



Article

Test-Driven Development of a Substructuring Technique for the Analysis of Electromagnetic Finite Periodic Structures

Ignacio Martínez-Fernández, Adrian Amor-Martin *  and Luis E. Garcia-Castillo 

Signal Theory and Communications Department, University Carlos III of Madrid, Leganes, 28911 Madrid, Spain; igmarfer@gmail.com (I.M.-F.); legcasti@ing.uc3m.es (L.E.G.-C.)

* Correspondence: aamor@ing.uc3m.es

Featured Application: The methodology presented in this work can be applied to develop reliable scientific codes, and specifically codes for the finite element analysis of finite periodic structures.

Abstract: In this paper, we follow the Test-Driven Development (TDD) paradigm in the development of an in-house code to allow for the finite element analysis of finite periodic type electromagnetic structures (e.g., antenna arrays, metamaterials, and several relevant electromagnetic problems). We use unit and integration tests, system tests (using the Method of Manufactured Solutions—MMS), and application tests (smoke, performance, and validation tests) to increase the reliability of the code and to shorten its development cycle. We apply substructuring techniques based on the definition of a unit cell to benefit from the repeatability of the problem and speed up the computations. Specifically, we propose an approach to model the problem using only one type of Schur complement which has advantages concerning other substructuring techniques.

Keywords: test-driven development; verification and validation; finite element method; substructuring techniques; finite periodic structures; computational electromagnetics



Citation: Martínez-Fernández, I.; Amor-Martin, A.; Garcia-Castillo, L.E. Test-Driven Development of a Substructuring Technique for the Analysis of Electromagnetic Finite Periodic Structures. *Appl. Sci.* **2021**, *11*, 11619. <https://doi.org/10.3390/app112411619>

Academic Editors: Daniele Funaro and Ernesto Limiti

Received: 30 October 2021

Accepted: 3 December 2021

Published: 7 December 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The Finite Element Method (FEM) has been extensively used for Computational Electromagnetics [1] due to its robustness and versatility. However, it requires the use of volumetric meshes, which limit the size of the problems to be analyzed.

To leverage this problem in FEM, some techniques have been proposed in the last years: Domain Decomposition Methods (DDM) [2–4] have increased the size of the problems that can be solved at the expense of a more difficult formulation and the use of iterative solvers; Model Order Reduction (MOR) techniques [5,6] reduce the number of simulations required for a given bandwidth using polynomial interpolation techniques; multigrid methods [7,8] are used as preconditioners for iterative solvers; and finally, substructuring methods that are purely algebraic approaches based on the Schur complement [9,10].

Our goal in this work is the analysis of finite periodic structures where a unit cell is repeated on space. Periodic structures are very common in microwave engineering, as frequency selective surfaces, phased array antennas, metamaterials, or electromagnetic bandgap structures [11–14]. An approximated analysis of a finite periodic structure can be performed considering the structure to be infinitely periodic. Thus, the analysis is restricted to the analysis of the unit cell with periodic boundary conditions on the sides along with the directions on which there is periodicity, e.g., see [15,16]. Although the infinitely periodic approximation is useful to model large structures, it neglects the border effects due to the finite size of the structures in practice. In this context, substructuring methods are useful as they provide an algebraically exact solution for the analysis of the finite periodic structure while alleviating the computational effort by working with a finite number of different Schur complements (being that number independent of the size of the structure). Specifically, we propose an approach in which there is only one type of Schur

complement involved in the computations with the corresponding advantages in terms of computational resources.

One of our contributions is to show the step-by-step development process we have followed in the introduction of this approach in an existing general-purpose FEM in-house code called HOFEM [17]. Specifically, we follow the Test-Driven Development (TDD) paradigm extensively used in other research fields as scientific computing and software engineering. We use unit and integration tests, system tests (using the Method of Manufactured Solutions—MMS [18,19]), and application tests (smoke, performance, and validation tests). As a result, we managed to increase the reliability of the code and shorten its development cycle compared to other software development plans. This process can be also applied in different codes, e.g., [20–23].

The paper is structured as follows: In Section 2, we define the formulation used in the FEM code, the Schur complement-based substructuring methods that we use, and the types of validation procedures we have considered to introduce the technique; in Section 3, we show the unit and integration tests that we have considered, we use the MMS to show the accuracy of the implementation and to conduct error convergence tests, we define smoke tests and use a pyramidal horn antenna array as a real-world application, and we observe the performance of the introduced technique. Finally, in Section 4, we draw the conclusions we have obtained in this work.

2. Methods

2.1. Formulation

We define a partial differential operator \mathcal{L} as

$$\mathcal{L} = \nabla \times (\bar{\mu}_r^{-1} \nabla \times) - k_0^2 \bar{\epsilon}_r, \quad (1)$$

where $\bar{\mu}_r$ is the relative magnetic permeability tensor, $\bar{\epsilon}_r$ is the relative electric permittivity tensor, and k_0 is the wavenumber in vacuum.

If we apply \mathcal{L} to the electric field E , we obtain

$$\mathcal{L}E = f, \quad (2)$$

where f is the excitation of our problem, i.e.,

$$f = -jk_0\eta_0 J - \nabla \times (\bar{\mu}_r^{-1} M), \quad (3)$$

being η_0 the vacuum impedance, whereas J and M denote the electric and magnetic currents, respectively. We use bold characters for vector magnitudes.

Thus, we use as the mathematical model for the finite element analysis the well-known double-curl wave equation in a given domain Ω [1],

$$\nabla \times (\bar{\mu}_r^{-1} \nabla \times E) - k_0^2 \bar{\epsilon}_r E = -jk_0\eta_0 J - \nabla \times (\bar{\mu}_r^{-1} M). \quad (4)$$

To have a unique solution, we use as boundary conditions

$$\hat{n} \times E = 0, \text{ on } \Gamma_D, \quad (5a)$$

$$\hat{n} \times (\bar{\mu}_r^{-1} \nabla \times E) = 0, \text{ on } \Gamma_N, \quad (5b)$$

$$\hat{n} \times (\bar{\mu}_r^{-1} \nabla \times E) + \gamma \hat{n} \times (\hat{n} \times E) = \Phi, \text{ on } \Gamma_C, \quad (5c)$$

where \hat{n} is the outward normal vector, γ and Φ are the propagation constant and the excitation term that depend on the boundary condition (e.g., $\gamma = \omega\sqrt{\mu\epsilon}$, $\Phi = 0$ for absorbing boundary conditions), and Γ_D , Γ_N , and Γ_C stand for Dirichlet, Neumann, and Cauchy boundary conditions, respectively.

The vector electric field E belongs to the curl-conforming space [24],

$$H(\text{curl}) = \{N \in (L^2(\Omega))^3 : \nabla \times N \in (L^2(\Omega))^3\}, \tag{6}$$

where $L^2(\Omega)$ is the space of square-integrable functions and, specifically, here we use the space

$$H_0(\text{curl}, \Omega) = \{N \in H(\text{curl}, \Omega) : \hat{n} \times N = 0 \in \Gamma_D\} \tag{7}$$

with homogeneous Dirichlet boundary conditions on Γ_D .

Now, we define the residual $\mathbf{R} = \mathcal{L}E - \mathbf{f}$ that we test with the weight function F to obtain the variational formulation, i.e.,

$$\langle F, \mathcal{L}E - \mathbf{f} \rangle = 0, \tag{8}$$

so we need to find $E \in H_0(\text{curl}, \Omega)$ such that

$$\begin{aligned} & \int_{\Omega} (\nabla \times F) \cdot (\bar{\mu}^{-1} \nabla \times E) d\Omega + \gamma \int_{\Gamma_c} (\hat{n} \times F) \cdot (\hat{n} \times E) d\Gamma_c \\ & - k_0^2 \int_{\Omega} F \cdot (\bar{\epsilon} E) d\Omega - jk_0 \eta_0 \int_{\Omega} F \cdot J d\Omega \\ & - \int_{\Omega} F \cdot \nabla \times (\bar{\mu}_r^{-1} M) d\Omega + \int_{\Gamma_c} F \cdot \Phi d\Gamma_c = 0, \end{aligned} \tag{9}$$

for all $F \in H_0(\text{curl}, \Omega)$.

