CG-FFT for Nonuniform Inverse Fast Fourier Transforms (NU-IFFT's)

Xue Yuan Tang and Qing Huo Liu
Klipsch School of Electrical and Computer Engineering
New Mexico State University
Las Cruces, NM 88003

I. Introduction

Fast Fourier Transform (FFT) [1] has become a popular computational tool in the study of physics and engineering since its development in 1960's. The condition of using FFT algorithm is that the data acquisition must be equispaced. However, in some cases, the data acquisition is not uniformly spaced. Dutt and Rokhlin [2] and Beylkin [3] presented some algorithms for the problems of FFT for nonuniform data. Their idea in solving these problems by a regular FFT is to approximate a function $F(x) = e^{-bx^2}e^{icx}$, by a small number of uniform points on the unit circle. Recently, a more accurate algorithm was developed by using the regular Fourier matrices for the nonuniform forward FFT (NUFFT) [5, 6]. Instead of $F(x)$, it uses $F(j) = s_j e^{i2\pi cj/N}$, where $j = -N/2, ..., N/2-1$, and s_j are chosen to minimize the approximation error. In this work, we use the conjugate-gradient method and regular FFT (CG-FFT) together with the NUFFT algorithm to develop an accurate algorithm for nonuniform inverse fast Fourier transform (NU-IFFT).

II. Formulation

Our goal is to develop fast algorithms to find the forward and inverse solutions for the following two problems:

$$f_j = \sum_{k=0}^{N-1} \alpha_k e^{i\omega_k 2\pi j/N}, \quad \text{for } \omega_k \in [-N/2, N/2],\ j = -N/2, ..., N/2-1 \tag{1}$$

$$g_j = \sum_{k=-N/2}^{N/2-1} \beta_k e^{ikx_j}, \qquad \text{for } \quad x_j \in [-\pi, \pi],\ \ j = 0, ..., N-1 \tag{2}$$

Note that unlike the regular FFT, ω_k and x_j in (1) and (2) are nonuniform.

A. The NUFFT Algorithm

The principle of using the regular Fourier matrices [5, 6] to solve (1) and (2) is as follows. For an integer $m \geq 2$, let $w = e^{i2\pi/mN}$, q be an even positive integer, $s_j(j = -N/2, ..., N/2-1)$ be positive numbers, and c be a real number. Here we approximate w^{jmc} by interpolation,

$$s_j w^{jmc} = \sum_{k=[mc]-q/2}^{[mc]+q/2} X_{k-[mc]}(c) w^{jk}, \qquad \text{for } \quad j = -N/2, ..., N/2-1 \tag{3}$$

where $X_{k-[mc]}$ are unknowns to be solved. From [5, 6], the least-squares solution is $X(c) = F^{-1}a(c)$, where $F = A^{\dagger}A$, and $A_{jk} = w^{jk}$ $(j = -N/2, ..., N/2-1, k = [mc]-q/2, ..., [mc]+$

This work was supported by Environmental Protection Agency through at PECASE grant CR-825-225-010, and by the National Science Foundation through a CAREER grant ECS-9702195.

0-7803-4478-2/98/\$10.00

$q/2$), and $A^\dagger$ denotes the complex conjugate transpose of A matrix. If we choose the cosine accuracy factors $s_j = \cos\frac{\pi j}{Nm}$, a closed-form solution can be found for $a_k(c)$

$$a_k(c) = i \sum_{\gamma=-1,1} \frac{\sin[\frac{\pi}{2m}(2k-\gamma-q-2(mc-[mc])]}{1-e^{i\frac{\pi}{Nm}(2(mc-[mc])+q-2k+\gamma)}} \tag{4}$$

The NUFFT algorithm for (1) has been given in [5, 6]. By modifying it, we have the following NUFFT algorithm for (2):

1) Compute $X_l(x_j N/2\pi)$ by $X(c) = F^{-1}a(c)$, for $l = 0, ..., q$ and $k = 0, ..., N-1$;
2) Calculate Fourier coefficients $u_k = \beta_k s_k^{-1}$;
3) Use uniform FFT to evaluate

$$U_l = \sum_{k=-N/2}^{N/2-1} u_k e^{i2\pi kl/mN}, \qquad \text{for } l = -mN/2, ..., mN/2-1;$$

4) Scale the values to obtain the approximate NUFFT

$$\tilde{g}_j = \sum_{l=-q/2}^{q/2} X_l(x_j N/2\pi) U_{l+[mNx_j/2\pi]}, \qquad \text{for } \ j = 0, ..., N-1.$$

The asymptotic number of arithmetic operations in this algorithm is $O(mN\log_2 N)$, where $m \ll N$. Usually we choose $m = 2$ and $q = 8$.

