Antennas and Electromagnetic Wave Propagation Assignment No.2 – 2022

Prof Costas Constantinou*
Department of Electronic, Electrical & Systems Engineering
University of Birmingham
c.constantinou@bham.ac.uk

Contents

*This assignment was originally devised by Dr Michael Neve of the University of Auckland and has been slightly modified for the purposes of the present module.

1 Introduction

The *Finite-Difference Time-Domain* (FDTD) method is a computational electromagnetic technique for solving for the electric and magnetic fields in arbitrary spatial domains in the time domain. In contrast to techniques such as the *Finite Element Method* (FEM) and the *Method of Moments* (MoM), this technique is straightforward to understand and is simple to program. A rudimentary 2D TM$_z$ code is included in Section §7 and is used to illustrate the main features of the method.

The aim of this assignment is to use the provided FDTD code in a series of numerical investigations, and compare quantitatively its predictions against theory, which the student is expected to research independently after the completion of the taught part of the EMAP module. A formal report is not required, but your assignment report needs to answer all the assignment questions, in a self-contained manner.

2 The basics behind the FDTD algorithm

2.1 Defining the Lattice

The basic FDTD method (in Cartesian coordinates) makes use of a regular lattice of interleaved electric and magnetic field components as originally proposed by Yee [1]. In the case of a 2D TM$_z$ lattice[1], it is possible to derive the following from Maxwell's equations:

$$
\begin{aligned}
\frac{\partial H_x}{\partial t} &= \frac{1}{\mu_0\mu_r}\left[-\frac{\partial E_z}{\partial y} - \sigma^* H_x\right] \\
\frac{\partial H_y}{\partial t} &= \frac{1}{\mu_0\mu_r}\left[\frac{\partial E_z}{\partial x} - \sigma^* H_y\right] \\
\frac{\partial E_z}{\partial t} &= \frac{1}{\epsilon_0\epsilon_r}\left[\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y} - \sigma E_z\right]
\end{aligned}
$$

where σ^* and σ are the magnetic loss (Ω/m) and electric conductivity (S/m) respectively. The Yee algorithm uses second-order central difference approximations to discretise the spatial and temporal partial differentiation operators. If Δx, Δy (m) are the dimensions of a lattice cell in the x and y directions respectively, and Δt (s) is the time step, it is useful to adopt the notation for a field component U (U may be either E or H) given by

$$U(x, y, t) = U(i\Delta x, j\Delta y, n\Delta t) = U|^n_{i,j}.$$

Consider now the (i, j)th TM$_z$ lattice cell, as shown in Fig. 1. Using the above notation, it is possible to form the *update equations* [2, p73ff] for the various field components, given by

$$
\begin{aligned}
E_z|^{n+0.5}_{i-0.5,j+0.5} = {} & C_a|_{i-0.5,j+0.5} E_z|^{n-0.5}_{i-0.5,j+0.5} + \\
& C_b|_{i-0.5,j+0.5}\Big[H_y|^n_{i,j+0.5} - H_y|^n_{i-1,j+0.5} + \\
& H_x|^n_{i-0.5,j} - H_x|^n_{i-0.5,j+1} - J_{\text{source}}|^n_{i-0.5,j+0.5}\Delta\Big]
\end{aligned}
\tag{1}
$$

[1] A TM$_z$ lattice only contains the field components H_x, H_y and E_z. The converse is a TE$_z$ lattice which only contains the field components E_x, E_y and H_z.

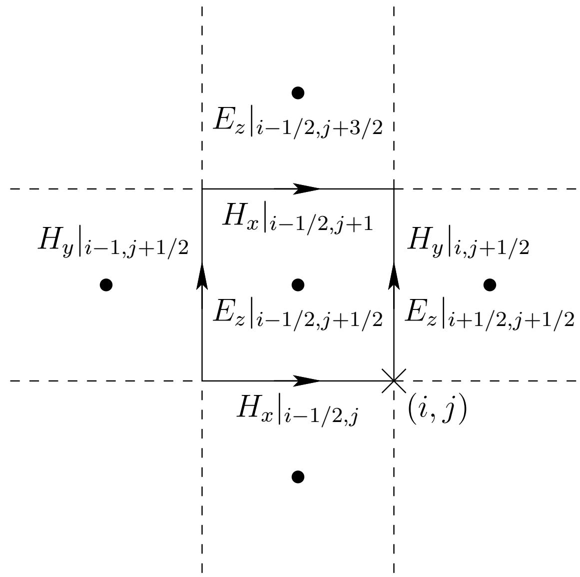

Figure 1: (i,j)th TM_z lattice cell.

