Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Fast transform from an adaptive multi-wavelet representation to a partial Fourier representation

Jun Jia [*], Robert Harrison, George Fann

Oak Ridge National Laboratory, Oak Ridge, TN 37831-6367, USA

ARTICLE INFO

Article history:
Received 28 May 2009
Received in revised form 1 April 2010
Accepted 6 April 2010
Available online 14 April 2010

Keywords:
Multi-wavelet
Multi-resolution analysis
Fourier transform
FFT

ABSTRACT

We present a fast algorithm to compute the partial transformation of a function represented in an adaptive pseudo-spectral multi-wavelet representation to a partial Fourier representation. Such fast transformations are useful in many contexts in physics and engineering, where changes of representation from a piece wise polynomial basis to a Fourier basis. The algorithm is demonstrated for a Gaussian in one and in three dimensions. For 2D, we apply this approach to a Gaussian in a periodic domain. The accuracy and the performance of this method is compared with direct summation.

1. Introduction

We describe a fast algorithm to compute the partial Fourier transform of a d-dimensional function $f(\mathbf{x})$ given in an adaptive discontinuous multi-wavelet expansion [2,3]. The partial transform is represented by an integral:

$$\hat{f}(\mathbf{p}) = \int_{[0,1]^d} f(\mathbf{x}) e^{-i\mathbf{x}\cdot\mathbf{p}} d\mathbf{x}, \quad \mathbf{p} \in \Omega^d, \tag{1}$$

where $\Omega = \{2\pi m | m = 0, 1, \ldots, c\}$. By partial we mean that we only compute the Fourier components less than some band limit $2\pi c$ in each dimension. Our algorithm may be regarded as a fast Fourier transform for a set of non-equally spaced sampling points [5,9,12] that use the adaptive multi-wavelet basis for interpolation and sampling.

Our primary applications are in physics and chemistry including the electronic band-structure of solids and photonics of nanoscale systems. These disciplines employ adaptive representations to capture localized fine-scale structure (e.g., to resolve the cusp in a wave function at the nucleus of an atom, or the high fields that arise in the vicinity of a corner of a nanoscale crystal), to robustly maintain precision (e.g., by treating both bound and continuum states with equal fidelity), and for efficient computation.

Despite the numerical and computational advantages of the adaptive algorithms, Fourier methods are still important. The physical interpretation of results is often best performed in Fourier space, such as in the generation of Wannier functions from Bloch states to form a local basis from the delocalized bands of solid-state electronic structure. Also, compact, albeit cumbersome, representations can be formed by combining coarse-scale Fourier modes with localized real-space expansions. The full potential linearized augmented plane wave (FLAPW) method for all-electron calculations in solids is an example of this [8].

[*] Corresponding author.
E-mail addresses: jiaj@ornl.gov (J. Jia), harrisonrj@ornl.gov (R. Harrison), fanngi@ornl.gov (G. Fann).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.04.006

Similarly, certain fast algorithms for the application of singular integral kernels using multi-resolution analysis [10] rely upon the kernel being smooth and oscillation free at long range, which, for example, is not the case with the important scattering-state Helmholtz kernel $\frac{e^{ir}}{r}$. Recently, Beylkin et al. [7] have combined Fourier and real-space representations of both the operator and the target function to construct fast algorithms for the Helmholtz and other similar kernels. However, this work relied upon a fast Fourier transform for non-equally spaced points [5,9,12] that employs multiple representations to attain efficiency and seems to be very complex to implement efficiently on a parallel processor. As we shall see below, it is usually the case that changing the representation or data layout is as or more expensive than the actual transformation; and, hence an algorithm tailored explicitly for the multi-wavelet basis that eliminates intermediate forms is ultimately more efficient in software such as MADNESS, Multi-resolution ADaptive Numerical Environment for Scientific Simulations [1].

For the above reasons it is highly desirable to have a numerical framework that permits us to switch as desired between the following equivalent representations:

- coefficients of the adaptively refined discontinuous spectral element (scaling function) basis,
- function values at the Gauss–Legendre quadrature points of the adaptively refined grid,
- coefficients of the multi-wavelets, and
- coefficients of selected Fourier components.

The first three are naturally provided by the multi-wavelet basis. Therefore, following the scheme presented below, we have extended the MADNESS software with the efficient and robust computation of the partial Fourier transform of a function represented in an adaptive, discontinuous spectral element basis.

The projection of a bandlimited function into a truncated multi-wavelet basis is no longer exactly bandlimited. However, the truncation of the basis is controlled by a user-specified precision and the bandlimit is maintained within that precision once some minimal resolution has been achieved. This is a critical point for designing fast algorithms. In this paper, the bandlimit of interest is not associated with the input function but is a user-specified parameter. The main content of the paper is presenting, analyzing and testing an algorithm designed to accurately and efficiently compute the partial Fourier transform.

