TECHNISCHE UNIVERSITÄT WIEN

DIPLOMARBEIT

Numerical Modeling of Diffusion and FDTD Wave Propagation in Semiconductor Devices

ausgeführt am

Institut für Mikroelektronik
Technische Universität Wien

unter der Anleitung von

Associate Prof. Dr.techn. Lado Filipovic

durch

Felix Strasser, BSc

Wien, im August 2025

Unterschrift Betreuer

Unterschrift Verfasser

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Hilfsmittel der künstlichen Intelligenz wurden allenfalls nur verwendet, um sprachliche Formulierungen und Zeichensetzung Korrektur zu lesen und gegebenenfalls zu verbessern.

Wien, am _____ _____

Felix Strasser

Abstract

In microelectronics, the simulation of physical processes is essential for predicting device structure and behavior during the design stage and for enabling the precise control of automated manufacturing steps. This field is broadly divided into two main categories: process simulation, which models the fabrication of semiconductor devices, and device simulation, which models their subsequent functionality. Together, they are known under the term technology computer-aided design (TCAD).

Building on the functionality provided by the ViennaPS process simulation framework, this work investigates physical simulation methods with a focus on solving partial differential equations on regular structured grids through the use of finite difference methods on the available volumetric cell representations of the material structures (cell set). The theory behind the utilized methods is first described, followed by the demonstration of two application examples, combining both fabrication process simulations and physical device simulations. In particular, a simple diffusion process serves as a first example for applying basic finite-difference schemes within the process simulation framework, and secondly, the finite-difference time-domain (FDTD) method for simulating electromagnetic wave propagation is explored and tested on a model of a photodiode. For the purpose of FDTD, a new and independent C++ library was developed, which allows integration within ViennaPS to perform testing of device behavior on the output structures resulting from completed process simulations.

Ultimately, the combination of process simulation with an electrodynamic device simulation method within a unified framework paves the way for the development of seamless iterative optimization workflows.

Kurzfassung

In der Mikroelektronik ist die Simulation physikalischer Prozesse essenziell für die Vorhersage der Struktur und des Verhaltens elektronischer Bauelemente während der Entwurfsphase, und um die präzise Steuerung automatisierter Herstellungsschritte zu ermöglichen. Das Gebiet umfasst im Wesentlichen zwei Hauptkategorien: Prozesssimulation, welche die Fertigung halbleiterbasierter Bauelemente modelliert, und Bauelementsimulation, welche deren Funktionalität modelliert. Diese werden zusammengefasst unter dem Begriff Technology Computer-Aided Design (TCAD).

Aufbauend auf der Funktionalität des Prozesssimulators ViennaPS untersucht diese Arbeit physikalisch basierte Simulationsmethoden mit Fokus auf dem Lösen partieller Differentialgleichungen auf regelmäßig strukturierten Gittern durch Finite-Differenzen-Methoden mit der vorhandenen volumetrischen Darstellung der Materialstrukturen (Cell-Set). Zunächst wird die zugrunde liegende Theorie der verwendeten Methoden erläutert, gefolgt von der Demonstration zweier Anwendungsbeispiele, die sowohl die Simulation von Fertigungsprozessen als auch der Funktionalität von Bauelementen umfassen. Im Speziellen dient ein einfacher Diffusionsprozess als erstes Beispiel für die Anwendung einfacher Finite-Differenzen-Verfahren innerhalb des Prozesssimulators, und als Zweites wird die Finite-Difference Time-Domain (FDTD) Methode für die Simulation elektromagnetischer Wellenausbreitung untersucht und am Modell einer Photodiode getestet. Für FDTD wurde eine neue und eigenständige Programmbibliothek in C++ entwickelt, die sich in ViennaPS integrieren lässt, um das Verhalten von Bauelementen auf Basis der aus abgeschlossenen Prozesssimulationen hervorgegangenen Strukturen zu untersuchen.

Die Kombination von Prozesssimulation mit einer elektrodynamischen Simulationsmethode für Bauelemente in einem gemeinsamen Rahmen legt letztendlich den Grundstein für die Entwicklung nahtloser iterativer Optimierungsabläufe.

Contents

1 Introduction

Microelectronics investigates the development and functioning of electronic components at very small scales, which revolves mainly around semiconductor materials and their use in complex integrated circuits (ICs). These technologies form the basis of most of today's ubiquitous electronic devices, notably the hardware components of general-purpose computers, including the central processing units (CPUs), random-access memory (RAM) and graphics processing units (GPUs). The small scale and complexity of these structures make computer models essential for designing, simulating and automating the manufacturing processes. The Institute for Microelectronics at the TU Wien works on building software tools that aid in these tasks. The project that is the subject of this thesis makes use of the **ViennaPS**[1] process simulation framework, exploring some aspects of the development and design of microelectronics hardware through computer simulations.

Technology computer-aided design (TCAD) is the overarching term in this field. It encompasses two primary disciplines: process simulation (process TCAD), which models semiconductor fabrication, and device simulation (device TCAD), which models functionality. Device simulation commonly uses the resulting structures from process simulation as its starting point, thus functioning as a postprocessing step. The target devices are typically electronic components in the form of active (e.g., transistors) and passive (e.g., metallization) devices, or other modern small devices such as micro-electromechanical systems (MEMS).

The great advantage of accurate computer simulations for semiconductor-based technologies is the ability to avoid or greatly reduce the high costs in resources and time associated with physical experiments.

Among other aspects, the numerical solution of partial differential equations (PDEs), which serve as the mathematical models for much of the underlying physics, is a central point of the simulation stages.

1.1 Process simulation

The goal of process simulation is to create and apply computational models for the processing steps required to fabricate microprocessors and related semiconductor technologies. The models should capture the governing physics well enough to allow for accurate predictions of the resulting material structures and their functionality, obtained from the real-word processes. These physics-heavy models may be combined with empirical models.

Important steps in process simulations include lithography (adding masks), etching, deposition, ion implantation, diffusion, surface removal and oxidation. Some of these processes come with multiple variations, for instance, wet etching and dry etching, or chemical vapor

[1] Website: `https://github.com/ViennaTools/ViennaPS`, `https://www.iue.tuwien.ac.at/viennacl0`

deposition and physical vapor deposition. Topography simulation, including deposition and etching, refers to the processing steps that change the surface shape of the wafer.

Some techniques available for a computer implementation of these steps are the level set method for the efficient definition and evolution of surface geometries [1, 2], voxel-based methods for interactions with volumetric cells (dense or sparse), ray tracing, numerical methods for solving PDEs such as finite difference methods, and Monte Carlo methods.

A basic implementation of a diffusion simulation with a simple example application, using finite differences on a dense structured grid, makes up the first part of this thesis.

1.2 Device simulation

Semiconductor device simulation deals with the operational modeling of devices and their components at different levels of abstraction and the numerical or algebraic computations necessary for a complete description.

Generally, the aim is to accurately reproduce the real physical effects and interactions that take place in and define the characteristics of the electronic devices under consideration, extending the physical simulations of process TCAD to the device operation stage; see Figure 1. Sometimes, however, even simplified physical models are too complex or costly for compound device structures, motivating the use of models at higher levels of abstraction for tasks such as circuit design and simulation (logical simulation).

The finite-difference time-domain (FDTD) method is an example of a physics-based analysis method for predicting the performance of feature-scale structures on the process simulation output, in particular by simulating electromagnetic wave propagation according to Maxwell's equations, supporting various frequency ranges, for example signals in the optical range.

The presentation of FDTD and its implementation forms the second and main part of this thesis.

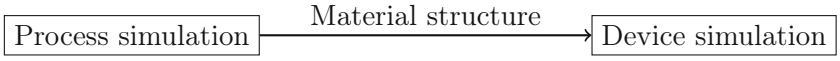


Figure 1: Sequence of TCAD stages.

1.3 Process simulation in ViennaPS

The process simulation framework **ViennaPS** is a suite of programming libraries that combines several tools used for the purpose of process TCAD mentioned above, including a realization of the level set method in **ViennaLS**, a ray tracing implementation in **ViennaRay** (for Monte Carlo ray tracing) and voxel-based functionality provided by the *cell set*. The individual libraries can be used independently for custom development, based on the more elementary functionalities, or in combination in the ViennaPS library, which already has some pre-assembled process models. Figure 2 gives a more complete overview of the current structure of the main components of ViennaPS.

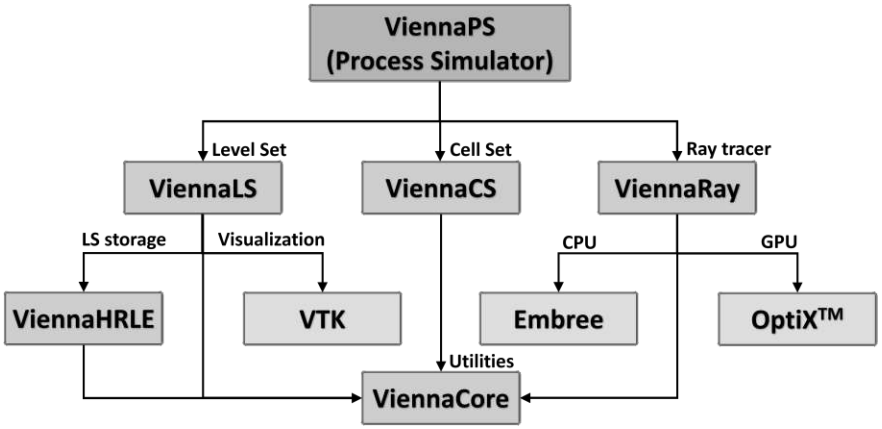


Figure 2: The main components of the ViennaPS framework.

The level set method relies on an implicit definition of a surface (or hypersurface) by storing discrete values on a Cartesian grid of an implicit function (the signed distance function) that defines the surface location in the domain. The hierarchical run-length-encoded (HRLE) data structure provided in **ViennaHRLE** makes possible a sparse representation, storing only the necessary grid points around the surface. This implements the narrow-band optimization, specifically the sparse-field version [1]. ViennaLS, and thereby also ViennaPS, makes use of this HRLE structure as its foundation.

The cell set builds on the same underlying level set grid, extending the sparse surface representations by generating a dense set of cells in a rectangular region around the surface geometries with a given depth. Optionally, the cells above the solid surface that are within the depth can be added to include part of the surrounding space in the simulation. The cell set also builds on the HRLE data structure for the structure generation and modification, utilizing its dense cell iterators.

Ray tracing allows tracing particle paths for ion implantation and other physical processes. The ray tracing functionality is used internally in some process models, including the SF_6/O_2 etching model, but it will not be discussed explicitly in this project.

The processing steps implemented in ViennaPS are etching, deposition, oxidation and others, including their variations and combinations. This is made possible by directly integrating the surface representation of the level set functionality with the volume cell representation of the domains, allowing more detailed three-dimensional interaction with the solid material below the surface and above it, including part of the surrounding gas or liquid. These representations are then useful for ray tracing.

1.4 Contributions

Originally, the code for the cell set was part of the ViennaPS library and, although it was mostly isolated as an independent part, some coupling between the two was present. The first task of this thesis project was to remove this coupling by making some minor modifications and then to extract the cell set into its own library, **ViennaCS**, which is now added as a dependency of ViennaPS and the cell set library itself depends on the level-set

and ray-tracing libraries.

Utilizing the new library, an example model was created to simulate a diffusion process using two different finite-difference discretization approaches to investigate numerical stability. This served as an introduction to finite difference methods and to demonstrate a possible use case for having the cell set with its volumetric capabilities available as an independent toolset. The geometric structure of the domain is first constructed from a surface representation, combining operations from the level set library, on top of which the voxel structure is then generated.

The final and most extensive part of this thesis consists in the development of a new implementation of the numerical method known as the finite-difference time-domain (FDTD) method, together with an example application. The implementation is managed as a separate and standalone C++ library, but it is designed to allow a straightforward integration with the dense data array format of the cell set. For the application, a model of a photodiode (as part of an image sensor pixel unit) is created in a process simulated within ViennaPS and it includes the construction of a simple layered geometry through the level set functionality and an intermediate chemical etching step that modifies a material layer by making small holes in the substrate. On the final geometry, a subsequent postprocessing step with FDTD simulates the propagation of light waves through the photodiode to investigate the effect of the hole pattern on the wave propagation. This part of the project thus continues the investigation into finite difference methods on regular structured grids and their generalizations.

1.4.1 Iterative device optimization workflows

The direct integration of fabrication process simulations with complementary device analysis capabilities in a combined system provides the benefit of a coherent unified setup and immediate feedback, thus providing a means to apply optimization cycles that iteratively run the simulations and adjust their parameters to improve upon a desired outcome, in a seamless and possibly automated fashion.

At a high level, this provides support for the following iterative device optimization workflow:

1. Perform process simulation to generate a physically meaningful geometric structure, providing a certain set of process parameters as input.

2. Execute the device simulation on the structure obtained in the previous step to predict the (physical) device behavior under certain conditions.

3. Assess the performance of the device with metrics calculated from the analysis, and, until a desired outcome is achieved, repeat the entire procedure, starting with the first step with appropriately modified input parameters according to an optimization scheme. The procedure should thereby provide the optimal process input parameters for a desired or targeted device performance.

The steps are visualized in Figure 3.

To properly realize a useful version of this workflow, future work could improve the ease of integration and develop a more complete example with appropriate analysis metrics and optimization targets.

The example of a photodiode (image sensor unit) presented in Chapter 5 would be a suitable model to optimize optical device performance in such a procedure, maximizing its transmittivity (or minimizing reflectivity). However, more work is necessary, as is further explained in that chapter.

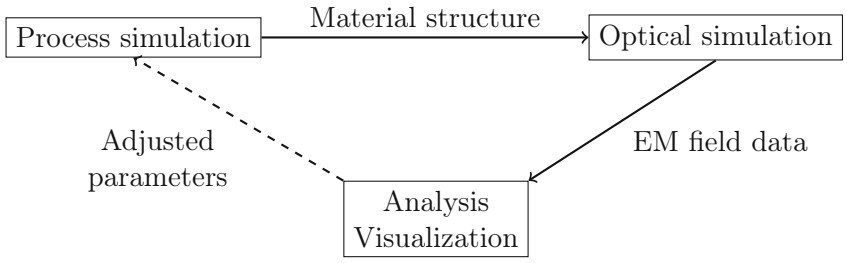


Figure 3: Stages in the combined TCAD workflow. Completing the cycle by systematically adjusting the process input parameters based on the device analysis is a subject for future work.

1.5 Technical notes

The **C++** programming language was the main tool used to translate the mathematical and physical theory that forms the basis of the treated technologies into executable computer programs.

The C++ libraries in the ViennaPS framework are open source and available under a permissive license.

Additional software used for the project were **ParaView**[2] and **Python 3** for visualization of the results and LaTeX for typesetting this text.

A Linux operating system was used as the main development environment.

Regarding the computer hardware, the kinds of simulations considered here are small-scale and can be run on any typical recent general-purpose laptop or desktop computer, without requiring specialized hardware components (i.e. CPU only).

The FDTD library is currently available at `https://github.com/exilief/FiDiTi`, see [3] for the version used here.

The diffusion examples are currently available in the ViennaCS code repository under the paths

- `ViennaCS/examples/diffusion/`,
- `ViennaCS/examples/diffusionImplicit/`,

see [4] for the library version used here. For the implicit diffusion example, the utilized Eigen[3] library is not a necessary dependency of ViennaCS and so it must be ensured that it is available on the system to build and run this example.

[2] Website: `https://www.paraview.org/`
[3] Website: `https://eigen.tuxfamily.org/`

The photodiode example, utilizing the FDTD library, is currently available in the ViennaPS code repository under the path

- `ViennaPS/examples/photodiode/`,

see [5] for the library version used here. It must be ensured that the FDTD library is available on the system.

2 Grid-based Numerical Simulations

Numerical methods for solving physical field problems on structured orthogonal grids with finite differences are a relatively old and simple approach. While often overshadowed by more complex methods, they remain in widespread use and have the advantage that they are quite straightforward to understand and implement. More advanced methods, such as the *finite element method* (FEM) and its variations, are available that allow performing simulations also on unstructured meshes and general curved geometries, but they are also more technically demanding and their advanced capabilities are not always necessary, or may not justify the trade-off in complexity or memory usage.

The regular grids that are already used in the level set method internally and in the cell set, which builds on and extends the level set grid, make finite difference methods a natural choice for our purposes.

In the following, simple low-order finite difference methods for solving PDEs on regular rectangular grids are introduced.

2.1 Finite differences

To solve differential equations numerically, the continuous-domain problem can be discretized using a regular grid of points and differential quotients are approximated with difference quotients of the relevant physical quantities at these grid points. This is the procedure followed by finite difference methods, at least in the simplest cases.

For example, an ordinary differential equation contains time-derivatives of a quantity $u(t)$ with the time variable t, which can be approximated as

$$\left.\frac{du}{dt}\right|_t \approx \frac{u(t+\delta) - u(t-\delta)}{2\delta}.$$

Throughout, we shall assume a regular discretization of each time and space dimension of the domain into a grid of equidistant points according to $t_n = n\,\Delta t = nk$ and $x_i = i\,\Delta x = ih$ with integers $n, i = 0, 1, ...$ and the time step Δt or spatial step Δx. Together, this forms an orthogonal (rectangular) grid in space with an additional dimension for time with regular intervals. The dependent quantity $u(t)$ to be solved for is then assigned discrete values u^n approximating $u(t_n)$ or u_i^n approximating $u(t_n, x_i)$. Another notation to express the assignment of values to discrete points is

$$u^n[i] = u_i^n$$

and it mimics the array indexing syntax of many programming languages. This will become useful later, when there is both a time- and space-dependence with multiple spatial dimensions.

Now, we can replace differential quotients with difference quotients, restricting the relevant quantities to the chosen discrete points, such as

$$\left.\frac{du}{dt}\right|_{t_n} \approx \frac{u^{n+1} - u^n}{\Delta t}.$$

Unlike the continuous case, there is no longer a fixed point at which the (approximated) derivative is unambiguously defined. Instead, it is approximated by a difference between values at points that are a finite distance apart, which requires a careful interpretation. Essentially, what matters is the relative location of the grid points between the different expressions in the equation that contain a dependence on t, so that for an ODE of the form

$$\frac{du}{dt} = f(u(t), t)$$

two possible choices are [6]

$$\frac{u^{n+1} - u^n}{\Delta t} = f(u^n, t_n) \tag{2.1}$$

and

$$\frac{u^{n+1} - u^n}{\Delta t} = f(u^{n+1}, t_{n+1}), \tag{2.2}$$

which are relations to be solved for the new value u^{n+1} in dependence of the previous value u^n. They are known as the *forward Euler* and *backward Euler* schemes, respectively. In the first case, an explicit formula can be directly obtained, while the second case where f is evaluated at the next step is an implicit relation that is, in general, more difficult to handle and requires solving a linear system of equations if f is linear in u and u is vector-valued.

