The Journal of Supercomputing (2019) 75:4947–4985
https://doi.org/10.1007/s11227-019-02772-2

# A parallel unified transform solver based on domain decomposition for solving linear elliptic PDEs

E. N. G. Grylonakis[1] · G. A. Gravvanis[1] · C. K. Filelis-Papadopoulos[1] · A. S. Fokas[2]

Published online: 15 February 2019

## Abstract

A hybrid approach for the solution of linear elliptic PDEs, based on the unified transform method in conjunction with domain decomposition techniques, is introduced. Given a well-posed boundary value problem, the proposed methodology relies on the derivation of an approximate global relation, which is an equation that couples the finite Fourier transforms of all the boundary values. The computational domain is hierarchically decomposed into several nonoverlapping subdomains; for each of those subdomains, a unique approximate global relation is derived. Then, by introducing a modified Dirichlet-to-Neumann iterative algorithm, it is possible to compute the solution and its normal derivative at the resulting interfaces. By considering several hierarchical levels, higher spatial resolution can be achieved. There are three main advantages associated with the proposed approach. First, since the unified transform is a boundary-based technique, the interior of each subdomain does not need to be discretized; thus, no mesh generation is required. Additionally, the Dirichlet and Neumann values can be computed on the interfaces with high accuracy, using a collocation technique in the complex Fourier plane. Finally, the interface values at each hierarchical level can be computed in parallel by considering a quadtree decomposition in conjunction with the iterative Dirichlet-to-Neumann algorithm. The proposed methodology is analysed both regarding implementation details and computational complexity. Moreover, numerical results are presented, assessing the performance of the solver.

**Keywords** Unified transform method · Elliptic PDEs · Domain decomposition · Multilevel methods · Parallel computations

✉ G. A. Gravvanis
ggravvan@ee.duth.gr

Extended author information available on the last page of the article

## 1 Introduction

The numerical solution of boundary value problems (BVPs) for linear elliptic partial differential equations (PDEs) represents an important topic in the area of applied mathematics and scientific computing. In recent years, efforts have been directed towards the development of novel numerical schemes based on the unified transform, or the Fokas method [16–19]. This method can be considered as the spectral analogue of the classical method of boundary integral equations. The unified transform that was introduced by one of the authors in the late 1990s has been used for the solution of a large class of BVPs, including linear and certain nonlinear PDEs. For the solution of a particular well-posed BVP in a polygonal domain, this method involves two basic steps: first, the solution is expressed in terms of integrals in the complex Fourier plane; these integrals contain certain integral transforms of the Dirichlet and Neumann boundary values. Second, the computation of these integral transforms through the analysis of the global relation. For the second task, several numerical schemes have been developed over the last years in order to determine the unknown boundary values [13, 20, 22, 23, 28, 41].

In this paper, considering the second ingredient of the unified transform, i.e. the analysis of the global relation, we propose a class of numerical schemes exhibiting the following properties: high accuracy, avoidance of mesh generation, and inherent parallelism. Regarding the first property, high accuracy is achieved by using a Legendre expansion collocation method in the complex plane for computing the unknown Dirichlet and Neumann values on the boundary of a computational domain. The second property relies on the fact that the collocation method for the global relation represents a boundary-based technique that involves the computation of the relevant expansion coefficients; hence, the solution can be evaluated anywhere on the boundary without the generation of a particular computational grid. Regarding the third property, the inherent parallelism is achieved by considering a domain decomposition technique. In particular, by redesigning an iterative Dirichlet-to-Neumann algorithm in terms of the global relation, and by considering a quadtree-type domain decomposition scheme, it is possible to derive several hierarchical partitions of the domain, where groups of subdomains can be processed independently.

Here, we present the details of the proposed methodology for the Laplace equation on a square as well as on an L-shaped domain; however, the relevant numerical schemes can be used with slight modifications for any linear elliptic PDE that admits a global relation, in a general polygonal domain with an arbitrary number of sides. In addition to the method for solving the given PDE in the interior of the domain, we also propose a technique for estimating the error of the approximated solution and the approximated normal derivative on the boundaries of the domain. This a posteriori error estimator is then used as a stopping criterion for terminating the relevant iterative procedures. Furthermore, it is also used for designing a novel technique for adaptively generating subdomains according to a prescribed threshold criterion. Moreover, we show that above approach can be possibly advantageous for solving PDEs formulated on nonconvex polygons with multiple re-entrant corners.

Recent developments concerning numerical techniques based on the unified transform can be found in [2, 10, 15, 24–27].

Over the last decades, the development in the field of high-performance computing (HPC) has directed the efforts of the research community to design highly parallel techniques. Domain decomposition techniques [34, 39, 42] have been used extensively for solving practical problems in science and engineering. There are several methods for solving PDEs using the domain decomposition approach, based on the finite difference (FDM) [38], the finite element (FEM) [48], the finite volume (FVM) [38], as well as the boundary element (BEM) [40] method and spectral methods [7, 8]. The techniques proposed in this paper share several characteristics with the boundary element method, the spectral as well as the meshless method, forming an entire new class of techniques. Regarding the scope of high-performance scientific computing, work related to the utilization of hardware and software on scalable multiprocessor systems can be found in [1, 4, 5, 30, 43]. In addition, recent developments in preconditioning techniques and parallel solution methods can be found in [21, 29, 31–33, 35, 36, 45, 47].

In Sect. 2, the collocation method for solving the global relation is briefly reviewed. In Sect. 3, an iterative Dirichlet-to-Neumann (DtN) algorithm based on the global relation is introduced. In Sect. 4, a parallel meshless hierarchical solver based on the iterative DtN algorithm is discussed. In Sect. 5, an adaptive technique for subdomain generation based on a posteriori error estimation is also proposed. In Sect. 6, the computational complexity of the proposed schemes is analysed and discussed. Finally, in Sect. 7, numerical results are presented demonstrating the applicability and possible advantages of the proposed techniques.

## 2 Discrete global relations

The global relation is an algebraic equation that couples the finite Fourier transforms of all boundary values. Given a well-posed boundary value problem subject to prescribed boundary conditions, the global relation can be solved either analytically or numerically in order to determine the unknown boundary values. In this paper, we consider the first approach, and below we review the collocation technique that will be used in conjunction with the proposed domain decomposition methodology.

Let us consider the Laplace equation in complex coordinates, on a convex two-dimensional polygon $\Omega$

$$\frac{\partial^2 u}{\partial z \partial \bar{z}} = 0, \; z = x + iy, \; z \in \Omega. \tag{1}$$

Additionally, let us consider the second identity of Green

$$\oint_{\partial\Omega} \left( u \frac{\partial v}{\partial n} - v \frac{\partial u}{\partial n} \right) \mathrm{d}s = 0, \tag{2}$$

as well as the identity

$$\frac{\partial v}{\partial n}\mathrm{d}s = i\left(\frac{\partial v}{\partial \bar{z}}d\bar{z} - \frac{\partial v}{\partial z}\mathrm{d}z\right). \tag{3}$$

Considering the particular solution $v = e^{-i\lambda z}, \lambda \in \mathbb{C}$, derived by the method of separation of variables, and using Eqs. (2) and (3), the following global relation is derived

$$\oint_{\partial\Omega} e^{-i\lambda z}\left[\frac{\partial u}{\partial \eta} + \lambda u \frac{\mathrm{d}z}{\mathrm{d}s}\right]\mathrm{d}s = 0. \tag{4}$$

In order to solve numerically Eq. (4), each side of the polygonal domain $\Omega$ is parameterized using a local parameter $t$, as follows

$$\begin{aligned} z &= m_j + t h_j, t \in [-1, 1], \quad j = 1, 2, \dots, n, \\ m_j &= \frac{z_{j+1} + z_j}{2}, h_j = \frac{z_{j+1} - z_j}{2}. \end{aligned} \tag{5}$$

Then, the discrete version of Eq. (4) takes the form

$$\sum_{j=1}^{n} e^{-i\lambda m_j} \int_{-1}^{1} e^{-i\lambda h_j t}\left[\left|h_j\right| \frac{\partial u_j(t)}{\partial \eta} + \lambda h_j u_j(t)\right]\mathrm{d}t = 0, \lambda \in \mathbb{C}. \tag{6}$$

The collocation-type technique requires the expansion of the boundary values in terms of appropriate basis functions. Following [28], we choose the Legendre polynomials; hence,

$$\begin{aligned} u_j(t) &= \sum_{\ell=0}^{N_\ell - 1} b_\ell^j P_\ell(t), \quad j = 1, \dots, n, t \in [-1, 1], \\ \frac{\partial u_j(t)}{\partial \eta} &= \sum_{\ell=0}^{N_\ell - 1} d_\ell^j P_\ell(t), \quad j = 1, \dots, n, t \in [-1, 1], \end{aligned} \tag{7}$$

where [12]

$$P_\ell(t) = \frac{1}{2^\ell \ell!} \sum_{i=0}^{\ell} (-1)^{\ell - i} \binom{\ell}{i} \frac{(2i)!}{(2i-\ell)!} t^{2i-\ell}. \tag{8}$$

The finite Fourier transform of the Legendre polynomials can be computed using the modified Bessel functions of the first kind, as follows

$$\widehat{P}_\ell(\omega) = \frac{\sqrt{2\pi\omega}}{\omega} I_{\ell + \frac{1}{2}}(\omega). \tag{9}$$

Hence, the discrete global relation (6) becomes

$$\sum_{j=1}^{n} \sum_{\ell=0}^{N_\ell - 1} e^{-i\lambda m_j}\left[\left|h_j\right| b_\ell^j + \lambda h_j d_\ell^j\right]\widehat{P}_\ell(-i\lambda h_j) = 0, \lambda \in \mathbb{C}. \tag{10}$$

By choosing a multitude of values for the complex parameter $\lambda$, according to the heuristic rule

$$\lambda = -\bar{h}_p \frac{R}{M} r, R > 0, M \in \mathbb{Z}^+, p = 1, \dots, n, r = 1, \dots, M, \tag{11}$$

a linear system is derived and its solution determines the expansion coefficients for the unknown boundary values. For the particular case of the Dirichlet problem, the linear system has the following form

$$Ga = Hb, \tag{12}$$

where $G$ and $H$ are nonsquare matrices associated with the Fourier transforms of the Neumann and Dirichlet boundary values, respectively. It should be noted that mixed-type boundary conditions can be implemented by appropriately exchanging columns between the matrices $G$ and $H$.

## 3 An iterative unified transform Dirichlet-to-Neumann algorithm

In this section, we present the main domain decomposition technique, which constitutes the backbone of the proposed methodology and is referred to as the Steklov–Poincare framework [34].

Let us consider the following model problem

$$\begin{aligned} Lu &= f \quad \text{in}\, \Omega, \\ u &= g \quad \text{in}\, \partial\Omega, \end{aligned} \tag{13}$$

where $L$ is a general elliptic operator and $\Omega$ a bounded two-dimensional domain.

Let us also consider a nonoverlapping decomposition of two subdomains $\Omega_1$ and $\Omega_2$, separated by an interface $\Gamma$. At this interface, the following equations, also called as transmission conditions, hold [34]

$$u_1 = u_2, \tag{14}$$

$$\frac{\partial u_1}{\partial n} = \frac{\partial u_2}{\partial n}. \tag{15}$$

The solution of the boundary value problem defined in Eq. (13) can be determined by solving the following coupled system of partial differential equations [34]

$$\begin{aligned} Lu_1 &= f, \quad \text{in}\, \Omega_1, \\ u_1 &= g, \quad \text{on}\, \Omega_1 \backslash \Gamma, \\ u_1 &= u_2, \quad \text{on}\, \Gamma, \\ Lu_2 &= f, \quad \text{in}\, \Omega_2, \\ u_2 &= g, \quad \text{on} \Omega_2 \backslash \Gamma, \\ \frac{\partial u_2}{\partial n} &= \frac{\partial u_1}{\partial n}, \quad \text{on}\, \Gamma. \end{aligned} \tag{16}$$

The Steklov–Poincare system (16) can be solved iteratively using a modified Dirichlet-to-Neumann algorithm based on the Fokas method, namely unified transform Dirichlet-to-Neumann (*UTDtN*). In particular, by solving in an iterative approach a Dirichlet problem on domain $\Omega_1$, as depicted in Fig. 1, followed by the solution of a mixed problem on domain $\Omega_2$ with Neumann boundary conditions on $\Gamma$, using the approximate global relations, the resulting approximate solution converges to the exact solution of problem (13).

Let $P$ be the matrix corresponding to the Legendre polynomials, (8), in $t \in [-1, 1]$. Let us consider a relaxation parameter $\theta$, with $0 < \theta < 1$, in order to ensure the convergence of the iterative Dirichlet-to-Neumann algorithm [34]. The solution of the coupled system (16) is given by the algorithm unified transform Dirichlet-to-Neumann (*UTDtN*), (Algorithm 1).

**Algorithm 1** Unified Transform Dirichlet-to-Neumann (UTDtN) Algorithm

```
1: Form matrices $\{G_j\}_1^2$, $\{H_j\}_1^2$, for each subdomain
2: Solve $Pb = g$ on $(\partial\Omega_1 \cup \partial\Omega_2) \setminus \Gamma$
3: Choose $\theta \in (0,1)$
4: Let $b_\Gamma^0$ denote an initial guess on $\Gamma$
5: for $k = 0, 1, \ldots,$ until convergence do
6:     solve $G_1 \left[a_{\partial\Omega_1}\ a_\Gamma^k\right]^T = H_1 \left[b_{\partial\Omega_1}\ b_\Gamma^k\right]^T + \hat{f}_1$
7:     solve $G_2 \left[a_{\partial\Omega_2}\ b_\Gamma^{k+1}\right]^T = H_2 \left[b_{\partial\Omega_2}\ a_\Gamma^k\right]^T + \hat{f}_2$
8:     $b_\Gamma^{k+1} = \theta b_\Gamma^{k+1} + (1-\theta) b_\Gamma^k$
9: end for
```

In Algorithm 1, $\hat{f}$ denotes the integral transform of the source term, given by

$$\hat{f} = \iint_\Omega e^{-i\lambda z} f(x, y) \mathrm{d}x\mathrm{d}y. \tag{17}$$

Furthermore, in lines 6 and 7 of Algorithm 1, the corresponding columns of matrices $G$ and $H$ should be exchanged accordingly in order to account for the mixed-type boundary conditions.

It should be stated that the *UTDtN* (Algorithm 1) does not require a mesh generation for the two subdomains. Particularly, the solution is computed indirectly by

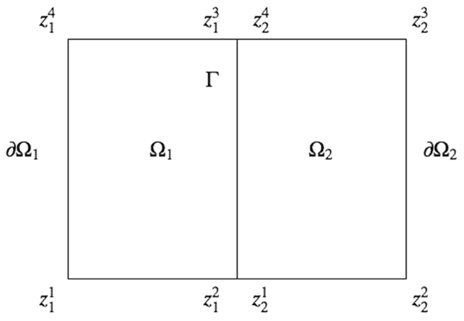


**Fig. 1** A two-subdomain decomposition with an interface $\Gamma$

alternately solving for the expansion coefficients on the boundaries of the two subdomains, $\partial\Omega_1 \cup \Gamma$, $\partial\Omega_2 \cup \Gamma$. The solution and the normal derivative can then be evaluated anywhere along the curve $\partial\Omega_1 \cup \partial\Omega_1 \cup \Gamma$.

