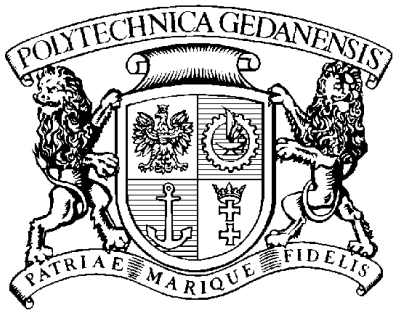
High Performance Algorithms for Large Scale Electromagnetic Modeling

Michał Rewieński, M. Sc.

Ph. D. Thesis

Department of Electronics, Telecommunications
and Computer Science,
Technical University of Gdańsk



Thesis supervisor: Prof. Michał Mrozowski

Gdańsk 1999

To my Grandfather,
who was the first to show me
the beauty of mathematics.

Abstract

This study discusses efficient numerical algorithms for solving operator boundary value problems arising in electromagnetics. It focuses on developing low cost methods capable of dealing with eigenproblems which describe complicated 2D and 3D electromagnetic structures. Original finite-dimensional operator projection techniques, exploiting the concept of implicit operator projection and based on the Finite Difference Frequency Domain (FDFD) and eigenfunction expansion methods are developed, offering reduced computational and memory cost as compared to orthodox approaches. This thesis also proposes a new method of reducing the effect of numerical dispersion, being an important source of errors while modeling electrically large structures using FDFD algorithm. The presented techniques applied jointly with modern Krylov subspace methods lead to numerical solvers which may be efficiently implemented in scalable parallel systems. Examples of application of these solvers to modeling selected waveguiding structures and electrically large resonators are described. The presented computational results validate the proposed algorithms and confirm their high performance and scalability in parallel systems.

Contents

Symbol conventions and abbreviations

General symbols

$\mathbf{A}$	-	linear operator
$\mathbf{A}^*$	-	adjoint operator associated with $\mathbf{A}$
$\underline{\underline{A}}$	-	matrix
$\underline{a}$	-	vector
$\overleftrightarrow{a}$	-	tensor
$B(X,Y)$	-	space of linear operators $\{\mathbf{A}\|\mathbf{A}: X \to Y\}$
C	-	the set of complex numbers
C^1	-	the class of functions with continuous derivatives
C^2	-	the class of functions from C^1 class with continuous second derivatives
$\delta(x)$	-	the Dirac delta distribution
$h(x)$	-	the Heaviside function
$L_2(\Omega)$	-	space of square integrable functions defined over the region Ω
R	-	the set of real numbers
$\sigma(\mathbf{A})$	-	point spectrum of the operator $\mathbf{A}$
v, w	-	functions
X	-	complete, linear (Banach) space
$(\cdot,\cdot)$	-	inner product in a Hilbert space
$\|\|\cdot\|\|$	-	norm in a Hilbert space induced by the inner product

Physical quantities

β	-	propagation constant
ϵ	-	relative permittivity of medium
ϵ_0	-	permittivity of the free space
$\vec{E}_t$	-	transverse electric field intensity
f	-	frequency
$\vec{H}_t$	-	transverse magnetic field intensity
k_0	-	wavenumber in the free space
μ_0	-	permeability of the free space
Z	-	normalized propagation constant

Selected abbreviations

DFT	-	Discrete Fourier Transform
FD	-	Finite Difference discretization method
FEM	-	Finite Element Method
FFT	-	Fast Fourier Transform
GM	-	Galerkin Method
IEEM	-	Iterative Eigenfunction Expansion Method
IRAM	-	Implicitly Restarted Arnoldi Method
MGS	-	Modified Gram-Schmidt orthonormalization algorithm
P_ARPACK	-	Parallel ARnoldi PACKage
TRM	-	Transverse Resonance Method

Symbols related to computational complexity

k, NEV	-	number of eigenvalues to be found
K	-	number of sample points in spatial domain
M	-	number of Fourier coefficients
N	-	matrix problem size
NCV	-	number of computed eigenvalues (size of the constructed Krylov subspace)
P	-	number of applied processors
$O(N)$	-	linear asymptotic growth
$O(N \log N)$	-	linear-logarithmic asymptotic growth
$O(N^2)$	-	quadratic asymptotic growth

Chapter 1

Introduction

1.1 Motivation and background

Dynamic progress occurring in telecommunication and information systems over the past few years, including advances in high frequency telecommunications, navigation, radar systems and computer networks, created an enormous demand for modeling various electromagnetic systems. The problem of characterization of new materials or structures such as waveguides or resonators in 'high frequency' range, became essential e.g. in fiber optics technology or in the design of modern microwave and millimeter wave circuits. Due to progress in manufacturing technology, the mentioned problem refers typically to structures in which the size of geometrical features is comparable to the wavelength. This situation requires application of rigorous analysis of electromagnetic wave interaction with matter. Moreover, the structures being of current interest of science and technology are usually characterized by a non-trivial geometry and/or complex materials. These factors imply the necessity of applying numerical modeling of electromagnetic fields and waves in order to be able to investigate the properties of these structures.

Typically, the physical problems to be modeled include investigating:

1. Electromagnetic response of a structure to the applied fields. – In numerical terms, this normally implies solving a deterministic problem, e.g. a linear set of equations, which allows one to find electromagnetic fields within the considered structure.

2. Free oscillations or waveguiding properties of a given structure. – In this case one usually arrives at an eigenproblem, whose solution involves finding modes of a structure, consisting both of electromagnetic field distributions and corresponding eigenfrequencies or propagation constants.

A typical numerical modeling scenario arising while dealing with the above modeling problems includes the following steps. First, a mathematical formulation of the problem is developed. In electromagnetics, this step consists in building an initial value problem or a boundary value problem involving an integral or differential equation or a set of equations derived from Maxwell's equations. In these formulations the electromagnetic

fields are represented as elements of infinite-dimensional functional spaces. Consequently, in the next step a method of finite-dimensional projection is developed in order to convert the initial problem into the form allowing one to apply an algorithmic procedure to find a desired solution. The last step consists of solving approximately the projected problem in a discrete domain using a computer program. The computer solver usually implements a method capable of dealing with a finite-dimensional (discrete) problem, e.g. a matrix eigenproblem or a system of linear equations. The above general methodology is used to develop virtually all numerical solvers of electromagnetic problems and has served to design various classical methods of solving operator boundary or initial value problems.

The necessity to model continually more complicated structures or systems has led to defining a class of most complicated computational problems which is commonly referred to as 'large scale' problems, which may be characterized by a number of factors, including:

- complicated geometries of the modeled structures,
- large dimensions of the structures, which in case of electromagnetic problems are usually related to the wavelength of modes excited/propagating in a given structure,
- complicated characteristics of materials forming a modeled system.

In computational terms all the mentioned factors, imply:

- introducing sophisticated models which require advanced numerical treatment,
- appearance of large problem sizes (large projection space), usually related to fine discretization,
- application of hi-end computers, often including scalable parallel systems, which may handle computationally demanding tasks.

Consequently, 'large scale' problems generally lie beyond the range of applicability of many orthodox numerical algorithms due to generally high computational and memory complexity of the classical methods, as well as inadequate approximations applied in these methods. In other words, a number of factors or effects, which become more acute in large scale problems, e.g. numerical dispersion, slow convergence rate, high density of operator spectrum cannot be handled efficiently by many 'older' algorithms.

Although the discussion or comparison of specific features of e.g. classical operator projection methods or basic algorithms of solving matrix eigenproblems is far beyond the scope of this work, it may be observed that in general the solvers based on the orthodox numerical techniques are characterized by a fairly high numerical complexity e.g. of $O(N^3)$ and high memory cost of $O(N^2)$, where N is the size of the discrete problem. This kind of numerical and memory cost results unacceptable while trying to model complicated electromagnetic systems. With a problem size N of order 10^4 or 10^5 the quadratic growth of memory complexity implies that the memory storage requirements approach or even exceed the storage capabilities of most available computer systems

(including massively parallel supercomputers). The computation time, associated with the numerical complexity, blows up already for much smaller problem sizes in the case of many classical methods. These facts indicate that many existing techniques of solving e.g. operator boundary value problems become useless while dealing with large scale electromagnetic problems.

In order to be able to model complicated, large scale electromagnetic structures the complexity of the applied techniques has to be reduced. This can be achieved by devising entirely new numerical algorithms and techniques, which allow one to reduce both memory and computational costs at different stages of constructing a numerical solver. This provides the main motivation for this work which investigates different approaches towards numerical solving of operator boundary value problems arising in large scale electromagnetic modeling. These approaches include:

- Applying new formulations of electromagnetic problems which are better suited for modeling of certain classes of structures, as well as allow effective finite-dimensional projection of the problem.
- Finding low cost methods for finite-dimensional mapping of electromagnetic operator equations.
- Investigating fast, low complexity methods of solving discrete operator problems, e.g. matrix eigenproblems.

The issues pointed out above correspond to the basic numerical modeling scenario and define the most up-to-date problems in computational electromagnetics and scientific computing in general. The first of the mentioned points defines the scope of application of a given method and influences the size of the emerging computational problem, the second allows one to reduce primarily the memory storage requirements while the last one has the most important impact on the computational complexity of the resulting numerical solver. In fact, all the three points are closely related to each other and provide the necessary conditions to be fulfilled by high performance numerical solvers, i.e. algorithms which may be used to model efficiently complicated, large scale electromagnetic structures.

1.1.1 Previous work on the efficient algorithms for large scale electromagnetic problems.

In the recent years a number of sophisticated numerical methods have been widely investigated in various application fields within computational electromagnetics. At different stages of construction of a numerical solver, including mathematical problem formulation, designing finite-dimensional projection method and solving the discrete problem a number of techniques have been developed aiming at enhancing efficiency and broadening the range of application of electromagnetic modeling by reducing size of the computational problems, improving numerical properties of the solvers and shortening computation time. These issues are considered in more detail in the following paragraphs.

1.1.1.1 Reducing size of the computational problem

The first group of developments to be considered refers to designing more efficient formulations and projection techniques for electromagnetic modeling, allowing one to reduce both computational and memory cost of numerical solution. These efforts generally concentrate on reducing the number of variables (degrees of freedom) involved in certain mathematical formulations by applying model order reduction, hybrid projection techniques and reducing errors associated with projection/discretization.

The model order reduction techniques aim at representing the dynamics of a given electromagnetic system using a reduced number of degrees of freedom, which allows a more efficient numerical treatment. Recent developments in the area of constructing reduced-order models for electromagnetic systems include application of Krylov subspace methods. Within this research field Lanczos and Arnoldi algorithms [100], [102] seem to have most significance. The Arnoldi iterative scheme has been successfully applied to build reduced-order models in simulation of 3D integrated circuit interconnects. It has been shown that the reduced-order models are very efficiently generated from surface/volume integral formulations [18], [62]. Fast Arnoldi-based techniques of generating reduced-order models for RC and RLC circuits which guarantee stability and passivity have been proposed [38], [105]. A number of methodologies combining Finite Difference Time Domain (FDTD) method with non-symmetric Lanczos algorithm for fast extraction of reduced-order models of electromagnetic responses have been developed and applied to calculation of broad-band response of passive waveguiding structures [16] or transient electromagnetic wavefields in inhomogeneous and lossy media [93]. Lanczos-based model-order reduction techniques, including Padé-via-Lanczos [39] or adaptive Lanczos-Padé sweep (ALPS) [9], have also been applied to simulating multiconductor transmission lines [17] or characterization of certain electromagnetic devices [9].

In problems involving modeling electromagnetic structures, such as waveguides or resonators, for both integral and differential problem formulations, the number of field components is often reduced to two components (e.g. transverse components in case of waveguiding structures) of either magnetic or electric field. This approach is applied in [37], [64], [73], [118] (Finite Difference-based methods, differential formulations), [44] (FD, integral formulations), [81] (Finite Element Method, differential formulation) or [55] (finite volume, integral formulation). Apart from reducing the computational effort associated with solving the problem (related to the size of the variables), these approaches achieve an additional goal of eliminating spurious modes or solutions, due to application of the divergence condition.

Another example is the accuracy improvement in the methods based on finite difference approach. In this class of numerical techniques the accuracy of results depends on the applied computational grid, which directly determines the number of unknowns. This in turn substantially influences both memory and computational cost of the numerical solution of a given problem. Savings in computing resources can be achieved by devising techniques which ensure greater accuracy for the same number of unknowns (the

same computational grid). These techniques include methods of correcting the errors associated with discretization. For instance, the phase errors arising in finite difference-based methods (due to numerical dispersion – cf. [110]) may be corrected by introducing modified FD schemes – cf. [52] (modified higher-order schemes for FDTD – integral formulations) or [83] (modified standard second order schemes for FDTD or FDFD), instead of applying costly finite difference mesh refinement.

1.1.1.2 Improving the underlying linear algebra solvers

Apart from deriving mathematical problem formulations and designing numerical techniques aiming at reducing size of the computational problem, one of crucial paths for development of modern fast and efficient numerical techniques was improving the underlying linear algebra solvers for problems arising from electromagnetic modeling. It appears that one of the most important steps was application of Krylov subspace iterative algorithms to solving eigenproblems and linear systems arising in electromagnetic modeling. The first group of applications of Krylov subspace methods refers to solving eigenproblems arising in modeling of waveguiding structures and resonators. Approaches towards solving non-symmetric standard eigenproblems for large, sparse matrices obtained by discretizing integral forms of Maxwell's equations using Finite Difference technique include application of Arnoldi method [102] with Chebyshev polynomial preconditioning ([44] – modeling dielectric channel waveguides), ([74] – modeling dielectric resonators). The Arnoldi algorithm is also used jointly with multiple deflation techniques and inverse power method to solve non-symmetric standard eigenproblems for dielectric waveguides obtained using integrated Finite Difference (FD) technique over non-uniform meshes [37]. Finite volume discretization applied jointly with Arnoldi solver (instead of QR-based solver) has been shown to reduce dramatically the solution time and memory requirements for the problem of modeling microwave transmission lines [55].

The generalized non-symmetric eigenproblems which arise in numerical modeling of waveguiding structures based on Finite Element Method (FEM) have also been effectively and efficiently treated by using Arnoldi method accelerated using Chebyshev polynomial and inflated inverse iteration technique [81] or the Arnoldi algorithm applied jointly with shift and inverse strategy [72]. In the case of methods applying FEM the size of the numerical problems to be solved appears to be somewhat limited due to costly matrix inverse operations associated with solving a generalized eigenproblem [72]. Still, in general, the proposed techniques succeed in substantially accelerating convergence to the desired set of eigenvalues and eigenvectors and reducing memory requirements of the algorithms as compared to orthodox methods, e.g. based on solving dense matrix eigenproblems. Due to its advantages, the Arnoldi method is also applied to finding eigenvalues in the small signal stability analysis of large electric power systems. In this case the Arnoldi method used together with shift-invert and/or semi-complex Cayley preconditioning [3], [71] results in fast and robust algorithms for selective spectral analysis. It may be concluded that among Krylov subspace methods used to solve eigenproblems the Arnoldi algorithm gained most importance in the electromagnetic modeling due to its fast convergence,

reliability, flexibility (which permits to use it efficiently with other methods and preconditioners) and low memory requirements as compared to older methods.

For the cases when linear systems arise instead of eigenproblems Krylov subspace methods other than Arnoldi process (or Full Orthogonalization Method (FOM) [100]) become more eminent. These include e.g. Biorthogonal Conjugate Gradient (BiCG), Generalized Minimal Residual (GMRES) or Quasi-Minimum-Residual (QMR) algorithms. (cf. [100]). The mentioned methods combined with minimal residual smoothing techniques have been shown to provide very efficient numerical solvers for large sparse complex systems of linear equations arising e.g. from discretization of the electro-quasistatic problems using Finite Integration Technique (FIT) [7], [19].

1.1.1.3 Parallel computation

We have mentioned a number of recently developed approaches with their applications which generally aim at reducing computational effort associated with modeling various electromagnetic problems. These high performance techniques become even more significant in the context of 'large scale' problems, where addressing issues related to memory and computational costs and adequacy of finite-dimensional projection often determines whether a given problem may be numerically tractable. In this case it is necessary to consider high performance algorithms also within the context of parallel processing. From this point of view the term of high performance algorithm gains additional meaning, appearing as a method which makes efficient use of computational and memory resources offered by multiprocessor systems. The computational power offered by modern supercomputers, especially scalable parallel distributed memory systems, providing now the gigaflop or even teraflop peak performance may be used to model more complex, large scale problems *only* if the applied algorithm satisfies additional requirements. These refer mainly to balancing the workload and minimizing data interchange across a large number of processing elements and are found to have a substantial impact on the efficiency of the algorithm executed in parallel environment. Consequently, the existing numerical techniques e.g. of solving operator boundary value problems have to be revised. It is necessary to find out e.g. whether a given problem formulation and numerical finite-dimensional projection technique produce a problem which may be efficiently distributed across the processors and whether a corresponding method of solving a discrete problem, e.g. a matrix eigenproblem may be parallelized efficiently. The challenging problems of designing new scalable numerical solvers, being in the mainstream of current research efforts in scientific computing, provide an important motivation for this study.

In recent years, the research in the fields of scientific computing and computational electromagnetics focused on various numerical techniques and parallelization strategies, which would result in high performance scalable algorithms suitable for different parallel architectures, including e.g. shared or distributed memory systems. (cf. [30] or [43]). A considerable effort has been devoted to investigate whether it is possible to apply efficiently Krylov subspace method in scalable supercomputer systems. An early article by Saad [99] describes efficient parallelization strategies for Conjugate Gradient (CG)

and Generalized Minimal Residual (GMRES) methods. A number of studies present high performance parallel algorithms based Krylov subspace methods applied to solving partial differential equations [51], large scale differential-algebraic systems [12] (GMRES) or large, sparse nonsymmetric systems of equations [8], [88], [103]. The cited works indicate that Krylov subspace methods such as conjugate gradient (CG), biconjugate gradient (BiCG), GMRES or quasi-minimal residual (QMR) are very well suited for parallelization in different system architectures, including the currently most popular multiple instruction multiple data (MIMD) systems [103]. The theoretical performance models also imply that this class of methods may be efficiently applied in parallel systems [108]. The most recent developments include parallelization of Krylov subspace methods for structural finite element analysis [68] (based on domain decomposition scheme) or designing Krylov-based acceleration schemes for solving linear systems in parallel [14].

The apparent success of Krylov subspace methods in parallel environments caused their widespread application in a number of fields which make extensive use of high performance processing, including computational electromagnetics. These highly efficient parallel iterative algorithms for solving linear systems or eigenproblems have been successfully applied in modeling cavity resonators [82], waveguiding structures [34], [72], scattering and radiation problems [25], [113], solving electromagnetic inverse problems [86] or generally wideband electromagnetic simulations [2].

Clearly, the main problem encountered in the mentioned application fields was matching the characteristics of the specific electromagnetic problems and associated formulations or finite-dimensional projection techniques with high performance parallel solvers of linear systems or finite-dimensional eigenproblems. In other words, the existing numerical approaches, e.g. projection methods based on the Finite Element Method, needed to be revised in order to match the requirements of high performance parallel computing, such as balancing the workload across the processors or minimizing the inter-processor communication. Various parallelization strategies or entirely new approaches were developed with the aim to assure that e.g. characteristics of the involved operators or projection procedures would not 'spoil' the parallel performance and scalability of a given numerical solver implemented for a certain system architecture.

In electromagnetic applications, for numerical methods based on the Finite Element (FE) projection techniques, a number of (often sophisticated) parallelization schemes or paradigms had to be developed in order to achieve reasonable parallel performance and scalability. Early works developing parallel FE solvers for electromagnetic scattering problems (for Hypercube system architecture) applied parallel domain decomposition, based on inertial partitioning in order to assure load balancing across the processors and good performance (speedup) [41] [42]. Due to the structure of the operator matrices created by FE projection technique, characterized by random non-zero element distribution, parallelization of FEM-based solvers experienced problems with performance in distributed memory MIMD systems [46], [113]. These difficulties associated with load balancing and amount of inter-processor communications were solved using different strate-

gies e.g. duplicating entire datasets to local memories of all processors (which had a side effect of often unacceptably increasing the total memory storage requirements) [113] (modeling scattering problems), applying load balancing based on finite element tearing and interconnecting scheme [82] (fullwave analysis of microwave circuits and resonators) or matrix reordering techniques reducing matrix bandwidth and increasing its regularity, which also reduced the amount of inter-processor communication [23] (scattering problems). Parallel implementations of solvers for complicated scattering problems have also been successfully developed for distributed memory computers using unstructured finite element method, which does not use mesh partitioning techniques [25]. Other works indicated the role of preconditioners while trying to obtain load balancing in FEM-based method by applying matrix assembly by degrees of freedom [115]. It has been shown that matrix assembly by degrees of freedom also reduces the amount of required inter-processor communication which enhances performance of the parallel solvers [116]. The FEM-based electromagnetic codes were also implemented in shared memory systems using parallel mesh partitioning [114] or bulk synchronous parallel approach [58]. The latter approach has also been applied for distributed memory systems, resulting in rather moderate parallel performances for a number of different FEM-based solvers.

Summing up, the parallelization of electromagnetic solvers based on the Finite Element Method (FEM), especially for distributed memory parallel systems, requires relatively complicated schemes in order to achieve good rates of e.g. parallel speedup. Additionally, it is important to note that FEM-based methods require performing a certain form of matrix inversion, which may result costly, particularly in parallel processing environments [46]. Comparisons made between FEM-based and Finite Difference-based parallel electromagnetic codes [22], [46], [111] indicate that finite difference approach results in much simpler and often more efficient parallelization schemes (as compared to FEM), so in order for FEM codes to be competitive with Finite Difference algorithms in parallel environments, they need to be characterized by significantly better convergence properties. In fact, very good scalable performance is observed for finite difference parallel solvers (FDTD) used in modeling microwave circuit devices [48] or electrically large microwave circuits [49] (using Planar Generalized Yee algorithm) implemented in distributed memory systems.

One should also mention parallel algorithms using Discrete Surface Integral Equation technique to model microwave circuit devices [46], characterized by good parallel performance or parallel solvers utilizing the Method of Moments (MoM) for modeling electromagnetic scattering problems [21], [29], [61], [87]. It is important to note that algorithms in the cited works are based on the explicit method of moments (Galerkin) schemes, which construct appropriate dense operator matrices. Consequently, the main part of parallelization refers not to solution process, but to costly phase of operator projection associated with computing matrix elements (which may be performed very efficiently in parallel). Since, in general, solving very large linear systems involving dense matrices (stored explicitly) is unsuitable for implementation in distributed memory systems due to large inter-processor communication overheads, different approaches towards solving

very large MoM-based problems in scalable systems need to be developed.

1.2 Scope, goals and claim of this work

The main interest of this work are the numerical methods suitable for solving large scale electromagnetic problems. Moreover, the focus is on those numerical algorithms which may be efficiently applied in parallel processing environment. This study does not aim at presenting or proposing several advanced numerical techniques, each applied to a specialized class of problems, e.g. initial value problems for electromagnetic differential operators or boundary value problems for given integral operators. Instead, it tries to perform a careful selection of paths to be followed or approaches to be applied at each stage of construction of a numerical solver (mathematical formulation, projection, numerical solution) which allow one to design high performance parallel algorithms applicable to electromagnetic modeling. A few techniques which, in author's opinion, appear most effective for the investigated class of physical problems are proposed and systematically investigated. These techniques are then applied to the design of numerical solvers for large scale electromagnetic eigenproblems involving differential operators. This class of problems has been selected as it constitutes one of the broadest and most important classes of formulations used in modeling of electromagnetic structures. The discussed algorithms are validated and their performance in scalable parallel systems is assessed. According to the above description, the scope of this work may be summarized in the following points:

- Presenting operator formulations of electromagnetic BVPs involving differential operators and discussing their applicability to solving complex, large scale problems.
- Proposing methods of finite-dimensional projection of electromagnetic operators offering reduced numerical cost as compared to classical techniques.
- Analyzing and comparing selected numerical properties of different operator projection strategies in the context of large scale electromagnetic problems.
- Presenting numerical methods of solving discrete operator eigenproblems and discussing their applicability while dealing with discretized large scale electromagnetic problems in parallel processing environment.
- Applying the proposed numerical techniques in design of high performance algorithms of solving large scale electromagnetic eigenproblems.
- Performing numerical validation of the proposed methods applied to solving selected electromagnetic problems.
- Carrying out performance tests of the discussed algorithms in scalable parallel systems.

In this context, the emerging main goal of this work is constructing high performance parallel algorithms which may be effectively and efficiently applied to solving those electromagnetic problems, in particular eigenproblems of given differential operators, which are too complex to be treated using classical, orthodox numerical methods.

This thesis makes the claim, that high performance numerical methods for large-scale electromagnetic modeling can be obtained by applying several new advanced techniques at different stages of the design of the solver as well as selecting an appropriate operator formulation, which include:

- Methods of implicit finite-dimensional operator projection which reduce memory and computational costs and assure high locality of data and computations as well as good workload balancing.
- Low cost iterative algorithms based on the concept of Krylov subspaces, allowing one to embed finite-dimensional projection of the operator into the process of solution of a numerical problem.
- Methods of reducing errors due to numerical dispersion, for algorithms applying finite difference approximations.
- Algorithms of accelerating the convergence of the Krylov subspace algorithms.

1.3 Chapter outline

This work starts with a formulation (in Chapter 2) of electromagnetic problems in both physical and mathematical terms. The description concentrates on boundary value problems for selected differential operators, suitable for modeling either two-dimensional waveguiding structures or three-dimensional resonant cavities. The derivations of the operators for 2D problems are based on the book by Mrozowski [80]. Chapter 2 also describes some characteristics of large scale electromagnetic boundary value problems.

Chapter 3 presents iterative algorithms of solving operator and matrix eigenproblems, with an emphasis on Krylov subspace methods. The description of the algorithms is based on the paper by Sorensen [107] and the books by Saad [100], [102].

Chapter 4 discusses methods of projection of infinite-dimensional operators, based on the Finite Difference Frequency Domain (FDFD) method and the eigenfunction expansion technique. The chapter includes an originally developed technique for reducing the effects of numerical dispersion in FDFD discretization method for problems defined in the cylindrical coordinate system. It also contains a new hybrid projection scheme for operators arising in modeling 3D electromagnetic systems in cylindrical coordinates. The scheme combines standard FDFD technique with eigenfunction expansion technique. The discussed design of the fast method of calculating inner products for the operators and functions applying the Method of Moments representation has been first proposed

by Mrozowski [78]. This chapter also describes the novel methodology of implicit finite-dimensional projection of electromagnetic operators [95]– [97]. The methodology is applied both to FDFD-based techniques as well as algorithms exploiting eigenfunction expansions, including the FDFD discretization scheme for 3D systems in cylindrical systems developed by Mielewski, Ćwikła and Mrozowski [74].

Chapter 5 concentrates on describing original parallel designs and/or implementations of the algorithms of solving operator eigenproblems. Also a description of a new eigensolver (based on the IRAM algorithm and implicit representation of the input operator) is given.

Chapter 6 discusses application of the previously described eigensolvers to the problems of modeling dielectric waveguides with arbitrary permittivity profiles as well as cylindrical and hemispherical resonators and shows the results of tests validating the algorithms. It also discusses the applicability of the algorithms based of the Method of Moments projection to modeling dielectric waveguides with discontinuous, rectangular permittivity profiles by presenting modifications to the basic eigensolvers and applying two different operator formulations.

Chapter 7 presents a collection of the results of performance tests in selected parallel distributed memory systems as well as some comparison of execution times among the discussed solvers.

Chapter 2

Operator boundary value problems arising in electromagnetics

This chapter defines and briefly characterizes electromagnetic boundary value problems (BVPs), which provide a background for application of the numerical methods to be proposed later on in this work. It focuses on presenting BVPs arising in modeling of electromagnetic structures, such as closed or open waveguides or resonators with dielectric filling. The problems are represented as eigenproblems involving specific differential operators. Their solutions bring essential information on the modeled systems, including propagation constants (or wavenumbers) of modes in waveguides with corresponding modal field distributions or eigenfrequencies in resonators. The knowledge of mentioned parameters is crucial e.g. in the design of certain filters, passive elements or investigation of properties of materials used in microwave frequency range. Moreover, the following description consciously focuses on the operator formulations which:

- use a reduced number of unknowns, e.g. four field components instead of six,
- may be represented as standard eigenvalue problems,

as these two factors have a significant impact on the size of the corresponding discrete problem and performance of the numerical algorithms which apply these formulations to solving large scale electromagnetic eigenproblems. As shown in Chapters 6 and 7 the proposed operator formulations allow one to develop methods which efficiently and effectively deal with complicated electromagnetic problems.

2.1 Formulation of 2D and 3D electromagnetic boundary value problems

This study investigates algorithms applied to the analysis of two general types of electromagnetic structures, i.e. waveguides and resonators (resonant cavities). In physical terms the problems to be considered involve:

- Finding the wavelength or propagation constant of one or more waveguide modes at a given frequency.

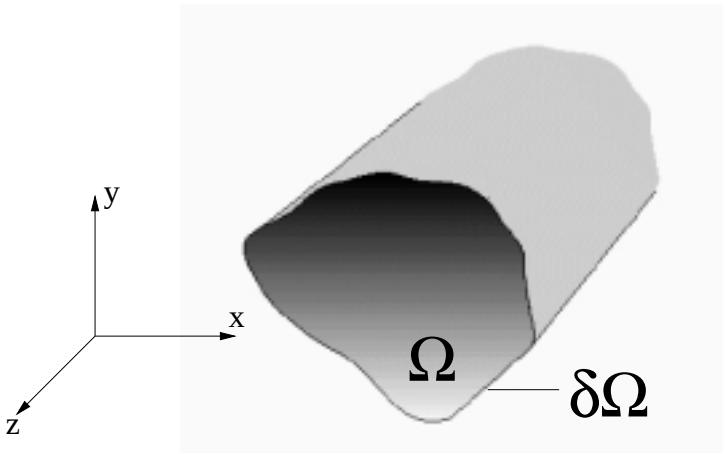


Figure 2.1: A waveguide loaded with inhomogeneous dielectric material in a rectangular coordinate system.

- Finding electromagnetic field distributions corresponding to appropriate waveguide modes.
- Finding resonant frequencies (and corresponding field distributions) in given cavities.

The stated problems are mathematically formulated using the frequency domain version of Maxwell's equations and appropriately defined boundary conditions. Before writing the operator equations one should define properties of spatial domains for problems involving either waveguiding structures or resonant cavities.

A rectangular coordinate system is to be used in the analysis of dielectric waveguides (cf. Fig. 2.1). It is assumed that a waveguide has an arbitrarily shaped cross-section Ω in the $x-y$ plane bounded by periphery $\delta\Omega$ and is filled with an inhomogeneous dielectric material. The structure is uniform along the waveguide axis aligned with the z direction. Consequently, the equations describing this system are invariant with respect to the translation in the z direction which allows one to assume a harmonic variation of the modal field along the waveguide axis z. Then, the 3D problem domain may be easily reduced to a 2D domain, containing the $x-y$ cross-section of the waveguide.

For problems involving resonant cavities (e.g. circular or hemispherical resonators) a cylindrical coordinate system will be used (cf. Fig. 2.2) which gives more convenient mathematical formulations for some classes of structures, e.g. structures with rotational symmetry filled with isotropic materials and/or structures whose boundaries coincide with surfaces $r = \text{const}$, $\phi = \text{const}$ or $z = \text{const}$. In the case of systems with rotational symmetry a 3D problem of finding resonant field distributions is straightforwardly reduced to a problem with a 2D domain having a Cartesian metrics. In a more general case, it is assumed that volume Ω of the resonant cavity is filled with an anisotropic

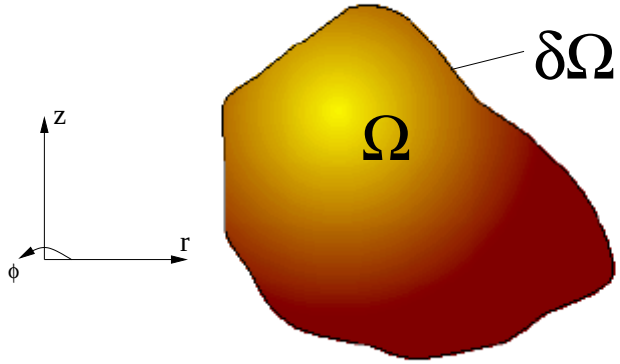


Figure 2.2: An arbitrarily shaped electromagnetic resonant cavity in a cylindrical coordinate system.

material and is bounded by the surface $\delta\Omega$. If the constitutive parameters are described by diagonal tensors, e.g. the permittivity is given as follows:

$$\overleftrightarrow{\epsilon}(r,\phi,z) = \begin{bmatrix} \epsilon_r(r,\phi,z) & 0 & 0 \\ 0 & \epsilon_\phi(r,\phi,z) & 0 \\ 0 & 0 & \epsilon_z(r,\phi,z) \end{bmatrix} \tag{2.1}$$

then, as shown later on, application of cylindrical coordinate system allows one to obtain operator formulation with a reduced number of unknowns. This may be also achieved if the system has a rotational symmetry but an anisotropic medium is rotated about the z axis, so that:

$$\overleftrightarrow{\epsilon}(r,z) = \begin{bmatrix} \epsilon_r(r,z) & \epsilon_{r\phi}(r,z) & 0 \\ \epsilon_{\phi r}(r,z) & \epsilon_\phi(r,z) & 0 \\ 0 & 0 & \epsilon_z(r,z) \end{bmatrix} \tag{2.2}$$

It is also assumed that the electromagnetic field components normal or tangential to the bounding surface $\delta\Omega$ satisfy Dirichlet or Neumann conditions. In physical terms it means that $\delta\Omega$ is either a perfect magnetic or electric conductor and therefore this kind of boundary may be used to describe closed electromagnetic systems or non-radiative open structures in which fields vanish at infinity. Whenever possible the bounding surface coincides with surfaces $x = \text{const}$ or $y = \text{const}$ in rectangular coordinates or $r = \text{const}$, $\phi = \text{const}$ or $z = \text{const}$ in cylindrical coordinates, which simplifies analysis of a given electromagnetic system. The non-radiating modes in open structures are modeled by moving the walls defining the bounding surface sufficiently far away e.g. from the waveguide core.

Assuming a time harmonic ($e^{j\omega t}$) variation of both electric and magnetic fields, the fields in any electromagnetic system may be described by Maxwell's equations:

$$\nabla \times \vec{E}\,(\vec{r}) = -j\mu_0 \overleftrightarrow{\mu}\,(\vec{r})\,\omega \vec{H}\,(\vec{r}) \tag{2.3}$$

$$\nabla \times \vec{H}\,(\vec{r}) = j\epsilon_0 \overleftrightarrow{\epsilon}\,(\vec{r})\,\omega \vec{E}\,(\vec{r}) \tag{2.4}$$

In the above equations $\vec{E}$ and $\vec{H}$ denote the electric and magnetic field intensities, μ_0 and ϵ_0 are the permeability and permittivity of the vacuum, $\overleftrightarrow{\mu}$ and $\overleftrightarrow{\epsilon}$ represent the tensors of relative permeability and relative permittivity of the medium, ω is the angular frequency, j is the imaginary unit and $\vec{r}$ is the coordinate vector, either in rectangular or cylindrical coordinates.

The above equations are complemented by the conditions imposed on the electromagnetic fields at the boundary $\delta\Omega$. If $\hat{n}$ denotes a unit vector normal to $\delta\Omega$, then the boundary conditions at $\delta\Omega$ may be written as follows:

$$\left.\begin{array}{r} \nabla \cdot \overleftrightarrow{\epsilon}\vec{E} \\ \hat{n} \times \vec{E} \\ \hat{n} \cdot \overleftrightarrow{\mu}\vec{H} \end{array}\right\} = 0 \tag{2.5}$$

if $\delta\Omega$ is a perfect electric conductor (PEC) or:

$$\left.\begin{array}{r} \nabla \cdot \overleftrightarrow{\mu}\vec{H} \\ \hat{n} \times \vec{H} \\ \hat{n} \cdot \overleftrightarrow{\epsilon}\vec{E} \end{array}\right\} = 0 \tag{2.6}$$

if $\delta\Omega$ is a perfect magnetic conductor (PMC).

With the general form of electromagnetic boundary value problems defined above one may now derive operator equations suitable for analysis of either waveguiding structures or resonant cavities and using a reduced number of unknowns (field components).

2.1.1 Formulations of electromagnetic BVPs for waveguiding structures

As assumed, the electromagnetic field time variation is described by the $e^{j\omega t}$ factor, which allows one to use equations (2.3) and (2.4). These equations may be further customized for the case of a waveguiding structure shown in Figure 2.1. Due to the uniformity of the waveguide in the propagation (z) direction, Maxwell's equations should be invariant with respect to the translation in that direction. Consequently, one may impose a harmonic variation of electric and magnetic fields in the z direction in the form of $e^{-j\beta z}$, where β is a propagation constant.

One may now decompose the differential operator $\nabla(\cdot)$ in two parts: transverse and longitudinal with respect to the propagation direction:

$$\nabla(\cdot) = \nabla_t(\cdot) + \hat{z}\frac{\partial}{\partial z}(\cdot) \tag{2.7}$$

where $\hat{z}$ is a unit vector in the z direction and ∇_t is a differentiation operator in the $x-y$ plane. Using this decomposition and assuming the harmonic field variation in the z direction, Maxwell's equations (2.3) and (2.4) may be written in the following form:

$$\nabla_t \times \vec{E}\,(x,y) - j\beta\hat{z} \times \vec{E}\,(x,y) = -j\mu_0 \overleftrightarrow{\mu}\,(x,y)\,\omega\vec{H}\,(x,y) \tag{2.8}$$

$$\nabla_t \times \vec{H}\,(x,y) - j\beta\hat{z} \times \vec{H}\,(x,y) = j\epsilon_0 \overleftrightarrow{\epsilon}\,(x,y)\,\omega\vec{E}\,(x,y) \tag{2.9}$$

Next, the electric and magnetic field intensities can also be decomposed into their transverse and longitudinal parts:

$$\vec{E}(x,y) = \vec{E}_t(x,y) + \hat{z}E_z(x,y) \quad \text{and} \quad \vec{H}(x,y) = \vec{H}_t(x,y) + \hat{z}H_z(x,y) \tag{2.10}$$

Substituting the above expressions to the left hand sides of equations (2.8) and (2.9) one gets:

$$\nabla_t \times \vec{E}_t\,(x,y) + \nabla_t \times \hat{z}E_z\,(x,y) - j\beta\hat{z} \times \vec{E}_t\,(x,y) = -j\mu_0 \overleftrightarrow{\mu}\,(x,y)\,\omega\vec{H}\,(x,y) \tag{2.11}$$

$$\nabla_t \times \vec{H}_t\,(x,y) + \nabla_t \times \hat{z}H_z\,(x,y) - j\beta\hat{z} \times \vec{H}_t\,(x,y) = j\epsilon_0 \overleftrightarrow{\epsilon}\,(x,y)\,\omega\vec{E}\,(x,y) \tag{2.12}$$

Performing simple vector manipulations on the above equations gives the following relations for the transverse and longitudinal parts (cf. [80]):

$$\hat{z} \times \nabla_t \times \hat{z}E_z\,(x,y) + j\beta\vec{E}_t\,(x,y) = -j\omega\mu_0\hat{z} \times \overleftrightarrow{\mu}\,(x,y)\,\vec{H}\,(x,y) \tag{2.13}$$

$$\hat{z} \times \nabla_t \times \hat{z}H_z\,(x,y) + j\beta\vec{H}_t\,(x,y) = j\omega\epsilon_0\hat{z} \times \overleftrightarrow{\epsilon}\,(x,y)\,\vec{E}\,(x,y) \tag{2.14}$$

$$\nabla_t \cdot \left(\vec{E}_t\,(x,y) \times \hat{z}\right) = -j\omega\mu_0\hat{z} \times \overleftrightarrow{\mu}\,(x,y)\,\vec{H}\,(x,y) \tag{2.15}$$

$$\nabla_t \cdot \left(\vec{H}_t\,(x,y) \times \hat{z}\right) = j\omega\mu_0\hat{z} \times \overleftrightarrow{\mu}\,(x,y)\,\vec{E}\,(x,y) \tag{2.16}$$

Using equations (2.15) and (2.16) one may derive formulae for E_z and H_z components, which substituted to (2.13) and (2.14) give equations involving only four transverse field components. Although this can be done for the most general form of the permeability and permittivity tensors $\overleftrightarrow{\mu}$ and $\overleftrightarrow{\epsilon}$ (cf. [80]), below it is assumed that they both have a diagonal form:

$$\overleftrightarrow{\epsilon} = \begin{bmatrix} \underline{\underline{\epsilon}}_t & 0 \\ 0 & \epsilon_{zz} \end{bmatrix} \quad \overleftrightarrow{\mu} = \begin{bmatrix} \underline{\underline{\mu}}_t & 0 \\ 0 & \mu_{zz} \end{bmatrix} \tag{2.17}$$

where $\underline{\underline{\epsilon}}_t$ and $\underline{\underline{\mu}}_t$ are diagonal 2×2 matrices.

Such situation occurs for strictly bidirectional waveguides, i.e. guides with an arbitrary cross-section filled either with nongyrotropic uniaxial or biaxial materials whose one principal axis is aligned with the z direction or gyrotropic materials magnetized along the propagation direction. One may note that this class of structures includes all waveguides filled with isotropic media, being of particular interest of many engineering applications.

After performing the appropriate substitutions one obtains the following equations:

$$j\beta\hat{z}\times\vec{E}_t = j\omega\mu_0\underline{\underline{\mu}}_t\cdot\vec{H}_t + \nabla_t\times\left(\frac{1}{j\omega\epsilon_0\epsilon_{zz}}\nabla_t\times\vec{H}_t\right) \tag{2.18}$$

and

$$j\beta\hat{z}\times\vec{H}_t = -j\omega\epsilon_0\underline{\underline{\epsilon}}_t\cdot\vec{E}_t - \nabla_t\times\left(\frac{1}{j\omega\mu_0\mu_{zz}}\nabla_t\times\vec{E}_t\right) \tag{2.19}$$

The above two equations may be cast in the form of the following operator equation:

$$\mathbf{L_T}\Psi_t - \beta\mathbf{M_T}\Psi_t = 0 \tag{2.20}$$

or

$$\begin{bmatrix} \mathbf{L_{Tee}} & 0 \\ 0 & \mathbf{L_{Thh}} \end{bmatrix}\begin{bmatrix} \vec{E}_t \\ \vec{H}_t \end{bmatrix} - \beta\begin{bmatrix} 0 & -\hat{z}\times \\ \hat{z}\times & 0 \end{bmatrix}\begin{bmatrix} \vec{E}_t \\ \vec{H}_t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.21}$$

where

$$\mathbf{L_{Tee}}(\cdot) = \omega\epsilon_0\underline{\underline{\epsilon}}_t(\cdot) - \nabla_t\times\frac{1}{\omega\mu_0\mu_{zz}}\nabla_t\times(\cdot) \tag{2.22}$$

$$\mathbf{L_{Thh}}(\cdot) = \omega\mu_0\underline{\underline{\mu}}_t(\cdot) - \nabla_t\times\frac{1}{\omega\epsilon_0\epsilon_{zz}}\nabla_t\times(\cdot) \tag{2.23}$$

$$\Psi_\mathbf{t} = \left[\vec{E}_t \;\; \vec{H}_t\right]^T \tag{2.24}$$

In the operator equation (2.21) the transverse magnetic and electric fields are coupled. This equation may be viewed as a generalized eigenproblem involving four field components. Still, one may note that this equation can be easily transformed to give two separate equations, each involving only electric or magnetic field. Rewriting (2.21) gives:

$$\mathbf{L_{Tee}}\vec{E}_t = -\beta\hat{z}\times\vec{H}_t \tag{2.25}$$

$$\mathbf{L_{Thh}}\vec{H}_t = \beta\hat{z}\times\vec{E}_t \tag{2.26}$$

Multiplying the first equation by $\hat{z}\times$ and the second equation by $-\hat{z}\times$ one obtains:

$$\hat{z}\times\mathbf{L_{Tee}}\vec{E}_t = \beta\vec{H}_t \tag{2.27}$$

$$-\hat{z}\times\mathbf{L_{Thh}}\vec{H}_t = \beta\vec{E}_t \tag{2.28}$$

If one derives a formula for $\vec{E}_t$ from equation (2.28), substitute it into equation (2.27) and proceed analogously for $\vec{H}_t$ the following two equations are obtained:

$$-\hat{z}\times\mathbf{L_{Tee}}\hat{z}\times\mathbf{L_{Thh}}\vec{H}_t = \beta^2\vec{H}_t \tag{2.29}$$

$$-\hat{z}\times\mathbf{L_{Thh}}\hat{z}\times\mathbf{L_{Tee}}\vec{E}_t = \beta^2\vec{E}_t \tag{2.30}$$

Introducing operators $\mathbf{Q_H}$ and $\mathbf{Q_E}$, given by the left hand sides of equations (2.29) and (2.30), one gets:

$$\mathbf{Q_H}\vec{H}_t = \beta^2\vec{H}_t \tag{2.31}$$

$$\mathbf{Q_E}\vec{E}_t = \beta^2\vec{E}_t \tag{2.32}$$

Comparing operator equation (2.21) to equations (2.29) and (2.30) it is clearly seen that an equivalent physical problem may be solved using the equation involving either a second order differential operator and four field components or a fourth order differential operator and only two field components. Another difference refers to the form of the resulting operator equations. Comparison of equations (2.20) and (2.31) (or (2.32)) yields that the first operator equation may be viewed as a generalized operator eigenproblem with β as an eigenvalue and Ψ_t as a corresponding eigenfunction while the other operator equation is a standard eigenproblem with eigenvalue β^2 and eigenfunction $\vec{H}_t$ (or $\vec{E}_t$). Still, equation (2.20) may also be transformed into a standard eigenproblem by deriving the inverse of operator $\mathbf{M_T}$ which can be done analytically due to the simple form of $\mathbf{M_T}$.

One should be aware that for both formulations only the curl Maxwell's equations have been used to derive the final forms of the operator equations. Consequently, it is possible that, in general, the solutions of (2.21), (2.29) or (2.30) with appropriately defined boundary conditions do not satisfy the divergence equations:[1]

$$\nabla\cdot\epsilon_0\overleftrightarrow{\epsilon}\vec{E} = 0 \quad \text{or} \quad \nabla\cdot\mu_0\overleftrightarrow{\mu}\vec{H} = 0 \tag{2.33}$$

[1]Such solutions may emerge if the domain used in the operator projection is too broad.

A different formulation suitable for modeling of waveguiding structures may also be proposed. In that approach (presented below) the divergence equations are applied to eliminate the longitudinal field components.

Starting with equations (2.8) and (2.9) one may derive formula for $\vec{E}$ from the right hand side of (2.9) and substitute it into equation (2.8). Then one obtains:

$$\nabla_t \times \overleftrightarrow{\epsilon}^{-1} \left(\nabla_t \times \vec{H} - j\beta\hat{z} \times \vec{H} \right) - j\beta\hat{z} \times \overleftrightarrow{\epsilon}^{-1} \left(\nabla_t \times \vec{H} - j\beta\hat{z} \times \vec{H} \right) = k_0^2 \overleftrightarrow{\mu} \vec{H} \quad (2.34)$$

where $k_0^2 = \epsilon_0, \mu_0 \omega^2$.

Assuming that the permittivity and permeability tensors are quasi-diagonal (cf. equation (2.17)) and taking the transverse part of the above vector equation yields:

$$\nabla_t \times \frac{1}{\epsilon_{zz}} \nabla_t \times \vec{H}_t - j\beta\hat{z} \times \underline{\underline{\epsilon}}_t^{-1} \nabla_t \times \hat{z} H_z - k_0^2 \underline{\underline{\mu}}_t \vec{H}_t - \beta^2 \hat{z} \times \underline{\underline{\epsilon}}_t^{-1} \hat{z} \times \vec{H}_t = 0 \quad (2.35)$$

It follows from the divergence equation $\nabla \cdot \vec{B} = 0$ that:

$$H_z = \frac{-j}{\beta \mu_{zz}} \nabla_t \cdot \underline{\underline{\mu}}_t \vec{H}_t \quad (2.36)$$

Substituting the above formula into (2.35) and then multiplying the equation by $\hat{z} \times \underline{\underline{\epsilon}}_t \hat{z} \times$ gives:

$$\hat{z} \times \underline{\underline{\epsilon}}_t \hat{z} \times \left[\nabla_t \times \frac{1}{\epsilon_{zz}} \nabla_t \times \vec{H}_t \right] + \nabla_t \frac{1}{\mu_{zz}} \nabla_t \underline{\underline{\mu}}_t \vec{H}_t - k_0^2 \hat{z} \times \underline{\underline{\epsilon}}_t \hat{z} \times \underline{\underline{\mu}}_t \vec{H}_t - \beta^2 \vec{H}_t = 0 \quad (2.37)$$

An analogous relation holds for the transverse electric field:

$$\hat{z} \times \underline{\underline{\mu}}_t \hat{z} \times \left[\nabla_t \times \frac{1}{\mu_{zz}} \nabla_t \times \vec{E}_t \right] + \nabla_t \frac{1}{\epsilon_{zz}} \nabla_t \underline{\underline{\epsilon}}_t \vec{E}_t - k_0^2 \hat{z} \times \underline{\underline{\mu}}_t \hat{z} \times \underline{\underline{\epsilon}}_t \vec{E}_t - \beta^2 \vec{E}_t = 0 \quad (2.38)$$

As one notes, the above equations provide formulations for waveguiding problems involving separated transverse magnetic and electric fields. Unlike in the former formulation (cf. equations (2.29) and (2.30)) the operators defined by the left hand sides of (2.37) and (2.38) are second order differential equations. As shown later on in this work, the methods applying these second order differential operators are characterized by faster convergence than algorithms which make use of the formulation involving the fourth order operator. Still, in both cases only two electromagnetic field components are involved, which implies that smaller number of variables will be needed to solve numerically the discussed problems.

2.1.1.1 The case of isotropic media

A fairly complicated form of equations (2.37) and (2.38) may be simplified for the case of waveguides filled with isotropic dielectric material. Then $\underline{\underline{\epsilon}}_t = \epsilon\underline{\underline{I}}$, $\epsilon_{zz} = \epsilon$, $\underline{\underline{\mu}}_t = \mu\underline{\underline{I}}$, $\mu_{zz} = \mu$. One may also assume that μ is constant. In such a case, equation (2.37) becomes:

$$-\epsilon\nabla_t \times \frac{1}{\epsilon}\nabla_t \times \vec{H}_t + \nabla_t\nabla_t \cdot \vec{H}_t + k_0^2\epsilon\mu\vec{H}_t - \beta^2\vec{H}_t = 0 \tag{2.39}$$

Applying the vector identity:

$$\nabla \times (A\vec{a}) = \nabla A \times \vec{a} + A\nabla \times \vec{a} \tag{2.40}$$

yields:

$$-\epsilon\left[\nabla_t\frac{1}{\epsilon} \times \nabla_t \times \vec{H}_t + \frac{1}{\epsilon}\nabla_t \times \nabla_t \times \vec{H}_t\right] + \nabla_t\nabla_t \cdot \vec{H}_t + k_0^2\epsilon\mu\vec{H}_t - \beta^2\vec{H}_t = 0 \tag{2.41}$$

Using another identity: $\nabla_t^2\vec{A}_t = \nabla_t(\nabla_t \cdot \vec{A}_t) - \nabla_t \times \nabla_t \times \vec{A}_t$ one obtains:

$$\nabla_t^2\vec{H}_t + k_0^2\epsilon\mu\vec{H}_t + \frac{1}{\epsilon}\left[\nabla_t\epsilon \times (\nabla_t \times \vec{H}_t)\right] - \beta^2\vec{H}_t = 0 \tag{2.42}$$

Using the duality principle an analogous equation may be written for the electric field formulation:

$$\nabla_t^2\vec{E}_t + k_0^2\epsilon\mu\vec{E}_t + \frac{1}{\mu}\left[\nabla_t\mu \times (\nabla_t \times \vec{E}_t)\right] - \beta^2\vec{E}_t = 0 \tag{2.43}$$

The above equations may be rewritten in the operator form:

$$\mathbf{T_E}\vec{E}_t = \beta^2\vec{E}_t \tag{2.44}$$

$$\mathbf{T_H}\vec{H}_t = \beta^2\vec{H}_t \tag{2.45}$$

where operators $\mathbf{T_E}$ and $\mathbf{T_H}$ are defined by the left hand sides of equations (2.42) and (2.43), respectively. One may note, that both equations (2.31) and (2.32) as well as equations (2.44) and (2.45) may be viewed as standard eigenproblems. Then, the squared propagation constants are eigenvalues of $\mathbf{T_E}$ and $\mathbf{T_H}$ and transverse magnetic or electric fields are corresponding eigenfunctions.

2.1.2 Operator equations for resonant cavities

One of the most important cases to be considered while investigating formulations of electromagnetic boundary value problems for resonant cavities is a BVP defined in cylindrical coordinates for a system with rotational symmetry (i.e. a system homogeneous in the ϕ direction). As will result in the chapters that follow, developing a formulation for the mentioned case allows one to propose a discretization scheme and numerical algorithms valid for systems which do or do not possess rotational symmetry.

The derivation starts with a general form of a wave equation obtained from substituting the formula for magnetic field $\vec{H}$ given by equation (2.3) into equation (2.4):

$$c^2 \nabla \times \overleftrightarrow{\mu}^{-1} \nabla \times \vec{E} = \omega^2 \overleftrightarrow{\epsilon} \vec{E} \tag{2.46}$$

where c is the speed of light in the vacuum. The above equation may be viewed as a generalized eigenproblem.

This equation may be rewritten using electric flux density $\vec{D}$ instead of electric field intensity:

$$c^2 \nabla \times \overleftrightarrow{\mu}^{-1} \nabla \times \overleftrightarrow{\epsilon}^{-1} \vec{D} = \omega^2 \vec{D} \tag{2.47}$$

The equation above may be already viewed as a *standard* eigenproblem of an operator given by the left hand side with eigenvalue ω^2 and corresponding eigenfunction $\vec{D}$. The above formulation involves all three electric field components. Nevertheless, it may be transformed so that only two field components appear in the final equation. This operation reduces the size of the numerical problem to be solved. This reduction has in turn a principal influence on memory requirements as well as performance of a numerical solver. Both factors are extremely important while dealing with large scale problems. The reduction of the number of unknowns can be achieved if a modeled structure has rotational symmetry, i.e. its geometry and constitutive parameters do not depend on ϕ. Then, a natural choice of the form of both electric and magnetic field defined in (r, ϕ, z) coordinates is:

$$\vec{D}(r, \phi, z) = \vec{D}_n(r, z) e^{(-jn\phi)} \quad \text{and} \quad \vec{B}(r, \phi, z) = \vec{B}_n(r, z) e^{(-jn\phi)} \tag{2.48}$$

where n is an integer number, $\vec{B}_n = [B_n^r, B_n^\phi, B_n^z]^T$ and $\vec{D}_n = [D_n^r, D_n^\phi, D_n^z]^T$.[2] With a given, fixed value of n the differentiation operation $\frac{\partial}{\partial \phi}(\cdot)$ on magnetic or electric field may be replaced with the multiplication by $(-jn)$ term. The divergence operator (in cylindrical coordinates) may be decomposed as follows:

$$\nabla \cdot \vec{A} = \tilde{\nabla} \cdot \tilde{A} - jnA_\phi \tag{2.49}$$

where $\tilde{A} = [A_r, A_z]^T$ and the definition of $\tilde{\nabla} \cdot (\cdot)$ follows naturally from the form of the divergence operator in cylindrical coordinates. Applying the zero divergence condition for the electric flux density one gets:

$$D_\phi = \frac{-jr}{n} \tilde{\nabla} \cdot \tilde{D} \tag{2.50}$$

where $\tilde{D} = [D_r, D_z]^T$. Using the above formula equation (2.47) may be transformed as follows:

$$c^2 \Gamma \nabla \times \overleftrightarrow{\mu}^{-1} \nabla \times \overleftrightarrow{\epsilon}^{-1} \left(\Pi \tilde{D} - \hat{\phi} \frac{jr}{n} \tilde{\nabla} \cdot \tilde{D} \right) = \omega^2 \tilde{D} \tag{2.51}$$

[2]For a structure which does not possess rotational symmetry the magnetic and electric field may be represented as a superposition of the fields defined by (2.48) with different values of n.

where $\Gamma[A_r, A_\phi, A_z] = [A_r, A_z]$ and $\Pi[A_r, A_z] = [A_r, 0, A_z]$. The above equation defines an eigenproblem involving only two electric field components. This means that the number of variables for this problem is reduced which, as shown in the following chapters, has a substantial impact on the performance of a numerical algorithm used to model the discussed system in a discrete domain.

