SIAM J. SCI. COMPUT.
Vol. 44, No. 2, pp. A964–A992

TWO-DIMENSIONAL FOURIER CONTINUATION AND APPLICATIONS*

OSCAR P. BRUNO† AND JAGABANDHU PAUL†

Abstract. This paper presents a fast "two-dimensional Fourier continuation" (2D-FC) method for construction of biperiodic extensions of smooth nonperiodic functions defined over general two-dimensional smooth domains. The approach, which runs at a cost of $\mathcal{O}(N \log N)$ operations for an N-point discretization grid, can be directly generalized to domains of any given dimensionality, but such generalizations are not considered in this contribution. The 2D-FC extensions are produced in a two-step procedure. In the first step the *one-dimensional* Fourier continuation method is applied along a discrete set of outward boundary-normal directions to produce, along such directions, continuations that vanish outside a narrow interval beyond the boundary. Thus, the first step of the algorithm produces "blending-to-zero along normals" for the given function values. In the second step, the extended function values are evaluated on an underlying Cartesian grid by means of an efficient, high-order boundary-normal interpolation scheme. A Fourier continuation expansion of the given function can then be obtained by a direct application of the two-dimensional fast Fourier transform (FFT). Algorithms of arbitrarily high order of accuracy can be obtained by this method. The usefulness and performance of the proposed 2D-FC method are illustrated with applications to the Poisson equation and the time-domain wave equation within a bounded domain. As part of these examples the novel "Fourier forwarding" solver is introduced which, *propagating plane waves as they would in free space* and relying on certain boundary corrections, can solve the time-domain wave equation and other hyperbolic partial differential equations *within general domains* at computing costs that grow *sublinearly* with the size of the spatial discretization.

Key words. two-dimensional Fourier continuation, Poisson equation, wave equation, FC solver, Fourier forwarding, FFT

AMS subject classifications. 35J05, 78A45, 35L05, 42A10, 42A15

DOI. 10.1137/20M1373189

1. Introduction.

This paper presents a fast "two-dimensional Fourier Continuation" (2D-FC) method for construction of biperiodic extensions of smooth nonperiodic functions defined over general two-dimensional (2D) smooth domains. The algorithm, which runs at a cost of $\mathcal{O}(N \log N)$ operations for an N-point discretization grid, converges with a user-prescribed dth order of convergence, and can be directly generalized to domains of any given dimensionality, but such generalizations are not considered here. The usefulness and performance of the proposed 2D-FC method are illustrated with applications to the Poisson equation and the time-domain wave equation within a bounded domain. As part of these examples the novel "Fourier forwarding" solver is introduced which, *propagating plane waves as they would in free space* and relying on certain boundary corrections, can solve the time-domain wave equation and other constant-coefficient hyperbolic partial differential equations *within general domains* at computing costs that grow *sublinearly* with the size of the spatial discretization.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section October 13, 2020; accepted for publication (in revised form) November 18, 2021; published electronically April 25, 2022.

https://doi.org/10.1137/20M1373189

Funding: This work was supported by the NSF through grants DMS-1714169, DMS-2109831, by DARPA through grant HR00111720035, by AFOSR through grant FA9550-21-1-0373, and by the Vannewar Bush Foundation through grant N00014-16-1-2808.

†Computing and Mathematical Sciences, Caltech, Pasadena, CA 91125 USA (obruno@caltech.edu, jpaul@caltech.edu).

The periodic-extension problem has actively been considered in the recent literature, in view, in particular, of its applicability to the solution of various types of partial differential equations (PDEs) [1, 3, 4, 6, 7, 8, 10, 11, 14, 17, 20, 21, 24]. The contributions [3, 4, 11, 20], in particular, utilize the Fourier continuation (FC) method in one dimension in conjunction with dimensional splitting for the treatment of multidimensional PDE problems. The dimensional splitting is also used in [12] to produce Fourier extensions to rectangular domains in two dimensions, where the FC is effected by separately applying the one-dimensional FC-Gram method [3, 4, 11] first to the columns and then to the rows of a given data matrix of function values. The method does assume that the given smooth function is known on a rectangular region containing the domain for which the continuation is sought. The 2D-FC algorithm proposed in this paper does not rely on such a stringent requirement.

The approach to periodic function extension presented in [6, 24] is based on the solution of a high-order PDE, where the extension shares the values and normal derivatives along the domain boundary. [14], in turn, presents a function-extension method based on the use of radial basis functions (RBFs). In that approach, overlapping circular partitions, or patches, are placed along the physical boundary of the domain, and a local extension is defined on each patch by means of RBFs. A second layer of patches is placed outside the first, on which the local values are set to vanish. The zero patches are used in conjunction with a partition of unity function to blend the local extensions into a global counterpart. The choice of functions used to build the partition of unity determines the regularity of the extended function. [21] proposes an accurate and superalgebraically convergent function approximation method using Fourier extension frames in general 2D domains at a cost of $\mathcal{O}(N^2 \log^2 N)$ operations: a significantly higher cost than the $\mathcal{O}(N \log N)$ cost required by the 2D-FC method introduced in the present contribution. In fact, as indicated in that reference, the $\mathcal{O}(N^2 \log^2 N)$ cost reported only improves upon the $\mathcal{O}(N^3)$ cost required by a corresponding full SVD-based algorithm for discretizations containing more than 8,100 points. As an example, cost and accuracy comparisons presented in Example 3.3 below, which concerns approximation of a function exhibiting 36 oscillations on a circular domain, and for both high ($7 \cdot 10^{-8}$) and low ($1.7 \cdot 10^{-3}$) accuracy, the 2D-FC method is approximately 10,000 times faster and requires over 20 times less memory than the approach in [21], with significant increases in improvement factors expected as discretizations are enlarged further.

The 2D-FC extensions proposed in this paper are produced in a two-step procedure. In the first step the *one-dimensional* FC (1D-FC) method [4] is applied along a discrete set of outward boundary-normal directions to produce, along such directions, continuations that vanish outside a narrow interval beyond the boundary. Thus, the first step of the algorithm produces "blending-to-zero along normals" for the given function values. In the second step, the extended function values are evaluated on an underlying Cartesian grid by means of an efficient, high-order boundary-normal interpolation scheme. Since the continuation-along-normals procedure is a fixed cost operation for each normal line, the cost of this procedure grows only linearly with the size of the boundary discretization. An FC expansion of the given function can then be obtained by a direct application of the 2D FFT algorithm. Algorithms of arbitrarily high order of accuracy can be obtained by this method. In view of its construction on the basis of combined 1D-FC approximation and regular polynomial interpolation, the convergence theory for the 2D-FC method follows directly from the theoretical results presented in [20].

As mentioned above, this paper demonstrates the usefulness of the proposed general-domain 2D-FC technique via applications to both the Poisson problem for the Laplace equation and the time-domain wave equation. In the Poisson case the 2D-FC method is utilized to obtain a *particular solution* for a given right-hand side; the boundary conditions are then made to match the prescribed boundary data by adding a solution of the Laplace equation which is produced by means of boundary-integral methods. The Fourier forwarding approach, in turn, uses the 2D-FC method to solve the spatio-temporal PDE in the interior of the domain and it then corrects the solution values near the boundary by means of a classical time-stepping solver. The overall procedure, which utilizes large time steps for the interior solver and small CFL-constrained time steps for the near-boundary solver, runs in computing times per small time step that grow sublinearly with the size of the spatial discretization mesh.

It is interesting to note that the primary continuation device in the 2D-FC method, namely, continuation along normals to the domain boundary, is a *one-dimensional procedure*. This one-dimensional continuation procedure can be utilized in a generalization of the method to n-dimensional domains with $n > 2$. This is in contrast to other extension methods mentioned above. For example, the RBF-based extension method [14] requires solution of boundary problems of increasing dimensionality as the spatial dimension grows, which, given the method's reliance on dense-matrix linear algebra for the local-extension process, could have a significant impact on computing costs. Similar comments apply to PDE-based extension methods such as [24].

The proposed 2D-FC algorithm performs favorably in the context of existing related approaches. Specific comparisons with results presented in [14] are provided in subsection 4.1.1 for a Poisson problem considered in that reference. The recent contribution [13], in turn, presents an FFT-based high-order solver for the Poisson problem for *rectangular* domains, namely, Cartesian products of one-dimensional (1D) intervals in either 2D or three-dimensional space. The present 2D-FC based Poisson solver achieves, *for general domains*, a similar performance (similar accuracy and computing time) to that demonstrated in [13, Tables 3 and 4] under the Cartesian-domain assumption.

As discussed in [3, 4], and particularly e.g. in [4, Figure 12], in view of the spectral character of the 1D-FC approach, which incurs errors in finite-degree polynomial approximation near boundaries, but which propagates the polynomial boundary accuracy to the domain interior via Fourier series, enjoys excellent dispersion characteristics as well: using FC derivatives as part of a PDE solver results in errors that, asymptotically, do not grow as the number of oscillations in the PDE domain and the discretization size are simultaneously and proportionally increased. This dispersion characteristic is also enjoyed by the 2D-FC algorithm, as illustrated in the top graph of Figure 11.

This paper is organized as follows. After a brief review of the 1D-FC method presented in section 2, the proposed 2D-FC method is introduced in section 3. The two main applications considered, namely, solution of the Poisson and Fourier-forwarding for the wave equation, are presented in subsections 4.1 and 4.2. Finally our conclusions are presented in section 5.

2. Background: 1D "blending-to-zero" FC algorithm. A brief review of the 1D-FC method [3, 4] is presented in this section, with an emphasis on one of its key components, the *blending-to-zero* procedure—which is employed in the normal direction continuation portion of the proposed 2D-FC approach presented in section 3.

2.1. 1D-FC algorithm: Outline. Given the vector $\boldsymbol{\phi}_D = (\phi_0, \ldots, \phi_{N-1})^t$ of values of a smooth function $\phi : [0,1] \to \mathbb{C}$ on the equispaced grid

$$D = \{x_j = jk : 0 \le j \le N-1\}$$

of step size $k = 1/(N-1)$, the 1D-FC method [3, 4] of order d (with, e.g., $4 \le d \le 12$) produces, at first, an $(N+C)$-dimensional vector $\boldsymbol{\phi}^c$ of discrete continuation function values (including the N given function values) over an extended interval $[0,b], b > 1$. To do this, the algorithm utilizes the d-dimensional vectors $\boldsymbol{\phi}_\ell = (\phi_0, \ldots, \phi_{d-1})^t$ and $\boldsymbol{\phi}_r = (\phi_{N-d}, \ldots, \phi_{N-1})^t$ of values of the function ϕ on the left and right "matching-point" sets $D_\ell = \{x_0, \ldots, x_{d-1}\}$ and $D_r = \{x_{N-d}, \ldots, x_{N-1}\}$, respectively, each one of which is contained in a small subinterval of length $(d-1)k$ near the corresponding endpoint of the containing interval $[0,1]$. In order to obtain the C necessary continuation values, the 1D-FC method blends $\boldsymbol{\phi}_\ell$ and $\boldsymbol{\phi}_r$ to zero (see subsection 2.2), towards the left and right, respectively, resulting in two zero-blending vectors of length C. The sum of these two vectors is then utilized as a rightward discrete continuation to the set $D^c = \{x_j = 1 + jk : 1 \le j \le C\}$ of points in the interval $(1,b]$—as described in subsection 2.3. As indicated in that section, the overall 1D-FC procedure is then completed via an application of the FFT algorithm to the $(N+C)$-dimensional vector $\boldsymbol{\phi}^c$ (cf. (2.7) below) of "smoothly periodic" discrete continued function values. The following two subsections describe the blending-to-zero and 1D-FC approaches, respectively.

2.2. Blending-to-zero algorithm. In our description of the order-d blending-to-zero algorithm [4] we only present details for the *rightward* blending-to-zero technique, since the leftward blending-to-zero problem can easily be reduced to the rightward problem. Thus, given the column vector $\boldsymbol{F}_{\mathcal{D}} = (F_0, \ldots, F_{d-1})^t$ of values of a complex-valued function F on the set $\mathcal{D} = \{x_0, x_1, \ldots, x_{d-1}\}$, the rightward blending-to-zero approach starts by producing a polynomial interpolant for F over the interval $[x_0, x_{d-1}]$ relying on the Gram polynomial basis

$$G_d = \{g_0(x), g_1(x), \ldots, g_{d-1}(x)\} \tag{2.1}$$

for this interval. The functions $g_j(x)$ $(j = 0, \ldots, d-1)$ are the polynomials with real coefficients that are obtained as the Gram–Schmidt orthogonalization procedure is applied, in order of increasing degree, to the polynomials in the set $\{1, x, x^2, \ldots, x^{d-1}\}$, with respect to the discrete scalar product

$$(g,h) = \sum_{j=0}^{d-1} g(x_j)h(x_j).$$

Discrete values of the Gram polynomials on the set $\mathcal{D}$ can be computed on the basis of the QR factorization [15]

$$\boldsymbol{P} = \boldsymbol{QR} \quad \text{of the matrix} \quad \boldsymbol{P} = (x_i{}^{j-1})_{0 \le i,j \le d-1}. \tag{2.2}$$

(Note that the jth column of $\boldsymbol{Q}$ contains the values of the jth Gram polynomial on the set $\mathcal{D}$.) Following [4] we obtain the necessary QR factorization by applying the stabilized Gram–Schmidt orthogonalization method to the matrix $\boldsymbol{P}$.

In order to closely approximate each one of the Gram polynomials in G_d, throughout the continuous interval $[0, (d-1)k]$ containing $\mathcal{D}$, by corresponding trigonometric

polynomials, as described below, we use a certain "oversampled matching" method. According to this method the polynomials in G_d are oversampled to an equispaced set of discretization points with step size k/n_{os} (containing $n_{\text{os}}(d-1)+1$ points), where n_{os} denotes the oversampling factor, and where the oversampled values are used as part of a certain low-dimensional SVD matching procedure described in what follows. Note that the aforementioned oversampled values on the refined grid $\mathcal{D}_{\text{os}} \coloneqq \{\tilde{x}_j = jk/n_{os} : 0 \le j \le n_{\text{os}}(d-1)\}$ coincide with the columns of the matrix

$$\boldsymbol{Q}_{\text{os}} = \boldsymbol{P}_{\text{os}}\boldsymbol{R}^{-1}, \tag{2.3}$$

where $\boldsymbol{P}_{\text{os}}$ is the Vandermonde matrix of size $(n_{\text{os}}(d-1)+1)\times d$ corresponding to the oversampled discretization $\mathcal{D}_{\text{os}}$, and where $\boldsymbol{R}$ is the upper triangular matrix in (2.2).

