Trigonometric Interpolation on Non-Periodic Functions and its Applications

Xiaorong Zou (xiaorong.zou@bofa.com)
Global Market Risk Analytic, Bank of America

November 24, 2025

Abstract. The aims of the study are twofold. First, we propose a novel trigonometric interpolation algorithm to estimate non-periodic functions defined on bounded intervals. The algorithm is computationally effective by using Fast Fourier Transform (FFT), and achieves optimal convergent rates of the approximation for not only a target function, but also the associated high-order derivatives of the target function. Some numerical experiments show decent accuracy with a moderate number of grid points. The algorithm has been applied to estimate integrals and certain testing results show that it outperforms significantly Trapezoid and Simpson method.

Secondly, we show how a trigonometric-interpolation based optimization (TIBO) can be used to solve a non-linear ODE. We demonstrate the idea by developing a new algorithm to cope with first-order non-linear ODEs. Numerical experiments show that it outperforms significantly the classic Runge-Kutta method. The algorithm has been generalized to second-order ODEs in a separate paper with decent performance and flexibility to address various boundary conditions and attack challenging issues like identifying the solution with a predefined property (e.g. being positive or increasing) when there are multiple solutions.

Keywords: Fourier Series, Trigonometric Interpolation, Fast Fourier Transformation (FFT), Ordinary Differential Equation, Runge-Kutta method.

MSC2000: Primary 65T40; Secondary 65L05

1. Introduction

Let $f(x)$ be a periodic function with period $2b$. Assume that its $K+1$-th derivative $f^{(K+1)}$ is bounded by D_{K+1} for certain $K \geq 1$. A trigonometric interpolant of degree n with given nodes $x_0 < x_1 \ldots, < x_{N-1}$ is a trigonometric polynomial

$$f_n(x) = a_{0,n} + \sum_{1 \leq j \leq n} a_{j,n} \cos \frac{j\pi x}{b} + b_{j,n} \sin \frac{j\pi x}{b},$$

such that

$$y_k := f(x_k) = f_n(x_k), \quad k = 0, \cdots, N-1.$$

As an example, let

$$x_j = \frac{2bj}{2n+1} = j\lambda, \qquad \lambda := \frac{2b}{2n+1}, \qquad -n \leq j \leq n.$$

be $2n+1$ equispaced nodes over $[-b,b]$, and

$$a_j^{(2n+1)} := \frac{2}{2n+1}\sum_{k=-n}^{n} y_k \cos\frac{2\pi kj}{2n+1}, \tag{1}$$

$$b_j^{(2n+1)} := \frac{2}{2n+1}\sum_{k=-n}^{n} y_k \sin\frac{2\pi kj}{2n+1}. \tag{2}$$

It is not hard to verify that

$$f_n^{(2n+1)}(x) := \frac{a_0^{(2n+1)}}{2} + \sum_{1\le j\le n} a_j^{(2n+1)}\cos\frac{j\pi x}{b} + b_j^{(2n+1)}\sin\frac{j\pi x}{b}.$$

is the unique trigonometric interpolant of degree n. $\{a_j^{(2n+1)}, b_j^{(2n+1)}\}$ can be obtained by Fast Fourier Transform (FFT). Note that odd number $N=2n+1$ of terms occurs in FFT while the optimal performance of FFT can only be reached when $N=2^h$ is a radix-2 integer for some h [3]. A trigonometric interpolation algorithm by even number $N=2M$ of nodes is available in the literature [6] (Theorem 3.5-3.6) and it converges uniformly at order $K-\frac{1}{2}$, i.e.

$$|Q_M(x)-f(x)| \le \frac{\xi_M}{N^{K-\frac{1}{2}}}, \quad \xi_M = o(1). \tag{3}$$

where $Q_M(x)$ is the interpolant uniquely determined by N evenly-spaced grid points.

The above algorithm has been recently enhanced in [8] as follows

Theorem 1.1. *Let $f(x)$ be a periodic function with period $2b$, $N=2M$ be an even integer and define*

$$x_j = -b + j\lambda, \quad \lambda = \frac{2b}{N}, \quad y_j = f(x_j), \qquad 0\le j<N.$$

- *If $f(x)$ is even, then there is a unique $M-1$ degree trigonometric polynomial*

$$f_M(x) = \sum_{0\le j<M} a_j \cos\frac{j\pi x}{b},$$

$$a_0 = \frac{1}{M}\sum_{0\le j<M} y_{2j}, \quad a_j = \frac{1}{M}\sum_{0\le k<N}(-1)^j y_k\cos\frac{2\pi jk}{N}, \quad j\ge 1,$$

such that for $0\le k<M$,

$$f_M(x_{2k}) = y_{2k}, \quad f_M(x_{2k+1}) = y_{2k+1} + \epsilon_M, \tag{4}$$

where

$$\epsilon_M = \frac{1}{M} \sum_{0 \le j < N} (-1)^j y_j. \tag{5}$$

In another word, $f_M(x)$ fits to all even grid points and shifts away in parallel from all odd grid points by ϵ_M.

