ISSN 1990-4789, Journal of Applied and Industrial Mathematics, 2023, Vol. 17, No. 4, pp. 928–942. © Pleiades Publishing, Ltd., 2023.

Numerical Differentiation by the Polynomial-Exponential Basis

P. M. Nguyen[1*], T. T. Le[1], L. H. Nguyen[1***], and M. V. Klibanov[1****]**

[1] *Department of Mathematics and Statistics, University of North Carolina at Charlotte, Charlotte, NC 28223 USA*

*e-mail: *pnguye45@charlotte.edu, **tle55@charlotte.edu, ***loc.nguyen@charlotte.edu, ****mklibanv@charlotte.edu*

Received January 18, 2024; revised January 18, 2024; accepted January 18, 2024

Abstract—Our objective is to calculate the derivatives of data corrupted by noise. This is a challenging task as even small amounts of noise can result in significant errors in the computation. This is mainly due to the randomness of the noise, which can result in high-frequency fluctuations. To overcome this challenge, we suggest an approach that involves approximating the data by eliminating high-frequency terms from the Fourier expansion of the given data with respect to the polynomial-exponential basis. This truncation method helps to regularize the issue, while the use of the polynomial-exponential basis ensures accuracy in the computation. We demonstrate the effectiveness of our approach through numerical examples in one and two dimensions.

Keywords: *numerical differentiation, derivatives, polynomial-exponential basis, truncation*

DOI: 10.1134/S1990478923040191

1. INTRODUCTION

We revisit the problem *how to differentiate noisy data*. The problem is formulated as follows:

Problem 1. Let $\Omega = (-R, R)^d$, $d \geq 1$, be an open and bounded domain of $\mathbb{R}^d$ where R is a positive number. Let $f^* : \Omega \to \mathbb{R}$ be a function. Given the noisy data

$$f^\delta(\mathbf{x}) = f^*(\mathbf{x}) + \delta\eta(\mathbf{x}) \quad \mathbf{x} \in \Omega, \tag{1.1}$$

where η is any function taking random numbers in the range $[-1, 1]$ and $\delta \in (0, 1)$ is the noise level, find an approximation of the derivatives of f^*. More precisely, compute

1. the gradient vector $\nabla f^* = (\partial_{x_i} f^*)_{i=1}^d$,
2. the Hessian matrix $D^2 f^* = (\partial^2_{x_i x_j} f^*)_{i,j=1}^d$.

In (1.1), the statement that η can take on random values does not place any specific constraints on the noise present within the data. The task of differentiating noisy data holds great importance in various fields, including applied mathematics, statistics, data science, physics, engineering, etc. However, the problem of computing derivatives is severely ill-posed. Even a minor amount of noise in the data can result in significant errors in computation, see e.g., [1, 3]. We present a well-known example in 1D. Consider $\eta(x) = \sin(nx)$, where n is an integer and $x \in \mathbb{R}$. We have

$$\begin{aligned} f^\delta(x) &= f^*(x) + \delta \sin(nx), \\ f^{\delta\prime}(x) &= f^{*\prime}(x) + n\delta \cos(nx), \end{aligned} \tag{1.2}$$

for $x \in \mathbb{R}$. The absolute error in this example is $\|n\delta \cos(nx)\|_{L^\infty(\mathbb{R})} = n\delta \gg 1$ as $n \to \infty$, even though the perturbation term $\delta \sin(nx)$ is compatible to the noise level δ, which is assumed to be small. Due to the importance of differentiating noisy data, much research has been conducted in this area. In the following, we review some commonly employed methods for carrying out this task.

1. The finite difference method is the most natural approach for differentiation. An approximation of the derivative of f^δ can be obtained using the following formula:

$$f^{\delta\prime}(x) \approx \frac{f^\delta(x+h) - f^\delta(x)}{h} \approx f^{*\prime}(x) + \frac{\delta\big(\eta(x+h) - \eta(x)\big)}{h} \tag{1.3}$$

for $x \in \mathbb{R}$ where h is a step size. The step size h, which serves as a regularization factor, must be chosen appropriately to ensure stability. It is important to note that if the step size h is too small, the expression on the right-hand side of (1.3) becomes too large, leading to significant computational errors. This is because $\eta(x+h)$ and $\eta(x)$ might take any value and, hence, the term $\frac{\delta(\eta(x+h)-\eta(x))}{h}$ might diverge as $h \to 0$. For further details and the strategy to choose the step size h, we refer the reader to [5, 20].

2. Tikhonov optimization is a more commonly used approach that involves introducing cost functionals with a Tikhonov regularization term. An example of such a cost functional is

$$J(u) = \left\| f^\delta(x) - f^\delta(-R) - \int\limits_{\mathbb{R}}^{x} u(s)\,ds \right\|^2_{L^2(-R,R)} + \epsilon\|u\|^2_X, \tag{1.4}$$

where $\epsilon > 0$ is the regularization parameter and X is a functional space. The absolute minimum of this cost functional provides an approximation to $f^{*\prime}$. This approach is discussed in more details in [15]. Selecting the appropriate regularization parameter ϵ for each functional space X for the regularization term in (1.4) is not a trivial task. A well-known method for determining the regularization parameter is Morozov's discrepancy principle, which is detailed in [3, 18, 19, 23].

3. An alternative method for computing derivatives of data involves the use of cubic spline curves to smooth the data. It is worth mentioning that the cubic smoothing spline was initially introduced in [21, 22, 24]. Suppose that f^δ is the given data, and let S be the natural cubic spline that approximates f^δ. Here, by natural cubic spline, we mean that the function S is twice continuously differentiable with $S''(-R) = S''(R) = 0$, and S coincides on each subinterval $[x_{i-1}, x_i]$ with some cubic polynomial, where $x_0 = -R < x_1 < x_2 < \cdots < x_n = R$, $n > 1$, is a partition of $(-R, R)$. One can then use the derivatives of S to approximate the derivatives of f^*. For a method that combines cubic spline and Tikhonov optimization technique to differentiate noisy data, we refer the reader to [6].

4. Other methods for differentiation are discussed in [2, 4, 15, 16] and their references.

The example of $f^\delta(x) = f^*(x) + \delta\sin(nx)$ in (1.2) highlights the ill-posed nature of the problem, which arises from the high-frequency components of the data. It suggests cutting off these components to increase stability. More precisely, given a function $f^\delta \in L^2(\Omega)$, we approximate

$$f^\delta(\mathbf{x}) = \sum_{n=1}^{\infty} f_n^\delta \Phi_n(\mathbf{x}) \simeq \sum_{n=1}^{N} f_n^\delta \Phi_n(\mathbf{x}) \quad \mathbf{x} \in \Omega \tag{1.5}$$

for some cut-off number N, where $\{\Phi_n\}_{n=1}^{\infty}$ is an orthonormal basis of $L^2(\Omega)$ and

$$f_n^\delta(\mathbf{x}) = \int\limits_{\Omega} f^\delta(\mathbf{x})\Phi_n(\mathbf{x})d\mathbf{x}.$$

Then, we can approximate the desired derivatives using the following formulas

$$\begin{aligned} \nabla f^\delta(\mathbf{x}) &= \sum_{n=1}^{N} f_n^\delta \nabla\Phi_n(\mathbf{x}), \\ D^2 f^\delta(\mathbf{x}) &= \sum_{n=1}^{N} f_n^\delta D^2\Phi_n(\mathbf{x}), \end{aligned} \tag{1.6}$$

for $\mathbf{x} \in \Omega$. Although the numerical method based on (1.6) may not be rigorously valid due to the term-by-term differentiation of the series in (1.5), it is an effective technique for numerical computation. Proving the analytical validity of this approach as N approaches ∞ poses a significant challenge, which is not addressed in our paper. Nevertheless, our method remains acceptable. In fact, it is well-known that Problem 1 is highly unstable, and therefore, a regularized technique is necessary. Truncating the series in (1.5) serves as an option for this purpose.

The main concern in (1.6) is to identify a suitable basis $\{\Phi_n\}_{n\geq 1}$ that enables reliable numerical differentiation through (1.6). We observe that using a popular basis such as Legendre polynomials or trigonometric functions in the well-known Fourier expansion might not be appropriate. This is mainly due to the fact that the initial function of these bases is a constant, which has an identically zero derivative. Consequently, the contribution of f_1 in (1.6) is overlooked. The lack of this information significantly reduces the accuracy of the method. The basis $\{\Phi_n\}_{n\geq 1}$ we use in this paper is constructed by extending the polynomial-exponential basis initially introduced by Klibanov, the fourth author of this paper, in [10] and [12, Sec. 6.2.3]. The original goal of Klibanov was to use this basis for the numerical solution of some coefficient inverse problems, see, e.g. [7–9, 11, 13, 17], [12, Chs. 7, 10–12], and [14]. This basis is an appropriate choice for (1.6) since it satisfies the necessary condition that derivatives of Φ_n, $n \geq 1$, are not identically zero. We will demonstrate the effectiveness of the polynomial-exponential basis by comparing some numerical results due to (1.6) in two cases when $\{\Phi_n\}_{n\geq 1}$ is the polynomial-exponential basis and $\{\Phi_n\}_{n\geq 1}$ is the widely-used trigonometric basis. Additionally, we will compare the numerical results obtained by our method with those obtained by the Tikhonov regularization and the cubic spline methods, which will serve as further evidence of the efficiency of our approach.