The aforementioned SVD matching procedure, which is one of the crucial steps in the FC approach [4], produces a band-limited Fourier series of the form

$$g_j^c(x) = \sum_{m=-J}^{J} a_m^j e^{\frac{2\pi i m x}{(d+2C+Z-1)k}} \tag{2.4}$$

for each polynomial $g_j \in G_d$ $(0 \le j \le d-1)$, where C is the number of blending-to-zero values to be produced, and Z being the number of "zero-matching" points. The Fourier coefficients are selected so as to match, in the least-squares sense, both the oversampled polynomial values over the interval $[0,(d-1)k]$, and identically zero values on an equally fine discretization of the "zero matching" interval

$$[(d+C)k,(d+C+Z-1)k]$$

of length $(Z-1)k$. The coefficients in (2.4) are taken to equal the solution $\boldsymbol{a}$ of the minimization problem

$$\min_{\boldsymbol{a}=(a_{-J},\ldots,a_J)^T} \left\| \boldsymbol{B}_{\text{os}}\boldsymbol{a} - \begin{pmatrix} \boldsymbol{q}_{\text{os}}^j \\ \boldsymbol{0} \end{pmatrix} \right\|_2, \tag{2.5}$$

where $\boldsymbol{B}_{\text{os}}$ is a matrix whose entries are values of (2.4) at all points in the set $\mathcal{D}_{\text{os}}$ as well as all the set of k/n_{os}-spaced points in the zero matching interval mentioned above (which, in particular, contains the endpoints $(d+C)k$ and $(d+C+Z-1)k$). The minimizing Fourier coefficients $\boldsymbol{a}$ are then found via an SVD-based [15] least-squares approach. Once the coefficients $\boldsymbol{a}$ have been obtained, the resulting Fourier expansions (2.4) are used to produce a certain "continuation matrix" $\boldsymbol{A} \in \mathbb{C}^{C\times d}$, whose columns equal the values of the expression (2.4) at the C (unrefined) k-discretization points in the interval $[dk,(d+C-1)k]$ (cf. Remark 2.2 below). The desired vector $\boldsymbol{F}^r$ of rightward blending-to-zero function values at the C continuation points in the interval $[dk,(d+C-1)k]$ is then given by the expression

$$\boldsymbol{F}^r = \boldsymbol{A}\boldsymbol{Q}^T\boldsymbol{F}. \tag{2.6}$$

2.3. 1D-FC algorithm. As outlined in subsection 2.1, the 1D-FC algorithm requires use of a certain rightward (resp., leftward) blending-to-zero vector $\boldsymbol{\phi}_r^r$ (resp., $\boldsymbol{\phi}_\ell^\ell$) for a given matching-point vector $\boldsymbol{\phi}_r$ (resp., $\boldsymbol{\phi}_\ell$). In view of (2.6) we define $\boldsymbol{\phi}_r^r = \boldsymbol{A}\boldsymbol{Q}^T\boldsymbol{\phi}_r$. To obtain the leftward extension $\boldsymbol{\phi}_\ell^\ell$, in turn, we first introduce the "order reversion" matrix $R^e \in \mathbb{C}^{e\times e}$ $(e \in \mathbb{N})$ by

$$R^e(g_0,g_1\ldots,g_{e-2},g_{e-1})^t = (g_{e-1},g_{e-2},\ldots,g_1,g_0)^t,$$

and we then define $\boldsymbol{\phi}_\ell^\ell = R^C \boldsymbol{A}\boldsymbol{Q}^T R^d \boldsymbol{\phi}_\ell$. A vector $\boldsymbol{\phi}^c$ containing both the N given values in the vector $\boldsymbol{\phi} = (\phi_0, \phi_1, \ldots, \phi_{N-1})^t$ as well as the C "continuation" function values is constructed by appending the sum $\boldsymbol{\phi}_\ell^\ell + \boldsymbol{\phi}_r^r$ at the end of the vector $\boldsymbol{\phi}$, so that we obtain

$$\boldsymbol{\phi}_j^c = \begin{cases} \boldsymbol{\phi}_j & \text{for } 0 \le j \le N-1, \\ \left(\boldsymbol{\phi}_\ell^\ell + \boldsymbol{\phi}_r^r\right)_{(j-N)} & \text{for } N \le j \le N+C-1. \end{cases} \tag{2.7}$$

Following the various stages of the construction of the vector $\boldsymbol{\phi}^c$ it is easy to check that, up to numerical error, this vector contains point values of a smoothly periodic function defined over the interval $[0, b]$. An application of the FFT algorithm to this vector therefore provides the desired continuation function in the form of a trigonometric polynomial,

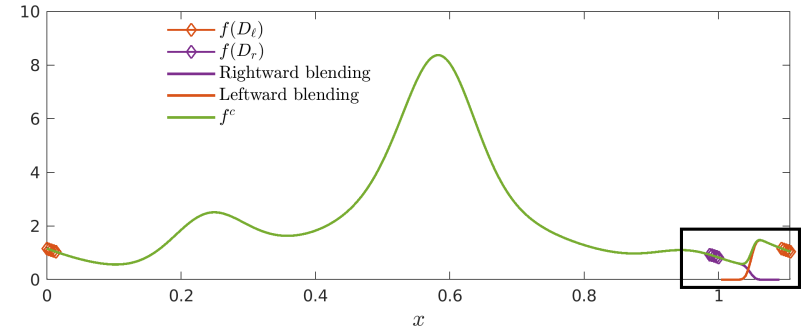
$$\phi^c(x) = \sum_{\ell=-(N+C)/2}^{(N+C)/2} \hat{\phi}_\ell^c e^{\frac{2\pi i \ell x}{b}}, \tag{2.8}$$

which closely approximates ϕ in the interval $[0, 1]$. In fact, as demonstrated in previous publications (including [3, 4]), for sufficiently smooth functions ϕ, the corresponding 1D-FC approximants converge to ϕ with order $\mathcal{O}(k^d)$—so that, as expected, the number d of points used in the blending-to-zero procedures determines the convergence rate of the algorithm, while, as mentioned in section 1, giving rise to essentially dispersionless numerics in the context of the PDE solution. The 2D-FC algorithm introduced in the following section also relies on the 1D blending-to-zero procedure described in subsection 2.2, and its convergence in that case is once again of the order $\mathcal{O}(k^d)$, and, when applied to solution of PDE problems, essentially dispersionless.

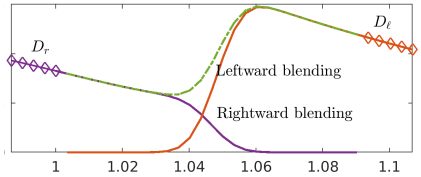
It is important to note that, for a given order d, the matrices $\boldsymbol{A}$ and $\boldsymbol{Q}$ can be computed once and permanently stored on disc for use whenever an application of the blending-to-zero algorithm is required—as these matrices do not depend on the point spacing k. A graphical demonstration of various elements of the 1D-FC procedure is presented in Figure 1.

Remark 2.1 (extra vanishing values). The 1D-FC implementations [3, 4] allow for an additional number $E \ge 0$ of identically zero "extra" function values to be added on an (unrefined) k-discretization of the interval $[(d+C)k, (d+C+E-1)k]$, as illustrated in Figure 1(c), to obtain a desired overall number of discrete function values (including the given function values and the continuation values produced) such as, e.g., a power of two or a product of powers of small prime numbers, for which the subsequent application of the FFT is particularly well suited. The corresponding use of extra vanishing values for the 2D continuation problem is mentioned in Remark 3.2.

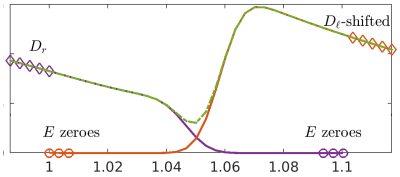
Remark 2.2 (blending-to-zero on a refined grid). As indicated above in the present section, the 2D-FC procedure introduced in section 3 utilizes the 1D blending-to-zero strategy described above in this section to extend a function given on a 2D domain Ω along the normal direction to Γ; the continuation values obtained at all normals are then utilized to obtain the continuation function on the Cartesian grid by interpolation. As detailed in subsection 3.4, in order to prevent accuracy loss in the 2D interpolation step we have found it necessary to use 1D normal-direction grids finer than the grids inherent in the blending-to-zero process itself. To easily provide the necessary fine-grid values, a modified fine-grid continuation matrix $\boldsymbol{A}_r \in \mathbb{C}^{C_r \times d}$ is constructed, where $C_r > C$ denotes the number of fine-grid points utilized.



(a) Demonstration of the 1D-FC method. The continuation values are computed as the sum of the blended-to-zero rightward and leftward extensions.



(b) Inset of Figure 1a.



(c) Numbers E of leftward and rightward extra zeroes.

FIG. 1. *Illustration of the 1D-FC procedure. Figure 1(a) depicts the FC of the nonperiodic function* $\phi : [0,1] \to \mathbb{R}$ *given by* $\phi(x) = \exp(\sin(5.4\pi x - 2.7\pi) - \cos(2\pi x)) - \sin(2.5\pi x) + 1$. *Figure 1(b) presents a close up of the right continuation region* $[1-(d-1)k, b]$. *Subsequently Figure 1(c) illustrates the use of a number* E *of extra zeroes in the blending-to-zero process, to yield a continuation mesh containing FFT-friendly numbers (products of powers of small prime numbers) of point values.*

The modified continuation matrix $\boldsymbol{A}_r$ can be built on the basis of the minimizing coefficients $\boldsymbol{a}$ in (2.5): the corresponding columns of the fine-grid continuation matrix $\boldsymbol{A}_r$ are obtained by evaluating (2.4) on the given fine-grid points in the interval $((d-1)k, (d+C-1)k]$. The necessary blending-to-zero function values at the C_r fine-grid points are given by $\boldsymbol{A}_r \boldsymbol{Q}^T \boldsymbol{\phi}_D$.

3. 2D-FC method. This section presents the proposed volumetric FC method on 2D domains $\Omega \subset \mathbb{R}^2$ with a smooth boundary $\Gamma = \overline{\Omega} \setminus \Omega$, some elements of which are illustrated in Figure 2. Let a smooth function $f : \overline{\Omega} \to \mathbb{C}$ be given; we assume that values of f are known on a certain uniform Cartesian grid within $\overline{\Omega}$ as well as a grid of points on the boundary Γ. The 2D-FC algorithm first produces 1D blending-to-zero values for the function f *along directions normal to the boundary* Γ, yielding continuation values on a certain 2D tangential-normal curvilinear grid around Γ, as detailed in sections 3.1 and 3.2 and illustrated in Figure 2. These continuation values, which are produced on the basis of the corresponding blending-to-zero procedure presented in subsection 2.2 in the context of the 1D-FC method, are then interpolated onto a Cartesian grid around the domain boundary, to produce a 2D blending-to-zero continuation of the function f. The necessary interpolation from the curvilinear grid to the Cartesian grid is accomplished by first efficiently obtaining the foot of the normal that passes through a given Cartesian grid point $\boldsymbol{r} = (x, y)$ exterior to Ω and near the boundary Γ (section 3.3), and then using a local 2D interpolation procedure

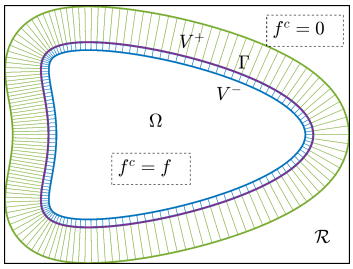


FIG. 2. *Geometrical constructions underlying the* 2*D-FC procedure, with reference to the various regions defined in subsection* 3.1.

to produce the corresponding continuation value at the point $\boldsymbol{r}$ (section 3.4). Once the interpolated values have been obtained throughout the Cartesian mesh around Γ, the desired 2D-FC function

$$f^c(x,y) = \sum_{\ell=-N_x/2+1}^{N_x/2} \sum_{m=-N_y/2+1}^{N_y/2} \hat{f}^c_{\ell,m} e^{2\pi i\left(\frac{\ell x}{L_x}+\frac{my}{L_y}\right)} \tag{3.1}$$

(where L_x and L_y denote the period in the x and y directions, respectively) is obtained by means of a 2D FFT. Following the algorithmic prescriptions presented in sections 3.1 through 3.4, a summary of the overall 2D-FC approach is presented in section 3.5.

3.1. 2D tangential-normal curvilinear grid. The necessary curvilinear grids around Γ can be produced on the basis of a (smooth) parametrization

$$\boldsymbol{r} = \boldsymbol{q}(\theta) = (x(\theta), y(\theta)), \quad 0 \le \theta \le 2\pi, \tag{3.2}$$

of the boundary Γ. We assume the boundary to be of class C^{d+1} to ensure the interpolation processes described in subsection 3.4 enjoy the necessary $\mathcal{O}(h^d)$ convergence order. However, per Example 3.2, use of interpolation of order $M = d + 2$ in that context yields somewhat improved accuracy. In order to achieve such improvements it is necessary to assume the boundary Γ is a C^{d+3} curve.

