Tom Penick

EE 325K Homework 10, November 30, 2000

Problem:

Write a 2D FDTD (finite difference time domain) computer code to simulate wave propagation due to a line current J_z at (5m,10m). Plot E_z at (15m, 10m) from 0 to 200 ns.

Source: $$J_z(t) = \begin{cases} 0, & t < 0 \\ 1 - \cos\left(\frac{\pi t}{T}\right), & 0 \le t \le T \\ 2, & t > T \end{cases}$$

T = 10 ns, Δ = 0.1 m, Δt = 0.2 ns

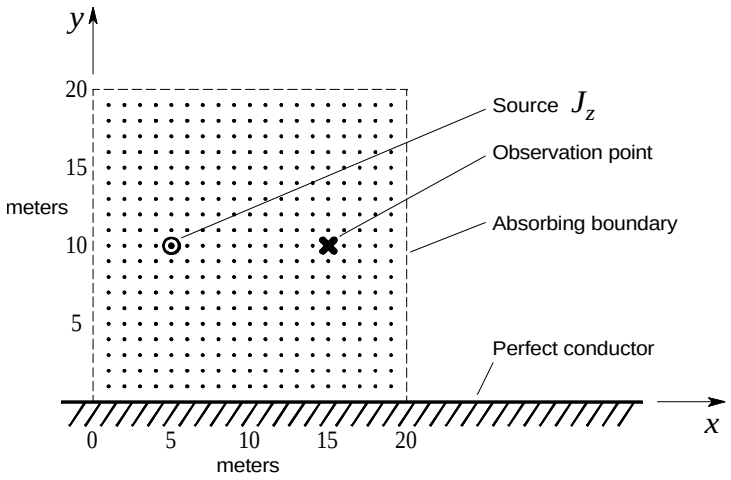
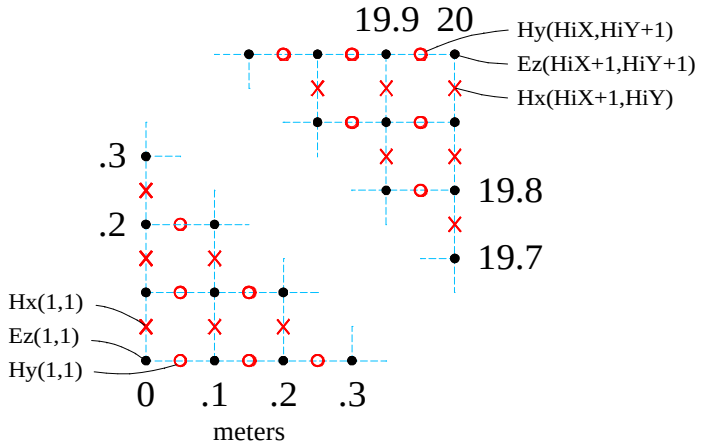
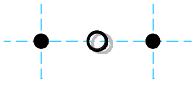
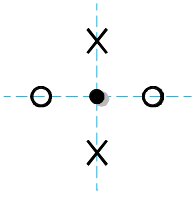


Diagram from the problem statement showing the area of calculation.



Detail showing the upper right and lower left corners of the Yee grid. This shows how matrix elements written in Matlab code correspond to H_x, H_y, and E_z of the Yee grid. Matlab rows are the *x*-values (*i*-values) and Matlab columns are the *y*-values (*j*-values).

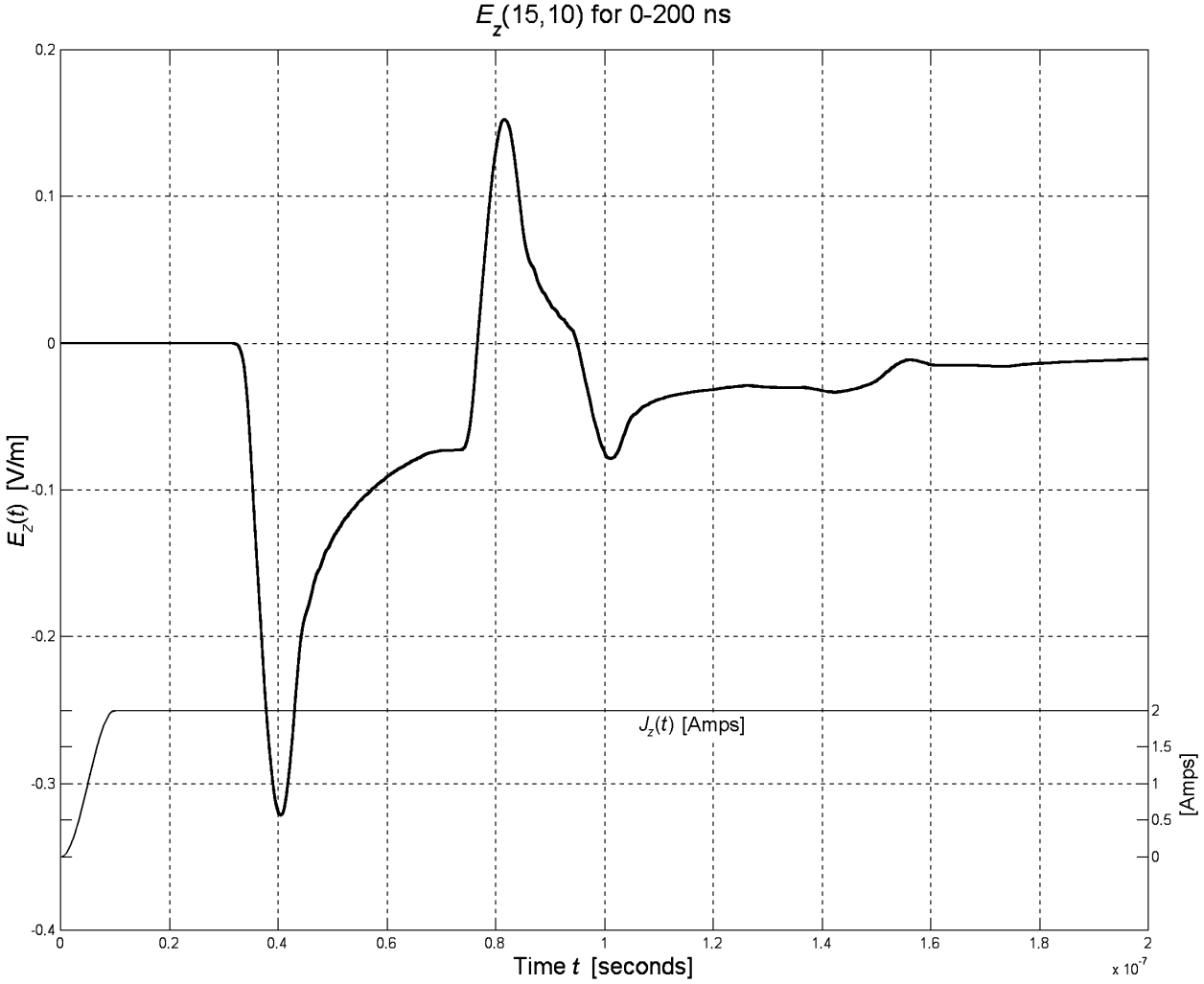
Legend	$\boldsymbol{H_x}$ is calculated based on the past value of H_x and the values of E_z above and below	$\boldsymbol{H_y}$ is calculated based on the past value of H_y and the values of E_z to the left and right	$\boldsymbol{E_z}$ is calculated based on the past value of E_z and the values of H_x above and below, and the values of H_y to the left and right.
● — X -- ○ --			

Detail showing how the Matlab calculations propagate through the Yee grid.

The project was done using Matlab. Two 201×201 matrices are created for present and future values of E_z, one 200×200 matrix is created for J_z, one 201×200 matrix for H_x, and one 200×200 matrix for H_y. These values are easily scaled for other problems by changing parameters in the first section of the code. Execution time using a Celeron 450 MHz processor is 93 seconds.

The wave calculations propagate by one grid point ($\delta = 0.1$m) horizontally and vertically each time step ($\Delta t = 0.2$ ns). So propagation to a diagonal grid point takes two time steps ($2\Delta t = 0.4$ ns). The propagation of calculations (not the wave itself) along the direct path from source to observation point takes $\Delta t \times 10\text{m}/\delta = 20$ ns. The reflected path from the perfect conductor takes a minimum of $\Delta t \times 30\text{m}/\delta = 60$ ns for the calculation and $22\text{m}/c = 73.4$ ns for the wave propagation. Should there be a reflection from the nearest "absorbing" boundary, that would take $\Delta t \times 20\text{m}/\delta = 40$ ns for the calculations to propagate, and $20\text{m}/c = 66.7$ ns for the wave to propagate at the speed of light. I am using the MUR1 method for dealing with boundaries, so some error is possible.