Equation (2.51) may be written using compact operator notation:

$$\mathbf{S}\tilde{D} = \omega^2 \tilde{D} \tag{2.52}$$

where $\tilde{D} = [D_r, D_z]^T$.

It defines a *standard* operator eigenproblem suitable for modeling resonant cavities with rotational symmetry. Operator $\mathbf{S}$ in the above equation is non-symmetric. The eigenvalues to be found determine resonant frequencies and the corresponding eigenfunctions – modal field distributions in a given structure. An analogue of the above derivation of equation (2.52) is given in Appendix A. The appendix presents a more 'practical' derivation using expanded form of vector Maxwell's equations.

Summing up the above derivations of electromagnetic operator boundary value problems, one may note that in all cases we were able to formulate these problems as *standard* eigenvalue problem involving a *reduced* number of unknowns (two and not three electric or magnetic field components). As shown later on this has important implications on the efficiency of the constructed numerical solvers.

2.2 Numerical modeling of electromagnetic systems using boundary value problems: a typical scenario

The above sections presented derivations of different operator equations which may be used to model electromagnetic fields and waves in selected structures if appropriate boundary conditions are specified. All the derived equations may be viewed as operator eigenproblems with eigenvalues and eigenfunctions to be found. Below, a general scenario of numerical modeling of electromagnetic systems involving solution of operator eigenproblems is presented. The aim of this description is to indicate the key steps of the process of numerical solution typically encountered in modeling of electromagnetic systems.

The first step consists of a selection of a form of an operator boundary value problem to be used to model a given structure. At this stage one determines which quantities will be found, e.g. propagation constants or eigenfrequencies. If the operator problem takes the form of an eigenproblem then its solution consists of eigenpairs having different physical meanings. By selecting a certain eigenproblem one decides whether the eigenfunctions will consist of e.g. three or two field components, which essentially influences the size (number of variables) of the discrete problem to be solved later on. This, in turn, has an

important influence on the performance of the numerical solvers for large scale problems. The choice of an eigenproblem also determines the properties of the discrete problem developed in the subsequent steps, such as e.g. the spectrum of the discrete operator, symmetry or definiteness of the resulting discrete operator. Last but not least, operator equations are obtained in two forms. The first is a standard eigenproblem and the other is a generalized eigenproblem. Each type of the problem requires a different numerical treatment. In general, standard eigenproblems should be preferred in the context of large scale modeling, as solution of generalized eigenproblems typically involves some kind of matrix inversion to be performed which may be very costly in case of large matrix problems.

After the operator formulation has been selected and appropriate boundary conditions have been defined it is necessary to transform the operator, namely perform an operator projection, so as to permit numerical, algorithmic treatment of the problem. The projection process consists of selecting a finite-dimensional basis to be used to represent (and approximate) both the functions from the operator's domain and the operator itself. The finite-dimensional representations of functions include e.g. representation by a finite number of samples (function values), representation using finite Fourier series, orthogonal or non-orthogonal polynomials etc. The mapping of the operator usually follows from the selected finite-dimensional representation of functions from its domain. The properties of the emerging finite-dimensional, projected operator eigenproblem differ from the properties of the initial (infinite-dimensional) eigenproblem. Most importantly, the spectrum of the projected operator is different from the spectrum of the initial operator. Firstly, the number of eigenvalues becomes finite so that an infinite set of eigenvalues from the spectrum of the initial operator does not have its counterparts in the spectrum of the projected operator. The common eigenvalues of the two operators include only those whose corresponding eigenfunctions belong both to the initial operator domain and the applied finite-dimensional projection of the domain. The eigenfunctions of the initial operator which are outside the domain of the projected operator may only have approximate counterparts in the discussed finite-dimensional domain. Concluding, the selected projection method determines whether a certain eigenvalue or its approximation may be found as an eigenvalue of the projected operator or not. Therefore, the projection should be selected so as to assure that the class of desired solutions is included in the set of solutions of the projected problem. The projection method also significantly influences e.g. size of the discrete problem to be solved or definiteness of the resulting operator matrix. Application of either explicit or implicit projection methods (described in detail in Chapter 4) also affects the complexity of the algorithms solving the discussed operator problem.

After the finite-dimensional projection of the initial operator has been performed one is ready to solve numerically an (approximate) operator eigenproblem. A wide variety of algorithms known from linear algebra may be applied. The range of methods which can be used is restricted by the form and properties of the projected operator, including implicit or explicit operator representation, definiteness of the operator, hermitian

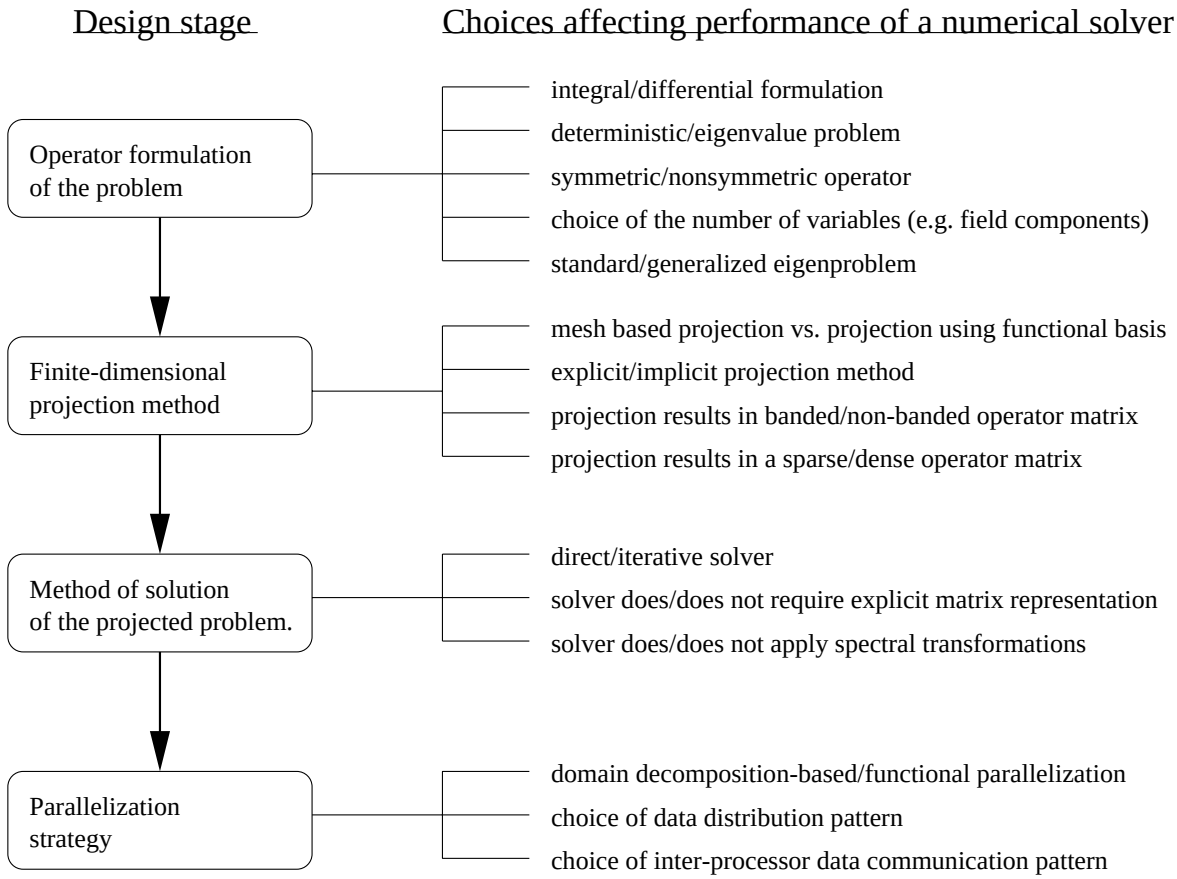


Figure 2.3: Factors affecting performance of a numerical solver at different stages of its construction.

or non-hermitian (symmetric or non-symmetric) form of the operator. A choice of the solution method is also affected by the form of an eigenproblem which may be standard or generalized. Another issue which has to be addressed is whether it is necessary to find e.g. all the eigenvalues of the finite-dimensional operator or only a small subset of the spectrum. In the former case, one may apply e.g. direct methods based on matrix transformations. In the latter case it is often justified to apply e.g. methods which create smaller subspaces including desired solutions or methods which use spectral transformations in order to accelerate the solution process.

Once the method of solving of a discrete operator eigenproblem is selected it is possible to implement a solver of a given operator problem in a computer algorithm. The questions which arise at this stage refer mainly to the problem of parallelization of a given method of solving operator eigenproblem. It is necessary to establish whether an algorithm may be efficiently implemented for use in scalable supercomputer systems or whether further modifications need to be introduced in order to assure high performance of the parallel solver.

Summing up, the cost of obtaining the final solution of a given problem may be controlled by making judicious choices at different stages of the construction of a numerical algorithm. These stages include:

- selecting an operator formulation,
- selecting a method of projection of the operator,
- choosing a numerical algorithm capable of solving the eigenproblem for the approximate, projected operator,
- designing a parallelization strategy for the numerical solver.

The choices affecting performance of a numerical solver available at different stages of its construction are summarized in Figure 2.3.

This chapter aimed at developing operator formulations for selected electromagnetic problems which involve a reduced number of variables (field components) and may be represented as standard eigenvalue problems. The following chapters discuss further stages of construction of a numerical solver. Their main goal is to propose projection and solution techniques satisfying a number of a-priori requirements, which, as shown in the chapters containing results of numerical tests, assure high performance while solving large scale electromagnetic eigenproblems.

Chapter 3

Methods of solving operator and matrix eigenproblems

The previous chapter presented derivations of selected equations involving differential operators which may be applied to modeling either waveguiding structures or resonant cavities. As already mentioned, all of the derived operator equations are characterized by a reduced number of variables and may be viewed as standard or generalized eigenproblems with eigenvalues and eigenfunctions which have to be found. In order to solve numerically the arising problems one constructs an algorithm involving several general stages as described in Section 2.2. Apart from finite-dimensional operator projection, which has to be applied to operators, the most essential part of the algorithm refers to actual solving of a (discrete) operator eigenproblem. This chapter focuses on this stage of the design of numerical solvers.

The following sections give a description of approaches towards numerical solution of operator eigenproblems. They refer either to infinite-dimensional operators or finite-dimensional operators. In the latter case the operator problems may be associated with appropriate matrix problems, which allows one to exploit the ample set of techniques of the linear algebra.

Before starting the presentation of the methods one should define whether all the solutions or only a certain subset of the space of solutions is to be found. This selection determines a class of methods of solving operator eigenproblems which has to be applied in order to meet the needs. For instance, if all the eigenvalues of an operator defined over a finite-dimensional space are to be found the choice of the methods which may be applied is typically restricted to algorithms using diagonalization or tridiagonalization routines e.g. based on orthogonal matrix transformations. The most eminent representatives of this class of methods are classical Jacobi and QR algorithms [50]. Both methods are characterized by a relatively high computational complexity of $O(N^3)$, where N is the matrix operator size (dimension of the operator domain), which limits the scope of applications of the methods to small or medium size problems (N of order 10^3). For larger problems the computation time blows up so that a solution cannot be obtained in

a 'reasonable' time period. Consequently, the problem of finding all eigenvalues of a given operator may be solved only for smaller problems. Nevertheless, in a great deal of engineering and scientific applications it is necessary to find only one or several eigenvalues and eigenfunctions of a given operator. In such a case a much wider spectrum of methods may be used. It includes mainly iterative algorithms capable of dealing with much larger problems, characterized by reduced complexity compared to methods of finding all matrix eigenvalues.

The scope of interest of this work is limited to large scale problems in which size of the matrix obtained as a result of the operator projection is of order $10^5 - 10^6$ or more. Application of methods of finding the entire spectrum of an operator (which, in fact, is not of interest in most electromagnetic applications), e.g. the QR algorithm, to such large scale problems is inevitably associated with enormous cost or simply results in a failure due to both high memory and computational complexity of the algorithm. Consequently, the description below focuses on low-cost methods of finding only selected eigenpairs from the operator's spectrum. Obviously, it is far beyond the scope of this study to present or even mention all the available methods of solving large eigenproblems. Instead, a reference may be be given at this point to an excellent book by Saad [102]. The discussion below is confined to only two methods, namely the Arnoldi and Lanczos algorithms. The mentioned methods belong, in author's opinion, to the most efficient and flexible numerical algorithms which, as shown in a number of publications ([33], [34], [37], [44], [69], [72], [74], [81], [94]– [97]), provide a truly high performance and reliability while solving large scale operator eigenproblems. Before presenting the Arnoldi and Lanczos methods it is worthwhile to give an introduction concerning the simple iteration algorithm (the Power Method).

3.1 The Power Method

The first method to be described is the classical Power Method (a simple iteration method) [50]. Not only is this the simplest but also the most important iterative algorithm for finding operator eigenvalues due to its numerous implications for modern iterative eigensolvers. Numerical methods used throughout this study originate precisely in the Power Method which serves as a basis for the iterative processes. Given the operator $\mathbf{A}$ the steps of the basic version of the Power Method are given as follows:

> Algorithm 1: The Power Method.
>
> Step 0: Choose an initial function v_1 such that $||v_1|| = 1$, assume $k = 1$.
>
> Step 1: Iterate:
>
> Step 1.1: Calculate $w_{k+1} = \mathbf{A}v_k$.
>
> Step 1.2: Normalize: $v_{k+1} = w_{k+1}/||w_{k+1}||$.
>
> Step 1.3: $k := k + 1$.

The main feature of the above method is that it converges to the eigenfunction corresponding to the dominant eigenvalue (the eigenvalue with the largest modulus) of the operator $\mathbf{A}$. In the case of a symmetric operator $\mathbf{A}$ with its eigenfunctions forming an orthonormal basis in the operator's domain it is easy to show (cf. [117]) that:

$$\lim_k \frac{||\mathbf{A}^k v_1||}{||\mathbf{A}^{k-1} v_1||} = \lim_k ||\mathbf{A} v_{k-1}|| = |\lambda_{\max}| \tag{3.1}$$

where $\lambda_{\max}$ is the dominant eigenvalue of the operator $\mathbf{A}$. The other important feature of the above algorithm is that the information on the operator $\mathbf{A}$ is passed to the iterative process only via the $\mathbf{A}v_k$ operation which allows one to apply any kind of implicit representation of the input operator.[1] The main drawback of the above simple method is that it is able to find only a single, dominant eigenvalue and eigenfunction of the operator. Still, the functionality of this algorithm may be extended if deflation and shifting techniques and/or spectral transformations are applied within the iterative process (cf. e.g. inverse iteration method – [50], [102]) allowing one to find other eigenvalues of the input operator.

3.2 Krylov subspace methods

The main reason for presenting the Power Method in the previous section was that during the iterative process *the Krylov subspace* K_m is being constructed:

$$K_m = \mathrm{Span}\{v_1, \mathbf{A}v_1, \ldots, \mathbf{A}^{m-1} v_1\} \tag{3.2}$$

At this point it should be noted that the Power Method exploits only the last two vectors from the basis of the Krylov subspace K_m shown above. This fact provided a basis for the development of the *iterative subspace methods* which exploit the whole Krylov subspace in order to achieve quicker convergence than in the Power Method. These algorithms, which may be used to solve eigenproblems both for infinite-dimensional linear operators and finite-dimensional matrix operators, are currently the most dynamically developing field of research in numerical analysis ([50], [100]). This is mainly due to the fact that the methods may be efficiently parallelized and then applied to construct scalable parallel algorithms capable of solving most complicated large scale eigenproblems arising in mathematical modeling. The most representative examples of modern iterative subspace methods are the Lanczos method (for symmetric or non-symmetric operators), the Arnoldi method (non-symmetric case) or the Davidson algorithm ([31]) (originally designed for symmetric matrices). In these highly effective methods the problem, defined usually for a sparse or structured matrix operator of very large dimension, is reduced to a much smaller dense matrix operator problem. This smaller problem may then be solved by any of the standard techniques used for dense matrix operators. Due to the structure of the three mentioned algorithms they are normally used to find several eigenvalues from

[1] Implicit representation means that the linear operator does not have to be stored (explicitly) e.g. as a matrix of elements, but instead through e.g. a series of linear transformations performed on an input vector.

a spectrum of a given operator. Another numerical method which, to a certain extent, contains the Power Method is the Iterative Eigenfunction Expansion Method (IEEM) [78] which may be used to solve non-symmetric operator and matrix eigenproblems.

3.2.1 The Arnoldi algorithm

This section presents the Arnoldi method which belongs to a class of iterative subspace algorithms capable of approximating a few eigenvalues and the corresponding eigenvectors of a general square matrix. The method is based on constructing an orthonormal basis in a Krylov subspace K_m, spanned by a number of generally non-orthogonal vectors (functions) v_1, $\mathbf{A}v_1$, $\mathbf{A}^2v_1$ and so on. In a classical approach ([4]) the applicability of this technique was strongly limited due to a potentially unbounded growth in storage as well as the lack of numerical stability of the iterative process resulting e.g. in the loss of orthogonality of the computed eigenvectors. These problems have been successfully solved by Sorensen [107] who proposed a modification of the initial Arnoldi algorithm called the Implicitly Restarted Arnoldi Method (IRAM). Exploiting the analogy between the Arnoldi process and the QR iteration the IRAM provides an iterative scheme which has a fixed memory complexity if the number of eigenvalues to be sought is pre-specified. The other advantage of the method is that it preserves the orthogonality of the Arnoldi basis in the Krylov subspace (compare the previous section) if the number of the eigenvalues to be found is not too large.

The Implicitly Restarted Arnoldi Method was found to be a highly efficient tool for solving eigenproblems, capable of reducing both storage requirements and the computation time for a very wide class of large structured non-symmetric matrices in different fields of applications. (cf. [33], [55], [69], [72]) The problem which was found to occur with the IRAM [95] is the significant increment in the number of update iterations with the increasing size of the input matrix.

3.2.1.1 The Arnoldi factorization

In the approach proposed by Sorensen ([107]) the Arnoldi factorization may be treated as a truncated reduction of a given square matrix $\underline{\underline{A}}$ to a form of an upper Hessenberg matrix. This operation is performed in an iterative process and the k-th step of the factorization may be described by the following formula (cf. [50], [102]):

$$\underline{\underline{A}}\,\underline{\underline{V}}_k = \underline{\underline{V}}_k\underline{\underline{H}}_k + \underline{f}_k\underline{e}_k^T \tag{3.3}$$

where:

$\underline{\underline{A}}$ is the input $n \times n$ matrix,

$\underline{\underline{H}}_k$ is a $k \times k$ upper Hessenberg matrix ($k < n$),

$\underline{\underline{V}}_k$ is an $n \times k$ matrix whose columns are Arnoldi vectors ,

$\underline{f}_k$ is a residual vector of size n, satisfying the relation $\underline{\underline{V}}_k^T\underline{f}_k = \underline{0}$.

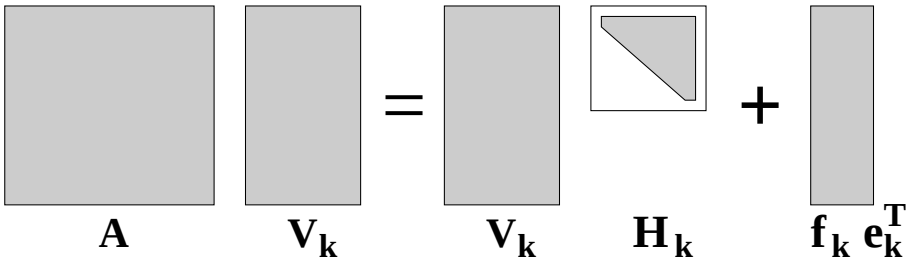


Figure 3.1: The schematic of the Arnoldi factorization.

The idea of Arnoldi factorization is illustrated in Figure 3.1. From equation (3.3) it may been noticed that the process is a truncation of the complete reduction to the Hessenberg form and if the vector $\underline{f}_k$ becomes zero the eigenvalues of the Hessenberg matrix will equal the eigenvalues of the given matrix $\underline{\underline{A}}$. The columns of the matrix $\underline{\underline{V}}_k = [\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_k]$ constructed in the Arnoldi process form an orthonormal basis in the Krylov subspace K_k:

$$K_k = \text{Span}\left\{\underline{v}, \underline{\underline{A}}\,\underline{v}, \underline{\underline{A}}^2\underline{v}, \ldots, \underline{\underline{A}}^{k-1}\underline{v}\right\}$$

where $v \in R^n$ ($v \in C^n$). The basis $\{v_i\}_{i=1,\ldots k}$ is formed in k iterations of the basic Arnoldi algorithm, which may be implemented in a few ways, including the most common, known as the Arnoldi Modified Gram Schmidt algorithm. The steps of this algorithm are given as follows (cf. [102]):

ALGORITHM 2: ARNOLDI-MGS.

STEP 0: Choose an initial vector $\underline{v}_1$ such that $||\underline{v}_1||_2 = 1$

STEP 1: Iterate: For $j = 1, 2, \ldots, k$ do:

STEP 1.1: $\underline{w} := \underline{\underline{A}}\,\underline{v}_j$

STEP 1.2: For $i = 1, 2, \ldots, j$ do:

STEP 1.2A: $h_{ij} = (\underline{w}, \underline{v}_i)$,

STEP 1.2B: $\underline{w} = \underline{w} - h_{ij}\underline{v}_i$

STEP 1.3: $h_{j+1,j} = ||\underline{w}||_2$

STEP 1.4: $\underline{v}_{j+1} = \underline{w}/h_{j+1,j}$

where h_{ij} are the elements of the upper Hessenberg matrix $\underline{\underline{H}}$. It has to be noted that during the factorization process the information on the input matrix ($\underline{\underline{A}}$) is passed to the algorithm only via the matrix-vector product $\underline{\underline{A}}\,\underline{v}_j$. This is an extremely important feature since $\underline{\underline{A}}$ does not have to be known explicitly.

As already mentioned, the reduction of a given matrix $\underline{\underline{A}}$ to the upper Hessenberg matrix is incomplete, and usually $k << n$. This means that only k eigenvalues from the spectrum of A are found. Consequently, the important questions which arise are: What is the accuracy of the computed eigenvalues? Which eigenvalues may be found? These issues are addressed below.

3.2.1.2 Filtering eigenvalues in the Arnoldi method

In the basic Arnoldi algorithm two main problems arise. The first is an undefined number of iterations k necessary to obtain a desired accuracy of the eigenvalues (estimated by calculating the residual norm) which leads to unbounded memory complexity. If one recalls equation (3.3) describing the k-th step of the Arnoldi factorization, it becomes clear that the quality of approximation of the eigenvalues of matrix $\underline{\underline{A}}$ by the eigenvalues of matrix $\underline{\underline{H}}$ is determined by the norm of the residual vector: $||f_k||$. The number of iterations k needed to reduce this norm to an acceptable level is unknown and consequently the size of the matrix $\underline{\underline{V}}_k$ containing the computed orthogonal basis vectors may grow indefinitely. This constitutes the first important problem of the basic Arnoldi algorithm. This problem may be solved by restarting the iterative process after a given, fixed number of iterations with a updated initial vector $\underline{v}_1$, which zeroes the residual vector [106]. At this point one may ask which eigenvalues may be found if this restarted procedure is applied. Sorensen has shown in [106] that the restarted algorithm may be forced to converge to eigenvalues from the edges of the spectrum of $\underline{\underline{A}}$, i.e. eigenvalues with the smallest or largest modulus, smallest or largest real part by applying polynomial filtering and eliminating the "unwanted" eigenvalues at each restart of the method. In this technique, after initial k steps of the basic Arnoldi algorithm, additional p iterations are performed. Next, $k+p$ eigenvalues are found as the eigenvalues of the upper Hessenberg matrix $\underline{\underline{H}}_k$ (the Ritz values) and p "unwanted" eigenvalues are then being filtered out using the implicit shift algorithm with an appropriate filtering polynomial. The algorithm is then restarted with an updated initial vector $\underline{v}_1$ and the subsequent p basic Arnoldi iterations are being performed. The modifications described above introduced to the basic Arnoldi algorithm form a new algorithm known as the Implicitly Restarted Arnoldi Method (IRAM). The question of convergence to eigenvalues located 'far' from the edges of the operator spectrum are discussed in Section 3.2.3.

3.2.1.3 Numerical and memory complexity of the IRAM algorithm

As explained in the previous sections, the Implicitly Restarted Arnoldi Method (IRAM) demonstrates a fixed memory complexity. If the number of the eigenvalues to be found equals k, the number of additional eigenvalues to be computed is p and the input matrix size is n then, denoting $l = k+p$, the algorithm requires $n \cdot O(l) + O(l^2)$ storage. Lehoucq et al. suggest ([67]) that p should equal k in order to obtain an efficient algorithm with good convergence rate. Then the memory complexity equals $n \cdot O(k) + O(k^2)$. If one keeps in mind that k is much smaller than n, it results that the IRAM itself requires very little storage. (Obviously some extra storage may be required to perform the matrix-vector

product $\underline{\underline{A}} \cdot \underline{v}$ operation, but it should not exceed n^2).[2]

The numerical complexity may only be assessed for a single update in a p-step IRAM algorithm. If the cost of the matrix-vector product (step 1.1 of the factorization) is excluded then the complexity equals $O(p^2 n)$. If one assumes that $p = k$ ($p = O(k)$) then the numerical cost becomes of $O(k^2 n)$. Once again, as the number of eigenvalues k to be found is normally much smaller than the problem size n, a linear complexity is obtained. Obviously the cost of the operation of matrix-vector product may be significantly higher, reaching $O(n^2)$ in the worst case and result in a quadratic overall complexity. Still, as shown later in this work, this cost may be reduced in two cases. The first case occurs when the given input operator matrix $\underline{\underline{A}}$ is sparse. The other case occurs when the discrete operator does not have to be represented explicitly by computing (and storing) the elements of the corresponding matrix $\underline{\underline{A}}$.

3.2.2 The Lanczos algorithm

This section briefly describes the other algorithm belonging to a class of iterative Krylov subspace methods, characterized by low computational and memory cost. Besides the Arnoldi method, the Lanczos algorithm appears to be the most widely used computational procedure, applied in various scientific and engineering problems, e.g. simulation of multiconductor transmission lines [17] or characterization of different electromagnetic devices (e.g. T-junctions between waveguides) [9]. The basic version of Lanczos algorithm applies to symmetric operators and is a simplification of the Arnoldi method for the particular symmetric case (cf. [100] or [102]). In this case the Hessenberg matrix $\underline{\underline{H}}_k$ constructed during Arnoldi/Lanczos factorization becomes symmetric tridiagonal, which leads to a three-term recurrence in the Arnoldi process. A much more important version of the Lanczos algorithm is the non-symmetric Lanczos method (Lanczos biorthogonalization method). In this case significant differences between Lanczos and Arnoldi algorithms show up. In the Arnoldi method an orthogonal basis in the Krylov subspace K_m was being constructed. The non-symmetric Lanczos method builds a pair of biorthogonal bases for the two subspaces:

$$K_m = \mathrm{Span}\{v_1, \mathbf{A}v_1, \ldots, \mathbf{A}^{m-1}v_1\} \tag{3.4}$$

and

$$\tilde{K}_m = \mathrm{Span}\{w_1, \mathbf{A}^T w_1, \ldots, (\mathbf{A}^T)^{m-1}w_1\} \tag{3.5}$$

The construction of the bases is performed in the following steps [100]:

[2]Other authors [33] indicate that the choice $p = k$ may not be the optimal one and propose a choice of the value of p as a function of the problem dimension n in order to obtain faster convergence. In this case the theoretical memory complexity is $O(n^2)$, still in the applications presented in [33] it does not result in high memory requirements.

ALGORITHM 3: LANCZOS METHOD.

STEP 0: Choose the initial vectors $\underline{v}_1$, $\underline{w}_1$ such that $(\underline{v}_1, \underline{w}_1) = 1$

STEP 1: Set $\beta_1 = \delta_1 \equiv 0$, $\underline{w}_0 = \underline{v}_0 \equiv 0$

STEP 2: Iterate: For $j = 1, 2, \ldots, m$ do:

STEP 2.1: $\alpha_j = \left(\underline{\underline{A}}\,\underline{v}_j, \underline{w}_j\right)$

STEP 2.2: $\underline{\hat{v}}_{j+1} = \underline{\underline{A}}\,\underline{v}_j - \alpha_j \underline{v}_j - \beta_j \underline{v}_{j-1}$

STEP 2.3: $\underline{\hat{w}}_{j+1} = \underline{\underline{A}}^T \underline{w}_j - \alpha_j \underline{w}_j - \delta_j \underline{w}_{j-1}$

STEP 2.4: $\delta_{j+1} = |(\underline{\hat{v}}_{j+1}, \underline{\hat{w}}_{j+1})|^{1/2}$

STEP 2.5: $\beta_{j+1} = \left(\underline{\hat{v}}_{j+1}, \underline{\hat{w}}_{j+1}\right)/\delta_{j+1}$

STEP 2.6: $\underline{w}_{j+1} = \underline{\hat{w}}_{j+1}/\beta_{j+1}$

STEP 2.7: $\underline{v}_{j+1} = \underline{\hat{v}}_{j+1}/\delta_{j+1}$

The scalars α_m, β_m and δ_m computed in the above algorithm may be treated as elements of the following tridiagonal matrix $\underline{\underline{T}}_m$:

$$\underline{\underline{T}}_m = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \ldots & 0 & 0 & 0 \\ \delta_2 & \alpha_2 & \beta_3 & \ldots & 0 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & \ldots & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ 0 & 0 & 0 & \ldots & 0 & \delta_m & \alpha_m \end{bmatrix} \tag{3.6}$$

The following relations link matrices $\underline{\underline{A}}$ and $\underline{\underline{T}}_m$:

$$\underline{\underline{A}}\,\underline{\underline{V}}_m = \underline{\underline{V}}_m \underline{\underline{T}}_m + \delta_{m+1} \underline{v}_{m+1} e_m^T \tag{3.7}$$

$$\underline{\underline{A}}^T \underline{\underline{W}}_m = \underline{\underline{W}}_m \underline{\underline{T}}_m^T + \beta_{m+1} \underline{w}_{m+1} e_m^T \tag{3.8}$$

$$\underline{\underline{W}}_m^T \underline{\underline{A}}\,\underline{\underline{V}}_m = \underline{\underline{T}}_m \tag{3.9}$$

where $\underline{\underline{A}}$ is the input $n \times n$ matrix and $\underline{\underline{V}}_m$ and $\underline{\underline{W}}_m$ are $n \times m$ matrices consisting of columns of biorthogonal vectors $\underline{v}_m$ and $\underline{w}_m$ respectively, computed in Algorithm 3.

In the context of the above formulae one notes that $\underline{\underline{T}}_m$ is a projection of $\underline{\underline{A}}$ obtained from oblique projection procedure onto K_m and orthogonally to $\tilde{K}_m$ (cf. (3.4) and (3.5)). It is also clear that selected eigenvalues of matrix $\underline{\underline{A}}$ are approximated by eigenvalues of tridiagonal matrix $\underline{\underline{T}}_m$.

The memory cost of the non-symmetric Lanczos method is very low, as only 6 vectors of size n are needed during biorthogonalization procedure, for any value of m. This is usually less than the storage cost needed to perform Arnoldi factorization, which equals $O(mn)$ (cf. Section 3.2.1.3). Still, analogously as for the Arnoldi method, the additional cost of representing the operator matrix has to be taken into account. This cost often determines the overall storage requirements, e.g. if the operator is represented by a dense matrix with matrix elements computed explicitly.

The memory cost of the Lanczos algorithm is lower than for the Arnoldi (IRAM) method. On the other hand, the disadvantage of the Lanczos method is that there are potentially more opportunities for breakdown than in the Arnoldi algorithm. These breakdowns happen when the computed values of δ_{j+1} from Step 2.4 of the Lanczos biorthogonalization procedure become close or equal zero (cf. Algorithm 3). The computed basis vectors are then scaled by small quantities and the cumulative effect of these scalings may introduce very significant rounding errors. This problem is solved by applying the *look-ahead* strategy [89]. Unfortunately, this modification is associated with significant extra computational complexity, as well as causes the matrix $\underline{\underline{T}}_m$ to cease to be tridiagonal.

Referring to computational complexity of the non-symmetric Lanczos algorithm, the cost of performing m iterations of the algorithm is $O(mn)$. In this estimation the cost of performing the matrix-vector products $\underline{\underline{A}}\,\underline{v}_j$ and $\underline{\underline{A}}^T\underline{w}_j$ in steps 2.1 and 2.3 has been excluded. Once again, this cost is theoretically lower than for the Arnoldi method (which equals $O(m^2n)$. Still, if the IRAM algorithm is considered then m becomes constant and the complexity of this algorithm becomes linear. In fact as $m << n$ the complexity of both algorithms is quasi-linear and a similar performance is observed. Obviously, similarly as for the Arnoldi method, the cost of the matrix-vector product operation may significantly raise the overall computational cost of the algorithm. Additional complexity in the Lanczos algorithm is associated with the fact that products involving both initial and transposed matrix have to be computed.

Nevertheless, as already mentioned, the Lanczos algorithm is used in a great variety of application fields, including large scale electromagnetic modeling. In the applications the algorithm appears either in standard version valid for solving eigenproblems or versions valid for solving linear systems of equations. Often more specialized algorithms are constructed on top of Lanczos biorthogonalization, e.g. Padé via Lanczos (PVL) method used in modeling of certain electromagnetic devices (cf. [16], [17], [93], [100]).

3.2.3 Convergence in the Krylov subspace methods

Although, as shown in the previous sections, both the Arnoldi and Lanczos methods are characterized by relatively low memory and computational complexity their performance was found to depend on various properties of the input operator matrix for which an

eigenproblem is being solved. Below a few factors affecting the convergence of the iterative Krylov subspace methods are outlined.

The rate of convergence is determined by the number of iterations e.g. during Lanczos biorthogonalization or e.g. the number of implicit restarts of the iterative process in the IRAM algorithm. A noticeable effect while using the discussed methods of solving operator eigenproblems is the increment of the number of iterations (and implicit restarts) with the increasing problem size (This problem is discussed in Chapter 4. See also Table 4.1). The substantial reason for this phenomenon is the increment of the spectral radius of an operator matrix with the increasing problem size. This effect widely occurs virtually always if the matrix is a projection (discretization) e.g. of a differential or integral equation. For instance, if the Laplace operator is discretized using Finite Difference technique (discussed in the following chapter), then doubling the number of sampling points in every spatial direction results in an approximately four-fold increment of the spectral radius.

There is a number of other factors which influence the number of iterations needed to obtain convergence, which include: existence of multiple eigenvalues in the matrix spectrum (e.g. corresponding to degenerate modes), definiteness of the matrix, hermitian or non-hermitian form of the operator matrix (which determines existence of complex eigenvalues).

Another issue which arises while analyzing the Krylov subspace eigensolvers is the convergence to eigenvalues located 'far' from both ends of the operator spectrum, i.e. eigenvalues separated from $\lambda_{\min}$ and $\lambda_{\max}$ by a large number of other eigenvalues, where $\lambda_{\min}$ and $\lambda_{\max}$ are correspondingly the eigenvalues with the smallest and the largest modulus. If e.g. the 1000-th eigenvalue (looking from hi-end of the spectrum) is to be found, application of the Arnoldi (IRAM) algorithm will result in excessively long iterative process, due to a large size of the Krylov subspace which has to be considered and the obtained results are likely to contain significant errors. Consequently, additional convergence acceleration techniques have to be applied. (The techniques based on polynomial filtering (already described in Section 3.2.1.2) applied in Implicitly Restarted Arnoldi Method accelerate the convergence to the eigenvalues located on the 'edges' of the operator spectrum, i.e. to eigenvalues with the largest or smallest modulus or the largest or smallest real part.) The acceleration methods are most frequently based on transformations of the operator spectrum, i.e. operator preconditioning. Applying the appropriate polynomials (e.g. based on Chebyshev polynomials) the initial operator problem is transformed so that the desired eigenvalues are moved from the interior of the spectrum of the initial operator to the far end of the spectrum of the modified operator. Then, the eigenvalues (and corresponding eigenfunctions) may be easily found as e.g. largest modulus eigenvalues of the modified operator. As the operator eigenfunctions are invariant to polynomial transformations, they may serve to compute the actual eigenvalues of the initial, non-transformed operator. An example of an algorithm performing a spectral transformation has been described in detail in Section 6.2.1 and Appendix C.

3.3 Other methods of solving operator and matrix eigenproblems

Above we have focused on presenting the two Krylov subspace algorithms of solving operator and matrix eigenproblems playing a crucial role in modern computational electrodynamics and scientific computing in general. At this point let us only mention some other recent developments concerning algorithms of solving operator eigenproblems. The most important include modifications in the Davidson method leading to algorithms suitable for non-symmetric matrices, e.g. the Jacobi-Davidson algorithm or the introduction of look-ahead strategy to two-sided Lanczos algorithms. The investigations also include the designs of algorithms which inherently assume parallel computations. An example for such method is the divide and conquer algorithm ([5]) with extensions exploiting the relationship between a certain matrix algebra and complex polynomials ([6]). The detailed description of the methods outlined above is clearly far beyond the scope of this limited study and may be found in many excellent books, including classical book by Wilkinson and Reinsch [119], the monograph by Golub and van Loan [50] which broadly covers the questions of non-symmetric eigenproblems, the book by Saad [100] or the paper by van der Vorst and Golub [117] which presents a review of recent developments.

3.4 Summary

This chapter presented in detail two modern Krylov subspace methods, namely the Arnoldi and Lanczos algorithms. Both methods are generally characterized by low computational and memory cost. Still, as noted earlier, this cost largely depends on the cost of performing the $\underline{\underline{A}}\,v$ product, where $\underline{\underline{A}}$ is a given discrete operator, which, in turn, depends primarily on the projection method applied to the initial operator $\mathbf{A}$. Consequently, this imposes a number of requirements which should be satisfied by a finite-dimensional projection method, as to ensure high performance of the numerical solver based on a Krylov subspace method. These include:

- low cost of the projection procedure,
- low memory cost associated with the representation of the projected operator,
- low (preferably linear) cost of computing the $\underline{\underline{A}}\,\underline{v}$ product,
- possibly a small spectral radius of the projected operator,
- ability to exploit the fact that only the outcome of the $\underline{\underline{A}}\,\underline{v}$ operation has to be known during the iterative process.

Additionally the projection method should allow efficient and simple parallelization and should feature scalability comparable to the scalability of the Krylov subspace methods (This issue is discussed later on in Chapter 5). The main goal of the following chapter is to find projection methods which would most closely fit the requirements set above.

Chapter 4

Cost reducing projection methods of infinite-dimensional electromagnetic operators

The previous chapter described selected methods of solving operator eigenproblems putting aside the questions concerning the form, representation or domain of the operator involved in the computations. In the case of linear operators in finite-dimensional domains, the algorithms presented above, e.g. the Arnoldi or Lanczos methods give recipes ready to be used to find numerically eigenvalues and corresponding eigenvectors of the discrete operator at hand. Still, if an infinite-dimensional operator is involved one has to project it onto a certain finite-dimensional space before the mentioned methods of solving eigenproblems may be applied. In the discussion it was found that the key factor influencing the performance of the numerical solvers based on the Krylov subspace methods is the form and representation of the projected operator. A number of requirements to be satisfied by an 'ideal' projection method have been listed. Consequently, in this chapter we will try to propose the projection techniques which meet the following general goals:

- Assure low memory cost associated with a discrete operator representation and the low cost of projection procedure.
- Allow low cost of computing the $\underline{\underline{A}}\,\underline{v}$ product.
- Generate projected operators with relatively small spectral radii.
- Allow efficient parallel implementation in scalable systems.

Before moving to the descriptions of specific numerical methods, it is worthwhile to study in more detail the above goals in the context of finite-dimensional operators.

The first general goal (requirement) which has been pointed out is the low memory cost of the representation of the projected, discrete operator. This cost depends on several factors, such as:

- The dimension of the domain of the discrete operator or the size of the matrix associated with the discrete operator, which is directly related to the size of the basis of the space onto which an initial operator has been projected as well as the number of variables (e.g. field components) associated with a given operator formulation,

- Representation of the operator matrix which may be: 1) explicit, when the elements of the matrix are computed and stored and the operation of a matrix vector product consists of computing appropriate inner products of matrix and vector elements; 2) implicit, when the operations performed by the matrix are stored (implemented) instead of matrix elements and the computation of a matrix-vector product consists now of performing a number of linear transformations on the input vector,

- The form of the discrete operator matrix, e.g. sparse or dense character of the matrix, regular or irregular distribution pattern of non-zero matrix elements, existence of block structure of the matrix or bandwidth of the matrix.

In this context it is apparent, that if a matrix of a projected operator is sparse, banded and has a very regular pattern of distribution of non-zero elements, then it is very likely that memory cost associated with its representation will be lower than for a dense matrix or a non-banded sparse matrix with irregular distribution of non-zero elements. Moreover, if a discrete operator may be represented implicitly, rather than explicitly, using a matrix of elements, then memory requirements may further be reduced.

One may note that the cost of computing the matrix-vector product is also directly related to the size and form of the matrix. Once again it turns out that this cost is usually lower for sparse, 'structured' matrices, i.e. matrices which are e.g. banded, regular or have a block structure.

The spectral radius, which has an important influence on the performance of the Krylov subspace methods, depends on various factors. For instance, as shown later in this chapter, in a simple case of discrete operators constructed using Finite Difference technique it may be directly controlled by the number of applied discretization points. Another requirement to be met by a projection method is the ability to exploit the fact that only the outcome of the $\underline{\underline{A}}\,\underline{v}$ operation is used by iterative Krylov subspace solvers. This requirement is directly related to the question of representation of the projected operator.

This short discussion reveals the extremely important role of finite-dimensional operator projection. The various projection (and discretization) methods which exist allow one to control the numerical properties (e.g. spectral radius), the form and representation of the emerging finite-dimensional operator and consequently provide means to substantially influence the convergence of the Krylov subspace methods as well as to reduce the memory storage requirements and/or the numerical cost of performing the $\mathbf{A}v$ operation, where $\mathbf{A}$ should be understood as a finite approximation of the initial operator and v should as perceived as a corresponding representation of the function from the operator's

domain. Another issue which has not been addressed yet is the impact of the projection method and the resulting form and representation of the discrete operator on the scalability of the constructed numerical eigensolver. In other words, the applied projection method often determines whether the emerging discrete operator may be efficiently represented in a parallel environment and whether the entire method of solving the discrete eigenproblem may be parallelized in order to obtain a high performance solver. This in turn, defines the scope of application of the solver, determining whether is may be applied to modeling large scale ('grand challenge') computational problems, requiring the processing power of massively parallel supercomputers. All the issues concerning parallel design and performance of the algorithms of solving operator eigenproblems in scalable parallel systems and relations between parallelization strategy and the form of the discrete operator are discussed in Chapter 5.

Returning to the projection techniques, the problem of defining a finite-dimensional mapping refers obviously both to operators and to the functions belonging to the operator's domain. There is a great variety of finite representations of functions, with a vast majority based on expansions in terms of a chosen set of basis functions. Obviously, even a short description of the most popular functional bases lies far beyond the scope of this work. Nevertheless, some general classes of representations may be distinguished, starting from simple representations based on regular or irregular sampling of a function in its domain to the *entire domain expansions*, *entire subdomain expansions* or *domain subdivision expansions* [79] in which accuracy of the representation depends correspondingly on the number of expansion terms or (in the third case) the number of subdomains or sampling points within the domain. (The Finite Difference (FD) discretization method presented later on in this chapter belongs clearly to the domain subdivision methods.) Apart from different finite mappings of functions also various operator representations may be selected which gives rise to a number of numerical procedures. If, for instance, the operator projection is achieved by calculating scalar products, as in the Method of Moments (the Galerkin Method) then for different representations of functions various methods are obtained e.g. the Finite Element Method (FEM) with a resulting sparse operator matrix having usually an irregular distribution of non-zero elements or the collocation (or point matching) technique with a resulting sparse or dense matrix. The discussion of functional expansion and discretization techniques may be found in a number of books – cf. [28], [40], [59], [80], [79].

The rest of this chapter aims at presenting and proposing projection methods which would meet the requirements concerning the reduction of computational cost as well as produce discrete operators which suit the Krylov subspace methods of solving operator eigenproblems described in the previous chapter. The discussion below concentrates on two basic types of finite-dimensional mapping methods: the algorithms based on finite difference discretization and techniques based on the functional expansion. It also presents a hybrid approach in which both techniques are used together. Although the mentioned projection methods produce approximate finite-dimensional operators with entirely different properties, they all try to meet the outlined general goals. In other

words, the specifics of all representations enable one to 1) reduce the cost of performing the $\mathbf{A}v$ operation, which has a substantial impact on the efficiency of numerical solving of given eigenvalue problems; 2) efficiently implement operations involving the discrete operators in parallel distributed memory environments (as shown in the following Chapters); 3) efficiently apply Krylov subspace methods to solve the emerging eigenproblems; 4) Control the spectral radius of the emerging discrete operator.

4.1 Finite difference methods in frequency domain (FDFD)

The sections below describe discretization techniques based on the finite difference (FD) approximation. Generally speaking, the FD approach results in finite-dimensional problems that involve highly regular sparse matrix operators which, as shown in the next Chapter, allow straightforward and efficient parallel decomposition. Therefore they appear to be well suited to solving large scale electromagnetic problems. On the other hand, the resulting discrete problem sizes may become very large in the case of complicated electromagnetic systems which causes increment in memory complexity. Moreover, significant errors associated with numerical dispersion appear for large scale problems and serious decrement in convergence rate of the iterative eigensolvers, such as IRAM is observed (associated with increment of the spectral radius of the matrix) (cf. Table 4.1). These problems may be treated correspondingly by applying implicit operator projection, using methods reducing the effect of numerical dispersion and introducing hybrid discretization techniques, joining eigenfunction expansion techniques with the finite difference method (e.g. in order to obtain operators with reduced spectral radius). All the mentioned techniques are addressed below and indicate that the solvers applying the FD method, if carefully implemented, may be effectively used to model numerically large scale problems.

4.1.1 Simple FDFD discretization for 2D problems

The Finite Difference (FD) method is one of the simplest and very commonly used algorithms of operator discretization. In this method, the functions from the domain of the given operator are represented either as simple sets of values sampled over a certain region or as expansions with simple (usually piecewise linear) expansion functions defined over rectangular subdomains. As already mentioned, this method belongs to a class of domain subdivision expansions which become more accurate with a growing number of subdomains or sampling points [110].

The FD method is most frequently used to discretize various differential operators and consists in substituting the differentials by the finite-dimensional difference operators. The finite difference operators may yield various forms, starting from simple 2-point stencils valid for approximating the first order derivatives in one dimension to complex multipoint stencils used to obtain higher accuracy or deal with higher order derivatives,

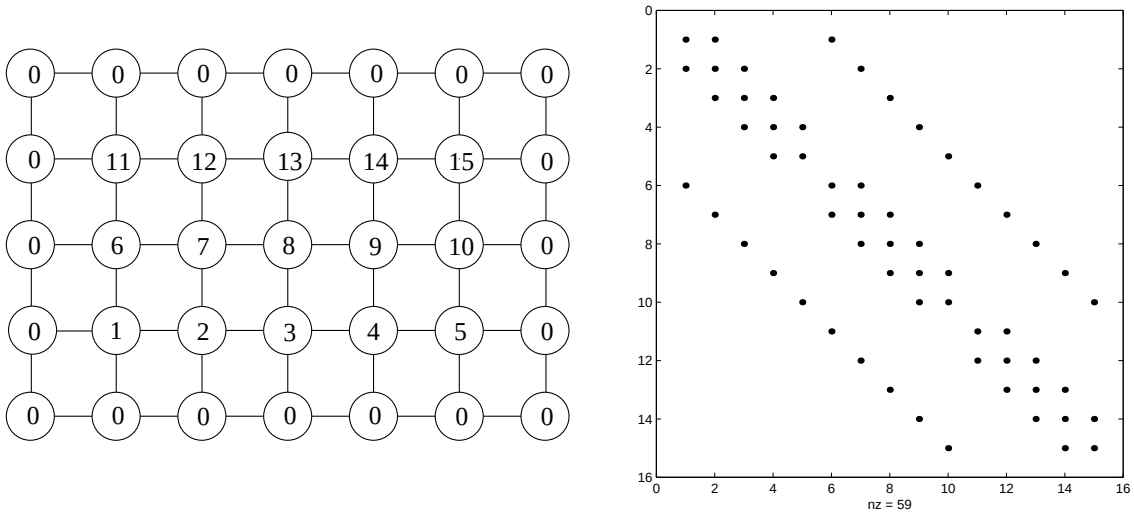


Figure 4.1: Finite Difference grid over a rectangular 2D region with zero boundary conditions and distribution of non-zero elements in the corresponding discrete 2D Laplace operator.

directional derivatives and so forth. It is worthwhile to give at this point a simple example. One may consider a problem for two-dimensional Laplace's operator given as follows:

$$\nabla^2(\cdot) = \frac{\partial^2(\cdot)}{\partial x^2} + \frac{\partial^2(\cdot)}{\partial y^2} \tag{4.1}$$

defined for a rectangular 2D region with a zero boundary condition on the edges of the region. The grid for this domain is shown in Figure 4.1. The numbers at the grid points determine the assumed ordering (usually referred to as natural ordering). (Zeros denote grid points located on the boundary of the region). The same figure shows the distribution of non-zero elements in a discrete version of the Laplace operator, given by the following formula:

$$\mathbf{L}v_{ij} = \frac{-2v_{ij} + v_{i+1,j} + v_{i-1,j}}{\Delta x^2} + \frac{-2v_{ij} + v_{i,j+1} + v_{i,j-1}}{\Delta y^2} \tag{4.2}$$

As one may note the operator matrix has a regular 5-diagonal structure which corresponds to a 5-point stencil applied to approximate second derivatives in both spatial directions using central difference discrete operators, as shown above. One should stress that also different stencils are possible for the discrete Laplace's operator based on non-collocated meshes, such as Yee's mesh.

As observed, the general feature of the matrices generated by the finite difference scheme is a highly sparse structure and a usually very regular pattern of distribution of non-zero elements. It is also characteristic that in most modern applications these matrices have very large dimensions that equal the number of sampling points (which is the product of the numbers of sampling points (of order 10^2-10^3 or more) in each of the spatial dimensions).

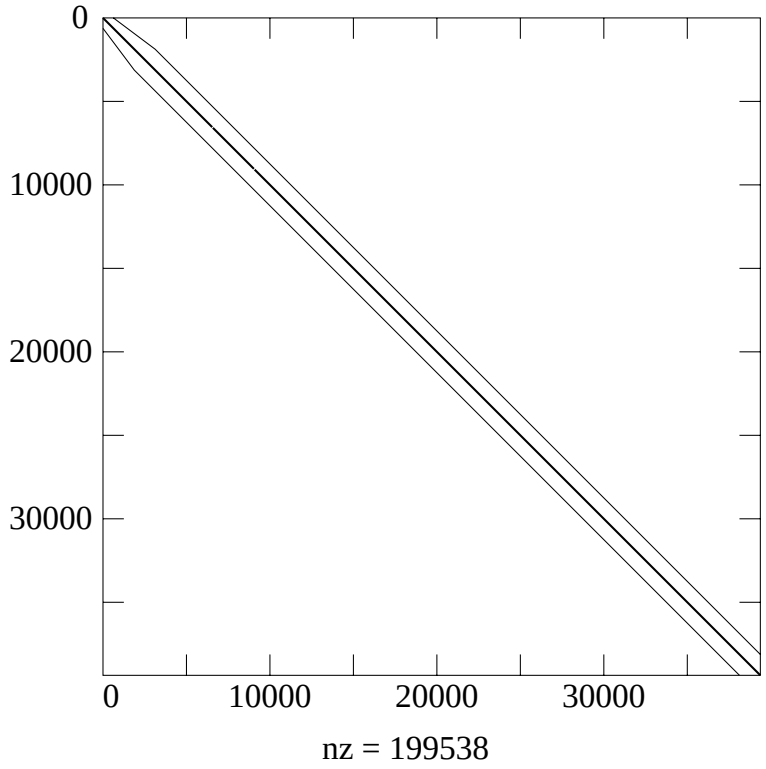


Figure 4.2: Distribution of non-zero elements in operator matrix obtained using the FD discretization.

A more advanced example of a structure of the operator matrix obtained using the FD discretization has been shown in Figure 4.2. The figure presents the distribution of non-zero elements in the matrix approximating the following second order non-symmetric differential operator (derived in Section 2.1.1) (cf. equation (2.42)):

$$\mathbf{A}v = \nabla_t^2 v + k_0^2 \epsilon(x,y) v + \frac{1}{\epsilon(x,y)} [\nabla_t \epsilon(x,y) \times (\nabla_t \times v)] \tag{4.3}$$

where $\nabla_t(\cdot) = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)(\cdot)$, $\epsilon(x,y)$ is a fixed, arbitrary function defined over 2D space and v is an appropriate two-dimensional vector field defined over a 2D spatial domain.

The matrix shown in Figure 4.2 has a dimension of approximately 40000, which corresponds to a discretization of a 2D vector field $\vec{v} = (v_x, v_y)$ over a 200×100 regular spatial grid and the number of the non-zero matrix elements equals approximately 200000. Although the matrix is non-symmetric, it has a highly regular structure with 95% of its elements located on 5 diagonals: 0 (main diagonal), +2, -2, +199, -199. These five diagonals reflect the 5-point finite difference stencils replacing the appropriate derivatives. At this point it should be noted that the bandwidth of the discussed matrix depends substantially on the ordering of the elements of vector functions, obtained from discretizing the vector field $\vec{v} = (v_x, v_y)$. With an inappropriate ordering of elements one may obtain

a matrix with a substantially increased bandwidth (or even a non-banded matrix). In our example the bandwidth equals approximately 400 and is minimal for the applied ordering, which puts first all the elements of the v_x field component before all the elements of the v_y field component. Still, in this case the bandwidth is increased if the elements of the v_x and v_y field components are mixed. It should be noted that the situation may be different if we consider a vector operator which couples H_x and H_y field components, instead of the scalar Laplace operator (in which both field components are decoupled). In such a case applying the ordering in which field components are mixed results in a matrix with substantially lower bandwidth than if the elements of v_x are all stored before the elements of v_y.

4.1.1.1 Explicit representation of the discrete operator

One may consider an explicit representation of a matrix constructed using the Finite Difference Frequency Domain (FDFD) method. As shown in the above examples the matrix of the projected operator is sparse. Consequently, it may be noted that, although the dimension of the matrix n is large, the memory requirements are not of order $O(n^2)$ but of order $O(n)$ and the matrix may be stored in one of the sparse matrix storage formats, e.g. Compressed Sparse Row (CSR) or Compressed Sparse Column (CSC) which save memory and enable efficient handling of sparse matrices using specifically designed numerical procedures (cf. the description of the SPARSKIT numerical library – [101]). In the above example the storage requirements may be further reduced if the five diagonals are stored separately and solely the irregularly located elements are stored using e.g. the CSR format.

4.1.1.2 Implicit representation of the discrete operator

Memory savings may also be achieved in a different way – by applying implicit representation of the discrete operator. One may easily note that the equal values of matrix elements associated with five-point finite difference stencils can be excluded from the stored elements and included implicitly only while calculating e.g. a matrix-vector product. The following simple example for implicit representation of the discrete 2D Laplace operator may be given. Recalling equation (4.1) and the form of the matrix of a discrete Laplace operator shown in Figure 4.1 an algorithm of computing the matrix-vector product may be easily developed. If the input vector is:

$$\underline{v} = [v_1, \ldots, v_{N_x \cdot N_y}]^T \tag{4.4}$$

where N_x and N_y equal the number of grid points in the respective spatial directions. For the case $N_x = 5$ and $N_y = 3$, the fragment of algorithm used to compute the first N_x components of the vector $\underline{w} = \underline{\underline{L}}\,\underline{v}$, where $\underline{\underline{L}}$ denotes a discrete Laplace operator, is as follows:

N_x	N_y	largest modulus eigenvalue	Number of iterations
20	10	-1.244D+01	155
40	20	-5.089D+01	271
80	40	-2.047D+02	556

Table 4.1: Eigenvalues with largest modulus computed using IRAM algorithm for the operator (4.3) discretized using FDFD technique. The results are shown for different numbers of grid points applied. The table also gives the number of iterations of the IRAM algorithm needed to obtain convergence.

