Optimized, Parallel Nonuniform Fast Fourier Transform (FFT) for Radiation From Aperiodic Arrays

Amedeo Capozzoli, Claudio Curcio, and Angelo Liseno

Abstract – We deal with the fast evaluation of the radiation operator of aperiodic arrays. We develop a fast numerical algorithm using optimized two-dimensional nonuniform fast Fourier transform (NUFFT) of nonequispaced data type and parallel processing on graphics processing units (GPUs). A fast GPU-implementation of the NUFFT routine is achieved by exploiting a polynomial representation of the involved spatial and spectral windows and by using the Horner polynomial calculation scheme.

1. Introduction

Today, high performance computing, including parallel programming on graphics processing units (GPUs), is indispensable to face computationally burdened design problems requiring fast and accurate simulation tools as direct solvers [1–4]. Electromagnetic examples are the synthesis of devices, as microwave circuits or antennas, exploiting all available degrees of freedom to reach high performance.

Similarly, numerical computing is taking on a major role, especially in applied electromagnetics [5]. Accurate and fast numerical methods, facing the difficult trade-off between accuracy and computational speed, are a basic component of scattering calculations [6], fast antenna analysis [7], and radar cross-section predictions [8].

Here, we benefit from efficient and effective algorithms to reduce the computational complexity and from high performance, massive parallel computing platforms as GPUs to analyze aperiodic arrays [1–3]. Compared with periodic ones, they are indeed raising great interest in the electromagnetic community because they can reach varieties of equivalent amplitude tapers and higher performance thanks to the larger number of degrees of freedom. Nevertheless, reliably designing an aperiodic array is a burdened process requiring iterative and accurate calculations of the radiation from the current structure.

Fortunately, the field radiated by an aperiodic array can be expressed as a two-dimensional (2D) nonuniform discrete Fourier transform (NUDFT) of nonequispaced data (NED) type that can be efficiently and effectively calculated by nonuniform fast Fourier transforms (NUFFTs), with a computational complexity close to standard FFTs. NUFFTs use interpolation windows and the Kaiser–Bessel window (KBW) of [9] has been often considered as state-of-the-art in performance. Recently, an optimized window (OW) exploiting a prolate spheroidal wave function (PSWF) representation that reached accuracy close to machine precision was introduced in [10, 11]. The computational complexity of the OW is the same as the KBW provided to precalculate the involved windows, while the OW outperforms the KBW in accuracy.

However, from the aperiodic array application perspective, the windows calculation must be performed on the fly because the element positions change during synthesis iterations. On the other hand, the OW requires the computation of special functions (PSWFs), so the question arises on whether the use of OW is more computationally burdened or more memory demanding than the KBW.

We show how, by proper polynomial expansion of the OW and by an appropriate computational strategy, the performance of the OW-based 2D NED-NUFFT can be better than the KBW-based one both for ordinary sequential processing on a CPU and for parallel processing on a GPU.

2. NUFFT-Based Field Calculation of Aperiodic Arrays

In this section, we briefly recall how to compute, by a 2D NED-NUFFT, the field radiated by a 2D, planar aperiodic array. To this end, we will focus on the array factor only.

Let $Oxyz$ be the relevant reference system, N the total number of array elements on the $z = 0$ plane, and (x_n, y_n), $n = 1, \ldots, N$, the coordinates of the nth element. The array factor F can be expressed as

$$F(u,v) = \sum_{n=1}^{N} c_n e^{j\beta(ux_n + vy_n)} \tag{1}$$

where c_n is the element coefficient, $u = \sin\theta\cos\phi$, $v = \sin\theta\sin\phi$, $\beta = 2\pi/\lambda$ is the wavenumber, with λ being the wavelength, and (θ, ϕ) indicate the observation direction.

If we are interested in the radiated field at a regular (Cartesian) spectral grid $(u_h, v_k) = (h\Delta u, k\Delta v)$, $h = -N_u/2, \ldots, N_u/2 - 1$, $k = -N_v/2, \ldots, N_v/2 - 1$, then the relevant array factor samples are

$$F_{hk} = \sum_{n=1}^{N} c_n e^{j2\pi\left[\frac{h}{N_u}\tilde{x}_n + \frac{k}{N_v}\tilde{y}_n\right]} \tag{2}$$

Manuscript received 27 December 2022.