This formulation is discretized with a collection of finite elements (e.g., tetrahedral mesh) that we assemble for each element e obtaining

$$\sum_{\forall e} \langle F^e, \mathcal{L}E^e - \mathbf{f}^e \rangle = 0, \tag{10}$$

where we have used the superscript e to denote magnitudes relative to each element. We use basis functions $N \in H_0(\text{curl}, \Omega)$ to approximate E^e in each element, i.e., $E^e = \sum_{j=1}^{N_e} g_j N_j^e$, with N_e as the number of unknowns in each element, whereas we use the same set of basis functions N for F (Galerking approach) yielding

$$\sum_{\forall e} \langle N_i^e, \mathcal{L} \sum_{j=1}^{N_e} g_j N_j^e - \mathbf{f}^e \rangle = 0 \rightarrow \sum_{\forall e} \sum_{j=1}^{N_e} g_j \langle N_i^e, \mathcal{L} N_j^e - \mathbf{f}^e \rangle = 0, \tag{11}$$

thanks to the linearity of (1). Thus, we can obtain a global system of equations where an array \mathbf{g} (comprised of all g_j) is solved through

$$L\mathbf{g} = \mathbf{f}. \tag{12}$$

2.2. Schur-Based Substructuring Methods

In this work, we address electromagnetic structures of the kind shown in Figure 1, where we have a two-dimensional finite grid of the so-called here unit cells. We want to have a full-wave, accurate solution of the problem taking into account the periodicity of the problem; thus, we use substructuring methods (i.e., the computation of the Schur complement) to solve the original system of equations using only a subset of the unknowns. In this way, we obtain a small, condensed matrix from the huge, sparse original matrix that, otherwise, we would need to solve if we would not apply substructuring techniques. Thus, let us divide the original set of unknowns from (12) into two disjoint sets, \mathbf{g}_1 and \mathbf{g}_2 , leading to

$$\begin{aligned} L_{1,1} \cdot \mathbf{g}_1 + L_{1,2} \cdot \mathbf{g}_2 &= \mathbf{f}_1, \\ L_{2,1} \cdot \mathbf{g}_1 + L_{2,2} \cdot \mathbf{g}_2 &= \mathbf{f}_2, \end{aligned} \tag{13}$$

where we apply a partial LU decomposition, here noted as $L = MU$,

$$\begin{pmatrix} L_{1,1} & L_{1,2} \\ L_{2,1} & L_{2,2} \end{pmatrix} = \begin{pmatrix} M_{1,1} & 0 \\ M_{2,1} & I \end{pmatrix} \begin{pmatrix} U_{1,1} & U_{1,2} \\ 0 & S \end{pmatrix}, \tag{14}$$

that maps the original matrix to a condensed matrix as in Figure 2. We have used I to denote the identity matrix, whereas the S corresponds to the Schur complement of the original matrix L , i.e., $S = L_{2,2} - L_{2,1}L_{1,1}^{-1}L_{1,2}$. We can observe that a sparse matrix is decomposed into the product of two block-triangular dense matrices: we work with the right lower dense block of the right matrix, S , which is expected to be much smaller than the original matrix. Specifically, we choose as g_2 the unknowns on the interfaces between unit cells. This is the best option to profit from the periodicity of the problem because it provides the minimal number of unknowns to solve the original problem.

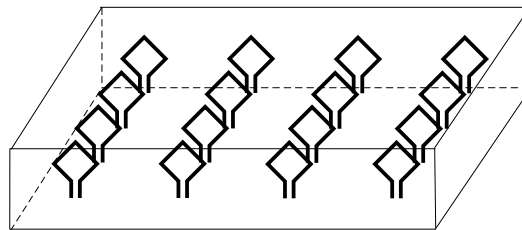


Figure 1. Regular periodic structures addressed in this work.

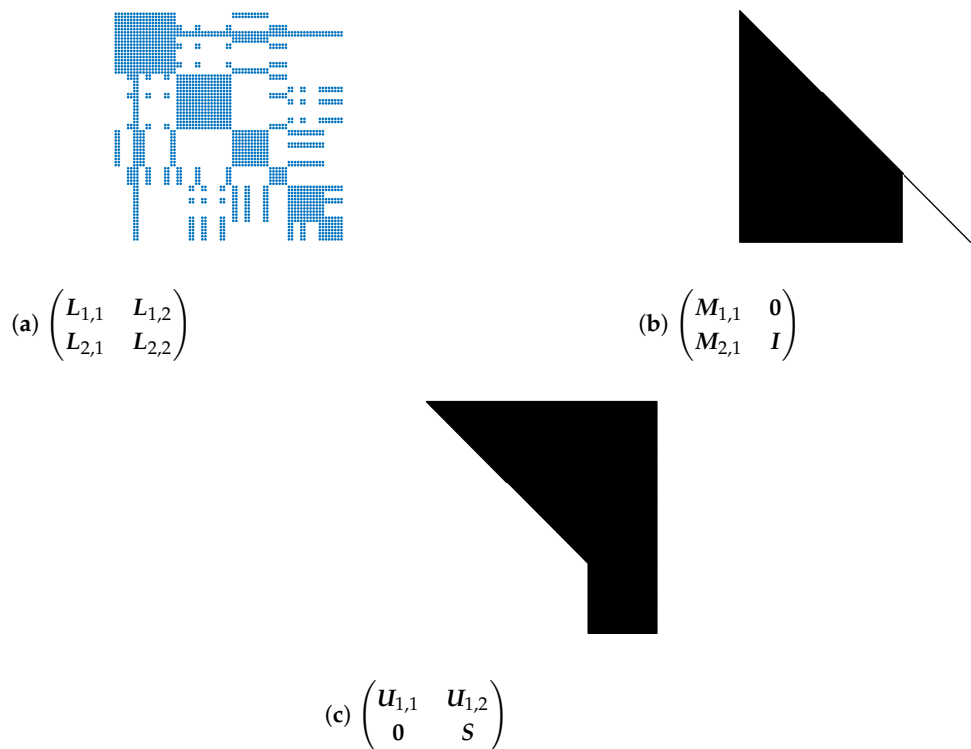


Figure 2. Schur factorization of the left matrix, (a) into lower, (b) and upper, (c) block-triangular matrices.

The original system may be solved in three steps:

1. Solve $M_{1,1}y_1 = f_1$ and obtain y_2 from $M_{2,1}y_1 + Iy_2 = f_2$.
2. Solve the condensed problem $Sg_2 = y_2$.
3. Expand the solution g_2 to the whole system with $U_{1,1}g_1 + U_{1,2}g_2 = y_2$.

From a physical point of view, we construct the surface condensed problem; then, we solve this small problem and, after that, we propagate the solution from the surface to the interior of each unit cell.

The main advantage of this method is that we can extend the dimension of the problems that can be solved in a given machine since the original big matrix is not even constructed. Additionally, the data FEM structures of the whole periodic problem are not built (e.g., those supporting the mesh of the whole problems) since we can operate only with the unit cell.

As seen in (10), we use a discretization of finite elements to solve a problem with FEM. This leads to having a direct correspondence between the local (to each element) unknown and the global (in the final system of equations) unknown that identifies univocally that unknown in the whole problem. However, the introduction of a unit cell leads us to introduce an intermediate correspondence between the cell unknown and the global unknown. Finally, we extract the surface unknowns from the global set of unknowns.

We use three algorithms as intermediate steps to achieve the final solution:

1. Full mesh approach.
2. Nine Schur complement approach.
3. One Schur complement approach.

Moreover, we follow the TDD paradigm: we write the tests shown in Section 2.3, and we code each step until we obtain the right solution for each algorithm. Since this is an algebraic technique, the solution must be numerically equal for all the approaches.

