Progress In Electromagnetics Research, PIER 92, 33–46, 2009

A NEW EFFICIENT FDTD TIME-TO-FREQUENCY-DOMAIN CONVERSION ALGORITHM

Y.-H. Liu and Q. H. Liu †

Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708, USA

Z.-P. Nie

School of Electronic Engineering
University of Electronic Science and Technology of China
Chengdu, Sichuan 610054, China

Abstract—The time-to-frequency-domain conversion is often required in many applications of the finite-difference time-domain (FDTD) method. This paper presents a new FDTD time-to-frequency-domain conversion algorithm based on the optimization of nonuniform fast Fourier transform (NUFFT) with several redundancy-reduction techniques. The proposed algorithm can perform the FDTD conversion at multiple desired frequencies without the limitation of uniformly spaced frequencies in the fast Fourier transform (FFT). In addition, with a very low storage cost, the algorithm can be much more efficient than other FDTD conversion techniques if a moderate number of frequencies or more are of interest. This algorithm is very useful for some FDTD applications.

1. INTRODUCTION

The finite-difference time-domain (FDTD) method [1] has been extensively used to provide broad-band results for electromagnetic parameters in various applications, such as radar cross section (RCS) calculations [2, 13], analysis of antenna radiation patterns [3], and specific absorption rate (SAR) of the human body [4]. In these

Corresponding author: Q. H. Liu (qhliu@ee.duke.edu).
† Y.-H. Liu is also with School of Electronic Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan 610054, China.

applications, one needs to convert a large number of FDTD sequences to multiple frequencies which may be nonuniformly spaced. Several FDTD time-to-frequency-domain conversion techniques have been developed previously. They include the discrete Fourier transform (DFT) [5], the fast Fourier transform (FFT), the recursive Goertzel algorithm [3] and linear equation methods [6]. Despite the success of these techniques, the problem of converting a large number of FDTD sequences to multiple nonuniformly spaced frequencies is not well solved yet, due to the following three reasons:

1) The DFT and the recursive Goertzel algorithm implement the multiple-frequency conversion by equivalently performing multiple single-frequency conversions. This is a computationally efficient way for a few but not for many frequencies.
2) Linear equation methods need to find the inverse of a matrix. The computational time is very small for a few frequencies, but increases very quickly as the number of frequencies increases [6]. In addition, they have the problem that attends a poorly-conditioned matrix [6] if the conversion over a large number of frequencies is required.
3) The FFT is a very efficient tool for computing the DFT for multiple frequencies. However, the FFT cannot obtain the results exactly located at desired nonuniformly spaced frequencies even if zero-padding is used. In addition, the FFT algorithm requires storage of complete time history of the FDTD sequence. This is very expensive in disk storage for our concerned situation.

Recent developments in nonuniform fast Fourier transform (NUFFT) algorithms [7–10] provide more efficient tools to evaluate the DFT for nonuniformly spaced frequencies. We adopt the basic idea of the NUFFT algorithm presented by Liu and Nguyen in [8] and [9], and introduce several redundancy-reduction techniques to further reduce the requirements in both computation and storage of this NUFFT algorithm for the specific application of this paper. First, we modify the original NUFFT of [8] to have real interpolation coefficients, which halves the computational cost of the interpolation processing. Second, we utilize a real-valued FFT technique [11] to reduce the oversampling FFT computation required for the NUFFT implementation. Third, we develop a segmentation strategy to remove the requirement of storing the complete time histories of FDTD sequences. Consequently, the proposed algorithm is storage and computationally efficient, and is particularly suitable for converting a large number of FDTD sequences to multiple frequencies. The comparison with other FDTD conversion techniques is also given in this paper.

