

Sparse Fourier Transform via Butterfly Algorithm

Lexing Ying

Department of Mathematics, University of Texas, Austin, TX 78712

December 2007, revised June 2008

Abstract

This paper introduces a fast algorithm for computing sparse Fourier transforms with spatial and Fourier data supported on curves or surfaces. This problem appears naturally in several important applications of wave scattering, digital signal processing, and reflection seismology. The main idea of the algorithm is that the interaction between a frequency region and a spatial region is approximately low rank if the product of their widths are bounded by the maximum frequency. Based on this property, we can approximate the interaction between these two boxes accurately and compactly using a small number of equivalent sources. The overall structure of the algorithm follows the butterfly algorithm. The computation is further accelerated by exploiting the tensor-product property of the Fourier kernel in two and three dimensions. The proposed algorithm is accurate and has the optimal complexity. We present numerical results in both two and three dimensions.

Keywords. Fourier transform; Butterfly algorithm; Multiscale methods; Far field pattern.

AMS subject classifications. 65R10, 65T50, 65Y20.

1 Introduction

We consider the rapid computation of the following sparse Fourier transform problem. Let N be a large integer. The spatial domain X and the Fourier domain Ω are both equal to $[0, N]^d$ where $d = 2, 3$. M_X and M_Ω are two smooth or piecewise smooth manifolds of codimension 1 in X and Ω , respectively, and we assume that both of them are smooth on the scale of N . Suppose that the point sets $\{x_i\}$ and $\{\xi_j\}$ are, respectively, samples of M_X and M_Ω (see Figure 1). Given a set of sources $\{f_j\}$ located at $\{\xi_j\}$, the sparse Fourier transform that we consider is the computation of the potentials $\{u_i\}$ defined by

$$u_i = \sum_j e^{2\pi i x_i \cdot \xi_j / N} f_j. \quad (1)$$

In most of the applications that we are concerned with, $\{x_i\}$ and $\{\xi_j\}$ sample the manifolds M_X and M_Ω with a constant number of points per unit length. As a result, the two sets $\{x_i\}$ and $\{\xi_j\}$ are both of size $O(N^{d-1})$. Direct evaluation of (1) for all $\{u_i\}$ clearly requires $O(N^{2d-2})$ steps, which can be quite expensive for large values of N , especially in the three dimensional case.

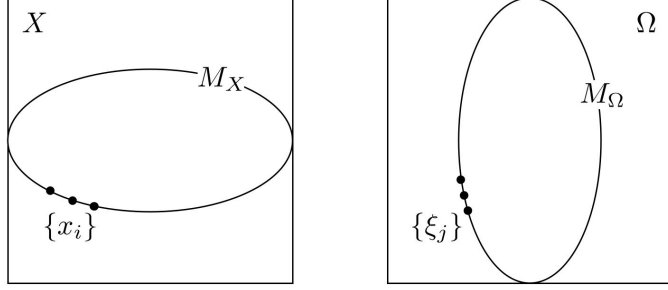


Figure 1: The formulation of the sparse Fourier transform. $X = [0, N]^d$ and $\Omega = [0, N]^d$ are the spatial and Fourier domains. M_X and M_Ω are two manifolds of codimension 1 in X and Ω . The point sets $\{x_i\}$ and $\{\xi_j\}$ sample M_X and M_Ω with a constant number of points per unit length.

1.1 Outline of the approach

In this paper, we propose an algorithm that computes the values of all $\{u_i\}$ accurately in only $O(N^{d-1} \log N)$ steps. This complexity bound is optimal up to a logarithmic factor from the information-theoretical viewpoint. The discussion of our approach focuses on the two dimensional case ($d = 2$); the algorithm can be extended to the three dimensional case ($d = 3$) without any significant modification.

The main idea of our algorithm is the following geometric low rank property of the Fourier kernel $e^{2\pi i x \cdot \xi / N}$. Let $A \subset X$ and $B \subset \Omega$ be two square boxes in the spatial and Fourier domains, respectively. If the widths w^A and w^B of these two boxes satisfy the condition $w^A w^B \leq N$, then the interaction $e^{2\pi i x \cdot \xi / N}$ between $x \in A$ and $\xi \in B$ is approximately of low rank and the approximation rank is independent of the value of N . If we define

$$u^B(x) := \sum_{\xi_j \in B} e^{2\pi i x \cdot \xi_j / N} f_j \quad (2)$$

to be the potential function generated by the sources $\{f_j\}$ in B , this low rank property guarantees that one can approximate the potential $u^B(x)$ in the box A with a set of equivalent sources in B and the number of equivalent sources required is bounded by a constant independent of N .

Based on this observation, our approach follows the general structure of the butterfly algorithm proposed in [22, 23]. The data structure of the algorithm consists of two quadtrees T_X and T_Ω of the points sets $\{x_i\}$ and $\{\xi_j\}$. T_X takes the spatial domain X as its root node and partitions the set $\{x_i\}$ recursively. Similarly, the quadtree T_Ω takes the Fourier domain Ω as its root node and partitions $\{\xi_j\}$. Both of these quadtrees are constructed adaptively and all the leaf boxes are at level $L = \log_2 N$ and of unit size. In the first step, the algorithm constructs the equivalent sources for the interaction between the root box of T_X and the leaf boxes of T_Ω . Next, we traverse down in T_X and up in T_Ω simultaneously. For each level l of T_X , we construct the equivalent sources for the interaction between all pairs of boxes (A, B) with A at level l of T_X and B at level $L - l$ of T_Ω . Each of these pairs satisfies the constraint $w^A w^B = N$. Finally, we visit all of the leaf boxes of T_X and use the equivalent sources of the interaction between the leaf boxes of T_X and the root box of T_Ω to compute the potentials $\{u_i\}$ at points $\{x_i\}$.

1.2 Applications

Sparse Fourier transforms of form (1) appear naturally in several contexts. An important example is the calculation of the far field pattern of scattered fields [12]. Suppose that D is a smooth scatterer in \mathbb{R}^d illuminated by a time harmonic incoming wave field. The scattered field $u(x)$ satisfies the Helmholtz equation

$$\Delta u(x) + k^2 u(x) = 0, \quad x \in \mathbb{R}^d \setminus D \quad (3)$$

with appropriate boundary conditions. Here k is the wave number and large values of k correspond to high frequency waves. The far field pattern $u_\infty(\hat{x})$ of the scattered field $u(x)$ is defined by

$$u_\infty(\hat{x}) = \lim_{r \rightarrow \infty} r^{(d-1)/2} e^{-ik|r|} u(r\hat{x}), \quad \hat{x} \in \mathbb{S}^{d-1}.$$

and is of great importance to many applications. One of the most efficient approaches to solve (3) uses boundary integral equation formulations where the unknown quantity is a surface density $\phi(x)$ supported on ∂D . Once $\phi(x)$ is resolved, the far field pattern $u_\infty(\hat{x})$ can be computed easily using an integral such as

$$u_\infty(\hat{x}) = \int_{\partial D} e^{-ik\hat{x} \cdot y} \phi(y) ds(y). \quad (4)$$

Since we often discretize the integral (4) with a fixed number of points per wavelength in high frequency scattering applications, the discrete version of (4) takes exactly the form of (1) if we scale \hat{x} and y by a factor of k .

Another application of the sparse Fourier transform comes from the one-way seismic wave extrapolation procedure [21] of reflection seismology, where an important computation task is the evaluation of the following partial Fourier transform. Suppose that X and Ω now are the $N \times N$ Cartesian grid $\{(n_1, n_2), 0 \leq n_1, n_2 < N, n_1, n_2 \in \mathbb{Z}\}$. Given a function $f(\xi)$ for $\xi \in \Omega$, the *partial Fourier transform* computes for each $x \in X$,

$$u(x) = \sum_{|k| < c(x)} e^{2\pi i x \cdot k / N} f(k)$$

where $c(x)$ is a given function (often smooth on the scale of N). The main difficulty of this computational problem is that the summation constraint of k now depends on x through the function $c(x)$. This prevents us from using the standard fast Fourier transform (FFT) to compute $u(x)$ efficiently. Recently, [26] proposed an efficient solution to this problem by decomposing the summation domain hierarchically into simpler components. Each of these components then takes the form of the sparse Fourier transform in (1) and hence can be computed efficiently using the algorithm proposed in the current article.