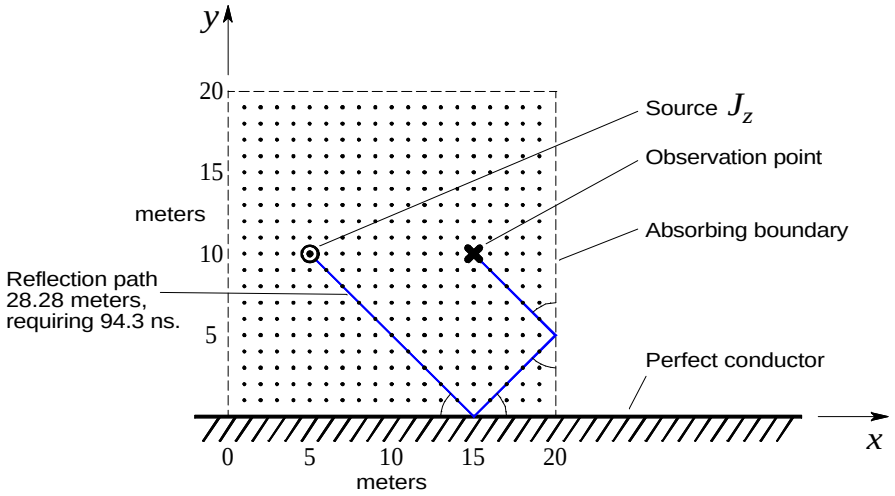
At the speed of light, the wave should propagate from source to observation point in 33.4 ns. Since the propagation of calculations for this path is 20 ns, it should be possible to produce a



The problem result. The electric field E_z at the observation point is shown with a bold line using the scale at left, and the input current J_z is shown with a thin line at the bottom of the plot using the scale at right.

realistic result using Δt = 0.2 ns and δ = 0.1 m. The plot confirms this, showing a 32 ns interval from t = 0 until the disturbance is seen at the observation point. At this point, the observed electric field swings negative in response to the increasing current J_z. The duration of this negative swing is about 10 ns, same as the ramp time for J_z. With J_z a constant 2 amps, the response is an exponential decay of the electric field at the observation point.

At t = 74 ns the electric field moves sharply positive. This is the amount of time required for a reflection from the perfect conductor to reach the observation point (22m/c = 73.4 ns). The duration of this positive swing is about 10 ns, corresponding to the ramp time for J_z. Following this positive swing, another exponential decay is observed terminating in a sharply negative swing at 94 ns. The negative direction and the remaining strength of the response suggests a 2-reflection wave that uses the perfect conductor for at least one reflection. It is found that a wave reflecting first from the perfect conductor and then from the right-hand "absorbing" boundary would have a path length of 28.3 meters yielding a propagation time of 28.3m/c = 94.3 ns. This appears to be the cause of the second steep, almost linear downward excursion and points out a problem with the MUR1 method of handling the absorbing boundary.



Detail showing possible path of second significant reflection.

Following this excursion, the E_z component begins to return exponentially (more or less) to zero with some minor ripples evidently due to other reflections.

The 2D FDTD Equations

where *i* is a value on the horizontal axis, *j* is a value on the vertical axis, and *n* is a time value.

$$H_x^{n+\frac{1}{2}}\left(i, j+\tfrac{1}{2}\right) = H_x^{n-\frac{1}{2}}\left(i, j+\tfrac{1}{2}\right) - \frac{\Delta t}{\delta \mu_0}\left[E_z^n\left(i, j+1\right) - E_z^n\left(i, j\right)\right]$$

$$H_y^{n+\frac{1}{2}}\left(i+\tfrac{1}{2}, j\right) = H_x^{n-\frac{1}{2}}\left(i+\tfrac{1}{2}, j\right) + \frac{\Delta t}{\delta \mu_0}\left[E_z^n\left(i+1, j\right) - E_z^n\left(i, j\right)\right]$$

$$E_z^{n+1}\left(i, j\right) = E_z^n\left(i, j\right) + \frac{\Delta t}{\delta \varepsilon_0}\left[H_y^{n+\frac{1}{2}}\left(i+\tfrac{1}{2}, j\right) - H_y^{n+\frac{1}{2}}\left(i-\tfrac{1}{2}, j\right) - H_x^{n+\frac{1}{2}}\left(i, j+\tfrac{1}{2}\right) + H_x^{n+\frac{1}{2}}\left(i, j-\tfrac{1}{2}\right)\right] - \frac{\Delta t}{\varepsilon_0} J_z^{n+\frac{1}{2}}\left(i, j\right)$$

The Courant Stability Condition: $\Delta t \le \frac{\delta}{\sqrt{2}}\left(\frac{1}{c}\right)$

MUR1 Absorbing Boundary Conditions

where *N* is the boundary element (opposite 0)

Left Boundary: $E_z^{n+1}\left(0, j\right) = E_z^n\left(1, j\right) + \frac{c\Delta t - \delta}{c\Delta t + \delta}\left[E_z^{n+1}\left(1, j\right) - E_z^n\left(0, j\right)\right]$

Right Boundary: $E_z^{n+1}\left(N, j\right) = E_z^n\left(N-1, j\right) + \frac{c\Delta t - \delta}{c\Delta t + \delta}\left[E_z^{n+1}\left(N-1, j\right) - E_z^n\left(N, j\right)\right]$

Top Boundary: $E_z^{n+1}\left(i, N\right) = E_z^n\left(i, N-1\right) + \frac{c\Delta t - \delta}{c\Delta t + \delta}\left[E_z^{n+1}\left(i, N-1\right) - E_z^n\left(i, N\right)\right]$

MUR2 Absorbing Boundary Conditions

Left Boundary: $E_z^{n+1}\left(0, j\right) = E_z^n\left(1, j\right) + \frac{c\Delta t - \delta}{c\Delta t + \delta}\left[E_z^{n+1}\left(1, j\right) - E_z^n\left(0, j\right)\right]$

$$- \frac{\mu_0 c}{2\left(c\Delta t + \delta\right)}\left[H_x^{n+\frac{1}{2}}\left(0, j+\tfrac{1}{2}\right) - H_x^{n+\frac{1}{2}}\left(0, j-\tfrac{1}{2}\right) + H_x^{n+\frac{1}{2}}\left(1, j+\tfrac{1}{2}\right) - H_x^{n+\frac{1}{2}}\left(1, j-\tfrac{1}{2}\right)\right]$$

(Right boundary and top boundary are similarly modified.)