ALGORITHM 4: MATRIX-VECTOR PRODUCT (FRAGMENT)

STEP 1: Do for $j = 1, N_x$:

STEP 1.1: If $j = 1$ then $w(1) = 4v(1) - v(2) - v(N_x + 1)$

STEP 1.2: else if $j = N_x$ then $w(N_x) = 4v(N_x) - v(N_x - 1) - v(2N_x)$

STEP 1.3: else $w(j) = 4v(j) - v(j - 1) - v(j + 1) - v(N_x + j)$

STEP 1.4: end if.

In the above fragment of the matrix-vector product computation algorithm the assumed spacing between the grid points equaled 1. Analogous algorithms may be developed to compute the remaining elements of vector $\underline{w}$. As one may note the above algorithm has a linear computational cost and consequently application of the above 'functional' representation of the discrete Laplace instead of the explicit matrix storage does not increase the cost of computing the matrix-vector product. At the same time the storage requirements are significantly reduced, as for the implicit representation the storage requirements are in fact constant, independent of the matrix size $N = N_x \cdot N_y$, while for the explicit representation they grow linearly with the problem size.

The above example shows the possible optimizations due to specific form of the matrix obtained in the Finite Difference discretization. — It is found that although the size of matrix is inevitably large the memory cost of storing the matrix may be reduced to even a constant value and numerical complexity of calculating the matrix-vector product may be kept linear.

Referring to the spectral radius of the discrete operator obtained during FDFD discretization, once again the example of the 2D Laplace operator will be used. If the distance between the grid points in the domain shown in Figure 4.1 equals Δ, then the function:

$$v = A \sin\left(\frac{m\pi x}{(N_x + 1)\Delta}\right) \sin\left(\frac{n\pi y}{(N_y + 1)\Delta}\right) \tag{4.5}$$

evaluated at grid points defines an eigenfunction of the discrete 2D Laplace operator satisfying the boundary condition, as shown in Figure 4.1. The corresponding eigenvalue is (cf. [50]):

$$\lambda_{mn} = -\frac{2}{\Delta^2}\left(1-\cos(m\pi/(N_x+1))\right) - \frac{2}{\Delta^2}\left(1-\cos(n\pi/(N_y+1)\right) \tag{4.6}$$

The finite difference grid defines the maximum values of m and n: $m_{\max} = N_x + 1$ and $n_{\max} = N_y + 1$. If the Finite Difference grid is sufficiently fine ($\Delta \to 0$), then the largest modulus eigenvalue (highest order eigenvalue) of the discrete Laplace operator (corresponding to $m_{\max}$ and $n_{\max}$) equals approximately:

$$\lambda_{\max} = -4\left(m_{\max}+1\right)^2 - 4\left(n_{\max}+1\right)^2 \tag{4.7}$$

If now a grid refinement is applied so that $\Delta := \Delta/2$ (and consequently N_x, N_y, $m_{\max}$ and $n_{\max}$ are doubled), then it is clear from the above formula that the modulus of the of the largest modulus eigenvalue will increase approximately four times. Consequently, the matrix spectral radius will also increase approximately four times. Table 4.1 shows eigenvalues with the largest modulus computed using IRAM algorithm for a more complicated operator, given by equation (4.3) with appropriately defined boundary conditions. The operator includes a 2D Laplace operator and has been discretized using FDFD technique as described above. The results clearly indicate that doubling the number of grid points in every spatial direction within the problem domain (equivalent to the grid refinement) results in a four fold increment in the spectral radius of the matrix. The table also gives the number of iterations of the IRAM algorithm needed to obtain convergence. One notes that the number of iterations approximately doubles while doubling the number of discretization points in every spatial direction.

4.1.2 FDFD discretization in cylindrical coordinates – 3D problems

The previous section focused on a simple FDFD technique in Cartesian coordinates which may be suitable for modeling e.g. longitudinally homogeneous waveguiding structures. This section presents a more sophisticated application of the FDFD method by describing a derivation of a discrete operator which may be used to model 3D electromagnetic systems possessing rotational symmetry (homogeneous in the ϕ direction) in a cylindrical coordinate system. In this case the $r-z$ plane is covered with the Yee mesh (cf. Figure 4.3), defining dual grids of magnetic and electric fields. The operator to be constructed will be an analogue of an infinite-dimensional operator, given by equations (A.17) or (A.18). The derivation is entirely analogous as in Appendix A, still it starts with Maxwell's equations discretized using the finite difference scheme.

In order to obtain discrete Maxwell's equations on the Yee mesh (cf. Figure 4.3) in cylindrical coordinates from the initial Maxwell's equations (A.3)-(A.9) the finite difference (central difference) scheme is applied in r and z directions and a harmonic variation

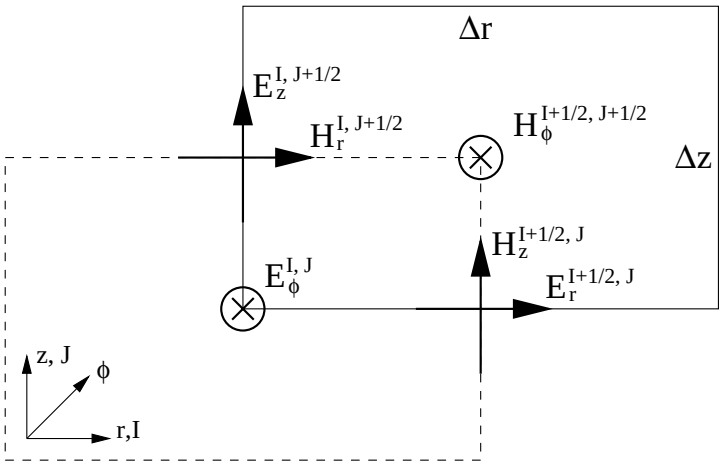


Figure 4.3: Yee's mesh in cylindrical coordinates.

on ϕ of all the field components, i.e. $\vec{E}(r,\phi,z) = \vec{E'}(r,z)\exp(jn\phi)$ and $\vec{H}(r,\phi,z) = \vec{H'}(r,z)\exp(jn\phi)$, where n is an integer number, is assumed. Then, one obtains the following set of equations for any grid cell (I,J):

$$\frac{jn}{I\Delta r}E_z^{I,J+1/2} - \frac{1}{\Delta z}\left[E_\phi^{I,J+1} - E_\phi^{I,J}\right] = j\omega\mu_0 H_r^{I,J+1/2} \tag{4.8}$$

$$\frac{1}{\Delta z}\left[E_r^{I+1/2,J+1} - E_r^{I+1/2,J}\right] - \frac{1}{\Delta r}\left[E_z^{I+1,J+1/2} - E_z^{I,J+1/2}\right] = j\omega\mu_0 H_\phi^{I+1/2,J+1/2} \tag{4.9}$$

$$\frac{1}{(I+1/2)\Delta r}\left[(I+1)E_\phi^{I+1,J} - IE_\phi^{I,J}\right] - \frac{jn}{(I+1/2)\Delta r}E_r^{I+1/2,J} = j\omega\mu_0 H_z^{I+1/2,J} \tag{4.10}$$

$$\frac{jn}{(I+1/2)\Delta r}H_z^{I+1/2,J} - \frac{1}{\Delta z}\left[H_\phi^{I+1/2,J+1/2} - H_\phi^{I+1/2,J-1/2}\right] = -j\omega\epsilon_0\epsilon_r E_r^{I+1/2,J} \tag{4.11}$$

$$\frac{1}{\Delta z}\left[H_r^{I,J+1/2} - H_r^{I,J-1/2}\right] - \frac{1}{\Delta r}\left[H_z^{I+1/2,J} - H_z^{I-1/2,J}\right] = -j\omega\epsilon_0\epsilon_\phi E_\phi^{I,J} \tag{4.12}$$

$$\begin{aligned}\frac{1}{I\Delta r}\Big[(I+1/2)H_\phi^{I+1/2,J+1/2} &- \\ -(I-1/2)H_\phi^{I-1/2,J+1/2}\Big] - \frac{jn}{I\Delta r}H_r^{I,J+1/2} &= -j\omega\epsilon_0\epsilon_z E_z^{I,J+1/2}\end{aligned} \tag{4.13}$$

$$\begin{aligned}\frac{1}{I\Delta r}\left[(I+1/2)D_r^{I+1/2,J} - (I-1/2)D_r^{I-1/2,J}\right] &+ \\ +\frac{jn}{I\Delta r}D_\phi^{I,J} + \frac{1}{\Delta z}\left[D_z^{I,J+1/2} - D_z^{I,J-1/2}\right] &= 0\end{aligned} \tag{4.14}$$

where Δr and Δz are FD grid sizes in r and z directions, respectively. In the above formulae it is assumed that the permittivity tensor has a diagonal form: $\overleftrightarrow{\epsilon} = diag(\epsilon_r, \epsilon_\phi, \epsilon_z)$, where ϵ_r, ϵ_ϕ and ϵ_z vary only in the $r-z$ plane. Also, for simplicity, the permeability is assumed to be constant in the entire domain and equals μ_0. It is worthwhile to note that the equations (4.14)–(4.10) may also be obtained by discretizing Maxwell's equations in the integral form applying the Finite Integration Technique (FIT) on the Yee mesh (cf. Figure 4.3) [66].

Denoting as $\underline{D}^r$, $\underline{D}^\phi$, $\underline{D}^z$, $\underline{E}^r$, $\underline{E}^\phi$, $\underline{E}^z$, $\underline{H}^r$, $\underline{H}^\phi$, $\underline{H}^z$ the vectors containing values of respective fields at the appropriate grid points one may write equations (4.14)–(4.10) for the entire discretized 2D domain using the following matrix notation:

$$\underline{D}^\phi = \underline{\underline{D\phi divDr}}\,\underline{D}^r + \underline{\underline{D\phi divDz}}\,\underline{D}^z \tag{4.15}$$

$$\omega\underline{\epsilon_r}\,\underline{E}^r = \underline{\underline{ErrotHz}}\,\underline{H}^z + \underline{\underline{ErrotH\phi}}\,\underline{H}^\phi \tag{4.16}$$

$$\omega\underline{\epsilon_\phi}\,\underline{E}^\phi = \underline{\underline{E\phi rotHr}}\,\underline{H}^r + \underline{\underline{E\phi rotHz}}\,\underline{H}^z \tag{4.17}$$

$$\omega\underline{\epsilon_z}\,\underline{E}^z = \underline{\underline{EzrotH\phi}}\,\underline{H}^\phi + \underline{\underline{EzrotHr}}\,\underline{H}^r \tag{4.18}$$

$$\omega\mu_0\underline{H}^r = \underline{\underline{HrrotEz}}\,\underline{E}^z + \underline{\underline{HrrotE\phi}}\,\underline{E}^\phi \tag{4.19}$$

$$\omega\mu_0\underline{H}^\phi = \underline{\underline{H\phi rotEr}}\,\underline{E}^r + \underline{\underline{HfrotEz}}\,\underline{E}^z \tag{4.20}$$

$$\omega\mu_0\underline{H}^z = \underline{\underline{HzrotE\phi}}\,\underline{E}^\phi + \underline{\underline{HzrotEr}}\,\underline{E}^r \tag{4.21}$$

where $\underline{\epsilon_r}$, $\underline{\epsilon_\phi}$ and $\underline{\epsilon_z}$ are vectors containing the values of the elements of the permittivity tensor at the subsequent grid points. (In fact those vectors may contain values of effective permittivity for the cells of the defined Yee mesh as to improve the model of geometry of the investigated boundary value problem. [26], [27])

Using equation (4.15) one may derive a formula for $\underline{E}^\phi$ and substitute it into equations (4.19) and (4.21). Then, equations (4.19)–(4.21) may be used to derive formulae for $\underline{H}^r$, $\underline{H}^\phi$ and $\underline{H}^z$ which may be substituted to equations (4.16)–(4.18). Eventually one gets:

$$\begin{aligned}\omega^2\underline{D}^r &= \underline{\underline{ErrotHz}}\,(\underline{\underline{HzrotE\phi}}\,\underline{\epsilon_\phi^{-1}}\,(\underline{\underline{D\phi divDr}}\,\underline{D}^r + \underline{\underline{D\phi divDz}}\,\underline{D}^z)\\ &+ \underline{\underline{HzrotEr}}\,\underline{\epsilon_r^{-1}}\,\underline{D}^r)\\ &+ \underline{\underline{ErrotH\phi}}\,(\underline{\underline{H\phi rotEr}}\,\underline{\epsilon_r^{-1}}\,\underline{D}^r + \underline{\underline{H\phi rotEz}}\,\underline{\epsilon_z^{-1}}\,\underline{D}^z)\end{aligned} \tag{4.22}$$

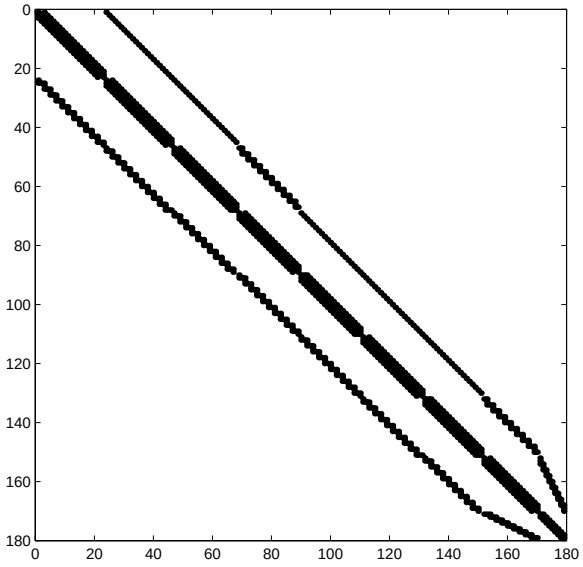


Figure 4.4: Sample matrix of the operator $\underline{\underline{S}}$ – a discrete mapping of differential operator **S**, given by equation (2.52).

$$\begin{aligned}
\omega^2 \underline{D}^z &= \underline{\underline{EzrotHr}}\,(\underline{\underline{HrrotE\phi}}\,\underline{\epsilon_\phi^{-1}}\,(\underline{\underline{D\phi divDr}}\,\underline{D}^r + \underline{\underline{D\phi divDz}}\,\underline{D}^z) \\
&+ \underline{\underline{HrrotEz}}\,\underline{\epsilon_z^{-1}}\,\underline{D}^z) \\
&+ \underline{\underline{EzrotH\phi}}\,(\underline{\underline{H\phi rotEr}}\,\underline{\epsilon_r^{-1}}\,\underline{D}^r + \underline{\underline{H\phi rotEz}}\,\underline{\epsilon_z^{-1}}\,\underline{D}^z)
\end{aligned} \tag{4.23}$$

The above two equations may be cast in the following compact form:

$$\omega^2 \underline{D} = \underline{\underline{S}}\,\underline{D} \tag{4.24}$$

where $\underline{D} = [\underline{D}^r \& \underline{D}^z]^T \equiv [\underline{D}^r_1, \underline{D}^z_1, \ldots, \underline{D}^r_N, \underline{D}^z_N]$ and $\underline{\underline{S}}$ is an operator defined by the right hand side of equations (4.22) and (4.23). Discrete operator $\underline{\underline{S}}$ is a finite-dimensional projection of operator **S** given by equation (2.52). A sample form of the matrix of operator $\underline{\underline{S}}$ is given in Figure 4.4 which shows non-zero elements of the matrix.

The matrix has been obtained by mapping the operator **S**, given by equation (2.52) for certain boundary conditions (discussed in Chapter 6). One may note that the operator matrix does not have an ideally regular structure. A tapered end appears in the pattern of non-zero element distribution. This is due to the applied boundary conditions, and is explained in Figure 4.5. The figure shows two domains: a regular rectangular domain and a domain with a semicircular boundary (marked with a dashed line). Outside these domains the values of functions are imposed to equal zero. If, once again, the example of

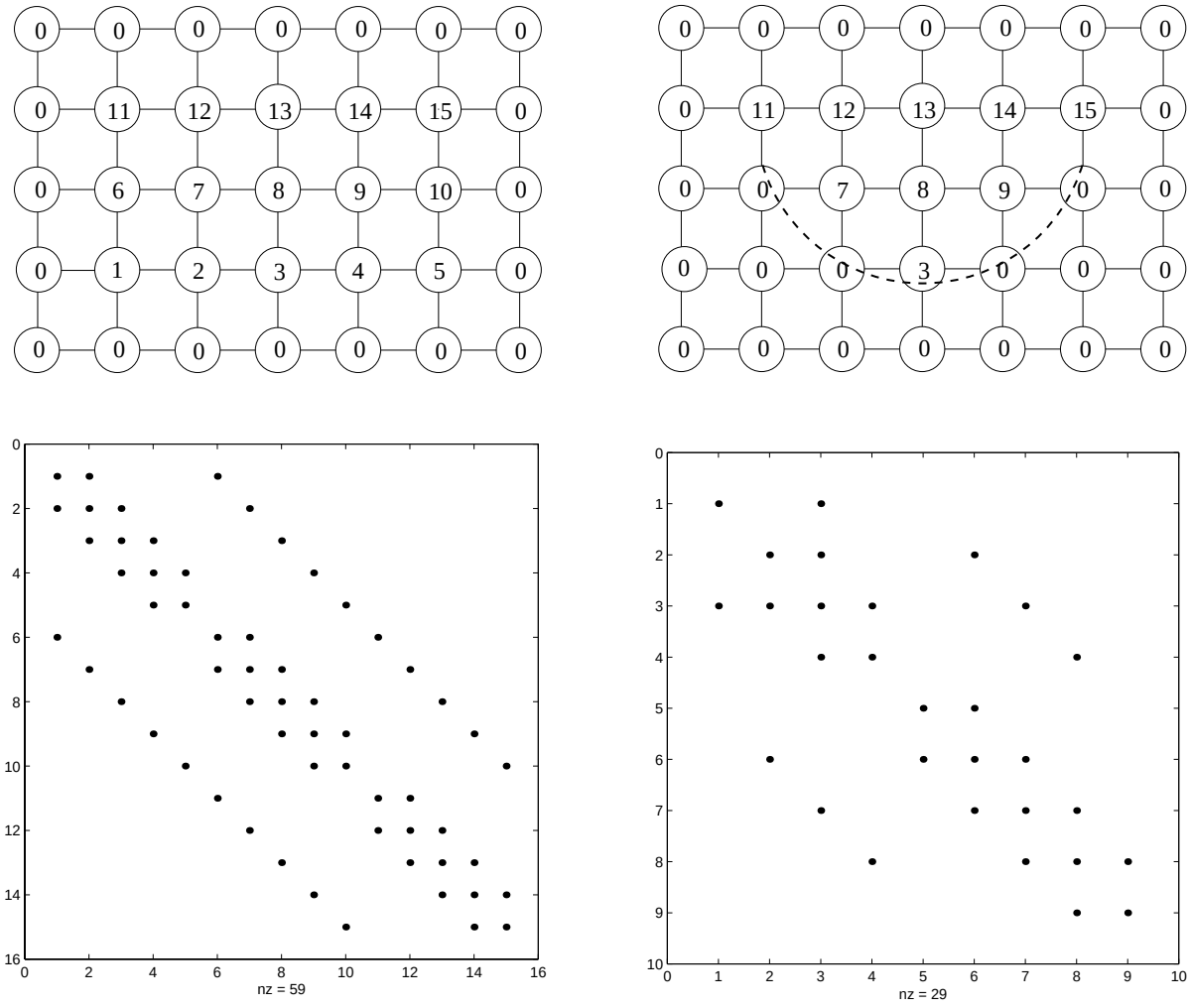


Figure 4.5: Forms of a discrete 2D Laplace operator with different boundary conditions applied. The left pictures show the case of a rectangular domain, while the right one – the case of of a domain with semicircular boundary.

a 2D Laplace operator is considered, then the forms of discrete, projected operators with boundary conditions implemented for the respective domains are also shown in Figure 4.5. In fact the matrix on the right hand side is formed by removing columns and rows with indices 1, 2, 4, 5, 6 and 10 from the matrix on the left hand side. This is done as it is unnecessary to represent the grid points at which the zero boundary conditions are assumed.[1] One may note that even for this extremely coarse grid a tapered end appears in the matrix shown on the lower-right picture of Figure 4.5.

[1]Still, under certain circumstances, discussed in Section 5.3.2, it is advantageous to include the zeroed grid points in the domain representation in order to preserve a more regular matrix structure.

$\Delta = \Delta z = \Delta r$ [mm]	Spectral radius	Matrix size N
$6 \cdot 0.237$	$9.175 \cdot 10^{23}$	2146
$3 \cdot 0.237$	$2.515 \cdot 10^{24}$	8322
$1 \cdot 0.237$	$1.652 \cdot 10^{25}$	73780

Table 4.2: The spectral radii computed for the $\underline{\underline{S}}$ operator using IRAM algorithm. During computations $n = 1$ (in $\exp(jn\phi)$ term).

n ($\exp(jn\phi)$)	Computed $\omega^2_{\max}$	Estimation for $\omega^2_{\max}$
128	$2.9176 \cdot 10^{27}$	$2.9171 \cdot 10^{27}$
256	$1.1668 \cdot 10^{28}$	$1.1668 \cdot 10^{28}$
512	$4.6671 \cdot 10^{28}$	$4.6671 \cdot 10^{28}$
1024	$1.8668 \cdot 10^{29}$	$1.8668 \cdot 10^{29}$

Table 4.3: The comparison of estimated and computed spectral radii for the $\underline{\underline{S}}$. It has been assumed that $\Delta r = \Delta z = \Delta = 6 \cdot 0.237$ mm.

Referring to the numerical costs associated with applying the presented FDFD mapping technique the situation is similar as in the previous section. Due to a highly sparse character of the operator matrix the memory cost of storing explicitly the matrix is linear ($O(N)$). For the matrix shown in Figure 4.4 the storage requirements equal approximately $11N$, where N is the matrix size. If the operator matrix is represented implicitly, then this complexity may be reduced to a constant complexity. This estimations do not take into account the cost of storing the elements of the permittivity tensor $\overleftrightarrow{\epsilon}$. If the values of permittivity tensor may be computed rather than stored, then the overall matrix storage cost remains constant. Otherwise, it becomes linear and equals approximately $3N$. The cost of computing the matrix-vector product for both implicit and explicit representations is linear.

Another important issue which has to be addressed at this point is the spectral radius of the discrete operator obtained by applying the discussed FDFD projection procedure over the Yee mesh in cylindrical coordinates. The following discussion will focus solely on the properties of the discrete operator $\underline{\underline{S}}$ derived above. It is expected that, analogously as in the previous section, the spectral radius increases while refining the grid in the $r-z$ plane. This is verified by the numerical results shown in Table 4.2. One may note that also in this case the spectral radius is inversely proportional to Δ^2.

Additionally, the spectral radius (in physical terms the squared modal angular frequency) should depend on the value of integer n, associated with the term $\exp(jn\phi)$ and determining the azimuthal variation of the modal fields. The following estimation

has been developed by Dehler [36] by using eigenvalue localization methods (Gershgorin theorem):

$$\omega_{\max}^2 \leq c^2 \frac{4n_{\max}^2 + 4}{\Delta^2} \tag{4.25}$$

where $n_{\max}$ is a maximum possible value of n from the $\exp(jn\phi)$ term and Δ determines the grid size. The estimation is valid for $n_{\max} >> 1$. In the case of operator $\underline{\underline{S}}$ one has $n_{\max} = n$, as n is a pre-determined, fixed number. One may note that the term:

$$\frac{4n_{\max}^2}{\Delta^2}$$

relates to the squared wavenumber in the ϕ direction for a wave described by the $\exp(jn\phi)$ term, propagating along the cylindrical surface of the smallest cylinder defined by the Yee mesh, having the radius $R = \Delta/2$. Comparison of the estimations to the computed values of the eigenvalues with the largest modulus are presented in Table 4.3. The results imply that for larger values of n the inequality (4.25) gives correct and accurate estimations for the spectral radius.

It is worthwhile to note that estimation (4.25) may also be used to bound the spectral radius for the case, when the discretization is applied not only in the $r - z$ plane, but in all the three spatial dimensions. In this case the value of $n_{\max}$ equals:

$$n_{\max} = N_\phi/2 \tag{4.26}$$

where N_ϕ is the number of discretization points in the ϕ direction. (It is assumed that N_ϕ is even.) The discussed estimation is also valid for the method described in Section 4.3.1, which uses eigenfunction expansion representation in the ϕ direction.

Concluding the discussion, it may be pointed out that the FDFD technique results in discrete operators which may be represented as highly regular, banded matrices, that in turn may be easily represented implicitly. Moreover, computing the matrix-vector product is straightforward and is associated with only a linear cost. As shown in Chapter 5 these properties allow efficient parallel implementation of the problems associated with this class of operators. Unfortunately, application of FDFD method has one important disadvantage, which is appearance of numerical dispersion. While the effects of numerical dispersion can be neglected in most small scale or medium scale problems, they become very pronounced in large scale numerical modeling. This issue is addressed in detail in the following section.

4.1.3 Reduction of numerical dispersion in finite difference methods

This section discusses the problem of numerical dispersion which inevitably arises while applying the finite difference approximation to differential operators. In general, the

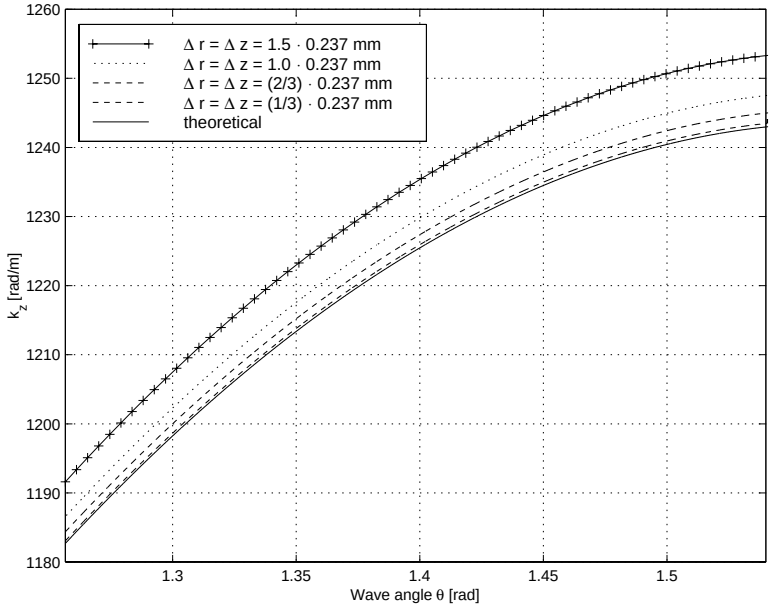


Figure 4.6: Reducing the effect of numerical dispersion. The error in the values of wavenumber k_z may be lowered by refining the FDFD discretization grid.

numerical dispersion is attributed to the fact that numerically modeled electromagnetic waves propagate not in a continuous (possibly homogeneous) space but in a finite difference grid, which may be compared to a crystal lattice. In this context the numerical 'free space' becomes an anisotropic medium. Consequently, the following important effects are observed. Most importantly, for the simulated wave modes, being eigenfunctions of e.g. operator $\underline{\underline{S}}$ (cf. previous section, equation (4.24)), propagating in an FDFD grid it is observed that their phase velocity differs from the speed of light c in the vacuum and varies with the wavelength, the direction of propagation with respect to the grid and the grid discretization. The numerical errors, e.g. the phase velocity error, accumulate along the propagation direction. In other words, the longer is the distance traveled by a propagating wave, the larger are the errors due to numerical dispersion. This explains why the problem of numerical dispersion is most visible in large scale electromagnetic systems whose geometrical dimensions are by tens larger than the wavelength of the propagating modes. The further consequences of numerical dispersion include significant errors in computed propagation constants or resonant frequencies of modes in resonators.

The most common way to overcome the effect of numerical dispersion is to refine the discretization grid. Figure 4.6 illustrates such approach. The Figure shows values of wavenumbers (propagation constants) k_z computed for different directions of wave propagation θ (which is the angle between the wavevector and the $r - \phi$ plane) and different refinements of the Yee mesh applied. The computed wavenumbers are compared

to the theoretical values of k_z. The waves propagate in a hollow cylindrical region. The values of $\Delta r = \Delta z$ vary from about $\lambda/14 = 1.5 \cdot 0.237$ mm (coarse grid) to $\lambda/64 = (1/3) \cdot 0.237$ mm (fine grid), where λ is the wavelength of the propagating wave. For the coarse grid the relative error of computing k_z equals about 1%. As one notes this error is very much reduced for the fine grid.

Still, this method of reducing the effects of numerical dispersion is an expensive one. The technique of grid refinement has a disadvantage of increasing the size of the solved matrix problem and in consequence increasing memory and computational cost of solving the eigenproblem. For instance, if a 2D domain having dimensions of 20λ in each direction is considered, where λ is the wavelength of the propagating wave, then applying a grid spacing of $\lambda/10$ in each direction yields a matrix problem size of about 80000 for the formulation (4.24). Refining a grid so that the cell spacing becomes $\lambda/20$ results in a matrix of the size $N = 320000$, with four times larger bandwidth and approximately four times larger spectral radius (compare the discussion in the previous section). This in turn causes a significant increment in solution time of the resulting matrix eigenproblem.

Consequently, a different method has to be applied to reduce the effects of numerical dispersion if excessive memory and computational costs are to be avoided while modeling large scale problems. The approach towards reducing errors due to numerical dispersion proposed below does not change the size of the problem and consequently does not increase memory and computational cost associated with the solution of the eigenproblem. Therefore, it appears to be well suited to be used together with FDFD technique to model complicated large scale electromagnetic problems.

The proposed method of reducing the effect of numerical dispersion is based on introducing a correction into the discrete operator equations. The correction annihilates or reduces the error associated with numerical dispersion for a certain range of solutions. In general, the idea is based on replacing the standard finite difference operator, e.g. the central difference operator:

$$\mathbf{D}F(x) = \frac{F(x + \Delta x/2) - F(x - \Delta x/2)}{\Delta x} \tag{4.27}$$

with a modified operator:

$$\mathbf{D}F(x) = A\frac{F(x + \Delta x/2) - F(x - \Delta x/2)}{\Delta x} \tag{4.28}$$

where $A(\neq 1)$ is a certain constant number. In this way a modified discrete operator with corrected eigenvalues is constructed. A broad discussion of this approach for the Finite Difference Time Domain (FDTD) method applied to solving Maxwell's equations in rectangular coordinate system may be found in a book by Taflove [110]. Somewhat different approaches towards reducing numerical dispersion, also referring to rectangular coordinate system and FDTD method, may be found in [52] and [83].

The sections below present an originally developed technique of reducing the effect of numerical dispersion for the Finite Difference Frequency Domain (FDFD) discretization method in cylindrical coordinates. This technique provides a way to compute the value of constant A from equation (4.28) in the case of modes propagating in homogeneous 3D regions. It is assumed that the FD discretization has been performed in the r and z directions only using a standard Yee's mesh (cf. previous section).

If a 3D region whose boundaries coincide with cylindrical coordinate surfaces is considered, then the electromagnetic fields may be found analytically using a scalar Helmholtz equation in cylindrical coordinates:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \psi}{\partial r}\right)+\frac{1}{r^2}\frac{\partial^2 \psi}{\partial \phi^2}+\frac{\partial^2 \psi}{\partial z^2}+k^2\psi=0 \tag{4.29}$$

The class of solutions of the above equation to be considered takes the following form:

$$\psi(r,\phi,z)=B_n(k_r r)\exp(jn\phi+k_z z) \tag{4.30}$$

where $B_n(\cdot)$ is the n-th Bessel (or Hankel) function of the first or the second kind, n is an integer number, and the following relation is satisfied:

$$k^2=k_z^2+k_r^2 \tag{4.31}$$

where k_z and k_r denote the components of the free-space wavenumber k in the z and r directions, respectively (cf. Figure 4.7).

Modes in cylindrical coordinates may be described by means of the TE or TM potentials (cf. Rozzi and Mongiardo [98]) which give the following expressions for TM and TE fields, respectively:

$$\begin{array}{ll} E_r=-j\omega\mu\frac{1}{r}\frac{\partial\psi}{\partial\phi} & H_r=\frac{\partial^2\psi}{\partial r\partial z} \\ E_\phi=j\omega\mu\frac{\partial\psi}{\partial r} & H_\phi=\frac{1}{r}\frac{\partial^2\psi}{\partial\phi\partial z} \\ E_z=0 & H_z=\left(k^2+\frac{\partial^2}{\partial z^2}\right)\psi \end{array} \tag{4.32}$$

and:

$$\begin{array}{ll} E_r=\frac{\partial^2\psi}{\partial r\partial z} & H_r=j\omega\epsilon\frac{1}{r}\frac{\partial\psi}{\partial\phi} \\ E_\phi=\frac{1}{r}\frac{\partial^2\psi}{\partial\phi\partial z} & H_\phi=-j\omega\epsilon\frac{\partial\psi}{\partial r} \\ E_z=\left(k^2+\frac{\partial^2}{\partial z^2}\right)\psi & H_z=0 \end{array} \tag{4.33}$$

Consequently, the fields may be represented in the following form (TM case):

$$E_r=E_{r0}jB_n'(k_r r)\exp(jn\phi+k_z z) \tag{4.34}$$

$$E_\phi=E_{\phi 0}\frac{1}{r}B_n(k_r r)\exp(jn\phi+k_z z) \tag{4.35}$$

$$E_z = E_{z0} B_n(k_r r) \exp(jn\phi + k_z z) \tag{4.36}$$

$$H_r = H_{r0} \frac{1}{r} B_n(k_r r) \exp(jn\phi + k_z z) \tag{4.37}$$

$$H_\phi = H_{\phi 0} j B'_n(k_r r) \exp(jn\phi + k_z z) \tag{4.38}$$

$$H_z = 0 \tag{4.39}$$

and (TE case):

$$H_r = H_{r0} j B'_n(k_r r) \exp(jn\phi + k_z z) \tag{4.40}$$

$$H_\phi = H_{\phi 0} \frac{1}{r} B_n(k_r r) \exp(jn\phi + k_z z) \tag{4.41}$$

$$H_z = H_{z0} B_n(k_r r) \exp(jn\phi + k_z z) \tag{4.42}$$

$$E_r = E_{r0} \frac{1}{r} B_n(k_r r) \exp(jn\phi + k_z z) \tag{4.43}$$

$$E_\phi = E_{\phi 0} j B'_n(k_r r) \exp(jn\phi + k_z z) \tag{4.44}$$

$$E_z = 0 \tag{4.45}$$

If the TE case is considered, the formulae (4.40)–(4.45) may be substituted to discrete Maxwell's curl equations for cylindrical coordinates (4.11)-(4.13), (4.9) and (4.10) in which the standard central difference operators (4.27) are replaced with modified operators (4.28). The substitution yields the following set of equations:

$$\underline{\underline{W}}\,\underline{\Psi} = 0 \tag{4.46}$$

where $\underline{\Psi} = \begin{bmatrix} H_{r0} & H_{\phi 0} & H_{z0} & E_{r0} & E_{\phi 0} \end{bmatrix}^T$ is a vector of the electromagnetic field amplitudes and $\underline{\underline{W}}$ is a 5×5 matrix. If the above set of equations is about to have a non-zero solution, then matrix $\underline{\underline{W}}$ has to be singular:

$$\det(\underline{\underline{W}}) = 0$$

The explicit form of the above equation is presented below:

$$\begin{vmatrix} -\omega\mu & 0 & 0 & 0 & \frac{2A\sin(k_z\Delta z/2)}{\Delta z} \\ 0 & \omega\mu & 0 & \frac{2A\sin(k_z\Delta z/2)}{\Delta z} & 0 \\ 0 & 0 & \omega\mu B_n(\alpha) & \frac{-n}{((I+1/2)\Delta r)^2} \times B_n(\alpha) & \frac{A}{(I+1/2)\Delta r}\times [B_n'(\gamma)(I+1) - B_n'(\delta)I] \\ 0 & -\frac{2A\sin(k_z\Delta z/2)}{\Delta z} & n & -\omega\epsilon & 0 \\ -\frac{2A\sin(k_z\Delta z/2)}{\Delta z} \times B_n'(k_r I\Delta r) & 0 & -\frac{B_n(\alpha)}{\Delta r}A + \frac{B_n(\beta)}{\Delta r}A & 0 & \omega\epsilon B'(k_r I\Delta r) \end{vmatrix} = 0 \quad (4.47)$$

where

$$\alpha = k_r(I+1/2)\Delta r \quad \beta = k_r(I-1/2)\Delta r$$
$$\gamma = k_r(I+1)\Delta r \quad \delta = k_r I\Delta r$$

An analogous relation may be developed for a TM field.

It is important to note that if $\Delta r \to 0$ and $\Delta z \to 0$ (and imposing that $(I+1/2)\Delta r = r$) the discrete dispersion relation (4.47) reduces to the Bessel equation:

$$r\frac{d}{dr}\left(r\frac{dB_n(k_r r)}{dr}\right) + r^2(k^2 - k_z^2)B_n(k_r r) - n^2 B_n(k_r r) = 0 \quad (4.48)$$

which is satisfied if and only if:

$$k^2 - k_z^2 = k_r^2 \quad (4.49)$$

Equation (4.47) may therefore be used to establish relation between k, k_z and k_r in the case of a discretized problem domain. If the values of $k = \omega/c$, k_r, I, n, Δr, Δz and A are fixed then one may compute the value of k_z using (4.47). (Introducing $q = \sin(k_z\Delta z/2)$, equation (4.47) becomes a biquadratic equation with an unknown q. If the solutions are limited to k_z ranging from 0 to $\pi/4\Delta z$ then only two of the four solutions q of the biquadratic equation are significant.) The computed values of k_z may then be compared to the theoretical value of k_{zt} computed using the 'continuous' dispersion relation (4.49):

$$k_{zt} = \sqrt{k^2 - k_r^2} \quad (4.50)$$

Consequently, for given wavenumbers k and k_r one may optimize the value of A as to minimize the difference between k_z and k_{zt}, computed using equations (4.47) and (4.50) respectively. The value of A can be optimized for a single value of k_r which is equivalent to minimizing the function:

$$f(A) = |k_{zt}(k_r) - k_z(k_r, A)| \quad (4.51)$$

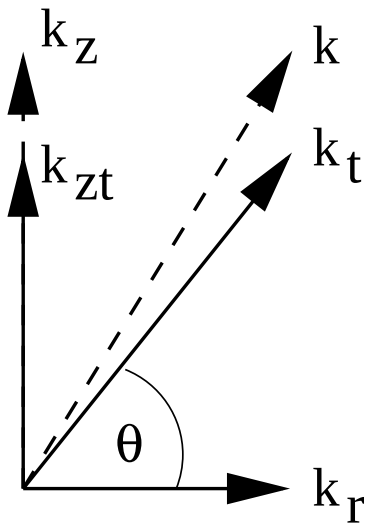


Figure 4.7: Relation between the wavenumbers. Note that $|k| > |k_t|$ due to numerical dispersion.

or can be optimized for a certain range of wavenumbers $k_r = k\cos(\theta)$ (cf. Figure 4.7) which is equivalent to minimizing the following function:

$$g(A) = \int_{\theta_0-\Delta\theta}^{\theta_0+\Delta\theta} |k_{zt}(k\cos(\theta)) - k_z(k\cos(\theta), A)|\, d\theta \tag{4.52}$$

The minimization of either $f(A)$ or $g(A)$ may be performed by applying any from a wide spectrum of numerical recipes. In the case of computations performed within this study, a simple 'fmin' MATLAB function [70] for finding local minima of a user defined single-variable function was applied. This routine was able to find optimized values of A at virtually no numerical cost.

Before giving an example of application of the proposed technique, the following observation has to be made. The main difference between the theoretical dispersion relation (4.49) and the discrete dispersion relation (4.47) is that satisfying the first relation ensures that the fields given by formulae (4.40)-(4.45) are solutions of our problem for the entire domain, i.e. for any values or r, ϕ and z, while satisfying the second relation determines solutions for a single value of I, i.e. for a single cylindrical surface $I\Delta = const.$. Nevertheless, as shown in Chapter 6 containing numerical results for the discussed algorithm applied to the problem of modeling a large hemispherical resonator, for a wide range of values of I (referring to surfaces with radii ranging from $0.1R$ to R, where R is the radius of the resonator), the computed values of A do not differ significantly and in any case the effect of numerical dispersion is reduced. This means that the choice of I is of secondary importance.

To illustrate the proposed technique we applied it to find optimized value of the A in formula (4.28). The following set of input parameters has been used: $\epsilon = \epsilon_0$, $\mu = \mu_0$

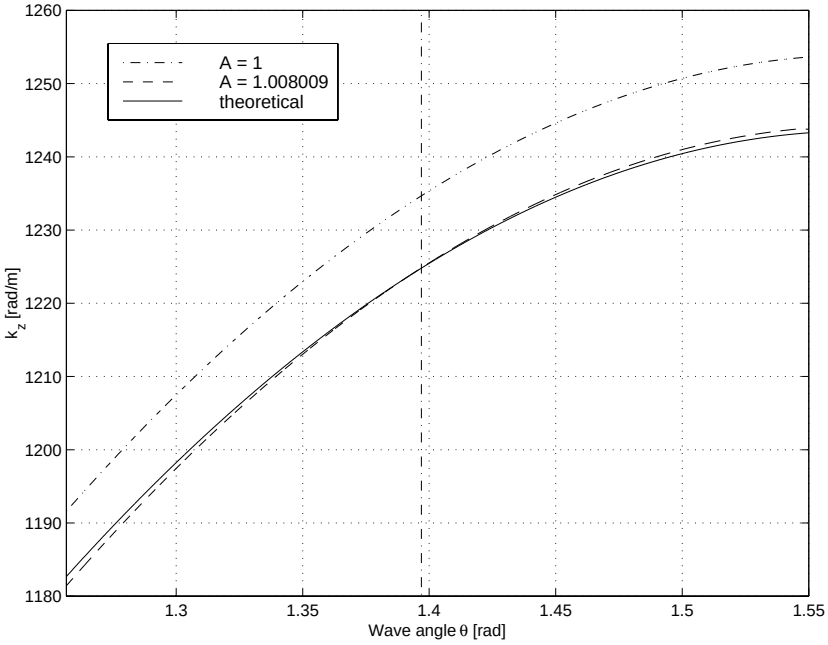


Figure 4.8: Wavenumber k_z computed for different values of $k_r = k\cos(\theta)$ and two different values of A. The vertical line indicates the angle between the wavevectors k and k_r for which the value of A has been optimized.

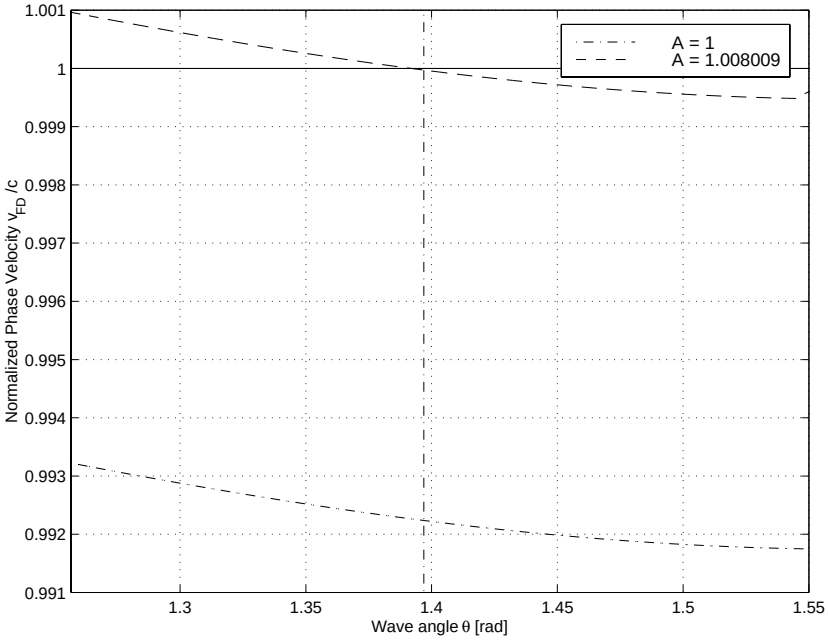


Figure 4.9: Variation of the numerical phase velocity with the wave propagation angle for the optimized and non-optimized values of A.

(a hollow cylindrical cavity), $\omega = 2\pi f$, where $f = 59.375168$ GHz, $\Delta R = \Delta z = 1.5 \cdot 0.237$ mm, $n = 1$. The A factor has been optimized for the value of the wavenumber $k_r = k \cdot \cos(0.889 \cdot \pi/2) = \sqrt{k^2 - k_z^2}$, where $k = \omega/c = 2\pi f/c$ and $k_z = 2\pi N/L = 2\pi \cdot 10$ rad/(51.3 mm), where L is the length of the cavity and the value of $I = 113$, which corresponds to the cavity radius $R = 40.17$ mm.

For the listed parameters the optimized value of A equals 1.008009. Figure 4.8 shows the values of the wavenumber k_z, computed using the 'continuous' dispersion relation (4.49) and the discrete dispersion relation (4.47) for the non-optimized and optimized value of A. It is apparent that if $A = 1$ in the finite difference operator then values of k_z are significantly larger from the theoretical values k_{zt}. This, in turn, causes the numerical modes to propagate with phase velocity which is always less than c, as shown in Figure 4.9. For the optimized value of A the error in computed wavenumbers k_z is significantly reduced for the shown range of propagation angles (and annihilated for the $k_r = k \cdot \cos(0.889 \cdot \pi/2)$). Consequently, also the error in phase velocity is very significantly diminished. Clearly the effect of numerical phase velocity anisotropy, inherent to Yee's algorithm cannot altogether be eliminated using this approach.

It has been shown above that the proposed technique introduces a correction to the discrete dispersion relation which allows a better approximation of the theoretical 'continuous' dispersion relation. The following section presents application of this technique to modeling an electrically large hollow cylindrical resonator with a Finite Difference Frequency Domain (FDFD) method.

4.1.4 Iterative scheme for reducing dispersion error in a cylindrical resonator: a numerical example

The aim of this series of numerical tests was to assess the applicability of the technique of reducing numerical dispersion presented above and test an iterative scheme for reducing numerical dispersion error (presented below). The modeled structure was a simple hollow cylindrical resonator shown in Figure 4.10.

During the tests the resonant frequencies of the TE_{116} mode have been computed. The theoretical frequency for this mode is: $f_t = 17.6803011514791$ GHz. This refers to $\lambda \approx 17$ mm. One may note, that the dimensions of the resonator equal approximately $2.4\lambda \times 3\lambda$. Figure 4.11 shows the field plot of the E_r component computed using the IRAM-FDFD algorithm (with implicit matrix representation - cf. Section 4.1.1.2) with the discretization grid: $\Delta r = \Delta z = 1.442$ mm $= \lambda/12$. For this discretization grid the computed numerical resonant frequency equals $f = 17.48441386145861$ GHz. Consequently, the relative error in computing the resonant frequency is significant and equals 1.11%. If now a correction of the finite difference scheme, discussed in Section 4.1.3 is applied this error may be reduced. The value of A factor (cf. equation (4.28)) optimized for the theoretical frequency f_t and the applied discretization grid equals: $A = 1.01133318543985$. Introducing this modified value of A into the constructed discrete operator yields a new

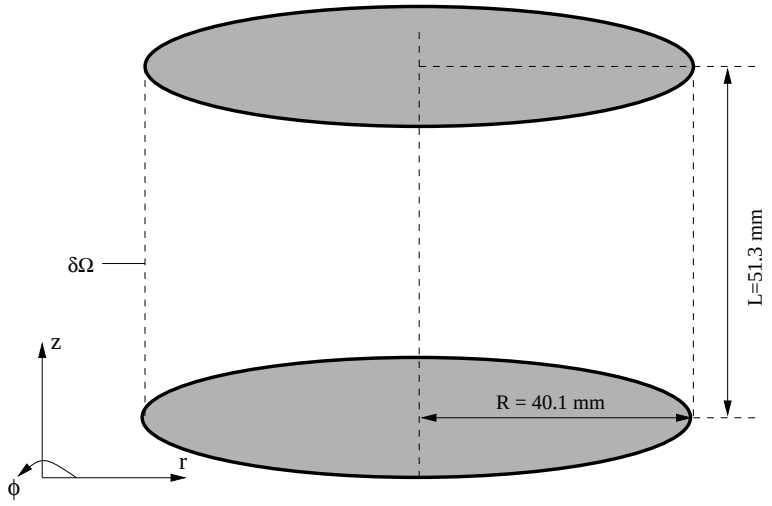


Figure 4.10: Schematic of a hollow cylindrical resonator. At the boundary $\delta\Omega$ the Perfect Electric Conductor (PEC) condition has been applied.

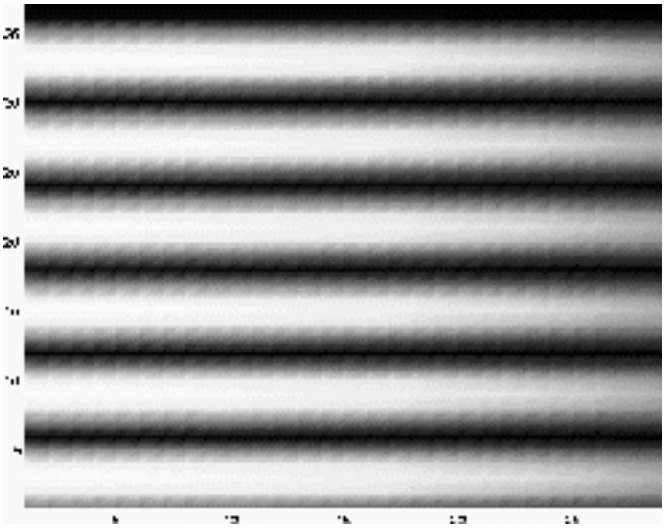


Figure 4.11: Rms E_r field plot for the TE_{116} mode in a homogeneous cylindrical resonator (cf. Figure 4.10). The discretization grid applied: $\Delta r = \Delta z = 1.422$ mm $= \lambda/12$.

matrix eigenproblem. The corrected resonant frequency computed as an eigenvalue of the discussed matrix equals $\tilde{f} = 17.68041080944422$ GHz. Now, the relative error between $\tilde{f}$ and f_t equals 0.0006% instead of more than 1%!.

This spectacular error reduction has been obtained for the value of A optimized exactly for the theoretical frequency which is known in this case. Clearly, in all practical situations situation the resonant frequency to be found is (by definition) unknown. Nevertheless, the correction technique may be still applied. Table 4.4 shows a number of computed

Optimization frequency [GHz]	Value of A factor	Computed frequency [GHz]	Relative error [%]
-	1.00000000000000	17.48441386145861	-1.1079
17.2382939769208	1.01106733874854	17.67581308125628	-0.0254
17.3382939769208	1.01119712304638	17.67805767786787	-0.0127
17.4382939769208	1.01132768160268	17.68031564133214	0.0001
17.5382939769208	1.01145901482931	17.68258697358160	0.0129
17.6382939769208	1.01139396076993	17.68146186865034	0.0066
17.6803011514791	1.01133318543985	17.68041080944422	0.0006

Table 4.4: The table shows values of coefficients A optimized for different frequencies close to the theoretical resonant frequency $f_t = 17.68030115147906$ GHz of the TE_{116} mode and respective computed resonant frequencies for the same mode. The first row shows the frequency computed for non-optimized value of A. The last row shows the value of A optimized for the theoretical resonant frequency. The last column shows the relative error between computed and theoretical resonant frequencies. Discretization parameters: $\Delta r = \Delta z = 1.422$ mm $= \lambda/12$.

Optimization frequency [GHz]	Value of A factor	Computed frequency [GHz]	Relative error [%]
-	1.00000000000000	17.48441386145861	-1.1079
17.48441386145861	1.01138815606153	17.68136149647179	0.0060
17.68136149647179	1.01132331826806	17.68024016426026	-0.0003

Table 4.5: The table shows an iterative scheme of reducing error due to numerical dispersion. The resonant frequencies for the TE_{116} mode computed in one step are used in the next step to find the optimized value of A. The last column shows the relative error between computed and theoretical resonant frequencies. Discretization parameters: $\Delta r = \Delta z = 1.422$ mm $= \lambda/12$.

values of A, optimized for frequencies that are close to the theoretical resonant frequency f_t. For these values of A the resonant frequencies of TE_{116} are computed using IRAM-FDFD algorithm. It is clear that, although the values of A have been optimized for frequencies only approximating f_t, in all cases a very substantial improvement in the computed results has been obtained.

This observation leads to proposing the following iterative scheme of reducing the error due to numerical dispersion. If the approximation of the resonant frequency for a given mode is unknown, then an approximate value of this frequency may be computed using IRAM-FDFD and non-optimized value of A, i.e. $A = 1$. The approximate frequency may then be used to optimize the value of A and IRAM-FDFD method may be run with this optimized value of A. This step may be iterated as many times as needed. Table 4.5 shows

Optimization frequency [GHz]	Value of A factor	Computed frequency [GHz]	Relative error [%]
-	1.00000000000000	17.63133404107683	-0.2770
17.63133404107683	1.00281701766091	17.68046874620936	0.0009

Table 4.6: The table shows an iterative scheme of reducing error due to numerical dispersion. The resonant frequencies for the TE_{116} mode computed in one step are used in the next step to find the optimized value of A. The last column shows the relative error between computed and theoretical resonant frequencies. Discretization parameters: $\Delta r = \Delta z = 0.711$ mm $= \lambda/24$.

such a scheme for the example discussed. One notes that already the second step gives a substantial improvement of the results and only 3 steps are needed to reduce the relative error to less than 0.001%. An analogous iterative scheme has been shown in Table 4.6 for a refined discretization grid $\Delta r = \Delta z = 0.711$ mm $= \lambda/24$. Here only two steps are needed to reduce the error to less than 0.001%. By comparing the tables one also notes that applying a single optimization of A for a coarser grid gives more accurate results than applying a more refined grid. For the fine grid ($\Delta r = \Delta z = 0.711$ mm $= \lambda/24$), the matrix is approximately 4 times larger than for the coarse grid (which implies roughly four-time increment in the cost of computing matrix-vector product and the number of iterations performed by the solved) and consequently it is faster to run the algorithm with a coarse grid twice than to run the algorithm with a fine grid just once (cf. Table 6.16.

The above results indicate that the proposed technique of reducing the effect of numerical dispersion may be effectively applied to enhance quality of the obtained numerical results. Chapter 6 presents further numerical results validating the discussed technique in the case of a significantly more complicated structure of a large hemispherical resonator.

4.1.5 Implicit operator projection in FD methods

Although, as shown in the previous section, the memory cost of storing the matrix operator constructed using the FD technique generally equals $O(N)$ rather than $O(N^2)$, where N is the matrix size this cost may become significant with the increasing number of discretization points. It has been shown that for the 2D FDFD discretization the number of matrix elements to be stored equals approximately $11N$. (cf. [94]) This cost may be significantly reduced if the matrix is not stored explicitly. This, in turn, can be achieved if the Krylov subspace methods are applied, in which information on the operator is passed only via the the $\mathbf{A}v$ product.

The idea behind implicit representation of the matrix operator has already been introduced in Section 4.1.1 using the example of a 2D discrete Laplace operator.

