Progress In Electromagnetics Research, Vol. 156, 83–103, 2016

Fast, Phase-Only Synthesis of Aperiodic Reflectarrays Using NUFFTs and CUDA

Amedeo Capozzoli[1, *], Claudio Curcio[1], Angelo Liseno[1], and Giovanni Toso[2]

Abstract—We deal with one of the computationally most critical steps of the Phase-Only synthesis of aperiodic reflectarrays, namely the fast evaluation of the radiation operator. We present an approach exploiting the use of a fast numerical algorithm using 2D Non-Uniform FFTs (NUFFTs) of NED (Non-Equispaced Data) and NER (Non-Equispaced Results) type and of parallel processing on Graphics Processing Units (GPUs). We extend the approach in K. Fourmont, *J. Fourier Anal. Appl.*, Vol. 9, No. 5, 431–540, 2013 for implementing NUFFT routines to the 2D case and illustrate the parallel strategies to accelerate the approach. In particular, we show how the two levels of parallelism intrinsic in the interpolation step of the 2D NED-NUFFT can be fruitfully exploited by adopting dynamic parallelism, a feature made available in one of the latest architecture of NVIDIA cards. The presented synthesis results show that the introduction of further degrees of freedom (positions) allows improving the performance with respect to periodic reflectarrays. Also, the possibility of adopting aperiodic reflectarrays of reduced number of elements for fixed performance is demonstrated.

1. INTRODUCTION

The application of numerical computing has come to occupy a major role in the field of applied electromagnetism [1]. Indeed, many accurate and fast numerical methods have been developed in the last decades for scattering calculations, fast antenna analysis and RCS predictions, facing the important trade-off between the accuracy of the results and the rapidity of the simulations [2–7].

In the very last years, the cost of computing has decreased significantly, thanks to the exploitation of Graphics Processing Units (GPUs) to speed up the computations. This has come following the introduction, in 2007, of the NVIDIA Compute Unified Device Architecture (CUDA) [8], a powerful and simple-to-use extension of C++, which makes GPU computing accessible to developers not expert in the field of computer graphics.

Parallelism is the future of computing [9] and the interest of the Antennas and Propagation community in topics of high performance computing and, in particular, of parallel programming on GPUs to face computationally burdened problems has been remarkable, as witnessed by [2–7] and by other electromagnetic numerical methods which have benefitted from GPU computing [10–17]. From this starting point, it is clear that the electromagnetic community can take advantage of this technological evolution to employ ever-more sophisticated numerical methods. These are especially required in the framework of iterative design approaches, when the use of fast and accurate simulation tools as direct solvers must be guaranteed and repeated many times. Indeed, to achieve high performance electromagnetic devices, as microwave circuits or antennas, exploiting all the available degrees of freedom of the structure is required. Of course, higher performance is attained at the cost of more

Received 19 February 2016, Accepted 11 May 2016, Scheduled 27 June 2016
* Corresponding author: Amedeo Capozzoli (a.capozzoli@unina.it).
[1] Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione, Università di Napoli Federico II, via Claudio 21, Napoli I 80125, Italy. [2] European Space Agency, Antenna and Sub-Millimeter Wave Section, Kepleerlan 1, PB 299, Noordwijk, AG 2200, The Netherlands.

burdened models and synthesis procedures, based on more refined optimization processes. Such a higher computational complexity, arising, for example, from the use of global optimization approaches, must be properly managed to make the design affordable. Accordingly, properly conceived efficient and effective design strategies are requested.

Efficiency and effectiveness can be pursued along two lines: the use of efficient and effective numerical algorithms capable to reduce the computational complexity and the exploitation of high performance, massively parallel computing platforms as GPUs. These two aspects are not disjoined since the appropriate exploitation of parallel hardware requires the choice of conveniently parallelizable numerical algorithms. From this point of view, a new paradigm emerges in electromagnetic numerical computing. Reaching advanced numerical computations is not anymore limited to devise and exploit numerical tricks, but requires picking new or well assessed numerical approaches from fields other than applied electromagnetics (including, but not limited to, numerical parallel computing), a deep knowledge of the new available computing architectures and a clever exploitation of their features by smart programming practices and profiling tools. Failing to set up proper implementations on such new architectures entails failing to fully exploit their capabilities, dramatically limiting also their performance.

The purpose of this paper is to apply this new paradigm of numerical computing to a case of interest in the analysis and design of a particular class of antennas, namely, reflectarrays.

Microstrip reflectarrays are a special class of antennas combining some of the appealing features

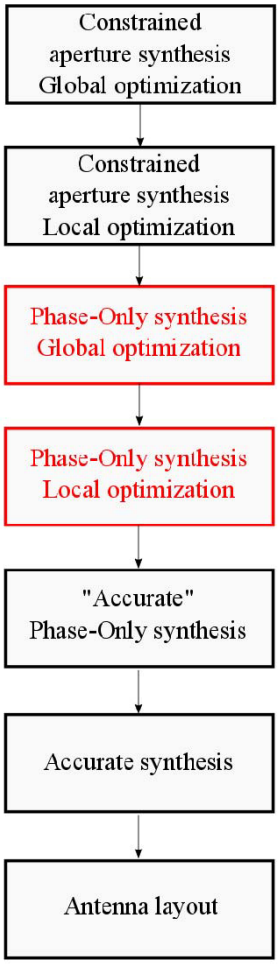

Figure 1. Illustrating the different steps of the whole approach. The steps dealt with in this paper are in red.

of microstrip patch arrays and reflectors [18, 19]. Indeed, they are flexible like arrays, and they do not require cumbersome beamforming networks as reflectors [20, 21]. As compared to periodic ones, aperiodic reflectarrays can reach a wide variety of equivalent amplitude tapers [22–27] and higher performance [24, 25–27] thanks to the larger number of involved degrees of freedom (including the positions of all the reflectarray elements).

In order to guarantee satisfactory reliability and accuracy in the design of aperiodic reflectarrays, the multistage approach devised in [21, 27] is here considered as the synthesis procedure. The idea, whose concept can be extended to other synthesis problems, is simple: employ a rough modeling of the structure to enable robustness against suboptimal solutions at the early stages and then refining, from stage to stage, the model by renouncing to use optimization strategies which are too demanding from the computational point of view. The multistage strategy is summarized in Fig. 1 along with the steps particularly dealt with in this paper (in red), while the details of the whole approach are reported in [21].

In this paper, we highlight and discuss in detail, for the very first time, the computationally most critical step (the evaluation of the radiation operator) of one of the stages of the mentioned approach, namely, the Phase-Only (PO) synthesis stage [21]. In particular, the key synthesis/high performance computing innovations of this paper are the following:

1. Concerning the use of efficient and effective algorithms as mentioned above, a 2D Non-Uniform FFT (NUFFT) algorithm of NED (Non-Equispaced Data) type [28], having a computational complexity comparable to that of a standard FFT, is here employed for the direct evaluation of the field radiated by the aperiodic reflectarray. To this end, the NED-NUFFT approach devised in [28] and therein presented for the 1D case is here properly extended to the 2D case. NUFFT was first introduced in the synthesis of aperiodic arrays in [25, 26] and then adopted in the synthesis of aperiodic reflectarrays [27]. The accuracy of calculating the field radiated by aperiodic arrays has been proven in [12] and the issues arising from the segmentation of the radiating cells in sub-apertures already faced by the Authors in [27].

2. A 2D NUFFT of NER (Non-Equispaced Results) type is adopted for the computation of the analytical functional gradient of the relevant cost functional to be minimized during the synthesis process. To this end, the NER-NUFFT approach devised in [25, 26] and therein presented for the 1D case is here properly extended to the 2D case. The use of a NER-NUFFT-based analytical gradient calculation not only dramatically speeds up the computation, but also improves the convergence properties of the algorithm.

3. Regarding the adoption of high performance, massively parallel computing platforms, we here present the implementation of the reflectarray radiation step as well as of the gradient calculation on GPUs in CUDA language by using a "standard" programming style and an "advanced" programming style, exploiting one of the ultimate features of NVIDIA cards, namely, *dynamic parallelism* [8, 29–31]. Both the NED and NER NUFFTs have been evaluated in terms of accuracy and computational convenience so to convince the Reader of the usefulness of their adoption.

4. Although the use of the *iterated projections* technique proposed in [32] is established for the synthesis of reflectarray antennas [18–20], we show that such an approach lacks the required flexibility to deal with aperiodic reflectarrays on the one hand side and that our synthesis strategy outperforms the iterated projections approach even for the case of periodic reflectarrays on the other hand side.

Since Matlab has become a common platform for technical computing, the whole synthesis procedure has been conveniently implemented as a high-level Matlab script, compiled as a mex file [33, 34].

The paper is organized as follows. In Section 2, the PO radiation model is briefly recalled. In Section 3, the evaluation of the radiation from aperiodic reflectarrays is cast as the evaluation of a 2D NED-NUFFT. Consequently, a 2D extension of the algorithm introduced in [28] is discussed and its very good accuracy is highlighted. Section 4 presents the details of the implementations of the 2D NED-NUFFT by both, a "standard" CUDA approach and by the approach using dynamic parallelism. The numerical results therein contained show how the GPU implementation of the 2D NED-NUFFT is able to significantly improve the computational speed as compared to an optimized sequential approach, when using dynamic parallelism. Section 5 is devoted to provide a short recall of the aperiodic reflectarray synthesis strategy. Furthermore, the analytical calculation of the cost functional gradient by 2D NER-NUFFTs is briefly discussed. The synthesis results are provided in Section 6 and show how an aperiodic

reflectarray is able to outperform a periodic reflectarray in terms of different quality parameters. The conclusions are gathered in Section 7.

2. THE PO RADIATING MODEL

In this section, we shortly illustrate the PO radiative model, of interest in this paper, which is employed in the synthesis algorithm recalled in Section 5. To this end, we consider the planar microstrip reflectarray in Fig. 2.

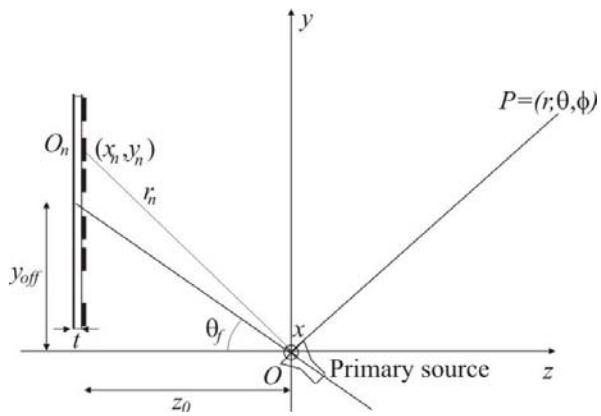

Figure 2. Geometry of the planar, aperiodic microstrip reflectarray.