– *If $f(x)$ is odd, then there is a unique $M-1$ degree trigonometric polynomial*

$$\begin{aligned} f_M(x) &= \sum_{0 \le j < M} a_j \sin \frac{j\pi x}{b}, \\ a_j &= \frac{2}{N} \sum_{0 \le k < N} (-1)^j y_k \sin \frac{2\pi jk}{N}, \quad 0 \le j < M \end{aligned}$$

such that it fits to all grid points, i.e.

$$f_M(x_k) = y_k, \quad 0 \le k < N.$$

The error ϵ_M in Eq (5) is $O(\frac{1}{N^{K+1}})$ by Euler-Maclaurin identity [4]. f_M can be implemented by Inverse Fast Fourier Transform (ifft).

Compared to the method in [6], $f_M(x)$ in Theorem 1.1 generates the error ϵ_M at half of grid nodes as disadvantage if f is even. The advantage is significant. The number of coefficients can be an radix-2 integer to fully leverage FFT power; more importantly, one can not only improve convergence rate in Eq. (3) to the optimal level N^{-K} for a target function, but also obtain the optimal convergent rate at N^{K-k} for the associated k-order derivatives of the target function as follows

Theorem 1.2. *Assume that $|f^{(K+1)}(x)|$ exists with an upper bound D_{K+1}, then*

$$|f_M^{(k)}(x) - f^{(k)}(x)| \le \frac{C(D_{K+1})}{N^{K-k}}, \quad 0 \le k < K. \tag{6}$$

where $C(D_{K+1})$ is a constant depending on D_{K+1}.

Theorem 1.2 provides a solid base for such applications as solving differential equations where an interpolant of a function can be used to approximate derivatives of the function. The proof of Theorem 1.2 and relevant properties of the trigonometric interpolation are refereed to [8].

For a non-periodic function $f(x)$ defined over bounded interval such as $[-1, 1]$, one can transform it to a periodic function $F(\phi)$ as follows:

$$F(\phi) = f(\cos \phi), \quad x = \cos \phi, \qquad \phi \in [0, \pi].$$

It is clear that $F(\phi)$ can be interpreted as an even 2π-periodic function of ϕ and be approximated by a trigonometric interpolation in ϕ-space, which can be further transformed back to the summation of a list of Chebyshev polynomials in x [6].

Trigonometric interpolation is believed to be suitable for periodic functions while Chebyshev polynomial interpolation is generally preferred for non-periodic functions in the literature [7].

The aims of the study are twofold. We first develop a trigonometric interpolation algorithm suitable for a non-periodic function defined over a bounded interval as described in Algorithm 2.1. The idea is to extend a target function periodically with same degree of smoothness. As such, the algorithm is expected to converge at the rate described in Theorem 1.2. The test results in Subsection 3.3 suggest that error of trigonometric approximation often exhibits cancellation effect and thus does not propagate into significant compounding errors, a remarkable advantage compared to polynomial-based approximation [6].

Considering the analytic attractiveness of trigonometric polynomial, especially in treatment of differential and integral operations, we expect that Algorithm 2.1 can be used in a wide spectrum. We start the application by using it to approximate an integral. The numerical experiments in Section 4 show that the new method outperforms significantly the classic Trapezoid and Simpson rules, especially in treatment of volatile integrands.

As the second purpose of this study, we show how a trigonometric-interpolation based optimization (TIBO) can be used to solve a non-linear ODE. The fundamental idea is to transform a non-linear ODE to a non-linear algebraic system, which can be solved by an optimization problem; the key ingredient is Fast Fourier Transformation (FFT), which can be used to computer efficiently the gradient vector of the optimization with sufficient number of variables for desired accuracy. As a demonstrative example, we develop Algorithm 4.1 to cope with first-order non-linear ODEs. The numerical experiments in Section 4.2 show that it outperforms significantly the classical Runge-Kutta method (RK4) [5], especially in the treatment of highly oscillating solutions. The same idea has been used to develop an algorithm for the solutions of second-order ODEs. The algorithm has consistent performance as observed in this paper and is flexible to cope with general boundary conditions and attack certain challenging issues such as identifying the solution with a predefined property (e.g. being positive or increasing) when there are multiple solutions [9].

The rest of the paper is structured as follows. Section 2 is devoted to develop Algorithm 2.1 used for non-periodic function. In Section 3, we conduct some numerical experiments to assess the performance of the algorithm. Section 4 is devoted to the applications of Algorithm 2.1. The summary is made on Section 5.

2. Trigonometrical Interpolation of Non-periodic Functions

This section is used to develop a trigonometric interpolation algorithm that can be applied to a general function f over a bounded interval $[s, e]$ with bounded $K+1$-th derivative $f^{(K+1)}(x)$ for some $K \geq 0$.

We can shift $f(x)$ by s and then evenly extend it to $[s-e, e-s]$. It is well-known that such direct extension deteriorates the smoothness at $0, \pm(e-s)$, which leads to a poor convergence performance as confirmed in Section 3. To seek for a smooth periodic extension, we assume that f can be extended smoothly such that $f^{(K+1)}$ exists and is bounded over $[s-\delta, e+\delta]$ for certain $\delta > 0$ and leverage a cut-off smooth function $h(x)$ with following property:

$$h(x) = \begin{cases} 1 & x \in [s, e], \\ 0 & x < s - \delta \text{ or } x > e + \delta. \end{cases}$$

