Table of Contents

Contents

Chapter 1

GIF Animation

```
% Chapter1_GIFanimation.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% UNITS
degrees = pi/180;

% DASHBOARD GRID
lam0  = 0.1;
theta = 30*degrees;
Sx    = 1;
Sy    = 1;
Nx    = 100;
Ny    = 100;

NFRAMES = 40;
gif_name = 'FDFD_animation.gif';

% CALCULATE MESHGRID
dx = Sx/Nx;
xa = [1:Nx]*dx;
xa = xa - mean(xa);
dy = Sy/Ny;
ya = [1:Ny]*dy;
ya = ya - mean(ya);
[Y,X] = meshgrid(ya,xa);

% CALCULATE A WAVE (REPLACES FDFD SIMULATION)
k0 = 2*pi/lam0;
kx = k0*sin(theta);
ky = k0*cos(theta);
f  = exp(-1i*(kx*X + ky*Y));

% VISUALIZE USING IMAGESC
for nframe = 1 : NFRAMES

    % Add Phase
    phase = 2*pi*nframe/NFRAMES;
```

```
    fi = f*exp(1i*phase);

    % Draw Field
    pcolor(xa,ya,real(fi).');
    shading interp;
    axis equal tight off;
    set(gca,'YDir','reverse');
    drawnow;

    % Capture Frame
    F = getframe(gca);
    F = frame2im(F);
    [ind,cmap] = rgb2ind(F,256,'nodither');

    % Add Frame to GIF
    if nframe == 1
        imwrite(ind,cmap,gif_name,'gif',...
                'DelayTime',0,'Loopcount',inf);
    else
        imwrite(ind,cmap,gif_name,'gif',...
                'DelayTime',0,'WriteMode','append');
    end
end
```

Chapter 3

One-Dimensional Finite-Difference Method

```
% Chapter3_fdm1d.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% DEFINE AND CALCULATE GRID
a  = 0;
b  = 10;
Nx = 100;
xa = linspace(a,b,Nx);
dx = xa(2) - xa(1);

% BUILD DERIVATIVE MATRIX DX
d  = ones(Nx,1)/(2*dx);
DX = spdiags(-d,-1,sparse(Nx,Nx));
DX = spdiags(+d,+1,DX);
DX(Nx,Nx-1:Nx) = [-1 +1]/dx;

% BUILD MATRIX EQUATION A*f=b
A = DX;
b = -(1/3)*ones(Nx,1);

% ADD BOUNDARY VALUE f(0)=1
A(1,:) = 0;
A(1,1) = 1;
b(1)   = 1;
```

```
29
30  % SOLVE PROBLEM
31  f = A\b;
32
33  % PLOT SOLUTION
34  plot(xa,f);
35  axis equal tight;
```

Multi-Variable Finite-Difference Method

```
1   % Chapter3_mvfdm.m
2
3   % INITIALIZE MATLAB
4   close all;
5   clc;
6   clear all;
7
8   % GRID
9   a  = 0;
10  b  = 10;
11  Nx = 200;
12  dx = (b - a)/Nx;
13  xa = [0.5:Nx-0.5]*dx;
14
15  % BUILD DERIVATIVE MATRICES
16  d   = ones(Nx,1)/dx;
17
18  DFX = sparse(Nx,Nx);
19  DFX = spdiags(-d,0,DFX);
20  DFX = spdiags(+d,+1,DFX);
21
22  DGX = -DFX';
23
24  % BUILD MATRIX EQUATION
25  I = speye(Nx,Nx);
26  A = DGX*DFX + 6*I;
27  b = zeros(Nx,1);
28
29  % ADD BOUNDARY VALUES
30  A(1,:) = 0;
31  A(1,1) = 1;
32  b(1)   = 10;
33
34  A(Nx,:)  = 0;
35  A(Nx,Nx) = 1;
36  b(Nx)    = 1;
37
38  % SOLVE
39  f = A\b;
40  g = -(1/3)*DFX*f;
41
42  % PLOT RESULTS
43  plot(xa,f,'-k','LineWidth',2);
44  hold on;
45  plot(xa,g,'--k','LineWidth',2);
46  hold off;
```

```
47  xlabel('$x$','Interpreter','LaTex');
48  legend('$f(x)$','$g(x)$',...
49         'Interpreter','LaTex',...
50         'Location','NorthEastOutside');
51  set(gca,'FontSize',18,'LineWidth',2);
```

Chapter 4

Demonstrate YEEDER2D()

```
1   % Chapter4_yeeder2d_demo.m
2
3   % INITIALIZE MATLAB
4   close all;
5   clc;
6   clear all;
7
8   % DEFINE INPUT ARGUMENTS FOR YEEDER2D
9   NS  = [3 4];
10  RES = [0.2 0.1];
11  BC  = [0 0];
12
13  % CALL YEEDER2D
14  [DEX,DEY,DHX,DHY] = yeeder2d(NS,RES,BC);
15
16  % SHOW DERIVATIVE MATRICES
17  disp('DEX = ');
18  disp(full(DEX))
19
20  disp('DEY = ');
21  disp(full(DEY))
22
23  disp('DHX = ');
24  disp(full(DHX))
25
26  disp('DHY = ');
27  disp(full(DHY));
```

YEEDER2D() – Two-Dimensional Derivative Matrices on Yee Grid

```
1    function [DEX,DEY,DHX,DHY] = yeeder2d(NS,RES,BC,kinc)
2    % YEEDER2D      Derivative Matrices on a 2D Yee Grid
3    %
4    % [DEX,DEY,DHX,DHY] = yeeder2d(NS,RES,BC,kinc);
5    %
6    % INPUT ARGUMENTS
7    % =================
8    % NS    [Nx Ny] Grid Size
9    % RES   [dx dy] Grid Resolution
10   % BC    [xbc ybc] Boundary Conditions
11   %         0: Dirichlet boundary conditions
12   %         1: Periodic boundary conditions
13   % kinc  [kx ky] Incident Wave Vector
14   %       This argument is only needed for PBCs.
15   %
16   % Note: For normalized grids use k0*RES and kinc/k0
17   %
```

```
% OUTPUT ARGUMENTS
% =================
% DEX   Derivative Matrix wrt x for Electric Fields
% DEY   Derivative Matrix wrt to y for Electric Fields
% DHX   Derivative Matrix wrt to x for Magnetic Fields
% DHY   Derivative Matrix wrt to y for Magnetic Fields

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% HANDLE INPUT ARGUMENTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% EXTRACT GRID PARAMETERS
Nx = NS(1);     dx = RES(1);
Ny = NS(2);     dy = RES(2);

% DEFAULT KINC
if ~exist('kinc')
    kinc = [0 0];
end

% DETERMINE MATRIX SIZE
M = Nx*Ny;

% ZERO MATRIX
Z = sparse(M,M);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD DEX
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% HANDLE IF SIZE IS 1 CELL
if Nx==1
    DEX = -1i*kinc(1)*speye(M,M);

% HANDLE ALL OTHER CASES
else

    % Center Diagonal
    d0 = -ones(M,1);

    % Upper Diagonal
    d1 = ones(M,1);
    d1(Nx+1:Nx:M) = 0;

    % Build Derivative Matrix with Dirichlet BCs
    DEX = spdiags([d0 d1]/dx,[0 1],Z);

    % Incorporate Periodic Boundary Conditions
    if BC(1)==1
        d1 = zeros(M,1);
        d1(1:Nx:M) = exp(-1i*kinc(1)*Nx*dx)/dx;
        DEX = spdiags(d1,1-Nx,DEX);
    end

end

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD DEY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% HANDLE IF SIZE IS 1 CELL
if Ny==1
    DEY = -1i*kinc(2)*speye(M,M);

% HANDLE ALL OTHER CASES
else

    % Center Diagonal
    d0 = -ones(M,1);

    % Upper Diagonal
    d1 = ones(M,1);

    % Build Derivative Matrix with Dirichlet BCs
    DEY = spdiags([d0 d1]/dy,[0 Nx],Z);

    % Incorporate Periodic Boundary Conditions
    if BC(2)==1
        d1 = (exp(-1i*kinc(2)*Ny*dy)/dy)*ones(M,1);
        DEY = spdiags(d1,Nx-M,DEY);
    end

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD DHX AND DHY
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

DHX = -DEX';
DHY = -DEY';
```

Demonstrate YEEDER3D()

```
% Chapter4_yeeder3d_demo.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% DEFINE INPUT ARGUMENTS FOR YEEDER2D
NS  = [2 2 2];
RES = [0.3 0.2 0.1];
BC  = [0 0 0];

% CALL YEEDER3D
[DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(NS,RES,BC);

% SHOW DERIVATIVE MATRICES
disp('DEX = ');
disp(full(DEX));

```

```
disp('DEY = ');
disp(full(DEY));

disp('DEY = ');
disp(full(DEZ));

disp('DHX = ');
disp(full(DHX));

disp('DHY = ');
disp(full(DHY));

disp('DHZ = ');
disp(full(DHZ));
```

YEEDER3D() – Three-Dimensional Derivative Matrices on Yee Grid

```
function [DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(NS,RES,BC,kinc)
% YEEDER3D      Derivative Matrices on a 3D Yee Grid
%
% [DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(NS,RES,BC,kinc);
%
% INPUT ARGUMENTS
% ================
% NS    [Nx Ny Nz] Grid Size
% RES   [dx dy dz] Grid Resolution
% BC    [xbc ybc zbc] Boundary Conditions
%         0: Dirichlet boundary conditions
%         1: Periodic boundary conditions
% kinc  [kx ky kz] Incident Wave Vector
%       This argument is only needed for PBCs.
%
% Note: For normalized grids use k0*RES and kinc/k0
%
% OUTPUT ARGUMENTS
% ================
% DEX   Derivative Matrix wrt x for Electric Fields
% DEY   Derivative Matrix wrt y for Electric Fields
% DEZ   Derivative Matrix wrt z for Electric Fields
% DHX   Derivative Matrix wrt x for Magnetic Fields
% DHY   Derivative Matrix wrt y for Magnetic Fields
% DHZ   Derivative Matrix wrt z for Magnetic Fields

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% HANDLE INPUT ARGUMENTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% EXTRACT GRID PARAMETERS
Nx = NS(1);     dx = RES(1);
Ny = NS(2);     dy = RES(2);
Nz = NS(3);     dz = RES(3);

% DEFAULT KINC
if ~exist('kinc')
    kinc = [0 0 0];
end
```

```
40
41  % DETERMINE MATRIX SIZE
42  M = Nx*Ny*Nz;
43
44  % ZERO MATRIX
45  Z = sparse(M,M);
46
47  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
48  %% BUILD DEX
49  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50
51  % HANDLE IF SIZE IS 1 CELL
52  if Nx==1
53      DEX = -1i*kinc(1)*speye(M,M);
54
55  % HANDLE ALL OTHER CASES
56  else
57
58      % Center Diagonal
59      d0 = -ones(M,1);
60
61      % Upper Diagonal
62      d1 = ones(M,1);
63      d1(Nx+1:Nx:M) = 0;
64
65      % Build Derivative Matrix with Dirichlet BCs
66      DEX = spdiags([d0 d1]/dx,[0 1],Z);
67
68      % Incorporate Periodic Boundary Conditions
69      if BC(1)==1
70          d1 = zeros(M,1);
71          d1(1:Nx:M) = exp(-1i*kinc(1)*Nx*dx)/dx;
72          DEX = spdiags(d1,1-Nx,DEX);
73      end
74
75  end
76
77  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78  %% BUILD DEY
79  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80
81  % HANDLE IF SIZE IS 1 CELL
82  if Ny==1
83      DEY = -1i*kinc(2)*speye(M,M);
84
85  % HANDLE ALL OTHER CASES
86  else
87
88      % Center Diagonal
89      d0 = -ones(M,1);
90
91      % Upper Diagonal
92      d1 = [ ones((Ny-1)*Nx,1) ; zeros(Nx,1) ];
93      d1 = repmat(d1,[Nz-1 1]);
94      d1 = [ zeros(Nx,1) ; d1 ; ones((Ny-1)*Nx,1) ];
95
```

```
96        % Build Derivative Matrix with Dirichlet BCs
97        DEY = spdiags([d0 d1]/dy,[0 Nx],Z);
98
99        % Incorporate Periodic Boundary Conditions
100       if BC(2)==1
101           ph = exp(-1i*kinc(2)*Ny*dy)/dy;
102
103           d1 = [ ones(Nx,1) ; zeros((Ny-1)*Nx,1) ];
104           d1 = repmat(d1,[Nz-1 1]);
105           d1 = [ d1 ; ones((Ny-1)*Nx,1) ; zeros(Nx,1) ];
106
107           DEY = spdiags(ph*d1,-Nx*(Ny-1),DEY);
108       end
109
110   end
111
112   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
113   %% BUILD DEZ
114   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
115
116   % HANDLE IF SIZE IS 1 CELL
117   if Nz==1
118       DEZ = -1i*kinc(3)*speye(M,M);
119
120   % HANDLE ALL OTHER CASES
121   else
122
123       % Center Diagonal
124       d0 = ones(M,1);
125
126       % Build Derivative Matrix with Dirichlet BCs
127       DEZ = spdiags([-d0 +d0]/dz,[0 Nx*Ny],Z);
128
129       % Incorporate Periodic Boundary Conditions
130       if BC(3)==1
131           d0  = (exp(-1i*kinc(3)*Nz*dz)/dz)*ones(M,1);
132           DEZ = spdiags(d0,-Nx*Ny*(Nz-1),DEZ);
133       end
134
135   end
136
137   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
138   %% BUILD DHX, DHY AND DHZ
139   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
140
141   DHX = -DEX';
142   DHY = -DEY';
143   DHZ = -DEZ';
```