2. FORMULATION AND ALGORITHM

We assume that $\beta_n = \beta(n\Delta_t)$ is an FDTD simulation sequence to be converted, where Δ_t is the sampling period (β_n and Δ_t represent the desampled sequence and time resolution if desampling is used). The problem is to find the following Fourier summation for $k = 0, 1, \ldots, N_f - 1$:

$$g_k = \sum_{n=0}^{N_t-1} \beta_n e^{j2\pi f_k n\Delta_t} \tag{1}$$

where $j = \sqrt{-1}$, and $\{f_k\}$ are frequencies of interest and are not necessarily uniformly spaced. Direct DFT computation requires $O(N_t N_f)$ CPU time complexity. The NUFFT algorithms in [7–10] can be used to speed up the computation of DFT. In particular, the NUFFT algorithm presented in [8] and [9] calculates the least-square error (LSE) interpolation coefficients to improve the accuracy performance. The dominant computational complexity is about $O(\mu N_t \log_2(\mu N_t))$, and the storage complexity is $O(\mu N_t)$, where μ is the FFT oversampling factor. We shall show that for the case of FDTD conversion, this cost can be further reduced by removing redundant computation and storage in the original NUFFT algorithm.

2.1. Modified NUFFT with Real Interpolation Coefficients

Here we modify the original NUFFT algorithm in [8] and [9] such that the resulting interpolation coefficients are real-valued. For simplicity, we assume that the length N of data is odd, otherwise one zero-point can be padded to the original data. We consider the following DFT computation

$$\tilde{g}_k = \sum_{n=-(N-1)/2}^{(N-1)/2} \beta_n e^{j2\pi c_k n/N} \tag{2}$$

where $c_k = \Delta_t f_k N$. The key step of the NUFFT algorithm is to approximate an exponential by linear combination of some other exponentials with uniformly spaced frequencies. That is to find $x_{r-q/2}$ $(r = 0, 1, \ldots, q)$ to satisfy the following condition:

$$s_n w^{n\mu c} = \sum_{k'=[\mu c]-q/2}^{[\mu c]+q/2} x_{k'-[\mu c]}(c) w^{nk'} \tag{3}$$

where $w = e^{j2\pi/N_{\mathrm{FFT}}}$, $\mu = N_{\mathrm{FFT}}/N$ $(\mu \geq 1)$, and $[\mu c]$ denotes the integer nearest to μc. The above condition should be satisfied for

$n = (-N+1)/2, (-N+3)/2, \ldots, (N-1)/2$. It can be proved that if s_n is conjugate-symmetric, the least-squares (LS) solution $\mathbf{x}(c)$ to the above equations are real-valued (the proof is given in Appendix A). This is different from the case of the original NUFFT algorithm in [8] and [9] where the LS interpolation coefficients are always complex-valued.

In practice, $\mathbf{x}(c_k)$ $(k = 0, 1, \ldots, N_f - 1)$ can be calculated by

$$\mathbf{x}(c_k) = \mathbf{F}^{-1}\mathbf{a}(c_k) \tag{4}$$

where $\mathbf{F}$ is a $(q+1) \times (q+1)$ Toeplitz matrix

$$F_{i_1 i_2} = \begin{cases} N, & i_1 = i_2 \\ \dfrac{w^{(i_2-i_1)/2}\left[w^{(i_1-i_2)N/2} - w^{(i_2-i_1)N/2}\right]}{1 - w^{(i_2-i_1)}}, & i_1 \neq i_2 \end{cases} \tag{5}$$

and

$$a_r(c_k) = \sum_{n=-(N-1)/2}^{(N-1)/2} s_n e^{j\frac{2\pi n}{N_{\mathrm{FFT}}}(\mu c_k - [\mu c_k] + q/2 - r)}. \tag{6}$$

In particular, for an accuracy factor $s_n = \cos\frac{\pi n}{N_{\mathrm{FFT}}}$,

$$a_r(c_k) = -j \sum_{\gamma=-1,1} \frac{e^{j\phi_k/N}}{1 - e^{j2\phi_k/N}} \sin\phi_k \tag{7}$$

where $\phi_k = \frac{\pi}{2\mu}(2\mu c_k - 2[\mu c_k] + q - 2r + \gamma)$. Since the cosine accuracy factor is symmetric, $\mathbf{x}(c_k)$ is theoretically real. Numerical error in the evaluation of (4), (5) and (7) results in an imaginary part that is usually more than ten orders of magnitude smaller than the real part. We can use only the real part of $\mathbf{x}(c_k)$ for interpolating FFT results onto the desired frequency-domain locations without decreasing the accuracy. This halves storage and computational requirements of the interpolation processing.

