SIAM REVIEW
Vol. 46, No. 3, pp. 443–454

Accelerating the Nonuniform Fast Fourier Transform*

Leslie Greengard†
June-Yub Lee‡

Abstract. The nonequispaced Fourier transform arises in a variety of application areas, from medical imaging to radio astronomy to the numerical solution of partial differential equations. In a typical problem, one is given an irregular sampling of N data in the frequency domain and one is interested in reconstructing the corresponding function in the physical domain. When the sampling is uniform, the fast Fourier transform (FFT) allows this calculation to be computed in $O(N \log N)$ operations rather than $O(N^2)$ operations. Unfortunately, when the sampling is nonuniform, the FFT does not apply. Over the last few years, a number of algorithms have been developed to overcome this limitation and are often referred to as *nonuniform FFTs* (NUFFTs). These rely on a mixture of interpolation and the judicious use of the FFT on an oversampled grid [A. Dutt and V. Rokhlin, *SIAM J. Sci. Comput.*, 14 (1993), pp. 1368–1383].

In this paper, we observe that one of the standard interpolation or "gridding" schemes, based on Gaussians, can be accelerated by a significant factor without precomputation and storage of the interpolation weights. This is of particular value in two- and three-dimensional settings, saving either $10^d N$ in storage in d dimensions or a factor of about 5–10 in CPU time (independent of dimension).

Key words. nonuniform fast Fourier transform, fast gridding, FFT, image reconstruction

AMS subject classifications. 42A38, 44A35, 65T50, 65R10

DOI. 10.1137/S003614450343200X

1. Introduction.

In this note, we describe an extremely simple and efficient implementation of the nonuniform fast Fourier transform (NUFFT). There are a host of applications of such algorithms, and we refer the reader to the references [2, 6, 8, 11, 13, 14, 17] for examples. We restrict our attention here to one: function (or image) reconstruction from Fourier data as discussed in [6, 8, 11, 14]. Let us begin, however, with a more precise description of the computational task. In two dimensions, we define the nonuniform discrete Fourier transform of types 1 and 2 according to the formulae

$$F(k_1, k_2) = \frac{1}{N} \sum_{j=0}^{N-1} f_j \, e^{-i(k_1,k_2)\cdot \mathbf{x}_j} \ , \tag{1}$$

*Received by the editors July 23, 2003; accepted for publication (in revised form) December 1, 2003; published electronically July 30, 2004.

http://www.siam.org/journals/sirev/46-3/43200.html

†Courant Institute of Mathematical Sciences, New York University, New York, NY 10012 (greengard@cims.nyu.edu). The work of this author was supported by the Applied Mathematical Sciences Program of the U.S. Department of Energy under contract DEFGO288ER25053.

‡Department of Mathematics, Ewha Womans University, Seoul, 120-750, Korea (jylee@math.ewha.ac.kr). The work of this author was supported by the Korea Research Foundation under grant 2002-015-CP0044.

$$f(\mathbf{x}_j) = \sum_{k_1} \sum_{k_2} F(k_1, k_2)\, e^{i(k_1,k_2)\cdot \mathbf{x}_j}, \tag{2}$$

respectively, where $\mathbf{x}_j \in [0, 2\pi] \times [0, 2\pi]$ and $-\frac{M}{2} \le k_1, k_2 < \frac{M}{2}$.

It is, perhaps, convenient to think of (1) as a discretization of the Fourier integral

$$F(k_1, k_2) = \frac{1}{(2\pi)^2} \int_0^{2\pi} \int_0^{2\pi} f(\mathbf{x})\, e^{-i(k_1,k_2)\cdot \mathbf{x}}\, d\mathbf{x} \tag{3}$$

with $\{\mathbf{x}_j\}$ serving as the discretization points. If we let w_j denote the quadrature weight corresponding to $\{\mathbf{x}_j\}$, then we obtain (1) by setting $f_j = f(\mathbf{x}_j)\, w_j$. Equation (2), of course, is simply the evaluation of a finite Fourier series

$$f(\mathbf{x}) = \sum_{k_1} \sum_{k_2} F(k_1, k_2)\, e^{i(k_1,k_2)\cdot \mathbf{x}} \tag{4}$$

at an arbitrary set of targets.

Nonuniform FFTs of the types discussed here are based, in essence, on combining some interpolation scheme with the standard FFT. Oddly enough, it was a number of years after their use in applications before a rigorous analysis of such schemes was introduced by Dutt and Rokhlin [5]. Subsequent papers, such as [1, 3, 9, 10], described variants based on alternative interpolation/approximation approaches.