In view of their intended application (blending-to-zero along the normal direction in accordance with subsection 2.2), the curvilinear grids are introduced within interior and exterior strips V^- and V^+ (illustrated in Figure 2) given by

$$\begin{cases} V^- = \{\boldsymbol{q}(\theta) - \boldsymbol{n}(\theta)\gamma : 0 \le \theta \le 2\pi \text{ and } 0 \le \gamma \le (d-1)k_1\}, \\ V^+ = \{\boldsymbol{q}(\theta) + \boldsymbol{n}(\theta)\gamma : 0 \le \theta \le 2\pi \text{ and } 0 \le \gamma \le Ck_1\}, \end{cases} \tag{3.3}$$

where $\boldsymbol{n}(\theta) = (n_x(\theta), n_y(\theta))$ denotes the unit normal to the boundary Γ at $\boldsymbol{q}(\theta)$, and where d, C, and k_1 denote, respectively, the number of matching points, the number of blending-to-zero points, and the step size used, in the present application of the 1D blending-to-zero procedure described in subsection 2.2. Using, in addition, the uniform discretization

$$I_B = \{\theta_p = pk_2 : 0 \le p < B\}, \quad k_2 = \frac{2\pi}{B}, \tag{3.4}$$

of the interval $[0, 2\pi]$, we then construct a curvilinear 2D discretization

(3.5)
$$V^-_{B,d} = \{\boldsymbol{r}_{p,q} :\ \boldsymbol{r}_{p,q} = \boldsymbol{q}(\theta_p) + \boldsymbol{n}(\theta_p)(q - d + 1)k_1;\ 0 \le p < B \text{ and } 0 \le q \le d - 1\},$$

(3.6)
$$V^+_{B,C_r} = \{\boldsymbol{s}_{p,q} :\ \boldsymbol{s}_{p,q} = \boldsymbol{q}(\theta_p) + \boldsymbol{n}(\theta_p)qk_1/n_r;\ 0 \le p < B \text{ and } 0 \le q \le C_r\},$$

within V^- and V^+ respectively, for the given step size k_1, where $C_r = Cn_r$ for certain integer (refinement factor) n_r; note that the points in $V^-_{B,d}$ for $q = d-1$ and the points in V^+_{B,C_r} for $q = 0$ coincide and that they lie on Γ. Here the constants d, C, and n_r are independent of B. The continuation function is constructed so as to vanish at all points $\boldsymbol{s}_{p,q} \in V^+_{B,C_r}$ with $q = C_r$. Let now $\mathcal{R} = [a_0, a_1] \times [b_0, b_1]$ denote the smallest closed rectangle containing $\Omega \cup V^+$, and consider the equispaced Cartesian grid of step size h,

$$H = \{\boldsymbol{z}_{i,j} = (x_i, y_j) : x_i = a_0 + ih;\ y_j = b_0 + jh :\ 0 \le i < N_x, 0 \le j < N_y\} \tag{3.7}$$

on $\mathcal{R}$, where the 2D continuation function values are to be computed. We note that the size of the rectangle $\mathcal{R}$ along with the strips V^- and V^+ decrease as the step size k_2 is decreased.

3.2. Computation of FC values on V^+_{B,C_r}. A continuation of the function f to the exterior of Ω is obtained via application of the blending-to-zero procedure presented in subsection 2.2 (cf. Remark 2.2) along each one of the normal directions inherent in the definition of the set V^+_{B,C_r}. For given p, the d equidistant points $\boldsymbol{s}_{p,q} \in V^-_{B,d}$ $(0 \le q \le d - 1)$, which are indicated by the solid circles in Figure 3, constitute a set $\mathcal{D}_p$ of matching points that are used to effect the blending-to-zero procedure per the prescriptions presented in subsection 2.2. To obtain the desired continuation function values it is necessary to first obtain the vector $\boldsymbol{f}_{\mathcal{D}_p}$ of the values of the function f (or suitable approximations thereof) on the set $\mathcal{D}_p$. In the proposed method, the needed function values $\boldsymbol{f}_{\mathcal{D}_p}$ are computed on the basis of a two-step polynomial interpolation scheme, using polynomials of a certain degree $(M - 1)$, as briefly described in what follows.

With reference to the right image of Figure 3, and considering first the case $|n_x(\theta_p)| \ge |n_y(\theta_p)|$, the algorithm initially interpolates vertically the function values at M open-circle Cartesian points selected as indicated in the figure, onto the points of intersection, shown as red stars, of the normal line and the vertical Cartesian grid lines. For intersection (red-star) points close enough to the boundary, boundary function values at boundary points shown as squares in the figure, are utilized in the 1D interpolation process as well. Once the red-star function values are acquired, the function value at the matching solid-black point is effected by interpolation from the M red-star point values previously obtained, on the basis of a polynomial of degree $(M - 1)$.

The case $|n_x(\theta_p)| < |n_y(\theta_p)|$ is treated similarly, substituting the initial interpolation along vertical Cartesian lines, by interpolation along horizontal Cartesian lines; the algorithm then proceeds in an entirely analogous fashion.

3.3. Proximity map. As described below in section 3.4, the 2D-FC algorithm interpolates the FC values on V^+_{B,C_r} onto the Cartesian mesh H. The interpolation algorithm used in that section relies on a certain "proximity map" $\mathcal{P} : H \cap V^+ \to V^+_{B,C_r}$ which associates a curvilinear grid point $\boldsymbol{s}_{p,q} = \mathcal{P}(\boldsymbol{z}_{i,j}) \in V^+_{B,C_r}$ in the "proximity"

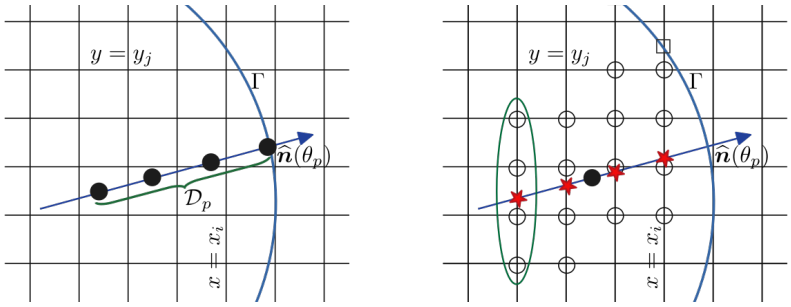


FIG. 3. *Interpolation scheme for evaluation of* $\boldsymbol{f}_{\mathcal{D}_p}$ *in the case* $|n_x(\theta_p)| \geq |n_y(\theta_p)|$. *Left: Black solid circles indicate the matching points that define the set* $\mathcal{D}_p$. *Right: Known function values at the Cartesian points (denoted by the empty circles) and, in some cases, at the point of intersection (represented by an empty square) of the vertical grid lines with* Γ, *are used to interpolate the matching function values at the red-star intersection points of the normal with the vertical grid lines. The function values at the red points are then used to obtain, by interpolation, the function values at the matching points.*

of each given Cartesian grid point $\boldsymbol{z}_{i,j}$. The algorithm described in the next section primarily uses the associated point $\boldsymbol{s}_{p,q}$ along with the corresponding discretized boundary parameter value θ_p to interpolate the foot of the normal for the Cartesian point $\boldsymbol{z}_{i,j}$. The proximity function we use is obtained by first associating with each curvilinear discretization point $\boldsymbol{s}_{p,q}$ the nearest Cartesian point, a procedure that results in a set $\mathcal{P}_0 \subseteq (H \cap V^+) \times V^+_{B,C_r}$ of pairs of points, one in the Cartesian grid and the other in the curvilinear grid. (The initial set $\mathcal{P}_0$ can easily be obtained by using the "integer part" *floor* ($\lfloor . \rfloor$) and the *ceil* ($\lceil . \rceil$) operators.) The set $\mathcal{P}_0$ is then modified by removing multiple associations for a given Cartesian point, and, if necessary, by adding a "next-nearest" curvilinear neighbor to Cartesian points that previously remained unassociated. The next-nearest curvilinear neighbor can be obtained simply by subtracting one from the floor value and adding one to the ceil value and, taking all the combinations of the results for each point in V^+_{B,C_r}. The resulting set $\mathcal{P}$ defines the desired function.

3.4. FC values on the Cartesian grid. Once FC values on V^+_{B,C_r} have been obtained, per the procedure presented in subsection 3.2, the 2D-FC scheme can be completed by (a) interpolation onto the set $H \cap V^+$ of outer Cartesian grid points; and (b) subsequent evaluation of the corresponding FCs in (3.1) by means of an FFT. (Note that since the continuation function f^c is a smooth function which vanishes outside $\overline{\Omega} \cup V^+$, this function can be viewed as the restriction of a smooth and biperiodic function with periodicity rectangle $\mathcal{R}$—whose Fourier series approximates f^c and, therefore f, accurately.) The efficiency of the interpolation scheme is of the utmost importance in this context—since interpolation to a relatively large set $H \cap V^+$ of Cartesian points is necessary. An accurate and efficient interpolation strategy is obtained by combining two 1D (local) interpolation procedures based on nearby normal directions. The first interpolation procedure produces the parameter value θ of the foot of the normal to Γ passing through a given Cartesian point; the second procedure then approximates the continuation function value utilizing the parameter value θ just mentioned and the continuation function values at the points in V^+_{B,C_r} around the given Cartesian point. A detailed description of the combined interpolation methodology is presented in what follows. Specifically, we describe the strategy

we use to interpolate the continuation function onto each point $Q = \boldsymbol{z}_{i,j} \in H \cap V^+$ and, to do this, we first obtain the foot $\mathcal{F}(Q)$ of this point. Using the proximity map $\mathcal{P}$ described in subsection 3.3, the algorithm utilizes the curvilinear discretization point $\boldsymbol{s}_{p,q} = \mathcal{P}(\boldsymbol{z}_{i,j}) \in V^+_{B,C_r}$ as well as the corresponding boundary discretization parameter value θ_p; according to (3.6), the point $\boldsymbol{q}(\theta_p)$ equals the foot of the normal passing through $\boldsymbol{s}_{p,q}$: $\boldsymbol{q}(\theta_p) = \mathcal{F}(\boldsymbol{s}_{p,q})$. The algorithm then seeks approximation of the foot $\mathcal{F}(Q)$ and the corresponding parameter value $\theta = \theta^Q$ via a preliminary interpolation step, as indicated in what follows. The foot $\mathcal{F}(Q)$ and the corresponding parameter value $\theta = \theta^Q$ are then used by the algorithm to produce the desired interpolated continuation value at Q.

In order to obtain $\mathcal{F}(Q)$ the algorithm uses the M boundary parameter values in the set

$$S_{\theta_p} = \{\theta_{p-K_\ell}, \theta_{p-K_\ell+1}, \ldots, \theta_p, \theta_{p+1}, \ldots, \theta_{p+K_r}\} \subset I_B \tag{3.8}$$

around θ_p (where $K_r + K_\ell + 1 = M$, and where $K_r = K_\ell$ if M is odd and $K_r = K_\ell + 1$ if M is even. Parameter values θ_k with negative values of k, which may arise in (3.8), are interpreted by periodicity: $\theta_k = \theta_{B+k}$).

The algorithm then utilizes the line $L^\perp_Q$ passing through Q that is orthogonal to the normal vector $\boldsymbol{n}(\theta_p)$ (see left image in Figure 4), together with the parametrization $\boldsymbol{\ell}^{\mathrm{tr}}_Q(\tau)$ of $L^\perp_Q$, where the parameter τ represents the signed distance of the points on $L^\perp_Q$ from the point Q. Clearly, then, $\boldsymbol{\ell}^{\mathrm{tr}}_Q(0) = Q$. Each point of intersection of $L^\perp_Q$ with the normals $\boldsymbol{n}(\theta_j)$ ($\theta_j \in S_{\theta_p}$), on the other hand, equals $\boldsymbol{\ell}^{\mathrm{tr}}_Q(\tau_j)$, where τ_j denotes the signed distance between Q and the corresponding intersection point. Thus, defining the function $\theta = \mathcal{T}(\tau)$, where $\mathcal{T}(\tau)$ gives the parameter value of the foot of the normal through the point $\boldsymbol{\ell}^{\mathrm{tr}}_Q(\tau)$, we clearly have

$$\theta_j = \mathcal{T}(\tau_j), \quad p - K_\ell \le j \le p + K_r. \tag{3.9}$$

It follows that a 1D interpolation procedure on the function $\mathcal{T}(\tau)$ can be used to obtain the desired approximation of the value $\theta^Q = \mathcal{T}(0)$ of the parameter corresponding to the foot of the point $Q = \boldsymbol{z}_{i,j}$: $\mathcal{F}(Q) = \boldsymbol{q}(\theta^Q) = \boldsymbol{q}(\mathcal{T}(0))$.

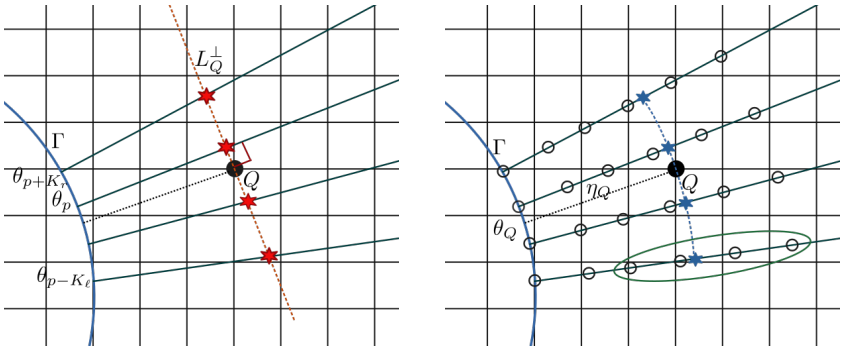


FIG. 4. *Interpolation schemes utilized to obtain the continuation function values on $H \cap V^+$ on the basis of the continuation values on V^+_{B,C_r}. Left: Evaluation of the boundary parameter value for the foot of the normal line passing through Q, depicted as a finely dotted line passing through that point. The left image also displays the set of red-star interpolation points along the dashed-orange line $L^\perp_Q$. Right: Interpolation of continuation values from the curvilinear mesh to a point Q on the Cartesian mesh.*

Once we have the corresponding foot parameter value θ^Q for the given Cartesian point Q, the distance η_Q of the point Q to the boundary Γ is easily computed. Let $S_{\theta^Q} \subset I_B$ be a set of M boundary parameter values, similar to the set S_{θ_p} defined in (3.8), but with S_{θ^Q} "recentered" around θ^Q. In order to obtain the continuation function value at the point Q using the continuation function values on V^+_{B,C_r}, we employ a local 2D polynomial interpolation scheme based on the set S_{θ^Q} and the distance η_Q, as indicated in what follows and illustrated in the right image of Figure 4. First, the continuation values are obtained at each point (marked by blue asterisks in the figure) at a distance η_Q from the boundary Γ along the normal grid lines in V^+_{B,C_r} that correspond to boundary parameter values in the set S_{θ^Q}. Each one of these values is obtained via 1D interpolation of the continuation function values on V^+_{B,C_r} along the corresponding normal grid line in V^+_{B,C_r}. The desired continuation value at the point Q, then, is obtained via a final 1D degree-$(M-1)$ interpolation step based on the parameter set S_{θ^Q} and values at the "blue-asterisk" points just obtained. Finally, by applying the 2D FFT to the continuation function values computed above we then obtain the desired Fourier series expression in (3.1) for the continuation function.