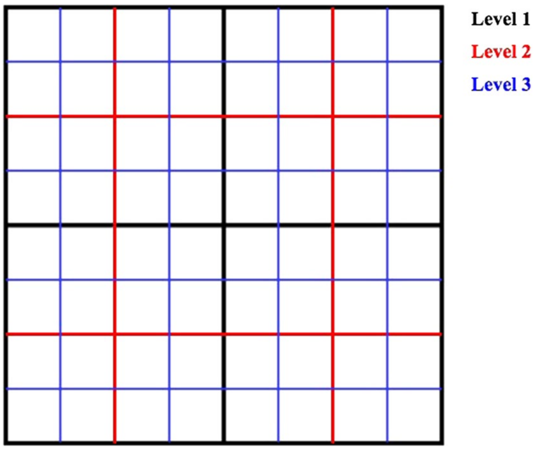
## 4 A parallel meshless hierarchical solver based on the unified transform

In this section, a parallel, meshless, hierarchical solver is proposed for solving linear elliptic PDEs. The solver is based on a modified version of the iterative Dirichlet-to-Neumann algorithm, cf. *UTDtN* (Algorithm 1), and has the particular advantage of avoiding the task of mesh generation, and also it can be implemented in parallel. The proposed method requires a quadtree-type decomposition of the given rectangular domain. In Fig. 2, a hierarchical, quadtree-type decomposition of a square domain, using three successive levels, is depicted. Initially, at Level 1, the domain consists of four subdomains; for each successive level, each subdomain is further decomposed into four new subdomains until the prescribed spatial resolution is achieved. For each successive level, the solution as well as the normal derivative on the interfaces of a group of four subdomains can be computed independently.

The procedure begins by decomposing the initial rectangular domain into four nonoverlapping subdomains. At the resulting interfaces, the Dirichlet as well as the Neumann values are required. The determination of those values is accomplished by modifying the iterative algorithm *UTDtN* (Algorithm 1), in order to be used with four, instead of two, subdomains. In Fig. 3, the process of computing the unknown Dirichlet and Neumann values across the interfaces $\Gamma_{ij}$, $i = 1, 3$, $j = 2, 4$, as implemented by the *UTDtN_init* algorithm, is depicted.

Initially, the four subdomains are grouped into two nonadjacent pairs, as shown in Fig. 3. The process starts by iteratively solving two Dirichlet problems for the subdomains 1 and 3 (red colour), followed by the solution of two mixed problems with Neumann conditions across the interfaces for the subdomains 2 and 4 (blue colour).

**Fig. 2** Quadtree-type decomposition of a square domain using three levels



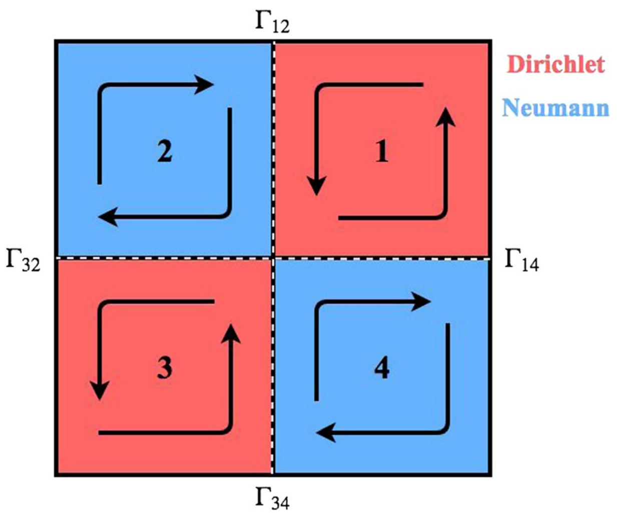


**Fig. 3** The process of computing the unknown Dirichlet and Neumann values across the interfaces of the decomposed domain (color figure online)

The arrows on each subdomain of Fig. 3 indicate the orientation of computing the expansion coefficients of the unknown interface values. For subsequent spatial levels, depending on the position of the new subdomains, the orientation for computing the new interface values changes; in that case, the orientation of the previously computed interface values should be reversed. This is a crucial implementation detail, which is taken into account when proceeding on subsequent spatial levels, as the interface values are used as input in the forthcoming computations. This procedure is described in algorithm *UTDtN_init*, (Algorithm 2). By selecting a maximum number of iterations *MaxIt*, the unknown Dirichlet and Neumann values are computed across the interfaces $\Gamma_{ij}$, $i = 1, 3$, $j = 2, 4$, by solving two Dirichlet problems (lines 8 and 9) followed by the solution of two mixed problems (lines 10 and 11). Then, each of the four subdomains is further decomposed into four new subdomains subject to Dirichlet boundary conditions. In *UTDtN_init* (Algorithm 2), $\partial\Omega$ denotes the boundary of the initial domain (Level 1) and $\left\{\partial\Omega_j\right\}_1^4$ the boundaries of the four subdomains (Level 2). Moreover, $\left\{\bar{a}_{\partial\Omega_j}\right\}_1^4$, $\left\{\bar{b}_{\partial\Omega_j}\right\}_1^4$ are subsets of the expansion coefficients $a$, $b$, corresponding to the sides $\left\{\partial\Omega_j \cap \partial\Omega\right\}_1^4$. In what follows we denote by $a$, $b$ the expansion coefficients of the Neumann and Dirichlet values, respectively, and by $[a, b, c, d]$ the indices for identifying a second-level subdomain.

In Fig. 4, a two-level decomposition of a square domain is depicted, indicating the orientations for computing the respective interface values. In Fig. 5, the process of computing the expansion coefficients of the boundary values of the second level, using the first-level interface values, is shown in detail for the upper left subdomain. The expansion coefficients of the interface values on $\Gamma_{12}$, $\Gamma_{32}$ are used for determining the Dirichlet boundary values on side 1 of the subdomains a, d and on side 4 of

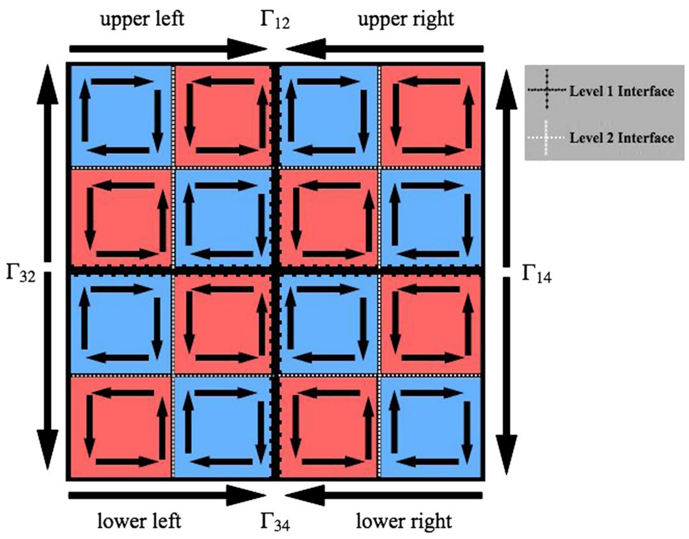


**Fig. 4** The process of computing the unknown Dirichlet and Neumann values across the interfaces of the decomposed domain, using two hierarchical levels

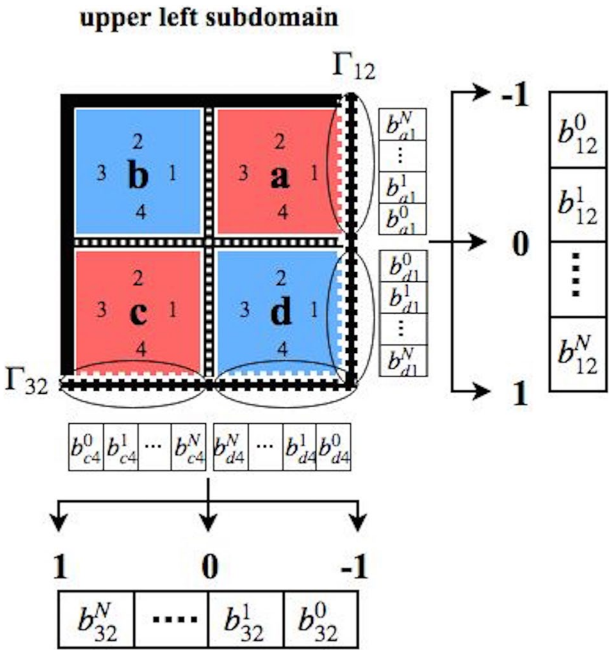


**Fig. 5** The process of obtaining the Legendre expansion coefficients for the Dirichlet boundary values of the second-level subdomains, corresponding to the first-level, upper left subdomain

the subdomains c, d, as depicted in Fig. 5. The interfaces $\Gamma_{12}$, $\Gamma_{32}$ are parameterized on the interval $[-1, 1]$ using equidistant nodes; hence, the boundary values on side 1 of the subdomain a are evaluated on the interval $[-1, 0]$, whereas the boundary values on side 1 of the subdomain d are evaluated on the interval [0, 1], using Eq. (7). Similarly, the boundary values on side 4 of the subdomain d are evaluated on the interval $[-1, 0]$, whereas the boundary values on side 4 of the subdomain c are evaluated on the interval [0, 1], using Eq. (7). Then, the respective Dirichlet expansion coefficients are obtained by solving the following linear systems:

$$\begin{aligned} Pb_{a1} &= u_{a1} \\ Pb_{d1} &= u_{d1} \\ Pb_{c4} &= u_{c4} \\ Pb_{d4} &= u_{d4}. \end{aligned} \tag{18}$$

The process of computing the boundary values on the upper right, lower right, and lower left subdomains can be performed similarly. The above process is implemented for each of the four subdomains, cf. Fig. 3, using four functions that take into consideration the position of each subdomain.

**Algorithm 2** *UTDtN_init*

```
    Choose the order of the Legendre basis functions, $N_\ell$
 2: Choose the maximum number of iterations, $MaxIt$
    Form matrices $\{G_j\}_1^4$, $\{H_j\}_1^4$, for each subdomain
 4: Solve $Pb = g$ on $(\partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \cup \partial\Omega_4) \setminus (\Gamma_{12} \cup \Gamma_{14} \cup \Gamma_{32} \cup \Gamma_{34})$
    Initialize the Dirichlet interface values, $b^0_{\Gamma_{12}}, b^0_{\Gamma_{14}}, b^0_{\Gamma_{32}}, b^0_{\Gamma_{34}}$
 6: Choose $\theta \in (0,1)$
    for $k = 0, 1, ..., MaxIt$ do
 8:     solve $G_1 \left[\bar{a}_{\partial\Omega_1}\ a^k_{\Gamma_{12}}\ a^k_{\Gamma_{14}}\right]^T = H_1 \left[\bar{b}_{\partial\Omega_1}\ b^k_{\Gamma_{12}}\ b^k_{\Gamma_{14}}\right]^T + \hat{f}_1$
        solve $G_3 \left[\bar{a}_{\partial\Omega_3}\ a^k_{\Gamma_{32}}\ a^k_{\Gamma_{34}}\right]^T = H_3 \left[\bar{b}_{\partial\Omega_3}\ b^k_{\Gamma_{32}}\ b^k_{\Gamma_{34}}\right]^T + \hat{f}_3$
10:     solve $G_2 \left[\bar{a}_{\partial\Omega_2}\ b^{k+1}_{\Gamma_{12}}\ b^{k+1}_{\Gamma_{32}}\right]^T = H_2 \left[\bar{b}_{\partial\Omega_2}\ a^k_{\Gamma_{12}}\ a^k_{\Gamma_{32}}\right]^T + \hat{f}_2$
        solve $G_4 \left[\bar{a}_{\partial\Omega_4}\ b^{k+1}_{\Gamma_{14}}\ b^{k+1}_{\Gamma_{34}}\right]^T = H_4 \left[\bar{b}_{\partial\Omega_4}\ a^k_{\Gamma_{14}}\ a^k_{\Gamma_{34}}\right]^T + \hat{f}_4$
12:     $b^{k+1}_{\Gamma_{12}} = \theta b^{k+1}_{\Gamma_{12}} + (1-\theta) b^k_{\Gamma_{12}}$
        $b^{k+1}_{\Gamma_{14}} = \theta b^{k+1}_{\Gamma_{14}} + (1-\theta) b^k_{\Gamma_{14}}$
14:     $b^{k+1}_{\Gamma_{32}} = \theta b^{k+1}_{\Gamma_{32}} + (1-\theta) b^k_{\Gamma_{32}}$
        $b^{k+1}_{\Gamma_{34}} = \theta b^{k+1}_{\Gamma_{34}} + (1-\theta) b^k_{\Gamma_{34}}$
16: end for
```

The respective procedures are given in *UTDtN_upper_right* (Algorithm 3), *UTDtN_upper_left* (Algorithm 4), *UTDtN_lower_right* (Algorithm 5), and *UTDtN_lower_left* (Algorithm 6). Those algorithms receive as input the appropriate Legendre expansion coefficients for the Dirichlet values on the interfaces $\Gamma_{ij}$, $i = 1, 3$, $j = 2, 4$ of the first-level decomposition, which are used for evaluating the Dirichlet boundary values of the newly formatted subdomains. If $\left\{\partial\Omega_j\right\}_1^4$ denote the boundaries of a first-level subdomain and $\left\{\partial\Omega_j\right\}_a^d$ the boundaries of a second-level subdomain, then $\bar{a}_{\partial\Omega_i^j}$, $\bar{b}_{\partial\Omega_i^j}$ are subsets of the expansion coefficients $a$, $b$, corresponding to the sides $\{\partial\Omega_i \cap \partial\Omega_j\}$, $i \in [1,2,3,4]$, $j \in [a,b,c,d]$.

After the initial partitioning of the computational domain, the aforementioned algorithms are applied for each of the four subdomains, independently. In particular, the given domain is firstly decomposed into four subdomains, using *UTDtN_init*, and the respective interface values on $\Gamma_{12}$, $\Gamma_{14}$, $\Gamma_{32}$ and $\Gamma_{34}$ are also computed. Afterwards, for each of the four subdomains, using Algorithms 3, 4, 5, and 6 results in a subsequent decomposition into four new subdomains where the respective interface values are also computed. At the end of this step, a total of 16 subdomains are derived with known Dirichlet boundary values. In a similar manner, at the third level, Algorithms 3, 4, 5, and 6 can be used for each of the 16 subdomains independently for a subsequent decomposition and computation of the corresponding interface values.

At the end of the third-level process, a total of 64 subdomains are derived. The whole procedure can be implemented in parallel, and the parallelism as expected increases when choosing a large number of hierarchical levels.