This section develops a method of computing the matrix-vector product with an implicitly stored operator for the discrete operator $\underline{\underline{S}}$ from equation (4.24). If the input vector $\underline{v}$ is given in the following form:

$$\underline{v} = [\underline{v}^r \& \underline{v}^z]^T \equiv [\underline{v}^r_1, \underline{v}^z_1, \ldots, \underline{v}^r_N, \underline{v}^z_N]^T \tag{4.53}$$

then equations (4.22) and (4.23) provide an algorithm for computing the $\underline{\underline{S}}\,\underline{v}$ vector. If the vector components $\underline{v}^r$ and $\underline{v}^z$ have a meaning of the the r and z components of the electric flux density, respectively, then the algorithm may be summarized in the following steps:

1. Compute the ϕ component of the electric field intensity:

$$\underline{div}^r = \underline{\epsilon_\phi^{-1}} \left(\underline{\underline{D\phi divDr}}\, \underline{v}^r + \underline{\underline{D\phi divDz}}\, \underline{v}^z \right)$$

2. Compute the components of H_z and H_r associated with E_ϕ:

$$\underline{div}^z := \underline{\underline{HrrotE\phi}}\, \underline{div}^r$$

$$\underline{div}^r := \underline{\underline{HzrotE\phi}}\, \underline{div}^r$$

3. Compute the H_ϕ component using E_r and E_z:

$$\underline{vtemp}^r := \underline{\underline{H\phi rotEr}}\, \underline{\epsilon_r}^{-1} \underline{v}^r + \underline{\underline{H\phi rotEz}}\, \underline{\epsilon_z}^{-1} \underline{v}^z$$

$$\underline{vtemp}^z := \underline{vtemp}^r$$

4. Compute the components of $\omega^2 D_r$ and $\omega^2 D_z$ associated with H_ϕ:

$$\underline{w}^r = \underline{\underline{ErrotH\phi}}\, \underline{vtemp}^r$$

$$\underline{w}^z = \underline{\underline{EzrotH\phi}}\, \underline{vtemp}^z$$

5. Compute the components of H_z and H_r associated with E_r and E_z correspondingly:

$$\underline{vtemp}^r := \underline{\underline{HzrotEr}}\, \underline{\epsilon_r}^{-1} \underline{v}^r$$

$$\underline{vtemp}^z := \underline{\underline{HrrotEz}}\, \underline{\epsilon_z}^{-1} \underline{v}^z$$

6. Compute H_z and H_r:

$$\underline{div}^r := \underline{div}^r + \underline{vtemp}^r$$

$$\underline{div}^z := \underline{div}^z + \underline{vtemp}^z$$

7. Compute the components of $\omega^2 D_r$ and $\omega^2 D_z$ associated with H_z and H_r, respectively:

$$\underline{div}^{\,r} := \underline{\underline{ErrotHz}}\,\underline{div}^{\,r}$$

$$\underline{div}^{\,z} := \underline{\underline{EzrotHr}}\,\underline{div}^{\,z}$$

8. Add the components of $\omega^2 D_r$ and $\omega^2 D_z$:

$$\underline{w}^{\,r} := \underline{w}^{\,r} + \underline{div}^{\,r}$$

$$\underline{w}^{\,z} := \underline{w}^{\,z} + \underline{div}^{\,z}$$

It is important to stress at this point that the matrices involved in the above computation e.g. $\underline{\underline{D\phi divDr}}$ are *not* stored explicitly. Instead, the finite difference operations performed on the input vector elements are implemented, just as shown on a simple example presented in Section 4.1.1.2. Consequently, the only significant storage needed to complete the above matrix-vector product consists of: 1) the workspace of four vectors of length N: $\underline{v}$, $\underline{w}$, $\underline{vtemp}$ and $\underline{div}$, 2) storage for the elements of the permittivity tensor $\overleftrightarrow{\epsilon}$ which may equal up to $3N$. (In case of isotropic media this cost reduces to N.)

The computational complexity of the algorithm presented above equals $O(N)$ where N is the matrix (or vector) size. The memory storage needed to perform the matrix-vector product equals roughly $4N+3N$, as compared to $11N+2N$ in the case when matrix $\underline{\underline{S}}$ is stored explicitly. This means almost a two-time reduction of the required memory storage with an unchanged computational complexity of the algorithm. Moreover, application of the implicit projection scheme, implies that the projection of the operator may be performed simultaneously with the iterative solution of the eigenproblem.

4.2 Eigenfunction expansion based methods

The previous sections discussed the methods of finite-dimensional mapping of operators arising in modeling of 2D and 3D problems, based of the Finite Difference Frequency Domain technique. Below a different approach is applied based on representing fields as series of eigenfunctions (or basis functions). The considerations focus on applying the described technique to operators arising in modeling two dimensional systems.

4.2.1 Method of moments and the Galerkin algorithm

The method of finite-dimensional mapping described in this section is based on the representation of an operator by its products with chosen basis or testing functions spanning a given functional space. This approach is known from the Method of Moments or its most important version – the Galerkin method, using explicit scheme of projecting the input operator.

4.2.1.1 Explicit projection of the operator

Let us start the description of this discretization method with discussing the finite representation of functions. Let the domain X of a given operator $\mathbf{T}$ be a functional Hilbert space with a properly defined scalar product and $\{h_i\}_{i=1}^{\infty}$ be a complete orthonormal set of functions in the space X. According to the definition of completeness every function $u \in X$ is convergent to its Fourier series, being the expansion of u in terms of the basis functions:

$$\lim_{n} ||u - \sum_{i=1}^{n} (u, h_i)\, h_i|| = 0 \tag{4.54}$$

Consequently, any function from the space X is represented by a sequence of the Fourier coefficients $\{f_i\}_{i=1}^{\infty} = \{(u, h_i)\}_{i=1}^{\infty}$. Truncating this sequence to a finite number of terms gives a wanted finite mapping of the function u:

$$\underline{u} = [(u, h_1), (u, h_2), \ldots, (u, h_n)]^T \tag{4.55}$$

The method of discretization of the operator $\mathbf{T}$ immediately follows from the above representation of the functions. Defining the elements of the $n \times n$ matrix $\underline{\underline{T}} = [c_{ij}]_{i,j=1}^{n}$ as:

$$c_{ij} = (\mathbf{T}h_j, h_i) \tag{4.56}$$

we obtain a finite-dimensional linear operator being a mapping of the operator $\mathbf{T}$ which has the following property:

$$\underline{\underline{T}}\,\underline{u} = \underline{\mathbf{T}u} = [(\mathbf{T}u, h_1), (\mathbf{T}u, h_2), \ldots, (\mathbf{T}u, h_n)]^T \tag{4.57}$$

As already mentioned, the representation of the operator involving the matrix of scalar products given by the equation (4.56) is used by the method of moments in which this matrix is constructed explicitly. Unfortunately this may bring about a series of negative effects. Firstly, the matrix (4.56) may be dense and its explicit storage may require n^2 memory locations. Secondly, the matrix-vector product $\underline{\underline{T}}\,\underline{u}$ can involve $O(n^2)$ operations which may cause the computation time in the methods which are based on this representation to blow up for the increasing problem size n. (This effect is widely known e.g. for the Galerkin method used with the QR algorithm (cf. [95] or Chapter 6).

4.2.1.2 Implicit projection of the operator

The question which emerges is whether it is possible to find an orthonormal basis (a complete set of functions) in the Hilbert space X such that either the storage cost of the discretized operator or the cost of calculating the discussed matrix-vector product may be significantly reduced even if the matrix $\underline{\underline{T}}$ (cf. equation (4.57)) is dense. The answer is positive for a certain class of functional Hilbert spaces chosen for the operator's domain. This wide class, being the most important one in a variety of application fields,

may be defined as the space of square integrable functions defined over a bounded region Ω, namely the $L_2(\Omega)$ space, with the scalar product defined as follows:

$$(u, v) = \int uv^* d\Omega \tag{4.58}$$

If, without significant loss of generality (cf. [84]), we shall limit our discussion to the case of the L_2 space defined over a two-dimensional bounded rectangular region $\Omega = ([0, b] \times [0, a]) \in R^2$ then the orthonormal bases which have the desired feature are the trigonometric complete sets of functions:

$$h_{ij}^x = A_{ij} \sin\left(\frac{i\pi x}{b}\right) \cos\left(\frac{j\pi y}{a}\right) \tag{4.59}$$

or

$$h_{ij}^y = B_{ij} \cos\left(\frac{i\pi x}{b}\right) \sin\left(\frac{j\pi y}{a}\right) \tag{4.60}$$

where A_{ij} and B_{ij} are properly defined normalization constants. If we represent magnetic field in terms of the above basis, the Perfect Electric Conductor (PEC) condition will be satisfied at the boundary of the region $[0, b] \times [0, a]$. If we consider e.g. a two-dimensional vector field $u = (u^x, u^y)$, where $u^x, u^y \in L^2([0, b] \times [0, a])$, then u may be represented e.g. by the following series:

$$u^x = \sum_{ij} c_{ij}^x h_{ij}^x$$

$$u^y = \sum_{ij} c_{ij}^y h_{ij}^y$$

If the above series are truncated then the emerging finite approximation of the function u is a vector of the Fourier coefficients:

$$\begin{aligned} \underline{u} &= [c_{11}^x, c_{12}^x, \ldots, c_{mn}^x, c_{11}^y, c_{12}^y, \ldots, c_{mn}^y] \\ &= [(u^x, h_{11}^x), (u^x, h_{12}^x), \ldots, (u^x, h_{mn}^x), (u^y, h_{11}^y), (u^y, h_{12}^y), \ldots, (u^y, h_{mn}^y)] \end{aligned} \tag{4.61}$$

The most significant observation about the above vector is that it is simply a vector of samples of the two-dimensional Fourier transform of the function $u = (u^x, u^y)$. Consequently, keeping in mind that the inner products are given by the integrals (4.58), the approximations of the vector elements may be numerically found using the two-dimensional Discrete Fourier Transforms (DFTs). In turn, the two-dimensional DFTs may be very efficiently computed by applying the Fast Fourier Transform (FFT) algorithm, proposed first by Cooley and Tukey (cf. [20]). The following section shows how this observation may be used to reduce both cost of calculating the $\underline{\underline{T}}\,\underline{u}$ product as well as cost of storing the operator matrix by applying the implicit, instead of explicit, matrix representation.

4.2.2 Calculation of the inner products and implicit operator projection

According to the previous section computing of the $\underline{\underline{T}}\,\underline{u}$ product may be viewed as calculating the inner products (in a finite-dimensional space) of $(\mathbf{T}u, h^x_{ij})$ and $(\mathbf{T}u, h^y_{ij})$ with the given vector of Fourier coefficients (4.61). This kind of approach enables one to develop a procedure of computing the $\underline{\underline{T}}\,\underline{u}$ product which does not require the explicit storage of the *dense* matrix $\underline{\underline{T}}$ [96]. Thanks to this both memory and computational cost may be reduced. The discussed operation may be performed in the following steps:

1. Using the given Fourier coefficients (4.61) calculate the values of the function u for a discrete set of points from the Ω spatial domain by computing a two-dimensional *backward* FFT.

2. Calculate the values of the $\mathbf{T}u$ function at the gridpoints of the domain Ω using the previously calculated values of u.

3. Compute the inner products (the Fourier coefficients) $(\mathbf{T}u, h^x_{ij})$ and $(\mathbf{T}u, h^y_{ij})$ by performing a two-dimensional *forward* FFT.

The above scheme has been illustrated in Figure 4.12 for the function $u \equiv H$ being a two-dimensional vector field. In the Figure, the function $H = (H^x, H^y)$ is represented in the DFT domain by 200 Fourier coefficients, which corresponds to 10 expansion functions in every spatial direction for both H^x and H^y. (Referring to the equations (4.59) and (4.60) the indices run as follows: $i = 1, 2, \ldots, 10$ and $j = 0, 1, \ldots, 9$.) Then, the discrete values of the function H (i.e. the actual field distribution) are computed using two 2D backward FFTs. In the example we obtain two 256×256 arrays of samples of the function H in the 2D spatial domain. Then, the operation $\mathbf{T}$ on H gives a matrix of samples of the $\mathbf{T}H$ function which is subsequently transformed using forward 2D FFTs to obtain the desired Fourier coefficients. One should note at this point an important relation which links the discussed discretization (based on finite expansion series) and the Finite Difference (FD) method. In the Step 2 of the above scheme one calculates the values of the $\mathbf{T}u$ function at the discrete gridpoints in entirely the same way as in the FD method. The additional Steps (1 and 3) are required to move back and forth between the spatial domain and the DFT domain.

It is important to note that computational procedure shown in Figure 4.12 performs a projection onto a 10×10 space of Fourier coefficients from a much larger (256×256) space and the actual operation of $\mathbf{T}$ on u is performed in a 'large' space domain. The function is oversampled in the space domain, which means that a reduced number of Fourier coefficients is calculated using more samples than necessary. Applying oversampling, i.e. a finer grid in the spatial domain reduces the effect of numerical dispersion (cf. Figure 4.6 in Section 4.1.3). This allows one to compute the first e.g. 100 Fourier coefficients with greater accuracy, while omitting all the other, containing larger (and often very serious) numerical errors. In this way the effect of numerical dispersion is reduced, while maintaining relatively small problem size, associated with a 'compact' representation of functions in the DFT domain.

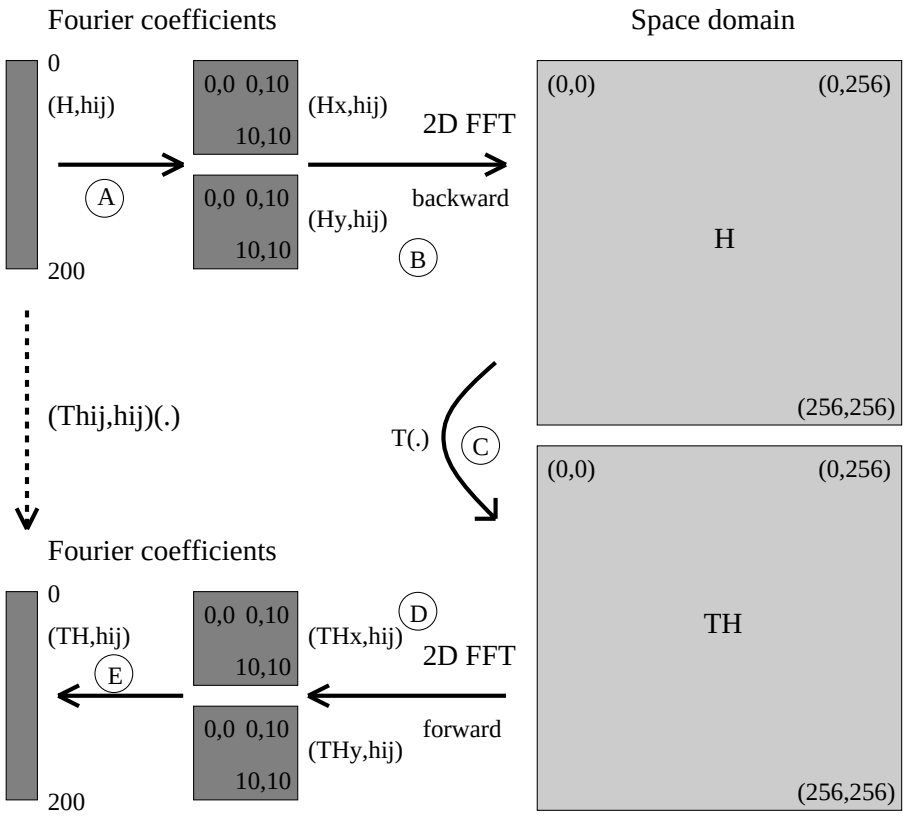


Figure 4.12: Calculation of the matrix-vector product for the DFT domain operator formulation is performed in a series of steps: A – rearranging a 1D vector in the form of two 2D matrices, B – performing two 2D backward FFTs, C – performing linear transformation $\mathbf{T}$ of the two fields in the space domain, D – performing two 2D forward FFTs, E – rearranging the fields in the form of a 1D vector.

The other question is: What is the numerical complexity of the algorithm? If K_x and K_y denote the lengths of the Fast Fourier Transforms, i.e. the number of sample points in the spatial domain in the x and y directions, respectively and the numbers of expansion functions used to represent the functions in the DFT domain equal N_x and N_y in the respective directions then the cost of performing the steps 1 and 3 in the calculation of the matrix-vector product equals $O(4N_xK_y \log K_y + 4K_yK_x \log K_x)$. (The cost of performing a one-dimensional FFT of the length N is $O(N \log N)$. – cf. [11]). Denoting $K = K_x \cdot K_y$ and $N = N_x \cdot N_y$, this cost may be estimated at a level $O(K \log K)$ since the length of the FFTs (the number of sampling points in the spatial domain) should be proportional to the number of expansion functions. The next issue is estimating the cost of the $\mathbf{T}u$ product (step 2), where function u is represented by N samples in the spatial domain. It is hard to give a non-conservative bound for this cost in the case of a general linear operator. Still, if only differential operators are considered (just as in the previous section) then this cost is given by $O(N)$. It is now seen that the overall cost of calculating the matrix-vector product in this representation equals $O(K \log K)$.

One may ask what are the advantages of this representation as compared either to the FD finite-dimensional mapping in which the matrix-vector product could be calculated within the linear time cost or the classical Galerkin representation involving a dense matrix.. Clearly, in the DFT representation only the step 2 involves a similar number of computations as the entire matrix-vector product in the FD discretization. The advantage of the DFT representation may be seen if one compares the dimensions of the resulting discrete operators. The size of the vectors in the DFT domain equals K which, due to the oversampling (needed to reduce the dispersion error in the discrete spatial domain), is usually considerably smaller than the vector size resulting from the FD mapping. (e.g. For the FFT length which equals 256, the number of applied expansion functions usually equals 20, 40 or at most 60. Consequently in two dimensions the problem size equals e.g. 3600 in the DFT space as compared to approximately 66000 in the spatial domain.) So, the DFT representation usually reduces the problem size which influences the execution time the program solving the discrete eigenvalue problem. Summing up, if the DFT representation is applied, the extra time spent on calculating in matrix-vector products is then regained by spending less time on solving the eigenproblem.

Referring to the memory complexity, this method needs relatively little space to calculate the matrix-vector product. The memory requirements include the space necessary to store the samples of the function $u = (u^x, u^y)$ in the spatial domain whose size equals $2N_xN_y = 2N$ and the space needed to perform Fourier transforms. In the Winograd version of the FFT algorithm (cf. [120] or [91]) the extra workspace needed to perform the one-dimensional transform will equal approximately $3N_x$ ($3N_y$). If the space needed to store the input/output vectors of Fourier coefficients is taken into account then the overall memory complexity equals: $O(2K+2N+6\sqrt{N})$ (assuming that $N_x = O(N_y)$). In this estimation the cost of storing the operator matrix $\underline{\underline{T}}$ is not taken into account as the implicit storage is assumed. Clearly, if this matrix is stored explicitly (as in the classical Galerkin approach), the memory requirements may increase dramatically to $O(n^2 + 2n)$ (dense matrix representation).

Last, but not least, it should be pointed out, that analogously as for the FD case, the implicit representation of the discrete operator presented above allows one to perform projection of the operator simultaneously with the $\mathbf{A}v$ operation. This means that if a Krylov subspace method is selected to solve the discrete problem, then the projection may be embedded into the actual solution process [96]. As shown in Chapter 6, this fact has serious implications, referring to performance of numerical solvers.

4.3 Hybrid methods

Previous sections presented finite-dimensional projection techniques based on either the Finite Difference Frequency Domain (FDFD) algorithm or eigenfunction expansion method. Below a hybrid discretization algorithm is presented joining FDFD and eigenfunction expansion representation of fields and operators. It concentrates on proposing a method

applicable to a full 3D problem that uses finite difference approximation in two dimensions and eigenfunction expansion in the third dimension.

The main goal of the proposed technique is to reduce the size of the resulting discrete eigenproblem and reduce the spectral radius of the emerging discrete operator. As shown in Section 4.1.2 application of finite difference approximation in all three spatial dimensions, necessary to model a general 3D system e.g. defined in cylindrical coordinate system, results in the operator matrix which has a very large size as well as relatively large spectral radius (compare Table 4.3). For instance, if we consider modeling a cylindrical resonant cavity with a formulation involving e.g. two electric field components using $28 \times 128 \times 36$ $(r - \phi - z)$ finite difference discretization grid then the resulting problem size equals approximately 258040. The spectral radius of the discrete operator matrix equals $\approx 3 \cdot 10^{27}$. The method proposed in the following section tries develop a more efficient projection scheme, which will allow a considerably more compact representation of the discrete operator and significant reduction of the spectral radius of the operator.

4.3.1 Joint FDFD and eigenfunction expansion discretization for 3D problems

So far methods of finite-dimensional mapping valid either for 2D problems or 3D problems reducible to 2D problems (e.g. due to rotational symmetry of the structure) have been discussed. This section presents a discretization method valid for 3D systems defined over cylindrical coordinate system which, unlike in sections 2.1.2 or 4.1.2, do not need be homogeneous in the ϕ direction (do not need to possess rotational symmetry).

Although such systems may still be described by two field components (cf. Section 2.1.2), e.g. D_r and D_z components of the electric flux density, the simple representation:

$$D_r(r_j, \phi, z_i) = \vec{D}^r_{i,j,n_0} e^{(-jn_0\phi)} \tag{4.62}$$

from Section 2.1.2 may no longer be applied. Instead one should assume the following form of the fields:

$$\begin{aligned} D_r(r_j, \phi, z_i) &= D^r_{ij1} + (-1)^{N_\phi - 1} D^r_{ijN_\phi} \\ &+ \sum_{l=2}^{N_\phi/2} \left(2D^r_{ij(2l-2)} \cos((l-1)\phi) - 2D^r_{ij(2l-1)} \sin((l-1)\phi) \right) \end{aligned} \tag{4.63}$$

$$\begin{aligned} D_z(\hat{r}_j, \phi, \hat{z}_i) &= D^z_{ij1} + (-1)^{N_\phi - 1} D^z_{ijN_\phi} \\ &+ \sum_{l=2}^{N_\phi/2} \left(2D^z_{ij(2l-2)} \cos((l-1)\phi) - 2D^z_{ij(2l-1)} \sin((l-1)\phi) \right) \end{aligned} \tag{4.64}$$

The above equations establish functional representation for the fields in the ϕ direction. A finite series is associated with every FD grid point in the $r - z$ plane. In this

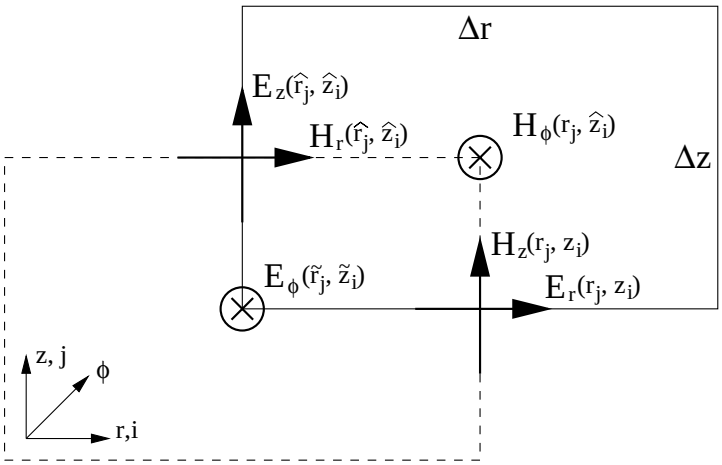


Figure 4.13: Yee's mesh in the $r-z$ plane in cylindrical coordinates.

way the finite difference projection in r and z directions is joined with the eigenfunction expansion, known from the method of moments, in the ϕ direction. In the above formulae r_j, z_i, $\hat{r}_j$ and $\hat{z}_i$ define the grid points on the Yee mesh (cf. Figure 4.13):

$$\begin{aligned} r_j &= j\Delta r + \Delta r/2, \quad \hat{r}_j = j\Delta r \\ z_i &= i\Delta z + \Delta z/2, \quad \hat{z}_i = i\Delta z \end{aligned} \tag{4.65}$$

where: $i = 1, \ldots, N_z$, $j = 1, \ldots, N_r$ and N_z and N_r determine the number of discretization points in the z and r directions.

It will be shown below that also this hybrid representation allows one to develop implicit representation of the discrete operator and consequently, the finite-dimensional projection may be performed simultaneously with the $\mathbf{A}u$ operation. The implicit representation of the discrete operator also allows one to reduce the required memory storage. If explicit matrix representation were used the resulting matrix would have either a high bandwidth (which would deteriorate parallel performance of the solver – cf. Chapter 5) or quasi 5-diagonal block structure with non-zero blocks being dense matrices. This would imply significant increment in both memory and computational cost.

One may note that equations (4.63) and (4.64) are in fact inverse real Discrete Fourier Transforms (DFTs) in the ϕ direction with Fourier coefficients D^r_{ijn} and D^z_{ijn} for fixed i and j. The discrete representation of the electric field components which follows is:

$$\underline{D}_r = \left[D^r_{1,1,1}, \ldots, D^r_{N_z,1,1}, \ldots D^r_{N_z,N_r,1}, \ldots, D^r_{1,1,M}, \ldots, D^r_{N_z,N_r,M}\right]^T \tag{4.66}$$

$$\underline{D}_z = \left[D^z_{1,1,1}, \ldots, D^z_{N_z,1,1}, \ldots D^z_{N_z,N_r,1}, \ldots, D^z_{1,1,M}, \ldots, D^z_{N_z,N_r,M}\right]^T \tag{4.67}$$

where $M \leq N_\phi$. Clearly, the number of discretization points in the ϕ direction does not have to (or even should not) equal the number of Fourier coefficients used to represent

the appropriate fields. Usually, in order to reduce the effect of numerical dispersion (and enhance the accuracy of computing of the Fourier coefficients), the oversampling in the spatial domain is applied so that $M < N_\phi$. If this is the case, then M is assumed to be an odd number and in equations (4.63) and (4.64) it is imposed that for $n = M+1, \ldots, N_\phi$:

$$D^r_{ijn} = D^z_{ijn} = 0$$

One should also note that, with Fourier coefficients given by (4.66) and (4.67), performing the inverse real DFT of length N_ϕ (using e.g. the FFT algorithm) gives values of D_r and D_z at discrete grid points defined by (4.65) and for $\phi = \phi_k = (k-1)\Delta\phi$, where $k = 1, \ldots, N_\phi$ and $\Delta\phi = 2\pi/N_\phi$.

Below an implicit definition of the discrete electromagnetic operator is given, based on the presented hybrid representation of the functional space. In fact, the algorithm for computing the matrix-vector product is developed, which implicitly defines the discrete operator for which an eigenproblem is then solved. The derivation of the algorithm is analogous as in Section 4.1.5 and also applies results from Section 4.2.2.

Given discrete field components D_r and D_z defined by the vectors (4.66) and (4.67) the discrete representation of D_ϕ is computed in the proposed algorithm using the divergence equation discretized in r and z directions using FD method:

$$\begin{gathered}\frac{1}{\Delta r}\left[r_j D_r(r_j,\phi,z_i) - r_{j-1} D_r(r_{j-1},\phi,z_i)\right] + \\ \frac{\hat{r}_j}{\Delta z}\left[D_z(\hat{r}_j,\phi,\hat{z}_{i+1}) - D_z(\hat{r}_j,\phi,\hat{z}_i)\right] = -\frac{\partial}{\partial\phi}\left[D_\phi(\tilde{r}_j,\phi,\tilde{z}_i)\right]\end{gathered} \tag{4.68}$$

where:

$$\tilde{r}_j = j\Delta r \quad \tilde{z}_i = i\Delta z + \Delta z/2 \tag{4.69}$$

Substituting $D_r(r_j,\phi,z_i)$ and $D_z(\hat{r}_j,\phi,\hat{z}_i)$ with the finite series (4.63) and (4.64) and integrating the above equation in the ϕ direction yields:

$$D_\phi(\tilde{r}_j,\phi,\tilde{z}_i) = A_{ij}\phi + B_{ij}\sin(N_\phi\phi/2) + \sum_{l=2}^{N_\phi/2}\left[2C_{ijl}\sin((l-1)\phi) + 2D_{ijl}\cos((l-1)\phi)\right] \tag{4.70}$$

where:

$$A_{ij} = \frac{1}{\Delta r}\left[r_{j-1}D^r_{i,j-1,1} - r_j D^r_{i,j,1}\right] + \frac{\hat{r}_i}{\Delta z}\left[D^z_{i,j,1} - D^z_{i+1,j,1}\right] \tag{4.71}$$

$$B_{ij} = \frac{2}{N_\phi\Delta r}\left[r_{j-1}D^r_{i,j-1,N_\phi} - r_j D^r_{i,j,N_\phi}\right] + \frac{2\hat{r}_i}{N_\phi\Delta z}\left[D^z_{i,j,N_\phi} - D^z_{i+1,j,N_\phi}\right] \tag{4.72}$$

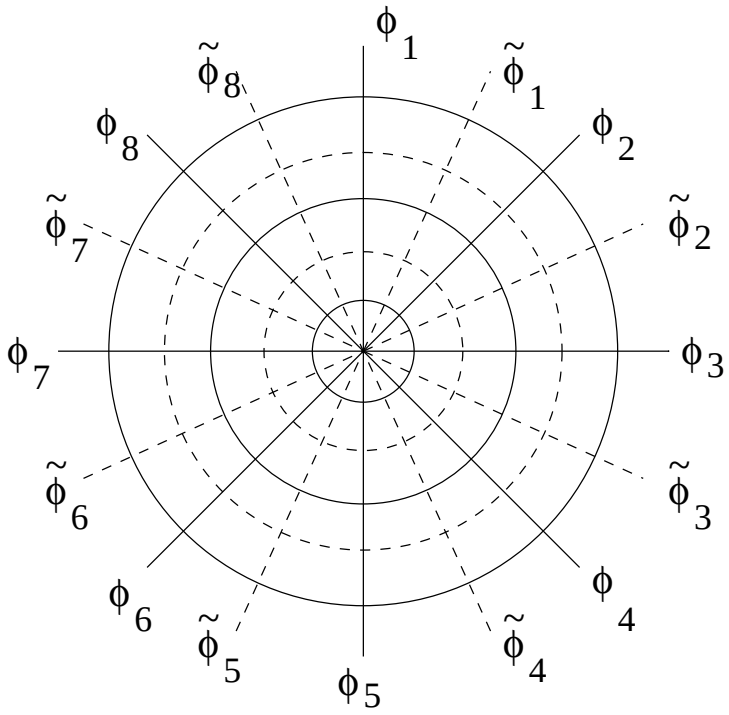


Figure 4.14: Yee's mesh in the $r-\phi$ plane in cylindrical coordinates for the case $N_\phi = 8$.

$$C_{ijl} = \frac{1}{(l-1)\Delta r}\left[r_{j-1}D^r_{i,j-1,(2l-2)} - r_j D^r_{i,j,(2l-2)}\right] + \frac{\hat{r}_i}{(l-1)\Delta z}\left[D^z_{i,j,(2l-2)} - D^z_{i+1,j,(2l-2)}\right] \tag{4.73}$$

$$D_{ijl} = \frac{1}{(l-1)\Delta r}\left[r_{j-1}D^r_{i,j-1,(2l-1)} - r_j D^r_{i,j,(2l-1)}\right] + \frac{\hat{r}_i}{(l-1)\Delta z}\left[D^z_{i,j,(2l-1)} - D^z_{i+1,j,(2l-1)}\right] \tag{4.74}$$

From the above equations it follows that, given coefficients A_{ij}, B_{ij}, C_{ij} and D_{ij}, we may compute the values of the field component D_ϕ at the discrete points $(\tilde{r}_j, \phi_k, \tilde{z}_i)$, where $\phi_k = (k-1)\Delta\phi$. As we apply the Yee mesh, so the values of the magnetic field are defined over a dual or 'staggered' grid, as compared to the grid used to evaluate the electric field (cf. Figure 4.14). Consequently, we will also need the values of D_ϕ field component at the grid points $(\tilde{r}_j, \tilde{\phi}_k, \tilde{z}_i)$, where $\tilde{\phi}_k = (k-1)\Delta\phi + \Delta\phi/2$. In order to compute these values we transform equation (4.70) in the following way:

$$\begin{aligned} D_\phi(\tilde{r}_j, \tilde{\phi}_k, \tilde{z}_i) &= A_{ij}\tilde{\phi}_k + B_{ij}\cos(N_\phi\phi_k/2) + \\ \sum_{l=2}^{N_\Phi/2} &\{2\left[C_{ijl}\sin((l-1)\Delta\phi/2) + D_{ijl}\cos((l-1)\Delta\phi/2)\right]\cos((l-1)\phi_k) \\ &-2\left[-C_{ijl}\cos((l-1)\Delta\phi/2) + D_{ijl}\sin((l-1)\Delta\phi/2)\right]\sin((l-1)\phi_k)\} \end{aligned} \tag{4.75}$$

where $\phi_k = (k-1)\Delta\phi$. The main reason for applying the above equation is that values of D_ϕ at grid points of the 'staggered' grid (with $\phi = \tilde{\phi}_k$) may be computed using inverse Discrete Fourier Transforms defined over the 'primary' grid with $\phi = \phi_k$.

In the next steps of the proposed algorithm of computing the matrix-vector product, the values of the components of magnetic field intensity are computed using one of the Maxwell's curl equations. By applying analogous techniques as those presented above one finds: $H_r(r_j, \tilde{\phi}_k, z_i)$, $H_\phi(r_j, \phi_k, \hat{z}_i)$ and $H_z(r_j, \tilde{\phi}_k, z_i)$. The last step involves computation of the Fourier coefficients of $\omega^2 D_r$ and $\omega^2 D_z$. This step involves performing forward DFTs, using the values of previously computed magnetic field components.

The procedure outlined the previous paragraph is systematically summarized below. If the input vector $\underline{D}$ (cf. equations (4.63) and (4.64)) is given in the following form:

$$\underline{D} = [\underline{D}_r \& \underline{D}_z]^T = \left[\underline{D}_r^1, \underline{D}_z^1, \ldots, \underline{D}_r^N, \underline{D}_z^N\right]^T \tag{4.76}$$

then the steps of the algorithm of computing the matrix-vector product using the implicit operator representation are as follows (note the analogy with algorithm presented in Section 4.1.5):

1. Compute $E_\phi(\tilde{r}_j, \tilde{\phi}_k, \tilde{z}_i)$ from the Fourier coefficients of D_r and D_z and store the result in $\underline{div}^r$. This step involves $N_r N_z$ inverse real FFTs.

2. Compute the components of H_z and H_r at the grid points (r_j, ϕ_k, z_i) and $(r_j, \tilde{\phi}_k, z_i)$ respectively associated with E_ϕ. Store the results in $\underline{div}^r$ and $\underline{div}^z$.

3. Compute $E_r(r_j, \phi_k, z_i)$ and $E_z(\hat{r}_j, \phi_k, \hat{z}_i)$ from the Fourier coefficients of D_r and D_z and store the result in $\underline{wtemp}^r$ and $\underline{wtemp}^z$. This step involves $N_r N_z$ inverse real FFTs.

4. Compute $H_\phi(r_j, \phi_k, \hat{z}_i)$ using $E_r(r_j, \phi_k, z_i)$ and $E_z(\hat{r}_j, \phi_k, \hat{z}_i)$. Store the result in $\underline{vtemp}^r$.

5. Copy $\underline{vtemp}^r$ to $\underline{vtemp}^z$.

6. Compute the 'parts' of Fourier coefficients for $\omega^2 D_r$ and $\omega^2 D_z$ associated with H_ϕ, using $H_\phi(r_j, \phi_k, \hat{z}_i)$. Store the result in $\underline{w}^r$ and $\underline{w}^z$. This step involves $2 \cdot N_z N_r$ forward FFTs.

7. Compute the Fourier coefficients for E_r and E_z from $E_r(r_j, \phi_k, z_i)$ and $E_z(\hat{r}_j, \phi_k, \hat{z}_i)$. Store the result in $\underline{wtemp}^r$ and $\underline{wtemp}^z$. This step involves $2 \cdot N_z N_r$ forward FFTs.

8. Compute the components of H_z and H_r at the grid points $(r_j, \tilde{\phi}_k, z_i)$ and $(r_j, \tilde{\phi}_k, z_i)$ respectively associated with E_r and E_z, correspondingly. Store the result in $\underline{wtemp}^r$ and $\underline{wtemp}^z$. This step involves $2 \cdot N_z N_r$ inverse FFTs.

9. Compute $H_z(r_j, \tilde{\phi}_k, z_i)$ and $H_r(r_j, \tilde{\phi}_k, z_i)$:

$$\underline{div}^{\,r} := \underline{div}^{\,r} + \underline{wtemp}^{\,r}$$

$$\underline{div}^{\,z} := \underline{div}^{\,z} + \underline{wtemp}^{\,z}$$

10. Compute the Fourier coefficients for H_z and H_r respectively using $H_z(r_j, \tilde{\phi}_k, z_i)$ and $H_r(r_j, \tilde{\phi}_k, z_i)$. Store the results in $\underline{div}^{\,r}$ and $\underline{div}^{\,z}$. This step involves $2 \cdot N_z N_r$ forward FFTs.

11. Compute the 'parts' of Fourier coefficients for $\omega^2 D_r$ and $\omega^2 D_z$ associated with H_z and H_r respectively by using the previously computed Fourier coefficients for H_z and H_r. Store the result in $\underline{div}^{\,r}$ and $\underline{div}^{\,z}$.

12. Add the two parts of the Fourier coefficients for $\omega^2 D_r$ and $\omega^2 D_z$:

$$\underline{w}^{\,r} := \underline{w}^{\,r} + \underline{div}^{\,r}$$

$$\underline{w}^{\,z} := \underline{w}^{\,z} + \underline{div}^{\,z}$$

As computation of the Fast Fourier Transform of length N_ϕ involves $O(N_\phi \log_2(N_\phi))$ operations, the overall computational complexity of the algorithm presented above equals $O(K)\log_2(N_\phi)$, where $K = 2N_z N_r N_\phi$ is the size of the spatial domain. (The matrix (vector) size equals $N = 2N_x N_r M$.) One may note that this complexity is higher than the linear complexity observed for matrices obtained using the FDFD technique. The memory storage needed to perform the matrix-vector product consists of the workspace needed to store vectors $\underline{v}$ (N), $\underline{w}$ (N), $\underline{div}$ (K), $\underline{vtemp}$ (K) and $\underline{wtemp}$ (K) plus a negligible storage needed to implement the performed linear operations, where K is the number of samples in the spatial domain (size of the spatial domain). Additionally, the diagonal permittivity tensor $\overleftrightarrow{\epsilon}$ requires up to $3K$ memory locations. Consequently, the memory cost equals roughly $2N + 6K$.

4.3.1.1 Spectral radius of the discrete operator

Referring to the spectral radius of the implicitly represented discrete operator it is determined by: 1) the parameters of the discretization grid in $r - z$ plane: $\Delta r = \Delta z$, 2) the number of Fourier coefficients M (cf. equations (4.63) and (4.64)) used to represent the functions in the ϕ direction. Once again the estimation (4.25) may be used. Still, in this case $n_{\max} = M$ (and not $n_{\max} = N_\phi/2$). One may note that even if a fine grid is applied in the spatial domain in order to reduce the negative effects of numerical dispersion, application of the expansion-based representation allows one to reduce the negative effects associated with a large spectral radius. Consequently, the convergence of the iterative solvers is improved in this way. For example, if a four fold oversampling is applied, e.g. $M = 64$ and $N_\phi = 256$, then the above hybrid method gives a discrete operator with a spectral radius which is 4 times smaller than for the operator obtained using FD mapping in all three spatial dimensions.

Projection method	Problem type	Operator represen -tation	Memory cost	Cost of computing $\underline{\underline{A}}\,\underline{v}$ product	Spectral radius
FDFD	2D	explicit	$7N$	$O(N)$	medium
EE	2D	implicit	$2K + 2N + 6\sqrt{N}$	$O(K\log(K))$	small
EE	2D	explicit	$2N + N^2$	$O(N^2)$	small
FDFD	3D→2D	explicit	$13N$	$O(N)$	medium
FDFD	3D→2D	implicit	$7N$	$O(N)$	medium
HYBRID	3D	implicit	$2N + 6K$	$O(K\log(N_\phi))$	medium
FDFD	3D	explicit	$> 13N$	$O(N)$	large

Table 4.7: Comparison of memory and computational costs associated with different finite-dimensional mapping techniques. In the table the problem type denoted as '3D→2D', refers to problems reducible to 2D problems. N denotes the problem size, K is the overall number of samples in the spatial domain, N_ϕ is the number of samples in the ϕ direction, EE refers to Eigenfunction Expansion-based algorithms.

4.4 Comparison of FDFD and eigenfunction expansion finite-dimensional mapping techniques

This section presents a comparison of selected numerical properties of discrete operators obtained using different finite-dimensional mapping techniques presented in this chapter.

First, the following general remark should be made about the two discussed classes of finite-dimensional mapping techniques: i.e. FDFD-based methods and algorithms using functional expansions. In order to meet the requirements of large scale modeling both approaches try to deal with the errors due to numerical dispersion.[2] Still, in order to avoid excessive memory and computational cost (which in fact may also preclude a certain method from modeling large scale problems) different techniques have to be applied to this end. For the FDFD-based methods an inexpensive technique of correcting the effects of numerical dispersion is introduced, which does not increase the size of the solved numerical problem. On the other hand, methods using eigenfunction expansion techniques apply oversampling in the spatial domain, so that operations in that domain are performed using a refined grid while the size of the solved eigenproblem remains unchanged.

Referring to memory costs related to the representation (storage) of the discrete operators as well as costs of computing the matrix-vector product for different proposed finite-dimensional projection techniques, they are collected in Table 4.7. It may be noted that the costs of computing the matrix-vector product in methods applying eigenfunction

[2]In the case of algorithms using functional expansions the dispersion error arises only in implicit schemes, which operate both in space and transform domains – cf. Section 4.2.

expansion techniques are generally higher than in method applying the FDFD technique only. Still, due to the fact that often the problem size is smaller for the former methods than for the latter ones the computation time may become faster in the first case. (This particularly applies if FD discretization is applied in all three spatial dimensions.) Another general conclusion which can be drawn, is that if memory cost becomes crucial (which often happens in large scale modeling) then application of implicit operator representation results in important memory savings.

Summing up, the presented finite-dimensional projection methods meet the goals set forth in the beginning of this chapter. They appear to be well suited for solving large scale operator problems due to the following general features:

- The memory cost associated with storage of the resulting discrete operator is low (i.e. linear and not e.g. quadratic).
- The cost of computing the $\underline{\underline{A}}\,\underline{v}$ product involving the discrete operator is linear or linear-logarithmic (also less than quadratic).
- The projection methods include techniques reducing the effects of numerical dispersion allowing significant reduction of errors in large scale problems without increasing the size of the solved problem (and the spectral radius of the discrete operator).
- All the methods make use of implicit projection strategy which allows one to perform projection of the operator simultaneously e.g. with solution of the discrete eigenproblem while using Krylov subspace methods.

The mentioned features of the proposed projection techniques are truly advantageous while dealing with complicated, large scale eigenproblems. Consequently, the proposed projection techniques should provide a proper base for developing high performance algorithms. So far, we were focusing on the requirements such as low memory and computational complexity or reduced spectral radius to be satisfied by the proposed algorithms. Deliberately we have not touched the last of the goals pointed out in the beginning of this chapter, i.e. efficient parallelization in scalable systems. This means that we need to assure that proposed numerical techniques satisfy additional requirements related to parallel processing in scalable supercomputer systems, if they are to be applied to solving the most complicated large scale problems. The following chapter is entirely dedicated to consider these important issues.

Chapter 5

Solving large scale operator eigenproblems in scalable parallel systems

The previous chapters defined a number of requirements which have to be satisfied by a numerical algorithm if it is to be applied to solving complex, large scale electromagnetic eigenproblems. Various issues crucial to a broadly understood efficiency of the algorithm and accuracy of the computations have been discussed. Based on these guidelines a few low-cost methods of finite-dimensional projection methods have been proposed meeting the requirements of large scale modeling. These methods may be used jointly with modern iterative Krylov subspace methods to solve electromagnetic boundary value problems.

Nevertheless, even if low-cost, efficient numerical methods are used certain most complex problems may only be modeled using massively parallel processing, i.e. the multiple power of a large number of processing elements working in a parallel supercomputer architecture. This chapter addresses issues of high performance parallel computing in the context of large scale electromagnetic modeling and tries to present factors which determine the applicability of the discussed numerical techniques in parallel processing environment. They refer either to finite-dimensional projection techniques or methods of solving discrete operator eigenproblems. A number of technical issues is described during the following discussion. Still, in author's opinion their presentation is necessary for understanding the problems arising in parallel computing using scalable supercomputer systems.

The following section discusses some aspects of finite-dimensional mapping which play an important role if the algorithms involving discrete operators and functions are to be implemented in scalable parallel systems. The general conclusions drawn from this discussion substantiate the choice of the operator projection methods (already presented in Chapter 4) applied to build parallel solvers for large scale operator eigenproblems.

Later on in this chapter, parallel design and implementation of the algorithms of solving electromagnetic operator eigenproblems described in Chapter 2 is presented. The implementations of the iterative solvers are discussed jointly with the parallel designs of matrix-vector products for the projection schemes discussed in Chapter 4, as to provide a complete description of methods which can be used to solve eigenproblems for a wide class of electromagnetic operators.

The parallel solvers presented in this chapter include:

- *IRAM-FDFD solvers*, based on the IRAM iterative process and the Finite Difference (FD) discretization of the input operator, presented in Section 4.1, valid for modeling of 2D structures or 3D structures homogeneous in ϕ direction.

- *IRAM-FFT solver*, based on the IRAM iterative process and the implicit discrete representation of the operator, applied jointly with the FFT algorithms to enhance the efficiency of the method. (cf. Section 4.2)

- *IRAM-HYBRID solver*, based on IRAM and the hybrid method of implicit operator projection based on FDFD and eigenfunction expansion techniques, as described in Section 4.3.1.

The base for the implementation of all the parallel solvers is the P_ARPACK library, described in Section 8 and offering portable parallel implementation of the IRAM iterative algorithm, ready for use in distributed memory systems. The tasks which have to parallelized independently include:

- The matrix-vector product operation for the matrix operator discretized using the FD mapping technique with an explicitly or implicitly stored operator matrix.

- Two-dimensional backward and forward Fast Fourier Transforms.

- The matrix-vector product in the IRAM-FFT algorithm which requires calculation of the appropriate inner products, involving computation of 2D FFTs.

- The matrix-vector product for the IRAM-HYBRID technique.

5.1 Discussion of discretization aspects in scalable parallel systems

Before presenting any particular parallel implementations associated with some finite-dimensional mapping strategy it is worthwhile to make a few remarks on the relations between the efficiency of parallel matrix computations and the choice of a finite-dimensional mapping technique for a given operator.

Let us first define the main parameters for measuring performance of a given parallel algorithm. One of the main goals of parallel processing is reducing the wall-clock computation time. Consequently, the most substantial quantity which we usually want to assess is the speedup of computations versus the number of applied processing elements. The speedup is usually defined as:

$$S_{relative} = \frac{T_1}{T_P} \tag{5.1}$$

where T_1 is the execution time of the parallel program on one processor and T_P is the execution time of the same parallel program on P processors. The above definition is known as a definition of a 'relative speedup', because it takes as a reference the single-processor execution time of the investigated parallel algorithm. The absolute value of the speedup is obtained if T_1 from formula (5.1) is taken as the execution time of the best-known sequential algorithm. Another parameter, closely related to speedup is the efficiency of a parallel program:

$$E_{relative} = \frac{S_{relative}}{P} \tag{5.2}$$

where P is the number of processing elements involved in the computation. The two above parameters most clearly determine the gain from applying parallel processing to solve a given computational problem.

Another important reason for applying parallel processing is the possibility of distributing problem data, which may not fit into memory of a single, sequential computer, across a large number of local memories of the processing elements. In this way, thanks to parallel processing, one may apply a given algorithm to more complicated, 'large scale' problems of numerical modeling.

The essential factors which affect performance of a solver in a parallel system and may be directly controlled during design of a numerical algorithm include:

- *parallel domain decomposition* (or *parallel mapping*), determining which part of data (e.g. of matrix elements) will be stored by which processing element and which parts of data (often related to geometrical subdomains of the problem domain) will be processed by specific processors (cf. Figure 5.1),
- *data and computation locality*, which determines what computations may be performed by each processing element (PE) using its own data, without the necessity to refer to data stored by other PE.
- *inter-processor communication*, determining the scheme and amount of data to be exchanged between the processors in order to complete the computations,
- *load balancing* which refers to the differences in the amount of processing performed by each of the PEs involved in the computation. Clearly, the more balanced is the workload among the processors, the better overall performance may be achieved.

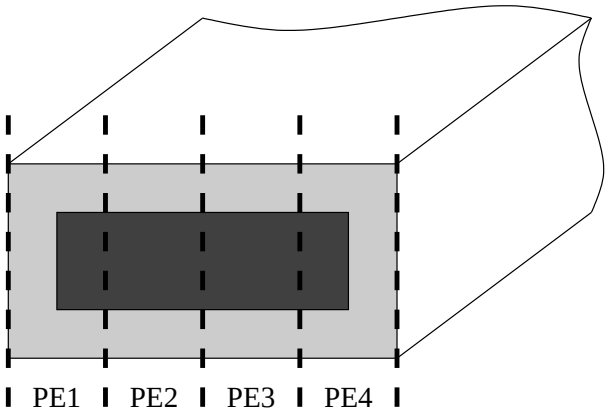


Figure 5.1: A sample parallel decomposition of a computational domain across processing elements (PEs) for a problem of modeling of a dielectric waveguide.

Designs of the numerical methods performing various matrix calculations, such as computing the matrix-vector product, matrix-matrix product or deriving matrix transposition, are determined primarily by the format of the input matrix operator. Application of these algorithms in the environment with multiple processing elements (PEs) requires developing suitable parallel mapping techniques of both data and computations to the processors in order to achieve the main goal of parallel processing, i.e. minimization of the total execution (wall-clock) time. Although these parallel mapping techniques certainly depend on the representation of the input matrix operators and the specifics of the matrix computations to be parallelized, the basic two strategies will certainly be applied: 1) Place the computational tasks on different processing elements in order to enhance concurrency, 2) Place the computational tasks which make use of the same data on the same processor to increase the locality.

These strategies may sometimes turn out very conflicting which would require tradeoffs in design of the parallel mapping techniques. At the same time, an inadequate exploitation of any of these strategies will usually reduce or even eliminate the gain in performance of the numerical algorithms implemented in a parallel environment. This fact is particularly true for scalable parallel systems where often a large number of processing elements is involved in the computations.

Let us consider the simplest, still up to now the most important parallel mapping technique, i.e. the static domain decomposition technique. In this mapping method all the data (e.g. matrix or vector elements or a grid in the spatial domain) as well as computational tasks are distributed among the processing elements in an fixed manner. In the method the properties of the domain being decomposed determine whether a computational task may be efficiently mapped to the available processing elements. Concentrating on the techniques of distributing matrix operators let us discuss the specifics of parallel decomposition for some classes of matrices:

1. *Matrices with a block structure.* In this case an ideal locality of data and computations (within a single PE or a group of PEs) may be achieved if all the elements of a given block in the matrix are local to a single processor or a group of processors. The most favorable case occurs if the number blocks of the matrix is a multiple of the number of PEs and all the blocks have equal sizes. Then all data (e.g. necessary to perform the matrix transposition) may be stored locally and the amount of computations may be perfectly balanced across the processors. The problems with balancing the computations will occur if the sizes of the blocks are not equal and/or the number of processors does not correspond directly to the number of the matrix blocks. In this case, the assignment of matrix blocks to PEs has to take into account the numerical complexity of the operations performed on each block in order to obtain balancing of the workload. (Still, the workload balancing may cause an imbalance in the local storage requirements.) The question that emerges is: Which operators may be discretized to produce an operator matrix with a block structure? The first group of such operators are scalar operators acting on multidimensional vector fields. Separating the field components in a finite-dimensional representation may give a block structure of the resulting matrix. The other group of operators may be defined as operators modeling short-range, local interactions in a number of disjoint subsystems. Applying e.g. the Finite Difference (FD) discretization may then result in a block-structured matrix or a banded matrix. [43]

2. *Banded matrices* also have a very favorable structure while investigating their parallel distribution using domain decomposition method. In most cases the amount of non-local data which is used by the processing elements is of order $O(b^2)$ or $O(b)$ (depending on the mapping and computational task), where b is the matrix bandwidth. If the bandwidth is small relatively to the matrix size then the majority of necessary data is stored locally by each PE and most of the computations involve only local data. Banded matrices are frequently obtained by using the Finite Difference (FD) discretization scheme. The FD technique has also the advantage of producing a highly regular matrix with an even distribution of its non-zero elements. This has a very positive impact on workload balancing which may be easily achieved by applying regular domain decomposition. Figure 5.2 presents a matrix obtained from discretizing the 2D Laplace operator with FD technique. The Figure also shows that one may easily control the matrix bandwidth by changing the ordering of the grid (the ordering of matrix rows and columns). The importance of bandwidth is also illustrated in Figure 5.3 which shows speedup for parallel computation of the matrix-vector product for a matrix constructed from FD discretization of 2D Laplace's operator. The matrix of size $N = 40000$ had highly regular structure and the bandwidth of 200. If we apply an inter-processor communication scheme, which does *not* take into account the banded character of the matrix, the speedup is very poor. (This is the speedup we should expect to obtain if a naive parallelization strategy is applied for computations involving non-banded matrices). The speedup improves considerably if we use an optimized inter-processor communication scheme, exploiting the structure of the matrix. The details of the

mentioned schemes are discussed in Section 5.3.

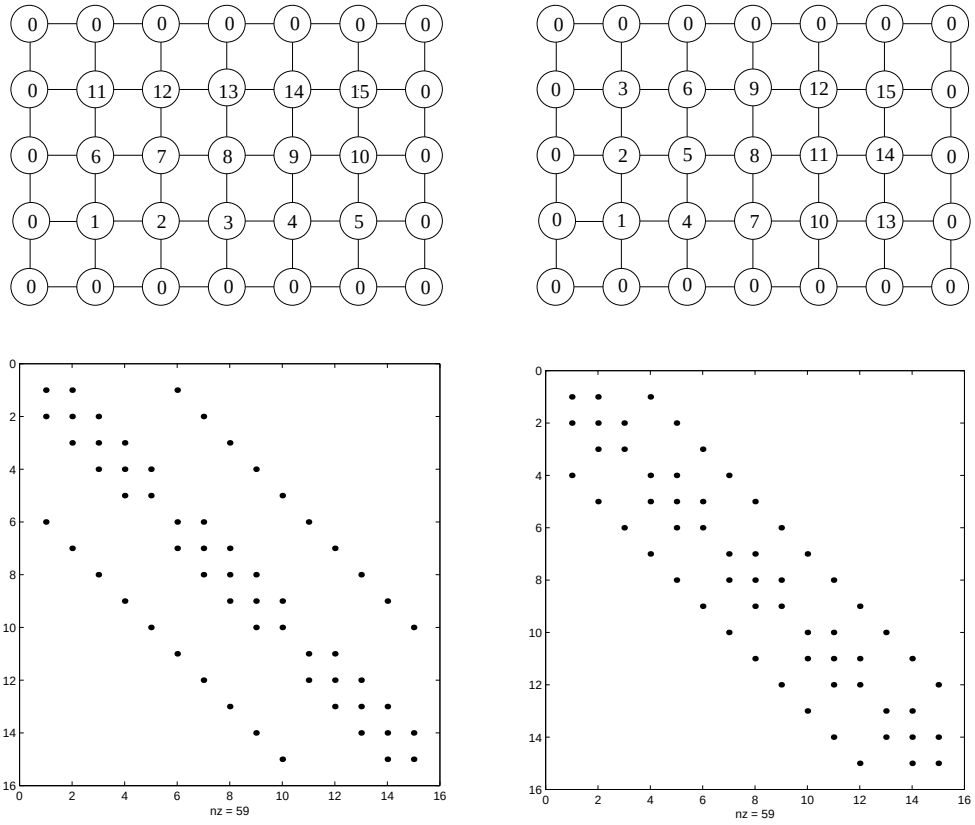


Figure 5.2: Reducing the bandwidth of a discrete 2D Laplace operator by changing the grid ordering.