MATLAB CODE

function FDT2()

```
%   Finite Difference Time Domain, EE325K HW 10, by Tom Penick
%   This function plots Ez at (15m, 10m) from 0<t<200 ns in response
%   to signal Jz(t) = 0, t<0; = 1-cos(pi*t/T), 0<t<T; = 2, t>T.
%   Jz(t) is located at (5m, 10m). The solution domain is (0-20m, 0-20m)
% INDEXING CONVENTIONS:  Since the problem calls for indices with some integer+1/2 values and Matlab uses
%   only whole number indexing, I need a plan.  For the first three expressions, the matrices calculated
%   are the interior matrices which do not include the boundary rows and columns.
%   (Those rows and columns are present, I just exclude them when calculating.) The other matrices in the
%   calculations are shifted as required with respect to these defining matrices.  Matrix row indices
%   correspond to graphical X positions and matrix column indices correspond to graphical Y positions,
%   e.g. column 1 corresponds to Y=0 and holds the values for the perfect conductor at the bottom of the
%   graphical representation (except for the Hx matrix where it's Y=1/2).

%******************  VARIABLES, PARAMETERS, AND INITIAL CALCULATIONS  ***********************************
T = 10e-9; t = 0; TT = 200e-9;           % ramp time 10 ns, start time 0 s, plot duration 200 ns
JzLoc = [5 10];                          % initial source current location
sRange = [0 20]; sDomain = [0 20];       % solution domain, y=sRange x=sDomain
D = .1;                                  % spatial discretization, delta
dt = .2e-9                               % time discretization, delta t = 0.2 ns
EzObs = []; Time = []; JzPlt = [];       % matrices, observed Ez and time for plotting
EzLoc = [15 10];                         % location of the observation point
uo = 1.25663706144e-6;                   % permiability of free space
eo = 8.85418781762e-12;                  % permittivity of free space
c = 299.792458e6;                        % speed of light
HiX = (sDomain(2)-sDomain(1))/D;         % one less than the number of X-values (rows)
HiY = (sRange(2)-sRange(1))/D;           % one less than the number of Y-values (columns)
Jz = zeros([HiX-1,HiY-1]);               % create current source matrix that doesn't include boundaries
Ez = zeros([HiX+1,HiY+1]);               % create electric field matrix that includes boundaries
Hx = zeros([HiX+1,HiY]);                 % create magnetic field matrix that includes left & right boundaries
Hy = zeros([HiX,HiY+1]);                 % create magnetic field matrix that includes upper & lower boundaries
Ezp=Ez;                                  % create matrices for present grid values, Ez-n
Courant = D/2^.5/c                       % Courant stability condition, must be > dt
                                         % Do some precalculations to speed this dog up.
M1=dt/D/uo; M2=dt/D/eo; M3=dt/eo; M4=(c*dt-D)/(c*dt+D); % some multipliers to be used later

%*****************************  THE EXPRESSIONS TO CALCULATE  ********************************************
ExprHx = 'Hx(2:HiX,1:HiY)=Hx(2:HiX,1:HiY)-M1*(Ezp(2:HiX,2:HiY+1)-Ezp(2:HiX,1:HiY));';
ExprHy = 'Hy(1:HiX,2:HiY)=Hy(1:HiX,2:HiY)+M1*(Ezp(2:HiX+1,2:HiY)-Ezp(1:HiX,2:HiY));';
ExprEz = 'Ez(2:HiX,2:HiY)=Ezp(2:HiX,2:HiY)+M2*(Hy(2:HiX,2:HiY)-Hy(1:HiX-1,2:HiY)-Hx(2:HiX,2:HiY)+Hx(2:HiX,1:HiY-
         1))-M3.*Jz;';
MURlft = 'Ez(1,2:HiY)=Ezp(2,2:HiY)+M4*(Ez(2,2:HiY)-Ezp(1,2:HiY));';
MURrgt = 'Ez(HiX+1,2:HiY)=Ezp(HiX,2:HiY)+M4*(Ez(HiX,2:HiY)-Ezp(HiX+1,2:HiY));';
MURtop = 'Ez(1:HiX+1,HiY+1)=Ezp(1:HiX+1,HiY)+M4*(Ez(1:HiX+1,HiY)-Ezp(1:HiX+1,HiY+1));';

%*********************************  THE PROGRAM CODE  ****************************************************
while t<=TT
  if t <= T                                % select the proper value for Jz
    Jz(JzLoc(1)/D,JzLoc(2)/D)=1-cos(pi*t/T); % Jz (current) is 2x2 smaller than the other matrices, but
    JzPlt = [JzPlt 1-cos(pi*t/T)];         %   since it is the 0th row and column that are dropped, no
  else                                     %   correction is needed to index the source current location.
    Jz(JzLoc(1)/D,JzLoc(2)/D)=2;           % the source current, Jz, after t = T
    JzPlt = [JzPlt 2];                     % the source current, Jz, saved for plotting
  end
  eval(ExprHx); eval(ExprHy); eval(ExprEz);  % evaluate the first three expressions
  eval(MURlft); eval(MURrgt); eval(MURtop);  % evaluate the MUR1 calculations for the perimeter
  EzObs = [EzObs Ez(EzLoc(1)/D+1,EzLoc(2)/D+1)]; % electric field at the observation point for plotting
  Time = [Time t]; t = t+dt;               % save the time value for plotting, then increment the time
  Ezp = Ez;                                % transfer the new values to the matrices for the past values
end

%*************************  PLOT Ez AND Jz VERSUS TIME  ***************************************************
figure('Position',[-10 -110 1700 1100])                                  % gimme a big window
[Ax,H1,H2] = plotyy(Time,EzObs,Time,JzPlt); grid on;                     % dual y-axis plot
set(Ax(1),'Ylim',[-.4 .2],'Ytick',[-.4 -.3 -.2 -.1 0 .1 .2]);            % left-hand y-axis settings
set(Ax(2),'Ylim',[-1 11],'Ytick',[0 .5 1 1.5 2]);                        % right-hand y-axis settings
set(H1,'LineWidth',2);                                                   % set the line width
title('{\itE{_z}}(15,10) for 0-200 ns', 'FontSize',18, 'Color',[0 0 0])  % title
xlabel('Time {\itt}  [seconds]','FontSize',16, 'Color',[0 0 0])          % x-axis label
set(Ax(1),'Ylabel',text('String','{\itE{_z}}({\itt})  [V/m]','FontSize',16, 'Color',[0 0 0]))
```

Optional Part

Tom Penick, EE 325K Homework 10

Problem:

Study the propagation loss due to a structure blocking direct transmission to the observation point.

Line current J_z at (2m,10m). Plot E_z at (18m, 10m) from 0 to 200 ns.

Source: $J_z(t) = 1 - \cos\left(\frac{\pi t}{T}\right)$

T = 10 ns, Δ = 0.1 m, Δt = 0.2 ns

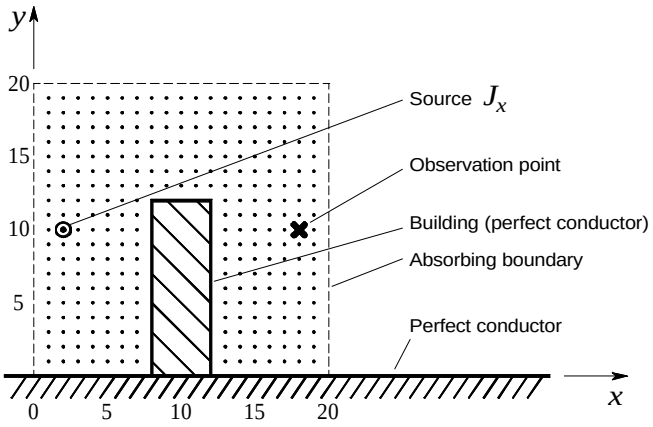
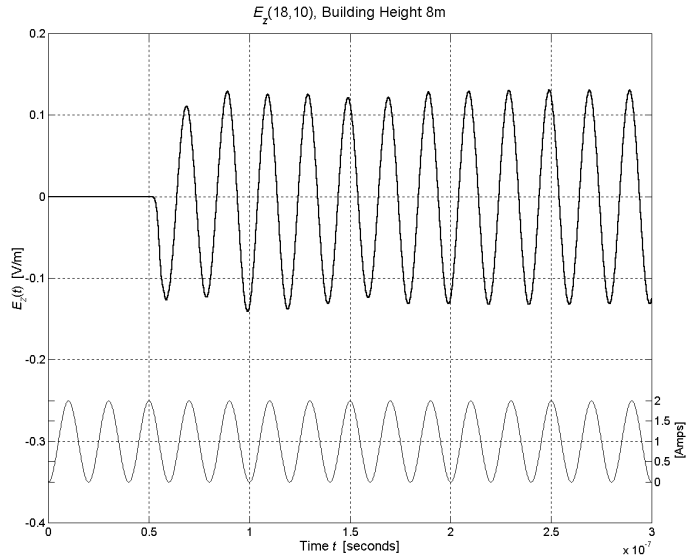
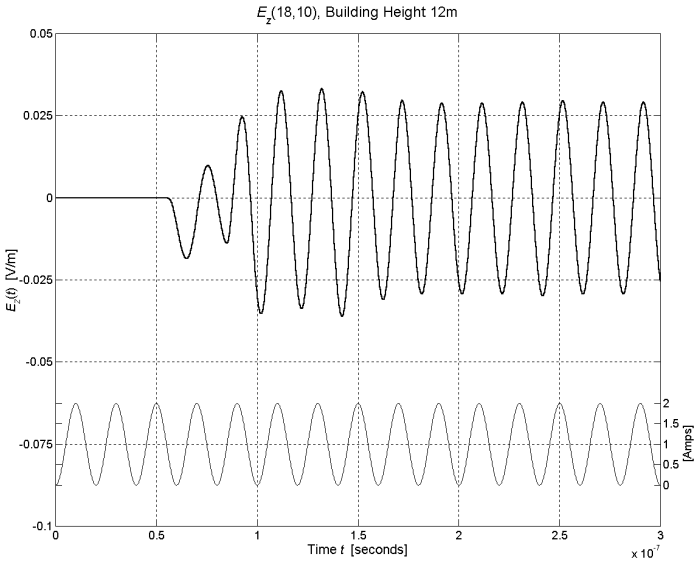
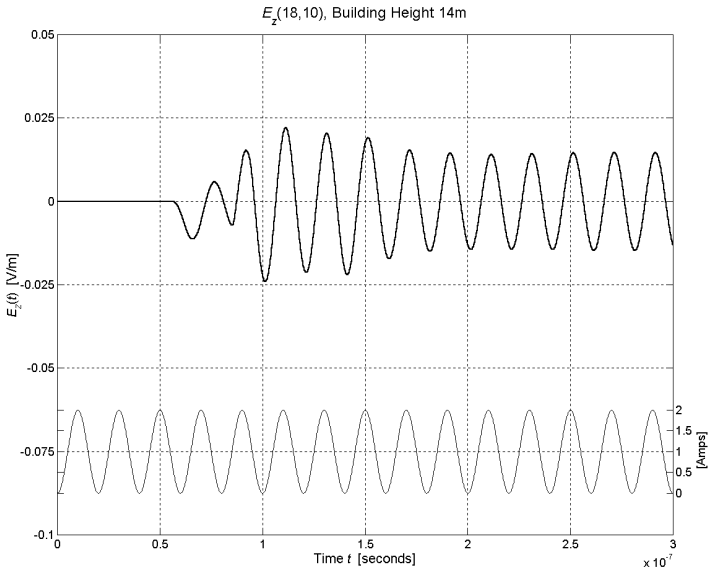
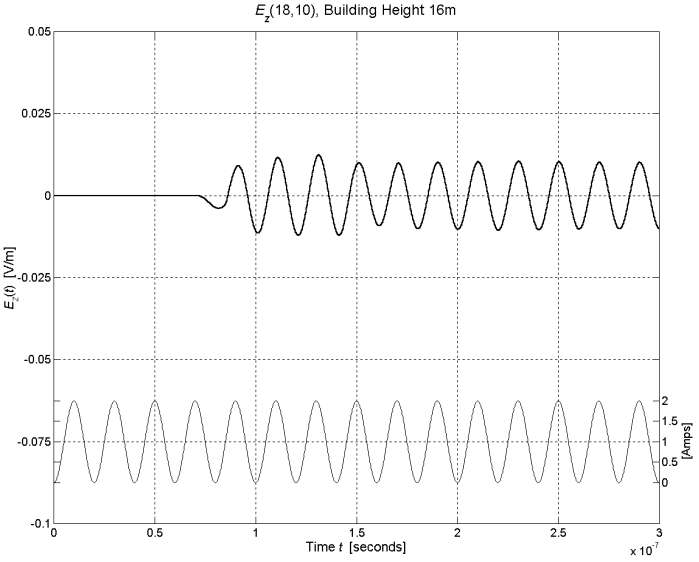


Diagram of the calculation area showing the obstructing building at its 12 meter height.