**Algorithm 3** $UTDtN\_upper\_right(b_{12}, b_{14})$

```
    Choose the order of the Legendre basis functions, $N_\ell$
 2: Choose the maximum number of iterations, $MaxIt$
    Form matrices $\{G_j\}_a^d$, $\{H_j\}_a^d$, for each subdomain
 4: Initialize the Dirichlet interface values, $b^0_{\Gamma_{12}}, b^0_{\Gamma_{14}}, b^0_{\Gamma_{32}}, b^0_{\Gamma_{34}}$
    Choose $\theta \in (0,1)$.
 6: $u_{b3}(t) = \sum_{\ell=0}^{N_\ell-1} b_{12}^\ell P_\ell(t),\ t \in [-1,0]$
    Solve $Pb_{b3} = u_{b3}$
 8: $u_{c3}(t) = \sum_{\ell=0}^{N_\ell-1} b_{12}^\ell P_\ell(t),\ t \in [0,1]$
    Solve $Pb_{c3} = u_{c3}$
10: $u_{c4}(t) = \sum_{\ell=0}^{N_\ell-1} b_{14}^\ell P_\ell(t),\ t \in [-1,0]$
    Solve $Pb_{c4} = u_{c4}$
12: $u_{d4}(t) = \sum_{\ell=0}^{N_\ell-1} b_{14}^\ell P_\ell(t),\ t \in [0,1]$
    Solve $Pb_{d4} = u_{d4}$
14: for $k = 0, 1, ..., MaxIt$ do
        Solve $G_a \left[\bar{a}_{\partial\Omega_1^a}\ a^k_{\Gamma_{ab}}\ a^k_{\Gamma_{ad}}\right]^T = H_a \left[\bar{b}_{\partial\Omega_1^a}\ b^k_{\Gamma_{ab}}\ b^k_{\Gamma_{ad}}\right]^T + \hat{f}_a$
16:     Solve $G_c \left[\bar{a}_{\partial\Omega_1^c}\ a^k_{\Gamma_{cb}}\ a^k_{\Gamma_{cd}}\right]^T = H_c \left[\bar{b}_{\partial\Omega_1^c}\ b^k_{\Gamma_{cb}}\ b^k_{\Gamma_{cd}}\right]^T + \hat{f}_c$
        Solve $G_b \left[\bar{a}_{\partial\Omega_1^b}\ b^{k+1}_{\Gamma_{ab}}\ b^{k+1}_{\Gamma_{cb}}\right]^T = H_b \left[\bar{b}_{\partial\Omega_1^b}\ a^k_{\Gamma_{ab}}\ a^k_{\Gamma_{cb}}\right]^T + \hat{f}_b$
18:     Solve $G_d \left[\bar{a}_{\partial\Omega_1^d}\ b^{k+1}_{\Gamma_{ad}}\ b^{k+1}_{\Gamma_{cd}}\right]^T = H_d \left[\bar{b}_{\partial\Omega_1^d}\ a^k_{\Gamma_{ad}}\ a^k_{\Gamma_{cd}}\right]^T + \hat{f}_d$
        $b^{k+1}_{\Gamma_{ab}} = \theta b^{k+1}_{\Gamma_{ab}} + (1-\theta) b^k_{\Gamma_{ab}}$
20:     $b^{k+1}_{\Gamma_{ad}} = \theta b^{k+1}_{\Gamma_{ad}} + (1-\theta) b^k_{\Gamma_{ad}}$
        $b^{k+1}_{\Gamma_{cb}} = \theta b^{k+1}_{\Gamma_{cb}} + (1-\theta) b^k_{\Gamma_{cb}}$
22:     $b^{k+1}_{\Gamma_{cd}} = \theta b^{k+1}_{\Gamma_{cd}} + (1-\theta) b^k_{\Gamma_{cd}}$
    end for
```

**Algorithm 4** $UTDtN\_upper\_left(b_{12}, b_{32})$

```
    Choose the order of the Legendre basis functions, $N_\ell$
2:  Choose the maximum number of iterations, $MaxIt$
    Form matrices $\{G_j\}_a^d$, $\{H_j\}_a^d$, for each subdomain
4:  Initialize the Dirichlet interface values, $b^0_{\Gamma_{12}}, b^0_{\Gamma_{14}}, b^0_{\Gamma_{32}}, b^0_{\Gamma_{34}}$
    Choose $\theta \in (0,1)$.
6:  $u_{a1}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{12} P_\ell(t),\ t \in [-1,0]$
    Solve $Pb_{a1} = u_{a1}$
8:  $u_{d1}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{12} P_\ell(t),\ t \in [0,1]$
    Solve $Pb_{d1} = u_{d1}$
10: $u_{d4}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{32} P_\ell(t),\ t \in [-1,0]$
    Solve $Pb_{d4} = u_{d4}$
12: $u_{c4}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{32} P_\ell(t),\ t \in [0,1]$
    Solve $Pb_{c4} = u_{c4}$
14: for $k = 0, 1, ..., MaxIt$ do
        Solve $G_a \left[\bar{a}_{\partial\Omega_1^a}\ a^k_{\Gamma_{ab}}\ a^k_{\Gamma_{ad}}\right]^T = H_a \left[\bar{b}_{\partial\Omega_1^a}\ b^k_{\Gamma_{ab}}\ b^k_{\Gamma_{ad}}\right]^T + \hat{f}_a$
16:     Solve $G_c \left[\bar{a}_{\partial\Omega_1^c}\ a^k_{\Gamma_{cb}}\ a^k_{\Gamma_{cd}}\right]^T = H_c \left[\bar{b}_{\partial\Omega_1^c}\ b^k_{\Gamma_{cb}}\ b^k_{\Gamma_{cd}}\right]^T + \hat{f}_c$
        Solve $G_b \left[\bar{a}_{\partial\Omega_1^b}\ b^{k+1}_{\Gamma_{ab}}\ b^{k+1}_{\Gamma_{cb}}\right]^T = H_b \left[\bar{b}_{\partial\Omega_1^b}\ a^k_{\Gamma_{ab}}\ a^k_{\Gamma_{cb}}\right]^T + \hat{f}_b$
18:     Solve $G_d \left[\bar{a}_{\partial\Omega_1^d}\ b^{k+1}_{\Gamma_{ad}}\ b^{k+1}_{\Gamma_{cd}}\right]^T = H_d \left[\bar{b}_{\partial\Omega_1^d}\ a^k_{\Gamma_{ad}}\ a^k_{\Gamma_{cd}}\right]^T + \hat{f}_d$
        $b^{k+1}_{\Gamma_{ab}} = \theta b^{k+1}_{\Gamma_{ab}} + (1-\theta) b^k_{\Gamma_{ab}}$
20:     $b^{k+1}_{\Gamma_{ad}} = \theta b^{k+1}_{\Gamma_{ad}} + (1-\theta) b^k_{\Gamma_{ad}}$
        $b^{k+1}_{\Gamma_{cb}} = \theta b^{k+1}_{\Gamma_{cb}} + (1-\theta) b^k_{\Gamma_{cb}}$
22:     $b^{k+1}_{\Gamma_{cd}} = \theta b^{k+1}_{\Gamma_{cd}} + (1-\theta) b^k_{\Gamma_{cd}}$
    end for
```

**Algorithm 5** $UTDtN\_lower\_right(b_{14}, b_{34})$

```
    Choose the order of the Legendre basis functions, $N_\ell$
2:  Choose the maximum number of iterations, $MaxIt$
    Form matrices $\{G_j\}_a^d$, $\{H_j\}_a^d$, for each subdomain
4:  Initialize the Dirichlet interface values, $b^0_{\Gamma_{12}}, b^0_{\Gamma_{14}}, b^0_{\Gamma_{32}}, b^0_{\Gamma_{34}}$
    Choose $\theta \in (0,1)$.
6:  $u_{a2}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{14} P_\ell(t),\ t \in [0,1]$
    Solve $Pb_{a2} = u_{a2}$
8:  $u_{b2}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{14} P_\ell(t),\ t \in [-1,0]$
    Solve $Pb_{b2} = u_{b2}$
10: $u_{b3}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{34} P_\ell(t),\ t \in [0,1]$
    Solve $Pb_{b3} = u_{b3}$
12: $u_{c3}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{34} P_\ell(t),\ t \in [-1,0]$
    Solve $Pb_{c3} = u_{c3}$
14: for $k = 0, 1, ..., MaxIt$ do
        Solve $G_a \left[\bar{a}_{\partial\Omega_1^a}\ a^k_{\Gamma_{ab}}\ a^k_{\Gamma_{ad}}\right]^T = H_a \left[\bar{b}_{\partial\Omega_1^a}\ b^k_{\Gamma_{ab}}\ b^k_{\Gamma_{ad}}\right]^T + \hat{f}_a$
16:     Solve $G_c \left[\bar{a}_{\partial\Omega_1^c}\ a^k_{\Gamma_{cb}}\ a^k_{\Gamma_{cd}}\right]^T = H_c \left[\bar{b}_{\partial\Omega_1^c}\ b^k_{\Gamma_{cb}}\ b^k_{\Gamma_{cd}}\right]^T + \hat{f}_c$
        Solve $G_b \left[\bar{a}_{\partial\Omega_1^b}\ b^{k+1}_{\Gamma_{ab}}\ b^{k+1}_{\Gamma_{cb}}\right]^T = H_b \left[\bar{b}_{\partial\Omega_1^b}\ a^k_{\Gamma_{ab}}\ a^k_{\Gamma_{cb}}\right]^T + \hat{f}_b$
18:     Solve $G_d \left[\bar{a}_{\partial\Omega_1^d}\ b^{k+1}_{\Gamma_{ad}}\ b^{k+1}_{\Gamma_{cd}}\right]^T = H_d \left[\bar{b}_{\partial\Omega_1^d}\ a^k_{\Gamma_{ad}}\ a^k_{\Gamma_{cd}}\right]^T + \hat{f}_d$
        $b^{k+1}_{\Gamma_{ab}} = \theta b^{k+1}_{\Gamma_{ab}} + (1-\theta) b^k_{\Gamma_{ab}}$
20:     $b^{k+1}_{\Gamma_{ad}} = \theta b^{k+1}_{\Gamma_{ad}} + (1-\theta) b^k_{\Gamma_{ad}}$
        $b^{k+1}_{\Gamma_{cb}} = \theta b^{k+1}_{\Gamma_{cb}} + (1-\theta) b^k_{\Gamma_{cb}}$
22:     $b^{k+1}_{\Gamma_{cd}} = \theta b^{k+1}_{\Gamma_{cd}} + (1-\theta) b^k_{\Gamma_{cd}}$
    end for
```

**Algorithm 6** *$UTDtN\_lower\_left(b_{32}, b_{34})$*

```
    Choose the order of the Legendre basis functions, $N_\ell$
 2: Choose the maximum number of iterations, $MaxIt$
    Form matrices $\{G_j\}_a^d$, $\{H_j\}_a^d$, for each subdomain
 4: Initialize the Dirichlet interface values, $b^0_{\Gamma_{12}}, b^0_{\Gamma_{14}}, b^0_{\Gamma_{32}}, b^0_{\Gamma_{34}}$
    Choose $\theta \in (0,1)$.
 6: $u_{b2}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{12} P_\ell(t)$, $t \in [0,1]$
    Solve $Pb_{b2} = u_{b2}$
 8: $u_{a2}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{12} P_\ell(t)$, $t \in [-1,0]$
    Solve $Pb_{a2} = u_{a2}$
10: $u_{a1}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{14} P_\ell(t)$, $t \in [0,1]$
    Solve $Pb_{a1} = u_{a1}$
12: $u_{d1}(t) = \sum_{\ell=0}^{N_\ell-1} b^\ell_{14} P_\ell(t)$, $t \in [-1,0]$
    Solve $Pb_{d1} = u_{d1}$
14: for $k = 0, 1, ..., MaxIt$ do
        Solve $G_a \left[\bar{a}_{\partial\Omega_1^a}\; a^k_{\Gamma_{ab}}\; a^k_{\Gamma_{ad}}\right]^T = H_a \left[\bar{b}_{\partial\Omega_1^a}\; b^k_{\Gamma_{ab}}\; b^k_{\Gamma_{ad}}\right]^T + \hat{f}_a$
16:     Solve $G_c \left[\bar{a}_{\partial\Omega_1^c}\; a^k_{\Gamma_{cb}}\; a^k_{\Gamma_{cd}}\right]^T = H_c \left[\bar{b}_{\partial\Omega_1^c}\; b^k_{\Gamma_{cb}}\; b^k_{\Gamma_{cd}}\right]^T + \hat{f}_c$
        Solve $G_b \left[\bar{a}_{\partial\Omega_1^b}\; b^{k+1}_{\Gamma_{ab}}\; b^{k+1}_{\Gamma_{cb}}\right]^T = H_b \left[\bar{b}_{\partial\Omega_1^b}\; a^k_{\Gamma_{ab}}\; a^k_{\Gamma_{cb}}\right]^T + \hat{f}_b$
18:     Solve $G_d \left[\bar{a}_{\partial\Omega_1^d}\; b^{k+1}_{\Gamma_{ad}}\; b^{k+1}_{\Gamma_{cd}}\right]^T = H_d \left[\bar{b}_{\partial\Omega_1^d}\; a^k_{\Gamma_{ad}}\; a^k_{\Gamma_{cd}}\right]^T + \hat{f}_d$
        $b^{k+1}_{\Gamma_{ab}} = \theta b^{k+1}_{\Gamma_{ab}} + (1-\theta) b^k_{\Gamma_{ab}}$
20:     $b^{k+1}_{\Gamma_{ad}} = \theta b^{k+1}_{\Gamma_{ad}} + (1-\theta) b^k_{\Gamma_{ad}}$
        $b^{k+1}_{\Gamma_{cb}} = \theta b^{k+1}_{\Gamma_{cb}} + (1-\theta) b^k_{\Gamma_{cb}}$
22:     $b^{k+1}_{\Gamma_{cd}} = \theta b^{k+1}_{\Gamma_{cd}} + (1-\theta) b^k_{\Gamma_{cd}}$
    end for
```

**Algorithm 7** *$UTDtN\_main$*

```
    Choose the number of hierarchical levels, $lev$
 2: $doms\_num = [1, 4, ..., 4^{lev}]$
    $[b_{12}, b_{14}, b_{32}, b_{34}] = UTDtN\_init$
 4: for $k = 1 : lev$ do
        for $i = 1 : doms\_num(k)$, in parallel do
 6:         $UTDtN\_upper\_right(b_{12}, b_{14})$
            $UTDtN\_upper\_left(b_{12}, b_{32})$
 8:         $UTDtN\_lower\_right(b_{14}, b_{34})$
            $UTDtN\_lower\_left(b_{32}, b_{34})$
10:     end for
        update $b_{12}, b_{14}, b_{32}, b_{34}$ for each resulting subdomain
12: end for
```

In Algorithm 7, the *UTDtN_main* parallel procedure is presented. Initially, the number of hierarchical levels, *lev,* is chosen, and afterwards, the *UTDtN_init* algorithm is used in order to partition the original domain into four subdomains. This algorithm also returns the Legendre expansion coefficients for the Dirichlet interface values $b_{ij}$, $i = 1, 3, j = 2, 4$, on $\Gamma_{ij}$, $i = 1, 3, j = 2, 4$. Subsequently, since the Dirichlet boundary values are known for each of the resulting subdomains, the four algorithms *UTDtN_upper_right*, *UTDtN_upper_left*, *UTDtN_lower_right*, and *UTDtN_lower_left* are used for each of the resulting subdomains, in a parallel manner. The above procedure is repeated until the prescribed spatial resolution, i.e. number of levels, is achieved.

In Fig. 6, the parallel computing technique for a three-level domain decomposition process is depicted. Within the scope of a parallel computing environment, a set of tasks is executed by a worker $W_j$ (thread) on a single core of a multicore machine. Hence, on a multicore system, ideally each worker is associated with each core. In this example, the task of decomposing each of the four subdomains of level 1 into a group of four new subdomains at level 2 is taken over by a total of four workers (or four cores). The 16 resulting subdomains can be further decomposed independently by 16 workers (or 16 cores).

## 5 An adaptive subdomain generation technique based on a posteriori error estimation

In this section, we propose a stopping criterion for the iterative procedures in Algorithms 2, 3, 4, 5, and 6. This criterion is based on an a posteriori error estimation technique, using the approximate global relations. The proposed criterion is then used for the design of a self-adaptive, subdomain generation algorithm, in order to increase the spatial resolution in specific areas of the computational domain, where the estimated errors are large.

Let us consider again the approximate global relation in matrix form for a linear elliptic PDE with source term

$$Ga = Hb + \hat{f} + \varepsilon, \tag{19}$$

where $\varepsilon$ is a term associated with the truncation error of the Legendre series expansion, as well as the error introduced when computing the integral transform of the source term, $\hat{f}$, using a numerical quadrature method. Since the exact expansion coefficients for the Dirichlet and Neumann values, $a$, $b$, should satisfy exactly the global relation, the estimator is defined as the maximum error arising when substituting the computed expansion coefficients into Eq. (19).

$$E = \left\| Ga^* - Hb^* - \hat{f} \right\|_\infty . \tag{20}$$

Using Eq. (20), it is possible to prescribe a tolerance parameter, $TOL$, to be used as a stopping criterion for the aforementioned iterative algorithms. Then, the order of the Legendre polynomials $N_\ell$ should be chosen according to this prescribed criterion. In [7], it is shown that the approximation error when using a truncated Legendre series is given by

$$\left\| u - u_{N_\ell} \right\|_{L^2(-1,1)} \leq c N_\ell^{-\sigma} \|u\|_{H^\sigma(-1,1)}, \tag{21}$$

where $H^\sigma(-1,1)$ denotes the Hilbert space with continuous derivatives of order $\sigma$. Using the above relation, and under the hypothesis of a sufficiently smooth function $u$, an exponential convergence rate can be assumed [44]

$$\left\| u - u_{N_\ell} \right\|_{L^2(-1,1)} \sim e^{-\mu N_\ell}, \tag{22}$$

where $\mu \in \mathbb{R}^+$. Hence, the following heuristic formula can be prescribed for the choice of the order of the Legendre polynomials,

$$N_\ell = \left\lceil \frac{-\ln(TOL)}{\mu} \right\rceil. \tag{23}$$

By examining a variety of experimental results, we have found that a good choice for the parameter $\mu$ is $\mu \approx 2$.

