

Fast Fourier Transform of Sparse Spatial Data to Sparse Fourier Data

W. C. Chew* and J.M. Song
 Center for Computational Electromagnetics and Electromagnetics Laboratory
 Department of Electrical and Computer Engineering
 University of Illinois, Urbana, IL 61801-2991

Dec 15, 1999

(*Dedicated to the memory of my mother who labored all her life for the next generation.)

1. Introduction

Nonuniform Fast Fourier transform on a line has been of interest to a number of scientists for its practical applications [1-3]. However, not much has been written on Fourier transforming sparse spatial data where the Fourier transform is needed at only sparse data points in the Fourier space in 2D or 3D. It finds applications in remote sensing, inverse problems, and synthetic aperture radar where the scattered field is related to the Fourier transform of the scatterers. In many instances, the scatterer is describable by a few scattering centers. Also, since the number of receivers is usually finite, the scattered field is only available at a few Fourier data points. Moreover, these scattering data may be collected over different frequencies. Hence, the Fourier data generated may be randomly scattered over the Fourier space.

In this paper, we outline an algorithm to perform this transform in $N \log N$ operations, where N is the number of spatial data available, and we assume that the number of Fourier data desired is also of $O(N)$. The algorithm described here is motivated by the multilevel fast multipole algorithm (MLFMA) [4,5], but is different from that described in [6]. In MLFMA, an embedded fast Fourier transform algorithm is inherent, where the spatial data is arbitrarily distributed, but the Fourier data is required on the Ewald sphere. In the present algorithm, the restriction of the Fourier data to an Ewald sphere is lifted, so that it can be arbitrarily distributed as well. The present algorithm can be easily generalized to 3D.

2. Statement of the Problem

Let us consider the 2D case. The statement of the problem is to calculate

$$F(k_x, k_y) = \int_{S_r} dS f(x, y) e^{ik_x x + ik_y y} \quad (1)$$

where $(k_x, k_y) \in S_k$. Let us assume that $f(x, y)$ is sufficiently smooth (bandlimited) so that (1) can be replaced by a quadrature rule with exponentially small error. Also, $F(k_x, k_y)$ is bandlimited, and hence, will be required only at a finite number of points on S_k . Therefore, a restatement of the problem is:

$$F(\mathbf{k}_i) = \sum_{j=1}^{N_r} \tilde{f}(\mathbf{r}_j) e^{i\mathbf{k}_i \cdot \mathbf{r}_j}, \quad \begin{array}{l} \mathbf{r}_j \in S_r, \\ j=1, \dots, N_r, \end{array} \quad \begin{array}{l} \mathbf{k}_i \in S_k \\ i=1, \dots, N_k \end{array} \quad (2)$$

where $\mathbf{k} = (k_x, k_y)$, $\mathbf{r} = (x, y)$, $\tilde{f}(\mathbf{r}_j) = w_j f(\mathbf{r}_j)$, where w_j is the weight that follows from a quadrature rule. Let us assume that data both in \mathbf{r} space and \mathbf{k} space are dense in 1D, and hence, sparse in 2D. We further assume sampling at the Nyquist rate, and that $N_k \sim O(N_r)$.

Definition: We define the data to be dense in αD if N/L^α is of $O(1)$ when L , a characteristic length of the box, increases. On the contrary if N/L^α is of $o(1)$ when L increases, we say that the data is sparse in αD .

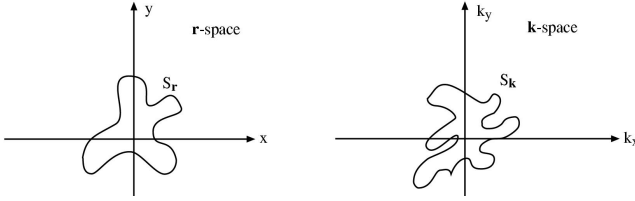


Fig. 1. Fourier transforming sparse spatial data (left) to sparse Fourier data (right).

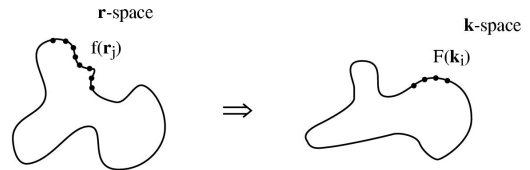


Fig. 2. Discrete version of the problem.

3. Algorithm I (Slow)

Let discuss a multilevel way of computing (2) that is not efficient, but will set the stage for a fast method.