The modified NUFFT algorithm is described as follows:

1) Scale the input data by the accuracy factor $\beta_n \leftarrow s_n^{-1}\beta_n$.
2) Apply FFT to evaluate

$$T_k = \sum_{n=-N_{\mathrm{FFT}}/2}^{N_{\mathrm{FFT}}/2-1} \beta_n e^{j2\pi nk/N_{\mathrm{FFT}}}. \tag{8}$$

3) Calculate $\tilde{g}_k = \sum_{r=0}^{q} \mathrm{Re}[x_{r-q/2}(k)] T_{[\mu c_k]+r-q/2}$.

Usually, the oversampling condition $\mu \geq 1.5$ is required for a good accuracy. Hence, if the radix-2 FFT [12] is used, a reasonable choice is to set $N_{\mathrm{FFT}} = 2^{[\log_2(3N/\sqrt{2})]}$, so that $v = N_{\mathrm{FFT}}/N$ is always between 1.5 and 3, which allows for a good balance between accuracy and efficiency.

In the above procedure, scaling the data costs N real multiplications, (μN)-point FFT requires $0.5\mu N \log_2(\mu N)$ complex multiplications (suppose the radix-2 algorithm [12] is used), and the interpolation processing requires $2(q+1)N_f$ real multiplications. If we implement each complex multiplication by four real multiplications and two additions, the total number of real multiplications for this modified NUFFT algorithm is about $2\mu N \log_2(\mu N) + N + 2(q+1)N_f$ (this excludes the precomputation cost). Compared with the original NUFFT algorithm in [8], this algorithm saves $2(q+1)N_f$ multiplications.

2.2. Real-Valued FFT for Real Data

The FDTD data are real-valued. Utilizing this information can halve the computational cost of the FFT of (8). A number of techniques for efficiently computing the FFT of a real-valued sequence have been presented by Sorensen et al. in [11] and other researchers. Among them, a simple technique is to use the symmetry of the FFT to transform two real-valued sequences simultaneously by computing one complex-FFT. Thus, transforming two real-valued sequences costs only the same as one complex-FFT. More details about this technique can be found in [11]. We incorporate this technique into the modified NUFFT algorithm for reducing the computational cost of (8). In our concerned situation, a large number of FDTD conversions at different locations are required. We can perform time-to-frequency-domain conversions for two FDTD sequences simultaneously. The cost per computation of (8) is halved. Therefore, the total number of multiplications per modified NUFFT is reduced to $\mu N \log_2(\mu N) + N + 2(q+1)N_f$ in this situation.

2.3. Segmentation Strategy

Here we develop a segmentation strategy to remove the requirement of storing complete time histories of FDTD sequences. This segmentation strategy is to break the FDTD sequence into multiple small segments and perform multiple small-size NUFFTs in instead of one N_t-point NUFFT. Assume that N_s (an odd number) is the length of data per segment, and $L = \mathrm{ceil}[N_t/N_s]$. Thus, we can extend the original sequence β_n by padding few zeros to an (LN_s)-length sequence and

split this sequence into L segments of length of N_s. Let us define

$$\beta_p^{(l)} = \beta_{p+lN_s+(N_s-1)/2} \tag{9}$$

where $l = 0, 1, \ldots, L-1$, and $p = (-N_s+1)/2, (-N_s+3)/2, \ldots, (N_s-1)/2$. Substituting (9) into (1), we have

$$g_k = \sum_{l=0}^{L-1} \sum_{p=-(N_s-1)/2}^{(N_s-1)/2} \beta_p^{(l)} e^{j2\pi f_k \Delta_t [p+lN_s+(N_s-1)/2]}. \tag{10}$$