Propagation Loss Due to a Structure

For this experiment I have moved the source and observation points further apart to (2m,10m) and 18m,10m) respectively. The source current J_z is now a sinusoidal wave and the observation time period has been increased to 300 ns to allow the sinusoidal response to stabilize. A "building" has been erected between the source and observation point. The building is modeled as a perfect conductor. Its height is varied for the four plots to observe the result of the signal blockage.

The 2 A_{pp} source current is plotted in the lower portion of the graph and the response E_z is plotted in the upper portion. The scale of the first plot of E_z has been changed to accommodate the much higher amplitude of the unobstructed response.

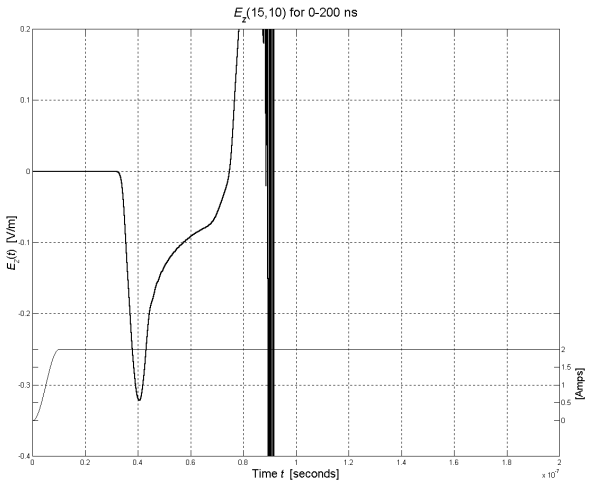
In the first plot shown on the preceding page, the building is only 8 meters high so that it does not obstruct the line of sight path between source and observation point. The observed electric field is 0.26 V_{pp}/m. When the building height is increased to 12 meters, 2 meters higher than the line of sight path, the observed voltage drops to 0.06 V_{pp}/m. The third and fourth plots are for building heights of 14 and 16 meters respectively. Steady state voltage response at the observation point is 0.03 V_{pp}/m and 0.02 V_{pp}/m for the two cases.

3D Observation of the Wave Encountering the Building

To observe the effects of the wave encountering the building, I returned the source J_z to its original pulse configuration and plotted 3D graphs of the pulse-induced wave encountering the side of the building. On the following page is a series of 15 time lapse views of the absolute value of the electric field from t = 12.5 ns to 47.5 ns. This appears as a mirror image of my original problem statement diagram, with the source now located on the right side of the building and the ground in the left foreground. It can be seen that the MUR1 absorbing boundary condition allows the electric field to flow across the boundary in the right foreground while the field remains anchored to zero at ground. A steep gradient appears next to the building and ground and a strong peak is observed at the upper corner of the building.

MUR2 Attempt

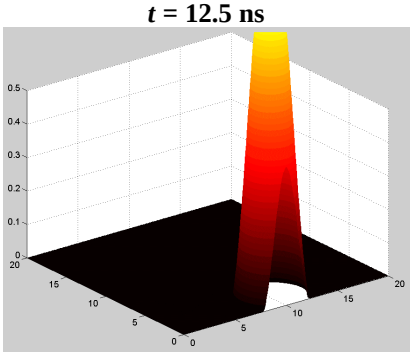
I revised the Matlab code for the original problem statement to employ the MUR2 method of handling absorbing boundary conditions as described by G. Mur in IEEE Trans. Electromagnetic Compatibility, vol. 23, pp. 377-382, Nov. 1981. This revision extended the execution time of the program from 93 seconds to 100 seconds. Unfortunately the result produced instability at t = 80 ns. I have tried experimenting with different values of Δt and δ, with no success.



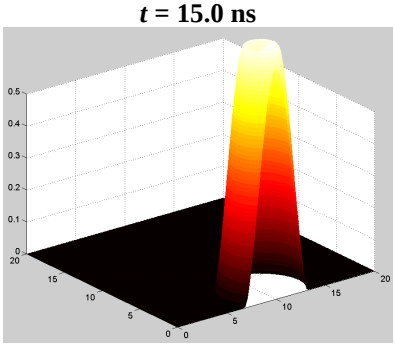
Implementation of MUR2, showing instability.

This is a 3D time lapse series of the absolute value of the electric field E_z in response to a 2-Amp 10 ns impulse J_z, showing the effect of the wave encountering the 16-meter building.