Amedeo Capozzoli, Claudio Curcio, and Angelo Liseno are with the Universitá di Napoli Federico II, DIETI, via Claudio 21, I 80125 Napoli, Italy; e-mail: a.capozzoli@unina.it.

where

$$\begin{cases} \tilde{x}_n = \frac{\Delta u N_u}{\lambda} x_n \\ \tilde{y}_n = \frac{\Delta v N_v}{\lambda} y_n \end{cases} \tag{3}$$

and $N_u \times N_v$ are the number of spectral grid points.

Thus, (2) is the expression of a 2D NED-NUDFT [3], and it can be rewritten as

$$\hat{z}_{kl} = \sum_{i=1}^{M} z_i e^{-j2\pi x_i \frac{k}{N_1}} e^{-j2\pi y_i \frac{l}{N_2}} \quad \begin{cases} k = -N_1, \ldots, N_1 - 1 \\ l = -N_2, \ldots, N_2 - 1 \end{cases} \tag{4}$$

where z_i represents the sequence to be transformed, $\hat{z}_{kl}$ the transformed sequence, and (x_i, y_i) the nonuniform sampling points of the input sequence. Therefore, (4) can be efficiently and effectively computed by 2D NED-NUFFTs.

The NUFFT approach consists of properly interpolating the "nonuniformly sampled" exponentials $\exp(-j2\pi x_i k/N_1)$ and $\exp(-j2\pi y_i l/N_2)$ in (4) by "uniformly sampled" exponentials $\exp(-j2\pi mk/(cN_1))$ and $\exp(-j2\pi nl/(cN_2))$, according to the Poisson summation formula [10, 11]

$$e^{-jx\xi} = \frac{(2\pi)^{-1/2}}{\phi(\xi)} \sum_{m\in\mathbb{Z}} \hat{\phi}(x-m) e^{-jm\xi}, \ |\xi| \le \pi/c, \ \forall x \in \mathbb{R} \tag{5}$$

where $\phi(\xi)$ is a window having compact support $(-\alpha', \alpha')$, $\alpha' = \pi\ (2 - 1/c)$, c is an oversampling factor, and $\hat{\phi}(x)$ is the Fourier transform of $\phi(\xi)$, namely,

$$\hat{\phi}(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{jx\xi} \phi(\xi) d\xi \tag{6}$$

To be of interest, ϕ must be essentially bounded to $[-K, K]$, with integer K, to ensure a finite summation in (5) [11]. The choice of ϕ is crucial for achieving a highly performing NUFFT in terms of the trade-off between accuracy and computational burden.

3. Optimized Window

In our optimized approach, for fixed values of K and c, ϕ is chosen as the window "minimizing" the representation error of $e^{-j\xi x}$ in (5) when the summation on the right-hand side is truncated to a finite number of $2K + 1$ terms. Because of its properties, ϕ must belong to the space of PSWFs corresponding to the singular values of the truncated Fourier transform operator before the "knee" [10, 11]. In other words

$$\phi(\xi) = \sum_{t=0}^{T} \gamma_t \Psi_{2t}[w, \xi], \quad |\xi| \le (2 - 1/c)\pi \tag{7}$$

where Ψ_k denotes the kth PSWF with space–bandwidth product (SBP) w, T the number of expansion functions, and γ_t the unknown expansion coefficients. Only the even PSWFs are considered in (7) because of the symmetry of ϕ. To enforce the desired properties, the SBP is chosen as $w = \chi\alpha' K$, where χ is a coefficient that is subject to optimization. The number of retained expansion PSWFs is chosen as, where $2T \le \lfloor (2w/\pi) \rceil$ is the largest integer smaller than v.

The expansion coefficients of the PSWFs are determined to optimize the functional [10, 11], and the results of the optimization are reported in [12] for four relevant cases: case A, $c = 1.5$ and $K = 3$; case B, $c = 2$ and $K = 3$; case C, $c = 1.5$ and $K = 6$; and case D, $c = 2$ and $K = 6$.

4. Computation of the OW and of Its Fourier Transform

We now illustrate the efficient calculation of both ϕ and $\hat{\phi}$.