Defining

$$c_k = \Delta_t f_k N_s \tag{11}$$

and

$$g_k^{(l)} = \sum_{p=-(N_s-1)/2}^{(N_s-1)/2} \beta_p^{(l)} e^{j2\pi c_k p/N_s} \tag{12}$$

we rewrite (10) as

$$g_k = e^{j\pi c_k (N_s-1)/N_s} \sum_{l=0}^{L-1} g_k^{(l)} z_k^l \tag{13}$$

where $z_k = e^{j2\pi c_k}$. Clearly, (12) has the same form as (2). Hence, $g_k^{(l)}$ can be calculated efficiently by performing N_s-point modified NUFFT. The final results g_k are just summations of all the $g_k^{(l)}$ with weight of z_k^l.

This segmentation strategy can reduce the storage requirement from $O(\mu N_t)$ to $O(\mu N_s)$. In addition, since usually $N_f \ll N_t$ in the FDTD conversion situation [5], this segmentation can also reduce the computational requirement by choosing a suitable N_s.

2.4. The Proposed Conversion Algorithm

We incorporate the modified NUFFT algorithm and the real-valued FFT technique into the above segmentation scheme, and construct an efficient way of implementing the FDTD time-to-frequency-domain conversion. The proposed algorithm is described below:

1) Set the parameters N_s and N_{FFT} (or $\mu = N_{\mathrm{FFT}}/N_s$).
2) Precompute the inverse of cosine accuracy factor $s_p^{-1} = \sec\left(\frac{\pi p}{N_{\mathrm{FFT}}}\right)$ for $p = (-N_s+1)/2, (-N_s+3)/2, \ldots, (N_s-1)/2$.

3) Precompute $x_{r-q/2}(c_k)$ by replacing N with N_s in (4), (5) and (7), where $r = 0, 1, \ldots, q$ and $k = 0, 1, \ldots, N_f - 1$. This requires $O((q+1)^2 N_f)$ complex multiplications.
4) Initialize $\hat{g}_k = 0$ for $k = 0, 1, \ldots, N_f - 1$, and for $l = 0, 1, \ldots, L-1$:
   (i) Scale FDTD data by $\beta_p^{(l)} \leftarrow s_p^{-1} \beta_p^{(l)}$. This costs LN_s multiplications for L loops.
   (ii) Apply the real-valued FFT to evaluate

$$T_k = \sum_{p=-N_{\mathrm{FFT}}/2}^{N_{\mathrm{FFT}}/2-1} \beta_p^{(l)} e^{j2\pi pk/N_{\mathrm{FFT}}}.$$

   This costs about $\mu L N_s \log_2(\mu N_s)$ multiplications.
   (iii) $\hat{g}_k \leftarrow z_k^{-1} \hat{g}_k + \sum_{r=0}^{q} \mathrm{Re}[x_{r-q/2}(c_k)] T_{[\mu c_k]+r-q/2}$. This costs $2(q+1)LN_f$ real multiplications and $(L-1)N_f$ complex multiplications.
5) Finally $\hat{g}_k \leftarrow e^{j\pi c_k[(2L-1)N_s-1]/N_s} \hat{g}_k$. This costs N_f complex multiplications.

Since $q, N_f \ll N_t$, the time cost of $O((q+1)^2 N_f)$ for the precomputation is very small. Especially for our concerned situation where a large number of conversions over the same frequency locations are required, we just need to run the precomputation once, thus the precomputation time is completely negligible in this situation. Hence, the total number of multiplies is about $L\mu N_s \log_2(\mu N_s) + 2(q+3)LN_f + LN_s$ (suppose we count one complex multiplications by four real multiplications), which is approximately equal to $N_t[\mu \log_2(\mu N_s) + 2(q+3)N_f/N_s + 1]$. Note that in the proposed algorithm, the summation $\hat{g}_k$ can be updated once the FDTD procedure finishes each N_f-step simulation. Hence, the storage requirement (per each field component) of the proposed algorithm is μN_s real values for the real-valued FFT and N_f complex conversion results.