2.2.1. Full Mesh Approach

Here, the original problem is generated from the unit cell: i.e., we copy and translate the unit cell along the two dimensions, thus obtaining the matrix that would be generated with that mesh (but not storing the mesh itself). This step allows us to have a reference solution to compare with the following two approaches. Thus, we solve the problem without substructuring techniques, i.e., with (12).

To generate the new mesh points from the unit cell, we compute offset vectors (in all the periodic directions of our problem) and use a master–slave hierarchy to store the correspondence between unknowns. For illustration purposes, let us consider a two-dimensional problem. We establish a master–slave hierarchy for the four edge corners in the unit cell, taking the southwest corner as the master edge. Thus, we identify the master edge and we assign the unknowns (in the same order as in the master edge) in the slave edges. A similar procedure is performed with the four faces that constitute the interface with the other unit cells. There, we take as master faces the south and west faces, and we assign the unknowns on the slave faces (north and east, respectively) in the same order as on the master face. Finally, we assign the remaining unknowns in the unit cell.

Note that this hierarchy is not needed for this approach (that could work only with the offset vectors), but it is very useful for parallelization (we can assign the unknowns concurrently for the different cells since we have the same relative order in each cell) and for the substructuring techniques that need the correspondence between local, cell, and global unknowns. Therefore, it is convenient to introduce it in this approach to generate the replication of the mesh along the two dimensions of the grid.

Once we have obtained the numbering of the unknowns, we compute for each element the different integral terms (including boundary conditions, if applicable), we assemble in a finite element sense all the contributions and, finally, we solve the resulting matrix using a direct method.

2.2.2. Nine Schur Complement Approach

In this approach, we introduce the substructuring technique in Equation (14). The main difference of this approach is that we do not need to generate the whole matrix to solve the problem, saving memory. We can work only with the unit cell using the hierarchy presented in Section 2.2.1, computing nine Schur complements for a two-dimensional periodicity, as seen in Figure 3, where we can see that the Schur complement for, e.g., cells 5 and 6, is the same. Note that for the one-dimensional periodicity case, we would

have only three different Schur complements. Thus, the procedure that we follow for each kind of unit cell is:

1. Generate the numbering of the unknowns following the hierarchy in Section 2.2.1.
2. Compute the integral terms (including terms related to boundary conditions) from (9) that are applicable to each kind of unit cell. Note that the volume terms (first, third, fourth, and fifth terms in (9)) are common to each cell (and can be computed once for all the cells), whereas the surface terms (second and sixth terms in (9)) depend on the kind of the cell: e.g., the cell number 1 needs to apply on the south face an absorbing boundary condition whereas the cell number 5 does not need to compute any surface integral term.
3. Extract the subset of unknowns that are involved at the interface with other unit cells.
4. Condense the original problem, i.e., compute the Schur complement.

Once we have the nine different kinds of Schur complements, we assign them to the different cells taking into account its relative position as in Figure 3. Then, we apply the three-step solver detailed in Section 2.2 solving the surface problem and then propagating this surface solution to the interior of each cell. Note that this approach also allows to introduce additional Schur complements when we have defects on the periodic grid or variations in some cells.

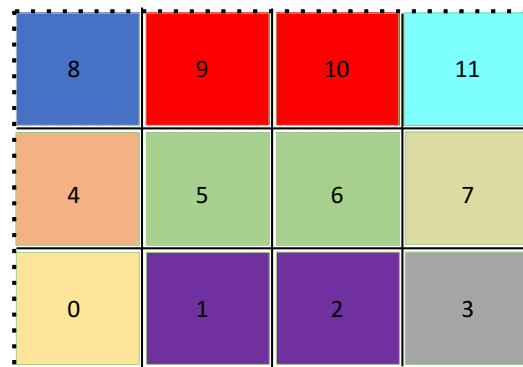


Figure 3. Illustration of the approach followed in Section 2.2.2. Each color means a different Schur complement, showing 9 different complements.

2.2.3. One Schur Complement Approach

We can improve the former approach by imposing the boundary conditions after the condensation of the problem since we only need to compute (and store) one Schur complement. The procedure that we follow in this approach is:

1. Generate the unknowns following the same hierarchy as in Section 2.2.1.
2. Compute the integral terms related to Ω in each element from (9), i.e., without including any boundary term.
3. Extract the subset of unknowns that belongs to any boundary (not only the interfaces between cells but also those of the boundaries of the original problem domain Ω).
4. Condense the problem by computing the Schur complement of this generic unit cell.

Once we have the matrix S , we assign the different boundary conditions for each cell. Let us follow the three-step procedure detailed in Section 2.2, being g_2 all the unknowns extracted in the third step of the former procedure. We have to distinguish between the three boundary conditions considered in Section 2.1:

- For Dirichlet boundary conditions in (5a), we directly remove the unknowns.
- For Neumann boundary conditions in (5b), we do not need to add anything since it is a natural boundary condition.
- For Cauchy boundary conditions in (5c), we can use the fact that the Schur complement is computed as $S = L_{2,2} - L_{2,1}L_{1,1}^{-1}L_{1,2}$. Since all the unknowns related to the Cauchy boundary condition belongs to g_2 , the only block that changes in the original matrix L

is $L_{2,2}$. If we call C to this change, we would obtain that the block $L_{2,2}^C$ (with Cauchy boundary conditions assigned) is $L_{2,2}^C = L_{2,2} + C$, so the difference when adding the Cauchy boundary condition to the Schur matrix is that same contribution, i.e., $S^C = S + C$. Analogously, on the right-hand side, the only term that changes is f_2 from (13), so we compute the sixth term in (9) and obtain the reduced right-hand side y_2 .

Thus, we only need to condense the problem once, and then, we particularize this condensed problem to each unit cell. If we take the same example as Section 2.2.2, we would obtain the same S for all the domains, which would be valid for cells 5 and 6, whereas for cell 1 we would compute the terms related to the absorbing boundary condition on the south face and add it to that generic S .

This approach improves the procedure explained in Section 2.2.2 since we store only one Schur complement and we add the boundary conditions on the reduced set of unknowns, which is computationally cheap. In addition, we can tune the phase of each excitation in every unit cell without needing to compute a different Schur complement, as would be the case in Section 2.2.2. Here, we only need to include the unknowns associated to the waveport to the subset of unknowns g_2 . This is extremely useful when having an array of antennas since we can steer the main beam of radiation.

2.3. Validation Procedures

In this work, we have followed the TDD paradigm [25–28]: we write the tests for small increments of functionality; then, we develop the code to pass the tests; and finally, we use refactoring to increase the performance and readability of the code. In our case, the targets are the different Schur-based approaches presented in Section 2.2.

We further subdivide the tests according to four levels, as referred in Figure 4:

- Unit tests, Figure 5, for assessing single routines in the code.
- Integration tests, to ease the modular development of the code.
- System tests, to verify the compliance of the code with its expected features. Specifically in our case, we will use the Method of Manufactured Solutions (MMS) to test the formulation introduced in Section 2.1.
- Application tests, to define a benchmark of tests to ensure that the new developments do not break established features of the code (including performance features) and as validation of the code.

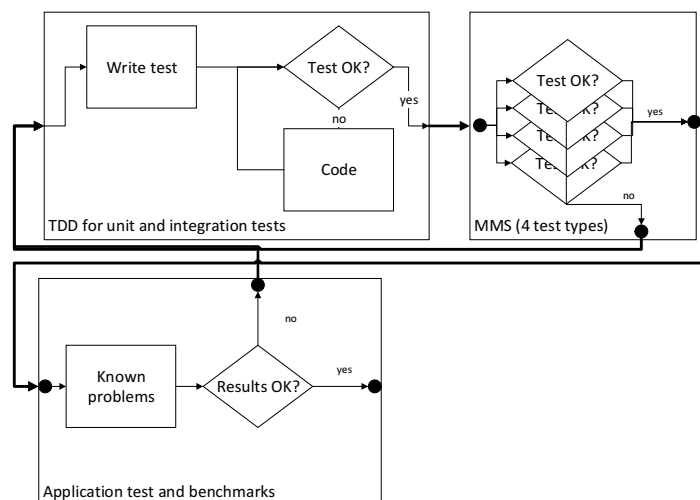


Figure 4. Development cycle diagram.