4.1 Computation of

Thanks to [13], the PSWFs are computed using a Legendre polynomial expansion:

$$\Psi_j[w, \xi] = \sum_{k=0}^{K_{\text{leg}}} \beta_k^{(j)} \bar{P}_k\left(\frac{\xi}{\alpha'}\right) \tag{8}$$

where $\alpha' = (2 - 1/c)\pi$, $\bar{P}_k(s) = \sqrt{k + 0.5} P_k(s)$ is the normalized kth Legendre polynomial and $P_k(s)$ is the kth Legendre polynomial defined by

$$\begin{cases} P_0(s) = 1 \\ P_1(s) = s \\ P_{k+1}(s) = \frac{2k+1}{k+1} P_k(s) - \frac{k}{k+1} P_{k-1}(s) \quad k > 1 \end{cases} \tag{9}$$

and $\beta_k^{(j)}$ superalgebrically decay and enable truncating the series at a certain index K_{leg}. The choice of K_{leg} and the determination of $\beta_k^{(j)}$ depend on c and K and are discussed in [13].

Thus, (7) and (8) define an expansion of ϕ in normalized Legendre polynomials:

$$\phi(\xi) = \sum_{k=0}^{K_{\text{leg}}} \underbrace{\left\{ \sum_{t=0}^{T} \gamma_t \beta_k^{(2t)} \right\}}_{\bar{\beta}_k} \bar{P}_k\left(\frac{\xi}{\alpha'}\right) \tag{10}$$

With $\bar{P}_k(s)$ as a polynomial of degree K_{leg}, it is convenient to express, starting from the values of both γ_t and $\beta_k^{(2t)}$, the window $\phi(\xi)$ as a polynomial of degree K_{leg} as

$$\phi(\xi) = \sum_{k=0}^{K_{\text{leg}}} \delta_k \left(\frac{c\xi}{\pi}\right)^k \tag{11}$$

where the normalization of ξ by π/c is due to ϕ being of interest in only $(-\pi/c, \pi/c)$. The coefficients δ_k are reported in [14] for the four considered cases A to D.

4.2 Computation of $\hat{\phi}$

The Fourier transform of ϕ can be obtained by using the representation (10) and by exploiting the identity (see formula 18.17.19 of [15])

$$\frac{1}{j^k}\sqrt{\frac{2\pi}{x}}J_{k+0.5}(x) = \int_{-1}^{1} P_k(\xi)e^{-jx\xi}dx \tag{12}$$

where $J_{k+0.5}$ is the Bessel function of the first kind with fractional index $k + 0.5$ [16]. Consequently,

$$\hat{\phi}(x) = \sum_{k=0}^{K_{\text{leg}}} \bar{\beta}_k \frac{\alpha'}{\pi j^k} j_k(\alpha' x) \tag{13}$$

where $j_k(y) = \sqrt{\pi/(2y)}J_{k+05(y)}$ is the kth spherical Bessel function [16].

From (13), the efficient calculation of $\hat{\phi}$ is submitted to the efficient computation of the spherical Bessel functions. To face this issue, we notice that the argument of $j_k(\alpha' x)$ ranges in $(-(K+0.5)(2-1/c), (K+0.5)(2-1/c))$ [1, 11]. Considering the typical ways the Bessel functions are approximated in practice, such a range can be "wide," putting the approximation schemes to the test. For example, in case A, the range amounts to about (–30.6, 30.6), which strongly affects the computational difficulty.

A largely used technique, especially adopted in library routines, for the computation of j_k exploits continued fractions [17]. They have the advantage of expressing the kth spherical Bessel function uniquely starting from the k to (k–1)th one. Moreover, as the index k increases, the number of terms in the continued fractions necessary for convergence decreases and increases accuracy for k large. However, the number of terms involved in the continued fractions keeps high due to the large range of possible arguments. Moreover, the use of the continued fractions does not match the CUDA programming model, even though the Milne–Thomson theorem offers a way to reduce computational complexity and to perform a parallel implementation [18].

Other possibilities for the computation of j_k are series representations as ascending series (see formula 10.1.2 of [16]), representations with elementary functions (see formula 10.1.8 of [16]), and polynomial expansions [19]. Unfortunately, the large range of arguments requires a large number of series terms whose coefficients become more and more difficult to be calculated within a standard double-precision arithmetic being either extremely low or extremely large.

