Contents lists available at SciVerse ScienceDirect

Applied and Computational Harmonic Analysis

www.elsevier.com/locate/acha

Letter to the Editor

A fast algorithm for multilinear operators

Haizhao Yang [a], Lexing Ying [b,*]

[a] Department of Mathematics, The University of Texas at Austin, Austin, TX, United States
[b] Department of Mathematics and ICES, The University of Texas at Austin, Austin, TX, United States

ARTICLE INFO

Article history:
Received 18 January 2012
Revised 19 March 2012
Accepted 24 March 2012
Available online 30 March 2012
Communicated by M.V. Wickerhauser

Keywords:
Multilinear operators
Fast Fourier transform
Multiscale decomposition
Low-rank approximation

ABSTRACT

This paper introduces a fast algorithm for computing multilinear integrals which are defined through Fourier multipliers. The algorithm is based on generating a hierarchical decomposition of the summation domain into squares, constructing a low-rank approximation for the multiplier function within each square, and applying an FFT based fast convolution algorithm for the computation associated with each square. The resulting algorithm is accurate and has a linear complexity, up to logarithmic factors, with respect to the number of the unknowns in the input functions. Numerical results are presented to demonstrate the properties of this algorithm.

1. Introduction

This paper is concerned with a class of multilinear integrals. Let $m(\xi_1,\dots,\xi_k)$ for $\xi_i \in \mathbb{R}^d$ be a bounded *multiplier function* that is smooth away from the origin. Given k functions $f_1,\dots,f_k$ in the Schwartz space $S(\mathbb{R}^d)$, we define the multilinear operator $T:(f_1,\dots,f_k)\to T(f_1,\dots,f_k)$ by

$$T(f_1,\dots,f_k)(x)=\int_{\mathbb{R}^d} d\xi\left(e^{2\pi ix\xi}\int_{\xi=\xi_1+\cdots+\xi_k} m(\xi_1,\dots,\xi_k)\hat{f}_1(\xi_1)\dots\hat{f}_k(\xi_k)\,d\xi_1\dots d\xi_{k-1}\right),$$

or, equivalently in the Fourier domain,

$$\widehat{T(f_1,\dots,f_k)}(\xi)=\int_{\xi=\xi_1+\cdots+\xi_k} m(\xi_1,\dots,\xi_k)\hat{f}_1(\xi_1)\dots\hat{f}_k(\xi_k)\,d\xi_1\dots d\xi_{k-1}.$$

This type of multilinear integral operator has been studied extensively in harmonic analysis. When $m(\xi_1,\dots,\xi_k)=1$, then $T(f_1,\dots,f_k)=f_1\dots f_k$ reduces to the pointwise multiplication operator. An important class of multiplier functions $m(\xi_1,\dots,\xi_k)$ that plays an important role is the symbols of order 0 in the sense that

$$\left|\nabla^j m(\xi_1,\dots,\xi_k)\right|\lesssim\left|(\xi_1,\dots,\xi_k)\right|^{-|j|} \tag{1}$$

for any multiindex $j=(j_1,\dots,j_k)\geqslant 0$ (see Fig. 1 (left) for an example). The Coifman–Meyer theorem [2,3] states that, when m is a symbol of order 0, the operator T maps $L^{p_1}\times\cdots\times L^{p_k}\to L^p$ if $1<p_i<\infty$, $1\leqslant i\leqslant k$, $1/p_1+\cdots+1/p_k=1/p$, and

* Corresponding author.
E-mail address: lexing@math.utexas.edu (L. Ying).

1063-5203/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.acha.2012.03.010

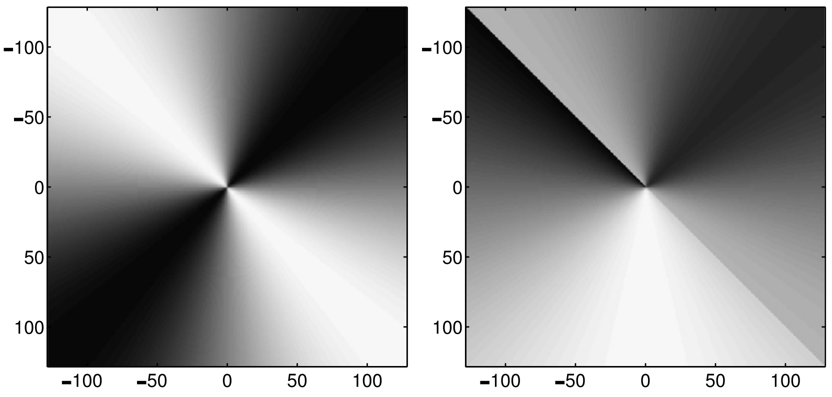

Fig. 1. Symbols with $N = 256$. Left: a symbol of order 0. Right: a piecewise symbol of order 0 with a diagonal discontinuity.

$1 < p < \infty$. Sometimes one also considers a multiplier $m(\xi_1, \ldots, \xi_k)$ that has linear discontinuities through the origin but satisfies (1) otherwise. For example, if $m(\xi_1, \xi_2) = \frac{1}{i}(\operatorname{sgn}(\xi_2) - \operatorname{sgn}(\xi_1 + \xi_2))$, then $T(f_1, f_2) = f_1 H(f_2) - H(f_1 f_2)$ where H stands for the Hilbert transform. Fig. 1 (right) gives an example of such a symbol with a diagonal discontinuity.

