3. The Finite-Difference Time-Domain Method (FDTD)

The Finite-Difference Time-Domain method (FDTD) is today's one of the most popular technique for the solution of electromagnetic problems. It has been successfully applied to an extremely wide variety of problems, such as scattering from metal objects and dielectrics, antennas, microstrip circuits, and electromagnetic absorption in the human body exposed to radiation. The main reason of the success of the FDTD method resides in the fact that the method itself is extremely simple, even for programming a three-dimensional code. The technique was first proposed by K. Yee, and then improved by others in the early 70s.

Theory

The theory on the basis of the FDTD method is simple. To solve an electromagnetic problem, the idea is to simply discretize, both in time and space, the Maxwell's equations with central difference approximations. The originality of the idea of Yee resides in the allocation in space of the electric and magnetic field components, and the marching in time for the evolution of the procedure. To better understand the theory of the method, we will start considering a simple one-dimensional problem. Assume, at this stage, "free space" as propagation medium. In this case, Maxwell's equations can be written as

$$\frac{\partial \mathbf{E}}{\partial t} = \frac{1}{\varepsilon_0} \nabla \times \mathbf{H} \tag{1}$$

$$\frac{\partial \mathbf{H}}{\partial t} = -\frac{1}{\mu_0} \nabla \times \mathbf{E} \quad . \tag{2}$$

In the one-dimensional case, we can use only E_x and H_y, and (1), (2) can be rewritten as

$$\frac{\partial E_x}{\partial t} = -\frac{1}{\varepsilon_0} \frac{\partial H_y}{\partial z} \tag{3}$$

$$\frac{\partial H_y}{\partial t} = -\frac{1}{\mu_0} \frac{\partial E_x}{\partial z} \tag{4}$$

that represents a plane wave traveling in the z direction.

Yee's scheme consists in considering Ex and Hy shifted in space by half a cell and in time by half a time step when considering a central difference approximation of the derivatives. In such a case, equations (3) and (4) can be written as

$$\frac{E_x^{n+1/2}(k) - E_x^{n-1/2}(k)}{\Delta t} = -\frac{1}{\varepsilon_0}\frac{H_y^n(k+1/2) - H_y^n(k-1/2)}{\Delta z} \tag{5}$$

$$\frac{H_y^{n+1}(k+1/2) - H_y^n(k+1/2)}{\Delta t} = -\frac{1}{\mu_0}\frac{E_x^{n+1/2}(k+1) - E_x^{n+1/2}(k)}{\Delta z} \quad . \tag{6}$$

Equations (5) and (6) show the usefulness of Yee's scheme in order to have a central difference approximation for the derivatives. In particular, the left term in equation (5) says that the derivative of the E field at time $n\Delta t$ can be expressed as a central difference using E field values at times $(n+1/2)\Delta t$ and $(n-1/2)\Delta t$. The right term in equations (5) approximates instead the derivative of the H field at point $k\Delta x$ as a central difference using H field values at points $(k+1/2)\Delta x$ and $(k-1/2)\Delta x$. This approximations oblige us to calculate always the E field values at points …, $(k-1)\Delta x$, $k\Delta x$, $(k+1)\Delta x$, … and times …, $(n-3/2)\Delta t$, $(n-1/2)\Delta t$, $(n+1/2)\Delta t$, … and to calculate always the H field values at points …, $(k-3/2)\Delta x$, $(k-1/2)\Delta x$, $(k+1/2)\Delta x$, … and times …, $(n-1)\Delta t$, $n\Delta t$, $(n+1)\Delta t$, …

This scheme is known as "leap-frog" algorithm. Practically, it means that to approximate Maxwell's equations in space and time using this algorithm, one should calculate first all H field values, then all E field values, remembering always that E and H are shifted also in space by half of the discretization Δx. Figure 1 shows schematically the algorithm.