The reflecting surface is illuminated by a primary source F, located at the origin of the $Oxyz$ reference system and radiating a field $\underline{E}_f$. On assuming each reflectarray patch to be located in the far-zone of F and under the phase-only (PO) model [27], the far-field pattern of the reflectarray can be written as

$$\begin{pmatrix} E_{co} \\ E_{cr} \end{pmatrix}(u,v) = \underline{\underline{Q}}(u,v)\underline{\underline{S}}_0(u,v)\underline{\tilde{E}}_f \sum_{n=1}^{N} \cos^{m_f}(\theta_n)\frac{e^{-j\beta r_n}}{r_n} e^{j\psi_n} e^{j\beta(ux_n+vy_n)} \tag{1}$$

where N is the total number of reflectarray elements, E_{co} and E_{cr} are the co-polar and cross-polar components of the far-field, respectively, $u = \sin\theta\cos\varphi$, $v = \sin\theta\sin\varphi$, $\underline{\underline{S}}_0$ is a term common to the scattering matrices $\underline{\underline{S}}_n$'s of all the elements so that $\underline{\underline{S}}_n(u,v) \cong \underline{\underline{S}}_0(u,v)\exp(j\psi_n)$, ψ_n are the patch control phases, r_n is the distance between the feed phase center and the n-th element, the (x_n, y_n)'s are the aperiodic positions of the patches on the reflecting surface, $\underline{\underline{Q}}$ is the matrix converting the Cartesian components in the $Oxyz$ system of the field scattered by the n-th patch to the co-polar and cross-polar components of the reflectarray far-field, and $\beta = 2\pi/\lambda$ is the wavenumber, λ being the wavelength. Regarding the primary source field, $\underline{\tilde{E}}_f$ accounts for its vector aspects (assuming that the angle subtended by the reflecting surface at O is sufficiently small, see [27]), and a $\cos^{m_f}(\theta_n)$ type pattern, typically sufficient for PO applications, has been assumed (θ_n is defined in Fig. 2).

According to Eq. (1), under the PO model, the field radiated by an aperiodic reflectarray can be written as the product between an "element factor" $\underline{\underline{Q}}(u,v) \cdot \underline{\underline{S}}_0(u,v) \cdot \underline{\tilde{E}}_f$ and an "array factor"

$$F(u,v) = \sum_{n=1}^{N} w_n^{m_f} \frac{e^{-j\beta r_n}}{r_n} e^{j\psi_n} e^{j\beta(ux_n+vy_n)} \tag{2}$$

where $w_n^{m_f} = \cos^{m_f}(\theta_n)$.

Let us observe that reflectarray elements are meant to introduce a phase shift only without scattering amplitude changes and this justifies the use of the PO model at first. In practice, PO is an approximated model which, nevertheless, can be refined by subsequent stages, as suggested by Fig. 1. This has been widely detailed in the synthesis procedure introduced in [21].

The validity of the PO model for the common case of a rectangular patch has been discussed in [27].

Furthermore, the PO approach is able to take into account, although of course in an approximate way, the mutual coupling effects between each element and the surrounding reflectarray elements as well as the specular reflection from the ground plane supporting the patches. This point has been extensively discussed in [35, 36]. As indicated, to include mutual interactions, the scattering behavior of a reflecting element in an aperiodic lattice can be evaluated by embedding the element into a "computational cell" recreating, in an approximate way, the influence of the "environment". The finite cell includes, apart from the current one, a proper number of rings of surrounding patches, trading off accuracy and computational burden. Two cases are possible. In the first case, the representation of the field radiated by the reflectarray through the array factor is retained, as it occurs in this paper, and $\underline{\underline{S}}_0(u,v)$ takes into account for the "environment" "on average". In the second case, the representation of the radiated field through the array factor is dismissed and $\underline{\underline{S}}_0(u,v)$ is replaced by a matrix $\underline{\underline{S}}_n(u,v)$ which depends on the environment and takes into account for it in a more accurate way. This "more accurate" POS possibility is represented by the "Accurate Phase-Only synthesis" stage of Fig. 1.

Finally, numerical simulations have indicated that the radiation of reflectarrays synthesized under the PO model have already a good degree of matching with the radiation of the same reflectarray as evaluated by more "accurate" models [37].

The PO approximation is then in many cases adequate enough and very convenient at a first stage or can be refined by further synthesis stages (the accurate synthesis step of Fig. 1). Whenever very high performance is not required, PO synthesis can provide already satisfactory results.

3. FAST EVALUATION OF THE RADIATED FIELD AND THE 2D NED-NUFFT ALGORITHM

Let us now describe in detail the approach adopted in this paper for the fast evaluation of the "array factor" F in Eq. (2).

Throughout the paper, we will assume to be interested in the calculation of the radiated field at a regular (Cartesian) spectral grid $(u_h, v_k) = (h\Delta u, k\Delta v)$, $h = -N_u/2, \ldots, N_u/2$, $k = -N_v/2, \ldots, N_v/2$, so that the array factor (2) can be recast as

$$F_{hk} = \sum_{n=1}^{N} a_n e^{j2\pi\left[\frac{h}{N_u}\tilde{x}_n + \frac{k}{N_v}\tilde{y}_n\right]} \tag{3}$$

where

$$\begin{cases} \tilde{x}_n = \dfrac{\Delta u N_u}{\lambda} x_n \\ \tilde{y}_n = \dfrac{\Delta v N_v}{\lambda} y_n \\ a_n = w_n^{m_f} \dfrac{e^{-j\beta r_n}}{r_n} e^{j\psi_n} \end{cases} . \tag{4}$$

where $N_u \times N_v$ is the number of spectral grid points.

To save computing time, the design specifications will be enforced, for the test cases reported in Section 6, only on a sub-domain of the visible space. It should be also noticed that the sampling points (x_n, y_n) are written using one index and, at variance with the spectral points, they do not form a tensor product of two coordinates. Obviously, for the CUDA implementations detailed in Section 4, they are stored as one dimensional vectors.

Equation (3) can be recast as a matrix-vector multiplication

$$F_{hk} = \sum_{n=1}^{N} B_{hkn} a_n \tag{5}$$

where

$$B_{hkn} = e^{j2\pi\left[\frac{h}{N_u}\tilde{x}_n + \frac{k}{N_v}\tilde{y}_n\right]}. \tag{6}$$

Equation (5), in turn, is amenable to be evaluated by optimized matrix-vector multiplication routines [12] which, in general, perform as $O(N^2)$ or, in the case of particular symmetries of $\underline{\underline{B}}$, as $O(M \log^5 M)$ [38, 39].

Fortunately, apart from conjugation operations, Eq. (3) is the expression of a 2D Non-Uniform Discrete Fourier Transform (NUDFT) of NED type [28]. As known, NUDFTs can be efficiently and effectively calculated by NUFFT routines, with a computational complexity close to that of standard FFTs. Accordingly, since we are interested here in the clever implementation of the calculation of the "array factor" on GPUs exploiting at the best the potentialities of such computing platforms, an illustration of the 2D NED-NUFFT calculation scheme is now in order. In this paper, the approach devised in [28] for the implementation of the NUFFT algorithm is adopted and properly extended from the 1D case to the 2D case. In the sequel, such an extension, far from being trivial, is detailed by retaining the same symbols and indices used in [28], as far as it is possible. A detailed discussion will be useful to all the Readers aiming to implement their own version of the approach.

The expression of a 2D NED-NUDFT is the following

$$\hat{z}_{kl} = \sum_{i=1}^{M} e^{-j2\pi x_i k/N_1} e^{-j2\pi y_i l/N_2} z_i \quad \begin{cases} k = -N_1, \ldots, N_1 \\ l = -N_2, \ldots, N_2 \end{cases}, \tag{7}$$

where the z_i's represent the sequence to be transformed, the $\hat{z}_{kl}$'s the transformed sequence and the (x_i, y_i)'s the non-uniform sampling points of the input sequence.

The 2D NED-NUFFT exploits the Poisson summation formula expressing each "non-uniformly sampled" exponential $\exp(-j2\pi x_i k/N_1)$ and $\exp(-j2\pi y_i l/N_2)$ into an infinite number of "uniformly sampled" exponentials $\exp(-j2\pi mk/(cN_1))$ and $\exp(-j2\pi nl/(cN_2))$, i.e.,

$$\begin{aligned} e^{-j2\pi x_i k/N_1} e^{-j2\pi y_i l/N_2} &= \frac{(2\pi)^{-1/2}}{\phi(2\pi k/cN_1)} \sum_{m=-\infty}^{\infty} \hat{\phi}(cx_i - m) e^{-j2\pi mk/cN_1} \\ &\quad \frac{(2\pi)^{-1/2}}{\phi(2\pi l/cN_2)} \sum_{n=-\infty}^{\infty} \hat{\phi}(cy_i - n) e^{-j2\pi nl/cN_2} \end{aligned} \tag{8}$$

where ϕ is a proper window function having support in $(-\pi/c,\ \pi/c)$; $\hat{\phi}$ is its Fourier transform; c is an oversampling factor [28]. The Poisson summation formula is exploited by the NUFFT schemes as an interpolator specifically tailored to "non-uniformly sampled" exponentials $\exp(-j2\pi x_i k/N_1)$ and $\exp(-j2\pi y_i l/N_2)$.

Assuming that $\hat{\phi}$ has support in $(-K,\ K)$, then the series in Eq. (8) can be truncated as follows

$$\begin{aligned} e^{-j2\pi x_i k/N_1} e^{-j2\pi y_i l/N_2} &\cong \frac{(2\pi)^{-1/2}}{\phi(2\pi k/cN_1)} \sum_{m=\mu_i-K}^{\mu_i+K} \hat{\phi}(cx_i - m) e^{-j2\pi mk/cN_1} \\ &\quad \frac{(2\pi)^{-1/2}}{\phi(2\pi l/cN_2)} \sum_{n=\nu_i-K}^{\nu_i+K} \hat{\phi}(cy_i - n) e^{-j2\pi nl/cN_2} \end{aligned} \tag{9}$$

where $\mu_i = \mathrm{Int}[cx_i]$, $\nu_i = \mathrm{Int}[cy_i]$ and $\mathrm{Int}[\cdot]$ denotes the integer part.