$$\begin{aligned} H_x|^{n+1}_{i-0.5,j+1} &= D_a|_{i-0.5,j+1} H_x|^{n}_{i-0.5,j+1} + \\ & D_b|_{i-0.5,j+1}\left[E_z|^{n+0.5}_{i-0.5,j+0.5} - E_z|^{n+0.5}_{i-0.5,j+3/2}\right] \end{aligned} \tag{2}$$

and
$$\begin{aligned} H_y|^{n+1}_{i,j+0.5} &= D_a|_{i,j+0.5} H_y|^{n}_{i,j+0.5} + \\ & D_b|_{i,j+0.5}\left[E_z|^{n+0.5}_{i+0.5,j+0.5} - E_z|^{n+0.5}_{i-0.5,j+0.5}\right]. \end{aligned} \tag{3}$$

The current term J_{source} can be used to excite the lattice. The coefficient matrices $C_a(\cdot)$, $C_b(\cdot)$, $D_a(\cdot)$ and $D_b(\cdot)$ are used to incorporate different materials within the lattice and are are given by

$$\begin{aligned} C_a|_{i,j} &= \frac{\left(1 - \frac{\sigma_{i,j}\Delta t}{2\epsilon_{i,j}}\right)}{\left(1 + \frac{\sigma_{i,j}\Delta t}{2\epsilon_{i,j}}\right)} \\ C_b|_{i,j} &= \frac{\left(\frac{\Delta t}{\epsilon_{i,j}\Delta}\right)}{\left(1 + \frac{\sigma_{i,j}\Delta t}{2\epsilon_{i,j}}\right)} \\ D_a|_{i,j} &= \frac{\left(1 - \frac{\sigma^*_{i,j}\Delta t}{2\mu_{i,j}}\right)}{\left(1 + \frac{\sigma^*_{i,j}\Delta t}{2\mu_{i,j}}\right)} \\ \text{and} \quad D_b|_{i,j} &= \frac{\left(\frac{\Delta t}{\mu_{i,j}\Delta}\right)}{\left(1 + \frac{\sigma^*_{i,j}\Delta t}{2\mu_{i,j}}\right)}. \end{aligned}$$

It should be noted that the indices in the coefficient matrices correspond to the locations of the field components that are being updated. Although appearing cumbersome, these update equations can be programmed in a straightforward fashion. Initially, all field components are initialized to zero, and the field components updated in the order E_z (1) followed by H_x (2) and H_y (3). These calculations are then repeated in sequence until sufficient number of iterations have been performed[2].

2.2 Excitation

It is always necessary to excite the lattice in some fashion, and the specific method by which this is done is problem dependent. One way is to specify the term J_{source} according to a predefined time sequence; alternatively a given field component can be directly specified in a similar fashion (in the TM_z case it is usual to specify a single E_z component). As the simulation progresses, the field will be observed to propagate outwards from the source.

In many cases it is desirable to obtain the response of a system at a fixed frequency[3]. Although it would seem logical to use sinusoidal excitation to determine this, in practice is is usually better to estimate the *impulse response* of the system using a wideband pulse such as a Gaussian derivative pulse [3, p88] given by

$$v_0(t) = -\frac{e^{1/2}}{s}(t-m)e^{-(t-m)^2/(2s^2)} \tag{4}$$

which provides a unit peak amplitude at $t - m = \pm s$. Plotting the impulse response can provide a very good qualitative understanding of how the propagating electromagnetic wave interacts with objects in the problem domain.

If the time-harmonic response of the system is required, this can be straightforwardly determined from the impulse response. For example, if the time-harmonic response at frequency f is required for the E_z component, the time-harmonic electric field $\mathcal{E}_z(f)$ is given by [4, p169]

$$\mathcal{E}_z(f) = \sum_{n=0}^{T} [E_z|^n \cos(2\pi f n\Delta t) - jE_z|^n \sin(2\pi f n\Delta t)] \tag{5}$$

In practice, $\mathcal{E}_z(f)$ can be calculated by maintaining a separate complex field buffer which is incrementally determined by adding the new contribution at each time step.

2.3 Spatial Step Size

In the FDTD method it is necessary to select a spatial step size of approximately $\lambda/20$ in the most electromagnetically 'dense' material in the solution domain (i.e. in the region with the greatest value of ϵ_r).

2.4 Time Step Size

To ensure stability, it is necessary to select a time step size that is less than or equal to the *Courant limit*. In the case of a uniform mesh in 2D with cell size Δ, the Courant limit is given by [3, p70]

$$\Delta t_{\text{Courant}} = \frac{\Delta}{u\sqrt{2}}$$

[2] The number of iterations required is problem dependent. Usually it is necessary to perform sufficient iterations such that any transients have decayed to an acceptable level.