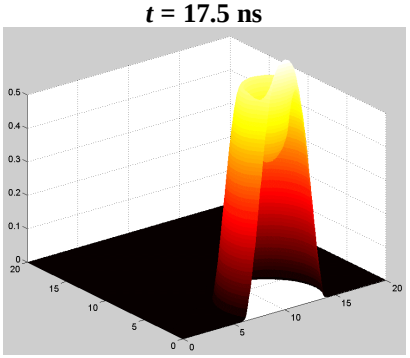
t = 12.5 ns



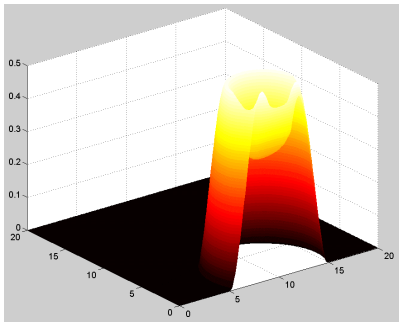
t = 15.0 ns



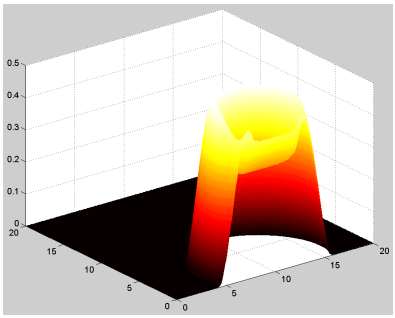
t = 17.5 ns



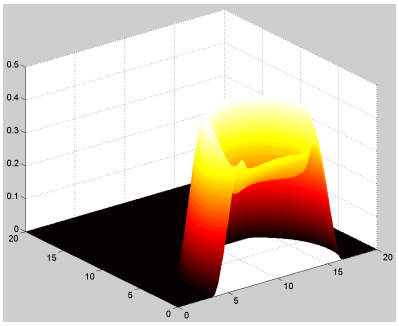
t = 20.0 ns



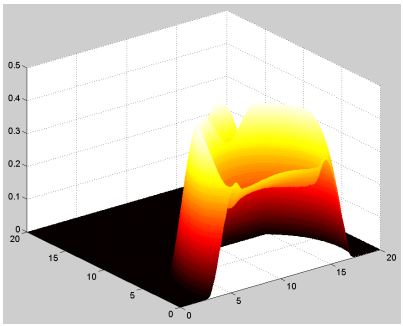
t = 22.5 ns



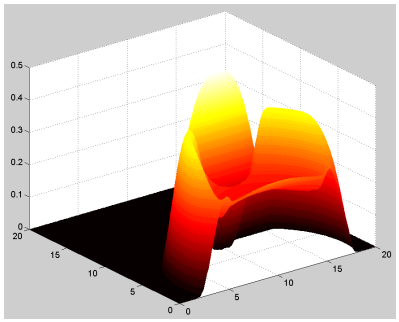
t = 25.0 ns



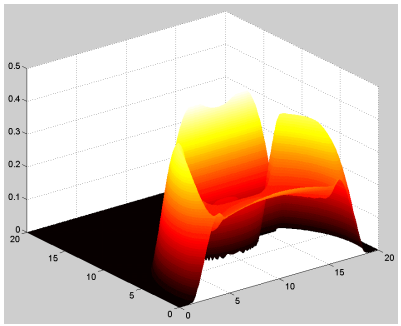
t = 27.5 ns



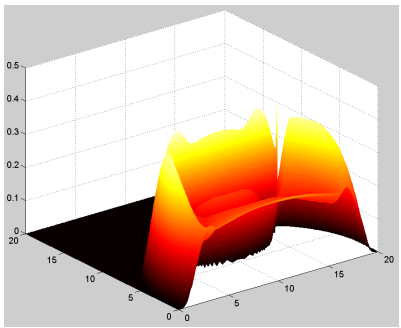
t = 30.0 ns



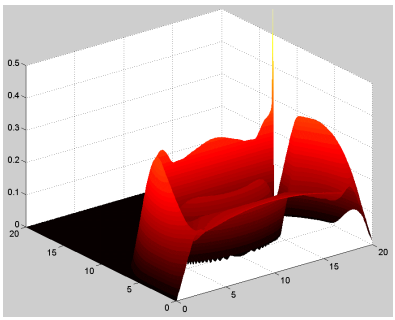
t = 32.5 ns



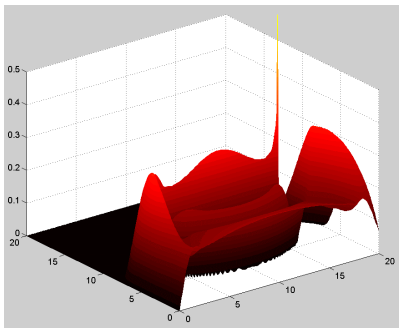
t = 35.0 ns



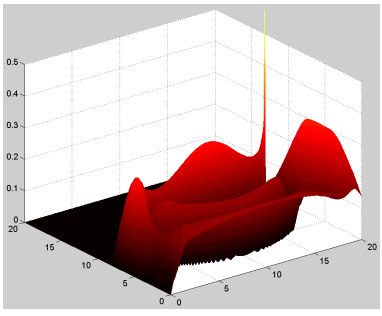
t = 37.5 ns



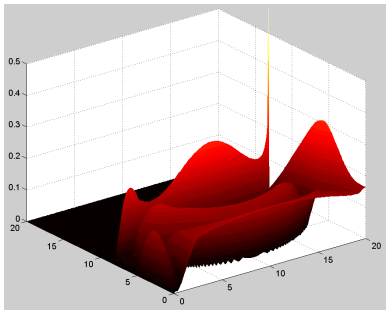
t = 40.0 ns



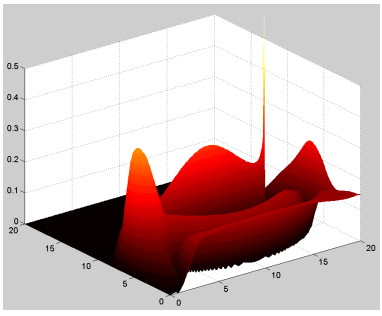
t = 42.5 ns



t = 45.0 ns



t = 47.5 ns



MATLAB CODE (Optional part – loss due to a building)

```
function FDT3()
%   Finite Difference Time Domain, EE325K HW 10, Optional part, by Tom Penick
%   This function plots Ez at (18m, 10m) from 0<t<300 ns in response
%   to signal Jz(t) = 1-cos(pi*t/T).  Jz(t) is located at (2m, 10m). A building
%   (perfect conductor) blocks the direct path between current source and
%   observation point.  The solution domain is (0-20m, 0-20m).
% INDEXING CONVENTIONS:  Since the problem calls for indices with some integer+1/2 values and Matlab uses
%   only whole number indexing, I need a plan.  For the first three expressions, the matrices calculated
%   are the interior matrices which do not include the boundary rows and columns.
%   (Those rows and columns are present, I just exclude them when calculating.) The other matrices in the
%   calculations are shifted as required with respect to these defining matrices.  Matrix row indices
%   correspond to graphical X positions and matrix column indices correspond to graphical Y positions,
%   e.g. column 1 corresponds to Y=0 and holds the values for the perfect conductor at the bottom of the
%   graphical representation (except for the Hx matrix where it's Y=1/2).
% HANDLING THE BUILDING:  I will allow field points to be calculated as before in function FDT2().  But
%   after each iteration dt, I will reset the E-field values on and within the perimeter of the building
%   lines to zero.  This will keep the H-field within the building at zero.

%*******************  VARIABLES, PARAMETERS, AND INITIAL CALCULATIONS  ***********************************
T = 10e-9; t = 0; TT = 300e-9;           % ramp time 10 ns, start time 0 s, plot duration 200 ns
JzLoc = [2 10]; EzLoc = [18 10];         % source current and observation point locations
Bldg = [8 12 16];                        % location parameters of the building [left right height]
sRange = [0 20]; sDomain = [0 20];       % solution domain, y=sRange x=sDomain
D = .1;                                  % spatial discretization, delta
dt = .2e-9                               % time discretization, delta t = 0.2 ns
EzObs = []; Time = []; JzPlt = [];       % matrices, observed Ez and time for plotting
uo = 1.25663706144e-6;                   % permiability of free space
eo = 8.85418781762e-12;                  % permittivity of free space
c = 299.792458e6;                        % speed of light
HiX = (sDomain(2)-sDomain(1))/D;         % one less than the number of X-values (rows)
HiY = (sRange(2)-sRange(1))/D;           % one less than the number of Y-values (columns)
Jz = zeros([HiX-1,HiY-1]);               % create current source matrix that doesn't include boundaries
Ez = zeros([HiX+1,HiY+1]);               % create electric field matrix that includes boundaries
Hx = zeros([HiX+1,HiY]);                 % create magnetic field matrix that includes left & right boundaries
Hy = zeros([HiX,HiY+1]);                 % create magnetic field matrix that includes upper & lower boundaries
Ezp=Ez;                                  % create matrix for present grid values, Ex-n
Courant = D/2^.5/c                       % Courant stability condition, must be > dt
                                         % Do some precalculations to speed this dog up.
M1=dt/D/uo; M2=dt/D/eo; M3=dt/eo; M4=(c*dt-D)/(c*dt+D); % some multipliers to be used later

%*****************************  THE EXPRESSIONS TO CALCULATE  ********************************************
ExprHx = 'Hx(2:HiX,1:HiY)=Hx(2:HiX,1:HiY)-M1*(Ezp(2:HiX,2:HiY+1)-Ezp(2:HiX,1:HiY));';
ExprHy = 'Hy(1:HiX,2:HiY)=Hy(1:HiX,2:HiY)+M1*(Ezp(2:HiX+1,2:HiY)-Ezp(1:HiX,2:HiY));';
ExprEz = 'Ez(2:HiX,2:HiY)=Ezp(2:HiX,2:HiY)+M2*(Hy(2:HiX,2:HiY)-Hy(1:HiX-1,2:HiY)-Hx(2:HiX,2:HiY)+Hx(2:HiX,1:HiY-
1))-M3.*Jz;';
MURlft = 'Ez(1,2:HiY)=Ezp(2,2:HiY)+M4*(Ez(2,2:HiY)-Ezp(1,2:HiY));';
MURrgt = 'Ez(HiX+1,2:HiY)=Ezp(HiX,2:HiY)+M4*(Ez(HiX,2:HiY)-Ezp(HiX+1,2:HiY));';
MURtop = 'Ez(1:HiX+1,HiY+1)=Ezp(1:HiX+1,HiY)+M4*(Ez(1:HiX+1,HiY)-Ezp(1:HiX+1,HiY+1));';

%*********************************  THE PROGRAM CODE  ************************************************
while t<=TT
  Jz(JzLoc(1)/D,JzLoc(2)/D)=1-cos(pi*t/T);   % Jz (current) is 2x2 smaller than the other matrices, but
  JzPlt = [JzPlt 1-cos(pi*t/T)];             % since it is the 0th row and column that are dropped, no
                                             % correction is needed to index the source current location.
  eval(ExprHx); eval(ExprHy); eval(ExprEz);  % evaluate the first three expressions
  eval(MURlft); eval(MURrgt); eval(MURtop);  % evaluate the MUR1 calculations for the perimeter
  EzObs = [EzObs Ez(EzLoc(1)/D+1,EzLoc(2)/D+1)]; % electric field at the observation point for plotting
  Time = [Time t]; t = t+dt;                 % save the time value for plotting, then increment the time
  Ezp = Ez;                                  % transfer the new values of Ex to the matrix for the past values
                                             % return the building E-field to zero
  Ezp(Bldg(1)/D+1:Bldg(2)/D+1,1:Bldg(3)/D+1) = zeros((Bldg(2)-Bldg(1))/D+1,(Bldg(3))/D+1);
end
```

```
%*************************  PLOT Ez VERSUS TIME  ************************************************************
figure('Position',[-10 -110 1700 1100])                                          % gimme a big window
[Ax,H1,H2] = plotyy(Time,EzObs,Time,JzPlt); grid on;                             % dual y-axis plot
%set(Ax(1),'Ylim',[-.4 .2],'Ytick',[-.4 -.3 -.2 -.1 0 .1 .2],'FontSize',14);   % left-hand y-axis settings
set(Ax(1),'Ylim',[-.1 .05],'Ytick',[-.1 -.075 -.05 -.025 0 .025 .05],'FontSize',14);   % left-hand y-axis
settings
set(Ax(2),'Ylim',[-1 11],'Ytick',[0 .5 1 1.5 2],'FontSize',14);                  % right-hand y-axis settings
set(H1,'LineWidth',2);                                                           % set the line width
title('{\itE{_z}}(18,10), Building Height 16m', 'FontSize',18, 'Color',[0 0 0])    % title
xlabel('Time {\itt}  [seconds]','FontSize',16, 'Color',[0 0 0])                  % x-axis label
set(Ax(1),'Ylabel',text('String','{\itE{_z}}({\itt})  [V/m]','FontSize',16, 'Color',[0 0 0]))
```