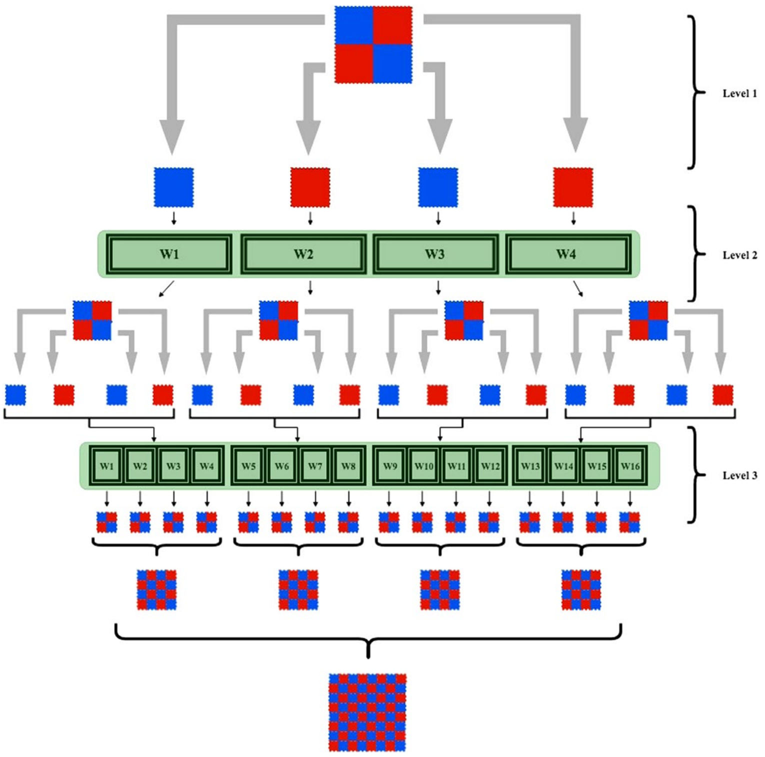


**Fig. 6** The parallel computing technique for a three-level hierarchical decomposition, resulting in 64 subdomains. For each level, each worker (core) $W_j$ takes over the task of generating a group of four subdomains along with approximating the corresponding interface values

**Algorithm 8** *UTDtN_init_m*

```
    Choose the stopping criterion, $TOL$
 2: $N_\ell = \left\lceil \frac{-\ln(TOL)}{2} \right\rceil$
    Form matrices $\{G_j\}_1^4$, $\{H_j\}_1^4$, for each subdomain
 4: Solve $Pb = g$ on $(\partial\Omega_1 \cup \partial\Omega_2 \cup \partial\Omega_3 \cup \partial\Omega_4) \setminus (\Gamma_{12} \cup \Gamma_{14} \cup \Gamma_{32} \cup \Gamma_{34})$
    Initialize the Dirichlet interface values, $b^0_{\Gamma_{12}}, b^0_{\Gamma_{14}}, b^0_{\Gamma_{32}}, b^0_{\Gamma_{34}}$
 6: Choose $\theta \in (0,1)$
    Initialize $E_1$, $E_2$, $E_3$, $E_4$
 8: i=0
    while $(E_1 \geq TOL)$ OR $(E_2 \geq TOL)$ OR $(E_3 \geq TOL)$ OR $(E_4 \geq TOL)$ do
10:     solve $G_1 \left[\bar{a}_{\partial\Omega_1}\ a^k_{\Gamma_{12}}\ a^k_{\Gamma_{14}}\right]^T = H_1 \left[\bar{b}_{\partial\Omega_1}\ b^k_{\Gamma_{12}}\ b^k_{\Gamma_{14}}\right]^T + \hat{f}_1$
        solve $G_3 \left[\bar{a}_{\partial\Omega_3}\ a^k_{\Gamma_{32}}\ a^k_{\Gamma_{34}}\right]^T = H_3 \left[\bar{b}_{\partial\Omega_3}\ b^k_{\Gamma_{32}}\ b^k_{\Gamma_{34}}\right]^T + \hat{f}_3$
12:     solve $G_2 \left[\bar{a}_{\partial\Omega_2}\ b^{k+1}_{\Gamma_{12}}\ b^{k+1}_{\Gamma_{32}}\right]^T = H_2 \left[\bar{b}_{\partial\Omega_2}\ a^k_{\Gamma_{12}}\ a^k_{\Gamma_{32}}\right]^T + \hat{f}_2$
        solve $G_4 \left[\bar{a}_{\partial\Omega_4}\ b^{k+1}_{\Gamma_{14}}\ b^{k+1}_{\Gamma_{34}}\right]^T = H_4 \left[\bar{b}_{\partial\Omega_4}\ a^k_{\Gamma_{14}}\ a^k_{\Gamma_{34}}\right]^T + \hat{f}_4$
14:     $b^{k+1}_{\Gamma_{12}} = \theta b^{k+1}_{\Gamma_{12}} + (1-\theta) b^k_{\Gamma_{12}}$
        $b^{k+1}_{\Gamma_{14}} = \theta b^{k+1}_{\Gamma_{14}} + (1-\theta) b^k_{\Gamma_{14}}$
16:     $b^{k+1}_{\Gamma_{32}} = \theta b^{k+1}_{\Gamma_{32}} + (1-\theta) b^k_{\Gamma_{32}}$
        $b^{k+1}_{\Gamma_{34}} = \theta b^{k+1}_{\Gamma_{34}} + (1-\theta) b^k_{\Gamma_{34}}$
18:     $E_1 = \left\| G_1 \left[\bar{a}_{\partial\Omega_1}\ a^k_{\Gamma_{12}}\ a^k_{\Gamma_{14}}\right]^T - H_1 \left[\bar{b}_{\partial\Omega_1}\ b^{k+1}_{\Gamma_{12}}\ b^{k+1}_{\Gamma_{14}}\right]^T - \hat{f}_1 \right\|_\infty$
        $E_2 = \left\| G_2 \left[\bar{a}_{\partial\Omega_2}\ a^k_{\Gamma_{12}}\ a^k_{\Gamma_{32}}\right]^T - H_2 \left[\bar{b}_{\partial\Omega_2}\ b^{k+1}_{\Gamma_{12}}\ b^{k+1}_{\Gamma_{32}}\right]^T - \hat{f}_2 \right\|_\infty$
20:     $E_3 = \left\| G_3 \left[\bar{a}_{\partial\Omega_3}\ a^k_{\Gamma_{32}}\ a^k_{\Gamma_{34}}\right]^T - H_3 \left[\bar{b}_{\partial\Omega_3}\ b^{k+1}_{\Gamma_{32}}\ b^{k+1}_{\Gamma_{34}}\right]^T - \hat{f}_3 \right\|_\infty$
        $E_4 = \left\| G_4 \left[\bar{a}_{\partial\Omega_4}\ a^k_{\Gamma_{14}}\ a^k_{\Gamma_{34}}\right]^T - H_4 \left[\bar{b}_{\partial\Omega_4}\ b^{k+1}_{\Gamma_{14}}\ b^{k+1}_{\Gamma_{34}}\right]^T - \hat{f}_4 \right\|_\infty$
22:     i=i+1
    end while
```

In Algorithm 8, a modified version of Algorithm 2 is presented, using a stopping criterion for the termination of the iterative procedure. For each iteration, and for each of the resulting subdomains, the maximum norm of the respective global relation, using the resulting expansion coefficients, is computed. The algorithm is terminated once all the estimated errors for each subdomain become less than or equal to the prescribed error tolerance, *TOL*. It should be noted that Algorithms 3, 4, 5, and 6 can be modified similarly.

Using the presented error estimators, it is possible to design an alternative algorithm that adaptively generates subdomains on specific areas and not across the whole domain, based on a certain criterion. Let us consider a first-level decomposition, as depicted in Fig. 3. Then, instead of deriving all 16 subdomains at the second hierarchical level, only the first-level subdomains whose error estimates are above a certain threshold parameter, *TSH*, are further decomposed. The process is then repeated until the desired spatial resolution is obtained.

In Fig. 7, an example of a decomposed computational domain using the adaptive subdomain generation technique, with five hierarchical levels, is shown. For this particular example, a threshold parameter, *TSH*, is assumed to be prescribed. Then, for each hierarchical level, the error estimates of each subdomain are compared to *TSH*; the estimates exceeding the value of *TSH* indicate the subdomains to be further decomposed. For the example presented in Fig. 7, the following subdomains are generated: Level 1: 2/4, Level 2: 4/16, Level 3: 8/64, Level 4: 7/256. In Algorithm 9, a modified version of Algorithm 7 is proposed. For each hierarchical level, the error estimate of each derived subdomain is computed along with the respective interface values. Consequently, only the subdomains that their respective error estimates satisfy the threshold criterion, *TSH*, are decomposed at the next level.

**Algorithm 9** *UTDtN_main_adaptive*

```
    Choose the number of hierarchical levels, $lev$
 2: Choose the adaptivity threshold, $TSH$
    $doms\_num = [1, 4, ..., 4^{lev}]$
 4: $[b_{12}, b_{14}, b_{32}, b_{34}, E_1, E_2, E_3, E_4] = UTDtN\_init$
    for $k = 1 : lev$ do
 6:     for $i \in \left[\{E_j\}_1^{doms\_num(k+1)} > TSH\right]$, in parallel do
            $UTDtN\_upper\_right(b_{12}, b_{14})$
 8:         $UTDtN\_upper\_left(b_{12}, b_{32})$
            $UTDtN\_lower\_right(b_{14}, b_{34})$
10:         $UTDtN\_lower\_left(b_{32}, b_{34})$
        end for
12:     update $b_{12}, b_{14}, b_{32}, b_{34}, E_1, E_2, E_3, E_4$ for each resulting subdomain
    end for
```

## 6 Computational complexity

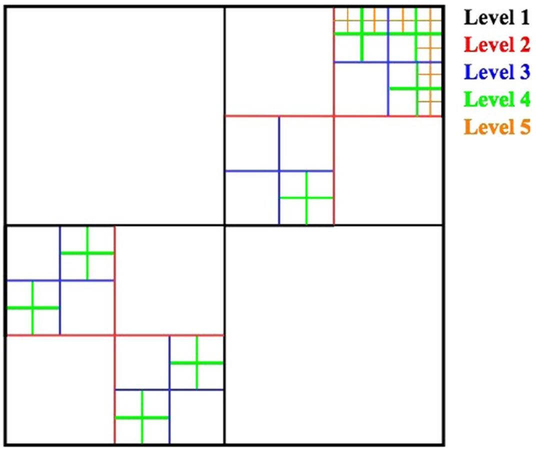
The key step of the proposed algorithms is the evaluation of the discrete global relation, cf. (10). This procedure requires the formation of the coefficient matrices $G$, $H$ and the solution of the resulting linear system. In order to form the coefficient matrices, the finite Fourier transform of the Legendre polynomials has to be computed for each entry, using Eq. (9). The modified Bessel function of the first kind is given by the following equation [37]

$$I_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{\left(\frac{z^2}{4}\right)^k}{k!\Gamma(\nu + k + 1)}. \tag{24}$$

The series described by Eq. (24) is convergent for all $z$; however, when $z \gg 1$, it is not (computationally) practical [37]. For large $z$, a computational approach and its implementation are given in [37]. Here, we discuss the computational complexity considering Eq. (24). It should be noted that for each entry of the coefficient matrices $G$, $H$, the argument of $z$ is given by Eqs. (10) and (11). The magnitude of the argument is given by:

$$|z| = \left|-i\lambda h_j\right| = \left|-i\left(-\bar{h}_p \frac{R}{M} r\right) h_j\right| = \left|i\bar{h}_p h_j \frac{R}{M} r\right|, \; p,j = 1, \ldots, n, \; r = 1, \ldots, M. \tag{25}$$

**Fig. 7** An adaptively decomposed computational domain with five hierarchical levels



Using $M = nN_\ell$, $R = 2M$ [28], for a polygonal domain lying on the unit circle, we have

$$\left|i\bar{h}_p h_j 2r\right| \le 2\left|\bar{h}_p\right|\left|h_j\right||r| = 2nN_\ell,\ p,j = 1,\dots,n. \tag{26}$$

Considering a truncated series $k = 0,\dots,c$ for Eq. (24), we have $\frac{3}{2}c^2 + \frac{7}{2}c + (c+2)\nu + 2$ arithmetic operations (additions and multiplications). The coefficient matrices $G$, $H$ consist of $(2nM)\times(nN_\ell)$ elements. By considering Eqs. (9), (10), setting up the linear system corresponding to the global relation requires $4n^2MN\left(\frac{3}{2}c^2 + \frac{7}{2}c + (c+2)\nu + 3\right) + k_1 + k_2 - 2nM$ arithmetic operations. The constants $k_1$, $k_2$ are associated with the number of arithmetic operations for computing the exponential coefficients appearing in matrices $G$, $H$. Hence, the computational complexity is $\mathcal{O}\left((6c^2 + 4\nu c)n^2MN\right)$.

The next step for determining the unknown boundary values requires the solution of the resulting overdetermined linear system. Here, we have used the QR factorization based on Householder transformations and the computational complexity for a matrix $A \in \mathbb{R}^{m\times n}$ is $2mn^2 - \frac{2}{3}n^3$ [6]. Hence, in our case solving the resulting linear system requires $4n^3MN_\ell^2 - \frac{2}{3}n^3N_\ell^3$. It should be stated that discretizing the global relation results in relatively small, dense coefficient matrices. Increased accuracy can be achieved in most cases by considering Legendre polynomials of order between 10 and 15. In that case, for a square domain, the order of the coefficient matrix is $480 \times 60$, with $N_\ell = 15$.

The proposed solvers *UTDtN_main* and *UTDtN_main_adaptive* rely on the algorithmic schemes *UTDtN_init*, *UTDtN_upper_right*, *UTDtN_upper_left*, *UTDtN_lower_right* and *UTDtN_lower_left*. As discussed above, the most computationally demanding part of formulating the global relation and determining the unknown boundary values is the solution of the resulting linear system using the QR method. Since the key part of the aforementioned algorithms is the repetitive solution of the global relation, the total computational work is determined by the number of linear systems that need to be solved. Each of the algorithmic schemes *UTDtN_init*, *UTDtN_upper_right*, *UTDtN_upper_left*, *UTDtN_lower_right* and *UTDtN_lower_left* requires $4(MaxIt+1)\zeta$ arithmetic operations, where $\zeta = 4n^3MN_\ell^2 - \frac{2}{3}n^3N_\ell^3$. When the a posteriori error estimator is used as a stopping criterion, we have $4i_{l,s}\zeta$ arithmetic operations, where $i_{l,s}$ is the number of iterations performed at the $l_{th}$ level, in the $s_{th}$ subdomain. The total number of arithmetic operations of the algorithmic scheme *UTDtN_main* is described by the following expression:

$$\begin{aligned} &4(MaxIt+1)\zeta + \sum_{j=0}^{lev} 4^j 16(MaxIt+1)\zeta \\ &= 4(MaxIt+1)\zeta + 16(MaxIt+1)\zeta\left(-\frac{1-4^{lev}}{3} + 4^{lev}\right) \\ &= 4(MaxIt+1)\zeta + 16(MaxIt+1)\zeta\left(\frac{4^{lev+1}-1}{3}\right) \\ &= (MaxIt+1)\zeta\left\{4 + 16\left(\frac{4^{lev+1}-1}{3}\right)\right\} \\ &= \left(\frac{4^{lev+3}-4}{3}\right)(MaxIt+1)\zeta. \end{aligned} \tag{27}$$

Analogously, in the case of using the a posteriori estimator, we have:

$$\begin{aligned} &4i_{0,0}\zeta + 4(i_{1,1}+i_{1,2}+i_{1,3}+i_{1,4})\zeta + \cdots \\ &= 4i_{0,0}\zeta + 4\zeta\sum_{m=1}^{lev}\sum_{n=1}^{4^m} i_{m,n} = 4\zeta\left\{i_{0,0} + \sum_{m=1}^{lev}\sum_{n=1}^{4^m} i_{m,n}\right\}. \end{aligned} \tag{28}$$

Hence, the overall computational complexity of *UTDtN_main* is approximately:

$$\mathcal{O}\left(\left(\frac{4^{lev+3}-4}{3}\right)(MaxIt+1)\left(4n^3MN_\ell^2 - \frac{2}{3}n^3N_\ell^3\right)\right). \tag{29}$$

Additionally, when using the a posteriori estimator, the overall computational complexity of *UTDtN_main* is approximately:

$$\mathcal{O}\left(4\left(4n^3MN_\ell^2 - \frac{2}{3}n^3N_\ell^3\right)\left\{i_{0,0} + \sum_{m=1}^{lev}\sum_{n=1}^{4^m} i_{m,n}\right\}\right). \tag{30}$$

The *UTDtN_main_adaptive* algorithm is similar to *UTDtN_main*, based on a posteriori error estimation. The main difference is that at each hierarchical level, a limited number of subdomains are further discretized, according to the threshold parameter *TSH*. In the worst-case scenario, where all of the subdomains are discretized, the approximate computational complexity of *UTDtN_main_adaptive* is given by Eq. (30).