B. The NU-IFFT Algorithm

Defining $D_{jk} = e^{ikx_j}$, we rewrite (2) as

$$g_j = \sum_{k=-N/2}^{N/2-1} \beta_k D_{jk}, \qquad \text{for } \ j = 0, ..., N-1 \tag{5}$$

From elementary matrix identities we see that

$$(D^\dagger D)D^{-1} = D^\dagger \ \text{ and } \ D^{-1} = (D^\dagger D)^{-1} D^\dagger \tag{6}$$

Combining (5) and (6), we have

$$\tilde{\beta} = (D^\dagger D)^{-1} D^\dagger g \tag{7}$$

Equation (7) is further divided into two consecutive steps for its solution:

1) Utilize the NUFFT algorithm for (1) to solve both $D^\dagger D = \sum_{k=0}^{N-1} e^{-i(j-l)x_k}$ $(j, l = -N/2, ..., N/2-1)$, and $y_k = \sum_{j=0}^{N-1} D_{kj}^* g_j$ $(k = -N/2, ..., N/2-1)$;
2) Use the following CG-FFT method to solve $\tilde{\beta} = (D^\dagger D)^{-1} y = B^{-1} y$.

In the iterative procedure of the conjugate gradient method for $\tilde{\beta} = B^{-1}y$, $z = By$ is calculated many times and can be expedited by using the regular FFT. Recognizing that

$B_{j,l} = \sum_{k=0}^{N-1} e^{-i(j-l)x_k}$ is a Toeplitz matrix ($j, l = -N/2, ..., 0, ...N/2-1$), and by denoting $B_{j,l}$ as 1-D form $B_{j,l} = b(j-l)$, we obtain

$$z_j = \sum_{l=-N/2}^{N/2-1} b(j-l)y(l), \quad \text{for} \quad j = -N/2, ..., N/2-1 \tag{8}$$

Applying the convolution theorem, (8) can be derived as:

$$z_j = \{\mathrm{FFT}^{-1}[\mathrm{FFT}(b)\mathrm{FFT}(y)]\}_j \tag{9}$$

If k iterations are required for the CG-FFT method to converge to an accurate solution, the total number of complex multiplications is proportional to $O[kN(2\log_2 2N + \log_2 N + 1)]$. Similarly, we can obtain the NU-IFFT algorithm for (1).

III. Numerical Results

We compare Dutt-Rokhlin's algorithm and our algorithm with cosine accuracy factors, and $m = 2$ and $q = 8$. Figures 1 and 2 are the examples of the NUFFT and NU-IFFT algorithms for (2). Figures 1(a) and 1(b) show the real and imaginary parts of input data β_k, respectively. Figure 1(c) shows the spatial distribution of output data g_j. Figure 1(d) displays the absolute error from our NUFFT and that from Dutt-Rokhlin's [2]. Quantitatively, the E_2 and E_∞ defined in [2] are $E_2 = 2.713 \times 10^{-6}$ and $E_\infty = 1.8511 \times 10^{-6}$ for our algorithm, and $E_2 = 5.238 \times 10^{-5}$ and $E_\infty = 3.535 \times 10^{-5}$ for the Dutt-Rokhlin's algorithm. Figure 2(a) shows inverse input data g_j. Figures 2(b) and 2(c) represent the real and imaginary parts of output data β_k for the inversion, respectively. Figure 1(d) indicates the absolute error from our NU-IFFT and that from Dutt-Rokhlin's [2]. Quantitatively, $E_2 = 3.156 \times 10^{-6}$ and $E_\infty = 2.390 \times 10^{-7}$ for our algorithm, and $E_2 = 2.3069 \times 10^{-5}$ and $E_\infty = 1.549 \times 10^{-6}$ for the Dutt-Rokhlin's algorithm. From these results, the accuracy of our algorithms is about one order of magnitude higher than Dutt-Rokhlin's.

IV. Conclusions

Based on the CG-FFT method and the NUFFT algorithm, a new nonuniform inverse fast Fourier transform (NU-IFFT) algorithm is developed for nonuniform data. With a comparable complexity of $O(N\log_2 N)$, this algorithm is much more accurate than the previously reported results since it is optimal in the least squares sense.

References

[1] J. W. Cooley and J. W. Tukey, "Algorithm for the machine computation of complex Fourier series," *Math. Comp.*, vol. 19, pp. 297-301, 1965.