MATLAB CODE (Optional part – 3D Graphics)

```
function FDT4()
%   Finite Difference Time Domain, EE325K HW 10, Optional part, by Tom Penick
%   3D experiment
%   This function plots Ez at (18m, 10m) from 0<t<300 ns in response
%   to signal Jz(t) = 1-cos(pi*t/T).  Jz(t) is located at (2m, 10m). A building
%   (perfect conductor) blocks the direct path between current source and
%   observation point.  The solution domain is (0-20m, 0-20m).

% INDEXING CONVENTIONS:  Since the problem calls for indices with some integer+1/2 values and Matlab uses
%   only whole number indexing, we need a plan.  For the first three expressions, the matrices calculated
%   are the interior matrices which do not include the boundary rows and columns.
%   (Those rows and columns are present, we just exclude them when calculating.) The other matrices in the
%   calculations are shifted as required with respect to these defining matrices.  Matrix row indices
%   correspond to graphical X positions and matrix column indices correspond to graphical Y positions,
%   e.g. column 1 corresponds to Y=0 and holds the values for the perfect conductor at the bottom of the
%   graphical representation (except for the Hx matrix where it's Y=1/2).
% HANDLING THE BUILDING:  I will allow field points to be calculated as in the problem without the
%   building present as in function FDT2().  But after each iteration dt, I will reset the E-field values
%   on and within the perimeter of the building lines to zero.  This will keep the H-field within the
%   building at zero.

%******************  VARIABLES, PARAMETERS, AND INITIAL CALCULATIONS  ***********************************
T = 10e-9; t = 0; TT = 50e-9;           % ramp time 10 ns, start time 0 s, plot duration 200 ns
JzLoc = [2 10]; EzLoc = [18 10];         % source current and observation point locations
Bldg = [8 12 16];                        % location parameters of the building [left right height]
sRange = [0 20]; sDomain = [0 20];       % solution domain, y=sRange x=sDomain
D = .1;                                  % spatial discretization, delta
dt = .2e-9;                              % time discretization, delta t = 0.2 ns
EzObs = []; Time = []; JzPlt = [];       % matrices, observed Ez and time for plotting
uo = 1.25663706144e-6;                   % permiability of free space
eo = 8.85418781762e-12;                  % permittivity of free space
c = 299.792458e6;                        % speed of light
HiX = (sDomain(2)-sDomain(1))/D;         % one less than the number of X-values (rows)
HiY = (sRange(2)-sRange(1))/D;           % one less than the number of Y-values (columns)
Jz = zeros([HiX-1,HiY-1]);               % create current source matrix that doesn't include boundaries
Ez = zeros([HiX+1,HiY+1]);               % create electric field matrix that includes boundaries
Hx = zeros([HiX+1,HiY]);                 % create magnetic field matrix that includes left & right boundaries
Hy = zeros([HiX,HiY+1]);                 % create magnetic field matrix that includes upper & lower boundaries
Ezp=Ez;                                  % create matrix for present grid values, Ex-n
Courant = D/2^.5/c;                      % Courant stability condition, must be > dt
Repeat = 2.5e-9;
                                         % Do some precalculations to speed this dog up.
M1=dt/D/uo; M2=dt/D/eo; M3=dt/eo; M4=(c*dt-D)/(c*dt+D); % some multipliers to be used later

%****************************  THE EXPRESSIONS TO CALCULATE  ********************************************
ExprHx = 'Hx(2:HiX,1:HiY)=Hx(2:HiX,1:HiY)-M1*(Ezp(2:HiX,2:HiY+1)-Ezp(2:HiX,1:HiY));';
ExprHy = 'Hy(1:HiX,2:HiY)=Hy(1:HiX,2:HiY)+M1*(Ezp(2:HiX+1,2:HiY)-Ezp(1:HiX,2:HiY));';
ExprEz = 'Ez(2:HiX,2:HiY)=Ezp(2:HiX,2:HiY)+M2*(Hy(2:HiX,2:HiY)-Hy(1:HiX-1,2:HiY)-Hx(2:HiX,2:HiY)+Hx(2:HiX,1:HiY-
1))-M3.*Jz;';
MURlft = 'Ez(1,2:HiY)=Ezp(2,2:HiY)+M4*(Ez(2,2:HiY)-Ezp(1,2:HiY));';
MURrgt = 'Ez(HiX+1,2:HiY)=Ezp(HiX,2:HiY)+M4*(Ez(HiX,2:HiY)-Ezp(HiX+1,2:HiY));';
MURtop = 'Ez(1:HiX+1,HiY+1)=Ezp(1:HiX+1,HiY)+M4*(Ez(1:HiX+1,HiY)-Ezp(1:HiX+1,HiY+1));';
```

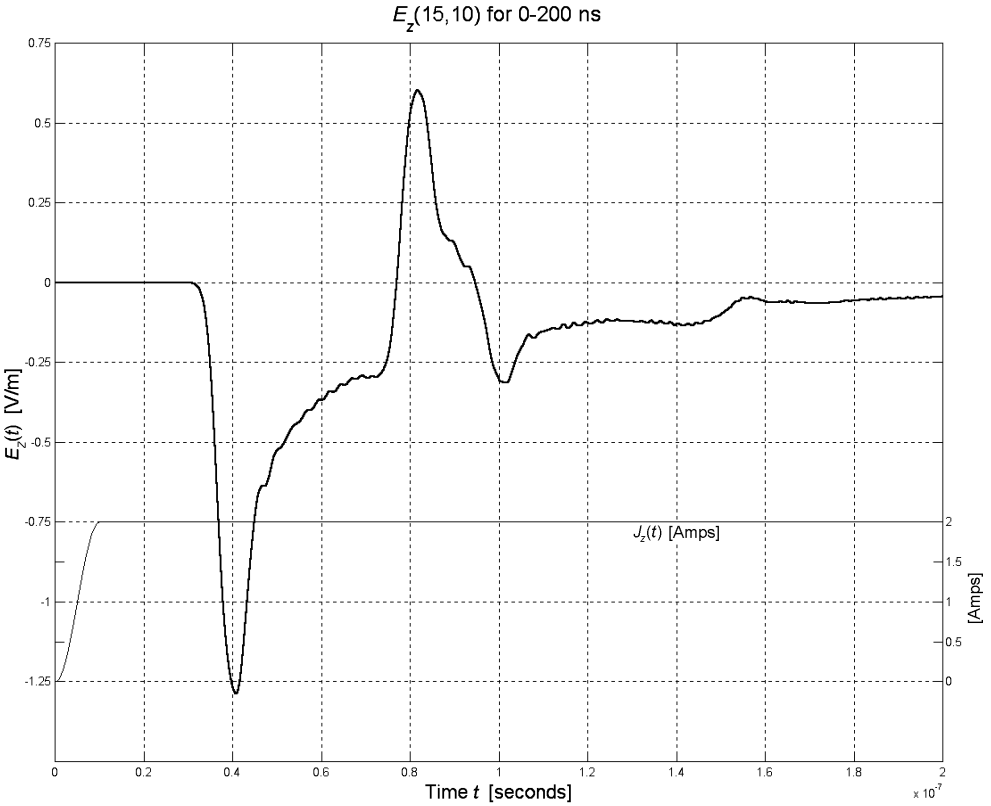
```
%*********************************  THE PROGRAM CODE  ******************************************************
Cnt=4;
while t<=TT
  if t <= T                                  % select the proper value for Jz
    Jz(JzLoc(1)/D,JzLoc(2)/D)=1-cos(pi*t/T); % Jz (current) is 2x2 smaller than the other matrices, but
    JzPlt = [JzPlt 1-cos(pi*t/T)];           % since it is the 0th row and column that are dropped, no
                                             % correction is needed to index the source current location.
  else
    Jz(JzLoc(1)/D,JzLoc(2)/D)=2;             % the source current, Jz, after t = T
    JzPlt = [JzPlt 2];                       % the source current, Jz, saved for plotting
  end
  eval(ExprHx); eval(ExprHy); eval(ExprEz);  % evaluate the first three expressions
  eval(MURlft); eval(MURrgt); eval(MURtop);  % evaluate the MUR1 calculations for the perimeter
  EzObs = [EzObs Ez(EzLoc(1)/D+1,EzLoc(2)/D+1)]; % electric field at the observation point for plotting
  if t >= Cnt*Repeat & t < Cnt*Repeat + dt   % create a series of 3D plots
    figure('Position',[-10 -110 1000 800])   % gimme a big window
    [X,Y] = meshgrid(0:.1:20);               % Create a grid
    surf(X,Y,abs(Ez))
    set(gca,'Zlim',[0 .5],'FontSize',14);    % z-axis settings
    colormap hot
    shading interp
    Cnt = Cnt+1;
  end
  Time = [Time t]; t = t+dt;                 % save the time value for plotting, then increment the time
  Ezp = Ez;                                  % transfer the new values of Ex to the matrix for past values
                                             % return the building E-field to zero
  Ezp(Bldg(1)/D+1:Bldg(2)/D+1,1:Bldg(3)/D+1) = zeros((Bldg(2)-Bldg(1))/D+1,(Bldg(3))/D+1);
end

%*************************  PLOT Ez VERSUS TIME  ***************************************************
figure('Position',[-10 -110 1100 800])                                    % gimme a big window
[Ax,H1,H2] = plotyy(Time,EzObs,Time,JzPlt); grid on;                      % dual y-axis plot
%set(Ax(1),'Ylim',[-.4 .2],'Ytick',[-.4 -.3 -.2 -.1 0 .1 .2],'FontSize',14); % left-hand y-axis settings
set(Ax(1),'Ylim',[-.1 .05],'Ytick',[-.1 -.075 -.05 -.025 0 .025 .05],'FontSize',14); % left-hand y-axis settings
set(Ax(2),'Ylim',[-1 11],'Ytick',[0 .5 1 1.5 2],'FontSize',14);           % right-hand y-axis settings
set(H1,'LineWidth',2);                                                    % set the line width
title('{\itE{_z}}(18,10), Building Height 16m', 'FontSize',18, 'Color',[0 0 0])    % title
xlabel('Time {\itt}  [seconds]','FontSize',16, 'Color',[0 0 0])           % x-axis label
set(Ax(1),'Ylabel',text('String','{\itE{_z}}({\itt})  [V/m]','FontSize',16, 'Color',[0 0 0]))
```

