Finite-Difference Time-Domain Method

MATLAB Challenge - Summer Term 2021/22

March 24, 2022

1 Motivation

With the advent of computer technology, analytical approaches to solving technical problems fall into oblivion. Powerful computing machines can now solve problems whose complexity would not have occurred to anyone a hundred years ago, and if they did, they would be considered unsolvable. Today's times offer a great deal of opportunity to solve complex problems. Since the members of Department of Electromagnetic Field create the team responsible for MATLAB course, the goal of the competition project is to implement Finite-Difference Time-Domain method (FDTD), one of the most widespread and intuitive methods for simulating electromagnetic fields in the time domain [1]. The method is a part of plenty of commercial softwares [2, 3, 4] and offers unsuspected possibilities, but it also has disadvantages. Immerse yourself in programming one of the present electromagnetic field solvers, rediscover its secret nooks and crannies and try to humiliate commercial programs.

2 Task

The idea of the challenge project is to implement 2D FDTD and to apply it to wave equation. The implementation requires at least basic knowledge of calculus and numerical mathematics.

FDTD frequently serves as a tool capable of computing propagation of electromagnetic field in various scenarios. In many cases, the description of electromagnetic field propagation can be reduced to wave propagation. The problem is related to the wave equation

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u + f, \tag{1}$$

second-order linear partial differential equation, where c is phase velocity, u is wave amplitude, and f is an arbitrary function of sources. Apart from electromagnetism, the wave equation is an important concept in mathematics and in physics generally, *i.e.* mechanical waves in acoustics and in fluid dynamics.

The dicretized form for evolution in 2D space reads

$$\begin{aligned} u_{i,j}^{k+1} = {} & \Delta_t^2 f_{i,j}^k + 2u_{i,j}^k - u_{i,j}^{k-1} + \\ & + \left(\frac{c\Delta_t}{\Delta_x}\right)^2 \left[u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k\right] \\ & + \left(\frac{c\Delta_t}{\Delta_y}\right)^2 \left[u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k\right], \end{aligned} \tag{2}$$

where $i \in \{1, \ldots, N_\mathrm{x}\}$, $j \in \{1, \ldots, N_\mathrm{y}\}$ are indexes in spatial discretization and $k \in \{1, \ldots, N_\mathrm{t}\}$ in time discretization.

The important points to solution of each differential equation are the boundary condition. In this case the boundary is ideally reflective. Its implementation can be done via ghost cell, focusing on the boundary point $u_{1,j}^k$, the value of ghost cell reads

$$u_{0,j}^k = u_{2,j}^k. \tag{3}$$

For more details, information about implementation see documentation, for inspiration and examples visit YouTube video.

3 Equations Implementation

3.1 Space-time Discretization

At first, we need to discretize time and space into cells. Our simulation still has to obey laws of physics. Therefore, the discretization is not arbitrary. The most used discretization criteria is Courant's condition [5, 1]

$$\Delta_t \leq \frac{1}{c_0\sqrt{\Delta_x^{-2} + \Delta_y^{-2}}}, \tag{4}$$

which will prevent nonphysical wave propagation. We can freely select discretization Δ_x and Δ_y and we get Δ_t from (4).

The condition has to be kept in mind while defining a periodic source, so that the period of source T_s has to obey

$$T_\mathrm{s} > 2\Delta_t, \tag{5}$$

as a consequence of sampling theorem [6].

3.2 Wave Equation

The discrete form of wave equation shown above may still look quite complicated for numerical evaluation. However, in the following MATLAB code snippet, you can see that solving this equation is very straightforward.

```
 t |«j  ne®O}
Tn} c Q oli§
   uuu
   t ªqL9§O _j §_hO
   «rIJlJ¨s Q «rlJlJ©s¡
   «rIJlJ©s Q «rlJlJos¡

   t ªqL9§O _j  q9HO
   Tn} _ Q ©li²]o
     Tn} a Q ©li²]o
        «r_JaJos Q L§;©=Xr_Jas v ©=«r_JaJ©s ] «r_JaJ¨s v uuu
        rL§=H·¤L²s;©=r«r_]oJaJ©s ] ©=«r_JaJ©s v «r_voJaJ©ss v uuu
        rL§=H·¤L´s;©=r«r_Ja]oJ©s ] ©=«r_JaJ©s v «r_JavoJ©ss¡
     OjL
   OjL
   uuu
OjL
```

This is the core of the numerical evaluation of the equation above, and it will be present in some form in your code. This particular implementation can still be improved in many directions.