[3] This is often referred to as the *time-harmonic* response.

where u is the speed of light in the most electromagnetically 'dense' material in the lattice. In practice a time step of $\Delta t = 0.95\Delta t_{\text{Courant}}$ is used to ensure any finite precision rounding errors do not cause numerical instability [5, p31].

2.5 Absorbing Boundary Conditions

The FDTD lattice is, by default, terminated on its periphery by a perfect conductor which acts to reflect any outwardly propagating fields (can you figure out why?). However, this is not appropriate for problems which have open boundaries, in which any outwardly propagating fields should be absorbed. In these cases it is necessary to modify the material coefficient matrices and/or the update equations in the vicinity of the boundaries to minimize any reflections. The development of high performance absorbing boundaries has been an area of active research for some time, and boundaries such as the *Uniaxial Perfectly Matched Layer* (UPML) [4, p212] and *Convolutional Perfectly Matched Layer* (CPML) [4, p225] can achieve very high levels of performance. These boundaries can, however, be complex to implement.

A much simpler boundary condition is the *Absorbing Boundary Condition* (ABC) discussed in [3, pp82-83]. In the case of the boundary at $+x$, the new value of a tangential field component $\phi^{n+1}_{N_x}$ is given by

$$\phi^{n+1}_{N_x} = \phi^{n}_{N_x}\left(1 - \frac{c\Delta t}{\Delta x}\right) + \frac{c\Delta t}{\Delta x}\phi^{n}_{N_x-1}$$

i.e., the new field on the boundary is a function only of the old field on the boundary and the field one lattice cell in from the boundary. In the case of the TM_z case being considered here, this condition need only be specified for H_x at the $\pm y$ boundaries and H_y at the $\pm x$ boundaries — the remaining field components are established by the update equations.

3 Examples

A rudimentary FDTD code (`fdtd_1`) has been written in MATLAB and is included in Section §7. Various examples using this code will be investigated in this section.

3.1 `example1` — Propagation in Free Space ($t_{\max} = 10$ ns)

This example is for the code included in Section §7. The source is located at (20,200), and the total simulation time is 10 ns. You must run the code as is and observe the excitation waveform, and the pulse response at 10 ns. Why do you think it is not meaningful to extract the time-harmonic response from this result? Can you observe any unwanted numerical reflections from the ABC?

3.2 `example2` — Propagation in Free Space ($t_{\max} = 50$ ns)

The magnitudes of the fields in Fig. 3 are noticeably smaller than those in the earlier example, as all propagating fields have encountered the ABC on the periphery of the computational domain at least once. However, the residual field is still of appreciable magnitude, and the only way to reduce these is to use a higher performance absorbing boundary such as the UPML or CPML. The time-harmonic response in Fig. 4 shows a dominant cylindrically-propagating wave,

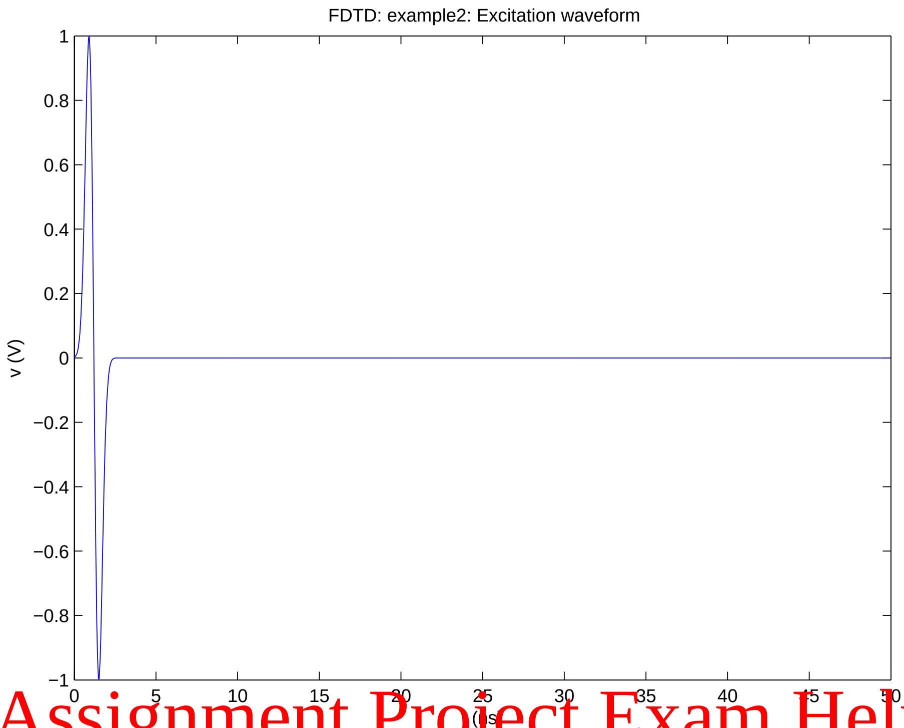

Figure 2: `example2` — Excitation waveform.

although some ripple is present and is due to the presence of reflections from the boundaries of the computational domain.