This paper is organized as follows: in Section 2, we summarize and recall the relevant aspects of multi-wavelet representations of a function. In Section 3, we describe computation of the fast transform from a function represented at a fixed level to the corresponding Fourier space, and in Section 4 extend the transform to an adaptively refined function representation. In Section 5, several possible performance enhancements are discussed. Finally, we demonstrate the transforms with several numerical examples.

2. Multi-wavelet representation

We use the multi-resolution analysis approach described in [3] with the multi-wavelet basis constructed by Alpert [2]. In this approach, the underlying basis functions, or scaling functions ϕ_j, are the first k Legendre polynomials P_j, scaled, shifted and normalized on $(0,1)$

$$\phi_j(x) = \begin{cases} \sqrt{2j+1}P_j(2x-1), & x \in (0,1), \\ 0, & \text{otherwise}. \end{cases} \tag{2}$$

Dyadically refining the unit interval n times, we obtain 2^n equally spaced intervals $[(2^{-n}l, 2^{-n}(l+1)),\ l = 0,\ldots,2^{n-1}]$, each of width 2^{-n}. On each of the refined intervals the set of rescaled and translated scaling functions

$$\phi^n_{jl}(x) = 2^{n/2}\phi_j(2^n x - l) \tag{3}$$

is defined and forms a basis on V^k_n, which is the space of scaling functions at level n. The multi-wavelet basis, denoted by $\{\psi_j(x),\ j = 0,\ldots,k-1\}$, is an orthonormal basis that spans the orthogonal complement of V^k_0 in V^k_1 with similar rescaling and translation properties. Thus, $V^k_1 = V^k_0 \oplus W_0$. Alpert [2] imposes an additional k vanishing moments constraint to uniquely define this basis. In many dimensions we use tensor products of the scaling functions and the multi-wavelets.

Since $V^k_0 \subset V^k_1$ and $W_0 \subset V^k_1$, there is an exact two-scale relationship between the two sets of scaling bases and thus we can write

$$\begin{pmatrix} \bar{\phi}(x) \\ \bar{\psi}(x) \end{pmatrix} = \sqrt{2}\begin{pmatrix} H^{(0)} & H^{(1)} \\ G^{(0)} & G^{(1)} \end{pmatrix}\begin{pmatrix} \bar{\phi}(2x) \\ \bar{\phi}(2x-1) \end{pmatrix}, \tag{4}$$

which defines an orthonormal transformation between the scaling and multi-wavelet functions and the scaling functions at the next finest scale. Here $\bar{\phi}(x)$ and $\bar{\psi}(x)$ represent the vectors $(\phi_0,\ldots,\phi_{k-1})$ and $(\psi_0,\ldots,\psi_{k-1})$. This is the multi-wavelet version of the fast wavelet transform [6] and is used throughout this paper to obtain coarse-scale representations of adaptively refined functions. In higher dimensions, we use these two-scale relationships to simultaneously refine in all dimensions.

Since $L^2([0,1]) = \overline{\lim_{n\to\infty} V^k_n}$, we have the L_2 norm, $\langle f, g\rangle = \int_0^1 f(x)g(x)dx$, defined on each V^k_n. By construction, the scaling and wavelet functions satisfy the following orthogonality properties

$$\begin{aligned}
&\langle \psi^n_{jl}, \psi^{n'}_{j'l'}\rangle = \delta_{nn'}\delta_{ll'}\delta_{jj'},\\
&\langle \phi^n_{jl}, \phi^n_{j'l'}\rangle = \delta_{ll'}\delta_{jj'},\\
&\langle \phi^n_{jl}, \psi^{n'}_{j'l'}\rangle = 0 \quad n' \geqslant n.
\end{aligned} \tag{5}$$

A one dimensional function $f \in L^2([0,1])$ can be projected to V^k_n and is represented as

$$f^n(x) = \sum_{l=0}^{2^n-1}\sum_{j=0}^{k-1} s^n_{jl}\phi^n_{jl}(x), \tag{6}$$

where s^n_{jl} are the scaling function coefficients $\langle f, \phi^n_{jl}\rangle$. Computationally, this may be recast as an interpolation between function values at the Gauss–Legendre quadrature points in each sub-interval. Since the basis $\{\phi^0_{jl}, \psi^m_{jl}\}$ also spans V^k_n, f^n can be rewritten as

$$f^n(x) = \sum_{j=0}^{k-1} s^0_{j0}\phi_j(x) + \sum_{m=0}^{n-1}\sum_{l=0}^{2^m-1}\sum_{j=0}^{k-1} d^m_{jl}\psi^m_{jl}(x), \tag{7}$$

in the wavelet basis where $d^m_{jl} = \langle f, \psi^m_{jl}\rangle$ are the wavelet coefficients. Since the first k moments of the wavelets are zero by construction, the wavelet coefficients become sparse once the refinement level is sufficiently fine for the function to be locally smooth and truncation of small wavelet coefficients within a specified tolerance can be applied. This local truncation enables adaptive refinement and fast algorithms with guaranteed precision [6].