For partial differential equations (PDEs), the most common finite-difference solution approach is to first apply a finite-difference discretization in space, keeping the time (and time-derivatives) continuous and treating it as a fixed parameter. This yields a semi-discretized system and ensures that time-derivatives are evaluated at the desired point x_i in the spatial grid. The semi-discretized system represents a system of ordinary differential equations with one degree of freedom for each grid point x_i and finite differences can again be applied as explained above with time as the variable. Both explicit and implicit methods in time are also possible here. This is known as the *method of lines*.

The Euler formulas given above are not symmetric with respect to the application point. As another choice for the discretization, the midpoint rule uses centered finite differences with a symmetric difference quotient around the evaluation point. It is given as

$$\frac{u^{n+1} - u^{n-1}}{2\,\Delta t} = f(u^n, t_n).$$

This is a two-step formula since it contains values of u separated by two time steps. The related *leap-frog method* for the first-order wave equation (advection equation) $\frac{\partial u}{\partial t} = -\frac{\partial u}{\partial x}$, a hyperbolic PDE, is given as [6]

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\,\Delta t} = -\frac{u_{i+1}^n - u_{i-1}^n}{2\,\Delta x}$$

and it will be seen later that the finite-difference time-domain method uses a scheme very similar to the leap-frog method for its time and space discretizations of the relevant differential operators.

Although the leap-frog method works well for some wave-propagation problems, it is not suitable for parabolic problems like diffusion, as it does not guarantee stability and convergence to the exact solution as the grid is refined. Therefore, it is necessary to be familiar with multiple types of discretization methods and to be aware of their strengths and weaknesses in order to apply an appropriate method for the problem at hand.

Convergence of a numerical method refers to convergence of the approximation to the exact solution in the limit $\Delta t \to 0$ or $\Delta x \to 0$, while the order of convergence is the behavior of a suitable measure for the magnitude of the error in the limit as a power of Δt or Δx.

Consistency of a finite-difference approximation means that the discretized equation converges to the continuous PDE as the grid is refined (at least of order 1). This can be checked by performing a Taylor series expansion of the solution around a discrete point to relate the values of neighboring points to each other, resulting in an equation containing the PDE and additional terms with varying orders of Δx and Δt, see [7], Section 3.11, for the exact procedure.

Stability is an important feature that decides the viability of a finite difference method, as will be seen in the following.

An important theoretical result is the *Lax equivalence theorem*, which states that for a linear PDE, a consistent numerical scheme is convergent if and only if it is stable [6]. For our purposes we can thus identify stability and convergence as synonymous.

This highlights the importance of a stability analysis, which must, in general, be performed separately for each type of equation, although some general statements can also be made.

3 Application: Diffusion

Diffusion processes that occur during the fabrication of semiconductor devices are an important aspect that must be understood in order to reliably control the desired properties of the resulting structure. Frequently, diffusion is an undesired side-effect of processing steps that require operating with high temperatures, but it can also be used to introduce certain modifications into a substrate (i.e. doping) through its surface in a controlled manner [8].

A simple finite-difference approximation of a diffusion problem representative of that occurring in process simulation steps is given in this chapter and it will serve as a demonstration of the usefulness of a separate voxel-based data structure, to be used in conjunction with surface-based level-set representations. The geometric structure is generated using the available level set functionality to create a simple flat solid substrate with a partially obstructed surface by an inert mask material. This can be done by creating geometric primitives including planar layers and boxes in the surface format (using signed distances internally) and combining them using Boolean operations such as volumetric union, intersection and differences (set operations). The entire domain is laterally bounded but can be freely extended in the vertical direction. Finally, layers of cells are generated around the surface structures with a chosen depth, filling the layers above the surface with air. The geometry of the setup is shown in Figure 4.

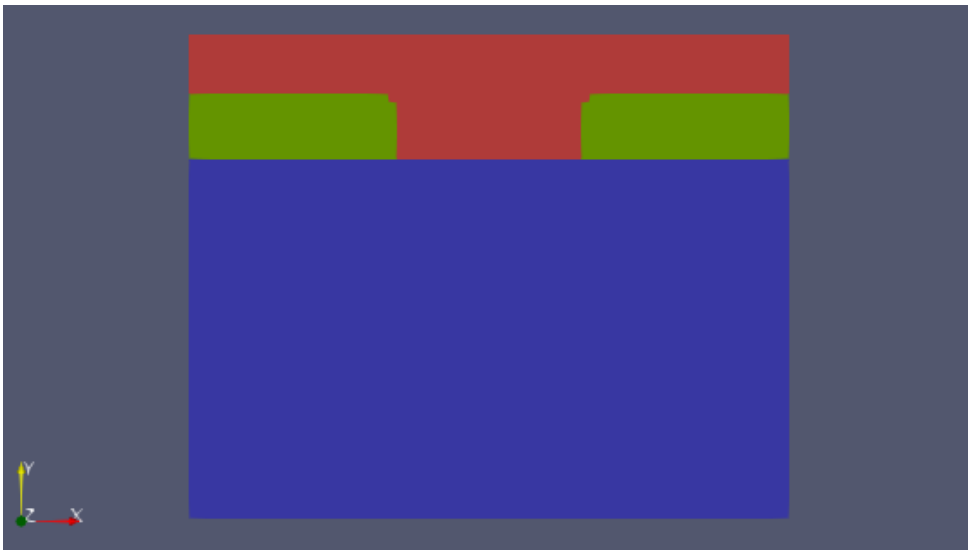


Figure 4: The geometry for the dopant in-diffusion example. The semiconductor substrate at the bottom (blue) is covered with a mask layer (green) with a hole in the middle, where it is exposed to the surrounding fluid or gas (red) with a certain material concentration.

3.1 Mathematical problem statement

A simple model for the diffusion of a single species with concentration c is the PDE known as Fick's diffusion law

$$\frac{\partial c}{\partial t} = \nabla \cdot (D \nabla c)$$

with the diffusivity D (diffusion coefficient) as a material parameter. The component $J = -D \nabla c$ can be identified as the diffusion flux of particles, representing the material flow when a concentration gradient is present. For a more realistic process simulation of dopant diffusion, additional physical considerations would have to be made due to anomalous diffusion behaviors in semiconductors [9], but the simple model shall suffice here.

The diffusion problem is exemplary for a dissipative system, which is marked by irreversible evolution. Heat conduction is another physical process driven by diffusive behavior, with the heat equation being analogous to the diffusion equation if the material parameter is a scalar constant. In this analogy, temperature T is the intensive property that drives diffusion, and heat energy is the extensive quantity that is transported.

Mathematically, diffusion is described by a partial differential equation of the parabolic type, including a time dependence, where both boundary conditions and initial conditions must be provided for a complete problem statement.

Two different types of boundary conditions are used for the example of dopant diffusion. The in-flow of material through the exposed surface at the top from the external region is modeled by setting a constant concentration outside, in the form of an inhomogeneous Dirichlet boundary condition, which can be interpreted as an inexhaustible source of new particles for the duration of the simulation. On the remaining parts of the substrate boundary, which is comprised of the interface to the mask and the outer domain boundary, a zero-flow condition is imposed by setting the normal derivative of the concentration to zero (Neumann boundary conditions). The initial values of the concentration inside the substrate volume is set to zero.

3.2 Discretized formulation

A suitable finite-difference discretization in one-dimensional space is given by

$$\frac{c_i^{n+1} - c_i^n}{\Delta t} = D \frac{c_{i+1}^n + c_{i-1}^n - 2c_i^n}{\Delta x^2},$$

where the material parameter D is assumed to be a scalar constant, to further simplify the model (and $\Delta x^2 = (\Delta x)^2$). The semi-discretization in space and the following time-discretization are already applied in this expression. A simple forward-Euler explicit time stepping is chosen. The combined finite-difference formula for the space and time derivatives is known as the forward-in-time and centered-in-space (FTCS) scheme, specifically for the equation that is first order in time and second order in space. It is generally preferred to analyze the finite-difference discretization of a PDE as a whole, taking into account that stability is defined for the combined scheme. In two-dimensional (2D) and three-dimensional (3D) space, analogous expressions are added for the terms from the differential operator

(Laplacian) $\Delta c = \nabla \cdot \nabla c = \sum_{i=1}^{d} \frac{\partial^2 c}{\partial x_i^2}$ (d dimensions), resulting in the five-point stencil in 2D

$$\frac{c_{i,j}^{n+1} - c_{i,j}^{n}}{\Delta t} = D \frac{c_{i+1,j}^{n} + c_{i-1,j}^{n} + c_{i,j+1}^{n} + c_{i,j-1}^{n} - 4c_{i,j}^{n}}{\Delta x^2},$$

illustrated in Figure 5, and the seven-point stencil in 3D

$$\frac{c_{i,j,k}^{n+1} - c_{i,j,k}^{n}}{\Delta t} = D \frac{c_{i+1,j,k}^{n} + c_{i-1,j,k}^{n} + c_{i,j+1,k}^{n} + c_{i,j-1,k}^{n} + c_{i,j,k+1}^{n} + c_{i,j,k-1}^{n} - 6c_{i,j,k}^{n}}{\Delta x^2}.$$

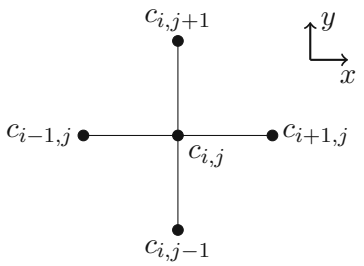


Figure 5: Five-point stencil in 2D, for discretizing $\Delta c = \frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}$.

Dirichlet boundary conditions can be added by prescribing the constant concentration as initial values in the layer of cells in the source region that is in direct contact with the substrate boundary and then never updating the values in this outer region, keeping them constant. The adjacent internal cells then use these values of neighboring cells in the regular update step. Alternatively, the numerical domain could be restricted to the internal points and the fixed boundary value contributions added to the equation as a source term. This approach is equivalent in the present case, but it would be more flexible, allowing for a generalized treatment of boundary and source conditions in the future. This idea is not pursued further.

To impose the Neumann boundary conditions, consider the spatial difference quotient as derived from the combination of forward and backward differences around the point x_i,

$$\frac{(c_{i+1}^{n} - c_{i}^{n}) - (c_{i}^{n} - c_{i-1}^{n})}{\Delta x^2},$$

and set this one-sided difference at the boundary to zero. This is done by individually adding the one-sided first-order differences for all spatial dimensions in both directions and simply leaving out the contributions from neighboring points where the zero flow condition should be applied.

A general concern with boundary conditions that themselves involve a finite-difference approximation of a derivative is that the order of convergence should match that of the main finite-difference scheme to avoid it reducing the overall order of accuracy to that of the boundary discretization. Thus, second-order one-sided differences [10] or centered differences might be more adequate. For example, centered differences would be $(c_2^n - c_0^n)/(2\,\Delta x) = 0$, see [7], Appendix B. The value at the boundary c_0^n would thus enter the update equation of the internal point c_1^n, modifying the coefficient for c_2^n. This is ignored here and simple differences with first-order accuracy are used.

Stability demands that

$$\Delta t \leq \frac{\Delta x^2}{2d \cdot D}$$

for the time step Δt and the grid interval Δx, with d being the number of spatial dimensions, see [7], Sections 3.7 and 4.4, for a derivation (or [10], Section 5.7). This can be proven with a von Neumann stability analysis and convergence to the real solution follows from the Lax equivalence principle. Because a globally constant time step Δt is used, the diffusivity D must, in general, be replaced by the maximum value over all materials in the domain to ensure stability everywhere.

It is interesting to note that in the limit $\Delta x \to 0$, the time step Δt approaches zero faster than the spatial step Δx by one order of convergence and since a signal can travel through the grid by a distance of Δx per time step (nearest neighbors), the maximum speed approaches infinity. This, in fact, matches the continuous behavior, where the initial conditions at one point can affect all other points in the domain after any finite time interval. The requirement that the numerical domain of dependence asymptotically matches (or contains) the analytical one is the well-known Courant-Friedrichs-Lewy condition (CFL condition) for parabolic PDEs [11], and it is a necessary condition for stability (for linear problems) and convergence, but not always a sufficient one, as is the case here (it only requires Δt to approach zero faster than Δx). The CFL condition gains a more direct and intuitive interpretation for hyperbolic PDEs such as wave equations that have a finite speed of information travel and it will be explained more thoroughly for electromagnetic wave propagation when using the finite-difference time-domain method in Chapter 4.

We can identify

$$S = 2d \cdot D \frac{\Delta t}{\Delta x^2}$$

as a stability factor so that $S = 1$ defines the stable limit, and this factor is added to the simulation parameters along with the spatial step Δx. This way, the stability parameter can be fixed and the time step Δt is calculated automatically according to the expression above, which makes it easier to investigate the stability behavior.

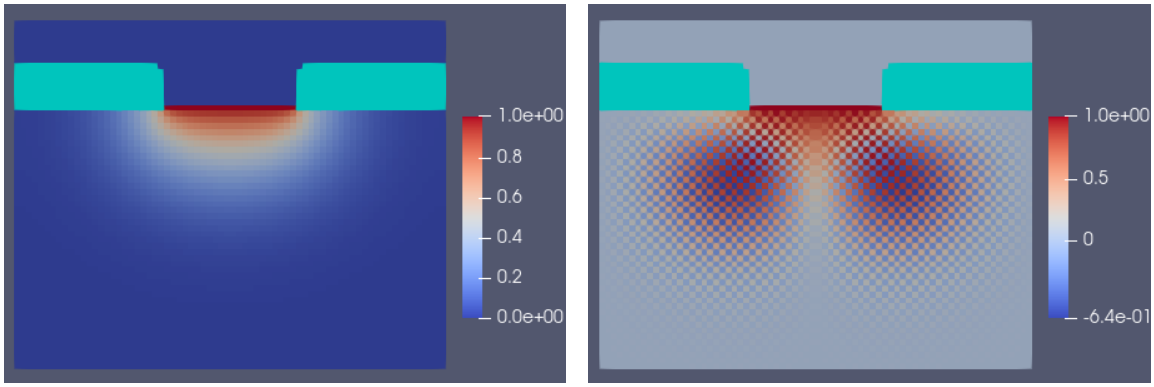
The stability requirement for the time step in the explicit method, which can become quite restrictive in certain cases, can be avoided by using a numerically more demanding but unconditionally stable implicit time-stepping scheme, which makes the choice of much larger time steps permissible. Implicit finite-difference schemes also have an unbounded numerical domain of dependence — new values at each point can depend on the previous values at any other point [6]. A relatively straightforward implementation is possible by using the **Eigen**[1] numerical library for solving the linear system of equations that arises.

3.3 Simulation results

The simulation results for both the explicit and implicit implementation for different values of the stability factor, choosing two space dimensions, are displayed in Figure 6 and Figure 7. When the stability factor S is chosen to be 1.024, only slightly above the maximum stable

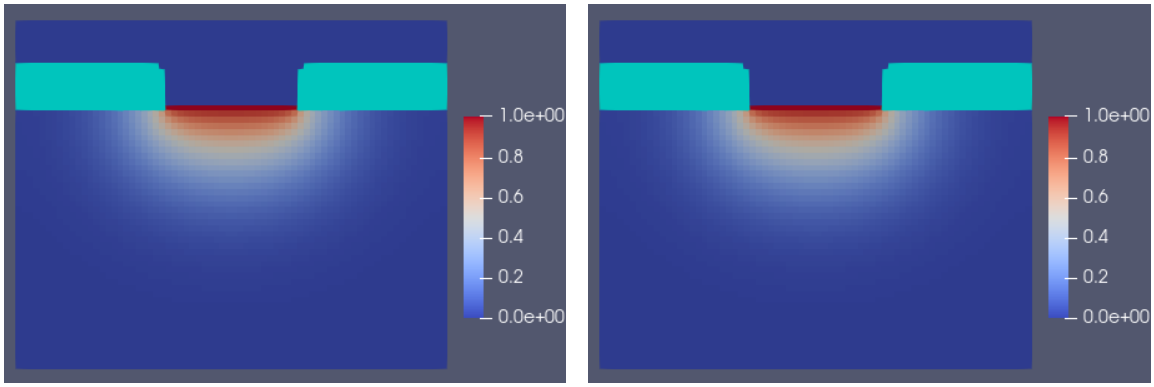
[1] Website: `https://eigen.tuxfamily.org/`

value 1, it is evident that the behavior becomes unphysical with negative values appearing at some cells and forming a periodic pattern of rapidly oscillating extremal values. If the value of S is further increased (or the simulation is run for a longer period of time), the values of the concentration further diverge, demonstrating an exponential blow-up of both positive and negative values. In contrast, the implicit version continues to provide seemingly correct results even when further increasing the time step beyond the explicit stable limit.



(a) $S = 0.95$ (b) $S = 1.024$

Figure 6: Diffused dopant concentration using the explicit finite difference method for different values of the stability factor S. In this simplified model normalized parameters are used without a real physical meaning. The inert mask is displayed as a turquoise solid.



(a) $S = 0.95$ (b) $S = 1.024$

Figure 7: Diffused dopant concentration using the implicit finite difference method for different values of the stability factor S. No instability occurs for $S > 1$.

The Euler integration methods used here have a low order of convergence. In practice, more accurate finite-difference schemes would normally be used, but this is not the main topic of this thesis.

4 Finite-Difference Time-Domain Method

The *finite-difference time-domain (FDTD) method* is a popular numerical method for simulating transient electromagnetic wave propagation that has proven itself to be very robust within its range of applicability. It is compatible with a wide range of scales, spanning the electromagnetic frequency spectrum, and it has a broad range of applications due to the many different manifestations of electromagnetic radiation, including light (optics), microwaves, radio frequencies and many others. The limitations are well-understood and many extensions and generalizations as well as combinations with other methods have been — and are still being — developed. The method is suitable for a problem if the required simulation time is not much longer than the time it takes for an electromagnetic wave to move through the domain and if simultaneously the characteristic lengths are comparable to the wavelength. This makes it unusable for the category of eddy current problems, which require larger relative time frames, but FDTD works well for optical and microwave problems and similar wave propagation problems at different length scales [12]. The reason for this is the explicit time stepping and the stability requirement that comes with it, imposing an upper limit on the permissible time step Δt, namely $\Delta t \leq \Delta x/(c\sqrt{d})$ in d-dimensional space, where c is the speed of light through the medium (material) being simulated. While this can be restrictive, it makes it a more efficient and scalable method than other (implicit) methods. For a similar reason it is also a relatively simple method, being based on finite differences, usually defined on a uniform rectangular grid, and it does not require storing a system matrix. Compared to more sophisticated (higher-order) methods, FDTD is easier to understand and implement from the ground up.

One difficulty arising in a practical setting is the need for formulating boundary conditions and, in particular, curved boundaries and other sub-cell features pose a challenge, resulting in staircase patterns that can become a major source of error if no special treatment is given to such geometric features, as with sub-cell techniques [13] or generalized methods for unstructured meshes [14]. However, such techniques are available if the need arises and the additional complexity can be justified.

The finite-difference scheme with its special alignment of electromagnetic field components in the space-time grid is also theoretically attractive, because it seems to reflect the geometry of Maxwell's equations in a discrete setting, preserving important symmetries and invariants and therefore closely matching the physics. It is not necessary to introduce artificial dissipation and the symplectic and energy-conserving nature of the scheme can be explained with a solid theoretical basis. A result of these favorable properties is the method's robustness, avoiding spurious effects that can appear in other methods [14].