Chapter 5

Demonstrate ADDUPML2D()

```
1   % Chapter5_addumpl2d_demo.m
2
3   % INITIALIZE MATLAB
```

```
close all;
clc;
clear all;

% DEFINE GRID
Nx = 5;
Ny = 5;

% DEFINE PML
NPML = [2 2 2 2];

% BUILD ER2 AND UR2 ARRAYS
ER2 = ones(2*Nx,2*Ny);
UR2 = ones(2*Nx,2*Ny);

% CALL ADDUPML2D
[ERxx,ERyy,ERzz,URxx,URyy,URzz] ...
               = addupml2d(ER2,UR2,NPML);

% DISPLAY THE RESULTS
disp('ERxx =');
disp(ERxx);

disp('ERyy =');
disp(ERyy);

disp('ERzz =');
disp(ERzz);

disp('URxx =');
disp(URxx);

disp('URyy =');
disp(URyy);

disp('URzz =');
disp(URzz);
```

ADDUPML2D() – Add Two-Dimensional Uniaxial Perfectly Matched Layer to Grid

```
function [ERxx,ERyy,ERzz,URxx,URyy,URzz] = addupml2d(ER2,UR2,NPML)
% ADDUPML2D     Add UPML to a 2D Yee Grid
%
% [ERxx,ERyy,ERzz,URxx,URyy,URzz] = addupml2d(ER2,UR2,NPML);
%
% INPUT ARGUMENTS
% ================
% ER2       Relative Permittivity on 2x Grid
% UR2       Relative Permeability on 2x Grid
% NPML      [NXLO NXHI NYLO NYHI] Size of UPML on 1x Grid
%
% OUTPUT ARGUMENTS
% ================
% ERxx      xx Tensor Element for Relative Permittivity
% ERyy      yy Tensor Element for Relative Permittivity
% ERzz      zz Tensor Element for Relative Permittivity
```

```
% URxx      xx Tensor Element for Relative Permeability
% URyy      yy Tensor Element for Relative Permeability
% URzz      zz Tensor Element for Relative Permeability

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INITIALIZE FUNCTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE PML PARAMETERS
amax = 4;
cmax = 1;
p    = 3;

% EXTRACT GRID PARAMETERS
[Nx2,Ny2] = size(ER2);

% EXTRACT PML PARAMETERS
NXLO = 2*NPML(1);
NXHI = 2*NPML(2);
NYLO = 2*NPML(3);
NYHI = 2*NPML(4);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE PML PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE PML PARAMETERS TO PROBLEM SPACE
sx = ones(Nx2,Ny2);
sy = ones(Nx2,Ny2);

% ADD XLO PML
for nx = 1 : NXLO
    ax = 1 + (amax - 1)*(nx/NXLO)^p;
    cx = cmax*sin(0.5*pi*nx/NXLO)^2;
    sx(NXLO - nx + 1,:) = ax*(1 - 1i*60*cx);
end

% ADD XHI PML
for nx = 1 : NXHI
    ax = 1 + (amax - 1)*(nx/NXHI)^p;
    cx = cmax*sin(0.5*pi*nx/NXHI)^2;
    sx(Nx2 - NXHI + nx,:) = ax*(1 - 1i*60*cx);
end

% ADD YLO PML
for ny = 1 : NYLO
    ay = 1 + (amax - 1)*(ny/NYLO)^p;
    cy = cmax*sin(0.5*pi*ny/NYLO)^2;
    sy(:,NYLO - ny + 1) = ay*(1 - 1i*60*cy);
end

% ADD YHI PML
for ny = 1 : NYHI
    ay = 1 + (amax - 1)*(ny/NYHI)^p;
    cy = cmax*sin(0.5*pi*ny/NYHI)^2;
    sy(:,Ny2 - NYHI + ny) = ay*(1 - 1i*60*cy);
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INCORPORATE PML
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CALCULATE TENSOR ELEMENTS WITH UPML
ERxx = ER2./sx.*sy;
ERyy = ER2.*sx./sy;
ERzz = ER2.*sx.*sy;

URxx = UR2./sx.*sy;
URyy = UR2.*sx./sy;
URzz = UR2.*sx.*sy;

% EXTRACT TENSOR ELEMENTS ON YEE GRID
ERxx = ERxx(2:2:Nx2,1:2:Ny2);
ERyy = ERyy(1:2:Nx2,2:2:Ny2);
ERzz = ERzz(1:2:Nx2,1:2:Ny2);

URxx = URxx(1:2:Nx2,2:2:Ny2);
URyy = URyy(2:2:Nx2,1:2:Ny2);
URzz = URzz(2:2:Nx2,2:2:Ny2);
```

Demonstrate CALCPML3D()

```
% Chapter5_calcpml3d_demo.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% DEFINE INPUT ARGUMENTS
NGRID = [6 6 6];
NPML  = [2 2 2 2 2 2];

% CALL CALCPML3D
[sx,sy,sz] = calcpml3d(NGRID,NPML);

% DISPLAY THE RESULTS
disp('sx =');
disp(sx);

disp('sy =');
disp(sy);

disp('sz =');
disp(sz);
```

CALCPML3D() – Calculate Three-Dimensional Perfectly Matched Layer

```
function [sx,sy,sz] = calcpml3d(NGRID,NPML)
% CALCPML3D     Calculate PML Parameters
%
% [sx,sy,sz] = calcpml3d(NGRID,NPML);
%
```

```
% INPUT ARGUMENTS
% ================
% NGRID     [Nx Ny Nz] Number of Cells on Grid
% NPML      [NXLO NXHI NYLO NYHI NZLO NZHI] Size of PML

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% HANDLE INPUT AND OUTPUT ARGUMENTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE PML PARAMETERS
amax = 4;
cmax = 1;
p    = 3;

% EXTRACT GRID SIZE
Nx = NGRID(1);
Ny = NGRID(2);
Nz = NGRID(3);

% EXTRACT PML SIZE
NXLO = NPML(1);
NXHI = NPML(2);
NYLO = NPML(3);
NYHI = NPML(4);
NZLO = NPML(5);
NZHI = NPML(6);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE PML PARAMETERS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE PML PARAMETERS TO PROBLEM SPACE
sx = ones(Nx,Ny,Nz);
sy = ones(Nx,Ny,Nz);
sz = ones(Nx,Ny,Nz);

% CALCULATE XLO PML
for nx = 1 : NXLO
    ax = 1 + (amax - 1)*(nx/NXLO)^p;
    cx = cmax*sin(0.5*pi*nx/NXLO)^2;
    sx(NXLO - nx + 1,:,:) = ax*(1 - 1i*60*cx);
end

% CALCULATE XHI PML
for nx = 1 : NXHI
    ax = 1 + (amax - 1)*(nx/NXHI)^p;
    cx = cmax*sin(0.5*pi*nx/NXHI)^2;
    sx(Nx - NXHI + nx,:,:) = ax*(1 - 1i*60*cx);
end

% CALCULATE YLO PML
for ny = 1 : NYLO
    ay = 1 + (amax - 1)*(ny/NYLO)^p;
    cy = cmax*sin(0.5*pi*ny/NYLO)^2;
    sy(:,NYLO - ny + 1,:) = ay*(1 - 1i*60*cy);
end
```

```
62
63  % CALCULATE YHI PML
64  for ny = 1 : NYHI
65      ay = 1 + (amax - 1)*(ny/NYHI)^p;
66      cy = cmax*sin(0.5*pi*ny/NYHI)^2;
67      sy(:,Ny - NYHI + ny,:) = ay*(1 - 1i*60*cy);
68  end
69  % CALCULATE ZLO PML
70  for nz = 1 : NZLO
71      az = 1 + (amax - 1)*(nz/NZLO)^p;
72      cz = cmax*sin(0.5*pi*nz/NZLO)^2;
73      sz(:,:,NZLO - nz + 1) = az*(1 - 1i*60*cz);
74  end
75
76  % CALCULATE ZHI PML
77  for nz = 1 : NZHI
78      az = 1 + (amax - 1)*(nz/NZHI)^p;
79      cz = cmax*sin(0.5*pi*nz/NZHI)^2;
80      sz(:,:,Nz - NZHI + nz) = az*(1 - 1i*60*cz);
81  end
```

Chapter 6

Rib Waveguide Analysis

```
1    % Chapter6_ribwaveguide.m
2
3    % INITIALIZE MATLAB
4    close all;
5    clc;
6    clear all;
7
8    % UNITS
9    micrometers = 1;
10   nanometers  = 1e-3 * micrometers;
11
12   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13   %% DASHBOARD
14   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16   % WAVELENGTH
17   lam0 = 1.55 * micrometers;
18
19   % SOI RIB WAVEGUIDE PARAMETERS
20   rib_n1 = 1.0;
21   rib_n2 = 3.5;
22   rib_n3 = 1.5;
23   rib_h  = 0.60 * micrometers;
24   rib_t  = 0.60 * micrometers;
25   rib_w  = 0.80 * micrometers;
26
27   % GRID PARAMETERS
28   nmax   = max([rib_n1 rib_n2 rib_n3]);
29   NRES   = 20;
30   SPACER = lam0*[1.5 1.5 0.5 0.5];
31
```

```
% NUMBER OF MODES TO CALCULATE
NMODES = 3;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% COMPUTE OPTIMIZED GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FIRST GUESS AT GRID RESOLUTION
dx = lam0/nmax/NRES;
dy = lam0/nmax/NRES;

% SNAP GRID TO CRITICAL DIMENSIONS
nx = ceil(rib_w/dx);
dx = rib_w/nx;
ny = ceil(rib_t/dy);
dy = rib_t/ny;

% GRID SIZE
Sx = SPACER(1) + rib_w + SPACER(2);
Nx = ceil(Sx/dx);
Sx = Nx*dx;

Sy = SPACER(3) + rib_h + rib_t + SPACER(4);
Ny = ceil(Sy/dy);
Sy = Ny*dy;

% 2X GRID
Nx2 = 2*Nx;          dx2 = dx/2;
Ny2 = 2*Ny;          dy2 = dy/2;

% GRID AXES
xa = [1:Nx]*dx;      xa = xa - mean(xa);
ya = [1:Ny]*dy;      ya = ya - mean(ya);
xa2 = [1:Nx2]*dx2;      xa2 = xa2 - mean(xa2);
ya2 = [1:Ny2]*dy2;      ya2 = ya2 - mean(ya2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD DEVICE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE TO AIR
ER2 = rib_n1^2*ones(Nx2,Ny2);
UR2 = ones(Nx2,Ny2);

% CALCULATE POSITION INDICES
nx1 = 1 + round(SPACER(1)/dx2);
nx2 = nx1 + round(rib_w/dx2) - 1;

ny1 = 1 + round(SPACER(3)/dy2);
ny2 = ny1 + round(rib_h/dy2) - 1;
ny3 = ny2 + 1;
ny4 = ny3 + round(rib_t/dy2) - 1;
ny5 = ny4 + 1;

% BUILD RIB WAVEGUIDE
ER2(nx1:nx2,ny1:ny2) = rib_n2^2;
```

```
ER2(:,ny3:ny4)       = rib_n2^2;
ER2(:,ny5:Ny2)       = rib_n3^2;

% EXTRACT MATERIALS TENSORS
ERxx = ER2(2:2:Nx2,1:2:Ny2);
ERyy = ER2(1:2:Nx2,2:2:Ny2);
ERzz = ER2(1:2:Nx2,1:2:Ny2);

URxx = UR2(1:2:Nx2,2:2:Ny2);
URyy = UR2(2:2:Nx2,1:2:Ny2);
URzz = UR2(2:2:Nx2,2:2:Ny2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FINITE-DIFFERENCE ANALYSIS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DIAGONALIZE MATERIAL ARRAYS
ERxx = diag(sparse(ERxx(:)));
ERyy = diag(sparse(ERyy(:)));
ERzz = diag(sparse(ERzz(:)));
URxx = diag(sparse(URxx(:)));
URyy = diag(sparse(URyy(:)));
URzz = diag(sparse(URzz(:)));

% BUILD DERIVATE MATRICES
k0  = 2*pi/lam0;
NS  = [Nx Ny];
RES = [dx dy];
BC  = [0 0];
[DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);

% BUILD P AND Q MATRICES
P = [ DEX/ERzz*DHY , -(DEX/ERzz*DHX + URyy) ; ...
      DEY/ERzz*DHY + URxx , -DEY/ERzz*DHX ];
Q = [ DHX/URzz*DEY , -(DHX/URzz*DEX + ERyy) ; ...
      DHY/URzz*DEY + ERxx , -DHY/URzz*DEX ];

% SOLVE EIGEN-VALUE PROBLEM
ev       = -rib_n2^2;
[Exy,D2] = eigs(P*Q,NMODES,ev);
D        = sqrt(D2);
NEFF     = -1i*diag(D);

% CALCULATE OTHER FIELD COMPONENTS
M  = Nx*Ny;
Ex = Exy(1:M,:);
Ey = Exy(M+1:2*M,:);

Hxy = (Q*Exy)/D;
Hx  = Hxy(1:M,:);
Hy  = Hxy(M+1:2*M,:);

Ez = ERzz\(DHX*Hy - DHY*Hx);
Hz = URzz\(DEX*Ey - DEY*Ex);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% VISUALIZE THE MODES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% VISUALIZE CALCULATED MODES
for m = 1 : NMODES
    % Extract Fields
    ex = reshape(Ex(:,m),Nx,Ny);
    ey = reshape(Ey(:,m),Nx,Ny);

    % Scale Fields
    fmax = max(abs([ ex(:) ; ey(:) ]));
    ex   = ex/fmax;
    ey   = ey/fmax;

    % Plot Modes
    subplot(NMODES,2,(m - 1)*2 + 1);
    imagesc(xa,ya,abs(ex).');
    axis equal tight off;
    colorbar;
    caxis([0 1]);
    title(['Ex, n_{eff} = ' num2str(NEFF(m))]);

    subplot(NMODES,2,(m - 1)*2 + 2);
    imagesc(xa,ya,abs(ey).');
    axis equal tight off;
    colorbar;
    caxis([0 1]);
    title('Ey');

end
```