Such cut-off function can be constructed in different ways and we shall adopt one with a closed-form analytic expression as follows:

$$h(x; r, s, e, \delta) = B(\frac{x - (s - \delta)}{\delta}, r) \times B(\frac{e + \delta - x}{\delta}, r), \tag{7}$$

where

$$B(x; r) = \frac{G(x; r)}{G(x; r) + G(1 - x; r)}, \quad G(x; r) = \begin{cases} e^{-\frac{r}{x^2}} & x > 0, \\ 0 & x \leq 0. \end{cases}$$

To see the effect of parameter r, Figure 2 plots left wing of cut-off functions with three scenarios $r \in \{0.1, 0.5, 1\}$ for $(s, e, \delta) = (-1, 1, 1)$. As r increases from 0.1 to 1, h takes more space in $x - axis$ to increase around 0 to near 1 and therefore change is less dramatically over the process, which is preferable. On the other hand, smaller r provides more spaces for $h(x)$ converges to 0 and 1, which is also preferable for the performance of h around s and $s - \delta$.

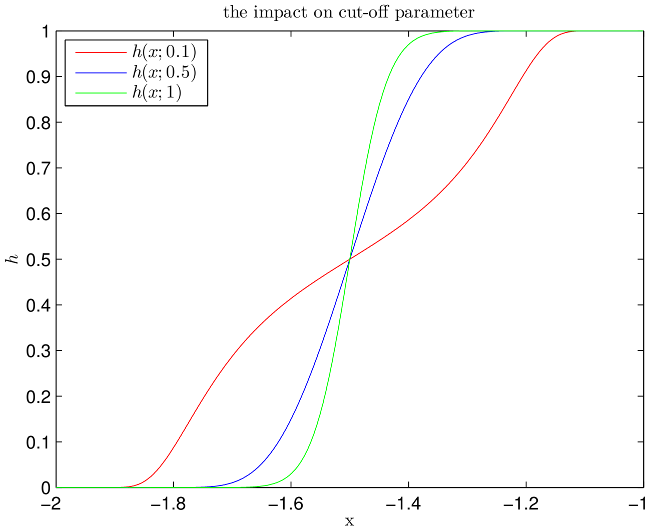

Figure 1. The graphs of h_M over $[s-\delta, s]$ with $(s,e,\delta)=(-1,1,1)$ and $M=2^8$ for three cases: $r=0.1$ (red), $r=0.5$ (blue) and $r=1$ (green).

Trigonometric expansion of h can be useful in applications, and Table I shows max error in two cases with $q\in\{7,8\}$ under $(s,e,\delta)=(-1,1,0.5)$, which recommends $r=0.5$ and we shall take it on all tests reported in this paper. With the cut-off $h(x)$, $h(x)f(x)$ can be

Table I. the max error of interpolant h_M with $(s,e,\delta)=(-1,1,0.5)$

δ	r	error	δ	r	error
7	0.1	1.4E-04	8	0.1	1.0E-06
7	0.5	1.6E-06	8	0.5	2.6E-10
7	1	2.2E-04	8	1	1.7E-07

smoothly extended to $[2s-e-3\delta, s-\delta]$ symmetrically with respect to vertical line $x=s-\delta$. The idea is demonstrated by an example where $f(x)=(x-2.5)^2$ for $x\in[2,3]$ with $(s,e,\delta)=2,3,1$, as shown in Figure 2.

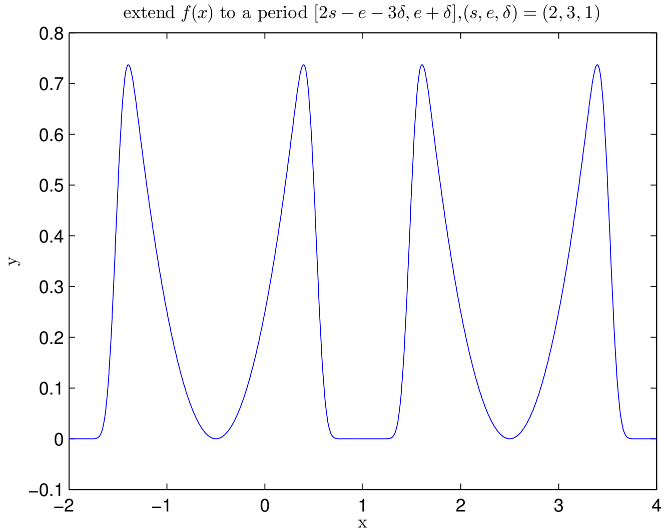

Figure 2. The graphs of the fh's extension $(hf)_{ext}$ over a period $[2s-e-3\delta, e+\delta]$ with $(s,e,\delta)=(2,3,1)$. The function is even after parallel shift left by $s-\delta=1$. Note that $fh=f=(x-2.5)^2$ over $[s,e]$ as expected.

We summarize the enhanced trigonometric interpolation algorithm for non-periodic function as follows:

Algorithm 2.1. *Let $f(x)$ be defined over $[s,e]$.*

1. *Select integers $0<p<q$ such that $f(x)$ can be smoothly extended to $[s-\delta, e+\delta]$, where*

$$\begin{aligned} n &= 2^p, \quad M = 2^q, \quad \lambda = \frac{e-s}{n}, \\ m &= \frac{M-n}{2}, \qquad \delta = m\lambda. \end{aligned}$$

2. *Construct the cut-off function $h(x)$ with parameter (s,e,δ).*

3. *Let*

$$o = s-\delta, \quad b = e+\delta-o, \tag{8}$$

and define $F(x) := h(x+o)f(x+o)$ for $x\in[0,b]$.