Before discussing the algorithm itself, we would like to comment briefly on applications that involve evaluation of (3) or the Fourier integral

$$H(s_1, s_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(\mathbf{x})\, e^{-i(s_1,s_2)\cdot \mathbf{x}}\, d\mathbf{x} \tag{5}$$

when the transform data $h(\mathbf{x})$ is known at a scatter of points rather than on a regular Cartesian mesh. There is some confusion in the literature about the use of NUFFTs in this context (see Remark 1 below). There are three separate issues involved: acquisition of data $h(\mathbf{x}_j)$ in the Fourier domain, the choice of a quadrature scheme $\{\mathbf{x}_j, w_j\}$, and the availability of a fast algorithm for computing the discrete transform itself. They are often blended together when they should not be. Dutt and Rokhlin [5] appear to have been the first to try to isolate one of these problems; they addressed the algorithmic question and showed that sums of the form (1) or (2) can be computed in $O(N \log N)$ time with complete control of precision. While they concentrated on the one-dimensional case, higher dimensional versions have been considered by a variety of authors [3, 6, 14]. The first rigorous two-dimensional version can be found in a paper by Strain [15], which uses the NUFFT to solve a class of elliptic partial differential equations.

REMARK 1. *Not all schemes for reconstructing Fourier integrals of the type* (3) *can be represented formally as a quadrature of the type* (1). *Different schemes for interpolating* $f(\mathbf{x}_j)$ *to a uniform mesh* do *give rise to different reconstructed functions* F. *However, if the decision has been made to use a quadrature approach such as* (1), *then the remaining task is entirely computational. The best algorithm is the one that evaluates the relevant sums as quickly and as accurately as possible.*

2. The NUFFT. Let us first consider the one-dimensional analogs of the summation problems (1) and (2). With $x_j \in [0, 2\pi]$, the type-1 NUFFT is defined by the

calculation of

$$F(k) = \frac{1}{N} \sum_{j=0}^{N-1} f_j \, e^{-ikx_j} \quad \text{for} \quad k = -\frac{M}{2}, \ldots, \frac{M}{2} - 1 \,. \tag{6}$$

It is based on the following set of observations:

1. Equation (6) describes the exact Fourier coefficients of the function

$$f(x) = \sum_{j=0}^{N-1} f_j \delta(x - x_j), \tag{7}$$

viewed as a periodic function on $[0, 2\pi]$. Here, $\delta(x)$ denotes the Dirac delta function. It is clearly not well-resolved by a uniform mesh in x.

2. Let $g_\tau(x)$ denote the one-dimensional periodic heat kernel on $[0, 2\pi]$, given by

$$g_\tau(x) = \sum_{l=-\infty}^{\infty} e^{-(x-2l\pi)^2/4\tau}.$$

If we define $f_\tau(x)$ to be the convolution

$$f_\tau(x) = f * g_\tau(x) = \int_0^{2\pi} f(y) g_\tau(x - y) \, dy, \tag{8}$$

then f_τ is a 2π-periodic C^∞ function and can be well-resolved by a uniform mesh in x whose spacing is determined by τ (Figure 1). The Fourier coefficients of f_τ, namely,

$$F_\tau(k) = \frac{1}{2\pi} \int_0^{2\pi} f_\tau(x) e^{-ikx} \, dx \,,$$

can be computed with high accuracy using the standard FFT on an oversampled grid

$$F_\tau(k) \approx \frac{1}{M_r} \sum_{m=0}^{M_r - 1} f_\tau(2\pi m / M_r) e^{-ik2\pi m/M_r}, \tag{9}$$

where

$$f_\tau(2\pi m / M_r) = \sum_{j=0}^{N-1} f_j g_\tau(2\pi m / M_r - x_j). \tag{10}$$

3. Once the values $F_\tau(k)$ are known, an elementary calculation shows that

$$F(k) = \sqrt{\frac{\pi}{\tau}} \, e^{k^2 \tau} F_\tau(k). \tag{11}$$

(This is a direct consequence of the convolution theorem and the fact that the Fourier transform of g_τ is $G_\tau(k) = \sqrt{2\tau} \, e^{-k^2\tau}$.)

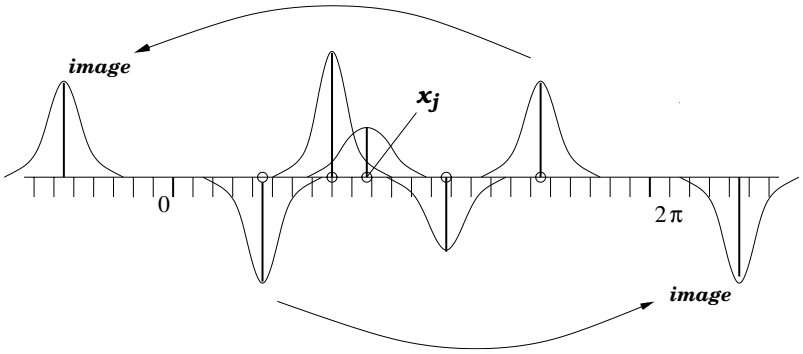

Fig. 1 *In the version of the NUFFT described here, each delta function source at a point such as* $\mathbf{x}_j$ *in* (7) *is replaced by a Gaussian. This smears the source strength to nearby regular grid points. The regular grid must be fine enough to resolve the smeared function* f_τ *in* (8). *Note that we include* 2π*-periodic images of the sources in the definition of the heat kernel* g_τ. *They decay sufficiently rapidly that all but the nearest ones can be ignored.*

