Module 3 Assignment

Alessandro Abbati

7/29/23

HW. Implement the Yee scheme from the paper [1]. Try to match outputs as in the paper, with snapshots in time that capture the wave propagation.

Solution:

Introduction

Any electromagnetic field can be decomposed into “transverse electric” and “transverse magnetic” fields with the correct choice of constants μ and ϵ, [1]. The constants μ and ϵ, are the vacuum permeability and vacuum permittivity constants, respectively. In order to recreate the results in [1], we carry out the numerical computations for the transverse magnetic wave (TM), which are characterized by:

$$E_x = E_y = 0, \qquad H_z = 0,$$

$$\epsilon \frac{\partial E_z}{\partial t} = \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}$$

$$\mu \frac{\partial H_x}{\partial t} = -\frac{\partial E_z}{\partial y}, \qquad \mu \frac{\partial H_y}{\partial t} = \frac{\partial E_z}{\partial x},$$

Where, E is the electric field, and H is the magnetic field strength.

We model the interaction between a generated incident wave at a source location that will propagate as a plane wave, which will then be “scattered” after encountering an obstacle, and give rise to a reflected wave.

The Numerical Scheme

We use finite difference equations (14a)-(14c) for the TM wave:

$$E_z^{n+1}(i, j) = E_z^n(i, j) + Z\frac{\Delta\tau}{\Delta x}\left[H_y^{n+\frac{1}{2}}\left(i + \frac{1}{2}, j\right) - H_y^{n+\frac{1}{2}}\left(i - \frac{1}{2}, j\right)\right]$$

$$- Z\frac{\Delta\tau}{\Delta y}\left[H_y^{n+\frac{1}{2}}\left(i, j + \frac{1}{2}\right) - H_x^{n+\frac{1}{2}}\left(i, j - \frac{1}{2}\right)\right] \qquad \text{…(14a)}$$

$$H_x^{n+\frac{1}{2}}(i,\ j+\frac{1}{2}) = H_x^{n-\frac{1}{2}}(i,\ j+\frac{1}{2}) - \frac{1}{Z}\frac{\Delta\tau}{\Delta y}\left[E_z^n(i,\ j+1) - E_z^n(i,\ j)\right] \quad \text{...(14b)}$$

$$H_y^{n+\frac{1}{2}}(i+\frac{1}{2},\ j) = H_y^{n-\frac{1}{2}}(i+\frac{1}{2},\ j) + \frac{1}{Z}\frac{\Delta\tau}{\Delta x}\left[E_z^n(i+1,\ j) - E_z^n(i,\ j)\right] \quad \text{...(14c)}$$

For the initial condition $(t = 0)$, the values of $E_z^0(i,\ j)$, $H_y^{\frac{1}{2}}(i+\frac{1}{2},\ j)$, and $H_x^{\frac{1}{2}}(i,\ j-\frac{1}{2})$ are obtained from the incident wave, where "we choose t such that when $t = 0$, the incident wave has not encountered the obstacle" [1]. We will implement a left traveling plane wave.

The Grid

We implement a two-dimensional grid with coordinates $(i,\ j) = (i\Delta x,\ j\Delta y)$. We will compute and store the $E_z,\ H_x, H_y$ values at each time step t, where $0 \le t \le 95$, $0 \le i \le 81$, and $0 \le j \le 98$.

The Obstacle

The obstacle of choice is a perfectly conducting square, of length 32 (4 alpha), whose side boundaries are along the horizontal lines from $i = 17$, to $i = 49$, along $j = 33$, and $j = 65$, and is characterized by the following boundary conditions:

Left Side	$E_z = 0, H_x = 0$
Right Side	$E_z = 0, H_x = 0$
Top	$E_z = 0, H_y = 0$
Bottom	$E_z = 0, H_y = 0$

The Incident Wave

We let the left-traveling incident wave be plane with a profile of a half sine wave. The incident wave will only have E_z and H_y components, with a pulse width of 2 alpha units.

The Updates

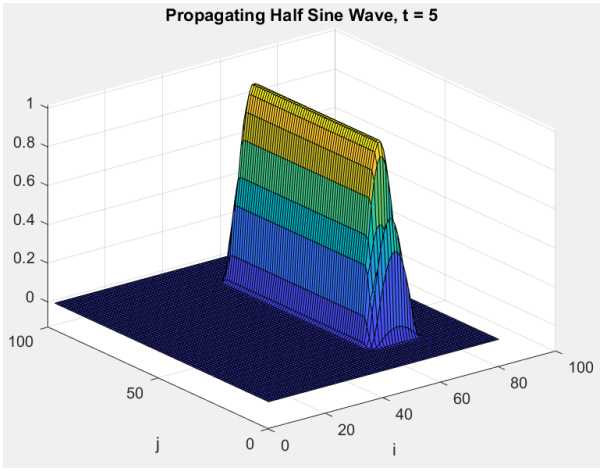
The order in which the updates take place matter the most when trying to attain an accurate model of wave propagation. In [1], it is not readily obvious the order in which E_z, H_y, H_x, is updated with respect to the incident wave data. The order that appeared to produce the best visual results was:

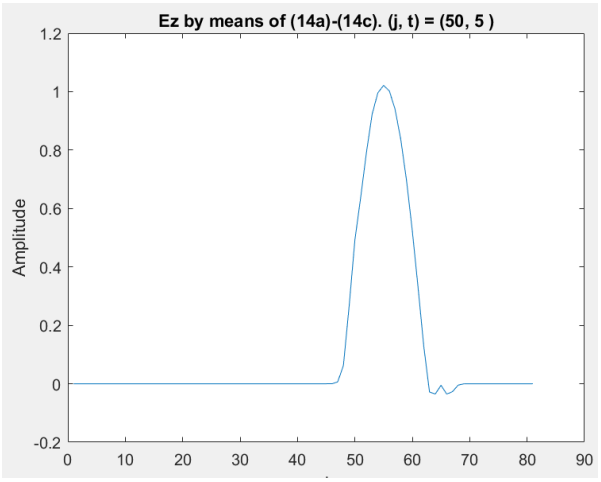
1. Generate initial conditions E_z, H_y for the incident wave using 16(a) and 16(b).
2. Begin temporal loop.
    a. Generate boundary conditions of grid & obstacle.
    b. Update the magnetic field using 14(b) & 14(c).
    c. Update the electric field with 14(a).

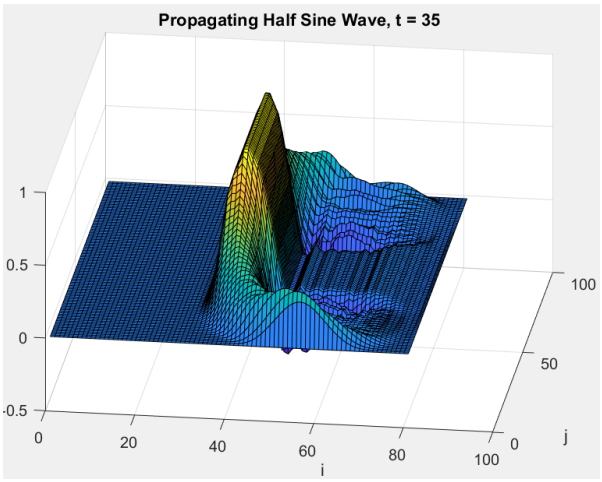
See the appendix for the code used, which will show the order in which the updates were applied, as well as when the boundary conditions were applied, and when the incident wave data was incorporated.

