MULTISCALE MODEL. SIMUL.
Vol. 8, No. 1, pp. 110–124

FAST COMPUTATION OF PARTIAL FOURIER TRANSFORMS*

LEXING YING† AND SERGEY FOMEL‡

Abstract. This paper is concerned with the evaluation of the partial Fourier transform $u_x = \sum_{|k|<c(x)} e^{2\pi\imath xk/N} f_k$ in one and two dimensions. The motivating application is the wave extrapolation procedure in reflection seismology. As the summation in the frequency variable depends on the location x, the standard fast Fourier transform does not apply here. Direct summation requires quadratic complexity, which is extremely expensive for large values of N. The main idea of our approach is to decompose the summation domain in the $(x, |k|)$ space hierarchically into dyadic squares or cubes. The computation associated with each square or cube is accelerated by the fractional Fourier transform in one dimension and by the butterfly algorithm for the sparse Fourier transform in two dimensions. The resulting algorithm in one dimension has an $O(N \log^2 N)$ complexity for an input of size N and is numerically exact. The algorithm in two dimensions takes $O(N^2 \log^2 N)$ steps for an input of size $N \times N$ and is also highly accurate. Numerical examples are included to demonstrate the efficiency of these algorithms.

Key words. fast Fourier transform, multiscale decomposition, butterfly algorithm, fractional Fourier transform, wave extrapolation

AMS subject classifications. 65R99, 65T50

DOI. 10.1137/080715457

1. Introduction. We consider the following partial Fourier transforms in one and two dimensions. Let N be a large integer, which is assumed to be a power of 2 without loss of generality. In the one-dimensional (1D) case, define two grids $X = \{0, 1, \ldots, N-1\}$ and $K = \{-N/2, \ldots, N/2-1\}$. Let $\alpha(t)$ be a smooth function on $[0, 1]$ with $0 \le \alpha(t) \le 1/2$, and introduce the cutoff function $c(x) = N \cdot \alpha(x/N)$ for $x \in X$. Given a sequence of N numbers $\{f_k\}_{k\in K}$, the 1D partial Fourier transform computes

$$u_x = \sum_{|k|<c(x)} e^{2\pi\imath xk/N} f_k, \qquad x \in X. \tag{1}$$

Here we use $\imath$ for $\sqrt{-1}$ throughout this paper.

In the two-dimensional (2D) case, the problem is formulated in a similar way. Define two Cartesian grids $X = \{(x_1, x_2), 0 \le x_1, x_2 < N, x_1, x_2 \in \mathbb{Z}\}$ and $K = \{(k_1, k_2), -N/2 \le k_1, k_2 < N/2, k_1, k_2 \in \mathbb{Z}\}$. Now let $\alpha(t)$ be a smooth function on $[0, 1]^2$ with $0 \le \alpha(t) \le 1/2$, and introduce the cutoff function $c(x) = N \cdot \alpha(x/N)$ for $x \in X$. Given a sequence of N^2 numbers $\{f_k\}_{k\in K}$, the 2D partial Fourier transform computes

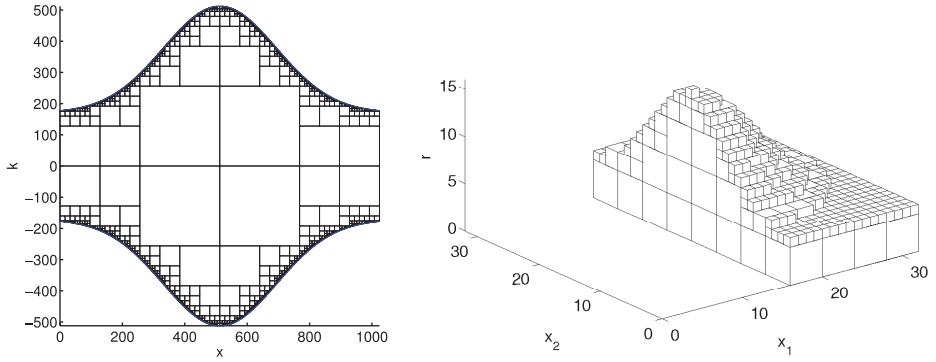
$$u_x = \sum_{|k|<c(x)} e^{2\pi\imath x\cdot k/N} f_k, \qquad x \in X. \tag{2}$$

*Received by the editors February 11, 2009; accepted for publication (in revised form) August 17, 2009; published electronically November 11, 2009. This research was partially supported by the Alfred P. Sloan Foundation and the National Science Foundation.

http://www.siam.org/journals/mms/8-1/71545.html

†Department of Mathematics and ICES, University of Texas, Austin, TX 78712 (lexing@math.utexas.edu).

‡Jackson School of Geosciences and ICES, University of Texas, Austin, TX 78712 (sergey.fomel@beg.utexas.edu).



(a) (b)

FIG. 1. (a) 1*D case. A multiscale decomposition of* $D = \{(x, k) : |k| < c(x)\}$. (b) 2*D case. A cross section view of a multiscale decomposition of* $D = \{(x_1, x_2, r) : r < c(x_1, x_2)\}$.

As the summation over $k \in K$ in (1) and (2) depends on the location x, one cannot apply the fast Fourier transform (FFT) directly. Direct evaluation of (1) and (2) has quadratic time complexity, i.e., $O(N^2)$ operations for (1) and $O(N^4)$ operations for (2). This can be extremely expensive when N is large, especially in 2D.

1.1. Outline of the approach. In this paper, we propose efficient algorithms for the partial Fourier transform with optimal complexity. The main idea of our approach is to construct a multiscale decomposition of the summation domain and apply existing fast algorithms to each piece of the decomposition.

In 1D, the summation domain is $D = \{(x, k) : |k| < c(x)\}$. The algorithm first constructs a multiscale decomposition of D by partitioning $\{(x, k) : 0 \le x < N, -N/2 \le k < N/2\}$ dyadically and recursively until the each leaf square is either inside D or outside of D (see Figure 1(a)). One then considers the computation associated with each leaf square inside D one by one. It turns out that the computation associated with each square is a fractional Fourier transform, which can be computed in almost linear time. The overall complexity of the algorithm is $O(N \log^2 N)$.