Let us now define $\phi_k^{(1)} = \phi(2\pi k/(cN_1))$, $\phi_l^{(2)} = \varphi(2\pi l/(cN_2))$, and $\hat{\phi}_{it} = \hat{\phi}(cx_i - (\mu_i + t))/\sqrt{2\pi}$ and $\hat{\psi}_{it} = \hat{\phi}(cy_i - (\nu_i + t))/\sqrt{2\pi}$, where the index t stands for m or n. On substituting Eq. (9) into Eq. (7), then we have

$$\hat{z}_{kl} = \frac{1}{\phi_k^{(1)} \phi_l^{(2)}} \sum_{i=1}^{M} \sum_{m=\mu_i-K}^{\mu_i+K} \sum_{n=\nu_i-K}^{\nu_i+K} z_i \hat{\phi}_{i,m-\mu_i} \hat{\phi}_{i,n-\nu_i} e^{-j2\pi \frac{mk}{cN_1}} e^{-j2\pi \frac{nl}{cN_2}}. \tag{10}$$

If we redefine $\hat{\phi}_{it}$ and $\hat{\psi}_{it}$ so that it is vanishing outside the indices ranges $1 \le i \le M$ and $-K \le t \le K$, then the summations in Eq. (10) can be extended between $-\infty$ and $+\infty$ as

$$\hat{z}_{kl} = \frac{1}{\phi_k^{(1)} \phi_l^{(2)}} \sum_{i=1}^{M} \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} z_i \hat{\phi}_{i,m-\mu_i} \hat{\psi}_{i,n-\nu_i} e^{-j2\pi \frac{mk}{cN_1}} e^{-j2\pi \frac{nl}{cN_2}}. \tag{11}$$

Finally, since $\exp(-j2\pi(m + cN_1)k/(cN_1)) = \exp(-j2\pi mk/(cN_1))$ and $\exp(-j2\pi(n + cN_2)l/(cN_2)) = \exp(-j2\pi nl/(cN_2))$, then the two infinite summations in m and n can be rewritten as four infinite summations as:

$$\hat{z}_{kl} = \underbrace{\frac{1}{\phi_k^1 \phi_l^2} \underbrace{\sum_{m=-cN_1/2}^{cN_1/2} \sum_{n=-cN_2/2}^{cN_2/2} \underbrace{\sum_{i=-\infty}^{+\infty} \sum_{p=-\infty}^{+\infty} \sum_{q=-\infty}^{+\infty} z_i \hat{\phi}_{i,m+cpN_1-\mu_i} \hat{\psi}_{i,n+cqN_2-\nu_i}}_{\text{Interpolation} \to u_{mn}} e^{-j2\pi \frac{mk}{cN_1}} e^{-j2\pi \frac{nl}{cN_2}}}_{\text{Standard FFT on } cN \text{ points}}}_{\text{Scaling and decimation}} \tag{12}$$

Equation (12) highlights the three main steps needed for the implementation of the 2D NED-NUFFT.

The first one is an *interpolation* step needed to calculate

$$u_{mn} = \sum_{i=-\infty}^{+\infty} \sum_{p=-\infty}^{+\infty} \sum_{q=-\infty}^{+\infty} z_i \hat{\phi}_{i,m+cpN_1-\mu_i} \hat{\psi}_{i,n+cqN_2-\nu_i}. \tag{13}$$

Note that nonzero terms of $\hat{\phi}_{i,m+cpN_1-\mu_i}$ and $\hat{\psi}_{i,m+cqN_2-\nu_i}$ occur only for

$$\begin{cases} |m + cpN_1 - \mu_i| \leq K \ll cN_1 \\ |n + cqN_2 - \nu_i| \leq K \ll cN_2 \end{cases}. \tag{14}$$

Accordingly, since typical values for K are 3 or 6, each u_{mn} receives contributions from only a very limited number of interpolation terms. Accordingly, the computational cost of such an interpolation is $M(2K + 1)^2$.

The second step consists in the calculation of the *standard FFT on* $cN_1 \times cN_2$ *points* of the sequence u_{mn} and costs $O((cN)^2 \log(cN))$ operations in the typical case $N_1 = N_2 = N$.

Finally, in the third step, *decimation and scaling*, is performed, requiring $N_1 N_2$ operations.

For N_1, N_2, $M \gg K$, the computational complexity is $O((cN)^2 \log(cN))$, proportional to that of a standard FFT.

It should be finally noticed that the computational cost associated to the calculation of the spatial and spectral windows ϕ and $\hat{\phi}$ depends on the particular choice of the windows functions and on the accuracy required by their calculations. However, typical window functions have a computational cost which is significantly smaller than that needed by the above mentioned three steps and also they must be computed only once for fixed computational grids. Although slightly suboptimal, in this paper we will exploit the convenient choice of ϕ and $\hat{\phi}$ adopted in [28]. In particular, an approximation of the Prolate Spheroidal Wave Function of zero-th order is adopted for ϕ as

$$\phi(\xi) = \begin{cases} I_0\left(K\sqrt{\alpha^2 - \xi^2}\right), & |\xi| \leq \alpha \\ 0, & |\xi| > \alpha \end{cases} \tag{15}$$

where I_0 is the modified Bessel function of zeroth order, and α is a constant slightly smaller than $\pi(2 - 1/c)$. The Fourier transform of ϕ is the Kaiser window function

$$\hat{\phi}(x) = \sqrt{\frac{2}{\pi}} \frac{\sinh\left(\alpha\sqrt{\alpha^2 - x^2}\right)}{\sqrt{\alpha^2 - x^2}}. \tag{16}$$

Let us now finally assess the accuracy of the 2D NED-NUFFT. Throughout the cases in this paper, we have considered $c = 2$ and $K = 6$. Fig. 3 shows the percentage Root Mean Square (RMS) error achieved by the implemented 2D NED-NUFFT against the exact evaluation of the corresponding 2D NUDFT. For that results, the implemented 2D NUFFT algorithm has been run on stochastic complex data with real and imaginary parts uniformly distributed in (0, 1) and assuming input sampling point coordinates x_n and y_n uniformly distributed in $(-M/2, M/2)$. The output sequences have been chosen so that $N_1 N_2 = M^2$, M being the array dimension. As it can be seen, a very high accuracy, close to that dictated by IEEE double precision arithmetics, is reached. This clearly shows that, whenever a 2D NED-NUDFT is to be calculated, choosing a 2D NED-NUFFT is a very convenient choice in terms of accuracy.

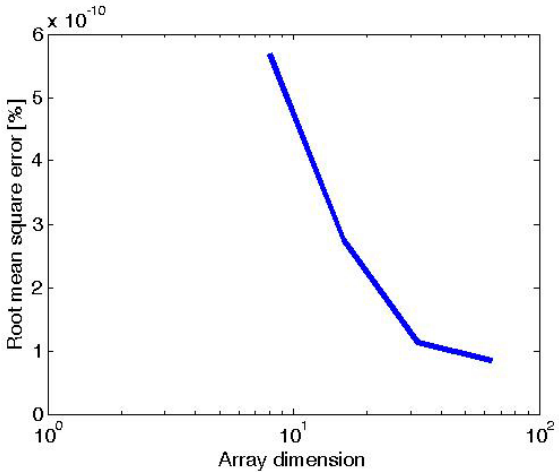

Figure 3. Accuracy of the 2D NED-NUFFT: percentage RMS error against an exact calculation of the corresponding 2D NUDFT.

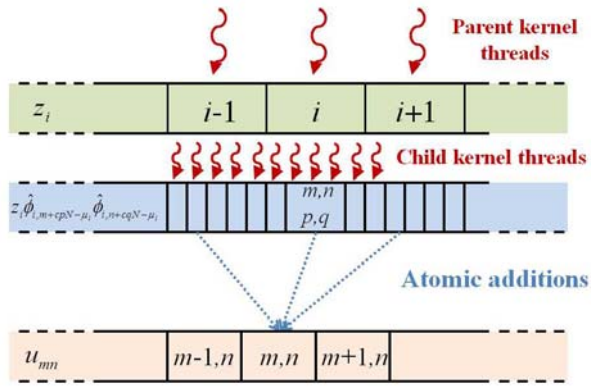

Figure 4. Illustrating the CUDA implementation of the interpolation step based on dynamic parallelism.

4. FAST EVAUATION OF THE RADIATED FIELD BY THE CUDA IMPLEMENTATION OF THE 2D NED-NUFFT ALGORITHM

The 2D NED-NUFFT has been implemented in CUDA language both avoiding dynamic parallelism and by using it. In this section, both, a "standard" CUDA version avoiding dynamic parallelism and a version using dynamic parallelism are sketched, to highlight how the different versions require different implementations of the interpolation scheme in Eq. (13).

4.1. Computation of the Window Functions

Concerning the evaluation of the spatial window function ϕ, it should be noticed that the Bessel function I_0 is not available in CUDA. Consequently, a customized implementation has been set up according to the Blair's approach [40]. It is based on a rational Chebyshev approximation of I_0 and is a good compromise between accuracy and calculation efficiency. The window function ϕ over all the points of interest is calculated in advance by a specific kernel, while its Fourier transform is evaluated on-the-fly in the interpolation step. This step is common to the "standard" CUDA implementation and to the version of the code using dynamic parallelism.

4.2. Interpolation Step for the "Standard" Implementation

The interpolation step appearing in Eq. (13) is the most difficult part of the 2D NED-NUFFT to be parallelized and has been implemented by a specific kernel. The devised computational scheme is illustrated in Fig. 4. More in detail, each thread of the computational grid is assigned to a different input index i and is appointed to accumulate the contribution $z_i\hat{\phi}_{i,m+cpN_1-\mu_i}\hat{\psi}_{i,n+cqN_2-\nu_i}$ to different output locations m and n. The accumulation is performed by each thread through two nested *for* loops and, thanks to conditions (14), each *for* loop spans only a reduced number of cycles $2K+1$. In other words, each input term z_i contributes to only a small number of $(2K+1)\times(2K+1)$ output terms u_{mn} or, saying it differently, each thread is in charge to spread the contribution of z_i to a number of $(2K+1)\times(2K+1)$ output locations mn. It should be noticed that, since different input terms z_i can contribute to the same output value u_{mn}, then race conditions could occur. To avoid them, the accumulation process needs the use of atomic operations (atomicAdd) which serialize the accesses of the different threads to the global memory location of the output value u_{mn}. Although atomic operations imply slowing down the memory accesses, it should be mentioned that they benefit of a very fast implementation for the Kepler architecture.

4.3. CUDA Dynamic Parallelism

Prior to the Kepler architecture, CUDA kernels could only be launched from the host code so that only the CPU was appointed to provide work to the GPU [8].