3. *Sparse, non-banded matrices* are the class of matrices which may be encountered if the Finite Element Method (FEM) is used to discretize the operator's domain. Figure 5.4 shows a typical form of a matrix obtained using the FEM. It presents the matrix for a 2D Laplace operator on a coarse mesh. Although, as seen in the figure, the matrix is sparse, the irregular distribution of their non-zero elements may result in problems while seeking an efficient parallel mapping using static domain decomposition. The first problem is that potentially large amount of non-local data has to be used by each processing element in order to perform parallel matrix operations. One of the solutions to this situation is designing specific procedures of accessing or communicating non-local data in order to avoid bottlenecks and reduce the number of non-local data accesses. The other problem is the irregular non-zero element distribution which may cause an imbalance in the workload across the PEs. Summing up, in the case of sparse, non-banded matrices the static parallel domain decomposition schemes may turn out unsuitable if high performance in a scalable parallel execution environment is to be achieved (cf. the previous example – Figure 5.3).

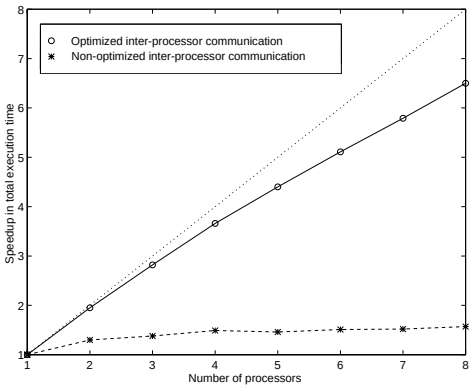


Figure 5.3: Speedup in the parallel execution of a matrix-vector product, involving a Finite Difference sparse matrix for non-optimized and optimized inter-processor communication. The matrix size equaled $N = 40000$ and its bandwidth equaled 200. The tests have been performed in the IBM SP2 parallel system.

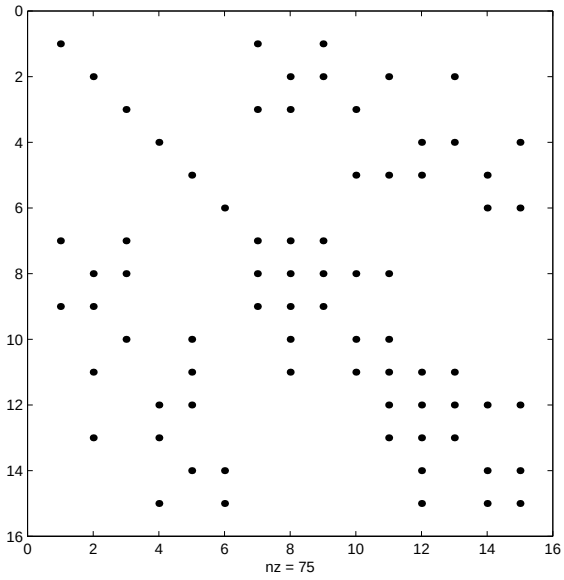


Figure 5.4: A matrix obtained from discretization of the 2D Laplace operator using the Finite Element Method on a coarse mesh.

4. *Dense matrices* appear when entire domain or entire subdomain expansion discretization techniques are used. (The example of such technique – the Method of Moments representation will be described in one of the following sections of this chapter.) The parallel decomposition of dense matrices may potentially result in very large amount of non-local data which has to be accessed by the processing elements while performing such operations as e.g. matrix transposition. There is

usually little that can be done to avoid a great deal of computations involving non-local data. Still, in order to maintain high level of parallel performance one may increase the computation time involving solely local data as compared to the time spent on accessing or using non-local data by applying appropriate scaling of the problem size. Unfortunately this cannot be done if the complexity of operations involving non-local data is higher than the numerical cost of the local computations. The positive feature while dealing with dense matrices is that the workload balance may be achieved by applying a simple regular domain mapping scheme.

Summing up, the characteristics of different types of matrices obtained in various methods of finite-dimensional mapping of linear operators may affect favorably or adversely the performance of parallel algorithms involving operating on distributed matrices. Within the limits of the static domain decomposition parallel mapping techniques the positive features of matrices to be distributed include block structure, sparsity of the matrix, relatively narrow matrix bandwidth, while the negative ones include irregular non-zero element distribution in sparse matrices or dense non-zero element packing. Some of these negative factors may even preclude the static domain decomposition technique if an efficient parallelization of a given computational problem is to be achieved. In this case different parallel mapping techniques have to be applied. At this point the following mapping schemes may be mentioned:

- *load balancing algorithms* which include: *probabilistic load balancing* or *cyclic mapping* – the static methods which exploit structure of the computations and data to distribute the domain of computations and may be used e.g. to problems involving matrices with an irregular distribution of non-zero elements or irregular distribution of computations; *dynamic load balancing* in which the parallel mappings change during execution of the algorithm – this method may be applied e.g. in the multigrid algorithms (cf. [10], [85]).
- *task scheduling algorithms* which explore the potential for *functional* parallel decomposition of the computational tasks and may be applied to obtain parallel mapping of problems with FEM-based discretization, multigrid approach, etc.

A much broader discussion of parallel mapping strategies with various case study presentations may be found in a book by Foster [43] or the materials from the Edinburgh Parallel Computing Centre [76] (in which mainly static domain decomposition techniques are described).

This section presented general issues concerning parallel mapping techniques of discrete matrix operators and related computational tasks. In the above description, some potential problems occurring during parallel mapping of different classes of matrices obtained during projection of linear operators were discussed. In the above approach we tried to answer whether a suitable parallel mapping may be found for a given type of matrix. Still, these general guidelines may be applied in a somewhat inverse approach. This second approach consists of exploring the possible parallel mapping techniques for a given

parallel system architecture *before* selecting projection and finite representation scheme for a given input linear operator. In this way the finite-dimensional representations of operators which will not fit any efficient parallel mapping technique may be immediately excluded.

With this general conclusions and guidelines one may now turn to discussion of more specific and to a certain extent technical issues related to parallel design of selected numerical methods.

5.2 Parallel design of the Arnoldi factorization

This section discusses parallel design features of the basic Arnoldi factorization proposed in this library by Maschhoff and Sorensen ([69]). This design has been applied to develop P_ARPACK (Parallel Arnoldi Package) library (described in Appendix B), which is a portable collection of routines providing parallel implementations of the Implicitly Restarted Arnoldi Method (IRAM).

If, once again, the formula for the Arnoldi factorization is examined:

$$\underline{\underline{A}}\,\underline{\underline{V}}_k = \underline{\underline{V}}_k \underline{\underline{H}}_k + \underline{f}_k \underline{e}_k^T \tag{5.3}$$

where the symbols have the same meaning as in Section 3.2.1, then the parallelization scheme illustrated in Figure 5.5 may be described as follows:

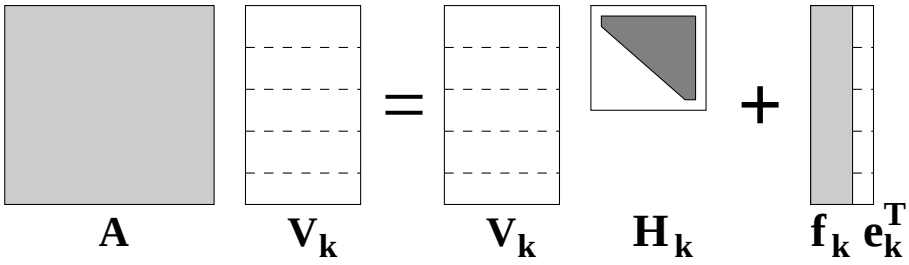


Figure 5.5: Parallel block data distribution during the Arnoldi factorization.

- the $k \times k$ upper Hessenberg matrix $\underline{\underline{H}}_k$ is replicated on every processor,
- the matrix $\underline{\underline{V}}_k$ is block-distributed across a one-dimensional processor grid,
- $\underline{f}_k$ and workspace are distributed accordingly,
- parallel data distribution in the input matrix $\underline{\underline{A}}$ is chosen by the user. Still, as the outcome of the matrix-vector product has to be distributed analogously as the matrix $\underline{\underline{V}}_k$, the decomposition of the matrix will typically be commensurate with this block distribution.

According to the conclusions obtained in Section 3.2.1.3, the memory storage requirements for the applied data distribution equal $n_{loc}O(l) + O(l^2)$ per processor, where $n_{loc} \approx n/P$ (P equals the number of processors) and $l = k + p$ equals the sum of the number of eigenvalues to be found and to be filtered-out.

A crucial aspect of parallel implementation in distributed memory systems is the size of messages communicated between the processors during the execution of the algorithm. Referring to P_ARPACK (cf. Appendix B) and Arnoldi factorization there are only two communication points. One of them is computation of the norm of the distributed vector $\underline{f}_k$ and the other is the orthogonalization of $\underline{f}_k$ to $\underline{\underline{V}}_k$ using the MGS algorithm, where the global scalar products of a given vector with the columns of the matrix $\underline{\underline{V}}_k$ have to be computed. In the MPI implementation these global norms and sums are calculated using a global reduction procedure `MPI_Allreduce(.)`. For a single iteration in the Arnoldi factorization, the overall size of elements communicated across the processors is extremely low and is of order $O(Pk)$, where P denotes the number of processors and k equals the number of eigenvalues to be found.

A certain kind of trade-off may be observed in the parallelization strategy applied in the IRAM iteration. As all the operations on the upper Hessenberg matrix $\underline{\underline{H}}_k$ are replicated on each processor, the communication of the results is not needed. Nevertheless, this introduces some redundancy to the algorithm that may lead to a serial bottleneck as the size k of the matrix increases. This may eventually cause the lack of scalability of the method.

According to the results obtained in Section 3.2.1.3 the numerical complexity of the parallel version of a single update in a p-step IRAM algorithm equals $O(p^2 n_{loc})$ or (if $p = O(k)$) $O(k^2 n_{loc})$ per processor, where n_{loc} is the local dimension of the problem.

Clearly, in the above estimations the costs of performing the parallel matrix-vector operation and storing the operator matrix, which largely depend on the choice of finite-dimensional mapping method, have been excluded. This problem will be addressed in the following sections.

5.3 Parallelization of the FDFD methods

This section presents implementations of two parallel eigensolvers based on the Implicitly Restarted Arnoldi Method (IRAM), for operators discretized using the Finite Difference Frequency Domain method. First of the solvers implements the operator problem discussed in 4.1.1, applicable to modeling waveguiding structures, while the other one implements the operator eigenproblem defined for 3D electromagnetic systems possessing rotational symmetry (cf. Section 4.1.2).

The implementations are based on calling the P_ARPACK library routines (cf. Appendix B) which perform the IRAM iteration. Consequently, according to the description

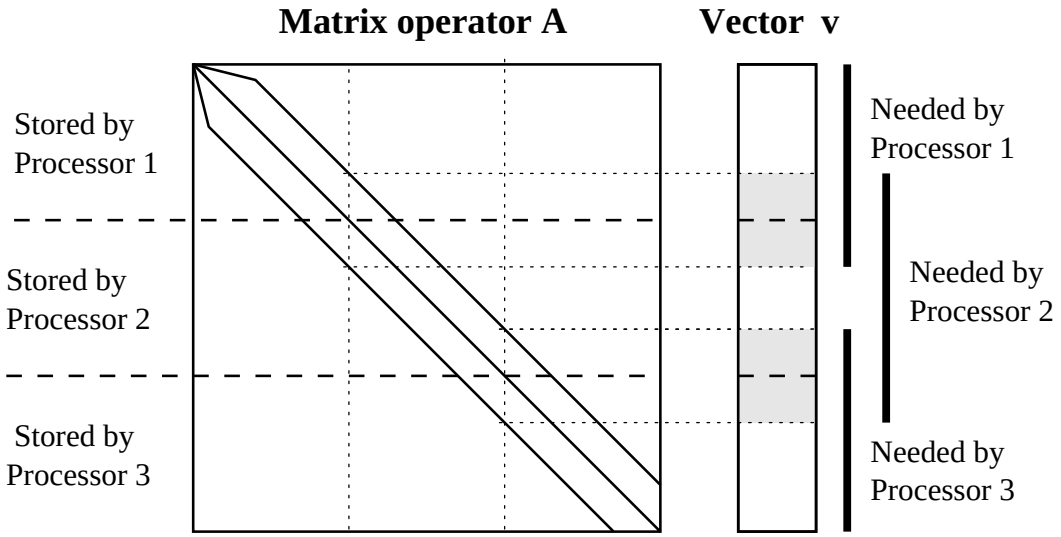


Figure 5.6: Schematic of a block distribution among the processors of a quasi five-diagonal sparse matrix and the corresponding vector. The figure shows that in order to calculate the matrix-vector product with such distribution the grayed parts of the vector $\underline{v}$ need to be communicated between the processors (processes).

of P_ARPACK from Appendix B, the implementation of the solver follows the *reverse communication* scheme shown in Figure A.1. Therefore, the implementation of the solver may concentrate on only two aspects: 1) Defining parallel data distribution, which includes distribution of the vectors and the discrete operator and 2) Implementing the parallel operation of matrix-vector product which corresponds to the applied parallel data distribution.

5.3.1 Implementation of the matrix-vector product in parallel

This section concentrates on describing the parallel implementation of the matrix-vector product involving discrete operator used in modeling 2D waveguiding structures. As discussed in Section 4.1.1 the matrix obtained in the FDFD mapping applied is a highly sparse matrix with very regular structure. Consequently, a simple parallel block data distribution scheme may be applied. In this distribution each of the processors (processes) stores a specific range of rows of the operator matrix and a corresponding range of elements of the input vectors. This has been illustrated in Figure 5.6.

The matrix presented in the Figure shows the discretized differential operator discussed in Section 4.1.1. With most of the non-zero elements located on five diagonals, the presented distribution minimizes the inter-processor communication necessary to compute the matrix-vector product. The regions of the input vector $\underline{v}$ which have to be communicated between the pairs of neighboring processors have been shown in the Figure as grayed regions. In our example, as the matrix (and the vector) size equals 39700, the

number of the vector elements to be communicated between each pair is approximately 400, which equals the doubled number of discretization points in the x direction. This fact is explained, if one recalls the ordering of field samples in the input vector:

$$\underline{H} = \left[H^x_{1,1}, H^y_{1,1}, \ldots, H^x_{N_x,1}, H^y_{N_x,1}, \ldots, H^x_{N_x,N_y}, H^y_{N_x,N_y}\right]^T \tag{5.4}$$

where N_x and N_y are the number of discretization points in x and y direction respectively. Then the finite (central) difference operator corresponding to the first derivative in the y direction is reflected in the operator matrix by off-diagonal elements located at a distance of 400 elements from the main diagonal. Note that ordering:

$$\underline{H} = \left[H^x_{1,1}, \ldots, H^x_{N_x,N_y}, H^y_{1,1}, \ldots, H^y_{N_x,N_y}\right]^T \tag{5.5}$$

is unsuitable in the context of parallel processing, as due to the coupling of both field components in the initial operator it would result in a matrix with a dramatically increased bandwidth, which would imply significantly increased inter-processor communication costs.

If an explicit matrix storage is used then, in order to reduce the memory storage cost of the matrix, the following 'hybrid' type of storage is applied. The five diagonals (containing 95% of the non-zero elements) are stored separately in the $5 \times n$ matrix and the remaining 5% of the elements are stored in the Compressed Sparse Row (CSR) format. (The irregularities in the structure of the matrix are due to boundary conditions – cf. Section 4.1.) Assuming that $5n \approx 0.95nnz$, where n is the matrix size and nnz is the number of non-zero matrix elements, the resulting storage requirements equal approximately $0.95nnz + 2 \cdot 0.05nnz + n + 1 = 1.05nnz + n + 1$ elements. In a more sophisticated approach the implicit matrix representation is applied, which does not require storage of matrix elements. Still, even if stored implicitly, the discrete operator retains its properties e.g. its bandwidth.

As already mentioned, in order to complete the operation of the matrix-vector product with each of the processing elements computing the local part of the outcome some non-local elements of the input vector need to be communicated. The number of vector elements to be communicated equals the bandwidth of the matrix (understood as the maximum difference between the row indices for non-zero elements located in the same column). It the example this bandwidth equals 400. Consequently, only 400 elements have to be communicated as to enable every processor to compute its part of the matrix-vector product (cf. Figure 5.6). In consequence, no collective communication routines are necessary and a simple two-step inter-processor communication scheme shown in Figure 5.7 may be used. This figure shows how the necessary parts of the input vector $\underline{v}$ are communicated using a series of simple blocking send and receive procedures. In our example the overall size of the data communicated between the processors equals approximately $(P-1) \cdot 0.01n$ elements, as $400 = 0.01 \cdot 40000 \approx 0.01n$. This means the size of communication equals one-hundredth of the vector size. Clearly, the presented

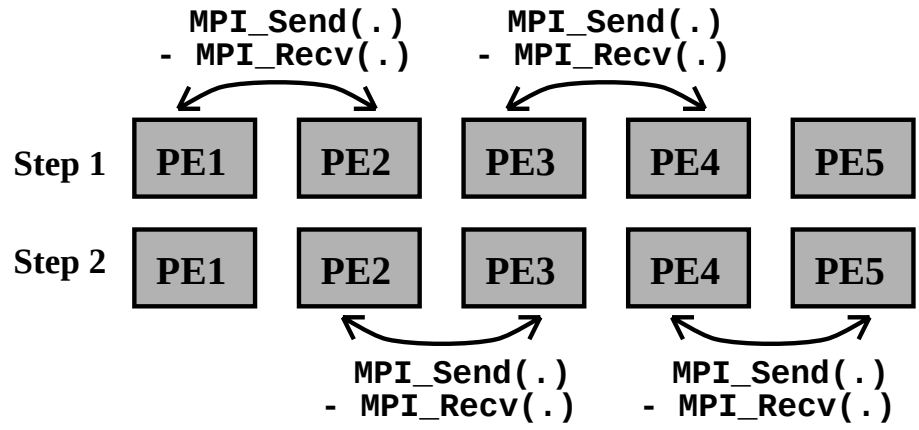


Figure 5.7: Two-step communication scheme used in the parallel matrix-vector product calculation.

scheme of communication may be applied to arbitrary banded matrices provided that $b < (2 \cdot n/P)$, where b is the bandwidth of the matrix of size n, block-distributed among P processors. If this condition is not satisfied more complicated schemes have to be applied involving not only pairs of neighboring processors.

5.3.2 FDFD operator matrix customized for parallel data distribution

The previous section presented a parallel design of the matrix-vector product in the case of the matrix obtained with the FDFD discretization method. The description referred to the discrete operator obtained from differential operator given by (2.42) used for modeling 2D waveguiding structures. The second of the parallel eigensolvers using IRAM and FDFD method implements the operator problem defined by equations (A.16) and (A.17) which may be used to model 3D systems with rotational symmetry and for which boundary conditions are commensurate with the grid defined by the cylindrical coordinate system.

The parallel design of the matrix-vector product is entirely analogous as in the case of the previously discussed solver, including the inter-processor communication scheme shown in Figure 5.7. This is due to a similar pattern of distribution of non-zero elements of the operator matrix. The left picture in Figure 5.8 shows an example of a matrix obtained by discretizing operator $\mathbf{S}$ given by equation (2.52).

In this case, the form of the matrix may be further customized as to improve the regularity of the distribution of the matrix non-zero elements and, in consequence, enhance the balancing of parallel data distribution and facilitate parallel computation.

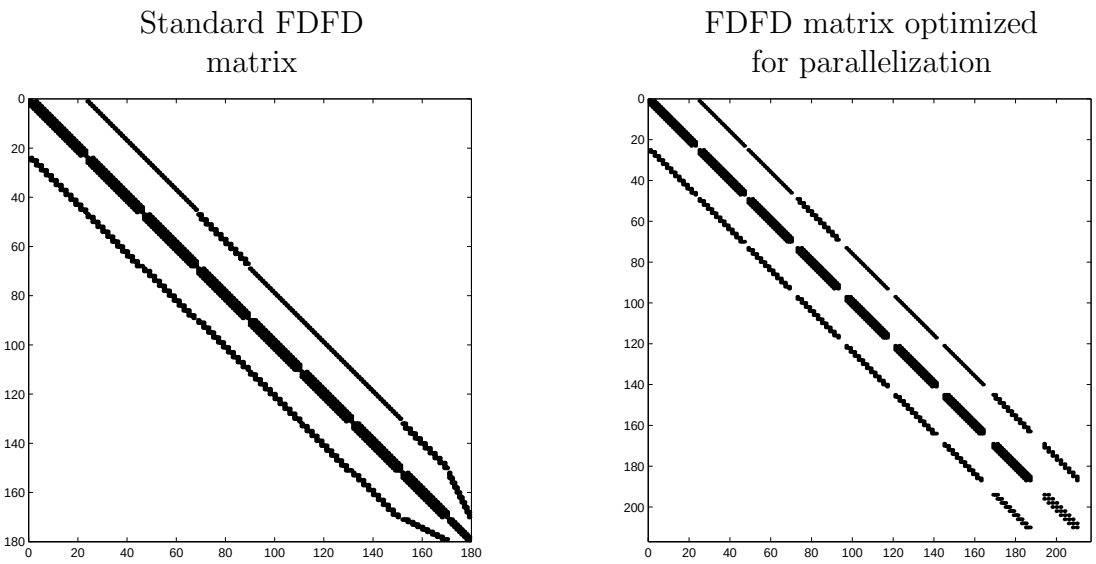


Figure 5.8: Modification of the initial operator matrix by inserting all-zero rows and columns.

One may note, that due to boundary conditions the regular 11-diagonal structure of the matrix (observed for rows corresponding to grid points located far from the boundary) is spoiled by the applied e.g. homogeneous Dirichlet boundary conditions. There are still 11 non-zero elements in a row, but they are located at various distances from the main diagonal (– a tapered end appears). To overcome this problem one may apply the idea based on inserting into the initial matrix rows and corresponding columns containing only the zero elements. The inserted elements correspond to the points in the discretization grid zeroed due to the applied boundary conditions. In the initial approach the elements are excluded from the discrete representation of field, while in the modified approach they are included in this representation. The main consequence of applying the mentioned modification is the further regularization of the matrix structure. Recalling the example of 2D Laplace operator from Figure 4.5 one notes that the matrix rows and columns corresponding to the zeroed grid points are removed. In the proposed approach, instead of excluding the rows and columns they are simply replaced by all-zero rows and columns. One has to note that by inserting all-zero rows and columns zero eigenvalues are introduced to the operator spectrum. In general, this may deteriorate convergence of the Krylov subspace methods (cf. [100]). Still, the numerical experiments show that for the considered class of problems, the performance of the Implicitly Restarted Arnoldi Method (IRAM) is not affected if we apply the discussed technique of matrix regularization.

Figure 5.8 shows the form of the initial matrix (on the left) and the modified matrix with inserted zero rows and columns (on the right). While the bandwidth of both matrices is the same, the number of non-zero matrix diagonals approaches the value of bandwidth for the first case and equals exactly 11 for the second matrix. This situation occurs for

any size of the given matrix.

It has also to be stressed that in the case of operators and boundary conditions which are more sophisticated than the basic 2D discrete Laplace's operator with homogeneous Dirichlet boundary conditions the simple insertion of zero rows and columns is no longer a valid approach. Instead, the appropriate boundary conditions are imposed during construction of the discrete operator. This may be conveniently performed within the implicit projection scheme, which allows one to force given homogeneous/inhomogeneous Dirichlet/Neumann conditions on appropriate elements of the input vector during the matrix-vector product operation. For instance, if we consider the operators $\underline{\underline{HzrotE\phi}}$ and $\underline{\underline{HzrotEr}}$ from the left hand side of equation (4.21) we apply the desired boundary conditions by e.g. zeroing operator entries corresponding to elements located on the top and side boundary walls, respectively. So, in each of the 'partial' operators the appropriate boundary conditions are imposed separately (i.e. no vector elements are globally zeroed while performing the matrix-vector product).

Clearly, if the information on the discrete operator is passed to the solver only via the matrix-vector product (as in the case of Krylov subspace methods) this implicit scheme may be applied during numerical solution and, consequently, the appropriate conditions may be imposed 'on the fly'. One recalls that including vector elements corresponding to points located on the boundary, where Dirichlet condition is imposed is redundant are causes a slight increment in the size of the problem being solved. Nevertheless, a significantly simpler parallel algorithm of computing the matrix-vector product emerges. First of all, the sequential part of the algorithm executed by each of the processors is simplified. Secondly, in this case only 10 vector elements (instead of the entire bandwidth of elements) need to be communicated between each pair of neighboring processing elements. This trade-off allows one to obtain higher performance of the numerical solver.

5.4 Parallel design of methods using eigenfunction expansion techniques

This section presents a design of parallel algorithm which uses the projection technique based on eigenfunction expansion, presented in Section 4.2 in order to solve the given operator eigenproblem using the Implicitly Restarted Arnoldi Method, implemented in the P_ARPACK library (cf. Appendix B). The salient feature of the applied projection is that the operator is represented implicitly, resulting in reduced storage requirements and allowing much more efficient implementation of the matrix-vector product operation. In fact, as shown below due to this implicit representation the projection of the operator is embedded in the eigenproblem solution process.

In the same way as in the previous section, the implementation of the solver is based on the reverse communication scheme (presented in Figure A.1) in which calls to P_ARPACK

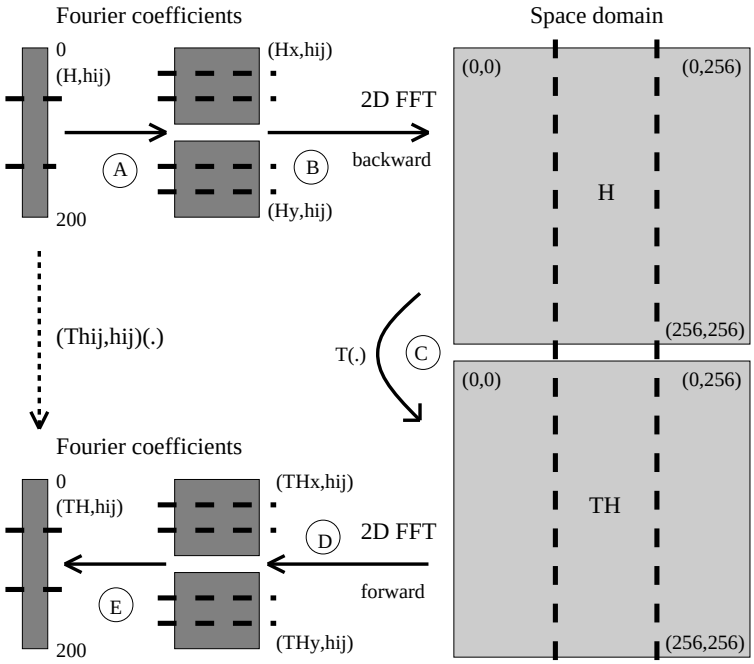


Figure 5.9: Schematic of parallel data distribution in matrix-vector product design for the DFT-based operator and function representation. The dashed lines mark the block data distribution pattern among the processors. Calculation of the matrix-vector product is performed in a series of steps: A – rearranging a 1D vector in the form of two 2D matrices, B – performing two parallel 2D backward FFTs, C – performing linear transformation **T** of the two fields in the space domain, D – performing two parallel 2D forward FFTs, E – rearranging the fields in the form of a 1D vector. Compare also Section 4.2.2.

library routines (mainly the `pdnaupd()` routine) performing the Arnoldi factorization are followed by calls to user-supplied routines calculating the matrix-vector product.

The following section presents the parallel implementation of the matrix-vector product jointly with the description of the parallel distribution of the elements of the input vector.

5.4.1 Parallel implementation of the matrix-vector product using two-dimensional Fast Fourier Transform

Assuming that the domain of the given linear operator **T** is the space of 2D vector fields $\vec{H} = (H^x, H^y)$ where $H^x, H^y \in L_2([0,b] \times [0,a])$ the following representation for the functions in this domain has been defined in Section 4.2.1:

$$\begin{aligned} \underline{H} &= [c^x_{11}, c^y_{11}, c^x_{12}, c^y_{12}, \ldots, c^x_{mn}, c^y_{mn}] \\ &= [(H^x, h^x_{11}), (H^y, h^y_{11}), (H^x, h^x_{12}), (H^y, h^y_{12}), \ldots (H^x, h^x_{mn}), (H^y, h^y_{mn})] \end{aligned} \quad (5.6)$$

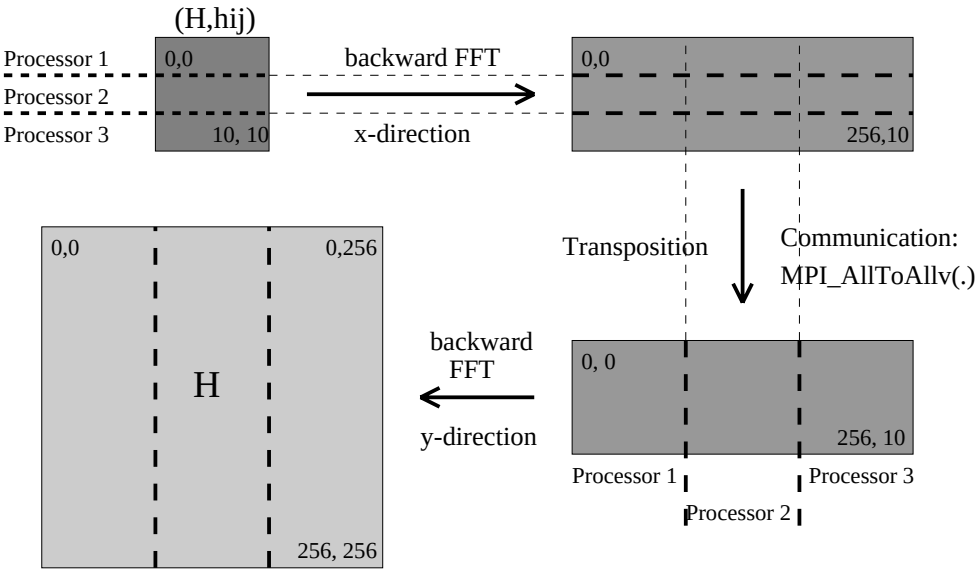


Figure 5.10: Idea of the parallel backward two-dimensional FFT algorithm design. The scheme of performing a forward 2D FFT is entirely analogous. In the above example the field components are represented in the transform domain by 10 Fourier coefficients in each direction. Due to applied oversampling these field components are represented on a 256×256 grid in the spatial domain.

where $\underline{H}$ is a finite representation for the vector field $\vec{H} = (H^x, H^y)$, c_{ij}^x and c_{kl}^y are Fourier coefficients defined by appropriate inner products and $\{h_{ij}^x\}$ and $\{h_{kl}^y\}$ form orthonormal bases in the $L_2([0,b] \times [0,a])$ functional space.

As described in Section 4.2.2, calculating the matrix-vector product in the case of the discussed representation may be performed using an efficient method which dramatically reduces the computational cost of this operation, as compared to the classical approach used in the Galerkin Method (GM). In this unorthodox approach the operation of calculating matrix-vector product involves three steps: 1) calculating the backward 2D FFTs, 2) calculating the $\mathbf{T}\vec{H}$ product in the spatial two-dimensional domain and 3) calculating forward 2D FFTs. This has been illustrated in Figure 5.9. This Figure also shows the main idea of parallelization of this matrix-vector product, which is based on block distribution of the input elements of the vector $\underline{H}$, given by the equation (5.6). In other words, each processor stores a range of rows of the matrices of coefficients $[c_{ij}^x]$ and $[c_{ij}^y]$. The number of rows stored by each processors is balanced, as to assure a similar workload for all the processors. After completing the computation of the matrix-vector product each processor stores the same range of rows of the Fourier coefficient matrices for the $\mathbf{T}\vec{H}$ field.

In Figure 5.9, it may also be noted that after computing the two-dimensional backward FFTs, the elements of the matrices H^x and H^y are block distributed by columns and not

by rows. This is the effect of the parallel design of the two-dimensional FFT algorithm. Let us look in more detail at the parallel algorithm of computing the backward two-dimensional FFT. The schematic of this operation has been shown in Figure 5.10. The computation involves three steps:

1. As the matrices (from which only one was shown for simplicity) of the Fourier coefficients are distributed by rows, each processor computes a backward one-dimensional FFT in the x-direction for a locally stored range of rows.

2. In order to perform the backward one-dimensional FFT in the y-direction the processors need to have access to a full range of coefficients from specified columns. Consequently, a parallel transposition of the distributed matrices obtained after completing the backward FFTs in the x-direction has to be performed. This operation involves mainly the inter-processor communication, as each processor has to send $(P-1)$ blocks of the locally stored part of the matrix and has to receive also $(P-1)$ different blocks from other processors. In the MPI implementation of the solver this operation may be performed by using a high-level collective communication routine `MPI_Alltoall(.)` (or `MPI_Alltoallv(.)` for non-equal sizes of the transmitted matrix blocks) which sends from all the processors to all the processors the specified blocks of data. Clearly, this operation may also be performed by using simple send and receive operations by scheduling these operations appropriately. Still, if the high-level message-passing routine is applied, the programming complexity is passed to the library implementation. Another advantage of such an approach is that we may achieve better performance if a native implementation of the MPI library, which optimizes collective communication routines for a specific interconnection network topology, is applied in a given testing platform. In the actual implementation this approach has been successfully used, producing a highly efficient parallel routine as shown in Chapter 7.

3. After the transposition each processor computes a one-dimensional FFT in the y-direction for a locally stored range of columns.

This completes the parallel operation of computing the two-dimensional FFT. One may ask whether the elements of the output matrices should be block distributed among the processors by rows rather than by columns. The answer is negative. The main reason is that there is no need to perform an extra transposition operation (which involves a considerable amount of inter-processor communication) in order to obtain a parallel block distribution by columns. As may be seen from Figure 5.9, after computing the backward 2D FFT and performing the $\mathbf{T}\vec{H}$ operation a forward 2D FFT is performed. During the forward 2D FFT the parallel block distribution by rows is restored. The forward FFT involves analogous steps as those shown in Figure 5.10, namely: 1) Computing one-dimensional FFTs in the y-direction, 2) Performing a parallel matrix-transposition using the `MPI_Alltoallv(.)` routine, 3) Computing one-dimensional FFTs in the x-direction.

By reversing the order of computing one-dimensional FFTs, two unnecessary (and costly) transposition operations are avoided. The numerical tests performed by the author

comparing the two versions of the algorithm for computing the matrix-vector product – the one described above and the older serial implementation which performed additional transpositions (applied e.g. in [78]) show that for a single-processor execution the first algorithm was by about 30% faster than the second serial algorithm. Even the overheads due to initiating the MPI communication and additional computations needed to establish the parallel data distribution scheme did not prevent the parallel algorithm from running faster on one processor than the second algorithm. This fact implies that the execution times of the solver, given in [78] may be further reduced by up to 30%.

So far nothing has been told about the operation $\mathbf{T}\vec{H}$ performed in the spatial domain during the matrix-vector product. This step is entirely dependent on the form of the operator $\mathbf{T}$. Still, in many applications this operation may be completed in a linear time with respect to $N = N_x \cdot N_y$, where N_x and N_y denote the FFT lengths in the x- and y-directions respectively. More details are given in the next chapter, presenting applications of the proposed eigensolvers.

5.4.2 Numerical and memory complexity of the method

In this section the numerical and memory complexity of a single p-step update of the IRAM algorithm involving the FFT-based matrix-vector product will be investigated. Applying the results from Section 3.2.1 and Chapter 3, Section 4.2 one may estimate the overall memory storage needed by the parallel solver as the sum of the storage needed by the IRAM procedure $((N/P)\cdot O(k)+O(k^2))$ and the memory required in the matrix-vector product computation $(O(2K/P + 2N/P + 6\sqrt{K}))$, where P is the number of processors, k is the number of eigenvalues to be found ($p = O(k)$), $K = K_x \cdot K_y$, where K_x and K_y denote the FFT lengths in the x- and y-direction respectively and N is the problem size. ($N = N_x \cdot N_y$, where N_x and N_y denote the number of expansion functions used to represent the functions in the respective spatial dimensions.)

The numerical complexity of a single update involves the time cost of performing the Arnoldi factorizations $(O(p^2N/P))$ and the cost of computing p matrix-vector products which equals $O(pK/P \log K)$. The overall cost is given by the formula:

$$O((p^2N + pK \log K)/P) = O((k^2N + kK \log K)/P) \tag{5.7}$$

with all the symbols having the same meaning as above.

Another aspect which has to be addressed is the size of messages communicated in the algorithm which in this case is dominated by the size of the messages communicated during the matrix-vector product computation. In a single matrix-vector product the communication occurs during the two transposition operations. The size of the communicated data equals $O((K_xN_y + K_yN_x)(P-1)/P)$ elements. Consequently in a k-step IRAM algorithm the communication size equals:

$$O(p(K_xN_y + K_yN_x)(P-1)/P) = O(p\sqrt{KN}(P-1)P) \tag{5.8}$$

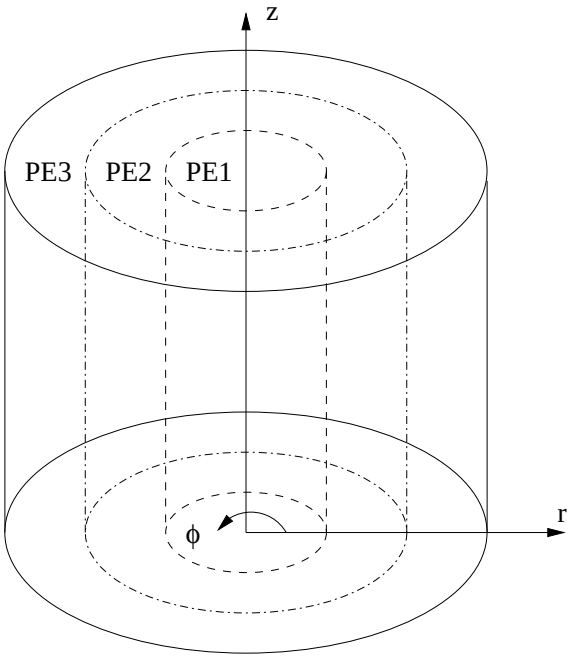


Figure 5.11: Parallel data distribution applied to a cylindrical domain in the IRAM-based hybrid eigensolver.

assuming that $K_y = O(K_x)$ and $N_y = O(N_x)$.

5.5 Parallel design of the IRAM eigensolver exploiting the hybrid projection method in 3D space

Once again the Implicitly Restarted Arnoldi Method (IRAM) implemented in the parallel P_ARPACK library is used to solve the problem for the discrete operator developed in Section 4.3.1. The projection method applied to construct the implicitly represented operator is a hybrid technique based on FD discretization in the $r - z$ plane and eigenfunction expansion in the ϕ direction. The parallel distribution of the problem domain is shown in Figure 5.11. Each of the processing elements (PEs) stores data enclosed in a cylindrical ring: $[r_i, r_{i+1}] \times [0, 2\pi] \times [z_{\min}, z_{\max}]$. If the number of discretization points in the r, ϕ and z directions equals respectively N_r, N_ϕ and N_z then the local size of the problem for each processing elements equals roughly $N_z \cdot N_\phi \cdot N_{rloc}$, where $N_{rloc} \approx N_r/P$ and P is the number of processors.

The proposed parallel data distribution implies that very little inter-processor communication is required to compute the matrix vector product $\underline{\underline{S}}\,\underline{v}$ ($\underline{\underline{S}}$ is the operator developed in Section 4.3.1). Unlike in the case of the algorithm presented in the previous section where computation of 2D FFTs involved serious communication, the computation of 1D Fast Fourier Transforms in the ϕ direction for the currently discussed method does

not require any inter-processor communication at all. Also in the case of applying finite difference operator in the z directions (referring to derivatives in the z direction) to vector elements no communication is required. The only operation when the inter-processor communication is required is applying the finite difference operators in the r direction (corresponding to derivatives in the r direction) to the vector elements. Consequently, N_z elements have to be communicated between each pair of the neighboring processors. The overall communication size equals $P \cdot N_z$ which is considerably less than the problem size $N = N_z \cdot N_\phi \cdot N_r$.

The memory and computational cost for the algorithm may be easily assessed if the results from Section 4.3.1 are exploited. The memory cost equals $5N_{loc} \approx 5N/P$ and the computational complexity equals $O(N_{loc} \log_2(N_\phi)) = O(N \log_2(N_\phi)/P)$.

Chapter 6

Application and validation of the algorithms

The previous chapters discussed a number of key factors determining the computational cost and performance of methods of solving operator boundary value problems in the context of large scale numerical modeling. Different requirements, to be satisfied by the numerical techniques, have been pointed out and according to them a number of low cost numerical algorithms have been proposed. This chapter aims at constructing numerical methods for solving selected 'real life' electromagnetic problems. Before we turn to describing specific numerical procedures, capable of solving certain electromagnetic boundary value problems, we should define what are the general performance characteristics of the solvers we want to obtain. These characteristics may be summarized in terms of an 'ideal algorithm'.

An 'ideal numerical algorithm' for solving electromagnetic boundary value problems can be characterized by the following features:

- Linear ($O(N)$) memory and computational complexity.
- As small as possible problem size N.
- Accuracy adequate for a given electromagnetic application.
- Good scalability in parallel distributed memory systems.

According to the conclusions drawn in all the previous chapters, the 'ideal algorithm' is clearly the algorithm which is best suited for large scale electromagnetic modeling, reaching high, scalable performance if applied to complicated problems of computational electromagnetics.

Below we construct numerical solvers composed of different 'bits' already developed in Chapters 2, 3, 4 and 5, i.e. we apply derived operator formulations, presented eigenproblem solution algorithms and proposed projection techniques, together with discussed parallelization strategies to arrive with numerical techniques suitable for solving specified problems.

We will try to answer whether and/or to what extent the designed techniques meet the characteristics of an 'ideal algorithm' defined above. This will also show whether the developed techniques are adequate to be used in large scale electromagnetic modeling. Below we present the results of performed numerical tests which validate the proposed solvers. The discussion of parallel performance is postponed until Chapter 7.

6.1 Modeling waveguiding structures

This section focuses on presenting application of the proposed numerical techniques to finding modes in dielectric waveguides. The operator eigenproblems for waveguiding structures to be used in this part of the work have been formulated in Section 2.1.1. They provide examples which generate medium-size computational problems, as compared to problems presented in Section 6.2. Nevertheless, even for these relatively simple modeling examples, advantages of the proposed algorithms over some classical methods may be observed (cf. Section 7.4). The common feature of these eigenproblems is that the operator eigenvalues are squared propagation constants and the corresponding eigenfunctions are transverse magnetic or electric field intensities. The discussion below concentrates on the formulations involving transverse magnetic field, still it applies to entirely analogous formulation for transverse electric field.

The waveguiding structures used to validate the numerical algorithms are shown in Figure 6.1 and include a slab waveguide (A), image guides (B, C) with discontinuous permittivity profiles as well as an open structure of an elliptical guide (D) with a continuous permittivity profile $\epsilon(x, y)$.

6.1.1 Application of the IRAM-FDFD solver

The first of the operators to be considered is a vector non-symmetric differential operator derived in Section 2.1.1 (cf. equation (2.42)):

$$\mathbf{T}(\cdot) = \nabla_t^2(\cdot) + k_0^2\epsilon(x,y)(\cdot) + \frac{1}{\epsilon(x,y)}\left[\nabla_t\epsilon(x,y) \times (\nabla_t \times (\cdot))\right] \tag{6.1}$$

where $\nabla_t(\cdot) = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)(\cdot)$, k_0 is the wavenumber in the free space and $\epsilon(x, y)$ is the permittivity profile of a waveguide in the $x-y$ plane. The domain of the operator is defined as the space of 2D vector fields $\vec{H} = (H^x, H^y)$, where H^x and H^y are square integrable functions defined over a bounded rectangular region containing the cross-section of the examined waveguiding structure. In the presented formulation, the eigenfunctions of the operator $\mathbf{T}$ correspond to the transverse magnetic field and its eigenvalues correspond to squared propagation constants β^2.

If operator (6.1) is treated with the FD discretization method, then the IRAM-FDFD solver, described in Sections 4.1.1 and 5.3 may be applied. The numerical tests validating

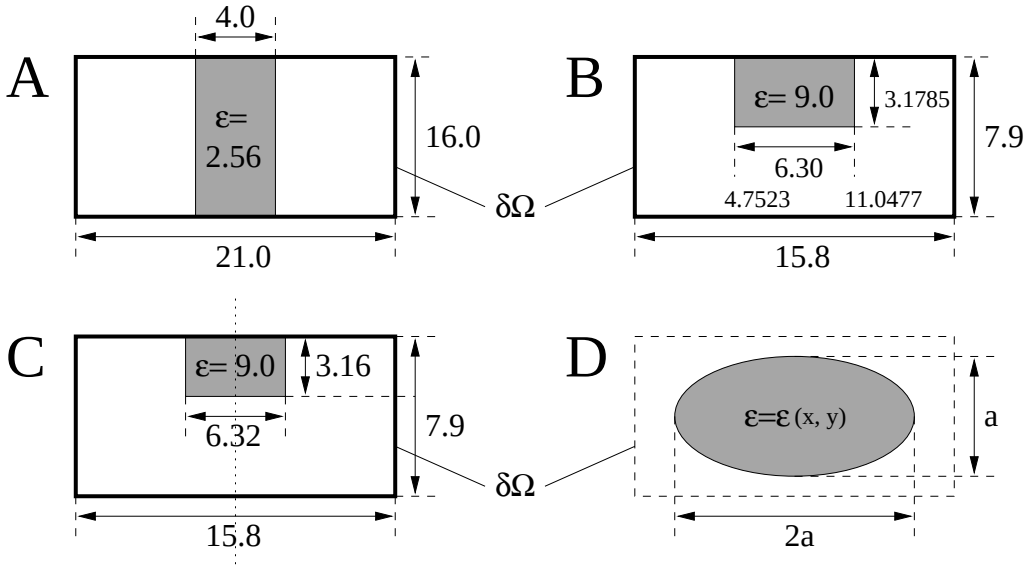


Figure 6.1: Cross-sections of the waveguiding structures used during validation tests. All dimensions are given in millimeters. At the boundaries $\delta\Omega$ the Perfect Electric Conductor (PEC) conditions are assumed – cf. Section 2.1.

the IRAM-FDFD algorithm have been performed for rectangular waveguides with discontinuous permittivity profiles. For these problems the PEC (Perfect Electric Conductor) conditions have been applied at the boundary $\delta\Omega$ (cf. Section 2.1). Table 6.1 shows the normalized propagation constants β/k_0 for an image guide with a discontinuous permittivity profile (structure C in Figure 6.1), computed using the IRAM-FDFD solver. The values are compared to those computed using a classical algorithm based on the dense matrix constructed with the Galerkin Method (GM) whose eigenvalues are found with the QR method [50]. (The algorithm is referred in the table as GM-QR.)

Although the IRAM-FDFD and GM-QR algorithms apply entirely different strategies for operator discretization and different methods of computing eigenvalues the errors for eigenvalues obtained by the IRAM-FDFD method stay at an acceptable level. In fact, we do not know which propagation constants are computed with greater accuracy, as we do not know whether e.g. the 200×100 FD discretization grid provides a more accurate finite representation of the input operator than the 20×20 matrix of Fourier coefficients, constructed by the Galerkin Method, also representing the same differential operator. In the considered case the problem solved by the IRAM-FDFD algorithm equaled $N = 39750 \approx 2 \times 200 \times 100$.

6.1.2 Application of the basic IRAM-FFT solver

The other finite-dimensional representation of the operator (6.1) which may be used is based on the Method of Moments (eigenfunction expansion). This approach has been

IRAM-FDFD (200×100)	GM-QR 20×20	Relative error [%]
2.3058	2.3110	0.23
1.5840	1.5947	0.67
0.49398+0.32770j	0.51314+0.27998j	1.41
0.49398j-0.32770j	0.51314-0.27998j	1.41
1.1297j	1.1278j	0.17
1.2639j	1.2571j	0.54

Table 6.1: Comparison of the normalized propagation constants β/k_0 computed using the IRAM-FDFD algorithm and the QR method with the matrix constructed using the Galerkin Method. The structure used in the tests was an image guide (structure C in Figure 6.1). Other test parameters: f=15 GHz, NEV=8 (number of eigenvalues to be computed), NCV=40 (size of the constructed Krylov subspace) (IRAM-FDFD). The relative error was computed using the formula: $E = 100|\beta_{IRAM-FDFD} - \beta_{GM-QR}|/|\beta_{GM-QR}|$.

widely discussed in Section 4.2 and the solver implementing the method applying implicit operator projection (below referred to as IRAM-FFT algorithm) has been described in Section 5.4.

The first class of applications of the IRAM-FFT algorithm to be considered are waveguiding structures with continuous (and infinitely smooth) permittivity profiles $\epsilon(x, y)$. In this case the gradient of $\epsilon(x, y)$ appearing in the definition of the operator (6.1) has a standard mathematical meaning and consequently computation of Fourier integrals (inner products) has also a standard functional meaning and may be performed by applying the numerical algorithm defined in Section 4.2.

During the validation tests, the structure modeled with the basic IRAM-FFT algorithm was an elliptical waveguide with continuous permittivity profile (structure D in Figure 6.1). The tested structure was an elliptical waveguide with the semi-axes ratio $a_x/a_y = 2/1$ and the permittivity profile given by the function:

$$\epsilon(x, y) = \epsilon_0 \epsilon \left[1 - \left((x/2a)^2 + (y/a)^2\right)^{\alpha/2}\right] \tag{6.2}$$

where α is the permittivity profile exponent.

Non-radiative modes in an open structure were modeled by taking the screening walls sufficiently far away from the guide (at the distance of $20a_x$ from the center of the waveguide). The results presented in Table 6.2 show a comparison of the propagation constants (for different profile exponents α) computed using Iterative Eigenfunction Expansion Method (IEEM) [78] and obtained by the author using the IRAM-FFT method. The table shows non-dimensional normalized propagation constants Z computed with

α	IEEM [78]	IRAM-FFT	Difference [%]
2	0.4894	0.4907	0.27
4	0.6254	0.6258	0.06
6	0.6740	0.6742	0.03
8	0.6976	0.6978	0.03
10	0.7112	0.7114	0.03

Table 6.2: A comparison of the normalized propagation constants Z computed in the IEEM method and the IRAM-FFT algorithms for the fundamental mode in the elliptical waveguide with a continuous permittivity profile (structure D in Figure 6.1), for different permittivity profile exponents α (cf. formula 6.2). In the tests: $V = 3$, NEV=1, NCV=20, FFT length=256, number of expansion function used to represent functions = 75 (in every spatial direction).

the following formula:

$$Z = \frac{\beta^2/k_0^2 - 1}{\epsilon - 1} \tag{6.3}$$

In the tests the normalized frequency V, given by formula: $V = k_0 \cdot 2a_x \cdot \sqrt{\epsilon - 1}$ equaled 3, the number of expansion functions equaled 75 in x and y directions and the respective FFT lengths equaled 256. Consequently, the size of the problem solved by the IRAM algorithm equaled $N = 2 \cdot 75^2 = 11250$, while in the spatial domain the grid consisted of $2 \cdot 256^2 = 131072$ points. It is apparent that the obtained results agree almost perfectly, confirming that the IRAM-FFT algorithm may be successfully applied to deal with the discussed class of structures.

6.1.3 Coping with discontinuous permittivity profiles in the DFT representation.

As pointed out in the previous sections, if the eigenfunction expansion representation is to be applied it is necessary to find out whether this representation is suitable for all the operators in the form given by Equation (6.1). As described in the previous chapters, in the finite-dimensional mapping technique based on eigenfunction expansion the operator is represented by certain inner products – the Fourier coefficients, such as $(\mathbf{T}h_{ij}^x, h_{kl}^y)$ (cf. Section 4.2). These coefficients are in fact 2D definite integrals whose values are computed numerically by using the Discrete Fourier Transform (DFT). Using the DFT we calculate approximate values of these integrals using a regular grid of samples of the integrated 2D function. The numerical error in the integration depends on the class of the integrated function. If the form of the operator $\mathbf{T}$ given by formula (6.1) is examined, one may note the term $\nabla_t \epsilon(x, y)$. If the permittivity profile is a C^1 class function then the operation $\mathbf{T}$ on an arbitrary function from the C^2 class results in a continuous function. In this case, as shown in the previous paragraphs, the DFT-based representation of the operator provides an adequate finite approximation of the operator.

TRM [78]	IEEM-FFT [95]	GM-QR	IRAM-FFT
1.2353	1.2352	1.2344	1.2339
1.0833	1.0756	1.0813	1.0818
1.0648	1.0622	1.0648	1.0641
0.41570	0.41563	0.41146	0.41412

Table 6.3: Comparison of the normalized propagation constants β/k_0 calculated for a few low order modes in a slab guide (structure A in Figure 6.1) with a discontinuous permittivity profile. In the tests the frequency f equaled 15 GHz, the number of expansion functions used equaled 10 in the x (horizontal) direction, the FFT length equaled 256, NEV=4, NCV = 20 (IRAM-FFT - cf. the list of symbols).

If the permittivity profile is a discontinuous function the situation is very different. Now, the $\nabla\epsilon$ term may correctly be considered only within the theory of distributions. Consequently, a modified scheme of computing Fourier integrals should be applied instead of the basic algorithm presented in Section 4.2.2. Before presenting this modified technique, applied to a waveguiding structure with a rectangular core, one more example will be given. The example presented below shows that for simple waveguiding structures the basic IRAM-FFT may still be used.

The structure to be modeled is a slab waveguide (cf. structure A in Figure 6.1). In this case the value of gradient of a discontinuous permittivity profile at the grid points close to discontinuity was simply approximated by the value computed with the standard central difference scheme. (So, for the grid points located closest to the discontinuity, $\partial\epsilon/\partial x$ was approximated by $(\epsilon - 1)/2\Delta x$.) Table 6.3 shows the values of normalized propagation constants β/k_0 computed for this structure with the basic IRAM-FFT algorithm. The results are compared to the corresponding eigenvalues computed with three different algorithms: 1) the Transverse Resonance Method (TRM) which is regarded as one of the most accurate algorithms for finding propagation constants, suitable while dealing with certain relatively simple waveguiding structures; 2) the Galerkin Method (GM-QR) in which the operator is represented explicitly by the appropriate inner products (Fourier integrals) computed analytically and the eigenvalues are computed with the QR method; 3) the Iterative Eigenfunction Expansion Method using FFT integration (IEEM-FFT – cf. [78], [95]) using eigenfunction expansion technique and a specific iterative process in order to find operator eigenvalues. Although the number of expansion functions used in discrete representations of functions and operators is very small and equals 10 in the x (horizontal) direction (for IEEM-FFT, IRAM-FFT and GM-QR algorithms) a very good convergence of results is obtained. It may easily be found that the relative differences between the corresponding eigenvalues do not exceed 1%. This is probably because there is a low contrast between permittivities inside and outside the waveguide core ($\epsilon/\epsilon_0 = 2.5$).

A different series of tests performed for the same simple waveguiding structure compared the values of normalized propagation constants computed with the Galerkin Method

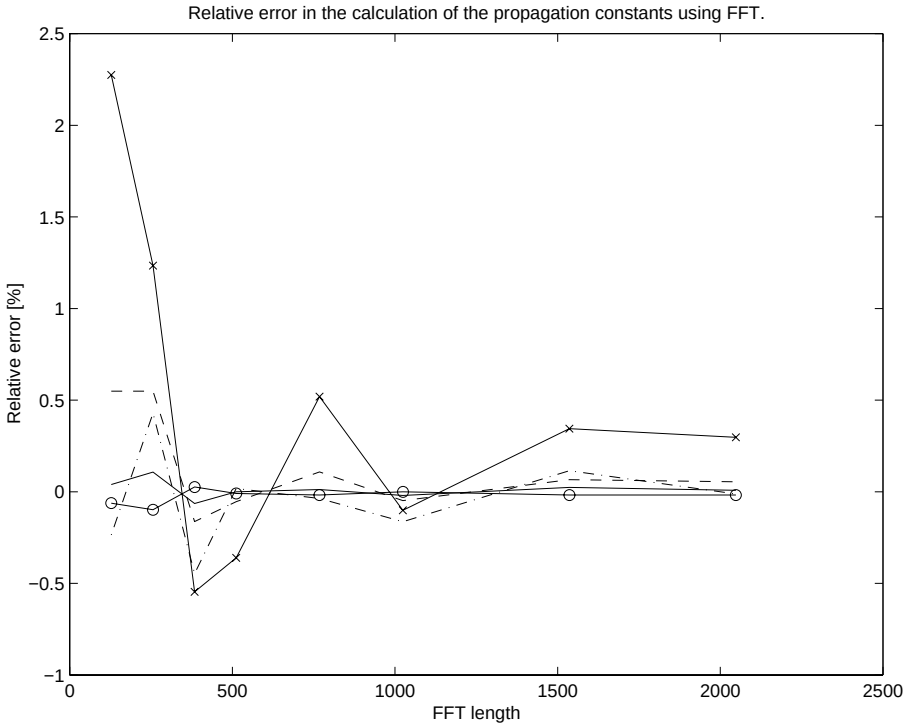


Figure 6.2: Relative difference in the values of normalized propagation constants β/k_0 computed for a slab guide (structure A in Figure 6.1) using the IRAM-FFT solver and the Galerkin Method (GM) as a function the length of the FFT applied in the IRAM-FFT algorithm. During the tests the frequency f equaled 15 GHz, the number of expansion functions used equaled 10 (in the x direction), NEV=5, NCV=20 (IRAM-FFT). The errors were computed for the first 5 eigenvalues found by the methods.