In this paper, we are concerned with the numerical computation of such multiplier integrals, particularly the 1D bilinear case (i.e., $d = 1$ and $k = 2$)

$$\widehat{T(f_1, f_2)}(\xi) = \int\limits_{\xi=\xi_1+\xi_2} m(\xi_1, \xi_2)\hat{f}_1(\xi_1)\hat{f}_2(\xi_2)\,d\xi_1 = \int\limits_{\Omega} m(\xi_1, \xi - \xi_1)\hat{f}_1(\xi_1)\hat{f}_2(\xi - \xi_1)\,d\xi_1. \tag{2}$$

Let us start by introducing a discrete analog of the multiplier operator. Let N be an integer which is also an integral power of 2 for simplicity. We define the Cartesian grid X and the associated Fourier grid Ω:

$$X = \left\{\frac{0}{N}, \ldots, \frac{N-1}{N}\right\} \quad \text{and} \quad \Omega = \left\{-\frac{N}{2}, \ldots, \frac{N}{2} - 1\right\}.$$

For any discrete function f defined on X, its Fourier transform $\hat{f}(\xi)$ is defined by

$$\hat{f}(\xi) = \frac{1}{N}\sum_{x \in X} e^{-2\pi i x\xi} f(x)$$

and the inverse Fourier transform is

$$f(x) = \sum_{\xi \in \Omega} e^{2\pi i x\xi}\hat{f}(\xi).$$

For two discrete functions defined on X, say f_1, f_2, the discrete multilinear operator, also denoted by T, computes a function $u(\xi)$ defined by

$$u(\xi) := \widehat{T(f_1, f_2)}(\xi) = \sum_{\xi=\xi_1+\xi_2,\ \xi_1,\xi_2 \in \Omega} m(\xi_1, \xi_2)\hat{f}_1(\xi_1)\hat{f}_2(\xi_2),$$

for $\xi \in \Omega$, where the summation in the ξ variable is taken modulo N (i.e., $\xi = \xi_1 + \xi_2$ for $\xi, \xi_1, \xi_2 \in \Omega$ if and only if $\xi \equiv \xi_1 + \xi_2 \pmod N$). When f_1 and f_2 are compactly supported in $[0, 1]$ and nearly band-limited, the discrete operator defined above provides an accurate approximation to the continuous one when N is sufficiently large.

The direct computation of the discrete operator would go through all $\xi \in \Omega$ and, for each fixed ξ, take the sum over all $\xi_1 \in \Omega$ (see (2)). Since $|\Omega| = N$, this naive algorithm takes $O(N^2)$ steps, which can be quite expensive for large values of N. In this paper, we introduce a fast algorithm of which the complexity scales essentially linear with respect to N for a suitable subset of the symbols of order 0.

1.1. *Outline of the approach*

We restrict ourselves to an analytic subset of symbols of order 0, where we call m an *analytic* symbol of order 0 if

$$\left|\nabla^j m(\xi_1, \xi_2)\right| \leqslant D \cdot C^{|j|} \cdot j! \cdot \left|(\xi_1, \xi_2)\right|^{-|j|}$$

away from the origin for any multiindex $j = (j_1, \ldots, j_k) \geqslant 0$. Here C and D are uniform constants independent of j. Compared to the definition of symbols of order 0, we have more precise control of the norm of the derivatives of the symbol. For such a symbol $m(\xi_1, \xi_2)$, let us first consider a square $S = S_1 \times S_2$ in the (ξ_1, ξ_2) plane, where S_1 and S_2 are two intervals in ξ_1 and ξ_2 coordinates, respectively. Let w_S be the width of S and d_S be the distance between S and the origin. As we shall see, if w_S/d_S is less than a constant depending on C and D, then $m(\xi_1, \xi_2)$ is numerically low-rank. More specifically, for any $\varepsilon > 0$, there exists an integer $t = O(\log(1/\varepsilon))$ and two classes of functions $\{\alpha^S_{1,p}(\xi_1)\}_{1 \leqslant p \leqslant t}$ and $\{\alpha^S_{2,p}(\xi_2)\}_{1 \leqslant p \leqslant t}$ such that

$$\left| m(\xi_1, \xi_2) - \sum_{p=1}^{t} \alpha^S_{1,p}(\xi_1) \alpha^S_{2,p}(\xi_2) \right| \leqslant \varepsilon$$

for any $(\xi_1, \xi_2) \in S$. Here the superscript S emphasizes dependence on the square S.

Once the low-rank approximation is identified, the partial sum $u^S(\xi)$ associated with the square S,

$$u^S(\xi) = \sum_{\xi = \xi_1 + \xi_2,\, (\xi_1, \xi_2) \in S} m(\xi_1, \xi_2) \hat{f}_1(\xi_1) \hat{f}_2(\xi_2),$$

can be approximated with

$$u^S(\xi) \approx \sum_{p=1}^{t} \sum_{\xi = \xi_1 + \xi_2,\, (\xi_1, \xi_2) \in S} \big(\alpha^S_{1,p}(\xi_1) \hat{f}_1(\xi_1)\big)\big(\alpha^S_{2,p}(\xi_2) \hat{f}_2(\xi_2)\big).$$

For each fixed p, the discrete convolution can be performed with an FFT in $O(w_S \log w_S)$ steps.

The above discussion addresses the computation for one single square S. For the whole computation, we generate a hierarchical decomposition that subdivides the domain $[-N/2, N/2]^2$ into multiple squares S that satisfy the condition on w_S/d_S and then apply the fast convolution algorithm to each square. As we shall see, the number of square scales like $O(\log N)$ since the ratio bound on w_S/d_S is a constant independent of N. As a result, the overall complexity of the algorithm is $O(N \log N \log(1/\varepsilon))$. This algorithm can be easily generalized to piecewise analytic symbols of order 0, for which the complexity scales like $O(N \log^2 N \log(1/\varepsilon))$ as shown in Section 2.2.

1.2. Related work

Multilinear operators have been intensively studied in harmonic analysis. However, there has been little research on their numerical application and the current work is probably the first attempt at fast evaluation of such operators. However, the main idea behind the proposed algorithm, i.e., generating hierarchical decomposition of the integral operator and applying low-rank and/or FFT based techniques is well known.

Hairer, Lubich, and Schlichte considered the fast evaluation of temporal convolution in [6]. The summation domain, which was geometrically a triangle, was partitioned hierarchically into squares and the computation associated with each square was accelerated using FFT. In [1,7,8], the authors constructed exponential-based low-rank approximations of the convolution kernel so that the integration could be performed using explicit time-stepping.

In [11], Fomel and Ying considered the computation of partial Fourier transforms where the summation frequencies are spatially dependent, which have several applications in high frequency wave propagation and computational seismology. The computational domain is again hierarchically partitioned into manageable components. In 1D, the summation associated with each component is commutated using fractional Fourier transform, while in higher dimensions, the computation of each component is accelerated using the butterfly algorithm [10].