## 7 Numerical results

In this section, the applicability of the proposed methodology is illustrated. In particular, the Laplace equation is solved on a square domain, $[-1, 1]^2$, using the proposed algorithms. In order to assess the accuracy of the approximated solutions, we have used the following closed-form expression

$$u(x, y) = e^{1+x}cos(2 + y). \tag{31}$$

Initially, numerical results concerning the accuracy of the proposed method are given, using the two versions of the *UTDtN* algorithm (maximum iterations, error tolerance) for the Laplace equation on a square domain. Secondly, results are given for the *UTDtN* algorithm, including the adaptive subdomain generation technique based on error estimation for a square domain as well. The third set of numerical results involves the solution of Laplace PDE in an L-shaped domain in the presence of singularity, whereas in the fourth set, results are given for a nonconvex domain with multiple re-entrant corners. The final set of numerical results involves the performance of the parallel version of the *UTDtN* algorithm, whereas a brief comparison to a standard finite element domain decomposition scheme is conducted.

### 7.1 Numerical convergence on a square domain

In this subsection, the accuracy of the approximated Legendre expansion coefficients is assessed. These coefficients correspond to the solution as well as its normal derivative on the interfaces of the resulting subdomains, at each hierarchical level. The numerical errors have been computed with reference to the following equation

$$\frac{\left\| v_{\text{analytical}} - v_{\text{approximated}} \right\|_{\infty}}{\left\| v_{\text{analytical}} \right\|_{\infty}}. \tag{32}$$

In Fig. 8, the computed errors for the Dirichlet $(b_{12}, b_{32}, b_{14}, b_{34})$ and the Neumann $(a_{12}, a_{32}, a_{14}, a_{34})$ expansion coefficients at the interfaces of a first-level decomposed domain, for each iteration with *MaxIt* = 500, are given. In Fig. 9, the computed errors for the Dirichlet and the Neumann expansion coefficients at the interfaces of the four subdomains of a second-level decomposed domain, for each iteration with *MaxIt* = 400, are given. In Figs. 10 and 11, the computed errors for the Dirichlet and the Neumann expansion coefficients at the interfaces of the subdomains 1–8 and 9–16, respectively, are shown for a third-level decomposed domain, for each iteration with *MaxIt* = 400. In Tables 1 and 2, the estimated error, the exact errors for the Dirichlet and the Neumann values, and the total DtN iterations, for three hierarchical levels with TOL = 1E−08, are presented.

It should be noted that the a posteriori error estimation technique is a preferable termination criterion since it is possible to avoid the execution of excess iterations which can occur when assigning an arbitrarily large value for the *MaxIt* parameter.

In Tables 3 and 4, the estimated error, cf. Eq. (20), the exact errors for the Dirichlet and the Neumann values, as well as the total DtN iterations, for three hierarchical levels with TOL = 1E−03, are given.

## 7.2 Adaptive subdomain generation results

In this subsection, numerical results are presented for the *UTDtN* algorithm with adaptive subdomain generation, based on a posteriori error estimation. We present three different cases with various numbers for the error estimate parameter *TOL* and the adaptivity threshold parameter *TSH*. The results involve the presentation of the computational domain after the final adaptive decomposition as well as the corresponding maximum computed errors at each hierarchical level. The numerical results have been obtained with reference to the exact solution given by Eq. (31).

In Fig. 12, an adaptively decomposed subdomain into seven hierarchical levels, with TOL = 1E−03 and TSH = 1E−04, is presented. The corresponding maximum computed error for each hierarchical level is given in Fig. 13.

In Fig. 14, an adaptively decomposed subdomain into seven hierarchical levels, with TOL = 1E−04 and TSH = 5E−06, is shown, and the corresponding maximum computed error for each hierarchical level is given in Fig. 15. In Fig. 16, an adaptively decomposed subdomain into six hierarchical levels, with TOL = 1E−05, is depicted. The threshold parameter *TSH* has been adaptively selected with TSH = 1E−07 at levels 1–2 and TSH = 1E−06 at levels 3–6. The corresponding maximum computed error for each hierarchical level is given in Fig. 17.

The numerical results indicate that the given computational domain can be possibly further decomposed in particular areas where the a posteriori error estimate does not satisfy the prescribed threshold parameter. Moreover, the corresponding computed errors point out that the order of the accuracy is maintained through all hierarchical levels. These adaptive decompositions result in domains that are finer in areas with relatively large error estimates and coarser in the other areas.

## 7.3 L-shaped domain with singularity

Here, we present a test case where the Laplace equation is solved over an L-shaped domain in the presence of a singularity near the origin. Generally, there are two effective ways to successfully solve such a problem. The first approach requires the numerical solution using adaptive mesh refinement techniques near the singular point [46]. The second approach involves the inclusion of singular functions directly in the numerical method [14]. In the context of the Fokas method, supplementing the Legendre basis functions with singular functions results in exponential convergence rates as opposed to algebraic ones when those functions are not included; however, conditioning issues are expected to arise as noted in [11]. Here, the *UTDtN* method without singular functions is considered for solving the above problem and the numerical results are compared to a solution obtained by an adaptive finite element method. Although we expect algebraic convergence as indicated by Theorem 2.8 in [3], we show that in this numerical example, our method is faster than the

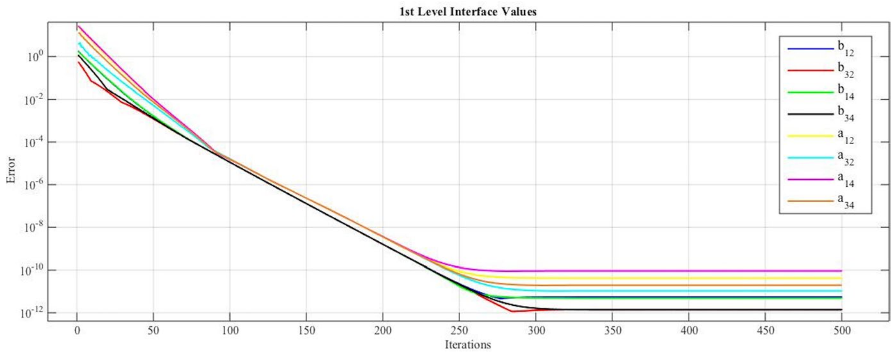


**Fig. 8** The computed errors at each iteration for the Dirichlet and Neumann expansion coefficients, on the interfaces of a first-level decomposed domain, with *MaxIt* = 500

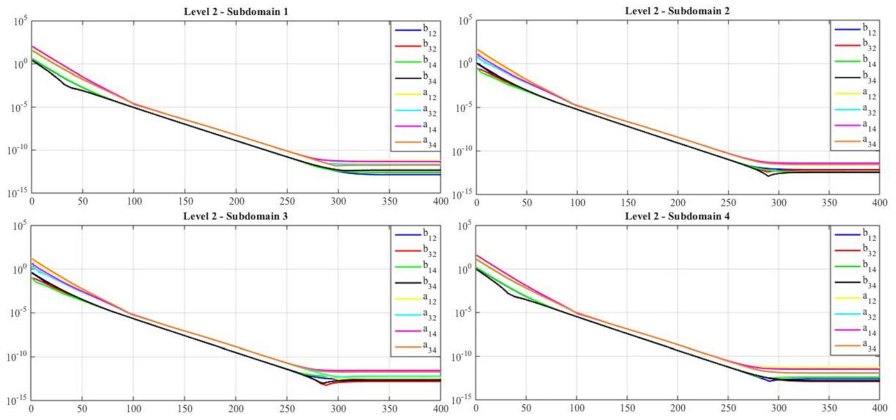


**Fig. 9** The computed errors ($y$-axis) at each iteration ($x$-axis) for the Dirichlet and Neumann expansion coefficients, on the interfaces of a second-level decomposed domain, with *MaxIt* = 400

adaptive FEM. In a future publication, we aim to combine our domain decomposition approach with singular functions in order to improve both the accuracy and the relevant condition number of the linear system.

In Fig. 18, the L-shaped domain and a two-level decomposition are depicted. In Fig. 19, the computational mesh generated by an adaptive finite element method using 22172 triangles is depicted on the left. On the right, the subdomains generated by the *UTDtN* technique where the solution approximated on each boundary are shown. In Fig. 20, the solutions obtained using the adaptive FEM (red colour) and the *UTDtN* technique (black colour) are plotted against each other. Using $N_\ell = 15$ basis functions, we obtain an error of 2.95E−04 near the singularity, by comparing to the solution computed by the finite element method. The time required for computing the solution with the adaptive FEM was 8.21 s, whereas for the *UTDtN* was 4.87 s. It should be mentioned that *UTDtN* used 83 DtN iterations at the first level as well as 145, 148, and 149 iterations for each of the three subdomains at level 2.

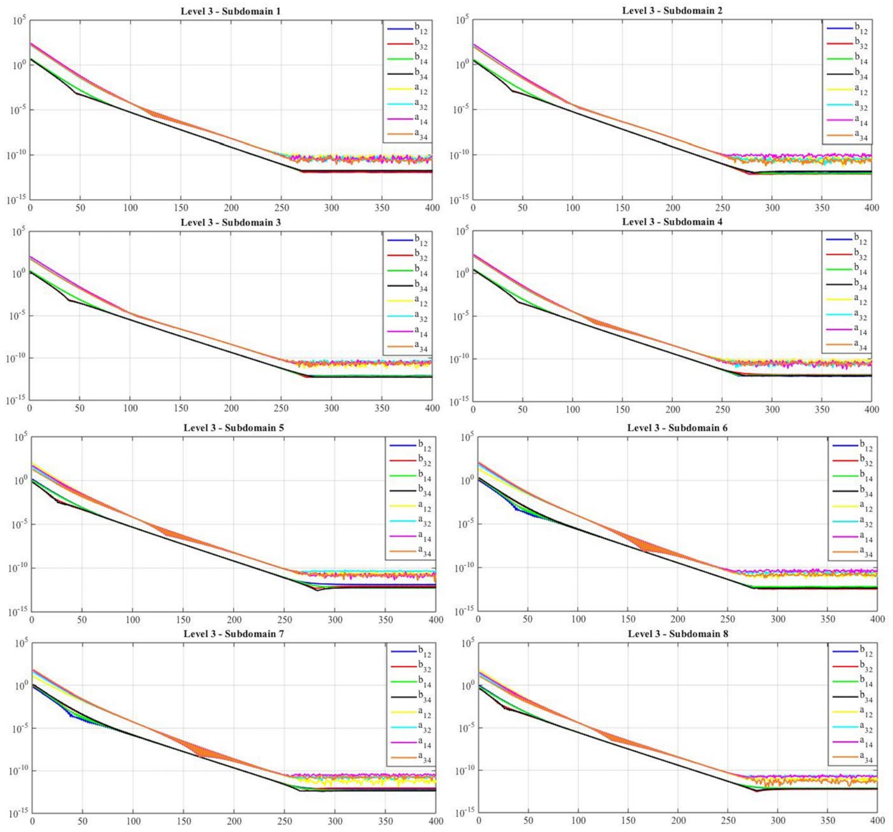


**Fig. 10** The computed errors ($y$-axis) at each iteration ($x$-axis) for the Dirichlet and Neumann expansion coefficients, on the interfaces of a third-level decomposed domain, with $MaxIt = 400$ (subdomains 1–8)

At the third level, three subdomains were generated with 149 DtN iterations, respectively. At the fourth level, four subdomains were generated with 140, 140, 141, and 143 iterations, respectively. Finally, at the fifth level, four subdomains were again generated with 154, 154, 156, and 158 DtN iterations, respectively.

## 7.4 Nonconvex domain with multiple re-entrant corners

In this subsection, we consider the Laplace equation on a relatively complicated nonconvex domain with multiple re-entrant corners. We compare the usual unified transform on the boundary versus the proposed iterative domain decomposition algorithm (UTDtN). For the numerical results, we have used the exact solution given by Eq. (31). In Fig. 21, the nonconvex domain is depicted along with the corresponding ten-subdomain decomposition. We use the UTDtN algorithm in order to compute the solution on the interfaces $\Gamma$. In Fig. 22, the numerical convergence for the solution across the interfaces $\Gamma$ after 1000 iterations and using $N_\ell = 7$

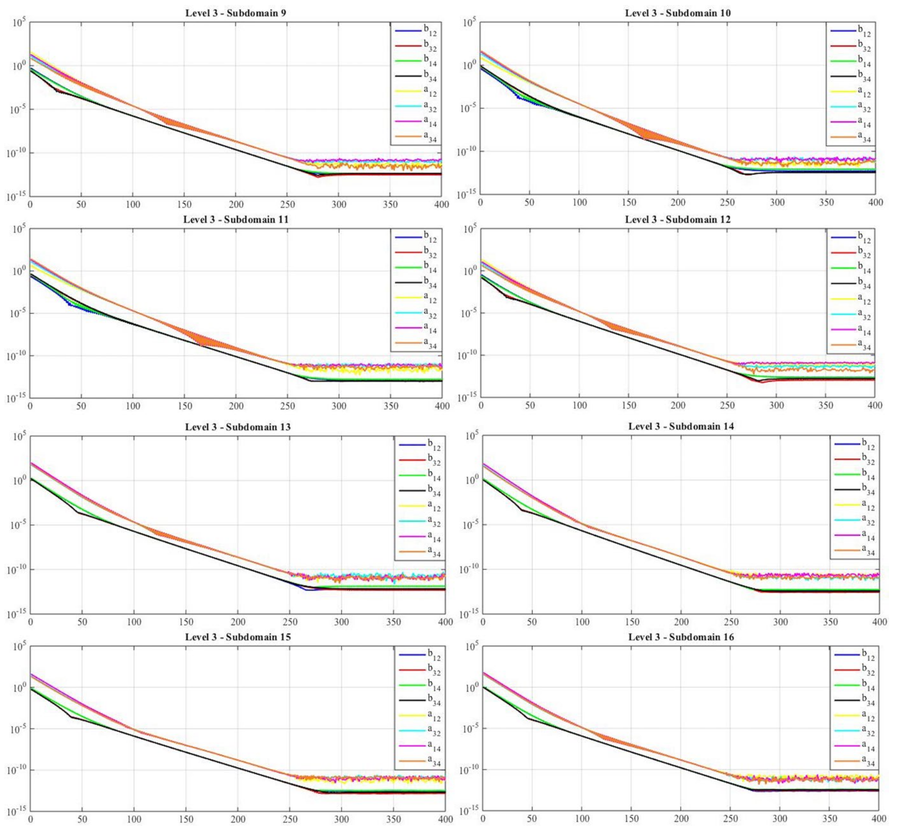


**Fig. 11** The computed errors ($y$-axis) at each iteration ($x$-axis) for the Dirichlet and Neumann expansion coefficients, on the interfaces of a third-level decomposed domain, with $MaxIt = 400$ (subdomains 9–16)

**Table 1** Computed and estimated errors for the interface values at levels 1 and 2, as well as total DtN iterations, with TOL = 1E−03

| | Level 1 | Level 2 | | | |
|---|---|---|---|---|---|
| | – | Subdomain 1 | Subdomain 2 | Subdomain 3 | Subdomain 4 |
| Estimated error | 9.32E−04 | 9.45E−04 | 9.22E−04 | 9.80E−04 | 9.15E−04 |
| Max. error (Dirichlet) | 1.85E−04 | 2.79E−04 | 2.66E−04 | 2.71E−04 | 3.10E−04 |
| Max. error (Neumann) | 6.53E−04 | 9.97E−04 | 1.11E−03 | 1.42E−03 | 1.27E−03 |
| Total DtN iterations | 70 | 67 | 63 | 53 | 58 |

basis functions is given. In Table 5, the condition numbers of the corresponding linear systems are given for the entire nonconvex domain as well as the decomposed subdomains. It is worth mentioning that using the *UTDtN* algorithm, the condition number decreases from 2.84E+18 to a maximum of 4.16E+04. Also, the time to compute all interface values after 1000 iterations was 4.16 s.