(GM-QR) and the IRAM-FFT algorithm for different lengths of the Fourier Transforms, applied in the IRAM-FFT method and ranging from 128 to 2048. The results of these tests are shown in Figure 6.2. As one could expect the relative differences between the computed eigenvalues become smaller as the FFT length increases. This means that the approximations of inner products computed using the FFT algorithm approach the values of the inner products computed analytically in the Galerkin Method with application of a more refined discretization grid (determined by the FFT length).

The favorable situation described above changes if a more complex waveguiding structure is considered, e.g. an image guide with a discontinuous permittivity profile (structure C in Figure 6.1) for which the contrast between permittivities inside and outside the waveguide core is larger ($\epsilon/\epsilon_0 = 9.0$) than in the former case. In this case, substantial problems appear with the simple IRAM-FFT algorithm. The eigenvalues found by the IRAM-FFT algorithm for the vector operator $\mathbf{T}$ (compare equation 6.1) differ very significantly (by 10-20%) from the corresponding eigenvalues found using the Galerkin Method [95]. The situation may improve considerably if a modification of the IRAM-FFT

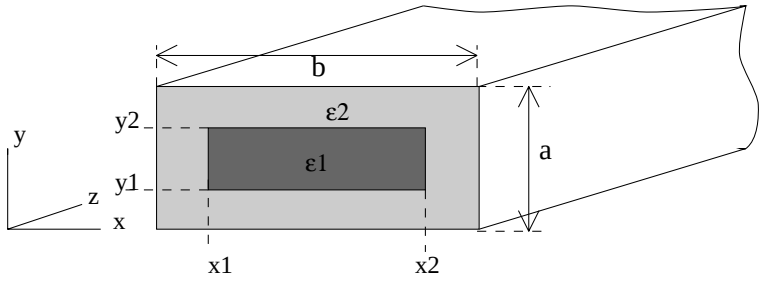


Figure 6.3: Schematic of a dielectric waveguide with a rectangular, discontinuous permittivity profile $\epsilon(x, y)$.

algorithm described below is applied.

6.1.3.1 Application of numerical integration

The modification of the basic IRAM-FFT algorithm is presented for the class of dielectric waveguides with one or more rectangular cores. For simplicity of the considerations it is assumed here that the geometry of the system is as shown in Figure 6.3. The permittivity profile $\epsilon(x, y)$ for the structure shown in the figure is a discontinuous function which is given by the following formula:

$$\epsilon(x, y) = (\epsilon - 1)h(x_2 - x)h(x - x_1)h(y_2 - y)h(y - y_1) + 1 \tag{6.4}$$

where $h(x)$ denotes the Heaviside function. Only the derivatives in a generalized sense exist for $\epsilon(x, y)$:

$$\frac{\partial}{\partial x}\epsilon(x, y) = (\epsilon - 1)h(y_2 - y)h(y - y_1)\left(\delta(x - x_1) - \delta(x_2 - x)\right) \tag{6.5}$$

where $\delta(x)$ denotes the Dirac distribution.

Obviously applying 'sampling' to a distribution is impossible. Consequently calculating the Fourier integrals by using the Discrete Fourier Transform does not have any correct mathematical meaning in this case, which results in severe numerical errors which are indeed observed in many applications (including the currently discussed one) if this kind of approach is applied.

The solution which may be proposed is a modification of the solvers which use the discussed implicit representation of operators (e.g. IRAM-FFT). The modification refers to the method of calculating the matrix-vector product which implicitly contains the form of the operator. The proposed method is a hybrid algorithm which uses both DFT (FFT) and numerical integration to calculate the matrix-vector product. The method starts with decomposing the initial operator $\mathbf{T}$ given by the equation (6.1):

$$\mathbf{T} = \mathbf{L} - \mathbf{F} \tag{6.6}$$

where:

$$\mathbf{L}(\cdot) = \nabla_t^2(\cdot) + k_0^2\epsilon(x,y)(\cdot) \tag{6.7}$$

$$\mathbf{F}(\cdot) = \frac{1}{\epsilon(x,y)}\left[\nabla_t\epsilon(x,y) \times (\nabla_t \times (\cdot))\right] \tag{6.8}$$

Denoting as $\mathbf{F_x}$ and $\mathbf{F_y}$ the projections of the vector operator $\mathbf{F}$ into x- and y- directions, the inner products $(\mathbf{F_x}\vec{H}_t, h_{ij}^x)$ and $(\mathbf{F_y}\vec{H}_t, h_{kl}^y)$ for the structure shown in Figure 6.3 are given by the following 1D integrals:

$$\left(\mathbf{F_x}\vec{H}_t, h_{ij}^x\right) = \int_{x_1}^{x_2} \frac{2(\epsilon-1)}{(\epsilon+1)}\left[\left(\nabla_t\vec{H}_t(x,y)\right)h_{ij}^x(x,y)\right]_{y=y_2}^{y=y_1} dx \tag{6.9}$$

$$\left(\mathbf{F_y}\vec{H}_t, h_{kl}^y\right) = \int_{y_1}^{y_2} \frac{2(\epsilon-1)}{(\epsilon+1)}\left[\left(\nabla_t\vec{H}_t(x,y)\right)h_{kl}^y(x,y)\right]_{x=x_2}^{x=x_1} dy \tag{6.10}$$

where the term $2(\epsilon-1)/(\epsilon+1)$ is obtained while integrating the permittivity profile, under the assumption that the Heaviside function is given by the formula:

$$h(x) = \begin{cases} 0 & x < 0 \\ 0.5 & x = 0 \\ 1 & x > 0 \end{cases} \tag{6.11}$$

The above linear integrals may be computed using any standard method of numerical integration. If we denote by $\mathbf{L_x}$ and $\mathbf{L_y}$ the obvious projections of the operator $\mathbf{L}$ (cf. equation (6.7)) then the steps of the hybrid algorithm calculating the inner products $\left(\mathbf{T_x}\vec{H}_t, h_{ij}^x\right)$ and $\left(\mathbf{T_y}\vec{H}_t, h_{kl}^y\right)$ may be given as follows:

1. Given the Fourier coefficients $\{c_{ij}^x\}$ and $\{c_{kl}^y\}$ compute the values of the vector field $\vec{H}_t = (H^x, H^y)$ in the spatial domain by applying backward 2D FFTs.

2. Apply numerical integration (NI) to compute the following inner products:

$$g_{ij}^x = (\mathbf{F_x}\vec{H}_t, h_{ij}^x) \quad g_{kl}^y = (\mathbf{F_y}\vec{H}_t, h_{kl}^y)$$

3. Derive the Fourier coefficients $(\mathbf{L_x}H_x, h_{ij}^x)$ i $(\mathbf{L_y}H_y, h_{kl}^y)$ using the 2D FFT algorithm.

4. Compute the final Fourier coefficients:

$$\left(\mathbf{T_x}\vec{H}_t, h_{ij}^x\right) = (\mathbf{L_x}H_x, h_{ij}^x) + g_{ij}^x$$

$$\left(\mathbf{T_y}\vec{H}_t, h_{kl}^y\right) = (\mathbf{L_y}H_y, h_{kl}^y) + g_{kl}^y$$

where $\mathbf{T_x}$ and $\mathbf{T_y}$ denote the projections of the initial operator $\mathbf{T}$ onto x- and y- dimensions.

GM-QR 20×20	IRAM-FFT-NI 20×20 FFT length 2048	Relative error [%]
2.3110	2.3132	0.10
1.5947	1.5951	0.03
0.85624	0.85142	0.56
0.51314+0.27998j	0.51403+0.27986j	0.16
0.51314-0.27998j	0.51403-0.27986j	0.16
0.69140j	0.70192j	1.52
1.1278j	1.1276j	0.02
1.2571j	1.2568j	0.03

Table 6.4: Comparison of the normalized propagation constants β/k_0 for several low order modes, computed using the modified IRAM-FFT algorithm (IRAM-FFT-NI) and the Galerkin Method. The structure used in the tests was an image guide (structure C in Figure 6.1). Other test parameters: f=15 GHz, NEV=8, NCV=40. The relative error was computed using the formula: $E = 100|\beta_{IRAM} - \beta_{GM}|/|\beta_{GM}|$.

This algorithm is validated by computing the propagation constants for the image guide shown on picture C in Figure 6.1. Application of the basic IRAM-FFT procedure for this case resulted in unacceptable numerical errors. Table 6.4 and Figure 6.4 show the results for the modified algorithm (IRAM-FFT-NI) compared to eigenvalues computed using the GM-QR algorithm. One may note that the computed propagation constants stay very close to each other (especially for lower-order modes). The results confirm the usefulness of the investigated IRAM-FFT-NI algorithm in modeling structures with discontinuous rectangular permittivity profiles, such as the tested image guide (cf. structure C in Figure 6.1).

Apart from the obvious advantage of being able to deal with discontinuous permittivity profiles, the above algorithm has also a very substantial drawback of increasing the computational complexity of the matrix-vector product algorithm by $O(\sqrt{K}N)$ (in the worst case this means the increment of the complexity to $O(K^{3/2})$, as compared to $O(K \log K)$), where K is the product of the DFT lengths in the x- and y-dimensions and N is the problem size (the product of the number of expansion terms used to approximate functions in the x- and y-dimensions). This drawback may be eliminated by applying FFT integration instead of numerical integration at a cost of imposing certain restrictions on the geometry of the modeled system. This issue is addressed in the next section.

6.1.3.2 Application of 1D FFT integration

If one takes a look at the formulae (6.9) and (6.10) it becomes clear that, because h^x_{ij} and h^y_{kl} are composed of trigonometric functions, the integrals appearing on the right hand side of these formulae may be computed using one dimensional sine and cosine DFTs

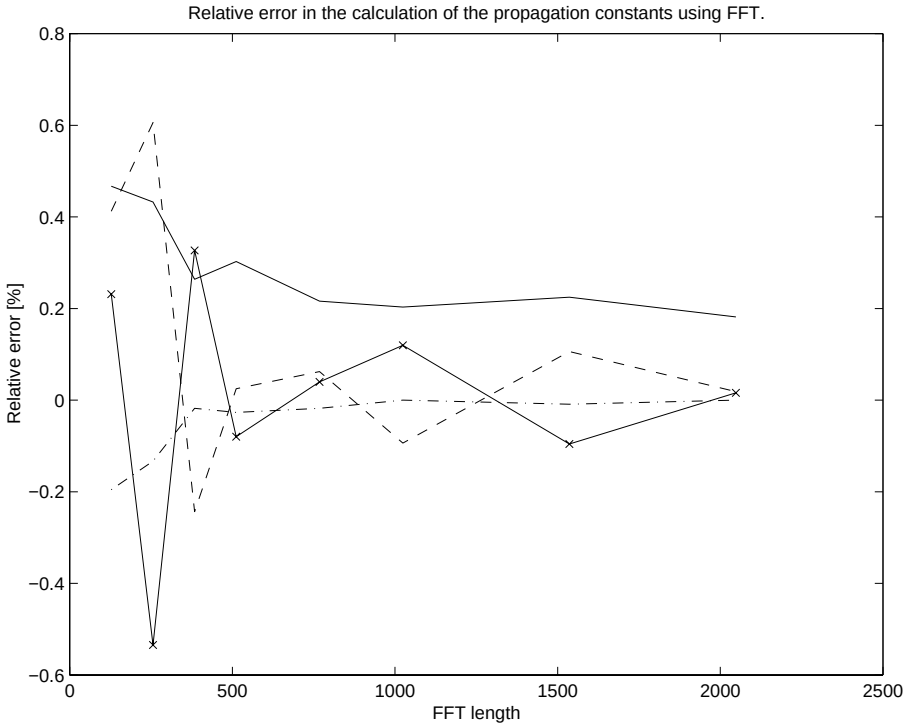


Figure 6.4: Comparison of selected real normalized propagation constants β/k_0 computed using the modified IRAM-FFT algorithm (IRAM-FFT-NI) and the Galerkin Method for different FFT lengths applied. The structure used in the tests was an image guide (structure C in Figure 6.1). Other test parameters: f=15 GHz, NEV=8, NCV=40. The relative error was computed using the formula: $E = 100(\beta_{IRAM} - \beta_{GM})/(\beta_{GM})$.

(e.g. Fast Fourier Transforms). Still, this may be done only if the grid points are located *exactly* along the discontinuity, i.e. the values of fields are defined for points located on the material discontinuity. In other words the material discontinuities as well as structure boundaries must be aligned with the discretization grid in the spatial domain. (If this is not the case the DFTs may still be used, but computation of the discussed integrals should then involve interpolation procedures necessary to compute field values at the discontinuity.)

The algorithm of computing the matrix-vector product in this case is the same as in the previous section, except Step 2 where the values of g_{ij}^x and g_{kl}^y are computed by applying FFT integration instead of standard numerical integration. The consequence of this fact is that the overall numerical complexity of the algorithm is linear-logarithmic, i.e. equals $O(K\log(K))$ as compared to the computational complexity of $O(K^{3/2})$ for the previous version of the algorithm. This in turn, has a positive impact on the performance of the solver.

IRAM-FFT^2 40×20	IRAM-FFT-NI 40×20	GM-QR 20×20	Relative difference [%]
2.3209	2.3209	2.3140	0.30
1.6039	1.6039	1.5970	0.44
0.85224	0.85224	0.86023	0.93
0.53456+0.28183j	0.53456+0.28183j	0.51740+0.27472j	3.18
0.53456-0.28183j	0.53456-0.28183j	0.51740-0.27472j	3.18
0.70161j	0.70161j	0.69346j	1.18
1.1263j	1.1263j	1.1277j	0.13
1.2588j	1.2588j	1.2589j	0.01

Table 6.5: Comparison of the computed normalized propagation constants β/k_0 for three different algorithms: IRAM-FFT^2, IRAM-FFT-NI and GM-QR. For the first two algorithms the FFT length equaled 256 in every spatial direction. The propagation constants have been computed for the frequency $f = 15$ GHz.

Table 6.5 shows propagation constants computed for the waveguiding structure B from Figure 6.1. This structure is a slightly modified structure C, in which the permittivity discontinuities have been moved to the closest grid line (i.e. each of the boundaries of the core is shifted by at most $\Delta x/2$ or $\Delta y/2$, where $\Delta x = a/N_x$ and $\Delta y = b/N_y$ ($a = 15.8$ mm, $b = 7.9$ mm, N_x and N_y are FFT length in the x and y directions respectively). The results are obtained using the two modified IRAM-FFT algorithms: the first, using the FFT 1D integration (IRAM-FFT^2) and the second using the numerical integration (IRAM-FFT-NI). The computed propagation constants are identical to numerical precision. The results are compared to the values computed with the GM-QR algorithm.

Table 6.6 shows a relative performance of the two algorithms while solving the problem discussed above. One may note that although the number of implicit updates and matrix-vector products is the same in both cases, the $IRAM-FFT^2$ solver is significantly faster than the $IRAM - FFT - NI$ solver. The computation of the matrix-vector product is almost three times faster for the first solver. This provides a rationale for applying FFT integration instead of numerical integration method.

6.1.3.3 Application of different operator formulations

This section describes a different approach in order to cope with the discontinuities of the permittivity profiles than presented in the two above paragraphs. The main difficulty which appeared in the previously presented methods was associated with the initial definition of the electromagnetic differential operator and consisted in the necessity of computing $\nabla\epsilon$, i.e. a gradient of a discontinuous permittivity profile. (cf. equation (6.1)) Consequently, this gradient could not have been computed using finite difference approximation. Instead a correction involving computation of an integral along the discontinuity had to be introduced. The correction was associated with either increasing the numerical

Integration method at the discontinuity	Number of implicit restarts	Number of $\mathbf{A}v$ operations	Total $\mathbf{A}v$ time [s]	Total time [s]
$IRAM - FFT^2$	25	758	66.18	76.59
$IRAM - FFT - NI$	25	758	184.89	194.69

Table 6.6: Comparison of the execution time of the two modified IRAM-FFT algorithms using different integration algorithms at the discontinuity of permittivity profile. The tests have been performed in the IBM SP2 system.

complexity of the algorithm (in the case of the method using numerical integration) or restricting the allowed geometries of a waveguiding structure (in the case of the method using FFT integration).

The problem of modeling discontinuities in algorithms using implicit operator projection may be approached using a different strategy. The main idea behind it is to avoid direct computation of permittivity gradients in the spatial domain, by applying a different operator formulation. This formulation (derived in Section 2.1.1 – cf. (2.29)) allows one to develop a scheme, in which the arising discontinuities (due to the problem geometry) are in fact 'hidden' on the side of the DFT domain and do not show directly in the spatial domain. The price which has to be paid for using this more sophisticated scheme (derived below) is a greater complication of the algorithm (although the theoretical computational complexity remains low, i.e. linear-logarithmic) and degradation of some properties of the resulting discrete operator.

We start the derivation with operator equation (2.29), obtained in Section 2.1.1. This operator equation may be written in the following compact form:

$$\mathbf{B}^{-1}\mathbf{A}\vec{H}_t = \beta^2\vec{H}_t \tag{6.12}$$

where $\vec{H}_t = [H^x, H^y]^T$ and

$$\mathbf{A}(\cdot) = \mathbf{L}_{Thh}(\cdot) \tag{6.13}$$

and

$$\mathbf{B}^{-1}(\cdot) = -\hat{z} \times \mathbf{L}_{Tee}\hat{z} \times (\cdot) \tag{6.14}$$

If $\{h^x_{mn}\}$, and $\{h^y_{mn}\}$ are functions given by equations (4.59) and (4.60) forming an orthonormal basis, then the fields are represented as follows:

$$H^x = \sum_m \sum_n c^x_{mn} h^x_{mn} \qquad H^y = \sum_m \sum_n c^y_{mn} h^y_{mn} \tag{6.15}$$

So, a discrete representation of fields typical of eigenfunction expansion techniques is applied.

The essential point of the proposed algorithm is the method of computing the following integrals:

$$\int \left(\Pi_{\mathbf{x}} \cdot \mathbf{A}\vec{F}\right) h^x_{mn} d\Omega \qquad \int \left(\Pi_{\mathbf{y}} \cdot \mathbf{A}\vec{F}\right) h^y_{mn} d\Omega \tag{6.16}$$

$$\int \left(\Pi_{\mathbf{x}} \cdot \mathbf{B}^{-1}\vec{F}\right) h^x_{mn} d\Omega \qquad \int \left(\Pi_{\mathbf{y}} \cdot \mathbf{B}^{-1}\vec{F}\right) h^y_{mn} d\Omega \tag{6.17}$$

where $\Pi_{\mathbf{x}}$ and $\Pi_{\mathbf{y}}$ denote the operators of projection onto x and y directions respectively and $\vec{F} = [F^x, F^y]^T$ is a vector field, where F^x and F^y denote functions also given by the Fourier expansions:

$$F^x = \sum_m \sum_n c^x_{mn} h^x_{mn} \qquad F^y = \sum_m \sum_n c^y_{mn} h^y_{mn} \tag{6.18}$$

(At this point we try to abstract from the physical meaning of $\vec{F}$.)

The computation of the integrals (6.16) and (6.17) may be performed in the following manner. We start with writing down explicit form of the differential operators involved:

$$\Pi_{\mathbf{x}} \cdot \mathbf{B}^{-1}(\cdot) = \omega\epsilon_0\epsilon\Pi_{\mathbf{x}}(\cdot) + \frac{1}{\omega\mu_0}\frac{\partial}{\partial x}\left[\left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right] \cdot (\cdot)\right] \tag{6.19}$$

and

$$\Pi_{\mathbf{y}} \cdot \mathbf{B}^{-1}(\cdot) = \omega\epsilon_0\epsilon\Pi_{\mathbf{y}}(\cdot) + \frac{1}{\omega\mu_0}\frac{\partial}{\partial y}\left[\left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right] \cdot (\cdot)\right] \tag{6.20}$$

Then, the formula for the first of the integrals (6.17) may be written as follows:

$$\begin{aligned}\int \left(\Pi_{\mathbf{x}} \cdot \mathbf{B}^{-1}\vec{F}\right) h^x_{mn} d\Omega &= \int \omega\epsilon_0(\epsilon - 1)F^x h^x_{mn} d\Omega \\ &+ \int \omega\epsilon_0 F^x h^x_{mn} + \frac{1}{\omega\mu_0}\left(\frac{\partial}{\partial x}\left[\frac{\partial F^x}{\partial x} + \frac{\partial F^y}{\partial y}\right]\right) d\Omega\end{aligned} \tag{6.21}$$

Making use of representation (6.18) and the orthogonality of basis functions one gets:

$$\begin{aligned}\int \left(\Pi_{\mathbf{x}} \cdot \mathbf{B}^{-1}\vec{F}\right) h^x_{mn} d\Omega &= \int \omega\epsilon_0(\epsilon - 1)F^x h^x_{mn} d\Omega \\ &+ \omega\epsilon_0 c^x_{mn} - \frac{1}{\omega\mu_0}\left[\left(\frac{m\pi}{b}\right)^2 c^x_{mn} + \left(\frac{mn\pi^2}{ab}\right) c^y_{mn}\right]\end{aligned} \tag{6.22}$$

An analogous formula is obtained for the second of the integrals:

$$\begin{aligned}\int \left(\Pi_{\mathbf{y}} \cdot \mathbf{B}^{-1}\vec{F}\right) h^y_{mn} d\Omega &= \int \omega\epsilon_0(\epsilon-1)F^y h^y_{mn} d\Omega \\ &+ \omega\epsilon_0 c^y_{mn} - \frac{1}{\omega\mu_0}\left[\left(\frac{mn\pi^2}{ab}\right)c^x_{mn} + \left(\frac{n\pi}{a}\right)^2 c^y_{mn}\right] \quad (6.23)\end{aligned}$$

As one may note, the above formula contains an 'analytical part' and an integral which may be computed numerically, e.g. by performing a 2D forward FFT. One should also note that the domain of integration is reduced only to the region of the waveguide core, as outside the core one has: $\epsilon - 1 \equiv 0$. This fact allows one to reduce significantly the cost of computing this Fourier integral. One also notes that the above integrals do not require computation of the gradient of permittivity profile ϵ.

The computation of the integrals (6.16) may be performed analogously. The explicit form of the involved operators is given as follows:

$$\Pi_x \cdot \mathbf{A}(\cdot) = \omega\mu_0\Pi_x(\cdot) - \frac{\partial}{\partial y}\left(\frac{1}{\omega\epsilon_0\epsilon}\left[-\frac{\partial}{\partial y}, \frac{\partial}{\partial x}\right]\cdot(\cdot)\right) \quad (6.24)$$

$$\Pi_y \cdot \mathbf{A}(\cdot) = \omega\mu_0\Pi_y(\cdot) + \frac{\partial}{\partial x}\left(\frac{1}{\omega\epsilon_0\epsilon}\left[-\frac{\partial}{\partial y}, \frac{\partial}{\partial x}\right]\cdot(\cdot)\right) \quad (6.25)$$

Making use of representation (6.18) and the orthogonality of basis functions one gets:

$$\begin{aligned}\int \left(\Pi_{\mathbf{x}} \cdot \mathbf{A}\vec{F}\right) h^x_{mn} d\Omega &= \omega\mu_0 c^x_{mn} + \frac{1}{\omega\epsilon_0}\left(\frac{mn\pi^2}{ab}\right)c^y_{mn} - \frac{1}{\omega\epsilon_0}\left(\frac{n\pi}{a}\right)^2 c^x_{mn} \\ &+ \int \frac{1}{\omega\epsilon_0}\frac{\partial}{\partial y}\left(\left[\frac{\epsilon-1}{\epsilon}\right]\left[\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right]\right) h^x_{mn} d\Omega \quad (6.26)\end{aligned}$$

In order to eliminate the appearing partial derivative the above formula is further transformed by applying Green's theorem (or simply integration by parts in the one-dimensional case):

$$\begin{aligned}\int \left(\Pi_{\mathbf{x}} \cdot \mathbf{A}\vec{F}\right) h^x_{mn} d\Omega &= \omega\mu_0 c^x_{mn} + \frac{1}{\omega\epsilon_0}\left(\frac{mn\pi^2}{ab}\right)c^y_{mn} - \frac{1}{\omega\epsilon_0}\left(\frac{n\pi}{a}\right)^2 c^x_{mn} \\ &+ \int \frac{1}{\omega\epsilon_0}\left[\frac{\epsilon-1}{\epsilon}\right]\left[\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right] \\ &\times \left(\frac{n\pi}{a}A_{mn}\sin\left(\frac{m\pi x}{b}\right)\sin\left(\frac{n\pi y}{a}\right)\right) d\Omega \quad (6.27)\end{aligned}$$

Once again, in the above integral the domain of integration is limited to the waveguide core and the integral may be computed using a 2D sine FFT. We note that using the

above formula one may enclose the information on the permittivity discontinuity at the core-cladding interface inside the Fourier coefficients, without the necessity of evaluating the derivative of ϵ at the interface.

An analogous result is obtained for the second of the integrals:

$$\begin{aligned}\int \left(\Pi_{\mathbf{y}} \cdot \mathbf{A}\vec{F}\right) h_m^y n d\Omega &= \omega\mu_0 c_{mn}^y + \frac{1}{\omega\epsilon_0}\left(\frac{mn\pi^2}{ab}\right) c_{mn}^x - \frac{1}{\omega\epsilon_0}\left(\frac{m\pi}{b}\right)^2 c_{mn}^y \\ &+ \int \frac{1}{\omega\epsilon_0}\left[\frac{\epsilon-1}{\epsilon}\right]\left[\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right] \\ &\times \left(-\frac{m\pi}{b} B_{mn} \sin\left(\frac{m\pi x}{b}\right) \sin\left(\frac{n\pi y}{a}\right)\right) d\Omega \end{aligned} \tag{6.28}$$

Knowing how to compute integrals (6.16) and (6.17) one is ready to formulate the algorithm of computing the matrix-vector product, which implicitly defines the discrete operator obtained from operator equation (6.12).

The steps of the algorithm for computing the matrix-vector product in this case are summarized in the following steps:

1. Compute the 'analytical parts' of integrals (6.27) and (6.28):

$$a_{mn}^x = \omega\mu_0 c_{mn}^x + \frac{1}{\omega\epsilon_0}\left(\frac{mn\pi^2}{ab}\right) c_{mn}^y - \frac{1}{\omega\epsilon_0}\left(\frac{n\pi}{a}\right)^2 c_{mn}^x \tag{6.29}$$

$$a_{mn}^y = \omega\mu_0 c_{mn}^y + \frac{1}{\omega\epsilon_0}\left(\frac{mn\pi^2}{ab}\right) c_{mn}^x - \frac{1}{\omega\epsilon_0}\left(\frac{m\pi}{b}\right)^2 c_{mn}^y \tag{6.30}$$

2. Using Fourier coefficients c_{mn}^x and c_{mn}^y compute:

$$F^x = \sum_{mn} c_{mn}^x h_{mn}^x \quad F^y = \textstyle\sum_{mn} c_{mn}^y h_{mn}^y \tag{6.31}$$

   with 2D inverse FFTs.

3. In the spatial domain compute:

$$\omega\epsilon_0\left[\frac{\epsilon-1}{\epsilon}\right]\left[\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right] \tag{6.32}$$

4. Using 2D forward sine FFTs compute the integrals:

$$b_{mn}^x = \int \frac{1}{\omega\epsilon_0}\left[\frac{\epsilon-1}{\epsilon}\right]\left[\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right]\left(-\frac{m\pi}{b} B_{mn} \sin\left(\frac{m\pi x}{b}\right) \sin\left(\frac{n\pi y}{a}\right)\right) d\Omega \tag{6.33}$$

and

$$b^y_{mn} = \int \frac{1}{\omega\epsilon_0}\left[\frac{\epsilon-1}{\epsilon}\right]\left[\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right]\left(-\frac{m\pi}{b}B_{mn}\sin\left(\frac{m\pi x}{b}\right)\sin\left(\frac{n\pi y}{a}\right)\right)d\Omega \tag{6.34}$$

5. Compute the Fourier coefficients of the functions $\Pi_{\mathbf{x}} \cdot \mathbf{A}\vec{H}_t$ and $\Pi_{\mathbf{y}} \cdot \mathbf{A}\vec{H}_t$:

$$d^x_{mn} = a^x_{mn} + b^x_{mn} \tag{6.35}$$

$$d^y_{mn} = a^y_{mn} + b^y_{mn} \tag{6.36}$$

6. Compute the 'analytical parts' of integrals (6.22) and (6.23):

$$\tilde{a}^x_{mn} = \omega\epsilon_0 d^x_{mn} - \frac{1}{\omega\mu_0}\left[\left(\frac{m\pi}{b}\right)^2 d^x_{mn} + \left(\frac{mn\pi^2}{ab}\right)d^y_{mn}\right] \tag{6.37}$$

$$\tilde{a}^y_{mn} = \omega\epsilon_0 c^y_{mn} - \frac{1}{\omega\mu_0}\left[\left(\frac{mn\pi^2}{ab}\right)c^x_{mn} + \left(\frac{n\pi}{a}\right)^2 c^y_{mn}\right] \tag{6.38}$$

7. Using Fourier coefficients d^x_{mn} and d^y_{mn} compute:

$$\tilde{F}^x = \sum_{mn} d^x_{mn}h^x_{mn} \quad \tilde{F}^y = \textstyle\sum_{mn} d^y_{mn}h^y_{mn} \tag{6.39}$$

with 2D inverse FFTs.

8. In the spatial domain compute:

$$\omega\epsilon_0(\epsilon-1)\tilde{F}^x \tag{6.40}$$

$$\omega\epsilon_0(\epsilon-1)\tilde{F}^y \tag{6.41}$$

9. Using 2D forward sine FFTs compute the integrals:

$$\tilde{b}^x_{mn} = \int \omega\epsilon_0(\epsilon-1)\tilde{F}^x h^x_{mn}d\Omega \tag{6.42}$$

$$\tilde{b}^y_{mn} = \int \omega\epsilon_0(\epsilon-1)\tilde{F}^y h^y_{mn}d\Omega \tag{6.43}$$

10. Compute the Fourier coefficients of the functions $\Pi_{\mathbf{x}} \cdot \mathbf{B}^{-1}\mathbf{A}\vec{H}_t$ and $\Pi_{\mathbf{y}} \cdot \mathbf{B}^{-1}\mathbf{A}\vec{H}_t$:

$$\tilde{c}^x_{mn} = \tilde{a}^x_{mn} + \tilde{b}^x_{mn} \tag{6.44}$$

$$\tilde{c}^y_{mn} = \tilde{a}^y_{mn} + \tilde{b}^y_{mn} \tag{6.45}$$

where:

$$\Pi_{\mathbf{x}} \cdot \mathbf{B}^{-1}\mathbf{A}\vec{H}_t = \sum_m \sum_n \tilde{c}^x_{mn} h^x_{mn} \qquad \Pi_{\mathbf{y}} \cdot \mathbf{B}^{-1}\mathbf{A}\vec{H}_t = \sum_m \sum_n \tilde{c}^y_{mn} h^y_{mn} \tag{6.46}$$

As already mentioned, the above algorithm defines an *implicit representation* of the operator $\mathbf{B}^{-1}\mathbf{A}$ from equation (6.14). This means that the finite-dimensional operator projection (of operator $\mathbf{B^{-1}A}$ is performed simultaneously with computation of the matrix-vector product $\underline{\underline{C}}\,\underline{v}$, where $\underline{\underline{C}}$ denotes here the projection of $\mathbf{B^{-1}A}$. One may note that Step 3, performed in the spatial domain is limited to operations within the waveguide core, as outside the core $\epsilon - 1 \equiv 0$. The same refers to the integral computed in Step 9. This allows one to reduce the computations in the spatial domain only to the region of the waveguide core.

The algorithm presented above may be used jointly with the Implicitly Restarted Arnoldi Method (IRAM) in order to find numerically eigenvalues and eigenfunctions of the given operator. The performed validation tests included finding propagation constants for modes in structure C from Figure 6.1. Table 6.7 shows the normalized propagation constants β/k_0 computed using the discussed algorithm (IRAM-FFT*). For comparison the results of computations are also given for the IRAM-FDFD algorithm. In the case of the first algorithm the number of expansion functions used to represent the fields equaled 40 in x direction and 20 in y direction, while for the second algorithm the 80×40 FD discretization grid has been applied. One may observe that the results are in relatively good accordance. It is important to stress here that the basic IRAM-FFT algorithm discussed in Section 6.1.2 would result in unacceptable $\approx 10\% - 20\%$ errors. For instance, the propagation constant for the dominant mode, computed using the basic IRAM-FFT algorithm was 2.5406, which means that the relative error equals in this case approximately 10% (cf. Table 6.7).

While investigating numerical properties of the discussed algorithm one may easily find that both the computational and memory complexity are the same as for the IRAM-FFT algorithm using the basic algorithm of computing matrix-vector product. (cf. Section 4.2.2) Still, compared to IRAM-FFT method algorithm IRAM-FFT* requires computation of twice as many Fast Fourier Transforms.

IRAM-FFT* 40×20	IRAM-FDFD 80×40	Relative error [%]
2.2907	2.3059	-0.66
1.5969	1.5845	0.78
0.81459	0.80937	0.64
0.45661-0.33488j	0.49408-0.32459j	7.20
0.45661+0.33488j	0.49408+0.32459j	7.20
0.71145j	0.73155j	-2.75
1.1282j	1.1289j	-0.06
1.2551j	1.2615j	-0.51

Table 6.7: Comparison of the computed normalized propagation constants β/k_0 for two different algorithms: IRAM-FFT* and IRAM-FDFD. For the IRAM-FFT* algorithm the FFT length equaled 256 in every spatial direction. The propagation constants have been computed for the frequency $f = 15$ GHz.

Expansion functions	No. of implicit restarts	No. of $\underline{\underline{A}}\,\underline{v}$ operations	Spectral radius
20×10	7	212	3.744D+01
40×20	15	459	1.477D+02
80×40	37	1122	6.326D+02

Table 6.8: Number of implicit IRAM updates and $\underline{\underline{A}}\,\underline{v}$ operations performed in order to obtain convergence to the wanted eigenvalues for different number of expansion functions used to represent the fields. During the computations FFT length equaled 256.

The significant differences of numerical properties between the eigensolver using the first of the operator formulations (IRAM-FFT) and the currently discussed formulation (IRAM-FFT*) concern the spectral radius of the matrix for which the problem is being solved. As already pointed out, the value of a spectral radius of a matrix has a substantial impact on the speed of convergence of the eigensolver. This is verified by data shown in Table 6.8. This table presents the number of IRAM implicit restarts and $\underline{\underline{A}}\,\underline{v}$ operations which have to be performed in order to obtain results shown in Table 6.7. The data are shown for different number of expansion functions used to represent the fields. A visible correlation between the amount of computation and the spectral radius of the operator matrix (computed as the eigenvalue with the largest modulus) is observed.

Another table (Table 6.9) compares the spectral radii of the operator matrices constructed with the two discussed operator formulations using finite-dimensional projection methods embedded in the IRAM-FFT algorithm. The table shows results for three different waveguiding structures shown in Figure 6.5. In the figure, structure 3 is the same image guide as shown in Figure 6.1C, structure 1 is a hollow waveguide and structure 2 is

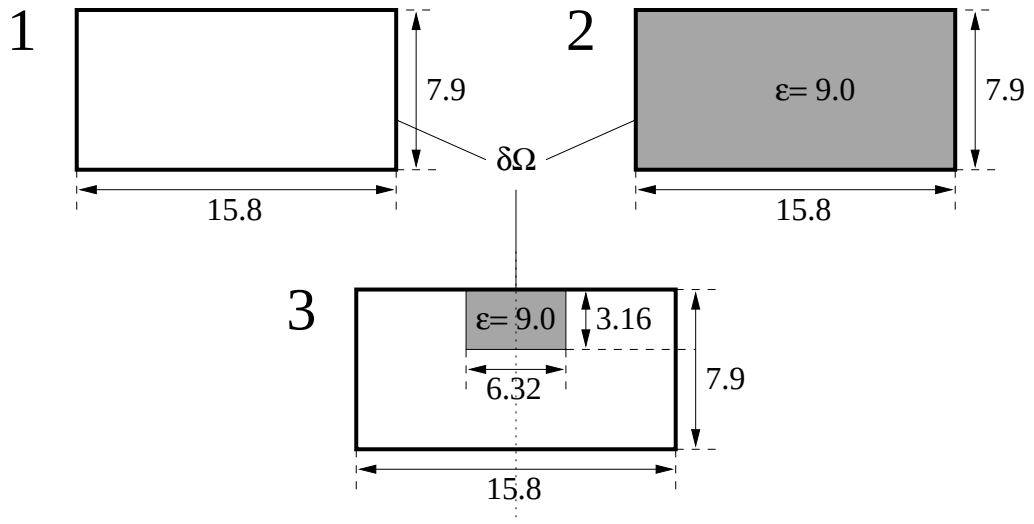


Figure 6.5: Cross-sections of the waveguiding structures used to investigate the spectral radius of the operator constructed in the IRAM-FFT* algorithm. All dimensions are given in millimeters.

Structure	IRAM-FD 80×40	IRAM-FFT-NI 40×20	IRAM-FFT* 40×20
1	2.048D+02	2.832D+02	2.687D+02
2	2.040D+02	2.832D+02	2.885D+02
3	2.047D+02	2.832D+02	6.127D+02

Table 6.9: Spectral radii of matrices used to model the three waveguiding structures shown in Figure 6.5. During the tests: FFT length equaled 512, and frequency $f = 15$ GHz.

a waveguide completely filled with dielectric material with relative permittivity $\epsilon = 9$. As shown in Table 6.9 for the formulation applied in the previous section (IRAM-FFT-NI) the spectral radius of the matrix remains the same for all the modeled structures. In turn, for the algorithm using current operator formulation (IRAM-FFT*), the spectral radius is significantly increased for the structure with a discontinuous permittivity profile. While the problem has had been investigated in sufficient detail, it is suspected that this effect is due to the fact that the problem implemented in IRAM-FFT* algorithm involves a fourth-order differential operator, while IRAM-FFT-NI solver implements the problem for the second-order differential operator.

Clearly, the increased spectral radius deteriorates the rate of convergence of the IRAM-FFT* solver, which becomes lower than for the IRAM-FFT-NI (or IRAM-FFT^2) solver. Moreover, as shown in Table 6.10 the spectral radius of the matrix constructed implicitly by the IRAM-FFT* algorithm varies with the FFT length applied to perform the Discrete

Structure	FFT length = 256	FFT length = 512	FFT length =1024
1	2.687D+02	2.687D+02	2.687D+02
2	3.495D+02	2.885D+02	2.731D+02
3	6.326D+02	6.127D+02	6.108D+02

Table 6.10: Spectral radii for the matrices implicitly constructed using IRAM-FFT* method used to model different waveguiding structures shown in Figure 6.5. The table shows data for different FFT lengths applied during computation.

Fourier Transforms (except the case of the hollow waveguide). Summing up, application of the more complicated operator formulation, as discussed in this section allows one to model waveguiding structures with discontinuous permittivity profiles (with no restrictions concerning the geometry of the permittivity profile) at the cost of a deteriorated convergence rate.

6.2 Modeling electromagnetic resonators

The previous sections presented the application of the discussed numerical techniques to modeling dielectric waveguides. Due to relatively small dimensions of the structures and 2D character of the problems the resulting computational problems may be qualified as small- or medium-sized. Below we present examples of applications of the proposed algorithms to modeling electrically large electromagnetic resonators, particularly open hemispherical resonators. In this case, the resulting problems are characterized by very large 2D and 3D domains, operator matrix sizes of order 10^5 and difficulties with accuracy of discretization, providing truly challenging computational tasks for today's algorithms and computer systems.

Modeling of electromagnetic properties of resonant cavities has numerous applications in microwave technology, e.g. microwave filter design, as well as other engineering and scientific disciplines, such as material science, e.g. in characterization of properties of different materials, including high temperature superconductors and semiconductors working in microwave or millimeter wave frequency range [1], [27], [54], [60], [65]. In the latter case the system shown in Figure 6.6 is used for measurements. The sample of interest is placed on the flat mirror and high-order modes, such as TEM_{00q}, where $q > 20$ are excited in a resonator structure. Then, the resonant frequencies of these modes are measured in order to determine the response of the structure loaded with an investigated material. It has to be stressed that very high accuracy results are needed in the presented application. Although in many cases the parameters of the sample may be modeled analytically with sufficient accuracy using paraxial wave propagation approximation [53], this method fails if the measured sample is relatively thick, consists of many layers or one wants to investigate effects related to finite sample size or a non-ideal mirror shape.

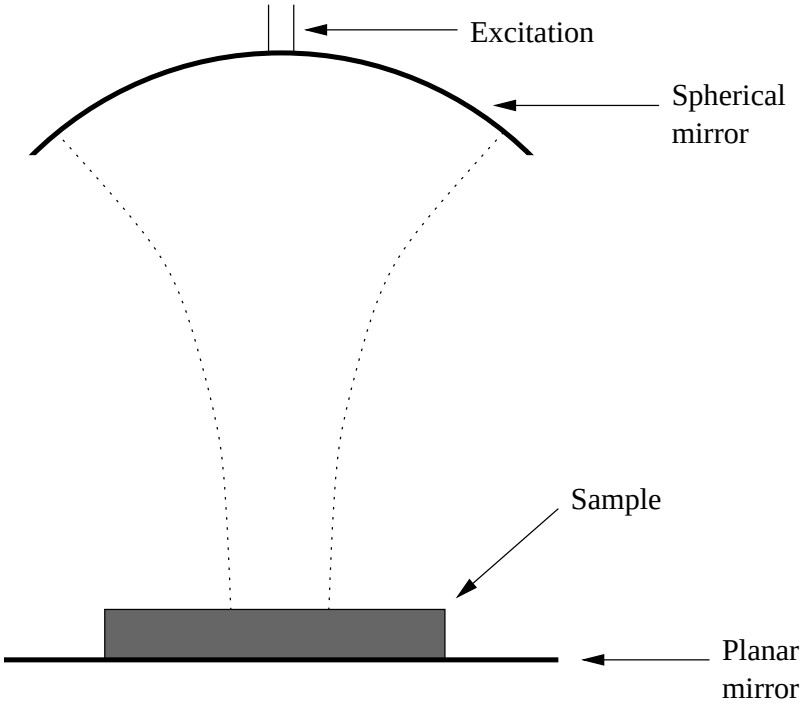


Figure 6.6: Open resonator structure loaded with a dielectric disc.

In all the mentioned cases numerical treatment has to be applied.

Still, modeling of the described open resonator problem is a challenging computational problem which until recently remained intractable. This has been caused by the following reasons:

- The accuracy of the numerical solution has to be comparable to the analytical one and the solution method should be versatile in order to allow one to investigate effects not modeled by paraxial approximation.
- The structure is electromagnetically large, i.e. its dimensions are large compared to the wavelength λ of the excited modes. Even if, due to rotational symmetry of the structure, the problem is reduced to two dimensions, the size of computational domain equals typically a few hundred λ^2.
- The resonator spectrum is very dense (e.g. 20 modes per 1 gigahertz), which causes problems with mode identification.
- The location of the mode to be computed on a mode chart ranges from 1000th to 5000th position, far from the fundamental mode, which alone makes extraction of this eigenvalue a very challenging problem.

- Huge size of the corresponding discrete problem causes serious problems related to numerical cost of computations.

- Numerical dispersion arising if finite difference (e.g. Finite Difference Time Domain or FDFD) methods are applied causes severe degradation in the accuracy of computations, which is unacceptable e.g. in modeling of properties of new materials.

The above factors clearly indicate that the presented problem of modeling of high order modes in resonant cavities belongs to complex, large scale electromagnetic problems and requires application of special methods which reduce the cost of computations, enhance the accuracy of computations and allow finding desired modes from the resonator spectrum.

In the past, attempts have been made using methods such as Boundary Element Method (BEM) [15] and FDTD [54]. In the cited works, FDTD provided only qualitative results for low order modes, while BEM generated a dense matrix problem which was very costly to solve. This section shows how the advanced techniques proposed in the previous chapters allow one to deal effectively and efficiently with this class of large scale computational problems.

Below the two algorithms:

1. IRAM-FDFD: based on IRAM, finite difference frequency domain operator projection in cylindrical coordinate system (presented in Section 4.1.2), implicit operator representation (presented in Section 4.1.5), and using algorithm of correcting numerical dispersion proposed in Section 4.1.3.

2. IRAM-HYBRID: based on IRAM and hybrid operator projection using FDFD and eigenfunction expansion in three dimensions (presented in Section 4.3.1).

are applied to modeling cylindrical and hemispherical electromagnetic resonators, using operator formulation developed in Section 2.1.2 with appropriately defined PEC boundary conditions. The structures to be modeled include an open hemispherical resonator with rotational symmetry and cylindrical resonator lacking rotational symmetry. By solving the appropriate operator boundary value problems one obtains resonant frequencies of the resonators and the corresponding modal fields.

Before we turn to presenting numerical results, it is worthwhile to discuss the numerical techniques which allow one to extract high order modes (eigenvalues) from the dense spectrum of the emerging discrete operators. This question is addressed in the following section.

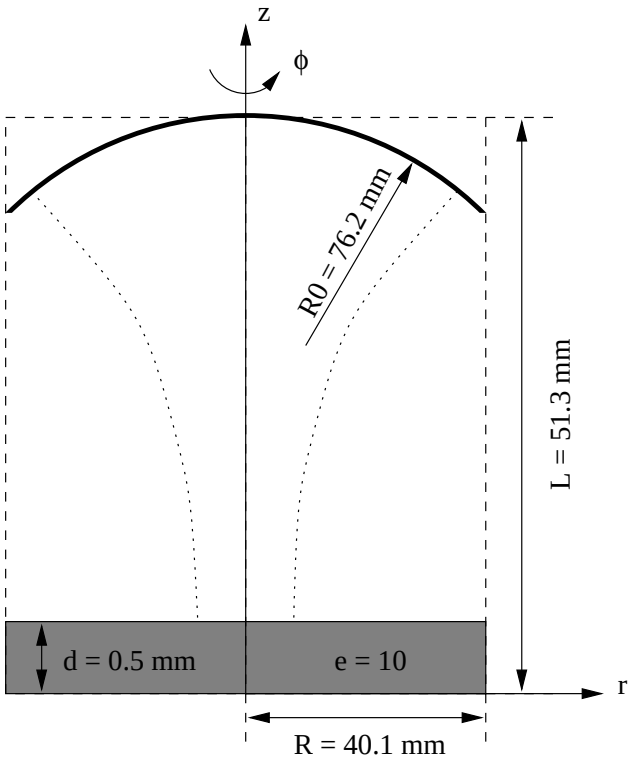
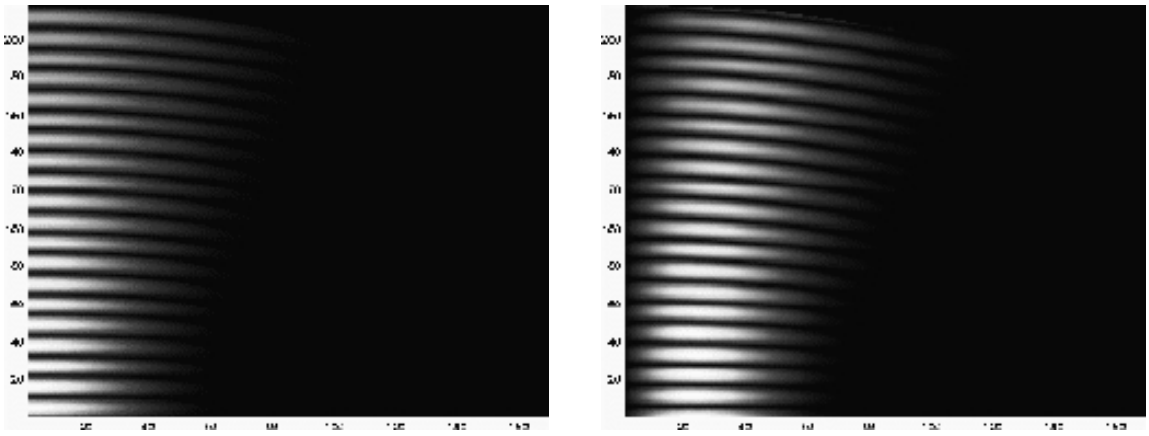


Figure 6.7: Schematic of an open hemispherical resonator loaded with a dielectric material.

6.2.1 Finding higher modes in resonators: Chebyshev polynomial filtration

To illustrate the problem, one may consider a task of finding the resonant frequency of e.g. the quasi-$TEM_{0,0,20}$ mode in a hemispherical resonator shown in Figure 6.7 using the IRAM-FDFD algorithm. The frequency to be found equals approximately $f_t = 57.691$ GHz. The eigenvalue to be found equals: $\omega^2 = (2 * \pi * f_t)^2 = 1.3139 \cdot 10^{23}$ rad^2/m^2 . If the discretization grid size equals $\lambda/12$ in both r- and $z-$directions, then the resulting operator matrix has the size of 18530. The spectral radius of the matrix equals $4.587e \cdot 10^{24}$ and the eigenvalue corresponding to the resonant frequency of the lowest order mode is $5.8 \cdot 10^{20}$. Consequently, assuming that the density of eigenvalues is the uniform along the matrix spectrum, one finds that the resonant frequency of the quasi-$TEM_{0,0,20}$ corresponds approximately to the 500-th eigenvalue of the matrix. Since the eigenvalue is located far from both ends of the matrix spectrum the Krylov subspace methods (such as IRAM) cannot find efficiently as many as 500 eigenvalues with the smallest modulus. (This would require construction of an orthonormal basis in a Krylov subspace of order 500 or more, which is inefficient due to a very high cost of orthogonalization process.) Therefore it is necessary to modify the spectrum of the matrix operator, so that eigen-



E_r E_z

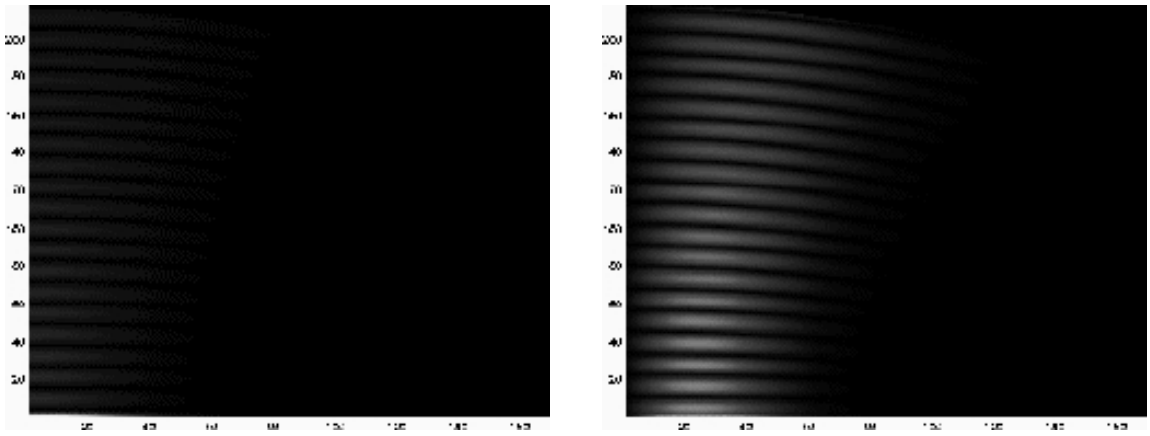
Figure 6.8: Normalized rms E_r and E_z field plots for the quasi-$TEM_{0,0,20}$ mode in a homogeneous hemispherical resonator (cf. Figure 6.7 - the dielectric filling is removed in this case). The discretization grid applied: $\Delta r = \Delta z = 1.5 \cdot 0.237$ mm $= \lambda/12$.

values from the desired range are moved towards any of the ends of the matrix spectrum. This can be achieved by applying polynomial filtering, based on Chebyshev polynomials and designing bandpass digital filters. This technique is described in detail in Appendix C. It transforms the spectrum of the matrix so that the eigenvalues located around a given central frequency of the filter form the hi-end of the matrix spectrum and consequently may be found by searching the 'largest modulus' eigenvalues with IRAM. Below, results of validation tests for the proposed algorithms, using the mentioned polynomial filtering technique, are presented.

6.2.2 Large hemispherical resonators: numerical results

This section describes the tests validating the IRAM-FDFD algorithm, using technique for correcting the effects of numerical dispersion for the case of hemispherical resonator shown in Figure 6.7. Also, the case of a homogeneous resonator with no dielectric filling is considered. Curved mirror were modeled by means of conformal technique and effective permittivity concept was applied to dielectric boundaries [26]. The numerical tests aimed at finding the resonant frequency of the quasi-$TEM_{0,0,20}$. The field plots for this mode, computed using IRAM-FDFD algorithm for a homogeneous and inhomogeneous hemispherical resonator have been shown in Figures 6.8 and 6.9, respectively.

In the first series of tests the basic algorithm, which does not include the technique for reducing numerical dispersion, has been investigated. For the homogeneous resonator and the discretization grid $\Delta r = \Delta z = 1.5 \cdot 0.237$ mm $= \lambda/15$ the computed resonant frequency of the quasi-$\mathrm{TEM}_{0,0,20}$ (Gaussian beam notation) equals $f = 58.913499369858$ GHz. The error relative to the theoretical frequency $f_t = 59.375168474867$ GHz equals 0.777%. The results for different discretization grids are shown in Table 6.11. For



E_r E_z

Figure 6.9: Normalized rms E_r and E_z field plots for the quasi-$TEM_{0,0,20}$ mode in an inhomogeneous hemispherical resonator (cf. Figure 6.7). The discretization grid applied: $\Delta r = \Delta z = 1.5 \cdot 0.237$ mm $= \lambda/12$.

FD grid	Matrix size	Resonant frequency [GHz]	Relative error [%]
$\lambda/15$	32770	58.913499369858	-0.777
$\lambda/20$	73780	59.168570954669	-0.348
$\lambda/30$	165100	59.266261667606	-0.183
λ/∞	–	59.375168474867	0.000

Table 6.11: Error due to numerical dispersion in the FDFD algorithm on Yee's mesh for an empty hemispherical resonator. The table entry denoted λ/∞ gives the theoretical, reference resonant frequency.

the finest grid applied the error is still 0.183%. In the case of the resonator with the dielectric filling and the same discretization grid the computed resonant frequency equaled $f = 57.02053612372188$ GHz. The error relative to the theoretical frequency $f_t = 57.69091347456410$ GHz equals 1.163%. The results for different discretization grids are shown in Table 6.12. In this case the smallest error equals 0.196%.

The values of relative errors obtained for the applied grids are still too large for the numerical method to be used in practical measurements. This problem may be solved by further refinement of the grid, but this appears to be impractical since it increases (already large) matrix size and slows down the convergence of the IRAM solver. Consequently, a different strategy, based on technique of correcting the effects of numerical dispersion (proposed in Section 4.1.3) should be applied, in order to improve the properties of the numerical solver. This strategy is discussed in the following paragraph.

FD grid	Matrix size	Resonant frequency [GHz]	Relative error [%]
$\lambda/15$	32770	57.02053612372188	-1.163
$\lambda/20$	73780	57.45054906432008	-0.417
$\lambda/30$	165100	57.57789349284087	-0.196
λ/∞	–	57.69091347456410	0.000

Table 6.12: Error due to numerical dispersion in the FDFD algorithm on Yee's mesh for a hemispherical resonator with dielectric filling (cf. Figure (6.9)). The table entry denoted λ/∞ gives the theoretical, reference resonant frequency.