For our purposes, the most accurate and efficient approach has revealed performing a polynomial interpolation of $\hat{\phi}$ using Legendre polynomials. Analogously to the case of ϕ, the polynomial expansion of $\hat{\phi}$ can be expressed as

$$\hat{\phi}(x) = \sum_{k=0}^{K} \lambda_k \left(\frac{x}{\bar{x}}\right)^k \tag{14}$$

where $(-\bar{x}, \bar{x})$ is the interval in which the interpolation was performed. To avoid boundary inaccuracies, $(-\bar{x}, \bar{x})$ is chosen slightly larger than $(-(K+0.5)\alpha', (K+0.5)\alpha')$ so that the representation in (14) is reliable in the whole interval of interest. In the examined cases, $\bar{x}$ has been chosen equal to $(K+1)\alpha'$. The coefficients λ_k are reported in [14] for the four considered cases.

5. Parallel Implementation of the Approach

The 2D NED-NUFFT has been implemented in CUDA 11.2 language by using both the Kaiser–Bessel window and the developed OW. Concerning the evaluation of ϕ for the approach in [9], note that the Bessel function I_0 is computed in CUDA according to a modified Blair's approach [20], which is based on a rational Chebyshev approximation of I_0. Furthermore, the spectral window $\hat{\phi}$ for [9] involving the sinh function is computed thanks to the use of a table lookup, followed by an interpolation step [21].

Opposite to that, in the developed approach, both the computation of ϕ and of $\hat{\phi}$ are based on the evaluation of polynomials. Having cast the calculation as a polynomial series has the advantage of enabling exploiting the Horner scheme [22]. To illustrate it, we consider the case of a third-order polynomial

$$p(x) = v_3x^3 + v_2x^2 + v_1x + v_0 \tag{15}$$

According to the Horner scheme, such a polynomial can be calculated as

$$(x) = \underbrace{\underbrace{\underbrace{\left[\underbrace{(v_3x + v_2)}_{\text{FMA}(v_3,x,v_2)}x + v_1\right]}_{\text{FMA}(\text{FMA}(v_3,x,v_2),x,v_1)}x + v_0}_{\text{FMA}(\text{FMA}(\text{FMA}(v_3,x,v_2),x,v_1),x,v_0)}}$$

$$\text{FMA}(\text{FMA}(\text{FMA}(v_3, x, v_2), x, v_1), x, v_0) \tag{16}$$

In (16), FMA means *fused multiply add* and represents operations that advantageously combine, in CUDA, multiply–add operations into a single instruction per clock cycle.

6. Numerical Results

The developed approach has been implemented in both a sequential and parallel code. For the sequential approach, the Python language has been used. The parallel implementation, on the other side, has been achieved by kernel functions written in CUDA 11.2 language and called under Python 3.9 thanks to the functionalities of the PyCUDA 2022.2.2 library. The code has been executed on an Intel Xeon 2.00 GHz equipped with a Tesla T4 GPU.

In Figure 1, results regarding the sequential approach are presented. In particular, we show the speedup of our version against the KBW for different values of the NUFFT size, having assumed $N_1 \times N_2 =$

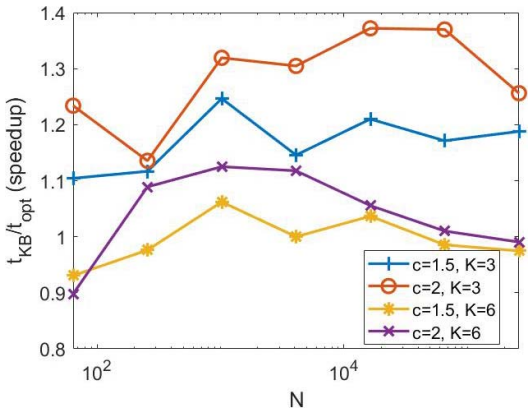

Figure 1. Speedup of the optimized Python version against KBW.