4. *Extend $F(x)$ evenly by $F(x)=F(-x)$ for $x\in[-b,0]$ [1]. It is clear that $F(x)$ can be treated as an periodic even function.*

[1] Alternatively, extend $F(x)$ oddly by $F(x)=-F(-x)$ for $x\in[-b,0]$ if odd trigonometric estimation is desired.

5. *Define grid nodes by*

$$x_j = -b + j\lambda, \quad j = 0, 1, \cdots N-1, \quad N = 2M,$$

and apply them to construct trigonometric expansion F_M by Theorem 1.1 [2] *:*

$$F_M(x) = \sum_{0 \le j < M} a_j \cos \frac{j\pi x}{b}.$$

6. *Let*

$$\hat{f}_M(x) = F_M(x - o) = \sum_{0 \le j < M} a_j \cos \frac{j\pi(x-o)}{b}.$$

$\hat{f}$ will be used to denote the extended periodic function by a cut-off function h and $\hat{f}_M$ be the interpolant of $\hat{f}$ by Algorithm 2.1 in the rest of this paper. Clearly, it inherits same smoothness as $f(x)$ does. We shall discuss its performance and applications in Section 3 and 4 respectively.

Figure 2 compares $f = (x - 2.5)^2$ and $\hat{f}_M$ over $[s - \delta, e + \delta]$ with $[s, e] = [2, 3]$. Note that $\hat{f}_M$ recovers f over $[s, e]$ and approaches to 0 around boundary points $s - \delta, e + \delta$ as expected.

[2] Alternatively, $F(x) = \sum_{1 \le j < M} a_j \sin \frac{j\pi x}{b}$ if odd trigonometric interpolant is desired.

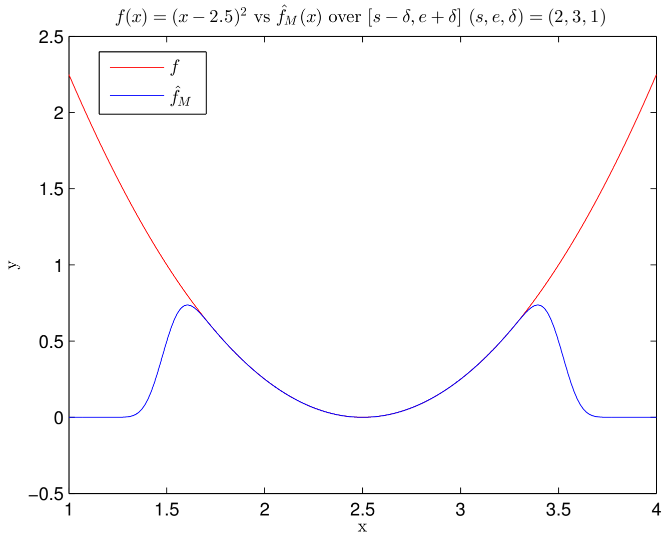

Figure 3. The graphs of $f(x)$ vs $\hat{f}_M(x)$ over $[e-\delta, s+\delta]$ with $(s,e,p,q,\delta) = (2,3,7,8,1)$. The figure is plotted by 2^{12} sample points.

3. Numerical Performance

This section provides some numerical results to test performance of Algorithm 2.1. First, we test convergence performance of periodic function, and show that it is sensitive to smoothness of underlying function f as expected. Secondly, we apply the enhanced algorithm to two sets of typical functions whose values can be highly oscillated and rapidly changed, and show that it does exhibit stable and accurate performance.

3.1. Numerical Performance on Periodic Functions

Let f be the even periodic function with period 2π and be defined as follows over $[-\pi, \pi]$:

$$f(x;d)|_{[-\pi,\pi]} = (1 - (\frac{x}{\pi})^2)^d,$$

where d can be $1, 2$. It is clear that $f(x;1)$ is not differentiable at $\pm\pi$ and $f(x;2)$ is 2-th continuous differentiable. We expect that interpolant of $f_M(x;2)$ has significant better performance than $f_M(x;1)$.

Table II provides max errors at grid points under various settings on d, M. It confirms that performance is sensitive to number of grid points and especially degree of f's smoothness as expected.

Table II. The max errors $|f_M(x) - f(x)|$ and $|f'_M(x) - f'(x)|$

d	M	$\max(\lvert f - f_M\rvert)$	$\max(\lvert f' - f'_M\rvert)$
1	64	6.0E-03	6.4E-01
1	256	1.5E-03	6.4E-01
1	1024	3.6E-04	6.4E-01
2	64	5.6E-07	3.5E-05
2	256	8.3E-09	2.2E-06
2	1024	1.3E-10	1.4E-07

3.2. Numerical Performance on General Functions

Let $\hat{f}$ be the smooth extension described in Section 2 and $\hat{f}_M$ be the trigonometric estimation of $\hat{f}$ by Algorithm 2.1.