The polynomial-exponential basis $\{\Phi_n\}_{n\geq 1}$ has an additional significant property, in addition to the "nonidentically zero" characteristic mentioned earlier. This property states that for all values of $N \geq 2$, the matrix $(\langle \Phi_n, \Phi_m' \rangle_{L^2})_{1\leq m,n\geq N}$ is invertible. This property guarantees a global convergence of Carleman-based convexification methods to solve inverse scattering problems in [7–9, 12, 17].

The organization of this paper is as follows. Section 2 provides an overview of our approach. Section 3 presents numerical results for one and two dimensions. A comparison between our method and other numerical approaches can be found in Sec. 4. Section 5 is for concluding remarks.

2. THE NUMERICAL METHOD BASED ON THE TRUNCATION TECHNIQUE

In [10], Klibanov proposed a special orthonormal basis of $L^2(-R, R)$ to numerically solve inverse problems. For any $x \in (-R, R)$, we define $\phi_n(x) = x^{n-1}e^x$. It is clear that the set $\{\phi\}_{n\geq 1}$ is complete in $L^2(-R, R)$. By applying the Gram-Schmidt orthonormalization process to this set, we obtain an orthonormal basis for $L^2(-R, R)$, which is denoted by $\{\Psi_n\}_{n\geq 1}$ and named the polynomial-exponential basis. The basis $\{\Psi_n\}_{n\geq 1}$ plays a vital role in our method for addressing Problem 1. The product of $\{\Psi_n\}_{n\geq 1}$ is the basis we use to approximate the noisy data f^δ. For $\mathbf{n} = (n_1, n_2, \ldots, n_d)$, we define

$$P_{\mathbf{n}}(\mathbf{x}) = \Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\ldots\Psi_{n_d}(x_d) \quad \text{for all} \quad \mathbf{x} = (x_1, \ldots, x_d) \in \Omega. \tag{2.1}$$

It is obvious that $\{P_{\mathbf{n}}\}_{\mathbf{n}\in\mathbb{N}^d}$ forms an orthonormal basis of $L^2(\Omega)$. For any function $f \in L^2(\Omega)$, we can write

$$f(\mathbf{x}) = \sum_{\mathbf{n}\in\mathbb{N}^d} a_{\mathbf{n}} P_{\mathbf{n}}(\mathbf{x}) = \sum_{\mathbf{n}=(n_1,\ldots,n_d)\in\mathbb{N}^d} a_{\mathbf{n}}\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\ldots\Psi_{n_d}(x_d) \quad \mathbf{x} \in \Omega, \tag{2.2}$$

where

$$a_{\mathbf{n}} = a_{n_1 n_2 \ldots n_d} = \int\limits_\Omega f(\mathbf{x})P_{\mathbf{n}}(\mathbf{x})d\mathbf{x} = \int\limits_\Omega f(x_1, x_2, \ldots, x_d)\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\ldots\Psi_{n_d}(x_d)(\mathbf{x})\, d\mathbf{x}. \tag{2.3}$$

Motivated by the Galerkin approximation, we truncate the representation of the function f in (2.2) as

$$f(\mathbf{x}) = \sum_{n_1=1}^{\infty}\cdots\sum_{n_d=1}^{\infty} a_{\mathbf{n}}\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\ldots\Psi_{n_d}(x_d) \simeq \sum_{n_1=1}^{N_1}\cdots\sum_{n_d=1}^{N_d} a_{\mathbf{n}}\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\ldots\Psi_{n_d}(x_d)$$

Algorithm 1 (the procedure to compute derivatives of noisy data).

1: Choose cut-off numbers $N_1, N_2, \dots, N_d$.

2: Construct the orthonormal basis $\{\Psi_n\}_{n\geq 1}$ defined in the first paragraph of Sec. 2.

3: For each $\mathbf{n} \in \{(n_1, \dots, n_d) : 1 \leq n_1 \leq N_1, \dots, 1 \leq n_d < N_d\}$, compute the coefficients $a_{\mathbf{n}}$

$$a_{\mathbf{n}} = a_{n_1 n_2 \dots n_d} = \int_\Omega f(x_1, x_2, \dots, x_d)\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\dots\Psi_{n_d}(x_d)d\mathbf{x}. \tag{2.7}$$

4: Find the first derivatives of f via formulas (2.5).

5: Find the second derivatives of f via formulas (2.6).

for some positive integers $N_1, \dots, N_d$ where $\mathbf{n} = (n_1, \dots, n_d)$ and $\mathbf{x} = (x_1, \dots, x_d)$. Hence, the function f approximately becomes

$$f(\mathbf{x}) \simeq \sum_{n_1=1}^{N_1} \cdots \sum_{n_d=1}^{N_d} a_{\mathbf{n}}\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\dots\Psi_{n_d}(x_d). \tag{2.4}$$

Remark 2.1. Truncating the Fourier series, as depicted in equation (2.4), provides an effective regularization technique for addressing ill-posed problems like Problem 1. This approach involves eliminating the high-frequency components of function f, resulting in a partial reduction of the ill-posedness of the problem. This technique has been successfully employed in solving inverse problems, where it enables the computation of numerical solutions even when significant noise levels are present, as evidenced by [7, 11, 13]. In particular, highly noisy experimental data are treated by this technique in [7] and [12, Chapter 10]. The cut-off numbers $N_1, \dots, N_d$ are utilized as regularization parameters in our method, and their determination can be achieved through numerical methods using the available data without requiring any knowledge of the noise level. We refer to Sec. 3.1 for further details on selecting these cut-off numbers.

After smoothing the data using (2.4), we can compute the first derivatives of f via the formulas

$$\partial_{x_i} f(\mathbf{x}) \simeq \sum_{n_1=1}^{N_1} \cdots \sum_{n_d=1}^{N_d} a_{\mathbf{n}}\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\dots\Psi'_{n_i}(x_i)\dots\Psi_{n_d}(x_d). \tag{2.5}$$

for $i \in \{1, \dots, d\}$. The second derivatives of f are computed using

$$\partial_{x_i x_j} f(\mathbf{x}) \simeq \begin{cases} \displaystyle\sum_{n_1=1}^{N_1} \cdots \sum_{n_d=1}^{N_d} a_{\mathbf{n}}\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\dots\Psi'_{n_i}(x_i)\dots\Psi'_{n_j}(x_j)\dots\Psi_{n_d}(x_d), & i \neq j \\ \displaystyle\sum_{n_1=1}^{N_1} \cdots \sum_{n_d=1}^{N_d} a_{\mathbf{n}}\Psi_{n_1}(x_1)\Psi_{n_2}(x_2)\dots\Psi''_{n_i}(x_i)\dots\Psi_{n_d}(x_d), & i = j, \end{cases} \tag{2.6}$$

for $i, j \in \{1, \dots, d\}$. The approximations in (2.5) and (2.6) suggest Algorithm 1 to solve Problem 1. We refer the reader to Sec. 3.1 for a numerical method to choose the cut-off numbers in the first step of Algorithm 1.

Remark 2.2. The selection of the polynomial-exponential basis $\{P_{\mathbf{n}}\}_{\mathbf{n}\in\mathbb{N}^d}$ is critical for achieving success in our approach. Upon revisiting equations (2.5) and (2.6), it becomes apparent that the accuracy of the computed results is heavily reliant on the contribution of the Fourier coefficients $a_{\mathbf{n}}$, $\mathbf{n} \in \mathbb{N}^d$. Therefore, if an inappropriate basis such as $\{Q_{\mathbf{n}}\}_{n\geq 1}$ is chosen, which contains an entry $Q_{\mathbf{m}}$ with zero derivative for some $\mathbf{m} \in \mathbb{N}^d$, then the corresponding Fourier coefficient $b_{\mathbf{m}} = \int_\Omega f(\mathbf{x})Q_{\mathbf{m}}(\mathbf{x})d\mathbf{x}$ would not be present in equations (2.5) and (2.6). For instance, in 1D,

the first entry of the trigonometric basis involving $\cos\left(\frac{\pi n x}{R}\right)$, $n \geq 0$, and $\sin\left(\frac{\pi m x}{R}\right)$, $m \geq 1$, used in the Fourier expansion is the constant $\cos 0 = 1$, whose derivative vanishes. Consequently, the first term in equations (2.5) and (2.6) for computed derivatives will be missing. This leads to a decrease in accuracy during computation as the first term in the series could be the most crucial one. Thus, the trigonometric basis is unsuitable for our method. This observation will be confirmed through numerical experiments in this paper.