CUDA dynamic parallelism, introduced with the Kepler architecture, is an extension to the CUDA programming model enabling a CUDA kernel to launch new kernels and thus enabling a GPU to give work to itself. Dynamic parallelism improves the CUDA programming model since:

1. It supports algorithms that dynamically discover new work to prepare and launch kernels without burdening the host with kernel launch overheads.
2. It supports algorithms that involve recursion for applications such as hierarchical grid evaluations; in general, it supports algorithms that do not fit a flat single level of parallelism, but need to be implemented with multiple kernel launches.
3. It simplifies programming as it avoids, besides other things, the tedious coding of returning the command to the host whenever required to launch new operations.

4.4. Interpolation Step for the Version Exploiting Dynamic Parallelism

As mentioned in the previous Subsection, dynamic parallelism potentially improves the implementation of algorithms with nested levels of parallelism, so that it is highly suitable to be used to code the interpolation step in Eq. (13), as it was recognized for the very first time in [29].

The idea behind implementing the interpolation step in Eq. (13) by dynamic parallelism is to parallelize the two nested summations in p and q, thus creating a newer level of parallelism. More in detail, and as in the "standard" CUDA implementation, a different *parent* thread is assigned to a different input index i of Eq. (13). However, opposite to the previous case, each *parent* thread (first level of parallelism) now generates a number of $(2K+1) \times (2K+1)$ *child* threads (second level of parallelism). Each child threads takes now care of a different loop cycle (nested summations) of the previous approach (see Fig. 4). In particular, it is assigned to unique values of i, m, n, p and q and its purpose is to sum a specific value $z_i \hat{\phi}_{i,m+cpN_1-\mu_i} \hat{\psi}_{i,n+cqN_2-\nu_i}$ to a specific u_{mn} of interest. Since multiple child threads can contribute to the same output location mn, the contributions must be summed up again by using atomic operations. The approach is illustrated in Fig. 5.

4.5. FFT and Scaling

The last two steps to be considered are the standard FFT on $cN_1 \times cN_2$ points and the scaling and decimation step (see Eq. (10)). The former can be easily implemented by exploiting the functionalities of the *cuFFT* library. The latter is intrinsically parallel and is implemented by a specific CUDA kernel.

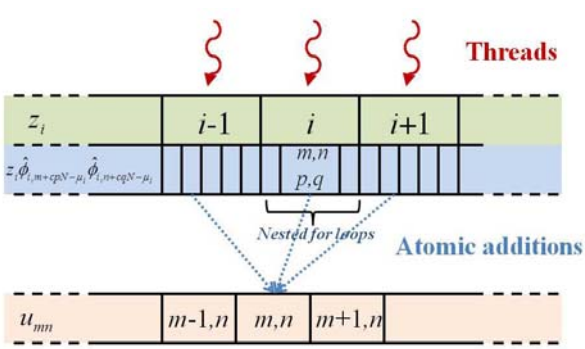

Figure 5. Illustrating the "standard" CUDA implementation of the interpolation step.

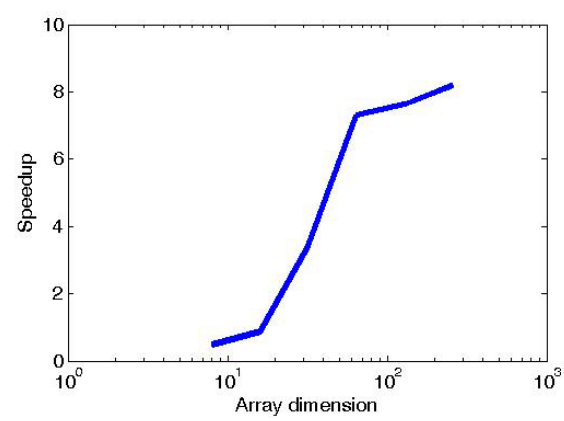

Figure 6. GPU vs CPU speedup for the 2D NED-NUFFT.

4.6. Computational Performance of the CUDA Implementation of the 2D NED-NUFFT

We finally assess the computational performance of the implemented 2D NED-NUFFT CUDA code on one of the latest CUDA architecture, namely, a Kepler K20c card.

All the CUDA codes developed in this work have been deeply optimized for speed by using different strategies. Indeed, mastering the canonical optimization approaches is a key step to maximize performance.

Furthermore, apart from the mentioned GPU implementation, a CPU version of the 2D NED-NUFFT routine has been also implemented and deeply optimized for comparison and run on an Intel Xeon E5-2650 2.00 GHz Eight Core Processor equipped with 256 GB of RAM. The FFT step needed therein has been accomplished by the FFTW library [41]. In addition, a GPU implementation of the 2D NED-NUDFT has been worked out based on the use of cuBLAS routines, as they will mainly serve to provide a reference for the speed up tests.

In Fig. 6, the speedup of the GPU implementation as compared to the CPU one for the 2D-NED NUFFT is reported. As it can be seen from Fig. 6, for values of M corresponding to a 100×100 reflectarray, the GPU vs CPU speedup is about 8. For a value of $M = 64$, the speedup of the CUDA implementation against the cuBLAS based 2D-NUDFT has been around 100. It should be mentioned that using the cuBLAS based 2D-NUDFT has not been possible beyond the mentioned value of M due to the unmanageable memory requirements of optimized and parallel matrix-vector multiplication routines.

5. THE SYNTHESIS STRATEGY

After having provided the details of the analysis approach and of its implementation, in this section we shortly describe the synthesis scheme.

The synthesis problem is formulated as an optimization one [42] and the approach aims at determining the control phases ψ_n and the positions of the patch elements (x_n, y_n). The approach enforces also proper constraints on the element locations which is a crucial point due to the need of guaranteeing adequate inter-element spacings [21, 25–27, 43].

Since the number of control parameters of typical aperiodic reflectarrays can be very large, using global optimizations becomes impractical. Accordingly, the use of local, gradient-based searches would become mandatory. Unfortunately, although computationally efficient, local, gradient-based searches suffer from the local optima issue.

To face this problem, the synthesis is preceded by a pre-synthesis stage, in which the amplitude and phase distributions of a continuous aperture are efficiently (in terms of computational times) and effectively (in terms of robustness against local minima) determined from the design specifications [21, 27]. The determined amplitude distribution is then interpreted as the equivalent tapering by which choosing the initial positions of the subsequent PO stage [25, 26]. The phase distribution, on the other side, dictates the initial command phase distribution of the patches [25, 26].

Using the results of the pre-synthesis stage as a starting point, the unknown control phases and element positions are obtained by minimizing the following objective functional Φ given by

$$\Phi(\underline{\underline{\Psi}}, \underline{\underline{P}}) = \left\| \left| A_{co}(\underline{\underline{\Psi}}, \underline{\underline{P}}) \right|^2 - \mathcal{P}_{U_{co}} \left(\left| A_{co}(\underline{\underline{\Psi}}, \underline{\underline{P}}) \right|^2 \right) \right\|^2, \tag{17}$$

where A_{co} denotes the co-polar component of the relevant radiation operator provided by Eq. (1), namely $A_{co} = E_{co}$, $\underline{\underline{\Psi}}$ and $\underline{\underline{P}}$ are the $1 \times M$ and $2 \times M$ matrices of the command phases and element positions, respectively, $\mathcal{P}_{U_{co}}$ is the metric projector onto the set U_{co} [44] that contains all the power patterns satisfying the specifications for the co-polar component, and $\| \cdot \|$ is a properly chosen norm. The set U_{co} is defined by mask functions (m_{co}, M_{co}) determining upper and lower bounds for $|A_{co}|$ [44] and the metric projector is given by

$$\mathcal{P}_{U_{co}} \left(|A_{co}|^2 \right) = \begin{cases} |M_{co}|^2 & |A_{co}|^2 \geq |M_{co}|^2 \\ |A_{co}|^2 & |m_{co}|^2 \leq |A_{co}|^2 \leq |M_{co}|^2 \\ |m_{co}|^2 & |A_{co}|^2 \leq |m_{co}|^2 \end{cases} . \tag{18}$$

Note that, in Eq. (17) the functional Φ refers only to the co-polar term A_{co} operator is relevant for our purposes due to the retained PO model.

When optimizing the functional in Eq. (17), again to mitigate the false solution issue, the unknowns of the problem are given proper representations enabling a progressive enlargement of the number of parameters to be determined as well as a modulation of the computational complexity of the approach [21, 25–27, 43]. Two steps are so devised. In the first step, few properly chosen polynomials are employed for the command phases and the element positions (Zernike polynomials for the phases and Lagrange polynomials for the positions in this paper) and progressively increased in number [21, 25–27, 43]. It should be noticed that, thanks to such a polynomial representation, global optimization [45, 46] can be employed at this stage to provide an initial guess for the lowest order (Zernike and Lagrange) polynomial coefficients. The so obtained global solution can be subsequently improved by a progressive enlargement of the unknowns space with higher order polynomial coefficients and by local searches. The choice of the global optimizer is briefly discussed in the following Subsection. In the second stage, impulsive functions are adopted to represent the control phases individually [21, 25–27, 43]. In order to keep low the number of unknowns, a proper modal representation is introduced also for the element positions.

Finally, let us stress that a critical computational point for the optimization of functional (17) is the calculation of its gradient. Details on how this point has been dealt with are provided in Subsection 5.2.

5.1. Choice of the Global Optimization Algorithm

Choosing the global optimization algorithm most suited to the synthesis problem dealt with in this paper is not an easy task since "no free lunch" theorems exist for optimization [47]. Generally speaking, effectiveness in finding the global minimum and computational efficiency are the main criteria for selecting the optimizer and they are problem dependent [45, 46].

In this paper, global optimization is based on an iterative multistart procedure which efficiently and effectively combines global and local searches. In particular, the procedure randomly generates starting points for local searches in a "feasible region" to obtain a mapping of the local minima and subsequently recovering the global one. As a multistart algorithm, the Multi-Level Single Linkage (MLSL) method [48, 49] is exploited, which is particularly efficient and effective in avoiding useless local searches and in guaranteeing the convergence to the global optimum with probability one within a finite number of iterations (see Theorem 8, Ref. [48]). Moreover, the total number of local searches started by the MLSL method is finite with probability one even if the sampling continues forever.

5.2. Functional Gradient Calculation

Concerning the derivatives of the cost functional against the phase unknowns, when a representation in terms of Zernike polynomials is adopted, thanks to the significantly limited number of unknowns, the gradient of Φ is evaluated by finite differences. Similarly, and regarding the derivatives of the cost functional against the position unknowns, thanks to the representation in terms of Lagrange polynomials significantly limiting the number of unknowns, the gradient of Φ is evaluated again by finite differences.