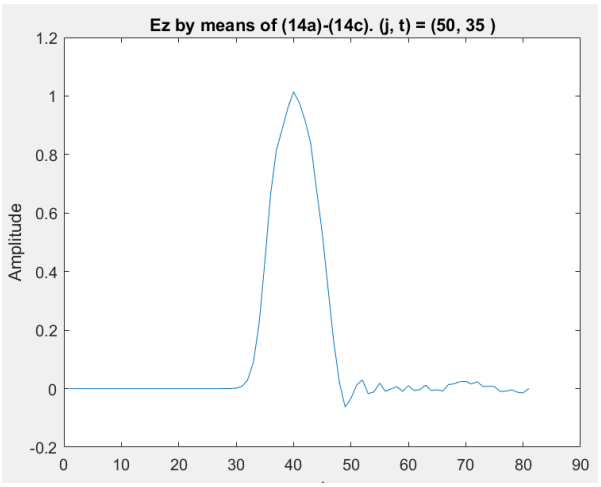
Numerical Results

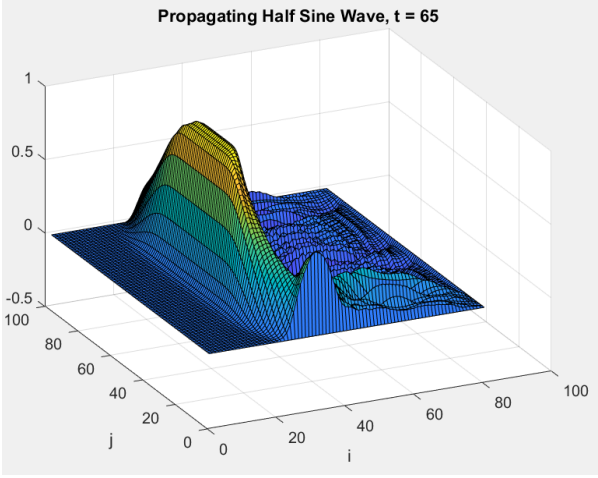
Absence of Obstacle, j = 50

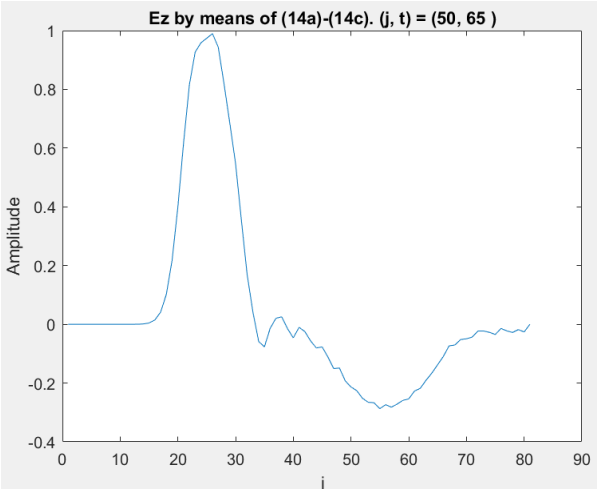


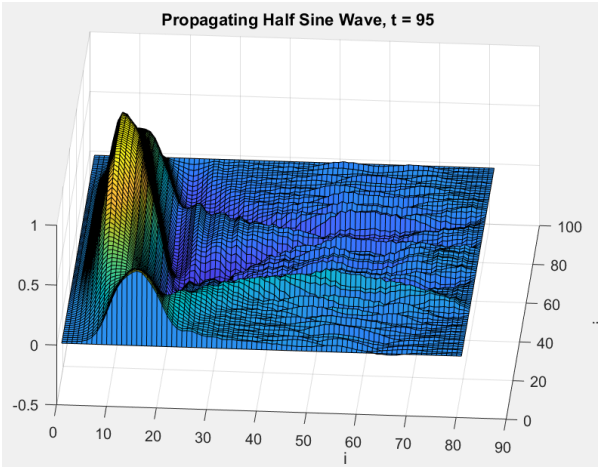


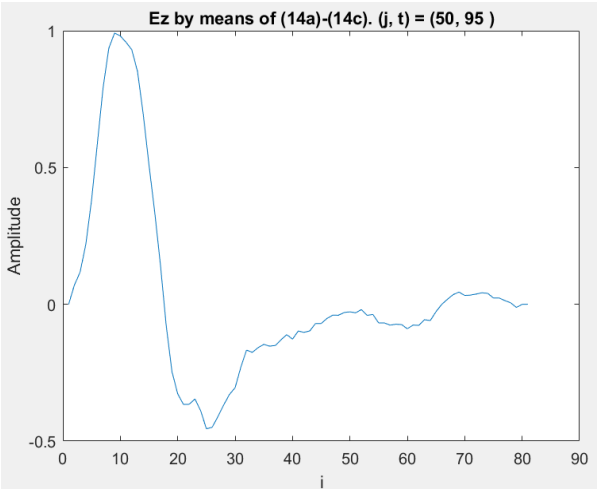




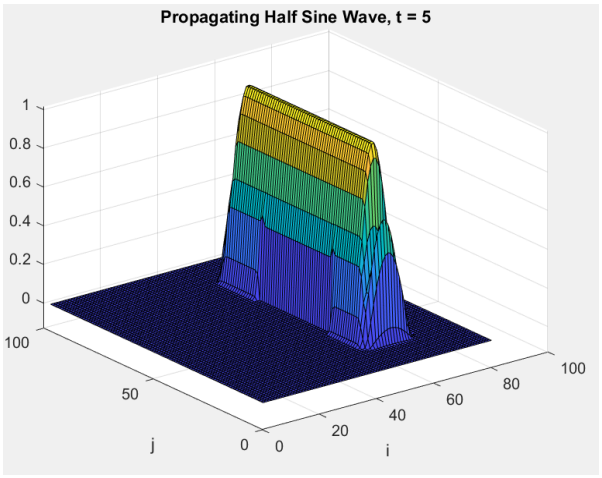

Propagating Half Sine Wave, t = 65
1
0.5
0
-0.5
100
80
60
40
20
0
j
0
20
40
60
80
100
i

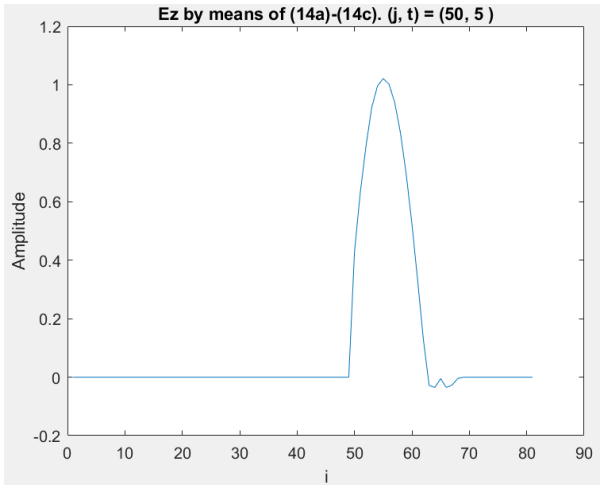

Ez by means of (14a)-(14c). (j, t) = (50, 65)
Amplitude
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
10
20
30
40
50
60
70
80
90
i

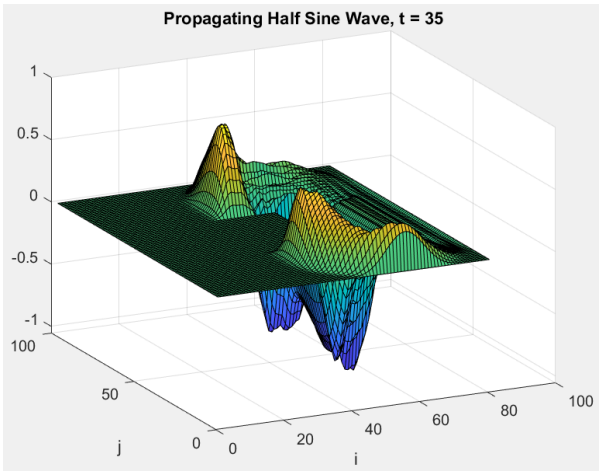

Propagating Half Sine Wave, t = 95
1
0.5
0
-0.5
0
10
20
30
40
50
60
70
80
90
i
100
80
60
40
20
0