It is also possible to adapt FDTD for simulating other physical wave propagation problems, for example acoustics.

Two main sources were used as learning material for the theoretical background of the FDTD numerical method and also for helping assess the technical aspects that a correct implementation must take into account.

The first source, the book *Computational Electrodynamics: The Finite-Difference Time-Domain Method* [13] by Taflove and Hagness, is a comprehensive treatment of the most important theory that was developed for the method in the decades after its inception, while also presenting many applications with their technical considerations in great detail. This book was used as a general reference text and it was helpful because it offers more thorough explanations and rigorous derivations; it was not used as the main guide to follow step by step, as it often goes into much greater detail than needed for the foundational implementation that is pursued here, to be used as a starting point for more advanced and specialized techniques in the future. A shorter treatment covering much of the essential theory of FDTD, including a basic dispersion and stability analysis, can alternatively be found in [12].

The second source, the book *Understanding the Finite-Difference Time-Domain Method* [15] by Schneider, takes a different approach, focusing on a high-level understanding with an easy-to-follow presentation of the basic theory and offering guidance for technical issues. It provided a good starting point for learning how to manage the technical challenges that one faces when creating a computational realization of the main components. It is not formally published, but it is available online, free of charge. It includes an implementation written in the C programming language, the main parts of it being listed in the corresponding chapters in the book, but also available separately. The code relies heavily on C preprocessor macros and raw for-loops and it was not used for this thesis project, although it did provide some insights into how certain aspects can be implemented.

A new and independent C++ implementation was instead developed for this thesis, which has no dependencies of its own, using only the functionality provided by the C++ standard library, while requiring the language version C++17. An overview of the implementation is given in the next section.

4.1 Implementation

The program code relies on the use of C++ templates and is written in a highly general way, such that essentially the same source code can produce the machine code for setting up and running the FDTD simulation for all spatial dimensions, supporting one, two and three dimensions. Care has been taken to minimize code duplication, avoiding the need for large nested loops and long chains of conditional statements in most places, concentrating them to a few central locations, with the aim of making the code more declarative to reduce the chance of errors. It was helpful to identify basic building blocks, creating auxiliary data structures such as vector and rectangle types supporting arbitrary dimension and various scalar types. This effort was made with the hope that it would increase readability and thus improve future maintainability.

The update loop for each physical field in the finite-difference algorithm that advances the simulation in time is split into two parts: first, one time step of the discretized Maxwell equations is performed, and subsequently corrections are applied to selected regions, as necessary. This way, the main field updates of the interior points can operate most efficiently, applying the general update uniformly, avoiding conditional expressions for points that need a differentiated treatment. The points that require application of special behavior such as

those at the boundary and source points are processed in a separate step, modifying their values based on current or past field values or overwriting them according to some fixed function of time. Sources and boundary conditions are bundled together and both applied as corrections. While this approach results in redundant calculations for interior points that are overwritten, the performance impact is negligible and it simplifies the main update loop.

Integrating the FDTD functionality with ViennaPS (or directly with the cell set) works simply by accepting and returning contiguous arrays of integers for the material identifiers and of floating-point values for each physical field component, sorted by the cell index (flattened multidimensional arrays). The material parameters for each occurring identifier can be registered separately. It may be necessary to embed the geometric structure generated externally (extending over the whole domain) in a slightly larger grid with some padding with empty space to allow specifying absorbing boundary conditions or for other techniques that restrict the structure of interest to a smaller region. For this purpose, resizing the grid can be performed by adapting the arrays of materials as input and the field values as output, allocating a new array for each one and copying the data, adding or stripping away the extra cells.

The integration with ViennaPS will be elaborated further in Chapter 5, where an application that combines both is described.

FDTD is known to be amenable to parallelization thanks to its explicit time stepping and the decoupled nature of the electric and magnetic fields and their components. An attempt was made to parallelize the update equations with **OpenMP**[1], using a very simple form of domain decomposition, but it did not lead to a significant speedup, possibly due to false sharing (cache invalidation) or some other undiscovered problem. A more primitive type of parallelization can be implemented by performing the updates of the individual components in directions x, y and z for each of the electric and magnetic fields concurrently, which did lead to a useful speedup, but it is of course limited to the number of dimensions of the domain space, utilizing only up to three processor cores. The lack of proper parallelization limits the scope to two-dimensional and smaller three-dimensional problems. Further investigation is needed to make the method scale well to larger problems.

Further notes on the implementation are given in the sections that explain some of the more technical aspects of the numerical method, following after a section that covers the theoretical background of the involved physics and mathematics.

4.2 Electromagnetic theory and optics

The presentation of the basics of electromagnetic theory below follows [13], Chapter 3, [16], Chapter 2, and [17].

Maxwell's equations describe the fundamental interactions and the evolution of electric and magnetic fields (electrodynamics) in a compact form. Collectively they can be written

[1] Website: `http://www.openmp.org`

in the vector calculus formulation as

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{4.1a}$$

$$\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J} \tag{4.1b}$$

$$\nabla \cdot \mathbf{D} = \rho \tag{4.1c}$$

$$\nabla \cdot \mathbf{B} = 0. \tag{4.1d}$$

In particular, the first two equations of (4.1) are known as Faraday's law (4.1a) for the electric field $\mathbf{E}$ and the magnetic field $\mathbf{B}$ and Ampere's law (4.1b), where a term for the so-called displacement current for a changing $\mathbf{D}$ is added to the actual charge current $\mathbf{J}$. $\mathbf{D}$ is known as the *electric flux density* (sometimes called *charge potential*) and $\mathbf{H}$ is the *magnetic field intensity*. In the literature $\mathbf{B}$ may also be called magnetic flux density and $\mathbf{H}$ then the magnetic field instead.

The displayed form of Maxwell's equations is the differential form. An alternative integral form of the equations can be derived and shown to be equivalent.

The negative sign in (4.1a) is consistent with Lenz's law, which states that a rotational electric current (eddy current) induced by a changing magnetic field generates its own magnetic field that opposes the initial change, thus effectively creating a resistance to it.

To complete the set of equations, constitutive equations must be added, which link the involved fields:

$$\mathbf{D} = \varepsilon \mathbf{E} \tag{4.2a}$$

$$\mathbf{B} = \mu \mathbf{H} \tag{4.2b}$$

The material parameters are the (absolute) electric permittivity $\varepsilon = \varepsilon_r \varepsilon_0$ and the (absolute) magnetic permeability $\mu = \mu_r \mu_0$ where the values ε_0 and μ_0 are for free space (vacuum) and the relative permittivity ε_r and permeability μ_r were introduced.

The electric current $\mathbf{J}$ can represent multiple types of current with different origins. The most useful ones for our case will be a current source term $\mathbf{J}_{src}$ that can be used to additively introduce electric energy at certain points and a conduction term $\mathbf{J} = \sigma \mathbf{E}$ according to Ohm's law with the electric conductivity parameter σ, causing energy dissipation due to heat loss. Both types of current can be combined by adding them together.

When moving charges are present, Maxwell's equations can be supplemented with the Lorentz force law to obtain a complete foundation from which all classical electromagnetic phenomena can be theoretically derived.

Maxwell's equations permit solutions $\mathbf{E}(t, \mathbf{x})$ in the form of transversal waves, as will be seen. The magnetic field always accompanies the electric field in a freely moving wave. These transversal electromagnetic waves can also advance through empty space, at the speed of light of vacuum.

Two of Maxwell's equations, (4.1a) and (4.1b), together form a type of a system of coupled first-order three-dimensional wave equations and they can be reduced to one second-order wave equation for only one field $\mathbf{E}$ or $\mathbf{H}$. However, the first-order system will be used for the numerical implementation. In a sense, it more fully captures the physics by including both the electric and magnetic fields. As wave equations, mathematically the PDEs are of the hyperbolic type.

For pure wave propagation (no charge production), the set of equations (4.1a) and (4.1b) describes the evolution in time and the two balance equations (4.1c) and (4.1d) are preserved by the temporal changes to $\mathbf{B}$ and $\mathbf{D}$ due to the differential operator identity $\nabla \cdot \nabla \times = 0$ (and $\nabla \cdot \mathbf{J} = -\frac{\partial \rho}{\partial t}$) and thus they are automatically satisfied for all times if the initial conditions satisfy them. It must still be confirmed that this is also true numerically in an analogous way for the discretized update formulas of the numerical solution scheme, but it is true for FDTD [14].

Throughout this thesis an isotropic medium is presupposed, simplifying the material parameters ε and μ to become scalars instead of general second-order tensors, as is the case for a material with direction-dependent electric or magnetic behavior, i.e. anisotropy. The materials that are are commonly encountered in optical problems are also usually non-magnetic ($\mu_r = 1$) and non-conducting (insulating) and they may be assumed non-dispersive (explained further below), or at least approximately.

Adding an equivalent magnetic current term $-\mathbf{J}_m$ to the right side of (4.1a) can be useful for artificially introducing an additive magnetic signal in the same way as it is done for electric signals with $\mathbf{J}$. It furthermore allows adding a magnetic conduction term $\mathbf{J}_m = \sigma_m \mathbf{H}$, to simulate magnetic loss with an equivalent magnetic conductivity σ_m. This also creates symmetry between the time-dependent relations and this translates to an equivalent symmetry in the update equations, which is also mirrored in the program implementation.

The electromagnetic field contains energy when non-zero electric and/or magnetic field excitations are present. This is similar to the interplay between kinetic and potential energy in elasticity for a continuum mechanics system. For pure electromagnetic wave problems, the electric and magnetic contributions to the energy should be about the same (under some simplifying assumptions), and the two fields are directly related and their orientations are such that in an isotropic medium $\mathbf{B}$ is perpendicular to $\mathbf{E}$ and both are perpendicular (transversal) to the propagation direction given by the wave vector $\mathbf{k}$ [18].

The instantaneous energy density in the electric and magnetic fields can be expressed in combination as [17]

$$u = \frac{1}{2}(\mathbf{E} \cdot \mathbf{D} + \mathbf{B} \cdot \mathbf{H}).$$

When the field varies with time, as is the case for a wave advancing through space, energy flows between different regions and the rate of the associated work is power, applied through a surface separating the regions.

The Poynting vector $\mathbf{P}$ (often named $\mathbf{S}$) represents the surface power density in the electromagnetic field. In three-dimensional space it is a vectorial quantity associated with surfaces and must be multiplied by an oriented surface element by forming the inner product with its representative vector. The Poynting vector is given as the cross product [17]

$$\mathbf{P} = \mathbf{E} \times \mathbf{H}.$$

Poynting's theorem introduces this quantity and it derives a fundamental identity that expresses the conservation of energy in the field, which in differential form is

$$-\frac{\partial u}{\partial t} = \nabla \cdot \mathbf{P} + \mathbf{J} \cdot \mathbf{E}. \tag{4.3}$$

It allows for electric currents $\mathbf{J}$ that can dissipate the electromagnetic energy, converting it to other forms of energy. If $\mathbf{J} = 0$, (4.3) is a continuity equation with exact conservation.

The integral of the Poynting vector over a finite surface, taking into account also its orientation, results in the (signed) power applied through that surface. This integral is commonly called power flow, although technically only energy flows, so a more directly interpretable choice of words would be surface power (or simply power), energy flow rate or energy current. Additionally integrating over a finite time interval gives one the energy flowing through the surface during that time period. The magnitude of the Poynting vector, $|\mathbf{P}|$, is known as the irradiance or intensity.

In optics, which explains wave-like phenomena such as refraction and reflection that are typically associated with visible light, it is common to work mainly with monochromatic plane waves (sinusoidal wave forms of a single frequency) as an idealization, because of the mathematical simplicity and the more natural frequency-based description of wave behaviors. For example, the electric field can have an amplitude $\mathbf{E}_0$ and follow an evolution in time t and space $\mathbf{x}$ with the angular frequency $\omega = 2\pi f$ (frequency f) and wave vector $\mathbf{k}$ where $k = |\mathbf{k}| = 2\pi/\lambda$ (wavelength λ) to take the form

$$\mathbf{E}(t, \mathbf{x}) = \mathbf{E}_0 \cos(\omega t - \mathbf{k} \cdot \mathbf{x}). \tag{4.4}$$

The complex exponential notation $\mathbf{E}(t, \mathbf{x}) = \mathbf{E}_0 \, e^{j(\omega t - \mathbf{k} \cdot \mathbf{x})}$ with the imaginary unit j simplifies the manipulation and combination of such expressions, taking the real part as physically meaningful. The amplitude components can then also take on complex values to allow a constant phase shift. Because arbitrary wave shapes can often be equivalently represented (at least approximately) as the superposition of many single-frequency waves (making up the Fourier spectrum), the harmonic wave is really the more elementary concept in this context and this formulation is also the basis for investigating dispersion relations $\omega(k)$, and it is essential for analyzing the stability and accuracy of FDTD. The phase velocity (or phase speed) $v_{ph} = \omega/k = \lambda f$, which is known as the (medium-dependent) speed of light, defines the travel speed of the planar wave fronts of constant phase $\omega t - \mathbf{k} \cdot \mathbf{x} = \text{const}$. If ω is linearly related to k, as is the case in vacuum, v_{ph} is constant, reflecting their ratio, and the wave movement is said to be free of dispersion since all frequencies move at the same speed, namely the speed of light c if in vacuum. In any other case there is dispersion and a material with a nonlinear relation $\omega(k)$ is called dispersive. This causes different frequency components to be shifted relative to each other over time and a combined wave changes its shape.

The cause for dispersion in a material can be a frequency-dependence in the material parameters, $\varepsilon_r(f)$ or $\mu_r(f)$. Handling dispersive materials in FDTD is covered in [15], Chapter 10, but here we assume no dispersion to be present, making the approximation that within the investigated frequency range the parameters are constants.

Numerical dispersion is another phenomenon that is relevant for discretized models of wave propagation and it can occur independently of physical dispersion caused by a material dependence on the frequency, $\varepsilon_r(f)$. Numerical dispersion will be discussed later, mainly in Section 4.6.

Assuming scalars ε and μ and assuming $\mathbf{J} = 0$, we can extract some useful information from Maxwell's equations by using the expressions for waves (4.4). Substituting analogous

plane waves for both $\mathbf{E}(t, \mathbf{x}) = \mathbf{E}_0 \cos(\omega t - \mathbf{k} \cdot \mathbf{x})$ and $\mathbf{H}(t, \mathbf{x}) = \mathbf{H}_0 \cos(\omega t - \mathbf{k} \cdot \mathbf{x})$ in Faraday's law (4.1a) and Ampere's law (4.1b), we see that both $\mathbf{E}$ and $\mathbf{H}$ (and therefore also $\mathbf{D}$ and $\mathbf{B}$) are orthogonal to $\mathbf{k}$ and they are also orthogonal to each other, with the relations $\omega\varepsilon\mathbf{E} = -\mathbf{k} \times \mathbf{H}$ and $\omega\mu\mathbf{H} = \mathbf{k} \times \mathbf{E}$. Furthermore, the two circuital Maxwell's equations can be combined to obtain a form of a second-order wave equation

$$\varepsilon \frac{\partial^2 \mathbf{E}}{\partial t} = -\mu^{-1} \nabla \times \nabla \times \mathbf{E}. \tag{4.5}$$

Entering the harmonic plane wave (4.4) in (4.5) results in $\omega^2 \mathbf{E}_0 = -\mathbf{k} \times \mathbf{k} \times \mathbf{E}_0$. Since $\mathbf{E}$ is orthogonal to $\mathbf{k}$, $\mathbf{k} \times \mathbf{k} \times \mathbf{E}_0$ is equal to $-k^2 \mathbf{E}_0$ and therefore we obtain the relation $\omega^2/k^2 = 1/(\varepsilon\mu)$. Thus, the phase speed is, in general, given by

$$v_{ph} = \frac{\omega}{k} = \frac{1}{\sqrt{\varepsilon\mu}} = \frac{c}{\sqrt{\varepsilon_r \mu_r}}.$$

In vacuum, $c = 1/\sqrt{\varepsilon_0 \mu_0}$ is approximately $3 \cdot 10^8$ m/s.

The *refractive index*[2] n of a material is defined as the ratio of the vacuum speed of light to the speed of light in the medium, $n = c/c' = c/v_{ph}$. In optics, one usually makes the assumption that $\mu_r \approx 1$, ignoring magnetic material effects. In a dielectric medium, $\varepsilon_r \neq 1$ and so we can set $n = \sqrt{\varepsilon_r}$ where usually $n > 1$. The frequency of an intruding electromagnetic plane wave remains unchanged, but the wave slows down. The wavelength λ is also scaled by the refractive index in the same way relative to the wavelength λ_0 in vacuum, as $\lambda = v_{ph}/f = (c/f)/n = \lambda_0/n$ with the same frequency f.

Important uses for the refractive index are Snell's law for determining the angles of refraction and reflection and the Fresnel equations, from which the reflectivity and transmittivity at a material interface can be obtained.

It is possible to define a complex-valued refractive index and permittivity to account for material absorption in the imaginary part.

Another quantity that can be defined is the wave impedance $Z = \sqrt{\mu/\varepsilon}$ or η.

Thin dielectric films can be used to reduce surface reflections, a principle that will be applied in Chapter 5. It works through a combination of constructive and destructive interference between the partially reflected and transmitted waves, increasing the transmitted wave signal and reducing the reflected energy due to a relative phase difference between the reflected wave at the first (n_1, n_2) and second interface (n_2, n_3), caused by the additional traveled distance through the film with refractive index n_2. There is always an additional phase shift of π in the electric component of the reflected wave if $n_1 < n_2$, but this cancels out if it occurs at both the first and second material interface. The phase difference from moving through the layer (twice) depends on the wavelength and so the layer thickness d should be chosen for a desired wavelength range. The value n_2 of the coating layer must be intermediate between the surrounding regions with n_1 and n_3. For a maximal antireflection effect, an ideal value can be chosen, if this is practical. In fact, repeated reflections in the film must also be considered in this case, see [18], Section 1.14 for the details. According to

[2] The symbol n is also used for the discrete time step, but it should be clear from the context which one is meant.

[18], Section 1.7, such an antireflection coating of thickness d should satisfy

$$d = \frac{\lambda}{4n_2}(2m+1), \quad m = 0, 1, 2, ..., \tag{4.6}$$

where the modified wavelength $\lambda' = \lambda/n_2$ must be used, because $2\pi \cdot 2d/\lambda'$ is the phase difference $\Delta\phi = 2dk'$ when moving through the layer twice. The refractive index would ideally be $n_2 = \sqrt{n_1 n_3}$, which ensures that the relative refractive indices n_2/n_1 and n_3/n_2 are equal. If multiple coatings are used on top of each other, equivalent relations should hold (approximately) at each interface, with $n_i < n_{i+1}$. The formulas do not need to be satisfied exactly, an antireflection effect can still be achieved with parameters that inevitably deviate from the theoretical values within a limited range.