Remark 3.1 (function values on Γ). I. For definiteness, in this paper we have assumed that the boundary data are provided in the form of values of the given function—which correspond to the the Dirichlet boundary data in the PDE context. But the approach is also applicable in cases for which the boundary data are given as the normal derivative of the function, (Neumann boundary data), or even a combination of function and normal derivative values (Robin data) by relying on a slightly modified blending-to-zero procedure of the type presented in [4]. II. If no boundary data are available, the 2D-FC method can still be utilized on the basis of interior data only, albeit with a certain reduction in accuracy near the boundary.

Remark 3.2 (extra vanishing values in 2 dimensions). As in the 1D case, prior to the FFT procedure the grid H can be enlarged, with vanishing function values assigned to the added discretization points to obtain a discretization containing a number of discretization points equal to a power of two (or a product of powers of small prime numbers) along each Cartesian direction, which leads to especially fast evaluations by means of the FFT.

3.5. Summary of the 2D-FC procedure. This section presents a summary of the 2D-FC procedure described in sections 3.1 through 3.4 for a function f given on a uniform Cartesian grid $H \cap \Omega$ within the domain of definition Ω, where H is a Cartesian mesh over the rectangle $\mathcal{R}$ containing both Ω and the near-boundary outer region V^+; see subsection 3.1 and, in particular, Figure 2. The construction of the continuation function f^c for the given function f relies on the use of three main parameters associated with the 1D blending-to-zero approach presented in section 2, namely, d (number of points in the set $\mathcal{D}$), C (number of unrefined discrete continuation points), and n_{os} (oversampling factor for the 1D blending-to-zero FC procedure), together with the parameters n_r (refinement factor for the discrete continuation points; Remark 2.2 and subsection 3.1), and $M-1$ (degree of the interpolating polynomials; subsections 3.2 and 3.4). Additionally, the 2D-FC procedure utilizes the precomputed matrices $\boldsymbol{A}_r$ and $\boldsymbol{Q}$, which, with reference to subsection 2.2, are obtained as per the description provided in Remark 2.2.

Using the aforementioned parameters and matrices, the algorithm proceeds in two main steps, namely, step (a) A *geometrical setup* precomputation procedure (comprising points 1 through 4 below); and step (b) A *Cartesian evaluation* procedure

(comprising points 5 through 7 below). Part (a) only concerns geometry, and, for a given domain and configuration, could be produced once and used for evaluation of FCs for many different functions, as is often necessary in practice. The necessary normals may be obtained by employing explicit expressions for the domain boundary when available (as is the case for all of the numerical examples considered in this paper, as well as cases for which computer aided design representations are used), but numerical evaluation of the boundary normals, e.g., based on polynomial or even Fourier approximation, could be utilized instead.

The full 2D-FC algorithm thus proceeds according to the following seven-step procedure:

1. Discretize Γ using a smooth parametrization $\boldsymbol{q}(\theta) = (x(\theta), y(\theta))$ $(0 \le \theta \le 2\pi)$ and the uniform discretization $I_B = \{0 = \theta_0 < \theta_1, \ldots, < \theta_{B-1} < 2\pi\}$ of the interval $[0, 2\pi]$ (subsection 3.1).
2. Using the discretization I_B, construct two curvilinear meshes $V^-_{B,d}$ and V^+_{B,C_r} in the near-boundary regions V^- (within $\overline{\Omega}$) and V^+ (outside Ω), respectively ((3.5) and (3.6)). Note that the discrete boundary points $\boldsymbol{q}(\theta_j)$ with $\theta_j \in I_B$ are common to both $V^-_{B,d}$ and V^+_{B,C_r}.
3. Determine the set $H \cap V^+$ of Cartesian grid points and construct the proximity map $\mathcal{P} : H \cap V^+ \to V^+_{B,C_r}$ (subsection 3.3).
4. For all $Q \in H \cap V^+$ obtain the the parameter value $\theta^Q \in [0, 1]$ of the foot $\mathcal{F}(Q)$ of the normal through Q (subsection 3.4 and the left image in Figure 4).
5. For each normal grid line (inherent in $V^-_{B,d}$ and V^+_{B,C_r}) given by the discretization I_B, compute the blending-to-zero function values along that normal (subsection 3.2).
6. For all the points $Q \in H \cap V^+$, obtain the continuation function value at Q by local 2D interpolation (subsection 3.4 and the right image in Figure 4).
7. Apply the 2D FFT once to the continuation function values to obtain the desired Fourier series in (3.1).

3.6. 2D-FC approximation: Numerical results. This section demonstrates the accuracy and efficiency of the proposed 2D-FC method. Use of the 2D-FC method requires selection of specific values for each one of the following parameters (all of which are introduced in sections 2 and 3):

- d: number of points in the boundary section (subsection 2.2).
- C: number of continuation points (subsection 2.2).
- Z: number of zero-matching points (subsection 2.2).
- n_{os}: oversampling factor for the matching procedure (subsection 2.2).
- n_r: refinement factor along the normal directions in V^+_{B,C_r} (Remark 2.2).
- $\mathcal{R}$: the smallest rectangle containing $\Omega \cup V^+$ (section 3).
- $N = N_x \times N_y$: number of points in the uniform spatial grid H (section 3).
- B: number of points in the boundary discretization (section 3).
- $M - 1$: interpolating polynomial degree (sections 3.2 and 3.4).

All the errors reported in this section (namely, the absolute maximum error $\varepsilon^{\text{Abs}}_\infty$, the relative maximum error $\varepsilon^{\text{Rel}}_\infty$ and the relative ℓ_2 error $\varepsilon^{\text{Rel}}_2$) were computed on a Cartesian grid of step size $h/2$ within Ω. The approximations on the finer grid are obtained by means of zero padding [22]. In all of the numerical examples considered in this article the parameter selections were made in accordance with Remark 3.3. The computer system used, in turn, is described in Remark 3.4.

Remark 3.3 (parameter selections). For a given step size h used for the 2D Cartesian grid H, the normal and the boundary step sizes k_1 and k_2 (section 3) were taken to coincide with h: $k_1 = k_2 = h$. The parameter values $C = 27$, $n_{\text{os}} = 20$, $Z = 12$, and $n_r = 6$ were used in the evaluation of the matrices $\boldsymbol{A}_r$ and $\boldsymbol{Q}$ (see subsection 2.2 and Remark 2.2). And, finally, with the exception of the interpolation-degree experiments presented in Example 3.2 (Table 2) and Example 4.2 (Table 3), the interpolating-polynomial degree $(M-1) = (d+2)$ was used for the various matching-point numbers d considered.

Remark 3.4 (computer system). All of the numerical results reported in this paper were run on a single thread of a 3.40 GHz Intel Core i7-6700 processor with 15.4 Gb of 2133 MHz memory, with the exception of Example 3.3, where a single thread of a 2.9 GHz Intel Core i7 processor with 32 Gb of memory was used.

Example 3.1 (performance and efficiency of the 2D-FC method). In our first example we consider a problem of FC approximation of the function $f : \Omega \to \mathbb{R}$ given by

$$f(x, y) = -\sin(5\pi x)\sin(5\pi y) \tag{3.10}$$

on the unit disc $\Omega = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$. The left graph in Figure 5 displays the relative ℓ_∞ and ℓ_2 errors as functions of the number $N_{1\text{D}}$ of grid points in one spatial direction within Ω, in log-log scale, obtained from 2D-FC approximations of the function f, for three different values of polynomial degree d defined in section 3.2, namely, $d = 4$, $d = 5$, and $d = 10$ —demonstrating the respective fourth, fifth, and tenth orders of convergence expected. Higher rates of convergence, which are useful in some cases, can be achieved by using higher values of d, as demonstrated in the context of the Poisson solver in subsection 4.1.1. The corresponding computing costs, including "geometrical setup" cost as well as the "Cartesian evaluation" cost are presented in the right graph of Figure 5. The geometrical setup cost combines the setup time for the grids (section 3.1) $V^-_{B,d}$, V^+_{B,C_r}, and H; the time $T_{\mathcal{P}}$ required for construction of the proximity map $\mathcal{P}$ (subsection 3.3); and the time $T_{\mathcal{F}}$ required

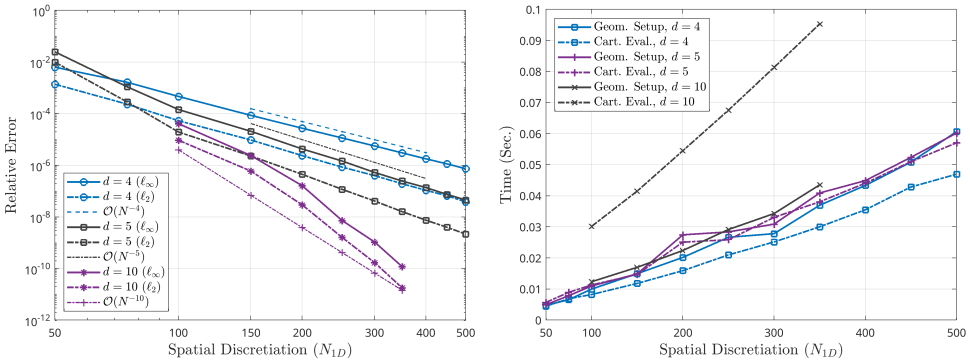


FIG. 5. *Numerical errors in* log-log *scale (left graph) and computing times required (right graph) in the* 2*D-FC approximation of the function* f *in* (3.10) *in the setting of Example* 3.1. *The interpolating polynomial degree* $M = d+3$ *was used in all cases. The integer* $N_{1\text{D}}$ *equals the number of spatial grid points in one spatial direction within* Ω. *Times reported correspond to averages over* 10 *runs.*

TABLE 1

Times (in sec) required by the various tasks in the 2D-FC algorithm in the setting of Example 3.1. *The times reported were calculated as time averages over* 10 *runs. The integer* N_{1D} *equals the number of spatial grid points in one spatial direction within* Ω.

d	h	N_{1D}	B	$T_{\mathcal{P}}$	$T_{\mathcal{F}}$	T_V	T_H	$\varepsilon_\infty^{\text{Rel}}$
4	$2\cdot10^{-2}$	100	313	$2.1\cdot10^{-3}$	$2.6\cdot10^{-3}$	$1.7\cdot10^{-3}$	$6.5\cdot10^{-3}$	$4.7\cdot10^{-4}$
	$1\cdot10^{-2}$	200	628	$4.2\cdot10^{-3}$	$5.2\cdot10^{-3}$	$3.8\cdot10^{-3}$	$1.2\cdot10^{-2}$	$2.7\cdot10^{-5}$
	$5\cdot10^{-3}$	400	1250	$1.0\cdot10^{-3}$	$1.2\cdot10^{-2}$	$1.0\cdot10^{-4}$	$2.5\cdot10^{-2}$	$1.8\cdot10^{-6}$
5	$2\cdot10^{-2}$	100	313	$2.1\cdot10^{-3}$	$2.9\cdot10^{-3}$	$2.2\cdot10^{-3}$	$9.0\cdot10^{-3}$	$1.4\cdot10^{-4}$
	$1\cdot10^{-2}$	200	628	$5.9\cdot10^{-3}$	$6.9\cdot10^{-3}$	$5.8\cdot10^{-3}$	$1.9\cdot10^{-2}$	$4.3\cdot10^{-6}$
	$5\cdot10^{-3}$	400	1250	$1.1\cdot10^{-2}$	$1.3\cdot10^{-2}$	$1.2\cdot10^{-2}$	$3.1\cdot10^{-2}$	$1.4\cdot10^{-7}$
10	$2\cdot10^{-2}$	100	313	$2.2\cdot10^{-3}$	$3.9\cdot10^{-3}$	$4.2\cdot10^{-3}$	$2.6\cdot10^{-2}$	$4.1\cdot10^{-5}$
	$1\cdot10^{-2}$	200	628	$4.3\cdot10^{-3}$	$6.6\cdot10^{-3}$	$8.8\cdot10^{-3}$	$4.7\cdot10^{-2}$	$1.6\cdot10^{-7}$

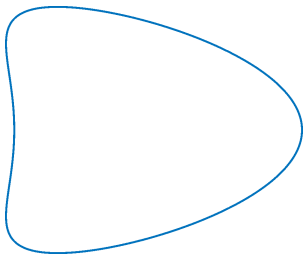


FIG. 6. *Kite-shaped domain.*

for evaluation of the foot of the normal for all points in $H \cap V^+$ (section 3.4). The Cartesian evaluation time, in turn, equals the sum of the time T_V required for evaluation of the f^c values on the curvilinear grid V^+ and time T_H required for subsequent interpolation onto the Cartesian grid H. Table 1 reports additional details concerning computing times required by various tasks associated with the geometrical setup and Cartesian evaluation for this example, including the times $T_{\mathcal{P}}$, $T_{\mathcal{F}}$, T_V, and T_H. The Cartesian interpolation time T_H dominates the overall Cartesian evaluation step (cf. Table 1). In all of these cases we see that the computation time grows linearly with $1/h$. Also, the slope of the Cartesian evaluation cost depends on the degree d whereas the geometrical setup cost, which remains similar in all the cases considered in this example, depends mainly on B and the refinement factor n_r. Note that these costs do not depend upon the approximated function f; rather the costs depend on the resolution of the grids $V^-_{B,d}$, V^+_{B,C_r}, and H, and values of the relevant parameters utilized by various parts of the algorithm.

Remark 3.5 (use of higher degree Gram polynomials). Comparison of the various accuracy and timing values reported in Figure 5 suggests that use of lower 2D-FC degrees such as $d = 4$ or $d = 5$ may provide the highest efficiency for approximation accuracies up to single precision.