The algorithm of this paper is very much along the same line of thought. The rest of this paper is organized as follows: we present the fast computation of multilinear operators in Section 2. In Section 3, some numerical examples are provided. Finally, we talk about the conclusion and future work in Section 4.

2. Algorithm description

In this section, we first consider an analytic symbol $m(\xi_1, \xi_2)$ and then extend the algorithm to piecewise analytic symbols with linear discontinuities. Finally, we comment on more general cases with $d \geqslant 2$ and $k \geqslant 3$.

2.1. Analytic symbols

We shall work with analytic symbols of order 0 given by the following definition.

Definition 2.1. A bounded function $m(\xi_1, \xi_2)$ is an analytic symbol of order 0 if

$$\left|\nabla^j m(\xi_1, \xi_2)\right| \leqslant D \cdot C^{|j|} \cdot j! \cdot \left|(\xi_1, \xi_2)\right|^{-|j|}$$

for some constants C and D, for all $j \geqslant 0$.

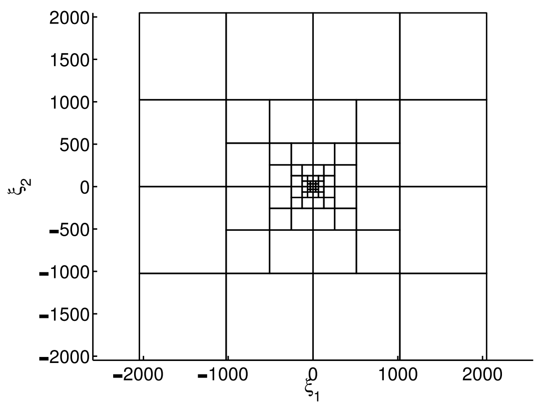

Fig. 2. Hierarchical decomposition of the summation domain for a smooth symbol with $N = 4096$.

The computation of (2) involves all possible pairs (ξ_1, ξ_2) for $\xi_1, \xi_2 \in \Omega$, which are the integer points in the domain $S_0 = [-N/2, N/2)^2$. The main idea of the algorithm is to partition this summation domain S_0 hierarchically into appropriate squares and then apply fast algorithms to speed up the computation associated with each square.

For each square S, we define w_S to be its width and d_S to be its distance to the origin. This decomposition step splits S_0 recursively into squares S until either $w_S/d_S \leqslant 2/C$ or w_S is less than or equal to a prescribed constant width w_0. More precisely, the algorithm proceeds as follows.

Algorithm 2.2 *(Hierarchical decomposition of an analytic symbol).*

```
1: Initialize the queue $\mathcal{Q}$ to contain only the square $S_0 = [-N/2, N/2)^2$ and the set $\mathcal{S}$ to be empty, and let $w_0$ be some
   small positive constant
2: while $\mathcal{Q}$ is not empty do
3:   Pop a square $S$ from $\mathcal{Q}$
4:   if $w_S/d_S \leqslant 2/C$ or $w_S \leqslant w_0$ then
5:     Put $S$ into $\mathcal{S}$
6:   else
7:     Partition $S$ uniformly into four squares and put them into $\mathcal{Q}$
8:   end if
9: end while
```

At the end of this algorithm, the union of the squares in $\mathcal{S}$ is equal to $[-N/2, N/2]^2$ (see Fig. 2 for a typical example). For each square S in $\mathcal{S}$, we define the partial sum $u^S(\xi)$ to be the summation associated with S, i.e.,

$$u^S(\xi) := \sum_{\xi=\xi_1+\xi_2,\, (\xi_1,\xi_2)\in S} m(\xi_1, \xi_2)\hat{f}_1(\xi_1)\hat{f}_2(\xi_2),$$

such that

$$u(\xi) = \widehat{T(f_1, f_2)}(\xi) = \sum_{S\in\mathcal{S}} u^S(\xi).$$

There are clearly two classes of squares in $\mathcal{S}$. The first class consists of small squares of size $w_S \leqslant w_0$. There are only $O(1)$ of them and we may simply use direct computation to evaluate their contribution. The second class consists of squares that satisfy $w_S/d_S \leqslant 2/C$, of which there are only $O(\log N)$ by construction (see Fig. 2). The following theorem provides the basis for speeding up the computation associated with these squares.

Theorem 2.3. *Let $m(\xi_1, \xi_2)$ be an analytic symbol of order 0 with constants C, D. Consider a square $S = S_1 \times S_2$ in the (ξ_1, ξ_2) plane. Let $w_S = |S_1| = |S_2|$ be the width of S and d_S be the distance between S and the origin. Then if $w_S/d_S \leqslant 2/C$, then for any $\varepsilon > 0$ there exist $t = O_{C,D}(\log(1/\varepsilon))$ and functions $\{\alpha^S_{1,p}(\xi_1)\}_{1\leqslant p\leqslant t}$ and $\{\alpha^S_{2,p}(\xi_2)\}_{1\leqslant p\leqslant t}$ such that*

$$\left| m(\xi_1, \xi_2) - \sum_{p=1}^{t} \alpha^S_{1,p}(\xi_1)\alpha^S_{2,p}(\xi_2) \right| \leqslant \varepsilon$$

for any $(\xi_1, \xi_2) \in S$.

Proof. The proof relies on Chebyshev interpolation. Let f be a smooth function on $B=[a,b]$ and $\mathcal{I}_n^B$ be the operator of n point Chebyshev interpolation. Then the following estimate holds (see [9] for example):

$$\left\| f-\mathcal{I}_n^B f\right\|_\infty \leqslant \frac{(b-a)^{n+1}}{2^{2n+1}(n+1)!}\max_{\xi\in B}\left|f^{(n+1)}(\xi)\right|$$

and

$$\left\|\mathcal{I}_n^B\right\|_{\infty\to\infty}\leqslant 1+\frac{2}{\pi}\log(n+1).$$