2.3.1. Unit and Integration Tests

The features that we need for a unit test are:

- Specificity, to check a defined section of the code.

- Orthogonality, to go through the same code only once.
- Coverage, to test all the features in the developed code.
- Independence, to have the same start and end state of the program.
- Automation, to be executed automatically.

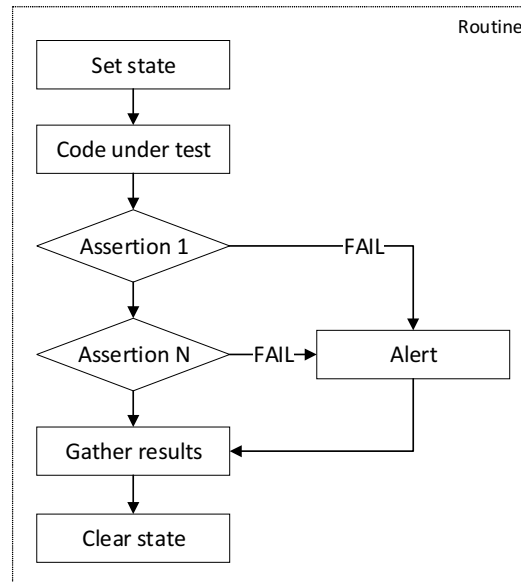


Figure 5. Unit test for a piece of code.

The unit tests translate a functional requirement to the code, adding sanity checks to small portions of the code. The integration tests consist of calling different routines of the code with a given input (e.g., mesh structure, FEM matrix...) asserting the expected output of the routine. For both, we have used pFUnit [29,30], which allows us to add automatic tests in MPI and OpenMP (used extensively in the code [31]).

We show an example of the application of unit and integration tests with pFUnit in Code 1. Note that the snippet contains Fortran free-format code, which is the language used to develop our code. There, we import the module of pFUnit with `USE pfunit_mod`, and we test the `finiteGrid` module, which is the part of the code devoted to managing the grid. We construct a mockup grid of 2×2 , and we have designed the cell numbering in a way that the second domain should be in the second row and the first column. We test those numbers with `@assertEqual` and, if true, we destruct the mockup grid previously created.

```

1  MODULE test_finiteGrid
2  IMPLICIT NONE
3  CONTAINS
4  @test
5  SUBROUTINE test_get2DLocation
6    USE finiteGrid
7    USE pfunit_mod
8    TYPE(GridStruct):: testGrid
9    INTEGER :: row, col, cell, m, n
10   m=2; n=2; cell=2
11   CALL gridConstructor(m,n,testGrid)
12   CALL get2Dlocation(testGrid,cell,row,col)
13   @assertEqual(2,row)
14   @assertEqual(1,col)
15   CALL gridDestructor(testGrid)
16  END SUBROUTINE test_get2DLocation
  
```

Code 1: Snippet for unit and integration tests in the code.

2.3.2. System Tests: MMS

The MMS is a mathematical technique that is typically used for code verification and can be understood as a system test since it checks the formulation [19], which is critical for the features of the application. The MMS consists of choosing (manufacturing) a solution of some (partial differential), calculating the excitation terms by plugging the manufactured solution in the equation, and running the code with that manufactured excitation. Then, the numerical solution provided by the code can be compared with the prescribed manufactured solution.

More specifically, we can summarize the whole procedure in four steps:

1. Pick the differential operator \mathcal{L} and a function as manufactured solution, E^{mms} .
2. Generate $f^{mms} = \mathcal{L}E^{mms}$.
3. Solve $\mathcal{L}E^{sol} = f^{mms}$.
4. Compare the numerical results E^{sol} with E^{mms} .

In our case, we use the following norms

$$e_{field} = \frac{\|E^{sol} - E^{mms}\|_2}{\|E^{mms}\|_2}, \tag{15a}$$

$$e_{rot} = \frac{\|\nabla \times E^{sol} - \nabla \times E^{mms}\|_2}{\|\nabla \times E^{mms}\|_2}. \tag{15b}$$

One of the advantages of MMS is the possibility of testing independently the different terms present in the formulation of the problem, (9). Thus, we can divide the application of MMS into four incremental tests:

1. $\mathcal{L} = I$ and Dirichlet boundary conditions. We test here the Dirichlet procedure and the third and fourth terms from (9), i.e.,
Find $F \in H_0(\text{curl}, \Omega)$ such as:

$$\int_{\Omega} F \cdot E d\Omega = \int_{\Omega} F \cdot E^{mms} d\Omega \tag{16}$$

2. \mathcal{L} from (1) and Dirichlet boundary conditions. We add now the first term from (9), as in:
Find $F \in H_0(\text{curl}, \Omega)$ such as:

$$\int_{\Omega} (\nabla \times F) \cdot (\nabla \times E) d\Omega - k_0^2 \int_{\Omega} F \cdot E d\Omega = \int_{\Omega} F \cdot [\nabla \times (\nabla \times E^{mms}) - k_0^2 E^{mms}] d\Omega \tag{17}$$

3. The same \mathcal{L} but now using Neumann instead of Dirichlet boundary conditions, adding the last term from (9). This means:
Find $F \in H_0(\text{curl}, \Omega)$ such as:

$$\int_{\Omega} (\nabla \times F) \cdot (\nabla \times E) d\Omega - k_0^2 \int_{\Omega} F \cdot E d\Omega = \int_{\Omega} F \cdot [\nabla \times (\nabla \times E^{mms}) - k_0^2 E^{mms}] d\Omega - \int_{\Gamma_N} F \cdot (\hat{n} \times (\nabla \times E^{mms})) d\Gamma_N \tag{18}$$

4. We change Neumann to Cauchy boundary condition, where the last term is the same, and we add the second term in (9), testing all the elementary terms from the variational formulation:

$$\begin{aligned}
& \int_{\Omega} (\nabla \times \mathbf{F}) \cdot (\nabla \times \mathbf{E}) d\Omega - k_0^2 \int_{\Omega} \mathbf{F} \cdot \mathbf{E} d\Omega \\
& + \int_{\Gamma_C} (\hat{\mathbf{n}} \times \mathbf{F}) \cdot (\hat{\mathbf{n}} \times \mathbf{E}) = \int_{\Omega} \mathbf{F} \cdot \left[\nabla \times (\nabla \times \mathbf{E}^{\text{mms}}) - k_0^2 \mathbf{E}^{\text{mms}} \right] d\Omega \quad (19) \\
& - \int_{\Gamma_C} \mathbf{F} \cdot (\hat{\mathbf{n}} \times (\nabla \times \mathbf{E}^{\text{mms}})) d\Gamma_C
\end{aligned}$$

Note that this way of application of MMS follows the TDD paradigm since we design tests for small, incremental terms in the formulation.

2.4. Application Tests

We have three different application tests:

1. Smoke tests: basic, fast (and not necessarily thorough) tests for sanity checks. For example, we are refactoring the code and we assess that the code is working as expected.
2. Performance tests: to assess the speedup of the code (including parallel scalability) and obtain computation metrics.
3. Validation tests: in addition to the verification made with the system tests, we use real (e.g., radiation) problems to validate the code, i.e., to be sure it returns a good approximation of the physical problem under analysis.

Without TDD paradigms, we could directly try to use application tests; however, the detection of errors in the code would be more difficult and cumbersome.

3. Results

We employ GiD [32] for meshing purposes and also to show the results of the simulations (field plots). The programming language used for the code is modern Fortran (using features up to Fortran 2003 and following an object-oriented paradigm), whereas we use isotropic materials without loss of generality.

We have structured the results in three sets: unit and integration tests, system tests (using MMS), and application tests.