Concerning the material parameters, we must keep in mind that they are dependent on the frequency (wavelength), as their values can vary considerably over the large spectrum. For example, they should be chosen for optical frequencies, if that matches the given application, even if we assume that they are constant within a limited frequency range of interest.

4.3 Electric and magnetic fields on a staggered grid

To discretize Maxwell's equations in the differential formulation with finite differences both in space and in time, the electric and magnetic field components must be located at discrete points in the grid and on the timeline, while taking into account their relative positions to define discrete versions of the differential operators (difference operators) that have good accuracy and satisfy some physical principles. The particular arrangement chosen in the FDTD method, also known as *Yee's scheme*, after the original description by Kane Yee in [19], uses a staggered grid in both space and time, so that each field component is located at a different point in a grid cell, either at the center of a cell edge (for electric components) or the center of a cell face (for magnetic components) and also with a half-step offset of all magnetic components in time, relative to the electric components. The arrangement in the spatial grid is illustrated in Figure 8 with a cube that displays all six discretized components of **E** and **H** belonging to the same cell.

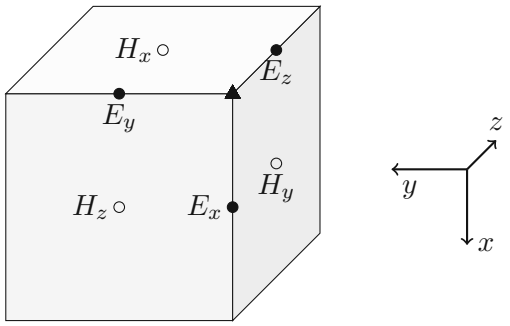


Figure 8: Locations of the nodes for the components of the **E** and **H** fields, marked with filled or hollow circles. The triangle marks the reference point of the cell.

This choice is motivated by looking at the structure of the differential operators in Maxwell's equations and the relations between them. It was already mentioned in Section 2.1 that centered differences used for approximating first-order differential quotients (in a one-dimensional problem) and in particular the leap-frog scheme can be an appropriate choice for wave propagation problems. The leap-frog scheme for the advection equation $\frac{\partial u}{\partial t} = -\frac{\partial u}{\partial x}$ with one scalar function u

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\,\Delta t} = -\frac{u_{i+1}^n - u_{i-1}^n}{2\,\Delta x}$$

relates values at even-numbered time steps to values at odd-numbered time steps, where the pairs of points are either at even-numbered and odd-numbered grid points or vice versa. Continuing the pattern in the space-time grid by applying this four-point stencil to the participating points in all directions, one obtains all points that are dependent on each other through the update equations. This pattern contains only half of the space-time points, however, and the other half, which is interspersed between them, is shifted by one cell length and is also updated according to the same pattern, see Figure 9. Even though two consecutive values in time are localized at the same point in space, they are not related by the update scheme in any way due to the fact that two sets of initial values for the instants with index $n = 0$ and $n = 1$ must be provided. In a sense, the 1D leap-frog scheme performs two independent simulations in parallel that each process half of the grid values of u (counting all values that must be stored in memory from two consecutive instants).

We can now take one of these two partial schemes, operating on only one of the two independent sets of grid points, and transfer the pattern to the pair of Maxwell's equations in 1D,

$$\frac{\partial(\mu H_y)}{\partial t} = \frac{\partial E_z}{\partial x},$$
$$\frac{\partial(\varepsilon E_z)}{\partial t} = \frac{\partial H_y}{\partial x}.$$

There are now two scalar functions, where E_z is assigned to even-numbered time and space points and H_y to odd-numbered points in the analogous scheme. However, half-step indices are commonly used in FDTD instead and values are denoted by $E_z^n[i]$ and $H_y^{n+1/2}\left[i + \frac{1}{2}\right]$, for example. The two schemes are illustrated and compared in Figure 9, displaying the relevant points for centered finite differences around one space-time point. The discretized forms of the two Maxwell equations are applied in turns, each advancing time by $\Delta t/2$. The exact procedure, resulting in a complete finite-difference scheme, is explained in Section 4.4.

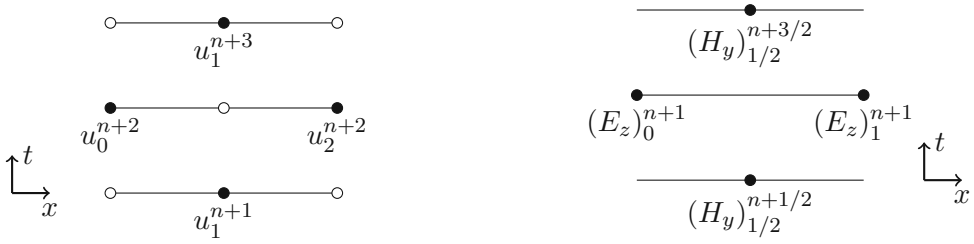


Figure 9: Leap-frog scheme (left) and Yee's scheme (right) in 1D. The labeled nodes are related to each other in an update step through centered finite differences in space and time around the central point. White nodes do not participate in the update of labeled nodes and thus the leap-frog scheme effectively does two independent FDTD-like simulations on one grid.

Extending this idea to the discretization of Maxwell's equations in three dimensions with their rotation (curl) operators applied to the fields $\nabla \times \mathbf{E}$ and $\nabla \times \mathbf{H}$ naturally leads one to look at grid planes that contain all values necessary for the spatial centered differences placed symmetrically around the other field component from $\partial(\mu\mathbf{H})/\partial t$ or $\partial(\varepsilon\mathbf{E})/\partial t$ that is normal to the plane. See Figure 10 for the arrangement.

Two different types of such layers can be identified in the Yee lattice, transverse electric (TE) layers corresponding to Faraday's law with $\nabla \times \mathbf{E}$ (aligned with cell faces) and transverse magnetic (TM) layers corresponding to Ampere's law with $\nabla \times \mathbf{H}$ (cutting through the centers of cells). Figure 10 shows examples for both types of layers oriented in z-direction, denoted by TE(z) and TM(z).

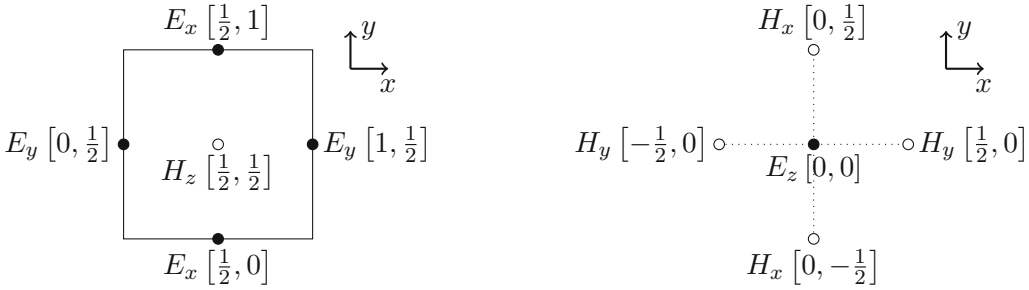


Figure 10: Locations of the nodes for the components of the $\mathbf{E}$ and $\mathbf{H}$ fields in a transverse electric layer TE(z) (left) and a transverse magnetic layer TM(z) (right). The electric and magnetic values are defined at different time instants (n and $n+1/2$). The TM(z) layer is positioned in the plane at $z = \Delta x/2$. The dotted lines indicate where the edges of the primary grid are located (in parallel planes above and below).

In the case of a two-dimensional simulation, one considers fields that do not vary in one direction so that the problem is effectively reduced to a single layer. This means that one must choose either a TE or a TM layer. Which of the three layers of a type one chooses is not important for the results since a rotation of the grid followed by a coordinate transformation can transfer it to another one. The update steps for the transverse components take place in layers that are orthogonal to the main layer and are effectively one-dimensional, containing variation in only one spatial direction. A fully one-dimensional simulation performs two

such 1D update steps that vary in the same direction. In the pair of 1D Maxwell equations used above, the time steps for H_y and E_z would take place in orthogonal TE(y) and TM(z) planes in 3D where the intersection line in x-direction goes through nodes belonging to both components that are shared by the intersecting planes.

The choice for assigning values to points in the space and time grids as explained above has been proven to lead to a robust algorithm and to be compatible with the physics and geometry of Maxwell's equations.

4.3.1 Integral interpretation

An alternative viewpoint on Yee's scheme is based on the integral formulation of Maxwell's equations, which can be derived from the differential form with Stokes theorem, relating integrals over surfaces to integrals over their closed bounding curves:

$$\frac{\partial}{\partial t}\int_S \mathbf{B}\cdot d\mathbf{S} + \int_S \mathbf{J}_m \cdot d\mathbf{S} = -\oint_{\partial S} \mathbf{E}\cdot d\mathbf{L}$$
$$\frac{\partial}{\partial t}\int_S \mathbf{D}\cdot d\mathbf{S} + \int_S \mathbf{J} \cdot d\mathbf{S} = \oint_{\partial S} \mathbf{H}\cdot d\mathbf{L}$$

The numerical values for the components of **E** and **H** take on the meaning of integrals of the vector fields over straight lines or rectangular surfaces in the staggered grid, if a distinction between the pairs **E**, **H** and **D**, **B** is made. The values in the differential formulation can then be interpreted as average densities over the sub-cells.

An advantage of this viewpoint is that smaller geometric features within cells such as wires and thin plates can be included with the integral formulation, and discontinuities can be treated in a natural way [13]. It can also provide additional insight into the geometry of the physics.

Because components H_i are located at cell faces, but **H** is integrated over lines, the values essentially belong to additional lines that pass centrally through cell faces in their normal direction. When these lines are all connected, they form a *dual grid* that has the same shape as the *primal grid* but is staggered so that corners from one grid are at centers of volume cells from the other (0-cells are paired with dual 3-cells) and lines are paired with dual faces and vice versa. One can pass from components of **E** and **H** (line integrals) to **D** and **B** (surface integrals) by multiplying with the material parameters ε or μ and additionally by the surface area (dual or primal faces) divided by the line length (primal or dual lines).

Physical fields have an association with geometric elements so that it only makes sense to integrate them over regions of a certain dimension. For example, the electric field **E** can be integrated over lines to obtain a voltage and the electric flux density **D** can be integrated over surfaces to obtain an electric charge. Similarly, **B** is integrated over surfaces and **H** over lines. In classical vector calculus, no explicit distinction is made between vectors that are associated with lines or surfaces and their orientation (inner or outer).

A more thorough justification for the specific alignment of the field components can be given on the grounds of differential geometry, specifically the calculus of differential forms (exterior calculus) in the discrete setting, see [14]. It can be seen as a mathematical theory that takes the additional geometric structure inherent in physical theories that the vector

calculus formulation ignores and formalizes it in a coherent way. The existence of a dual grid is recognized and it may be explicitly constructed.

This theory enables the formulation of generalized numerical methods.

An intuitive explanation for the association of physical quantities with space and time elements and their orientation, and also providing the motivation for it from the physical measurement processes, can be found in [20].

4.4 Finite differences on the staggered grid

A complete update scheme for Maxwell's equations in three dimensions with centered finite differences in space and time is now constructed by combining TM and TE layers in all directions.

Introducing the notation $i' = i + 1/2$ for spatial and temporal indices, we can write the update equations for TE(z) layers

$$\begin{aligned}&\mu \frac{H_z^{n'}[i',j',k] - H_z^{n'-1}[i',j',k]}{\Delta t} \\ &= -\frac{E_y^n[i+1,j',k] - E_y^n[i,j',k]}{\Delta x} + \frac{E_x^n[i',j+1,k] - E_x^n[i',j,k]}{\Delta x}\end{aligned}$$

and for TM(z) layers

$$\begin{aligned}&\varepsilon \frac{E_z^{n+1}[i,j,k'] - E_z^n[i,j,k']}{\Delta t} \\ &= \frac{H_y^{n'}[i',j,k'] - H_y^{n'}[i'-1,j,k']}{\Delta x} - \frac{H_x^{n'}[i,j',k'] - H_x^{n'}[i,j'-1,k']}{\Delta x}\end{aligned}$$

which can be rearranged to obtain expressions for future values $H_z^{n+\frac{1}{2}}$ and E_z^{n+1} in terms of known values at time $t = n\,\Delta t$. From these equations a general pattern for all components can be discerned.

In the implementation, the half-step offsets of values located within cells are discarded (rounded down to an integer), so that all components belonging to one cell have the same array index. Using the notation $\bar{E}_1^n[i,j,k] = E_x^n[i',j,k]$, $\bar{H}_1^n[i,j,k] = H_x^{n'}[i,j',k']$ etc., and the index vector $\mathbf{i} = (i_1, i_2, i_3) = (i,j,k)$ we can write

$$\begin{aligned}&\mu \frac{\bar{H}_I^n[\mathbf{i}] - \bar{H}_I^{n-1}[\mathbf{i}]}{\Delta t} \\ &= -\frac{\bar{E}_{I+2}^n[\mathbf{i}+\mathbf{e}_{I+1}] - \bar{E}_{I+2}^n[\mathbf{i}]}{\Delta x} + \frac{\bar{E}_{I+1}^n[\mathbf{i}+\mathbf{e}_{I+2}] - \bar{E}_{I+1}^n[\mathbf{i}]}{\Delta x} \\ &\varepsilon \frac{\bar{E}_I^{n+1}[\mathbf{i}] - \bar{E}_I^n[\mathbf{i}]}{\Delta t} \\ &= \frac{\bar{H}_{I+2}^n[\mathbf{i}] - \bar{H}_{I+2}^n[\mathbf{i}-\mathbf{e}_{I+1}]}{\Delta x} - \frac{\bar{H}_{I+1}^n[\mathbf{i}] - \bar{H}_{I+1}^n[\mathbf{i}-\mathbf{e}_{I+2}]}{\Delta x},\end{aligned}$$

where $\mathbf{e}_I$ is the unit vector in direction I and indices I are defined to continue cyclically so that $E_{3+1} = E_1 = E_x$ etc.

With the difference operators $\Delta_I^{\pm}$ defined by $\Delta_J^+ \bar{E}_I^n[\mathbf{i}] = \bar{E}_I^n[\mathbf{i}+\mathbf{e}_J] - \bar{E}_I^n[\mathbf{i}]$ and $\Delta_J^- \bar{H}_I^n[\mathbf{i}] = \bar{H}_I^n[\mathbf{i}] - \bar{H}_I^n[\mathbf{i}-\mathbf{e}_J]$ this is written more compactly as

$$\bar{H}_I^n[\mathbf{i}] = \bar{H}_I^{n-1}[\mathbf{i}] - \frac{\Delta t}{\mu\,\Delta x}\left(\Delta_{I+1}^+ \bar{E}_{I+2}^n[\mathbf{i}] - \Delta_{I+2}^+ \bar{E}_{I+1}^n[\mathbf{i}]\right)$$
$$\bar{E}_I^{n+1}[\mathbf{i}] = \bar{E}_I^n[\mathbf{i}] + \frac{\Delta t}{\varepsilon\,\Delta x}\left(\Delta_{I+1}^- \bar{H}_{I+2}^n[\mathbf{i}] - \Delta_{I+2}^- \bar{H}_{I+1}^n[\mathbf{i}]\right).$$

In general, the material parameters ε and μ can vary between cells (possibly even for each component independently), so that we introduce the update coefficient arrays $k_{EE}[\mathbf{i}] = k_{HH}[\mathbf{i}] = 1$, $k_{EH}[\mathbf{i}] = \Delta t/(\varepsilon[\mathbf{i}]\,\Delta x)$, $k_{HE}[\mathbf{i}] = \Delta t/(\mu[\mathbf{i}]\,\Delta x)$, and the final update formulas become

$$\bar{H}_I^n[\mathbf{i}] = k_{HH}[\mathbf{i}] \cdot \bar{H}_I^{n-1}[\mathbf{i}] - k_{HE}[\mathbf{i}] \cdot \left(\Delta_{I+1}^+ \bar{E}_{I+2}^n[\mathbf{i}] - \Delta_{I+2}^+ \bar{E}_{I+1}^n[\mathbf{i}]\right) \tag{4.7}$$
$$\bar{E}_I^{n+1}[\mathbf{i}] = k_{EE}[\mathbf{i}] \cdot \bar{E}_I^n[\mathbf{i}] + k_{EH}[\mathbf{i}] \cdot \left(\Delta_{I+1}^- \bar{H}_{I+2}^n[\mathbf{i}] - \Delta_{I+2}^- \bar{H}_{I+1}^n[\mathbf{i}]\right) \tag{4.8}$$

To also support lossy materials through a conduction term we must discretize

$$\frac{\partial(\mu\mathbf{H})}{\partial t} + \sigma_m \mathbf{H} = -\nabla \times \mathbf{E}$$
$$\frac{\partial(\varepsilon\mathbf{E})}{\partial t} + \sigma \mathbf{E} = \nabla \times \mathbf{H}.$$

The equations with centered differences relate values at instants n and $n+\frac{1}{2}$, but $\mathbf{H}$ and $\mathbf{E}$ are defined at instants of the respective other type. We therefore take the average values between two instants, $(\bar{H}_I^{n-1}[\mathbf{i}] + \bar{H}_I^n[\mathbf{i}])/2$ and $(\bar{E}_I^n[\mathbf{i}] + \bar{E}_I^{n+1}[\mathbf{i}])/2$, which can be combined with the time-difference terms to obtain the discretized equivalents for the left sides of above equations,

$$\mu\frac{\bar{H}_I^n[\mathbf{i}] - \bar{H}_I^{n-1}[\mathbf{i}]}{\Delta t} + \sigma_m \frac{\bar{H}_I^{n-1}[\mathbf{i}] + \bar{H}_I^n[\mathbf{i}]}{2}$$
$$= \left(\frac{\mu}{\Delta t} + \frac{\sigma_m}{2}\right)\bar{H}_I^n[\mathbf{i}] - \left(\frac{\mu}{\Delta t} - \frac{\sigma_m}{2}\right)\bar{H}_I^{n-1}[\mathbf{i}]$$

and

$$\varepsilon\frac{\bar{E}_I^{n+1}[\mathbf{i}] - \bar{E}_I^n[\mathbf{i}]}{\Delta t} + \sigma \frac{\bar{E}_I^n[\mathbf{i}] + \bar{E}_I^{n+1}[\mathbf{i}]}{2}$$
$$= \left(\frac{\varepsilon}{\Delta t} + \frac{\sigma}{2}\right)\bar{E}_I^{n+1}[\mathbf{i}] - \left(\frac{\varepsilon}{\Delta t} - \frac{\sigma}{2}\right)\bar{E}_I^n[\mathbf{i}].$$

Using this, we can derive the update formulas as above. Therefore, the only modifications that need to be made in the equations to add support for material conductivity are to the update coefficients k_{EE} etc., while the expressions for the update formulas remain

unchanged. The update coefficients including the conductivities are given by

$$k_{EE}[\mathbf{i}] = \frac{1 - \frac{\sigma[\mathbf{i}]\,\Delta t}{2\,\varepsilon[\mathbf{i}]}}{1 + \frac{\sigma[\mathbf{i}]\,\Delta t}{2\,\varepsilon[\mathbf{i}]}} \tag{4.9a}$$

$$k_{HH}[\mathbf{i}] = \frac{1 - \frac{\sigma_m[\mathbf{i}]\,\Delta t}{2\,\mu[\mathbf{i}]}}{1 + \frac{\sigma_m[\mathbf{i}]\,\Delta t}{2\,\mu[\mathbf{i}]}} \tag{4.9b}$$

$$k_{EH}[\mathbf{i}] = \frac{\frac{\Delta t}{\varepsilon[\mathbf{i}]\,\Delta x}}{1 + \frac{\sigma[\mathbf{i}]\,\Delta t}{2\,\varepsilon[\mathbf{i}]}} \tag{4.9c}$$

$$k_{HE}[\mathbf{i}] = \frac{\frac{\Delta t}{\mu[\mathbf{i}]\,\Delta x}}{1 + \frac{\sigma_m[\mathbf{i}]\,\Delta t}{2\,\mu[\mathbf{i}]}} \tag{4.9d}$$

In the code, a general function is implemented, which can perform any of the component updates defined above. It takes as arguments the participating field component arrays with their update coefficients and the orientation of the plane as well as information about which type of field is updated, which is used to select the appropriate difference operator and the sign, and also to exclude field-dependent boundary points. The sign could alternatively be incorporated into the update coefficients.