Grid	A	Resonant frequency [GHz]	Relative error [%]
$\lambda/15$	1.00800933665862	59.378210357138	-0.005
$\lambda/20$	1.00354468643016	59.375151192199	-0.00003
λ/∞	–	59.375168474867	0.00000

Table 6.13: Dispersion-corrected results of the resonant frequency for the quasi $TEM_{0,0,20}$ mode in a homogeneous hemispherical resonator.

6.2.2.1 Correction of numerical dispersion

In order to reduce the errors due to numerical dispersion the low-cost correction technique described in Section 4.1.3 has been applied in the IRAM-FDFD solver for the open resonator problem.

Initially, the case of empty hemispherical resonator has been investigated. Table 6.13 shows resonant frequencies computed with IRAM-FDFD algorithm using values of A optimized for the theoretical resonant frequency f_0. The other optimization parameters equaled $k_z = 2\pi N/L = 2\pi \cdot 10$ rad/(51.3 mm), where L is the length of the cavity, $k_r = sqrt(k^2 - k_z^2)$, where $k = 2\pi f_0/c$ and the value of I corresponds to the cylindrical plane with the radius $R/10$ (A certain rationale for this choice is that most of the power of the modal field is enclosed in that cylindrical region (cf. Figure 6.8)). Comparing the results with data shown in Table 6.11 one notes a substantial improvement in the computed resonant frequencies. Table 6.14 shows resonant frequencies computed with IRAM-FDFD using values of A optimized for different cylindrical planes. Although the best results are obtained for $R/10$, in any case the reduction of numerical error with respect to the non-optimized case is substantial.

Instead of performing optimization of the value of A for a fixed frequency one may apply an iterative scheme introduced in Section 4.1.4 in order to enhance the quality of the computed results. In this scheme (cf. Table 6.15) the computed resonant frequency, providing an approximation for the exact resonant frequency, is used to find an optimized

A	Resonant frequency [GHz]	Relative error [%]	Optimized for
–	59.37516847874867	0.000	–
1	58.91349971348569	-0.777	–
1.00800933665862	59.37821035713776	0.005	R/10
1.00777725841457	59.36474306226363	-0.018	R/5
1.00773735322444	59.36242819476590	-0.022	R/3
1.00771941522412	59.36138700979552	-0.0232	R/2
1.00771689609819	59.36124092386456	-0.0235	R

Table 6.14: Dispersion-corrected results of the resonant frequency for the quasi $TEM_{0,0,20}$ mode in a homogeneous hemispherical resonator. The optimization has been performed for different cylinder sizes. The discretization grid is $\Delta r = \Delta z = 1.5 \cdot 0.237$ mm.

Optimization frequency [GHz]	Value of A	Resonant frequency [GHz]	Relative error [%]
-	1.00000000000000	58.91349971348569	-0.777
58.91349971348569	1.00814383806358	59.38601254832548	0.018
59.38601254832548	1.00800626897733	59.37803159008240	0.005

Table 6.15: Iterative reduction of the error due to numerical dispersion for the homogeneous resonator problem. The resonant frequencies for the quasi-TEM_{0020} mode computed in one step are used in the next step to find the optimized value of A. $\Delta r = \Delta z = \lambda/15$ ($N = 32770$).

value of A in the next step, i.e. parameter A is found for the resonant frequency computed in the previous iteration. From Table 6.15 one may note that just after 2 iterations the result obtained with $\lambda/15$ grid for this scheme is approximately 10 times better (in terms of relative error) than the result for the $\lambda/30$ grid (compare Table 6.11 or Table 6.16).

In order to assess whether it is efficient to use the discussed iterative scheme we have compared the execution times of IRAM-FDFD algorithm for different grid sizes. Table 6.16 shows number of IRAM implicit update iterations and matrix-vector multiplications needed to obtain convergence along with execution times (for parallel execution on 4 processors) for different grid sizes. The last row shows data for 2 subsequent iterations of the IRAM-FDFD algorithm, corresponding to the iterative correction of the error due to numerical dispersion. One notes that this scheme is more than four times faster than the basic solver for the grid size $\lambda/30$, giving much better results.

The tests have also been performed for the case of hemispherical resonator loaded with a dielectric sample (cf. structure in Figure 6.7). Table 6.17 presents resonant frequencies for this case computed with IRAM-FDFD algorithm. The correction of numerical dispersion

Problem size	Grid size	No. of updates	No. of $\underline{\underline{A}}\,\underline{v}$ operations	Execution time [s]	Relative error [%]
32770	$\lambda/15$	85	2216	1091.43	-0.777
73780	$\lambda/20$	107	2699	4690.47	-0.348
165100	$\lambda/30$	181	4418	8884.84	-0.183
32770 2-iter.	$\lambda/15$	170	4432	2197.78	0.018

Table 6.16: Comparison of the number of of implicit updates, matrix-vector operations and execution times for the IRAM-FDFD algorithm, for different problem sizes. The last row shows data for the 2 repetitions of the IRAM-FFT algorithm, performed in order to reduce the dispersion error. The tests have been performed in the Cray T3E system using 4 processors.

Grid	A	Resonant frequency [GHz]	Relative error [%]
$\lambda/15$	1.00806364207505	57.47324359009531	-0.37
$\lambda/20$	1.00356777087715	57.65191932766050	-0.07
$\lambda/20$	1.00401803934350*	57.67794770894802	-0.03
λ/∞	–	57.69091347456410	0.00

Table 6.17: Dispersion-corrected results of the resonant frequency for the quasi $TEM_{0,0,20}$ mode in an inhomogeneous hemispherical resonator.

has been performed by using values of A optimized for the theoretical resonant frequency. Comparing the results with data shown in Table 6.12 one notes a good improvement in the computed resonant frequencies. It should be mentioned that the optimization of the values of A has been performed for the cylindrical plane with the radius $R/10$. Once again, the rationale for this choice is that most of the power of the modal field is enclosed in that cylindrical region (cf. Figure 6.9). In Table 6.17 the value of A marked with an asterisk(*) has been obtained using the following heuristic procedure. The two values of A have been computed: one for a hollow resonator and one for a resonator completely filled with dielectric material with relative permittivity $\epsilon = 10$. The final value of A has been computed as a weighted mean value of the two numbers, with weights reflecting the proportion of the volume of dielectric filling to the volume of the entire resonator. One notes that for this value of A results in a smaller relative error of the computed resonant frequency.

Analogously as for the case of an empty hemispherical resonator, the values of parameter A are computed without referring to any fixed resonant frequency. Instead, the iterative scheme discussed in the previous paragraph may be applied. Tables 6.18 and 6.19 show the results obtained in this way. The data in the second table refers to the

Optimization frequency [GHz]	Value of A	Resonant frequency [GHz]	Relative error [%]
-	1.00000000000000	57.02053612372188	-1.17
57.02053612372188	1.00788431822189	57.46329577804698	-0.40
57.46329577804698	1.00800251347183	57.46994349600465	-0.38

Table 6.18: Iterative reduction of the error due to numerical dispersion for the inhomogeneous resonator problem. The resonant frequencies for the quasi-TEM_{0020} mode computed in one step are used in the next step to find the optimized value of A, computed with assumption that $\epsilon = 1$. $\Delta r = \Delta z = \lambda/15$ ($N = 32770$).

Optimization frequency [GHz]	Value of A	Resonant frequency [GHz]	Relative error [%]
-	1.00000000000000	57.02053612372188	-1.17
57.02053612372188	1.00858350560113	57.50256536476118	-0.32
57.50256536476118	1.00872483204390	57.51050773068695	-0.31

Table 6.19: Iterative reduction of the error due to numerical dispersion for the inhomogeneous resonator problem. The resonant frequencies for the quasi-TEM_{0020} mode computed in one step are used in the next step to find the optimized value of A, computed as a weighted mean for the cases when $\epsilon = 1$ and $\epsilon = 10$. $\Delta r = \Delta z = \lambda/15$ ($N = 32770$).

scheme, in which A was computed iteratively as a weighted mean of A's for the region with $\epsilon = 1$ and $\epsilon = 10$. Apparently, application of the weighted sum gives slightly better results. Nevertheless, it appears that for the discussed grid size ($\lambda/15$ outside the dielectric, the error cannot be reduced to less than approximately 0.3%. This can be interpreted in the following way: The grids inside and outside dielectric filling may be considered as different grids, due to different permittivities, directly related to wave propagation velocity. In fact, inside the dielectric sample the wave propagates in a grid which is relatively much coarser than $\lambda/15$ (which is the grid size for the wave propagating outside the sample). So, the numerical wave propagates in two grids characterized by different velocity anisotropy, and consequently the phase error cannot be consistently canceled in both regions. Further improvement of the results seems to require *both* refinement of the FD grid and iterative correction of numerical dispersion.

6.2.3 Cylindrical resonator without rotational symmetry: numerical results

So far the considerations focused on modeling large open resonators which possess rotational symmetry (i.e. are homogeneous in the ϕ direction). As shown in the previous chapters the problem of modeling these structures could be reduced to two dimensions,

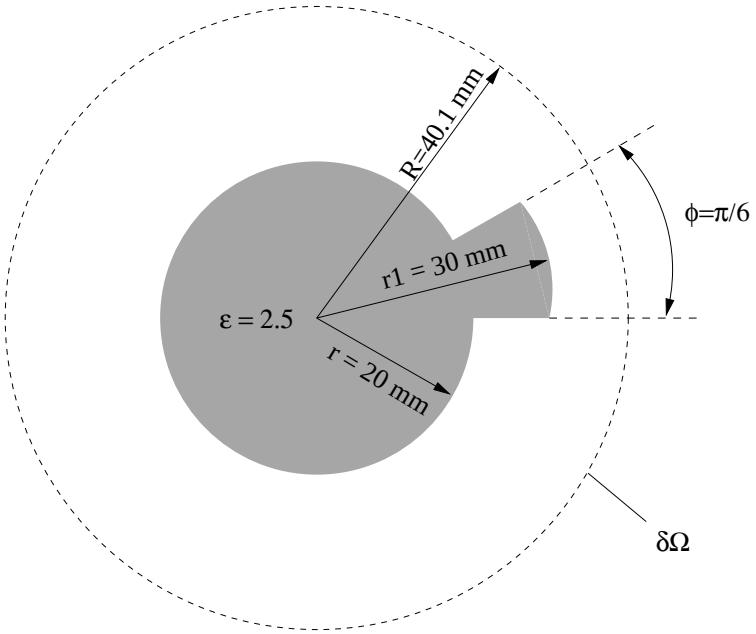


Figure 6.10: Schematic of the $r-\phi$ cross-section of a cylindrical resonator with a dielectric filling, inhomogeneous in the ϕ direction.

which allowed one to efficiently solve the numerical problem. If the structure lacks rotational symmetry, the problem domain may no longer be reduced to 2D. In this case a full three-dimensional computational domain has to be considered, which has serious implications on the size of the resulting discrete operator eigenproblem. If we consider a finite difference scheme involving e.g. 150 grid points in r, z and ϕ directions (which is typical for the discussed applications) the problem size (for the formulation involving transverse field components) will equal approximately $6.7 \cdot 10^6$, which means that the problem will be extremely large and difficult to handle by any numerical treatment. Consequently, a different strategy of finite-dimensional projection, resulting in a more compact discrete representation has to be applied. This strategy is based on the hybrid projection scheme proposed in Section 4.3.1, applying finite difference technique in r and z directions and functional representation in ϕ direction. In this approach the number of expansion functions applied to represent the respective fields is typically smaller than the number of discretization points in the spatial domain (in the ϕ direction). In this way the dispersion error is appropriately reduced, while avoiding excessive growth of the problem size.

Below an example of application of the mentioned technique is presented. In this application the hybrid projection method is implemented together with IRAM. The numerical solver (denoted as IRAM-HYBRID) is used to modeling electromagnetic structures, such as one shown in Figure 6.10. The Figure shows an $r-\phi$ cross-section of a cylindrical resonator filled with a dielectric material which has a 'disturbed' geometry. Clearly, this

ω^2 coupled fields	ω^2 decoupled fields
$3.40528869 \cdot 10^{20}$	$3.40485010 \cdot 10^{20}$
$3.40528869 \cdot 10^{20}$	$3.40091483 \cdot 10^{20}$

Table 6.20: Squared resonant frequencies computed for a 'symmetric' cylindrical cavity (coupled fields) and 'disturbed' cavity (decoupled fields) shown in Figure 6.10.

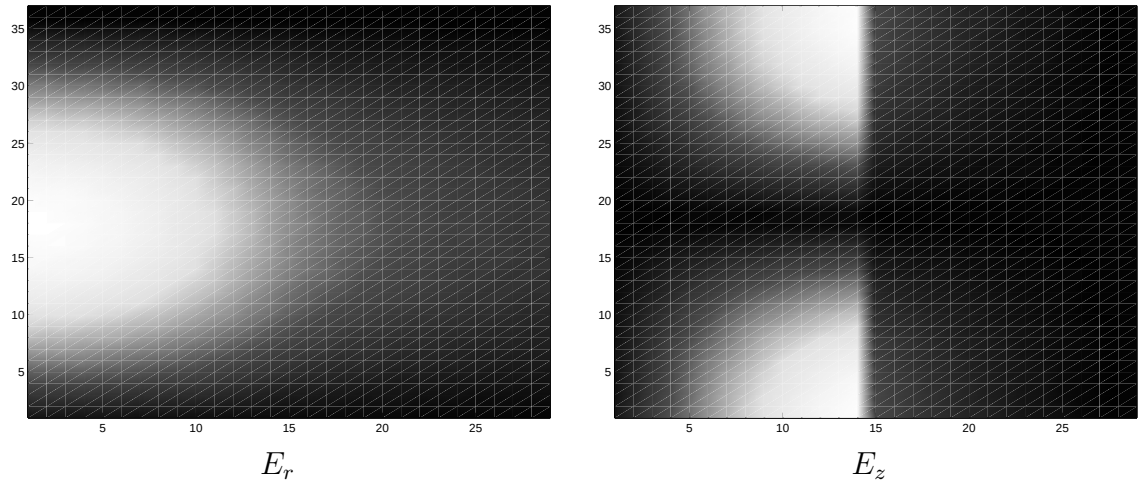


Figure 6.11: Normalized rms E_r and E_z field plots for the mode in the resonator shown in Figure 6.10. The discretization grid applied: $\Delta r = \Delta z = 6 \cdot 0.237$ mm, number of expansion terms applied in ϕ direction = 20, FFT length = 256.

structure is inhomogeneous in the ϕ direction. It has been selected for tests in order to investigate mode degeneration. In the case when the disturbance is removed, i.e. the dielectric filling is a uniform cylinder with radius $r1 = 20$ mm, it is expected that degenerate, coupled modes appear in the resonator. Due to the small insertion which disturbs the rotational symmetry of the structure, these modes should decouple into two separate modes with slightly different resonant frequencies.

This effect may be modeled by using the IRAM-HYBRID algorithm, in which degenerate modes easily appear due to selected Fourier functional basis in the ϕ direction. The degenerate eigenvalues have then a space of corresponding eigenvectors of dimension more than 1. Table 6.20 shows the results of computations for the symmetric and 'disturbed' structures (the 'disturbed' structure is shown in Figure 6.10). In both cases the distance between the flat mirrors equaled 51.3 mm and the screening walls were placed at the distance $R = 40.1$ mm from the axis of the structure. The table shows the computed values of squared resonant frequencies for decoupled modes for a selected low order mode. The modal field distributions for these modes (in both 'coupled' and 'decoupled' case) are very similar. One of them is shown in Figure 6.11.

In the performed computations the number of Fourier expansion terms used in the representation of fields in the ϕ direction equaled 21. The number of discretization points in the r and z direction equaled 29 and 37, respectively. Consequently, the size of the solved eigenproblem equaled $N = 45066$. The FFT length equaled 256, which corresponds to 256 discretization points in the ϕ direction for the spatial domain. It has to be stressed that identical results have been obtained if only 11 Fourier coefficients have been used. (Then, the problem size equaled $N = 23606$.) This observation indicates that a sufficient functional representation has been used. It also implies that important savings can be made by applying the discussed hybrid projection scheme. One should note that implicit projection scheme was used in the computation was used in this case (i.e. operator matrix was never formed). Now, if instead of hybrid discretization, the finite difference discretization were applied in all 3 spatial directions, with 256 grid points in the ϕ direction, the size of the problem would equal approximately 600000. This would imply significant increase in both memory storage requirements and computational effort needed to solve this problem.

6.3 Summary

In this chapter a number of previously proposed numerical techniques have been applied to solving selected problems arising in electromagnetic modeling. The solvers which have been constructed may generally be characterized as follows:

- For all the methods the memory cost associated with solving the eigenproblems is linear.
- The computational complexity is either linear (for solvers based entirely on FDFD) or linear-logarithmic (for methods using functional expansions).
- The size of the discrete eigenproblems derived from initial electromagnetic problems is kept as small as possible by using: 1) formulations involving reduced number of variables, 2) cost-efficient projection method based on eigenfunction expansions in 2D, 3) hybrid projection technique for 3D problems.
- Adequate accuracy of the solvers is obtained by dealing with errors e.g. due to numerical dispersion by 1) applying oversampling in the spatial domain (for eigenfunction expansion based algorithms), 2) introducing schemes correcting dispersion error for FDFD based solvers.

The above features bring the presented solvers, based on proposed numerical techniques, close to the case of an ideal algorithm, discussed in the beginning of this chapter. Clearly, a number of trade-offs is observed, e.g. reducing problem size versus increasing computational cost for algorithms based on eigenfunction expansions. Nevertheless, the validation tests show that the postulates or goals set forth while systematically constructing the discussed algorithms were adequate, leading to efficient solvers, capable of dealing with large scale electromagnetic problems. The results prove that satisfying

the requirements, such as low memory and computational complexity or reducing size of the discrete problems, provides means to construct effective and efficient numerical solvers. The following chapter shows that the proposed techniques are also characterized by good efficiency in parallel processing environments, which further extends their range of applicability.

Chapter 7

Performance of the solvers – numerical results

The previous chapter concentrated on presenting the scope of applications of the discussed numerical algorithms in modeling of large scale electromagnetic systems. This part of the study focuses on performance of the proposed solvers, assessed by investigating firstly their scalability in distributed memory systems and secondly their rate of convergence.

The parallel performance tests have been carried out using mainly the 24-processor Cray T3E system as well as the 15-node IBM SP2. Both supercomputers belong to a class of scalable parallel distributed memory systems. A broad discussion of their characteristics is presented in [94] and [95].

7.1 Parallel solvers using FDFD discretization

This section presents the results of performance tests of the parallel implementation of the IRAM-FDFD solver, described in Section 5.3. The tests aim at assessing mainly the speedup of computations achieved during parallel execution of the eigensolver. This parameter provides most substantial information about the performance of the parallel method, determining whether it may be efficiently used to model large scale problems. The presented tests, performed in a parallel system, involve solving electromagnetic eigenproblems widely discussed in the previous chapters. Consequently, the obtained results are relevant to situations in which the proposed algorithms are applied to solving 'real' engineering or scientific numerical problems.

7.1.1 Parallel Arnoldi solver applied to modeling waveguiding structures

The presentation of parallel performance of IRAM-FDFD solver starts with describing numerical tests involving application of the method to modeling an image guide (structure C in Figure 6.1). This operator problem has already been described in the previous

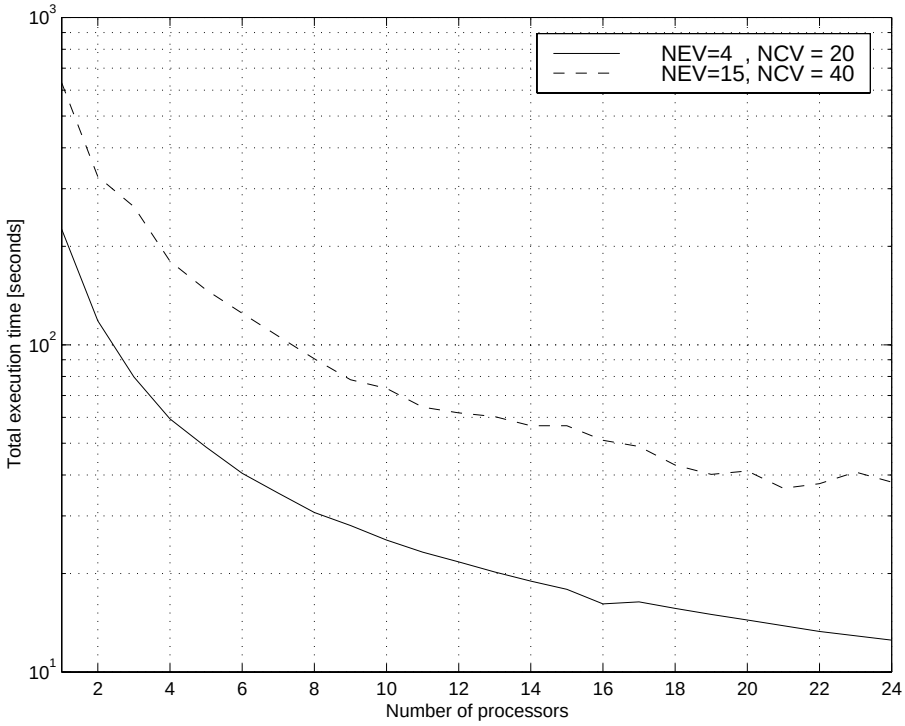


Figure 7.1: Execution time of the parallel IRAM-FDFD solver as a function of the number of processors involved in the computation. The tests were performed in the Cray T3E system.

chapter (Section 6.1.1), therefore only parameters relevant to the performance of the solver are given below. The essential input parameters defining the tests were as follows:

1. The size of the input matrix operator equaled $N = 39700$; the matrix was sparse with 199538 non-zero elements; among the non-zero elements 95% were located in the five diagonals: 0 (main diagonal), +2, -2, +199, -199; the bandwidth of the matrix equaled 402.

2. IRAM parameters: NEV = 4 (number of eigenvalues to be found), NCV = 20 (number of additional eigenvalues to be filtered out). For some tests: NEV=15 and NCV=40

3. The stopping criterion – the accuracy of computed eigenvalues equaled $tol = 1.2 \cdot 10^{-16}$.

The following series of tests was performed in the Cray T3E parallel system. Both the P_ARPACK library (cf. Appendix B) and the solver code were compiled using the following directive: `f90 -O3 -X m xxx.f -o xxx -lsci -lmpi`.

Number of PEs	Total time [s] NEV=4, NCV=20	Efficiency [%]	Total time [s] NEV=15, NCV=40	Efficiency [%]
1	225.26	100	631.42	100
2	118.27	95	326.33	97
4	59.38	95	180.10	88
8	30.70	92	90.56	87
16	16.15	87	51.05	77
24	12.52	75	37.99	69

Table 7.1: Selected total execution times of the parallel IRAM-FDFD solver as a function of the number of processors involved in the computation. The table also shows parallel efficiency of the solver. The tests were performed in the Cray T3E system.

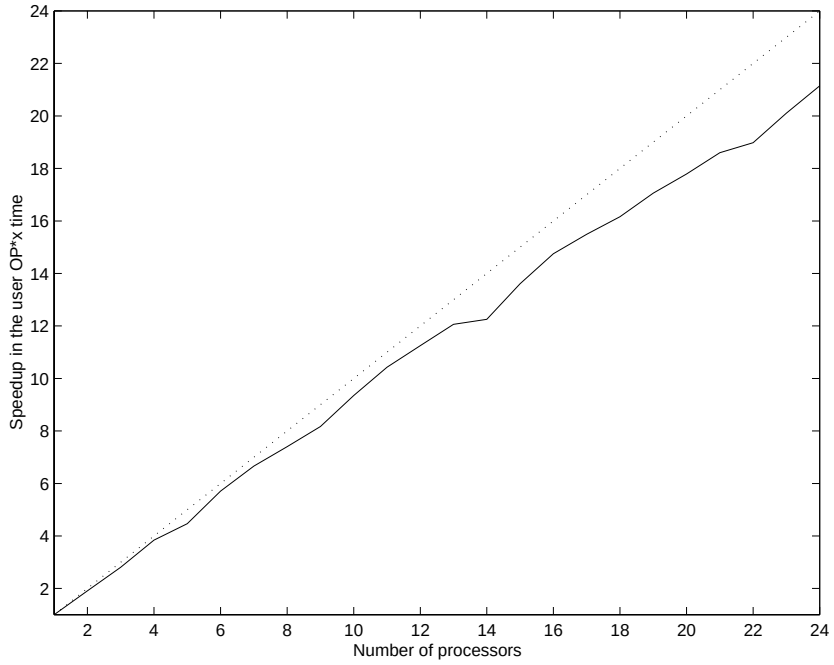


Figure 7.2: Speedup in the execution time while calculating the matrix-vector (OP*x) product in the IRAM-FDFD solver in the function of the number of processors involved in the computation. The tests were performed in the Cray T3E system. The dotted line shows perfect linear speedup.

Figure 7.1 shows the total execution time of the parallel IRAM-FDFD solver vs. the number of processors involved in the computation. For convenience, the same results for selected numbers of processors have also been shown in Table 7.1 along with parallel efficiency of the solver (cf. Section 5.1). Figures 7.2 and 7.3 show the speedups in the execution time of the parallel calculation of the matrix-vector (OP*x) product and the

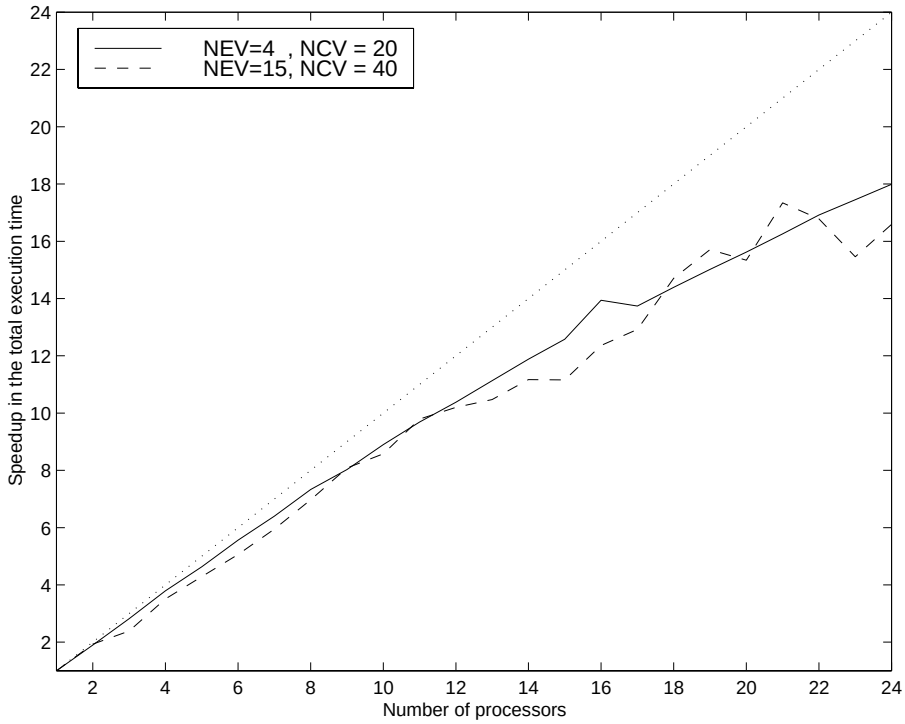


Figure 7.3: Speedup in the total execution time of the parallel IRAM-FDFD solver as a function of the number of processors involved in the computation. The tests were performed in the Cray T3E system. The dotted line shows perfect linear speedup.

total time used by the solver. It may be noted that the speedup while calculating the matrix-vector product is almost perfect. This very good result is the consequence of achieving appropriate workload balancing across the processors as well as limited inter-processor communication. These two goals could have been achieved using simple domain decomposition method due to highly regular FDFD matrix structure and low matrix bandwidth, respectively. It may also be observed that the speedup in total execution time is lower as compared to the speedup for the matrix-vector product operation, which means that overheads due to parallel execution are on the side of the parallel Arnoldi solver. Still, the speedup reaches 18 for 24 processors involved in the computation which is a fairly good result.

Another thing which may be noted in Figure 7.3 is a relatively unstable performance of the solver for NEV=15 and NCV=40 (cf. list of symbols). This effect is due to a different number of both Arnoldi update iterations and matrix-vector operations performed during the execution of the algorithm with different number of processors applied. This phenomenon does not occur for the parameters NEV=4 and NCV=20. Generally speaking, it has been noted that a variable number of iterations occurs if the problem becomes larger and if more iterations are necessary to obtain the convergence of the Arnoldi process. Still, this does not explain the dependence of the number of iterations on the number of

processors used. A possible explanation is that during the implicit updates and during the initial iteration of the Arnoldi factorization the vectors submitted to the iterative process are generated by each processor using only local data. In this case the global form of these vectors can be different for different number of processors applied. Consequently, the starting point of the iterative process before each implicit restart may be different for different number of processors involved in the computation.

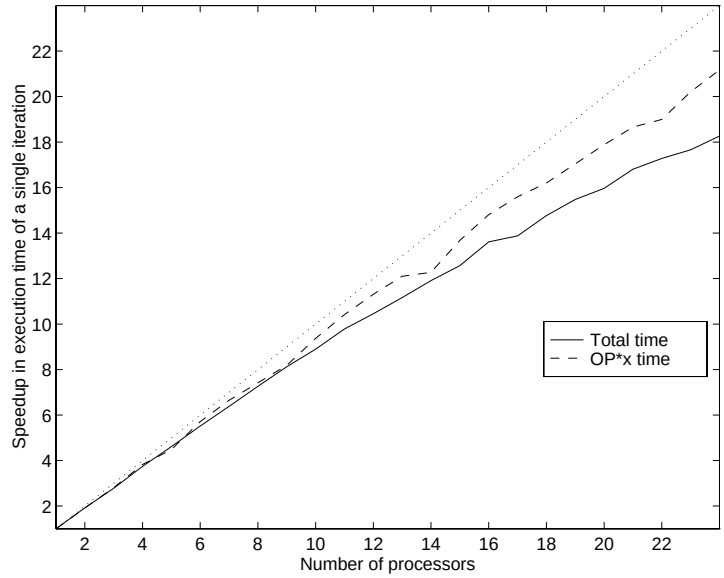


Figure 7.4: Speedup in execution time of a single iteration of the IRAM-FDFD algorithm and the single matrix-vector product computation for the case NEV=15, NCV=40. The tests were performed in the Cray T3E system. The dotted line indicates the linear speedup.

If now, for the case of NEV=15 and NCV=40, we compute the speedups in computation time per single iteration of the algorithm we shall obtain much more stable results, presented in Figures 7.4 and 7.5, which show a true speedup in computations due to parallelization. In fact the discussed graph of the speedup is almost identical to the case of NEV=4, NCV=20 (cf. Figures 7.2 and 7.3). The total parallel efficiency of a single iteration of the solver equals approximately 75%, which agrees with the result for NEV=4 and NCV=20.

Another aspect of parallel execution which has been investigated for the discussed parallel IRAM-FDFD solver is the workload balancing achieved for the applied parallel data distribution scheme. Figure 7.6 shows the relative difference of the execution times for the processors involved in a parallel computation. The investigated task was run on 24 processors and the differences (in %) are related to the execution time for the processor 1. It is apparent that the load balancing is almost perfect, with the largest

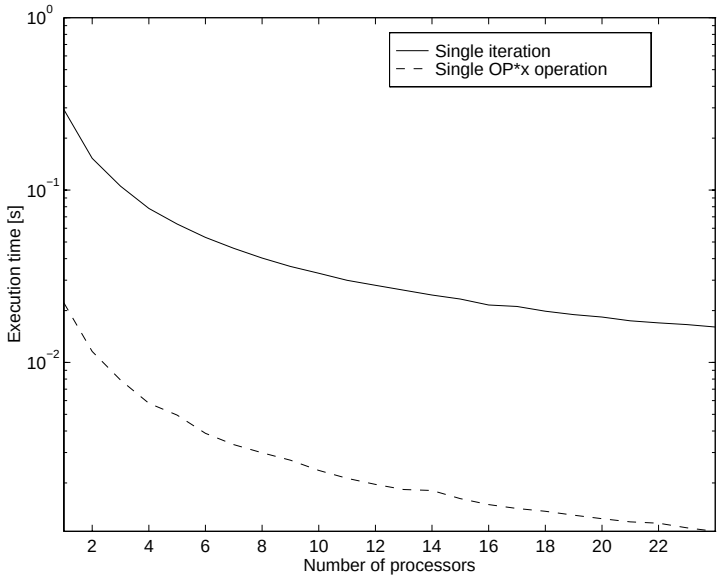


Figure 7.5: Execution time of a single iteration of the IRAM-FDFD algorithm and the single matrix-vector product computation in the function of the number of processors applied for the case: NEV=15, NCV=40. The tests were performed in the Cray T3E system.

relative difference in execution time equaling 0.4%. Consequently, it may be stated that an appropriate parallel data distribution scheme has been applied in the solver.

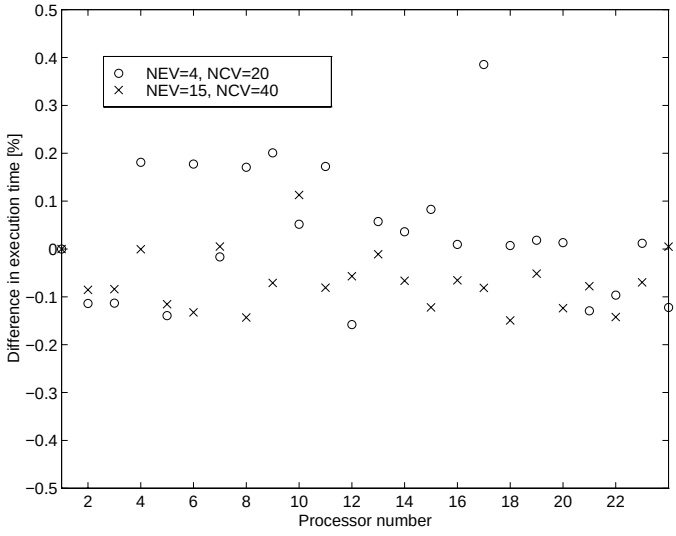


Figure 7.6: The per cent variation in execution times (as related to the execution time for processor 1) on different processors involved in a parallel computation for the IRAM-FDFD solver. The tests were performed in the Cray T3E system.

7.1.2 Parallel Arnoldi solver applied to modeling resonators

This section presents performance tests for the IRAM-FDFD solver applied to modeling resonators possessing rotational symmetry. In this case the operator matrix has a regular 11-diagonal form, customized for parallel distribution, as described in Section 5.3.2. During the numerical tests the matrix size equaled $N = 165100$, the number of eigenvalues to be found NEV = 30 and NCV = 60 (cf. list of symbols). The stopping criterion equaled $tol = 1 \cdot 10^{-6}$.

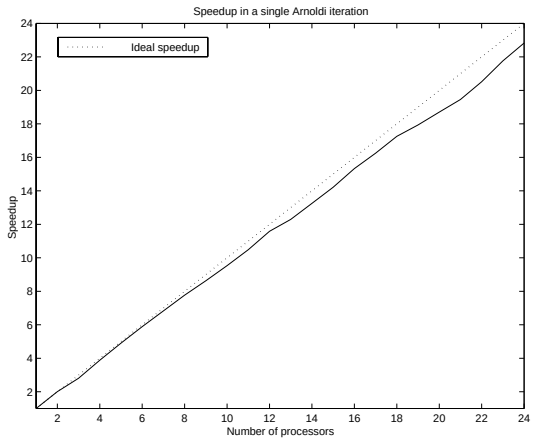


Figure 7.7: Scalability of the IRAM-FDFD solver in the Cray T3E parallel system. Matrix size $N = 165100$.

Figures 7.7 and 7.8 show, respectively, the speedup (per iteration) in the total execution time of the solver and speedup in computing the matrix-vector product. The total execution time for the IRAM-FDFD solver vs. the number of processors has been shown in Figure 7.9. The Figures show an almost perfect speedup of the considered parallel solver in a distributed memory system. In this case the parallel efficiency of the solver ranges from 100% to approximately 95%.

One may note that the speedup of the IRAM-FDFD solver applied to solving the current matrix problem is better than the speedup obtained for the IRAM-FDFD method applied to solving waveguiding problems, described in the previous section (compare Figure 7.3). The reason for this may be either the form of the operator matrix involved or the effect of scaling the problem. In the previous section the problem size equaled about 40000, while now it equals approximately 160000. The bandwidths of the involved matrices equal approximately 400 and 500, respectively. Consequently, *relatively* more time is spent on inter-processor communication (whose amount is proportional to the matrix bandwidth) in the first case than in the second case. This is most likely to be the main cause of the lower parallel speedup in the first case. The effect of matrix customization described in Section 5.3.2 is difficult to assess, due to the fact that if

'tapered end' appears, then the matrix looses its regular 11-diagonal structure and a different method of matrix-vector multiplication algorithm has to be applied.

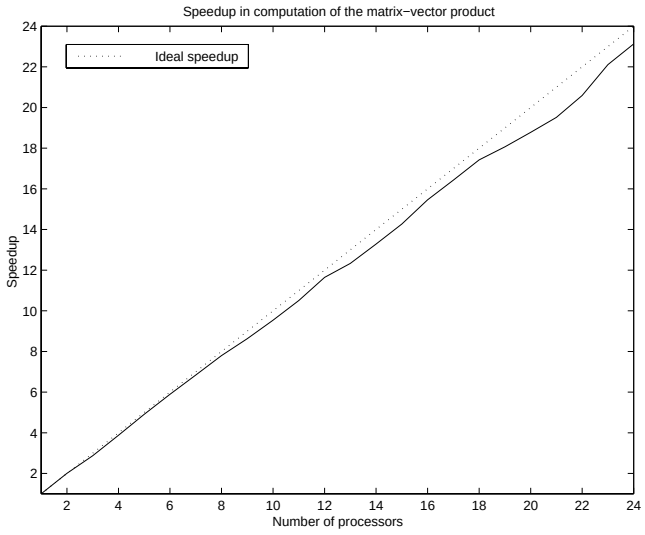


Figure 7.8: Scalability of the matrix-vector product in the Cray T3E parallel system for the IRAM-FDFD solver. Matrix size $N = 165100$.

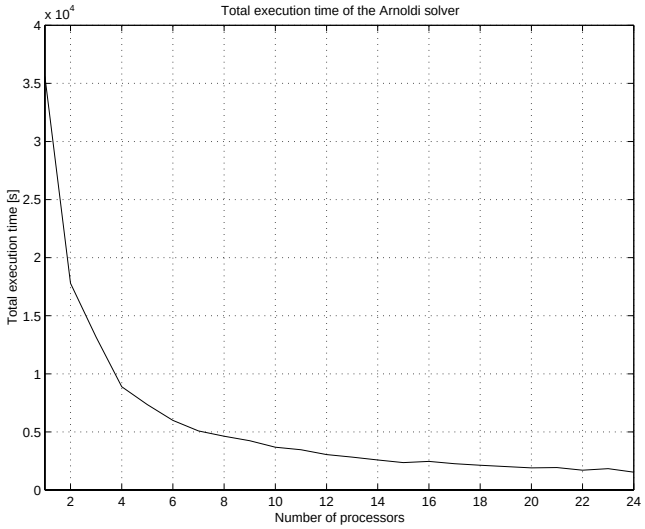


Figure 7.9: Total execution time of the IRAM-FDFD solver in the Cray T3E parallel system. Matrix size $N = 165100$.

7.2 Eigenfunction Expansion based algorithms

The performance tests of the parallel IRAM-FFT solver using implicit representation of operator and functional discrete representation (described in Section 5.4) have focused on measuring the speedup achieved by the program in given parallel distributed memory environments. All the following tests have been performed in the two scalable parallel systems: the IBM SP2 and the Cray T3E.

The first of the figures in this section (Figure 7.10) shows the execution times for the IRAM-FFT solver (number of the eigenvalues to be found NEV=8, size of the constructed Krylov subspace NCV=40) for different Fast Fourier Transform lengths and different number of expansion terms used to represent the functions in the input operator's domain. Analogous results are shown in Figure 7.11 for the tests performed in the Cray T3E system.

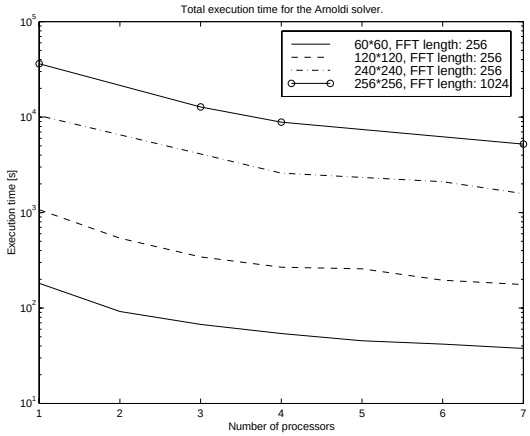


Figure 7.10: Total execution time of the IRAM-FFT parallel solver as a function of the FFT length. The tests have been performed in the IBM SP2 system. (NEV=8, NCV=40)

Both plots show comparable performance of the parallel solver. It may be noted that doubling the number of expansion terms (in every spatial direction) increases the total execution time roughly 10 times. This effect is clearly attributed to approximately four-fold increment in the computation time of the matrix-vector product and increment in the number of IRAM iterations, due to larger spectral radius. The results of the tests indicate that approximately two-fold increment in the number of iterations is observed, which is consistent with what we could expect due to the increased spectral radius.

Figure 7.12 shows the speedup in the total execution time of a single iteration of the IRAM-FFT solver in the Cray T3E platform. The best speedup may be observed in the case when the number of expansion functions equals 256 in every direction and the FFT length equals 1024 (in both directions). This indicates that applying a larger, more

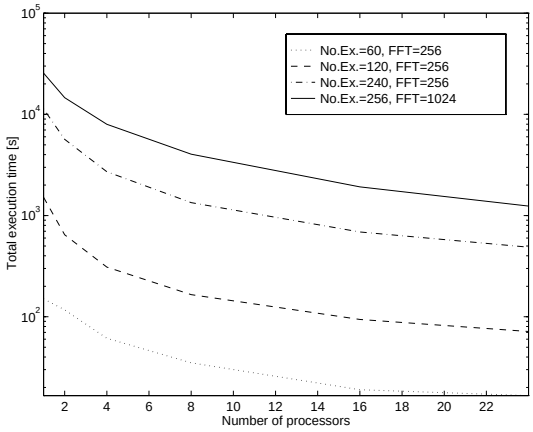


Figure 7.11: Total execution time of the IRAM-FFT parallel solver as a function of the FFT length. The tests have been performed in the Cray T3E system.

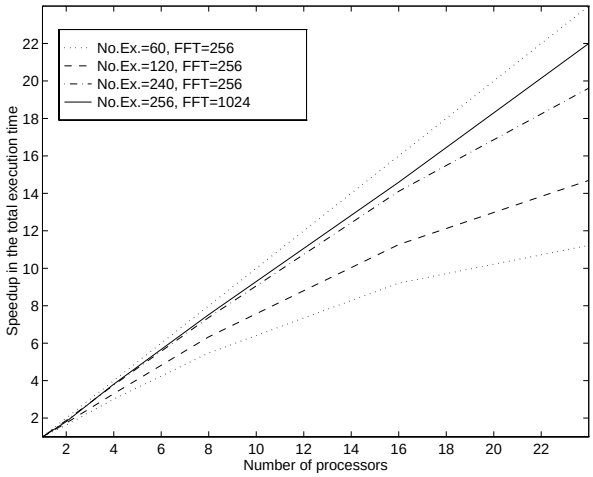


Figure 7.12: Speedup in the total execution time of a single iteration of the IRAM-FFT parallel solver vs. the number of processors. The tests have been performed in the Cray T3E system.

complex problem gives better performance. In other words, the solver scales well with the problem size. The other positive result which may be noted is that as the ratio between the number of expansion functions and the FFT length increases, the parallel performance also improves. This means that although the percentage of time spent on the matrix-vector product computation related to the total execution time increases and also the size of inter-processor communication during the parallel transposition operation becomes larger this does not cause a parallel communication bottleneck.

Figures 7.13 and 7.14 show the speedups in the execution time of a pair of operations: a backward 2D FFT and a forward 2D FFT, as a function of the number of processors

applied. This pair of operations is performed within each matrix-vector operation and includes virtually all inter-processor communication, therefore it is a suitable measure for assessing parallel performance of the solvers using implicit projection scheme based on FFTs. The speedups were computed for the average time of a single pair of the mentioned operations. It may be noted from Figure 7.14 that although the speedups are high, they are lower than the total speedup of the IRAM-FFT solver, which is an opposite situation as compared to the case of the parallel IRAM-FDFD solver.

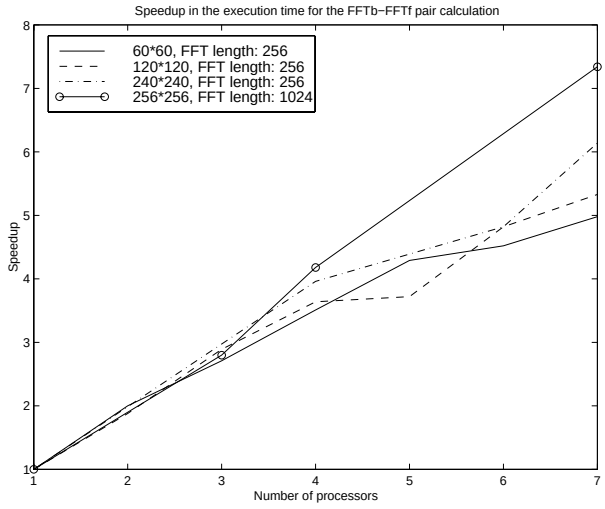


Figure 7.13: Speedup in the execution time of a pair of operations: a backward 2D FFT and a forward 2D FFT, as a function of the number of processors applied. The tests were performed in the IBM SP2 system.

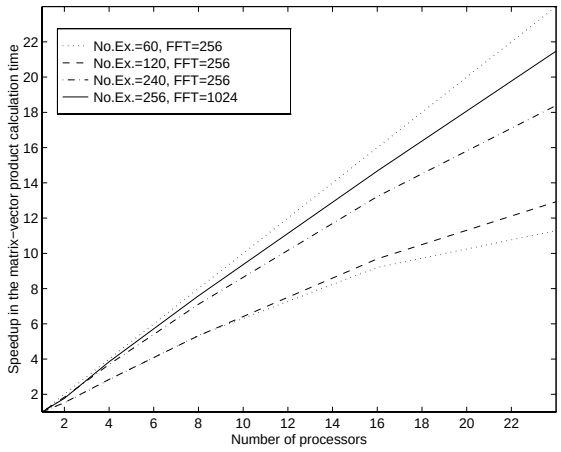


Figure 7.14: Speedup in the execution time of a pair of operations: a backward 2D FFT and a forward 2D FFT, as a function of the number of processors applied. The tests were performed in the Cray T3E system.

Number of expansion functions	Size of the operator matrix	Total execution time [s]
5×10	115	70.74
10×10	220	105.33
20×10	430	192.09
40×10	850	371.52
80×10	1690	892.69
160×10	3370	2216.21

Table 7.2: The total execution time (for one processor) of the IRAM-FFT solver for different numbers of expansion functions used to approximate the 2D fields by the Fourier expansions. The tests were performed in the IBM SP2 system (NEV=15, NCV=40, FFT length = 256).

DFT Length	Total execution time [s]
256×256	444.68
512×256	825.52
1024×256	1611.83
1024×512	4023.25
1024×1024	9235.12

Table 7.3: The total execution time (for one processor) of the IRAM-FFT solver for different discretization grids (FFT lengths). The number of expansion terms used to approximate the 2D fields equaled 20 in every spatial direction. The tests were performed in the IBM SP2 system.

In Tables 7.2 and 7.3 the total execution times (for a single-processor execution) of the IRAM-FFT solver are shown for different number of expansion functions used and different FFT lengths. The results confirm a rather stable behavior of the solver which is reflected in the linear or linear-logarithmic type of time growth with the growing problem size (N) or spatial domain size (FFT length). This type of growth may be contrasted with drastic time increment observed in the Galerkin Method (GM) (using the QR algorithm to find eigenvalues of the operator matrix) – compare Table 7.4. This last table shows the substantial difference in performance of the classical method (the Galerkin Method) in which an explicit representation of the input operator is applied producing a dense matrix and the proposed IRAM-FFT method which uses implicit operator representation.

Another series of tests compared performance of the IRAM-based solvers using explicit and implicit operator representation (cf. Section 4.2.1). Table 7.5 shows execution times for two solvers: IRAM-FFT - discussed above, using implicit operator projection and IRAM-GM - based on IRAM and using exactly the same operator discretization, as IRAM-FFT, still constructing the dense operator matrix explicitly by calculating its

Number of expansion functions:	Time [s] GM-QR:	Time [s] IRAM-FFT:
10 * 10	1.61	8.85
20 * 20	304.92	19.94
30 * 30	4254.86	38.22

Table 7.4: Comparison of the execution times (on one processor) between the Galerkin Method (GM) using QR solver and the IRAM-FFT solver. In case of the IRAM-FFT method: FFT length = 256, NEV=4, NCV=20. The tests were performed in the IBM SP2 system.

No. of expansion functions	Matrix generation time [s] IRAM-GM	Total time [s] IRAM-GM	Total time [s] IRAM-FFT
200	0.04	0.16	1.78
400	0.17	0.62	3.90
800	0.68	2.40	5.10
1600	2.75	13.96	11.53
3200	11.44	73.41	19.28
6400	46.17	521.53	40.06
7200	58.89	628.51	55.42

Table 7.5: Comparison of execution times of the IRAM-based solvers using explicit (IRAM-GM) and implicit (IRAM-FFT) operator projection.

elements. In both cases exactly the same discrete eigenproblem is being solved, so the number of implicit Arnoldi updates and matrix-vector products performed to obtain convergence also remains the same. The results show that for very small matrix sizes the solver using explicit matrix representation is faster than the solver using implicit matrix representation. For N=1600 (2 times 20×40 expansion terms) the IRAM-FFT solver becomes faster and for the matrix size N=7200 (2 times 60×60 expansion terms) its total execution time is shorter than the time needed to construct the explicit form of the matrix in the IRAM-GM algorithm! This result provides a clear argument in favor of using implicit operator representation rather than explicit representation in case of large scale operator eigenproblems.

7.3 Hybrid algorithms

This section presents results of parallel performance tests for the IRAM-HYBRID algorithm applying hybrid method of finite-dimensional operator projection in a 3D space defined in a cylindrical coordinate system. The algorithm has been described in Sections 4.3.1 and 5.5. Figure 7.15 shows the speedup in the execution time of a single iteration

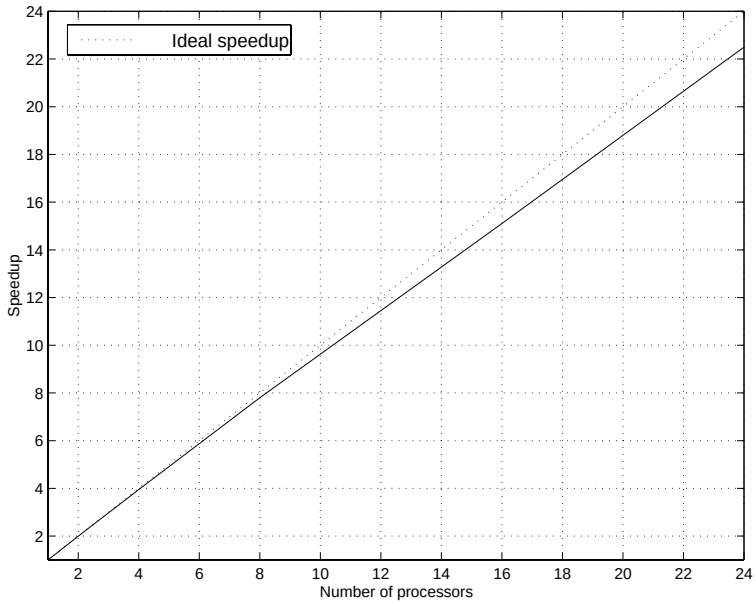


Figure 7.15: Speedup in a single iteration of the IRAM-HYBRID algorithm. Problem size $N = 45066$. The tests have been performed in the Cray T3E parallel system.

Number of processors	Total execution time [s]
1	13781.23
2	6898.75
4	3488.98
8	1766.79
16	912.66
24	612.51

Table 7.6: Speedup in a single iteration of the IRAM-HYBRID algorithm. Problem size $N = 45066$. The tests have been performed in the Cray T3E parallel system.

of the IRAM-HYBRID algorithm. In this case the global matrix problem size equaled $N = 45066$. The number of eigenvalues to be computed NEV=10 (NCV=40). Table 7.6 shows the total execution times for the solver. Once again, due to highly regular structure of the FDFD matrix and limited matrix bandwidth, good workload balancing and reduced amount of inter-processor communication have been achieved, which resulted in a nearly linear speedup of the parallel solver. One should note that this is also due to the fact that the Fast Fourier Transforms performed during matrix-vector product operation are computed only in one spatial dimension, which does not require any inter-processor communication.

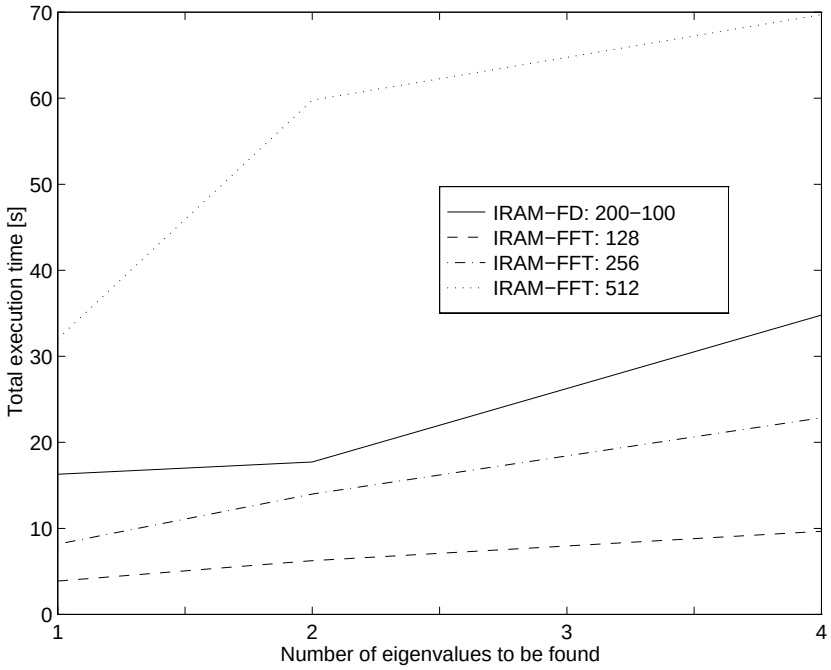


Figure 7.16: Comparison of single-processor execution times of IRAM-FDFD and IRAM-FFT solvers for different number of eigenvalues to be found and different FFT lengths applied. The tests have been performed in the Cray T3E system.

7.4 Comparison of performance of the proposed eigensolvers

In this section we shall present a short comparison of execution times of the IRAM-FFT and IRAM-FDFD algorithms. In the comparison we investigate single-processor execution times for the considered methods, applied to solve the same problem (in physical terms). The problem consists of finding propagation constants in one of the waveguiding structures discussed in Chapter 6 (cf. structure C in Figure 6.1). The test parameters were as follows: 1) For both algorithms the stopping criterion equaled $1.2e-16$. 2) In the case of the IRAM-FDFD algorithm the 200×100 discretization grid was used. 3) In the case of IRAM-FFT method the number of expansion functions equaled 40 in both x- and y- directions. The FFT lengths equaled 128, 256, 512 or 1024 in every direction. 4) For both algorithms the size of the constructed Krylov subspace equaled 20 (NCV=20) and the number of eigenfunctions to be found (NEV) equaled 1, 2 or 4. With this choice of input parameters one may expect that the quality of approximations of eigenvalues computed using the discussed solvers will roughly be the same.