3. Fast partial Fourier transform of $f^n \in V^n$

Without loss of generality, we discuss our method in 1D and outline the extension to multiple dimensions.

We first consider a function f projected to f^n and represented in the scaling function basis. At the fixed level n, f can be written as

$$f^n(x) = \sum_{l=0}^{2^n-1}\sum_{j=0}^{k-1} s^n_{jl}\phi^n_{jl}(x). \tag{8}$$

By linearity, the Fourier transform of $f^n(x)$ is reduced to the computation of the Fourier transform of the scaling functions,

$$\hat{f}^n(p) = \int_0^1 f^n(x)e^{-ix\cdot p}dx \quad p \in \Omega. \tag{9}$$

We first compute the Fourier transform of the scaling function $\phi^n_{jl}(x)$:

Lemma 1. *Let $\phi^n_{jl}(x)$ be a scaling function defined as in* Section 2. *Then its Fourier transform is*

$$\hat{\phi}^n_{jl}(p) = e^{-i2^{-n}lp}2^{-n/2}\hat{\phi}_j(2^{-n}p). \tag{10}$$

The derivation is straightforward. Thus, the translation dependence has factored out into an easily computed phase and hence we need only to compute the Fourier transform of the basis functions in one box at level n. For simpler notation, we denote

$$r^n_j(p) = 2^{-n/2}\hat{\phi}_j(2^{-n}p), \tag{11}$$

and by linearity we have:

Corollary 2

$$\hat{f}^n(p) = \sum_{l=0}^{2^n-1}\sum_{j=0}^{k-1} s^n_{jl}e^{-i2^{-n}lp}r^n_j(p) = \sum_{j=0}^{k-1} r^n_j(p)\sum_{l=0}^{2^n-1} s^n_{jl}e^{-i2^{-n}lp}. \tag{12}$$

The sum over l for $p = 2N\pi, (N = 0,1,\ldots,2^n-1)$ can be performed with an FFT of size 2^n, and the transform is then computed by summation of k such FFT's, which yield a total cost of $O(nk2^n)$.

The terms $r^n_j(p)$ can be precomputed and tabulated as desired for computational efficiency. The accuracy of the transform depends solely on how accurately this integral is computed. Let $\bar{r}^n_j(p)$ denote the numerically computed $r^n_j(p)$, $\bar{\varepsilon} = \max|r^n_j(p) - \bar{r}^n_j(p)|$ and define $\hat{\hat{f}}^n$ by substituting r by $\bar{r}$ in $\hat{f}^n$,

$$\hat{\hat{f}}^n = \sum_{j=0}^{k-1} \bar{r}^n_j(p)\sum_{l=0}^{2^n-1} s^n_{jl}e^{-i2^{-n}lp}. \tag{13}$$

Thus, we have

$$\hat{f}^n = \hat{\hat{f}}^n + O(\bar{\varepsilon}), \tag{14}$$

and we have total control over $\bar{\varepsilon}$.

Note that the above FFT's compute the full transform; however, we are only interested in the first $c + 1$ components instead of 2^n. Therefore, we can take advantages of the fractional FFT [4] by rewriting the sum as:

$$\hat{f}^n(p) = \sum_{j=0}^{k-1} r_j^n(p) \sum_{l=0}^{2^n-1} s_{jl}^n e^{-i2^{-n}lp} = \sum_{j=0}^{k-1} r_j^n(p) \sum_{b=0}^{q-1} \sum_{a=0}^{2^m-1} s_{j(b+aq)}^n e^{-i2^{-n}(b+aq)p} = \sum_{j=0}^{k-1} r_j^n(p) \sum_{b=0}^{q-1} e^{-i2^{-n}bp} \sum_{a=0}^{2^m-1} s_{j(b+aq)}^n e^{-i2^{-m}ap}, \tag{15}$$

where $m = \lceil \log_2(c+1) \rceil$ and $q = 2^{n-m}$.

Instead of k size 2^n FFT's in (12), the partial transform is a summation of $k \cdot q$ smaller size 2^m FFT's on the index a for each $p = 2N\pi, (N = 0,1,\ldots,c,\ldots,2^m-1)$. Thus, the total cost is reduced to $O(mk\,2^n)$.

