INFO 0939 Project 2025

Last updated: 16/09/2025

Goals of the Project

- Solve a physical problem modelled by partial differential equations using a finite difference scheme coded in C.
- Parallelize the code on distributed memory systems using MPI, and on shared memory systems using OpenMP.
- Test the acceleration potential of GPUs using CUDA.
- Learn to profile the code, run weak and strong scalability analyses, and understand the performance of the algorithm on each hardware.
- Explore one or more advanced topic(s) based on your own interests.

Project statement

The propagation of electromagnetic waves is governed by Maxwell's equations, which describe the behavior of electromagnetic fields. Without sources, they read as follows:

$$\frac{\partial \mathbf{B}}{\partial t} + \mathbf{curl}\,\mathbf{E} = 0, \tag{1}$$

$$\frac{\partial \mathbf{D}}{\partial t} - \mathbf{curl}\,\mathbf{H} = 0, \tag{2}$$

where, in three dimensions and assuming a Cartesian coordinate system and MKS units, $\mathbf{B} = \mathbf{B}(x,y,z,t)$ is the magnetic flux density (in T), $\mathbf{E} = \mathbf{E}(x,y,z,t)$ is the electric field (in V/m), $\mathbf{D} = \mathbf{D}(x,y,z,t)$ is the electric displacement field (in C/m^2) and $\mathbf{H} = \mathbf{H}(x,y,z,t)$ is the magnetic field (in A/m). These four vector fields are function of the position (x,y,z) and of time t. In linear materials the following constitutive laws link these fields:

$$\mathbf{B} = \mu\mathbf{H}, \tag{3}$$

$$\mathbf{D} = \varepsilon\mathbf{E}, \tag{4}$$

where the constants μ and ε denote the magnetic permeability and the dielectric permittivity, respectively. The speed of waves is $c = \frac{1}{\sqrt{\varepsilon\mu}}$. In vacuum, $\mu = \mu_0 = 4\pi \cdot 10^{-7}$ H/m and $\varepsilon_0 \approx 8.854 \cdot 10^{-12}$ F/m, which leads to a speed $c_0 = \frac{1}{\sqrt{\varepsilon_0\mu_0}} \approx 3 \cdot 10^8$ m/s.

The numerical scheme used to solve the problem is the Finite Difference Time Domain (FDTD) technique, first proposed by Kane S. Yee in 1966 [1, 2]. This scheme relies on the discretization of the electric and magnetic fields in space and time, on staggered grids.

In a 2D setting where the electric field has a single nonzero component along z and the magnetic field lies in the (x,y) plane (the so-called "transverse magnetic" polarization, TM^z), the discrete unknowns reduce to the x- and $y-$ components H_x and H_y of $\mathbf{H}$, and the z-component E_z of $\mathbf{E}$. Assuming

a grid spacing of Δx and Δy along x and y and a time step value of Δt, and denoting by $F^n[i,j]$ the (approximation of) field F at time $n\Delta t$ and position $(i\Delta x, j\Delta y)$, the FDTD scheme writes (see Figure 1):

$$\begin{aligned}
H_x^{n+1/2}[i,j+1/2] &= H_x^{n-1/2}[i,j+1/2] + \frac{\Delta t}{\mu \Delta y}\left(E_z^n[i,j+1] - E_z^n[i,j]\right) \\
H_y^{n+1/2}[i+1/2,j] &= H_y^{n-1/2}[i+1/2,j] + \frac{\Delta t}{\mu \Delta x}\left(E_z^n[i+1,j] - E_z^n[i,j]\right) \\
E_z^{n+1}[i,j] &= E_z^n[i,j] + \frac{\Delta t}{\varepsilon \Delta x}\left(H_y^{n+1/2}[i+1/2,j] - H_y^{n+1/2}[i-1/2,j]\right) - \\
&\qquad \frac{\Delta t}{\varepsilon \Delta y}\left(H_x^{n+1/2}[i,j+1/2] - H_x^{n+1/2}[i,j-1/2]\right)
\end{aligned}$$

for $n = 0, 1, \ldots$ The scheme is initialized with an homogeneous initial condition on all fields, i.e. $H_x^{-1/2}[i,j+1/2] = H_y^{-1/2}[i+1/2 = E_z^0[i,j] = 0$. Several boundary conditions can be considered. To simulate a perfectly conducting boundary a homogeneous Dirichlet condition is imposed on the electric field, i.e. $E_z^n[i,j] = 0$ on the boundary. For the first and second update equations one then considers $i = 0, \ldots, N_x - 1$ and $j = 0, \ldots, N_y - 2$, and $i = 0, \ldots, N_x - 2$ and $j = 0, \ldots, N_y - 1$, respectively; and for the third equation one considers $i = 1, \ldots, N_x - 2$ and $j = 1, \ldots, N_y - 2$.