To avoid unnecessary computations during iterations, update coefficients are precomputed at the beginning of the simulation and stored in scalar arrays alongside the field component values.

Even though we speak of two- and one-dimensional problems, one does not actually obtain a truly two-dimensional electromagnetic formulation, since Maxwell's equations, as formulated here, are inherently three-dimensional, relying on the vector field rotation. Instead, one should think of the lower-dimensional problems as still three-dimensional, but with no variation of the relevant field quantities in the removed dimensions. Therefore, instead of picturing a 2D simulation as taking place in an isolated plane, it can be imagined as a stack of parallel layers, both TE and TM, where all layers of a type are perfectly synchronized. The fields in the different types of layers are decoupled and they can evolve independently.

The updates that take place in a layer that contains a direction in which the fields are constant can still use the general update formulas; the difference-term for the transverse component that has no variation is zero in this case. In the implementation, the same general update function can be reused. It accepts any indexable array type for the transverse components, using a C++ template parameter, which allows to simulate a uniform field in one direction by passing an instance of a special type named `RepeatArray` that always returns the same value (zero) on element access, no matter what index is passed as the argument.

Being based on centered differences, this scheme, like the leap-frog scheme, is second-order accurate in space and time.

4.5 The finite-difference time-domain method

Having the necessary theory and the basic discretization scheme described, we can finally put together the pieces to create the necessary data structures and define the iterative algorithm in the implementation.

However, some more considerations are necessary to allow easily scaling the method to different domain sizes and frequency ranges.

An interesting and useful property of an FDTD simulation is the ability to easily scale it to various frequencies or wavelengths of the electromagnetic spectrum, as long as certain stability conditions are considered.

Since the numerical method can be implemented so that only the ratio $\Delta t/\Delta x$ is used for the internal calculations, the same simulation results can, in principle, be interpreted as belonging to one or more particular choices from a wide range of frequencies, with the domain and duration accordingly scaled.

Conversely, for a fixed domain size, it would be sufficient to specify a base frequency or wavelength for a wave in vacuum that represents the target frequency range of interest, and then automatically calculate suitable values for the spatial and time resolution to obtain a stable simulation. Important parameters that decide the stability of an FDTD simulation are the so-called *Courant number* and the number of grid cells per wavelength, which can be a non-integer value. Together they, once fixed, determine the spatial and consequently the temporal resolution of the discretized system.

The Courant number, denoted by S_C here, is defined as $S_C = c\,\Delta t/\Delta x$, with the free-space speed of light c, and it plays the role of a numerical stability factor. For a particular wavelength of interest, which can be the main wavelength component of a pulse signal, the number of cells per wavelength is $N_\lambda = \lambda_0/\Delta x$, defined relative to the spatial step Δx of the grid. It may also be called the grid sampling density. It is always defined with respect to the wavelength in vacuum λ_0; an appropriate interpretation in a dielectric medium must consider the scaling of the wavelength by a factor n, the refractive index. One should also note that the ideal continuous wavelength is used and not the numerical one, which can deviate from it. If a free-space frequency f_0 is given instead, it can be converted with the formula $\lambda_0 = c/f_0$.

To obtain accurate and stable results, N_λ should have a value of about 10-20 or more for the shortest wavelength of interest (relevant to the physical problem) [13]. In case there is a material in the domain for which $n = \sqrt{\varepsilon_r}$ is significantly larger than 1, it is necessary to refer to the dielectric wavelength $\lambda = \lambda_0/n$ for the maximum refractive index n in the domain and thus scale N_λ by n.

Due to errors accumulating over time, it may be necessary to further increase the resolution in a so-called *electrically large problem*, where a large number of time steps is simulated if a wave travels over a relatively long distance (meaning a large number of cells) in at least one direction of the domain.

We would like to make all involved parameters independent of the exact step sizes Δx and Δt. When the ratio $\Delta t/\Delta x$ is needed in an expression, it can be expressed using S_C,

which is known. In the update coefficients k_{EH} and k_{HE} in (4.9), we can insert

$$\frac{\Delta t}{\varepsilon[\mathbf{i}]\,\Delta x} = \frac{S_C\,\eta_0}{\varepsilon_r[\mathbf{i}]}$$
$$\frac{\Delta t}{\mu[\mathbf{i}]\,\Delta x} = \frac{S_C}{\mu_r[\mathbf{i}]\,\eta_0},$$

where $\eta_0 = \sqrt{\mu_0/\varepsilon_0}$ defines the wave impedance of free space. The material parameters ε_r and μ_r are already relative and can be used directly in a scale-independent implementation. However, the conductivity parameters σ and σ_m must be multiplied by the time step Δt to define their effect. To avoid this dependence, the values $\frac{\sigma[\mathbf{i}]\,\Delta t}{2\,\varepsilon[\mathbf{i}]}$ and $\frac{\sigma_m[\mathbf{i}]\,\Delta t}{2\,\mu[\mathbf{i}]}$ may be identified in (4.9) as an electric and a magnetic *loss factor*, which are the values that are eventually registered for a material and used to calculate the update coefficients. However, instead of specifying this value directly during the setup of the simulation, it is more practical to have it calculated from the skin depth δ_{skin} for a particular wavelength, expressed as a multiple N_L of Δx so that $\delta_{skin} = N_L\,\Delta x$, and the corresponding wavelength, expressed as cells per wavelength $N_\lambda = \lambda_0/\Delta x$. If the need arises, a given conductivity can be converted to the loss factor with the appropriate function in the code, passing the necessary parameters.

The skin depth is defined as the distance in the conductive material after which the amplitude of a wave of a particular wavelength has decayed to $1/e$. δ_{skin} coincides with the skin depth that characterizes the skin effect, which is the effect that a current through a wire is concentrated at the surface of the wire. Following a theoretical derivation [15], the loss factor can be expressed in terms of the parameters as

$$\frac{\sigma\,\Delta t}{2\,\varepsilon} = S_C \frac{\pi}{N_\lambda}\sqrt{\left(1 + \frac{N_\lambda^2}{2\pi^2 N_L^2 \varepsilon_r \mu_r}\right)^2 - 1}.$$

An equivalent loss can be defined for a magnetic conductivity σ_m with independent N_L and N_λ.

Source functions are defined with the number of time steps as parameters instead of the physical time (for the pulse width and delay).

During the simulation setup, the domain size and the stability factor S_C are first required. If S_C is not specified, it is set to the stable limit $1/\sqrt{d}$ by default, explained in Section 4.6. The domain size, as a multiple of the cell size in all directions, is selected depending on the physical problem size and Δx, determined from N_λ for a target wavelength. Internally in the simulation, N_λ and Δx do not need to be known.

As the next step, wave sources, boundary conditions and materials are added to the simulation domain.

Unused field component arrays in 2D and 1D simulations are left empty, which is also used to detect which components are active and should be updated.

The simulation then progresses for a selected number of time steps, iteratively updating the field arrays and applying source and boundary conditions after each update step per field type, taking into account the half-step offset in time.

The main update algorithm skips all outer boundary nodes that are missing at least one neighbor node so that by default the values at these points remain zero. Such points

therefore act like a perfect electric conductor (PEC) if they are electric nodes and like a perfect magnetic conductor (PMC) if they are magnetic nodes, and these perfect conductors cause full reflections of incoming waves. In particular, at a PEC boundary the reflected wave has an electric component with opposite sign (due to a phase shift of π), while the magnetic component is unchanged, and vice versa for a PMC boundary.

Some of the mentioned simulation components will be further explained in the following sections.

4.6 Stability

A requirement that is known to be essential for the numerical stability of a method is the CFL condition, which was described in Chapter 3 for a problem with a parabolic PDE, marked by dissipative behavior. It states that the numerical domain of dependence of a point must include the analytical (or physical) domain of dependence of that point in the limit of a finely resolved grid. In other words, physically meaningful behavior (causality) necessitates that the speed at which a signal can move from a point in the grid to a reference point must match or exceed the speed allowed by the continuous version of the mathematical model (the PDE).

The hyperbolic nature of PDEs underlying wave propagation problems, such as the pair of Maxwell's equations, is characterized by a finite maximum speed of propagation c (the free-space speed of light), which we will try to relate to the maximum signal travel speed in the grid. Let us look at a simplified model where a single point value can affect neighbor points in the main grid directions in one update step (in Yee's scheme it works somewhat differently due to the transverse nature and the half-cell offsets, which will be addressed below). The fact that in each time step the field values in the grid cells are updated using only information from direct cell neighbors means that the time required for information to travel between two points P_1 and P_2 in the grid is $\tau = \|P_2 - P_1\|_1 \, \Delta t$, where $\|\mathbf{d}\|_1 = \sum_i |d_i|$. This absolute-norm distance is also called the Manhattan norm. Thus, the maximum speed of information travel between points in the grid is equal to the Euclidean distance divided by this time interval,

$$v(P_1, P_2) = \frac{\|P_2 - P_1\|_2 \, \Delta x}{\|P_2 - P_1\|_1 \, \Delta t}.$$

In a d-dimensional grid, this speed attains its minimum in directions along cell diagonals, namely $\sqrt{d}\,\Delta x/(d\,\Delta t)$, and so we obtain the condition

$$\frac{\Delta x}{\sqrt{d}\,\Delta t} \geq c$$

and with the Courant number (stability factor) $S_C = c\,\Delta t/\Delta x$ it becomes $S_C \leq 1/\sqrt{d}$. The speed in grid axis directions is then $\Delta x/\Delta t = c/S_C \geq c$. The numerical domain of dependence for multiple time steps is illustrated in Figure 11; it must contain a circle of radius equal to the distance a real wave can cover within the time span.

The necessary condition given above is also sufficient for stability in FDTD, which is not immediately obvious. The separation of time steps into half-steps in the Yee algorithm

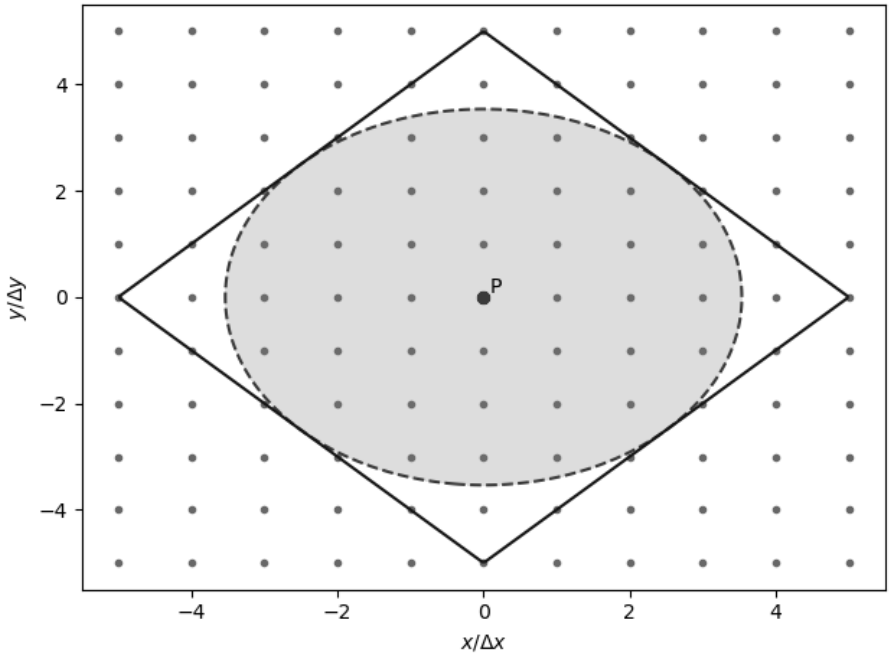
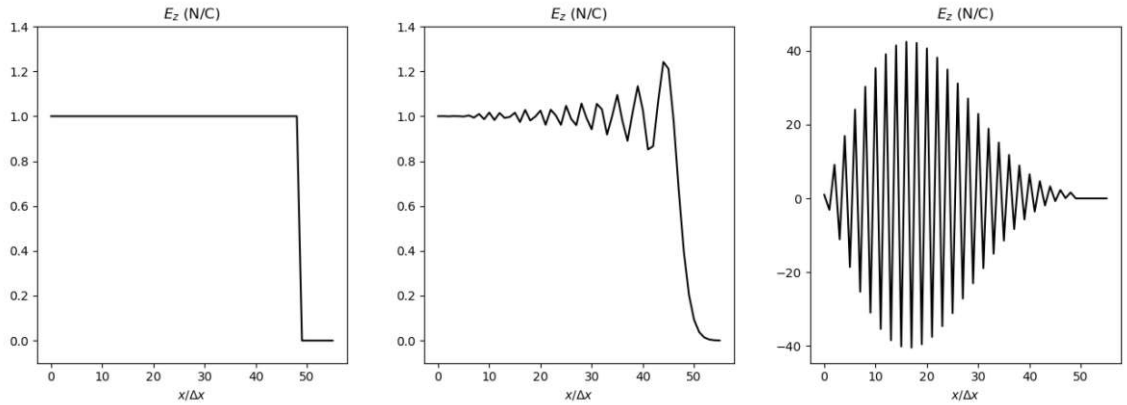


Figure 11: CFL condition: The numerical domain of dependence of the central point over five time steps (area enclosed in straight solid lines) must contain the analytical domain of dependence (circle with dashed line) with a speed $c = \Delta x/(\sqrt{2}\,\Delta t)$.

allows diagonal transfer in a grid plane even within one time step (for example from E_x to B_z to E_y), but this still takes two full time steps to reach the next diagonal cell (E_x again) and does not change the results above. However, because the waves are transversal, a field component cannot move directly in a direction parallel to it, and so the simplified model of field values affecting all direct neighbors does not work. Instead, monochromatic waves in different directions are the elementary concept, which are analyzed for a proper dispersion and stability analysis.

The fact that the speed in the grid must be permitted to exceed the speed of light for dimensions two and three does not mean that waves will actually travel faster than c, as a naive look may lead one to believe. Although it seems correct to assume that for a pulsed wave a part of the signal will in fact travel at the maximum allowed speed along the grid's main directions, this (superluminal) signal will decay exponentially, with a base factor proportional to the time step Δt (the Courant number $S_C < 1$) and it will therefore not be of much significance beyond numerical error or low-amplitude noise. For harmonic wave excitations in the grid, representing infinitely extended plane waves, it can be shown that there is an analytic solution, from which a dispersion relation can be derived to obtain the phase velocity in the grid, as it is performed in [12] or in [13], Chapter 4. The result is that well-resolved plane waves travel with a speed slightly below c when no material is present. Increasing the grid resolution reduces dispersion if the wavelength is fixed and the phase speed approaches c. The exact numerical dispersion characteristic also depends on the travel direction of the wave, making it anisotropic.

The effect of different stability factors S_C is demonstrated in Figure 12.



(a) $S_C = 1$ (b) $S_C = 0.5$ (c) $S_C = 1.005$

Figure 12: A 1D FDTD simulation with the result of a step source function originating from the left after some time has passed, for different choices of S_C. When $S_C = 1$, the input signal travels unaltered to the right with speed $c = \Delta x/\Delta t$. In any other case there is numerical dispersion, causing a distortion of the shape, and if $S_C > 1$, instability occurs.

4.7 Absorbing boundary conditions

When propagating electromagnetic waves are simulated numerically in a limited region of space, it is usually necessary to apply a special treatment to the boundaries of the discretized domain to ensure meaningful behavior in the case of a wave reaching those locations. Often, the desired effect is for outgoing waves to no longer affect the simulation in the restricted domain, as if they would simply vanish, i.e. the wave should continue moving away as if there was infinite empty space all around. Because the discretized domain necessarily has finite extent, a method must be applied to terminate the grid at the boundary while approximating the effect of extending it to infinity with vacuum. Simply cutting off the domain and ignoring the boundary (setting the field values outside to zero) is not acceptable, because it is equivalent to creating a perfect electric or magnetic conductor (depending on which field outside the domain has a direct neighbor inside), and it will cause incident signals to be fully reflected back into the domain, which is not the physical behavior that one wants in this case. Therefore, another way to terminate the grid must be found, avoiding reflections that are not appropriate for the physical problem that is simulated. Extending the domain with empty space, far enough to ensure that a reflected wave will not reach the main region within the simulation time, would be an option, but it would be inefficient, requiring a potentially large number of additional computations.

Several techniques are available for simulating an infinite domain, requiring a relatively smaller amount of added computational effort.

Analytical absorbing boundary conditions (ABCs) are one option. The idea is to explicitly set the field values (electric or magnetic) at the boundary points that are missing a neighboring point value on at least one side that would be needed for the regular update, in such a way that for an outgoing wave originating from inside the computational domain, its continuation into empty space is simulated approximately. All interior points close to the boundary are still processed in the normal update step. It works by solving a (one-directional) wave equation, approximating the exact differential equation with finite differences, using one or more neighbor values, possibly from multiple time steps. This is the technique chosen for this project, following [15]. Despite some limitations, it is sufficient for suppressing the reflections in an acceptable way for the target simulations. A problem was encountered with the Gauss pulse wave, showing noticeable reflections in certain cases, but with the Ricker wavelet, which is more suitable for electromagnetic wave simulations, the absorption shown is acceptable (i.e. not making interpretations of the relevant field behavior in the visualization output difficult).

A more advanced technique is the so-called *perfectly matched layer* (PML) absorbing boundary, which works by extending the domain by a thin layer of additional cells into which the incoming waves can propagate, but their signal strength will rapidly decay when traveling through the layer in a direction normal to the interface, due to material absorption. The *perfect matching* refers to a suitably chosen combination of material parameters that results in minimal reflections at the interface to the simulated domain (impedance matching). In the continuous realm, a truly perfect matching can be formulated in principle, with no reflections, but this ideal case cannot quite be realized numerically. Nevertheless, the absorption performance of a high-quality PML is typically superior to analytical ABCs. It is, however, more complicated to understand and implement. The implementation of a perfectly matched layer is a subject for future work.