3.1. Unit and Integration Tests

Snippet 1 is a minimal example where we assess one particular feature of the code. pFUnit works as a preprocessor with .pf files (as the Code 1, written in Fortran free format) that can use a Fortran library (in Code 1, finiteGrid) and will provide an executable whose outcome is displayed in Code 2. We have a successful test if the unit test is displayed as a dot (.), providing an F otherwise. For the latter case, we obtain the information of the position of the error.

```

1 ..F.
2 Time: 0.001 seconds
3
4 Failure in:
5 test_FiniteGrid2_suite.test_get2dlocation
6 Location:
7 [test_FiniteGrid2.pf:49]
8 expected 2 but found: 1; difference: |1|.
9
10 Tests run: 3, Failures: 1, Errors: 0

```

Code 2: Output in case of failure.

3.2. System Tests

We perform here two kinds of tests: accuracy tests (to debug possible losses of accuracy within the code) and error convergence tests. As a unit cell, we use a vacuum cube with dimensions $[0, 1] \times [0, 1] \times [0, 1]$ m. Regarding finite elements, we have used tetrahedra with second-order basis functions from [33]. We employ GiD [32] to generate the mesh and show the results of the simulations and employ Fortran as the programming language (following an object-oriented paradigm). The mesh that we use here is composed of 8324 tetrahedra for the unit cell, which produces a problem of 53,124 unknowns for the unit cell. We have considered a grid of 3×3 unit cells.

For the accuracy test, we want to detect if all the polynomials that belong to the space of functions we have used to numerically approximate the solution yields a numerically zero output. We test each of the monomials (three-dimensional polynomial vectors) whose linear combination yields the basis functions N_i of (11). Specifically, the monomials involved in these basis functions are the following (please refer to [33]):

$$\left\{ \begin{aligned} & \left[\begin{matrix} 1 \\ 0 \\ 0 \end{matrix} \right], \left[\begin{matrix} x \\ 0 \\ 0 \end{matrix} \right], \left[\begin{matrix} y \\ 0 \\ 0 \end{matrix} \right], \left[\begin{matrix} z \\ 0 \\ 0 \end{matrix} \right], \left[\begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \right], \left[\begin{matrix} 0 \\ x \\ 0 \end{matrix} \right], \left[\begin{matrix} 0 \\ y \\ 0 \end{matrix} \right], \left[\begin{matrix} 0 \\ z \\ 0 \end{matrix} \right], \left[\begin{matrix} 0 \\ 0 \\ 1 \end{matrix} \right], \left[\begin{matrix} 0 \\ 0 \\ x \end{matrix} \right], \left[\begin{matrix} 0 \\ 0 \\ y \end{matrix} \right], \left[\begin{matrix} 0 \\ 0 \\ z \end{matrix} \right], \dots \\ & \dots, \left[\begin{matrix} y^2 \\ -xy \\ 0 \end{matrix} \right], \left[\begin{matrix} 0 \\ -yz \\ y^2 \end{matrix} \right], \left[\begin{matrix} -xy \\ x^2 \\ 0 \end{matrix} \right], \left[\begin{matrix} -xz \\ 0 \\ x^2 \end{matrix} \right], \left[\begin{matrix} z^2 \\ 0 \\ -xz \end{matrix} \right], \left[\begin{matrix} 0 \\ z^2 \\ -yz \end{matrix} \right], \left[\begin{matrix} yz \\ -xz \\ 0 \end{matrix} \right], \left[\begin{matrix} 0 \\ xz \\ -xy \end{matrix} \right] \end{aligned} \right\} \quad (20)$$

We show in Table 1 the value of e_{field} from (15a) for all the monomials within the space of basis functions for Test 4 from Section 2.3.2. However, we have followed the same TDD approach suggested in Section 2.3.2: i.e., we have obtained good results with Test 1, then we use Test 2 until we obtain good results, and so on. For the sake of brevity, we do not show the value for all the polynomials but rather the maximum value in Table 2. Moreover, for brevity reasons, we only show the values of e_{field} , although similar results are obtained with e_{rot} . We obtain numerically equivalent values for the three algorithms since we follow a mathematical approach, and we should obtain the same solution for all of them. Furthermore, we use a double-precision accuracy in our code so the minimum resolution is 10^{-16} . The three approaches yield the same numerical solution, whereas the differences are due to the different operations applied to obtain the final solution. The accuracy of the results obtained (in the order of 10^{-12}) is due to the numerical noise, and it sets the numerical resolution for this problem.

Second, we perform a test on error convergence, specifically using a plane wave as the solution. We use a plane wave because it is an electromagnetically relevant example of a smooth function for which theoretical results concerning the convergence of the error exist so we can compare with it. Specifically, for smooth functions and the second-order finite element basis functions of [33], the theory predicts the error to decrease as $O(h^2)$ [34], i.e., the slope in a log-log plot of the error versus the discretization size h should be ideally equal to 2. The expression of the electric field of a linearly polarized (in $\hat{\theta} + \hat{\phi}$) plane wave incident from (θ, ϕ) direction is:

$$\mathbf{E} = E_0 \left\{ \begin{aligned} & \cos(\theta) \cos(\phi) - \sin(\phi) \\ & \cos(\theta) \cos(\phi) + \cos(\theta) \\ & - \sin(\theta) \end{aligned} \right\} e^{-jk_0 \mathbf{k}_p \cdot \mathbf{r}} \quad (21)$$

where \mathbf{r} is the position vector, $\mathbf{k}_p = -\sin(\theta) \cos(\phi) \hat{x} - \sin(\theta) \sin(\phi) \hat{y} - \cos(\theta) \hat{z}$, with \hat{x} , \hat{y} , \hat{z} the unit vectors in the Cartesian axis. The working frequency is set to 50 MHz. The results shown in the document correspond to $(\theta, \phi) = (\frac{\pi}{2}, \frac{\pi}{4})$. Similar results, i.e., the same slope of the error, are obtained with different angles of incidence.

In Table 3, we show the relation between the unknowns, the number of elements in the mesh, the maximum discretization size, and the value for e_{field} in two setups of the grid, i.e., 1×1 and 1×2 . As expected, the error is the same for both cases, as we should

expect from Table 1. Figure 6 shows the evolution of the error in log–log scale. The slope obtained is 1.9582 and 1.9141 for e_{field} and e_{rot} , respectively, with both slopes being very close to the predicted result of 2.

Table 1. Value of e_{field} for all the monomials within the space of second-order tetrahedra basis functions.

Monomials	Full Mesh	9 Compl.	1 Compl.
[1, 0, 0]	5.24×10^{-12}	5.32×10^{-12}	4.64×10^{-12}
[0, 1, 0]	4.74×10^{-12}	4.52×10^{-12}	4.56×10^{-12}
[0, 0, 1]	5.34×10^{-12}	4.91×10^{-12}	5.01×10^{-12}
[x, 0, 0]	5.33×10^{-12}	5.28×10^{-12}	4.90×10^{-12}
[y, 0, 0]	5.27×10^{-12}	5.37×10^{-12}	5.14×10^{-12}
[z, 0, 0]	5.30×10^{-12}	4.58×10^{-12}	4.72×10^{-12}
[0, x, 0]	5.24×10^{-12}	5.22×10^{-12}	4.87×10^{-12}
[0, y, 0]	5.13×10^{-12}	5.03×10^{-12}	4.53×10^{-12}
[0, z, 0]	4.63×10^{-12}	5.05×10^{-12}	5.26×10^{-12}
[0, 0, x]	5.45×10^{-12}	4.74×10^{-12}	4.85×10^{-12}
[0, 0, y]	4.96×10^{-12}	4.58×10^{-12}	4.51×10^{-12}
[0, 0, z]	5.29×10^{-12}	5.39×10^{-12}	5.38×10^{-12}
[y ² , -xy, 0]	5.37×10^{-12}	4.75×10^{-12}	5.32×10^{-12}
[0, -yz, y ²]	5.48×10^{-12}	4.79×10^{-12}	4.67×10^{-12}
[xy, -x ² , 0]	4.67×10^{-12}	4.75×10^{-12}	5.41×10^{-12}
[xz, 0, -x ²]	4.81×10^{-12}	4.55×10^{-12}	5.42×10^{-12}
[z ² , 0, -xz]	4.83×10^{-12}	5.15×10^{-12}	4.52×10^{-12}
[0, z ² , -xz]	4.54×10^{-12}	5.24×10^{-12}	4.62×10^{-12}
[yz, -xz, 0]	5.11×10^{-12}	4.68×10^{-12}	5.41×10^{-12}
[0, xz, -xy]	5.31×10^{-12}	4.88×10^{-12}	4.77×10^{-12}

Table 2. Maximum value of e_{field} for all the monomials for the three first incremental tests.