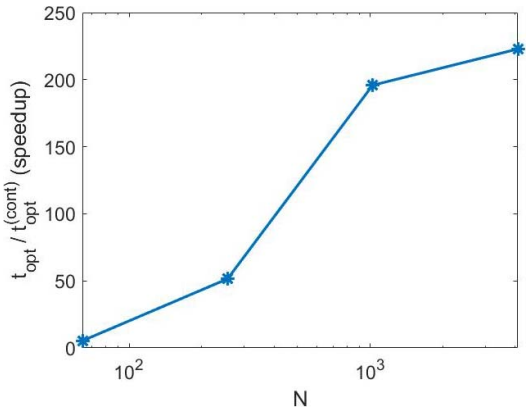

Figure 3. Speedup of the optimized CUDA version against a version based on the Milne–Thomson theorem for computing $\hat{\phi}$.

M. As can be seen, the optimized version is generally better performing or it has performance very similar to the KBW case, yet being significantly more accurate [11].

An analogous result is achieved for the parallel case, as shown in Figure 2. In this case, the optimized version resulted generally faster than the KBW one. Note that the parallel scalability of the proposed algorithm has been studied in [1]. The focus of Figure 2 is on a point not yet addressed throughout the literature, namely, a comparison between two NUFFT approaches sharing the same underlying structure but using different window designs.

To further valorize the devised scheme, we finally show, in Figure 3, a comparison with a version based on the use of the Milne–Thomson theorem for the calculation of the spectral window $\hat{\phi}$. The notable increase in performance of the developed scheme can be appreciated.

7. Conclusions and Future Developments

The presented optimized 2D NED-NUFFT shows improved accuracy as compared with [9], as pointed out in [11]. We have shown here how, by properly taking care of the numerical evaluation of the involved windows, it reaches better performance than [9] for the parallel case and similar performance to [9] for the sequential case, yet significantly more accurate [11].

The results of this article are of interest also for conformal arrays, thanks to the scheme in [23] and for reflectarray antennas. The approach can be extended to nonequispaced results and type-3 NUFFTs [11, 24].

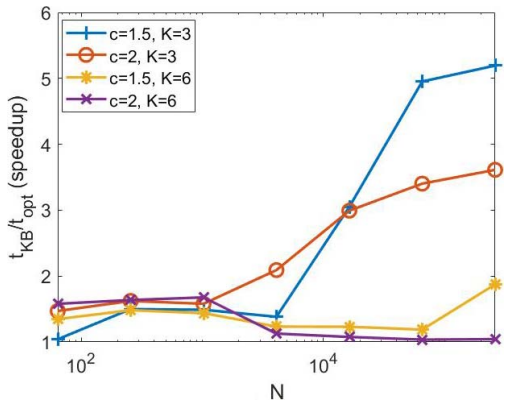

Figure 2. Speedup of the optimized CUDA version against KBW.

8. References

1. A. Capozzoli, C. Curcio, A. Liseno, and G. Toso, "Fast, Phase-Only Synthesis of Aperiodic Reflectarrays Using NUFFTs and CUDA," *Progress in Electromagnetics Research*, **156**, June 2016, pp. 83-103.
2. A. Capozzoli, C. Curcio, and A. Liseno, "CUDA-Based Particle Swarm Optimization in Reflectarray Antenna Synthesis," *Advanced Electromagnetics*, **9**, 2, October 2020, pp. 66-74.
3. A. Capozzoli, C. Curcio, A. Liseno, and G. Toso, "Phase-Only Synthesis of Flat Aperiodic Reflectarrays," *Progress in Electromagnetics Research*, **133**, 2013, pp. 53-89.
4. M. O. Akinsolu, B. Liu, V. Grout, P. I. Lazaridis, M. E. Mognaschi, et al., "A Parallel Surrogate Model Assisted Evolutionary Algorithm for Electromagnetic Design Optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, **3**, 9, April 2019, pp. 93-105.
5. K. Sharma and R. Mittra, "Novel Techniques for Numerically Efficient Solution of Multiscale Problems in Computational Electromagnetics," *International Journal of Numerical Modelling*, **33**, 2, March–April 2020, pp. 1-13.
6. D. P. Xiang and M. M. Botha, "MLFMM-Based, Fast Multiple-Reflection Physical Optics for Large-Scale Electromagnetic Scattering Analysis," *Journal of Computational Physics*, **368**, September 2018, pp. 69-91.
7. H. Bui-Van, J. Abraham, M. Arts, Q. Gueuning, C. Raucy, et al., "Fast and Accurate Simulation Technique for Large Irregular Arrays," *IEEE Transactions on Antennas and Propagation*, **66**, 4, April 2018, pp. 1805-1817.
8. A. Bilal, S. M. Hamza, Z. Taj, and S. Salamat, "Comparison of SBR and MLFMM Techniques for the Computation of RCS of a Fighter Aircraft," *IET Radar, Sonar & Navigation*, **13**, 10, October 2019, pp. 1805-1810.
9. K. Fourmont, "Non-Equispaced Fast Fourier Transforms