Recall that a type-2 transformation evaluates a regular Fourier series at irregular target points. Thus, in one dimension, for $x_j \in [0, 2\pi]$, the type-2 NUFFT is defined by the calculation of

$$f(x_j) = \sum_{k=-\frac{M}{2}}^{\frac{M}{2}-1} F(k)\, e^{ikx_j}\,. \tag{12}$$

It is based on a closely related idea:

1. We first deconvolve the Fourier coefficients, defining $F_{-\tau}(k)$ by

$$F_{-\tau}(k) = \sqrt{\frac{\pi}{\tau}}\, e^{k^2\tau}\, F(k), \tag{13}$$

and evaluate the corresponding function $f_{-\tau}(x)$ on a uniform mesh with M_r points on $[0, 2\pi]$ using the FFT

$$f_{-\tau}(x) = \sum_{k=0}^{M_r-1} F_{-\tau}(k)\, e^{ikx}. \tag{14}$$

In the preceding expression, we set $F_{-\tau}(k) = 0$ for $\frac{M}{2} \le k < M_r - \frac{M}{2}$ and view $F_{-\tau}(k)$ as an M_r-periodic function $F_{-\tau}(k) = F_{-\tau}(k - M_r)$.

2. We then compute the desired values $f(x_k)$ from

$$\begin{aligned} f(x_j) = f_{-\tau} * g_\tau(x_j) &= \frac{1}{2\pi}\int_0^{2\pi} f_{-\tau}(x) g_\tau(x_j - x)\, dx \\ &\approx \frac{1}{M_r}\sum_{m=0}^{M_r-1} f_{-\tau}(2\pi m/M_r)\, g_\tau(x_j - 2\pi m/M_r). \end{aligned} \tag{15}$$

This is again a direct consequence of the convolution theorem except that we deconvolve the effect of Gaussian smoothing in (13) before we actually carry out the smoothing in (15)!

REMARK 2. *There are a number of details that need to be fixed here, including the selection of* τ*, the convolution with a Gaussian in both procedures, and the length* M_r *of the FFTs used. We will not repeat the analysis of* [5]*, since the relevant results can be summarized very simply: with* $M_r = 2M$ *and* $\tau = 12/M^2$*, Gaussian spreading of each source to the nearest* 24 *grid points yields about* 12 *digits of accuracy. With* $\tau = 6/M^2$*, Gaussian spreading of each source to the nearest* 12 *grid points yields about* 6 *digits of accuracy.*

3. Fast Gaussian Gridding. The dominant task in the NUFFT is the calculation of $f_\tau(2\pi m/M_r)$ in (10) and $f(x_j)$ in (15). Following standard practice, we will refer to these processes as *gridding* and the M_r-point mesh as the *oversampled* mesh. In d dimensions, gridding requires $12^d N$ exponential evaluations for single precision accuracy and about $24^d N$ exponential evaluations for double precision accuracy. In order to avoid that cost using existing schemes, one can precompute all the necessary quantities, incurring a storage cost of $12^d N$ or $24^d N$ and a computational cost of $12^d N$ or $24^d N$ multiplications.

This cost (in either storage or CPU time or both) becomes a significant burden in two, three, and higher dimensions. It is sometimes called the curse of dimensionality; in the absence of a separable coordinate system, interpolation-type processes have costs that grow exponentially with dimension. In the remainder of this paper, we don't overcome the curse, but we show that $(1 + d)N$ exponential evaluations or $(1+d)N$ storage is sufficient, followed by $(12d+12^d)N$ or $(24d+24^d)N$ multiplications, depending on the required accuracy. The net reduction in CPU time is by a factor of 5–10.

By inspection of (10), it is evident that we only need values of the function f_τ at equispaced points on the oversampled mesh. For this, we have

$$f_\tau(2\pi m/M_r) = \sum_{j=0}^{N-1} f_j \sum_{l=-\infty}^{\infty} e^{-(x_j - 2\pi m/M_r - 2l\pi)^2/4\tau}.$$

This expression for f_τ looks much more expensive than it actually is. Since the Gaussian sources are sharply peaked (in a manner dependent on τ), each source of strength f_j located at x_j is nonnegligible only at nearby grid points. As mentioned above, we only need to compute its effect at the nearest 12 or 24 points (Figure 1) to achieve either 6- or 12-digit accuracy. Thus, for the purpose of computation, we change our point of view from the receiving point $(2\pi m/M_r)$ to the source point x_j and consider one Gaussian source at a time. An elementary calculation shows that

$$e^{-(x_j - 2\pi m/M_r)^2/4\tau} = e^{-x_j^2/4\tau} \left(e^{x_j \pi/M_r \tau}\right)^m e^{-(\pi m/M_r)^2/\tau}. \tag{16}$$

But this means that we can compute and store two exponentials, $e^{-x_j^2/4\tau}$ and $e^{x_j\pi/M_r\tau}$, for each source point. The third exponential, $e^{-(\pi m/M_r)^2/\tau}$, is independent of x_j.