1.3 Related work

Since the development of the fast Fourier transform (FFT) [13], there has been a lot of work devoted to the rapid computation of various Fourier summations. The nonuniform fast Fourier transform (NFFT) [1, 14, 18, 20, 24] generalizes the FFT and computes efficiently the Fourier summations when at least one of the spatial and Fourier grids is unequally spaced. The type-3 NFFT, which addresses the case that both grids are unequally spaced, can be used directly to compute the sparse Fourier transform in (1). However, since the complexity of the type-3 NFFT is of order $O(N^d \log N)$, the algorithm proposed in the current article is much more efficient for sparse data.

During the reviewing process of the current article, the referees pointed to our attention the articles [2, 11]. These papers considered the same sparse Fourier transform and the algorithm proposed there is similar to our approach. The main difference is on how to approximate the potential function $u^B(x)$ in (2). In [2, 11], $u^B(x)$ is approximated compactly by interpolating its values at a Cartesian grid with special interpolation schemes [6]. In our approach, however, we approximate $u^B(x)$ using equivalent charges. The accelerated computation of the equivalent sources using the tensor-product structure of the Fourier kernel is also new. The numerical results in Section 4 show that our approach results a significant improvement in accuracy when the same number of degrees of freedom is used for approximation.

Most functions or signals that we work with in high dimensional problems have strong smoothness properties for mixed derivatives. For many of them, the Fourier coefficients are only supported on the so-called hyperbolic cross

$$H = \{(\xi_1, \dots, \xi_d), \quad (1 + |\xi_1|) \cdots (1 + |\xi_d|) \leq N\}.$$

An obvious question here is how to evaluate these functions from their Fourier coefficients efficiently by exploiting the sparsity of the Fourier data. When the spatial points $\{x_i\}$ lie on the sparse grid [7], this task is called the *hyperbolic Fourier transform* and there exist *exact* fast algorithms [3, 17] of order $O(N \log^d N)$. In [16], Fenn et al. addressed the general case where the spatial points $\{x_i\}$ are chosen arbitrarily. However, their algorithm does not have the optimal complexity in the three dimensional case. Finally, when the spatial points $\{x_i\}$ are samples of a smooth curve, this problem starts to resemble the sparse Fourier transform considered here. In this case, we can generalize the algorithm proposed here to obtain an algorithm with optimal complexity.

1.4 Contents

The rest of this paper is organized as follows. In Section 2, we outline the overall structure of the butterfly algorithm. Section 3 proves the main analytic result and describe our algorithm in detail. We provide numerical results for both the two and three dimensional cases in Section 4. Finally, we conclude in Section 5 with some discussions on future work.

2 Butterfly Algorithm

In this section, we outline the butterfly algorithm described in [23] in the simple one dimensional case. Suppose now that $\{x_i\}_{1 \leq i \leq N}$ and $\{\xi_j\}_{1 \leq j \leq N}$ are two sets of points in $[0, N]$. Given the coefficients $\{f_j\}_{1 \leq j \leq N}$, the goal is to compute the values $\{u_i\}_{1 \leq i \leq N}$ defined by

$$u_i = \sum_j G(x_i, \xi_j) f_j$$

where $G(x, \xi)$ is a given kernel function. In order for us to use the butterfly algorithm, the kernel $G(x, \xi)$ should satisfy the following geometric low rank property. Suppose that A and B are two subintervals of $[0, N]$ of width w^A and w^B , respectively. If $w^A w^B \leq N$, then for any accuracy ε there exists a separated approximation of the Fourier kernel $G(x, \xi)$ of form

$$\left| G(x, \xi) - \sum_{t=1}^r \alpha_t^{AB}(x) \beta_t^{AB}(\xi) \right| < \varepsilon, \quad \forall x \in A, \forall \xi \in B \quad (5)$$

where $\alpha_t^{AB}(x)$ and $\beta_t^{AB}(\xi)$ are functions of only x and ξ , respectively. The separation rank r depends on only the accuracy ε and grows very slowly as ε decreases. This property holds for $G(x, \xi)$ appeared in many polynomial transforms and integral equations. The most important case is the Fourier transform where $G(x, \xi) = e^{2\pi i x \xi / N}$. Other examples include the Fourier-Bessel transform, the Legendre polynomial transform, and the Chebyshev polynomial transform. Geometrically, this property says that $\{G(x, \xi), \xi \in B\}$, viewed as functions of $x \in A$, stay approximately in a r -dimensional space. As a result, one expects that each member of $\{G(x, \xi), \xi \in B\}$ can be approximated by a linear combination of r carefully selected members of this set. It is shown in [10, 19] that this is indeed the case and one can choose $\alpha_t^{AB}(x) = G(x, \xi_t^{AB})$ with $\xi_t^{AB} \in B$ carefully chosen.

This low rank separated representation allows us to represent the interaction between two such sets A and B compactly. Let us recall that $u^B(x)$ is the potential generated by all the sources ξ_j in B , i.e.

$$u^B(x) := \sum_{\xi_j \in B} G(x, \xi_j) f_j.$$

Applying (5) for each ξ_j with weight f_j gives

$$\left| u^B(x) - \sum_{t=1}^r G(x, \xi_t^{AB}) \left(\sum_{\xi_j \in B} \beta_t^{AB}(\xi_j) f_j \right) \right| = O(\varepsilon), \quad \forall x \in A.$$

This says that the equivalent sources $\{\sum_j \beta_t^{AB}(\xi_j) f_j\}_{1 \leq t \leq r}$ located at points $\{\xi_t^{AB}\}_{1 \leq t \leq r}$ can serve as a compact representation for the interaction between A and B . We would like to emphasize that, even for a fixed B , the equivalent sources are often different for different A .

The basic idea of the butterfly algorithm is to combine the above observation with hierarchical decomposition. In order to simplify the description, we assume that N is an integer power of 2 and set $L = \log_2 N$. The target domain X and the source domain Ω are both equal to $[0, N]$. The algorithm starts by constructing hierarchical decompositions for X and Ω . At each level $l = 0, 1, \dots, L$, X and Ω are partitioned into 2^l subintervals $X_{l,m} = [N/2^l \cdot m, N/2^l \cdot (m+1)]$ with $m = 0, 1, \dots, 2^l - 1$ and $\Omega_{l,n} = [N/2^l \cdot n, N/2^l \cdot (n+1)]$ with $n = 0, 1, \dots, 2^l - 1$. In the special case of $l = 0$, we have $X_{0,0} = X$ and $\Omega_{0,0} = \Omega$.

The butterfly algorithm visits the levels of X from top to bottom and, at the same time, the levels of Ω from bottom to top.

1. Let us use l to denote the level of X that is currently visited by the algorithm. First, we set $l = 0$. The algorithm constructs the equivalent sources for the interaction between $X_{0,0}$ and $\Omega_{L,n}$ with $n = 0, 1, \dots, 2^L - 1$. In this case, the two boxes are $A = X_{0,0}$ and $B = \Omega_{L,n}$. From the previous discussion, we have

$$\left| u^B(x) - \sum_{t=1}^r G(x, \xi_t^{AB}) \sum_{\xi_j \in B} \beta_t^{AB}(\xi_j) f_j \right| = O(\varepsilon), \quad \forall x \in A,$$

and the equivalent sources are given by

$$\left\{ f_t^{AB} := \sum_{\xi_j \in B} \beta_t^{AB}(\xi_j) f_j \right\}_{1 \leq t \leq r}. \quad (6)$$