Finally, when the derivatives of Φ against the ψ_n's are concerned, they can be effectively evaluated as (see Appendix)

$$\frac{\partial \Phi}{\partial \psi_n} = -4\mathrm{Im}\left\{ f_n e^{j\psi_n} \sum_{h,k} E^*_{co_{hk}} \left(|E_{co_{hk}}|^2 - \mathcal{P}_{U_{co}}\left(|E_{co_{hk}}|^2 \right) \right) g_{hk} e^{j[u_h x_n + v_k x_n]} \right\} \tag{19}$$

where g_{hk} is the co-polar term of $\underline{\underline{Q}}(u,v) \cdot \underline{\underline{S}}_0(u,v) \cdot \underline{\tilde{E}}_f$ and

$$f_n = w_n^{m_f} \frac{e^{-j\beta r_n}}{r_n}. \tag{20}$$

As it can be inferred from Eq. (19), the whole gradient of Φ against the ψ_n's can be calculated by

a 2D NER-NUFFT. Indeed, the expression of a 2D NER-NUFFT is the following:

$$\hat{z}_i = \sum_{k=-N_1/2}^{N_1/2} \sum_{l=-N_2/2}^{N_{2/2}} z_{kl} e^{-j2\pi x_i k/N_1} e^{-j2\pi y_i l/N_2}, \tag{21}$$

where the z_{kl}'s represent the sequence to be transformed, the $\hat{z}_i$'s the transformed sequence and the (x_i, y_i)'s the non-uniform sampling points of the output sequence. It can be seen that, by exploiting Eq. (10), Eq. (21) can be rewritten as

$$\hat{z}_i = \underbrace{\sum_{m=\mu_i-k}^{\mu_i+k} \underbrace{\sum_{n=\nu_i-l}^{\nu_i+l} \hat{\phi}_{i,m-\mu_i} \hat{\psi}_{i,n-\nu_i} \underbrace{\sum_{k=-N_1/2}^{N_1/2} \sum_{l=-N_2/2}^{N_2/2} \underbrace{\frac{(2\pi)}{\phi_k^1 \phi_l^2} z_{kl}}_{\text{Scaling and zero padding}}}_{\text{Standard FFT on } cN_1 \times cN_2 \text{ points}} e^{-j2\pi \frac{mk}{cN_1}} e^{-j2\pi \frac{nl}{cN_2}}}_{\text{Interpolation}}} \tag{22}$$

which highlights the three different steps necessary to the implementation of the 2D NER-NUFFT.

Using analytical, instead of numerical (finite difference), derivatives is convenient not only from the computational point of view, but also in terms of synthesis performance, as it will be shown in Section 6.

The parallel implementation can be performed in a very similar way to [50] and the details are omitted here.

5.3. Performance of the 2D NER-NUFFT

In this Subsection, we assess the computational performance and the accuracy of the implemented 2D NER-NUFFT.

As for the NED case, a GPU implementation of the 2D NER-NUDFT has been worked out based on cuBLAS routines for reference.

In Fig. 7, the speedup of the GPU implementation as compared to the CPU one for the 2D-NER NUFFT is reported. As it can be seen from Fig. 7, for values of M corresponding to a 100×100 reflectarray, the GPU vs CPU speedup is about 80. The less speedup for the NED implementation (Fig. 6) as compared to the NER one is due to the use of atomic operations needed by the former algorithm, which, on the contrary, are not required for the latter one. Once again, for a value of $M = 64$, the speedup of the CUDA implementation against the cuBLAS based 2D-NUDFT has been around 100. Similarly to the NED case, using the cuBLAS based 2D-NUDFTs has not been possible beyond the mentioned value of M due to the unmanageable memory requirements.

Concerning the accuracy, Fig. 8 shows the percentage Root Mean Square (RMS) error achieved by the implemented 2D NER-NUFFT against the exact evaluation of the corresponding 2D NUDFT. Again, a very high accuracy, close to that dictated by IEEE double precision arithmetics, is reached.

6. SYNTHESIS RESULTS

In this Section, we report an abstract of the results of an extensive numerical analysis aimed at assessing the performance achievable by the developed technique. Some of the features of the technique are first illustrated with reference to periodic case. Subsequently, we point out the performance achievable with an aperiodic reflectarray and compare it with that of the periodic counterpart.

6.1. Periodic Reflectarray

Before discussing the aperiodic case, we first consider the synthesis of a 71×71 shaped beam, periodic reflectarray radiating according to the typical European coverage. The periodic reflectarray has been synthesized according to a procedure similar to that indicated above for the synthesis of the aperiodic case, while only avoiding the optimization of the element positions. The coverage masks have been generated by assuming a geostationary satellite reflectarray with position 5E0N and the main antenna

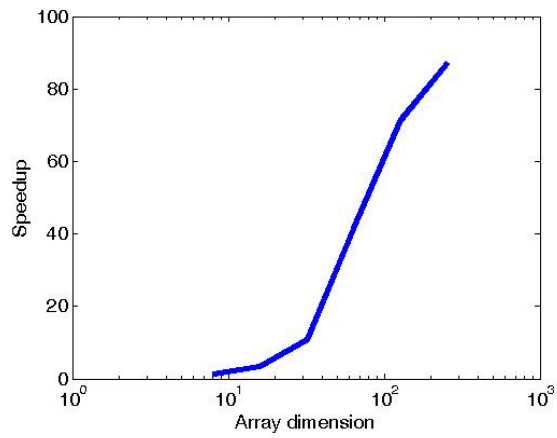

Figure 7. GPU vs CPU speedup for the 2D NER-NUFFT.

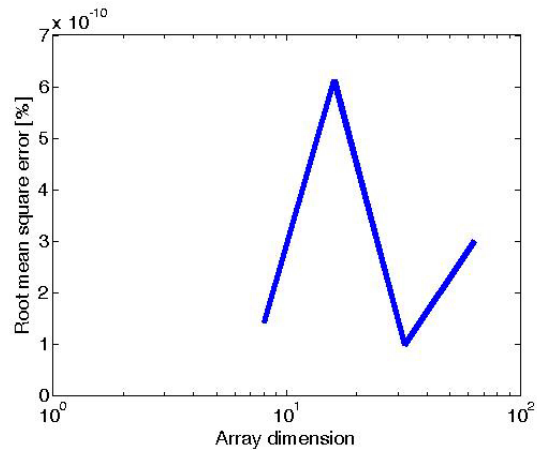

Figure 8. Accuracy of the 2D NER-NUFFT: percentage RMS error against an exact calculation of the corresponding 2D NUDFT.

parameters for this test-case are summarized in Tab. 1. The choice of the periodic reflectarray enables highlighting some useful features of the synthesis process.

For this test-case, we begin by showing in Fig. 9 the result achieved by the application of MLSL optimization algorithm to the first synthesis stage based on the use of Zernike polynomials. On the other side, Fig. 10 illustrates the final result following the application of the last synthesis stage using impulsive basis functions for phase representation and using the result illustrated in Fig. 9 as a starting point. The mean (D_{mean}) and minimum (D_{min}) directivities over the whole coverage are reported in Tab. 2. The very good shaping of the beam synthesized by the considered approach can be appreciated.

We remark that the result presented in Fig. 9 has been obtained by using the analytical expression of the gradient reported in Eq. (19), computed, as mentioned, by aid of the 2D NER-NUFFT algorithm. With this expedient, the computation time has been around 2 hours on the machine whose features have been detailed in Section 4. We have also run the synthesis algorithm with a numerical approximation of the gradient provided by Matlab's finite differences. In this case, the computation time has been significantly longer, about 4 days. Furthermore, the overall performance has been slightly worse than that achieved with the analytical gradient, as detailed in Tab. 2.

The performance of the developed algorithm has been compared with that achievable by the well-known, so-called *iterated projections* technique proposed in [32] for which the result is shown in Fig. 11 and summarized in Tab. 2 and which leads to an overall simpler optimization algorithm. Such an approach, however, suffers from a significant local minima problem, so that the solution in Fig. 11 has been obtained in a very large number of subsequent optimization steps in which the number of reflectarray elements has been gradually increased and θ_f has been gradually changed from 0° (centered reflectarray) to 26.9° [51]. This comparison shows that, already for the simpler case of a periodic

Table 1. Main parameters for the European coverage test-case.

Frequency	14.25 GHz
Number of elements	71×71
y_{off}	0.423 m
z_0	0.833 m
$\theta_f = a\tan(y_{\text{off}}/z_0)$	26.9°
Interelement spacing	0.5λ
Feed illumination factor m_f	12

reflectarray for which the cost functional is less affected from local minima, the developed synthesis strategy outperforms the approach in [32]. Obviously, when moving from a periodic reflectarray to an aperiodic one and so when increasing the number of unknowns and changing the functional topology, the local minima trapping issue is expected to become more relevant, as typically stressed in the literature concerning optimization. This suggests trusting in the generalized projections method for the optimization of the cost functional relevant to this paper.

6.2. Aperiodic Reflectarray

Let us consider now the synthesis of an aperiodic, 44×44 elements reflectarray, radiating the "Galaxy 17" coverage made up of Continental USA (ConUS) and Canada [35, 36], whose details are reported in Tab. 3.

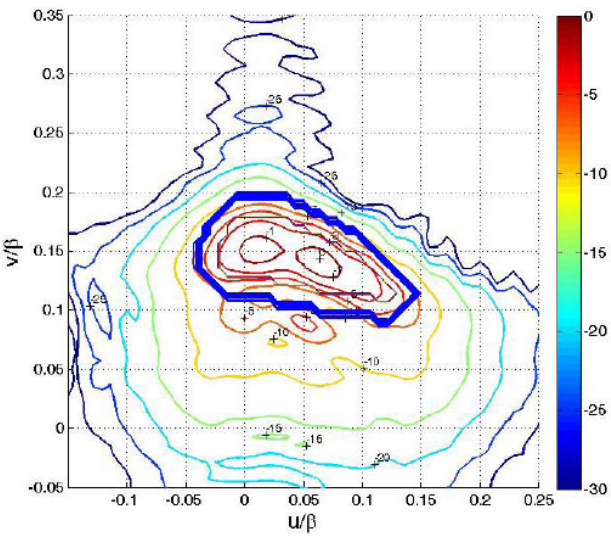

Figure 9. 71×71 sized periodic reflectarray: result of the first stage of the PO synthesis. The thick blue line indicates the coverage region.