4. Fast partial Fourier transform of *f*

In practice, a multi-wavelet function representation is adaptively refined instead of being at fixed at level n. It may be refined below the level necessary to represent the locally smooth Fourier components of interest. However, to compute the transforms accurately we only need to consider the projection of the function at the coarsest scale necessary to represent the sought Fourier components to the desired precision, i.e. V_n^k for some n. Although neither the scaling functions nor the corresponding wavelets are strongly bandlimited, the result immediately follows from the representation of both the function and the plane waves in the orthonormal wavelet basis.

Suppose a function $f \in L^2([0,1])$ is represented as a multi-wavelet expansion

$$f(x) = \sum_{l=0}^{2^n-1} \sum_{j=0}^{k-1} s_{jl}^n \phi_{jl}^n(x) + \sum_{m=n+1}^{\infty} \sum_{l=0}^{2^m-1} \sum_{j=0}^{k-1} d_{jl}^m \psi_{jl}^m(x) = f^n(x) + f_r^n(x),$$

where $f^n \in V_n^k$ is the projection of f in V_n^k, and f_r^n represents the contributions from finer scales.

Lemma 3. *Let $p \in \mathbb{Z}$ and $E^n(x)$ be the projection of e^{ixp} in V_n^k and $\varepsilon_p \doteq \max|E_r^n| \doteq \max|e^{ixp} - E^n(x)|, \varepsilon_f \doteq \max|f_r^n| \doteq \max|f - f^n|$, then*

$$|\hat{f}(p) - \hat{f}^n(p)| \leqslant \varepsilon_p \cdot \varepsilon_f. \tag{16}$$

Proof

$$|\hat{f}(p) - \hat{f}^n(p)| = \left| \int_0^1 e^{ixp} \cdot (f - f^n)dx \right| = \left| \int_0^1 (E^n + E_r^n) \cdot f_r^n dx \right| \leqslant \left| \int_0^1 E^n \cdot f_r^n dx \right| + \max|E_r^n| \cdot \left| \int_0^1 f_r^n dx \right|.$$

The first term vanishes due to (5) and we have $|\hat{f}(p) - \hat{f}^n(p)| \leqslant \varepsilon_p \cdot \varepsilon_f$. □

Therefore, together with (14), we have

$$\hat{f} = \hat{\hat{f}}^n + O(\bar{\varepsilon} + \varepsilon_p \cdot \varepsilon_f). \tag{17}$$

Typically $\varepsilon_p, \varepsilon_f \sim O(2^{-nk})$ [2]. This implies that the application of order *2 k* Gauss–Legendre quadrature rule on level n suffices to evaluate $r_j^n(p)$ in (11) if we choose the level such that $\bar{\varepsilon} \approx \varepsilon_p \cdot \varepsilon_f$.

5. Further discussion

In the following, we consider how projection into a higher-order polynomial basis affects the computational cost of the partial Fourier transform.

With a basis of order k uniformly refined at level n the number of sampling points is $k\,2^n$. To represent $\cos(6\pi x), x \in [0,1]$ to 12 digits with $k = 10$, we must refine to level $n = 4$ using 160 points. With $k = 30$, this can be accomplished at level $n = 0$ with just 30 points. Hence, it is advantageous to project f into a high-order basis and then truncate at the level necessary to represent the plane waves. The performance benefit of using higher-order polynomials will be sensitive to the relative speeds of the actual implementations of matrix–vector operations and FFT. However, if we choose a high-order basis with, for example, $k = 30$ in $[0,1]$ we can represent $\cos(24\pi x)$ to 12 digits at level two and $\cos(48\pi x)$ at level three. These Fourier components are adequate for the intended applications in solid-state physics, but the range of summation for l is probably too short to benefit significantly from fast summation. Hence, with $(c + 1)$ points being sampled in Fourier space it might be faster (and certainly easier) to re-order the summation as

$$\hat{f}^n(p) = \sum_{l=0}^{2^n-1} e^{-i2^{-n}lp} \sum_{j=0}^{k-1} r_j^n(p) s_{jl}^n. \tag{18}$$

This can be computed directly in $O((c+1)\,k\,2^n)$ operations.

We may also perform the sum in the multi-wavelet basis and exploit sparsity of the difference coefficients. In the target function, this corresponds to the function being locally less deeply refined than the level necessary to represent some Fourier components. For our intended applications the target function will usually be more deeply refined nearly everywhere than the level necessary to represent the maximum Fourier component and therefore not a source of sparsity. The multi-wavelet representation of a low-frequency Fourier component is naturally sparse relative to that of a higher-frequency component. However, this sparsity is limited due to the poor localization of multi-wavelets in Fourier space and the structure of this sparsity is equivalent to that exploited with great efficiency by the FFT algorithm and highly optimized implementations thereof. For these reasons we restrict our attention to constructing a fast algorithm based upon the FFT.