Let $\mathcal{I}_t^{S_1}$ be the Chebyshev interpolation operator in the ξ_1 direction on interval S_1 with t points and similarly $\mathcal{I}_t^{S_2}$ be the one in the ξ_2 direction on S_2. Using the fact that $|\nabla^j m(\xi_1,\xi_2)|\leqslant D\cdot C^{|j|}\cdot j!\cdot|(\xi_1,\xi_2)|^{-|j|}$, we find that

$$\begin{aligned}
\left\|m-\mathcal{I}_t^{S_1}\mathcal{I}_t^{S_2}m\right\|_\infty &= \left\|m-\mathcal{I}_t^{S_1}m\right\|_\infty+\left\|\mathcal{I}_t^{S_1}m-\mathcal{I}_t^{S_1}\mathcal{I}_t^{S_2}m\right\|_\infty\\
&\leqslant \left\|m-\mathcal{I}_t^{S_1}m\right\|_\infty+\left(1+\frac{2}{\pi}\log(t+1)\right)\left\|m-\mathcal{I}_t^{S_2}m\right\|_\infty\\
&= \left(2+\frac{2}{\pi}\log(t+1)\right)2\left(\frac{w_S}{4}\right)^{t+1}\frac{1}{(t+1)!}DC^{t+1}(t+1)!\frac{1}{d_S^{t+1}}\\
&= \left(2+\frac{2}{\pi}\log(t+1)\right)2D\left(\frac{w_S C}{4d_S}\right)^{(t+1)}\\
&\leqslant \left(2+\frac{2}{\pi}\log(t+1)\right)2D\left(\frac{1}{2}\right)^{t+1},
\end{aligned}$$

where we have used the condition $w_S/d_S\leqslant 2/C$ in the last line. In order to make the final estimate bounded by ε from above, we can choose $t=2\log(4D/\varepsilon)$.

The Chebyshev interpolation in fact provides a low-rank approximation. To see this, let $\xi_i^{S_1}$ be the i-th Chebyshev point in interval S_1 and $L_{t,i}^{S_1}$ to be the i-th Lagrange basis function of the Chebyshev grid on S_1. Define $\xi_i^{S_2}$ and $L_{t,i}^{S_2}$ for the ξ_2 variable in a similar way. Then,

$$\mathcal{I}_t^{S_1}\mathcal{I}_t^{S_2}m=\sum_{i_1,i_2=1}^{t}L_{t,i_1}^{S_1}(\xi_1)L_{t,i_2}^{S_2}(\xi_2)m\left(\xi_{i_1}^{S_1},\xi_{i_2}^{S_1}\right)=\sum_{p=1}^{t}\left(\sum_{i_1=1}^{t}L_{t,p}^{S_2}(\xi_2)m\left(\xi_{i_1}^{S_1},\xi_p^{S_2}\right)\right)L_{t,p}^{S_2}(\xi_2).$$

Now, defining

$$\alpha_{1,p}^S(\xi_1)=\sum_{i_1=1}^{t}L_{t,p}^{S_2}(\xi_2)m\left(\xi_{i_1}^{S_1},\xi_p^{S_2}\right)\quad\text{and}\quad\alpha_{2,p}^S(\xi_2)=L_{t,p}^{S_2}(\xi_2)$$

finishes the proof. □

This theorem explicitly constructs a low-rank approximation from Chebyshev interpolation. In practice, the rank of this approximation is far from optimal since its construction only exploits the smoothness of the multiplier $m(\xi_1,\xi_2)$. A more effective approximation is the pseudoskeleton decomposition [5], where the functions $\{\alpha_{1,p}^S(\xi_1)\}$ and $\{\alpha_{2,p}^S(\xi_2)\}$ behave roughly like $m(\cdot,\xi_p^{S_1})$ and $m(\xi_p^{S_2},\cdot)$ for some carefully chosen $\{\xi_p^{S_1}\}$ and $\{\xi_p^{S_2}\}$, respectively. A randomized procedure for constructing pseudoskeleton decompositions was proposed in [4], and empirically it has a complexity which is only proportional to $O(w_S)$. A brief outline of this procedure is given in Appendix A for completeness. In our numerical experiment, in Section 3, this procedure is used to generate the low-rank approximation for S.

The importance of the low-rank approximation is that, once $\{\alpha_{1,p}^S(\xi_1)\}$ and $\{\alpha_{2,p}^S(\xi_2)\}$ are available, the partial sum $u^S(\xi)$ associated with the square S can be approximated by

$$u^S(\xi)\approx\sum_{p=1}^{t}\sum_{\xi=\xi_1+\xi_2,\,(\xi_1,\xi_2)\in S}\left(\alpha_{1,p}^S(\xi_1)\hat{f}_1(\xi_1)\right)\left(\alpha_{2,p}^S(\xi_2)\hat{f}_2(\xi_2)\right).$$

First, we observe that $u^S(\xi)$ is zero if $\xi\notin S_1+S_2$. Therefore, when S_1 and S_2 are short intervals, the above sum will only update a small portion of ξ. Secondly, since the second sum is a discrete convolution, it can be computed in linear cost with the help of FFTs. More precisely, the algorithm goes as follows.

Algorithm 2.4 *(Fast computation of the partial sum $u^S(\xi)$ associated with a square S).*

```
1: for $p = 1, \dots, t$ do
2:   Compute $\alpha^S_{1,p}(\xi_1)\hat{f}_1(\xi_1)$ for $\xi_1 \in S_1$, extend by zero to a vector of length $2w_S$, and apply an FFT to the result.
3:   Compute $\alpha^S_{2,p}(\xi_2)\hat{f}_2(\xi_2)$ for $\xi_2 \in S_2$, extend by zero to a vector of length $2w_S$, and apply an FFT to the result.
4:   Multiply the two results and apply an inverse FFT to the product.
5:   Add the result to the $\xi$ locations at $S_1 + S_2$.
6: end for
```

Since this algorithm only uses FFTs of size $O(w_S)$ and $t = O(\log(1/\varepsilon))$, the total cost is $O(w_S \log w_S \log(1/\varepsilon))$. Piecing together all components, the full algorithm proceeds as follows:

Algorithm 2.5 *(Fast application of an analytic symbol).*

```
1: Apply Algorithm 2.2 to construct a hierarchical decomposition.
2: for each $S$ in $\mathcal{S}$ do
3:   if $S$ satisfies $w_S/d_S \leqslant 2/C$ then
4:     Compute its contribution using Algorithm 2.4.
5:   else
6:     Compute its contribution using direction computation.
7:   end if
8: end for
```

The following theorem provides the complexity and error estimates of the proposed algorithm.