3. NUMERICAL EXAMPLES

We implement Algorithm 1 for the cases of 1D and 2D. In all tests below, we compute the derivatives of a function f^* using the noisy data given by

$$f^\delta(\mathbf{x}) = f^*(\mathbf{x})\big(1 + \delta \text{rand}(\mathbf{x})\big) \tag{3.1}$$

where $\text{rand}(\mathbf{x})$ is a function that generates uniformly distributed random numbers in $[-1, 1]$. The noise levels are set to be 5%, 10%, and 20%.

3.1. Choosing the Cut-Off Number

Before presenting the numerical results, we explain the process of selecting the cut-off number in Step 1 of Algorithm 1. To simplify matters, we will only present our strategy for the one-dimensional case. The same approach is utilized for selecting the cut-off numbers in the two-dimensional case. For each $N > 1$, we compare noisy data f^δ with the sum $\sum_{n=1}^{N} f_n^\delta \Psi_n(x)$. In other words, for each

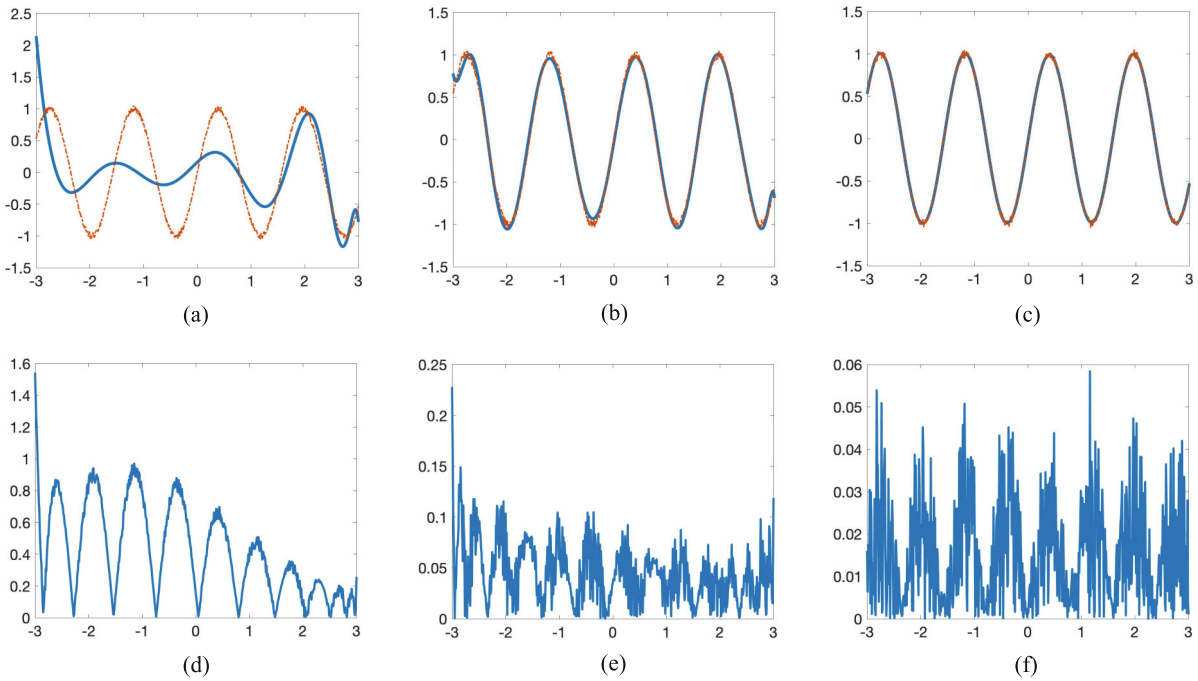

Fig. 1. An example of how to choose the cut-off number N where the data, with 5% noise, is taken from Test 1 below: (a) the data f^δ (dash-dot) and its approximation $\sum_{n=1}^{N} f_n^\delta \Psi_n(x)$ (solid) when $N = 10$; (b) the data f^δ (dash-dot) and its approximation $\sum_{n=1}^{N} f_n^\delta \Psi_n(x)$ (solid) when $N = 15$; (c) the data f^δ (dash-dot) and its approximation $\sum_{n=1}^{N} f_n^\delta \Psi_n(x)$ (solid) when $N = 20$; (d) the function φ_N when $N = 10$; (e) the function φ_N when $N = 15$; (f) the function φ_N when $N = 20$. It is evident that $N = 20$ should be chosen for Test 1.

given data f^δ, we compute

$$\varphi_N(x) = \frac{\left| f^\delta(x) - \sum_{n=1}^{N} f_n^\delta \Psi_n(x) \right|}{\|f^\delta\|_{L^\infty}}, \quad \text{where} \quad f_n^\delta = \int_{-R}^{R} f^\delta(x)\Psi_n(x)dx,$$

for several values of N. Then we choose N as the smallest number N such that $\|\varphi_N\|_{L^\infty(-R,R)}$ is sufficiently small. An illustration of this strategy is provided in Fig. 1.

This methodology is implemented in all of the following tests. The same procedure is used for the experiment involving two dimensions.

3.2. The Case of One Dimension

We compute the first and second derivatives of some functions $f^*(x)$ defined in $(-R, R)$ where $R = 3$. On $(-R, R)$, we arrange a partition

$$\mathcal{G} = \{x_i = -R + (i-1)h, i = 1, \ldots, N\} \subset [-3, 3] \tag{3.2}$$

where the step size $h = 0.001$ and $N = 6001$.

3.2.1. Test 1. In this test, we take $f^*(x) = \sin(4x)$ as our function of interest. The accurate first and second-order derivatives are $f'_{\text{true}}(x) = 4\cos(x)$ and $f''_{\text{true}}(x) = -16\sin(x)$, respectively. Our chosen cut-off number for this test is $N = 20$.

The graphical representation in Fig. 2 illustrates both the computed and true derivatives of f^*. Based on the outcomes shown in Fig. 2, it is apparent that Algorithm 1 performs effectively in computing derivatives, even when there is a high level of noise, up to 20%. It is worth mentioning

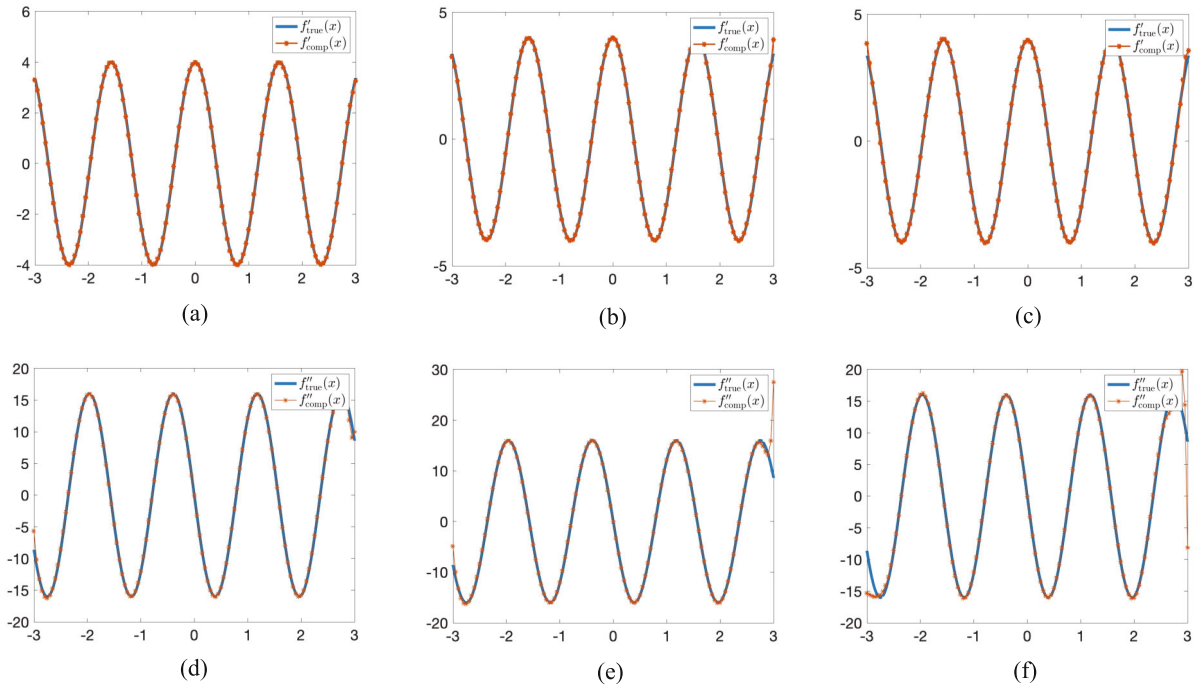

Fig. 2. *Test 1.* The true and computed derivatives of the function $f(x) = \sin(4x)$. As can be seen, our method is highly proficient in accurately computing derivatives of both the first and second order: (a) the true and computed $f'(x)$, noise level $\delta = 5\%$; (b) the true and computed $f'(x)$, noise level $\delta = 10\%$; (c) the true and computed $f'(x)$, noise level $\delta = 20\%$; (d) the true and computed $f''(x)$, noise level $\delta = 5\%$; (e) the true and computed $f'(x)$, noise level $\delta = 10\%$; (f) the true and computed $f''(x)$, noise level $\delta = 20\%$. Notably, the errors are primarily concentrated at the endpoints of the computed domain.