Slab Waveguide Analysis

```
% Chapter6_slabwaveguide.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% UNITS
micrometers = 1;
nanometers  = 1e-3 * micrometers;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% WAVELENGTH AND MODE
lam0 = 1.55 * micrometers;
MODE = 'H';

% SLAB WAVEGUIDE
a  = 1500 * nanometers;
n1 = 1.0;
n2 = 2.0;
```

```
n3 = 1.5;

% GRID PARAMETERS
nmax = max([n1 n2 n3]);
NRES = 20;
b    = 3*lam0;

% NUMBER OF MODES TO CALCULATE
NMODES = 4;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE OPTMIZED GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FIRST GUESS AT GRID RESOLUION
dx = lam0/nmax/NRES;

% SNAP GRID TO CRITICAL DIMENSIONS
nx = ceil(a/dx);
dx = a/nx;

% CALCULATE TOTAL GRID SIZE
Sx = b + a + b;
Nx = ceil(Sx/dx);
Sx = Nx*dx;

% GRID AXIS
xa = [1:Nx]*dx;
xa = xa - mean(xa);

% 2X GRID
Nx2 = 2*Nx;
dx2 = dx/2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD SLAB WAVEGUIDE ONTO GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE TO FREE SPACE
ER2 = ones(Nx2,1);
UR2 = ones(Nx2,1);

% CALCULATE START AND STOP ARRAY INDICES
nx1 =   1 + ceil(b/dx2);
nx2 = nx1 + round(a/dx2) - 1;

% BUILD SLAB WAVEGUIDE
ER2(1:nx1-1)   = n1^2;
ER2(nx1:nx2)   = n2^2;
ER2(nx2+1:Nx2) = n3^2;

% EXTRACT YEE GRID ARRAYS
ERxx = ER2(2:2:Nx2);
ERyy = ER2(1:2:Nx2);
ERzz = ER2(1:2:Nx2);
URxx = UR2(1:2:Nx2);
```

```
URyy = UR2(2:2:Nx2);
URzz = UR2(2:2:Nx2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FINITE-DIFFERENCE ANALYSIS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FORM DIAGONAL MATERIALS ARRAYS
ERxx = diag(sparse(ERxx(:)));
ERyy = diag(sparse(ERyy(:)));
ERzz = diag(sparse(ERzz(:)));
URxx = diag(sparse(URxx(:)));
URyy = diag(sparse(URyy(:)));
URzz = diag(sparse(URzz(:)));

% BUILD DERIVATIVE MATRICES
k0  = 2*pi/lam0;
NS  = [Nx 1];
RES = [dx 1];
BC  = [0 0];
[DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);

% BUILD EIGEN-VALUE PROBLEM
if MODE=='E'
    A = -(DHX/URzz*DEX + ERyy);
    B = inv(URxx);
else
    A = -(DEX/ERzz*DHX + URyy);
    B = inv(ERxx);
end

% SOLVE EIGEN-VALUE PROBLEM
ev      = -n2^2;
[Fy,D2] = eigs(A,B,NMODES,ev);
D       = sqrt(D2);
NEFF    = -1i*diag(D);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% VISUALIZE THE GUIDED MODES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% PREPARE THE FIGURE WINDOW
clf;
hold on;

% DRAW CORE
x = NMODES*[0 1 1 0 0] - 0.5;
y = 0.5*a*[-1 -1 +1 +1 -1];
fill(x,y,0.8*[1 1 1]);

% DRAW THE MODES
Fy = Fy/max(abs(Fy(:)));
for m = 1 : NMODES
    x0 = m - 1;
    line(x0+0.4*Fy(:,m),xa,'LineWidth',3,'Color','w');
    line(x0+0.4*Fy(:,m),xa,'LineWidth',1,'Color','b');
```

```
136      T = ['$n_{' num2str(m) ',\textrm{eff}} = ' ...
137           num2str(NEFF(m),'%4.2f') '$'];
138      text(x0,-b/2,T,'Interpreter','LaTex',...
139           'Rotation',-90,...
140           'VerticalAlignment','bottom');
141  end
142
143  % SET VIEW
144  hold off;
145  axis equal tight off;
```

Animating the Slab Waveguide Analysis

```
1    % Chapter6_animatedslabmode.m
2
3    % INITIALIZE MATLAB
4    close all;
5    clc;
6    clear all;
7
8    % UNITS
9    micrometers = 1;
10   nanometers  = 1e-3 * micrometers;
11
12   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13   %% DASHBOARD
14   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
16   % WAVELENGTH AND MODE
17   lam0 = 1.55 * micrometers;
18   MODE = 'H';
19
20   % SLAB WAVEGUIDE
21   a  = 1500 * nanometers;
22   n1 = 1.0;
23   n2 = 2.0;
24   n3 = 1.5;
25
26   % GRID PARAMETERS
27   nmax = max([n1 n2 n3]);
28   NRES = 20;
29   b    = 3*lam0;
30
31   % NUMBER OF MODES TO CALCULATE
32   NMODES = 4;
33
34   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
35   %% CALCULATE OPTMIZED GRID
36   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37
38   % FIRST GUESS AT GRID RESOLUION
39   dx = lam0/nmax/NRES;
40
41   % SNAP GRID TO CRITICAL DIMENSIONS
42   nx = ceil(a/dx);
43   dx = a/nx;
```

```
44
45  % CALCULATE TOTAL GRID SIZE
46  Sx = b + a + b;
47  Nx = ceil(Sx/dx);
48  Sx = Nx*dx;
49
50  % GRID AXIS
51  xa = [1:Nx]*dx;
52  xa = xa - mean(xa);
53
54  % 2X GRID
55  Nx2 = 2*Nx;
56  dx2 = dx/2;
57
58  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
59  %% BUILD SLAB WAVEGUIDE ONTO GRID
60  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61
62  % INITIALIZE TO FREE SPACE
63  ER2 = ones(Nx2,1);
64  UR2 = ones(Nx2,1);
65
66  % CALCULATE START AND STOP ARRAY INDICES
67  nx1 =   1 + ceil(b/dx2);
68  nx2 = nx1 + round(a/dx2) - 1;
69
70  % BUILD SLAB WAVEGUIDE
71  ER2(1:nx1-1)   = n1^2;
72  ER2(nx1:nx2)   = n2^2;
73  ER2(nx2+1:Nx2) = n3^2;
74
75  % EXTRACT YEE GRID ARRAYS
76  ERxx = ER2(2:2:Nx2);
77  ERyy = ER2(1:2:Nx2);
78  ERzz = ER2(1:2:Nx2);
79  URxx = UR2(1:2:Nx2);
80  URyy = UR2(2:2:Nx2);
81  URzz = UR2(2:2:Nx2);
82
83  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
84  %% PERFORM FINITE-DIFFERENCE ANALYSIS
85  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
86
87  % FORM DIAGONAL MATERIALS ARRAYS
88  ERxx = diag(sparse(ERxx(:)));
89  ERyy = diag(sparse(ERyy(:)));
90  ERzz = diag(sparse(ERzz(:)));
91  URxx = diag(sparse(URxx(:)));
92  URyy = diag(sparse(URyy(:)));
93  URzz = diag(sparse(URzz(:)));
94
95  % BUILD DERIVATIVE MATRICES
96  k0  = 2*pi/lam0;
97  NS  = [Nx 1];
98  RES = [dx 1];
99  BC  = [0 0];
```

```
100 [DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);
101
102 % BUILD EIGEN-VALUE PROBLEM
103 if MODE=='E'
104     A = -(DHX/URzz*DEX + ERyy);
105     B = inv(URxx);
106 else
107     A = -(DEX/ERzz*DHX + URyy);
108     B = inv(ERxx);
109 end
110
111 % SOLVE EIGEN-VALUE PROBLEM
112 ev      = -n2^2;
113 [Fy,D2] = eigs(A,B,NMODES,ev);
114 D       = sqrt(D2);
115 NEFF    = -1i*diag(D);
116
117 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
118 %% ANIMATE A SLAB WAVEGUIDE MODE
119 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
120
121 % DEFINE GIF ANIMATION
122 NFRAMES  = 40;
123 gif_name = 'slab_animation.gif';
124
125 % PICK A MODE
126 m = 2;
127
128 % CALCULATE A 2D GRID
129 Sy = 2*Sx;
130 dy = dx;
131 Ny = ceil(Sy/dy);
132 Sy = Ny*dy;
133 ya = [0:Ny-1]*dy;
134 [Y,X] = meshgrid(ya,xa);
135
136 % PROPAGATE MODE ACROSS GRID
137 F = zeros(Nx,Ny);
138 for ny = 1 : Ny
139     SM(:,ny) = Fy(:,m)*exp(-1i*k0*NEFF(m)*ny*dy);
140 end
141
142 % VISUALIZE USING IMAGESC
143 for nframe = 1 : NFRAMES
144
145     % Add Phase
146     phase = 2*pi*nframe/NFRAMES;
147     fi = SM*exp(1i*phase);
148
149     % Prepare Graphics
150     clf;
151     hold on;
152
153     % Draw Field
154     pcolor(ya,xa,real(fi));
155     shading interp;
```

```
156
157       % Calculate Transparencies
158       fmin = 0.05;
159       fmax = 0.5;
160       nmin = min([n1 n2 n3]);
161       nmax = max([n1 n2 n3]);
162       fa1  = fmin + (fmax - fmin)*(n1 - nmin)/(nmax - nmin);
163       fa2  = fmin + (fmax - fmin)*(n2 - nmin)/(nmax - nmin);
164       fa3  = fmin + (fmax - fmin)*(n3 - nmin)/(nmax - nmin);
165
166       % Draw Slab Waveguide
167       x = [ ya(1) ya(Ny) ya(Ny) ya(1) ya(1) ];
168       y = [ a/2 , a/2 , b+a/2 , b+a/2 , a/2 ];
169       fill(x,-y,'w','FaceAlpha',fa1);
170       fill(x,+y,'w','FaceAlpha',fa3);
171       y = [ -a/2 , -a/2 , a/2 , a/2 , -a/2 ];
172       fill(x,-y,'w','FaceAlpha',fa2);
173
174       % Set Graphics View
175       hold off;
176       axis equal tight off;
177       set(gca,'YDir','reverse');
178       drawnow;
179
180       % Capture Frame
181       F = getframe(gca);
182       F = frame2im(F);
183       [ind,cmap] = rgb2ind(F,256,'nodither');
184
185       % Add Frame to GIF
186       if nframe == 1
187           imwrite(ind,cmap,gif_name,'gif',...
188                   'DelayTime',0,'Loopcount',inf);
189       else
190           imwrite(ind,cmap,gif_name,'gif',...
191                   'DelayTime',0,'WriteMode','append');
192       end
193   end
```