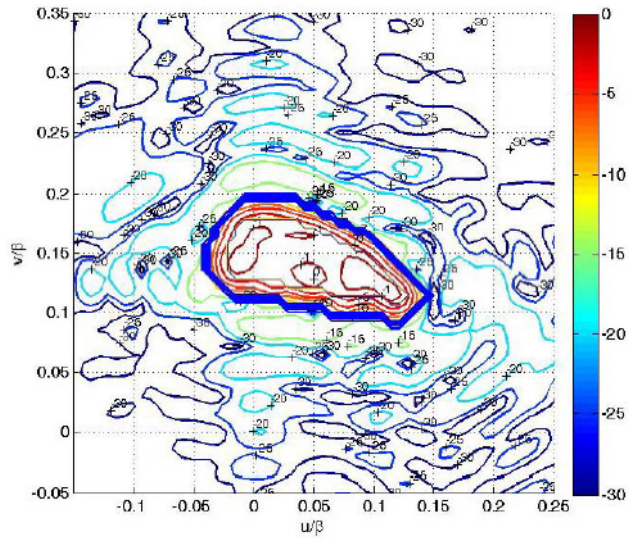

Figure 10. 71×71 sized periodic reflectarray: result of the final stage of the PO synthesis. The thick blue line indicates the coverage region.

Table 2. Performance results for the 71×71 sized periodic reflectarray.

	D_{mean}	D_{min}
Developed approach with analytical gradient	30.86	29.88
Developed approach with numerical gradient	30.5	29.5
Iterated projections	25.38	20.72

Table 3. Main parameters for the ConUS coverage test-case.

Frequency	14.25 GHz
Number of elements	44×44
y_{off}	0.276 m
z_0	1 m
$\theta_f = a\tan(y_{\text{off}}/z_0)$	14.9°
Min/max interelement spacing	$0.5\lambda/0.7\lambda$
Feed illumination factor m_f	12

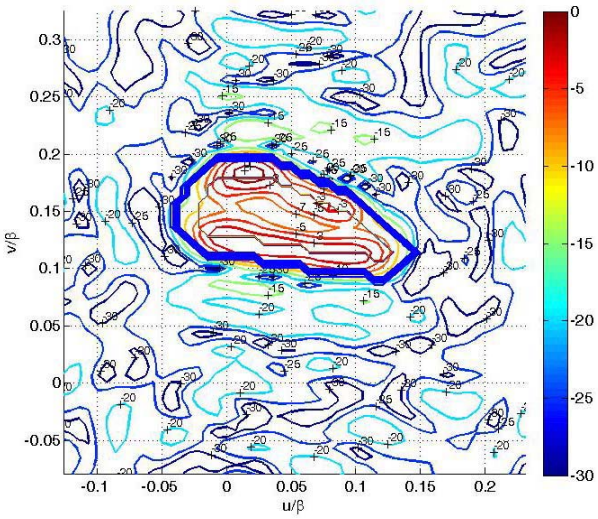

Figure 11. 71×71 sized periodic reflectarray: result of the *iterated projections* approach. The thick blue line indicates the coverage region.

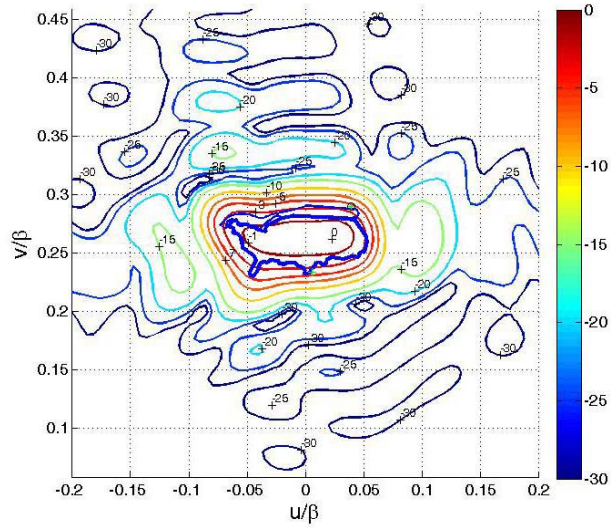

Figure 12. 44×44 sized aperiodic reflectarray: result of the PO synthesis. The thick blue line indicates the coverage region.

Table 4. Mean and minimum directivity (dB) over the coverage for the synthesized case in Fig. 12.

	D_{mean}	D_{min}
Whole Coverage	31.06	27.98
ConUs	31.32	28.34
Canada	29.92	27.98

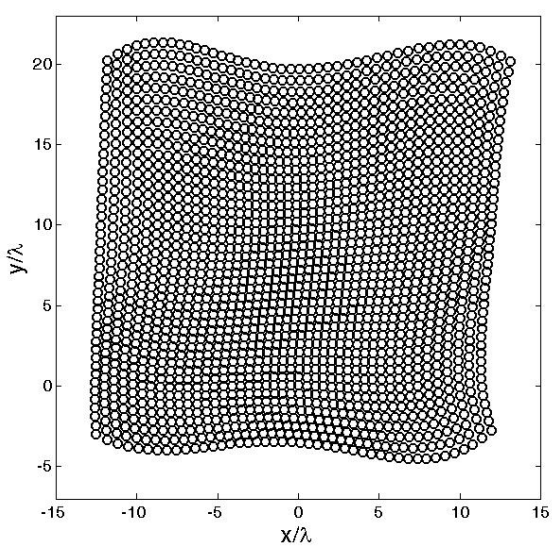

Figure 13. 44×44 sized aperiodic reflectarray: element positioning.

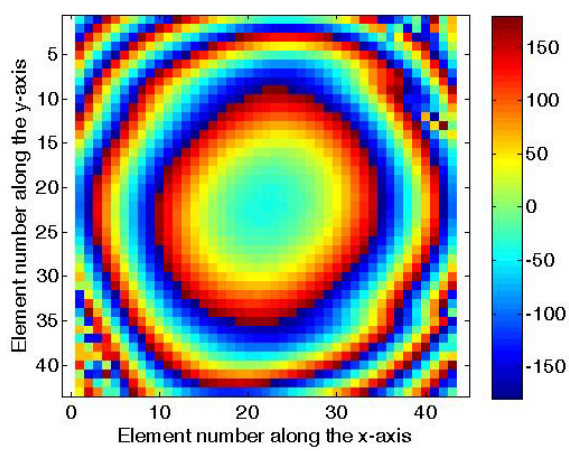

Figure 14. 44×44 sized aperiodic reflectarray: command phases in degrees.

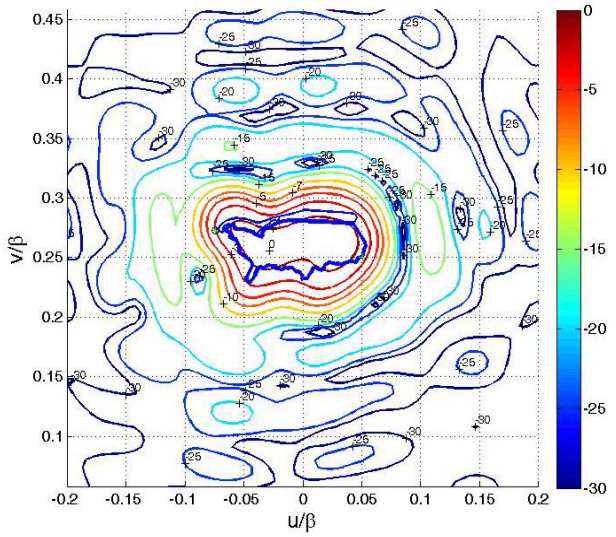

Figure 15. 44 × 44 sized periodic reflectarray: result of the PO synthesis. The thick blue line indicates the coverage region.

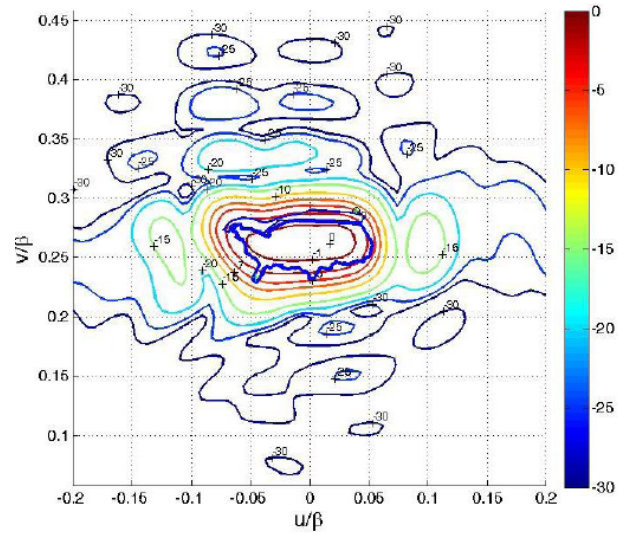

Figure 16. 42 × 42 sized aperiodic reflectarray: result of the PO synthesis. The thick blue line indicates the coverage region.

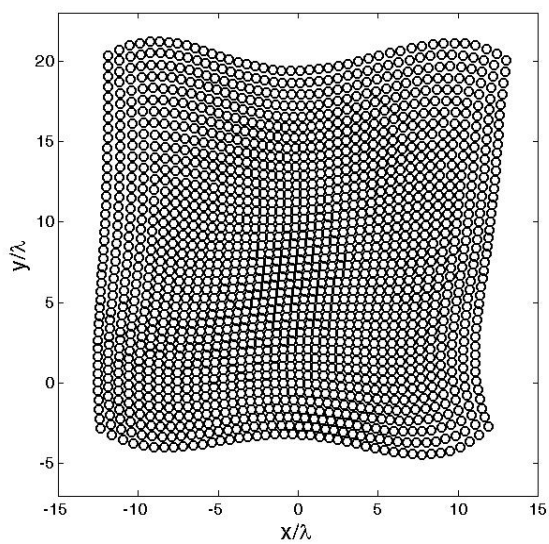

Figure 17. 42 × 42 sized aperiodic reflectarray: element positioning.

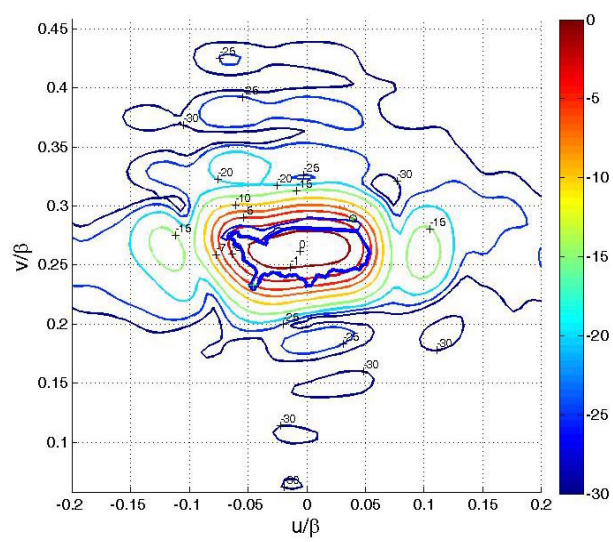

Figure 18. 40 × 40 sized aperiodic reflectarray: result of the PO synthesis. The thick blue line indicates the coverage region.