2. For $l = 1, 2, \dots, L$, the algorithm constructs the equivalent sources for the interaction between all pairs of boxes $A = X_{l,m}$ and $B = \Omega_{L-l,n}$ with $m = 0, 1, \dots, 2^l - 1$ and $n = 0, 1, \dots, 2^{L-l} - 1$. Let us use P to denote A 's parent, i.e. $P = X_{l-1, \lfloor m/2 \rfloor}$, and B_1 and B_2 to denote the children of B , i.e. $B_1 = \Omega_{L-l+1, 2n}$ and $B_2 = \Omega_{L-l+1, 2n+1}$. From the computation at the previous level, we have already computed the equivalent sources $\{f_t^{PB_1}\}_{1 \leq t \leq r}$ and $\{f_t^{PB_2}\}_{1 \leq t \leq r}$. Therefore, the interaction between P and B_1 is approximated by

$$\left| u^{B_1}(x) - \sum_{t=1}^r G(x, \xi_t^{PB_1}) f_t^{PB_1} \right| = O(\varepsilon), \quad \forall x \in P,$$

and the one between P and B_2 by

$$\left| u^{B_2}(x) - \sum_{t=1}^r G(x, \xi_t^{PB_2}) f_t^{PB_2} \right| = O(\varepsilon), \quad \forall x \in P.$$

Since $u^B(x) = u^{B_1}(x) + u^{B_2}(x)$, summing them up gives us

$$\left| u^B(x) - \left(\sum_{t=1}^r G(x, \xi_t^{PB_1}) f_t^{PB_1} + \sum_{t=1}^r G(x, \xi_t^{PB_2}) f_t^{PB_2} \right) \right| = O(\varepsilon), \quad \forall x \in P.$$

As $A \subset P$, this approximation is certainly true for any $x \in A$. Since A is on level l and B is on level $L-l$, $w^A w^B = N$. Therefore, we can take $\{f_t^{PB_1}\}_{1 \leq t \leq r}$ and $\{f_t^{PB_2}\}_{1 \leq t \leq r}$ as the sources in B and apply the low rank representation. The final result is that interaction between A and B can be approximated accurately by a set of equivalent sources

$$\left\{ f_t^{AB} := \sum_{u=1}^t \beta_t^{AB}(\xi_u^{PB_1}) f_u^{PB_1} + \sum_{u=1}^t \beta_t^{AB}(\xi_u^{PB_2}) f_u^{PB_2} \right\}_{1 \leq t \leq r}. \quad (7)$$

3. When $l = L$, we have $A = X_{L,m}$ with $m = 0, 1, \dots, 2^L - 1$ and $B = \Omega_{0,0}$. For each pair (A, B) , the equivalent sources $\{f_t^{AB}\}_{1 \leq t \leq r}$ provides an approximation:

$$\left| u^B(x) - \sum_{t=1}^r G(x, \xi_t^{AB}) f_t^{AB} \right| = O(\varepsilon), \quad \forall x \in A = X_{L,m}.$$

For each $x_i \in A = X_{L,m}$, we set

$$u_i \approx \sum_{t=1}^r G(x_i, \xi_t^{AB}) f_t^{AB}. \quad (8)$$

An schematic illustration of the algorithm is given in Figure 2.

Let us estimate the number of operations used in the butterfly algorithm. In the first step, the number of operations in (6) for each $B = \Omega_{L,n}$ is proportional to the number of points ξ_j in B since the rank r is a constant for a fixed ε . As there are only $O(N)$ points in $\{\xi_j\}$ in total, the first step takes $O(N)$ operations. At each level l of the second step, we construct the equivalent sources $\{f_t^{AB}\}_{1 \leq t \leq r}$ for all pairs of boxes (A, B) with $A = X_{l,m}$ and $B = \Omega_{L-l,n}$. It is clear from (7) the number of steps used for each $\{f_t^{AB}\}_{1 \leq t \leq r}$ is

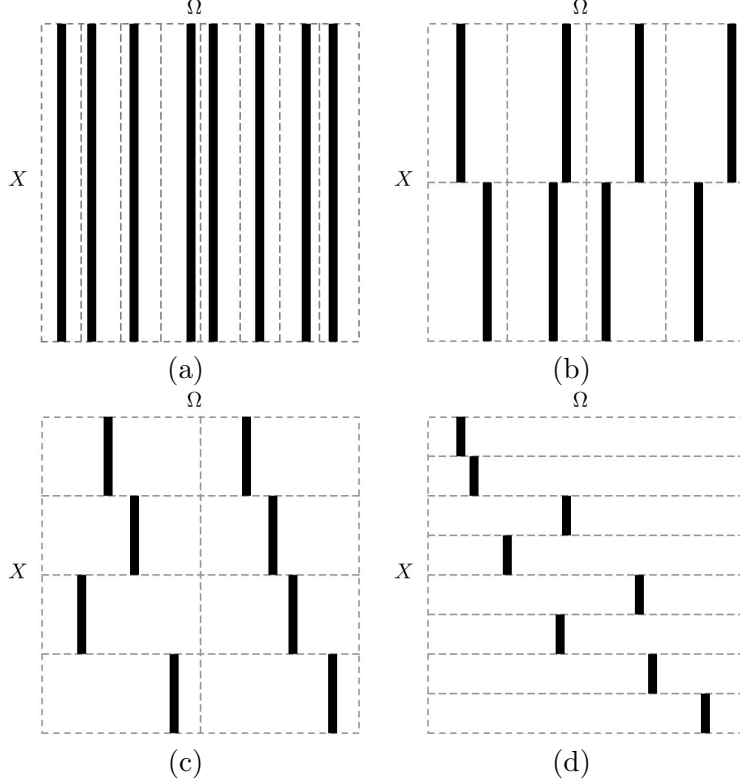


Figure 2: An schematic illustration of the butterfly algorithm when $N = 8$ and $L = 3$. The four plots (a)-(d) represent the status of the algorithm when $l = 0, 1, 2, 3$. At each level l , the interaction between X and Ω is partitioned into N blocks, each of which corresponds to the interaction between $A = X_{l,m}$ and $B = \Omega_{L-l,n}$. The black columns in each block stand for the components $\{G(x, \xi_t^{AB})\}_{1 \leq t \leq r}$ associated with the equivalent sources at $\{\xi_t^{AB}\}_{1 \leq t \leq r}$ and $x \in A$. In this figure, the rank r is set to be 1 for simplicity. The white columns give no contribution. It is clear from the algorithm that, as l increases, the size of $A = X_{l,m}$ decreases. Therefore, the lengths of the black columns shrink accordingly. In each plot, the whole potential function $u(x) := \sum_j G(x, \xi_j) f_j$ is approximated by the sum of all the dark columns. Finally in (d), the value of $u(x)$ at any $x \in X$ can be computed by summing the r columns in the horizontal block that contains x .

$O(r^2) = O(1)$ for fixed r . Since $m = 0, 1, \dots, 2^l - 1$ and $n = 0, 1, \dots, 2^{L-l} - 1$, there are exactly $2^l \cdot 2^{L-l} = 2^L = N$ pairs of (A, B) to consider. Therefore, the number of operations for each level l is $O(N)$. Summing over all $L = \log_2 N$ levels gives an $O(N \log N)$ operation count for the second step. In the final step, (8) clearly takes only $O(1)$ steps for each point x_i . Therefore, the total number of operations of the butterfly algorithm is of order $O(N \log N)$.

So far, we have not discussed how to select the locations $\{\xi_t^{AB}\}_{1 \leq t \leq r}$ and how to construct the functions $\{\beta_t^{AB}(\xi)\}_{1 \leq t \leq r}$. In [23], this is done with a preprocessing step using the so-called interpolative decomposition [10, 19]. The number of steps used in the preprocessing step is equal to $O(N^2)$, which can be quite costly for large values of N . The associated storage requirement is approximately $O(r^2 N \log N)$ because one needs to store the interpolation matrices for all pairs (A, B) appeared in the algorithm [23]. This storage requirement can become a bottleneck in practice when N is large and r is no longer negli-

gible. In Section 3, we will show how to remove these time and storage constraints in the case of the sparse Fourier transform.

From the above discussion, we conclude that the general structure of the butterfly algorithm utilizes three important components:

- Hierarchical structures for the target domain X and the source domain Ω .
- An approximate low rank property of the kernel $G(x, \xi)$ whenever the target region A and the source region B satisfy the condition $w^A w^B \leq N$, see (5).
- A method to represent $u^B(x)$ for $x \in A$ compactly, i.e. the locations $\{\xi_t^{AB}\}_{1 \leq t \leq r}$ and how to compute the equivalent sources $\{f_t^{AB}\}_{1 \leq t \leq r}$, see (6) and (7).