[2] A. Dutt and V. Rokhlin, "Fast Fourier Transform For Nonequispaced Data," *SIAM J. Sci. Comput.*, vol. 14, pp. 1368-1393, 1993.

[3] G. Beylkin, "On Fast Fourier Transform of Functions with Singularities," *Applied and Computational Harmonic Analysis*, vol. 2, pp. 363-382, 1995.

[4] Q. H. Liu and W. C. Chew, "Applications of the conjugate gradient fast Fourier Hankel transform method with an improved fast Hankel transform algorithm," *Radio science*, vol. 29, no. 4, pp. 1009-1022, July-August 1994.

[5] Q. H. Liu and N. Nguyen, "An Accurate Algorithm for Nonuniform Fast Fourier Transforms (NUFFT's)," *IEEE Microwave and Guided Wave Letters*, vol. 8, no. 1, pp. 18-20, January, 1998.

[6] N. Nguyen, and Q. H. Liu, " The regular Fourier matrices and nonuniform fast Fourier transforms," *New Mexico State University Technical Report,* July 1997.

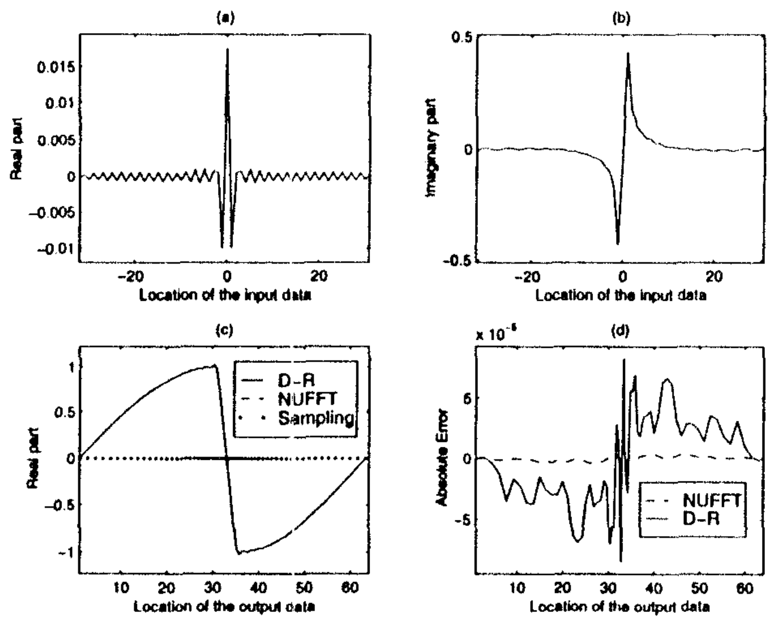

Figure 1. The NUFFT algorithm for (2). (a) Real part of input data (b) Imaginary part of input part. (b) Output data. (d) Absolute error of output data.

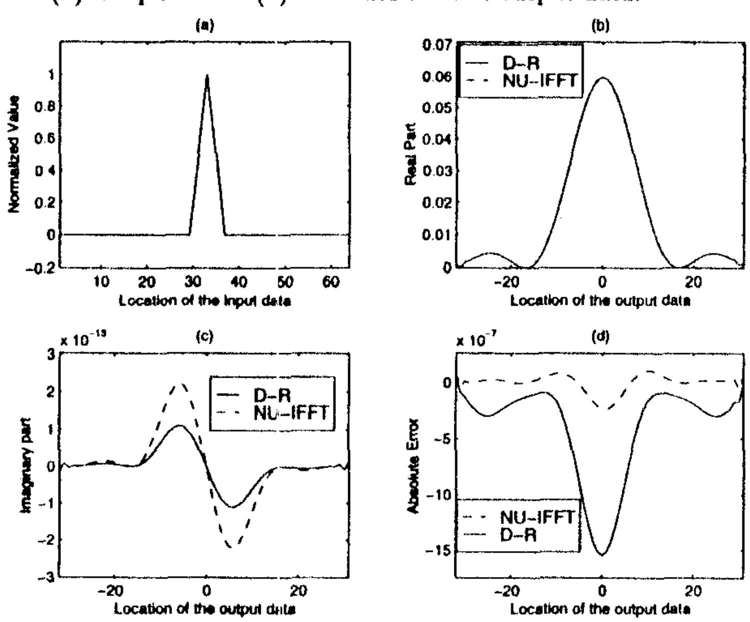

Figure 2. The NU-IFFT algorithm for (2) by CG-FFT method. (a) Input data. (b) Real part of output data. (c) Imaginary part of output part. (d) Absolute error of output data.