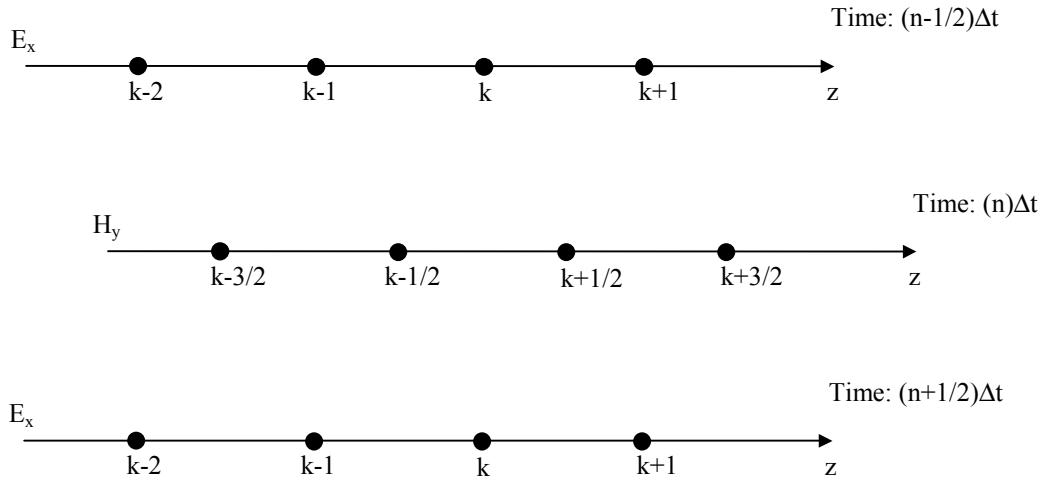

Figure 1. Yee's one-dimensional scheme for updating EM fields in space and time.

The explicit FDTD equations can be derived from (5) and (6) obtaining

$$E_x^{n+1/2}(k) = E_x^{n-1/2}(k) + \frac{\Delta t}{\varepsilon_0 \Delta z} \left(H_y^n(k - 1/2) - H_y^n(k + 1/2)\right) \tag{7}$$

$$H_y^{n+1}(k + 1/2) = H_y^n(k + 1/2) + \frac{\Delta t}{\mu_0 \Delta z} \left(E_x^{n+1/2}(k) - E_x^{n+1/2}(k + 1)\right) \quad . \tag{8}$$

To avoid computational problems due to the very different amplitudes of E and H, Taflove introduced a normalization of the E field:

$$\tilde{E} = \sqrt{\frac{\varepsilon_0}{\mu_0}}\, E \quad . \tag{9}$$

Equations (7) and (8), adopting this substitution and dropping the symbol "~" from now on, become

$$E_x^{n+1/2}(k) = E_x^{n-1/2}(k) + \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\Delta t}{\Delta z} \left(H_y^n(k - 1/2) - H_y^n(k + 1/2)\right) \tag{10}$$

$$H_y^{n+1}(k + 1/2) = H_y^n(k + 1/2) + \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\Delta t}{\Delta z} \left(E_x^{n+1/2}(k) - E_x^{n+1/2}(k + 1)\right) . \tag{11}$$

These equations can be directly implemented in a computer code. Note that the "1/2" in equations (10) and (11) do not need to be implemented in the computer code. The half a cell and half a time step are necessary in equations (10) and (11) just to remind us the physical definitions of E and H, and to remind us that E and H are actually offset by half a cell and half a time step. This information, anyway, **will never appear in our coding**. To implement the code for the calculation of the fields obeying to equations (10) and (11) we need to:

a) Define the size KE of the arrays E and H that, once we have chosen the spatial resolution Δz, will correspond to the absolute size of the computational domain;

b) Determine the time step necessary according to our resolution and excitation signal;

c) Implement a cycle to compute the fields for a certain number of time steps. Within the cycle, we need to include:

    d) A cycle to calculate the various EX(K) according to equation (10) for all the cells of the domain KE;

e) The excitation signal at the source point KS;

f) A cycle to calculate the various HY(K) according to equation (11) for all the cells of the domain KE; Note that in the computer code we do not need to include the information relative to the half a cell shift (i.e., the "1/2") since this is only the interpretation that we need to give to the field, and does not correspond to any practical modification in the algorithm.

The problem is now: how to choose Δz and Δt. How fine should our resolution be? The first parameter to choose is generally the resolution Dz. It has been verified that at least 10 cells per wavelength are necessary to ensure an adequate representation. The wavelength to consider is the smallest wavelength in the simulation. We did not consider yet the presence in our simulations of dielectrics, but we will shortly see that the presence of dielectric will just slightly modify the FDTD scheme given by equations (10) and (11). Anyway, it is well known that the signal wavelength is smaller in dielectrics with higher dielectric constants and losses. Therefore, in the case we have to consider dielectrics, we should first calculate the wavelength in the dielectric with higher dielectric constant and losses, and then consider the cell size to be about ten times smaller than this value.