3 Sparse Fourier Transform

In this section, we describe how to adapt the butterfly algorithm to the sparse Fourier transform. Let us recall that $X = \Omega = [0, N]^d$, and $\{x_i\}$ and $\{\xi_j\}$ are samples of the curves M_X and M_Ω , respectively.

As we mentioned at the end of Section 2, the butterfly algorithm uses three major components. For the hierarchical structures, we construct two adaptive quadtrees T_X and T_Ω for the sets $\{x_i\}$ and $\{\xi_j\}$, respectively. The quadtree T_X takes the whole spatial domain X as its root node and partitions the set $\{x_i\}$ recursively. Similarly, the quadtree T_Ω takes the whole Fourier domain Ω as its root node and partitions $\{\xi_j\}$. Since the curves M_X and M_Ω are assumed to be smooth on the order of N , at each level l of T_X and T_Ω there are at most $O(2^l)$ nonempty boxes. Figure 3 illustrates the quadtrees for a simple case with $N = 64$.

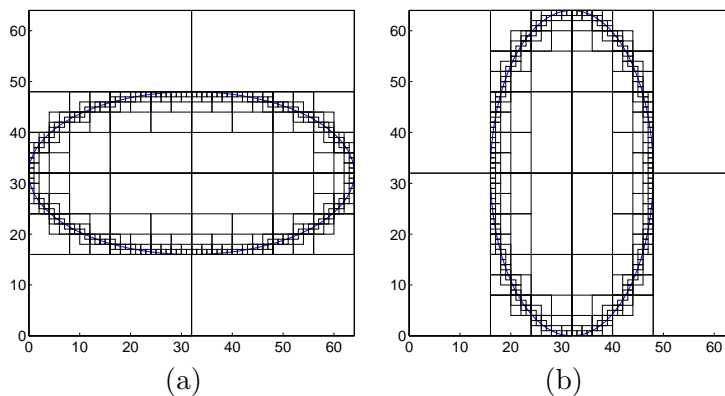


Figure 3: The quadtrees. (a) T_X . (b) T_Ω . Both quadtrees are constructed adaptively so that boxes with no points inside are discarded. Here $N = 64$ and $L = 6$.

3.1 Low rank property

In this section, we prove the low rank property of the interaction between two boxes $A \in T_X$ and $B \in T_\Omega$ that satisfy the condition $w^A w^B \leq N$.

Theorem 3.1. *Let A be a box in T_X and B be a box in T_Ω . Suppose the width w^A and w^B of A and B satisfy $w^A w^B \leq N$. Then, for any $\varepsilon > 0$, there exists a constant $T(\varepsilon)$ and functions $\{\alpha_t(x), 1 \leq t \leq T(\varepsilon)\}$ and $\{\beta_t(k), 1 \leq t \leq T(\varepsilon)\}$ such that*

$$\left| e^{2\pi i x \cdot k/N} - \sum_{t=1}^{T(\varepsilon)} \alpha_t(x) \beta_t(k) \right| \leq \varepsilon$$

for any $x \in A$ and $k \in B$.

The proof of Theorem 3.1 is based on the following elementary lemma (see [15] for a proof).

Lemma 3.2. *For any $Z > 0$ and $\varepsilon > 0$, let $S = \lceil \max(4e\pi Z, \log_2(1/\varepsilon)) \rceil$. Then*

$$\left| e^{2\pi i x} - \sum_{t=0}^{S-1} \frac{(2\pi i x)^t}{t!} \right| \leq \varepsilon$$

for any x with $|x| \leq Z$.

Proof of Theorem 3.1. The proof of the theorem is quite simple. Let us use $c^A = (c_1^A, c_2^A)$ and $c^B = (c_1^B, c_2^B)$ to denote the centers of boxes A and B , respectively. Writing $x = c^A + x'$ and $k = c^B + k'$, we have

$$e^{2\pi i x \cdot k/N} = e^{2\pi i (c^A + x') \cdot (c^B + k')/N} = e^{2\pi i (c^A + x') \cdot c^B/N} \cdot e^{2\pi i x' \cdot k'/N} \cdot e^{2\pi i c^A \cdot k'/N}.$$

Notice that the first and the third terms depends only on x' and k' , respectively. Therefore, we only need to construct a factorization for the second term. Since $|x'| \leq w^A/\sqrt{2}$ and $|k'| \leq w^B/\sqrt{2}$, $|x' \cdot k'|/N \leq \frac{1}{2}$. Invoking the lemma for $Z = \frac{1}{2}$, we obtain the following approximation with $S(\varepsilon)$ terms:

$$\left| e^{2\pi i x' \cdot k'/N} - \sum_{t=0}^{S(\varepsilon)-1} \frac{(2\pi i x' \cdot k'/N)^t}{t!} \right| \leq \varepsilon.$$

After expanding each term of the sum using $x' \cdot k' = (x'_1 k'_1 + x'_2 k'_2)$, we have an approximate expansion

$$\left| e^{2\pi i x \cdot k/N} - \sum_{t=1}^{T(\varepsilon)} \alpha_t(x) \beta_t(k) \right| \leq \varepsilon$$

where $T(\varepsilon)$ only depends on the accuracy ε . □

3.2 Equivalent sources

Once we have the low rank property of the interaction between two regions A and B satisfying $w^A w^B \leq N$, one approach to obtain an $O(N \log N)$ butterfly algorithm for the sparse Fourier transform is to use the interpolative decomposition as [23]. However, as we pointed out earlier, such an approach has two bottlenecks: the $O(N^2)$ precomputation step and the $O(rN \log N)$ storage requirement. In this section, we show how to remove these constraints by choosing the locations $\{\xi_t^{AB}\}$ carefully.

Given a box B in T_Ω , we recall that $u^B(x)$ is potential field generated by the sources inside B :

$$u^B(x) = \sum_{\xi_j \in B} e^{2\pi i x \cdot \xi_j / N} f_j.$$

For a fixed integer p , we define a Chebyshev grid in the interval $[-1/2, 1/2]$

$$\left\{ \alpha_i = \frac{1}{2} \cos(i/(p-1)\pi) \right\}_{0 \leq i \leq p-1}.$$

Let $c^B = (c_1^B, c_2^B)$ be the center of B . For the efficiency reason to be discussed shortly, we pick the equivalent sources to be located on the following tensor-product grid of B :

$$\{\xi_{st}^B := (c_1^B + \alpha_s w^B, c_2^B + \alpha_t w^B), \quad s, t = 0, 1, \dots, p-1\}$$

(see Figure 4). Notice that the locations $\{\xi_{st}^B\}$ are now independent of the box A . The size of the grid p depends on the prescribed accuracy ε (see Section 4).