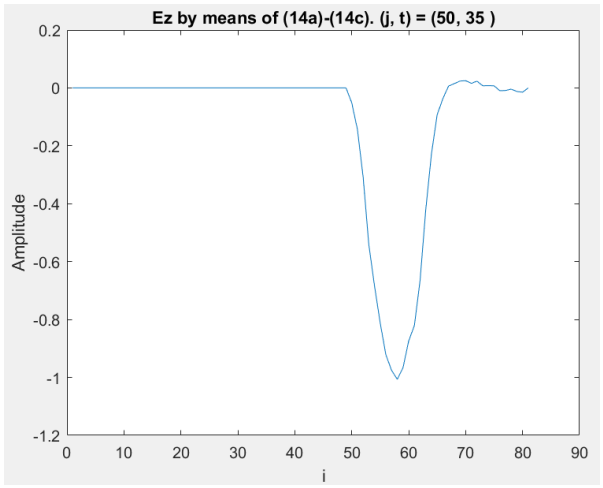

Ez by means of (14a)-(14c). (j, t) = (50, 95)
Amplitude
1
0.5
0
-0.5
0
10
20
30
40
50
60
70
80
90
i

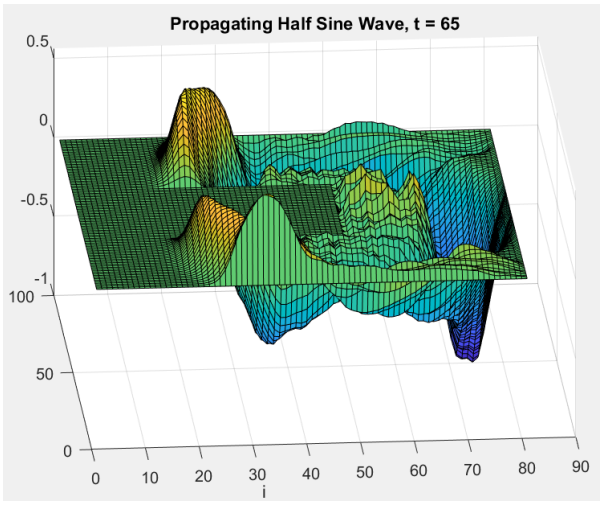
Obstacle Present, j = 50

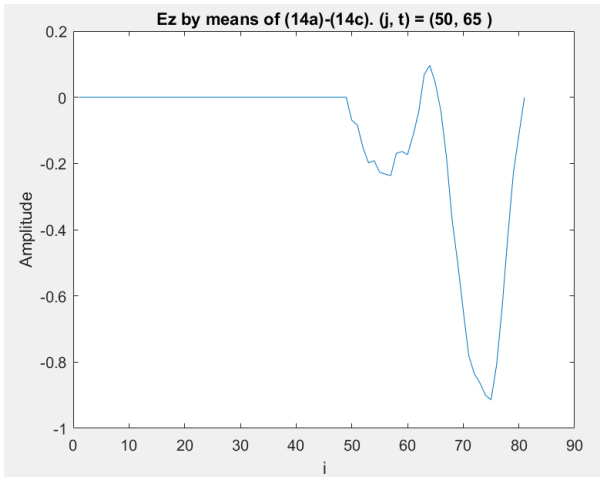


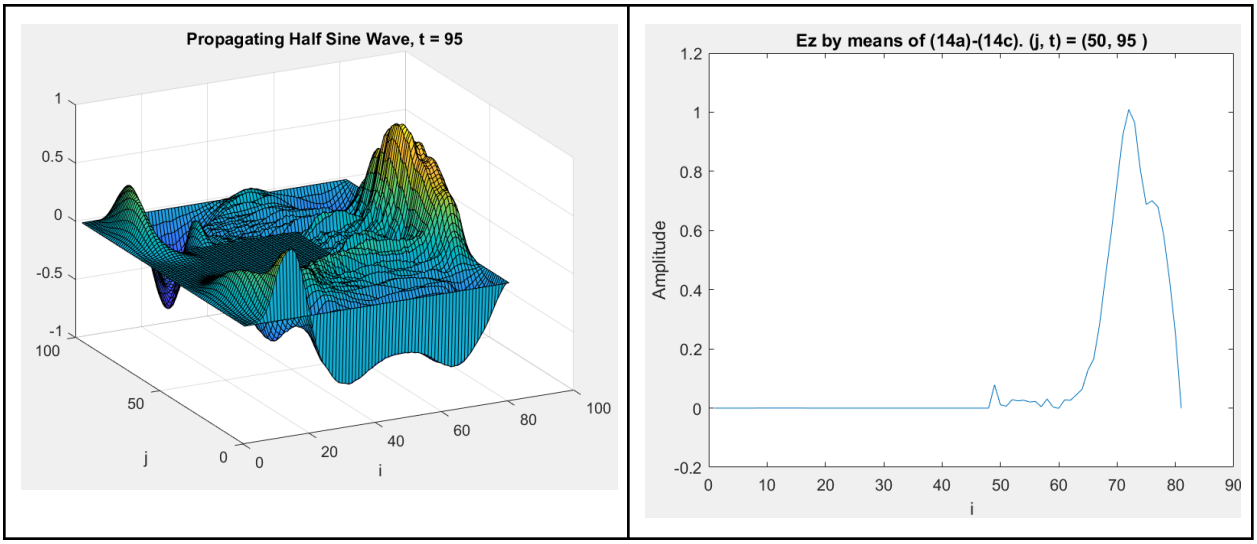




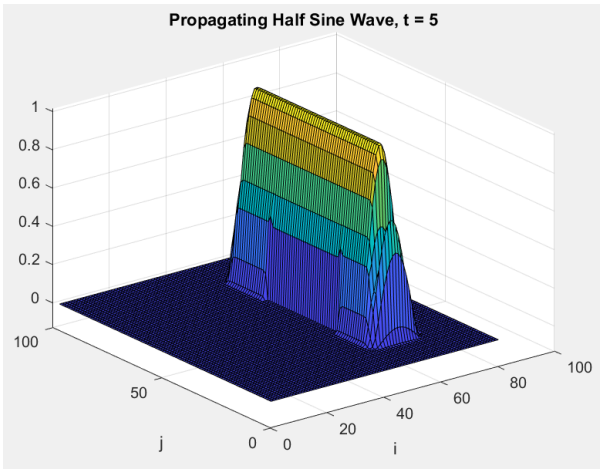


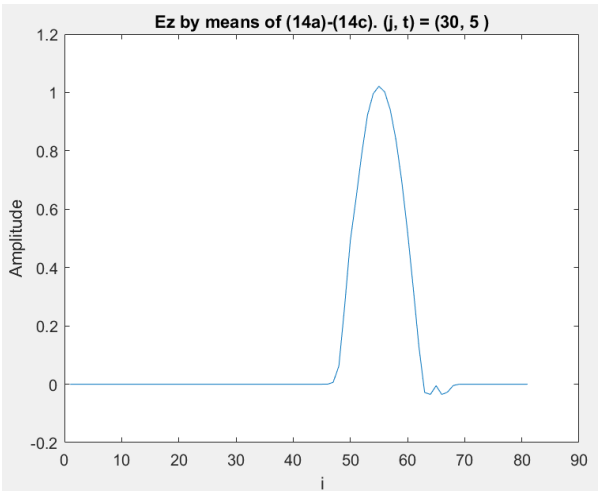


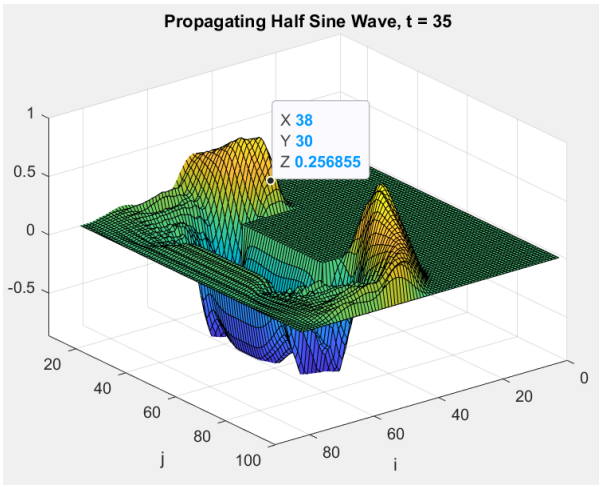


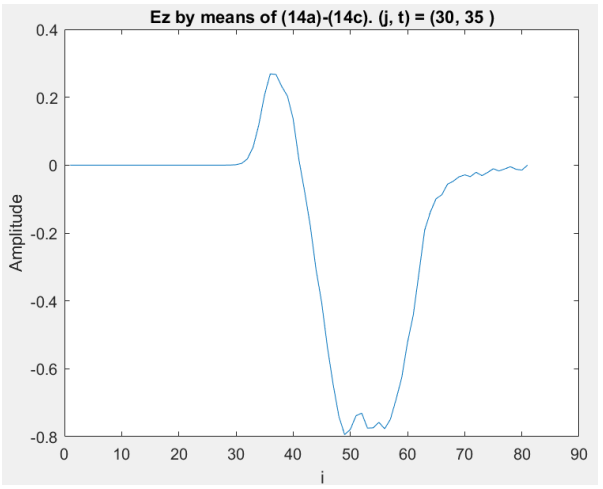

Propagating Half Sine Wave, t = 95
1
0.5
0
-0.5
-1
100
50
0
j
0
20
40
60
80
100
i
Ez by means of (14a)-(14c). (j, t) = (50, 95)
Amplitude
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
10
20
30
40
50
60
70
80
90
i