With Applications to Tomography," *Journal of Fourier Analysis and Applications*, **9**, 5, September 2003, pp. 431-450.
10. A. Capozzoli, C. Curcio, and A. Liseno, "An Approach to NUFFT Optimization," 22nd Riunione Nazionale di Elettromagnetismo, Cagliary, Italy, September 3–6, 2018, pp. 37-40.
11. A. Capozzoli, C. Curcio, and A. Liseno, "Optimized Non-Uniform FFTs (NUFFTs) and Their Application to Array Factor Computation," *IEEE Transactions on Antennas and Propagation*, **67**, 6, June 2019, 3924-3938.
12. A. Capozzoli, C. Curcio, A. Liseno, "Legendre Coefficients Representing the Optimized Window," https://archive.org/details/LegendreCoefficientsRepresentingTheOptimizedWindow (Accessed 19 December 2022).
13. H. Xiao, V. Rokhlin, and N. Yarvin, "Prolate Spheroidal Wavefunctions, Quadrature and Interpolation," *Inverse Problems*, **17**, 4, August 2001, pp. 805-838.
14. A. Capozzoli, C. Curcio, A. Liseno, "Tables of Coefficients," https://ia601402.us.archive.org/19/items/tables-of-coefficients/Tables_of_coefficients.pdf (Accessed 20 December 2022).
15. F.W.J. Olver, D.W. Lozier, R.F. Boisvert, C.W. Clark (eds), NIST Handbook of Mathematical Functions, Cambridge University Press, Cambridge, 2010, Chapter 18.
16. H. A. Antosiewicz, "Bessel Functions of Fractional Order," in M. Abramowitz and I. A. Stegun (eds.), *Handbook of Mathematical Functions*, Washington D.C., U.S. Government Printing Office, 1972, Chapter 10.
17. W. J. Lentz, "Generating Bessel Functions in Mie Scattering Calculations Using Continued Fractions," *Applied Optics*, **15**, 3, March 1976, pp. 668-671.
18. O. Eğecioğlu, Ç.K. Koç, and J. Rifà i Coma, "Fast Computation of Continued Fractions," *Computers & Mathematics with Applications*, **21**, 2–3, XXX 1991, pp. 167-169.
19. J. J. Wimp, "Polynomial Expansions of Bessel Functions and Some Associated Functions," *Mathematics of Computation*, **16**, 80, October 1962, pp. 446-458.
20. J. M. Blair, "Rational Chebyshev Approximations for the Modified Bessel Functions $I_0(x)$ and $I_1(x)$," *Mathematics of Computation*, **28**, 126, April 1974, pp. 581-583.
21. S. F. Oberman and M. Y. Siu, "A High-Performance Area-Efficient Multifunction Interpolator," 17th IEEE Symposium on Computer Arithmetic, Cape Cod, MA, USA, June 27–29, 2005, pp. 1-8.
22. B. Fine, A. Gaglione, A. Moldenhauer, G. Rosenberger, and D. Spellman, *Algebra and Number Theory*, Berlin, De Gruyter, 2017.
23. A. Capozzoli, C. Curcio, G. D'Elia, A. Liseno, P. Vinetti, "Fast CPU/GPU Pattern Evaluation of Irregular Arrays," *The Applied Computational Electromagnetics Society Journal*, **25**, 4, April 2010, pp. 355-372.
24. A. Capozzoli, C. Curcio, A. Liseno, and A. Riccardi, "Parameter Selection and Accuracy in Type-3 Non-Uniform FFTs Based on Gaussian Gridding," *Progress in Electromagnetics Research*, **142**, October 2013, pp. 743-770.