In the first step of the butterfly algorithm (see Section 2), one needs to construct the equivalent sources $\{f_{st}^{AB}\}$ (located at $\{\xi_{st}^B\}$) for the interaction between the root box A of T_X and a leaf box B of T_Ω . To do this, we first select a grid in A :

$$\{x_{st}^A := (c_1^A + \alpha_s w^A, c_2^A + \alpha_t w^A), \quad s, t = 0, 1, \dots, p-1\}$$

where $c^A = (c_1^A, c_2^A)$ is A 's center (see Figure 4). Then, $\{f_{st}^{AB}\}$ are computed so that they generate the field $\{u_{st}^{AB} := u^B(x_{st}^A)\}$ at the points $\{x_{st}^A\}$; i.e.

$$\sum_{s't'} e^{2\pi i x_{st}^A \cdot \xi_{s't'}^B / N} f_{s't'}^{AB} = u_{st}^{AB}.$$

Writing this into a matrix form $Mf = u$ and using the definitions of $\{x_{st}^A\}$ and $\{\xi_{s't'}^B\}$, we can decompose the $p^2 \times p^2$ matrix M into a Kronecker product $M = M_1 \otimes M_2$, where

$$M_1 = \left(e^{2\pi i (c_1^A + \alpha_s w^A)(c_1^B + \alpha_{s'} w^B) / N} \right)_{ss'}, \quad M_2 = \left(e^{2\pi i (c_2^A + \alpha_t w^A)(c_2^B + \alpha_{t'} w^B) / N} \right)_{tt'}.$$

Since $(M_1 \otimes M_2)^{-1} = M_1^{-1} \otimes M_2^{-1}$, in order to compute $f = M^{-1}u$ we only need to invert the $p \times p$ matrices M_1 and M_2 . Expanding the formula for M_1 , we get

$$(M_1)_{ss'} = e^{2\pi i (c_1^A + \alpha_s w^A) c_1^B / N} \cdot e^{2\pi i \alpha_s \alpha_{s'} \frac{w^A w^B}{N}} \cdot e^{2\pi i c_1^A (\alpha_{s'} w^B) / N} \quad (9)$$

Noticing that the first and the third terms depend only on s and s' respectively and $w^A w^B / N = 1$, we can rewrite M_1 into a factorization $M_1 = M_{11} \cdot G \cdot M_{12}$, where M_{11} and M_{12} are two diagonal matrices and the center matrix G given by $(G)_{ss'} = e^{2\pi i \alpha_s \alpha_{s'}}$ is independent of N and the boxes A and B . The situation for M_2 is exactly the same. The result of this discussion is that we have successfully reduced the complexity of $f = M^{-1}u$ from $O(p^4)$ operations to $O(p^3)$ operations using the Kronecker product structure of M .

Now, let us assume that B is a non-leaf box of T_Ω , A is a box in T_X with $w^A w^B = N$. In the second step of the butterfly algorithm (see Section 2), one needs to construct the equivalent sources $\{f_{st}^{AB}\}$ from the equivalent sources of B 's children. Let us denote the children of B by $B_c, c = 1, \dots, 4$ and the parent of A by P . As we construct the equivalent sources in a bottom-up traversal of T_Ω , when one reaches the box B the equivalent sources

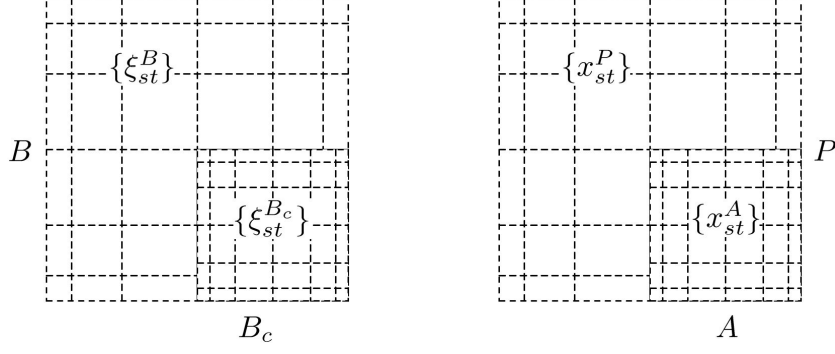


Figure 4: The construction of $\{f_{st}^{AB}\}$ using $\{f_{st}^{PB_c}\}$. B_c is one of B 's child boxes and P is A 's parent box. We first evaluate the potentials $\{u_{st}^{AB}\}$ located at $\{x_{st}^A\}$ using $f_{s't'}^{PB_c}$ and then solve for $\{f_{st}^{AB}\}$ so that they produce the same potentials.

of B_c have already been computed. Since $A \subset P$, the check potentials $\{u_{st}^{AB}\}$ can be approximated using the equivalent sources $\{f_{st}^{PB_c}\}$ of B_c :

$$u_{st}^{AB} \approx \sum_{c=1}^4 \left(\sum_{s't'} e^{2\pi i x_{st}^A \cdot \xi_{s't'}^{B_c} / N} f_{s't'}^{PB_c} \right)$$

for $0 \leq s, t \leq p-1$ (see Figure 4). The inner sum $\sum_{s't'} e^{2\pi i x_{st}^A \cdot \xi_{s't'}^{B_c} / N} f_{s't'}^{PB_c}$ for each fixed c can be rewritten into a matrix form Ef . Using again the Kronecker product, we can decompose E as $E_1 \otimes E_2$ where

$$E_1 = \left(e^{2\pi i (c_1^A + \alpha_s w^A) (c_1^{B_c} + \alpha_{s'} w^{B_c}) / N} \right)_{ss'}, \quad E_2 = \left(e^{2\pi i (c_2^A + \alpha_t w^A) (c_2^{B_c} + \alpha_{t'} w^{B_c}) / N} \right)_{tt'}.$$

Expanding the formula for E_1 , we get

$$(E_1)_{ss'} = e^{2\pi i (c_1^A + \alpha_s w^A) c_1^{B_c} / N} \cdot e^{2\pi i \alpha_s \alpha_{s'} \frac{w^A w^{B_c}}{N}} \cdot e^{2\pi i c_1^A (\alpha_{s'} w^{B_c}) / N} \quad (10)$$

Noticing that the first and the third terms depend only on s and s' respectively and $w^A w^{B_c} = N/2$, we can write E_1 into a factorization $E_1 = E_{11} \cdot H \cdot E_{12}$ where E_{11} and E_{12} are again diagonal matrices and the matrix H given by $(H)_{ss'} = e^{\pi i \alpha_s \alpha_{s'}}$ is independent of N . Now, it is clear that the computation of the product Ef is also reduced from $O(p^4)$ operations to $O(p^3)$ operations using the Kronecker product structure of M .

3.3 Algorithm

We now summarize the proposed algorithm. It contains the following steps:

1. Construct the quadtrees T_X and T_Ω for the point sets $\{x_i\}$ and $\{\xi_j\}$, respectively. These trees are constructed adaptively and all the leaf boxes are of unit size.
2. Construct the equivalent sources for the interaction between the root box A of T_X and the leaf boxes of T_Ω . For each leaf box B in T_Ω , we calculate $\{f_{st}^{AB}\}$ by matching the potentials $\{u_{st}^{AB}\}$ (see Section 3.2).