The directivity pattern of the synthesized aperiodic reflectarray is reported in Fig. 12, the achieved element positioning is displayed in Fig. 13, while the command phases are displayed in Fig. 14. The mean (D_{mean}) and minimum (D_{min}) directivities over the whole coverage as well as over the ConUS and Canada partial coverages are reported in Tab. 4. The whole synthesis has required approximately 3 hours to be accomplished on an a Genesis Tesla I-7950 workstation, with a 8-core Intel CPU i7-950, working at 3.06 GHz and with 6 GB of RAM, and equipped with an NVIDIA Tesla C2050 card.

For the sake of comparison, the directivity pattern of a periodic reflectarray, with elements arranged on a grid with $\lambda/2$ spacing, and synthesized according to a PO model, is reported in Fig. 15, while Tab. 5 summarizes the mean and minimum directivities over the coverages. As it can be seen, an aperiodic arrangement of the reflectarray elements allows better performance.

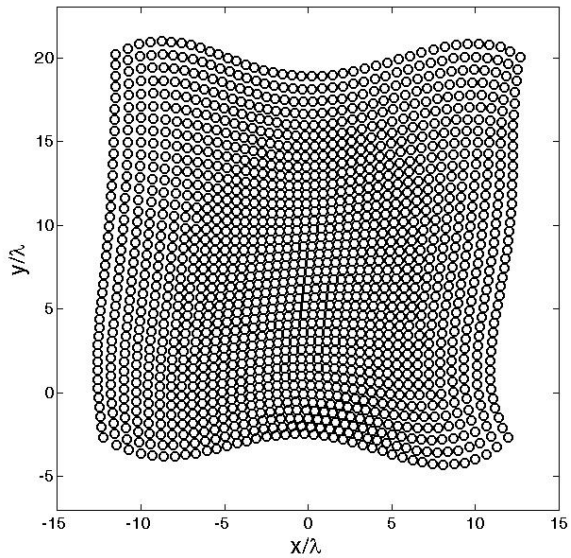

Figure 19. 40×40 sized aperiodic reflectarray: element positioning.

Table 5. Mean and minimum directivity (dB) over the coverage for the synthesized case in Fig. 15.

	D_{mean}	D_{min}
Whole Coverage	29.71	27.66
ConUs	29.91	28.31
Canada	29.05	27.66

Table 6. Mean and minimum directivity (dB) over the coverage for the synthesized cases in Figs. 14, 19 and 21.

Whole Coverage	D_{mean}	D_{min}
44×44	31.06	27.98
42×42	31.21	27.60
40×40	31.31	27.80

Test cases involving a reduced number of elements have been also considered. The results reported in Figs. 16–19 and Tabs. 4 and 6 allow evaluating the effect of the number of elements reduction when passing from a 44×44 aperiodic reflectarray (Figs. 12 and 13), to a 42×42 (Figs. 16 and 17) or even to a 40×40 aperiodic reflectarray (Figs. 18 and 19). As it can be seen, by a 40×40 aperiodic reflectarray, results comparable to that of the 44×44 case are obtained, with about 17% less elements.

7. CONCLUSIONS AND FUTURE DEVELOPMENTS

We have dealt with one of the computationally most critical steps of the PO synthesis of aperiodic reflectarrays, namely the fast evaluation of the radiation operator.

We have presented an approach exploiting the joint use of a fast numerical algorithm based on 2D NED- and NER-NUFFTs and on parallel processing on Graphics Processing Units (GPUs) in Compute Unified Device Architecture (CUDA) language. We have pointed out the programming strategies used to accelerate the implementation of the approach and assessed its computational performance. In particular, we have shown that:

1. when dealing with aperiodic reflectarrays, a 2D NED-NUFFT can be used for the direct radiation with accuracy comparable to that of a standard DFT;

2. the implementation and acceleration of 2D NED-NUFFTs on GPUs enables speedups of about 8/10 for reflectarrays of medium/large size;

3. the generalized projections technique outperforms the iterated projections technique;

4. the implementation of the proposed synthesis technique with analytical gradient calculation outperforms that with numerical gradient calculation;

5. the calculation of the analytical gradient can benefit of the use of a 2D NER-NUFFT which preserves the accuracy and guarantees speedups of about 80 for medium/large sized reflectarrays;

6. the introduction of further degrees of freedom (positions in aperiodic reflectarrays) allows improving the performance with respect to periodic reflectarrays;

7. it is possible to adopt aperiodic reflectarrays of reduced number of elements for fixed performance which is a very appealing feature to reduce the number of unknowns of interest, especially in the case of electronically reconfigurable reflectarray antennas;

8. the two levels of parallelism intrinsic in the interpolation step of the 2D NED-NUFFT can be fruitfully exploited by the use of dynamic parallelism made available in one of the latest architecture of CUDA cards.

Concerning the difficulty and the cost of constructing reflectarray antennas, it should be mentioned that these antennas are typically manufactured with printed technology which is cheap and simplifies their fabrication, especially regarding mispositioning errors. In other words, printing aperiodic reflectarrays does not appear more critical than printing periodic reflectarrays. Likewise, the feed alignment issue does not appear more critical in aperiodic reflectarrays than in periodic ones.

The use of an accurate radiative model [35, 36], the control of the shape of the reflecting surface by the use of conformal surfaces [35, 36, 52], for which fast analysis tools have been already developed [12], the use of the orientation of the reflecting elements [36], and the extension to multi-frequency design specifications [36] are continuations of the presented activity that we have already performed. A further future computational development will consist in the possibility of enforcing the design specifications on an arbitrary (non-Cartesian) spectral grid. This will require using Type-3 NUFFT algorithms [53] even for flat reflectarrays.

ACKNOWLEDGMENT

The Authors wish to thank the European Space Agency (ESA) and Space Engineering S.p.A. for the support provided during the LET-SME 2009 ESA Activity.

APPENDIX A.

First of all, let us remark that the discrete version of the operator Φ in Eq. (17) which is optimized by the algorithm is

$$\Phi = \sum_{h,k} \left| |E_{co_{hk}}|^2 - \mathcal{P}_{U_{co}} \left(|E_{co_{hk}}|^2 \right) \right|^2, \tag{A1}$$

where $E_{co_{hk}}$ is the co-polar component of the field radiated at the regular (u_h, v_k) (Cartesian) spectral grid location expressed as

$$E_{co_{h,k}} = E_{co}(u_h, v_k) = g_{hk} \sum_{n=1}^{N} w_n^{m_f} \frac{e^{-j\beta r_n}}{r_n} e^{j\psi_n} e^{j\beta(u_h x_n + v_k y_n)}, \tag{A2}$$

$g_{hk} = [\underline{\underline{Q}}(u_h, v_k)\underline{\underline{S}}_0(u_h, v_k)\underline{\tilde{E}}_f] \cdot \hat{e}_{co}$ and $\hat{e}_{co}$ is the unit vector corresponding to the co-polar component of the field. The functional Φ can be also rewritten as

$$\Phi = \sum_{h,k} \left(|E_{co_{hk}}|^2 - \mathcal{P}_{U_{co}} \left(|E_{co_{hk}}|^2 \right) \right) \left(|E_{co_{hk}}|^2 - \mathcal{P}_{U_{co}} \left(|E_{co_{hk}}|^2 \right) \right)^*. \tag{A3}$$

Accordingly,

$$\frac{\partial \Phi}{\partial \psi_n} = 2\sum_{h,k} \frac{\partial \left|E_{co_{hk}}\right|^2}{\partial \psi_n} \left(\left|E_{co_{hk}}\right|^2 - \mathcal{P}_{U_{co}}\left(\left|E_{co_{hk}}\right|^2\right)\right)^* \tag{A4}$$

Now,

$$\frac{\partial \left|E_{co_{hk}}\right|^2}{\partial \psi_n} = 2\mathrm{Re}\left\{\frac{\partial E_{co_{hk}}}{\partial \psi_n} E^*_{co_{hk}}\right\} = -2\mathrm{Im}\left\{g_{hk} w_n^{m_f} \frac{e^{-j\beta r_n}}{r_n} e^{j\psi_n} e^{j\beta(u_h x_n + v_k y_n)} E^*_{co_{hk}}\right\}. \tag{A5}$$

Finally,

$$\frac{\partial \Phi}{\partial \psi_n} = -4\mathrm{Im}\left\{w_n^{m_f} \frac{e^{-j\beta r_n}}{r_n} e^{j\psi_n} \sum_{h,k} E^*_{co_{hk}} \left(\left|E_{co_{hk}}\right|^2 - \mathcal{P}_{U_{co}}\left(\left|E_{co_{hk}}\right|^2\right)\right) g_{hk} e^{j[u_h x_n + v_k x_n]}\right\}. \tag{A6}$$