Example 3.2 (interpolation degree $(M-1)$ for a given 2D-FC order d). Our next example concerns the approximation of the trigonometric function

$$f = -(x^6 + y^6)\sin(10\pi x)\sin(10\pi y), \tag{3.11}$$

defined over the kite shaped domain (depicted in Figure 6) contained within the curve given by $x(\theta) = \cos(\theta) + 0.35\cos(2\theta) - 0.35$; $y(\theta) = 0.7\sin(\theta)$ for $0 \le \theta \le 2\pi$. A

TABLE 2

Convergence table for the 2D-FC method in the setting of Example 3.2. For both $d = 4$ and $d = 5$, we observe the expected fourth and fifth orders of convergence, respectively, for all the three choices of M, namely, $M = d + 1$, $M = d + 2$, and $M = d + 3$. The value $M = d + 3$ leads to somewhat improved accuracy.

d				$M = d+1$		$M = d+2$		$M = d+3$	
	h	N_x	N_y	$\varepsilon_\infty^{\text{Abs}}$	Ord.	$\varepsilon_\infty^{\text{Abs}}$	Ord.	$\varepsilon_\infty^{\text{Abs}}$	Ord.
4	$1 \cdot 10^{-2}$	297	197	$1.8 \cdot 10^{-3}$	—	$1.4 \cdot 10^{-3}$	—	$9.2 \cdot 10^{-4}$	—
	$5 \cdot 10^{-3}$	537	337	$2.0 \cdot 10^{-4}$	3.2	$9.0 \cdot 10^{-5}$	3.9	$3.1 \cdot 10^{-5}$	4.9
	$2.5 \cdot 10^{-3}$	1017	617	$1.0 \cdot 10^{-5}$	4.3	$4.6 \cdot 10^{-6}$	4.3	$2.3 \cdot 10^{-6}$	3.8
	$1.25 \cdot 10^{-3}$	1977	1177	$6.2 \cdot 10^{-7}$	4.0	$1.4 \cdot 10^{-7}$	5.0	$1.4 \cdot 10^{-7}$	4.0
	$6.25 \cdot 10^{-4}$	3897	2297	$3.4 \cdot 10^{-8}$	4.2	$9.0 \cdot 10^{-9}$	4.0	$9.0 \cdot 10^{-9}$	4.0
5	$1 \cdot 10^{-2}$	297	197	$4.4 \cdot 10^{-3}$	—	$2.3 \cdot 10^{-3}$	—	$2.5 \cdot 10^{-4}$	—
	$5 \cdot 10^{-3}$	537	337	$3.3 \cdot 10^{-4}$	3.7	$4.2 \cdot 10^{-5}$	5.8	$1.5 \cdot 10^{-5}$	4.1
	$2.5 \cdot 10^{-3}$	1017	617	$1.8 \cdot 10^{-5}$	4.2	$4.3 \cdot 10^{-7}$	6.6	$2.6 \cdot 10^{-7}$	5.8
	$1.25 \cdot 10^{-3}$	1977	1177	$4.1 \cdot 10^{-7}$	5.4	$6.4 \cdot 10^{-9}$	6.1	$4.1 \cdot 10^{-9}$	6.0
	$6.25 \cdot 10^{-4}$	3897	2297	$1.2 \cdot 10^{-8}$	5.0	$1.3 \cdot 10^{-10}$	5.6	$1.3 \cdot 10^{-10}$	5.0

TABLE 3

Convergence of the 2D-FC based solution of the Poisson problem described in Example 4.2.

			$M = M_P = d+1$		$M = M_P = d+2$		$M = M_P = d+3$	
h	N_x	N_y	$\varepsilon_2^{\text{Rel}}$	Ord.	$\varepsilon_2^{\text{Rel}}$	Ord.	$\varepsilon_2^{\text{Rel}}$	Ord.
$4 \cdot 10^{-2}$	117	93	$6.0 \cdot 10^{-6}$	—	$7.5 \cdot 10^{-7}$	—	$4.1 \cdot 10^{-7}$	—
$2 \cdot 10^{-2}$	177	127	$1.7 \cdot 10^{-7}$	5.1	$1.0 \cdot 10^{-8}$	6.2	$6.3 \cdot 10^{-9}$	6.0
$1 \cdot 10^{-2}$	297	197	$5.2 \cdot 10^{-9}$	5.0	$1.5 \cdot 10^{-10}$	6.1	$8.7 \cdot 10^{-11}$	6.2
$5 \cdot 10^{-3}$	537	337	$1.6 \cdot 10^{-10}$	5.1	$2.6 \cdot 10^{-12}$	5.9	$1.2 \cdot 10^{-12}$	6.2
$4 \cdot 10^{-3}$	657	407	$5.2 \cdot 10^{-11}$	4.9	$6.8 \cdot 10^{-13}$	6.0	$3.0 \cdot 10^{-13}$	6.2

convergence study for this test case is presented in Table 2 for the values $d = 4$ and $d = 5$, and with $M = d + 1$, $M = d + 2$, and $M = d + 3$. Clearly, the selections $M = d + 2$ and $M = d + 3$ provide similar accuracy in most of the 2D-FC approximation cases considered. The value $M = d + 3$, which yields somewhat better interpolation accuracy for larger step sizes (e.g., $h = 10^{-2}$), and which, as illustrated in Table 3, gives rise to some improvements for all step sizes in the Poisson-problem applications considered in subsection 4.1, was used in all of the numerical experiments presented in this article (except for the cases specifically designed to test the dependence of the accuracy on variations of the parameter M).

Example 3.3 (accuracy and computing-cost comparison). In this example we compare the performance of the 2D-FC method (with $d = 10$) to that provided by the 2D-FE (Fourier extension) method [21], using the author's implementation of the FE method available through the Julia package FrameFun, on a disc of area 4 and for the function

$$f(x, y) = e^{(x+y)} \cos(100xy)$$

(which exhibits approximately 36 oscillations along the diameter). We compare the computing time and memory requirments of these two methods to provide approximations of the function f with similar errors. To achieve an accuracy $\approx 1.7 \cdot 10^{-3}$ in the approximation of f, the Julia implementation of the FE method required 2,555 seconds and used approximately 12 Gb of computer memory whereas the the 2D-FC method required 0.2 seconds and it used less than 200 Mb of computer memory. To achieve an accuracy of $\approx 7 \cdot 10^{-8}$, in turn, the FE method required $4,200$ seconds and

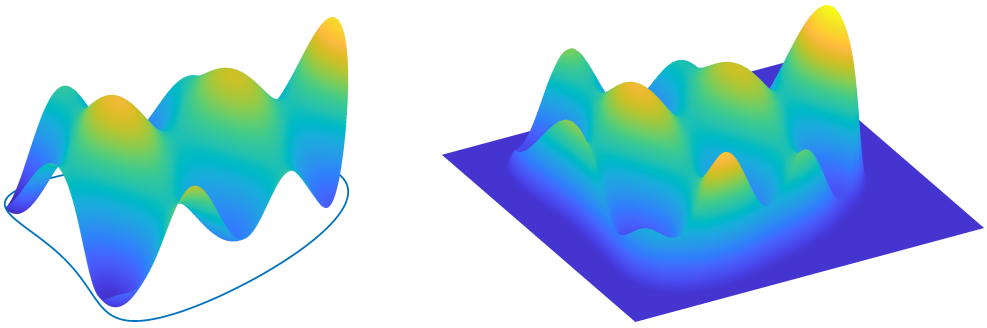


FIG. 7. *Demonstration of the 2D-FC procedure. Continuation of an oscillatory function defined on a kite shaped domain, as detailed in Example* 3.4. *The left and right images display the original and 2D-FC function values, respectively. The blue curve in the left image indicates the boundary of the kite-shaped domain* Ω. *Note the narrowness of the region wherein the transition to zero takes place.*

16 Gb of memory while the 2D-FC method required 0.5 seconds and less than 700 Mb of memory. This simple comparison clearly demonstrates the significance of the $\mathcal{O}(N \log N)$ complexity of the 2D-FC method relative to the $\mathcal{O}(N^2 \log^2 N)$ complexity of the FE approach.

Example 3.4 (graphical illustration of the 2D-FC method). Figure 7 demonstrates the 2D-FC extension method for the function defined by

$$f(x, y) = 4 + (1 + x^2 + y^2)(\sin(2.5\pi x - 0.5) + \cos(2\pi y - 0.5)),$$

over the kite-shaped domain considered in Example 3.2. Both the original function and its extension are presented in Figure 7.

4. Applications of the 2D-FC method. The 2D-FC method introduced in section 3 can be used to facilitate spectral treatment in cases for which iterated use of 1D Fourier expansions does not suffice, but for which use of full 2D Fourier expansions is beneficial. Subsections 4.1 and 4.2 describe two such cases, one concerning solution of the Poisson problem via FE, and, the other, the solution of the wave equation via a novel Fourier forwarding (FF) technique. Several numerical examples are presented for both applications to demonstrate the character of the resulting numerical solvers.

4.1. Application example I: Poisson problem. In this section we present a 2D-FC based method for the solution of the 2D Dirichlet Poisson problem

$$\begin{cases} \Delta u(x, y) = f(x, y), & (x, y) \in \Omega \\ u(x, y) = g(x, y), & (x, y) \in \Gamma; \end{cases} \tag{4.1}$$

the corresponding problem under Neumann or Robin boundary conditions can be treated similarly. The approach is outlined in what follows; a complete description of the method, including, most notably, details concerning the geometrical treatment used for accurate evaluation of the solution near boundaries, is presented in supplement section SM1.

The proposed Poisson solver obtains the solution u within the prescribed tolerance as a sum

$$u = u_p + v \tag{4.2}$$

of a "particular solution" u_p, produced by means of the 2D-FC method, which satisfies the Poisson equation $\Delta u = f$ (but which generically does not satisfy the boundary conditions), and a solution v of the "homogeneous problem," produced by means of a boundary-integral equation, which satisfies the Dirichlet boundary-value problem for Laplace's equation

$$\begin{cases} \Delta v(x,y) = 0, & (x,y) \in \Omega, \\ v(x,y) = g_{\text{hom}}(x,y), & (x,y) \in \Gamma, \end{cases} \tag{4.3}$$

where

$$g_{\text{hom}}(x,y) = g(x,y) - u_p(x,y)|_{\Gamma}. \tag{4.4}$$

A particular solution u_p for the problem (4.1) can easily be obtained from the 2D-FC expansion $f^c(x,y)$ of the right-hand function $f(x,y)$ (3.1)—in view of the diagonal character of the Laplace operator in Fourier space. We thus obtain

$$u_p(x,y) = -\hat{f}^c_{0,0}(x^2+y^2)/4 + \sum_{\ell=-N_x/2+1}^{N_x/2} \sum_{m=-N_y/2+1}^{N_y/2} b_{\ell,m} e^{2\pi i\left(\frac{\ell x}{L_x}+\frac{my}{L_y}\right)}, \tag{4.5}$$

where

$$b_{\ell,m} = \begin{cases} 0, & \text{if } (\ell,m) = (0,0), \\ \frac{-\hat{f}^c_{\ell,m}}{(2\pi\ell/L_x)^2+(2\pi m/L_y)^2}, & \text{if } (\ell,m) \neq (0,0), \end{cases} \tag{4.6}$$

where one of a variety of possible selections was made for the constant Laplacian term. In view of the asymptotically small factors that relate the Fourier coefficients $b_{\ell,m}$ to the original FC coefficients $\hat{f}^c_{\ell,m}$ it follows that, as illustrated in subsection 4.1.1, the rate of convergence in the overall numerical 2D-FC based solution u is of $\mathcal{O}(h^{d+2})$ if a 2D-FC algorithm of $\mathcal{O}(h^d)$ is utilized to compute the particular solution u_p (provided a sufficiently accurate method is subsequently used for evaluation of the homogeneous solution v).

Values $u_p(\boldsymbol{r})$ of the particular solution at points $\boldsymbol{r}$ on the boundary Γ are required as an input (via (4.4)) in the boundary value problem (4.3) for the Laplace solution v. It is therefore necessary to utilize an efficient method for evaluation of u_p at points $\boldsymbol{r}$ that are not part of the Cartesian mesh H. The straightforward procedure based on direct addition of all terms in (4.5) for each discretization point on Γ does not match the optimal $\mathcal{O}(N\log(N))$ cost asymptotics enjoyed by all the other elements of the algorithm and is therefore avoided. Instead, the proposed algorithm first obtains the values of the right-hand side of (4.5) for all $\boldsymbol{r} \in H$ via a direct application of the FFT algorithm, and, then, using these values, it produces the values for $\boldsymbol{r} \in \Gamma$ via iterated 1D interpolation, as described in [22, sect. 3.6.1]. In order to match the overall order $(d+2)$ accuracy of the overall Poisson solution, 1D polynomial interpolants of degree $(M_P - 1) \geq (d+2)$ (cf. Example 4.2) are used in this context for both the x and y interpolation directions.

The numerical solution of the Laplace equation in (4.3), in turn, can be obtained rapidly and efficiently on the basis of the boundary-integral method (see, e.g., [19]). Relying on the boundary parametrization (3.2), the proposed algorithm incorporates an integral equation with a smooth kernel together with the simple and effective Nyström algorithm presented in [19, sect. 12.2]. Based on trapezoidal-rule quadrature, this algorithm results in highly accurate solutions: in view of the periodicity and smoothness of the solution and the kernel, the approach yields superalgebraically small errors provided the boundary and right-hand side g_{hom} are both smooth. The associated linear system is solved by means of the iterative linear algebra solver *GMRES* [23]. Note that the integrand exhibits a near singular behavior for evaluation points that are near the boundary Γ but that are not on Γ. In order to address this difficulty, the proposed method uses a scheme (some elements of which were introduced in [2, 9]) which, based on local mesh refinement and subsequent interpolation using a polynomial of degree $(M_P - 1)$, successfully resolves this difficulty. A detailed description of this and other aspects concerning the 2D-FC based Poisson solver is presented in supplementary material section SM1.

Once the particular and homogeneous solutions u_p and v have been obtained, the solution u of the Poisson problem is given by (4.2). The numerical convergence rate of the solution produced by the algorithm is mainly determined by the order d of the 2D-FC algorithm used. In all, the method is fast and highly accurate; a few illustrations, including accuracy and timing comparisons with leading solvers, are presented in the following section.

4.1.1. Numerical illustrations for the Poisson problem. The numerical illustrations presented in this section utilize the 2D-FC parameter selections presented in Remark 3.3, with various choices of the order parameter d. In addition, the size of the uniform boundary discretization used by the trapezoidal-rule based Nyström method is taken, for simplicity, to equal N_x—but, of course, in view of the superalgebraic convergence of the trapezoidal-rule quadrature, a smaller discretization size could have been used without sacrificing accuracy. The ℓ_2 and ℓ_∞ errors reported in this section were computed over the Cartesian grid $H \cap \Omega$ unless indicated otherwise. The first Poisson-solver example concerns a simple problem previously considered in [14].