Table 1. *Test 1.* The relative computed errors with respect to the L^2 norms over the intervals $(-3, 3)$ and $(-2, 2)$. As can be observed, the relative errors are lower than the noise level

	$\frac{\|f'_{\text{true}} - f'_{\text{comp}}\|_{L^2(-3,3)}}{\|f'_{\text{true}}\|_{L^2(-3,3)}}$	$\frac{\|f''_{\text{true}} - f''_{\text{comp}}\|_{L^2(-3,3)}}{\|f''_{\text{true}}\|_{L^2(-3,3)}}$	$\frac{\|f'_{\text{true}} - f'_{\text{comp}}\|_{L^2(-2,2)}}{\|f'_{\text{true}}\|_{L^2(-2,2)}}$	$\frac{\|f''_{\text{true}} - f''_{\text{comp}}\|_{L^2(-2,2)}}{\|f''_{\text{true}}\|_{L^2(-2,2)}}$
$\delta = 0.05$	0.0060	0.0268	0.0030	0.0195
$\delta = 0.10$	0.0110	0.0996	0.0031	0.0201
$\delta = 0.20$	0.0260	0.1123	0.0073	0.0282

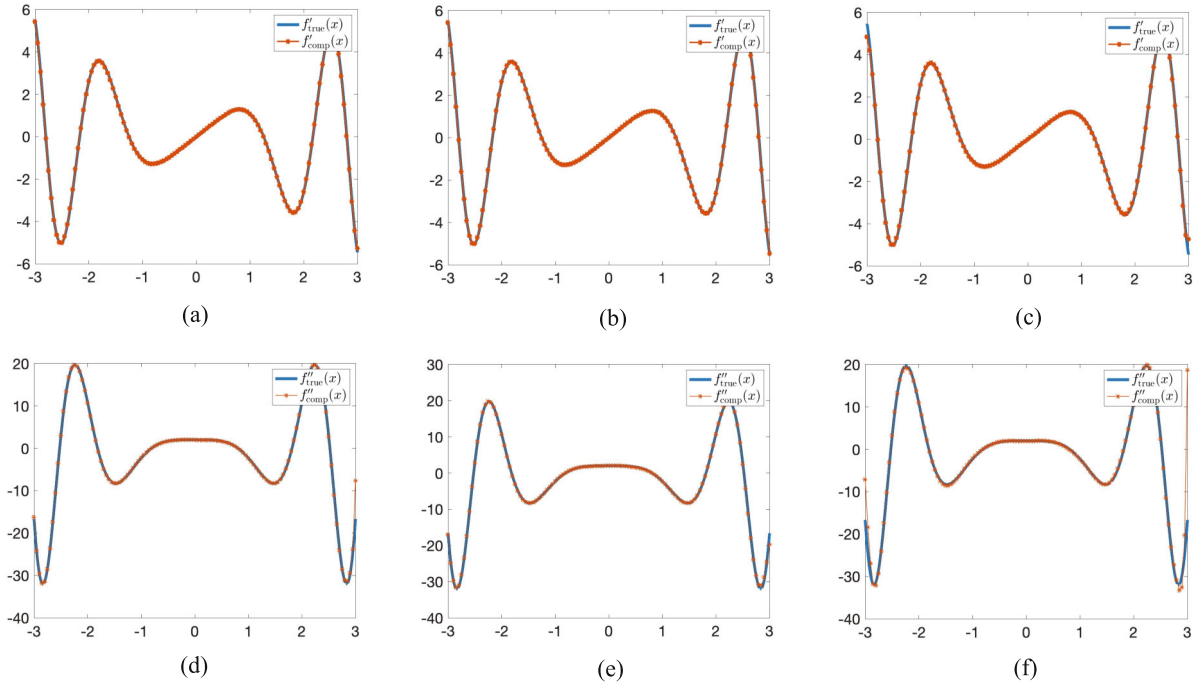

Fig. 3. *Test 2.* The true and computed derivatives of the function $f(x) = \sin(x^2)$: (a) the true and computed $f'(x)$, noise level $\delta = 5\%$; (b) the true and computed $f'(x)$, noise level $\delta = 10\%$; (c) the true and computed $f'(x)$, noise level $\delta = 20\%$; (d) the true and computed $f''(x)$, noise level $\delta = 5\%$; (e) the true and computed $f''(x)$, noise level $\delta = 10\%$; (f) the true and computed $f''(x)$, noise level $\delta = 20\%$. As seen, the performance of Algorithm 1 in this test is out of expectation. The errors occur at the endpoints $\{-3, 3\}$ while the computed derivatives are accurate inside the interval $(-3, 3)$.

that the errors mostly occur at the endpoints of the computational interval, while the computed function's accuracy within the interval exceeds expectations. The first two columns of Table 1 display the computed errors over the entire interval $(-3, 3)$, while the last two columns show the computed errors over the interval $(-2, 2)$.

3.2.2. Test 2. We calculate the derivatives for the function $f^*(x) = \sin(x^2)$. The true first-order and second-order derivatives for this function are given by $f'_{\text{true}} = 2x\cos(x^2)$ and $f''_{\text{true}} = 2\cos(x^2) - 4x^2\sin(x^2)$, respectively. For this test, we use a cut-off number of $N = 25$. Figure 3 displays the graphs of the computed derivatives.

Similar to Test 1, Fig. 3 visually confirms that our algorithms for computing the first and second derivatives are highly effective. The errors are primarily localized at the endpoints of the interval, whereas the computed function demonstrates noteworthy accuracy within the subinterval $(-2, 2)$. Detailed information can be found in Table 2.

Table 2. *Test 2.* The computed errors with respect to the L^2 norms over the intervals $(-3,3)$ and $(-2,2)$. It is apparent that the relative errors are below the noise level

	$\frac{\|f'_{\text{true}}-f'_{\text{comp}}\|_{L^2(-3,3)}}{\|f'_{\text{true}}\|_{L^2(-3,3)}}$	$\frac{\|f''_{\text{true}}-f''_{\text{comp}}\|_{L^2(-3,3)}}{\|f''_{\text{true}}\|_{L^2(-3,3)}}$	$\frac{\|f'_{\text{true}}-f'_{\text{comp}}\|_{L^2(-2,2)}}{\|f'_{\text{true}}\|_{L^2(-2,2)}}$	$\frac{\|f''_{\text{true}}-f''_{\text{comp}}\|_{L^2(-2,2)}}{\|f''_{\text{true}}\|_{L^2(-2,2)}}$
$\delta = 0.05$	0.0052	0.0380	0.0017	0.0309
$\delta = 0.10$	0.0074	0.0955	0.0047	0.0484
$\delta = 0.20$	0.0240	0.1734	0.0117	0.0704