Calculating Surface Plasmon Polaritons (SPPs)

```
1     % Chapter6_spp.m
2
3     % INITIALIZE MATLAB
4     close all;
5     clc;
6     clear all;
7
8     % UNITS
9     micrometers = 1;
10    nanometers  = 1e-3 * micrometers;
11
12    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13    %% DASHBOARD
14    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15
```

```
16  % WAVELENGTH AND MODE
17  lam0 = 500 * nanometers;
18
19  % SURFACE PROPERTIES
20  erd = 2.31;
21  erm = -9.98 - 0.26i;
22  b1  = 2*lam0;
23  b2  = 1*lam0;
24
25  % GRID PARAMETERS
26  nmax   = max(real([sqrt(erd) sqrt(erm)]));
27  NRES   = 200;
28
29  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30  %% CALCULATE OPTMIZED GRID
31  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32
33  % FIRST GUESS AT GRID RESOLUION
34  dx = lam0/nmax/NRES;
35
36  % CALCULATE TOTAL GRID SIZE
37  Sx = b1 + b2;
38  Nx = ceil(Sx/dx);
39  Sx = Nx*dx;
40
41  % GRID AXIS
42  xa = [0:Nx-1]*dx;
43  xa = xa - b1;
44
45  % 2X GRID
46  Nx2 = 2*Nx;
47  dx2 = dx/2;
48
49  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
50  %% BUILD SLAB WAVEGUIDE ONTO GRID
51  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
52
53  % INITIALIZE TO FREE SPACE
54  ER2 = ones(Nx2,1);
55  UR2 = ones(Nx2,1);
56
57  % BUILD MATERIAL INTERFACE
58  nx = 2*round(b1/dx2/2);
59  ER2(1:nx)     = erd;
60  ER2(nx+1:Nx2) = erm;
61
62  % EXTRACT YEE GRID ARRAYS
63  ERxx = ER2(2:2:Nx2);
64  ERyy = ER2(1:2:Nx2);
65  ERzz = ER2(1:2:Nx2);
66  URxx = UR2(1:2:Nx2);
67  URyy = UR2(2:2:Nx2);
68  URzz = UR2(2:2:Nx2);
69
70  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
71  %% PERFORM FINITE-DIFFERENCE ANALYSIS
```

```
72  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
73
74  % FORM DIAGONAL MATERIALS ARRAYS
75  ERxx = diag(sparse(ERxx(:)));
76  ERyy = diag(sparse(ERyy(:)));
77  ERzz = diag(sparse(ERzz(:)));
78  URxx = diag(sparse(URxx(:)));
79  URyy = diag(sparse(URyy(:)));
80  URzz = diag(sparse(URzz(:)));
81
82  % BUILD DERIVATIVE MATRICES
83  k0  = 2*pi/lam0;
84  NS  = [Nx 1];
85  RES = [dx 1];
86  BC  = [0 0];
87  [DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);
88
89  % BUILD EIGEN-VALUE PROBLEM (H MODE)
90  A = -(DEX/ERzz*DHX + URyy);
91  B = inv(ERxx);
92
93  % SOLVE EIGEN-VALUE PROBLEM
94  ev      = -erd*erm/(erd + erm);
95  [Fy,D2] = eigs(A,B,1,ev);
96  D       = sqrt(D2);
97  NEFF    = -1i*diag(D);
98
99  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
100 %% VISUALIZE THE SURFACE WAVE
101 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
102
103 % PREPARE THE FIGURE WINDOW
104 clf;
105 hold on;
106
107 % DRAW MEDIUMS
108 x = 1.2*[0 1 1 0 0] - 0.1;
109 y = [0 0 -b1 -b1 0];
110 fill(x,y,0.9*[1 1 1]);
111 y = [0 0 b2 b2 0];
112 fill(x,y,0.5*[1 1 1]);
113
114 % DRAW THE MODES
115 Fy = real(Fy)/max(abs(Fy));
116 line(Fy,xa,'LineWidth',3,'Color','w');
117 line(Fy,xa,'LineWidth',1,'Color','b');
118
119 % SET VIEW
120 hold off;
121 axis equal tight off;
122 set(gca,'YDir','reverse');
123 title(['$n_{\rm{eff}} = ' num2str(NEFF,'%4.2f') '$'],...
124       'Interpreter','LaTex');
```

Microstrip Transmission Line Analysis

```
% Chapter6_microstrip.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% UNITS
meters      = 1;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
micrometers = 1e-6 * meters;
seconds     = 1;
hertz       = 1/seconds;
kilohertz   = 1e3*hertz;
megahertz   = 1e6*hertz;
gigahertz   = 1e9*hertz;

% CONSTANTS
c0 = 299792458 * meters/seconds;
N0 = 376.73031346177;
e0 = 8.8541878176e-12 * 1/meters;
u0 = 1.2566370614e-6 * 1/meters;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SOURCE
freq = 1.0 * gigahertz;
lam0 = c0/freq;

% MATERIAL PROPERTIES
sigma = 5.8e7;
erm   = 1 + sigma/(1i*2*pi*freq*e0);

er    = 4.4;
tand  = 0.03;
erd   = er*(1 - 1i*tand);

era   = 1.0;

% MICROSTRIP PARAMETERS
w = 1 * millimeters;
h = 0.5 * millimeters;
t = 35 * micrometers;

% GRID PARAMETERS
nmax   = sqrt(real(erd));
NRES   = 40;
NDIM   = 10;
SPACER = w*[3 3 2 0];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%% COMPUTE OPTIMIZED GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GRID RESOLUTION
dx = lam0/nmax/NRES;
dy = lam0/nmax/NRES;

% RESOLVE MINIMUM DIMENSIONS
nx = round(w/dx);
if nx<NDIM
    nx = NDIM;
    dx = w/nx;
end

ny = round(t/dy);
if ny<NDIM
    ny = NDIM;
    dy = t/ny;
end

% SNAP GRID TO CRITICAL DIMENSIONS
nx = ceil(w/dx);
dx = w/nx;

ny = ceil(h/dy);
dy = h/ny;

% COMPUTE GRID SIZE
Sx = SPACER(1) + w + SPACER(2);
Nx = ceil(Sx/dx);
Sx = Nx*dx;

Sy = SPACER(3) + t + h + SPACER(4);
Ny = ceil(Sy/dy) + 1;
Sy = Ny*dy;

% GRID AXES
xa = [1:Nx]*dx;          xa = xa - mean(xa);
ya = [0:Ny-1]*dy;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD DEVICE ON GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE MATERIALS TO FREE SPACE
URxx = ones(Nx,Ny);
URyy = ones(Nx,Ny);
URzz = ones(Nx,Ny);
ERxx = era*ones(Nx,Ny);
ERyy = era*ones(Nx,Ny);
ERzz = era*ones(Nx,Ny);

% COMPUTE POSITION INDICES
nx  = round(w/dx);
nx1 = 1 + floor((Nx - nx)/2);
nx2 = nx1 + nx - 1;
```

```
111
112  ny6 = Ny;
113  ny5 = ny6 - 2;
114  ny4 = ny5 - 1;
115  ny3 = ny4 - round(h/dy) + 1;
116  ny2 = ny3 - 1;
117  ny1 = ny2 - round(t/dy) + 1;
118
119  % ADD DIELECTRIC
120  ERxx(:,ny3:Ny) = erd;
121  ERyy(:,ny3:Ny) = erd;
122  ERzz(:,ny3:Ny) = erd;
123
124  % ADD METAL
125  ERxx(nx1:nx2-1,ny1:ny2  ) = erm;
126  ERyy(nx1:nx2  ,ny1:ny2-1) = erm;
127  ERzz(nx1:nx2  ,ny1:ny2  ) = erm;
128  ERxx(:,ny5:ny6) = erm;
129  ERyy(:,ny5:ny6) = erm;
130  ERzz(:,ny5:ny6) = erm;
131
132  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
133  %% PERFORM FINITE-DIFFERENCE ANALYSIS
134  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
135
136  % FORM DIAGONAL MATERIALS MATRICES
137  ERxx = diag(sparse(ERxx(:)));
138  ERyy = diag(sparse(ERyy(:)));
139  ERzz = diag(sparse(ERzz(:)));
140
141  URxx = diag(sparse(URxx(:)));
142  URyy = diag(sparse(URyy(:)));
143  URzz = diag(sparse(URzz(:)));
144
145  % BUILD DERIVATIVE MATRICES
146  k0  = 2*pi/lam0;
147  NS  = [Nx Ny];
148  RES = [dx dy];
149  BC  = [0 0];
150  [DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);
151
152  % BUILD P AND Q MATRICES
153  P = [ DEX/ERzz*DHY , -(DEX/ERzz*DHX + URyy) ; ...
154        DEY/ERzz*DHY + URxx , -DEY/ERzz*DHX ];
155  Q = [ DHX/URzz*DEY , -(DHX/URzz*DEX + ERyy) ; ...
156        DHY/URzz*DEY + ERxx , -DHY/URzz*DEX ];
157
158  % SOLVING EIGEN-VALUE PROBLEM
159  ereff    = (erd+1)/2 + (erd-1)/(2*sqrt(1+12*(h/w)));
160  [Exy,D2] = eigs(P*Q,1,-real(ereff));
161  gamman   = sqrt(D2);
162
163  % CALCULATE E AND H FIELD COMPONENTS
164  M   = Nx*Ny;
165  Ex  = Exy(1:M,:);
166  Ey  = Exy(M+1:2*M,:);
```

```
167
168 Hxy = (Q*Exy)/(-1i*N0*gamman);
169 Hx  = Hxy(1:M,:);
170 Hy  = Hxy(M+1:2*M,:);
171
172 % RESHAPE FIELDS BACK TO 2D GRID
173 Ex = reshape(Ex,Nx,Ny);
174 Ey = reshape(Ey,Nx,Ny);
175 Hx = reshape(Hx,Nx,Ny);
176 Hy = reshape(Hy,Nx,Ny);
177
178 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
179 %% CALCULATE TRANSMISSION LINE PARAMETERS
180 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
181
182 % CALCULATE LINE VOLTAGE
183 nxc = round(Nx/2);
184 V0  = sum(Ey(nxc,ny2:ny4))*dy;
185
186 % CALCULATE LINE CURRENT
187 s  = 1;
188 I1 = sum(Hx(nx1-s:nx2+s,ny1-s))*dx;
189 I2 = sum(Hy(nx2+s        ,ny1-s+1:ny2+s))*dy;
190 I3 = sum(Hx(nx1-s:nx2+s,ny2+s))*dx;
191 I4 = sum(Hy(nx1-s        ,ny1-s+1:ny2+s))*dy;
192 I0 = I1 + I2 - I3 - I4;
193
194 % CALCULATE Z0 and gamma
195 Z0    = V0/I0;
196 gamma = k0*gamman;
197 neff  = imag(gamman);
198
199 % CALCUALTE DISTRIBUTED PARAMETERS
200 R = real(gamma*Z0);
201 L = imag(gamma*Z0)/(2*pi*freq);
202 G = real(gamma/Z0);
203 C = imag(gamma/Z0)/(2*pi*freq);
204
205 % REPORT RESULTS
206 disp(['gamma = ' num2str(gamma) ' 1/m']);
207 disp(['Z0    = ' num2str(Z0) ' ohms']);
208 disp(['R     = ' num2str(R/1e-3) ' mOhm/m']);
209 disp(['L     = ' num2str(L/1e-9) ' nH/m']);
210 disp(['G     = ' num2str(G/1e-6) ' uS/m']);
211 disp(['C     = ' num2str(C/1e-12) ' pF/m']);
212 disp(['neff  = ' num2str(neff)]);
213 disp(['ereff = ' num2str(neff^2)]);
214
215 % CALCULATE ELECTRIC FIELD MAGNITUDE
216 E = sqrt(abs(Ex).^2+abs(Ey).^2);
217 E = E/max(E(:));
218
219 % PLOT THE FIELD
220 pcolor(xa/millimeters,ya/millimeters,E.');
221 shading interp;
222 axis equal tight;
```

```
223  title(['Magnitude of Electric Field ' ...
224        '$\left| \vec{E} \right|$'],...
225        'Interpreter','LaTex');
226  set(gca,'YDir','reverse');
227  colormap(flipud(colormap('gray')));
228  xlabel('$x$ (mm)','Interpreter','LaTex');
229  ylabel('$y$ (mm)','Interpreter','LaTex',...
230         'HorizontalAlignment','right');
```