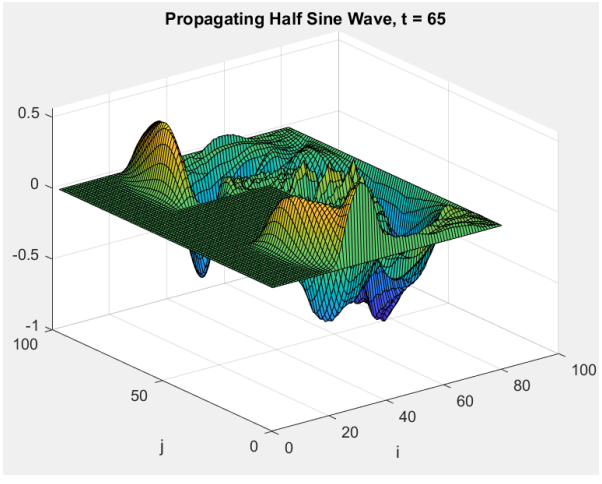
Obstacle Present, j = 30

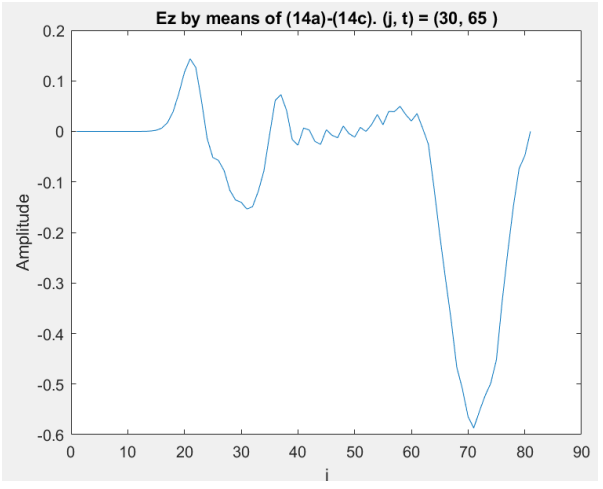


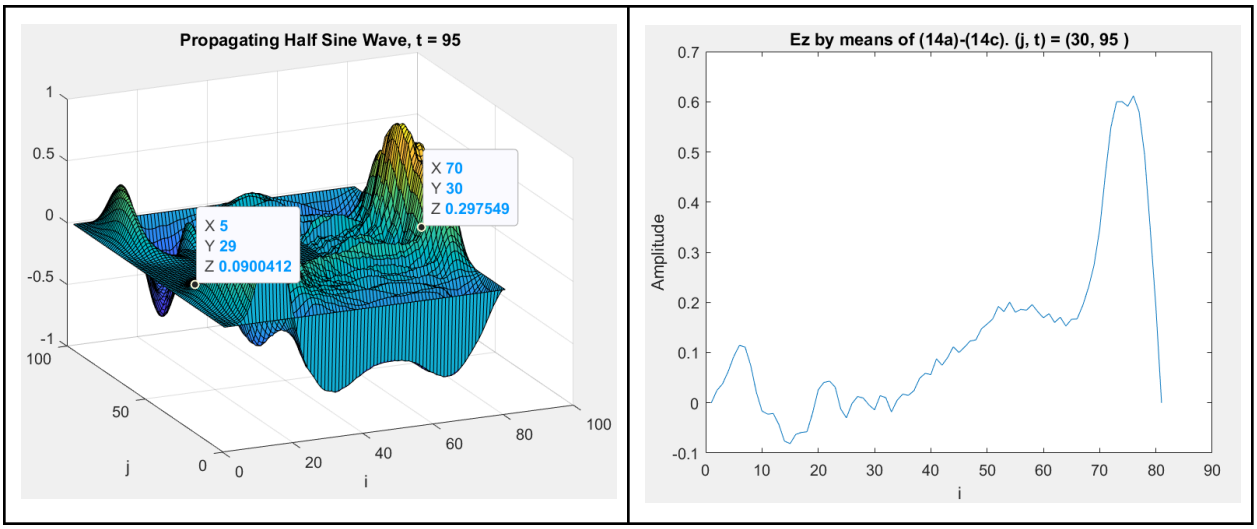












Overall, the generated plots at various time steps, and across various j values, appear to be in agreement with the numerical results in [1]. Potential improvements of the attached code would involve ensuring the amplitudes of the reflected wave as the time steps increase are in agreement with [1], as well as ensuring none of the boundary conditions are being violated after updating the magnetic and electric fields.

HW 4.4.3. Solve the initial-boundary-value problem given in HW 4.2.1 using the Peaceman-Rachford scheme and the same M_x, M_y and ν. Use $\Delta t = 0.01$. Compare your solutions with those of HW 4.2.1.
Solution:

The initial-boundary-value problem given in HW 4.2.1 is:
$v_t = \nu(v_{xx} + v_{yy}),\ (x,\ y) \in (0,\ 1)\ x\ (0,\ 1),\ t > 0$
$v(0,\ y,\ t) = v(1,\ y,\ t) = 0,\ y \in [0,\ 1],\ t \geq 0$
$v(x,\ 0,\ t) = v(x,\ 1,\ t) = 0,\ x \in [0,\ 1],\ t \geq 0$
$v(x,\ y,\ 0) = sin\pi x\ sin2\pi y, (x, y) \in [0,\ 1]\ x\ [0, 1].$
We are told to use $M_x = M_y = 20,\ \nu = 1.0$, and calculate solutions at $t = 0.6,\ t = 0.1, t = 0.9$, and $\Delta t = 0.001$.

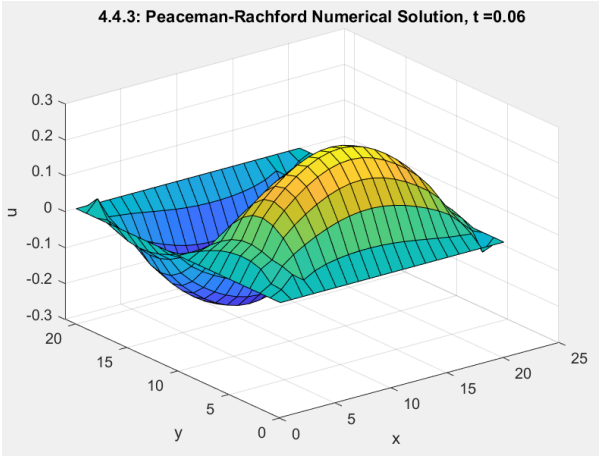
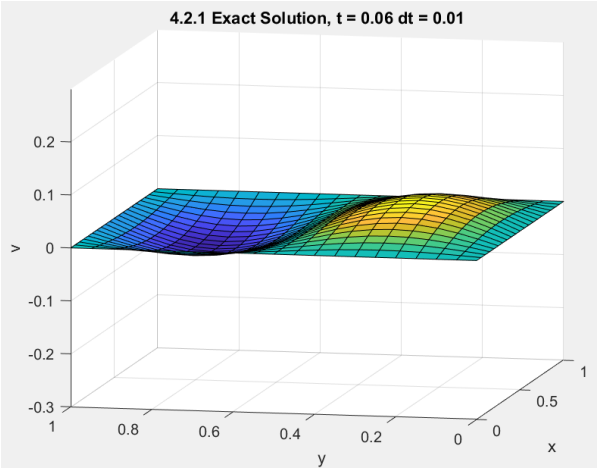
We note that in HW 2.6.2, in Thomas, the implicit Backward-Time Central-Space (BTCS) scheme (2.3.14) was used to solve problem (2.6.4)-(2.6.6), and the Crank-Nicolson scheme, (2.6.7). Which used a stencil to solve a tridiagonal matrix. We will pull implementation ideas from that example problem when solving this one.