Theorem 2.6. *For an analytic symbol $m(\xi_1, \xi_2)$, the proposed algorithm has a complexity of order $O(N \log N \log(1/\varepsilon))$ and the ℓ^∞ error of $\widehat{T(f_1, f_2)}(\xi)$ is bounded by*

$$\varepsilon \min\bigl(\|\hat{f}_1\|_1 \|\hat{f}_2\|_\infty, \|\hat{f}_1\|_\infty \|\hat{f}_2\|_1\bigr).$$

Proof. Let us consider the complexity first. Let $N = 2^n$. The computational cost of the squares S with $w_S \leqslant w_0$ is clearly constant since there are only $O(1)$ of them and w_0 is a prescribed constant. Therefore, the main cost is from the rest of the squares with $w_S/d_S \leqslant 2/C$. All squares in $\mathcal{S}$ are of size 2^s and, for each fixed s, there are $O(1)$ box of with 2^s. Therefore, the cost is bounded by

$$\sum_{s=0}^{n} O\bigl(2^s \log\bigl(2^s\bigr) \log(1/\varepsilon)\bigr) = O\bigl(N \log N \log(1/\varepsilon)\bigr).$$

To estimate the pointwise error, let us define $m_a(\xi_1, \xi_2)$ as our low-rank approximation to $m(\xi_1, \xi_2)$. By construction, we have

$$\bigl|m(\xi_1, \xi_2) - m_a(\xi_1, \xi_2)\bigr| \leqslant \varepsilon$$

for any (ξ_1, ξ_2). Therefore the error is bounded by

$$\begin{aligned}\sum_{\xi=\xi_1+\xi_2} \bigl|m(\xi_1, \xi_2) - m_a(\xi_1, \xi_2)\bigr|\bigl|\hat{f}_1(\xi_1)\bigr|\bigl|\hat{f}_2(\xi_2)\bigr| &\leqslant \varepsilon \sum_{\xi=\xi_1+\xi_2} \bigl|\hat{f}_1(\xi_1)\bigr|\bigl|\hat{f}_2(\xi_2)\bigr| \\ &\leqslant \varepsilon \min\bigl(\|\hat{f}_1\|_1 \|\hat{f}_2\|_\infty, \|\hat{f}_1\|_\infty \|\hat{f}_2\|_1\bigr)\end{aligned}$$

by Hölder's inequality. □

2.2. Piecewise analytic symbols

We now extend this algorithm to the case where the symbol has linear discontinuities through the origin.

Definition 2.7. A bounded function $m(\xi_1, \xi_2)$ is a piecewise analytic symbol of order 0 if there exist lines $L_1, \dots, L_\ell$ that go through the origin in the (ξ_1, ξ_2) plane and partition the plane into regions $R_1, \dots, R_{2\ell}$, and within each region

$$\bigl|\nabla^j m(\xi_1, \xi_2)\bigr| \leqslant D \cdot C^{|j|} \cdot j! \cdot \bigl|(\xi_1, \xi_2)\bigr|^{-|j|}$$

for some uniform constants C and D for all $j \geqslant 0$.

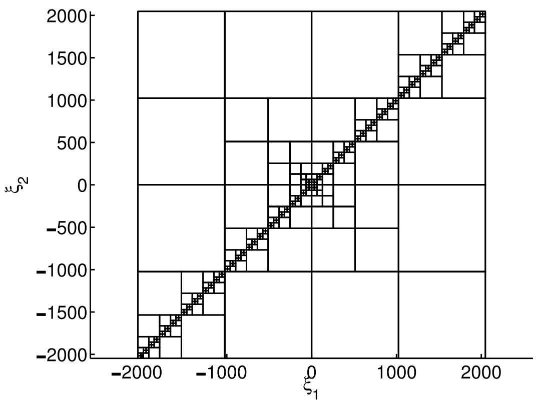

Fig. 3. Hierarchical decomposition of the summation domain for a piecewise smooth symbol with a diagonal discontinuity for $N = 4096$.

Relative to the smooth case, the only complication arises when a square S overlaps with more than one region. Due to its discontinuities, the symbol $m(\xi_1, \xi_2)$ is no longer numerically low-rank in general. Therefore, one needs to further partition these squares. The resulting hierarchical decomposition is very similar to the one for smooth symbols, except for a minor modification on the partitioning criteria.

Algorithm 2.8 *(Hierarchical decomposition of a piecewise analytic symbol).*

```
1: Initialize the queue $\mathcal{Q}$ to contain the square $[-N/2, N/2]^2$ and the set $\mathcal{S}$ to be empty. Let $w_0$ be a small constant
2: while $\mathcal{Q}$ is not empty do
3:   Pop a square $S$ from $\mathcal{L}$
4:   if $w_S/d_S \leqslant 2/C$ or ($w_S \leqslant w_0$ and $\mathrm{dist}(S, L_i) > 0$ for all $L_i$) then
5:     Put $S$ into $\mathcal{S}$
6:   else
7:     Partition $S$ uniformly into four squares and put them into $\mathcal{Q}$
8:   end if
9: end while
```

Now in the final set $\mathcal{S}$, the small squares of width w_0 are close to either the origin or the lines L_i. For any fixed s, there can now be as many as $O(N/2^s)$ squares of width 2^s due to the linear discontinuities. See Fig. 3 for an example.