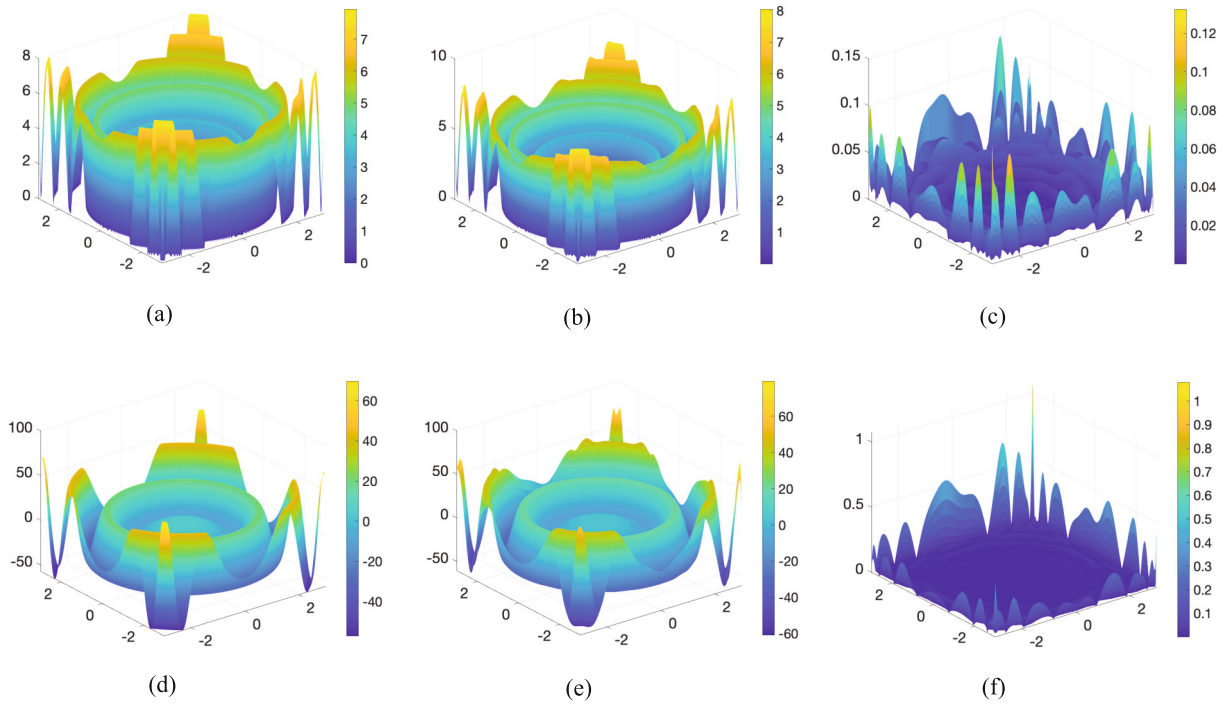

Fig. 4. *Test 3.* The true and computed $|\nabla f|$ and Δf, using data that has been corrupted with a 10% noise level: (a) the true function $|\nabla f^*|$; (b) the computed function $|\nabla f_{\text{comp}}|$; (c) $|\nabla f^* - \nabla f_{\text{comp}}|/\|\nabla f^*\|_{L^\infty}$; (d) the true function Δf^*; (e) the computed function Δf_{comp}; (f) $|\Delta f^* - \Delta f_{\text{comp}}|/\|\Delta f^*\|_{L^\infty}$.

3.3. The Case of Two Dimensions

Our computation involves setting $\Omega = (-R, R)^2$, where $R = 3$. Similar to the one-dimensional case, we implement Algorithm 1 in the finite difference scheme. To define our grid, we set

$$\mathcal{G} = \left\{\mathbf{x}_{ij} = (x_i = -R + (i-1)h, y_j = -R + (j-1)h : 1 \le i, j \le N_{\mathbf{x}}\right\}$$

where $N_{\mathbf{x}} = 601$ and $h = 2R/(N_{\mathbf{x}} - 1) = 0.01$. We choose the cut-off numbers N_1 and N_2 using the same strategy as in Sec. 3.2, i.e., we find N_1 and N_2 such that the approximation in (2.2) is satisfactory. For both tests presented below, we set $N_1 = N_2 = 20$.

3.3.1. Test 3. In this test, we take $f^*(x, y) = \sin(x^2 + y^2)$. Then

$$\begin{aligned}\left|\nabla f^*(x,y)\right| &= 2\sqrt{(x^2+y^2)\cos^2(x^2+y^2)},\\ \Delta f^*(x,y) &= 4\cos(x^2+y^2) - 4x^2\sin(x^2+y^2) - 4y^2\sin(x^2+y^2).\end{aligned}$$

In Fig. 4, we show the true and computed magnitudes of the gradient, $|\nabla f^*|$, and the Laplacian, Δf^*, are depicted using data that has been affected by a 10% noise level.

Table 3. *Test 3.* The relative computed errors with respect to the L^2 norms on $\Omega = (-3,3)^2$ and $\Omega' = (-2,2)^2$. As can be observed, the relative errors are lower than the noise level

	$\frac{\|\nabla f_{\text{true}} - \nabla f_{\text{comp}}\|_{L^2(\Omega)}}{\|\nabla f_{\text{true}}\|_{L^2(\Omega)}}$	$\frac{\|\Delta f_{\text{true}} - \Delta f_{\text{comp}}\|_{L^2(\Omega)}}{\|\Delta f_{\text{true}}\|_{L^2(\Omega)}}$	$\frac{\|\nabla f_{\text{true}} - \nabla f_{\text{comp}}\|_{L^2(\Omega')}}{\|\nabla f_{\text{true}}\|_{L^2(\Omega')}}$	$\frac{\|\Delta f_{\text{true}} - \Delta f_{\text{comp}}\|_{L^2(\Omega')}}{\|\Delta f_{\text{true}}\|_{L^2(\Omega')}}$
$\delta = 0.05$	0.0303	0.1282	0.0163	0.0320
$\delta = 0.10$	0.0313	0.1306	0.0165	0.0324
$\delta = 0.20$	0.0338	0.1610	0.0173	0.0337

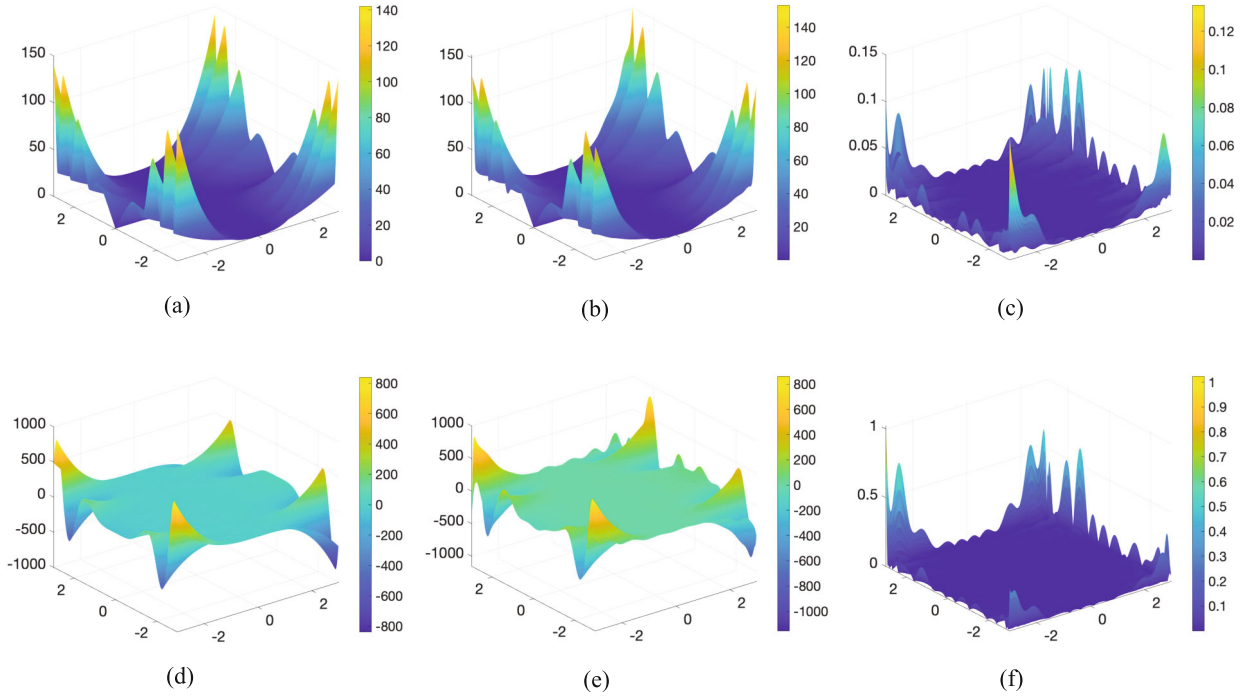

Fig. 5. *Test 4.* The true and computed $|\nabla f|$ and Δf, using data that has been corrupted with a 10% noise level: (a) the true function $|\nabla f^*|$; (b) the computed function $|\nabla f_{\text{comp}}|$; (c) $|\nabla f^* - \nabla f_{\text{comp}}|/\|\nabla f^*\|_{L^\infty}$; (d) the true function Δf^*; (e) The computed function Δf_{comp}; (f) $|\Delta f^* - \Delta f_{\text{comp}}|/\|\Delta f^*\|_{L^\infty}$.