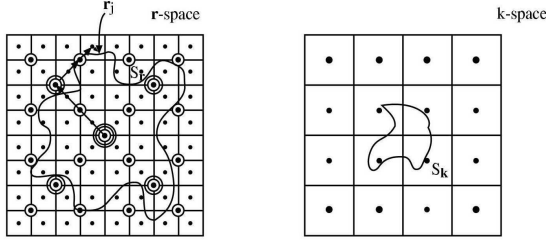


Fig. 3. Assigning boxes so that a tree structure can be defined.

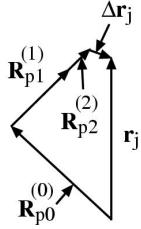


Fig. 4. Relationship between the vectors.

Step A: We immerse S_r in a box, and divide it into four smaller boxes recursively until the smallest box contains $O(1)$ number of \mathbf{r}_j points. All the boxes at different levels can be organized into a quad-tree structure. The centers of the boxes at different levels are denoted by circles. The more circles the dots have, the higher the level of boxes whose centers they represent.

Step B: Notice that

$$\mathbf{r}_j = \mathbf{R}_{p_0}^{(0)} + \mathbf{R}_{p_1}^{(1)} + \mathbf{R}_{p_2}^{(2)} + \dots + \mathbf{R}_{p_{L-1}}^{(L-1)} + \Delta \mathbf{r}_j \quad (3)$$

where $\mathbf{R}_p^{(i)}$, $p=1, \dots, 4$ are the vectors from the center of the box at level i to level $i+1$. Here, $i=0$ corresponds to the coarsest or the root level while $i=L$ corresponds to the finest level. $\Delta \mathbf{r}_j$ is the last vector that points from the center of the finest box to \mathbf{r}_j that is also closest to \mathbf{r}_j .

Step C: Using (3) in (2), we can sum (2) in a hierarchical or multilevel fashion. First the summation is taken over the finest boxes, and then over the next finest level, and so on.

$$F(\mathbf{k}_i) = \sum_{\substack{p_0=1 \\ B_{i_0}^{(1)} \in B_{i_0}^{(0)}}}^4 e^{i\mathbf{k}_i \cdot \mathbf{R}_{p_0}^{(0)}} \dots \sum_{\substack{p_{L-1}=1 \\ B_{i_{L-1}}^{(L)} \in B_{i_{L-1}}^{(L-1)}}}^4 e^{i\mathbf{k}_i \cdot \mathbf{R}_{p_{L-1}}^{(L-1)}} \left(\sum_{\mathbf{r}_j \in B_{i_L}^{(L)}} \tilde{f}(\mathbf{r}_j) e^{i\mathbf{k}_i \cdot \Delta \mathbf{r}_j} \right), \quad \mathbf{k}_i \in S_{\mathbf{k}} \quad (4)$$

where $B_{i_L}^{(L)}$ denotes the i_L -th box at the L -th level. The second sum is for all $B_{i_L}^{(L)} \in B_{i_{L-1}}^{(L-1)}$ and likewise for the other sums similarly denoted.

4. More Details and Complexity Analysis of the Slow Algorithm

The above is also a nested way of calculating (2), but it is not fast. To understand it more fully, let us describe the calculation in more details:

Step 0: Calculate

$$f_{i_L}(\mathbf{k}_i) = \sum_{\mathbf{r}_j \in B_{i_L}^{(L)}} \tilde{f}(\mathbf{r}_j) e^{i\mathbf{k}_i \cdot \Delta \mathbf{r}_j}, \quad \mathbf{k}_i \in S_{\mathbf{k}}, i=1, \dots, N_k \quad (5)$$

$$i_L = 1, \dots, N_B^{(L)}$$

where $N_B^{(L)}$ is the number of nonempty boxes at level L . Hence, the above calculation is repeated for each box which is nonempty. Since each box contains $O(1)$ points, the number of nonempty boxes, $N_B^{(L)} \sim N_r$. The calculation is repeated for $\mathbf{k}_i \in S_{\mathbf{k}}, i=1, \dots, N_k$. Therefore, the workload $W_L \sim C_1 N_r N_k$.

Step 1: Calculate

$$f_{i_{L-1}}(\mathbf{k}_i) = \sum_{\substack{p_{L-1}=1 \\ B_{i_{L-1}}^{(L)} \in B_{i_{L-1}}^{(L-1)}}}^4 e^{i\mathbf{k}_i \cdot \mathbf{R}_{p_{L-1}}^{(L-1)}} f_{i_L}(\mathbf{k}_i), \quad \mathbf{k}_i \in S_{\mathbf{k}}, i=1, \dots, N_k \quad (6)$$

The above calculation is repeated for each box at level $L-1$ that is nonempty. The number of nonempty boxes at level $L-1$ is $N_B^{(L-1)} \approx \frac{1}{2} N_B^{(L)}$. So calculating (6) approximately involves workload $W_{L-1} \sim C_1 N_r N_k / 2$.