The plot shown in Figure 7.16 presents the execution times for the IRAM-FDFD algorithm and IRAM-FFT method (for different FFT lengths) as a function of the number of eigenvalues to be found. One may note that the IRAM-FFT algorithm is faster if the

NEV	Problem size	No. of updates	No. of OP*x operations	Total time [seconds]	OP*x time [seconds]
IRAM-FFT	(128×128)				
1	33024	142	2860	112.35	23.07
4	33024	168	5983	171.57	47.91
IRAM-FDFD	(200×200)				
1	79600	61	1240	80.27	5.56
4	79600	121	4264	191.24	19.10
IRAM-FDFD	(128×128)				
1	32512	34	700	18.42	1.31
4	32512	77	2688	50.02	5.03

Table 7.7: Comparison of the number of updates, the number of matrix-vector product operations (OP*x) performed by the IRAM process and the execution times for IRAM-FFT and IRAM-FDFD solvers. In the case of IRAM-FFT algorithm both FFT length and the number of expansion functions used equaled 128. For IRAM-FDFD algorithm the discretization grid equaled 200×200 or 128×128. In all cases: NCV=40 and NEV = number of eigenvalues to be found.

FFT length equals 128 or 256. Still, if FFT length equals 512 then the IRAM-FDFD algorithm appears to be twice as fast as the IRAM-FFT method. It should be stressed here that the number of update iterations of the IRAM process did not change at all with the changing FFT lengths in the IRAM-FFT algorithm. It means that the growth in execution time while changing the FFT length is due solely to the increasing execution time of calculating the inner products (cf. Section 4.2.2). Another interesting observation can be made about the presented results. It is apparent that the execution time grows faster for the IRAM-FDFD algorithm than for the FFT-based algorithm with the growing number of eigenvalues to be found. It is not known whether this tendency is stable or for what range of parameters it occurs, as the computational complexity of the IRAM-FDFD solver is lower than the cost of FFT-based algorithms. Still, due to smaller size of the problem solved in the IRAM-FFT method (as compared to the IRAM-FDFD method) the growth of the number of iterations of the IRAM process necessary to obtain convergence is not so dynamic as in the IRAM-FDFD method and compensates the higher complexity of the IRAM-FFT algorithm.

Table 7.7 shows a comparison of the number of update iterations and OP*x operations for the IRAM-FFT and IRAM-FDFD algorithms if one and four eigenvalues are to be computed. One may note that while for the IRAM-FFT algorithm the number of update iterations increases by less than 20% (for NEV=4), it doubles for the IRAM-FDFD algorithm. The increment in the number of matrix-vector products (inner products) to be computed also changes more rapidly for the IRAM-FDFD algorithm. Consequently,

the growth in execution time is also faster for the IRAM-FDFD compared to IRAM-FFT solver. Lastly, it should be noted that the time spent on calculating the matrix-vector product does not exceed 10% of the total execution time of the IRAM-FDFD solver, while for the IRAM-FFT solver this percentage may range from about 20% to more than 90%. This fact is a consequence of transferring the complexity of the solver from the IRAM iterative process to the operation of computing inner products (using 2D FFTs) in the IRAM-FFT algorithm.

7.5 Summary

This chapter presented results of performance tests for the following parallel numerical solvers: IRAM-FDFD (using IRAM and Finite Difference-based projection), IRAM-FFT (using IRAM and implicit projection based on eigenfunction expansions), IRAM-HYBRID (using IRAM and hybrid type of operator projection, described in Section 4.3.1. The following general observations can be made, summarizing the obtained results:

- All the parallel solvers are characterized by very good speedup in distributed memory scalable systems.
- The algorithms show good overall performance if applied to large scale computational problems.
- Using implicit operator projection schemes may be extremely advantageous in terms of obtained performance while solving large problems.
- A comparable performance of IRAM-FFT and IRAM-FDFD algorithms has been observed.

Chapter 8

Conclusions

Previous chapters presented a discussion of a number of aspects arising in numerical solving of operator Boundary Value Problems, which constitute one of the most important classes of problems in scientific computing. The main context of this discussion, concerning methods of solving operator eigenproblems (in Chapter 3), techniques of finite-dimensional operator projection (in Chapter 4) and scalable, parallel design problems (in Chapter 5), was a broadly understood large scale electromagnetic modeling. This term has been perceived throughout this work as modeling of electromagnetic systems which are characterized by complicated geometry, complex material characteristics or relatively large dimensions, so that application of classical numerical techniques is strongly limited or simply impossible.

Within this study a number of new numerical algorithms have been constructed in a systematic manner by imposing various requirements, defining admissible eigenproblem solution methods, finite-dimensional operator projection techniques or parallelization strategies. Special attention has been paid to the following factors, related to large scale numerical modeling:

- Numerical complexity of algorithms solving discrete operator (matrix) eigenproblems, which plays particularly important role if the problem sizes become very large (e.g. over 10^5 unknowns)
- Memory storage requirements for a given discrete operator, often determining the largest possible size of the problem domain to be modeled.
- Complexity of the finite-dimensional operator projection technique.
- Accuracy of operator approximation associated with the projection method.
- Properties of the constructed discrete operator, including its size and spectral radius, spectrum characteristics, sparse or dense character of the operator matrix (if operator is represented explicitly).

Additionally, in the context of parallel processing, a few other factors characterizing a numerical solver in scalable computer systems have been considered, including:

- Properties of the operator matrix, particularly matrix bandwidth, sparsity and regularity of distribution of non-zero elements, which have impact on:
- Amount of inter-processor communication needed to complete the computations
- Locality of parallel computations.
- Balancing of the workload across the processing elements.

The formulated general goals or requirements to be satisfied by the constructed algorithms included:

- Assuring low memory and computational complexity of both projection and solution algorithms.
- Providing adequate accuracy of approximation of the initial operator, associated with the finite-dimensional projection method.
- Reducing the spectral radius of the discrete operator.
- Assuring high regularity and low bandwidth of the matrix associated with a discrete operator.

With these general goals to be met, the highly efficient scalable iterative Krylov subspace methods (e.g. IRAM) of solving operator eigenproblems have been selected as a basis for developing the following original numerical techniques:

- Implicit finite-dimensional projection techniques allowing substantial reduction of memory cost associated with solving an eigenproblem.
- FDFD projection algorithms producing sparse, highly regular banded matrix operators allowing efficient parallel mapping and reduction of the size of inter-processor communication.
- Techniques of regularization of operator matrix, allowing better load balancing and parallel data distribution.
- Correction method used to reduce the effects of numerical dispersion (and consequently reduce numerical errors) arising in large scale problems involving operators constructed the FDFD projection method. The proposed technique does not increase the size of the problem and does not increase significantly the numerical cost of the method.
- Finite-dimensional projection techniques based on eigenfunction expansions which generally result in problems with reduced size as compared to problems generated by FDFD method and allow one to deal with problems of numerical accuracy e.g. by applying oversampling.

- Hybrid finite-dimensional projection method, allowing one to reduce problem size and spectral radius while modeling 3D structures in cylindrical coordinates.
- Algorithm of accelerating convergence of Krylov subspace algorithms by applying Chebyshev polynomials (after [74], [75]).

Additionally a number of formulations of electromagnetic eigenproblems which allow reduction of 3D problems to 2D problems and reduction of the number of variables have been derived.

As shown in Chapter 6, the presented techniques have been successfully applied to model a number of electromagnetic systems including:

- dielectric waveguides,
- large cylindrical and hemispherical resonators.

The results of validation tests clearly indicate that relevant methods have been proposed and used to model discussed electromagnetic problems. Due to a number of advanced techniques applied, including acceleration techniques or correction of numerical dispersion, the complex large scale problem of modeling open hemispherical resonator has been efficiently solved. Until recently, this problem remained intractable due to excessive computational costs of classical, orthodox numerical methods.

As shown in Chapter 7, the proposed numerical methods are capable of solving large scale problems very efficiently. This refers either to serial performance which is often incomparably better as compared to performance of some classical techniques as well as parallel performance. The tests performed in massively parallel distributed memory supercomputers applying the parallel implementations of the discussed algorithms indicate their high efficiency and scalability in these processing environments. This considerably broadens the scope of application of the proposed methods to modeling very large, 'grand challenge' electromagnetic problems.

The results of both validation and performance tests clearly show that the goals set forth during construction of the proposed numerical algorithms constitute in fact key requirements for a good, high performance parallel numerical algorithm, applicable to solving large scale electromagnetic problems. These requirements may be summarized below in the form of the following guidelines:

- Apply low cost numerical algorithms of solving discrete operator eigenproblems.
- Avoid explicit representation of a discrete operator – use implicit operator schemes allowing reduction of memory cost.
- Apply operator formulations which allow reduction of the number of variables.

- Apply numerical techniques which enhance convergence and accuracy of the solutions without increasing the problem size.
- Apply solution techniques which allow one to embed projection procedures into the solution of a numerical problem.
- Apply finite-dimensional projection methods which accurately (in a certain sense) approximate the initial operator and at the same time assure that the discrete operator has a number of desired properties, including:
  - Reduced spectral radius.
  - Reduced bandwidth (enhancing locality of data and parallel computations).
  - Regular pattern of distribution of non-zero elements (if explicit representation of a discrete operator is used).
  - Sparse character (in the case explicit representation).

Summing up, the presented results prove that methodologies and requirements identified in the thesis of this study provide means to construct high performance parallel algorithms capable of dealing with large scale problems arising in electromagnetic modeling. The successful application of the proposed original methods to a number of electromagnetic problems arising in scientific and engineering domains indicates that the main goal of this work has been achieved.

Acknowledgments

First of all, I wish to thank very much Professor Michał Mrozowski for his important support, numerous valuable discussions concerning crucial problems arising during analysis and design of the algorithms proposed in this work as well as constant encouragement which enabled me to obtain the results presented in this study.

I am grateful to Jacek Mielewski and Andrzej Ćwikła for providing sequential codes of the Finite Difference solver in two dimensions and the Chebyshev preconditioner which have served as a basis for further developments of the methods and their parallel implementations, as well as for help concerning issues in modeling of hemispherical resonators.

I also acknowledge the support of the Academic Computer Centre TASK in Gdańsk and the Interdisciplinary Centre for Mathematical and Computational Modelling of the University of Warsaw in facilitating the access to supercomputer systems which served as platforms for all the numerical tests presented in this work.

Appendix A: Derivation of the operator eigenproblem for resonant cavities

This appendix presents an analogue of derivation of operator eigenproblem (2.51) presented in Section 2.1.2, using expanded forms of vector equations. The main reason of presenting this 'practical' derivation, which basically repeats the steps presented in Section 2.1.2 is that it provides a reference for developing (in Chapter 4) an algorithm of computing a matrix vector product $\underline{\underline{T}}\,\underline{v}$, where $\underline{\underline{T}}$ is a discrete analogue of operator from equation (A.18) (or (2.52)).

The derivation below refers to a simplified case when the permeability and permittivity tensors are diagonal:

$$\overleftrightarrow{\epsilon} = \mathrm{diag}[\epsilon_r, \epsilon_\phi, \epsilon_z], \quad \overleftrightarrow{\mu} = \mathrm{diag}[\mu_r, \mu_\phi, \mu_z] \tag{A.1}$$

The assumed form of both electric and magnetic field defined in (r, ϕ, z) coordinates is:

$$\vec{D}(r, \phi, z) = \vec{D}_n(r, z)e^{(-jn\phi)} \quad \text{and} \quad \vec{B}(r, \phi, z) = \vec{B}_n(r, z)e^{(-jn\phi)} \tag{A.2}$$

where n is an integer number.

One starts with Maxwell's curl equations (2.3) and (2.4) expanded in cylindrical coordinates:

$$\frac{1}{r}\frac{\partial E_z}{\partial \phi} - \frac{\partial E_\phi}{\partial z} = -j\omega\mu_0\mu_r H_r \tag{A.3}$$

$$\frac{\partial E_r}{\partial z} - \frac{\partial E_z}{\partial r} = -j\omega\mu_0\mu_\phi H_\phi \tag{A.4}$$

$$\frac{1}{r}\frac{\partial (rE_\phi)}{\partial r} - \frac{1}{r}\frac{\partial E_r}{\partial \phi} = -j\omega\mu_0\mu_z H_z \tag{A.5}$$

$$\frac{1}{r}\frac{\partial H_z}{\partial \phi} - \frac{\partial H_\phi}{\partial z} = j\omega\epsilon_0\epsilon_r E_r \tag{A.6}$$

$$\frac{\partial H_r}{\partial z} - \frac{\partial H_z}{\partial r} = j\omega\epsilon_0\epsilon_\phi E_\phi \tag{A.7}$$

$$\frac{1}{r}\frac{\partial (rH_\phi)}{\partial r} - \frac{1}{r}\frac{\partial H_r}{\partial \phi} = j\omega\epsilon_0\epsilon_z E_z \tag{A.8}$$

The divergence equation for electric flux density takes the following form:

$$\frac{1}{r}\frac{\partial (rD_r)}{\partial r} + \frac{1}{r}\frac{\partial D_\phi}{\partial \phi} + \frac{\partial D_z}{\partial z} = 0 \tag{A.9}$$

Substituting the $\vec{D}$ field in the form (A.2) into the above equation yields a formula for D_n^ϕ:

$$D_n^\phi = \frac{-j}{n}\frac{\partial\,(rD_n^r)}{\partial r} - \frac{jr}{n}\frac{\partial D_n^z}{\partial z} \tag{A.10}$$

In the same way one may transform equations (A.3)-(A.5):

$$\frac{-jn}{r}E_n^z - \frac{\partial E_n^\phi}{\partial z} = -j\omega\mu_0\mu_r H_n^r \tag{A.11}$$

$$\frac{\partial E_n^r}{\partial z} - \frac{\partial E_n^z}{\partial r} = -j\omega\mu_0\mu_\phi H_n^\phi \tag{A.12}$$

$$\frac{1}{r}\frac{\partial (rE_n^\phi)}{\partial r} - \frac{-jn}{r}E_n^r = -j\omega\mu_0\mu_z H_n^z \tag{A.13}$$

where $E_n^r = (\epsilon_0\epsilon_r)^{-1}D_n^r$, $E_n^\phi = (\epsilon_0\epsilon_\phi)^{-1}D_n^\phi$, $E_n^z = (\epsilon_0\epsilon_z)^{-1}D_n^z$.

Using equations (A.10)-(A.13) one may derive formulae for H_n^r, H_n^ϕ and H_n^z involving only D_n^r and D_n^z field components. The formulae may then be substituted to the following equations, derived from (A.6) and (A.8):

$$\frac{-jn}{r}H_n^z - \frac{\partial H_n^\phi}{\partial z} = j\omega\epsilon_0 D_n^r \tag{A.14}$$

$$\frac{1}{r}\frac{\partial (rH_n^\phi)}{\partial r} - \frac{-jn}{r}H_n^r = j\omega\epsilon_0 D_n^z \tag{A.15}$$

where $H_n^\phi = (\mu_0\mu_\phi)^{-1}B_n^\phi$ and $H_n^z = (\mu_0\mu_z)^{-1}B_n^z$.

Performing mentioned substitutions yields the following pair of equations:

$$\begin{aligned}\omega^2 D_n^r &= \frac{c^2}{\mu_z}\left[\frac{n^2}{r^2}\frac{1}{\epsilon_r}D_n^r + \frac{1}{r^2}\frac{\partial}{\partial r}\left(\frac{r^2}{\epsilon_\phi}\left(\frac{1}{r}\frac{\partial(rD_n^r)}{\partial r} + \frac{\partial D_n^z}{\partial z}\right)\right)\right] \\ &\quad -\frac{c^2}{\mu_\phi}\frac{\partial}{\partial z}\left[\frac{\partial}{\partial z}\left(\frac{1}{\epsilon_r}D_n^r\right) - \frac{\partial}{\partial r}\left(\frac{1}{\epsilon_z}D_n^z\right)\right] \qquad \text{(A.16)}\end{aligned}$$

$$\begin{aligned}\omega^2 D_n^z &= \frac{c^2}{\mu_r}\left[\frac{n^2}{r^2}\frac{1}{\epsilon_z}D_n^z - \frac{\partial}{\partial z}\left(\frac{1}{\epsilon_\phi}\left(\frac{1}{r}\frac{\partial(rD_n^r)}{\partial r} + \frac{\partial D_n^z}{\partial z}\right)\right)\right] \\ &\quad +\frac{c^2}{\mu_\phi}\frac{1}{r}\frac{\partial}{\partial z}\left[r\frac{\partial}{\partial z}\left(\frac{1}{\epsilon_r}D_n^r\right) - r\frac{\partial}{\partial r}\left(\frac{1}{\epsilon_z}D_n^z\right)\right] \qquad \text{(A.17)}\end{aligned}$$

The above set of equations may be written using compact operator notation:

$$\mathbf{T}\Theta = \omega^2\Theta \qquad \text{(A.18)}$$

where $\Theta = \left[\begin{array}{cc} D_n^r & D_n^z \end{array}\right]^T$. As already pointed out, it is an analogue of equation (2.52), also involving two field components D_n^r and D_n^z.

Appendix B: The P_ARPACK library: The Arnoldi solver for MPP platforms

This appendix presents a short description of the Parallel ARnoldi PACKage (P_ARPACK) – a portable library implementing the Implicitly Restarted Arnoldi Method (IRAM) for distributed memory parallel systems.

The P_ARPACK software has been developed at Rice University (cf. [69]) and provides a versatile package of Fortran77 subroutines for solving either symmetric or non-symmetric, real or complex matrix eigenproblems. The important feature of the Arnoldi algorithm which has been exploited in the design of the library routines is that the method does not require any explicit form of the input operator matrix to be used. Instead, all the information on the considered operator is passed via the matrix-vector product. This has been used by introducing the *reverse communication* interface. On one hand, this interface enables the subroutines that perform the Arnoldi algorithm iteration to be independent of the input matrix storage format and, on the other hand, it makes the user of P_ARPACK free to choose the most appropriate method of computing the matrix-vector product for a specified input matrix operator. The general framework of a parallel program calling P_ARPACK routines in a reverse communication loop is shown in Figure B.1 and constitutes a basis for the solvers presented in Chapters 5 and 6.

The central point of the presented program is a call to the `pdnaupd`() subroutine which implements the IRAM algorithm for a non-symmetric real eigenproblem. This call is preceded by the initialization of various parameters defining the problem, including: `n` – the global problem size, `nloc` – the local problem size for a given processor (process), `nev` – the number of eigenvalues to be found, `bmat` – type of the eigenproblem (standard/generalized), `which` – which part of the operator spectrum is to be considered (e.g. eigenvalues with the largest real part or the largest modulus). The `info` parameter determines whether an initial vector v_1 will be submitted. If `info=1` the initial vector is stored in the `resid` parameter. Otherwise, the initial vector is random. The `tol` parameter determines the stopping criterion for the Arnoldi factorization. The algorithm stops if the condition:

$$||Au_i - u_i\lambda_i||_2 \leq tol \cdot |\lambda_i| \tag{B.1}$$

```
c --------- Parameter selection for pdnaupd() -------------
c
      comm = MPI_COMM_WORLD ! Set the communicator
      call MPI_Comm_size(comm,  ! Determine the number of
                nprocs, ierr)   ! processors used
      n     = N      ! size of the problem
      nev   = NEV    ! number of eigenvalues to be computed
      ncv   = NCV    ! number of orthogonal columns of V
      nloc = n/nprocs ! Determine local size of the problem
      bmat  = 'I'     ! standard eigenvalue problem
      which = 'LM'    ! find eigenvalues with largest magn.
      tol    = 1.e-8  ! set the desired accuracy
      ido    = 0      ! first call to reverse communication
      info   = 1      ! resid contains the initial vector
      do 100 i = 1, nloc ! initialize resid as a vector
         resid(i) = 1.d0 ! with 1's as all elements
 100  continue
      iparam(1) = 1    ! exact shifts with respect to H
      iparam(3) = 1000 ! maximum number of updates
      iparam(7) = 1    ! Mode set to 1
c
c ------------------ Reverse communication loop ----------
c
 200  continue
         call pdnaupd(comm, ido, bmat, nloc, which, nev,
     &               tol, resid, ncv, v, ldv, iparam,
     &       ipntr, workd, workl, lworkl, info )
c
         if (ido .eq. -1 .or. ido .eq. 1) then
c           Compute  matrix-vector product: A*v
            call Av(nloc, workd(ipntr(1)), workd(ipntr(2))
            go to 200 ! Loop back to call pdnaupd() again
         endif
c ------------------------------------------------------
```

Figure B.1: Calling `pdnaupd()` P_ARPACK subroutine, solving a non-symmetric real eigenproblem in a reverse communication loop.

is satisfied for all λ_i. Parameters `iparam(1)` – `iparam(8)` define various options of the algorithm including the maximum number of Arnoldi updates allowed or types of shifts used in the polynomial filtering process. A detailed description of all the parameters of P_ARPACK routines may be found in [67].

Another important design feature of the P_ARPACK library is the possibility of applying the Single Program Multiple Data (SPMD) programming style, regarded the most efficient and transparent in the parallel message-passing programming. This programming technique allows one to write a single code (such as shown in Figure B.1) to be executed on all the processors. Once again the reverse communication interface to the P_ARPACK subroutines allows the user to choose a convenient parallelization strategy for the matrix-vector product operation.

Last but not least, the P_ARPACK library offers portability across a wide range of distributed memory parallel systems (including networks of workstations) by implementing its parallel routines using standard inter-processor communication libraries: the Message Passing Interface (MPI) ([77]) and the Basic Linear Algebra Communication Subprograms (BLACS) ([112]).

Appendix C: Matrix preconditioning using Chebyshev polynomials.

This appendix presents a method of matrix preconditioning applying Chebyshev polynomials. The method provides means to modify the matrix spectrum in order to enable one to find eigenvalues from a desired part of the spectrum (e.g. the center of the spectrum) by using Krylov subspace methods, such as Arnoldi algorithm, basically suitable only for computing several eigenvalues from either edge of the matrix spectrum. Consequently, the method significantly broadens the range of applications of Krylov subspace based eigensolvers to problem where higher order modes are to be found. At the same time it accelerates the convergence of the solver, as indicated in [102].

In the presented method the preconditioning is based on filtering the spectrum with an FIR bandpass digital filter. For a given center frequency f_0, determined by the range of eigenvalues to be found, a linear phase 80-th (or 60-th) order FIR filter with a flat monotonically decreasing pass band and an equiripple stop band is designed using Remez exchange algorithm. Figure C.1 shows an example of a frequency response of a filter designed for the center frequency $f_0 = 57.59$ GHz.

The design procedures for the FIR filters generally use the normalized ω domain, i.e. $\omega \in [0, \pi]$. Consequently, the filter has to be designed for the center frequency which equals:

$$\omega_0 = \arccos\left(\frac{f_0 - c}{\rho}\right) \tag{C.1}$$

where ρ is a spectral radius of the investigated matrix and c is the center frequency of the matrix spectrum.[1] Clearly, the entire frequency domain has to be transformed according to the formula:

$$\omega = \arccos\left(\frac{f - c}{\rho}\right) \tag{C.2}$$

[1]Both c and ρ may be found by computing numerically the matrix eigenvalues with the largest and the smallest modulus, which may be performed at relatively low cost with any of the basic Krylov subspace algorithms.

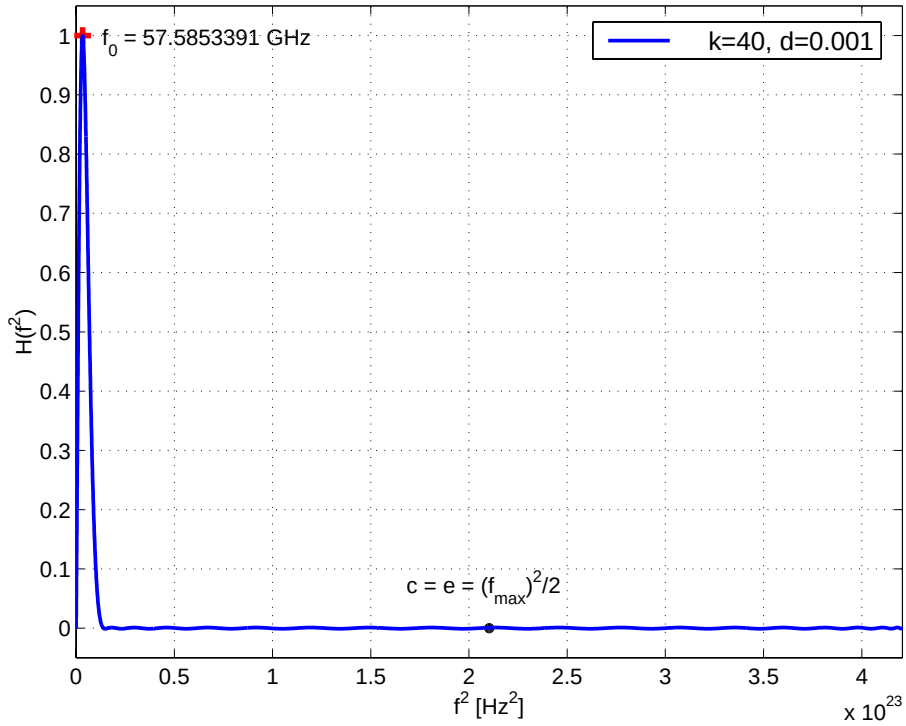


Figure C.1: Example of the frequency response of a 80-th order FIR filter, designed for the center frequency $f_0 = 57.59$ GHz and the ripple in a stop band $d = 0.001$. This figure has been adapted from Ph.D. Thesis by Jacek Mielewski [75].

The frequency response of the designed FIR filter of order $(2k)$ can be written in the form of the following series:

$$H(\omega) = \sum_{n=0}^{k} b_n \cos(n\omega) \tag{C.3}$$

where b_n are coefficients related to the impulse response of the filter. Substituting (C.2) to (C.3) yields:

$$H(f) = \sum_{n=0}^{k} b_n \cos\left(n \arccos\left(\frac{f-c}{\rho}\right)\right) = \sum_{n=0}^{k} b_n T_n\left(\frac{f-c}{\rho}\right) \tag{C.4}$$

where T_n is the n-th order Chebyshev polynomial.

If $\underline{\underline{A}}$ denotes a given matrix having a spectrum $\sigma(\underline{\underline{A}})$, then by trivial observation the matrix $\underline{\underline{B}} = H(\underline{\underline{A}})$ has the spectrum:

$$\sigma(\underline{\underline{B}}) = \left\{H(\lambda) : \lambda \in \sigma(\underline{\underline{A}})\right\} \tag{C.5}$$

where polynomial $H(\cdot)$ is given by (C.4). Suppose the frequency response for the designed filter is as shown in Figure C.1 and all the eigenvalues of matrix $\underline{\underline{A}}$ are included in the

shown frequency range. Let us consider eigenvalues of matrix $\underline{\underline{A}}$ located close to the frequency point referring to the peak value (f_0) of polynomial $H(f)$. According to the above formula, these eigenvalues of matrix $\underline{\underline{A}}$ will transform to 'very large' eigenvalues of matrix $\underline{\underline{B}}$. In other words they will relate (through polynomial transformation $H(\lambda)$) to largest modulus eigenvalues of matrix $\underline{\underline{B}}$. This relation becomes well established if we note that the eigenvectors of $\underline{\underline{A}}$ and $\underline{\underline{B}}$ are identical. (Matrix eigenvectors are invariant to the polynomial transformation of a matrix.)

As already discussed in Chapter 3, the Krylov subspace algorithms of solving operator eigenproblems generally do not need the operator to be defined explicitly. This is the consequence of the fact that the information on the operator is passed to the solver only via the $\mathbf{A}v$ products, where v are subsequent iterates of the solver. Consequently, application of the presented polynomial preconditioner given by (C.4) within the Krylov subspace eigensolver does not require the matrix $H(\underline{\underline{A}})$ to be constructed explicitly.

Computing the matrix-vector product $H(\underline{\underline{A}})\underline{v}$, where $\underline{v}$ is an arbitrary vector may be performed using a trivial recursive algorithm. Given c and ρ and denoting as $\tilde{b}_i$ the coefficients of the polynomial $H(\omega)$ in the basis $\{1, \omega, \omega^2, \ldots, \omega^k\}$ so that:

$$\sum_{n=0}^{k} \tilde{b}_n \omega^i \equiv H(\omega) \tag{C.6}$$

the algorithm for computing the matrix-vector product takes the following form:

> Algorithm 5: The Matrix-vector product with Chebyshev polynomial preconditioning.
>
> Step 1: $\underline{w} := \underline{v}$.
>
> Step 2: $\underline{w} := \tilde{b}_k \underline{w}$.
>
> Step 3: For $n = (k-1)$ to 0 iterate:
>
> Step 3.1: $\underline{wtemp} := \underline{\underline{A}}\,\underline{w}$.
>
> Step 3.2: $\underline{wtemp} := (\underline{wtemp} - c\underline{w})/\rho$.
>
> Step 3.3: $\underline{w} := \underline{wtemp} + \tilde{b}_n \underline{v}$

As already mentioned, preconditioning (C.4) preserves matrix eigenvectors. If $\underline{v}_i$ denotes an eigenvector of matrix $H(\underline{\underline{A}})$ found in a numerical procedure, then the corresponding eigenvalue of matrix $\underline{\underline{A}}$ may be found by computing a Rayleigh quotient:

$$\lambda_i = \frac{\underline{v}_i^H \underline{\underline{A}}\,\underline{v}_i}{||\underline{v}_i||^2} \tag{C.7}$$

where $||\cdot||$ denotes an Euclidean vector norm in C^n. Using the above formula one may find the desired eigenvalues of matrix $\underline{\underline{A}}$ by: 1) Finding eigenvectors of matrix $\underline{\underline{B}} = H(\underline{\underline{A}})$ corresponding to the largest modulus eigenvalues; and 2) Compute eigenvalues of $\underline{\underline{A}}$, using the previously found eigenvectors.

Bibliography

[1] S. Aditya, R. K. Arora, *A millimeter-wave open resonator with application in measurement of surface resistance of conducting materials*, IEEE Antennas and Propagation/URSI Symposium Digest, London, Ont., Canada, pp. 54-55, 1991.

[2] D. L. Alumbaugh, G. A. Newman, L. Prevost, J. N. Shadid, *Three-dimensional wideband electromagnetic modeling on massively parallel computers*, Radio Science, vol. 31, no. 1, pp. 1-23, Jan.-Feb. 1996.

[3] G. Angelidis, A. Semlyen, *Improved Methodologies for the Calculation of Critical Eigenvalues in Small Signal Stability Analysis*, IEEE Trans. on Power Systems, vol. 11, no. 3, Aug. 1996, pp. 1209-1215.

[4] W. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math. 9, 17-29, 1951.

[5] L. Auslander and A. Tsao, *On a divide and conquer algorithm for the eigenproblem via complementary invariant subspace decomposition*, Supercomputing Research Center Technical Report SRC-89-003, Bowie, USA, 1989.

[6] L. Auslander and A. Tsao, *On parallelizable eigensolvers*, Advances in Applied Mathematics 13, 253-261, 1992.

[7] K. Beilenhoff, W. Heinrich, H. L. Hartnagel, *Improved Finite-Difference Formulation in Frequency Domain for Three-Dimensional Scattering Problems*, IEEE Trans. Microwave Theory Tech., vol. 40, no. 3, Mar. 1992, pp. 540-546.

[8] H. Berryman, J. Saltz, W. Gropp, R. Mirchandaney, *Krylov Methods Preconditioned with Incompletely Factored Matrices on the CM-2*, Journal of Parallel and Distributed Computing, vol. 8, pp. 186-190, 1990.

[9] J. E. Bracken, Din-Kow Sun, Z. J. Cendes, *S-Domain Methods for Simultaneous Time and Frequency Characterization of Electromagnetic Devices*, IEEE Trans. Microwave Theory Tech. vol. 46, no. 9, pp. 1277-1290, Sep. 1998.

[10] W. Briggs, *Multigrid tutorial*, SIAM, Philadelphia, 1987.

[11] W. L. Briggs, Van Emden Henson, *The DFT. An Owner's Manual for the Discrete Fourier Transform*, SIAM, Philadelphia, 1995.

[12] P. N. Brown, A. C. Hindmarsh, L. R. Petzold, *Using Krylov Methods in the Solution of Large-Scale Differential-Algebraic Systems*, SIAM Journal of Scientific Computing, vol. 15, no. 6, pp. 1467-1488, Nov. 1994.

[13] F. W. Byron, R. W. Fuller, *Mathematics of classical and quantum physics*, Addison-Wesley, Reading (Polish edition: PWN, Warsaw, 1975).

[14] C. Le Calvez, Y. Saad, *Modified Krylov acceleration for parallel environments*, Applied Numerical Mathematics, vol. 30, pp. 191-212, 1999.

[15] H. Cam, S. Toutain, P. Gelin, G. Landrac, *Study of a Fabry-Perot cavity in the microwave frequency range by the Boundary Element Method*, IEEE Trans. Microwave Theory Tech., vol. 40, pp. 298-304, Feb. 1992.

[16] A. C. Cangellaris, L. Zhao, *Rapid FDTD Simulation Without Time Stepping*, IEEE Microwave and Guided Wave Letters, vol. 9, no. 1, pp. 4-6, Jan. 1999.

[17] M. Celik, A. C. Cangellaris, *Simulation of Multiconductor Transmission Lines Using Krylov Subspace Order-Reduction Techniques*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, vol. 16, no. 5, pp. 485-496, May 1997.

[18] M. Chou, J. White, *Efficient Reduced-Order Modeling for the Transient Simulation of Three-dimensional Interconnect*, ICCAD-95, 1995 IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, pp. 40-44, 1995.

[19] M. Clemens, T. Weiland, U. van Rienen, *Comparison of Krylov-Type Methods for Complex Linear Systems Applied to High-Voltage Problems*, IEEE Trans. on Magnetics, vol. 34, no. 5, Sep. 1998, pp. 3335-3338.

[20] J. W. Cooley, J. W. Tukey, *An algorithm for the machine evaluation of complex Fourier series*, Math. Comp. 19, pp. 297-301, 1965.

[21] T. Cwik, J. Partee, J. Patterson, *Method of Moment Solutions to Scattering Problems in a Parallel Processing Environment*, IEEE Transactions on Magnetics, vol. 27, no. 5, pp. 3837-3840, Sep. 1991.

[22] T. Cwik, Parallel Decomposition Methods for the Solution of Electromagnetic Scattering Problems, Electromagnetics, vol. 12, pp. 343-357, 1992.

[23] T. Cwik, D. S. Katz, *Scalable Solutions to Integral-Equation and Finite-Element Simulations*, IEEE Transactions on Antennas and Propagation, vol. 45, no. 3, pp. 544-555, Mar. 1997.

[24] T. Cwik, D. S. Katz, C. Zuffada, V. Jamnejad, *The application of scalable distributed memory computers to the finite element modeling of electromagnetic scattering*, Int. J. Num. Meth. Eng. , vol. 41, no. 4, pp. 759-776, Feb. 1998.

[25] T. Cwik, D. S. Katz, C. Zuffada, V. Jamnejad, *The Application of Scalable Distributed Memory Computers to the Finite Element Modeling of Electromagnetic Scattering*, International Journal for Numerical Methods in Engineering, vol. 41, pp. 759-776, 1998.

[26] A. Ćwikła, M. Mrozowski, *Zagadnienie aproksymacji warunk/ow brzegowych i granicznych dla siatki Yee*, Technical Report, Deptartmet of Electronics, Telecommunications and Computer Science, Technical University of Gdańsk, Gdańsk, 1999. (in Polish)

[27] A. Ćwikła, J. Mielewski, M. Mrozowski, J. Wosik, *Accurate Full Wave Analysis of Open Hemispherical Resonators Loaded with Dielectric Layers*, IEEE MTT-S Symposium Digest, vol. 3, pp. 1265-1268, 1999.

[28] R. Dautray, J.-L. Lions, *Mathematical Analysis and Numerical Methods for Science and Technology – Integral Equation and Numerical Methods*, vol. 4, Springer-Verlag, Berlin, 1990.

[29] D. B. Davidson, *Parallel Matrix Solvers for Moment Method Codes for MIMD Computers*, Applied Computational Electromagnetics Society Journal, vol. 8, no. 2, pp. 144-175, 1993.

[30] D. B. Davidson, *Parallel computing*, IEEE Potentials, vol. 14, no. 2, pp. 6-10, 1995.

[31] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices*, J. Comp. Phys., 17:87-94, 1975.

[32] J. B. Davies, F. A. Fernandez, Y. Fang, *Finite-Difference Solution of Inhomogeneous Waveguide Modes Using a Fast Direct Solver Routine*, IEEE Transactions on Magnetics, vol. 27, no. 5, Sep. 1991, pp. 4028-4031.

[33] M. P. Dębicki, P. Jędrzejewski, J. Mielewski, P. Przybyszewski, M. Mrozowski, *Application of the Arnoldi Method to the Solution of Electromagnetic Eigenproblems on the Multiprocessor Power Challenge Architecture*, Technical Report No. 19/95, Dept. of Electronics, Technical Univ. of Gdańsk, Gdańsk, 1995.

[34] M. P. Dębicki, P. Jędrzejewski, J. Mielewski, P. Przybyszewski, M. Mrozowski, *Solution of Electromagnetic Eigenproblems on the Multiprocessor Superscalar Architecture*, in Proceedings of the 12th Annual Review of Progress in Applied Computational Electromagnetics, Monterey, CA, pp. 413-420.

[35] P. Dębicki, P. Jędrzejewski, A. Kręczkowski, J. Mielewski, M. Mrozowski, K. Nyka, P. Przybyszewski, M. Rewieński, T. Rutkowski, *Coping with numerical complexity in computational electromagnetics*, Int. Microwave Symp. MIKON-98, Cracow, 1998.

[36] M. Dehler, *Numerische Lösung der Maxwellschen Gleichungen auf kreiszylindrischen Gittern*, Ph.D Thesis, Technische Hochschule Darmstadt, Darmstadt, 1993.

[37] H. Dong, A. Chronopoulos, J. Zou, A. Gopinath, *Vectorial Integrated Finite-Difference Analysis of Dielectric Waveguides*, IEEE Journal of Lightwave Technology, vol. 11, No. 10, Oct. 1993, pp. 1559-1564.

[38] I. M. Elfadel, D. D. Ling, *Zeros and Passivity of Arnoldi-Reduced-Order Models for Interconnect Networks*, in proceedings of the 34th Design Automation Conference, 1997, pp. 28-33.

[39] P. Feldman, R. W. Freund, *Efficient linear circuit analysis by Padé approximation via the Lanczos process*, IEEE Trans. on Computer-Aided Design, vol. 14, pp. 639-649, 1995.

[40] F. Fernandez, Y. Lu, *Microwave and Optical Waveguide Analysis by the Finite Element Method*, Research Studies Press, Taunton, Somerset, England, 1996.

[41] R. D. Ferraro, T. Cwik, N. Jacobi, P. C. Liewer, T. G. Lockhart, G. A. Lyzenga, J. Parker, J. E. Patterson, *Parallel Finite Elements Applied to Electromagnetic Scattering Problem*, Proceedings of the Fifth Distributed Memory Computing Conference, pp. 417-420, 1990.

[42] R. D. Ferraro, T. Cwik, N. Jacobi, P. C. Liewer, T. G. Lockhart, G. A. Lyzenga, J. Parker, J. E. Patterson, D. Simoni, *Parallel finite elements applied to 3D electromagnetic scattering problems*, IEEE Antennas and Propagation Society International Symposium, 1990. AP-S Digest, vol. 4, pp. 1360-1363, 1990.

[43] I. Foster, *Designing and Building Parallel Programs*, Addison-Wesley, Reading, 1995

[44] A. T. Galick, T. Kerkhoven, U. Ravaioli, *Iterative Solution of the Eigenvalue Problem for a Dielectric Waveguide*, IEEE Transactions on Microwave Theory and Techniques, vol. 40, no. 4, Apr. 1992, pp. 699-705.

[45] S. D. Gedney, J. F. Lee, R. Mittra, *A Combined FEM/MoM Approach to Analyze the Plane Wave Diffraction by Arbitrary Gratings*, IEEE Trans. on Microwave Theory and Techniques, vol. 40, no. 2, pp. 363-370, Feb. 1992.

[46] S. D. Gedney, F. Lansing, *A parallel discrete surface integral equation method for the analysis of three-dimensional microwave circuit devices with planar symmetry*, IEEE Antennas and Propagation Society International Symposium Digest, vol. 3, pp. 1778-1781, 1994.

[47] S. D. Gedney, *A comparison of the Performance of Finite Difference Time-Domain, Finite Element Time-Domain, and Discrete Surface Integral Equation Methods on High Performance Parallel Computers*, IEEE Antennas and Propagation Society International Symposium Digest, vol. 1, pp. 384-387, 1994.

[48] S. D. Gedney, *Finite-Difference Time-Domain Analysis of Microwave Circuit Devices on High Performance Vector/Parallel Computers*, IEEE Transactions on Microwave Theory and Techniques, vol. 43, no. 10, Oct. 1995.

[49] S. D. Gedney, F. Lansing, R. T. Kihm, N. Owona, K. Virga, *Simulating "Large" Microwave Circuits with the Parallel Planar Generalized Yee Algorithm*, IEEE MTT-S Digest, vol. 2, pp. 1011-1014, 1996.

[50] G. H. Golub, C. F. van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, 1996.

[51] W. D. Gropp, D. E. Keyes, *Parallel Performance of Domain-Decomposed Preconditioned Krylov Methods for PDEs with Locally Uniform Refinement*, SIAM Journal of Scientific and Statistical Computing, vol. 13, no. 1, pp. 128-145, 1992.

[52] M. F. Hadi, M. Piket-May, *A Modified FDTD (2,4) Scheme for Modeling Electrically Large Structures with High-Phase Accuracy*, IEEE Transactions on Antennas and Propagation, vol. 45, pp. 254-264, Feb. 1997.

[53] T. E. Harrington, *Open Resonators for Anisotropic Superconductor and Dielectric Testing*, M. Sc. Thesis, Dept. of Electrical Engineering, University of Houston, Aug. 1996.

[54] T. E. Harrington, J. Wosik, S. A. Long, *Open resonator mode patterns for characterization of anisotropic dielectric substrates for HTS thin films*, IEEE Trans. Applied Superconductivity, vol. 7, pp. 1861-1864, Jun. 1997.

[55] G. Hebermehl, R. Schlundt, H. Zscheile, W. Heinrich, *Eigen mode solver for microwave transmission lines*, Weierstrass Institute for Applied Analysis and Stochastics, Preprint No. 308, Berlin, 1997.

[56] T. Jabłoński, *Iterative Eigenfunction Expansion Method for Cylindrical Fibers*, IFTR Reports, 3/1986, Warsaw, 1986. (in Polish)

[57] T. Jabłoński, M. Sowiński, *Analysis of dielectric guiding structures by the iterative eigenfunction expansion method*, IEEE Trans. Microwave Theory Tech. vol. 37, pp. 63-70, Jan. 1989.

[58] R. Janssen, M. Dracopoulos, K. Parrott, E. Slessor, P. Alotto, P. Molfino, M. Nervi, J. Simkin, *Parallelisation of Electromagnetic Simulation Codes*, IEEE Transactions on Magnetics, vol. 34, no. 5, pp. 3423-3426, 1998.

[59] D. Jones, *Methods in Electromagnetic Wave Propagation, vol. 1: Theory and Guided Waves*, Clarendon Press, Oxford, 1987.

[60] R. G. Jones, *The measurement of dielectric anisotropy using a microwave open resonator*, J. Phys. D: Appl. Phys., vol. 9, 1976, pp. 819-827.

[61] D. I. Kaklamani, A. Marsh, *Solution of electrically large planar scattering problems using parallel computed method of moments technique*, Journal of Electromagnetic Waves and Applications, vol. 9, no. 10, pp. 1313-1337, 1995.

[62] M. Kamon, N. A. Marques, L. M. Silveira, J. White, *Automatic Generation of Accurate Circuit Models of 3-D Interconnect*, IEEE Trans. on Components, Packaging, and Manufacturing Technology – Part B, vol. 21, no. 3, pp. 225-240. Aug. 1998.

[63] D. S. Katz, T. Cwik, B.H. Kwan, J. Z. Lou, P.L. Springer, T.L. Sterling, P. Wang, *An Assessment of a Beowulf System for a Wide Class of Analysis and Design Software*, Advances in Engineering Software, vol. 29, no. 3-6, pp. 451-461, April-July 1998.

[64] G. Kondylis, F. De Flaviis, G. Pottie, M. Sironen, T. Itoh, *Reduced FDTD formulation (R-FDTD) for the Analysis of 30 GHz Dielectric Resonator Coupled to a Microstrip Line*, MTT-S Symposium Digest, Anaheim, CA, 1999.

[65] F. Königer, *Measurement System for the Precise Determination of Dielectric Properties in the mm-Wave Range Based on Hemispherical Open Resonators*, Frequenz, 43, 7-8, pp. 209-214, 1989.

[66] J. E. Labaric, D. Kajfez, *Analysis of Dielectric Resonator Cavities Using the Finite Integration Technique*, IEEE Trans. Microwave Theory Tech. vol. 37, no. 11, pp. 1740-1747, Nov. 1989.

[67] R. B. Lehoucq, D. C. Sorensen, C. Yang, *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Method*, available from: `ftp.caam.rice.edu`.

[68] I. Lenhardt, T. Rottner, *Krylov subspace methods for structural finite element analysis*, Parallel Computing, vol. 25, pp. 861-875, 1999.

[69] K. J. Maschhoff, D. .C. Sorensen, *P_ARPACK: An Efficient Portable Large Scale Eigenvalue Package for Distributed Memory Parallel Architectures*, Rice University, 1996, available at: `ftp.caam.rice.edu`.

[70] Matlab Reference Manual, The Math Works Inc., Aug. 1992.

[71] S. Matoba, R. Yokoyama, T. Nakazawa, *A High Accuracy Eigenvalue Analysis for Large Power Systems*, POWERCON'98. International Conference on Power System Technology, in proceedings, vol. 2, pp. 1388-1392, 1998.

[72] J. Mielewski, M. Mrozowski, *Application of the Arnoldi Method in FEM Analysis of Waveguides*, IEEE Microwave and Guided Wave Letters, vol. 8, no. 1, Jan. 1998, pp. 7-9.

[73] J. Mielewski, A. Ćwikła, M. Mrozowski, *Analysis of Shielded Anisotropic Dielectric Resonators Using FDFD and the Arnoldi method*, MIKON'98, International Conference on Microwaves and Radar, in proceedings, vol. 2, pp. 335-339, 1998.

[74] J. Mielewski, A. Ćwikła, M. Mrozowski, *Accelerated FD analysis of dielectric resonators*, IEEE Microwave and Guided Wave Letters, vol. 8, no. 11, pp. 375-377, Nov. 1998.

[75] J. Mielewski, *Classical and hybrid techniques for the analysis of microwave waveguides and resonators*, PhD. Thesis, Technical University of Gdańsk, 1999.

[76] E. Minty, R. Davey, A. Simpson, D. Henty, *Decomposing the potentially parallel*, Course Notes, Edinburgh Parallel Computing Centre, The University of Edinburgh, 1996, available: `http://www.epcc.ed.ac.uk`.

[77] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, International Journal of Supercomputer Applications and High Performance Computing, 8(3/4), 1994.

[78] M. Mrozowski, *IEEM FFT – A Fast and Efficient Tool for Rigorous Computations of Propagation Constants and Field Distributions in Dielectric Guides with Arbitrary Cross-Section and Permittivity Profiles*, IEEE Trans. Microwave Theory Tech. vol. 39, pp. 323-329, Feb. 1991.

[79] M. Mrozowski, *Eigenfunction expansion techniques in the numerical analysis of inhomogeneously loaded waveguides and resonators*, Zeszyty Naukowe Politechniki Gdańskiej, Elektronika, No. 81, 1994.

[80] M. Mrozowski, *Guided Electromagnetic Waves, Properties and Analysis*, Research Studies Press, Taunton, Somerset, England, 1997.

[81] M. A. Nasir, W. C. Chew, *Fast Solution of Large Complex Non-Hermitian Sparse Eigenvalue Problems*, Antennas and Propagation Society International Symposium, 1994, AP-S Digest, vol. 3, pp. 2100-2103.

[82] U. D. Navsariwala, S. Gedney, *An unconditionally Stable Parallel Finite Element Time Domain Algorithm*, IEEE Antennas and Propagation Society International Digest, vol. 1, pp. 112-115, 1996.

[83] J. W. Nehrbass, J. O. Jevtić, R. Lee, *Reducing the Phase Error for Finite-Difference Methods Without Increasing the Order*, IEEE Transactions on Antennas and Propagation, vol. 46, pp. 1194-1201, Aug. 1998.

[84] V. Nikolski, *Variational Methods for Internal Problems of Electromagnetics*, Science, Moscow, 1967. (in Russian)

[85] K. Nyka, M. Mrozowski, *Combining function expansion and multigrid method for efficient analysis of MMICs*, Int. Microwave Symp. MIKON-96, pp. 203-207, Warsaw, 1996.

[86] D. W. Oldenburg, *Practical strategies for the solution of large-scale electromagnetic inverse problems*, Radio Science, vol. 29, no. 4, pp. 1081-1099, July-August 1994.

[87] E. N. Opp, S. L. Geyer, R. Thomas, M. S. Willett, *Numerical Computation of Electromagnetic Scattering on the Connection Machine using the Method of Moments*, IEEE Transactions on Magnetics, vol. 25, no. 4, pp. 2907-2909, Jul. 1989.

[88] V. Y. Pan, *Parallel Computation of a Krylov Matrix for a Sparse and Structured Input*, Mathl. Comput. Modelling, vol. 21, no. 11, pp. 97-99, 1995.

[89] B. N. Parlett, D. R. Taylor, Z. S. Liu, *A look-ahead Lanczos algorithm for nonsymmetric matrices*, Mathematics of Computation, vol. 44, pp. 105-124, 1985.

[90] J. E. Patterson, T. Cwik, R. D. Ferraro, N. Jacobi, P. C. Liewer, T. G. Lockhart, G. A. Lyzenga, J. W. Parker, D. A. Simoni, *Parallel Computation Applied to Electromagnetic Scattering and Radiation Analysis*, Electromagnetics, vol. 10, pp. 21-39, 1990.

[91] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling, *Numerical recipes: The art of scientific computing*, Cambridge University Press, Cambridge, 1986.

[92] P. Przybyszewski, J. Mielewski, M. Mrozowski, *Efficient Eigenfunction Expansion Algorithms for Analysis of Waveguides*, Technical Report No. 88/96, Dept. of Electronics, Telecommunications and Computer Science, Technical University of Gdańsk, Gdańsk, 1996.

[93] R. F. Remis, P. M. van den Berg, *A Modified Lanczos Algorithm for the Computation of Transient Electromagnetic Wavefields*, IEEE Trans. Microwave Theory Tech. vol. 45, no. 12, pp. 2139-2149, Dec. 1997.

[94] M. Rewieński, *Implementation of the Parallel Arnoldi Method in the IBM SP2 Distributed Memory System*, TASK Quarterly, No. 1, vol. 1, pp. 109-126, Jul. 1997.

[95] M. Rewieński, *Methods of Solving Operator Eigenproblems in Parallel Distributed Memory Systems as Applied in Electromagnetics*, TASK Quarterly, No. 4/98, vol. 2, pp. 611-732, Oct. 1998.

[96] M. Rewieński, M. Mrozowski, *Finding Waveguide Modes via Implicit Operator Projection in the Krylov Subspace*, IEEE Microwave and Guided Wave Letters, vol. 9, no. 4, pp. 140-142, Apr. 1999.

[97] M. Rewieński, M. Mrozowski, *Scalable Algorithm for Solving Boundary Value Problems Arising in Electromagnetics Based on Implicit Operator Projection*, in Proceedings of the 15th Annual Review of Progress in Applied Computational Electromagnetics, March 15-20, Monterey, CA, pp. 563-570.

[98] T. Rozzi, M. Mongiardo, *Open electromagnetic waveguides*, The Institution of Electrical Engineers, London, UK, 1997.

[99] Y. Saad, *Krylov Subspace Methods on Supercomputers*, SIAM Journal of Scientific and Statistical Computing, vol. 10, no. 6, pp. 1200-1232, Nov. 1989.

[100] Y. Saad, *Numerical methods for large eigenvalue problems*, Manchester University Press, Manchester, UK, 1992.

[101] Y. Saad, *SPARSKIT: a basic tool kit for sparse matrix computations. Version 2*, CSRD – University of Illinois and RIACS (NASA Army Research Center), 1994, available: `ftp.cs.umn.edu/dept/sparse`.

[102] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halstead Press, NY, 1992.

[103] J. N. Shadid, R. S. Tuminaro, *A Comparison of Preconditioned Nonsymmetric Krylov Methods on a Large-Scale MIMD machine*, SIAM Journal of Scientific Computing, vol. 15, no. 2, pp. 440-459, Mar. 1994.

[104] V. P. Shestopalov, Y. V. Shestopalov, *Spectral theory and excitation of open structures*, IEE, London, UK, 1996.

[105] L. M. Silveira, M. Kamon, I. Elfadel, J. White, *A Coordinate-Transformed Arnoldi Algorithm for Generating Guaranteed Stable Reduced-Order Models of RLC Circuits*, ICCAD-96, 1996 IEEE/ACM International Conference on Computer-Aided Design, Digest of Technical Papers, pp. 288-294, 1996.

[106] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, Technical Report TR90-27, Rice University, Dept. of Math. Sci., Houston, TX, 1990.

[107] D. C. Sorensen, *Implicitly Restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations*, Proceedings of an ICASE/LaRC Workshop, May 23-25 1994, Hampton, VA, D. E. Keyes, A. Sameh and V. Venkatakrishnan, eds., Kluwer, 1995.

[108] E. de Sturler, *A performance model for Krylov subspace methods on mesh-based parallel computers*, Parallel Computing, vol. 22, pp. 57-74, 1996.

[109] P. N. Swarztrauber, *FFTPACK, version 4, A package of fortran subprograms for the Fast Fourier Transform of periodic and other symmetric sequences.*, 1985, available from: `http://www.netlib.org`.

[110] A. Taflove, *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, Norwood, MA, Artech House, 1995.

[111] K. D. Tatalias, J. M. Bornholdt, *Mapping Electromagnetic Field Computations to Parallel Processors*, IEEE Transactions on Magnetics, vol. 25, no. 4, pp. 2901-2906, Jul. 1989.

[112] *A User's Guide to the Basic Linear Algebra Communication Subprograms (BLACS) v. 1.1*, available: `ftp.netlib.org`.

[113] C. Vollaire, L. Nicolas, A. Nicolas, *Finite Elements and Absorbing Boundary Conditions for Scattering Problems on a Parallel Distributed Memory Computer*, IEEE Transactions on Magnetics, vol. 33, no. 2, pp. 1448-1451, Mar. 1997.

[114] C. Vollaire, L. Nicolas, *Implementation of a Finite Element and Absorbing Boundary Conditions Package on a Parallel Shared Memory Computer*, IEEE Transactions on Magnetics, vol. 34, no. 5, pp. 3343-3346, Sep. 1998.

[115] C. Vollaire, L. Nicolas, *Preconditioning Techniques for the Conjugate Gradient Solver on a Parallel Distributed Memory Computer*, IEEE Transactions on Magnetics, vol. 34, no. 5, pp. 3347-3350, Sep. 1998.

[116] C. Vollaire, L. Nicolas, A. Nicolas, *Parallel Computing for Electromagnetic Field Computation*, IEEE Transactions on Magnetics, vol. 34, no. 5, pp. 3419-3422, Sep. 1998.

[117] H. A. van der Vorst, G. H. Golub, *150 Years and still alive: eigenproblems*, Technical Report SCCM96-11, Dept. of Scientific Computing and Computational Mathematics, Stanford University, USA, 1996.

[118] T. Weiland, *Eine numerische Methode zur Lösung des Eigenwellenproblems längshomogener Wellenleiter*, AEÜ, Band 31, Heft 7/8, pp. 308-314, 1977.

[119] J. H. Wilkinson and C. Reinsch, editors, *Handbook for Automatic Computation, Vol. 2, Linear Algebra*, Springer Verlag, Heidelberg – Berlin – New York, 1971.

[120] S. Winograd, *On computing the discrete Fourier transform*, Math. Comp. , 32, pp. 175-199, 1978.

Copyright note