Test	Full Mesh	9 Compl.	1 Compl.
Test 1	1.45×10^{-12}	1.69×10^{-12}	2.47×10^{-12}
Test 2	2.14×10^{-12}	1.94×10^{-12}	1.97×10^{-12}
Test 3	3.48×10^{-12}	3.58×10^{-12}	3.24×10^{-12}

Table 3. e_{field} for the plane wave using five meshes.

Number of Elements	Degrees of Freedom	Discretization Size (m)	Discretization	
			1 × 1	1 × 2
617	4414	0.5	$3.44 \cdot 10^{-3}$	$3.44 \cdot 10^{-3}$
1236	8644	0.4	$2.38 \cdot 10^{-3}$	$2.38 \cdot 10^{-3}$
9773	65,122	0.2	$5.85 \cdot 10^{-4}$	$5.85 \cdot 10^{-4}$
23,452	154,544	0.15	$3.24 \cdot 10^{-4}$	$3.24 \cdot 10^{-4}$
36,060	236,300	0.13	$2.45 \cdot 10^{-4}$	$2.45 \cdot 10^{-4}$

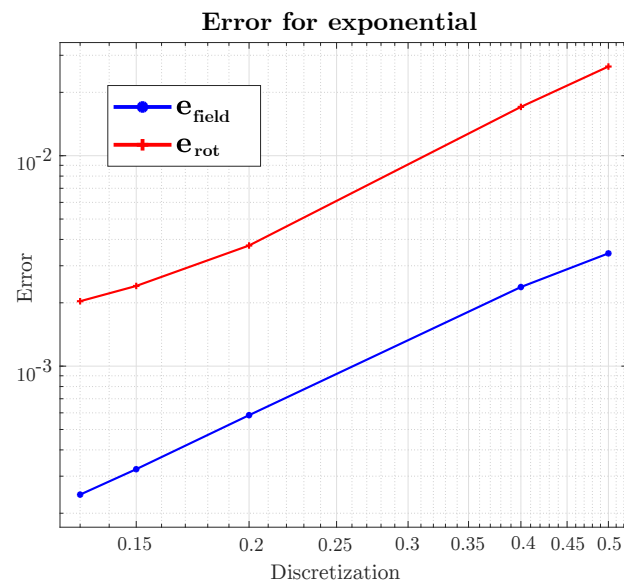


Figure 6. Convergence error.

3.3. Application Tests

3.3.1. Smoke Tests

For smoke tests, we would like to use small meshes that allow us to detect possible errors when we modify the code (e.g., when improving the performance or introducing new features). Thus, we have taken the smallest problem we can mesh with GiD (while maintaining conformal meshes along the vertical and horizontal directions). This setup leads to a mesh with 80 elements and 640 unknowns. As a unit cell, we use a vacuum cube where a plane wave (21) impinges from $(\theta, \phi) = (\frac{\pi}{2}, \frac{\pi}{4})$. We use a 9×9 grid that allows us to test every different interface in Figure 3, and it is large enough to test the index generation algorithm from Section 2.2.3 as it uses all types of possible domains. We show the analytic field E^{ana} , the solution obtained with the code E^{sol} , and the pointwise error between E^{ana} and E^{sol} in Figure 7. However, since we want to have fast tests, we introduce the energy error between the E^{ana} and E^{sol} similarly as in (15a). The energy error we obtain is $9.91 \cdot 10^{-2}$. Thus, this test is very fast to compute and allows us to test the integrity of the code: if we obtain a value other (higher) than $9.91 \cdot 10^{-2}$, we debug the code using, e.g., the system tests or the graphical error representation of the solution in Figure 7.

3.3.2. Performance Tests

It is important to include performance tests in the development process for several reasons. They are useful to detect possible bugs of the computer implementation (and not only the formulation) and inefficiencies. Specifically, tests showing the actual orders of the computational complexity of the code are very useful. In this context, performance tests help to know the limitations of the code and the situations in which it operates optimally. Additionally, performance tests are useful after refactoring the code.

In our case, we show the advantages and the limitations of the method making larger and larger grids. The outcome of the tests we should expect is that the introduction of the method is not worth it for small grids and it becomes more and more advantageous when the grid is larger. Without loss of generality, the problem that we solve is the propagation of a plane wave (expression (21)) whereas now we use 8324 tetrahedra and 53,124 unknowns for the unit cell. We use a personal workstation, equipped with a Linux distribution and with a six-core Intel Core i7-3970 and 32 GB of RAM, to obtain the results in this section.

We increase the size of the problem using a 1D periodic type expansion (i.e., $1 \times M$ configuration, with M as a variable number of cells) and a 2D periodic type expansion ($M \times M$ configuration) to detect possible differences between the two. In Figure 8, we

show the time that we need to solve the problem for the whole code (preprocessing, assembly, solving, and postprocessing) using the substructuring technique of one Schur complement of Section 2.2.3 and without using any substructuring technique (i.e., using the full mesh approach of Section 2.2.1). We can observe that the difference in the computational complexities between the two approaches is larger for the $1 \times M$ case: we obtain $\mathcal{O}(M)$ and $\mathcal{O}(0.67M)$ for the full mesh and one Schur complement approaches, respectively, whereas for the $M \times M$ configuration we have $\mathcal{O}(1.7M)$ and $\mathcal{O}(0.83M)$ for full mesh and the one Schur approaches, respectively. Moreover, the use of the one Schur approach proposed in this paper is more advantageous for a small number of domains (almost from the beginning with $M = 2$).

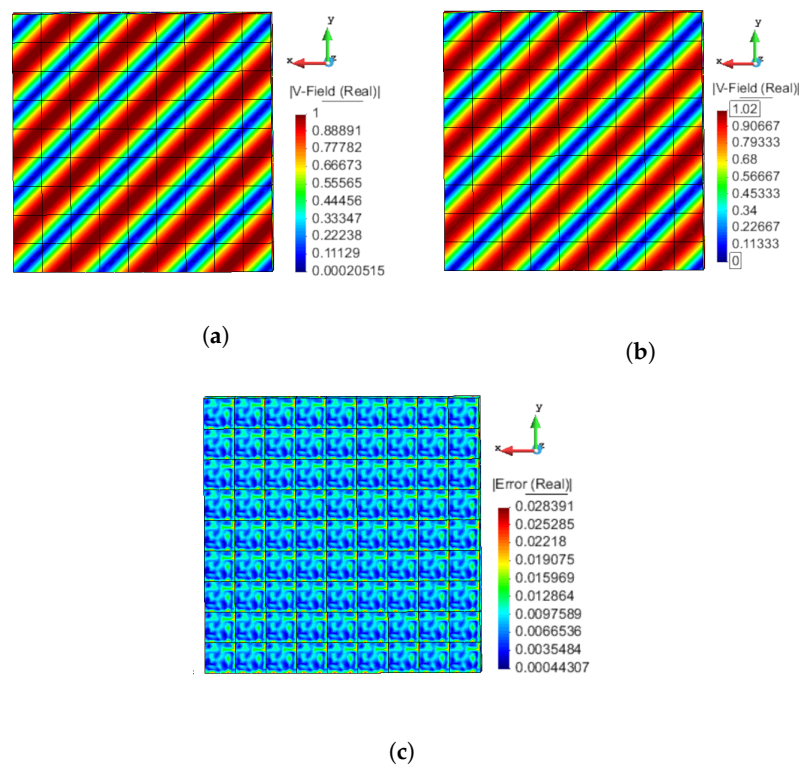


Figure 7. Smoke test with a plane wave, where (a) is the analytic solution E^{ana} , (b) is the solution provided by the code E^{sol} , and (c) is the pointwise error between (a,b).