Chapter 7

Calculating a Two-Dimensional Photonic Band Diagram

```
1    % Chapter7_photonicbanddiagram.m
2
3    % INITIALIZE MATLAB
4    close all;
5    clc;
6    clear all;
7
8    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9    %% DASHBOARD
10   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12   % PHOTONIC CRYSTAL PARAMETERS
13   a      = 1;
14   r      = 0.48*a;
15   erhole = 1.0;
16   erfill = 12.0;
17
18   % FDFD PARAMETERS
19   Nx     = 40;
20   Ny     = Nx;
21   NBETA  = 100;
22   NBANDS = 5;
23   wnmax  = 0.6;
24
25   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26   %% CALCULATE OPTIMIZED GRID
27   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28
29   % YEE GRID
30   dx = a/Nx;
31   dy = a/Ny;
32
33   % 2X GRID
34   Nx2 = 2*Nx;             dx2 = dx/2;
35   Ny2 = 2*Ny;             dy2 = dy/2;
36
37   % CALCULATE 2X MESHGRID
38   xa2 = [1:Nx2]*dx2;      xa2 = xa2 - mean(xa2);
39   ya2 = [1:Ny2]*dy2;      ya2 = ya2 - mean(ya2);
40   [Y2,X2] = meshgrid(ya2,xa2);
41
42   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
43   %% BUILD UNIT CELL
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BUILD UNIT CELL
ER2 = (X2.^2 + Y2.^2) <= r^2;
ER2 = erfill + (erhole - erfill)*ER2;
UR2 = ones(Nx2,Ny2);

% EXTRACT YEE GRID MATERIAL ARRAYS
ERxx = ER2(2:2:Nx2,1:2:Ny2);
ERyy = ER2(1:2:Nx2,2:2:Ny2);
ERzz = ER2(1:2:Nx2,1:2:Ny2);
URxx = UR2(1:2:Nx2,2:2:Ny2);
URyy = UR2(2:2:Nx2,1:2:Ny2);
URzz = UR2(2:2:Nx2,2:2:Ny2);

% SHOW UNIT CELL
subplot(131);
imagesc(xa2,ya2,ER2.');
axis equal tight;
colorbar;
title('Unit Cell');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCUALTE LIST OF BLOCH WAVE VECTORS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% RECIPROCAL LATTICE VECTORS
T1 = (2*pi/a) * [ 1 ; 0 ];
T2 = (2*pi/a) * [ 0 ; 1 ];

% KEY POINTS OF SYMMETRY
G = [ 0 ; 0 ];
X = 0.5*T1;
M = 0.5*T1 + 0.5*T2;

% CHOOSE PATH AROUND IBZ
KP = [ G X M G ];
KL = { '\Gamma' 'X' 'M' '\Gamma' };

% DETERMINE LENGTH OF IBZ PERIMETER
NKP  = length(KP);
LIBZ = 0;
for m = 1 : NKP-1
    LIBZ = LIBZ + norm(KP(:,m+1) - KP(:,m));
end

% GENERATE LIST OF POINTS AROUND IBZ
dibz  = LIBZ/NBETA;
BETA  = KP(:,1);
KT    = 1;
NBETA = 1;
for m = 1 : NKP-1
    dK      = KP(:,m+1) - KP(:,m);
    N       = ceil(norm(dK)/dibz);
    BETA    = [ BETA , KP(:,m) + dK*[1:N]/N ];
    NBETA   = NBETA + N;
```

```
100      KT(m+1) = NBETA;
101  end
102
103  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
104  %% PERFORM FDFD ANALYSIS
105  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
106
107  % DIAGONALIZE MATERIAL TENSORS
108  ERxx = diag(sparse(ERxx(:)));
109  ERyy = diag(sparse(ERyy(:)));
110  ERzz = diag(sparse(ERzz(:)));
111  URxx = diag(sparse(URxx(:)));
112  URyy = diag(sparse(URyy(:)));
113  URzz = diag(sparse(URzz(:)));
114
115  % INITIALIZE BAND DATA
116  WNTE = zeros(NBANDS,NBETA);
117  WNTM = zeros(NBANDS,NBETA);
118
119  %
120  % MAIN LOOP -- ITERATE OVER IBZ
121  %
122  for nbeta = 1 : NBETA
123
124      % Get Next Block Wave Vector
125      beta = BETA(:,nbeta);
126
127      % Build Derivative Matrices
128      NS  = [Nx Ny];
129      RES = [dx dy];
130      BC  = [1 1];
131      [DEX,DEY,DHX,DHY] = yeeder2d(NS,RES,BC,beta);
132
133      % TM Mode Analysis
134      A = - DHX/URyy*DEX - DHY/URxx*DEY;
135      B = ERzz;
136      D = eigs(A,B,NBANDS,0);
137      D = sort(D);
138      WNTM(:,nbeta) = D(1:NBANDS);
139
140      % TE Mode Analysis
141      A = - DEX/ERyy*DHX - DEY/ERxx*DHY;
142      B = URzz;
143      D = eigs(A,B,NBANDS,0);
144      D = sort(D);
145      WNTE(:,nbeta) = D(1:NBANDS);
146
147  end
148
149  % NORMALIZE THE FREQUENCIES
150  WNTE = a/(2*pi)*real(sqrt(WNTE));
151  WNTM = a/(2*pi)*real(sqrt(WNTM));
152
153  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
154  %% DRAW PROFESSIONAL BAND DIAGRAM
155  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
156
157  % PREPARE FIGURE WINDOW
158  subplot(1,3,2:3);
159
160  % DRAW BAND DATA
161  plot(1:NBETA,WNTM.','.b');
162  hold on;
163  plot(1:NBETA,WNTE.','.r');
164  hold off;
165
166  % SET VIEW
167  xlim([1 NBETA]);
168  ylim([0 wnmax]);
169  set(gca,'XTick',KT,'XTickLabel',KL);
170  xlabel('Bloch Wave Vector $\vec{\beta}$',...
171         'Interpreter','LaTex');
172  ylabel('Frequency $\omega_{\textrm{n}} = a/\lambda_0$',...
173         'Interpreter','LaTex');
174  title('Photonic Band Diagram');
```

Calculating a Two-Dimensional Isofrequency Contours

```
1    % Chapter7_IFCs.m
2
3    % INITIALIZE MATLAB
4    close all;
5    clc;
6    clear all;
7
8    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9    %% DASHBOARD
10   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
11
12   % PHOTONIC CRYSTAL PARAMETERS
13   a      = 1;
14   r      = 0.48*a;
15   erhole = 1.0;
16   erfill = 12.0;
17
18   % FDFD PARAMETERS
19   Nx     = 60;
20   Ny     = Nx;
21   NBx    = 41;
22   NBy    = NBx;
23   NBANDS = 10;
24
25   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26   %% CALCULATE OPTIMIZED GRID
27   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28
29   % YEE GRID
30   dx = a/Nx;
31   dy = a/Ny;
32
33   % 2X GRID
34   Nx2 = 2*Nx;             dx2 = dx/2;
```

```
Ny2 = 2*Ny;             dy2 = dy/2;

% CALCULATE 2X MESHGRID
xa2 = [1:Nx2]*dx2;      xa2 = xa2 - mean(xa2);
ya2 = [1:Ny2]*dy2;      ya2 = ya2 - mean(ya2);
[Y2,X2] = meshgrid(ya2,xa2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD UNIT CELL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BUILD UNIT CELL
ER2 = (X2.^2 + Y2.^2) <= r^2;
ER2 = erfill + (erhole - erfill)*ER2;
UR2 = ones(Nx2,Ny2);

% EXTRACT YEE GRID MATERIAL ARRAYS
ERxx = ER2(2:2:Nx2,1:2:Ny2);
ERyy = ER2(1:2:Nx2,2:2:Ny2);
ERzz = ER2(1:2:Nx2,1:2:Ny2);
URxx = UR2(1:2:Nx2,2:2:Ny2);
URyy = UR2(2:2:Nx2,1:2:Ny2);
URzz = UR2(2:2:Nx2,2:2:Ny2);

% SHOW UNIT CELL
subplot(131);
imagesc(xa2,ya2,ER2.');
axis equal tight;
colorbar;
title('Unit Cell');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCUALTE LIST OF BLOCH WAVE VECTORS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CREATE BETA MESHGRID
bx      = linspace(-pi/a,+pi/a,NBx);
by      = linspace(-pi/a,+pi/a,NBy);
[BY,BX] = meshgrid(by,bx);

% EXTRACT IBZ AREA
ind_ibz  = find(BX>=0 & BY>=0 & BX>=BY);
indx_ibz = 1 + mod(ind_ibz-1,NBx);
indy_ibz = 1 + floor((ind_ibz-1)/NBx);
NBETA    = length(ind_ibz);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FDFD ANALYSIS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DIAGONALIZE MATERIAL TENSORS
ERxx = diag(sparse(ERxx(:)));
ERyy = diag(sparse(ERyy(:)));
ERzz = diag(sparse(ERzz(:)));
URxx = diag(sparse(URxx(:)));
URyy = diag(sparse(URyy(:)));
```

```
URzz = diag(sparse(URzz(:)));

% INITIALIZE BAND DATA
WNTE = zeros(NBx,NBy,NBANDS);
WNTM = zeros(NBx,NBy,NBANDS);

%
% MAIN LOOP -- ITERATE OVER IBZ
%
for nbeta = 1 : NBETA

    % Get Next Block Wave Vector
    beta = [ BX(ind_ibz(nbeta)) ; BY(ind_ibz(nbeta)) ];

    % Build Derivative Matrices
    NS  = [Nx Ny];
    RES = [dx dy];
    BC  = [1 1];
    [DEX,DEY,DHX,DHY] = yeeder2d(NS,RES,BC,beta);

    % TM Mode Analysis
    A = - DHX/URyy*DEX - DHY/URxx*DEY;
    B = ERzz;
    D = eigs(A,B,NBANDS,0);
    D = real(sqrt(D));
    D = sort(D);
    WNTM(indx_ibz(nbeta),indy_ibz(nbeta),:) ...
                          = a/(2*pi)*D(1:NBANDS);

    % TE Mode Analysis
    A = - DEX/ERyy*DHX - DEY/ERxx*DHY;
    B = URzz;
    D = eigs(A,B,NBANDS,0);
    D = real(sqrt(D));
    D = sort(D);
    WNTE(indx_ibz(nbeta),indy_ibz(nbeta),:) ...
                          = a/(2*pi)*D(1:NBANDS);

end

% UNFOLD DATA
for nb = 1 : NBANDS
    wn = WNTE(:,:,nb);
    wn = wn + wn.';
    wn = wn - 0.5*diag(diag(wn));
    wn(1:floor(NBx/2),:) = wn(NBx:-1:2+floor(NBx/2),:);
    wn(:,1:floor(NBy/2)) = wn(:,NBy:-1:2+floor(NBy/2));
    WNTE(:,:,nb) = wn;

    wn = WNTM(:,:,nb);
    wn = wn + wn.';
    wn = wn - 0.5*diag(diag(wn));
    wn(1:floor(NBx/2),:) = wn(NBx:-1:2+floor(NBx/2),:);
    wn(:,1:floor(NBy/2)) = wn(:,NBy:-1:2+floor(NBy/2));
    WNTM(:,:,nb) = wn;
end
```

```
147
148  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
149  %% DRAW PRETTY ISO FREQUENCY CONTOURS
150  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
151
152  % SHOW TE BAND 1
153  subplot(2,3,2);
154  wn = WNTE(:,:,1);
155  v = linspace(min(wn(:)),max(wn(:)),20);
156  contour(bx,by,wn.',v,'LineWidth',2);
157  axis equal tight;
158  colorbar;
159  xlabel('\beta_x');
160  ylabel('\beta_y');
161  title('TE Mode 1');
162
163  % SHOW TE BAND 2
164  subplot(2,3,3);
165  wn = WNTE(:,:,2);
166  v = linspace(min(wn(:)),max(wn(:)),20);
167  contour(bx,by,wn.',v,'LineWidth',2);
168  axis equal tight;
169  colorbar;
170  xlabel('\beta_x');
171  ylabel('\beta_y');
172  title('TE Mode 2');
173
174  % SHOW TM BAND 1
175  subplot(2,3,5);
176  wn = WNTM(:,:,1);
177  v = linspace(min(wn(:)),max(wn(:)),20);
178  contour(bx,by,wn.',v,'LineWidth',2);
179  axis equal tight;
180  colorbar;
181  xlabel('\beta_x');
182  ylabel('\beta_y');
183  title('TM Mode 1');
184
185  % SHOW TM BAND 2
186  subplot(2,3,6);
187  wn = WNTM(:,:,2);
188  v = linspace(min(wn(:)),max(wn(:)),20);
189  contour(bx,by,wn.',v,'LineWidth',2);
190  axis equal tight;
191  colorbar;
192  xlabel('\beta_x');
193  ylabel('\beta_y');
194  title('TM Mode 2');
```