It follows from Fig. 4 that we successfully compute the first and second derivatives of the function $f^*(x,y) = \sin(x^2+y^2)$. It is interesting to mention that the errors in the computation are less than the noise level. As seen in the last column of Figs. 4c and 4f, the computed errors occur mainly on the boundary of the computed domain. This feature is true when we test our algorithm with up to the noise level 20%.

The performance of our method with different noise levels $\delta \in \{5\%, 10\%, 20\%\}$ is out of expectation. Table 3 shows that the relative computed errors are consistent with the corresponding noise levels. The relative error with respect to the $L^2(\Omega')$ norm is significantly low.

3.3.2. Test 4. We test our algorithms by computing derivatives of $f^*(x,y) = x^3\sin(y^2)$. As in Test 3, we will compute numerical versions of the functions

$$\begin{aligned} \left|\nabla f^*(x,y)\right| &= \sqrt{9x^4\sin^2(y^2) + 4x^6y^2\cos^2(y^2)} \\ \Delta f^*(x,y) &= (-4x^3y^2 + 6x)\sin(y^2) + 2x^3\cos(y^2). \end{aligned}$$

The numerical results are displayed in Fig. 5.

Like Test 3, it is evident from Fig. 5 and Table 4 that Algorithm 1 provides out-of-expectation numerical results for the first and second derivatives of the function f^* with acceptable relative errors.

Table 4. *Test 4.* The relative computed errors with respect to the L^2 norms on $\Omega = (-3,3)^2$ and $\Omega' = (-2,2)^2$. As can be observed, the relative errors are lower than the noise level

	$\frac{\|\nabla f_{\text{true}} - \nabla f_{\text{comp}}\|_{L^2(\Omega)}}{\|\nabla f_{\text{true}}\|_{L^2(\Omega)}}$	$\frac{\|\Delta f_{\text{true}} - \Delta f_{\text{comp}}\|_{L^2(\Omega)}}{\|\Delta f_{\text{true}}\|_{L^2(\Omega)}}$	$\frac{\|\nabla f_{\text{true}} - \nabla f_{\text{comp}}\|_{L^2(\Omega')}}{\|\nabla f_{\text{true}}\|_{L^2(\Omega')}}$	$\frac{\|\Delta f_{\text{true}} - \Delta f_{\text{comp}}\|_{L^2(\Omega')}}{\|\Delta f_{\text{true}}\|_{L^2(\Omega')}}$
$\delta = 0.05$	0.0336	0.1981	0.0128	0.0587
$\delta = 0.10$	0.0386	0.2301	0.0142	0.0645
$\delta = 0.20$	0.0571	0.3666	0.0154	0.0734

4. COMPARISON WITH SOME EXISTING DIFFERENTIATING METHODS

In this section, we compare the efficiency of Algorithm 1 with some other widely-used methods for differentiation. For brevity, we only implement and present the computation of these methods in one dimension.

4.1. Fourier Expansion with the Trigonometric Basis

If we substitute the polynomial-exponential basis with the trigonometric basis in Algorithm 1, *what would be the outcome?* Let $N_{\text{trig}} > 0$ be a cut-off number. Write

$$f^\delta(x) \simeq f^\delta_{\text{trig}}(x) = a_0 + \sum_{n=1}^{N_{\text{trig}}} a_n \cos\Big(\frac{\pi n x}{R}\Big) + \sum_{n=1}^{N_{\text{trig}}} b_n \sin\Big(\frac{n\pi x}{R}\Big) \tag{4.1}$$

where

$$\begin{aligned} a_0 &= \frac{1}{2R}\int\limits_{-R}^{R} f^\delta(s)ds, \\ a_n &= \frac{1}{R}\int\limits_{-R}^{R} f^\delta(s)\cos\Big(\frac{\pi n s}{R}\Big)ds, \quad n \geq 1, \\ b_n &= \frac{1}{R}\int\limits_{-R}^{R} f^\delta(s)\sin\Big(\frac{\pi n s}{R}\Big)ds, \quad n \geq 1. \end{aligned}$$

Like in Sec. 3.1, we choose the optimal cut-off number N_{trig} using the following formula

$$N_{\text{trig}} = \underset{3\leq N\leq 25}{\operatorname{argmin}} \left\{ \left\| f^\delta - \left[a_0 + \sum_{n=1}^{N} a_n \cos\Big(\frac{\pi n x}{R}\Big) + \sum_{n=1}^{N} b_n \sin\Big(\frac{n\pi x}{R}\Big)\right]\right\|_{L^\infty(-R,R)} \right\}. \tag{4.2}$$

The range $3 \leq N \leq 25$ in (4.2) can be changed. However, we observe that when N exceeds 25, the numerical results get worse. Using (4.1), we approximate $f^{*\prime}$ via the formula

$$f'_{\text{trig}}(x) \simeq -\sum_{n=1}^{N_{\text{trig}}} \frac{\pi n a_n}{R}\sin\Big(\frac{\pi n x}{R}\Big) + \sum_{n=1}^{N_{\text{trig}}} \frac{n\pi b_n}{R}\cos\Big(\frac{n\pi x}{R}\Big). \tag{4.3}$$

We also can approximate $f^{*\prime\prime}$ via

$$f''_{\text{trig}}(x) = -\sum_{n=1}^{N_{\text{trig}}} \Big(\frac{\pi n}{R}\Big)^2 a_n \cos\Big(\frac{\pi n x}{R}\Big) - \sum_{n=1}^{N_{\text{trig}}} \Big(\frac{\pi n}{R}\Big)^2 b_n \sin\Big(\frac{n\pi x}{R}\Big). \tag{4.4}$$

In rows 3 of Table 5 and Table 6, we present the relative errors of the method utilizing (4.3) and (4.4) with respect to the $L^2(-3,3)$ norm. The experiment employs data with 5%, 10%, and 20% noise components. The comparison of these results and those via Algorithm 1 in rows 2 and 5 reveals that, in terms of accuracy, the polynomial-exponential basis outperforms the trigonometric basis. We conclude that the use of the polynomial-exponential basis is crucial to the success of our approach.

4.2. Comparison with the Tikhonov Regularization Method

A widely used approach to differentiate the data is to employ the Tikhonov optimization. Define $u = f'$. By the fundamental theorem of calculus, we write

$$f^\delta(x) - f^\delta(-R) = \int_{-R}^{x} u(s)ds.$$

This suggests us to find u by minimizing the following Tikhonov cost functional

$$J_\gamma(u) = \int_{-R}^{R} \left| \int_{-R}^{x} u(s)ds - \left[f^\delta(x) - f^\delta(-R)\right] \right|^2 dx + \gamma \|u\|^2_{L^2(-R,R)}$$

where $\gamma > 0$ is a regularization parameter. The term $\gamma\|u\|^2_{L^2(-R,R)}$ is called the regularization term. It is possible to employ higher-order norms, such as H^1 and H^2, instead of the L^2 norm for the regularization term. The derivative ${f^\delta_{\text{Tik}}}'$ is determined to be $u^\gamma_{\min}$, which is the minimizer of J_γ. We compute the second derivative of f, denoted by ${f^\delta_{\text{Tik}}}''$, by differentiating ${f^\delta_{\text{Tik}}}'$ using the Tikhonov optimization approach again. While the Tikhonov regularization approach is known for its robustness, selecting the appropriate norm and regularization parameter is not trivial. A popular approach to selecting the regularization parameter is Morosov's discrepancy principle [18], in which the regularization parameter γ is determined such that

$$\left\| \int_{-R}^{x} u^\gamma_{\min}(s)ds - \left[f^\delta(x) - f^\delta(-R)\right] \right\|_{L^2(-R,R)} = O(\delta) \tag{4.5}$$

where δ is the noise level. Morosov's discrepancy method requires knowledge of the noise level, which is not always known in practical applications. The $L-$shape method is a good option to find γ. One sketches the graph of the function $l : (0,1) \to \mathbb{R}$ defined as

$$l(\gamma) = \left\| \int_{-R}^{x} u^\gamma_{\min}(s)ds - \left[f^\delta(x) - f^\delta(-R)\right] \right\|^2_{L^2(-R,R)}.$$

The graph of l is an L-shape curve, see Fig. 6. The optimal regularization parameter is determined by the horizontal coordinate of the corner of the L curve. We show in row 4 and row 8 of Table 5 the relative computing error using the Tikhonov regularization method, in which we use the L-shape method to choose the parameter γ. Comparing the relative errors in row 2 and row 6 of this Table, we conclude that our method performs better than the Tikhonov optimization method.