3.3 `example3` — Propagation in the Presence of a PEC Obstacle

A PEC obstacle has been defined with vertices at (150,100) and (300,250). This is done by including the following definition for `pec_blocks`:

```
pec_blocks = [150 100 300 250];
```

As in `example2`, $t_{\text{max}} = 50$ ns, and the pulse response at 50 ns is plotted in Fig. 5 and the time-harmonic response in Fig. 6 (the excitation waveform is the same as in `example2`). The pulse response in Fig. 5 is somewhat complex, as a result from waves reflecting from and diffracting around the PEC block. The effects of reflection can also be seen in Fig. 6 with the presence of a standing wave between the source and the box, and a significantly reduced field amplitude behind the box as a result of diffraction[4].

4 Assignment

Your report should be in four sections each providing an answer to the following questions. The assessment criteria for each section are, (a) a clear description of what you have done: 10%,

[4] An animation of the time-harmonic fields can also be useful in visualizing these effects. This can be done by enabling the option `want_plot_movie = 1;` in the header, and then executing the command `movie(mov,`n`)` where n is an integer specifying the number of times the movie should be played.

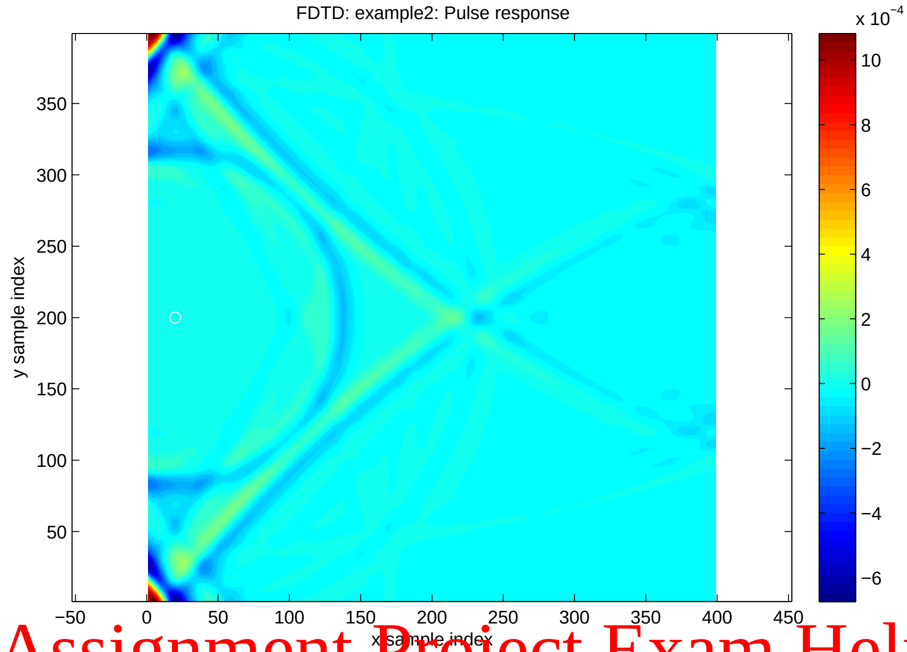

Figure 3: `example2` — Pulse response at $t = 50$ ns.