Example 4.1 (high-order 2D-FC based Poisson solution). We consider the Poisson problem 4.1 in the domain $\Omega = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$ with $f = -\sin(2\pi x)$ $\sin(2\pi y)$. The left portion of Figure 8 presents the numerical errors in the solutions produced by the 2D-FC based Poisson solvers for $d = 4$, $d = 6$, and $d = 8$ for $f = -\sin(2\pi x)\sin(2\pi y)$. The observed rates of convergence for all three cases match the expected increased rates of convergence, as discussed in subsection 4.1, that is, rate convergences of orders 6, 8, and 10, respectively. This problem was also considered in [14]. Comparison of the results presented in [14, Figure 8] and those on the left graph in Figure 8 suggests a somewhat better 2D-FC performance for lower accuracies, and it indicates a clearly favorable performance of the 2D-FC based Poisson solver for high accuracies. For instance, a discretization containing $N_{1\text{D}} = 100$ grid points in one spatial direction within Ω (leading to a total of $N_x = N_y \approx N_{1\text{D}} + 2C (C = 27)$ grid points over one length of the extended rectangular computational domain), provides, as shown in Figure 8, an ℓ_2 error $1.4 \cdot 10^{-12}$ whereas, in [14, Figure 8], a similar discretization provides ℓ_2 errors close to 10^{-9}. The 10^{-12} error is achieved in that reference at a number of approximately 275 points in spatial discretization points in each spatial direction. For a different test case in this problem setting we now take

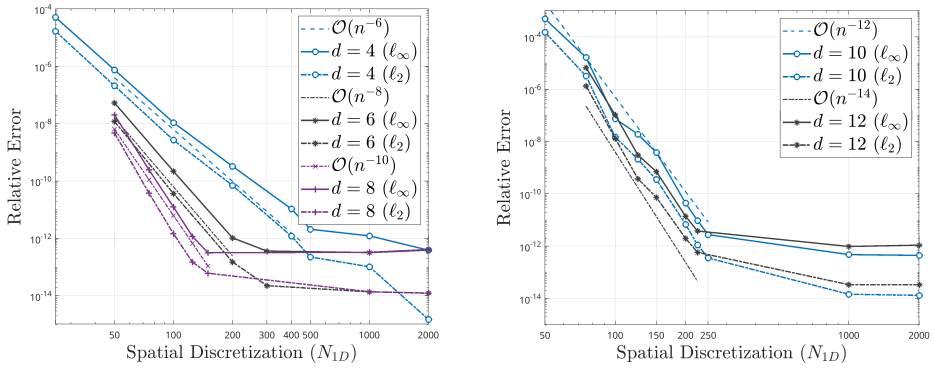


FIG. 8. *Numerical solution errors, in* log-log *scale against the numbers of spatial grid points in one spatial direction within* Ω*, resulting from use of the* 2*D-FC based Poisson solver over the unit disc (Example* 4.1*). Left graph: solution errors for* 2*D-FC methods of orders* $d = 4$*,* $d = 6$ *and* $d = 8$ *for the function* $f = -\sin(2\pi x)\sin(2\pi y)$*. Right graph: solution errors with* $d = 10$ *and* $d = 12$ *for the function* $f = -\sin(5\pi x)\sin(5\pi y)$*. The parameter* N_{1D} *denotes the maximum number of grid points in one spatial direction in* Ω*. A total of* $N_x = N_y \approx N_{1D} + 2C$ *points* ($C = 27$) *were used over each dimension of the periodic square* R*. As discussed in the text, a convergence rate of order* $(d+2)$ *results from use of a* 2*D-FC approximation of order* d*.*

$f = -\sin(5\pi x)\sin(5\pi y)$ (a function that was also used for the convergence study of the 2D-FC algorithm as presented in Example 3.1), and we report, on the right graph in Figure 8, the numerical errors in the solution produced by the solvers for higher values of d, namely, $d = 10$ and $d = 12$. Once again the expected convergence rates (in this case, of orders 12 and 14, respectively) are observed in practice. Additionally, we report the error resulting from a much finer grid than is required to achieve a close to machine level accuracy, illustrating the stability of the approximation method—in that use of arbitrarily fine grids does not cause accuracy degradation.

Example 4.2 (Poisson-solution interpolation degree $(M_P - 1)$). Once again we consider the problem (4.1) over a kite-shaped domain as in Example 3.2 with $f = -\sin(2\pi x)\sin(2\pi y)$. The errors in the solutions produced by the 2D-FC based Poisson solver and the corresponding convergence rates for $d = 4$ and three different values of $M_P (= M)$, namely, $M_P = d+1$, $M_P = d+2$, and $M_P = d+3$, are presented in Table 3. The observed rates of convergence for $M_P = d+2$ and $M_P = d+3$ show the increased $d + 2 =$ sixth-order convergence rate whereas the selection $M_P = d + 1 = 5$ shows a fifth-order convergence as the overall error in the Poisson solution is dominated by the error associated with the order-five interpolation process. The value $M_P = d + 3$ is utilized for all Poisson-problem numerical results presented in this paper.

Example 4.3 (highly oscillatory Poisson problem). Here we consider the problem (4.1) over the kite-shaped domain considered in Example 3.2 with the highly oscillatory right-hand side $f = -\sin(40\pi x)\sin(40\pi y)$. In this example, where we have used $d = 10$ for the 2D-FC particular-solution algorithm, the overall convergence rate, as reported in Table 4, is close to the expected convergence rate of order $(d + 2)$. In order to avoid near-singular integration problems which arise, under the fine discretizations considered in this example, as the numerical solution is evaluated at points very near the boundary Γ, here we report the error at all the Cartesian points within the computational domain that are at a greater distance from Γ than 0.2.

TABLE 4

Interior errors in the numerical solutions produced by the 2D-FC based Poisson solver of order $d = 10$, in the setting of Example 4.3. (Errors are computed over points lying at a distance 0.2 from the domain boundary; see Example 4.3.) A scaling-error even better than the expected order $(d + 2) = 12$ was observed in this case.

h	N_x	N_y	$\varepsilon_\infty^{\text{Rel}}$	Ord.	$\varepsilon_2^{\text{Rel}}$	Ord.
$5 \cdot 10^{-3}$	537	337	$9.3 \cdot 10^{-4}$	—	$3.4 \cdot 10^{-4}$	—
$2.5 \cdot 10^{-3}$	1017	617	$6.0 \cdot 10^{-8}$	13.9	$1.7 \cdot 10^{-8}$	14.3
$1.25 \cdot 10^{-3}$	1977	1177	$5.8 \cdot 10^{-12}$	13.3	$1.6 \cdot 10^{-12}$	13.4

4.2. Application example II: FF method for wave propagation problems. This section presents the 2D-FC based FF method for the solution of the initial boundary value problem

$$u_{tt} = c^2(u_{xx} + u_{yy}) \quad \text{for } (x, y, t) \in \Omega \times \mathbb{R}_+, \tag{4.7}$$

$$u(x, y, t)|_{t=0} = f(x, y), \quad (x, y) \in \Omega, \tag{4.8}$$

$$u_t(x, y, t)|_{t=0} = g(x, y), \quad (x, y) \in \Omega, \tag{4.9}$$

$$u(x, y, t) = h(x, y, t), \quad (x, y, t) \in \Gamma \times \mathbb{R}_+, \tag{4.10}$$

for the 2D wave equation with constant coefficients. A similar FF approach for linear hyperbolic systems with piecewise constant anisotropic media, which can also be developed, is left for future work.

In order to obtain a solution of this problem, the FF method (1) exploits the fact (also used in [16, 25] in the context of biperiodic problems) that the solution of the wave equation in all of $\mathbb{R}^2$ with the initial data $u(\boldsymbol{r}, 0) = ae^{i\boldsymbol{\kappa}\cdot\boldsymbol{r}}$ and $u_t(\boldsymbol{r}, 0) = be^{i\boldsymbol{\kappa}\cdot\boldsymbol{r}}$ is itself given in closed form as a combination of two time-domain plane waves; and, (2) using a sequence $T_j = j\Delta T$ $(j = 0, 1, \dots)$ of times, with an adequately selected "large" time step value ΔT, it constructs auxiliary solutions $U_{\text{FC}}(x, y, t; T_j)$ of (4.7) of the form

$$U_{\text{FC}}(x, y, t; T_j) = \sum_{\ell=-N_x/2+1}^{N_x/2} \sum_{m=-N_y/2+1}^{N_y/2} a_{\ell m}(t; T_j) e^{2\pi i \left(\frac{\ell x}{L_x} + \frac{my}{L_y}\right)}, \tag{4.11}$$

valid for $T_j \le t \le T_{j+1} + q\Delta t$ (with $q = 1$ or $q = 2$ and where Δt is a small time step, as described in what follows) in a spatial region away from the domain boundary, by applying the 2D-FC algorithm to the numerical solution at times $T_j = j\Delta T$ $(j = 0, 1, \dots)$. In detail, in view of the limited domain of dependence of solutions of the wave equation [18], the auxiliary solution $U_{\text{FC}}(x, y, t; T_j)$ with $T_j \le t \le T_{j+1} + q\Delta t$ provides a valid numerical approximation of the solution $u(x, y, t)$ over the subset $\Omega_\delta = \{\boldsymbol{r} = (x, y) \in \Omega : \text{dist}(\boldsymbol{r}, \Gamma) > \delta\} \subset \Omega$ with $\delta = c(\Delta T + q\Delta t)$. To compute the solution $U_B(x, y, t; T_j)$ on the region $\Omega \setminus \Omega_\delta$ adjacent to the physical boundary, the FF method uses a classical time-stepping scheme, with spatial derivatives obtained by means of the 1D-FC method, and using a (typically much smaller) time step Δt, which is taken small enough so as to ensure stability (as dictated by the CFL condition) and to yield an accuracy level consistent with that inherent in the 2D-FC approximation used. The selection $\Delta t = \Delta T/K$ for a suitably chosen integer $K > 0$ is used to ensure both the FC-forwarding and the time-stepping solutions are available at time $t = T_{j+1}$. The $q\Delta t$ term in the time interval $T_j \le t \le T_{j+1} + q\Delta t$ is included to enable evaluation of the time derivative of the near-boundary solution at time T_{j+1} by

means of a centered difference scheme, centered at $t = T_{j+1}$, with an order of accuracy consistent with those used for near-boundary time stepping. In particular, the value $q = 1$ (resp., $q = 2$) is utilized in conjunction with the third- (resp., fifth-) order Taylor series time-stepping method embodied in (4.16) (resp., (4.17)) in section 4.2.2.

The following two sections present the necessary details concerning the application of these ideas for the evaluation of the solution

$$u(x,y,t) = \begin{cases} U_{\mathrm{FC}}(x,y,t;T_j) & \text{for } (x,y,t) \in \Omega_\delta \times [T_j, T_{j+1}], \\ U_{\mathrm{B}}(x,y,t;T_j) & \text{for } (x,y,t) \in (\Omega \setminus \Omega_\delta) \times [T_j, T_{j+1}] \end{cases} \tag{4.12}$$

over a given time interval $T_j \le t \le T_{j+1}$, as well as the time derivative of the solution at $t = T_{j+1}$, on the basis of the numerical values of the solution and its time derivative at $t = T_j$ for $(x,y) \in H \cap \Omega$; this procedure can be used inductively starting from $j = 0$ to enable solution for all j, and, therefore, up to any given final time T. For notational simplicity the argument T_j is suppressed in what follows.

4.2.1. Evaluation of the forwarded solution U_{FC}. In order to obtain the auxiliary solution $U_{\mathrm{FC}}(x,y,t)$ the method utilizes the Cartesian grid H (3.7) together with the 2D-FC expansions

$$\begin{cases} F(x,y) = \sum_{\ell=-N_x/2+1}^{N_x/2} \sum_{m=-N_y/2+1}^{N_y/2} \hat{f}^c_{\ell m} e^{2\pi i \left(\frac{\ell x}{L_x} + \frac{m y}{L_y}\right)}, \\ G(x,y) = \sum_{\ell=-N_x/2+1}^{N_x/2} \sum_{m=-N_y/2+1}^{N_y/2} \hat{g}^c_{\ell m} e^{2\pi i \left(\frac{\ell x}{L_x} + \frac{m y}{L_y}\right)}, \end{cases} \tag{4.13}$$

(cf. (3.1)) of the approximate numerical solution $f_j(x,y) \approx u(x,y,t)|_{t=T_j}$ and its time derivative $g_j(x,y) \approx u_t(x,y,t)|_{t=T_j}$ for $(x,y) \in H \cap \Omega$. (Note that the values of f_0 and g_0 at $t = T_0 = 0$ are given by the initial data $f(x,y)$ and $g(x,y)$; see (4.8) and (4.9)). Clearly, provided the functions $a_{\ell m}(t)$ in (4.11) satisfy the equations

$$\begin{cases} a''_{\ell m}(t) + \alpha_{\ell m} a_{\ell m}(t) = 0, \\ a_{\ell m}(t)|_{t=T_j} = \hat{f}^c_{\ell m}, \\ \frac{\partial a_{\ell m}(t)}{\partial t}|_{t=T_j} = \hat{g}^c_{\ell m} \end{cases} \tag{4.14}$$

for $-N_x/2 + 1 \le \ell \le N_x/2$ and for $-N_y/2 + 1 \le m \le N_y/2$ (where $\alpha_{\ell m} = (2\pi c)^2 \times [(\ell/L_x)^2 + (m/L_y)^2]$), the function $U_{\mathrm{FC}}(x,y,t)$ satisfies (4.7) as well as the initial conditions $U_{\mathrm{FC}}(x,y,t)|_{t=T_j} = F(x,y)$ and $\frac{\partial U_{\mathrm{FC}}(x,y,t)}{\partial t}|_{t=T_j} = G(x,y)$ for $(x,y) \in \mathbb{R}^2$. Substituting the explicit solutions

$$a_{\ell m}(t) = \begin{cases} \hat{f}^c_{\ell m} + t\hat{g}^c_{\ell m} & \text{for } (\ell,m) = (0,0), \\ \hat{f}^c_{\ell m} \cos(\alpha_{\ell m} t) + \dfrac{\hat{g}^c_{\ell m}}{\alpha_{\ell m}} \sin(\alpha_{\ell m} t) & \text{for } (\ell,m) \ne (0,0) \end{cases} \tag{4.15}$$

of the ODE (4.14) into (4.11) the solution $U_{\mathrm{FC}}(x,y,t)$ for $(x,y) \in \mathbb{R}^2$ is obtained. An application of the 2D spatial inverse FFT over the Cartesian grid H to the coefficients $a_{\ell,m}(t)$ then yields, per the discussion above concerning domains of dependence, a numerical approximation of the solution $u(x,y,t)$ for $T_j \le t \le T_{j+1}$ and for all $(x,y) \in H \cap \Omega_\delta$. Similarly, an application of the 2D spatial inverse FFT to the coefficients $a'_{\ell m}(T_{j+1})$ provides a numerical approximation of the time derivative u_t at $t = T_{j+1}$ for $(x,y) \in H \cap \Omega_\delta$, which is needed as initial conditions for the subsequent time step $T_{j+1} \le t \le T_{j+2}$. The d value used for the computation of the solution U_{FC} is denoted by d_{FC} in what follows.