The gridding algorithm is straightforward:

- Let $\xi = 2\pi m/M_r$ denote the nearest regular grid point that is less than or equal to x_j on the oversampled grid. Beginning at ξ, let M_{sp} denote the number of grid points to which the spreading will be accounted for in each direction.
- Compute the two exponentials: $E_1 = e^{-(x_j-\xi)^2/4\tau}$, $E_2 = e^{(x_j-\xi)\pi/M_r\tau}$.
- The spreading contribution to $f_\tau(2\pi(m+m')/M_r)$ for $-M_{sp} < m' \le M_{sp}$ is $f_j\, E_1 \cdot E_2^{m'} \cdot E_3(m')$, where $M_{sp} = 6$ for single precision, $M_{sp} = 12$ for double precision, and $E_3(m') = e^{-(\pi m'/M_r)^2/\tau}$.

Careful organization of the loop shows that, for each source point, two exponential evaluations are required, followed by two multiplications at each of $2M_{sp}$ regular mesh points. The algorithm for (15) is similar.

A nice feature of Gaussian spreading is that the heat kernel is built as a tensor product:

$$\begin{aligned} &e^{-(x_j-2\pi m/M_r)^2/4\tau} \cdot e^{-(y_j-2\pi n/M_r)^2/4\tau} \\ &= e^{-(x_j+y_j)^2/4\tau} \left(e^{x_j\pi/M_r\tau}\right)^m \left(e^{y_j\pi/M_r\tau}\right)^n e^{-(\pi m/M_r)^2/\tau}\, e^{-(\pi n/M_r)^2/\tau}. \end{aligned}$$

Thus, one can carry out spreading one dimension at a time.

There follows an informal description of the implementation of the type-1 fast gridding algorithm in two dimensions,

$$F(k_1,k_2) = \frac{1}{N}\sum_{j=0}^{N-1} f_j\, e^{-i(k_1,k_2)\cdot \mathbf{x}_j},$$

for $-\frac{M}{2} \le k_1, k_2 < \frac{M}{2}$ and $\mathbf{x}_j \in [0,2\pi]\times[0,2\pi]$.

FAST GRIDDING ALGORITHM OF TYPE 1 IN TWO DIMENSIONS.

STEP I: INITIALIZATION

1. Set the oversampling ratio $R = M_r/M$, the spreading parameter M_{sp}, and the Gaussian kernel parameter τ according to the desired precision ϵ.
2. Precompute $E_3(l) = e^{-(\pi l/M_r)^2/\tau}$ for $0 \le l \le M_{sp}$ and $E_4(k) = E_4(M-k) = e^{\tau k^2}$ for $|k| \le \frac{M}{2}$.

STEP C: CONVOLUTION FOR EACH SOURCE POINT (x_j, y_j)

1. Find the nearest grid point $(\xi_1,\xi_2) = \frac{2\pi}{M_r}(m_1,m_2)$ with $\xi_1 \le x_j$, $\xi_2 \le y_j$.
2. Compute $E_1 = e^{-((x_j-\xi_1)^2+(y_j-\xi_2)^2)/4\tau}$, $E_{2x} = e^{\pi(x_j-\xi_1)/M_r\tau}$, $E_{2y} = e^{\pi(y_j-\xi_2)/M_r\tau}$ and $E_{2x}(l_1) = {E_{2x}}^{l_1}$, $E_{2y}(l_2) = {E_{2y}}^{l_2}$ for $-M_{sp} < l_1, l_2 \le M_{sp}$.
3. Convolve the Gaussian spreading function with f_j as follows:
   $V_0 = f_j \cdot E_1$
   for $l_2 = -M_{sp}{+}1,\ M_{sp}$
   $V_y = V_0 \cdot E_{2y}(l_2)$
   for $l_1 = -M_{sp}{+}1,\ M_{sp}$
   Add $V_y \cdot E_{2x}(l_1)$ to $f_\tau(m_1+l_1, m_2+l_2)$.

STEP D: FFT AND DECONVOLUTION

1. Compute two-dimensional FFT of $f_\tau(m_1,m_2)$ to obtain $F_\tau(k_1,k_2)$.
2. Set $F(k_1,k_2) = \sqrt{\frac{\pi}{\tau}} E_4(k_1)E_4(k_2)F_\tau(k_1,k_2)$ for $-\frac{M}{2} \le k_1, k_2 < \frac{M}{2}$.

The total cost is that of the oversampled two-dimensional FFT (Step D), $(d{+}1)$ exponential evaluations (Step C2), $(d{\cdot}2M_{sp})$ multiplications (Step C2), and $(2M_{sp})^d$ multiplications (Step C3) per source point.

4. Numerical Examples. The NUFFT of types 1 and 2 have been implemented (in Fortran) with fast gridding in one and two dimensions. The following is a selection of examples illustrating their performance.