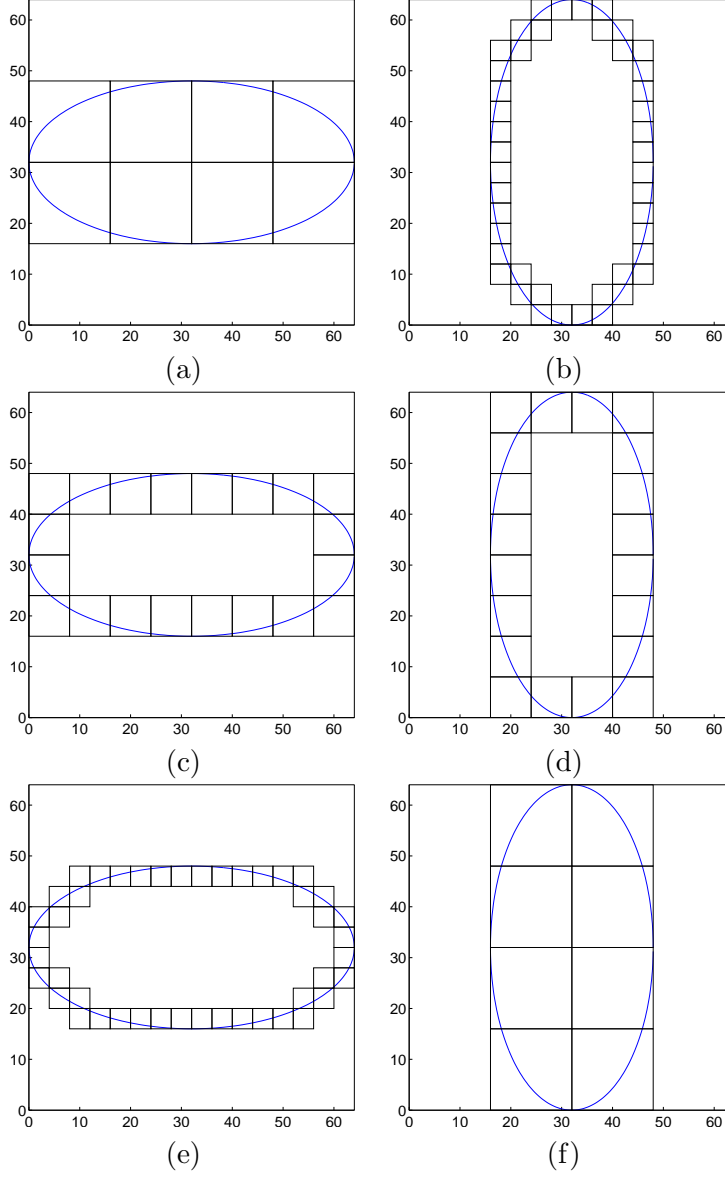


Figure 5: Computation at different levels of the algorithm. Here $N = 64$ and $L = 6$. (a) and (b) show the boxes in T_X and T_Ω that are visited when $l = 2$. For each pair of the boxes (A, B) , we construct the equivalent sources $\{f_{st}^{AB}\}$. (c) and (d) show the boxes in T_X and T_Ω that are visited when $l = 3$. (e) and (f) show the boxes in T_X and T_Ω that are visited when $l = 4$.

3. Travel down in T_X and up in T_Ω simultaneously. For each level l of T_X , we construct the equivalent sources $\{f_{st}^{AB}\}$ for the interaction between a box A at level l of T_X and a box B at level $L - l$ of T_Ω . The widths of these boxes satisfy $w^A w^B = N$. These equivalent sources $\{f_{st}^{AB}\}$ are computed from the equivalent sources of $\{f_{st}^{PB_c}\}$ where P is A 's parent and $\{B_c, c = 1, \dots, 4\}$ are the B 's children (see Section 3.2).
4. Compute $\{u_j\}$. Now B is the root box of T_Ω . For each leaf box A in T_X , we compute

for each $x_j \in A$

$$u_j \approx \sum_{st} e^{2\pi i x_j \cdot \xi_{st}^B / N} f_{st}^{AB}.$$

Figure 5 illustrates the pairs of boxes visited in T_X and T_Ω at different levels of the algorithm.

Let us first estimate the number of operations of the proposed algorithm. The first step clearly takes only $O(N \log N)$ operations since there are at most $O(N)$ points in both $\{x_i\}$ and $\{\xi_j\}$. Since the curves M_X and M_Ω are smooth on the scale of N , for a given size w , there are at most $O(N/w)$ non-empty boxes of size w in both T_X and T_Ω . In particular, we have at most $O(N)$ leaf boxes in T_X and T_Ω . This implies that the second and fourth steps take at most $O(N)$ operations. To analyze the third step, we estimate level by level. For a fixed level l , there are at most 2^l non-empty boxes in T_Ω on that level, each of size $N/2^l$. For each box B on level l , we need to construct $\{f_{st}^{AB}\}$ for all the boxes A in T_X with size $N/(N/2^l) = 2^l$. It is clear that there are at most $N/2^l$ non-empty boxes in T_X of that size. Since the construction for each set of equivalent sources take only a constant number of operations, the total complexity for level l is $O(2^l \times N/2^l) = O(N)$. As there are at most $O(\log N)$ levels, the third step takes at most $O(N \log N)$ operations. Summing over all the steps, we conclude that our algorithm is $O(N \log N)$.

Our algorithm is also efficient in terms of storage space. Since we give explicit construct formulas (9) and (10) for constructing the equivalent sources, we only need to store the equivalent sources during the computation. At each level, we only keep $O(N)$ set of equivalent sources, each of which takes $O(1)$ storage space. Noticing that, at any point of the algorithm, we only need to keep the equivalent sources for two adjacent levels, therefore the storage requirement of our algorithm is only $O(N)$. This is where our algorithm differs from the one in [23] where they need to store $O(N \log N)$ small matrices for the interpolative decomposition [10, 19], each of which is of size $O(r) \times O(r)$.

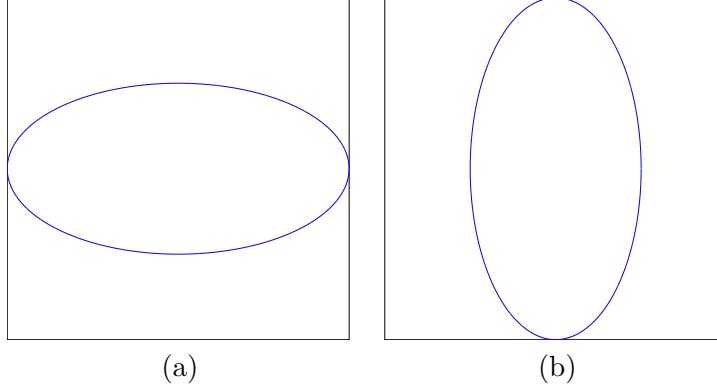
The three dimensional case is similar. Since M_X and M_Ω are smooth surfaces in $[0, 1]^3$, the number of points in $\{x_i\}$ and $\{\xi_j\}$ are $O(N^2)$ instead. The Kronecker product decomposition is still valid and, therefore, we can construct the equivalent sources efficiently in $O(p^4)$ operations instead of $O(p^6)$. The algorithm remains exactly the same and a similar complexity analysis gives an operation count of $O(N^2 \log N)$, which is almost linear in terms of the number of points.

4 Numerical Results

In this section, we provide some numerical examples to illustrate the properties of our algorithm. All of the numerical results are obtained on a desktop computer with a 2.8GHz CPU. The accuracy of the algorithm depends on p , which is the size of the Chebyshev grid used for the equivalent sources. In the following examples, we pick $p = 5, 7, 9$. The larger the value of p , the better the accuracy.

4.1 2D case

For each example, we sample the curves M_X and M_Ω with 5 points per unit length. $\{f_j\}$ are randomly generated numbers with mean 0. We use $\{u_i^a\}$ to denote the results of our algorithm. To study the accuracy of our algorithm, we pick a set S of size 200 and estimate



(N, p)	P	$T_a(\text{sec})$	$T_d(\text{sec})$	T_d/T_a	ε_a
(1024,5)	1.64e+4	1.23e+0	3.03e+1	2.46e+1	2.29e-3
(2048,5)	3.28e+4	2.78e+0	1.20e+2	4.30e+1	2.51e-3
(4096,5)	6.55e+4	6.12e+0	4.88e+2	7.98e+1	2.42e-3
(8192,5)	1.31e+5	1.35e+1	1.95e+3	1.45e+2	2.57e-3
(16384,5)	2.62e+5	2.96e+1	7.80e+3	2.64e+2	2.53e-3
(32768,5)	5.24e+5	6.43e+1	3.17e+4	4.94e+2	2.57e-3
(1024,7)	1.64e+4	2.03e+0	3.03e+1	1.49e+1	8.11e-6
(2048,7)	3.28e+4	4.54e+0	1.20e+2	2.63e+1	7.28e-6
(4096,7)	6.55e+4	1.00e+1	4.85e+2	4.83e+1	7.37e-6
(8192,7)	1.31e+5	2.25e+1	1.96e+3	8.71e+1	8.35e-6
(16384,7)	2.62e+5	5.12e+1	7.93e+3	1.55e+2	9.04e-6
(32768,7)	5.24e+5	1.18e+2	3.17e+4	2.70e+2	9.12e-6
(1024,9)	1.64e+4	3.17e+0	2.95e+1	9.30e+0	1.53e-8
(2048,9)	3.28e+4	7.02e+0	1.21e+2	1.73e+1	1.61e-8
(4096,9)	6.55e+4	1.55e+1	4.85e+2	3.13e+1	1.53e-8
(8192,9)	1.31e+5	3.45e+1	1.96e+3	5.68e+1	1.62e-8
(16384,9)	2.62e+5	7.57e+1	7.92e+3	1.05e+2	1.80e-8
(32768,9)	5.24e+5	1.83e+2	3.17e+4	1.73e+2	1.73e-8