REFERENCES

1. Itoh, T., *Numerical Techniques for Microwave and Millimeter-wave Passive Structures*, J. Wiley & Sons, New York, 1989.
2. Tao, Y., H. Lin, and H. Bao, "GPU-based shooting and bouncing ray method for fast RCS prediction," *IEEE Trans. Antennas Prop.*, Vol. 58, No. 2, 494–502, Feb. 2010.
3. Guan, J., S. Yan, and J.-M. Jin, "An OpenMP-CUDA implementation of multilevel fast multipole algorithm for electromagnetic simulation on multi-GPU computing systems," *IEEE Trans. Antennas Prop.*, Vol. 61, No. 7, 3607–3616, Jul. 2013.
4. Peng, S. and C.-F. Wang, "Precorrected-FFT method on Graphics Processing Units," *IEEE Trans. Antennas Prop.*, Vol. 61, No. 4, 2099–2107, Apr. 2013.
5. Ricciardi, G. F., J. R. Connelly, H. A. Krichene, and M. T. Ho, "A fast-performing error simulation of wideband radiation patterns for large planar phased arrays with overlapped subarray architecture," *IEEE Trans. Antennas Prop.*, Vol. 62, No. 4, 1779–1788, Apr. 2014.
6. Meng, H.-T. and J.-M. Jin, "Acceleration of the dual-field domain decomposition algorithm using MPI-CUDA on large-scale computing systems," *IEEE Trans. Antennas Prop.*, Vol. 62, No. 9, 4706–4715, Sep. 2014.
7. Mu, X., H.-X. Zhou, K. Chen, and W. Hong, "Higher order method of moments with a parallel out-of-core LU solver on GPU/CPU platform," *IEEE Trans. Antennas Prop.*, Vol. 62, No. 11, 5634–5646, Nov. 2014.
8. Kirk, D. B. and W.-M. W. Hwu, *Programming Massively Parallel Processors*, 2nd Edition, Moargan Kaufmann, Waltham, Morgan Kaufmann, 2013.
9. Owens, J. D., M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. of the IEEE*, Vol. 96, No. 5, 879–899, May 2008.
10. Rossi, F. V., P. P. M. So, N. Fichtner, and P. Russer, "Massively parallel two-dimensional TLM algorithm on Graphics Processing Units," *Proc. of the IEEE Microw. Theory Tech. Symp.*, 153–156, Atlanta, GA, Jun. 15–20, 2008.
11. De Donno, D., A. Esposito, G. Monti, and L. Tarricone, "MPIE/MOM with a general purpose Graphics Processing Unit," *IEEE Trans. Microw. Theory Tech.*, Vol. 60, No. 9, 2693–2701, Sep. 2012.
12. Capozzoli, A., C. Curcio, G. D'Elia, A. Liseno, and P. Vinetti, "Fast CPU/GPU pattern evaluation of irregular arrays," *Applied Comput. Electromagn. Soc. J.*, Vol. 25, No. 4, 355–372, Apr. 2010.
13. Capozzoli, A., C. Curcio, and A. Liseno, "GPU-based ω-k tomographic processing by 1D non-uniform FFTs," *Progress In Electromagnetics Research M*, Vol. 23, 279–298, 2012.
14. Breglia, A., A. Capozzoli, C. Curcio, and A. Liseno, "CUDA expression templates for electromagnetic applications on GPUs," *IEEE Antennas Prop. Mag.*, Vol. 55, No. 5, Oct. 2013.
15. Lezar, E. and D. B. Davidson, "GPU-accelerated method of moments by example: Monostatic scattering," *IEEE Antennas Prop. Mag.*, Vol. 52, No. 6, 120–135, Dec. 2010.

16. Topa, T., A. Karwowski, and A. Noga, "Using GPU with CUDA to accelerate MoM-based electromagnetic simulation of wire-grid models," *IEEE Antennas Wireless Prop. Lett.*, Vol. 10, 342–345, 2011.
17. Topa, T., A. Noga, and A. Karwowski, "Adapting MoM with RWG basis functions to GPU technology using CUDA," *IEEE Antennas Wireless Prop. Lett.*, Vol. 10, 480–483, 2011.
18. Huang, J. and J. A. Encinar, *Reflectarray Antennas*, J. Wiley & Sons, Hoboken, NJ, 2008.
19. Bialkowski, M. E. and J. A. Encinar, "Reflectarrays: Potentials and challenges," *Proc. of the Int. Conf. on Electromagn. in Adv. Appl.*, 1050–1053, Turin, Italy, Sep. 17–21, 2007.
20. Encinar, J. A., M. Arrebola, and G. Toso, "A parabolic reflectarray for a bandwidth improved contoured beam coverage," *Proc. of the Europ. Conf. on Antennas Prop.*, 1–5, Edinburgh, UK, Nov. 11–16, 2007.
21. Capozzoli, A., C. Curcio, G. D'Elia, A. Liseno, G. Toso, and P. Vinetti, "Aperiodic and non-planar array of electromagnetic scatterers, and reflectarray antenna comprising the same," World Patent Nr. WO/2011/033388.
22. Willey, R., "Space tapering of linear and planar arrays," *IEEE Trans. on Antennas Prop.*, Vol. 10, No. 4, 369–377, Jul. 1962.
23. Skolnik, M. I., "Nonuniform arrays," *Antenna Theory*, R. E. Colin and F. J. Zucker, Capter 6, McGraw-Hill, New York, 1969.
24. Morabito, A. F., T. Isernia, and L. Di Donato, "Optimal synthesis of Phase-Only reconfigurable linear sparse arrays having uniform-amplitude excitations," *Progress In Electromagnetics Research*, Vol. 124, 405–423, 2012.
25. Capozzoli, A., C. Curcio, G. D'Elia, A. Liseno, and P. Vinetti, "FFT & aperiodic arrays with phase-only control and constraints due to super-directivity, mutual coupling and overall size," *Proc. of the 30th ESA Antenna Workshop on Antennas for Earth Observ., Science, Telecomm. and Navig. Space Missions*, Noordwijk, The Netherlands, CD ROM, May 27–30, 2008.
26. Capozzoli, A., C. Curcio, G. D'Elia, A. Liseno, and P. Vinetti, "FFT & equivalently tapered aperiodic arrays," *Proc. of the XXIX General Assembly of the Int. Union of Radio Sci.*, Chicago, IL, CD ROM, Aug. 7–16, 2008.
27. Capozzoli, A., C. Curcio, A. Liseno, and G. Toso, "Phase-Only synthesis of flat aperiodic reflectarrays," *Progress In Electromagnetics Research*, Vol. 133, 53–89, 2013.
28. Fourmont, K., "Non-equispaced fast Fourier transforms with applications to tomography," *J. Fourier Anal. Appl.*, Vol. 9, No. 5, 431–450, 2003.
29. Capozzoli, A. and A. Liseno, "Fast reflectarray antenna analysis and synthesis on GPUs," *The GPU Technology Conference*, San Jose, California, Mar. 18–21, 2013.
30. Capozzoli, A., C. Curcio, A. Liseno, and G. Toso, "Dynamic parallelism in reflectarray antenna analysis and synthesis on GPU," *PIERS Abstracts*, 1003, Stockholm, Sweden, Aug. 12–15, 2013.
31. Dynamic Parallelism in CUDA, *NVIDIA White Pape*, 2012, http://developer.download.nvidia.com /assets/cuda/docs/TechBrief_Dynamic_Parallelism_in_CUDA_v2.pdf.
32. Bucci, O. M., G. Franceschetti, G. Mazzarella, and G. Panariello, "Intersection approach to array pattern synthesis," *IEE Proc. Pt. H*, Vol. 137, No. 6, 349–357, Dec. 1990.
33. Fatica, M. and W.-K. Jeong, "Accelerating Matlab with CUDA," *Proc. of the High Performance Embedded Comput. Workshop*, Lexington, MA, Sep. 18–20, 2007.
34. Suh, J. W. and Y. Kim, *Accelerating Matlab with GPU Computing: A Primer with Examples*, Waltham, MA, Morgan Kaufmann, 2014.
35. "Advanced reflectarray antennas," *ESA LET-SME 2009 Project*, Technical Report Nr. 1, Jul. 2010 (Available on Request to the European Space Agency).
36. "Advanced reflectarray antennas," *ESA LET-SME 2009 Project*, Technical Report Nr. 2, Mar. 2011 (Available on Request to the European Space Agency).
37. Capozzoli, A., C. Curcio, A. Liseno, M. Migliorelli, and G. Toso, "Power pattern synthesis of advanced flat aperiodic reflectarrays," *Proc. of the 33rd ESA Workshop on Challenges for Space Antenna Systems*, Noordwijk, The Netherlands, Oct. 18–21, 2011.

38. Bläser, M., "Lower bounds for the multiplicative complexity of matrix multiplication," *Comput. Complex.*, Vol. 8, No. 3, 203–226, Dec. 1999.
39. Atkinson, K. and D. D. K. Chien, "A fast matrix-vector multiplication method for solving the radiosity equation," *Adv. in Comput. Math.*, Vol. 12, No. 2–3, 151–174, Feb. 2000.
40. Blair, J. M., "Rational Chebyshev approximations for the modified Bessel functions I_0 and I_1," *Math. Comp.*, Vol. 28, No. 126, 581–583, Apr. 1974.
41. Frigo, M. and S. G. Johson, "The design and implementation of FFTW3," *Proc. of the IEEE*, Vol. 93, No. 2, 216–231, Feb. 2005.
42. Bulatsyk, O. O., B. Z. Katsenelenbaum, Y. P. Topolyuk, and N. N. Voitovich, *Phase Optimization Problems*, Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, 2010.
43. Capozzoli, A., C. Curcio, E. Iavazzo, A. Liseno, M. Migliorelli, and G. Toso, "Phase-only synthesis of a-periodic reflectarrays," *Proc. of the Europ. Conf. on Antennas Prop.*, 1031–1035, Rome, Italy, Apr. 11–15, 2011.
44. Bucci, O. M., G. D'Elia, G. Mazzarella, and G. Panariello, "Antenna pattern synthesis: A new general approach," *Proc. of the IEEE*, Vol. 82, No. 3, 358–371, Mar. 1994.
45. Capozzoli, A. and G. D'Elia, "Global optimization and antennas synthesis and diagnosis, Part I: Concepts, tools, strategies and performances," *Progress In Electromagnetics Research*, Vol. 56, 195–232, 2006.
46. Capozzoli, A. and G. D'Elia, "Global optimization and antennas synthesis and diagnosis, Part II: Applications to advanced reflector antenna synthesis and diagnosis techniques," *Progress In Electromagnetics Research*, Vol. 56, 233–261, 2006.
47. Wolpert, D. H. and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evolutionary Comput.*, Vol. 1, No. 1, 67–82, Apr. 1997.
48. Rinnooy Kan, A. G. H. and G. T. Timmer, "Stochastic global optimization methods, Part I: Clustering methods," *Math. Progr.*, Vol. 39, No. 1, 27–56, Sep. 1987.
49. Rinnooy Kan, A. G. H. and G. T. Timmer, "Stochastic global optimization methods, Part II: Multi level methods," *Math. Progr.*, Vol. 39, No. 1, 57–78, Sep. 1987.
50. Capozzoli, A., C. Curcio, and A. Liseno, "Fast GPU-based interpolation for SAR backprojection," *Progress In Electromagnetics Research*, Vol. 133, 259–283, 2013.
51. Zornoza, J. A. and J. A. Encinar, "Efficient phase-only synthesis of contoured-beam patterns for very large reflectarrays," *Int. J. of RF Microw. Comp.-Aided Design*, Vol. 14, No. 5, 415–423, Sep. 2004.
52. Capozzoli, A., C. Curcio, A. Liseno, M. Migliorelli, and G. Toso, "Aperiodic conformal reflectarrays," *Proc. of the Antennas Prop. Soc. Int. Symp.*, 361–364, Spokane, WA, Jul. 3–8, 2011.
53. Capozzoli, A., 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*, Vol. 142, 743–770, 2013.