In 2D, we consider a different set $D = \{(x_1, x_2, r) : r < c(x_1, x_2)\}$. To build a multiscale decomposition of D, the algorithm partitions the domain $\{(x_1, x_2, r) : 0 \le x_1, x_2, r < N\}$ dyadically and recursively into cubes, until each leaf cube is either inside D or outside D (see Figure 1(b)). The projection of each leaf cube onto the r axis is a dyadic interval. The computation associated with each dyadic interval I of the r axis is the interaction between a circular band of points in K and a set of points in X that lie in the (x_1, x_2) projection of the cubes that projects to I in the r axis. This interaction is a special case of the sparse Fourier transform, which can be computed efficiently and accurately using the butterfly algorithm in [19] in a number of steps that are linear to the input size. A careful complexity analysis shows that the overall number of steps is $O(N^2 \log^2 N)$, again almost linear in terms of the number of inputs.

1.2. Motivating application. Our study of the partial Fourier transform is motivated by the one-way wave extrapolation procedure in reflection seismology [4]. In this procedure, one often computes the integral [13]

$$u_z(x) = \int_{\mathbb{R}^2} e^{2\pi\imath(x\cdot k + \sqrt{\omega^2/v^2(x) - k^2}\cdot z)} \widehat{u_0}(k) dk, \tag{3}$$

where ω and z are fixed constants (frequency and extrapolation depth), $v(x)$ is a given function (layer velocity), and $\widehat{u_0}(k)$ is the Fourier transform of a function $u_0(x)$. The wave modes that correspond to $|k| < \omega/v(x)$ are called *propagating*, while the ones that correspond to $|k| > w/v(x)(x)$ are called *evanescent*. For the purposes of seismic imaging, often one is only interested in the propagating mode, and, therefore, we have the following restricted integral to evaluate:

$$\int_{|k|\le\omega/v(x)} e^{2\pi\imath(x\cdot k+\sqrt{\omega^2/v^2(x)-k^2}\cdot z)}\widehat{u_0}(k)dk. \tag{4}$$

To make the computation efficient, the term with the square root is often approximated by a separated representation in x and k with a small number of terms:

$$e^{2\pi\imath\sqrt{\omega^2/v^2(x)-k^2}\cdot z} \approx \sum_n f_n(x)g_n(k), \tag{5}$$

where z is sufficiently small. Putting this approximation back into (4) results in

$$\sum_n f_n(x)\left[\int_{|k|\le\omega/v(x)} e^{2\pi\imath x\cdot k}\left(g_n(k)\widehat{u_0}(k)\right)dk\right].$$

It is clear now that the main computational task is the evaluation of

$$\int_{|k|\le w/v(x)} e^{2\pi\imath x\cdot k}\left(g_n(k)\widehat{u_0}(k)\right)dk,$$

which takes the forms of the partial Fourier transform after discretization.

1.3. Related work. The multiscale decomposition adopted in the 1D partial Fourier transform has appeared in the fast computation of temporal convolutions. In the discrete setting, given a sequence $\{f_i\}_{1\le i\le N}$, the temporal convolution computes

$$u_i = f_i + \sum_{1\le j<i} w_{i-j}u_j, \qquad i = 1, 2, \ldots, N,$$

where w_i are the convolution weight. One approach for the fast temporal convolution is given by Hairer, Lubich, and Schlichte in [11]. Their algorithm partitions the triangular summation domain $\{(i, j), 1 \le j < i \le N\}$ hierarchically into squares and applies the FFT to accelerate the convolution associated with each square. The resulting algorithm has an $O(N \log^2 N)$ complexity.

A different approach for the temporal convolution problem was introduced in [12] and extended in [6, 17], where a more complicated decomposition is adopted. Within each part of the decomposition, the convolution kernel is approximated by a sum of exponentials with the help of the Talbot contours for the Laplace transform. The convolution associated with each exponential is then computed by explicit time integration. The running time of the resulting algorithm is of order $O(N \log N)$, where the prefactor is proportional to the number of terms in the exponential approximation.

The rest of this paper is organized as follows. In section 2, we describe the algorithm for the 1D partial Fourier transform. The algorithm for the 2D partial Fourier transform is presented in section 3. Finally, we conclude with some discussions for future work in section 4.

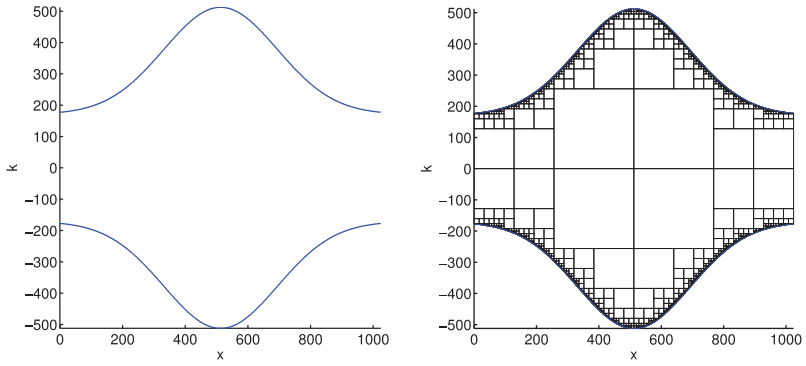


FIG. 2. (a) *The curves of* $|k| = c(x)$ *and the summation domain* D *is the region between the two curves.* (b) *The multiscale decomposition constructed of the summation domain* D*. The disjoint union of the squares is exactly equal to* D*.*

2. Partial Fourier transform in one dimension.

2.1. Algorithm description. Let us define the summation domain D to be $\{(x,k), |k| < c(x), 0 \le x < N, -N/2 \le k < N/2\}$. The idea behind our algorithm for computing

$$u_x = \sum_{|k|<c(x)} e^{2\pi\imath xk/N} f_k, \qquad 0 \le x < N,$$

is to construct a multiscale decomposition of D. Starting from the square $\{(x,k) : 0 \le x < N, -N/2 \le k < N/2\}$, we partition the domain recursively as follows:

1. If a square B is fully inside D, it is not subdivided, and we keep it in the decomposition.
2. If a square B is fully outside D, it is discarded.
3. Otherwise, B is further subdivided into four child squares with equal size. For each child, go to step 1.

At the end of this process, our decomposition contains a set of squares with dyadic sizes, and the union of these squares is exactly equal to D (see Figure 2).