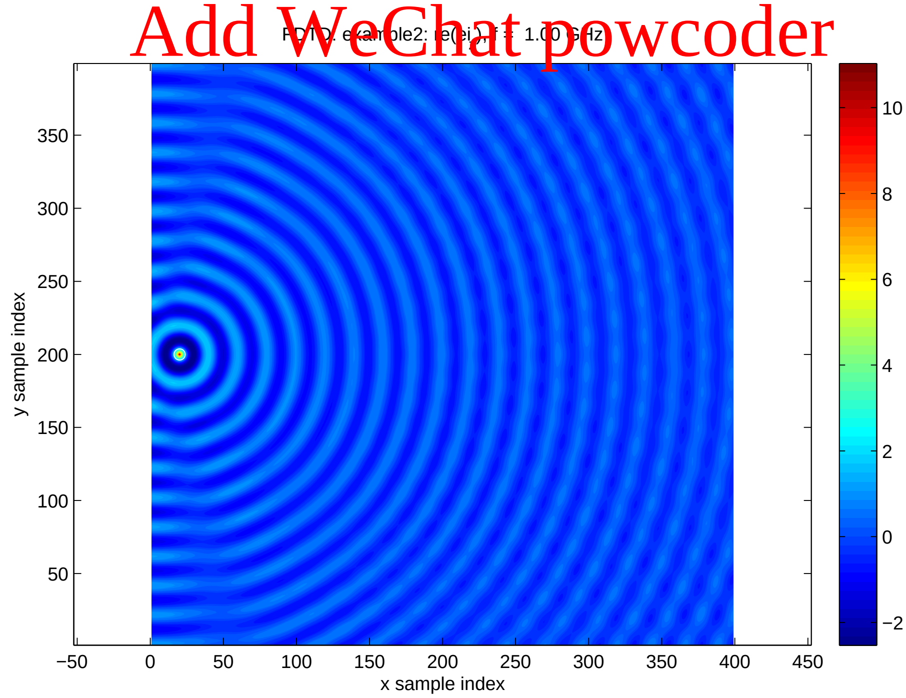

Figure 4: `example2` — Time-harmonic response.

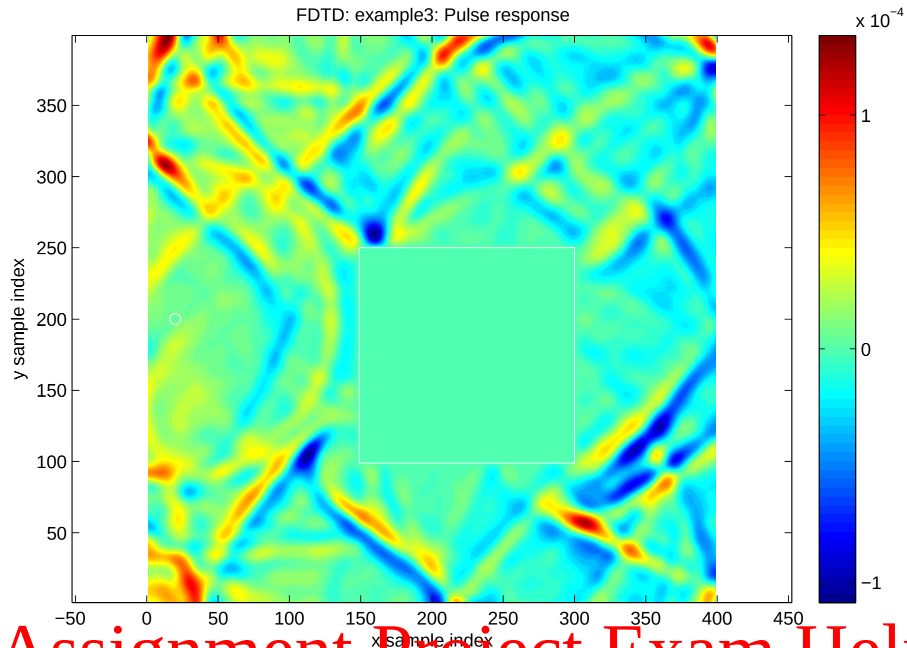

Figure 5: `example3` — Pulse response at $t = 50$ ns.

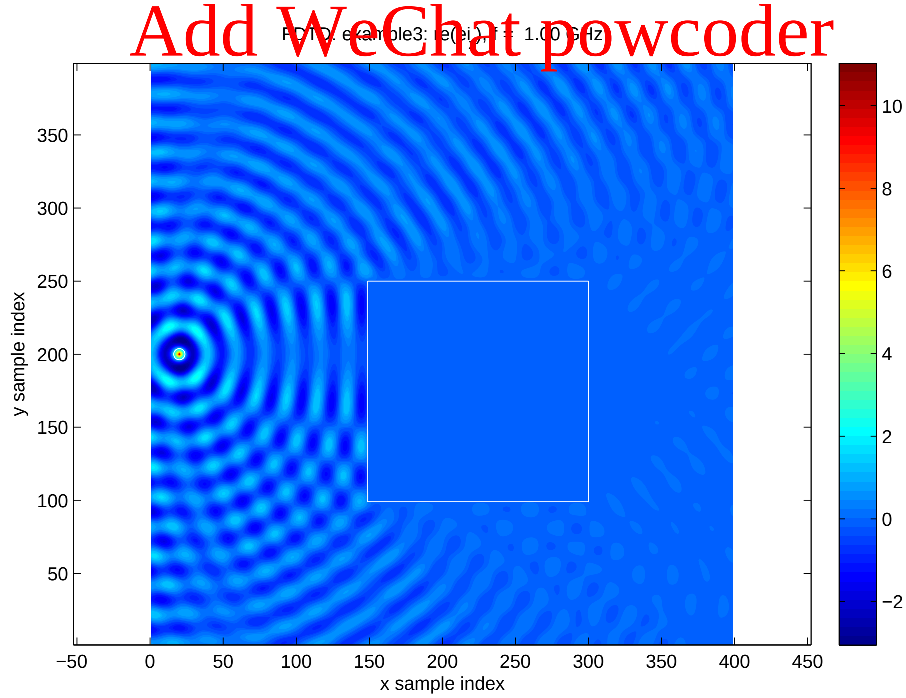

Figure 6: `example3` — Time-harmonic response.