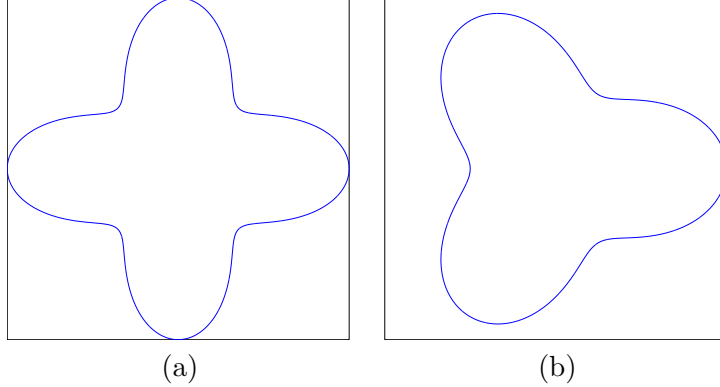
Table 1: 2D results. Top: Both M_X and M_Ω are ellipses in unit box $[0, 1]^2$. Bottom: Running time, speedup factor and accuracy for different combinations of N and p . N is the size of the domain, p is the size of the Cartesian grid used for the equivalent sources, P is the maximum of the numbers of points in $\{x_i\}$ and $\{\xi_j\}$, T_a is the running time of our algorithm in seconds, T_d is the estimated running time of the direct evaluation in seconds, T_d/T_a is the speedup factor, and finally ε_a is the approximation error.

the error by

$$\sqrt{\frac{\sum_{i \in S} |u_i - u_i^a|^2}{\sum_{i \in S} |u_i|^2}}$$

where $\{u_i\}_{i \in S}$ are the exact potentials computed by direct evaluation.

Before reporting the numerical results, let us summarize the notations that are used in the tables. N is the size of the domain, p is the size of the Chebyshev grid used for the equivalent sources, P is the maximum of the numbers of points in $\{x_i\}$ and $\{\xi_j\}$, T_a is the running time of our algorithm in seconds, T_d is the estimated running time of the direct



(N, p)	P	$T_a(\text{sec})$	$T_d(\text{sec})$	T_d/T_a	ε_a
(1024,5)	1.64e+4	1.95e+0	3.03e+1	1.55e+1	2.50e-3
(2048,5)	3.28e+4	4.37e+0	1.21e+2	2.77e+1	2.56e-3
(4096,5)	6.55e+4	9.75e+0	4.85e+2	4.97e+1	2.56e-3
(8192,5)	1.31e+5	2.11e+1	1.96e+3	9.29e+1	3.19e-3
(16384,5)	2.62e+5	4.82e+1	7.93e+3	1.65e+2	2.95e-3
(32768,5)	5.24e+5	1.02e+2	3.18e+4	3.12e+2	2.75e-3
(1024,7)	1.64e+4	3.16e+0	3.03e+1	9.59e+0	8.69e-6
(2048,7)	3.28e+4	7.08e+0	1.21e+2	1.71e+1	8.87e-6
(4096,7)	6.55e+4	1.58e+1	4.85e+2	3.06e+1	8.42e-6
(8192,7)	1.31e+5	3.52e+1	1.96e+3	5.56e+1	9.59e-6
(16384,7)	2.62e+5	8.22e+1	7.92e+3	9.63e+1	9.61e-6
(32768,7)	5.24e+5	1.93e+2	3.17e+4	1.64e+2	9.09e-6
(1024,9)	1.64e+4	4.94e+0	3.03e+1	6.14e+0	1.63e-8
(2048,9)	3.28e+4	1.11e+1	1.20e+2	1.08e+1	1.62e-8
(4096,9)	6.55e+4	2.44e+1	4.82e+2	1.97e+1	1.81e-8
(8192,9)	1.31e+5	5.56e+1	1.95e+3	3.51e+1	1.77e-8
(16384,9)	2.62e+5	1.21e+2	7.93e+3	6.55e+1	1.87e-8
(32768,9)	5.24e+5	2.85e+2	3.31e+4	1.16e+2	1.93e-8

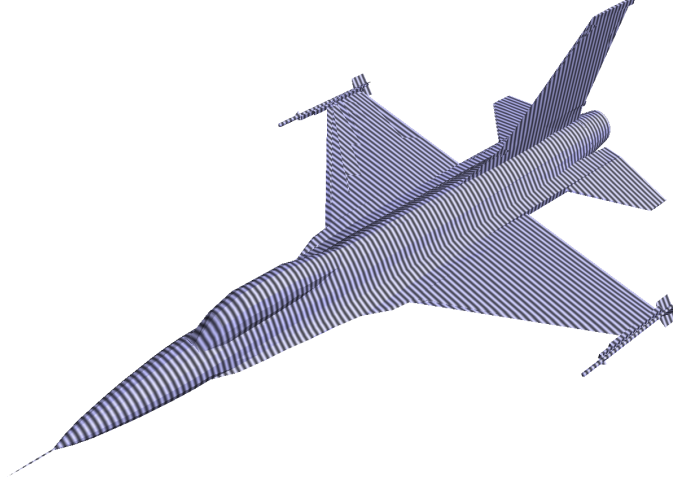
Table 2: 2D results. Top: M_X and M_Ω are two smooth curves in unit box $[0, 1]^2$. Bottom: Running time, speedup factor and accuracy for different combinations of N and p .

evaluation in seconds, T_d/T_a is the speedup factor, and finally ε_a is the approximation error.

Tables 1 and 2 report the results for two testing examples. From these tables, it is quite clear that the running time T_a grows indeed almost linearly in terms of the number of points, and the accuracy ε_a remains almost unchanged. For larger values of N , we obtain a substantial speedup over the direct evaluation.

4.2 3D case

We apply our algorithm to the problem of computing the far field pattern (4). In this setup, M_X is always a sphere and M_Ω is the boundary of the scatter. We sample the surface M_X and M_Ω with about $5 \times 5 = 25$ points per unit area. Tables 3 and 4 summarize the results of two typical examples in evaluating the far field pattern of scattering fields.



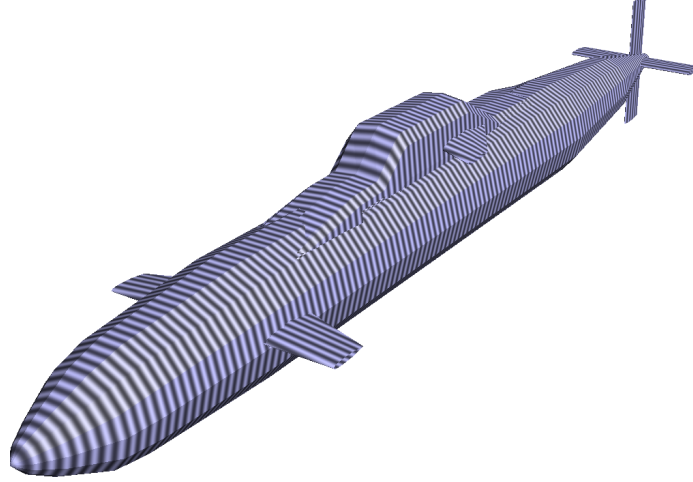
(N, p)	P	$T_a(\text{sec})$	$T_d(\text{sec})$	T_d/T_a	ε_a
(16,5)	2.14e+4	2.46e+0	1.50e+1	6.10e+0	1.79e-3
(32,5)	8.19e+4	9.66e+0	1.80e+2	1.87e+1	2.25e-3
(64,5)	3.22e+5	3.91e+1	2.77e+3	7.09e+1	2.35e-3
(128,5)	1.28e+6	1.64e+2	4.32e+4	2.63e+2	2.45e-3
(256,5)	5.13e+6	6.70e+2	6.91e+5	1.03e+3	2.80e-3
(16,7)	2.14e+4	6.69e+0	1.50e+1	2.24e+0	5.35e-6
(32,7)	8.19e+4	2.62e+1	1.88e+2	7.19e+0	8.06e-6
(64,7)	3.22e+5	1.06e+2	2.80e+3	2.65e+1	7.62e-6
(128,7)	1.28e+6	4.34e+2	4.32e+4	9.95e+1	8.67e-6
(16,9)	2.14e+4	1.42e+1	1.50e+1	1.06e+0	1.20e-8
(32,9)	8.19e+4	5.56e+1	1.80e+2	3.24e+0	1.54e-8
(64,9)	3.22e+5	2.26e+2	2.80e+3	1.24e+1	1.65e-8
(128,9)	1.28e+6	9.21e+2	4.37e+4	4.74e+1	1.57e-8

Table 3: 3D results. Top: the surface M_Ω is the boundary of an F16 airplane model. Bottom: Running time, speedup factor and accuracy for different combinations of N and p .