The extension to higher dimensions is straightforward. In d-dimensions, due to the effect of immediate summation, the lowest flop count is achieved by applying the transformation successively in each dimension. Theoretically, one can achieve $O\left(mk2^n\sum_{h=1}^{d}(c+1)^{h-1}(k2^n)^{d-h}\right)$ for FPFT and $O\left(\sum_{h=1}^{d}(c+1)^{h}(k2^n)^{d-h+1}\right)$ for direct summation, where $m=\lceil\log_2(c+1)\rceil$. However, in this paper we choose another approach to improve the practical performance by utilizing the already-optimized multi-dimensional FFT and BLAS subroutines. We arrange a d-dimensional transformation as

$$\hat{f}^n(\vec{p})=\sum_{\vec{j}=0}^{\mathbf{k-1}}\left(\prod_{h=1}^{d}r^n_{j_h}(p_h)\right)\sum_{\vec{b}=0}^{\mathbf{q-1}}e^{-i2^{-n}\vec{b}\cdot\vec{p}}\sum_{\vec{a}=0}^{\mathbf{2^m}-1}\left(\prod_{h=1}^{d}s^n_{(b_h+a_hq_h)j_h}\right)e^{-i2^{-m}\vec{a}\cdot\vec{p}},\tag{19}$$

where a bold number means a d-vector, $\mathbf{z}=(z,\ldots,z)$, and the last summation is computed by a d-dimensional FFT. The total cost is $O(mdk^d\,2^{dn})$. Similarly, when c is small, we re-order the summation as

$$\hat{f}^n(\vec{p})=\sum_{\vec{l}=0}^{\mathbf{2^n}-1}e^{-i2^{-n}\vec{l}\cdot\vec{p}}\sum_{\vec{j}=0}^{\mathbf{k-1}}\left(\prod_{h=1}^{d}r^n_{j_h}(p_h)s^n_{l_hj_h}\right),\tag{20}$$

and the last summation is computed by a tuned subroutine in MADNESS [1] with immediate summation, which calls BLAS. Thus the cost of the direct transformation is $O\left(2^{dn}\sum_{h=1}^{d}(c+1)^{h}k^{d-h+1}\right)$.

6. Numerical examples

We utilize the MADNESS (Multi-resolution Adaptive Numerical Environment for Scientific Simulation [1]) for multi-wavelet approximations, and FFTW-3.2 [11] for FFT related computations. The reference CPU time is obtained on a 1.8 GHz AMD Opteron 844 with the GCC-4.3.2 compiler and the BLAS library provided by ACML-4.2.0.

6.1. Example 1

On the interval $[-1/2,+1/2]$ in 1D, let $g(x,\sigma)=\sqrt{\frac{\sigma}{\pi}}e^{-\sigma|x|^2}$, i.e., a normalized Gaussian, where the parameter σ controls the length scales. We computed the Fourier transform of $g(x,\sigma)$ with a band limit of $c = 20$ and compared with the analytic result. The Fourier components were computed at the level where cos $(2\pi c)$ is resolved to machine precision with k=17 wavelets. The approximation error in the multi-wavelet representation of the target function was set to 10^{-15}. For various exponents, $\sigma = 2^m$ with $m = -1,0,1,\ldots,20$, the errors of the partial Fourier transforms were bounded by $2\cdot 10^{-15}$.

Subsequently, we chose a relatively low order basis $k = 5$ to explore the dependence of the error (16) upon the projection level n. For larger k's, it takes much fewer levels to resolve all the interested Fourier components.

The Gaussian function was projected into the scaling function basis at each level 4 to 7, in turn, and the partial Fourier transform was performed at the same level. In Fig. 1, with a band limit of $c = 15$, we check the convergence of the Fourier transform with the largest frequency computed, $2\cdot 15\cdot \pi$. Between level 4 and 5, the errors decay roughly as 2^{-kn} while between level 6 and 7, the errors decay as 2^{-2kn}. According to the error estimate (16), at coarser scales (smaller n), the error of the wavelet approximation of the transform is denoted by ε_f, for a large exponent Gaussian ($\sigma = 2^{20}$) does not decay rapidly as we refine the box. Therefore, we only observed convergence $O(2^{-5n})$ contributed by the error of the wavelet approximation for e^{ixp}, ε_p. As soon as the level is deep enough to start resolving the Gaussian, the error decays as $\varepsilon_p\cdot\varepsilon_f = O(2^{-10k})$, including the contribution from both approximation of the Gaussian and the plane wave.

In contrast, Fig. 2 shows the result with a much smoother Gaussian ($\sigma = 2^6$). We set $k = 5$ and computed the partial Fourier transform with a band limit of $c = 8$; the errors decay very fast at a rate of about 2^{-2nk} right from the coarsest level until the machine precision is reached.