(b) presentation of your simulation results in an appropriate form for interpretation and discussion: 20%, brief summary of relevant theory researched (including citations of key references, but *avoiding* giving an unnecessary tutorial), implementation and calculation of corresponding theoretical predictions: 30%, (c) discussion of numerical and theoretical results: 30%, and (d) drawing conclusions: 10%.

1. Investigate quantitatively the field scattered by a diffracting PEC right-angled wedge in the vicinity of its incidence and reflection shadow boundaries. Ensure that the simulation has converged and examine the field strength in logarithmic units. Is this result as expected according to the Uniform Theory of Diffraction? (*Hint*: You may use readily available UTD MATLAB code, e.g. [6], having investigated the use of the uniform geometric theory of diffraction in the literature). Provide a quantitative justification for your answer.
   (*Hint*: A right-angled wedge can be specified by setting `pec_blocks = [1 1 250 150]`.)
   [40 marks]

2. Investigate quantitatively the behaviour of the field in the shadow region on the PEC right-angled wedge. How does this result contrast with the shadow field distribution from a PEC knife-edge defined by `pec_blocks = [249 1 250 150]`? Discuss and justify the observed similarities or differences. **[20 marks]**

3. Compare quantitatively the field distribution in the shadow region of two cascaded PEC knife-edges furthest away from the source, to the corresponding shadow field distribution of a rectangular PEC block obtained by bridging the knife-edges. The knife-edges are defined using the multiple row format of the PEC blocks command by,
   ```
   pec_blocks = [150 1 151 150
   249 1 250 150];
   ```
   and the block is defined by `pec_blocks = [150 1 250 150]`. Discuss the physical reason for any observed differences. **[20 marks]**

4. When implementing the FDTD method, it is important to select a lattice size that is sufficiently small. Explore the consequences of choosing a lattice size that is too large. (Hint: Make the following change to `example1` in the header: `samples_per_wavelength = 5` to use only 5 samples per wavelength, and to compensate for the difference lattice size move the source to `xs_idx = 20` and `ys_idx = 50`. How does the result differ to that in Section §3.1?) **[20 marks]**

5 Submission details

You need to submit a short report, no more than 8 sides of A4 *excluding* figures, in 11 point Sans Serif font (e.g. Arial), single line spacing and 1.5 cm margins all round. The report should have a cover and feedback sheet which can be downloaded from the module's Canvas page and completed with your student ID number clearly visible on all pages.

Please ensure that all material included from the literature is adequately referenced to avoid any potential plagiarism penalties.

The report should be submitted in Acrobat PDF format, on Canvas, by 12:00 noon on 20 January 2023.

6 Statement of expectations

An excellent report will provide sufficient information to enable the assessor to reproduce its results independently, will have thoroughly researched the state-of-the-art literature for the appropriate theory to compare with numerical simulation results, will produce insightful discussions and will draw scientifically sound conclusions.

A failing report will generate numerical simulation results which cannot be verified independently for corectness, will simply cite relevant literature without justification of its appropriateness, and will limit its discussions to straight-forward observations.