Example 1 (Verification of Accuracy). In order to check the accuracy of the gridding algorithm numerically, we compare the results of the type-1 and -2 transforms

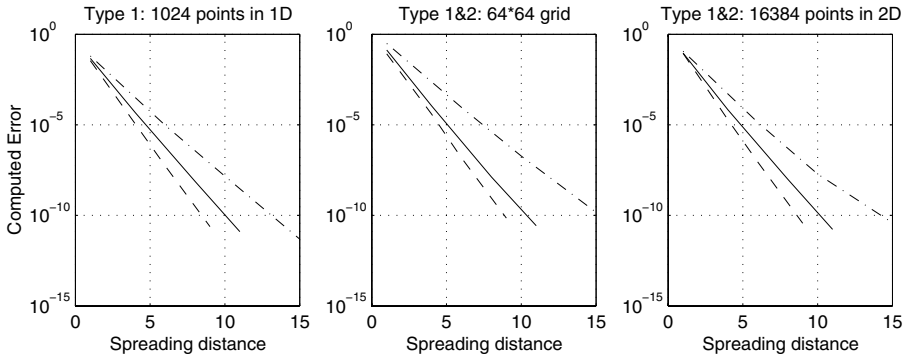

Fig. 2 Gridding error in the NUFFT. *In each figure, the x-axis indicates the spreading distance defined by parameter M_{sp} and the y-axis indicates the computed error. The leftmost figure corresponds to a one-dimensional example with* 1024 *points. The middle example uses a uniform set of data points in two dimensions and the rightmost figure uses a random distribution of data points in two dimensions. Dash-dotted lines show the error for an oversampled grid with $M_r = 1.5\,M$, solid lines for $M_r = 2\,M$, and dashed lines for $M_r = 3\,M$.*

Table I

M_{sp}	$R = 1.5$	$R = 2$	$R = 2.5$	$R = 3$	$R = 3.5$	$R = 4$
3	9.0E-3	1.9E-3	8.5E-4	5.3E-4	3.8E-4	3.1E-4
6	8.1E-5	3.5E-6	7.2E-7	2.8E-7	1.5E-7	1.0E-7
9	7.2E-7	6.5E-9	6.2E-10	1.5E-10	5.8E-11	3.0E-11
12	6.5E-9	1.2E-11	5.5E-13	8.0E-14	2.3E-14	9.2E-15

(6), (12) with uniformly distributed random data points using direct summation and the NUFFT. Figure 2 shows the errors in the l_2-norm.

In the fast gridding algorithm, the accuracy of the type-1 transformation (6) is controlled by three parameters: the Gaussian kernel parameter τ, the oversampling ratio $R = M_r/M$, and the spreading distance M_{sp}. We set τ according to the formula

$$\tau = \frac{1}{M^2}\,\frac{\pi}{R(R-0.5)}\,M_{sp}\,.$$

As indicated earlier, we refer to the paper [5] for a detailed analysis. Here, we present some numerical experiments verifying the precision estimates derived there (see Table 1).

Example 2 (Fast Gridding Compared to Gridding). The naive Gaussian gridding algorithm and the fast gridding method for (1), (2), (6), and (12) have been tested in various computing environments, including Solaris 2.7 with gcc-2.95 on a 450MHz Ultra-60 with 768MB RAM, cygwin-5.1 with gcc-3.2 on a 1GHz Pentium-3 with 384MB RAM, and cygwin-5.1 with gcc-3.2 on a 2.4GHz Pentium-4 with 1GB RAM. Figure 3 summarizes the computational costs for $M_r = 2M$, $M_{sp} = 6$ yielding six digits of accuracy.

Both algorithms use the standard FFT on the oversampled mesh, and the time for this step is indicated in Figure 3 by dotted lines. The actual computation time depends on a number of factors, including compiler options, type of CPU, performance