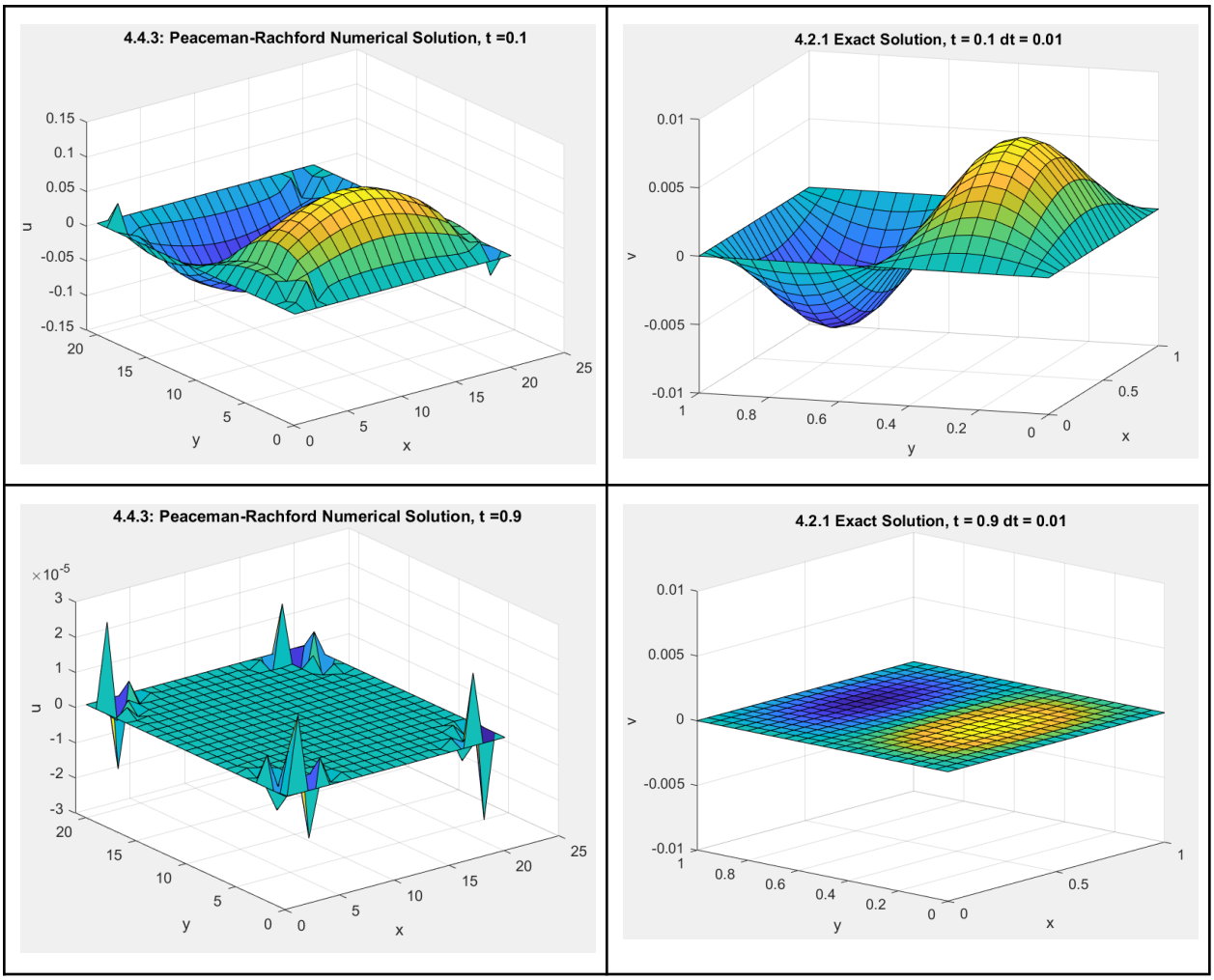
In this problem, we implement the Peaceman-Rachford scheme,
$(1 - \frac{r_x}{2}\delta_x^2)\mu_{jk}^{n+\frac{1}{2}} = (1 - \frac{r_y}{2}\delta_y^2)\mu_{jk}^{n} \quad \cdots (4.4.2)$
$(1 - \frac{r_y}{2}\delta_y^2)\mu_{jk}^{n+1} = (1 - \frac{r_x}{2}\delta_x^2)\mu_{jk}^{n+\frac{1}{2}} \quad \cdots (4.4.6)$
which is an alternating-direction implicit scheme. Expanding (4.4.2), and (4.4.6) to get RHS1 and RHS2, respectively:
$(1 - \frac{r_y}{2}\delta_y^2)\mu_{jk}^{n} = \mu_{jk}^{n} + \frac{r_y}{2}(\mu_{jk+1}^{n} - 2\mu_{jk}^{n} + \mu_{jk-1}^{n})$
$(1 - \frac{r_x}{2}\delta_x^2)\mu_{jk}^{n+\frac{1}{2}} = \mu_{jk}^{n+\frac{1}{2}} + \frac{r_x}{2}(\mu_{j+1\,k}^{n+\frac{1}{2}} - 2\mu_{jk}^{n+\frac{1}{2}} + \mu_{j-1\,k}^{n+\frac{1}{2}}).$

For the implementation, we follow the solution scheme outlined on page 175, Thomas:

1. Call Right Hand Side: (4.4.2)
2. Call Stencil: (4.4.2) to make Q
3. Call Trid to solve Qx = rhs1.
4. Call Right Hand Side: (4.4.6)
5. Call Stencil: (4.4.6) to make Q'
6. Call Trid. Q'x = rhs2.

Where Trid solves the tridiagonal matrix associated with solving equation 4.4.2 that's been created using the subroutine Stencil (4.4.2). Stencil fills in the diagonals of the associated tridiagonal matrix, along with the special zeros. The diagonal elements (a, b, c) of Q and Q' are found in 4.4.34 & 4.4.41, respectively.

Peaceman-Rachford	HW 4.2.1 Solutions
	



When t = 0.9, the u-axis values are close to zero. The numerical solution amplitudes using the Peaceman-Rachford scheme flatten out and go to zero as t increases to 0.9, and they appear to be in agreement with the solution plots from HW 4.2.1.

Bibliography

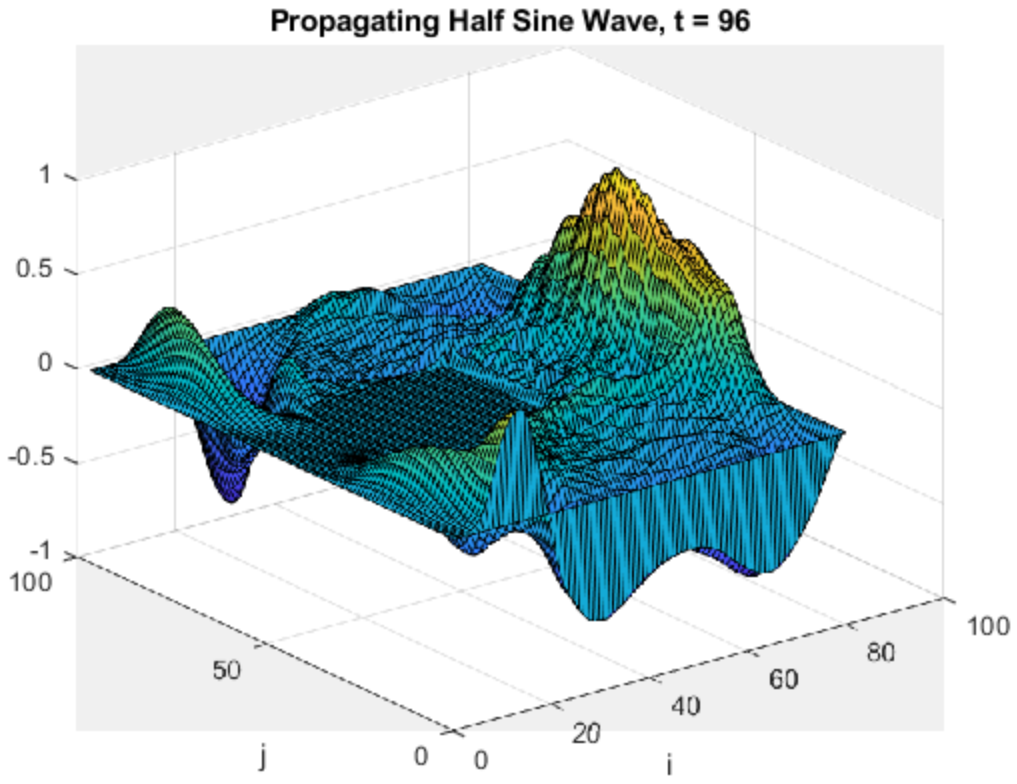
1. Yee, K. (1966). Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, *14*(3), 302–307. https://doi.org/10.1109/tap.1966.1138693