MATLAB CODE (Optional part – MUR2)

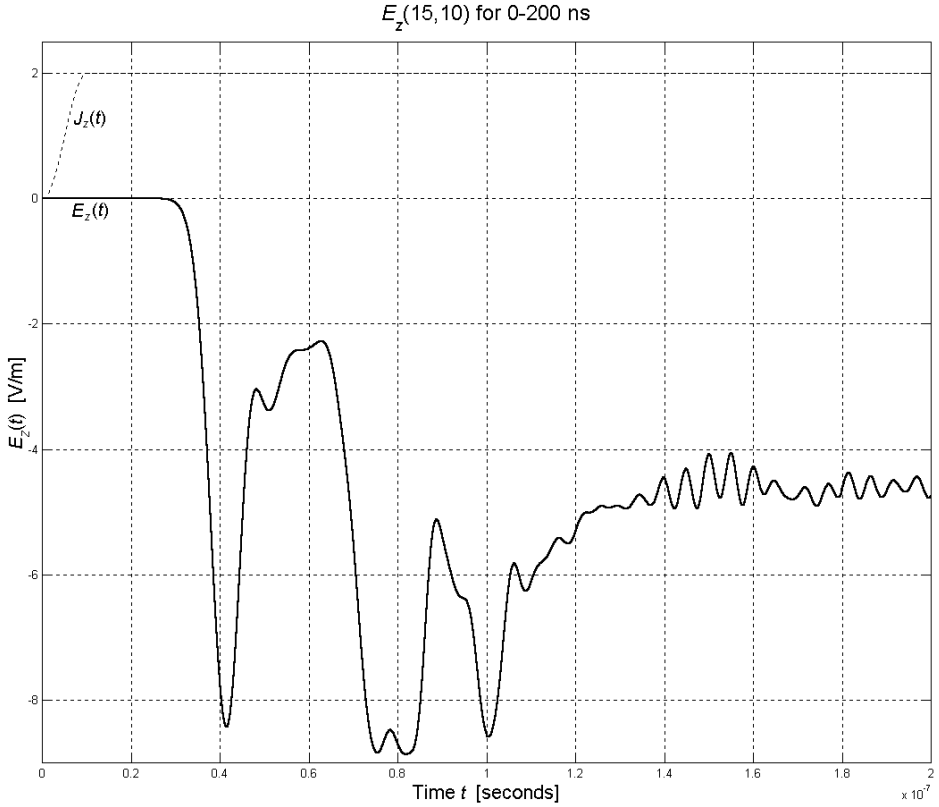
```
function FDT5()
%   Finite Difference Time Domain, EE325K HW 10, MUR2 Optional Part, by Tom Penick
%   This function plots Ez at (15m, 10m) from 0<t<200 ns in response
%   to signal Jz(t) = 0, t<0; = 1-cos(pi*t/T), 0<t<T; = 2, t>T.
%   Jz(t) is located at (5m, 10m). The solution domain is (0-20m, 0-20m)
%
% INDEXING CONVENTIONS:  Since the problem calls for indices with some integer+1/2 values and Matlab uses
%   only whole number indexing, we need a plan.  For the first three expressions, the matrices calculated
%   are the interior matrices which do not include the boundary rows and columns.
%   (Those rows and columns are present, we just exclude them when calculating.) The other matrices in the
%   calculations are shifted as required with respect to these defining matrices.  Matrix row indices
%   correspond to graphical X positions and matrix column indices correspond to graphical Y positions,
%   e.g. column 1 corresponds to Y=0 and holds the values for the perfect conductor at the bottom of the
%   graphical representation (except for the Hx matrix where it's Y=1/2).
```

```
%******************  VARIABLES, PARAMETERS, AND INITIAL CALCULATIONS  ************************************
T = 10e-9; t = 0; TT = 200e-9;           % ramp time 10 ns, start time 0 s, plot duration 200 ns
JzLoc = [5 10];                          % initial source current location
sRange = [0 20]; sDomain = [0 20];       % solution domain, y=sRange x=sDomain
D = .1;                                  % spatial discretization, delta
dt = .2e-9                               % time discretization, delta t = 0.2 ns
EzObs = []; Time = []; JzPlt = [];       % matrices, observed Ez and time for plotting
EzLoc = [15 10];                         % location of the observation point
uo = 1.25663706144e-6;                   % permiability of free space
eo = 8.85418781762e-12;                  % permittivity of free space
c = 299.792458e6;                        % speed of light
HiX = (sDomain(2)-sDomain(1))/D;         % one less than the number of X-values (rows)
HiY = (sRange(2)-sRange(1))/D;           % one less than the number of Y-values (columns)
Jz = zeros([HiX-1,HiY-1]);               % create current source matrix that doesn't include boundaries
Ez = zeros([HiX+1,HiY+1]);               % create electric field matrix that includes boundaries
Hx = zeros([HiX+1,HiY]);                 % create magnetic field matrix that includes left & right boundaries
Hy = zeros([HiX,HiY+1]);                 % create magnetic field matrix that includes upper & lower boundaries
Ezp=Ez;                                  % create matrix for present grid values, Ex-n
Courant = D/2^.5/c                       % Courant stability condition, must be > dt
                                         % Do some precalculations to speed this dog up.
M1=dt/D/uo; M2=dt/D/eo; M3=dt/eo; M4=(c*dt-D)/(c*dt+D); M5=uo*c/(2*(c*dt+D));% some multipliers to be used later

%*****************************  THE EXPRESSIONS TO CALCULATE  ********************************************
ExprHx = 'Hx(2:HiX,1:HiY)=Hx(2:HiX,1:HiY)-M1*(Ezp(2:HiX,2:HiY+1)-Ezp(2:HiX,1:HiY));';
ExprHy = 'Hy(1:HiX,2:HiY)=Hy(1:HiX,2:HiY)+M1*(Ezp(2:HiX+1,2:HiY)-Ezp(1:HiX,2:HiY));';
ExprEz = 'Ez(2:HiX,2:HiY)=Ezp(2:HiX,2:HiY)+M2*(Hy(2:HiX,2:HiY)-Hy(1:HiX-1,2:HiY)-Hx(2:HiX,2:HiY)+Hx(2:HiX,1:HiY-
1))-M3*Jz;';
MURlft = 'Ez(1,2:HiY)=Ezp(2,2:HiY)+M4*(Ez(2,2:HiY)-Ezp(1,2:HiY))-M5*(Hx(1,2:HiY)-Hx(1,1:HiY-1)+Hx(2,2:HiY)-
Hx(2,1:HiY-1));';
MURrgt = 'Ez(HiX+1,2:HiY)=Ezp(HiX,2:HiY)+M4*(Ez(HiX,2:HiY)-Ezp(HiX+1,2:HiY))-M5*(Hx(HiX+1,2:HiY)-Hx(HiX+1,1:HiY-
1)+Hx(HiX,2:HiY)-Hx(HiX,1:HiY-1));';
MURtop = 'Ez(2:HiX,HiY+1)=Ezp(2:HiX,HiY)+M4*(Ez(2:HiX,HiY)-Ezp(2:HiX,HiY+1))-M5*(Hy(2:HiX,HiY+1)-Hy(1:HiX-
1,HiY+1)+Hy(2:HiX,HiY)-Hy(1:HiX-1,HiY));';
Corners = 'Ez(1,HiY+1)=Ez(2,HiY); Ez(HiX+1,HiY+1)=Ez(HiX,HiY);';  % Copy the adjacent diagonals to the corners

%*********************************  THE PROGRAM CODE  ****************************************************
while t<=TT
  if t <= T                                % select the proper value for Jz
    Jz(JzLoc(1)/D,JzLoc(2)/D)=1-cos(pi*t/T); % Jz (current) is 2x2 smaller than the other matrices, but
    JzPlt = [JzPlt 1-cos(pi*t/T)];         % since it is the 0th row and column that are dropped, no
                                           % correction is needed to index the source current location.
  else
    Jz(JzLoc(1)/D,JzLoc(2)/D)=2;           % the source current, Jz, after t = T
    JzPlt = [JzPlt 2];                     % the source current, Jz, saved for plotting
  end
  eval(ExprHx); eval(ExprHy); eval(ExprEz);  % evaluate the first three expressions
  eval(MURlft); eval(MURrgt); eval(MURtop); eval(Corners); % evaluate the MUR2 calculations for the perimeter
  EzObs = [EzObs Ez(EzLoc(1)/D+1,EzLoc(2)/D+1)]; % electric field at the observation point for plotting
  Time = [Time t]; t = t+dt;               % save the time value for plotting, then increment the time
  Ezp = Ez;                                % transfer the new Ex values to the matrix for the past values
end

%**************************  PLOT Ez VERSUS TIME  ************************************************************
figure('Position',[-10 -110 1700 1100])                                    % gimme a big window
[Ax,H1,H2] = plotyy(Time,EzObs,Time,JzPlt); grid on;                       % dual y-axis plot
set(Ax(1),'Ylim',[-.4 .2],'Ytick',[-.4 -.3 -.2 -.1 0 .1 .2]);              % left-hand y-axis settings
set(Ax(2),'Ylim',[-1 11],'Ytick',[0 .5 1 1.5 2]);                          % right-hand y-axis settings
set(H1,'LineWidth',2);                                                     % set the line width
title('{\itE{_z}}(15,10) for 0-200 ns', 'FontSize',18, 'Color',[0 0 0])    % title
xlabel('Time {\itt}  [seconds]','FontSize',16, 'Color',[0 0 0])            % x-axis label
set(Ax(1),'Ylabel',text('String','{\itE{_z}}({\itt})  [V/m]','FontSize',16, 'Color',[0 0 0]))
```