**Table 2** Computed and estimated errors for the interface values at level 3, as well as total DtN iterations, with TOL = 1E−03

| | Level 3 | | | |
|---|---|---|---|---|
| | Subdomain 1 | Subdomain 2 | Subdomain 3 | Subdomain 4 |
| Estimated error | 9.70E−04 | 9.10E−04 | 9.37E−04 | 9.11E−04 |
| Max. error (Dirichlet) | 2.06E−04 | 2.51E−04 | 1.93E−04 | 2.67E−04 |
| Max. error (Neumann) | 1.14E−03 | 9.83E−04 | 1.63E−03 | 1.26E−03 |
| Total DtN iterations | 69 | 70 | 65 | 65 |
| | Level 3 | | | |
| | Subdomain 5 | Subdomain 6 | Subdomain 7 | Subdomain 8 |
| Estimated error | 9.94E−04 | 9.95E−04 | 9.52E−04 | 9.13E−04 |
| Max. error (Dirichlet) | 2.39E−04 | 2.09E−04 | 2.41E−04 | 2.44E−04 |
| Max. error (Neumann) | 1.17E−03 | 1.20E−03 | 1.53E−03 | 6.88E−04 |
| Total DtN iterations | 67 | 62 | 58 | 63 |
| | Level 3 | | | |
| | Subdomain 9 | Subdomain 10 | Subdomain 11 | Subdomain 12 |
| Estimated error | 9.45E−04 | 9.27E−04 | 9.37E−04 | 9.95E−04 |
| Max. error (Dirichlet) | 1.79E−04 | 2.83E−04 | 2.21E−04 | 2.29E−04 |
| Max. error (Neumann) | 1.37E−03 | 1.96E−03 | 2.09E−03 | 1.43E−03 |
| Total DtN iterations | 58 | 54 | 50 | 53 |
| | Level 3 | | | |
| | Subdomain 13 | Subdomain 14 | Subdomain 15 | Subdomain 16 |
| Estimated error | 9.81E−04 | 9.87E−04 | 9.51E−04 | 9.75E−04 |
| Max. error (Dirichlet) | 2.80E−04 | 2.89E−04 | 1.37E−04 | 2.66E−04 |
| Max. error (Neumann) | 1.71E−03 | 1.28E−03 | 1.65E−03 | 2.01E−03 |
| Total DtN iterations | 60 | 60 | 56 | 56 |

**Table 3** Computed and estimated errors for the interface values at Levels 1 and 2, as well as total DtN iterations, with TOL = 1E−08

| | Level 1 | Level 2 | | | |
|---|---|---|---|---|---|
| | – | Subdomain 1 | Subdomain 2 | Subdomain 3 | Subdomain 4 |
| Estimated error | 9.64E−09 | 9.72E−09 | 9.42E−09 | 9.50E−09 | 9.36E−09 |
| Max. error (Dirichlet) | 1.12E−09 | 4.23E−09 | 1.50E−09 | 1.65E−09 | 1.67E−09 |
| Max. error (Neumann) | 2.60E−09 | 5.31E−08 | 9.86E−08 | 4.71E−08 | 2.16E−08 |
| Total DtN iterations | 204 | 197 | 194 | 183 | 185 |

**Table 4** Computed and estimated errors for the interface values at level 3. as well as total DtN iterations, with TOL = 1E−08

| | Level 3 | | | |
|---|---|---|---|---|
| | Subdomain 1 | Subdomain 2 | Subdomain 3 | Subdomain 4 |
| Estimated error | 9.18E−09 | 9.70E−09 | 1.00E−08 | 9.55E−09 |
| Max. error (Dirichlet) | 4.98E−09 | 3.44E−09 | 1.97E−09 | 2.89E−09 |
| Max. error (Neumann) | 7.57E−08 | 6.99E−08 | 3.37E−08 | 3.75E−08 |
| Total DtN iterations | 200 | 200 | 193 | 195 |
| | Level 3 | | | |
| | Subdomain 5 | Subdomain 6 | Subdomain 7 | Subdomain 8 |
| Estimated error | 9.88E−09 | 9.83E−09 | 9.73E−09 | 9.32E−09 |
| Max. error (Dirichlet) | 1.10E−09 | 1.71E−09 | 1.02E−09 | 1.11E−09 |
| Max. error (Neumann) | 9.29E−08 | 1.08E−07 | 6.42E−08 | 5.57E−08 |
| Total DtN iterations | 198 | 195 | 189 | 193 |
| | Level 3 | | | |
| | Subdomain 9 | Subdomain 10 | Subdomain 11 | Subdomain 12 |
| Estimated error | 9.48E−09 | 9.19E−09 | 9.75E−09 | 9.81E−09 |
| Max. error (Dirichlet) | 1.20E−09 | 7.91E-10 | 9.31E-10 | 1.14E−09 |
| Max. error (Neumann) | 4.15E−08 | 4.74E−08 | 3.94E−08 | 2.17E−08 |
| Total DtN iterations | 187 | 184 | 178 | 181 |
| | Level 3 | | | |
| | Subdomain 13 | Subdomain 14 | Subdomain 15 | Subdomain 16 |
| Estimated error | 9.15E−09 | 9.80E−09 | 9.45E−09 | 9.64E−09 |
| Max. error (Dirichlet) | 1.87E−09 | 1.27E−09 | 1.19E−09 | 1.14E−09 |
| Max. error (Neumann) | 3.60E−08 | 3.05E−08 | 1.79E−08 | 2.13E−08 |
| Total DtN iterations | 188 | 187 | 182 | 182 |

Since the optimal choice for the $\theta$ value of the iterative *DtN* procedure is not known, we have considered a parameter testing numerical experiment. In Fig. 23, we present the maximum error across the interfaces $\Gamma$ for several values of the parameter $\theta$. It appears that the best numerical results in terms of accuracy are obtained in the interval $[0.01,\ \ 0.105]$.

It should be noted that the problem of decomposing a nonconvex polygon into a minimum number of convex polygons has been solved and “optimal decomposition algorithms” have been provided [9]. The above numerical results indicate that the proposed domain decomposition approach in conjunction with the method of Fokas can be useful for solving BVPs formulated on nonconvex domains.

An alternative approach using virtual sides has been introduced in [11] for solving BVPs on nonconvex domains via the Fokas method. This approach relies on introducing a virtual side between the appropriate corners of the domain in order

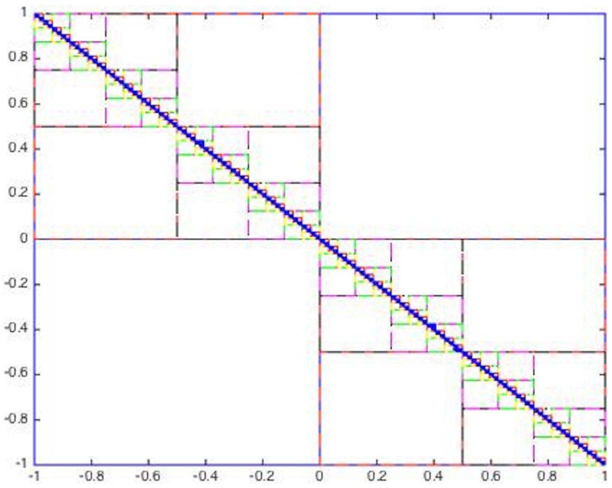


**Fig. 12** Adaptively generated subdomains for seven hierarchical levels, with TOL = 1E−03 and TSH = 1E−04

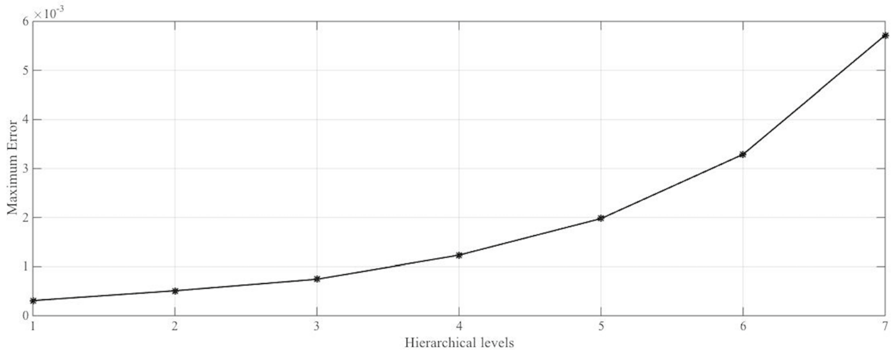


**Fig. 13** Maximum computed error for each of the seven hierarchical levels, with TOL = 1E−03 and TSH = 1E−04

to derive a number of convex polygons. The difference between this technique and ours lies in the solution procedure; in [11], one linear system is assembled including the matched Cauchy data across the virtual side in the solution vector. If $n_{vs}$ virtual sides are used, the coefficient matrix is then augmented by $2n_{vs}$ block columns of size depending on the order $N$ of the selected basis functions. The authors have reported a two-subdomain numerical experiment obtaining high accuracy in approximately 0.06 s; however, their method needs to be tested in a complicated nonconvex domain with multiple re-entrant corners in order to make a proper comparison to our approach. It is certain that including a large number of subdomains, hence a large number of virtual sides, the technique in [11] will result in a relatively large coefficient matrix. Using our technique, it is possible to handle *any* nonconvex domain by solving small linear systems of low condition number, cf. Table 5,

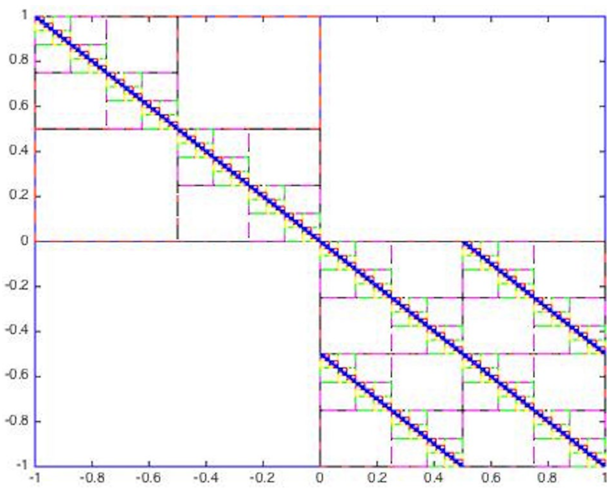


**Fig. 14** Adaptively generated subdomains for seven hierarchical levels, with TOL = 1E−04 and TSH = 5E−06

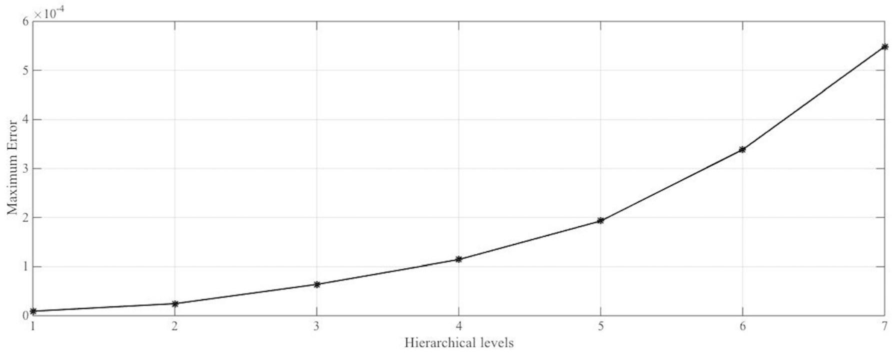


**Fig. 15** Maximum computed error for each of the 7 hierarchical levels, with TOL = 1E−04 and TSH = 5E−06

obtaining high-order accuracy as well, cf. Fig. 22. Hence, in cases of large nonconvex domains needing to be decomposed into a large number of subdomains, our method is expected to be faster since it is inherently parallel and each subdomain can be solved independently.

## 7.5 Parallel performance of UTDtN

In this subsection, the performance of the parallel implementation of the *UTDtN* algorithm is presented. We assess the proposed parallel algorithm by solving the Laplace equation on a square domain using the solution given by Eq. (31) as reference. Furthermore, to highlight the possible advantages, it is compared to a parallel

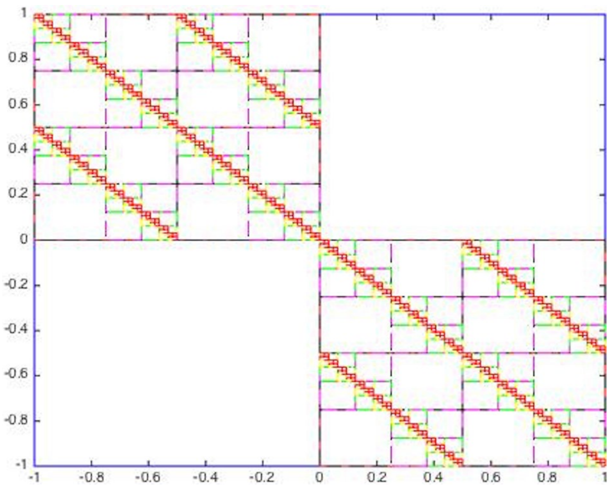


**Fig. 16** Adaptively generated subdomains for six hierarchical levels with TOL = 1E−05 and adaptive threshold; TSH = 1E−07 at levels 1–2, and TSH = 1E−06 at levels 3–6

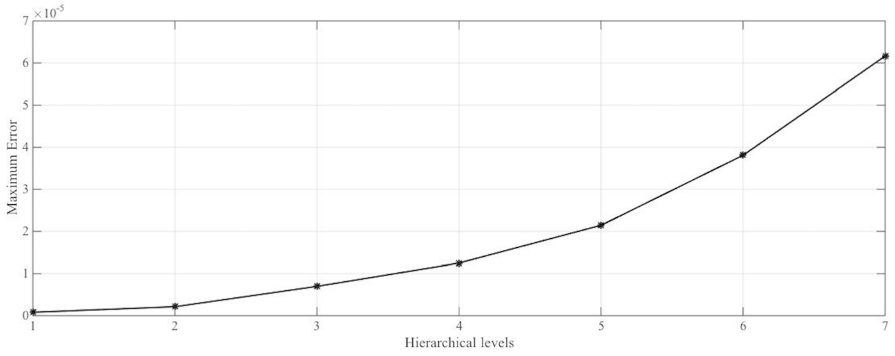


**Fig. 17** Maximum computed error for each of the six hierarchical levels with TOL = 1E−05 and adaptive threshold; TSH = 1E−07 at levels 1–2, and TSH = 1E−06 at levels 3–6

implementation of a standard Schur complement approach, based on the finite element method.

In Table 6, the total time (in seconds) to compute both the Dirichlet and the Neumann interface values, for various numbers of subdomains and cores, is given with *TOL* = 1E−03 and *TOL* = 1E−08. In addition, in Figs. 24 and 25, the speedups for various numbers of subdomains and cores are presented with TOL = 1E−03 and TOL = 1E−08, respectively. The numerical results depicted in Table 6 and Figs. 24 and 25 were obtained on a machine with an Intel Xeon CPU E5-2420v2, 2.2 GHz with 64 GB RAM.