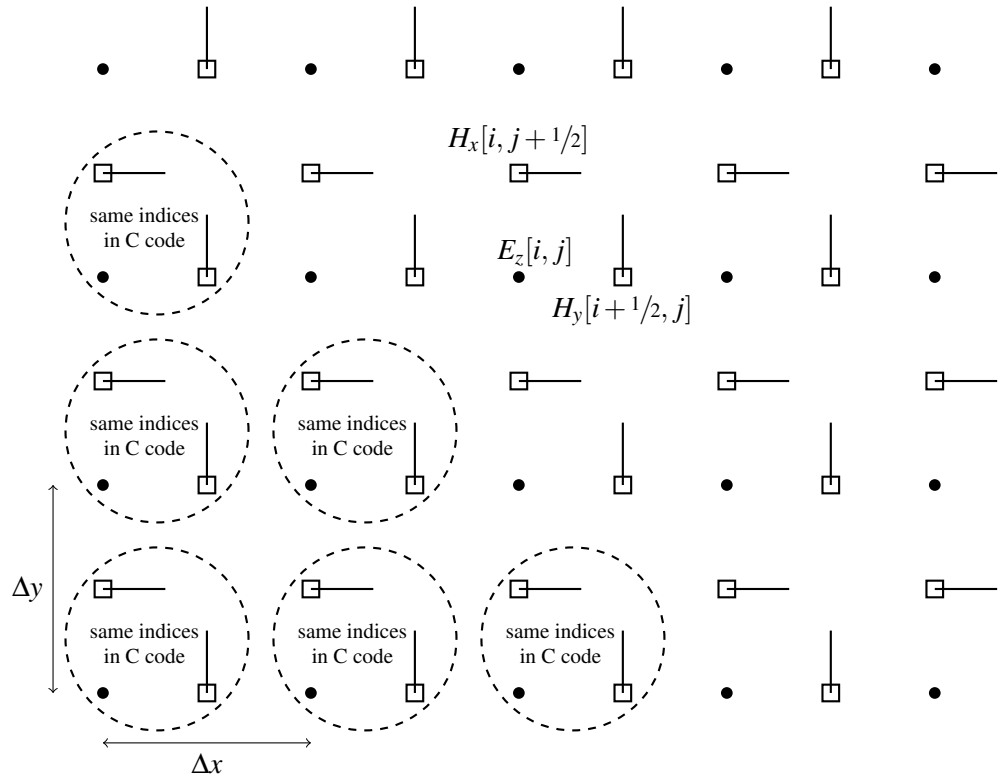

Figure 1: Schematic representation of the staggered FDTD discretization for TM^z polarization; adapted from [2].

Instructions

A serial C code that implements the FDTD scheme for TM^z polarization is available on `https://people.montefiore.uliege.be/geuzaine/INFO0939/project/`. The program takes a single integer argument, selecting the problem to solve, and creates output files that can be read with Paraview (`https://paraview.org`).

In a first project phase (deadline: 25/11/2025) you are asked to:

1. Experimentally study the stability of the FDTD scheme, depending on the values of the parameters.
2. Perform an analysis of the arithmetic intensity of the numerical method and determine the main limiting factor for the speed of the algorithm: is the code memory bound or CPU bound? What do you expect if you run the code on a machine capable of 200 GB/s of memory bandwidth and 2.8 TFLOPS/s of processing power?
3. Evaluate potential bottlenecks in the serial code, based on the lecture on CPU cache hierarchies.
4. Parallelize the serial code using MPI, by subdividing the domain into $P_x \times P_y$ partitions, where the positive integers P_x and P_y designate the number of partitions along x and y, respectively, and are chosen to minimize the amount of MPI communications.
5. Parallelize the serial code using OpenMP. Could you advantageously make use of the `collapse` clause?
6. Perform a scalability analysis of the MPI and the OpenMP codes, by evaluating both the strong and the weak scaling. Choose appropriate parameter values for these scaling runs, deactivating outputs to disk. Make sure to reserve appropriate resources on the clusters (e.g. full nodes), and perform measurements multiple times to reduce uncertainty. Explain the results of the scalability study using the concepts and the tools presented during the lectures. Do you observe NUMA effects?
7. Port the code to run in single precision on a single GPU of the Lyra cluster, using CUDA. Compare and analyze the performance of the CPU and GPU codes.

In a second phase (deadline: 19/12/2025), you are asked to explore **at least one** of the following topics:

1. Extend the physical model by adding dissipative (conductivity) effects and either an absorbing boundary condition or a perfectly matched layer to simulate a transparent boundary.
2. Run and analyze large scale simulations of physical relevance, by adapting the code to handle a spatially varying permittivity.
3. Extend the code to perform 3D simulations.

General remarks on C coding style

Your coding style will be evaluated. As such, some quality in the submitted code is expected, and you should strive to have a clean and neat coding style. In general, think about who will read your code and ask yourself if it is clear and understandable. Please beware of some common mistakes that will lead to penalties on your final grade:

- If you `malloc()` something, remember to `free()` it.
- Check return values for failure, especially from memory allocation, file handling and MPI functions. Don't assume that those calls will always succeed.
- Don't use Variable Length Arrays, i.e.

  ```
  int function(int N) {
      int array[N];
  }
  ```

  The value of `N` must be known at compile time. If it is not the case, use `malloc()`.
- Don't abuse comments. If something is obvious, don't comment it. If you feel the need to comment some code, ask yourself if perhaps you can write the code in a clearer way instead of putting that comment.
- Try to use relevant and appropriate variable and function names. Be coherent in naming things, in order to make it easy to track down where data goes.
- Whenever possible avoid global variables: they are one of the factors that make it harder to make your code thread safe.
- Limit the length of your functions: if a function does not fit on your screen, perhaps you should split it into smaller pieces. Also, if you have a function that takes more than 10 parameters, perhaps it is time create a `struct`?

References

[1] Kane Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on antennas and propagation*, 14(3):302–307, 1966.

[2] John B Schneider. Understanding the finite-difference time-domain method. Technical report, School of electrical engineering and computer science, Washington State University, 2025. `https://eecs.wsu.edu/~schneidj/ufdtd/ufdtd.pdf`.