Chapter 8

Sawtooth Diffraction Grating

```
1    % Chapter8_sawtooth.m
2
3    % INITIALIZE MATLAB
```

```
close all;
clc;
clear all;

% UNITS
degrees     = pi/180;
meters      = 1;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
micrometers = 1e-6 * meters;
nanometers  = 1e-9 * meters;
inches      = 2.54 * centimeters;
feet        = 12 * inches;
seconds     = 1;
hertz       = 1/seconds;
kilohertz   = 1e3 * hertz;
megahertz   = 1e6 * hertz;
gigahertz   = 1e9 * hertz;
terahertz   = 1e12 * hertz;
petahertz   = 1e15 * hertz;

% CONSTANTS
e0 = 8.85418782e-12 * 1/meters;
u0 = 1.25663706e-6 * 1/meters;
N0 = sqrt(u0/e0);
c0 = 299792458 * meters/seconds;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE PLANE WAVE SOURCE PARAMETERS
f0    = 30.0 * gigahertz;
theta = 30*degrees;
lam0  = c0/f0;
k0    = 2*pi/lam0;
MODE  = 'E';

% DEFINE DIFFRACTION GRATING PARAMETERS
L   = 1.5 * centimeters;
d   = 1.0 * centimeters;
er1 = 1.0;
er2 = 9.0;

% DEFINE FDFD PARAMETERS
NRES   = 100;
SPACER = lam0*[1 1];
NPML   = [20 20];
ermax  = max([er1 er2]);
nmax   = sqrt(ermax);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE OPTIMIZED GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GRID RESOLUTION
```

```
dx = lam0/nmax/NRES;
dy = lam0/nmax/NRES;

% SNAP GRID TO CRITICAL DIMENSIONS
nx = ceil(L/dx);
dx = L/nx;

ny = ceil(d/dy);
dy = d/ny;

% GRID SIZE
Sx = L;
Nx = ceil(Sx/dx);
Sx = Nx*dx;

Sy = SPACER(1) + d + SPACER(2);
Ny = NPML(1) + ceil(Sy/dy) + NPML(1);
Sy = Ny*dy;

% 2X GRID
Nx2 = 2*Nx;             dx2 = dx/2;
Ny2 = 2*Ny;             dy2 = dy/2;

% CALCULATE AXIS VECTORS
xa = [1:Nx]*dx;
ya = [1:Ny]*dy;
[Y,X] = meshgrid(ya,xa);
xa2 = [1:Nx2]*dx2;
ya2 = [1:Ny2]*dy2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD DIFFRACTION GRATING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE TO AIR
ER2 = er1*ones(Nx2,Ny2);
UR2 = ones(Nx2,Ny2);

% BUILD SAWTOOTH
ny1 = 2*NPML(1) + round(SPACER(1)/dy2) + 1;
ny2 = ny1 + round(d/dy2) - 1;
for ny = ny1 : ny2
    f = (ny - ny1 + 1)/(ny2 - ny1 + 1);
    nx = round(f*L/dx2);
    ER2(Nx2-nx+1:Nx2,ny) = er2;
end

% ADD SUBSTRATE
ER2(:,ny2+1:Ny2) = er2;

% SHOW DIFFRACTION GRATING
subplot(121);
imagesc(xa2,ya2,ER2.');
axis equal tight off;
colorbar;
title('ER2');
```

```
116
117 % INCORPORATE PML
118 [ERxx,ERyy,ERzz,URxx,URyy,URzz] ...
119                     = addpml2d(ER2,UR2,[0 0 NPML]);
120
121 % DIAGONALIZE MATERIAL TENSORS
122 ERxx = diag(sparse(ERxx(:)));
123 ERyy = diag(sparse(ERyy(:)));
124 ERzz = diag(sparse(ERzz(:)));
125 URxx = diag(sparse(URxx(:)));
126 URyy = diag(sparse(URyy(:)));
127 URzz = diag(sparse(URzz(:)));
128
129 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
130 %% PERFORM FDFD ANALYSIS
131 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
132
133 % INCIDENT WAVE VECTOR
134 nsrc  = sqrt(UR2(1,1)*ER2(1,1));
135 kxinc = k0*nsrc*sin(theta);
136 kyinc = k0*nsrc*cos(theta);
137 kinc  = [ kxinc ; kyinc ];
138
139 % BUILD DERIVATIVE MATRICES
140 NS  = [Nx Ny];
141 RES = [dx dy];
142 BC  = [1 0];
143 [DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC,kinc/k0);
144
145 % BUILD WAVE MATRIX
146 if MODE == 'E'
147     A = DHX/URyy*DEX + DHY/URxx*DEY + ERzz;
148 else
149     A = DEX/ERyy*DHX + DEY/ERxx*DHY + URzz;
150 end
151
152 % CALCULATE SOURCE FIELD
153 fsrc  = exp(-1i*(kxinc*X + kyinc*Y));
154
155 % CALCULATE SCATTERED-FIELD MASKING MATRIX
156 ny = NPML(1) + 2;
157 Q  = zeros(Nx,Ny);
158 Q(:,1:ny) = 1;
159 Q  = diag(sparse(Q(:)));
160
161 % CALCULATE SOURCE VECTOR
162 b = (Q*A - A*Q)*fsrc(:);
163
164 % SOLVE FOR FIELD
165 f = A\b;
166 f = reshape(f,Nx,Ny);
167
168 % SHOW FIELD
169 subplot(122);
170 pcolor(xa,ya,real(f).');
171 shading interp;
```

```
axis equal tight;
set(gca,'YDir','reverse');
caxis([-1 1]);
colorbar;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ANALYZE REFLECTION AND TRANSMISSION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% EXTRACT MATERIAL PROPERTIES IN EXTERNAL REGIONS
urref = UR2(1,1);
urtrn = UR2(1,Ny2);
erref = ER2(1,1);
ertrn = ER2(1,Ny2);
nref  = sqrt(urref*erref);
ntrn  = sqrt(urtrn*ertrn);

% CALCULATE WAVE VECTOR COMPONENTS
m     = [-floor(Nx/2):+floor((Nx-1)/2)].';
kx    = kxinc - m*2*pi/Sx;
kyref = sqrt((k0*nref)^2 - kx.^2);
kytrn = sqrt((k0*ntrn)^2 - kx.^2);

% EXTRACT REFELECTED AND TRANSMITTED WAVES
fref = f(:,NPML(1)+1)./fsrc(:,1);
ftrn = f(:,Ny-NPML(2))./fsrc(:,1);

% CALCULATE AMPLITUDES OF SPATIAL HARMONICS
aref = fftshift(fft(fref))/Nx;
atrn = fftshift(fft(ftrn))/Nx;

% CALCULATE DIFFRACTION EFFICENCIES
RDE = abs(aref).^2.*real(kyref/kyinc);
if MODE == 'E'
    TDE = abs(atrn).^2.*real(urref/urtrn*kytrn/kyinc);
else
    TDE = abs(atrn).^2.*real(erref/ertrn*kytrn/kyinc);
end

% CALCULATE OVERALL REFLECTANCE & TRANSMITTANCE
REF = sum(RDE(:))
TRN = sum(TDE(:))
CON = REF + TRN

% SHOW RESULTS
ind = find((RDE(:)+TDE(:))>1e-4);
[ m(ind) 100*RDE(ind) 100*TDE(ind) ]
```

Self-Collimating Photonic Crystal

```
% Chapter8_photoniccrystal.m

% INITIALIZE MATLAB
close all;
clc;
clear all;
```

```
7
8    % UNITS
9    degrees = pi/180;
10
11   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12   %% DASHBOARD
13   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14
15   % SOURCE
16   lam0  = 1;
17   k0    = 2*pi/lam0;
18   bw    = 1*lam0;
19   theta = 20*degrees;
20
21   % PHOTONIC CRYSTAL PARAMETERS
22   wn  = 0.217;
23   a   = wn*lam0;
24   r   = 0.48*a;
25   er1 = 1.0;
26   er2 = 12.0;
27   Lx  = 60*a;
28   Ly  = 40*a;
29
30   % FDFD PARAMETERS
31   NRES   = 20;
32   SPACER = lam0*[6 6 0 0];
33   NPML   = [20 20 20 20];
34   ermax  = max([er1 er2]);
35   nmax   = sqrt(ermax);
36
37   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
38   %% CALCULATE OPTIMIZED GRID
39   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40
41   % GRID RESOLUTION
42   dx = lam0/nmax/NRES;
43   dy = lam0/nmax/NRES;
44
45   % SNAP GRID TO CRITICAL DIMENSIONS
46   b  = a*sqrt(2);
47   nx = ceil(b/dx);
48   dx = b/nx;
49
50   ny = ceil(b/dy);
51   dy = b/ny;
52
53   % GRID SIZE
54   Sx = SPACER(1) + Lx + SPACER(2);
55   Nx = NPML(1) + ceil(Sx/dx) + NPML(2);
56   Sx = Nx*dx;
57
58   Sy = SPACER(3) + Ly + SPACER(4);
59   Ny = NPML(3) + ceil(Sy/dy) + NPML(4);
60   Sy = Ny*dy;
61
62   % 2X GRID
```

```
Nx2 = 2*Nx;             dx2 = dx/2;
Ny2 = 2*Ny;             dy2 = dy/2;

% CALCULATE AXIS VECTORS
xa = [1:Nx]*dx;
xa = xa - NPML(1)*dx - SPACER(1);
ya = [1:Ny]*dy;
ya = ya - mean(ya);

xa2 = [1:Nx2]*dx2;
xa2 = xa2 - 2*NPML(1)*dx2 - SPACER(1);
ya2 = [1:Ny2]*dy2;
ya2 = ya2 - mean(ya2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD LATTICE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% UNIT CELL GRID
Nxuc = round(b/dx2);
Nyuc = round(b/dy2);
xauc = [1:Nxuc]*dx2;      xauc = xauc - mean(xauc);
yauc = [1:Nyuc]*dy2;      yauc = yauc - mean(yauc);
[Y,X] = meshgrid(yauc,xauc);

% BUILD UNIT CELL
ERuc = (X.^2 + Y.^2) <= r^2;
ERuc = ERuc | ((X - b/2).^2 + (Y - b/2).^2) <= r^2;
ERuc = ERuc | ((X + b/2).^2 + (Y - b/2).^2) <= r^2;
ERuc = ERuc | ((X - b/2).^2 + (Y + b/2).^2) <= r^2;
ERuc = ERuc | ((X + b/2).^2 + (Y + b/2).^2) <= r^2;
ERuc = er2 + (er1 - er2)*ERuc;
URuc = ones(Nxuc,Nyuc);

% INITIALIZE MATERIALS ARRAYS
ER2 = ones(Nx2,Ny2);
UR2 = ones(Nx2,Ny2);

% STACK UNIT CELL TO BUILD LATTICE
for nyuc = 1 : round(Ly/b)
    ny1 = 2*NPML(3) + (nyuc - 1)*Nyuc ...
        + round(SPACER(3)/dy2) + 1;
    ny2 = ny1 + Nyuc - 1;
    for nxuc = 1 : round(Lx/b)
        nx1 = 2*NPML(1) + (nxuc - 1)*Nxuc ...
            + round(SPACER(1)/dx2) + 1;
        nx2 = nx1 + Nxuc - 1;
        ER2(nx1:nx2,ny1:ny2) = ERuc;
        UR2(nx1:nx2,ny1:ny2) = URuc;
    end
end

% SHOW UNIT CELL
subplot(131);
imagesc(xa2,ya2,ER2.');
axis equal tight;
```

```
colorbar;
title('Lattice');

% INCORPORATE PML
[ERxx,ERyy,ERzz,URxx,URyy,URzz] = addupml2d(ER2,UR2,NPML);

% DIAGONALIZE MATERIAL TENSORS
ERxx = diag(sparse(ERxx(:)));
ERyy = diag(sparse(ERyy(:)));
ERzz = diag(sparse(ERzz(:)));
URxx = diag(sparse(URxx(:)));
URyy = diag(sparse(URyy(:)));
URzz = diag(sparse(URzz(:)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FDFD ANALYSIS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BUILD DERIVATIVE MATRICES
NS  = [Nx Ny];
RES = [dx dy];
BC  = [0 0];
[DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);

% BUILD WAVE MATRIX
A = DHX/URyy*DEX + DHY/URxx*DEY + ERzz;

% CALCULATE SOURCE FIELD
nsrc      = sqrt(UR2(1,1)*ER2(1,1));
[Y,X]     = meshgrid(ya,xa);
[TH,R]    = cart2pol(X,Y);
[XR,YR]   = pol2cart(TH-theta,R);
fsrc      = exp(-(YR/bw).^2).*exp(-1i*k0*nsrc*XR);
fsrc(X>0) = 0;

% CALCULATE SCATTERED-FIELD MASKING MATRIX
nx1 = NPML(1) + 2;
nx2 = Nx - NPML(2) - 1;
ny1 = NPML(3) + 2;
ny2 = Ny - NPML(4) - 1;
Q   = ones(Nx,Ny);
Q(nx1:nx2,ny1:ny2) = 0;
Q  = diag(sparse(Q(:)));

% CALCULATE SOURCE VECTOR
b = (Q*A - A*Q)*fsrc(:);

% SOLVE FOR FIELD
f = A\b;
f = reshape(f,Nx,Ny);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DRAW PRETTY SOLUTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% PREPARE GRAPHICS WINDOW
```

```
clf;
hold on;

% DRAW FIELD
pcolor(xa,ya,real(f).');
shading interp;

% SUPERIMPOSE LATTICE
h = contour(xa2,ya2,ER2.',10,'Color','w','LineWidth',0.5);

% SET GRAPHICS VIEW
hold off;
axis equal tight;
set(gca,'YDir','reverse');
caxis([-1 1]);
colorbar;
```