Let us consider a single square B of the decomposition. Suppose that B is of size s and that its lower-left corner of B is (x^B, k^B). The summation associated with B is given by

$$\sum_{k^B \le k < k^B + s} e^{2\pi\imath xk/N} f_k \tag{6}$$

for $x^B \le x < x^B + s$. Denoting $x = x^B + x'$ and $k = k^B + k'$, we can write this into a matrix form Mf with

$$(M)_{x'k'} = e^{2\pi\imath(x^B+x')(k^B+k')/N} = e^{2\pi\imath(x^B+x')k^B/N} \cdot e^{2\pi\imath x'k'/N} \cdot e^{2\pi\imath x^B(k^B+k')/N},$$

with $0 \le x', k' < s$. Since the first and the third terms depend only on x' and k', respectively, one can factorize M as $M = M_1 \cdot M_2 \cdot M_3$, where M_1 and M_3 are diagonal matrices given by

$$(M_1)_{x'x'} = e^{2\pi\imath(x^B+x')k^B/N}, \qquad (M_3)_{k'k'} = e^{2\pi\imath x^B(k^B+k')/N},$$

and M_2 is a full matrix given by

$$(M_2)_{x'k'} = e^{2\pi\imath x'k'/N} \tag{7}$$

for $0 \le x', k' < s$. Applying M_2 to a vector of size s is the so-called fractional Fourier transform [3], which can be evaluated in $O(s \log s)$ operations using the FFT. Furthermore, as both M_1 and M_3 are diagonal matrices, (6) can be computed in $O(s \log s)$ steps as well.

In summary, the algorithm consists of the following steps:

1. Construct a decomposition for $D = \{(x,k), |k| < c(x)\}$ by partitioning the square domain $\{(x,k) : 0 \le x < N, -N/2 \le k < N/2\}$ recursively. The union of the squares in the final decomposition is equal to D.
2. For $s = 1, 2, 4, \ldots, N$, visit all of the squares of size s in the decomposition. Suppose B is one such square. Compute the summation associated with B:

$$\sum_{k^B \le k < k^B + s} e^{2\pi\imath xk/N} f_k$$

for $x^B \le x < x^B + s$ using the fractional Fourier transform, and add the result to $\{u_x, x^B \le x < x^B + s\}$.

The first step of our algorithm clearly takes at most $O(N \log N)$ steps. To estimate the complexity for the second step, one needs to obtain an upper bound on the number of squares of size s. Based on the construction of the decomposition, the center of a square B of size s must lie inside a band of width $\frac{3\sqrt{2}}{2}s$ around the curve $\{(x,k) : |k| = c(x)\}$. This is because otherwise the parent square of B would have been disjoint from the curve, and B would not have been generated. Recall now that the function $\alpha(t)$ used in the definition of $c(x)$ is smooth. Therefore, $c(x)$ itself is smooth on the scale of $O(N)$. As a result, the length of the curve $\{(x,k) : |k| = c(x)\}$ is at most of order $O(N)$. Hence, the area of the band is at most $O(Ns)$, and there are at most $O(Ns/s^2) = O(N/s)$ squares of size s inside this band. Since the algorithm spends $O(s \log s)$ operations in the fractional Fourier transform for each square of size s, the number of steps for a fixed s is $O(N/s \cdot s \log s) = O(N \log s) = O(N \log N)$. Finally, summing over all $\log N$ possible values of s, we conclude that our algorithm is $O(N \log^2 N)$. As no approximation has been made, the algorithm is exact.

2.2. Numerical results. All of the results presented here are obtained on a desktop computer with a 2.8GHz CPU. For each example, we use the following notations: T_a is the running time of our algorithm in seconds, $R_{d/a}$ is the ratio of the running time of direct evaluation to T_a, and $R_{a/f}$ is the ratio of T_a to the running time of a Fourier transform (timed using the package FFTW from [10]). As the complexity of our algorithm is $O(N \log^2 N)$, one expects $R_{d/a}$ to grow almost linearly and $R_{a/f}$ to grow like $\log N$.

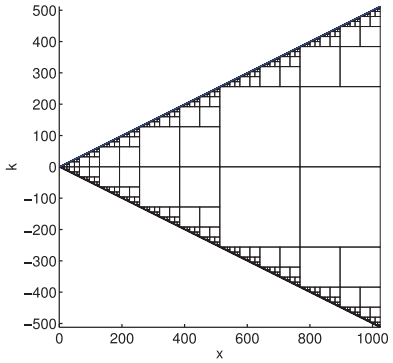
We apply the 1D partial Fourier transform algorithm to three test examples:

1. $\alpha(t) = t/2$. Then $c(x) = N \cdot \alpha(x/N) = x/2$.
2. $\alpha(t) = \sin(\pi t)/2$. Then $c(x) = N/2 \cdot \sin(\pi x/N)$.
3. $c(x)$ corresponds to a 100Hz wave propagation through a slice of the Marmousi velocity model [18] taken at 2 km depth.

Tables 1, 2 and 3 summarize the results for these three tests. From these examples, we observe clearly that $R_{d/a}$, the ratio between the running times of direct evaluation and our algorithm, indeed grows almost linearly in terms of N. Although the ratio $R_{a/f}$ has some fluctuations, its value grows slowly with respect to N.

TABLE 1

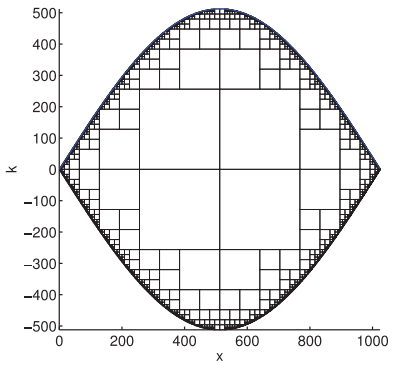
Results of test 1. *Top: the decomposition of* D. *Bottom: the results for different values of* N. T_a*: the running time of our algorithm.* $R_{d/a}$*: the ratio between direct evaluation and our algorithm.* $R_{a/f}$*: the ratio between our algorithm and one execution of FFT of size* N.