We apply Algorithm 2.1 to following functions to test convergence performance.

$$\begin{aligned} y &= \cos\theta x, \quad \theta = 1, 10, 100, \\ y &= x^n, \qquad n = 4, 8, 10. \end{aligned}$$

For any estimation $\tilde{w}(x)$ of a function $w(x)$, define max error in log space over a subset S of the domain of w by

$$E(w) = \max_{x \in S}\{log_{10}|\tilde{w}(x) - w(x)|\}. \tag{9}$$

Let f_M be the interpolant of direct periodic extension of f without using cut-off function. Table III shows such errors up to second derivatives of both f_M and $\hat{f}_M$ for the two functions with the parameters $(s, e, p, q, \delta) = (-1, 1, 7, 8, 1)$, and the subset $S = \{s + k\frac{e-s}{2^{12}}, 0 \leq k \leq 2^{12}\}$. Note that we deliberately make S larger than the set of grid nodes used in Theorem 1.1 to test that $\hat{f}_M$ converges at non-grid nodes. The performance of $\hat{f}_M$ is stable across all the test scenarios and max estimation errors are small, and $f_M(x)$ generates significant errors, especially on derivatives, which confirms the impact of smoothness on performance as expected.

Table III. Max error in log space (EL) with parameters $(s, e, p, q, \delta) = (-1, 1, 7, 8, 1)$.

$para$	$EL(f_M)$	$EL(\hat{f}_M)$	$EL(f'_M)$	$EL(\hat{f}'_M)$	$EL(f''_M)$	$EL(\hat{f}''_M)$
$f = \cos\theta x$						
$\theta = 1$	-3.2	-14.7	0.0	-13.1	6.4	-10.7
$\theta = 10$	-2.4	-14.8	-0.3	-14.2	3.7	-11.8
$\theta = 100$	-1.4	-14.0	-0.3	-14.0	1.3	-11.9
$f = x^n$						
$n = 4$	-2.5	-14.8	0.0	-13.6	5.5	-11.1
$n = 8$	-2.2	-14.3	0.0	-13.1	4.6	-10.6
$n = 10$	-2.1	-14.0	0.0	-12.9	4.4	-10.4

3.3. Discussion on Error behavior of $\hat{f}_M(x)$

In this subsection, we look into error patterns of $\hat{f}_M(x)$ and examine whether the errors exhibit certain “local property”, i.e. an error at a point would not propagate and cause large compounding error at other points. As shown in Section 4.2, polynomial-based methods lack of such local property. We conduct relevant tests on four basic functions power, exponential, sin and cos function. Figure 3.3 shows the normalized differences of consecutive errors defined by $\frac{1}{\max|f(x)|}\{\hat{f}_M(x_i) - f(x_i) - (\hat{f}_M(x_{i-1}) - f(x_{i-1}))\}$. A clear sawtooth pattern is shown in Figure 3.3 for all test cases. Magnitude of error keep reasonable stable, but goes in alternative directions, a strong sign that error is not accumulating, but canceled with each other when variable x moves around. Same phenomena has been observed on the estimation error about solution of non-linear ODE discussed in Observation 1.

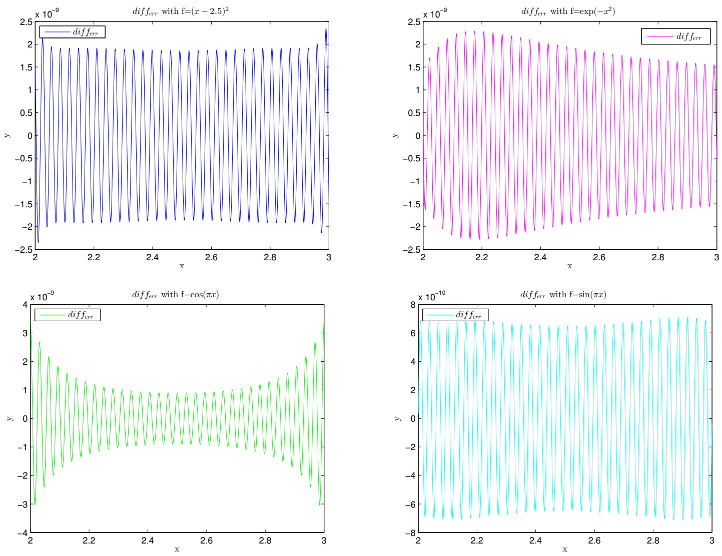

Figure 4. Plots of the normalized difference of consecutive error at grid points with $\lambda = 1/2^6$ with $[s, e] = [2, 3]$. They refer to $(x - 2.5)^2$,e^{-x^2},$\cos(\pi x)$, $\sin(\pi x)$ at top left and right, bottom left and right respectively.

4. Applications

In this section, we provide two applications of Algorithm 2.1. First, $\hat{f}_M$ is used to solve an integral. The test results show that the new method outperforms classic Trapezoid and Simpson methods, especially when the integrands are highly oscillated. Secondly, we demonstrate how the algorithm can be effectively used in optimization by solving first-order non-linear ODEs. The notations in Section 2 will be adapted in this Section.

4.1. Numerical Solution of Integrals

Let f be a $K + 1$ differentiable function over $[s - \delta, e + \delta]$ with $\delta > 0$. We aim to solve an integral $\int_s^e f(u)du$. Applying the approximation $\hat{f}$ described in Algorithm 2.1, the integral can be estimated by

$$\int_s^e f(u)du \approx a_0(e-s) + \sum_{1 \le j < M} \frac{ba_j}{j\pi}(\sin\frac{j\pi(e-o)}{b} - \sin\frac{j\pi(s-o)}{b}), \quad (10)$$

where o, b are defined in Eq. (8).