Integrated Optical Directional Coupler

```
% Chapter8_directionalcoupler.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% UNITS
micrometers = 1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SOURCE
lam0  = 1.55 * micrometers;
k0    = 2*pi/lam0;

% RIB WAVEGUIDE PARAMETERS
rib_n1 = 1.0;
rib_n2 = 3.5;
rib_n3 = 1.5;
rib_h  = 0.30 * micrometers;
rib_t  = 0.10 * micrometers;
rib_w  = 0.40 * micrometers;

% DIRECTIONAL COUPLER PARAMETERS
L1 = 0.5*lam0;
L2 = 5.0*lam0;
g  = 0.1*lam0;

% FDFD PARAMETERS
NRES   = 20;
SPACER = lam0*[2 2 1 1];
NPML   = [20 20 20 20];
nmax   = max([rib_n1 rib_n2 rib_n3]);

```

```
38  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39  %% CALCULATE OPTIMIZED GRID
40  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
41
42  % GRID RESOLUTION
43  dx = lam0/nmax/NRES;
44  dy = lam0/nmax/NRES;
45
46  % SNAP GRID TO CRITICAL DIMENSIONS
47  nx = ceil(rib_w/dx);
48  dx = rib_w/nx;
49  ny = ceil(rib_w/dy);
50  dy = rib_w/ny;
51
52  % GRID SIZE
53  Sx = SPACER(1) + L1 + L2 + SPACER(2);
54  Nx = NPML(1) + ceil(Sx/dx) + NPML(2);
55  Sx = Nx*dx;
56
57  Sy = SPACER(3) + rib_w + g + rib_w + SPACER(4);
58  Ny = NPML(3) + ceil(Sy/dy) + NPML(4);
59  Sy = Ny*dy;
60
61  % 2X GRID
62  Nx2 = 2*Nx;             dx2 = dx/2;
63  Ny2 = 2*Ny;             dy2 = dy/2;
64
65  % CALCULATE AXIS VECTORS
66  xa = [1:Nx]*dx;
67  ya = [1:Ny]*dy;
68  xa2 = [1:Nx2]*dx2;
69  ya2 = [1:Ny2]*dy2;
70
71  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
72  %% CALCULATE EFFECTIVE INDICES
73  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74
75  % ANALYZE OFF WAVEGUIDE
76
77      % Define Waveguide
78      a = rib_t;
79      b = 5*lam0;
80
81      % Calculate 1D Grid for Slab
82      dz  = lam0/nmax/NRES;
83      nz  = ceil(a/dz);
84      dz  = a/nz;
85      Sz  = b + a + b;
86      Nz  = ceil(Sz/dz);
87      Sz  = Nz*dz;
88      za  = [1:Nz]*dz;
89      za  = za - mean(za);
90      Nz2 = 2*Nz;
91      dz2 = dz/2;
92
93      % Build Slab
```

```
94        ER2             = ones(Nz2,1);
95        UR2             = ones(Nz2,1);
96        nz1             = 1 + ceil(b/dz2);
97        nz2             = nz1 + round(a/dz2) - 1;
98        ER2(1:nz1-1)    = rib_n1^2;
99        ER2(nz1:nz2)    = rib_n2^2;
100       ER2(nz2+1:Nz2) = rib_n3^2;
101
102       % Build Eigen-Value Problem
103       ERxx = diag(sparse(ER2(2:2:Nz2)));
104       ERzz = diag(sparse(ER2(1:2:Nz2)));
105       URyy = diag(sparse(UR2(2:2:Nz2)));
106
107       NS  = [1 Nz];
108       RES = [1 dz];
109       BC  = [0 0];
110       [~,DEZ,~,DHZ] = yeeder2d(NS,k0*RES,BC);
111
112       A = -(DEZ/ERxx*DHZ + URyy);
113       B = inv(ERzz);
114
115       % Solve Eigen-Value Problem
116       ev          = -rib_n2^2;
117       [Hy,D2]     = eigs(A,B,4,ev);
118       D           = sqrt(D2);
119       NEFF        = -1i*diag(D);
120       [NEFF,ind] = sort(NEFF,'descend');
121       Hy          = Hy(:,ind);
122
123       % Get Effective Index of Fundamental Mode
124       neff1 = NEFF(1);
125
126   % ANALYZE ON WAVEGUIDE
127
128       % Define Waveguide
129       a = rib_t + rib_h;
130       b = 5*lam0;
131
132       % Calculate 1D Grid for Slab
133       dz  = lam0/nmax/NRES;
134       nz  = ceil(a/dz);
135       dz  = a/nz;
136       Sz  = b + a + b;
137       Nz  = ceil(Sz/dz);
138       Sz  = Nz*dz;
139       za  = [1:Nz]*dz;
140       za  = za - mean(za);
141       Nz2 = 2*Nz;
142       dz2 = dz/2;
143
144       % Build Slab
145       ER2             = ones(Nz2,1);
146       UR2             = ones(Nz2,1);
147       nz1             = 1 + ceil(b/dz2);
148       nz2             = nz1 + round(a/dz2) - 1;
149       ER2(1:nz1-1)    = rib_n1^2;
```

```
150      ER2(nz1:nz2)   = rib_n2^2;
151      ER2(nz2+1:Nz2) = rib_n3^2;
152
153      % Build Eigen-Value Problem
154      ERxx = diag(sparse(ER2(2:2:Nz2)));
155      ERzz = diag(sparse(ER2(1:2:Nz2)));
156      URyy = diag(sparse(UR2(2:2:Nz2)));
157
158      NS  = [1 Nz];
159      RES = [1 dz];
160      BC  = [0 0];
161      [~,DEZ,~,DHZ] = yeeder2d(NS,k0*RES,BC);
162
163      A = -(DEZ/ERxx*DHZ + URyy);
164      B = inv(ERzz);
165
166      % Solve Eigen-Value Problem
167      ev         = -rib_n2^2;
168      [Hy,D2]    = eigs(A,B,4,ev);
169      D          = sqrt(D2);
170      NEFF       = -1i*diag(D);
171      [NEFF,ind] = sort(NEFF,'descend');
172      Hy         = Hy(:,ind);
173
174      % Get Effective Index of Fundamental Mode
175      neff2 = NEFF(1);
176
177  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
178  %% BUILD OPTICAL INTEGRATED CIRCUIT
179  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
180
181  % INITIALIZE MATERIALS ARRAYS
182  ER2 = neff1^2*ones(Nx2,Ny2);
183  UR2 = ones(Nx2,Ny2);
184
185  % INPUT WAVEGUIDE
186  ny1 = 2*NPML(3) + round(SPACER(3)/dy2) + 1;
187  ny2 = ny1 + round(rib_w/dy2) - 1;
188  ER2(:,ny1:ny2) = neff2^2;
189
190  % OUTPUT WAVEGUIDE
191  nx = 2*NPML(1) + round(SPACER(1)/dx2) + round(L1/dx2) + 1;
192  ny1 = ny2 + round(g/dy2) - 1;
193  ny2 = ny1 + round(rib_w/dy2) - 1;
194  ER2(nx:Nx2,ny1:ny2) = neff2^2;
195
196  % SHOW DEVICE
197  imagesc(xa2,ya2,ER2.');
198  axis equal tight;
199  colorbar;
200  title('Lattice');
201  drawnow;
202
203  % INCORPORATE PML
204  [ERxx,ERyy,ERzz,URxx,URyy,URzz] = addupml2d(ER2,UR2,NPML);
205
```

```
% DIAGONALIZE MATERIAL TENSORS
ERxx = diag(sparse(ERxx(:)));
ERyy = diag(sparse(ERyy(:)));
ERzz = diag(sparse(ERzz(:)));
URxx = diag(sparse(URxx(:)));
URyy = diag(sparse(URyy(:)));
URzz = diag(sparse(URzz(:)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ANALYZE INPUT WAVEGUIDE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% EXTRACT SLAB WAVEGUIDE FROM GRID
nx   = 2*(NPML(1) + 1);
erzz = diag(sparse(ER2(nx,1:2:Ny2)));
urxx = diag(sparse(UR2(nx,2:2:Ny2)));
uryy = diag(sparse(UR2(nx+1,1:2:Ny2)));

% BUILD DERIVATIVE MATRICES
NS  = [1 Ny];
RES = [1 dy];
BC  = [0 0];
[~,DEY,~,DHY] = yeeder2d(NS,k0*RES,BC);

% BUILD EIGEN-VALUE PROBLEM
A = -(DHY/urxx*DEY + erzz);
B = inv(uryy);

% SOLVE FULL EIGEN-VALUE PROBLEM
[Ez,D2]    = eig(full(A),full(B));
D          = sqrt(D2);
NEFF       = -1i*diag(D);
[NEFF,ind] = sort(real(NEFF),'descend');
Ez         = Ez(:,ind);

% GET SOURCE MODE
neff  = NEFF(1);
Ezsrc = Ez(:,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FDFD ANALYSIS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BUILD DERIVATIVE MATRICES
NS  = [Nx Ny];
RES = [dx dy];
BC  = [0 0];
[DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);

% BUILD WAVE MATRIX
A = DHX/URyy*DEX + DHY/URxx*DEY + ERzz;

% CALCULATE SOURCE FIELD
fsrc = zeros(Nx,Ny);
for nx = 1 : Nx
    fsrc(nx,:) = Ezsrc*exp(-1i*k0*neff*nx*dx);
```

```
end

% CALCULATE SCATTERED-FIELD MASKING MATRIX
nx        = NPML(1) + 2;
Q         = zeros(Nx,Ny);
Q(1:nx,:) = 1;
Q         = diag(sparse(Q(:)));

% CALCULATE SOURCE VECTOR
b = (Q*A - A*Q)*fsrc(:);

% SOLVE FOR FIELD
f = A\b;
f = reshape(f,Nx,Ny);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% ANALYZE TRANSMITTED AND REFLECTED
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% EXTRACT FIELDS
nx   = NPML(1) + 1;
fref = f(nx,:);
nx   = Nx - NPML(2);
ftrn = f(nx,:);

% CALCULATE MODE AMPLITUDES
aref = Ez\fref(:);
atrn = Ez\ftrn(:);

% CALCULATE REFLECTANCE AND TRANSMITTANCE
R = 100*abs(aref(1))^2
T = 100*abs(atrn(1))^2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DRAW PRETTY SOLUTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% PREPARE GRAPHICS WINDOW
clf;
hold on;

% DRAW FIELD
pcolor(xa,ya,real(f).');
shading interp;

% SUPERIMPOSE LATTICE
er0 = (neff1^2 + neff2^2)/2;
contour(xa2,ya2,ER2.',er0,'Color','w','LineWidth',0.5);

% SET GRAPHICS VIEW
hold off;
axis equal tight;
set(gca,'YDir','reverse');
caxis(0.25*[-1 1]);
colorbar;
colormap('gray');
```