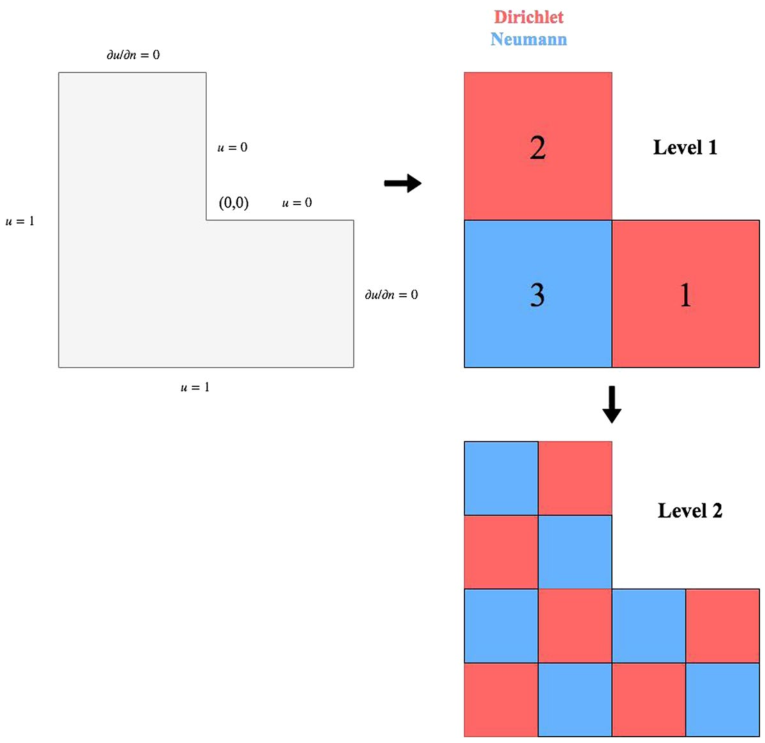


**Fig. 18** L-shaped domain with boundary conditions and a two-level decomposition

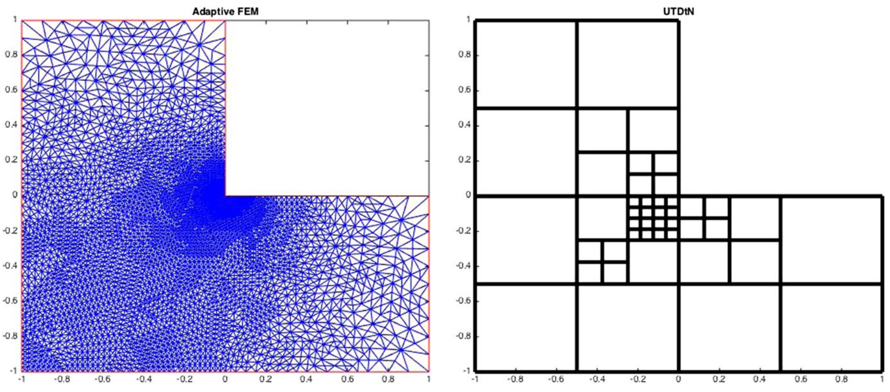


**Fig. 19** Left: The computational mesh using an adaptive FEM method. Right: the generated subdomains for the UTDtN technique

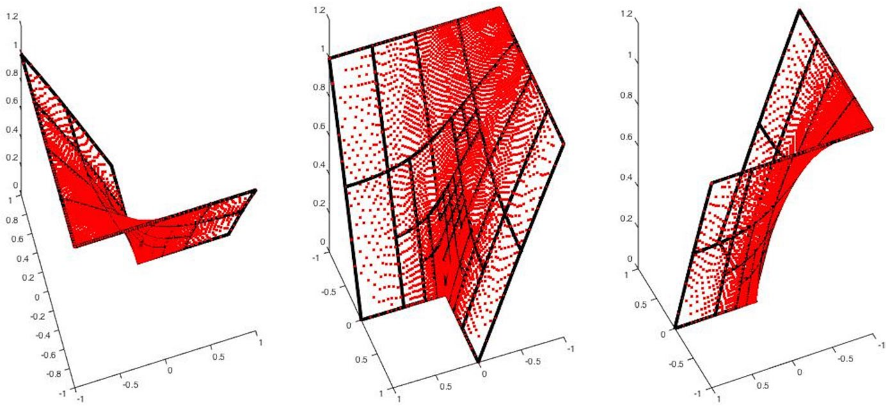


**Fig. 20** The numerical solution over the L-shaped domain using an adaptive FEM method (red) and the UTDtN algorithm (black), shown from multiple angles (color figure online)

Additionally, we have performed numerical experiments on the ARIS supercomputer (GRNET). Each compute node consists of 2× Ivy Bridge Intel Xeon E5-2680v2 (10 cores each) and 64 GB RAM. In Table 7, the total time (in seconds) to compute both the Dirichlet and the Neumann interface values, for 256 and 1024 subdomains, and several numbers of cores, is given with TOL = 1E−03 and TOL = 1E−08. Moreover, in Figs. 26 and 27, the speedups for 256 and 1024 subdomains, and various numbers of cores are presented with TOL = 1E−03 and TOL = 1E−08, respectively.

In Table 8, a comparison between the parallel *UTDtN* and a parallel finite element, Schur complement scheme (DDFEM), is presented. The total time to compute all interface values as well as the maximum computed errors on Intel Xeon CPU E5-2420v2, 2.2 GHz, 64 GB RAM, with six cores, is given.

The *DDFEM* scheme produces subdomains in the same manner as the *UTDtN* algorithm does. The main difference is the fact that *DDFEM* proceeds in the discretization of the interior of the subdomains, using a triangulation. Additionally, the initial mesh can be iteratively refined according to a parameter *MR*, which denotes the number of mesh refinements. It should be mentioned that the refinement procedure

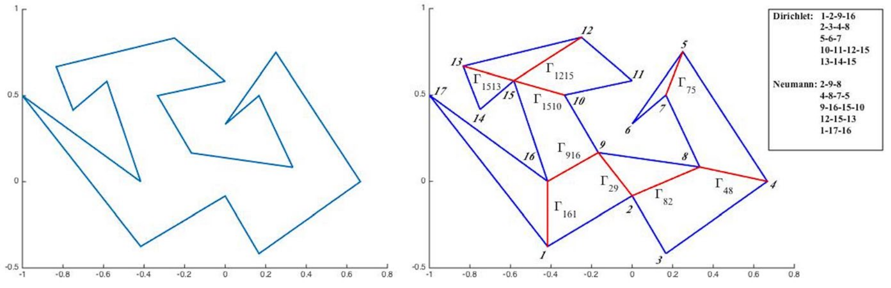


**Fig. 21** Nonconvex domain with multiple re-entrant corners and a ten-subdomain decomposition

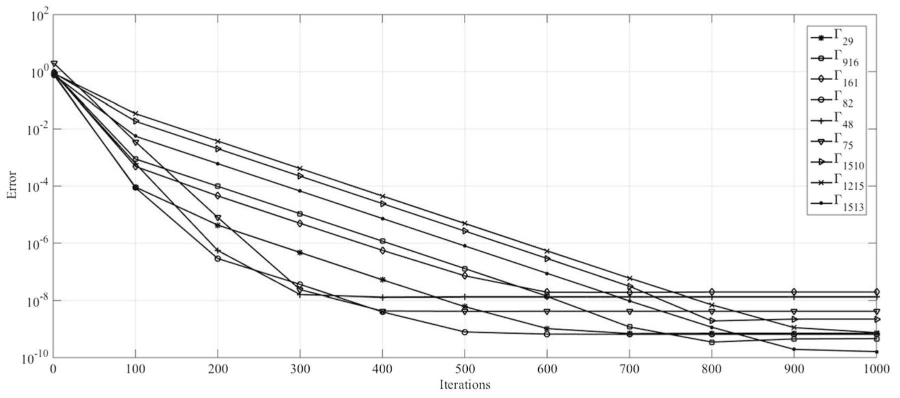


**Fig. 22** Numerical convergence of the interface values $\Gamma$ after 1000 iterations

**Table 5** Condition number of the linear system for the entire domain versus the decomposed subdomains

| Domain | Condition number |
|---|---|
| Entire | 2.84E+18 |
| 1-2-9-16 | 95.30 |
| 2-9-8 | 3.66E+02 |
| 2-3-4-8 | 57.10 |
| 4-8-7-5 | 2.55E+02 |
| 5-6-7 | 4.16E+04 |
| 9-16-15-10 | 5.11E+02 |
| 10-11-12-15 | 63.48 |
| 12-15-13 | 1.21E+02 |
| 13-14-15 | 3.58E+04 |
| 1-17-16 | 98.02 |

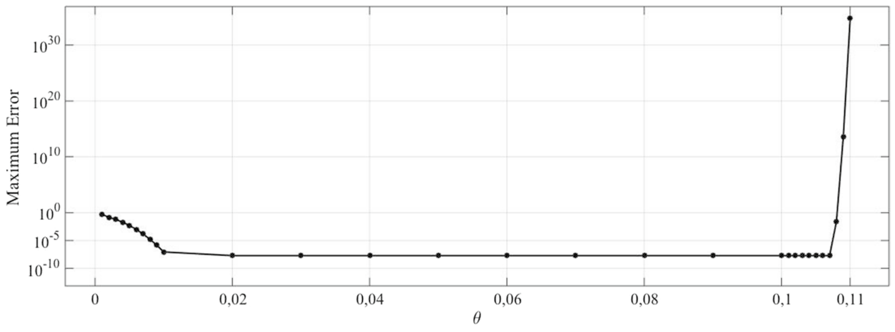


**Fig. 23** Numerical convergence of the interface values $\Gamma$ after 1000 iterations

involves the division of each triangle into four new triangles of the same shape. The *DDFEM* scheme relies on a Schur complement approach; however, the corresponding Schur complement matrix $S$ does not need to be explicitly formed; the associated linear system is solved using an iterative method where only matrix-by-vector operations are required. In this case, we have considered the preconditioned conjugate gradient (PCG) method [39], using 500 maximum iterations and 1E-10 tolerance. It should be noted that in order to be properly compared, only the interface values have been computed. In "Appendix", the details of the DDFEM algorithm are given.

The numerical results indicate that our proposed scheme is much better in terms of speed and accuracy, especially in the case of a large number of subdomains. The main disadvantage of the finite element method lies in the generation of the computational mesh, a task that is computationally expensive when higher accuracy is needed.

As it has been previously stated, our proposed technique does not require mesh generation; the solution is only computed on the subdomains' interfaces by solving smaller one-dimensional problems. In the case that higher resolution is required, the computational domain is further hierarchically decomposed, and the solution is obtained at the newly introduced interfaces.

## 8 Conclusion

A class of novel techniques for the solution of linear elliptic PDEs, based on the unified transform, has been presented. The proposed techniques rely on an iterative Dirichlet-to-Neumann approach, where the Fokas global relations constitute the essential component of the methodology. By reformulating an iterative Dirichlet-to-Neumann algorithm in terms of the approximate global relation, we have designed a domain decomposition-type class of techniques that have the following properties: inherent parallelism, high accuracy, and meshless subdomains resulting in reduced dimension approximations. In addition, we have presented an error estimation technique based on the global relation that has been initially used for the termination of the associated iterative procedures and secondly for the design of a numerical scheme that adaptively generates meshless subdomains. We have presented various sets of numerical results indicating the applicability of the proposed techniques. Furthermore, we have made brief comparisons to a Schur complement finite element method as well as an adaptive finite element method, highlighting the possible advantages. Future work will be directed towards the parallelization of the proposed scheme on distributed memory parallel systems and further study and application of the proposed techniques to a variety of practical problems arising in engineering and sciences as well as ill-posed problems.

**Table 6** The total time to compute all Dirichlet and Neumann interface values for various numbers of subdomains and cores

| Subdomains | Number of cores | | | | | |
|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 |
| | Time (s) | | | | | |
| TOL = 1E−03 | | | | | | |
| 4 | 0.72 | – | – | – | – | – |
| 16 | 2.75 | 1.89 | 1.80 | 1.21 | 1.17 | 1.01 |
| 64 | 5.50 | 3.24 | 2.72 | 2.17 | 2.00 | 1.82 |
| 256 | 20.44 | 10.87 | 7.81 | 6.05 | 5.56 | 4.92 |
| 1024 | 79.05 | 41.11 | 28.26 | 21.83 | 17.94 | 15.51 |
| TOL = 1E−08 | | | | | | |
| 4 | 5.39 | – | – | – | – | – |
| 16 | 23.64 | 15.26 | 14.61 | 10.76 | 10.73 | 10.61 |
| 64 | 96.88 | 51.70 | 41.04 | 30.07 | 28.82 | 25.16 |
| 256 | 371.33 | 192.75 | 138.18 | 101.30 | 90.67 | 79.98 |
| 1024 | 1546.53 | 788.48 | 539.29 | 413.49 | 341.79 | 299.47 |

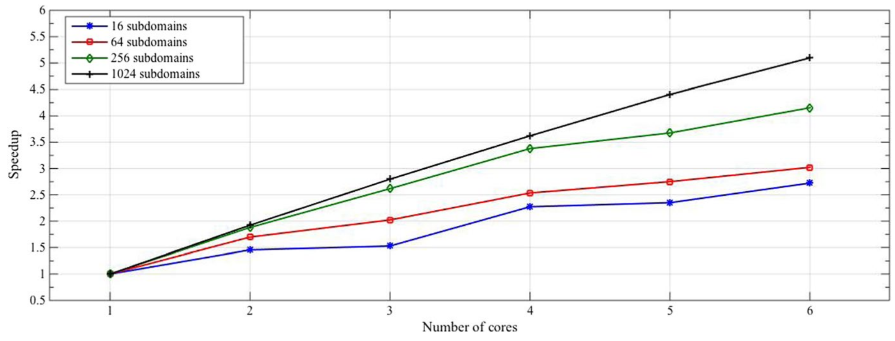


**Fig. 24** Speedups for various numbers of subdomains and cores, with TOL = 1E−03

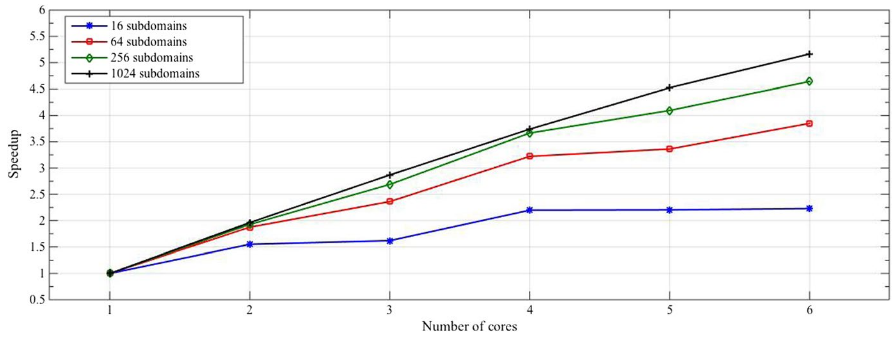


**Fig. 25** Speedups for various numbers of subdomains and cores, with TOL = 1E−08

**Table 7** The total time to compute all Dirichlet and Neumann interface values for various numbers of subdomains and cores, on ARIS HPC system

| Subdomains | Number of cores | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 |
| | Time (s) | | | | |
| TOL = 1E−03 | | | | | |
| 256 | 21.60 | 12.52 | 7.94 | 6.01 | 5.28 |
| 1024 | 74.88 | 40.07 | 22.14 | 14.37 | 10.00 |
| TOL = 1E−08 | | | | | |
| 256 | 346.32 | 179.31 | 96.12 | 55.85 | 37.33 |
| 1024 | 1405.50 | 710.50 | 364.02 | 192.74 | 108.44 |