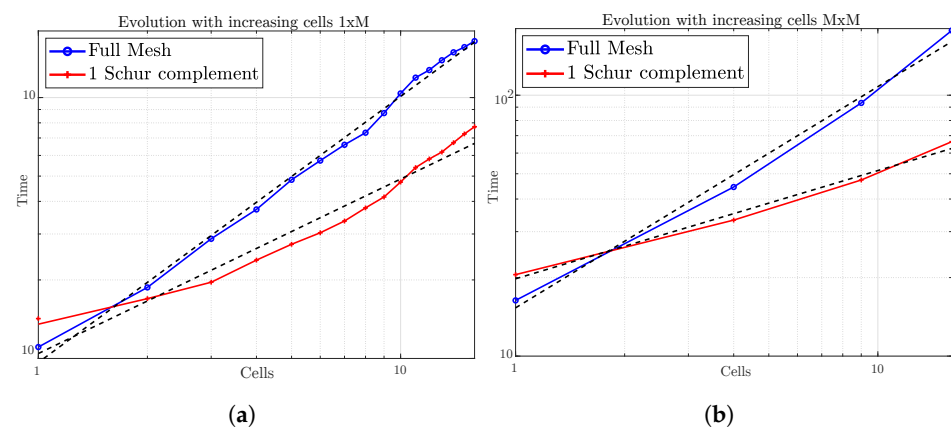


Figure 8. Performance tests. CPU comparison for (a) $1 \times M$ configuration, and (b) $M \times M$ configuration.

3.3.3. Validation Tests

Once the verification process (mathematical and computational) succeeds, it is necessary to validate the code, i.e., assure that the output from the code to a given physical (electromagnetic in this case) corresponds (up to a certain acceptable error level) to the physical solution of the problem. In this context, we show an example of validation with an array of pyramidal horn antennas. The horn antenna, shown in Figure 9a, is designed to work at 10 GHz, with a WR90 waveguide as feed (with dimensions 22.86×10.16 mm) and a straight section of 10.5 mm. To excite the problem, we have used the fundamental mode of the waveguide (transversal electric [35]) through Φ in (9). Note that no impressed currents J or M have been used to excite the problem. The pyramidal section has a height of 61.554 mm, whereas the size of the broader face is 40.131×29.31 mm. We set up two different problems and we qualitatively compare them. On one of the problems, we use the approach in Section 2.2.3 (Figure 9b) to set up a 2×2 grid. The unit cell is obtained by enclosing the horn antenna with an exterior box of dimensions $68.58 \times 50.8 \times 83$ mm. On the boundary of the whole problem, we place an Absorbing Boundary Condition (ABC), i.e., we set the boundary as Γ_C with $\Phi = 0$. We use Perfect Electric Conductor for the structure of the horn, i.e., these faces belong to Γ_D . We focus here on the validation of the substructuring technique and, therefore, we compare the solution of this problem with an equivalent problem set up without using substructuring techniques at all. The equivalent problem consists of four horn antennas placed in a 2×2 configuration and meshing at once the whole problem with the four antennas, where again we use an ABC on the outer boundary of the problem. We use a unit cell and the one Schur approach of Section 2.2.3 in Figure 9b, that we compare with the conventional FEM solution of the equivalent problem in Figure 9c. The differences between both approaches are not significant and they are due to the different meshes that are used (the division between unit cells forces us to have slightly different meshes even though we use the same size for the discretization elements). Moreover, no visible differences are observed between the different cells.

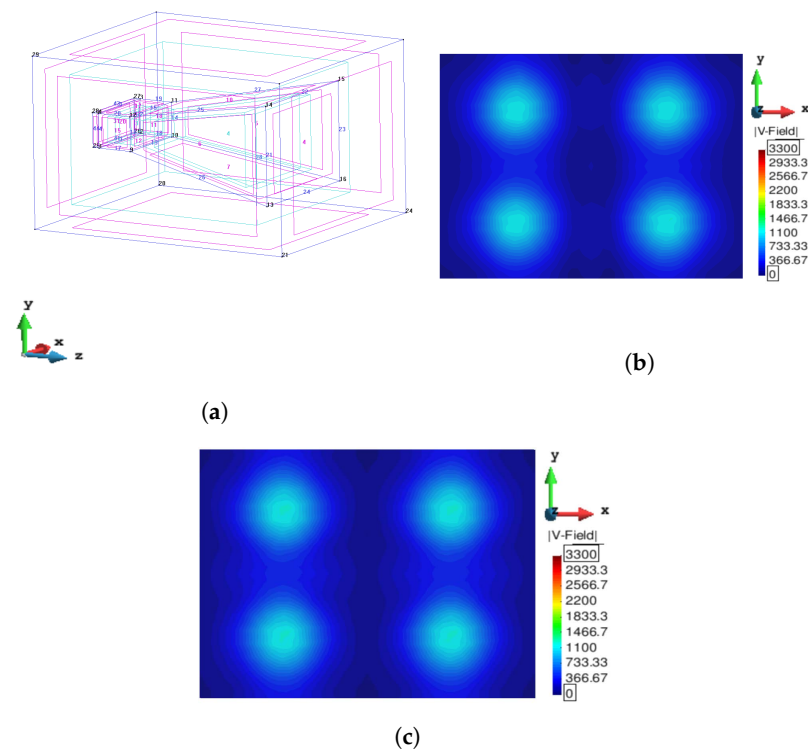


Figure 9. Pyramidal horn in 2×2 configuration. The subfigure (a) represents the definition of the model, (b) is the result without using unit cells, and (c) is the output of the code using one unit cell in a 2×2 configuration.

4. Conclusions

We have followed the TDD paradigm in the development of an in-house code to allow for the finite element analysis of finite periodic type electromagnetic structures (with many potential applications, e.g., analysis of antenna arrays, metamaterials, and so on). We have introduced substructuring techniques based on a unit cell that is virtually repeated on a $1 \times M$ or $M \times M$ to model the structure under analysis. Specifically, on the so-called one Schur complement approach, we manage to model the problem using only one type of Schur complement in contrast to the conventional use of a number of different Schur complements depending on the situation of the associated domain (unit cell) in the structure (i.e., on one side, corner, interior, etc.). As a result, we have obtained faster computations.

These substructuring techniques have been introduced in an existing general-purpose FEM code. Along the process of code development and integration of the software, we have written tests for small increments of functionality; then, we have developed the code to pass the tests; and finally, we have used refactoring to increase the performance and readability of the code. Thus, we have obtained a reliable and maintainable code in less time than using a conventional software development paradigm.

Specifically, we have used unit and integration tests, system tests (where we have applied MMS to isolate single terms within the FEM formulation), and application tests. We have shown convergence tests and the resolution of the code with the approximation of monomials contained in the space of functions that we use. Additionally, we have used smoke tests to have a fast way of debugging all the features of the code. Finally, we have shown a real-life example with an array of pyramidal horns, whereas we have realized a performance analysis of the code.