The above complexity analysis also implies that selection of different N_s allows the proposed algorithm to have different storage and computational requirement. In general, the best N_s in term of computational efficiency is usually larger than N_f but smaller than N_t. However, when the storage space is limited, N_s should be chosen such that the storage requirement can be satisfied. An automatic choice for the case of $N_f \geq 3$ is to set $N_{\mathrm{FFT}} = 2^{[\log_2(3N_f/\sqrt{2})]}$ and $N_s = 2[N_{\mathrm{FFT}}/3] - 1$. In this case, N_s is approximately within $0.9N_f \sim 2N_f$, and μ is equal to or marginally larger than 1.5. This choice usually gives a good balance among the computational efficiency, storage requirement and accuracy.

The advantages of the proposed algorithm are as follows.

- This algorithm retains the accuracy performace of the NUFFT presented in [8] and [9]. For arbitrarily spaced frequencies of interest, high accuracy conversions can be obtained through precalculating the LSE interpolation kernel.
- This algorithm not only takes advantage of the computational efficiency of the NUFFT, but also further avoids many redundant calculations for this specific application. Consequently, this algorithm becomes very efficient, especially when a moderate number of frequencies or more are of interest.
- No matter how long the length of FDTD input data is, the proposed algorithm only requires a memory space of $\mu N_s + 2N_f$ real values. The parameter N_s can be determined by users.

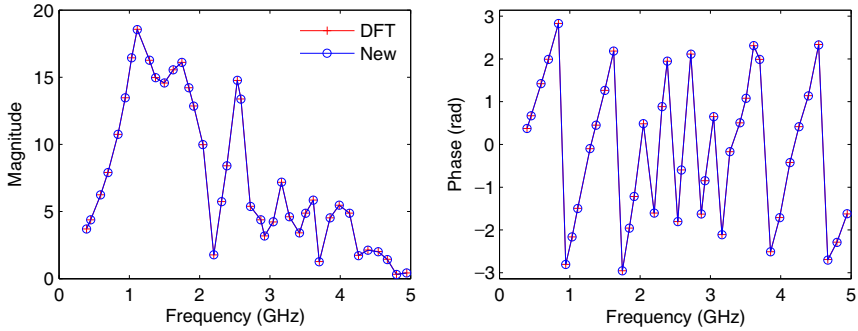
3. PERFORMANCE TEST AND COMPARISON

3.1. Accuracy Performance

We compare results of the proposed algorithm for (1) with those of direct summations by using the DFT. The relative errors is defined as:

$$E_{2,\infty} = \frac{\|\hat{g}_k - g_k\|_{2,\infty}}{\|g_k\|_{2,\infty}} \tag{14}$$

where $\|\cdot\|_2$ and $\|\cdot\|_\infty$ represent the second norm and infinity norm, respectively.



(a) (b)

Figure 1. Time-to-frequency-domain conversion results of the proposed algorithm and the DFT for an x-polarized electric field component at $(0, 0.04, 0.05)$ m of a 0.1 m dielectric cube with $\epsilon_r = 4$. (a) and (b) show amplitude and phase of the spectrum, respectively.