Used matrix calculations, but with D=0.2



Old one done with individual, looped calculations with D=0.5



MATLAB CODE (looped version)

```
function FDTD()
%   Finite Difference Time Domain, EE325K HW 10, by Tom Penick
%   This function plots Ez at (15m, 10m) from 0<t<200 ns in response
%   to signal Jz(t) = 0, t<0; = 1-cos(pi*t/T), 0<t<T; = 2, t>T.
%   Jz(t) is located at (5m, 10m). The solution domain is (0-20m, 0-20m)
%   This function takes a looooooooooooooong time to run!


%******************  VARIABLES, PARAMETERS, AND INITIAL CALCULATIONS  ************************************

T = 10e-9; t = 0; TT = 200e-9;           % ramp time 10 ns, start time 0 s, plot duration 200 ns
Jz = 0; JzLoc = [5 10];                  % initial current and location
sRange = [0 20]; sDomain = [0 20];       % solution domain, y=sRange x=sDomain
D = .5;                                  % spatial discretization, delta
dt = .2e-9                               % time discretization, delta t = 0.2 ns
EzObs = []; Time = []; JzPlt = [];       % matrices, observed Ez and time for plotting
EzLoc = [15 10];                         % location of the observation point
uo = 1.25663706144e-6;                   % permiability of free space
eo = 8.85418781762e-12;                  % permittivity of free space
c = 299.792458e6;                        % speed of light
Ez = zeros([(sDomain(2)-sDomain(1))/D+1,(sRange(2)-sRange(1))/D+1]);
Hx=Ez; Hy=Ez;                            % create matrices for future grid values, n+½, n+1
Ezp=Ez; Hxp=Ez; Hyp=Ez;                  % create matrices for past and present grid values, n-½, n
iNdx = 2;                                % matrix index for i and i-½
jNdx = 2;                                % matrix index for j and j-½
Courant = D/2^.5/c                       % Courant stability condition, must be > dt
                                         % Do some precalculations to speed this dog up.
M1=dt/D/uo; M2=dt/D/eo; M3=dt/eo; M4=(c*dt-D)/(c*dt+D); % some multipliers to be used later


%*****************************  THE EXPRESSIONS TO CALCULATE  ********************************************

ExprHx = 'Hx(iNdx,jNdx+1)=Hxp(iNdx,jNdx+1)-M1*(Ezp(iNdx,jNdx+1)-Ezp(iNdx,jNdx));';
ExprHy = 'Hy(iNdx+1,jNdx)=Hyp(iNdx+1,jNdx)+M1*(Ezp(iNdx+1,jNdx)-Ezp(iNdx,jNdx));';
ExprEz = 'Ez(iNdx,jNdx)=Ezp(iNdx,jNdx)+M2*(Hy(iNdx+1,jNdx)-Hy(iNdx,jNdx)-Hx(iNdx,jNdx+1)+Hx(iNdx,jNdx))-Jz;';
MURlft = 'Ez(iNdx,jNdx)=Ezp(iNdx+1,jNdx)+M4*(Ez(iNdx+1,jNdx)-Ezp(iNdx,jNdx));';
MURrgt = 'Ez(iNdx,jNdx)=Ezp(iNdx-1,jNdx)+M4*(Ez(iNdx-1,jNdx)-Ezp(iNdx,jNdx));';
MURtop = 'Ez(iNdx,jNdx)=Ezp(iNdx,jNdx-1)+M4*(Ez(iNdx,jNdx-1)-Ezp(iNdx,jNdx));';


%*********************************  THE PROGRAM CODE  ****************************************************

JzLoc = JzLoc/D+1;      % convert the location of the source from meters to values for matrix indices
EzLoc = EzLoc/D+1;      % convert the location of observation point from meters to values for matrix indices
while t<=TT
  while iNdx < size(Ez,1)               % iNdx is associated with x values and matrix rows
    while jNdx < size(Ez,2)             % jNdx is associated with y values and matrix columns
      eval(ExprHx); eval(ExprHy);       % evaluate the expressions for magnetic field
      if JzLoc == [iNdx jNdx]           % look for the source point
        if t <= T                       % select the proper value for Jz
          Jz = (1-cos(pi*t/T));         % the current, Jz
        else
          Jz = 2;                       % the current, Jz
        end
        JzPlt = [JzPlt Jz]; Jz = Jz*M3;  % save Jz for plotting and apply the multiplier dt/eo
      else
        Jz = 0;                         % we are not at the source
      end
      eval(ExprEz); jNdx = jNdx+1;      % evaluate the electric field and increment the j index
    end
    jNdx = 2; iNdx = iNdx+1;            % reset the j index, increment the i index
  end
```

```
                                    % Now do the MUR1 stuff
  iNdx = 1; jNdx = 2;               % reset the indices to prepare for the left side
  while jNdx < size(Ez,2)           % do the left side from one above the bottom to one below the top
    eval(MURlft); jNdx = jNdx+1;    % get Ez and increment j index
  end
  jNdx = 2; iNdx = size(Ez,1);      % reset the indices to prepare for the right side
  while jNdx < size(Ez,2)           % do the right side from one above the bottom to one below the top
    eval(MURrgt); jNdx = jNdx+1;    % get Ez and increment j index
  end
  iNdx = 1; jNdx = size(Ez,2);      % reset the indices to prepare for the top side
  while iNdx <= size(Ez,1)          % get Ez all the way across the top boundary
    eval(MURtop); iNdx = iNdx+1;    % get Ez and increment i index
  end
  EzObs = [EzObs Ez(EzLoc(1),EzLoc(2))]; % save the electric field at the observation point for plotting
  Time = [Time t];                  % save the time value for plotting
  Ezp = Ez; Hxp = Hx; Hyp = Hy;     % transfer the new values to the matrices for the past values
  t = t+dt; iNdx = 2; jNdx = 2;     % increment the time, reset the indices
end


%*************************  PLOT Ez VERSUS TIME  ***********************************************************

figure('Position',[-10 -110 1700 1100])  % gimme a big window
plot(Time,EzObs,'LineWidth',2); grid on; % create plot, set linewidth
hold on, plot(Time,JzPlt,':'), hold off
ylimits = [-9 2.5];                 % set the Y axis limits
set(gca,'Ylim',ylimits);            % apply the specified range
title('{\itE{_z}}(15,10) for 0-200 ns', 'FontSize',18, 'Color',[0 0 0])
xlabel('Time {\itt}  [seconds]','FontSize',16, 'Color',[0 0 0])
ylabel('{\itE{_z}}({\itt})  [V/m]','FontSize',16, 'Color',[0 0 0])
```