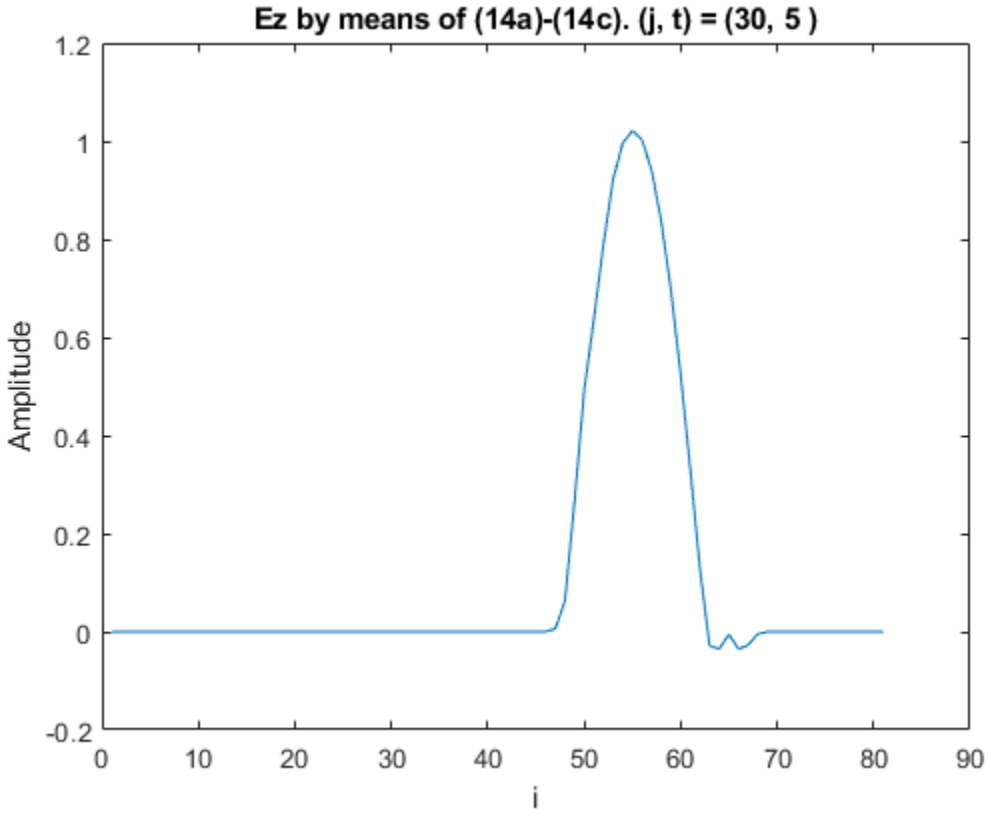
Appendix

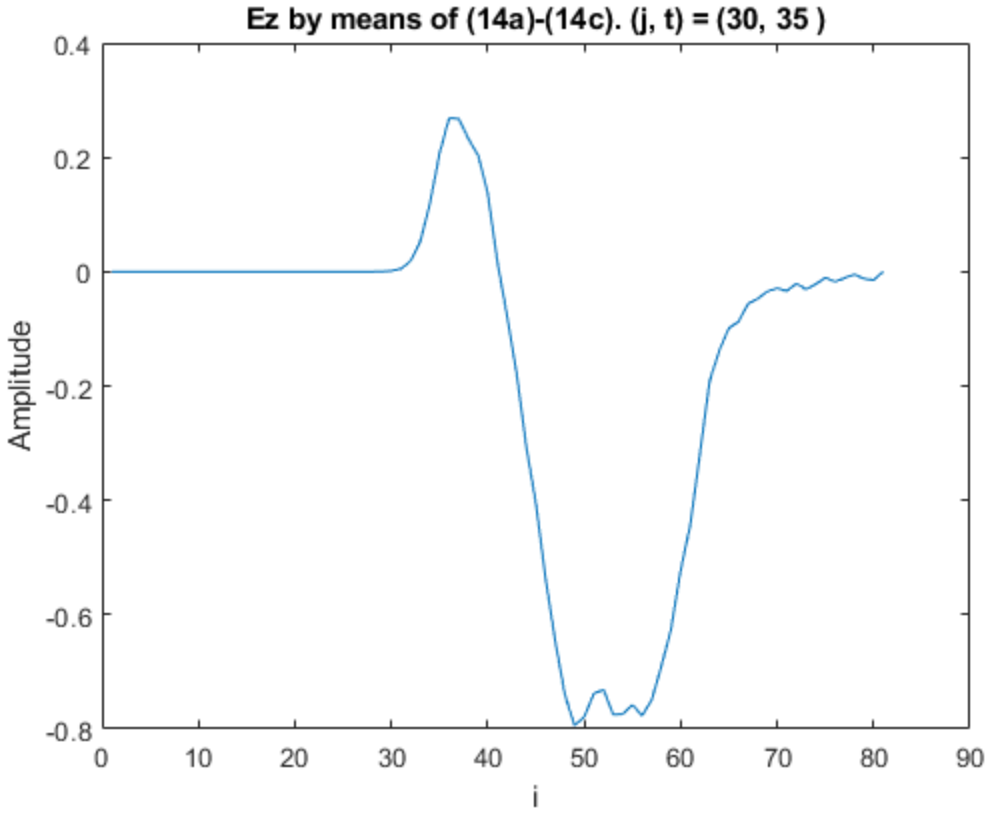
```
%=========================================================================
% EN.625.719 Numerical Diff. EQ.
% Alessandro Abbati
% Implement Yee Numerical Scheme
%=========================================================================
clear;
close all;
clc;


alpha = 8;
c = 3*(10^8);
Z = 376.7;
dtau = alpha/16;
dx = alpha/8;
dy = alpha/8;

t = 0;
x = linspace(((50*alpha) - (c*t)),((58*alpha) - (c*t)),2*alpha);
EzSource = sin(((x' - (50*alpha) + (c*t))*pi)/(8*alpha));
EzSource = repmat(EzSource,1,8*alpha);

Nx = 81;
Ny = 98;
nSteps = 95;

putObstacle = 1;
showPlots = 1;

waveStart = 50;
waveStop = waveStart + (2*alpha-1);

Ez(:,:,1) = zeros(Nx,Ny);
Ez(waveStart:waveStop,19:82,1) = EzSource;

Hy(:,:,1) = (1/Z)*Ez(:,:,1);
Hx(:,:,1) = zeros(Nx,Ny);

M=moviein(nSteps);
for t = 1:nSteps

    %boundary conditions
    Hx(:, 98, t) = 0;
    Hx(:, 1, t) = 0;
    Hx(81, :, t) = 0;
    Hx(1, :, t) = 0;
    Ez(81, :, t) = 0;
    Ez(1, :, t) = 0;

    %obstacle boundary conditions
    if putObstacle == 1
        %top and bottom boundary conditions of the square
```