Theorem 2.9. *For a piecewise analytic symbol $m(\xi_1, \xi_2)$, the proposed algorithm has a complexity of order $O(N \log^2 N \log(1/\varepsilon))$ and the error of $T\widehat{(f_1, f_2)}(\xi)$ is bounded in the infinity norm by*

$$\varepsilon \min\bigl(\|\hat{f}_1\|_1 \|\hat{f}_2\|_\infty, \|\hat{f}_1\|_\infty \|\hat{f}_2\|_1\bigr).$$

Proof. Let us consider the complexity first. Let $N = 2^n$. All the squares in $\mathcal{S}$ are of size 2^s and, for each fixed s, there are $O(N/2^s)$ boxes of width 2^s. Therefore, the total cost is bounded by

$$\sum_{s=0}^{n} O\bigl(N/2^s \cdot 2^s \log\bigl(2^s\bigr) \log(1/\varepsilon)\bigr) = O\bigl(N \log^2 N \log(1/\varepsilon)\bigr).$$

The error estimate is the same as in the proof of the previous theorem. □

2.3. Extensions

The algorithms for smooth symbols can naturally be extended to more general cases with $d \geqslant 2$ and $k \geqslant 3$. Parallel to the discretization of the 1D case, we define grids X and Ω as

$$X = \left\{\frac{0}{N}, \dots, \frac{N-1}{N}\right\}^d \quad \text{and} \quad \Omega = \{-N/2, \dots, N/2 - 1\}^d$$

and the discrete operator T as

$$T\widehat{(f_1, \dots, f_k)}(\xi) = \sum_{\xi = \xi_1 + \cdots + \xi_k,\, \xi_1, \dots, \xi_k \in \Omega} m(\xi_1, \dots, \xi_k) \hat{f}_1(\xi_1) \dots \hat{f}_k(\xi_k)$$

for $f_1, \dots, f_k$ defined on X. A direct computation clearly takes $O(N^{dk})$ steps. When $m(\xi_1, \dots, \xi_k)$ is an analytic symbol of order 0, we can again prove the theorem using Chebyshev interpolation.

Theorem 2.10. *Let $m(\xi_1, \dots, \xi_k)$ be an analytic symbol of order 0 with constants C, D. Consider a dk-dimensional hypercube $S = S_1 \times \dots \times S_k$ in the $(\xi_1, \dots, \xi_k)$ space, where each S_i is a d-dimensional hypercube. Let w_S be the width of S and d_S be the distance between S and the origin. Then if $w_S/d_S \leqslant 2/C$, then for any $\varepsilon > 0$ there exist an integer $t = O_{C,D}(\log^{dk-1}(1/\varepsilon))$ and k sets of functions $\{\alpha^S_{1,p}(\xi_1)\}_{1 \leqslant p \leqslant t}, \dots, \{\alpha^S_{k,p}(\xi_k)\}_{1 \leqslant p \leqslant t}$ such that*

$$\left| m(\xi_1, \dots, \xi_k) - \sum_{p=1}^{t} \alpha^S_{1,p}(\xi_1) \dots \alpha^S_{k,p}(\xi_k) \right| \leqslant \varepsilon$$

for any $(\xi_1, \dots, \xi_k) \in S$.

The hierarchical decomposition step proceeds almost the same, except that now each hypercube S splits into 2^{dk} smaller hypercubes. For a given hypercube $S \in \mathcal{S}$ that satisfies $w_S/d_S \leqslant 2/C$, the algorithm for its partial sum $u^S(\xi)$ proceeds as follows.

Algorithm 2.11 *(Fast computation of the partial sum $u^S(\xi)$ associated with a hypercube S).*

```
1: for $p = 1, \dots, t$ do
2:     Compute $\alpha^{1,S}_p(\xi_1)\hat{f}_1(\xi_1)$ for $\xi_1 \in S_1$, extend by zero to a hypercube of width $kw_S$, and apply a $d$-dimensional FFT to it.
3:     Do the same for $\alpha^S_{2,p}(\xi_2)\hat{f}_2(\xi_2), \dots, \alpha^{k,S}_p(\xi_k)\hat{f}_k(\xi_k)$.
4:     Multiply these results and apply a $d$-dimensional inverse FFT of width $kw_S$ to the product.
5:     Add the result to the $\xi$ locations of the hypercube $S_1 + \dots + S_k$.
6: end for
```

A similar theorem can be proved regarding the complexity of the whole algorithm and it offers a tremendous speedup over the naive $O(N^{dk})$ algorithm.

Theorem 2.12. *For a piecewise analytic symbol $m(\xi_1, \dots, \xi_k)$, the proposed algorithm has a complexity of order $O(N^d \log N \times \log^{dk-1}(1/\varepsilon))$.*

3. Numerical results

In this section, we provide several numerical examples to illustrate the accuracy and efficiency of the proposed algorithms. All results are obtained on a desktop computer with a 2.6 GHz CPU. The low-rank approximation for each $S \in \mathbb{S}$ is generated using the randomized algorithm described Appendix A, where the parameter ε is used to control the accuracy of the approximation. The discrete functions f_1 and f_2 are generated as Gaussian random noise. For each example, we test with different values of N and ε. The running time for evaluating the multilinear operator is reported in seconds. We denote the approximate solution computed from our algorithm by $u_a(\xi)$. To measure the error with reasonable efficiency, we randomly select a subset P of Ω and estimate the error using the relative ℓ^∞ error:

$$\frac{\max_{\xi \in P}\{|u_a(\xi) - u(\xi)|\}}{\max_{\xi \in P}\{|u(\xi)|\}}.$$

In the numerical experiment, P consists of 200 points from Ω.

We first test the algorithm in Section 2.1 for $d = 1$ and $k = 2$. In the first example, $m(\xi_1, \xi_2)$ is an analytic symbol of order 0 given by

$$m(\xi_1, \xi_2) = \frac{\xi_1}{\sqrt{1 + \xi_1^2 + \xi_2^2}}. \tag{3}$$

The results for different values of N and ε are reported in Table 1. In each test, the estimated relative ℓ^∞ error is well below the prescribed accuracy ε. For each fixed ε value, the running time scales approximately linearly with respect to the size of the problem, N. For a fixed value of N, the running time only grows slightly when the threshold ε is decreased.

In the second example, we set $m(\xi_1, \xi_2)$ to be

$$m(\xi_1, \xi_2) = \frac{\xi_1 \xi_2}{1 + \xi_1^2 + \xi_2^2}, \tag{4}$$

which is again an analytic symbol of order 0. The results of this symbol are summarized in Table 2 and the asymptotic behavior of the algorithm is again compatible with the theoretical claims.

Table 1
Results from the first example with $m(\xi_1, \xi_2)$ given in (3).