To see the performance, we conduct two sets of tests with integral x^n and $\cos\theta x$

$$\int_{-1}^{1} x^n dx, \qquad \int_{-1}^{1} \cos\theta x dx.$$

For each test, we use the metric defined in Eq. (9) to compare the performance of three methods: Trig Estimation by Eq. (10), Trapezoid as well as Simpson, and display the results in Table IV and V.

Table IV. The max errors in log space with three methods: Estimation Eq (10), Trapezoid, and Simpson. Note that 2^q denotes the number of grid points in interpolation algorithm with parameter $(s, e, p, q, \delta) = (-1, 1, 7, 8, 1)$.

n	Trig. Estimation	Trapezoid Method	Simpson Method
4	-15.5	-5.0	-10.2
8	-14.3	-4.7	-9.1
10	-14.3	-4.6	-8.7

Table V. Similar comparison as shown in Table IV for integral $\cos\theta x$

θ	Trig. Estimation	Trapezoid Method	Simpson Method
1	-15.4	-5.7	-11.7
10	-16.4	-4.9	-8.9
100	-16.8	-3.9	-5.9

One can see that Simpson outperforms significantly Trapezoid as expected. When n and θ increases, the performance of Simpson and Trapezoid deteriorates as expected since the integrals change more dramatically, especially for $\cos\theta x$. Estimation (10) turns out to be more robust to handle rapid changes and high oscillation of function values as long as the integrand is sufficient smooth.

4.2. Numerical Solution of First Order Differential Equation

In this subsection, we aim to develop an algorithm on numerical solution of a general first order ODE:

$$y'(x) = f(x, y), \quad x \in [s, e] \tag{11}$$

$$y(s) = \xi, \tag{12}$$

where $f(x,y)$ is continuously differential on the range $[s-\delta, e+\delta]\times R^1$. It is well-known that there is a unique solution of Eq. (11)-(12) [2].

Replacing $f(x,y)$ by $f(x+o,y)$ if needed, we assume that $o := s-\delta = 0$ and denote $h(x)$ as the cut-off function in Section 2. Apply h to extend $f(x,y)$ as follows [3]:

$$F(x,u) = \begin{cases} f(x,u)h(x) & \text{if } x \in [0,b], \\ -f(-x,u)h(-x) & \text{if } x \in [-b,0]. \end{cases} \tag{13}$$

We shall search a numerical solution for the extended ODE:

$$u'(x) = F(x,u), \quad x \in [-b,b], \tag{14}$$
$$u(s) = \xi. \tag{15}$$

Since $(u(x)-u(-x))' \equiv 0$, $u(x)$ is even and its derivative $z(x) := u'(x)$ is odd. It is clear that u can be smoothly extended to even periodic function with period $2b$ and $u(x)|_{[s,e]}$ solves Eq. (11)- (12). Let $\{(x_k,z_k)\}_{0\le k<N}$ be a grid set of $z(x)$:

$$x_k = -b + \frac{2b}{N}k, \qquad k = 0,1,\cdots,N-1, \tag{16}$$
$$z_0 = 0, \quad z_{N-k} = -z_k, \qquad 1 \le k < M, \tag{17}$$

and

$$z_M(x) = \sum_{0\le j<M} b_j \sin\frac{j\pi x}{b} \tag{18}$$

be the interpolant of $z(x)$ with

$$b_j = \frac{2}{N}\sum_{k=0}^{N-1}(-1)^j z_k \sin\frac{2\pi jk}{N} = \frac{4}{N}\sum_{k=0}^{M-1}(-1)^j z_k \sin\frac{2\pi jk}{N}, \quad 0 \le j < M. \tag{19}$$

It is clear

$$\frac{\partial b_j}{\partial z_k} = \frac{4}{N}(-1)^j \sin\frac{2\pi jk}{N}, \quad 0 \le j,k < M.$$

u can be approximated based on Eq. (18) by

$$\tilde{u}_M(x) = \sum_{0\le j<M} a_j \cos\frac{j\pi x}{b}, \quad a_j = -\frac{bb_j}{j\pi}, \quad 1 \le j < M, \tag{20}$$

and a_0 can be solved by the initial condition $u(-s) = u(x_{m+n}) = \xi$

$$a_0 = \xi - \sum_{1\le j<M}(-1)^j a_j \cos\frac{2\pi j(m+n)}{N}.$$

[3] We use u to denote the periodic extension of y.