N	T_a(sec.)	$R_{d/a}$	$R_{a/f}$
1024	6.25e-03	2.40e+01	1.02e+02
4096	3.25e-02	7.88e+01	1.56e+01
16384	1.80e-01	2.28e+02	1.08e+01
65536	7.30e-01	8.98e+02	6.19e+01
262144	3.10e+00	2.54e+03	5.98e+01
1048576	1.42e+01	8.13e+03	1.15e+02

TABLE 2

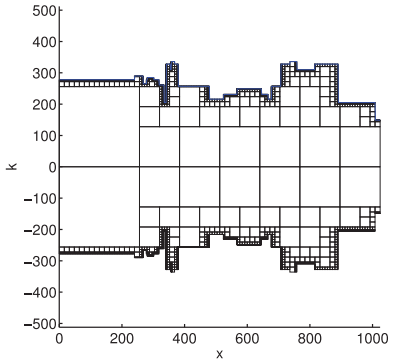
Results of test 2. *Top: the decomposition of* D. *Bottom: the results for different values of* N.



N	T_a(sec.)	$R_{d/a}$	$R_{a/f}$
1024	1.00e-02	1.50e+01	1.67e+02
4096	5.00e-02	5.12e+01	2.40e+01
16384	2.60e-01	1.58e+02	1.64e+02
65536	1.18e+00	5.55e+02	1.82e+02
262144	5.30e+00	1.61e+03	1.02e+02
1048576	2.45e+01	4.70e+03	1.80e+02

3. Partial Fourier transform in two dimensions. A direct extension of the 1D algorithm to the 2D case would partition the four-dimensional summation domain $D = \{(x, k), |k| < c(x)\}$ with a similar four-dimensional tree structure. However, this does not provide us with an algorithm with optimal complexity. To see this, let us count the number of cubes of size s in our tree structure. Repeating the argument used in the complexity analysis of the 1D algorithm shows that there are

TABLE 3
Results of test 3. Top: the decomposition of D. Bottom: the results for different values of N.



N	T_a(sec.)	$R_{d/a}$	$R_{a/f}$
1024	8.13e-03	1.72e+01	1.33e+02
4096	4.50e-02	5.69e+01	2.15e+01
16384	2.30e-01	1.78e+02	1.38e+01
65536	9.00e-01	6.83e+02	1.40e+02
262144	4.04e+00	2.11e+03	1.44e+02
1048576	1.70e+01	6.17e+03	1.36e+02

about $N^3 s/s^4 = N^3/s^3$ cubes of size s. Even though the computation associated with each cube can be done in about $s^2 \log s$ steps, the total operation count for a fixed s is about $N^3/s^3 \cdot s^2 \log s = N^3/s \log s$, which is much larger than the degree of freedom N^2 for small values of s.

3.1. Algorithm description. Since only $|k|$ appears in the constraint of the 2D partial Fourier transform

$$u_x = \sum_{|k|<c(x)} e^{2\pi \imath x \cdot k/N} f_k$$

for $x = (x_1, x_2)$ and $k = (k_1, k_2)$, we propose looking at a slightly different set $D = \{(x_1, x_2, r), 0 \le r < c(x_1, x_2)\}$ instead.

The algorithm first generates a multiscale decomposition for D. Starting from the cube $\{(x_1, x_2, r) : 0 \le x_1, x_2, r < N\}$, we partition the domain recursively as follows:

1. If a cube B is fully inside D, it is not subdivided, and we keep it in the decomposition.
2. If a cube B is fully outside D, it is discarded.
3. Otherwise, the cube B is further subdivided into eight child cubes with equal size. For each child cube, go to step 1.

At the end of this process, our decomposition contains a group of cubes with dyadic sizes, and the union of these cubes is exactly equal to D (see Figure 3).

The projection of any cube of the generated decomposition onto the r coordinate is a dyadic interval. Let us consider one such interval I of size s and denote by G^I the set of all cubes of size s that project onto I (see Figure 4(a)). Define K^I to be the set $\{k, |k| \in I\}$ (see Figure 4(b)) and X^I to be the image of the points in G^I under the projection onto the (x_1, x_2) plane (see Figure 4(c)). As I is an interval of size s, K^I is a circular band of points in the (k_1, k_2) plane with width equal of order $O(s)$. Since the function $\alpha(t)$ used to define $\{c(x), 0 \le x_1, x_2 < N\}$ is smooth, the function

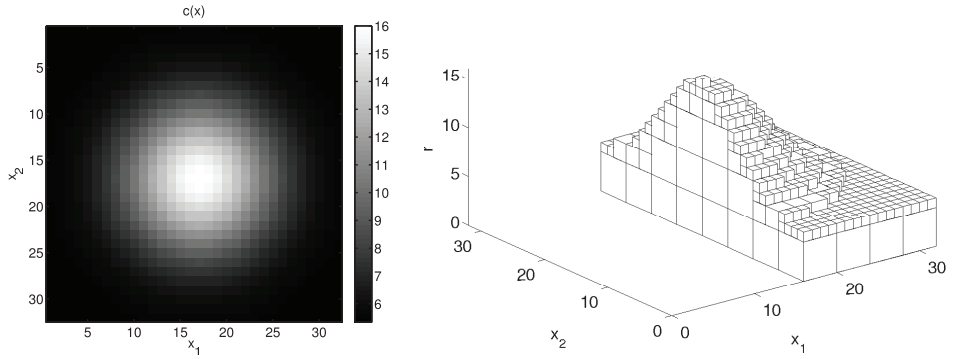


FIG. 3. *Left: $c(x_1, x_2)$ is a Gaussian function. Right: a cross section view of the multiscale decomposition of D (the domain below the surface $c(x_1, x_2)$). Here $N = 32$.*

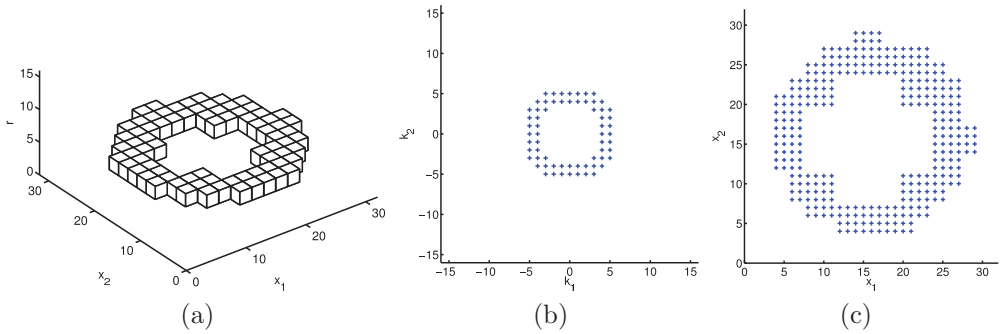


FIG. 4. (a) *The cubes inside the set G^I for an given interval I of the r coordinate.* (b) *The set K^I in the (k_1, k_2) plane.* (c) *The set X^I in the (x_1, x_2) plane. Here $N = 32$.*

$c(x)$ itself is smooth on a scale of $O(N)$. Hence, the set X_I is a set of points in the (x_1, x_2) plane with its boundary of length $O(N)$.