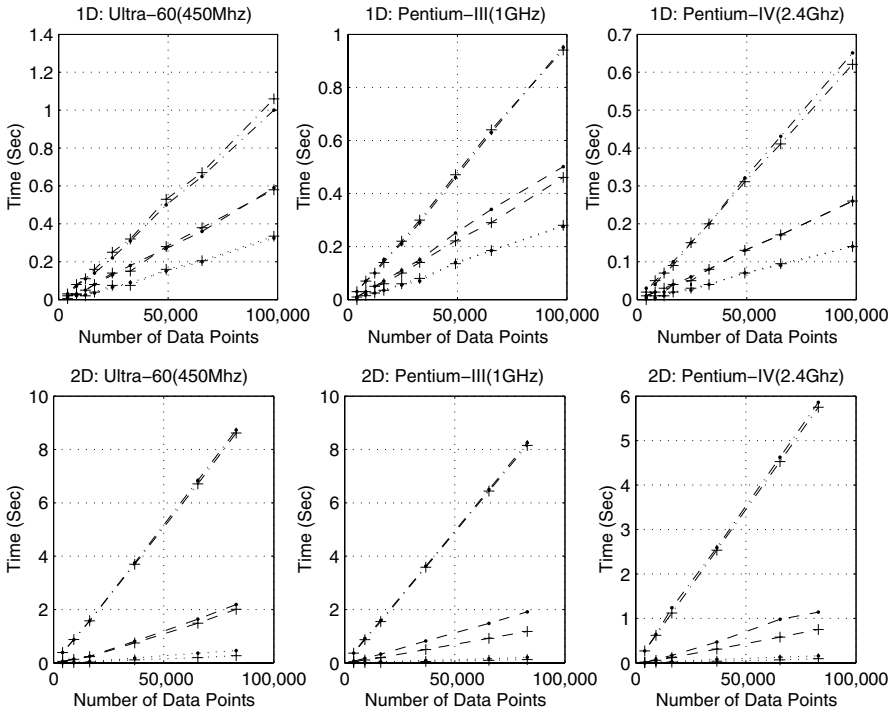

Fig. 3 *Computing time. Dotted lines represent the cost for FFT, dashed lines for fast gridding, and dash-dotted lines for naive gridding. Heavy dots (·) are used for type-*1 *and plus marks (+) for type-*2 *transformation.*

of the math coprocessor, cache size, etc. Note that the type-1 and type-2 transforms are very similar in terms of floating point operations; the differences in CPU time are due mainly to memory caching issues. In any case, the speed-up of the fast gridding algorithm is significant in two dimensions and would be even more significant in the three-dimensional case. We used an optimized version of the FFT and compiled all codes with the O2 optimization flag, but we did not carry out a complete optimization of the fast gridding algorithm itself. More detailed (but less portable) implementation work could probably yield a further factor of 5–10 in performance. We have not carried out such fine-tuning.

Example 3 (Comparison with Direct Method). In this example, we compare the computational performance of our fast gridding algorithm with direct summation and the standard FFT. The direct code was implemented and compiled with the same options: the gcc-2.95 compiler with -O2 optimization on a 450MHz Sparc Ultra-60. Figure 4 shows our results. We set $M_r = 2M$, and set the number of data points $N = M$ in one dimension and $N = M^2$ in two dimensions.

Using the data of Figure 4, we can summarize the performance of the method in one and two dimensions as follows. First, direct summation requires about $0.022\ N^2$ (μsec) in one dimension and $0.087\ N^2$ (μsec) in two dimensions. In one dimension, the oversampled FFT requires about $2.4\ M_r$ μsec (using a linear fit of the data over the range of M_r tested). Note that this is already twice as expensive as an M-point

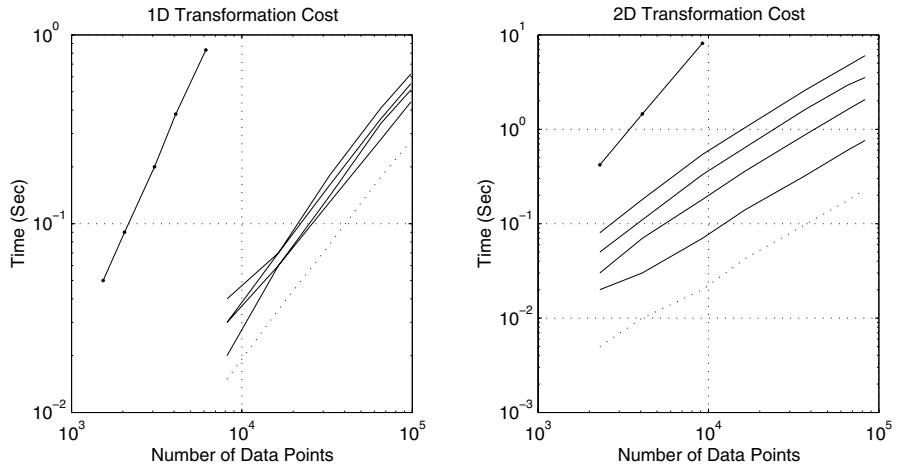

Fig. 4 *CPU requirements of the direct code (solid lines with dots), the fast gridding code (four solid lines with tolerance $\epsilon = 10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}$ from bottom to top), and the standard FFT for the oversampled mesh with $M_r = 2M$ (dotted lines).*

Table 2

1D	$\epsilon = 10^{-3}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-9}$	$\epsilon = 10^{-12}$
Time (μsec)	3.77 N	4.40 N	4.78 N	5.44 N
Break even (N)	170	200	220	250
2D	$\epsilon = 10^{-3}$	$\epsilon = 10^{-6}$	$\epsilon = 10^{-9}$	$\epsilon = 10^{-12}$
Time (μsec)	8.45 N	20.67 N	36.35 N	59.30 N
Break even ($N = M^2$)	10*10	15*15	20*20	26*26

FFT. In two dimensions, the oversampled FFT requires about 2.5 M_r^2 μsec. This is four times as expensive as an $M \times M$ FFT. Table 2 shows how the computational cost for gridding grows with precision. It also shows the break-even point with respect to the direct algorithm.