5 Conclusions and Discussions

In this paper, we introduced an efficient algorithm for computing Fourier transforms with sparse spatial and Fourier data supported on curves and surfaces. Our algorithm is accurate and has the optimal $O(N^{d-1} \log N)$ complexity. The main idea behind the algorithm is based on a low rank property concerning the interaction between spatial and frequency regions that follow a certain geometrical condition. We use the equivalent sources supported on tensor-product Chebyshev grids as the low rank representation and exploit the tensor-product property of the Fourier transform to achieve maximum efficiency. Furthermore, our algorithm requires only linear storage space.

The problem considered in this paper is only one example of many computational issues regarding highly oscillatory integrals. Some other examples include the computation of Fourier integer operators [25], scattering fields for high frequency waves [12], and Kirchhoff migrations [4]. In the past two decades, many algorithms have been developed to address



(N, p)	P	$T_a(\text{sec})$	$T_d(\text{sec})$	T_d/T_a	ε_a
(16,5)	2.14e+4	3.14e+0	1.07e+1	3.41e+0	1.55e-3
(32,5)	8.19e+4	1.26e+1	1.31e+2	1.04e+1	1.62e-3
(64,5)	3.22e+5	5.04e+1	2.13e+3	4.22e+1	2.18e-3
(128,5)	1.28e+6	2.04e+2	3.38e+4	1.65e+2	2.41e-3
(256,5)	5.13e+6	8.24e+2	5.58e+5	6.77e+2	2.26e-3
(16,7)	2.14e+4	8.59e+0	8.58e+0	9.98e-1	5.55e-6
(32,7)	8.19e+4	3.34e+1	1.39e+2	4.17e+0	6.36e-6
(64,7)	3.22e+5	1.33e+2	2.10e+3	1.57e+1	7.02e-6
(128,7)	1.28e+6	5.41e+2	3.48e+4	6.43e+1	8.43e-6
(16,9)	2.14e+4	1.87e+1	8.58e+0	4.58e-1	1.01e-8
(32,9)	8.19e+4	7.08e+1	1.31e+2	1.85e+0	1.16e-8
(64,9)	3.22e+5	2.83e+2	2.16e+3	7.63e+0	1.38e-8
(128,9)	1.28e+6	1.15e+3	3.46e+4	3.00e+1	1.47e-8

Table 4: 3D results. Top: the surface M_Ω is the boundary of a submarine model. Bottom: Running time, speedup factor and accuracy for different combinations of N and p .

these challenging computational tasks efficiently. Some examples include [5, 8, 9, 15]. It would be interesting to see whether the ideas behind these approaches can be used to study the problem addressed in this paper, and vice versa.

Acknowledgments. The research presented in this paper was supported by an Alfred P. Sloan Fellowship and a startup grant from the University of Texas at Austin. The author would like to thank the reviewers for their comments and suggestions.

References

- [1] C. Anderson and M. D. Dahleh. Rapid computation of the discrete Fourier transform. *SIAM J. Sci. Comput.*, 17(4):913–919, 1996.
- [2] A. Ayliner, W. C. Chew, and J. Song. A sparse data fast Fourier transform (sdfft) - algorithm and implementation. *Antennas and Propagation Society International Symposium, 2001. IEEE*, 4:638–641 vol.4, 2001.

- [3] G. Baszenski and F.-J. Delves. A discrete Fourier transform scheme for Boolean sums of trigonometric operators. In *Multivariate approximation theory, IV (Oberwolfach, 1989)*, volume 90 of *Internat. Ser. Numer. Math.*, pages 15–24. Birkhäuser, Basel, 1989.
- [4] G. Beylkin. The inversion problem and applications of the generalized Radon transform. *Comm. Pure Appl. Math.*, 37(5):579–599, 1984.
- [5] O. P. Bruno and L. A. Kunyansky. A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, tests, and applications. *J. Comput. Phys.*, 169(1):80–110, 2001.
- [6] O. Bucci, C. Gennarelli, and C. Savarese. Optimal interpolation of radiated fields over a sphere. *Antennas and Propagation, IEEE Transactions on*, 39(11):1633–1643, Nov 1991.
- [7] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numer.*, 13:147–269, 2004.
- [8] E. Candès, L. Demanet, and L. Ying. Fast computation of Fourier integral operators. *SIAM J. Sci. Comput.*, 29(6):2464–2493 (electronic), 2007.
- [9] 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(1):300–325, 2006.
- [10] H. Cheng, Z. Gimbutas, P. G. Martinsson, and V. Rokhlin. On the compression of low rank matrices. *SIAM J. Sci. Comput.*, 26(4):1389–1404 (electronic), 2005.
- [11] W. Chew and J. Song. Fast Fourier transform of sparse spatial data to sparse Fourier data. *Antennas and Propagation Society International Symposium, 2000. IEEE*, 4:2324–2327 vol.4, 2000.
- [12] D. L. Colton and R. Kress. *Integral equation methods in scattering theory*. Pure and Applied Mathematics (New York). John Wiley & Sons Inc., New York, 1983.
- [13] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19:297–301, 1965.
- [14] A. Dutt and V. Rokhlin. Fast Fourier transforms for nonequispaced data. *SIAM J. Sci. Comput.*, 14(6):1368–1393, 1993.
- [15] B. Engquist and L. Ying. Fast directional multilevel algorithms for oscillatory kernels. *SIAM Journal on Scientific Computing*, 29(4):1710–1737, 2007.
- [16] M. Fenn, S. Kunis, and D. Potts. Fast evaluation of trigonometric polynomials from hyperbolic crosses. *Numer. Algorithms*, 41(4):339–352, 2006.
- [17] V. Gradinaru. Fourier transform on sparse grids: code design and the time dependent Schrödinger equation. *Computing*, 80(1):1–22, 2007.
- [18] L. Greengard and J.-Y. Lee. Accelerating the nonuniform fast Fourier transform. *SIAM Rev.*, 46(3):443–454 (electronic), 2004.

- [19] M. Gu and S. C. Eisenstat. Efficient algorithms for computing a strong rank-revealing QR factorization. *SIAM J. Sci. Comput.*, 17(4):848–869, 1996.
- [20] J.-Y. Lee and L. Greengard. The type 3 nonuniform FFT and its applications. *J. Comput. Phys.*, 206(1):1–5, 2005.
- [21] G. F. Margrave and R. J. Ferguson. Wavefield extrapolation by nonstationary phase shift. *Geophysics*, 64(4):1067–1078, 1999.
- [22] E. Michielssen and A. Boag. A multilevel matrix decomposition algorithm for analyzing scattering from large structures. *IEEE Transactions on Antennas and Propagation*, 44(8):1086–1093, 1996.
- [23] M. O’Neil and V. Rokhlin. A new class of analysis-based fast transforms. Technical report, Yale University. YALE/DCS/TR1384, 2007.
- [24] D. Potts, G. Steidl, and M. Tasche. Fast Fourier transforms for nonequispaced data: a tutorial. In *Modern sampling theory*, Appl. Numer. Harmon. Anal., pages 247–270. Birkhäuser Boston, Boston, MA, 2001.
- [25] E. M. Stein. *Harmonic analysis: real-variable methods, orthogonality, and oscillatory integrals*, volume 43 of *Princeton Mathematical Series*. Princeton University Press, Princeton, NJ, 1993. With the assistance of Timothy S. Murphy, Monographs in Harmonic Analysis, III.
- [26] L. Ying and S. Fomel. Fast computation of partial Fourier transforms. Technical report, University of Texas at Austin, 2008.