In Fig. 3, we compare the CPU time used by the fast partial fourier transform (FPFT) and the direct summation. In this test, the order of the wavelets was set to $k = 8$, and the threshold was 10^{-15}. The FPFT was computed on level $n = 1$ to 11 and the band limit was set to $c = 2^n-1$, i.e. $m = \log_2(c + 1) = n$. The FPFT is accelerated by the FFT fast summation as soon as the number of computed Fourier components increased to 16. For direct summation, the elapsed CPU time increased at a rate of roughly 2^{2m}, while for the FPFT, the CPU time increased as $m2^m$; and these results agree with our computational cost estimates. The spuriously slow performance of $c = 127$ is reproducible and we speculate this is due to how FFTW is affected by power-of-two dimension arrays and memory layout.

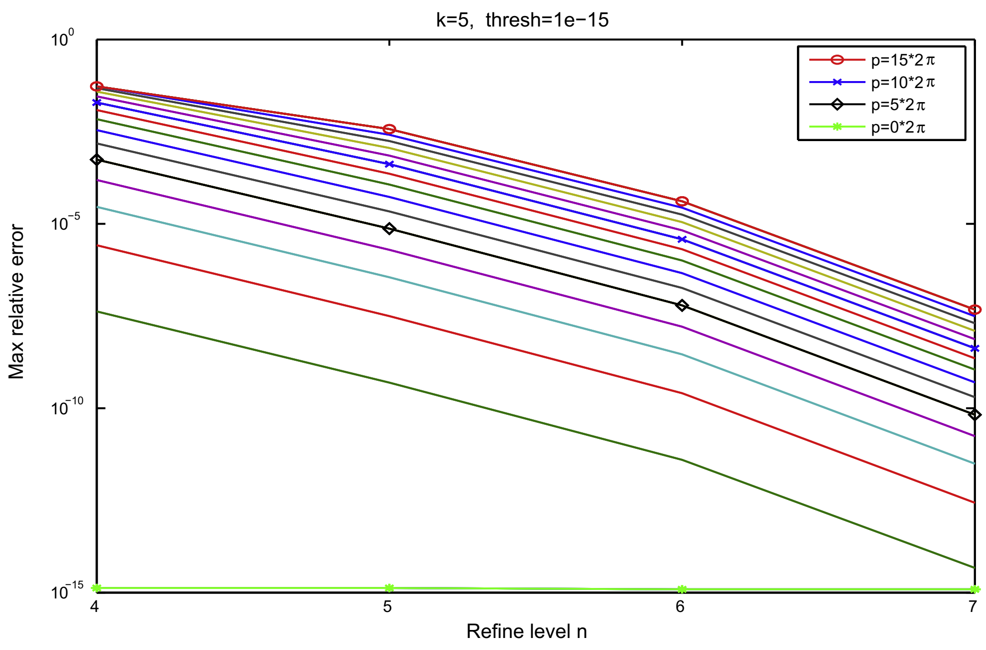

Fig. 1. FPFT of $g(x, 2^{20})$.

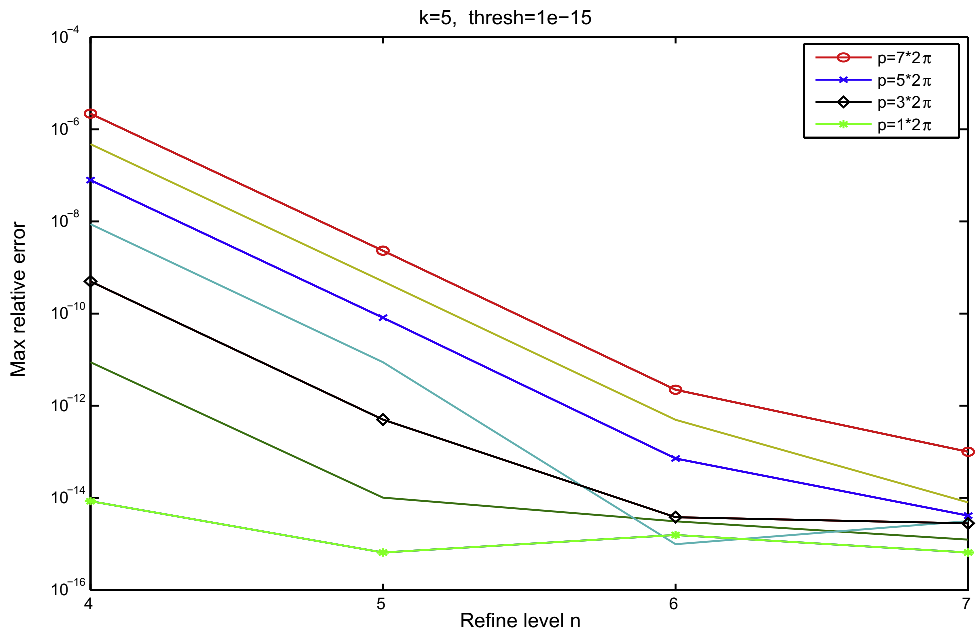

Fig. 2. FPFT of $g(x, 2^6)$.