Step 2: Repeat

$$f_{l_{j-1}}(\mathbf{k}_i) = \sum_{\substack{p_{j-1}=1 \\ B_j^{(j)} \in B_{j-1}^{(j-1)}}}^4 e^{i\mathbf{k}_i \cdot \mathbf{R}_{p_{j-1}}^{(j-1)}} f_{l_j}(\mathbf{k}_i), \quad \mathbf{k}_i \in S_{\mathbf{k}}, i=1, \dots, N_k \quad (7)$$

until $j-1=0$. The cost of performing the whole work is then

$$W \sim C_1 N_r N_k (1+1/2+1/4+\dots) \sim 2C_1 N_r N_k. \quad (8)$$

5. Algorithm II (Fast)

The above algorithm is not fast, but it can be sped up as follows:

Step a: Notice that $\Delta \mathbf{r}_j$ in (5) is small, and hence, $f_{l_j}(\mathbf{k}_i)$ in (5) has a small bandwidth in the \mathbf{k} -space.

Therefore (5) need not be computed for all points on $S_{\mathbf{k}}$.

Step b: We enclose $S_{\mathbf{k}}$ inside a smallest square box possible with side D_k , and then divide the box into four sub-boxes. Then 12 boxes are added to “buffer” the original four boxes as shown. There are a total of 16 boxes (see Fig. 3b). More buffer boxes can be added for higher interpolation accuracy to be described in Step e. We also check to see if the size of the boxes in the \mathbf{k} -space satisfy the Nyquist sampling criterion, i.e., $D_k \approx \pi / |\Delta \mathbf{r}_j|_{\max}$.

Step c: Equation (5) is computed with \mathbf{k} at the centers of these 16 boxes. Then each sub-box in the figure is further divided into four sub-boxes.

Step d: Sub-boxes that are further away from $S_{\mathbf{k}}$ than a buffer box is ignored. The boxes that are important have their centers denoted by crosses in Fig. 5.

Step e: The Fourier data of $f_{l_j}(\mathbf{k})$ on the crosses can be interpolated to exponential accuracy from the Fourier data on coarser grid, i.e. the dots, because $f_{l_j}(\mathbf{k})$ is a smooth or bandlimited function [7].

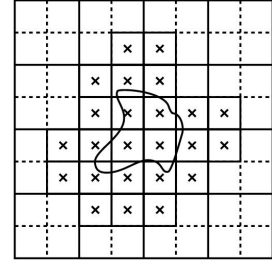


Fig. 5. The Fourier data available in the Fourier space.

Step f: Then, Equation (6) is invoked, but with \mathbf{k}_i belonging to the set shown above.

Step g: Next, further sub-divide the boxes, ignore boxes far away from $S_{\mathbf{k}}$, and interpolate from the coarser grid to the finer grid. Repeat the calculation of (7) on the subset of points in $S_{\mathbf{k}}$. Repeat Step g, until the whole of (4) is completed.

6. Complexity Analysis of the Fast Algorithm

Denoting the number of Fourier data in \mathbf{k} -space to be computed at level l to be $\tilde{N}_k^{(l)}$, then $\tilde{N}_k^{(L)} = 16$, $\tilde{N}_k^{(L-1)} = 24$, in the above example. It can be shown easily that $\tilde{N}_k^{(0)} \sim N_k$, $\tilde{N}_k^{(1)} \sim \frac{1}{2}N_k$, $\tilde{N}_k^{(2)} \sim \frac{1}{4}N_k$, when $N_k \rightarrow \infty$, because the Fourier data lives on a line. Every time when the size of the boxes is doubled, the number of nonempty boxes decreases by a factor of two.

- i) With this interpolation, the workload in Step 0 at level L to compute Eqn (5) is now reduced to about $16C_1 N_r \ll C_1 N_r N_k$ if $16 \ll N_k$.
- ii) The workload in Step 2 to compute Eqn. (6) at level $L-1$ is now reduced to about $C_1 \frac{N_r}{2} 24$ for the above example.
- iii) The workload at level $L-l$ is about $C_1 \frac{N_r}{2^l} \tilde{N}_k^{L-l}$.

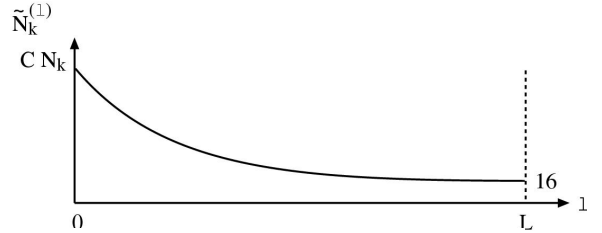


Fig. 6. The $\tilde{N}_k^{(l)}$ function versus l .