Once the cell size has been chosen, the time step is also chosen according to stability considerations. For stability reasons, a field component cannot propagate more than one cell size in the time step Δt. This means that

$$\Delta t \leq \frac{\Delta z}{c_0} \tag{12}$$

since the wave travels at the speed of the light c_0. This is the stability condition for one-dimensional problems. It can be proven that, in general, the stability condition (Courant condition) is given by

$$\Delta t \leq \frac{\Delta}{c_0 \sqrt{d}} \tag{13}$$

with d=1, 2, or 3 for one-, two-, or three-dimensional problems, respectively, and Δ the smallest cell size.

A common choice for Δt (in one-, two-, or thee-dimensional problems) is anyway given by

$$\Delta t = \frac{\Delta}{2\, c_0} \quad . \tag{14}$$

Source Signals

We have mentioned how to choose the cell-size Δz with respect to the minimum signal wavelength in the simulation. This makes sense in the case of **sinusoidal** excitation, but what about other signal waveforms that we may need to handle? If, for example, we have to consider a Gaussian pulse as excitation signal, how should we proceed to determine the cell size? A spectrum of a Gaussian pulse is, again, a Gaussian and, therefore, it contains infinite frequencies. In this case, we should choose the cell size according to the maximum frequency of interest. The remaining part of the spectrum will be filtered by the mesh, incapable of describing the propagation that will require smaller cell size (and time step). One should anyway be careful in choosing well the input signal, in the sense that energy associated with frequencies incapable of propagating should be small with respect to the others. If this is not satisfied, the propagating signal in our simulation may be affected significantly by **noise**, given by the frequencies that cannot propagate.

A wide variety of signals have been used as source in FDTD meshes. The most common are the sinusoidal signal and the Gaussian pulse. Sometimes, when a sinusoidal signal need to be used, it is preferred to use a modulated signal in order to avoid high frequency at the beginning of the simulation. Other choices, in order to reduce the error, consist in using the so called "soft" source, where the source signal of interest is added in the source point to the previous value of the field. In other words, for example, if the following is an "hard" source

```
EX(KS) = SIN (OMEGA* T)
```

the following is instead a "soft" source

```
EX(KS) = EX(KS) + SIN (OMEGA* T)
```

Basic Example of 1D FDTD Code in Matlab

The following is an example of the basic FDTD code implemented in Matlab. The code uses a pulse as excitation signal, and it will display a "movie" of the propagation of the signal in the mesh. If you are not using a workstation, Matlab might have difficulties in handling the movie due to the memory requirements. In this case you should use the second code given, where instead of the movie the code plots the e-field at each time step waiting for you to press a key to plot the next time step.

```
% This is a 1D FDTD simulation with pulse
% It displays a "movie" of the signal
% Size of the FDTD space
clear;
ke=50;

% Position of the source
ks=ke/2;

% Number of time steps
nsteps=100;

% Cell size and time stepping
c0=3.e8;
dx=0.01;
dt=dx/(2.*c0);

% Constants
cc=c0*dt/dx;

% Initialize vectors
ex=zeros(1,ke);
hy=zeros(1,ke);

% Gaussian pulse
t0=20;
spread=8;

% Start loop
M=moviein(nsteps);
for t=1:nsteps

% E field loop
  for k=2:ke-1
    ex(k)=ex(k)+cc*(hy(k-1)-hy(k));
  end

% Source
  ex(ks)=exp(-.5*((t-t0)/spread)^2);

% H field loop
  for k=1:ke-1
    hy(k)=hy(k)+cc*(ex(k)-ex(k+1));
  end

  plot(ex);axis([1 ke -2 2]);
  M(:,t) = getframe  ;
%  input('')
```

```
end
movie(M,1);
```

The following is the code without movie, but with plot at each time step and wait for you to press a key to continue. Of course, you can modify it to plot with the desired number of time steps interval between plots.

```
% This is a 1D FDTD simulation with pulse
% It displays a "movie" of the signal
% Size of the FDTD space
clear;
ke=50;

% Position of the source
ks=ke/2;

% Number of time steps
nsteps=100;

% Cell size and time stepping
c0=3.e8;
dx=0.01;
dt=dx/(2.*c0);

% Constants
cc=c0*dt/dx;

% Initialize vectors
ex=zeros(1,ke);
hy=zeros(1,ke);

% Gaussian pulse
t0=20;
spread=8;

% Start loop
for t=1:nsteps

% E field loop
  for k=2:ke-1
    ex(k)=ex(k)+cc*(hy(k-1)-hy(k));
  end

% Source
  ex(ks)=exp(-.5*((t-t0)/spread)^2);

% H field loop
  for k=1:ke-1
    hy(k)=hy(k)+cc*(ex(k)-ex(k+1));
  end

  plot(ex);axis([1 ke -2 2]);
  input('Type')

end
```

Understanding the Codes

We can now spend a few words on how the codes have been implemented, and the correspondence of equations (10) and (11) with those programmed. We can observe that, basically, we have just "dropped" the 1/2 in the position of the field H. The correspondence between physical and computer variables is indicated in the following figure.