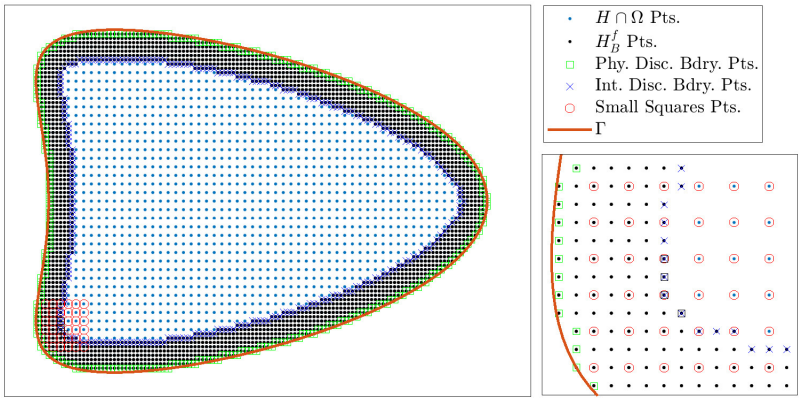


FIG. 9. *Illustration for the FF method. Small rectangular region utilized to forward the auxiliary solution for the black-square interior boundary points.*

4.2.2. Evaluation of the near-boundary solution U_{B}. As suggested above, to obtain the near boundary solution $U_{\mathrm{B}}(x, y, t)$ (see (4.12)) the method utilizes an explicit time stepping scheme in a certain open set $\Omega_B \supset \Omega \setminus \Omega_\delta$ adjacent to the boundary. In this region the method uses a refined uniform spatial grid $\widetilde{H}_B$ of step size $\widetilde{h} = h/2$ which, to maintain accuracy, is needed in view of the interpolatory strategy that is used for enforcement of boundary conditions, as described below. The refined boundary grid $\widetilde{H}_B$ is placed in such a way that alternate points in $\widetilde{H}_B$ coincide with points in $H \cap \Omega_B$ along both the x- and y-coordinates, as illustrated in the right image in Figure 9.

Boundary-condition enforcement. The time-stepping scheme requires that the boundary conditions for U_{B} be enforced at all Cartesian "boundary points" of the discrete set $\widetilde{H}_B$—including those near Γ and those near $\partial\Omega_\delta$, which are, henceforth, called "*physical* discrete boundary points" and "*interior* discrete boundary points" (illustrated as green squares and blue crosses in Figure 9), respectively.

The boundary conditions at physical discrete boundary points are enforced, with high-order accuracy, as proposed in [5], on the basis of a certain interpolation procedure, along grid lines, which utilizes the given boundary values on Γ as well as the solution values at the interior points of $\widetilde{H}_B$. The corresponding boundary condition at the interior discrete boundary points near $\partial\Omega_\delta$, on the other hand, are produced, for efficiency, by means of a special procedure which avoids a full evaluation of the FC expansion for U_{FC} at each small-Δt time interval, and which constructs and uses, instead, solutions similar to (4.11) but over some small square regions, such as the square region span by the red circles in Figure 9, without imposition of boundary conditions over the boundary of the square. In what follows the value of d used for the FC expansions on the small square regions is denoted by d_{ss}. Detailed descriptions of the strategies used for enforcement of the boundary conditions at the physical and interior boundary points are presented in section SM2.

Near-boundary time stepping. Finally, the near boundary solution U_{B} (which is based on use of the small time step Δt on the grid $\widetilde{H}_B$ in Ω_B) may be obtained by means of any adequate time-stepping approach; here we use the Taylor series methods (see [5, sect. 2.2]) of orders three and five (with respective local truncation error of orders $\mathcal{O}(\Delta t^4)$ and $\mathcal{O}(\Delta t^6)$):

$$u(t_{n+1}) \approx 2u(t_n) - u(t_{n-1}) + \Delta t^2 \frac{\partial^2 u}{\partial t^2} + \mathcal{O}(\Delta t^4), \quad \text{and} \tag{4.16}$$

$$u(t_{n+1}) \approx 2u(t_n) - u(t_{n-1}) + \Delta t^2 \frac{\partial^2 u}{\partial t^2} + \frac{\Delta t^4}{12} \frac{\partial^4 u}{\partial t^4} + \mathcal{O}(\Delta t^6), \tag{4.17}$$

respectively. Using (4.7), the time derivatives in the schemes above are replaced by the spatial derivatives, which are themselves computed by means of the 1D-FC method (subsection 2.2; cf. also [4, 11]). (The value of d employed in the application of the 1D-FC algorithm for evaluation of spatial derivatives in the boundary time-stepping procedure is denoted by $d_{\rm ts}$.) To ensure stability of the solver the method utilizes the spectral filter

$$w(x) = \sum_{\ell=-N/2}^{N/2} \hat{w}_\ell \exp(i\ell x) \longrightarrow \sum_{\ell=-N/2}^{N/2} \hat{w}_\ell \exp(-\alpha(2\ell/N)^{2p} \exp(i\ell x) \tag{4.18}$$

(cf. [4] and references therein) with values of the parameters p and α given in subsection 4.2.4. In the $j = 0$ case ($0 \le t \le T_1$) the initial values for $U_{\rm B}$ are obtained from the initial conditions $f(x, y)$ and $g(x, y)$ (see (4.8) and (4.9)); for subsequent time intervals the solution process for $U_{\rm B}$ is simply continued forward in time; no additional initial values are needed for $U_{\rm B}$ at the start of the time intervals $[T_j, T_{j+1}]$ for $j > 0$.

In order to compute numerical values of the time derivative $u_t(x, y, t)$ over the near-boundary grid $(\Omega\backslash\Omega_\delta)\cap H$, which are needed as part of the FC-forwarding initial conditions for the subsequent large time step ΔT (or, more precisely, for $T_{j+1} \le t \le T_{j+2} + q\Delta t$), the method employs the centered finite difference scheme on the basis of the values $U_{\rm B}(x, y, t)$ at the time steps $t = T_{j+1} - q\Delta t, T_{j+1} - (q-1)\Delta t, \ldots, T_{j+1} + q\Delta t$, where q depends on the overall order of the Taylor series method used. In particular, in the context of third-order (resp., fifth-order) Taylor series time stepping the value $q = 1$ (resp., $q = 2$) is used together with the centered difference finite difference scheme with truncation error of order three (resp., five).

4.2.3. Summary and computational cost of the FF method.

FF method summary. Combining the 2D-FC forwarded solution $U_{\rm FC}$ and the near boundary solution $U_{\rm B}$ according to (4.12) the desired FF numerical approximation of the solution u throughout the Cartesian set $H \cap \Omega$ is obtained up to time $t = T_{j+1}$. Repeating this procedure the solution may be advanced up to time $t = T_j = j\Delta T$ for arbitrarily large values of j, and, thus, up to an arbitrary final time T.

FF method computational cost. In view of the fact that auxiliary solutions $U_{\rm FC}(x, y, t)$ need only be computed once per large time step ΔT, a significant improvement results in the asymptotic global computational cost per small time step Δt over the cost required by classical finite difference and finite element spatial discretizations. Indeed, letting $n_c = [\delta/h] > 0$ and assuming the Cartesian mesh $H \cap \Omega$ contains a total of $\mathcal{O}(N)$ grid points, it follows that $\Omega\backslash\Omega_\delta$ contains a total of $\mathcal{O}(\sqrt{N}n_c)$ grid points. It is easy to check that the value of n_c that minimizes the computational cost is $n_c = \mathcal{O}(N^{1/4})$, and, thus, that the computational cost per time step of the overall FF algorithm is $\mathcal{O}(N^{3/4} \log N)$ operations. Interestingly, owing to the large multiplicative constant in front of the asymptotic cost estimate for the time-stepping portion in the boundary region $\Omega \setminus \Omega_\delta$, large increases in N are necessary for the optimal n_c value to increase by one or a few units. Thus, in some of the numerical examples considered in the present paper, for all of which we have $N \le 4 \cdot 10^6$, the

value of n_c is set to a constant. This selection leads to an overall cost estimate of approximately $\mathcal{O}(\sqrt{N}\log N)$ operations for some of the cases considered in this paper as the asymptotically large $\mathcal{O}(N\log N)$ FFT cost incurred by the algorithm has in fact a limited impact in such cases. The performance of the resulting FF method for a number of test cases is demonstrated in subsection 4.2.4 below.

4.2.4. FF: Numerical examples. The numerical examples presented in this section demonstrate the character of the FF method, including high-order convergence, limited dispersion, and sublinear computing cost. In all the cases considered, the 2D-FC parameter selections were made in accordance with Remark 3.3 and the simulations were run on the computer described in Remark 3.4. Either the third- or fifth-order time-stepping schemes in (4.16) and (4.17) were utilized for boundary time stepping (as indicated in each case) with CFL number $\Delta t/\widetilde{h} = 0.1$. The parameter values $p = 4$ and $\alpha = 5$ (resp., $p = 4$ and $\alpha = 8$) were used for the spectral filter (4.18) together with the third- (resp., fifth-) order Taylor series method. The value $e = d_{\rm ts}$ was used for the enforcement of the boundary condition at the physical discrete boundary points near Γ (SM2.2). It is useful to recall here the parameters used to indicate the numbers d of matching points utilized by the various FC-based operations performed, namely,

- the $d = d_{\rm FC}$ value used by the 2D-FC operation to compute the solution $U_{\rm FC}$;
- the $d = d_{\rm ss}$ value used by the FC operation over the small square regions to compute the boundary values at the interior discrete boundary points; and
- the $d = d_{\rm ts}$ value used by the 1D-FC operation for calculating the spatial derivatives in the near boundary region $\Omega \setminus \Omega_\delta$.

Example 4.4 (accuracy and high-order convergence of the FF method). This example employs the method of manufactured solutions to demonstrate the accuracy and convergence of the solutions provided by the FF method. We thus consider the wave equation (4.7) with the initial and (Dirichlet) boundary conditions selected in such a way that the exact solution is given by

$$u(x,y,t) = \cos(2\kappa(x+t)/3) + \cos(\kappa(y+t)) \tag{4.19}$$

(with $\kappa = 5\pi$) on two different domains, namely, the unit disc $\Omega = \{(x,y) \in \mathbb{R}^2 : x^2+y^2 \le 1\}$ and the kite-shaped domain depicted in Figure 6. Tables 5 to 7 report convergence results for the solution $u(x,y,t)$ at time $T = 2$. In particular, Table 5 presents results produced using the third-order Taylor series time-stepping scheme with FC d-values given by $d_{\rm FC} = d_{\rm ss} = d_{\rm ts} = 3$. Table 6 presents results produced by means of the fifth-order Taylor series time-stepping scheme with $d_{\rm FC} = 4, d_{\rm ss} = d_{\rm ts} = 5$. The value $d_{\rm FC} = 4$ was used in the latter test case as it was found that the value $d_{\rm FC} = 5$, matching the Taylor series order, results in numerical instability. Table 7, finally, presents a convergence study for the kite-shaped domain with parameter values $d_{\rm FC} = 4, d_{\rm ss} = d_{\rm ts} = 5$, and a fifth-order Taylor series method for boundary region time stepping. The fixed value $n_c = 6$ (section 4.2.3) was used for Table 5, but the variable relation $n_c = \lfloor (6/10)\sqrt{N_x} \rfloor$ was utilized to produce the results in Tables 6 and 7. In all three cases we observe the expected rate of convergence or better. In particular, in Tables 6 and 7 we observe a near fifth-order convergence despite the use of the value $d_{\rm FC} = 4$ (which yields a fourth-order-accurate 2D-FC algorithm). This behavior is attributed to the high accuracy in the $U_{\rm FC}$ solution for the initial (large) time steps ΔT.