Let 0_M be the M-dim zero vector and define $\frac{1}{0} := 0$. The following notations will be adopted in the rest of this subsection.

$$\begin{aligned}
u_k &= \tilde{u}_M(x_k), \quad F_k = F(x_k, u_k),\\
Z &= \{z_k\}_{0\le k<M}, \quad U = \{u_k\}_{0\le k<M}, \quad F = \{F_k\}_{0\le k<M},\\
DF_k &= \frac{\partial F}{\partial u}(x_k, u_k), \quad DF = \{DF_k\}_{0\le k<M},\\
J &= [0, 1, \frac{1}{2}, \dots, \frac{1}{M-1}, 0_M],\\
\Phi &= [\frac{1}{j}\cos\frac{2\pi j(m+n)}{N}]_{j=0}^{M-1}, \quad \Phi_N = [\Phi, 0_M],\\
\Psi &= \{(z_j - F_j)DF_j\}_{j=0}^{M-1}, \quad \Psi_N = [\Psi, 0_M], \qquad I = sum(\Psi).
\end{aligned}$$

ODE (14)-(15) can be solved by minimizing the following error function:

$$\phi(z_0, z_1, ..., z_{M-1}) = \frac{1}{2M}\sum_{0\le k<M}(z_k - F_k)^2. \tag{21}$$

We need an effective way to calculate its gradient $\frac{\partial\phi}{\partial Z}$ when M, the number of variables of ϕ, is not small.

$$M\frac{\partial\phi}{\partial z_t} = (z_t - F_t) - \sum_{0\le k<M}(z_k - F_k)DF_k\frac{\partial u_k}{\partial z_t}. \tag{22}$$

To cope with $\frac{\partial U}{\partial Z}$ in Eq. (22), we need express U in term of Z. By Eq (19) and $z_0 = z_M = 0$, we obtain for $0 \le k < N$

$$\begin{aligned}
u_k &= a_0 - \frac{2b}{\pi N}\sum_{0\le j<M}\frac{1}{j}\cos\frac{2\pi jk}{N}\sum_{0\le l<N}z_l\sin\frac{2\pi jl}{N} &(23)\\
&= a_0 - \frac{4b}{\pi N}\sum_{0\le l<M}z_l\sum_{0\le j<M}\frac{1}{j}\cos\frac{2\pi jk}{N}\sin\frac{2\pi jl}{N}. &(24)
\end{aligned}$$

The last step is due to $z_l\sin\frac{2\pi jl}{N} = z_{N-1}\sin\frac{2\pi j(N-l)}{N}$. One can rewrite (23) in term of ifft as follows:

$$U = a_0 - \frac{2bN}{\pi}Re\{ifft(J\circ Im\{ifft(Z)\})\}, \tag{25}$$

where $\circ$ denotes the Hadamard product, which applies the element-wise product to two metrics of same dimension. a_0 in (24) can be further interpreted by Z as follows:

$$a_0 = \xi + \frac{2b}{\pi N}\sum_{1\le j<M}\frac{1}{j}\cos\frac{2\pi j(m+n)}{N}\sum_{k=0}^{N-1}z_k\sin\frac{2\pi jk}{N}, \tag{26}$$

which implies

$$\frac{\partial a_0}{\partial z_k} = \frac{4b}{\pi N} \sum_{0 \le j < M} \frac{1}{j} \cos \frac{2\pi j(m+n)\pi}{N} \sin \frac{2\pi jk}{N}, \quad 0 \le k < M. \tag{27}$$

Combining (24) and (27), we obtain

$$\frac{\partial u_k}{\partial z_t} = \frac{4b}{\pi N} \sum_{0 \le j < M} \frac{1}{j} \sin \frac{2\pi jt}{N} (\cos \frac{2\pi j(m+n)}{N} - \cos \frac{2\pi jk}{N}). \tag{28}$$

We are ready to attack the non-trivial term in Eq. (22). Define

$$w_t := \sum_{0 \le k < M} (z_k - F_k) DF_k \frac{\partial u_k}{\partial z_t}.$$

By (28),

$$\begin{aligned} w_t &= \frac{4b}{\pi N} \sum_{0 \le j < M} \Phi_j \sin \frac{2\pi jt}{N} \sum_{0 \le k < M} (z_k - F_k) DF_k \\ &- \frac{4b}{\pi N} \sum_{0 \le j < M} J_j \sin \frac{2\pi jt}{N} \sum_{0 \le k < M} (z_k - F_k) DF_k \cos \frac{2\pi jk}{N}. \end{aligned}$$

The gradient vector (22) can be formulated by FFT as follows:

$$\begin{aligned} W &= \frac{4bI}{\pi} Im(ifft(\Phi_N)) - \frac{4\pi N}{b} Im\{ifft(J \circ Re[ifft(\Psi_N)])\}, \\ \frac{\partial \phi}{\partial Z} &= \frac{1}{M}(Z - F - W[0:M-1]). \end{aligned} \tag{29}$$

One can implement the algorithm by following steps.

Algorithm 4.1. *For a given ODE (11)-(12),*

1. *Select* (p, q, δ) *as in Algorithm 2.1 such that* $f(x, y)$ *can be smoothly extended to* $[s - \delta, e + \delta] \times R$.
2. *Construct the cut-off function* $h(x)$ *with parameter* $(s, e, \delta, r = 0.5)$ *by Eq. (7) and* $F(x, u)$ *by (13).*
3. *Apply an optimization function with the gradient function* $\frac{\partial \phi}{\partial Z}$ *formulated by Eq. (29).* [4]
4. *Apply the opt values of* Z *returned by the optimization function in previous step to calculate required* U *by (25) and (26) and return* $U|_{[s,e]}$.

[4] One can use RK4 to generate the initial values of the optimization function.

Algorithm 4.1 is expected to be efficient since the gradient of the target function can be carried out by $O(N \ln_2 N)$ operations.