As an example, the time-to-frequency-domain conversion of electric and magnetic fields tangential to the surface of an scattering object is considered; this problem often happens in broad-band RCS calculation from FDTD simulation. A $\hat{x}$-polarized plane wave propagating in $\hat{z}$ direction impinges on a dielectric ($\epsilon_r = 4$) cube of $0.1 \times 0.1 \times 0.1\text{m}^3$ in the vacuum background. We assume that results at 40 nonuniformly spaced frequencies over $(0.3 \sim 5)$ GHz are of interest; those frequencies in the experiment actually are generated by a random number distributed over that frequency band. We choose a Black-Harris window time function with a characteristic frequency of 1.5 GHz as excitation in the FDTD simulation. The FDTD space is $42 \times 42 \times 42$ cells ($34 \times 34 \times 34$ cells inside the cube), and the time resolution is 4.238×10^{-12} s. The simulation takes 5267 time steps to reach convergence for this example. The FDTD sequences are desampled by a factor of 4. The automatic segmentation criterion gives $N_{\text{FFT}} = 64$ and $N_s = 41$ in this case ($N_f = 40$). Fig. 1 shows time-to-frequency-domain conversion results of the proposed algorithm (with $q = 4$) and the DFT as well for the desampled sequence of an x-polarized electric field component at an arbitrarily chosen observation location $(0, 0.04, 0.05)$ m on the cubical surface (origin at cube center). As can be seen, the proposed algorithm achieves nearly the same conversion results as the DFT. The errors are $E_2 = 1.1 \times 10^{-3}$ and $E_\infty = 1.5 \times 10^{-3}$ in this case. Fig. 2 shows the errors E_2 and E_∞ as functions of q and μ (here we fixed $N_{\text{FFT}} = 64$). We can see that the errors decrease approximately exponentially as q or μ increases. Note that the total accuracy of the NUFFT calculation remains almost the same if we change the frequency distribution. From a large amount of numerical tests including different input data and different frequency sampling patterns, we found that using $q = 4$ at $\mu \geq 1.5$ always provides accurate calculations (less than 0.5% error) for all tested cases.

Table 1. The computational and storage requirements for time-to-frequency-domain conversion methods (N_I is the number of FDTD sequences to be converted).

	Multiplications	Storage Locations
DFT	$2N_I N_t N_f$	$2N_I N_f$
Goertzel	$N_I N_t N_f$	$N_I N_f$
NENU-Gauss	$8N_I N_f^3/3$	$4N_I N_f^2$
NENU-SVD	$12N_I N_f^3$	$4N_I N_f^2$
New	$N_I N_t[\mu \log_2(\mu N_s) + 2(q+3)N_f/N_s + 1]$	$N_I(\mu N_s + 2N_f)$

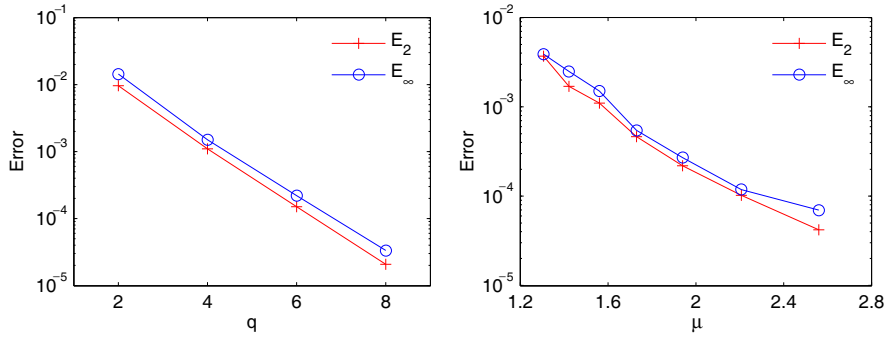


Figure 2. Error E_2 and E_∞ of the proposed algorithm. (a) Effect of changing q at a fixed μ ($N_{\text{FFT}} = 64$, $N_s = 41$, $q = 2, 4, 6, 8$). (b) Effect of changing μ at a fixed q ($N_{\text{FFT}} = 64$, $q = 4$, $N_s = 49, 45, 41, 37, 33, 29, 25$).