The summation associated with the interval I is given by

$$\sum_{k \in K^I} e^{2\pi \imath x \cdot k/N} f_k \tag{8}$$

for $x \in X^I$. Since X^I and K^I are two sets of points in $[0, N)^2$ and $[-N/2, N/2)^2$, respectively, the computation of (8) is a special case of the sparse Fourier transform considered in [19], where both the frequency and spatial data are sparse. The algorithm proposed in [19] is an extension of the butterfly algorithm [14, 15, 16] and computes the sparse Fourier transform in almost linear time. The description of this algorithm is postponed to the next section. Here we emphasize only that this algorithm computes an accurate approximation of (8) in $O((|X^I| + |K^I| + N) \cdot \log N)$ operations.

Combining these ideas together, we obtain the following algorithm for the 2D partial Fourier transform:

1. Construct a decomposition for $D = \{(x_1, x_2, r), r < c(x_1, x_2)\}$ by partitioning the domain $\{(x, k) : 0 \le x < N, -N/2 \le k < N/2\}$ recursively. The union of the cubes in the final decomposition is equal to D.

2. For $s = 1, 2, 4, \ldots, N$, visit all the dyadic intervals of size s in the r coordinate. Suppose that I is one such interval. Compute the summation associated with I:

$$\sum_{k \in K^I} e^{2\pi \imath x \cdot k/N} f_k$$

for $x \in X^I$ using the sparse Fourier transform algorithm proposed in [19], and add the result to $\{u_x\}_{x \in X^I}$.

Let us consider now the complexity of this algorithm. The first step of the algorithm takes $O(N^2 \log N)$ steps. In order to estimate the number of operations used in the second step, let us consider a fixed s. For each interval I of size s, the number of steps used is of order $O((|X^I| + |K^I| + N) \cdot \log N)$. Summing over all cubes of size s, we get

$$O\left(\sum_{I:|I|=s} |X^I| \log N\right) + O\left(\sum_{I:|I|=s} |K^I| \log N\right) + O\left(\sum_{I:|I|=s} N \log N\right).$$

As $\sum_{I:|I|=s} |X^I| = \sum_{I:|I|=s} |K^I| = N^2$, the above quantity is clearly bounded by $O(N^2 \log N)$. Finally, after summing over different values of s, we have a total complexity of order $O(N^2 \log^2 N)$.

3.2. Sparse Fourier transform. Here, we provide a brief description of the sparse Fourier transform, adapted to the current setting. Fix an dyadic interval I. To simplify the notation, we drop the superscript I in the following discussion. From the above discussion, K is a band of points in the frequency domain $[-N/2, N/2)^2$, X is a set of points in the spatial domain $[0, N)^2$, and the boundaries of both sets are of size $O(N)$, due to the smoothness of $c(x_1, x_2)$ on the $O(N)$ scale. The aim is to compute

$$u_x = \sum_{k \in K} e^{2\pi \imath x \cdot k/N} f_k, \qquad x \in X$$

efficiently.

The algorithm presented in [19] is a special case of the butterfly algorithm [14, 15, 16] for general oscillatory integral interactions. The algorithm first constructs adaptive quadtrees T_X and T_K for the sets X and K, respectively. The quadtree T_X takes $[0, N]^2$ as the top level square. Each square is partitioned recursively into four identical child squares until all leaf squares are of unit size, and only the squares that contain points in X are kept. The quadtree T_K is constructed in the same way with $[-N/2, N/2]^2$ as the top level square and K as the point set.

The main idea of the algorithm is based on the following geometric observation. Let A and B be two squares in T_X and T_K, respectively. If the product of their widths $w^A w^B$ is bounded by N, then the interaction $e^{2\pi \imath x \cdot k}/N$ for $x \in A$ and $k \in K$ is numerically low rank. More precisely, for any fixed ε, there exists a number $T_\varepsilon = O(\log(1/\varepsilon))$ and two sets of functions $\{\alpha_t^{AB}(x)\}_{1 \le t \le T_\varepsilon}$ and $\{\beta_t^{AB}(x)\}_{1 \le t \le T_\varepsilon}$ such that

$$\left| e^{2\pi \imath x \cdot k/N} - \sum_{t=1}^{T_\varepsilon} \alpha_t^{AB}(x) \beta_t^{AB}(k) \right| \le \varepsilon.$$

In fact, the function $\alpha_t^{AB}(x)$ can be chosen to be of form $e^{2\pi \imath x \cdot k_t^B/N}$, where $\{k_t^B\}_{1 \le t \le T_\varepsilon}$ belong to a Chebyshev grid of the square B. For a fixed accuracy ε, the size of this Chebyshev grid T_ε is independent of N (see [19] for details).

Let us define the *partial sum* $u^B(x)$ by

$$u^B(x) = \sum_{k \in B \cap K} e^{2\pi \imath x \cdot k/N} f_k \tag{9}$$

with the sum restricted to k inside B. The geometric observation implies that, for A and B with $w^A w^B \le N$, the restriction of $u^B(x)$ to $x \in A$ can be approximated by placing a set of *equivalent sources* $\{f_t^{AB}\}_{1 \le t \le T_\varepsilon}$ at location $\{k_t^B\}_{1 \le t \le T_\varepsilon}$. The computation of the equivalent sources $\{f_t^{AB}\}$ is done by equating the partial sum $u^B(x)$ and the potential generated by $\{f_t^{AB}\}$ at a Chebyshev grid $\{x_s^A\}$ inside A. More precisely, one solves for $\{f_t^{AB}\}$ from the following equation:

$$u_s^{AB} := u^B(x_s^A) = \sum_{t=1}^{T_\varepsilon} e^{2\pi \imath x_s^A \cdot k_t^B/N} f_t^{AB}, \qquad 1 \le s, t \le T_\varepsilon.$$

Solving this system requires inverting the matrix $(e^{2\pi \imath x_s^A \cdot k_t^B/N})_{st}$. However, due to the translation-invariant property of the Fourier kernel, the matrices to be inverted for different combinations of A and B are almost identical. Furthermore, the solution of $\{f_t^{AB}\}$ can be accelerated using the tensor-product structure of the 2D Fourier kernel.