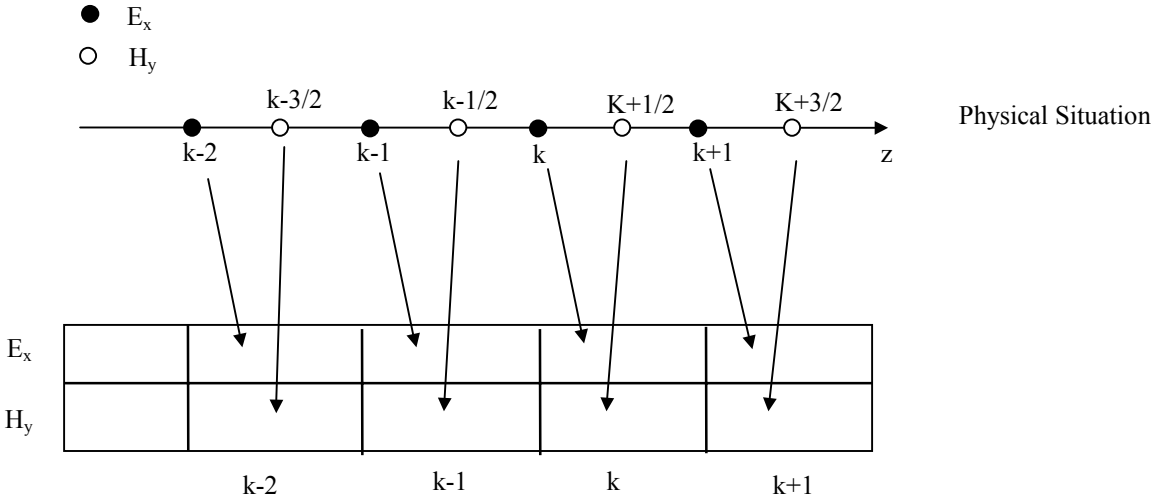

Figure 2. Correspondence between "physical arrays" and "programmed arrays".

Boundary Conditions

From the previous discussion, it is not clear what happens at the mesh termination. Of course, we cannot simulate the propagation of the signal indefinitely, and we need to terminate somehow the FDTD grid. The problem does not exist in the case of a spatially limited structure, like a waveguide, a resonator, etc., where we need to model a region that "trap" the field inside. In most of the problems, however, we need to simulate open space regions. In these cases, since our simulation region MUST be limited, we need to find a way to "simulate" the open space. These boundary conditions are called RADIATION BOUNDARY CONDITIONS (RBCs) or ABSORBING BOUNDARY CONDITIONS (ABCs).

In 1D (the formulation in eq. (10) and (11)) the problem can be easily solved. We can observe, in fact, that if we use Δt given by (14) ($\Delta t=\Delta z/(2\ c_0)$), since the field travels at the speed of the light c0, in one time step the field will travel only half a cell. This means that to entirely cross one cell, two time steps are necessary. The absorbing boundary condition for the 1D case can be therefore expressed by

$$E_x^{n+1/2}(1) = E_x^{n-2+1/2}(2) \qquad (15)$$

for the left side of the mesh, and by

$$E_x^{n+1/2}(KE) = E_x^{n-2+1/2}(KE - 1) \qquad (16)$$

for the right side of the mesh. With these conditions, in the 1D simulation described in the previous section the wave will be completely "absorbed" by the termination. Of course, "completely" means actually "relatively", since for numerical errors some small reflections from the boundary (noise) will be observed.