3.3 Boundary Condition

Figure 1 shows how to handle a presence of perfect electric conductor (PEC) boundary. The yellow discretization point is not present in our structure. However, we need it for the calculation of selected boundary conditions. This is handled by using a point $(i+1, j)$ and mirroring its value to the yellow point, see Figure 1b.

Considering PEC bounding box, the bounding condition is implemented as in the following code snippet.

```
t  |«j  ne®O}
Tn} c Q oli§
uuu
  t ?n«jL9}´ HnjL_§_nj
  Tn} _ Q ©li²]o
    «r_JEo OjLFJos Q «r_JE¨ OjL]©FJos¡
    «rEo OjLFJ_Jos Q «rE¨ OjL]©FJ_Jos¡
  OjL
uuu
OjL
```

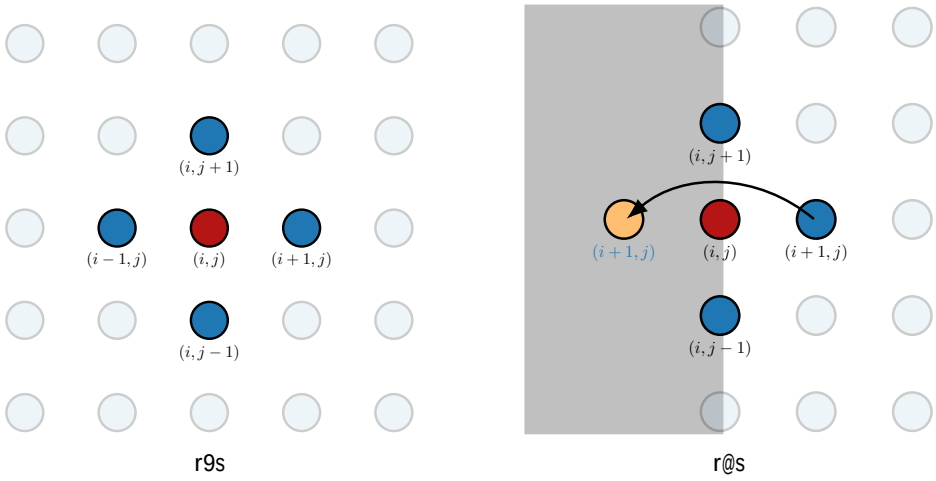

Figure 1: How to handle a boundary. Blue dots are discretization points in space. Red dot represents considered point. Yellow dot is ghost cell. Gray area is a boundary. Space far from boundary (a), space with boundary on the left side (b).

3.4 Source Definition

As we need some sort of field excitation, it is necessary to add it into the code. Considering TE modes, we assume only localized sources at position (x_0, y_0)

- Kronecker's pulse (a.k.a. unit pulse) source is defined in particular place as

$$u(x_0, y_0, t) = \delta(t - t_0), \tag{6}$$

  The following snippet shows how to add excitation into the code.

```
t N²H_§9§_nj LOT_j_§_nj
W Q ¶O}n ri§J os¡ t b}njOHcO}{  q«e O
Wros Q o¡
²W Q Tenn}ri² ¤ ©s v o¡ t _jLO² nT qn _§_nj nT TOOL_jX _j ²
´W Q Tenn}ri² ¤ ©s v o¡ t _jLO² nT qn _§_nj nT TOOL_jX _j ´

t |«j  ne®O}
Tn} c Q oli§
  t N²H_§9§_nj «qL9§O
  Xr²WJ´Ws Q Wrcs¡
```

- Another type of source is harmonic field source with angular frequency ω

$$u(x_0, y_0, t) = \cos \omega t. \tag{7}$$

  Once again, the addition of harmonic excitation is a straightforward task.

```
t N²H_§9§_nj LOT_j_§_nj
T Q o ¤ ro£=L§=q_s¡ t O²H_§9§_nj T}Ox«OjH´
W Q  _jr©=q_=T=L§=r·li§]oss¡ t O²H_§9§_nj  _Xj9e
²W Q Tenn}ri² ¤ ©s v o¡ t _jLO² nT qn _§_nj nT TOOL_jX _j ²
´W Q Tenn}ri² ¤ ©s v o¡ t _jLO² nT qn _§_nj nT TOOL_jX _j ´

t |«j  ne®O}
Tn} c Q oli§
   t N²H_§9§_nj «qL9§O
   Xr²WJ´Ws Q Wrcs¡
```

3.5 Region of Interest

We assume the region bounded by square PEC box (*i.e.* we are asking what is in the box). We are observing the most simple scenario in this example, see Figure 2a. Once we start to excite the field with a given source, we will see the evolution of wave distribution, see Figure 2b.