3.2. Comparison in the Storage and Computational Requirements

Table 1 shows the storage and computational requirements of the proposed algorithm and some other FDTD conversion techniques including the DFT, the recursive Goertzel algorithm [3], and N-equations N-unknowns methods (NENU) [6] with Gaussian elimination or with the singular value decomposition (SVD). From Table 1, the storage requirements for these methods can be easily compared. If the automatic segmentation criterion is used, the proposed algorithm requires about twice the memory space as the DFT. This is a very low requirement if we compare with the storage of FDTD data themselves. The comparison of the computational requirements can be visualized more clearly if we take some typical parameter values. For example, we do this for the above RCS calculation case where $N_f = 40$, $N_t = 1317$ (the length of desampled sequences), and $N_I = 27744$ (the total number of tangential field components on the cubic surface). Fig. 3 plots the computational requirements of these methods (for the proposed algorithm, two segmentation schemes are used: 1) fixing $N_s = 41$ and $N_{\text{FFT}} = 64$; 2) using the automatic segmentation criterion). As can be seen, the computational requirement of the NENU-Gauss method is the least if N_f is small ($N_f \leq 21$ in this case), but increases very quickly as N_f increases. Actually when N_f gets larger, the NENU method in general has a poorly-conditioned matrix problem and the SVD technique should be used for a good accuracy [6]. This further

lowers the computational efficiency (see results for the NENU-SVD). Unlike the NENU methods, the proposed algorithm always maintains a low computational requirement even if the number of frequencies of interest increases to 100 (or even more). This property is very useful in practical applications because the truly challenging problem for computing the FDTD conversion arises when a large number of nonuniformly spaced frequencies are of interest. This makes the proposed algorithm robust for different FDTD applications.

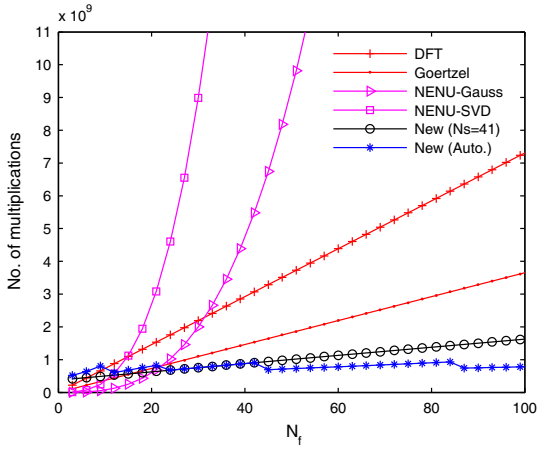


Figure 3. The computational requirements of FDTD time-to-frequency-domain conversion techniques for the parameter values associated with the presented RCS calculation case. Two segmentation schemes are used for the proposed algorithm: 1) fixing $N_s = 41$ and $N_{\text{FFT}} = 64$; 2) using the automatic segmentation criterion.

4. CONCLUSION

We present a new FDTD time-to-frequency-domain conversion algorithm which is based on the modification of the original NUFFT with several redundancy-reduction techniques for improved efficiency. The proposed algorithm can provide accurate calculations for the desired frequencies without the limitation of uniformly spaced frequencies required by the FFT. With a very low storage cost (the required memory space is about twice as that of the DFT), the proposed algorithm always maintains a relatively low computational requirement even if the conversion over a large number of frequencies is required. All these advantages make this algorithm robust for different FDTD application situations.

It is also worth noting that the modified NUFFT algorithm we developed for this specific application can be straightforwardly used to other applications related to the nonuniform DFT. Furthermore, the other redundancy-reduced techniques we used for this paper may be useful for other applications of the NUFFT.

ACKNOWLEDGMENT

This work was supported in part by U.S. National Science Foundation under Grant CCF-0621862, in part by the 111 project of China under Contract No. B07046, in part by the Natural Science Foundation of China under Grant No. 60431010 and No. 60728101, and in part by the Joint Ph.D. Fellowship Program of the China Scholarship Council.