The basic analytical ABC that was implemented supports multiple levels of accuracy. It is applied separately for each boundary field component that is in a direction parallel to the boundary, considering only field values of the same type at neighboring cells in normal direction for the PDE. Depending on the location of the boundary plane (aligned with a grid plane or at a half-cell offset), it is either the electric or magnetic fields. The components normal to the boundary have no adjacent interior points that need the values for their update and can therefore be left out, leaving them at zero.

In the simplest case, when the Courant number $S_C = 1$, a unidirectional wave moves exactly one spatial step per time step without changing shape. This makes it straightforward to simulate the continuation: one only needs to copy the direct neighbor value from the previous step, for example

$$E_z^{n+1}[0] = E_z^n[1]$$

at the low end of the domain and analogous on the other side, for example

$$H_y^{n+\frac{1}{2}}[N - 1/2] = H_y^{n-\frac{1}{2}}[N - 3/2]$$

if E_z and H_y are the parallel field components at the outer points. To apply the boundary update together with the source functions, after the main grid update, the field value from the previous time step for each boundary cell is stored in a buffer that is specially allocated. Saving previous values will be necessary for the first- and second-order ABCs in any case.

This can only work in a one-dimensional simulation if there is vacuum at the boundary. However, it is exact in this case. It can be selected in the code by setting the ABC order to zero.

A more general first-order ABC is based on an approximation of the first-order wave equation (advection equation)

$$\left(\frac{\partial}{\partial t} \pm c' \frac{\partial}{\partial x}\right) E_z = 0$$

with the material-dependent speed of light $c' = 1/\sqrt{\varepsilon\mu}$. The sign between the differential terms decides the direction of the wave; if it is positive, the wave moves in the positive x-direction and should be used at the high end of the domain, while a negative sign is for the low end. In general, replace x with the boundary surface normal direction and E_z with one of the parallel components of $\mathbf{E}$ or $\mathbf{H}$. To connect boundary points with their neighboring values, we apply centered finite differences in both space and time, and to relate them to the same intermediate space-time point, we can average two consecutive spatial differences in time and average adjacent temporal differences in space. This results in

$$\begin{aligned}&\frac{1}{2}\left(\frac{E_z^{n+1}[0] - E_z^n[0]}{\Delta t} + \frac{E_z^{n+1}[1] - E_z^n[1]}{\Delta t}\right)\\ &= \frac{c'}{2}\left(\frac{E_z^n[1] - E_z^n[0]}{\Delta x} + \frac{E_z^{n+1}[1] - E_z^{n+1}[0]}{\Delta x}\right)\end{aligned}$$

and from this an expression for the new boundary value is obtained:

$$E_z^{n+1}[0] = E_z^n[1] + \frac{c'\Delta t/\Delta x - 1}{c'\Delta t/\Delta x + 1}\left(E_z^{n+1}[1] - E_z^n[0]\right)$$

The simple rule of assigning the direct neighbor value in 1D emerges as a special case when $S_C = c\ \Delta t/\Delta x = 1$ and $c' = c$. It is necessary to store past values for the interior points in a dedicated array. The constant that appears can be expressed with the regular update coefficients k_i from the boundary cells:

$$\frac{c'\Delta t}{\Delta x} = \sqrt{\frac{\Delta t}{\varepsilon\Delta x} \cdot \frac{\Delta t}{\mu\Delta x}} = \sqrt{k_E[0] \cdot k_H[0]}$$

To obtain a second-order ABC, we apply the advection equation's differential operator twice,

$$\left(\frac{\partial}{\partial t} \pm c' \frac{\partial}{\partial x}\right)^2 E_z = 0$$

with the idea that a near-solution of the first-order equation should be even closer to a solution of the second-order equation. Applying the discrete equivalent of the differential operator twice, a larger equation relating points that are now separated by two steps in space and time is obtained. For the details see [15]. Five values from past time steps are needed now, while in practice six are stored to simplify the array indexing, including the

boundary point's current value. The final result is, setting $S'_C = c'\Delta t/\Delta x$,

$$\begin{aligned} E_z^{n+1}[0] = &\frac{1}{1/S'_C + 2 + S'_C}\{-(1/S'_C - 2 + S'_C)\left(E_z^{n+1}[2] + E_z^{n-1}[0]\right) \\ &- 2(S'_C - 1/S'_C)\left(E_z^{n}[0] + E_z^{n}[2] - E_z^{n+1}[1] - E_z^{n-1}[1]\right) \\ &+ 4(1/S'_C + S'_C)E_z^{n}[1]\} - E_z^{n-1}[2]. \end{aligned}$$

Note that this absorbing boundary only works properly if there are no input sources within the boundary region where the values for solving the differential equation are taken, and the material in this region should be homogeneous (e.g., vacuum). The boundary surfaces are also assumed to be flat (i.e. a rectangular domain).

Better and more generally applicable analytical ABCs are available in the literature [13].

4.8 Wave sources

Introducing electromagnetic energy into the grid can be achieved in several ways, either by placing a source inside the domain to obtain the near field, or using techniques to simulate the far-field response, for example a planar wave. Depending on which field components (electric or magnetic) are excited, the resulting signal can be a transversal wave with different polarizations.

With FDTD being a time-domain method, the shape of the excitation signal must also be chosen. Typically this is either a short pulse (or a series of them) or a continuously varying signal like a sinusoidal or other periodic excitation, or an arbitrary function of time.

4.8.1 Source functions

A Gaussian pulse or bell curve is a typical first choice for obtaining a smooth continuous curve usable as an excitation signal that is effectively limited to a configurable interval thanks to its fast exponential decay in either direction, starting from its peak value. It can thus be assumed to be non-zero in only a bounded interval. Its shape is symmetric around its single peak and it can be interpreted as a wave packet, comprised of the many frequencies in its spectrum.

An expression of a Gauss curve is

$$f_{Gauss}(t) = e^{-\left(\frac{t-d}{w/2}\right)^2}$$

with the width w and delay d as selectable parameters.

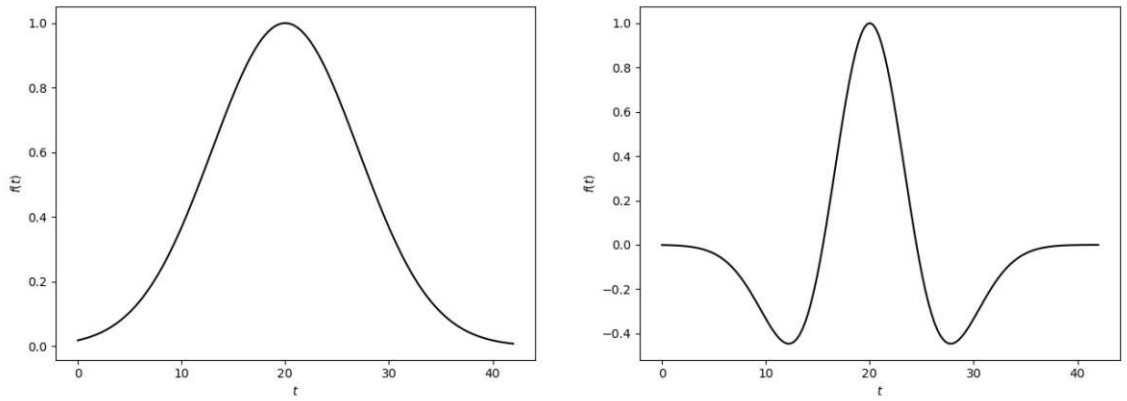
This type of curve was chosen initially during development, but when combining it with an absorbing boundary condition, it was found to result in numerical artifacts and it was not being absorbed at a desirable level. This is likely related to the fact that the main component (highest energy) is a zero-frequency (DC) component, which is, in general, not suitable for simulating traveling waves with FDTD and is known to have the potential to cause unphysical artifacts in the grid [15].

As an alternative, the Ricker wavelet was eventually chosen, following [15], and it did not exhibit the problems that were apparent with the Gaussian signal. This function is related to the second derivative of the Gauss curve and its shape is also symmetric around a central peak. However, it passes through zero, becoming negative on both sides of the main peak, but then also approaching zero at an exponential rate, effectively confining the curve to a limited interval.

One formulation of a Ricker wavelet is

$$f_{Ricker}(t) = \left(1 - 2\left(\pi \frac{t-d}{w}\right)^2\right) \; e^{-\left(\pi \frac{t-d}{w}\right)^2}$$

where, again, a width w and delay d can be selected as parameters. These parameters were chosen so that the same w and d in a Gauss curve result in a somewhat comparable shape, as shown in Figure 13. The width is actually the inverse of the peak frequency $f_P = 1/w$, which is the parameter that determines the frequency spectrum. The Ricker wavelet has no DC component.



(a) Gauss curve

(b) Ricker wavelet

Figure 13: Comparison of the Gauss and Ricker wavelet function graphs, both with a width $w = 20$ and a delay $d = 20$.

4.8.2 Point sources

The simplest possible source, applied at a single point, assigns a fixed value to one field component according to an arbitrary function of time in each step. This is called a *hard source* [13], shown in Figure 14.

Since a hard source overwrites the field value, this cell cannot react to changes in the neighboring cells and it therefore does not let an incoming signal pass through. Because of superposition, a hard source acts as a combination of a perfect conductor for incoming waves (zero internal field, fully reflective) and a perfect source for outgoing signals that it produces. A zero-value source can in fact be used to simulate a perfect conductor causing full reflections, by resetting the value to zero in each update step. The reflective property of a source is sometimes desirable, if the source object in the physical problem that is being modeled has similar reflective properties. However, it is more likely that the reflections are an undesirable side-effect and measures must be taken if it has a noticeable impact. For

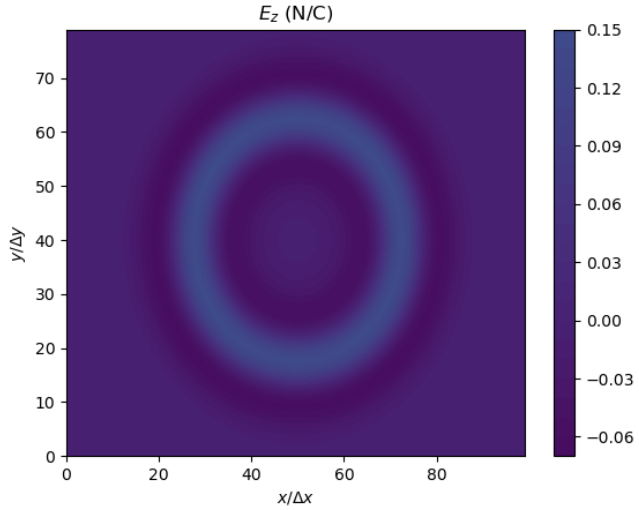


Figure 14: Snapshot of the E_z component of a wave originating from a point source with a Ricker wavelet excitation. The wave expands radially and the signal strength decays with the inverse distance from the source point. The peak source amplitude was normalized to 1.

a point source in two or three dimensions or a line of hard sources in three dimensions, the unwanted reflections may be negligible and can be ignored, depending on the setup. Otherwise, in the case of a narrow enough pulsed waveform, one can simply switch off the source as soon as the excitation signal has been fully transmitted and then replace it with a regular grid node. This is implemented by storing the point sources in a dynamic array, attaching an optional time limit and comparing the current time progress with the limit during the update steps, removing all expired sources. Care must be taken to ensure sufficient distance to other nearby sources or reflective surfaces to avoid retro-reflections while the pulse source is still active.

An additive source is another type of point source that originates from a discretized realization of the current source term $\mathbf{J}$ (or $\mathbf{J_m}$) in Maxwell's equations. Here, the field correction that is applied after the regular point update does not overwrite and discard the existing value, but instead it adds to it the value from the input source function, which is again a function of time. This avoids the problem with reflections and other signals can freely pass through it. However, the source node reacts to the outgoing signal and thus it results in a somewhat different pattern that can be more difficult to predict. An interior additive source causes a weakened response compared to a hard source.

Multiple point sources can be combined to form a continuous line of sources or a planar source in three dimensions. If all points in a line or plane region are synchronized, using the same source function, the created wave approximates an ideal plane wave in a space with the corresponding number of dimensions. A finite planar source region confined to the domain will not create a proper plane wave, because the boundary effects due to it being cut off cause it to be rounded off at the borders.

To model an ideal infinite planar wave, a different type of source condition is applied, which is described next.

4.8.3 Total-field/scattered-field source

An ideal plane wave has planar wave fronts of constant phase that extend infinitely. In a bounded and discretized domain, a finite cut-out of such a plane wave could possibly be simulated with an array of point sources spanning an entire domain wall, if certain measures are taken at the lateral boundaries. However, this is problematic and likely conflicts with an absorbing boundary condition. A simple planar array of sources contained inside the domain, which leaves some space necessary for boundary conditions, creates an imperfect plane wave that is rounded off at the sides.

The *total-field/scattered-field* (TF/SF) source solves this problem, modeling a section of an ideal plane wave, but confining it to a smaller region within the domain [13, 15].

This is realized by defining a special closed rectangular boundary inside the domain, which acts both as a source of the input signal and as a separation between the inner *total-field* region and the outer *scattered-field* region. The total field includes the effects of both the incoming plane wave and any modifications (scattering) it is subjected to inside, like in a regular simulation, but as if an actual infinite plane wave was incident; the scattered field is only non-zero if a scatterer is present in the inner region, so that it represents only the deviation from the ideal plane wave. If the plane wave travels through the total-field region undisturbed, it is (almost) perfectly "absorbed" at the opposite end, fully confining it to the inner domain section. Such an undisturbed wave is shown in Figure 15.

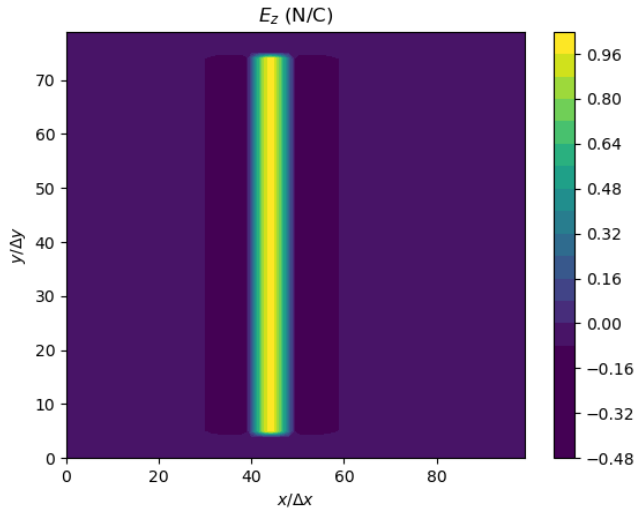


Figure 15: The E_z component of a plane wave moving in x-direction with components E_z and H_y and a Ricker wavelet excitation. It is confined to a slightly smaller region within the domain.

The boundary points participate in the regular update step like the rest of the domain, so that they still respond to the surrounding field values and other waves can freely pass

through, similar to an additive source. In addition to the main update, a correction is applied to boundary nodes. Nodes on the inner side of the boundary receive a virtual signal located at neighbor nodes on the opposite side, which introduces the source wave that then propagates into the total-field region. Nodes outside receive a corresponding virtual signal that is exactly opposite to the source wave, countering their neighbors' response signal due to a time difference. This ensures that in the absence of a scattering object, the outgoing wave from the total-field region is perfectly canceled, resulting in a zero field in the scattered-field region. However, if the wave inside is disturbed, the virtual and actual fields no longer match, and a response reaches the outer region, the scattered field.

To simplify matters, we restrict the wave to move in direction of a grid axis and choose the polarization so that only one transverse electric and magnetic component is excited; for example, a wave in x-direction with non-zero E_z and B_y. Generalizations are possible.

The boundary locations are chosen so that each boundary rectangle cuts between electric and magnetic components and the adjacent grid planes (one aligned with grid cells, the other with half-cells) have matching pairs of transverse electric and magnetic components across the boundary. The inner field components are always electric and the matching outer components always magnetic (other choices would be possible). Figure 16 shows the boundary node pairs in a TM(z) layer and Figure 17 for a TE(z) layer. Because the planar cuts at the higher ends of the region are each done behind a layer of tangential electric components, the remaining normal electric component of each of these layers of cells is not part of the inner region and it must be excluded when iterating the rectangular surfaces, shrinking each rectangle (in 3D) in direction of the relevant electric component by one cell. See Figure 17.

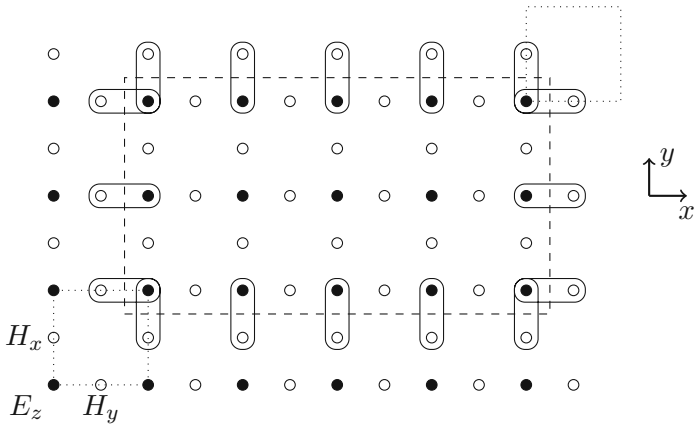


Figure 16: TF/SF boundary in a TM(z) plane, indicated by dashed lines. The dotted squares mark the first and last cell in the grid. Rounded boxes highlight pairs of matched boundary nodes. All electric components within the dashed rectangle belong to the total-field region. 5×3 cells have electric components in the inner region.

At every point along the boundary it essentially acts like a pair of additive sources, simulating the incoming source signal with a time delay depending on the point location, and set up in such a way that the undisturbed plane wave does not leak out of the inner region, canceling on the outer side due to an opposite sign. The implementation, however,

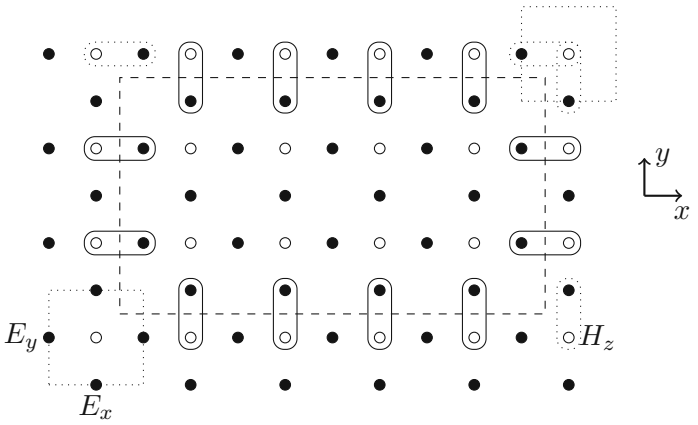


Figure 17: TF/SF boundary in a TE(z) plane. Even though 5×3 cells have electric components in the inner region (except for the top-right corner cell), only their tangential components are included in the cells along the top (E_x) and right boundary (E_y). When iterating the components normal to these boundaries (e.g., E_y at the top) along an adjacent boundary (e.g., left or right boundary), the last update pairs (highlighted with dotted outlines) are therefore excluded from the TF/SF correction.

works somewhat differently.