Author Contributions: Conceptualization, I.M.-F. and L.E.G.-C.; software, I.M.-F.; validation, I.M.-F. and A.A.-M.; investigation, A.A.-M. and L.E.G.-C.; data curation, I.M.-F. and A.A.-M.; writing—original draft preparation, I.M.-F. and A.A.-M.; writing—review and editing, L.E.G.-C.; supervision, A.A.-M. and L.E.G.-C.; resources, L.E.G.-C.; funding acquisition, L.E.G.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been financially supported by TEC2016-80386-P and PID2019-109984RB-C41.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ABC	Absorbing Boundary Condition
DDM	Domain Decomposition Method
FEM	Finite Element Method
MMS	Method of Manufactured Solutions
MPI	Message Passing Interface
TDD	Test-Driven Development

References

1. Jin, J.M. *The Finite Element Method in Electromagnetics*; John Wiley & Sons: Hoboken, NJ, USA, 2015.
2. Thierry, B.; Vion, A.; Tournier, S.; El Bouajaji, M.; Colignon, D.; Marsic, N.; Antoine, X.; Geuzaine, C. GetDDM: An Open Framework for Testing Optimized Schwarz Methods for Time-Harmonic Wave Problems. *Comput. Phys. Commun.* **2016**, *203*, 309–330. [[CrossRef](#)]
3. Xue, M.F.; Jin, J.M. A Hybrid Conformal/Nonconformal Domain Decomposition Method for Multi-Region Electromagnetic Modeling. *IEEE Trans. Antennas Propag.* **2014**, *62*, 2009–2021. [[CrossRef](#)]

4. Peng, Z.; Lim, K.H.; Lee, J.F. Nonconformal Domain Decomposition Methods for Solving Large Multiscale Electromagnetic Scattering Problems. *Proc. IEEE* **2013**, *101*, 298–319. [CrossRef]
5. Farle, O.; Hill, V.; Ingelström, P.; Dyczij-Edlinger, R. Multi-parameter polynomial order reduction of linear finite element models. *Math. Comput. Model. Dyn. Syst.* **2008**, *14*, 421–434. [CrossRef]
6. de la Rubia, V.; Mrozowski, M. A Compact Basis for Reliable Fast Frequency Sweep via the Reduced-Basis Method. *IEEE Trans. Microw. Theory Tech.* **2018**, *66*, 4367–4382. [CrossRef]
7. Zhu, Y.; Cangellaris, A.C. *Multigrid Finite Element Methods for Electromagnetic Field Modeling*; John Wiley & Sons: Hoboken, NJ, USA, 2006; Volume 28.
8. Dyczij-Edlinger, R.; Farle, O. Algorithmic Enhancements, Model-Order Reduction, and Multigrid Aspects in Contemporary Finite Element Implementations. *IEEE Trans. Magn.* **2009**, *45*, 1270–1275. [CrossRef]
9. Paszynski, M.; Pardo, D.; Torres-Verdin, C.; Demkowicz, L.; Calo, V. *A Multi Level Direct Sub-Structuring Multi-Frontal Parallel Solver for the hp-Finite Element Method*; ICES-Report; The University of Texas at Austin: Austin, TX, USA, 2007; pp. 7–33.
10. Martinez-Fernandez, I.; Wozniak, M.; Garcia-Castillo, L.E.; Paszynski, M. Mesh-based multi-frontal solver with reuse of partial LU factorizations for antenna array. *J. Comput. Sci.* **2017**, *18*, 132–142. [CrossRef]
11. Sarabandi, K.; Behdad, N. A frequency selective surface with miniaturized elements. *IEEE Trans. Antennas Propag.* **2007**, *55*, 1239–1245. [CrossRef]
12. Mailloux, R.J. *Phased Array Antenna Handbook*; Artech House: London, UK, 2017.
13. Caloz, C. Perspectives on EM metamaterials. *Mater. Today* **2009**, *12*, 12–20. [CrossRef]
14. Yang, F.; Rahmat-Samii, Y. *Electromagnetic Band Gap Structures in Antenna Engineering*; Cambridge University Press: Cambridge, UK, 2009.
15. de La Rubia, V.; Zapata, J.; Gonzalez, M.A. Finite element analysis of periodic structures without constrained meshes. *IEEE Trans. Antennas Propag.* **2008**, *56*, 3020–3028. [CrossRef]
16. Garcia-Donoro, D.; Garcia-Castillo, L.E.; Sarkar, T.K.; Zhang, Y. A Nonstandard Schwarz Domain Decomposition Method for Finite-Element Mesh Truncation of Infinite Arrays. *IEEE Trans. Antennas Propag.* **2018**, *66*, 6179–6190. [CrossRef]
17. Garcia-Donoro, D.; Amor-Martin, A.; Garcia-Castillo, L.E.; Salazar-Palma, M.; Sarkar, T.K. HOFEM: Higher Order Finite Element Method Simulator for Antenna Analysis. In Proceedings of the 2016 IEEE Conference on Antenna Measurements & Applications (CAMA), Syracuse, NY, USA, 23–27 October 2016; pp. 1–4.
18. Marchand, R.; Davidson, D.B. The Method of Manufactured Solutions in 3D for the verification of Computational Electromagnetics. In Proceedings of the 2012 IEEE International Symposium on Antennas and Propagation, Chicago, IL, USA, 2012; pp. 1–2.
19. Garcia-Donoro, D.; Garcia-Castillo, L.E.; Ting, S.W. Verification Process of Finite-Element Method Code for Electromagnetics: Using the method of manufactured solutions. *IEEE Antennas Propag. Mag.* **2016**, *58*, 28–38. [CrossRef]
20. Mudunuru, M.; Shabouei, M.; Nakshatrala, K. On local and global species conservation errors for nonlinear ecological models and chemical reacting flows. In *ASME International Mechanical Engineering Congress and Exposition*; American Society of Mechanical Engineers: New York, NY, USA, 2015; Volume 57526, p. V009T12A018.
21. Amor-Martin, A. A testbench of arbitrary accuracy for electromagnetic simulations. *Int. J. RF Microw. Comput.-Aided Eng.* **2020**, *30*, e22342. [CrossRef]
22. Jahandari, H.; Bihlo, A. Forward modelling of geophysical electromagnetic data on unstructured grids using an adaptive mimetic finite-difference method. *Comput. Geosci.* **2021**, *25*, 1083–1104. [CrossRef]
23. Adler, J.H.; Cavanaugh, C.; Hu, X.; Zikatanov, L.T. A finite-element framework for a mimetic finite-difference discretization of Maxwell's equations. *SIAM J. Sci. Comput.* **2021**, *43*, A2638–A2659. [CrossRef]
24. Monk, P. *Finite Element Methods for Maxwell's Equations*; Oxford University Press: Oxford, UK, 2003.
25. Dooley, J. *Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring*; Apress: New York, NY, USA, 2017. [CrossRef]
26. Rilee, M.; Clune, T. Towards Test Driven Development for Computational Science with pFUnit. In Proceedings of the 2014 Second International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering, New Orleans, LA, USA, 21 November 2014. [CrossRef]
27. Beck. *Test Driven Development: By Example*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 2002.
28. Clune, T. *Test Driven Development of Scientific Models*; Technical Report; Software Systems Support Office Earth Science Division NASA Goddard Space Flight Center: Greenbelt, GA, USA, 2012.
29. Clune, T.; Womack, B. pFUnit Web Page. 2021. Available online: <https://github.com/Goddard-Fortran-Ecosystem/pFUnit> (accessed on 21 October 2021).
30. Clune, T.; Rood, R. Software Testing and Verification in Climate Model Development. *IEEE Softw.* **2012**, *28*, 49–55. [CrossRef]
31. Garcia-Donoro, D.; Amor-Martin, A.; Garcia-Castillo, L.E. Higher-Order Finite Element Electromagnetics Code for HPC Environments. In *Procedia Computer Science*; Elsevier: Zurich, Switzerland, 2017; Volume 108, pp. 818–827. [CrossRef]
32. Melendo, A.; Coll, A.; Pasenau, M.; Escolano, E.; Monros, A. GiD Home. 2021. Available online: www.gidhome.com (accessed on 21 October 2021).
33. García-Castillo, L.E.; Salazar-Palma, M. Second-Order Nédélec Tetrahedral Element for Computational Electromagnetics. *Int. J. Numer. Model. Electron. Netw. Devices Fields* **2000**, *13*, 261–287. [CrossRef]

34. Salazar-Palma, M.; Sarkar, T.K.; García-Castillo, L.E.; Roy, T.; Djordjevic, A.R. *Iterative and Self-Adaptive Finite-Elements in Electromagnetic Modeling*; Artech House Publishers, Inc.: London, UK, 1998.
35. Balanis, C.A. *Advanced Engineering Electromagnetics*; John Wiley & Sons: Hoboken, NJ, USA, 1989.