Evaluating $u_s^{AB} := u^B(x_s^A)$ directly using (9) is computationally expensive when the square B is large. The next ingredient of the butterfly algorithm addresses how to do this efficiently. Let P be the parent of A and $B_c, c = 1, \ldots, 4$, be the children of B. From the definition (9),

$$u^B(x) = \sum_{c=1}^{4} u^{B_c}(x).$$

Now, suppose that the equivalent sources $\{f_t^{PB_c}\}$ are available already; then the quantities $\{u_s^{AB}\}$ can be approximated by

$$u_s^{AB} := u^B(x_s^A) = \sum_{c=1}^{4} u^{B_c}(x_s^A) \approx \sum_{c=1}^{4} \left(\sum_{t=1}^{T_\varepsilon} e^{2\pi \imath x_s^A k_t^{B_c}/N} f_t^{PB_c} \right), \qquad 1 \le s \le T_\varepsilon,$$

based on precisely the definition of the equivalent sources. This offers a much more efficient way for computing $\{u_s^{AB}\}$, as the size of the Chebyshev grid is a constant.

After putting these components together, the sparse Fourier transform algorithm in [19] is in fact a systematic way to construct $\{f_t^{AB}\}_{1 \le t \le T_\varepsilon}$ for all pairs of squares $A \in T_X$ and $B \in T_K$ with $w^A w^B = N$. It consists of the following steps.

1. Construct the quadtrees T_X and T_K for the point sets X and K, respectively. These trees are constructed adaptively, and all the leaf squares are of unit size.
2. Let A be the root square of T_X. For each leaf square B of T_K, compute

$$u_s^{AB} = \sum_{k \in B \cap K} e^{2\pi \imath x_s^A \cdot k/N} f_k, \qquad 1 \le s \le T_\varepsilon,$$

and solve for $\{f_t^{AB}\}_{1 \le t \le T_\varepsilon}$ from

$$u_s^{AB} = \sum_{t=1}^{T_\varepsilon} e^{2\pi \imath x_s^A \cdot k_t^B/N} f_t^{AB}, \qquad 1 \le s \le T_\varepsilon.$$

3. For each $\ell = 1, 2, \ldots, \log N$, construct the equivalent sources $\{f_t^{AB}\}$ for each pair (A, B) with A at level ℓ of T_X and B at level $(\log N - \ell)$ of T_K. Let P be the parent of A and B_c, $c = 1, \ldots, 4$, be the children of B. Compute u_s^{AB} using

$$u_s^{AB} = \sum_{c=1}^{4} \left(\sum_{t=1}^{T_\varepsilon} e^{2\pi \imath x_s^A k_t^{B_c}/N} f_t^{PB_c} \right), \qquad 1 \le s \le T_\varepsilon.$$

Next, solve for $\{f_t^{AB}\}_{1 \le t \le T_\varepsilon}$ from

$$u_s^{AB} = \sum_{t=1}^{T_\varepsilon} e^{2\pi \imath x_s^A \cdot k_t^B/N} f_t^{AB}, \qquad 1 \le s \le T_\varepsilon.$$

4. Finally, let B be the root square of T_K. For each leaf square A of T_X and for each $x \in A \cap X$, set

$$u_x = \sum_{t=1}^{T_\varepsilon} e^{2\pi \imath x \cdot k_t^B/N} f_t^{AB}.$$

Let us denote by N_X and N_K the numbers of points in X and K, respectively. The first step of the algorithm clearly takes at most $O(N_X \log N) + O(N_K \log N)$ steps. The second and fourth steps spend a constant number of steps per point in X and K, thus resulting in an extra $O(N_X) + O(N_K)$ steps. The dominant computation is in the third step. Let us use b to denote the side length of a generic square in T_X and T_K. Recall that in our setting K is a band of points with a boundary of size $O(N)$. As a result, each $b \times b$ square that overlaps with K must be less than $2b$ away from K. Therefore, each such square is contained in an area at most of size $O(Nb + N_K)$. Since there squares in T_K are nonoverlapping, there are at most $O((Nb + N_K)/b^2)$ squares that overlap with K. Similarly, for a fixed size b, there are at most $O((Nb + N_X)/b^2)$ squares that overlap with X.

At the ℓth step of the algorithm, the squares in T_K are of size 2^ℓ, and the squares in T_X are of size $N/2^\ell$. Therefore, the number of pairs (A, B) visited in the ℓth step is at most equal to

$$O\left(\frac{N2^\ell + N_K}{2^{2\ell}} \cdot \frac{N \cdot N/2^\ell + N_K}{N^2/2^{2\ell}} \right) = O(N_X + N_K + N).$$

This estimate is also the number of operations used in the ℓth step as one spends a constant number of operations for each pair. Finally, summing over all $\log N$ levels gives the $O((N_X + N_K + N) \log N)$ estimate of the sparse Fourier transform algorithm.

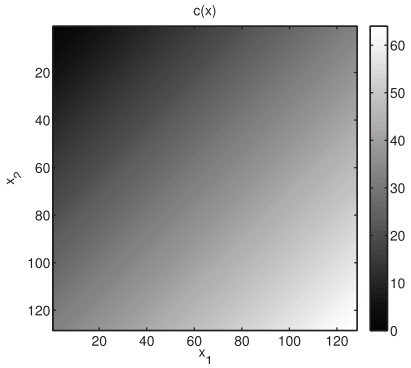
3.3. Numerical results. In the sparse Fourier transform algorithm, the size of the Chebyshev grid for the equivalent sources controls the accuracy. This accuracy further determines the accuracy of the 2D partial Fourier transform algorithm. In the following tests, we set the grid size to be 5×5 for low accuracy calculation and 9×9 for high accuracy calculation. To quantify the error, we select a set $S \subset \{0, 1, \ldots, N-1\}^2$ of size 100 and estimate the error by

$$\sqrt{\frac{\sum_{x \in S} |u_x - u_x^a|^2}{\sum_{x \in S} |u_x|^2}}, \tag{10}$$

where $\{u_x\}$ are the exact results and $\{u_x^a\}$ are our approximations.