7 Code Listing — fdtd_1

```
% fdtd_1
%
% TMz FDTD Code for EE4D Assignment #2
% Code written by Dr. M. J. Neve
%

clear all;   % Clear all variables from memory

% Problem parameters
id = 'example1';                     % Experiment identification string
f = 1e9;                             % Frequency (Hz)
samples_per_wavelength = 20;         % Samples per wavelength - 20 is good
lattice_size_in_wavelengths = 20;    % Lattice size in wavelengths
tmax = 10e-9;                        % Maximum simulation time (s)
want_abc = 1;                        % Should absorbing boundary condition be used? 1-yes, 0-no
want_plot_excitation = 1;            % Plot excitation waveform?
want_plot_pulse = 1;                 % Plot pulse waveform at final iteration?
want_plot_time_harmonic = 1;         % Plot time harmonic response at frequency f?
want_plot_lattice = 0;               % Plot lattice?
want_plot_geometry = 1;              % Overplot geometry?
want_plot_movie = 0;                 % Plot/create a movie/animation?
ncontours = 100;                     % Number of contours

% Constants
c = 2.997925e8;              % Speed of light in vacuum (m/s)
eps_0 = 8.854e-12;           % Permittivity of free space (F/m)
mu_0 = 4*pi*1e-7;            % Permeability of free space (H/m)
eta_0 = sqrt(mu_0/eps_0);    % Intrinsic impedance of free space (ohms)

% Excitation
s = 3e-10;
m = 4*s;
xs_idx = 20;    % Location of source (Ez field indices)
ys_idx = 200;

% Define PEC blocks
%   This section can be used to define PEC blocks
%   Each row defines a separate PEC block
%   Format is [xmin_idx ymin_idx xmax_idx ymax_idx]
%   Indices are specified relative to locations of the Ez component -
%     if Ez included components on all sides of cell included
%
%pec_blocks = [200 1 201 200];
pec_blocks = [];

% Preliminary calculations
lambda = c / f;    % Wavelength in free space
lx = lattice_size_in_wavelengths * lambda;    % x lattice size in m
ly = lx;    % y lattice size in m
nx = samples_per_wavelength * lattice_size_in_wavelengths; % Number of samples in x direction
ny = nx;
```

```
dx = lx / (nx - 1);    % delta x
dy = ly / (ny - 1);    % delta y
dt = 0.95 * dx / (c * sqrt(2));    % Time step according to Courant limit
nt = ceil(tmax / dt);    % Number of time steps required

% Preliminary calculations are now complete

% Preallocate field storage
ez = zeros(nx - 1, ny - 1);    % TMz field components
hx = zeros(nx - 1, ny);
hy = zeros(nx, ny - 1);
ei_z = zeros(size(ez));   % Time harmonic buffer at frequency f

% Preallocation material constants to free space
ca = ones(size(ez));
cb = ones(size(ez)) * dt / (eps_0 * dx);
dax = ones(size(hx));
day = ones(size(hy));
dbx = ones(size(hx)) * dt / (mu_0 * dx);
dby = ones(size(hy)) * dt / (mu_0 * dx);

% Process any defined PEC blocks
[n_pec_blocks, temp] = size(pec_blocks);    % n_pec_blocks is the number of PEC blocks
for ii = 1:n_pec_blocks
   xmin_idx = pec_blocks(ii,1);
   ymin_idx = pec_blocks(ii,2);
   xmax_idx = pec_blocks(ii,3);
   ymax_idx = pec_blocks(ii,4);
   ca(xmin_idx:xmax_idx,ymin_idx:ymax_idx) = -1.0;
   cb(xmin_idx:xmax_idx,ymin_idx:ymax_idx) = 0.0;
   % dax,dbx,day,dby do not change because magnetic loss of PEC is still 0
end

% Define excitation waveform
t = [0:nt-1] * dt;
v = -exp(0.5)*(t - m) / s .* exp(-(t - m).^2/(2*s^2));

% Main time step loop
for ii = 1:nt

    ez = ca .* ez + cb .* (hy(2:nx,:) - hy(1:(nx-1),:) + hx(:,2:ny) - hx(:,1:(ny-1)));

    ez(xs_idx,ys_idx) = v(ii);

    if (want_abc)
        hx(:,1) = hx(:,1) * (1 - c * dt / dx) + hx(:,2) * c * dt / dx;
        hx(:,ny) = hx(:,ny) * (1 - c * dt / dx) + hx(:,ny-1) * c * dt / dx;
        hy(1,:) = hy(1,:) * (1 - c * dt / dx) + hy(2,:) * c * dt / dx;
        hy(nx,:) = hy(nx,:) * (1 - c * dt / dx) + hy(nx-1,:) * c * dt / dx;
    end

    hx(:,2:ny-1) = dax(:,2:ny-1) .* hx(:,2:ny-1) + dbx(:,2:ny-1) .* (ez(:,2:ny-1) - ez(:,1:(ny-2)));
    hy(2:nx-1,:) = day(2:nx-1,:) .* hy(2:nx-1,:) + dby(2:nx-1,:) .* (ez(2:nx-1,:) - ez(1:(nx-2),:));

    % Update time harmonic buffer