APPENDIX A. PROOF OF REAL-VALUED PROPERTY OF INTERPOLATION COEFFICIENTS

Now assume that $x_{r-q/2}$ $(r = 0, 1, \ldots, q)$ is the least-square error solution of the following equations:

$$s_n w^{n\mu c} = \sum_{k'=[\mu c]-q/2}^{[\mu c]+q/2} x_{k'-[\mu c]}(c) w^{nk'} \tag{A1}$$

for $n = -(N-1)/2, -(N-1)/2+1, \ldots, (N-1)/2$, where N is a odd number. Taking the conjugate of the above equations, we have

$$s_n^* w^{-n\mu c} = \sum_{k'=[\mu c]-q/2}^{[\mu c]+q/2} x_{k'-[\mu c]}^*(c) w^{-nk'}. \tag{A2}$$

And taking the reverse of the index n of (A1) , we have

$$s_{-n} w^{-n\mu c} = \sum_{k'=[\mu c]-q/2}^{[\mu c]+q/2} x_{k'-[\mu c]}(c) w^{-nk'} \tag{A3}$$

By comparing (A2) and (A3), we immediately obtain the following property: if s_n is conjugate-symmetric, i.e., $s_n^* = s_{-n}$, the interpolation coefficients $x_{r-q/2}^* = x_{r-q/2}$. That is, they are real-valued.

REFERENCES

1. Taflove, A. and S. C. Hagness, *Computational Electrodynamics: the Finite-Difference Time-Domain Method*, 2nd edition, Artech House, Norwood, 2000.
2. Furse, C. M., S. P. Mathur, and O. P. Gandhi, "Improvements to the finite-difference time-domain method for calculating the radar cross section of a perfectly conducting target," *IEEE Trans. Microwave Theory Tech.*, Vol. 38, No. 7, 919–927, 1990.
3. Shum, S. M. and K. M. Luk, "An efficient FDTD near-to-far-field transformation for radiation pattern calculation," *Microwave and Optical Technology Letters*, Vol. 20, No. 2, 129–131, 1999.
4. Wang, J., O. Fujiwara, S. Kodera, and S. Watanabe, "FDTD calculation of whole-body average SAR in adult and child models for frequencies from 30 MHz to 3 GHz," *Phys. Med. Biol.*, Vol. 51, 4119–4127, 2006.
5. Furse, C. M. and O. P. Gandhi, "Why the DFT is faster than the FFT for the FDTD time-to-frequency domain ocnversions," *IEEE Trans. Microwave and Guided Wave Lett.*, Vol. 5, No. 10, 326–328, 1995.
6. Furse, C. M., "Faster than Fourier: ultra-efficient time-to-frequency-domain conversions for FDTD simulations," *IEEE Antennas and Propagation Magazine*, Vol. 42, No. 6, 24–34, 2000.
7. Dutt, A. and V. Rokhlin, "Fast Fourier transforms for nonequispaced data," *SIAM J. Sci. Comput.*, Vol. 14, 1368–1393, 1993.
8. Liu, Q. H. and N. Nguyen, "An accurate algorithm for nonuniform fast Fourier transform (NUFFT's)," *IEEE Trans. Microwave and Guided Wave Lett.*, Vol. 8, No. 1, 18–20, 1998.
9. Nguyen, N. and Q. H. Liu, "The regular Fourier matrics and nonuniform fast Fourier transforms," *SIAM J. Sci. Comput.*, Vol. 21, No. 1, 283–293, 1999.
10. Fessler, J. A. and B. P. Sutton, "Nonuniform fast Fourier transform using min-max interpolation," *IEEE Trans. Signal Process.*, Vol. 51, No. 2, 560–574, 2003.
11. Sorensen, H., D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech and Signal Process.*, Vol. 35, No. 6, 849–863, 1987.
12. Oppenheim, A. V. and R.W. Schafer, *Digital Signal Processing*, Prentice-Hall, Englewood Ciffs, 1975.
13. Li, C., G. W. Kattawar, and P. Yang, "A new algorithm to achieve

rapid field convergence in the frequency domain when using FDTD," *Journal of Electromagnetic Waves and Applications*, Vol. 18, No. 6, 797–807, 2004.