ε	N	Time	Error
1.00e–03	8192	4.27e–02	7.25e–05
1.00e–03	16 384	6.23e–02	7.39e–05
1.00e–03	32 768	1.22e–01	6.25e–05
1.00e–03	65 536	2.42e–01	9.94e–05
1.00e–06	8192	5.05e–02	4.63e–08
1.00e–06	16 384	9.39e–02	5.60e–08
1.00e–06	32 768	1.84e–01	3.02e–08
1.00e–06	65 536	3.75e–01	5.57e–08
1.00e–09	8192	6.43e–02	1.50e–10
1.00e–09	16 384	1.20e–01	1.29e–10
1.00e–09	32 768	2.36e–01	1.29e–10
1.00e–09	65 536	4.85e–01	1.48e–10

Table 2
Results from the second example with $m(\xi_1, \xi_2)$ given in (4).

ε	N	Time	Error
1.00e–03	8192	3.69e–02	1.25e–04
1.00e–03	16 384	6.78e–02	9.38e–05
1.00e–03	32 768	1.31e–01	6.53e–05
1.00e–03	65 536	2.65e–01	1.07e–04
1.00e–06	8192	5.35e–02	3.70e–08
1.00e–06	16 384	9.95e–02	6.15e–08
1.00e–06	32 768	1.96e–01	8.05e–08
1.00e–06	65 536	3.97e–01	5.22e–08
1.00e–09	8192	7.05e–02	7.09e–11
1.00e–09	16 384	1.32e–01	9.00e–11
1.00e–09	32 768	2.61e–01	1.12e–10
1.00e–09	65 536	5.34e–01	7.90e–11

Table 3
Results from the third example with $m(\xi_1, \xi_2)$ given in (5).

ε	N	Time	Error
1.00e–03	8192	2.80e–01	6.54e–05
1.00e–03	16 384	5.77e–01	5.50e–05
1.00e–03	32 768	1.13e+00	8.42e–05
1.00e–03	65 536	2.23e+00	1.87e–04
1.00e–06	8192	3.21e–01	6.79e–08
1.00e–06	16 384	6.16e–01	5.61e–08
1.00e–06	32 768	1.21e+00	1.08e–07
1.00e–06	65 536	2.43e+00	6.47e–08
1.00e–09	8192	3.40e–01	1.60e–10
1.00e–09	16 384	6.53e–01	1.12e–10
1.00e–09	32 768	1.32e+00	1.53e–10
1.00e–09	65 536	2.73e+00	9.30e–11

In the third example, $m(\xi_1, \xi_2)$ is a piecewise analytic symbol of order 0 given by

$$m(\xi_1, \xi_2) = \begin{cases} \frac{\xi_1 \xi_2}{1+\xi_1^2+\xi_2^2}, & \xi_2 \geqslant \xi_1, \\ \frac{\xi_1}{\sqrt{1+\xi_1^2+\xi_2^2}} & \xi_2 < \xi_1. \end{cases} \tag{5}$$

Clearly it has a diagonal discontinuity and the algorithms in Section 2.2 are used for the computation. The results for this symbol are summarized in Table 3. The actual running times are significantly higher due to the existence of the discontinuities near the diagonal. However, the asymptotic near-linear complexity is clear from the results.

In the final example, we test the 2D bilinear case (i.e., $d = 2$). Letting $\xi_1 = (\xi_{1,1}, \xi_{1,2})$ and $\xi_2 = (\xi_{2,1}, \xi_{2,2})$, we set

$$m(\xi_1, \xi_2) = m(\xi_{1,1}, \xi_{1,2}, \xi_{2,1}, \xi_{2,2}) = \frac{\xi_{1,1}}{\sqrt{1 + \xi_{1,1}^2 + \xi_{1,2}^2 + \xi_{2,1}^2 + \xi_{2,2}^2}}. \tag{6}$$

Table 4
Results from the fourth example with $m(\xi_1, \xi_2)$ given in (6).

ε	N	Time	Error
1.00e–03	256	2.09e+00	3.08e–04
1.00e–03	512	5.61e+00	1.36e–04
1.00e–03	1024	2.22e+01	1.26e–04
1.00e–06	256	2.67e+00	3.51e–07
1.00e–06	512	8.52e+00	4.17e–07
1.00e–06	1024	3.45e+01	1.27e–06
1.00e–09	256	3.74e+00	8.03e–10
1.00e–09	512	1.20e+01	6.93e–10
1.00e–09	1024	4.97e+01	4.01e–10

Now f_1 and f_2 are defined on an $N \times N$ grid and Section 2.3 claims that the running time should be almost linear with respect to N^2. The numerical results are summarized in Table 4 and the numbers are indeed compatible with the theoretical claim.

4. Conclusions and discussions

In this paper, we propose a fast algorithm for evaluating multilinear operator with a certain class of multipliers. For the 1D bilinear case, the algorithm starts by constructing a hierarchical decomposition of the summation domain in Fourier space into squares, and then performs FFT-based convolutions to speed up the computation associated with each individual square. The complexity of the algorithm is of order $O(N \log N \log(1/\varepsilon))$ and $O(N \log^2 N \log(1/\varepsilon))$ for smooth and piecewise symbols of order 0, respectively. We also generalize the algorithm to the higher-dimensional smooth symbol case. Numerical results are provided to demonstrate the efficiency and accuracy of the algorithm.

For the more general case, $k > 3$, Theorem 2.10 proves the existence of low-rank approximation of the symbol $m(\xi_1, \ldots, \xi_k)$ restricted to a hypercube S. As we mentioned earlier, the low-rank approximation based directly on Chebyshev interpolation is often not efficient and other approximations are more computationally favorable. The randomized procedure, which gives good practical results for the bilinear case, does not have a direct generalization for $k > 3$. In fact, generating low-rank approximations for higher-dimensional tensors is a very active field of research and has attracted a lot of attention in recent years.

In our discussion, we assume that N is the parameter while d and k are fixed small constants. However, in many applications, one can no longer assume that d and k are small. For these cases, the algorithms developed here are no longer efficient (for example see the $\log^{(dk-1)}(1/\varepsilon)$ dependence in Theorem 2.12) and new insights would be required to make them practical.

Acknowledgments

This work was partially supported by an NSF CAREER grant DMS-0846501. The authors thank Jack Poulson for providing suggestions and comments.