6.2. Example 2

We constructed a periodic Gaussian, p, in 3D from a normalized Gaussian, g, with the lattice sum,

$$p(x,\sigma) = \sum_{R\in\mathbb{Z}^3} g(x-R,\sigma), \quad x \in \left[-\frac{1}{2},\frac{1}{2}\right]^3, \tag{21}$$

the sum over R will be finite due to the decay of the Gaussian. We generate an input function that represents randomly distributed electronic and nuclear charge densities. The periodic charge density of an atom with nuclear charge Z centered at the origin is represented as

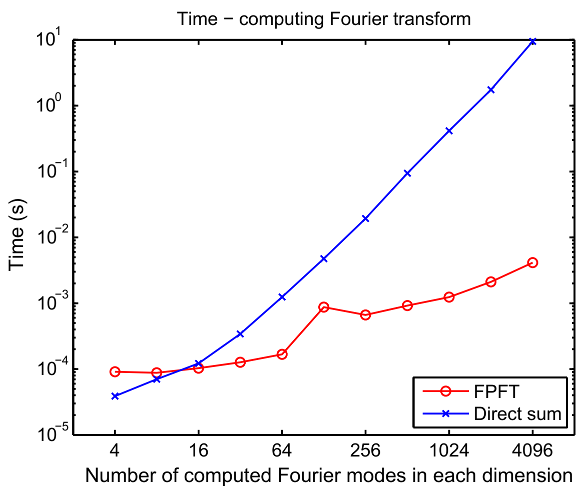

Fig. 3. CPU time comparison.

$$atom(x, Z) = 2p(x, 0.2) + (Z - 2) \cdot p(x, \sqrt{Z}) - Z \cdot p(x, 10,000). \tag{22}$$

This is overall charge neutral. The first term represents the electronic valence charge distribution, the second represents the core electrons, and the third approximates a point nuclear charge. Finally, the total distribution is formed by 22 atoms with Z from the list 8 to 29 (nuclear charge of oxygen to copper) randomly centered in the unit cell $[-1/2, 1/2]^3$.

We performed the Fourier transforms at level $n = 2, \ldots, 6$ with a band limit of $c = 2^{n-1} - 1$, i.e. $m = \log_2(c + 1) = n - 1$. In this example, we used order $k = 12$ wavelets with a threshold of 10^{-8}. In Fig. 4, we compare the CPU time used by the FPFT and the direct summation. As shown in the figure, the FPFT is accelerated by the fast summation as soon as the number of computed Fourier components in each dimension increased to 4.

6.3. Example 3

Let a periodic Gaussian function in $[-1/2, +1/2]^2$ be represented by

$$p(x, \sigma) = \sum_{R \in \mathbb{Z}^2} g(x - R, \sigma), \tag{23}$$

with now R a 2-vector. We set the exponent $\sigma = 2^{20}$, then used $k = 10$ wavelets and adaptive refinement with a threshold 10^{-12} to obtain the initial multi-wavelet representation for the 2D Gaussian. We computed the first 32^2 Fourier components with an accuracy of 10^{-12}, and the level of refinement was determined by the required accuracy.

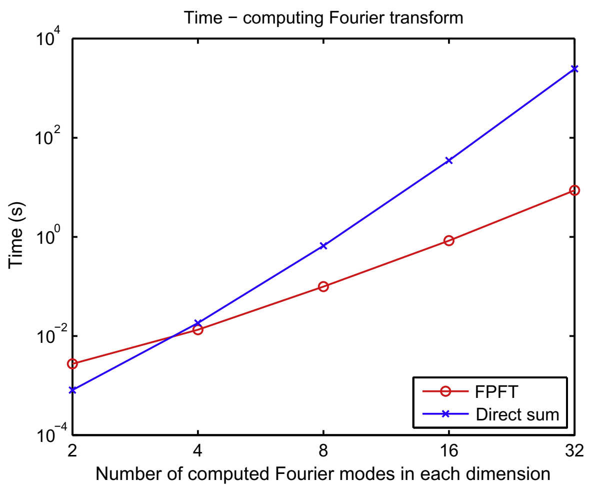

Fig. 4. CPU time comparison.