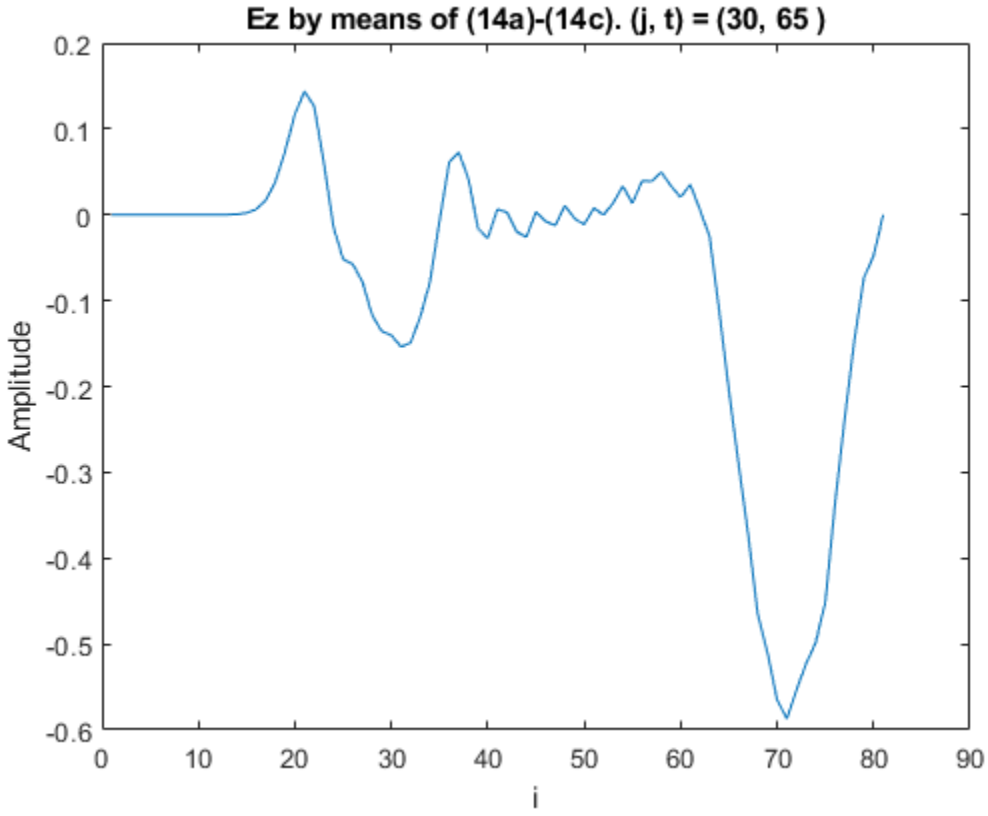
```
        Ez(17:49, 65, t) = 0;
        Hy(17:49, 65, t) = 0;
        Ez(17:49, 33, t) = 0;
        Hy(17:49, 33, t) = 0;

        %left and right boundary conditions of the square
        Ez(49, 33:65, t) = 0;
        Hx(49, 33:65, t) = 0;
        Ez(17, 33:65, t) = 0;
        Hx(17, 33:65, t) = 0;
    end


    % Update the Magnetic Field
    for i = 2:Nx-1
        for j = 2:Ny-1
            Hy(i,j,t+1) = Hy(i,j,t) + ((1/Z)*(dtau/dx)*(Ez(i+1,j,t)-
Ez(i,j,t)));
            Hx(i,j,t+1) = Hx(i,j,t) - ((1/Z)*(dtau/dy)*(Ez(i,j+1,t)-
Ez(i,j,t)));
        end
    end


    % Update the Electric Field
    for i = 2:Nx-1
        for j = 2:Ny-1
            Ez(i,j,t+1) = Ez(i,j,t) + ((Z*dtau/dx)*(Hy(i,j,t+1)-Hy(i-1,j,t
+1))) - ((Z*dtau/dy)*(Hx(i,j,t+1)-Hx(i,j-1,t+1)));
        end
    end


    surf(transpose(Ez(:, :, t+1)));
    title_str = strcat("Propagating Half Sine Wave, t = ", num2str(t+1));
    title(title_str);
    xlabel("i")
    ylabel("j")
    M(:,t) = getframe ;


end
movie(M,1);


if showPlots == 1

    t = 5;
    j = 30;
    figure(2)
    y = transpose(Ez(:, j, t+1));
    plot(y)
    title_str = strcat("Ez by means of (14a)-(14c). (j, t) = (",
 num2str(j), ", ", num2str(t), " )");
```

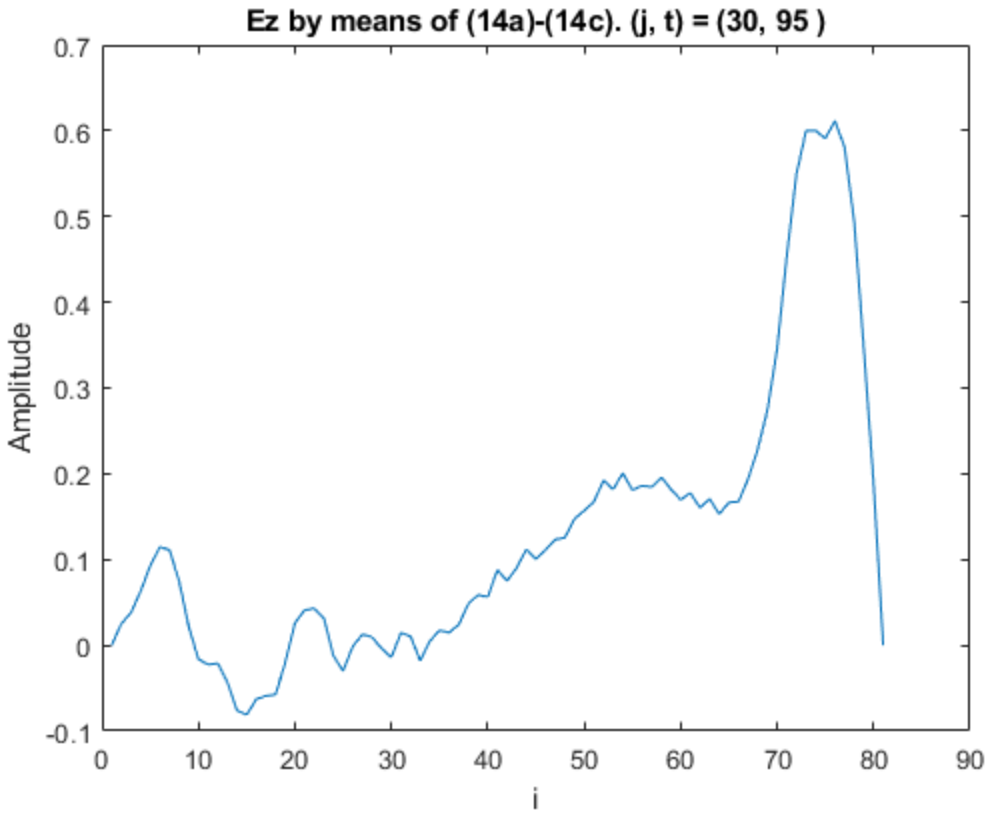
```
    title(title_str)
    xlabel("i")
    ylabel("Amplitude")

    t = 35;
    j = 30;
    figure(3)
    y = transpose(Ez(:, j, t+1));
    plot(y)
    title_str = strcat("Ez by means of (14a)-(14c). (j, t) = (",
 num2str(j), ", ", num2str(t), " )");
    title(title_str)
    xlabel("i")
    ylabel("Amplitude")


    t = 65;
    j = 30;
    figure(4)
    y = transpose(Ez(:, j, t+1));
    plot(y)
    title_str = strcat("Ez by means of (14a)-(14c). (j, t) = (",
 num2str(j), ", ", num2str(t), " )");
    title(title_str)
    xlabel("i")
    ylabel("Amplitude")


    t = 95;
    j = 30;
    figure(5)
    y = transpose(Ez(:, j, t+1));
    plot(y)
    title_str = strcat("Ez by means of (14a)-(14c). (j, t) = (",
 num2str(j), ", ", num2str(t), " )");
    title(title_str)
    xlabel("i")
    ylabel("Amplitude")

    % t = 5; surf(transpose(Ez(:, :, t)));
    % title_str = strcat("Propagating Half Sine Wave, t = ", num2str(t));
    % title(title_str);
    % xlabel("i")
    % ylabel("j")

end
```