Remark 4.1. Since we need to compute $l(\gamma)$, we have to repeatedly attempt to minimize J_γ for each value of γ, which makes the optimization-based method time-consuming. In the conducted experiments, we utilized an iMac 3.2 GHz Quad-Core Intel Core I5 to evaluate $l(\gamma)$ on uniformly partitioning the interval $(10^{-5}, 10^{-2})$ with a step size of 5×10^{-4}. The computation took 93.85 seconds. In contrast, the same computer required just 0.01167 seconds to compute both the first and second derivatives of the function in each of Test 1 and Test 2. Thus, our method is considerably faster.

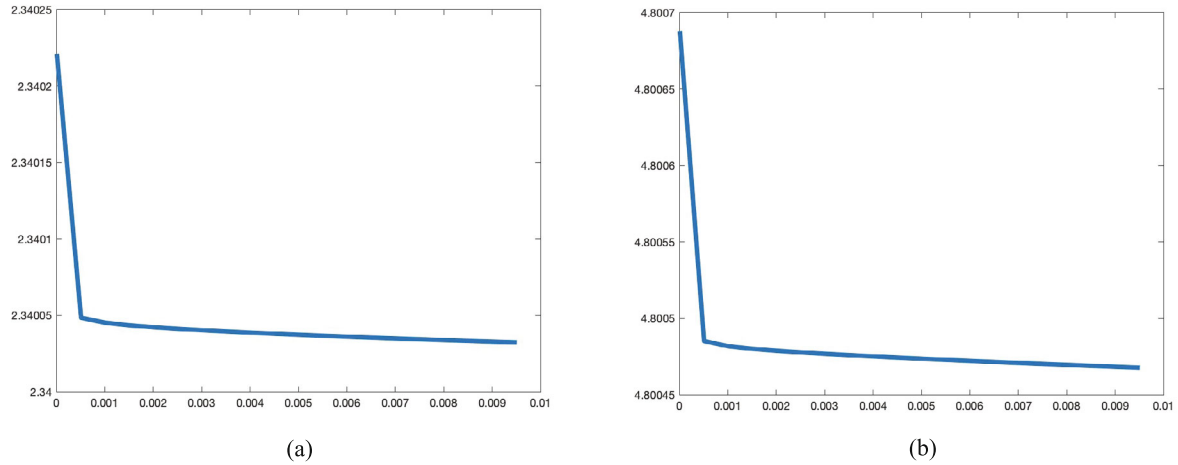

Fig. 6. The L-shape behavior of the function l: (a) the graph of the function $l(\gamma)$, $\gamma \in (10^{-5}, 10^{-2})$, computed by 5% noisy data taken in Test 1; (b) the graph of the function $l(\gamma)$, $\gamma \in (10^{-5}, 10^{-2})$, computed by 5% noisy data taken in Test 2. The optimal regularization parameter γ occurs at the corner of the graph.

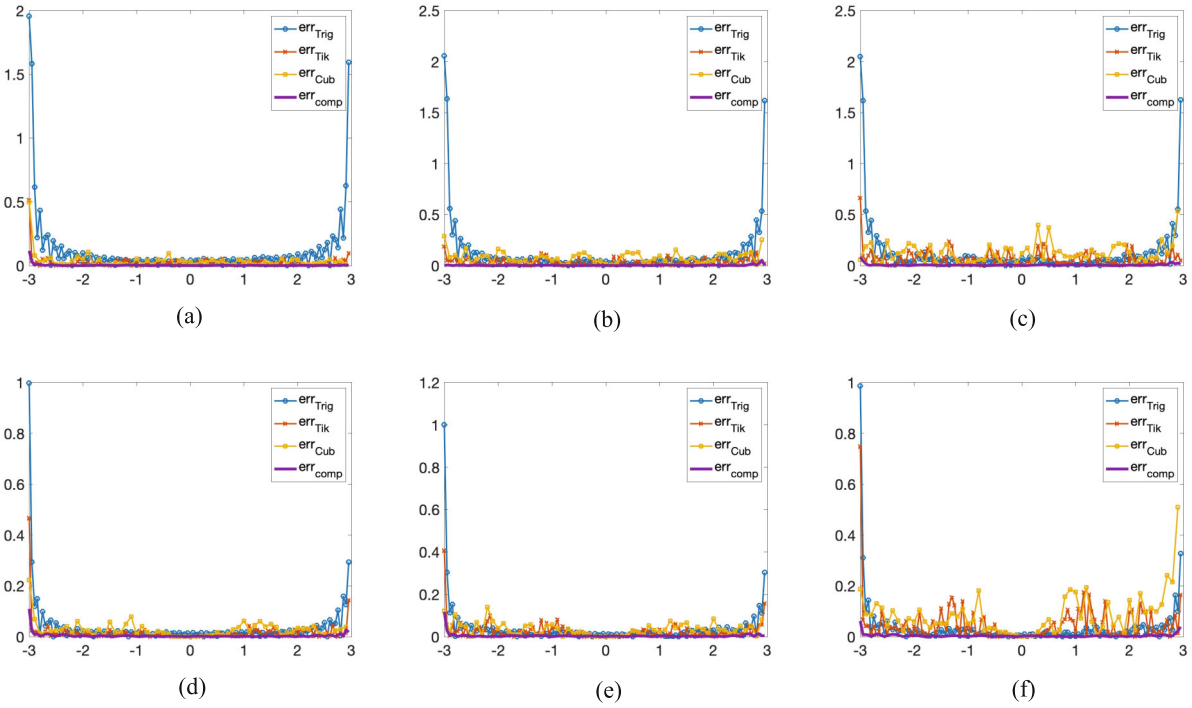

Fig. 7. The relative difference of the true and computed first derivatives of the functions in Test 1 and Test 2: (a) Test 1, noise level $\delta = 5\%$; (b) Test 1, noise level $\delta = 10\%$; (c) Test 1, noise level $\delta = 20\%$; (d) Test 2, noise level $\delta = 5\%$; (e) Test 2, noise level $\delta = 10\%$; (f) Test 2, noise level $\delta = 20\%$. These relative difference functions are defined in (4.6) for $x \in (-3, 3)$.

4.3. Comparison with the Method Based on Cubic Spline

One can apply cubic spline curves, introduced in [21, 22, 24], to smooth data before differentiating it. Define a partition $\{x_i = -R + (i-1)h, i = 1, 2, \ldots, N\}$ of the computational interval $(-R, R)$ similarly to the partition in (3.2). The difference between this partition and the one in (3.2) is that we change the step size to $h = 0.1$ because the cubic spline method works better with sparse data. Then, we use the "spline" command in Matlab to construct a set of spline curves that fit the data f^δ. The command spline provides the coefficients of each cubic function in each subdomain

Table 5. The relative errors in computing the first derivatives of our method and several other approaches are evaluated with respect to the $L^2(-3,3)$ norm. In this table, f'_{true} is the true first derivative that we aim to approximate. The function f'_{comp} is the first derivative computed by our approach. The functions f'_{trig}, f'_{Tik} and f'_{cub} are the derivative computed by the existing methods reviewed in Secs. 3.2, 4.2, and 4.3 respectively.

	Test 1 $\delta = 0.05$	Test 1 $\delta = 0.10$	Test 1 $\delta = 0.20$	Test 2 $\delta = 0.05$	Test 2 $\delta = 0.10$	Test 2 $\delta = 0.20$
$\frac{\|f'_{\text{true}} - f'_{\text{comp}}\|_{L^2(-3,3)}}{\|f'_{\text{true}}\|_{L^2(-3,3)}}$	**0.0060**	**0.0110**	**0.0260**	**0.0017**	**0.0047**	**0.0117**
$\frac{\|f'_{\text{true}} - f'_{\text{trig}}\|_{L^2(-3,3)}}{\|f'_{\text{true}}\|_{L^2(-3,3)}}$	0.4157	0.4397	0.4485	0.2216	0.2219	0.2260
$\frac{\|f'_{\text{true}} - f'_{\text{Tik}}\|_{L^2(-3,3)}}{\|f'_{\text{true}}\|_{L^2(-3,3)}}$	0.0652	0.0879	0.1484	0.1115	0.1248	0.1780
$\frac{\|f'_{\text{true}} - f'_{\text{cub}}\|_{L^2(-3,3)}}{\|f'_{\text{true}}\|_{L^2(-3,3)}}$	0.0775	0.1185	0.2450	0.0551	0.1904	0.2355

Table 6. The relative errors in computing the second derivatives of our method and several other approaches are evaluated with respect to the $L^2(-3,3)$ norm. In this table, f''_{true} is the true second derivative that we aim to approximate. The function f''_{comp} is the second derivative computed by our approach. The functions f''_{trig}, f''_{Tik} and f''_{cub} are the derivative computed by the existing methods reviewed in Secs. 3.2, 4.2, and 4.3 respectively.