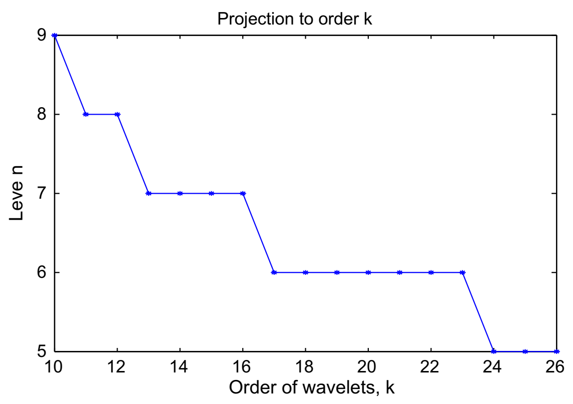

Fig. 5. Level of refinement.

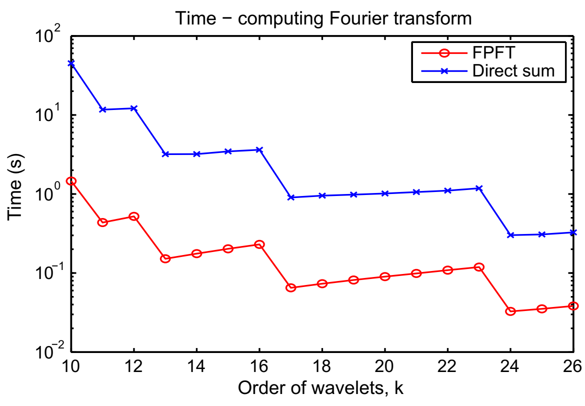

Fig. 6. CPU time comparison.

Fig. 5 shows which level of refinement is needed to resolve the largest frequency Fourier component. For a given accuracy ε, the required level n is asymptotically $-\frac{\log_2 \varepsilon}{2k}$ as a function of k according to (17). This implies projection to a higher-order wavelet basis will increase the performance if the projection results in a reduction of the level of refinement. Therefore, for a given accuracy requirement, we should choose the least of the largest possible k's that leads to the lowest level of refinement delivering the required accuracy. In Fig. 6, we compare the CPU time used by the FPFT and the direct sum with projection to different k's. According to these results, for the best execution-time performance overall, one should always pick the highest possible k. However memory limit constrains our choice of k. For the best performance at a given level of refinement we should use the smallest k according to Fig. 5.

Acknowledgements

This work was supported by the Scientific Discovery through Advanced Computing (SciDAC) program of the U.S. Department of Energy, the division of Basic Energy Science, Office of Science, under contract number DE-AC05–00OR22725 with Oak Ridge National Laboratory. G.Fann and J.Jia were partially supported by the Office of Advanced Scientific Computing, Applied Mathematics Program of SciDAC.

References

[1] <http://code.google.com/p/m-a-d-n-e-s-s/>.
[2] B. Alpert. Sparse Representation of Smooth Linear Operators, PhD Thesis, Yale University, 1990.
[3] B. Alpert, G. Beylkin, D. Gines, L. Vozovoi, Adaptive solution of partial differential equations in multiwavelet bases, J. Comput. Phys. 182 (1) (2002) 149–190.

[4] D.H. Bailey, P.N. Swarztrauber, The fractional Fourier transform and applications, SIAM Rev. 33 (1991) 389–404.
[5] G. Beylkin, On the fast Fourier transform of functions with singularities, Appl. Comput. Harmon. Anal. 2 (4) (1995) 363–381.
[6] G. Beylkin, R.R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms I, Comm. Pure Appl. Math. 44 (1991) 141–183.
[7] G. Beylkin, C. Kurcz, L. Monzn, Fast algorithms for Helmholtz Green's functions, Proc. Roy. Soc. A: Math. Phys. Eng. Sci. 464 (2100) (2008) 3301–3326. Dec.
[8] P. Blaha, K. Schwarz, P. Sorantin, S.B. Trickey, Full-potential, linearized augmented plane wave programs for crystalline systems, Comput. Phys. Commun. 59 (2) (1990) 399–415.
[9] A. Dutt, V. Rokhlin, Fast Fourier transforms for nonequispaced data, SIAM J. Sci. Comput. 14 (6) (1993) 1368–1393.
[10] G. Fann, G. Beylkin, R.J. Harrison, K.E. Jordan, Singular operators in multiwavelet bases, IBM J. Res. Dev. 48 (2) (2004) 161–171.
[11] M. Frigo, S.G. Johnson, The design and implementation of FFTW3, vol. 93, special issue on "Program Generation, Optimization, and Platform Adaptation", 2005, pp. 216–231.
[12] J.-Y. Lee, L. Greengard, The type 3 nonuniform FFT and its applications, J. Comput. Phys. 206 (1) (2005) 1–5.