An auxiliary 1D simulation performed alongside the main simulation can provide the input signal, where the field value at each point in the 1D grid represents the wave amplitude in a plane of constant phase in the main grid. It thus serves as a lookup table, updated in every time step, which can be more efficient than computing the field values by evaluating an analytical source function for each point, which may become expensive. Another advantage is that a 1D FDTD simulation with the same space to time step ratio also has the same dispersion characteristics as the wave moving along a coordinate axis. The update correction on each side is applied by reading the value that should be present on the other side according to the 1D simulation (which acts as a virtual hard source) and simulating a one-sided regular update step for the transversal component on this side, with reversed sign on the outer side. On the boundary side that is normal to the direction of the other side's wave component, the correction step can be skipped, because there is no matching component (only transverse pairs).

To simplify the setup, the same default x-directed TM(z) simulation in 1D can always be used and the values interpreted in the main grid by rotating the 1D grid and adjusting the sign depending on the main wave components, also accounting for the opposite half-step offset in the magnetic component if the wave direction is negative.

Reusing the notation from Section 4.4, we can write the one-sided update equations similar to equations (4.7) and (4.8). If the plane wave has transverse components $(E_{src})_j$ and $(H_{src})_k$ with coordinate indices j and k, then along each side of the boundary, oriented normal to axis I, if I is not the direction of the relevant component on the opposing side ($(E_{src})_j$ inside, $(H_{src})_k$ outside), the virtual plane wave component present on the other

side demands the update corrections of the main fields

$$\bar{H}_K^n[\mathbf{i} - \mathbf{e}_I] := \bar{H}_K^n[\mathbf{i} - \mathbf{e}_I] + s_{IKj} \cdot k_{HE}[\mathbf{i}] \cdot (\bar{E}_{src})_j^n[\mathbf{i}] \tag{4.10}$$

$$\bar{E}_J^{n+1}[\mathbf{i}] := \bar{E}_J^{n+1}[\mathbf{i}] - s_{IJk} \cdot k_{EH}[\mathbf{i}] \cdot (\bar{H}_{src})_k^n[\mathbf{i} - \mathbf{e}_I] \tag{4.11}$$

at the low ends of the TF/SF boundary and

$$\bar{H}_K^n[\mathbf{i}] := \bar{H}_K^n[\mathbf{i}] - s_{IKj} \cdot k_{HE}[\mathbf{i}] \cdot (\bar{E}_{src})_j^n[\mathbf{i}] \tag{4.12}$$

$$\bar{E}_J^{n+1}[\mathbf{i}] := \bar{E}_J^{n+1}[\mathbf{i}] + s_{IJk} \cdot k_{EH}[\mathbf{i}] \cdot (\bar{H}_{src})_k^n[\mathbf{i}] \tag{4.13}$$

at the high ends of the boundary, overwriting the previous values. Here, the direction J is orthogonal to both I and k (the remaining index in the set $\{1,2,3\} - \{I,k\}$) and K is orthogonal to I and j. The cell index $\mathbf{i}$ is that of the cell containing the $\mathbf{E}$-component on the inner side. The signs for the difference operators are $s_{ijk} = \mathbf{e}_j \cdot (\mathbf{e}_i \times \mathbf{e}_k)$ and the sign for the scattered-field correction was integrated directly.

These formulas are derived from the update equations (4.7) and (4.8) for special selections of non-zero components with the effect of one of the neighbor values left out for each difference operator.

Referring to Figure 16, if the wave moves in x-direction, the wave components may be E_z and H_y ($j = 3$, $k = 2$), and the update formulas (4.10) and (4.11) correspond to highlighted pairs of nodes along the left ($I = 1$) and bottom ($I = 2$) sides of the boundary, and formulas (4.12) and (4.13) to pairs along the right ($I = 1$) and top ($I = 2$) sides. For vertical lines of pairs ($I = 1$), both components must be updated because j and k are both tangential directions; for horizontal lines ($I = 2$), only H_x is corrected, but not E_z. This is because $I = k$ (the wave component H_y is not tangential to the boundary) and H_x remains zero, even though it must be corrected due to a non-zero E_z.

Note that the TF/SF boundary can only cause a wave in downstream direction (the direction of the source wave), either the main wave moving into the total-field region at the first impact boundary of the incoming wave, or a wave with opposite sign that escapes into the scattered-field region at the other end of the domain, if the wave inside was obstructed. It never creates a wave moving back into the region in the opposite direction. This is because of the half-step time difference, causing the node on the downstream side to receive the input signal before the upstream node has responded to it, which then sees both the neighbor's response and the negative input signal, canceling out each other.

The plane wave can be configured to travel along any of the coordinate axes, both in positive and negative directions. A wave that moves in a direction that is not aligned with a grid axis can also be implemented, but this necessitates some adjustments and interpolation to compensate for the misaligned points between the main grid and the auxiliary one-dimensional grid as well as some considerations for managing dispersion.

The scattering object for a given application is placed entirely within the total-field region and the external region would usually be left as empty space, which is terminated appropriately at the boundary. Detectors may be placed to capture and analyze the scattered field before it is absorbed.

4.9 Material interpolation

Inhomogeneities in the domain, such as an interface between two different materials, can be a source of error in the discretized formulation, resulting in a staircase pattern, and, in addition, the offsets in the placement of individual electric and magnetic field components in each cell (at cell faces or edges) cause a slight shift (or distortion) of the precise region where a material parameter is in effect.

The staircase effect is already inherent in the cell set with its assignment of materials to whole cells and mitigating it would require including additional information from the original surface representation to allow approximating sub-cell features. However, this could also be applied when adding geometric objects to the domain in a postprocessing step, such as thin plates, wires or curved objects like a spherical lens.

The effect of application points being at cell boundaries suggests combining the influence of multiple cells that are in contact with a point. This is currently not implemented, but it could quite easily be introduced by replacing the four update coefficient arrays, two for each of the electric and magnetic field vectors, with two coefficient arrays for each updated field component, and then initializing each one with the interpolated material parameters from multiple neighboring cells. In three dimensions, two adjacent cells always share an interior magnetic node, located at the center of their common surface, while each interior electric node, centered at an edge, is in direct contact with four cells, and thus averaging of material parameters could be performed over these cells. (A theoretical basis for using an arithmetic average for the electric permittivity ε_r is given in [15], Section 7.8.) This would increase memory consumption, while providing some benefits for applications where it is important that a material boundary is not shifted or distorted, or in the case of a continuously varying material over multiple cells. The smoothing of the material transition at the edge cases might also help reduce numerical artifacts and unphysically large reflections caused by a sudden jump in parameters.

Some sub-cell geometry techniques can also make use of the per-component material parameters ([13], Chapter 10).

5 Application: Photodiode

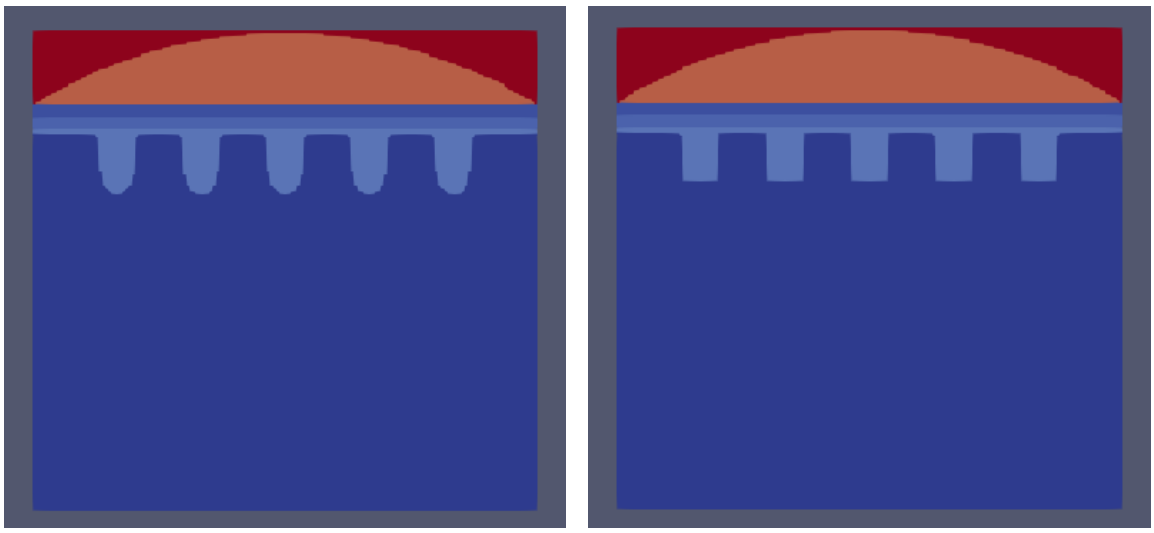
Oshiyama et al. [21] have recently described a back-illuminated CMOS image sensor, which is based on a photodiode with a special surface structure that the authors developed to reduce reflections for the incoming light. Its use case is the capturing of signals in the visible-light spectrum, such as for digital cameras (a single pixel). A photodiode (here assumed to be made of silicon) leverages the photoelectric effect to convert photon energy into conducting electrons in the semiconductor. The performance of an image sensor of this type can be quantified by its *quantum efficiency*, which is defined as the ratio between the number of detected electrons (the measured current) and the number of incoming photons. When light is reflected at the surface, it reduces the overall quantum efficiency of the device. The special surface structure involves making small circular holes into the photodiode at the surface that is exposed to light through an etching process, and it was shown to be more effective for visible light than previously investigated structures.

This work presents a simplified model of this image sensor and performs an FDTD simulation with an incident wave in the appropriate frequency range, using the developments from the previous sections.

The base structure of the image sensor is generated in ViennaPS step by step, starting with a uniform silicon block for the photodiode at the bottom. The holes in the silicon are equally spaced, with the distance and diameter being chosen as parameters. They can be created with the SF_6/O_2 plasma etching model[1], where the hole depth depends on the etching process duration, and a mask must first be applied that defines the locations of the holes. The ratio between the etchant (fluorine) and oxygen fluxes determines the shape of each hole, here chosen to be 1:1. For details and the theoretical background see [22, 23]. Alternatively, the etching process can be disabled, which then generates the holes in the geometry directly by cutting out rectangular (or cylindrical) shapes with a chosen depth, without a physical process. One can then compare the two variants with and without the physical etching. Next, multiple thin layers are added on top for the purpose of passivation and/or antireflection. Here, one passivation layer and two antireflection layers are in use, tightly fitted onto the material below, and for each one the thickness can be varied. Finally, a spherical lens shape of a certain material is added in a postprocessing step before the simulation, and the remaining empty space is assumed to be vacuum or air. The resulting two-dimensional geometric structures for both cases are shown in Figure 18. The original structure in [21] has a size of 1.4 µm, while we use a smaller model here for testing purposes.

The cell size of the underlying level set grid may be chosen suitably to allow resolving all the structural features without making it too small to avoid excessive memory consumption or performance degradation in the etching simulation step. This grid resolution could,

[1] `https://viennatools.github.io/ViennaPS/models/prebuilt/SF6O2Etching.html`



(a) Etched holes (b) Cut-out holes

Figure 18: The geometry for the photodiode example. The semiconductor region at the bottom has holes that are generated (a) using physical etching models or (b) by cutting out boxes using Boolean operations. This surface is covered with a passivation layer (filling the holes), two antireflective layers and a spherical lens on top. The photodiode has a width of 0.42 µm and the hole width is 30 nm.

however, be unsuitable for the FDTD simulation and its demands to ensure accuracy and stability. The appropriate cell size is dependent on the expected (minimum) wavelength, and therefore the grid and the surface domains should ideally be adapted if it is too coarse. This is achieved by converting each level set to an explicit surface mesh representation, which is no longer tied to a fixed-size grid, and then converting back to an implicit level set with a different grid resolution, from which the cell set is generated. While this may lead to loss in certain details such as sharp corners due to the interpolation, the effect was not observed to significantly impact the final simulations. The relevant parameter is the number of cells per wavelength.

The passivation layer that is directly placed on top of the exposed semiconductor surface suppresses the *dark current* by reducing the effect of surface states (dangling bonds), which is important to reduce noise in the electric signal. In this example it also serves a dual purpose as part of an additional antireflection layer.

The passivation layer is deposited on top of the diode structure to cover the flat outline with a thin film, while also filling in the holes with its material, creating a layer of mixed material.

For a photodiode with a flat surface, it is typical to apply an antireflective film on top of the surface to reduce losses for the incoming light. Multiple coatings of this type can also be combined to further reduce the amount of reflections, as is commonly done. Such an antireflective film must satisfy certain conditions to be effective. The refractive index n_1 of the material must have a value between those of the two surrounding media, for example $n_0 < n_1 < n_2$, with $n_0 \approx 1$ for air, n_2 for the next material underneath the layer. The optimal thickness depends on the wavelength and a formula was described in Section 4.2.

The refractive index n (or its real part) is related to the relative dielectric permittivity ε_r by $n = \sqrt{\varepsilon_r}$.

Region	Material	ε_r	n
Air	Vacuum	1.0	1.0
Lens	PDMS	1.96	1.4
Antireflection 1	a-SiN_x:H	2.6	1.6
Antireflection 2	a-SiN_x:H	3.8	1.9
Passivation	SiN	4.0	2.0
Semiconductor	Si	15.8	4.0

Table 1: Material parameters at optical frequencies ($\approx$ 500 nm wavelength, $\approx$ 600 THz frequency) for the different layers in the domain: relative permittivity ε_r and refractive index $n = \sqrt{\varepsilon_r}$.

Table 1 lists the constituent materials with approximate values for their relevant parameters at optical frequencies. All materials are assumed to be non-magnetic ($\mu_r = 1$) and isotropic. These values are representative for the optical range but are not intended to be physically exact, as they are assumed to be non-dispersive for this simulation, while in reality the variation with frequency can be significant. The values for Si, SiN and PDMS were taken from online sources[2], see the references listed there. PDMS refers to polydimethylsiloxane. Amorphous hydrogenated silicon nitrides (labeled a-SiN_x:H), a family of materials that can be created with variable fractions of the elemental compounds, were reported to possess refractive indices in the range from 1.8 to 2.6 and beyond and to be suitable for use in antireflective coatings on solar cells [24].

The refractive indices of the antireflective coatings n_{R1} and n_{R2} and of the mixed silicon-with-passivation layer n_3 and silicon n_{Si} should satisfy the relation $n_{R1} < n_{R2} < n_3 < n_{\mathrm{Si}}$ [21].

The layer made up of silicon with the holes filled with the passivation material is referred to as a *pseudo high refractive index film* (pHRF) by the authors in [21] and its value for n_3 refers to an effective refractive index that lies between that of the two materials, and it can be varied with the hole shapes and layout.

Selecting an appropriate number of grid cells per wavelength is important for FDTD simulations and it must be large enough to ensure accurate results. The vacuum value should be multiplied by the maximum refractive index of the materials in the domain, because the wavelength in matter is scaled by the inverse $1/n$. Silicon has the largest value of $n \approx 4$ in this example.

The type of the wave source used is the total-field/scattered-field source described in Section 4.8.3.

For the excitation signal, a simple value of 1.0 is used as the amplitude, without consideration for the physical meaningfulness of the numerical values. The values are to be interpreted only qualitatively and quantitatively relative to the input signal.

Snapshots of the simulation results, using the etched-hole geometry, can be seen in Figure 19, where the base wavelength of the incoming pulse is chosen to be 450 nm, which

[2] `https://refractiveindex.info/`

falls within the blue range of the visible spectrum. The visual results for the alternative geometry with rectangular holes are nearly identical and therefore not shown.

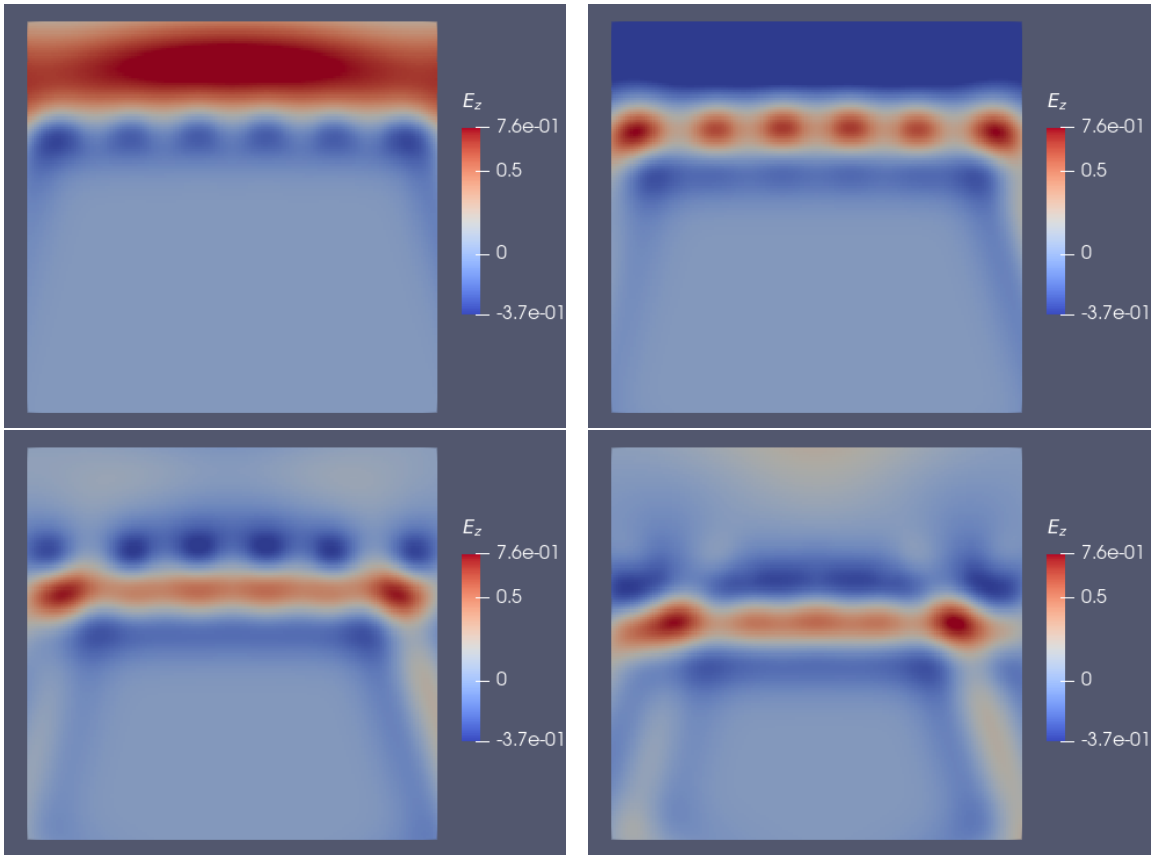


Figure 19: Snapshots of the simulation for an incoming planar wave from the top, showing the z-component of the electric field as it passes through the upper surface layers including the etched holes. The original wave signal had a maximum normalized value of 1.0 (units ignored).