	Test 1 $\delta = 0.05$	Test 1 $\delta = 0.10$	Test 1 $\delta = 0.20$	Test 2 $\delta = 0.05$	Test 2 $\delta = 0.10$	Test 2 $\delta = 0.20$
$\frac{\|f''_{\text{true}} - f''_{\text{comp}}\|_{L^2(-3,3)}}{\|f''_{\text{true}}\|_{L^2(-3,3)}}$	**0.0268**	**0.0996**	**0.1123**	**0.0380**	**0.0955**	**0.1734**
$\frac{\|f''_{\text{true}} - f''_{\text{trig}}\|_{L^2(-3,3)}}{\|f''_{\text{true}}\|_{L^2(-3,3)}}$	1.2419	1.4411	1.5455	1.0224	1.0314	1.0674
$\frac{\|f''_{\text{true}} - f''_{\text{Tik}}\|_{L^2(-3,3)}}{\|f''_{\text{true}}\|_{L^2(-3,3)}}$	0.3410	0.4504	0.6334	0.5912	0.6044	0.6700
$\frac{\|f'_{\text{true}} - f''_{\text{cub}}\|_{L^2(-3,3)}}{\|f''_{\text{true}}\|_{L^2(-3,3)}}$	0.5406	0.6970	1.2402	0.5118	0.7455	1.0360

$[x_i, x_{i+1}]$, $i \in \{1, \dots, N_x - 1\}$, that allows us to compute the first and second derivatives of the function f^δ.

To evaluate the performance of the cubic spline method, we show the relative error in computation in rows 5 of Table 5 and Table 6. Our experiment reveals that this method is well-suited for computing the first derivative; however, it may lead to a large computation error when computing the second derivative. A comparison of the first and final columns of the aforementioned tables leads us to the conclusion that our method yields markedly more accurate results.

4.4. Numerical Results

In Fig. 7, we show the graphs of the following functions

$$\begin{aligned} \text{err}_{\text{Trig}}(x) &= \frac{\left|f'_{\text{Trig}}(x) - f'_{\text{true}}(x)\right|}{\|f'_{\text{true}}\|_{L^\infty(-3,3)}}, \qquad \text{err}_{\text{Tik}}(x) = \frac{\left|f'_{\text{Tik}}(x) - f'_{\text{true}}(x)\right|}{\|f'_{\text{true}}\|_{L^\infty(-3,3)}}, \\ \text{err}_{\text{Cub}}(x) &= \frac{\left|f'_{\text{Cub}}(x) - f'_{\text{true}}(x)\right|}{\|f'_{\text{true}}\|_{L^\infty(-3,3)}}, \qquad \text{err}_{\text{comp}}(x) = \frac{\left|f'_{\text{comp}}(x) - f'_{\text{true}}(x)\right|}{\|f'_{\text{true}}\|_{L^\infty(-3,3)}} \end{aligned} \tag{4.6}$$

where f'_{Trig}, f'_{Tik}, f'_{Cub}, and f'_{comp} are the computed first derivatives via the methods in Secs. 3.2, 4.2, 4.3, and 2, respectively. The data is taken from Test 1 and Test 2 in Sec. 3.2 when the noise level are 5%, 10%, and 20%. We do not show similar graphs for the computed second derivatives since the methods mentioned in Secs. 3.2, 4.2, and 4.3 are not accurate, see Table 5. It is evident from Fig. 7 that our method performs better than the methods mentioned.

We also show in Table 5 and Table 6 the relative computational errors with respect to the $L^2(-3,3)$ norm. Based on the information in Table 5 and Table 6, we can infer that our approach is the most

precise one. The relative errors in computing the first derivative are roughly ten times lower than the noise levels, while the other methods have relative errors that are consistent with the noise levels. Additionally, our method's relative errors in computing second derivatives are in line with the noise levels, whereas other methods' relative errors in computing second derivatives are not satisfactory.

5. CONCLUDING REMARKS

We introduce a technique to compute derivatives of data affected by noise, which can be difficult due to the potential for significant computation errors even from a small amount of noise. Our method is straightforward. Firstly, we represent the data by its Fourier series with respect to the polynomial-exponential basis. Then, we remove all high-frequency components by truncating this series. The desired derivatives are then obtained directly. To demonstrate the efficacy of our algorithm, we provide numerical examples in both one and two dimensions. Furthermore, we compare our approach with some commonly used methods and conclude that our method is more precise.

FUNDING

The works of PMN, TTL, and LHN were partially supported by National Science Foundation grant no. DMS-2208159, and by funds provided by the Faculty Research Grant program at UNC Charlotte Fund no. 111272, and by the CLAS small grant provided by the College of Liberal Arts & Sciences, UNC Charlotte.

DATA AVAILABILITY

Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. K. Ahnert and L. A. Segel, "Numerical differentiation of experimental data: local versus global methods," Comput. Phys. Commun. **177**, 764–774 (2007).
2. F. V. Breugel, J. N. Kutz, and B. W. Brunton, "Numerical differentiation of noisy data: A unifying multi-objective optimization framework," IEEE Access **8**, 196865–196877 (2020).
3. H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems, Mathematics and Its Applications* (Kluwer, Dordrecht, 1996).
4. K. O. Friedrichs, "The identity of weak and strong extensions of differential operators," Trans. Am. Math. Soc. **55**, 132–151 (1944).
5. C. W. Groetsch, "Differentiation of approximately specified functions," Am. Math. Mon. **98**, 847–850 (1991).
6. M. Hanke and O. Sherzer, "Inverse problems light numerical differentiation," Am. Math. Mon. **108**, 512–521 (2001).
7. V. A. Khoa, G. W. Bidney, M. V. Klibanov, L. H. Nguyen, L. Nguyen, A. Sullivan, and V. N. Astratov, "Convexification and experimental data for a 3D inverse scattering problem with the moving point source," Inverse Probl. **36**, 085007 (2020).
8. V. A. Khoa, G. W. Bidney, M. V. Klibanov, L. H. Nguyen, L. Nguyen, A. Sullivan, and V. N. Astratov, "An inverse problem of a simultaneous reconstruction of the dielectric constant and conductivity from experimental backscattering data," Inverse Probl. Sci. Eng. **29** (5), 712–735 (2021).

9. V. A. Khoa, M. V. Klibanov, and L. H. Nguyen, "Convexification for a 3D inverse scattering problem with the moving point source," SIAM J. Imaging Sci. **13** (2), 871–904 (2020).
10. M. V. Klibanov, "Convexification of restricted Dirichlet to Neumann map," J. Inverse Ill-Posed Probl. **25** (5), 669–685 (2017).
11. M. V. Klibanov, T. T. Le, and L. H. Nguyen, "Convergent numerical method for a linearized travel time tomography problem with incomplete data," SIAM J. Sci. Comput. **42**, B1173–B1192 (2020).
12. M. V. Klibanov and J. Li, *Inverse Problems and Carleman Estimates: Global Uniqueness, Global Convergence and Experimental Data* (De Gruyter, Berlin, 2021).
13. M. V. Klibanov and L. H. Nguyen, "PDE-based numerical method for a limited angle X-ray tomography," Inverse Probl. **35**, 045009 (2019).
14. M. V. Klibanov and A. Timonov, "A comparative study of two globally convergent numerical methods for acoustic tomography," Commun. Anal. Comput. **1**, 12–31 (2023).
15. I. Knowles and R. J. Renka, "Methods for numerical differentiation of noisy data," Electron. J. Differ. Equat. Conf. **21**, 235–246 (2014).
16. I. Knowles and R. Wallace, "A variational method for numerical differentiation," Numer. Math. **70**, 91–110 (1995).
17. T. T. Le and L. H. Nguyen, "The gradient descent method for the convexification to solve boundary value problems of quasi-linear PDEs and a coefficient inverse problem," J. Sci. Comput. **91** (3), 74 (2022).
18. V. A. Morozov, *Methods for Solving Incorrectly Posed Problems* (Springer Verlag, New York, 1984).
19. R. Ramlau., "Morozov's discrepancy principle for Tikhonov regularization of nonlinear operators," J. Numer. Funct. Anal. Opt. **23**, 147–172 (2002).
20. A. G. Ramm and A. B. Smirnova, "On stable numerical differentiation," Math. Comput. **70**, 1131–1153 (2001).
21. C. H. Reinsch, "Smoothing by spline functions," Numer. Math. **10**, 177–183 (1967).
22. C. H. Reinsch, "Smoothing by spline functions. II," Numer. Math. **16**, 451–454 (1971).
23. O. Scherzer, "The use of Morozov's discrepancy principle for Tikhonov regularization for solving nonlinear ill-posed problems," SIAM J. Numer. Anal. **30**, 1796–1838 (1993).
24. I. J. Schoenberg, "Spline functions and the problem of graduation," Proc. Am. Math. Soc. **52**, 497–950 (1964).