TABLE 4

Results of test 1. *Top:* $c(x_1, x_2)$ *when* $N = 128$. *Bottom: the results for different combinations of* (N, p), *where* $p \times p$ *is the size of the Chebyshev grid used in the sparse Fourier transform computation.* T_a*: the running time of our algorithm.* $R_{d/a}$*: the ratio between direct evaluation and our algorithm.* $R_{a/f}$*: the ratio between our algorithm and one execution of FFT of size* N. E_a*: the estimated error.*



(N, p)	T_a(sec.)	$R_{d/a}$	$R_{a/f}$	E_a
(128,5)	1.05e+00	3.90e+01	1.12e+03	6.38e-04
(256,5)	7.36e+00	9.35e+01	1.68e+03	7.66e-04
(512,5)	3.99e+01	3.12e+02	1.11e+03	7.68e-04
(1024,5)	2.03e+02	1.11e+03	1.47e+03	1.07e-03
(2048,5)	1.03e+03	4.42e+03	1.05e+03	1.33e-03
(128,9)	7.70e-01	5.32e+01	8.21e+02	8.33e-15
(256,9)	7.55e+00	9.11e+01	1.73e+03	6.25e-09
(512,9)	5.27e+01	2.24e+02	2.14e+03	1.69e-08
(1024,9)	3.10e+02	9.38e+02	1.68e+03	1.47e-08
(2048,9)	1.59e+03	3.18e+03	1.96e+03	2.38e-08

Similar to the 1D case, the following notations are used: T_a is the running time of our algorithm in seconds, $R_{d/a}$ is the ratio of the running time of direct evaluation to T_a, $R_{a/f}$ is the ratio of T_a over the running time of a Fourier transform (timed using FFTW [10]), and finally E_a is the estimated error in (10).

We apply the 2D partial Fourier transform algorithm to the following three tests:

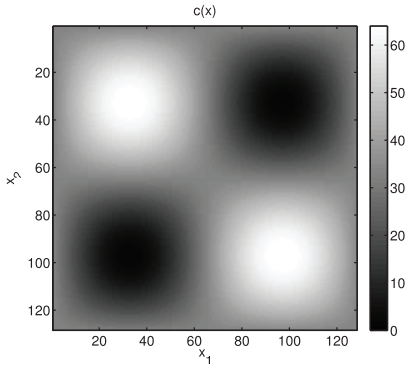
1. $\alpha(t_1, t_2) = (t_1 + t_2)/4$. Then $c(x_1, x_2) = (x_1 + x_2)/4$.
2. $\alpha(t_1, t_2) = \frac{1+\sin(2\pi t_1)\sin(2\pi t_2)}{4}$. Then $c(x_1, x_2) = \frac{N}{4} \cdot (1 + \sin\frac{2\pi x_1}{N} \sin\frac{2\pi x_2}{N})$.
3. $c(x_1, x_2)$ corresponds to a 50 Hertz wave propagation through a slice of the SEG/EAGE velocity model [1] taken at 1.5 km depth.

The numerical results of these three tests are summarized in Tables 4, 5, and 6. From these results, we see that the estimated relative error of the 2D partial Fourier transform is consistent with the error of the sparse Fourier transform presented in [19]. In terms of the running time, our implementation indeed has a complexity almost linear in terms of the number of grid points. However, due to the relatively complex structure of the sparse Fourier transform, the prefactor of our algorithm is quite large compared to the one of FFTW.

4. Conclusions and discussions. In this paper, we introduced two efficient algorithms for computing the partial Fourier transforms in one and two dimensions. In both cases, we decompose the appropriate summation domain in a multiscale way into simple pieces and apply existing fast algorithms on each piece to get optimal complexity. In one dimension, the fractional Fourier transform is used. In two dimen-

TABLE 5

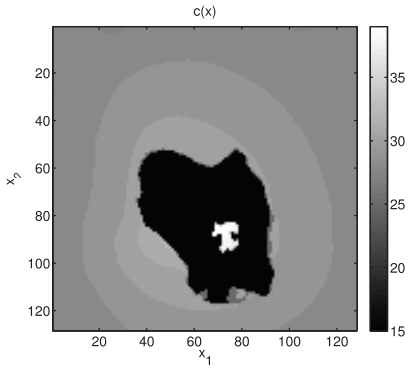
Results of test 2. *Top:* $c(x_1, x_2)$ *when* $N = 128$. *Bottom: the results for different combinations of* (N, p).



(N,p)	T_a(sec.)	$R_{d/a}$	$R_{a/f}$	E_a
(128,5)	1.64e+00	2.50e+01	1.61e+03	3.89e-04
(256,5)	1.36e+01	5.29e+01	3.11e+03	5.85e-04
(512,5)	7.99e+01	1.49e+02	2.61e+03	1.01e-03
(1024,5)	4.20e+02	7.32e+02	2.15e+03	1.53e-03
(2048,5)	2.11e+03	2.27e+03	2.65e+03	6.77e-04
(128,9)	1.10e+00	3.72e+01	1.08e+03	9.09e-15
(256,9)	1.57e+01	4.39e+01	3.58e+03	8.35e-09
(512,9)	1.13e+02	1.15e+02	3.19e+03	9.14e-09
(1024,9)	6.60e+02	4.66e+02	3.50e+03	1.96e-08
(2048,9)	3.25e+03	1.52e+03	4.11e+03	3.53e-08

TABLE 6

Results of test 3. *Top:* $c(x_1, x_2)$ *when* $N = 128$. *Bottom: the results for different combinations of* (N, p).