5.1 Transmission calculation

A key metric obtainable from an FDTD simulation of this type is the fraction of the electromagnetic energy that reaches the photodiode region, after loss due to reflection from the material interfaces and refraction at the given geometric structures and material features. Consequently, the effectiveness of the antireflective structure could be determined, or specifically the quantum efficiency of the entire photodetector can be obtained. This is defined as the ratio of the number of detected electrons to the number of incoming photons.

However, obtaining accurate and meaningful values for transmission is quite challenging with the setup that was chosen and satisfactory results are not really achieved. Nevertheless, multiple tested approaches and the problems associated with them are described in this

section, as well as possible improvements.

Typically, transmission and reflection coefficients are determined for a simulation using frequency-domain information, to obtain the coefficients on a per-frequency basis, which gives an entire spectrum of resulting values. This is meaningful when detailed information about the energy distribution over the different modes is desirable. Also, the concepts of transmission, reflection, absorption and scattering are most naturally defined for a single (monochromatic) plane wave in the continuous world. For this aim, a discrete Fourier transform (DFT) is performed, reading the time-domain field values at the points or the region of interest and summing up the individual contributions to each member in a chosen set of discrete frequencies, taking into account both the frequency and phase shift through the use of complex exponentials. This can be performed after an FDTD simulation run, using the recorded time-domain field values for all time steps, or instead as a *running DFT*, processing the field values as the simulation progresses, saving some storage space [15].

Here, however, a simpler approach is taken, capturing the instantaneous power applied through a surface and multiplying it with the time step Δt, and taking the sum over all steps to obtain the total energy flowing through that surface, combining all frequencies. This method does not require a DFT calculation.

The Poynting vector was introduced as part of the electromagnetic theory in Section 4.2 and is expressed as $\mathbf{P} = \mathbf{E} \times \mathbf{H}$. To get a discrete analog of the power through a surface, we take the values E_i and H_i sharing the same grid cell and interpret them as vectors. The cross product of these vectors is calculated and the power is subsequently obtained by taking the inner product with the surface normal of a surface element with the size of a cell boundary face and sum over the cells in a grid-aligned rectangular surface region. Multiplying by the time step Δt gives us an approximation for the energy flowing through the surface in one step.

As it is implemented, the Poynting vector is formed by combining the electric and magnetic field components from the staggered grid, and due to the different base locations of the individual components it is not quite clear how to correctly interpret this cross product and what is the extent of the error when using this approach. Interpolation could be used to localize the operands at a single point and thus possibly increase the accuracy, but it is not obvious where to place the new base point. When choosing the cell center, the nearest neighbor points include four electric and two magnetic contributions for each coordinate direction. Using a corner of the cell or the center of a face would reduce the number of points to interpolate, but it introduces an asymmetry. To be exact, the discretized magnetic field $\mathbf{H}$ would also have to be averaged in time to relate it to the shifted instants at which $\mathbf{E}$ is defined. Furthermore, the linear interpolation introduces an additional source of error, so it is not clear if it will actually be an improvement. Therefore, the uninterpolated components are used.

A more correct calculation uses individually interpolated components of $\mathbf{E}$ and $\mathbf{H}$, localized and combined at different points of the elementary cell and also taking into account the offset in time, reproducing a derivation of the continuous Poynting vector theorem in the discrete setting, as explained in [25]. This improvement was not implemented, but it serves as a suggestion for future work.

Finally, we can arrive at an estimate for the fraction of the transmitted energy through the layered surface by running the simulation first without the scattering structure (setting all the material parameters to the values of vacuum, i.e. $\varepsilon_r = \mu_r = 1$) to obtain a baseline energy transmission for the undisturbed plane wave, and then comparing this with the measurement from an actual simulation run with the materials present, taking the ratio of the energy values. The rectangular detector surface is placed below the upper surface layers so that it lies parallel and spans the inner region.

While testing the energy transmission calculation for different setups, the following problems were encountered. The measured fraction of the energy that is transmitted into the photodiode region, compared to the vacuum case, can actually be greater than one and thus the transmittivity calculated from that would exceed the physical maximum of one, at least for some choices of the parameters. The main problem is that the TF/SF source models the shape of an infinite plane wave that extends beyond the impact surface and, as can be observed, the parts of the wave front to the sides of the material structure enter the material region through the sidewalls when the primary wave inside is obstructed or slowed down. This can be understood with Huygens' principle, interpreting each point of the wave front as a small point source and if a neighboring source point is missing, it radially expands in that direction. This is explained further below. The effect is already visible in Figure 19. A first attempt to avoid this unwanted effect is to make the detector surface smaller, with the hope that most of the energy entering the region in this way is no longer captured. However, a new problem then arises due to the lens concentrating the main incoming energy in the middle of the area, which once again leads to a too large value for the transmitted signal. Simplifying the model by disabling the lens could avoid this and still provide useful information about the flat structure's reflectivity. But another problem then emerges: the arbitrary cut-off points for the detector region can have an unpredictable effect, because the holes cause an inhomogeneous scattering pattern, which may distort the captured energy, creating a dependence on the exact detector boundary location and thus unstable results. Moreover, a part of the additional wave from outside enters even at large deviating angles through the radial spread, meaning that a significant portion might still reach the smaller detector surface.

Figure 20 demonstrates this problem: when a plane wave strikes a reflective object, the unobstructed portions of the wave front, passing by the object, appear to bend around its edges. Even though there is no direct line of sight from the direction of the wave origin, the wave spreads into the area in the "shadow" of the reflective object, due to diffraction. Because the light wave in the photodiode example is significantly slowed down in the dielectric material (by a factor $n = \sqrt{\varepsilon_r} \approx 4$ for silicon), the part of the incoming wave outside the material region overtakes the wave inside and similarly enters the internal region at an angle, reaching the detector surface and adding extra energy that would not reach the detector in the vacuum case. Total (internal) reflection can only occur at a dielectric with a smaller refractive index than that of the incident wave, so this does not keep the wave outside if the surrounding material is vacuum or air.

One thing that can still be determined from the measured values, at least qualitatively, is that the antireflective layers did reduce reflections more effectively with the approximate theoretical values described in Section 4.2.

In conclusion, the chosen TF/SF plane wave source, though having many advantageous

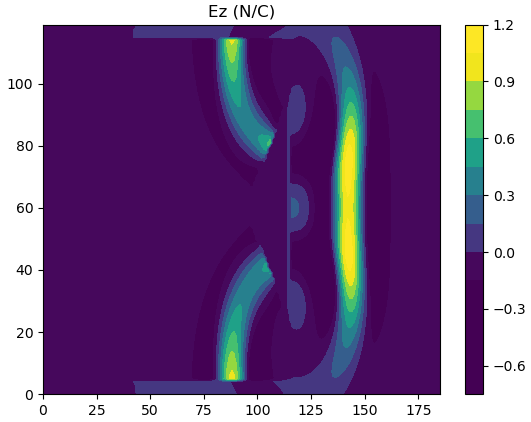


Figure 20: A reflective object and a wave coming in from the right. The central part of the wave is reflected back, but the parts of the wave above and below the reflector send out radial waves that reach the space behind it.

properties and being the right choice for simulating scattering of an ideal plane wave on a self-contained and isolated single object (surrounded by air), created difficulties for this setup, as the application did not align with the ideal use case for this type of source. In an array of photodiode cells, the individual sensors should be more or less isolated from each other and a significant signal would only be intruding through the top side.

Possible solutions to the problems described include adding additional neighboring diodes close-by (or other structures of a similar shape) that are somehow isolated from each other and taking the measurements only for an internal unit, or instead simulating a periodic boundary in some way. Such a setup would also more closely resemble the real use case of a photodiode as one element in a packed array of many such diodes in an image sensor. This was, however, not pursued further here, as it is not the main goal of this thesis to obtain accurate simulation results that would be suitable for making predictions for real physical systems.

A more suitable remedy could be to use a different type of source that is not modeling an infinite plane wave, such as a single point source that is located some distance away from the diode structure, or an array of point sources that would create an approximate plane wave that is rounded off at the edges and thus it would likely reduce — but probably not eliminate — the effect of incoming wave energy entering through the lateral boundaries.

Another possibility would be to measure the amount of reflected energy from the surface instead of the transmitted energy, which probably would not suffer so much from the problem explained above, as most of the signal moving in the direction back to the wave origin should come only from surface reflections; however, this method comes with its own set of challenges, requiring the capturing of energy reflected at an angle and deciding what fraction of the incoming wave energy to take as the reference value, and this was not attempted here.

6 Conclusion

This thesis explored numerical solution methods based on finite differences on structured grids to simulate the evolution in time of different physical processes relevant for microelectronics, where the aim is to accurately predict the material structures and behaviors of semiconductor devices. To begin with, an overview was provided of some important technologies available for this purpose, which are enabled by modern methods for the mathematical representation of geometries. This includes a surface representation through the level set method, based on a sparse data structure, and the dense volumetric representation in the cell set structure, which are together provided by the ViennaPS framework. Two physical problems served as exemplary models: a diffusion process, which forms part of some semiconductor fabrication steps, and the evolution of electromagnetic fields and their propagation as waves, simulated in a postprocessing step as an example of device simulation.

Utilizing the cell structure in combination with the geometric structures generated from the surface formulation, the first application of a diffusion simulation demonstrated the general procedure and highlighted the stability concerns that must be dealt with to obtain reliable results. It furthermore investigated the physical and mathematical nature behind the requirements for numerical stability that necessitates an application-specific treatment, such as for diffusion and wave propagation.

The existing functionality provided by the framework was then extended with a new C++ library for using the well-known finite-difference time-domain method with an application of visible-light wave propagation in a photodiode model.

A limitation of the presented applications is the fact that numerical values were not directly matched with a physical interpretation and they only serve to visualize the behavior of each model to evaluate if it matches the expected behavior. Further refinement of the model parameters would be necessary for more practical applications.

A particular limitation of the photodiode model was explained and some suggestions for resolving it to obtain more meaningful metrics for the transmittivity or reflectivity were provided.

Once the issues are resolved, iterative device optimization workflows could be developed in future work using the combined framework, which would demonstrate the power of such a unified TCAD tool.

More generally, possible improvements and extensions of the presented methods in various directions can be pointed out.

Future work on the diffusion implementation could improve the usability and expand the range of applicability by introducing a generalized diffusion solver module that supports multiple finite-difference solution schemes and it could also be made to support advection, including the additional stability considerations necessary depending on the maximum advection speed.

For the cell-set data structure itself, one interesting idea would be to modify it to not be a dense voxelization of a rectangular region bounding the surfaces in the level set domain, but instead to keep it in a sparse form, still using the HRLE data structure to only include a certain amount of cells between and around the surfaces.

For the FDTD implementation, much remains to be done to turn it into a fully featured and practical analysis tool for all the common use cases of semiconductor device modeling. Firstly, realizing proper parallelization with domain decomposition integrated into the algorithm will be essential for efficiently supporting more complex three-dimensional simulations. Secondly, it would be beneficial to perform numerical validation of the simulation results against existing well-tested FDTD implementations provided elsewhere or against analytical solutions of simple model problems, doing quantitative comparisons with some chosen metrics to confirm correctness. With a similar motivation, it would be useful to improve the tools for measuring the run time and doing more detailed performance testing.

It would also be useful to create additional facilities to ease the placement of wave sources and material structures, as well as to add custom detector regions that automatically record and print computational results, possibly integrated over time. A related useful tool would be a discrete Fourier transform mechanism that calculates the frequency contributions as the time-domain simulation progresses, as discussed previously.

More sophisticated techniques can be developed, with maybe the most important one being the perfectly matched layer technique, which counts as the state-of-the-art in grid termination, to replace the less effective analytical absorbing boundary condition, since it performs better in most practical cases. Interpolation of material parameters at interfaces was mentioned as a possibility to improve the accuracy of the method. Other methods that can improve the accuracy are, firstly, numerical dispersion compensation techniques and, secondly, local sub-cell resolution techniques and related methods that help reduce the errors associated with the staircase approximations of non-grid-aligned surfaces and features. The reduction of the staircase effects can significantly improve the accuracy in some cases (for example with spherical objects), while dispersion compensation may be less impactful ([13] Section 16.11.5). Support for dispersive materials would also be useful, as well as the near-to-far-field transformation technique.

As an extension of the method in a different direction, it is explained and demonstrated in [15], Chapter 12, how Yee's scheme can be adapted to simulate acoustic signals, with a mention of the possibility to further generalize the method to simulate other physical wave propagation problems such as elastodynamics. Even though the types of waves that are the subjects of these other physical theories can be of a different nature, like the sound waves of acoustics being longitudinal (scalar) waves, they can be treated analogously in some respects. For acoustics, the quantities to be solved for are the mean particle velocity and the pressure (alternatively mass density), modeling a compressible fluid with the usual linearization assumptions (small deviations from static pressure and small velocities). Velocity, a vectorial quantity, is located at cell edges, the pressure, a scalar, at cell corners. Both are also assigned to discrete time points that are, again, shifted by a half-step relative to each other. The differential operators would then be the gradient and divergence, which relate the values at points or edges to a change in the respective other quantity. Much of the theory of wave propagation carries over from the electromagnetic case, including the hyperbolic nature of the problem and its stability considerations. The speed of sound takes

the place of the speed of light and an analogous impedance can be defined.

It is conceivable that the implementation could be modified so that the parts that are specific to the electromagnetics formulation are extracted into a separate module (or class) with the necessary interface and inserted into the main simulation class as a C++ template parameter. The analogous data types and functionality (including discretized differential operators, tensorial physical field types and update equations) would then be created in new classes, satisfying the general interface. This would allow it to accommodate different physical field pairs that can be chosen by simply switching out template parameters. The task of identifying which parts of the code should be abstracted out and which parts can be reused by building on those abstractions in a clean way may, however, prove to be challenging.

In conclusion, this work can hopefully provide a useful introduction to finite difference methods on the cell set while favoring intuitive explanations, give pointers to some helpful literature and serve as a starting point for future investigations with the newly developed finite-difference time-domain library.

Bibliography

[1] Otmar Ertl. "Numerical Methods for Topography Simulation". PhD thesis. Technische Universität Wien, 2010.

[2] Clemens Heitzinger. "Simulation and inverse modeling of semiconductor manufacturing processes". PhD thesis. Technische Universität Wien, 2002. DOI: https://doi.org/10.34726/hss.2002.03696962.

[3] Felix Strasser. *FiDiTi - FDTD Library*. Commit ab3eec218e2c8e. URL: https://github.com/exilief/FiDiTi (visited on Aug. 4, 2025).

[4] Tobias Reiter et al. *ViennaCS*. Commit ebff0bae7c3a4b. URL: https://github.com/ViennaTools/ViennaCS (visited on Aug. 1, 2025).

[5] Tobias Reiter, Xaver Klemenschits, Lado Filipovic, et al. *ViennaPS - Vienna Process Simulation Library*. Commit 341ee28360a123. URL: https://github.com/ViennaTools/ViennaPS (visited on Aug. 4, 2025).

[6] Lloyd N. Trefethen. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*. unpublished text. 1996. URL: http://people.maths.ox.ac.uk/trefethen/pdetext.html (visited on Apr. 13, 2025).

[7] Klaus A. Hoffmann and Steve T. Chiang. *Computational Fluid Dynamics*. 4th. Vol. Volume 1. Engineering Education System, 2000.

[8] Andreas Hössinger. "Simulation of ion implantation for ULSI technology". Available at https://www.iue.tuwien.ac.at/phd/hoessinger/. PhD thesis. Technische Universität Wien, 2000.

[9] Helmut Puchner. "Advanced Process Modeling for VLSI Technology". Available at https://www.iue.tuwien.ac.at/phd/puchner/. PhD thesis. Technische Universität Wien, 1996.

[10] Parviz Moin. *Fundamentals of Engineering Numerical Analysis*. 2nd ed. Cambridge University Press, 2010.

[11] R. Courant, K. Friedrichs, and H. Lewy. "Über die partiellen Differenzengleichungen der mathematischen Physik". In: *Mathematische Annalen* 100 (Jan. 1928), pp. 32–74. DOI: https://doi.org/10.1007/BF01448839.

[12] Thomas Rylander, Pär Ingelström, and Anders Bondeson. *Computational Electromagnetics*. Second Edition. Springer, 2013. DOI: 10.1007/978-1-4614-5351-2.

[13] Allen Taflove and Susan C. Hagness. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*. Third Edition. Artech House, 2005.

[14] Ari Stern, Yiying Tong, Mathieu Desbrun, and Jerrold E. Marsden. "Geometric Computational Electrodynamics with Variational Integrators and Discrete Differential Forms". In: *Geometry, Mechanics, and Dynamics*. Online version: https://doi.org/10.48550/arXiv.0707.4470. Springer New York, 2015, pp. 437–475. DOI: 10.1007/978-1-4939-2441-7_19.

[15] John B. Schneider. *Understanding the Finite-Difference Time-Domain Method*. (Version: August 25, 2023). 2010. URL: www.eecs.wsu.edu/~schneidj/ufdtd (visited on Aug. 20, 2024).

[16] Robert E. Collin. *Foundations for Microwave Engineering*. Second Edition. John Wiley & Sons, 2001.

[17] John David Jackson. *Classical Electrodynamics*. 3rd ed. John Wiley & Sons, 1999.

[18] Safa O. Kasap. *Optoelectronics & Photonics: Principles & Practices*. 2nd ed. Pearson Education, 2013.

[19] Kane Yee. "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations in Isotropic Media". In: *IEEE Transactions on Antennas and Propagation* 14.3 (1966), pp. 302–307. DOI: 10.1109/TAP.1966.1138693.

[20] Enzo Tonti. *The Mathematical Structure of Classical and Relativistic Physics: A General Classification Diagram*. Springer New York, 2013. DOI: https://doi.org/10.1007/978-1-4614-7422-7.

[21] I. Oshiyama et al. "Visible Light Sensitivity Enhancement of CMOS Image Sensor with Pseudo High Refractive Index Film Integrated by Directed Self-Assembly Process". In: *2021 5th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*. 2021, pp. 1–3. DOI: 10.1109/EDTM50988.2021.9420898.

[22] Josip Bobinac et al. "Effect of Mask Geometry Variation on Plasma Etching Profiles". In: *Micromachines* 14.3 (2023). DOI: 10.3390/mi14030665.

[23] Rodolfo Jun Belen et al. "Feature-scale model of Si etching in SF6O2 plasma and comparison with experiments". In: *Journal of Vacuum Science & Technology A* 23.5 (2005), pp. 1430–1439. DOI: 10.1116/1.2013317.

[24] Janusz Jaglarz, Maria Jurzecka-Szymacha, Stanisława Kluska, and Katarzyna Tkacz-Śmiech. "Thermo-optical properties of high-refractive-index plasma-deposited hydrogenated amorphous silicon-rich nitride films on glass". In: *Optical Materials Express* 10.11 (2020), p. 2749. DOI: 10.1364/OME.396150.

[25] Jan De Moerloose and Daniel De Zutter. "Poynting's theorem for the finite-difference–time-domain method". In: *Microwave and Optical Technology Letters* 8.5 (1995), pp. 257–260. DOI: 10.1002/mop.4650080512.