Appendix A. A randomized approach for low-rank approximation

In this appendix, we briefly outline the randomized algorithm proposed in [4] for generating numerical low-rank approximation. For each square $S = S_1 \times S_2$ in $\mathcal{S}$, we define a matrix M as

$$M = \big(m(\xi_1, \xi_2)\big)_{\xi_1 \in S_1, \xi_2 \in S_2}.$$

The following algorithm generates an approximate low-rank factorization $M \approx M_1 M_2$ where the columns of M_1 and the rows of M_2 give the functions $\{\alpha^S_{1,p}\}$ and $\{\alpha^S_{2,p}\}$, respectively.

The ε-rank of an $m \times n$ matrix M, denoted by $r_\varepsilon(M)$ or just r_ε if M is fixed, is the number of singular values of M that are greater than or equal to ε. We call M numerically low-rank if r_ε is much smaller than the dimensions of M even for small ε. The algorithm described below aims to construct a separated approximation of the form

$$M \approx CDR$$

with accuracy $O(\varepsilon)$, where the number of columns in C and the number of rows in R are roughly r_ε. Here, we adopt the standard notation for a submatrix: given a row index set I and a column index set J, $M(I, J)$ is the submatrix with entries from rows in I and columns in J.

1. Randomly sample a set of βr_ε rows and denote the index set by $S = (s_i)$. Here β is the oversampling factor. Perform pivoted QR decomposition on the matrix $M(S, :)$ and obtain

$$M(S, P) = QR,$$

where $P = (p_i)$ is the resulting permutation vector of the columns and $R = (r_{ij})$ is upper triangular. Let k be the largest index such that $r_{kk} \geqslant \varepsilon$. Define the index set S_c to be $\{p_1, \ldots, p_k\}$.

2. Randomly sample a set of βr_ε columns and denote the index set by $S = (s_i)$. Perform a pivoted LQ decomposition on the rows of $M(:, S)$:

$$M(P, S) = LQ,$$

where P is the resulting permutation vector of the rows and $L = (\ell_{ij})$ is lower triangular. Let k be the largest index such that $\ell_{kk} \geqslant \varepsilon$. Define the index set S_r to be $\{p_1, \ldots, p_k\}$.

3. Perform a pivoted QR decomposition on the columns of $M(:, S_c)$ and a pivoted LQ decomposition on the rows of $M(S_r, :)$ respectively:

$$M(:, S_c) \cdot P_c = Q_c R_c, \qquad P_r \cdot M(S_r, :) = L_r Q_r,$$

where P_c and P_r are the resulting permutation matrices that reorder the columns of $M(:, S_c)$ and the rows of $M(S_r, :)$, respectively.

4. We seek a factorization of the form $M \approx Q_c \cdot D \cdot Q_r$. In order to do so efficiently, we restrict ourselves to the rows in S_r and columns in S_c and solve the following problem:

$$\min_D \left\| M(S_r, S_c) - Q_c(S_r, :) \cdot D \cdot Q_c(:, S_c) \right\|_F.$$

A simple least-squares solution gives $D = (Q_c(S_r, :))^+ M(S_r, S_c)(Q_r(:, S_c))^+$, where $(\cdot)^+$ stands for the pseudoinverse. Therefore, the resulting factorization is

$$M \approx Q_c \cdot \left(\left(Q_c(S_r, :)\right)^+ \cdot M(S_r, S_c) \cdot \left(Q_r(:, S_c)\right)^+ \right) \cdot Q_r.$$

Finally, we set

$$C = Q_c, \qquad D = \left(Q_c(S_r, :)\right)^+ \cdot M(S_r, S_c) \cdot \left(Q_r(:, S_c)\right)^+, \qquad R = Q_r.$$

Once we have $M \approx CDR$, setting $M_1 = CD$ and $M_2 = R$ gives the desired low-rank approximation.

In practice, setting the oversampling factor β to 5 is sufficient for an accurate approximation. Notice that the most computationally intensive steps of this algorithm are the pivoted QR decompositions of matrices of size $m \times O(r_\varepsilon)$ and the pivoted LQ decompositions of matrices of size $O(r_\varepsilon) \times n$. When ε is fixed and r_ε can be treated as a constant, the cost of this algorithm is only linear in $\max(m, n)$.

References

[1] G. Capobianco, D. Conte, I. Del Prete, E. Russo, Fast Runge–Kutta methods for nonlinear convolution systems of Volterra integral equations, BIT 47 (2) (2007) 259–275.
[2] R.R. Coifman, Y. Meyer, On commutators of singular integrals and bilinear singular integrals, Trans. Amer. Math. Soc. 212 (1975) 315–331.
[3] R.R. Coifman, Y. Meyer, Wavelets: Calderón–Zygmund and Multilinear Operators, Cambridge Stud. Adv. Math., Cambridge University Press, 1997.
[4] B. Engquist, L. Ying, A fast directional algorithm for high frequency acoustic scattering in two dimensions, Commun. Math. Sci. 7 (2) (2009) 327–345.
[5] S.A. Goreinov, E.E. Tyrtyshnikov, N.L. Zamarashkin, A theory of pseudoskeleton approximations, Linear Algebra Appl. 261 (1997) 1–21.
[6] E. Hairer, C. Lubich, M. Schlichte, Fast numerical solution of nonlinear Volterra convolution equations, SIAM J. Sci. Statist. Comput. 6 (3) (1985) 532–541.
[7] C. Lubich, A. Schädle, Fast convolution for nonreflecting boundary conditions, SIAM J. Sci. Comput. 24 (1) (2002) 161–182 (electronic).
[8] A. Schädle, M. López-Fernández, C. Lubich, Fast and oblivious convolution quadrature, SIAM J. Sci. Comput. 28 (2) (2006) 421–438 (electronic).
[9] E. Süli, D.F. Mayers, An Introduction to Numerical Analysis, Cambridge University Press, Cambridge, 2003.
[10] L. Ying, Sparse Fourier transform via butterfly algorithm, SIAM J. Sci. Comput. 31 (3) (2009) 1678–1694.
[11] L. Ying, S. Fomel, Fast computation of partial Fourier transforms, Multiscale Model. Simul. 8 (1) (2009) 110–124.