TABLE 5

Convergence study for the solution of the wave equation by means of the FF method in a disc-shaped domain. The third-order Taylor series method was used for time stepping over the boundary region with FC d-values given by $d_{\mathrm{FC}} = d_{\mathrm{ss}} = d_{\mathrm{ts}} = 3$. The fixed value of $n_c = 6$ was used in all cases.

h	N_x	N_y	$\varepsilon_\infty^{\mathrm{Abs}}$	Ord.	$\varepsilon_\infty^{\mathrm{Rel}}$	Ord.	$\varepsilon_2^{\mathrm{Rel}}$	Ord.
$2 \cdot 10^{-2}$	157	157	$1.4 \cdot 10^{-2}$	—	$6.8 \cdot 10^{-3}$	—	$3.5 \cdot 10^{-3}$	—
$1 \cdot 10^{-2}$	257	257	$1.6 \cdot 10^{-3}$	3.1	$7.8 \cdot 10^{-4}$	3.1	$4.1 \cdot 10^{-4}$	3.1
$5 \cdot 10^{-3}$	457	457	$1.4 \cdot 10^{-4}$	3.5	$6.8 \cdot 10^{-5}$	3.5	$3.3 \cdot 10^{-5}$	3.6
$2.5 \cdot 10^{-3}$	857	857	$1.2 \cdot 10^{-5}$	3.5	$5.9 \cdot 10^{-6}$	3.5	$2.9 \cdot 10^{-6}$	3.5
$1.25 \cdot 10^{-3}$	1656	1656	$1.2 \cdot 10^{-6}$	3.3	$6.0 \cdot 10^{-7}$	3.3	$3.0 \cdot 10^{-7}$	3.3

TABLE 6

Convergence study for the solution of the wave equation by means of the FF method in a disc-shaped domain. The fifth-order Taylor series method was used for time stepping over the boundary region with FC d-values given by $d_{\mathrm{FC}} = 4$ and $d_{\mathrm{ss}} = d_{\mathrm{ts}} = 5$.

h	N_x	N_y	$\varepsilon_\infty^{\mathrm{Abs}}$	Ord.	$\varepsilon_2^{\mathrm{Rel}}$	Ord.
$2 \cdot 10^{-2}$	157	157	$3.2 \cdot 10^{-4}$	—	$7.8 \cdot 10^{-5}$	—
$1 \cdot 10^{-2}$	257	257	$1.3 \cdot 10^{-5}$	4.6	$3.1 \cdot 10^{-6}$	4.6
$5 \cdot 10^{-3}$	457	457	$4.1 \cdot 10^{-7}$	5.0	$1.1 \cdot 10^{-7}$	4.8

TABLE 7

Convergence study for the solution of the wave equation by means of the FF method in a kite-shaped domain. The fifth-order Taylor series method was used for time stepping over the boundary region with FC d-values given by $d_{\mathrm{FC}} = 4$ and $d_{\mathrm{ss}} = d_{\mathrm{ts}} = 5$.

h	N_x	N_y	$\varepsilon_\infty^{\mathrm{Abs}}$	Ord.	$\varepsilon_2^{\mathrm{Rel}}$	Ord.
$2 \cdot 10^{-2}$	177	127	$3.8 \cdot 10^{-4}$	—	$1.3 \cdot 10^{-4}$	—
$1 \cdot 10^{-2}$	297	197	$1.7 \cdot 10^{-5}$	4.5	$4.0 \cdot 10^{-6}$	5.0
$5 \cdot 10^{-3}$	537	337	$5.8 \cdot 10^{-7}$	4.9	$1.6 \cdot 10^{-7}$	4.6

Example 4.5 (long-time stability and limited dispersion of the FF method). This test case demonstrates the long-time stability and low dispersion provided by the overall FF method for a propagation problem over the kite-shaped domain depicted in Figure 6, where the exact solution is given by (4.19) with $\kappa = 5\pi$. Figure 10 presents the solution errors up to time $T = 40$ (a total of one hundred time periods) resulting from use of 20 points per wavelength, $d_{\mathrm{FC}} = d_i = d_{\mathrm{ts}} = 4$, and third-order Taylor series time stepping over the near-boundary region. In particular, Figure 10 clearly demonstrates an essentially null dispersion (error growth over time) within this simulation time interval.

Example 4.6 (sublinear computing-cost asymptotics and limited dispersion over a frequency range). In order to demonstrate the favorable computing-cost asymptotics and limited dispersion associated with the FF algorithm over a wide frequency range we utilize once again the unit-disc problem considered in Example 4.4, with solution (4.19), for wave numbers κ varying from $\kappa = 10\pi$ to $\kappa = 120\pi$ (in such a way that the solution spans $w = 10$ to $w = 120$ wavelengths across the domain Ω). The top graph in Figure 11 reports the maximum absolute solution error (computed over all the Cartesian grid points within Ω) at time $T = 2$, at which all w wavelengths have crossed the domain once. The required computational times per small Δt as a function of the number w of wavelengths across Ω are presented on the bottom graph

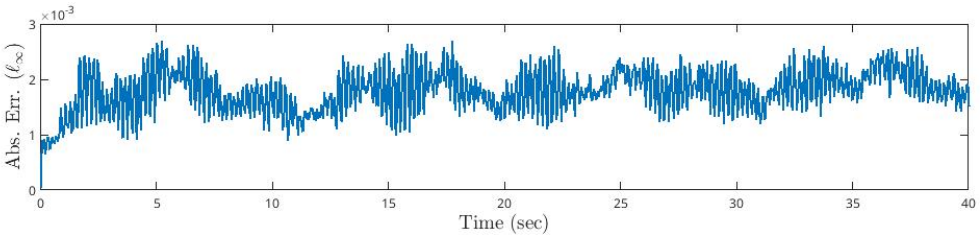


FIG. 10. *Long-time stability study.*

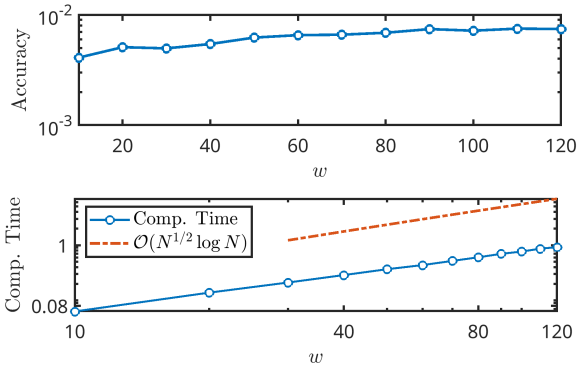


FIG. 11. *Accuracy and computing times (in seconds) for problems of the type considered in Example* 4.4 *on the unit disc, with varying values of* κ. *The third-order Taylor series method was used for time stepping over the boundary region with FC d-values given by* $d_{\mathrm{FC}} = d_{\mathrm{ss}} = d_{\mathrm{ts}} = 4$. *Top graph: maximum absolute errors at time* $T = 2$ *for problems spanning a number* w *of wavelengths across the disc. Bottom graph: computing times required per small-*Δt *time step. Fifteen points per wavelength and a fixed value* $n_c = 6$ *(*$\delta = n_c h$*) were used for this numerical experiment.*

in Figure 11. A fixed fifteen points per wavelength and a fixed value $n_c = 6$ (thus, $\delta = n_c h$ of the boundary region) were used for this numerical experiment. As discussed at the end of subsection 4.2.3, an $\mathcal{O}(N^{1/2}\log N)$ growth in the computational time is observed as the size N of the spatial discretization grows—significantly more favorable than the $\mathcal{O}(N)$ cost that would be required by a regular spatio-temporal finite difference algorithm.

Example 4.7 (interior multiple-scattering illustration). This example provides a graphical illustration of an FF solution, wherein a pulse excitation in the interior of the kite-shaped domain depicted in Figure 6 is enforced by means of the boundary condition

$$u = \cos(\kappa s)\exp(-s^2/\sigma^2)g(t) \tag{4.20}$$

with

$$s = (t - |\boldsymbol{r} - \boldsymbol{r_0}|)/|\boldsymbol{r} - \boldsymbol{r_0}|$$

and

$$g(t) = \begin{cases} 0 & \text{for } t = 0, \\ 1 - \exp\left(\frac{2\exp(-t_0/t)}{t/t_0 - 1.0}\right) & \text{for } t < t_0, \\ 1 & \text{for } t \geqslant t_0, \end{cases} \tag{4.21}$$

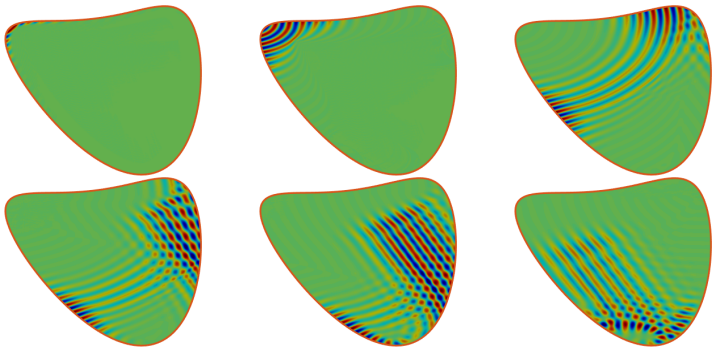


FIG. 12. *Fourier Forwarding method applied to the interior-domain wave propagation problem described in Example* 4.7. *The solution is shown, from top left to bottom right, after* 100, 500, 2050, 2750, 3200 *and* 4300 *timesteps.*

where $\boldsymbol{r_0} = (-1.1, -0.72)$, $t_0 = 0.05$, and $\kappa = 20\pi$. A fixed number $n_c = 8$ of points across the boundary region and slightly over 20 points per wavelength were used. Vanishing initial values of u and u_t at $t = 0$ were enforced; note that, like the initial condition, the imposed boundary values (4.20) vanish at $t = 0$. For improved visualization, the computed solution values presented in the figure were scaled by the maximum solution value at each specific point in time. (The scaling values range approximately between 1 and 3, and they are nearly equal to 1 for the first four images, all three in the upper row, and the leftmost image in the lower row.)

5. Conclusions. This paper introduced a novel 2D-FC method for biperiodic extension of functions defined on arbitrary smooth 2D domains. Applications to the Poisson and wave-equation problem, including the development of the FF method, have resulted in numerical PDE solvers of high orders of accuracy, and, most importantly, of extremely low numerical dispersion. Extensions of these methodologies to problems in higher dimensions, and to problems on nonsmooth domains, are left for future work.

REFERENCES

[1] B. ADCOCK, D. HUYBRECHS, AND J. MARTÍN-VAQUERO, *On the numerical stability of Fourier extensions*, Found. Comput. Math., 14 (2014), pp. 635–687, https://doi.org/10.1007/s10208-013-9158-8.

[2] E. AKHMETGALIYEV, O. P. BRUNO, AND N. NIGAM, *A boundary integral algorithm for the Laplace Dirichlet–Neumann mixed eigenvalue problem*, J. Comput. Phys., 298 (2015), pp. 1–28, https://doi.org/10.1016/j.jcp.2015.05.016.

[3] N. ALBIN AND O. P. BRUNO, *A spectral FC solver for the compressible Navier–Stokes equations in general domains* I*: Explicit time-stepping*, J. Comput. Phys., 230 (2011), pp. 6248–6270, https://doi.org/10.1016/j.jcp.2011.04.023.

[4] F. AMLANI AND O. P. BRUNO, *An FC-based spectral solver for elastodynamic problems in general three-dimensional domains*, J. Comput. Phys., 307 (2016), pp. 333–354.

[5] D. APPELÖ AND N. A. PETERSON, *A fourth-order accurate embedded boundary method for the wave equation*, SIAM J. Sci. Comput., 34 (2012), pp. 2982–3008.

[6] T. ASKHAM AND A. J. CERFON, *An adaptive fast multipole accelerated Poisson solver for complex geometries*, J. Comput. Phys. 344 (2017), pp. 1–22, https://doi.org/https://doi.org/10.1016/j.jcp.2017.04.063.

[7] J. P. BOYD, *A comparison of numerical algorithms for Fourier extension of the first, second, and third kinds*, J. Comput. Phys., 178 (2002), pp. 118–160, https://doi.org/10.1006/jcph.2002.7023.

[8] O. P. Bruno, *Fast, high-order, high-frequency integral methods for computational acoustics and electromagnetics*, in Topics in Computational Wave Propagation, Springer, Berlin, 2003, pp. 43–82, https://doi.org/10.1007/978-3-642-55483-4_2.
[9] O. P. Bruno and B. Delourme, *Rapidly convergent two-dimensional quasi-periodic Green function throughout the spectrum—including Wood anomalies*, J. Comput. Phys., 262 (2014), p. 262–290, https://doi.org/10.1016/j.jcp.2013.12.047.
[10] O. P. Bruno, Y. Han, and M. M. Pohlman, *Accurate, high-order representation of complex three-dimensional surfaces via Fourier continuation analysis*, J. Comput. Phys., 227 (2007), pp. 1094–1125.
[11] O. P. Bruno and M. Lyon, *High-order unconditionally stable FC-AD solvers for general smooth domains* I. *Basic elements*, J. Comput. Phys., 229 (2010), pp. 2009–2033, https://doi.org/10.1016/j.jcp.2009.11.020.
[12] K. T. Elgindy, *A high-order embedded domain method combining a predictor–corrector-Fourier-Continuation-Gram method with an integral Fourier pseudospectral collocation method for solving linear partial differential equations in complex domains*, J. Comput. Appl. Math., 361 (2019), pp. 372–395, https://doi.org/10.1016/j.cam.2019.03.032.
[13] H. Feng and S. Zhao, *FFT-based high order central difference schemes for three-dimensional Poisson's equation with various types of boundary conditions*, J. Comput. Phys., 410 (2020), 109391, https://doi.org/10.1016/j.jcp.2020.109391.
[14] F. Fryklund, E. Lehto, and A.-K. Tornberg, *Partition of unity extension of functions on complex domains*, J. Comput. Phys., 375 (2018), pp. 57–79, https://doi.org/10.1016/j.jcp.2018.08.012.
[15] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 2012.
[16] I. Haber, R. Lee, H. Klein, and J. Boris, *Advances in electromagnetic simulation techniques*, in Proceedings of the Sixth Conference on Numerical Simulation of Plasmas, Lawrence Livermore Laboratory, Berkeley, CA, 1973, pp. 46–48.
[17] D. Huybrechs, *On the Fourier extension of nonperiodic functions*, SIAM J. Numer. Anal., 47 (2010), pp. 4326–4355, https://doi.org/10.1137/090752456.
[18] F. John, *Partial differential equations*, Appl. Math. Sci., (1982), https://doi.org/10.1002/zamm.19730530416.
[19] R. Kress, *Linear Integral Equations*, Applied Math. Sci., 3rd ed., Springer, New York, 2014.
[20] M. Lyon and O. P. Bruno, *High-order unconditionally stable FC-AD solvers for general smooth domains* II. *Elliptic, parabolic and hyperbolic PDEs; theoretical considerations*, J. Comput. Phys., 229 (2010), pp. 3358–3381, https://doi.org/https://doi.org/10.1016/j.jcp.2010.01.006.
[21] R. Matthysen and D. Huybrechs, *Function approximation on arbitrary domains using Fourier extension frames*, SIAM J. Numer. Anal., 56 (2018), pp. 1360–1385, https://doi.org/10.1137/17M1134809.
[22] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd. ed., The Art of Scientific Computing*, Cambridge, Cambridge University Press, 2007.
[23] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., Other Titles Appl. Math. 82, SIAM, Philadelphia, 2003, https://doi.org/10.1137/1.9780898718003.
[24] D. B. Stein, R. D. Guy, and B. Thomases, *Immersed boundary smooth extension: A high-order method for solving PDE on arbitrary smooth domains using Fourier spectral methods*, J. Comput. Phys., 304 (2016), pp. 252–274, https://doi.org/10.1016/j.jcp.2015.10.023.
[25] J.-L. Vay, I. Haber, and B. B. Godfrey, *A domain decomposition method for pseudo-spectral electromagnetic simulations of plasmas*, J. Comput. Phys., 243 (2013), pp. 260–268, https://doi.org/10.1016/j.jcp.2013.03.010.