- iv) The workload at the last stage in Step 2 is about $C N_k$, because at the last stage the number of nonempty boxes in the Fourier (\mathbf{k}) space is proportional to N_k .

v) Therefore, the total workload now for all level ($\log_2 N_r$ levels in total) is

$$W = C_1 \sum_{l=0}^{\log_2 N_r} \left(\frac{N_r}{2^l} \right) \tilde{N}_k^{L-l} \sim C(N_r + N_k) \log_2 N_r \ll C_1 N_r N_k \quad (9)$$

vi) Note that the cost of interpolation is also proportional to $\frac{N_r}{2^l} \tilde{N}_k^{L-l}$ at each level. Therefore, the complexity remains the same with the interpolation work included.

vii) If $N_r \sim N_k \sim O(N)$, this is essentially an $N \log N$ algorithm.

7. Miscellaneous Details

The above analysis is based on Nyquist sampling rate both in \mathbf{r} - and \mathbf{k} -space. The same algorithm with some modifications for more efficiency can be used for some special cases where the conditions are different:

- Case a – Denser sampling rate in \mathbf{k} -space: We use (7) to calculate the values at the center of the boxes at the finest level in \mathbf{k} -space, then interpolate them to denser points thereafter.
- Case b – Sparser sampling rate in \mathbf{k} -space: We stop at a coarser level in \mathbf{r} -space (before the finest level in \mathbf{k} -space), interpolate and then perform a direction summation to calculate the \mathbf{k} -space data using interpolation.
- Case c – $D_k \ll \pi / |\Delta \mathbf{r}_j|_{\max}$ where D_k is defined in Step b above: We use larger boxes at the finest level in \mathbf{r} -space, or antepolate (transpose interpolate) the \mathbf{r} -space data to a coarser grid.
- Case d – $D_k \gg \pi / |\Delta \mathbf{r}_j|_{\max}$: We use smaller box at the finest level in \mathbf{r} -space or start with more boxes at the coarsest level in \mathbf{k} -space (more than 16 boxes are needed) to cover the desired \mathbf{k} -space.
- Case e – If the desired Fourier data is centered away from the origin, we can always redefine the transform (1) so that the desired data is centered about the origin, and apply the above algorithm.

8. Conclusions

We have outlined an $O(N \log N)$ algorithm to Fourier transform sparse spatial data to sparse Fourier data. We assume that the data is dense in 1D both in spatial and Fourier space, but live in a 2D space. The algorithm can be easily generalized to higher dimensions. For nD problems, it can be shown that the complexity remains $O(N \log N)$ when both the spatial and Fourier data are dense in αD where $0 < \alpha < n$. When the data in one space is denser than the other space, then the algorithm is essentially $O(N)$ where N is the larger of N_r or N_k .

Acknowledgement: This work is supported by the AFOSR under a MURI grant F49620-96-1-0025, DARPA grant F49620-98-1-0498 and National Science Foundation under grant NSF ECS 99-0051.

References:

1. A. Dutt and V. Rokhlin, “Fast Fourier transforms for nonequispaced data,” *SIAM J. Sci. Comp.*, vol. 14, pp. 1368-1393, 1993.
2. G. Beylkin, “On the fast Fourier transform of functions with singularities,” *Appl. Computat. Harmonic Anal.*, vol. 2, pp. 363-382, 1995.
3. Q.H. Liu and N. Nguyen, “An accurate algorithm for nonuniform Fast Fourier Transform (NUFFT),” *Micro. Guided Wave Lett.*, vol. 8, no. 1, pp. 18-20, 1998.
4. C.C. Lu and W.C. Chew, “A multilevel algorithm for solving boundary integral equation of scattering,” *Micro. Opt. Tech. Lett.*, vol. 7, no. 10, pp.466-470, July 1994.
5. J.M. Song and W.C. Chew, “Multilevel fast multipole algorithm for solving combined field integral equation of electromagnetic scattering,” *Mico. Opt. Tech. Lett.*, vol. 10, no. 1, pp 14-19, September 1995.
6. A. Brandt, “Multilevel computations of integral transforms and particle interactions with oscillatory kernels,” *Comput. Phys. Comm.*, vol. 65, pp. 24-38, 1991.
7. O.M. Bucci, C. Gennareli, and C. Savarese, “Optimal interpolation of radiated fields over a sphere,” *IEEE Trans. Ant. Propag.*, vol. 39, no. 11, pp. 1633-1643, Nov. 1991.