This is the simplest approach to derive an absorbing boundary condition. Several ABCs have been proposed in the literature that resort to analytical approximations of the wave equation or use fictitious absorbing materials to physically absorb the outgoing wave. One traditional ABC is that developed by G. Mur. The boundary condition proposed by Mur can be explained by considering the wave equation that the electromagnetic field obey. Considering the 1D case, we can write the following wave equation:

$$\frac{\partial^2 E_x}{\partial z^2} - \frac{1}{c_0^2}\frac{\partial^2 E_x}{\partial t^2} = 0 \qquad . \qquad (17)$$

This equation can be rewritten as

$$\left(\frac{\partial}{\partial z} + \frac{1}{c_o}\frac{\partial}{\partial t}\right)\left(\frac{\partial}{\partial z} - \frac{1}{c_o}\frac{\partial}{\partial t}\right) E_x = 0 \quad . \qquad (18)$$

Engquist and Madja have shown that an absorbing boundary condition for the left side of the grid (z=0) can be derived applying the condition

$$\left(\frac{\partial}{\partial z} - \frac{1}{c_o}\frac{\partial}{\partial t}\right) E_x = 0 \qquad (19)$$

while on the right side of the mesh (z=KE) the boundary condition can be derived by using

$$\left(\frac{\partial}{\partial z} + \frac{1}{c_o}\frac{\partial}{\partial t}\right) E_x = 0 \qquad . \qquad (20)$$

The idea of Mur is to express the derivatives around a point that is half a cell distant from the boundary calculated at time n. In this case, we will be able to express the derivatives in equations (19) and (20) by using the electric field values at the points and times known by the simulation. Consider, for example, equation (19) that we should use to derive the boundary condition at the left side of the mesh (z=0). We obtain, using central differences,

$$\frac{1}{\Delta z}\left(E_{x\,2}^{n} - E_{x\,1}^{n}\right) - \frac{1}{c_0 \Delta t}\left(E_{x\,1/2}^{n+1/2} - E_{x\,1/2}^{n-1/2}\right) = 0$$

and, therefore,

$$\frac{1}{\Delta z}\left(\frac{E_{x\,2}^{n+1/2}+E_{x\,2}^{n-1/2}}{2}-\frac{E_{x\,1}^{n+1/2}+E_{x\,1}^{n-1/2}}{2}\right)-\frac{1}{c_0\Delta t}\left(\frac{E_{x\,1}^{n+1/2}+E_{x\,2}^{n+1/2}}{2}-\frac{E_{x\,1}^{n-1/2}+E_{x\,2}^{n-1/2}}{2}\right)=0 \quad .$$

Simplifying, and solving the equation for $E_{x1}^{n+1/2}$, we obtain

$$E_{x\,1}^{n+1/2}=E_{x\,2}^{n-1/2}+\frac{c_0\Delta t-\Delta z}{c_0\Delta t+\Delta z}\left(E_{x\,2}^{n+1/2}-E_{x\,1}^{n-1/2}\right) \quad . \tag{21}$$

Applying the same approximation to the right boundary, we obtain

$$E_{x\,KE}^{n+1/2}=E_{x\,KE-1}^{n-1/2}+\frac{c_0\Delta t-\Delta z}{c_0\Delta t+\Delta z}\left(E_{x\,KE-1}^{n+1/2}-E_{x\,KE}^{n-1/2}\right) \quad . \tag{22}$$

Equations (21) and (22) can be applied directly in the code to update the left and right points in the mesh, respectively.

FDTD in Dielectrics

So far, we have considered the one-dimensional FDTD algorithm in free space. How the presence of a dielectric will affect the algorithm?

We can derive the modification we need to include in the algorithm by considering, again, Maxwell's equations. For a generic medium, Maxwell's equations can be written as

$$\frac{\partial \mathbf{E}}{\partial t}=\frac{1}{\varepsilon_0\varepsilon_r}\nabla\times\mathbf{H}-\frac{\sigma}{\varepsilon_0\varepsilon_r}\mathbf{E} \tag{23}$$

$$\frac{\partial \mathbf{H}}{\partial t}=-\frac{1}{\mu_0}\nabla\times\mathbf{E} \quad . \tag{24}$$

In 1D, these equations can be rewritten as follows:

$$\frac{\partial E_x}{\partial t}=-\frac{1}{\varepsilon_0\varepsilon_r}\frac{\partial H_y}{\partial z}-\frac{\sigma}{\varepsilon_0\varepsilon_r}E_x \tag{25}$$

$$\frac{\partial H_y}{\partial t}=-\frac{1}{\mu_0}\frac{\partial E_x}{\partial z} \quad . \tag{26}$$