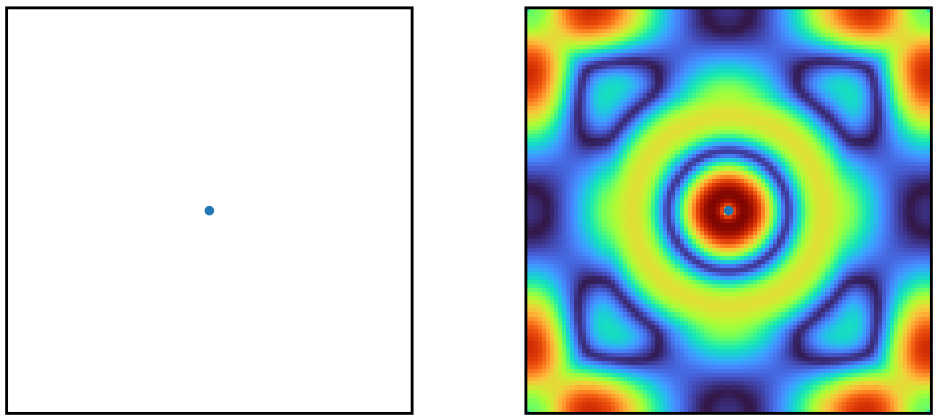

Figure 2: Regions of interest. Blue dot represents feeding point. Interior bounded by PEC (a), wave distribution in time $t = 46.7\,\text{ns}$ (b).

3.6 Example Code

We are providing you with the code of a simple implementation of the FDTD method. You can find many other FDTD codes in the depths of the internet. If you are still interested in participating in the competition, do not worry about asking for further information.

4 Criteria

4.1 Goal

The aim of the project is to:

1. implement the solution to the wave equation by FDTD in two-dimensional space with perfectly reflecting boundary,
2. build-up graphical user interface (GUI) allowing to set dumping coefficient, choose observation points, define perfectly reflecting obstacles and sources,
3. and animate the evolution during computation in every time step.

4.2 General

- An unlimited number of students can select this project. However, no collaboration between students is expected.

- Complete the challenge till **May 15, 2022, 23:59** and submit it via BRUTE.
- Contact `matlab@fel.cvut.cz` with any questions.
- The project should be submitted, including short documentation for each file describing how it works.
- Like for regular projects, a short presentation (a couple of minutes) is expected.
- The resulting project should be in the form of an application with GUI using `uifigure`.
- Use MATLAB version R2020a or newer (necessary for the comparison of normalized time discussed later).
- No external MATLAB toolboxes or third-party libraries are allowed.
- It is possible to always withdraw from the competition and select one of the regular projects. This decision should be discussed with the teachers, and their approval is required.

4.3 Technical

- Implement 2D FDTD in MATLAB based on the equation (2).
- Use discretization space 500×500.
- Build GUI, which allows to:
  1. choose entire domain dumping coefficient,
  2. add/delete perfectly reflecting obstacles and sources (unit pulse or periodic),
  3. set number of iterations N_t,
  4. measure evaluation time t_eval,
  5. pick points to display the whole evolution over time.
- Animate the evolution during computation.
- There are no limits to fantasy.

4.4 Evaluation

- Competition criteria is normalized time per computation iteration.
  1. In GUI, measure evaluation time t_eval,
  2. display normalized time in command line
  $$t_\mathrm{norm} = \frac{t_\mathrm{eval}}{K t_\mathrm{bench}}, \tag{8}$$
  where t_bench is evaluated as `tBench = sum(mean(bench(3)));` immediately after simulation,
  3. the normalized time will be credited.
- The fastest[1] project will be awarded.

5 List of Awards

By participating in competition you will automatically be rewarded with some of the unique MATLAB merchandise, see Figure 3.

The three best solutions will also be rewarded with Humusoft vouchers for MATLAB courses, where you can learn some of the advanced functionalities of MATLAB language, such as Simulink, parallel computations or embedded coding.

[1] When matching normalized times to 2 digits, the appearance and functionality of the GUI decide.