In the rest of this subsection, we study the performance of Algorithm 4.1, labeled *intp*, by solving the following ODE

$$y'(x) = f(x, y) + xy + y^2, \quad y(1) = 0, \quad x \in [1, 3] \tag{30}$$

where

$$f(x, y) = \cos\theta x - \theta x \sin\theta x - xy - y^2, \theta \in \{\frac{\pi}{2}, \frac{3\pi}{2}\}.$$

The analytic solution is available as follows:

$$y(x) = x \cos\theta x.$$

We shall compare *intp* to the classic Runge–Kutta method, labeled $rk4$. It is well-known that local truncation error of $rk4$ is on the order of $O(\lambda^5)$ and hence the total accumulated error is supposed to be $O(\lambda^4)$ [5].

In addition, we implement a benchmark method, labeled as *benc*, by adjusting $rk4$ in a "cheating" way that y_{n+1} is estimated by the true value Y_n at x_n. In this way, *benc* prevents accumulating error and is supposed to be on the order of $O(\lambda^5)$.

The overall performance is reported in Table VI, which includes the max magnitude of errors at grid nodes $\{x_j\}_{j=0}^{N-1}$ under three methods: *intp*, rk_4 and *benc*. To see the performance of *intp* at non-grid nodes, the max is taken over the set obtained by applying identified $\tilde{u}_M$ (see Eq. (20)) to the grid points with step size $\lambda/4$. We also look into changes

Table VI. The max magnitudes of three sets of errors and the optimization error described above. Algorithm 4.1 is implemented with $q = 7, p = 6, \delta = 1$.

θ	*intp*	$rk4$	*benc*
$\pi/2$	3.2E-09	7.7E-07	3.0E-08
$3\pi/2$	4.8E-07	2.1E-03	1.1E-05

of consecutive errors for three covered methods similar as we did in Section 3.3. Figure 5 plots such changes defined by

$$\{\hat{f}_M(x_i) - f(x_i) - (\hat{f}_M(x_{i-1}) - f(x_{i-1}))\}.$$

for *intp*, *benc* and $rk4$.

One can observe that $rk4$ has worst performance based on max error, especially for $\theta = 3\pi/2$; and *intp* also outperforms *benc* noticeably. In addition, Figure 5 shows

Observation 1. *The error of $rk4$ compounds and leads to significant aggregated error toward e. The error of intp moves in sawtooth around* 0 *without compounding effect.*

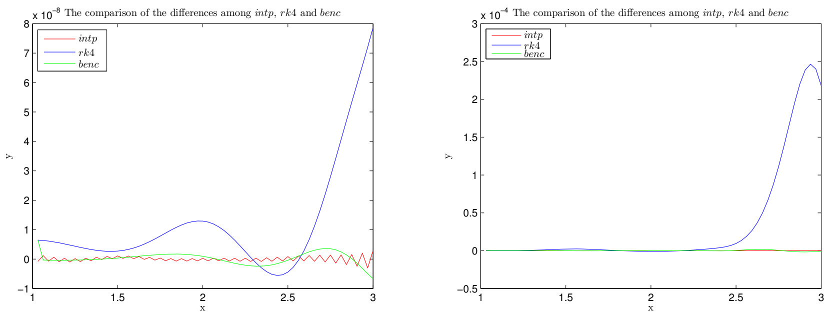

Figure 5. The comparison of changes of consecutive errors among *intp*, *benc*, $rk4$ for $\theta = \pi/2$ on the left panel and $\theta = 3\pi/2$ on the right panel.

5. Conclusions

In this paper, we propose a new trigonometric interpolation algorithm to estimate general functions defined on bounded intervals. The algorithm is computationally effective by leveraging the power of FFT. It achieves optimal convergent rates not only for a target function, but also for the associated high-order derivatives of the target function. The algorithm has been applied to estimate integrals and certain testing results show that it outperforms significantly Trapezoid and Simpson method.

We further show how a trigonometric-interpolation based optimization can be used to solve a non-linear ODE and demonstrate the idea by developing a new algorithm to cope with first-order non-linear ODEs. The test results show that it significantly outperforms the classic Runge-Kutta algorithm.

References

1. A. P. Austin, *On Trigonometric Interpolation In an Even Number of Points*, Electronic Transactions on Numerical Analysis, 58 (2023), 271–288.

2. W.E. Boyce and R.C. Diprima, *Elementary Differential Equations and Boundary Value Problems*, John Wiley and Sons, 1986.
3. James W. Cooley and John W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, SIAM Journal on Matrix Analysis and Applications, 19(4) (1990) 297–301.
4. R. Kress, *Numerical Analysis, volume 181 of Graduate Texts in Mathematics*, Springer-Verlag, 1998.
5. W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes*, Cambridge University Press, 2007.
6. V. S. Ryaben'kii and S. V. Tsynkov, *A Theoretical Introduction to Numerical Analysis*, Chapman & Hall/CRC, 2007.
7. G. B. Wright, M. Javed, H. Montanelli, and L. N. Trefethen *Extension of Chebfun to Periodic Functions*, SIAM Journal on Matrix Analysis and Applications, 37(5) (2015), pp. 554-573.
8. X. Zou, *On Trigonometric Approximation and Its Applications*, arxiv:2505.02330
9. X. Zou, *Trigonometric Interpolation Based Optimization for Second Order Non-Linear ODE with Mixed Boundary Conditions*, arXiv:2504.19280