Changing the variable Ex according to equation (9), we obtain

$$\frac{\partial E_x}{\partial t} = -\frac{1}{\varepsilon_r \sqrt{\mu_0 \varepsilon_0}} \frac{\partial H_y}{\partial z} - \frac{\sigma}{\varepsilon_0 \varepsilon_r} E_x \qquad (27)$$

$$\frac{\partial H_y}{\partial t} = -\frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\partial E_x}{\partial z} \; . \qquad (28)$$

At this point, we can just discretize equations (27) and (28) in the same we have done it for deriving equations (10) and (11). We obtain, after a few simple manipulations,

$$E_x^{n+1/2}(k) = \frac{1 - \dfrac{\sigma \Delta t}{2 \varepsilon_0 \varepsilon_r}}{1 + \dfrac{\sigma \Delta t}{2 \varepsilon_0 \varepsilon_r}} E_x^{n-1/2}(k) + \frac{\dfrac{1}{\sqrt{\mu_0 \varepsilon_0}} \dfrac{\Delta t}{\varepsilon_r \, \Delta z}}{1 + \dfrac{\sigma \Delta t}{2 \varepsilon_0 \varepsilon_r}} (H_y^n(k - 1/2) - H_y^n(k + 1/2)) \qquad (29)$$

$$H_y^{n+1}(k + 1/2) = H_y^n(k + 1/2) + \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\Delta t}{\Delta z} (E_x^{n+1/2}(k) - E_x^{n+1/2}(k + 1)) \; . \qquad (30)$$

Equation (30) is identical to equation (11), since we did not consider the possibility to have magnetic materials. Magnetic materials can be included easily modify equation (28) first, and the discretizing it as done for deriving equation (28).

Note that equations (29) and (30) are more general than equations (10) and (11). Equations (10) and (11) are obtained by simply imposing ε_r=1 and σ=0 in equations (29) and (30) and, therefore, we can focus on these last two equations for developing the general 1D FDTD algorithm.

Implementation of the 1D FDTD Algorithm for Generic Media.

As we have said before, the superscripts and subscripts containing '1/2' can be simplified in the computer implementation of the algorithm. We should always remember that, anyway, the half a cell and half a time step are "physically" present in the algorithm and, therefore, they should be taken into account in the interpretation of the results.

A computer version of equations (29), (30) is the following:

```
EX(K) = CA(K) * EX(K) + CB(K) * (HY(K-1) - HY(K))          (31)

HY(K) = HY(K) + CC * (EX(K) - EX(K+1))          ,          (32)
```

with

$$CA(K) = \frac{1 - \dfrac{\sigma(K)\,\Delta t}{2\,\varepsilon_0 \varepsilon_r(K)}}{1 + \dfrac{\sigma(K)\,\Delta t}{2\,\varepsilon_0 \varepsilon_r(K)}} \tag{33}$$

$$CB(K) = \frac{\dfrac{1}{\sqrt{\mu_0 \varepsilon_0}} \dfrac{\Delta t}{\varepsilon_r(K)\,\Delta z}}{1 + \dfrac{\sigma(K)\,\Delta t}{2\,\varepsilon_0 \varepsilon_r(K)}} \tag{34}$$

$$CC = \frac{1}{\sqrt{\mu_0 \varepsilon_0}} \frac{\Delta t}{\Delta z} \quad . \tag{35}$$

Each cell will have its own ε_r and σ, that we need to store somewhere, in order to calculate the coefficients CA and CB for each cell of the mesh. The coefficient CC is just a constant, not having any dependence from the dielectrics inside the mesh. A basic skeleton of the code is given in the following.

BASIC SKELETON FOR 1D FDTD CODE FOR GENERIC MEDIA

Define the cell size Δz to use (in meters) (typically, $\Delta z = \lambda_{min}/10$)
Define the time step Δt to use (in seconds) (typically, $\Delta t = \Delta z / 2\,c_0$)
Define how many cells will constitute the mesh (decide KE)
Define how many time steps for the simulation (decide NMAX)
Initialize the vectors CA and CB according to the dielectrics in the mesh
Initialize the variable CC
FOR N=1 : NMAX
    FOR K=2 : KE
        EX(K) = CA(K) * EX(K) + CB(K) * (HY(K-1) - HY(K))
    END
    Apply an electric field source somewhere in the mesh (i.e., sin, pulse, etc.)
    Apply Absorbing Boundary Conditions (if any)
    FOR K=1 : KE-1
        HY(K) = HY(K) + CC * (EX(K) - EX(K+1))
    END
END