In summary, the NUFFT is about 4 times more expensive in one dimension than a traditional M-point FFT for single precision accuracy. It is about 25 times slower in the current implementation than the traditional two-dimensional $M \times M$ FFT—a factor of 21 from gridding and a factor of 4 from the oversampled FFT.

Example 4 (MRI Image Reconstruction). One of the important applications of the nonuniform FFT is to magnetic resonance imaging (MRI) [6, 8, 11, 12, 13, 14]. The MRI hardware is able to acquire the Fourier transform of a particular tissue property at selected points in the frequency domain. In most clinical systems, the device is designed to acquire data on a uniform Cartesian mesh, from which a standard FFT can be used for image reconstruction. For a variety of technical reasons, however, nonuniform data sampling techniques are much better suited for fast data acquisition, motion correction, and functional MRI [4]. In this example, we create simulated MRI

data by using a type-2 transformation in two dimensions:

$$F(s_x^k, s_y^k) = \sum_{j_1} \sum_{j_2} f(j_1, j_2)\, e^{-i(j_1,j_2)\cdot(s_x^k, s_y^k)}\,, \tag{17}$$

followed by a type-1 transformation to reconstruct the image,

$$\tilde{f}(j_1, j_2) = \sum_{k=0}^{N-1} F_k\, e^{i(j_1,j_2)\cdot(s_x^k, s_y^k)}. \tag{18}$$

If the function $F(s_x, s_y)$ were known everywhere, then the exact reconstruction would obviously be the Fourier integral

$$\tilde{f}(j_1, j_2) = \int_0^{2\pi} \int_0^{\infty} F(r,\theta) e^{i(j_1,j_2)\cdot(r\cos\theta, r\sin\theta)} r\, dr\, d\theta, \tag{19}$$

written in polar coordinates. It is probably worth repeating a point made in the introduction: once the decision has been made to use (18) for reconstruction, one still has a number of degrees of freedom to work with. Engineering considerations determine the selection of points $\{(s_x^k, s_y^k)\}$, which will certainly affect the image quality. One must also select quadrature weights W_k so that, in the transform (18), $F_k \equiv W_k\, F(s_x^k, s_y^k)$. There are a number of interesting optimization questions that arise here, which will be addressed in subsequent work.

Here, we will simply use a radial grid and truncate the integral (19) at $r = \pi$, introducing a "ringing" artifact that can be seen in the reconstruction. For a fixed NUFFT tolerance ϵ, the result computed agrees with the exact *sum* (18) to within that error. More precisely, we let

$$(s_x^k, s_y^k) = r_j\, (\cos(\theta_i), \sin(\theta_i))\,,\ r_j = \frac{\pi j}{M},\ \theta_i = \frac{2\pi i}{2M} \tag{20}$$

for $k = i + M * j$, $0 \le j < M$, $0 \le i < 2M$, so that $N = 2M^2$. Thus, the quadrature weight for the point indexed by $k = i + M * j$ is $r_j\, \Delta\theta\, \Delta r = (j\pi/M)\cdot(2\pi/2M)\cdot(\pi/M)$.

Figure 5 shows the image reconstructed in this manner from the well-known Shepp–Logan phantom sampled on a 256×256 grid. We set the Fourier transform tolerance to $\epsilon = 10^{-6}$.

5. Conclusions. The nonuniform FFT (NUFFT) is an important, and relatively recent, algorithm. There is, however, some confusion in the literature about its use. It is simply a fast algorithm for computing discrete sums of a certain type. It is completely independent of considerations having to do with acquisition of data in the Fourier domain or the choice of a quadrature scheme in computing Fourier integrals. The use of Gaussian spreading inside the algorithm has no theoretical advantage over any other properly applied "spreading function." It does, however, allow a particularly simple and fast implementation, as described above. We believe it provides the first reasonably efficient three-dimensional scheme.

We do not mean to suggest, however, that all schemes for Fourier reconstruction must be based on a quadrature approach and the calculation of sums like (18) or (1). A more general linear reconstruction algorithm could take the form

$$F(k_1, k_2) = \frac{1}{N} \sum_{j=0}^{N-1} W(j, k_1, k_2) f(\mathbf{x}_j)\, e^{-i(k_1,k_2)\cdot\mathbf{x}_j}\,. \tag{21}$$