Propagating Half Sine Wave, t = 96
1
0.5
0
-0.5
-1
100
50
j
0
0
20
40
60
80
100
i


Ez by means of (14a)-(14c). (j, t) = (30, 5)
Amplitude
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
10
20
30
40
50
60
70
80
90
i


Ez by means of (14a)-(14c). (j, t) = (30, 35)
Amplitude
i
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
10
20
30
40
50
60
70
80
90


Ez by means of (14a)-(14c). (j, t) = (30, 65)
Amplitude
i
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
10
20
30
40
50
60
70
80
90



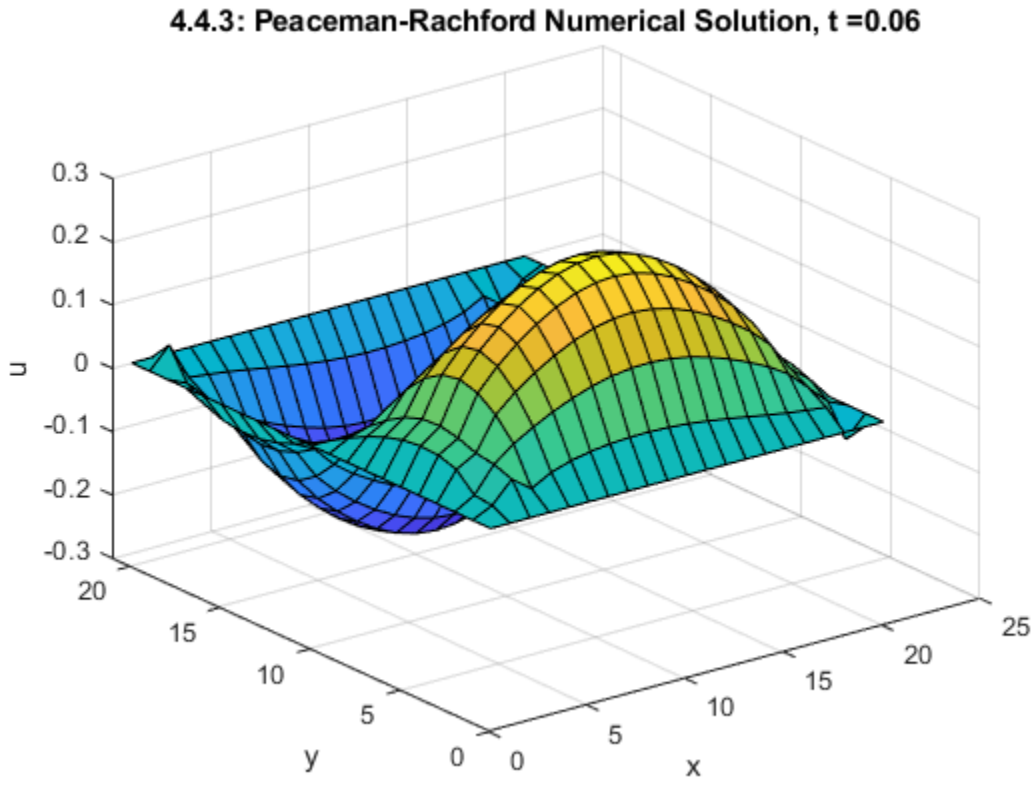
Published with MATLAB® R2023a

```
%========================================================================
% EN.625.719 Numerical Diff. EQ.
% Alessandro Abbati
%
% hw 4.4.3 2D parabolic equation using Peaceman-Rachford Scheme to solve
% f(x, y) = sin(pi*x)sin(2*pi*y) Initial condition (t = 0)
% a = 0 Left & Right boundary conditions (at x = 0 & x = 1)
% b = 0 Top & Bottom boundary conditions (at y = 0 & y = 1)
% nu = 1.0
% M_x = M_y = 20 (deltaT=0.001)
% find solutions at t = 0.06, 0.1, 0.9
% compare to hw 4.2.1 solutions.
%========================================================================

clc;
clear;
close all;

problemNumber = "4.4.3";
M_x = 20;
M_y = 20;

nu = 1;

dx = 1/M_x;
dy = 1/M_y;

dt = 0.01;
t_max = 0.06; %0.06, 0.1, 0.9
nSteps = t_max/dt;

r_x = nu * dt / dx^2;
r_y = nu * dt / dy^2;

x = 0:dx:1;
y = 0:dy:1;

u = zeros(M_x + 1, M_y + 1, nSteps);

%initial condition
for j = 1:M_y+1
    for i = 1:M_x+1
        u(i,j,1) = sin(pi*x(i))*sin(2*pi*y(j));
    end
end


for t = 1:nSteps

    %call RHS1 (4.4.2)
    rhs = [];
    kk = 0;
```

```
    for k = 2:M_y
        for j = 2:M_x
            kk = kk+1;
            rhs(kk) = u(j,k, t) + ((r_y/2)*(u(j,k+1, t) - (2*(u(j,k, t))) +
u(j,k-1, t)));
        end
    end


    %build stencil for (4.4.2) which generates the tridiagonal matrix Q.
    M = (M_x-1)*(M_y-1); %number of diagonal elements

    %diagonals
    for j = 1:M
        b(j) = 1+r_x;
    end

    %off diagonals
    for j = 1:M-1
        a(j) = -r_x/2;
        c(j) = -r_x/2;
    end

    %solve tridiagonal system equals rhs and reshape
    u_new = trid(a,b,c,rhs);
    u_new = reshape(u_new, M_x-1, M_y-1);

    %store u_new
    innerSize = size(u, 2) - 1;
    u(2:innerSize, 2:innerSize, t+1) = u_new;


    %call RHS2 (4.4.6)
    rhs = [];
    kk = 0;
    for j = 2:M_x
        for k = 2:M_y
            kk = kk+1;
            rhs(kk) = u(j,k, t) + ((r_x/2)*(u(j+1,k, t) - (2*u(j,k, t)) +
u(j-1,k, t)));
        end
    end


    %build stencil for (4.4.6) which generates the tridiagonal matrix Q'.
    M = (M_x-1)*(M_y-1); %number of diagonal elements

    %diagonals
    for j = 1:M
        b(j) = 1+r_y;
    end

    %off diagonals
    for j = 1:M-1
```

```
        a(j) = -r_y/2;
        c(j) = -r_y/2;
    end

    %solve tridiagonal system equals rhs and reshape
    u_new = trid(a,b,c,rhs);
    u_new = reshape(u_new, M_x-1, M_y-1);

    %store u_new
    innerSize = size(u, 2) - 1;
    u(2:innerSize, 2:innerSize, t+1) = u_new;


end


figure();
surf(u(:, :, t))
%axis([0,1,0,1,-0.01,0.01])
title_str = strcat('4.4.3: Peaceman-Rachford Numerical Solution, t = ',
 num2str(t_max));
title(title_str);
xlabel('x');
ylabel('y');
zlabel('u');


function rppp = trid(a,b,c,r)

    m = length(b);

    %initialize everything

    a = [a 0];
    c = [0 c];

    bp = 0 .* b;
    ap = bp;
    cp = ap;

    rp = 0 .* r;
    rpp = rp;
    rppp = rp;

    u = rp;

    cp(1) = c(1) / b(1);
    rpp(1) = r(1)/b(1);

    for j = 2:m-1
        bp(j) = b(j) - a(j)*cp(j-1);
        rp(j) = r(j) - a(j)*rpp(j-1);
        cp(j) = c(j) / bp(j);
        rpp(j) = rp(j) / bp(j);
```

```
    end
    bp(m) = b(m) - a(m)*cp(m-1);
    rp(m) = r(m) - a(m)*rpp(m-1);
    rppp(m) = rp(m) / bp(m);

    % go back
    for j = m-1:-1:1
        rppp(j) = rpp(j) - cp(j)*rppp(j+1);
    end

end
```



Published with MATLAB® R2023a