(N,p)	T_a(sec.)	$R_{d/a}$	$R_{a/f}$	E_a
(128,5)	7.00e-01	5.85e+01	7.47e+02	6.46e-04
(256,5)	4.24e+00	1.55e+02	1.13e+03	5.98e-04
(512,5)	2.41e+01	5.21e+02	8.99e+02	6.46e-04
(1024,5)	1.23e+02	2.50e+03	6.46e+02	1.03e-03
(2048,5)	6.13e+02	7.79e+03	5.40e+02	1.18e-03
(128,9)	5.60e-01	7.31e+01	5.51e+02	3.65e-15
(256,9)	4.71e+00	1.46e+02	1.21e+03	8.48e-09
(512,9)	3.39e+01	3.60e+02	1.03e+03	2.04e-08
(1024,9)	1.91e+02	1.60e+03	1.03e+03	1.77e-08
(2048,9)	1.00e+03	4.76e+03	8.70e+02	2.47e-08

sions, we resort to the butterfly algorithm in [19]. Asymptotically, both algorithms are only $O(\log N)$ times more expensive than the FFT.

In Tables 4, 5, and 6, we notice that our 2D algorithm has a relatively large constant. One direction of future research is to improve on our current implementation of the butterfly algorithm. Another direction is to seek different ways for computing the Fourier transforms with sparse data. In the past several years, several algorithms have been developed to address similar oscillatory behavior efficiently (see, for example, [2, 5, 7, 8, 9]). It would be interesting to see whether these ideas can be used in the setting of the Fourier transform with sparse data.

We have restricted our discussion to the case of N being a power of 2. For general values of N, the algorithm is almost the same. The only difference is that now the size of the squares or cubes are not necessarily integers. However, this difference does not affect the result of the computation, as neither the fractional Fourier transform used in one dimension nor the sparse Fourier transform used in two dimensions assumes an integer size.

So far, we have assumed that the spatial grid X and the frequency grid K are to be uniformly distributed Cartesian grids. The algorithms presented in this paper can also be extended to quasi-uniform grids. In the 1D case, since the points in X and K are no longer uniformly distributed, the fractional Fourier transform cannot be applied directly. Instead, we will need a butterfly algorithm to compute the summation associated with each box in the decomposition. In the 2D case, the algorithm remains unchanged as the sparse Fourier transform does not make any assumption on the uniformity of the grids X and K (see [19] for details).

As we mentioned earlier, this research is motivated mostly by the wave extrapolation procedure in seismic imaging. Our model problem considers only one of the challenges, i.e., the existence of the summation constraint. The other challenge is to improve the evaluation of the $e^{2\pi\imath\sqrt{\omega^2/v^2(x)-k^2}\cdot z}$ term, for example, by constructing different approximations within each component of the hierarchical decomposition.

Acknowledgments. The authors thank the reviewers for their comments and suggestions.

REFERENCES

[1] F. Aminzadeh, J. Brac, and T. Kunz, 3-*D Salt and Overthrust Models*, SEG/EAGE 3-D Modeling Series, Vol. 1, Society of Exploration Geophysicists, Tulsa, OK, 1997.

[2] A. Averbuch, E. Braverman, R. Coifman, M. Israeli, and A. Sidi, *Efficient computation of oscillatory integrals via adaptive multiscale local Fourier bases*, Appl. Comput. Harmon. Anal., 9 (2000), pp. 19–53.

[3] D. H. Bailey and P. N. Swarztrauber, *The fractional Fourier transform and applications*, SIAM Rev., 33 (1991), pp. 389–404.

[4] B. L. Biondi, 3*D Seismic Imaging*, Society of Exploration Geophysicists, Tulsa, OK, 2006.

[5] E. Candès, L. Demanet, and L. Ying, *A fast butterfly algorithm for the computation of Fourier integral operators*, Multiscale Model. Simul., 7 (2009), pp. 1727–1750.

[6] G. Capobianco, D. Conte, I. Del Prete, and E. Russo, *Fast Runge-Kutta methods for nonlinear convolution systems of Volterra integral equations*, BIT, 47 (2007), pp. 259–275.

[7] H. Cheng, W. Y. Crutchfield, Z. Gimbutas, L. F. Greengard, J. F. Ethridge, J. Huang, V. Rokhlin, N. Yarvin, and J. Zhao, *A wideband fast multipole method for the Helmholtz equation in three dimensions*, J. Comput. Phys., 216 (2006), pp. 300–325.

[8] B. Engquist and L. Ying, *Fast directional multilevel algorithms for oscillatory kernels*, SIAM J. Sci. Comput., 29 (2007), pp. 1710–1737.

[9] B. Engquist and L. Ying, *A fast directional algorithm for high frequency acoustic scattering in two dimensions*, Commun. Math. Sci., 7 (2009), pp. 327–345.

[10] M. Frigo and S. G. Johnson, *The design and implementation of FFTW*3, Proc. IEEE, 93 (2005), pp. 216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation," http://www.fftw.org.
[11] E. Hairer, C. Lubich, and M. Schlichte, *Fast numerical solution of nonlinear Volterra convolution equations*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 532–541.
[12] C. Lubich and A. Schädle, *Fast convolution for nonreflecting boundary conditions*, SIAM J. Sci. Comput., 24 (2002), pp. 161–182.
[13] G. F. Margrave and R. J. Ferguson, *Wavefield extrapolation by nonstationary phase shift*, Geophysics, 64 (1999), pp. 1067–1078.
[14] E. Michielssen and A. Boag, *A multilevel matrix decomposition algorithm for analyzing scattering from large structures*, IEEE Trans. Antennas and Propagation, 44 (1996), pp. 1086–1093.
[15] M. O'Neil and V. Rokhlin, *A New Class of Analysis-Based Fast Transforms*, Technical report, Yale University, New Haven, CT, 2007.
[16] M. O'Neil, F. Woolfe, and V. Rokhlin, *An algorithm for the rapid evaluation of special function transforms*, Technical report, Yale University, New Haven, CT, 2008.
[17] A. Schädle, M. López-Fernández, and C. Lubich, *Fast and oblivious convolution quadrature*, SIAM J. Sci. Comput., 28 (2006), pp. 421–438.
[18] R. Versteeg, *The Marmousi experience: Velocity model determination on a synthetic complex data set*, The Leading Edge, 13 (1994), pp. 927–936.
[19] L. Ying, *Sparse Fourier transform via butterfly algorithm*, SIAM J. Sci. Comput., 31 (2009), pp. 1678–1694.