    ei_z = ei_z + ez * (cos(2*pi*f*(ii-1)*dt) - j * sin(2*pi*f*(ii-1)*dt));

end

% The remaining code is for plotting/visualisation purposes only

if (want_plot_excitation)
    f0 = figure;
    plot(t*1e9,v);
    xlabel('t (ns)');
    ylabel('v (V)');
    title(sprintf('FDTD: %s: Excitation waveform', id));
    print(f0, '-depsc2', sprintf('%s_excitation.eps', id))
end

if (want_plot_pulse)  % Transpose of data needed to get matrix in correct orientation
```

```
    f1 = figure;
    [ch,ch]=contourf(ez',ncontours);
    set(ch,'edgecolor','none');
    axis equal;
    %caxis([-0.2 0.2]);
    colorbar;
    xlabel('x sample index');
    ylabel('y sample index');
    title(sprintf('FDTD: %s: Pulse response', id));
    if (want_plot_geometry)
        hold on;
        for ii = 1:n_pec_blocks
            x1 = (pec_blocks(ii,1) - 1);
            y1 = (pec_blocks(ii,2) - 1);
            x2 = pec_blocks(ii,3);
            y2 = pec_blocks(ii,4);
            plot([x1 x2 x2 x1 x1],[y1 y1 y2 y2 y1],'w');
        end
        plot(xs_idx, ys_idx, 'wo');
        hold off
    end
    print(f1, '-depsc2', sprintf('%s_pulse.eps', id))
end

if (want_plot_time_harmonic)
    f2 = figure;
    [ch,ch]=contourf(real(ei_z)',ncontours);
    set(ch,'edgecolor','none');
    axis equal;
    %caxis([-2 2]);
    colorbar;
    xlabel('x sample index');
    ylabel('y sample index');
    title(sprintf('FDTD: %s: re(ei_z), f = %5.2f GHz', id, f/1e9));
    if (want_plot_geometry)
        hold on;
        for ii = 1:n_pec_blocks
            x1 = (pec_blocks(ii,1) - 1);
            y1 = (pec_blocks(ii,2) - 1);
            x2 = pec_blocks(ii,3);
            y2 = pec_blocks(ii,4);
            plot([x1 x2 x2 x1 x1],[y1 y1 y2 y2 y1],'w');
        end
        plot(xs_idx, ys_idx, 'wo');
        hold off
    end
    print(f2, '-depsc2', sprintf('%s_timeharmonic.eps', id))
end

if (want_plot_lattice)
    f3 = figure;
    hold on;
    for ii = 1:nx-1
        for jj = 1:ny-1
            % hx
            x1 = (ii - 1) * dx;
            y1 = (jj - 1) * dx;
            x2 = ii * dx;
            y2 = (jj - 1) * dx;
            plot([x1 x2],[y1 y2],'b');

        end
    end
    hold off
    print(f3, '-depsc2', sprintf('%s_lattice.eps', id))
end

if (want_plot_movie)
    f4 = figure;
    frame = 0;
    mesh(real(ei_z));
```

```
    set(gca,'nextplot','replacechildren');
    for ii = 0:20:340
        frame = frame + 1;
        theta = ii * pi/180;
        [ch,ch]=contourf(real(ei_z * exp(j * theta))',ncontours);
        set(ch,'edgecolor','none');
        axis equal;
        caxis([-3 3]);
        colorbar;
        xlabel('x sample index');
        ylabel('y sample index');
        title(sprintf('FDTD: %s: re(ei_z), f = %5.2f GHz', id, f/1e9));
        if (want_plot_geometry)
            hold on;
            for ii = 1:n_pec_blocks
                x1 = (pec_blocks(ii,1) - 1);
                y1 = (pec_blocks(ii,2) - 1);
                x2 = pec_blocks(ii,3);
                y2 = pec_blocks(ii,4);
                plot([x1 x2 x2 x1 x1],[y1 y1 y2 y2 y1],'w');
            end
            plot(xs_idx, ys_idx, 'wo');
            hold off
        end
        drawnow;
        mov(frame) = getframe;
    end
end
```


References

[1] K. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Trans. Antennas Propagat.*, vol. 14, no. 3, pp.302307, 1966.

[2] A. Taflove and S. C. Hagness, *Computational electrodynamics: the finite-difference time-domain method*. Boston: Artech House, 2005.

[3] D. B. Davidson, *Computational Electromagnetics for RF and Microwave Engineering*, 2nd ed. Cambridge, 2011.

[4] U. S. Inan and R. A. Marshall, *Numerical Electromagnetics: The FDTD Method*. Cambridge, 2011.

[5] A. C. M. Austin, "Interference Modelling for IndoorWireless Systems using the Finite-Difference Time-Domain Method," Ph.D. dissertation, Department of Electrical and Computer Engineering, The University of Auckland, New Zealand, 2011.

[6] J. Carmo, "GTD-UTD diffraction" (https://www.mathworks.com/matlabcentral/fileexchange/2236-gtd-utd-diffraction), *MATLAB Central File Exchange*, 2022.