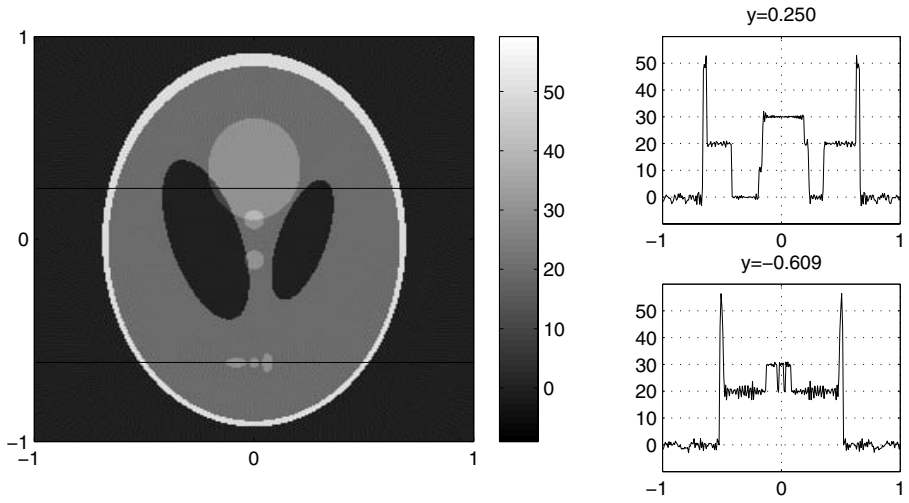

Fig. 5 *Reconstructed image from Fourier data sampled on a radial (polar coordinate) grid. The curves at the right show the reconstructed function along the lines* $y = 0.25$ *and* $y = -0.609$.

Here the weight W depends on both the "source" (j) and the "target" (k_1, k_2). The NUFFT does not apply to this calculation.

REFERENCES

[1] C. Anderson and M. D. Dahleh, *Rapid computation of the discrete Fourier transform*, SIAM J. Sci. Comput., 17 (1996), pp. 913–919.

[2] S. Bagchi and S. Mitra, *The Nonuniform Discrete Fourier Transform and Its Applications in Signal Processing*, Kluwer Academic, Boston, 1999.

[3] G. Beylkin, *On the fast Fourier transform of functions with singularities*, Appl. Comput. Harmonic Anal., 2 (1995), pp. 363–383.

[4] M. Bourgeois, F. Wajer, D. Ormondt, and D. Graveron-Demilly, *Reconstruction of MRI images from non-uniform sampling and application to Intrascan motion correction in functional MRI*, in Modern Sampling Theory: Mathematics and Applications, J. J. Benedetto and P. Ferreira, eds., Appl. Numer. Harmon. Anal., Birkhäuser, Boston, 2001, pp. 343–363.

[5] A. Dutt and V. Rokhlin, *Fast Fourier transforms for nonequispaced data*, SIAM J. Sci. Comput., 14 (1993), pp. 1368–1393.

[6] J. A. Fessler and B. P. Sutton, *Nonuniform fast Fourier transforms using min-max interpolation*, IEEE Trans. Signal Process., 51 (2003), pp. 560–574.

[7] L. Greengard and P. Lin, *Spectral approximation of the free-space heat kernel*, Appl. Comput. Harmonic Anal., 9 (2000), pp. 83–97.

[8] J. I. Jackson, C. H. Meyer, D. G. Nishimura, and A. Macovski, *Selection of a convolution function for Fourier inversion using gridding*, IEEE Trans. Med. Imag., 10 (1991), pp. 473–478.

[9] Q. H. Liu and N. Nguyen, *An accurate algorithm for nonuniform fast Fourier transforms (NUFFT)*, IEEE Microwave Guided Wave Lett., 8 (1998), pp. 18–20.

[10] N. Nguyen and Q. H. Liu, *The regular Fourier matrices and nonuniform fast Fourier transforms*, SIAM J. Sci. Comput., 21 (1999), pp. 283–293.

[11] J. D. O'Sullivan, *A fast sinc function gridding algorithm for Fourier inversion in computer tomography*, IEEE Trans. Med. Imag., MI-4 (1985), pp. 200–207.

[12] G. B. Pike, *Multidimensional Fourier transformation in magnetic resonance imaging*, in The Fourier Transformation in Biomedical Engineering, T. Peters and J. Williams, eds., Appl. Numer. Harmon. Anal., Birkhäuser, Boston, 1998, pp. 89–128.

[13] D. Potts, G. Steidl, and M. Tasche, *Fast Fourier transforms for nonequispaced data: A tutorial*, in Modern Sampling Theory: Mathematics and Applications, J. J. Benedetto and P. Ferreira, eds., Appl. Numer. Harmon. Anal., Birkhäuser, Boston, 2001, pp. 249–274.

[14] G. E. Sarty, R. Bennett, and R. W. Cox, *Direct reconstruction of non-Cartesian k-space data using a nonuniform fast Fourier transform*, Magn. Reson. Med., 45 (2001), pp. 908–915.

[15] J. Strain, *Fast potential theory* II: *Layer potentials and discrete sums*, J. Comput. Phys., 99 (1992), pp. 251–270.

[16] A. R. Thompson and R. N. Bracewell, *Interpolation and Fourier transformation of fringe visibilities*, Astronom. J., 79 (1974), pp. 11–24.

[17] A. F. Ware, *Fast approximate Fourier transforms for irregularly spaced data*, SIAM Rev., 40 (1998), pp. 838–856.