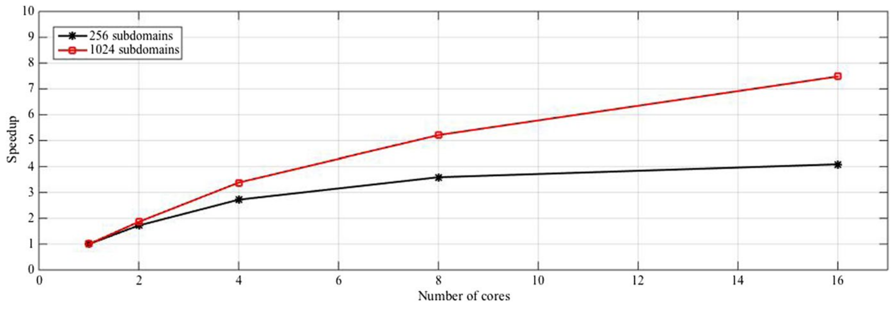


**Fig. 26** Speedups for various numbers of subdomains and cores, with TOL = 1E−03, on ARIS HPC system

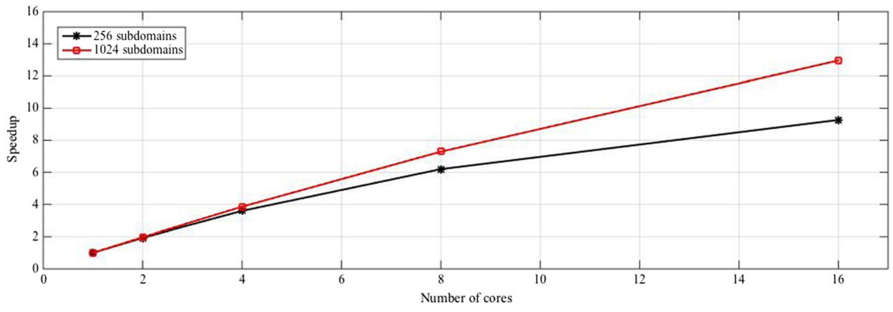


**Fig. 27** Speedups for various numbers of subdomains and cores, with TOL = 1E−08, on ARIS HPC system

**Table 8** Total time and corresponding computed error for approximating the interface values, using the *DDFEM* and *UTDtN_main* algorithms, for several values of the parameters *MR* and *TOL*, and for several numbers of subdomains, *sdoms*

| sdoms | DDFEM (MR = 1) | | UTDtN_main (TOL = 1E−01) | | DDFEM (MR = 2) | | UTDtN_main (TOL = 1E−02) | |
|---|---|---|---|---|---|---|---|---|
| | Time (s) | Error | Time (s) | Error | Time (s) | Error | Time (s) | Error |
| 4 | 0.07 | 7.70E−03 | 0.45 | 1.93E−01 | 0.17 | 2.10E−03 | 0.49 | 7.20E−03 |
| 16 | 0.14 | 1.74E−02 | 0.96 | 1.58E−01 | 0.32 | 4.90E−03 | 0.84 | 4.40E−03 |
| 64 | 0.72 | 6.10E−03 | 1.37 | 1.79E−01 | 2.79 | 1.60E−03 | 1.36 | 7.60E−03 |
| 256 | 15.11 | 1.90E−03 | 1.66 | 2.29E−01 | 93.61 | 4.98E−04 | 2.69 | 9.70E−03 |
| sdoms | DDFEM (MR = 3) | | UTDtN_main (TOL = 1E−03) | | DDFEM (MR = 4) | | UTDtN_main (TOL = 1E−04) | |
| | Time (s) | Error | Time (s) | Error | Time (s) | Error | Time (s) | Error |
| 4 | 0.49 | 6.21E−04 | 0.69 | 3.10E−04 | 2.89 | 1.71E−04 | 1.04 | 2.63E−05 |
| 16 | 0.94 | 1.30E−03 | 1.10 | 2.89E−04 | 4.51 | 3.51E−04 | 1.60 | 2.05E−05 |
| 64 | 14.69 | 4.12E−04 | 2.00 | 4.61E−04 | 92.02 | 1.05E−04 | 2.80 | 3.54E−05 |
| 256 | 573.01 | 1.29E−04 | 4.55 | 6.53E−04 | 3873.52 | 3.29E−05 | 9.36 | 5.17E−05 |
| sdoms | DDFEM (MR = 5) | | UTDtN_main (TOL = 1E−05) | | DDFEM (MR = 6) | | UTDtN_main (TOL = 1E−06) | |
| | Time (s) | Error | Time (s) | Error | Time (s) | Error | Time (s) | Error |
| 4 | 15.31 | 4.54E−05 | 1.43 | 2.33E−06 | 102.05 | 1.17E−05 | 1.90 | 2.28E−07 |
| 16 | 31.30 | 9.06E−05 | 2.11 | 1.69E−06 | 214.41 | 2.30E−05 | 3.22 | 1.58E−07 |
| 64 | 640.43 | 2.68E−05 | 4.10 | 2.82E−06 | >5000 | ~O(1E−06) | 6.37 | 2.02E−07 |
| 256 | >10,000 | ~O(1E−06) | 11.74 | 2.81E−06 | >10,000 | ~O(1E−06) | 19.57 | 2.99E−07 |

**Acknowledgements** The authors acknowledge the Greek Research and Technology Network (GRNET) for the provision of the National HPC facility ARIS under project PR004033—ScaleSciCompII, and support from EPSRC, UK. The authors are also thankful to Matt Colbrook for useful suggestions.

**Funding** Funding was provided by Engineering and Physical Sciences Research Council for A.S. Fokas.

## Appendix: Finite Element Domain Decomposition (DDFEM) algorithm

Let us consider a linear system Au = f, arising from a finite element discretization of a linear elliptic PDE. For a general decomposition into s subdomains, the linear system has the following structure [39]

$$\begin{bmatrix} K_1 & & & & E_1 \\ & K_2 & & & E_2 \\ & & \ddots & & \vdots \\ & & & K_s & E_s \\ E_1^T & E_2^T & \cdots & E_s^T & C \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_s \\ u_\Gamma \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_s \\ f_\Gamma . \end{bmatrix} \tag{33}$$

where matrix A can also be written as

$$A = \begin{bmatrix} K & E \\ E^T & C \end{bmatrix}. \tag{34}$$

The vectors $\{u_j\}_1^s$ represent the solution at the interior of the s subdomains, and $u_\Gamma$ represents the solution at the interfaces. Using block Gaussian elimination, the interface values are obtained by solving the following reduced system [39]

$$(C - E^T K^{-1} E)u_\Gamma = f_\Gamma - E^T K^{-1} f, \tag{35}$$

where

$$S = C - E^T K^{-1} E, \tag{36}$$

is called the Schur complement matrix [39]. The reduced system can be solved without explicitly assembling the Schur complement matrix S by considering a Krylov subspace iterative method. The matrix-by-vector operations $Su_\Gamma$ are performed as follows [39]:

$$\begin{aligned} &\text{compute}\, x = Eu_\Gamma \\ &\text{solve}|\; Ky = x \\ &\text{compute}\, Cu_\Gamma - E^T y. \end{aligned} \tag{37}$$

**Algorithm 10** *DDFEM*

```
    Choose the number of subdomains, $s$
 2: Choose the number of successive mesh refinements, $MR$
    Generate initial triangular mesh
 4: for $i = 1 : MR$ do
        Refine triangular mesh
 6: end for
    for $j = 1 : s$, in parallel do
 8:     Form $K^j, E^j, C^j, f^j, f_\Gamma^j, R^j$
        Form $r^j = f_\Gamma^j - (E^j)^T (C^j)^{-1} f^j$
10: end for
    Form $rhs = \sum_{j=1}^{s} (R^j)^T r^j$
12: Form $C = \sum_{j=1}^{s} (R^j)^T C^j R^j$
    Solve $Su_\Gamma = rhs$ using a Krylov subspace method
```

In Algorithm 10 the Schur complement, finite element procedure is described. It should be noted that R represents a restriction matrix. Further implementation details can be found in [34].

## References

1. Arabnia HR, Taha TR (1998) A parallel numerical algorithm on a reconfigurable multi-ring network. J Telecommun Syst 10:185–203
2. Ashton ACL (2013) The spectral Dirichlet–Neumann map for Laplace's equation in a convex polygon. SIAM J Math Anal 45(6):3575–3591

3. Babuska I, Guo B (2000) Optimal estimates for lower and upper bounds of approximation errors in the p-version of the finite element method in two dimensions. Numer Math 85(2):219–255
4. Balasubramanian P, Arabnia HR (2014) Computation of error resiliency of Muller C-element. In: Proceedings on International Conference on Computational Science and Computational Intelligence, pp 179–180
5. Bhandarkar SM, Arabnia HR (1995) The REFINE multiprocessor-theoretical properties and algorithms. Parallel Comput 21(11):1783–1806
6. Bjorck A (2015) Numerical methods in matrix computations. Texts in Applied Mathematics. Springer, Berlin
7. Canuto C, Hussaini MY, Quarteroni A, Zang TA (2006) Spectral methods. Springer, Berlin
8. Chan TF, Goovaerts D (1989) Schur complement domain decomposition algorithms for spectral methods. Appl Numer Math 6:53–64
9. Chazelle B, Dobkin D (1985) Optimal convex decompositions. In: Toussaint G (ed) Computational geometry. North-Holland, Amsterdam, pp 63–133
10. Colbrook MJ (2018) Extending the unified transform: curvilinear polygons and variable coefficient PDEs. IMA J Numer Anal. https://doi.org/10.1093/imanum/dry085
11. Colbrook MJ, Flyer N, Fornberg B (2018) On the Fokas method for the solution of elliptic problems in both convex and non-convex polygonal domains. J Comput Phys 374:996–1016
12. Courant R, Hilbert D (1989) Methods of mathematical physics, vol 1. Wiley, Hoboken
13. Davis C-IR, Fornberg B (2014) A spectrally accurate numerical implementation of the Fokas transform method for Helmholtz-type PDEs. Complex Var Elliptic Equ 59(4):564–577
14. Elliotis M, Georgiou G, Xenophontos C (2005) Solving Laplacian problems with boundary singularities: a comparison of a singular boundary integral method with the p/hp version of the finite element method. Appl Math Comput 169:485–499
15. Fernandez A, Baleanu D, Fokas AS (2018) Solving PDEs of fractional order using the unified transform method. Appl Math Comput 339:738–749
16. Fokas AS (1997) A unified transform method for solving linear and certain nonlinear PDEs. Proc R Soc Lond Ser A 453:1411–1443
17. Fokas AS (2001) Two-dimensional linear PDEs in a convex polygon. Proc R Soc Lond Ser A 457:371–393
18. Fokas AS (2002) A new transform method for evolution PDEs. IMA J Appl Math 67:559–590
19. Fokas AS (2008) A unified approach to boundary value problems. SIAM, Philadelphia
20. Fornberg B, Flyer N (2011) A numerical implementation of Fokas boundary integral approach: Laplace's equation on a polygonal domain. Proc R Soc A 467:2083–3003
21. Franceschini A, Paludetto Magri V, Ferronato M, Janna C (2018) A robust multilevel approximate inverse preconditioner for symmetric positive definite matrices. SIAM J Matrix Anal Appl 39:123–147
22. Fulton S, Fokas AS, Xenophontos C (2004) An analytical method for linear elliptic PDEs and its numerical implementation. J Comput Appl Math 167:465–483
23. Grylonakis E-NG, Filelis-Papadopoulos CK, Gravvanis GA (2015) A note on solving the generalized Dirichlet to Neumann map on irregular polygons using Generic Factored Approximate Sparse Inverses. CMES Comput Model Eng Sci 109(6):505–517
24. Grylonakis E-NG, Filelis-Papadopoulos CK, Gravvanis GA (2017) A hybrid method for solving inhomogeneous elliptic PDEs based on Fokas method. Comput Methods Appl Math. https://doi.org/10.1515/cmam-2017-0053
25. Grylonakis E-NG, Filelis-Papadopoulos CK, Gravvanis GA (2018) A class of unified transform techniques for solving linear elliptic PDEs in convex polygons. Appl Numer Math 129:159–180
26. Grylonakis E-NG, Filelis-Papadopoulos CK, Gravvanis GA, Fokas AS (2019) An iterative spatial-stepping numerical method for linear elliptic PDEs using the Unified Transform. J Comput Appl Math 352:194–209
27. Grylonakis E-NG, Filelis-Papadopoulos CK, Gravvanis GA, Fokas AS (2019) An adaptive complex collocation method for solving linear elliptic PDEs in regular convex polygons based on the unified transform. Numer Math Theory Methods Appl 12(2):348–369
28. Hashemzadeh P, Fokas AS, Smitheman SA (2015) A numerical technique for linear elliptic partial differential equations in polygonal domains. Proc Math Phys Eng Sci 471:20140747. https://doi.org/10.1098/rspa.2014.0747
29. Janna C, Castelletto N, Ferronato M (2015) The effect of graph partitioning techniques on parallel Block FSAI preconditioning: a computational study. Numer Algorithms 68(4):813–836

30. Jayashree HV, Thapliyal H, Arabnia HR, Agrawal VK (2016) Ancilla-input and Garbage-output Optimized Design of a Reversible Quantum Integer Multiplier. J Supercomput 72(4):1477–1493
31. Jiri K, Rozloznik M, Tuma M (2017) An adaptive multilevel factorized sparse approximate inverse preconditioning. Adv Eng Softw 113:19–24
32. Kyziropoulos PE, Filelis-Papadopoulos CK, Gravvanis GA (2018) A class of symmetric factored approximate inverses and hybrid two-level solver. Int J Comput Methods 15(2):1850050
33. Makaratzis AT, Filelis-Papadopoulos CK, Gravvanis GA (2016) Parallel multilevel recursive approximate inverse techniques for solving general sparse linear systems. J Supercomput 72(6):2259–2282
34. Mathew T (2008) Domain decomposition methods for the numerical solution of partial differential equations. Springer, Berlin
35. Moutafis BE, Filelis-Papadopoulos CK, Gravvanis GA (2017) Parallel multi-projection preconditioned methods based on semi-aggregation techniques. J Comput Sci 22:45–54
36. Moutafis BE, Filelis-Papadopoulos CK, Gravvanis GA (2018) Parallel Schur complement techniques based on multiprojection methods. SIAM J Sci Comput 40(4):634–654
37. Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical recipes. The art of scientific computing, 3rd edn. Cambridge University Press, Cambridge
38. Quarteroni A (2014) Numerical models for differential problems (*MS&A*), 2nd edn. Springer, Berlin
39. Saad Y (2003) Iterative methods for sparse linear systems, 2nd edn. Society for Industrial and Applied Mathematics, Philadelphia
40. Sauter SA, Schwab C (2010) Boundary element methods. Springer, Berlin
41. Sifalakis AG, Fokas AS, Fulton S, Saridakis YG (2008) The generalized Dirichlet–Neumann map for linear elliptic PDEs and its numerical implementation. J Comput Appl Math 219(1):9–34
42. Toselli A, Widlund O (2005) Domain decomposition methods—algorithms and theory. Springer, Berlin
43. Valafar H, Arabnia HR, Williams G (2004) Distributed global optimization and its development on the multiring network. Int J Neural Parallel Sci Comput 12(4):465–490
44. Wang H, Xiang S (2012) On the convergence rates of Legendre approximation. Math Comput 81:861–877
45. Xi Y, Li R, Saad Y (2016) An algebraic multilevel preconditioner with low-rank corrections for sparse symmetric matrices. SIAM J Matrix Anal Appl 37(1):235–259
46. Zhu JZ, Zienkiewicz OC (1988) Adaptive techniques in the finite element method. Commun Appl Numer Methods 4:197–204
47. Zhu Y, Sameh AH (2017) PSPIKE+: a family of parallel hybrid sparse linear system solvers. J Comput Appl Math 311:682–703
48. Zienkiewicz OC, Taylor OL, Zhu JZ (2013) The finite element method: its basis and fundamentals, 7th edn. Butterworth-Heinemann, Oxford

## Affiliations

**E. N. G. Grylonakis[1] · G. A. Gravvanis[1] · C. K. Filelis-Papadopoulos[1] · A. S. Fokas[2]**

E. N. G. Grylonakis
egrylona@ee.duth.gr

C. K. Filelis-Papadopoulos
cpapad@ee.duth.gr

A. S. Fokas
tf227@cam.ac.uk

[1] Department of Electrical and Computer Engineering, School of Engineering, Democritus University of Thrace, University Campus, Kimmeria, 67100 Xanthi, Greece

[2] Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge CB3 0WA, UK