Chapter 9

FDFD2D() – Generic Two-Dimensional FDFD for Periodic Structures

```
function DAT = fdfd2d(DEV,SRC)
% FDFD2D    Two-Dimensional FDFD for Periodic Structures
%
% DAT = fdfd2d(DEV,SRC);
%
% INPUT ARGUMENTS
% ===============
% DEV       Device Parameters
%   .ER2    relative permittivity on 2x grid
%   .UR2    relative permeability on 2x grid
%   .RES    [dx dy] grid resolution on Yee grid
%   .NPML   [NYLO NYHI] size of UPML on Yee grid
%
% SRC       Source Parameters
%   .lam0   free space wavelength
%   .theta  angle of incidence
%   .MODE   'E' or 'H'
%
% OUTPUT ARGUMENTS
% ===============
% DAT       Output Data
%   .f      simulated field
%   .m      diffraction order indices
%   .RDE    diffraction efficiencies of reflected waves
%   .REF    overall reflectance
%   .TDE    diffraction efficiencies of transmitted waves
%   .TRN    overall transmittance

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% HANDLE INPUT ARGUMENTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GET SIZE OF GRID
[Nx2,Ny2] = size(DEV.ER2);

% CALCULATE GRID
Nx = Nx2/2;
Ny = Ny2/2;
dx = DEV.RES(1);
dy = DEV.RES(2);
Sx = Nx*dx;
Sy = Ny*dy;
xa = [1:Nx]*dx;
ya = [1:Ny]*dy;
[Y,X] = meshgrid(ya,xa);

% WAVE NUMBER
k0 = 2*pi/SRC.lam0;

% EXTRACT MATERIAL PROPERTIES IN EXTERNAL REGIONS
urref = DEV.UR2(1,1);
urtrn = DEV.UR2(1,Ny2);
```

```
53  erref = DEV.ER2(1,1);
54  ertrn = DEV.ER2(1,Ny2);
55  nref  = sqrt(urref*erref);
56  ntrn  = sqrt(urtrn*ertrn);
57
58  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
59  %% PERFORM FDFD ANALYSIS
60  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61
62  % INCORPORATE UPML
63  [ERxx,ERyy,ERzz,URxx,URyy,URzz] ...
64      = addupml2d(DEV.ER2,DEV.UR2,[0 0 DEV.NPML]);
65
66  % DIAGONALIZE MATERIAL TENSORS
67  ERxx = diag(sparse(ERxx(:)));
68  ERyy = diag(sparse(ERyy(:)));
69  ERzz = diag(sparse(ERzz(:)));
70  URxx = diag(sparse(URxx(:)));
71  URyy = diag(sparse(URyy(:)));
72  URzz = diag(sparse(URzz(:)));
73
74  % INCIDENT WAVE VECTOR
75  kxinc = k0*nref*sin(SRC.theta);
76  kyinc = k0*nref*cos(SRC.theta);
77  kinc  = [ kxinc ; kyinc ];
78
79  % BUILD DERIVATIVE MATRICES
80  NS  = [Nx Ny];
81  BC  = [1 0];
82  [DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*DEV.RES,BC,kinc/k0);
83
84  % BUILD WAVE MATRIX
85  if SRC.MODE == 'E'
86      A = DHX/URyy*DEX + DHY/URxx*DEY + ERzz;
87  else
88      A = DEX/ERyy*DHX + DEY/ERxx*DHY + URzz;
89  end
90
91  % CALCULATE SOURCE FIELD
92  fsrc  = exp(-1i*(kxinc*X + kyinc*Y));
93
94  % CALCULATE SCATTERED-FIELD MASKING MATRIX
95  ny = DEV.NPML(1) + 2;
96  Q  = zeros(Nx,Ny);
97  Q(:,1:ny) = 1;
98  Q  = diag(sparse(Q(:)));
99
100 % CALCULATE SOURCE VECTOR
101 b = (Q*A - A*Q)*fsrc(:);
102
103 % SOLVE FOR FIELD
104 DAT.f = A\b;
105 DAT.f = reshape(DAT.f,Nx,Ny);
106
107 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
108 %% ANALYZE REFLECTION AND TRANSMISSION
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CALCULATE WAVE VECTOR COMPONENTS
DAT.m = [-floor(Nx/2):+floor((Nx-1)/2)].';
kx    = kxinc - DAT.m*2*pi/Sx;
kyref = sqrt((k0*nref)^2 - kx.^2);
kytrn = sqrt((k0*ntrn)^2 - kx.^2);

% EXTRACT REFELECTED AND TRANSMITTED WAVES
fref = DAT.f(:,DEV.NPML(1)+1)./fsrc(:,1);
ftrn = DAT.f(:,Ny-DEV.NPML(2))./fsrc(:,1);

% CALCULATE AMPLITUDES OF SPATIAL HARMONICS
aref = fftshift(fft(fref))/Nx;
atrn = fftshift(fft(ftrn))/Nx;

% CALCULATE DIFFRACTION EFFICENCIES
DAT.RDE = abs(aref).^2.*real(kyref/kyinc);
if SRC.MODE == 'E'
    DAT.TDE = abs(atrn).^2.*real(urref/urtrn*kytrn/kyinc);
else
    DAT.TDE = abs(atrn).^2.*real(erref/ertrn*kytrn/kyinc);
end

% CALCULATE OVERALL REFLECTANCE & TRANSMITTANCE
DAT.REF = sum(DAT.RDE(:));
DAT.TRN = sum(DAT.TDE(:));
```

Main Program to Simulate a Two-Dimensional Guided-Mode Resonance Filter

```
% Chapter9_GMRF.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% UNITS
micrometers = 1;
nanometers  = 1e-3 * micrometers;
degrees     = pi/180;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE PLANE WAVE SOURCE PARAMETERS
lam1      = 1.4 * micrometers;
lam2      = 1.6 * micrometers;
NLAM      = 1000;
LAMBDA    = linspace(lam1,lam2,NLAM);
SRC.theta = 0*degrees;
SRC.MODE  = 'E';

% DEFINE GMRF PARAMETERS
f  = 0.5;
```

```
n1 = 1.0;
n2 = 1.4;
nL = 1.9;
nH = 2.1;
d1 = 265 * nanometers;
d2 = 380 * nanometers;
L  = 870 * nanometers;

% DEFINE FDFD PARAMETERS
NRES     = 40;
SPACER   = max(LAMBDA)*[0.5 0.5];
DEV.NPML = [20 20];
nmax     = max([n1 n2 nL nH]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE OPTIMIZED GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GRID RESOLUTION
dx = min(LAMBDA)/nmax/NRES;
dy = min(LAMBDA)/nmax/NRES;

% SNAP GRID TO CRITICAL DIMENSIONS
nx = ceil(L/dx);
dx = L/nx;
ny = ceil(d2/dy);
dy = d2/ny;

% GRID SIZE
Sx = L;
Nx = ceil(Sx/dx);
Sx = Nx*dx;

Sy = SPACER(1) + d1 + d2 + d1 + SPACER(2);
Ny = DEV.NPML(1) + ceil(Sy/dy) + DEV.NPML(2);
Sy = Ny*dy;

% 2X GRID
Nx2 = 2*Nx;             dx2 = dx/2;
Ny2 = 2*Ny;             dy2 = dy/2;

% CALCULATE AXIS VECTORS
xa  = [1:Nx]*dx;
ya  = [1:Ny]*dy;
xa2 = [1:Nx2]*dx2;
ya2 = [1:Ny2]*dy2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD GUIDED-MODE RESONANCE FILTER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE TO AIR
DEV.ER2 = n1^2*ones(Nx2,Ny2);
DEV.UR2 = ones(Nx2,Ny2);

% COMPUTE ARRAY INDICES
```

```
nx  = round(f*L/dx2);
nx1 = 1 + floor((Nx2 - nx)/2);
nx2 = nx1 + nx - 1;

ny1 = 2*DEV.NPML(1) + round(SPACER(1)/dy2) + 1;
ny2 = ny1 + round(d1/dy2) - 1;
ny3 = ny2 + 1;
ny4 = ny3 + round(d2/dy2) - 1;
ny5 = ny4 + 1;
ny6 = ny5 + round(d1/dy2) - 1;

% BUILD DEVICE
DEV.ER2(:,ny1:ny2)       = n2^2;
DEV.ER2(:,ny3:ny4)       = nL^2;
DEV.ER2(nx1:nx2,ny3:ny4) = nH^2;
DEV.ER2(:,ny5:ny6)       = n2^2;

% SHOW DIFFRACTION GRATING
subplot(151);
imagesc(xa2,ya2,DEV.ER2.');
axis equal tight off;
colorbar;
title('ER2');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% SIMULATE DEVICE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FINISH INPUT ARGUMENTS
DEV.RES  = [dx dy];

% INITIALIZE RECORDS
REF = zeros(1,NLAM);
TRN = zeros(1,NLAM);
CON = zeros(1,NLAM);

%
% WAVELENGTH SWEEP
%
for nlam = 1 : NLAM

    % Set Wavelength
    SRC.lam0 = LAMBDA(nlam);

    % Call fdfd2d()
    DAT = fdfd2d(DEV,SRC);

    % Record Results
    REF(nlam) = DAT.REF;
    TRN(nlam) = DAT.TRN;
    CON(nlam) = DAT.REF + DAT.TRN;

    % Show Field
    subplot(152);
    imagesc(xa,ya,real(DAT.f).');
    axis equal tight;
```

```
    colorbar;
    title('Field');

    % Show Spectra
    subplot(1,5,[3:5]);
    plot(LAMBDA(1:nlam)/micrometers,CON(1:nlam),':k');
    hold on;
    plot(LAMBDA(1:nlam)/micrometers,TRN(1:nlam),'--b');
    plot(LAMBDA(1:nlam)/micrometers,REF(1:nlam),'-r');
    hold off;
    xlim([LAMBDA(1) LAMBDA(NLAM)]/micrometers);
    ylim([-0.05 1.05]);
    drawnow;
end
```

Sweep Depth of Metal Polarizer

```
% Chapter9_polarizer_d.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% UNITS
micrometers = 1;
nanometers  = 1e-3 * micrometers;
millimeters = 1e3 * micrometers;
meters      = 1e6 * micrometers;
degrees     = pi/180;

seconds   = 1;
hertz     = 1/seconds;
kilohertz = 1e3 * hertz;
megahertz = 1e6 * hertz;
gigahertz = 1e9 * hertz;
terahertz = 1e12 * hertz;

% CONSTANTS
c0 = 299279458 * meters/seconds;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE PLANE WAVE SOURCE PARAMETERS
f0        = 2.0 * terahertz;
SRC.lam0  = c0/f0;
SRC.theta = 0*degrees;

% DEFINE POLARIZER PARAMETERS
a = 4000 * nanometers;
t = 200 * nanometers;
w = 2000 * nanometers;

ersi = 11.8;
```

```
40  erau = 1e6;
41
42  d1    = 500 * nanometers;
43  d2    = 10 * micrometers;
44  NDAT  = 50;
45  d_DAT = linspace(d1,d2,NDAT);
46
47  % DEFINE FDFD PARAMETERS
48  NRES     = 200;
49  NDIM     = 1;
50  SPACER   = SRC.lam0*[0.1 0.1];
51  DEV.NPML = [10 10];
52  nmax     = sqrt(ersi);
53
54  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55  %% CALCULATE OPTIMIZED GRID
56  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57
58  % GRID RESOLUTION
59  dx0 = SRC.lam0/nmax/NRES;
60  dy0 = SRC.lam0/nmax/NRES;
61
62  % RESOLVE MINIMUM DIMENIONS
63  nx = w/dx0;
64  if nx<NDIM
65      dx0 = w/NDIM;
66  end
67
68  ny = t/dy0;
69  if ny<NDIM
70      dy0 = t/NDIM;
71  end
72
73  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
74  %% PERFORM PARAMETER SWEEP
75  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76
77  % INITIALIZE RECORDS
78  REF_H = zeros(1,NDAT);
79  REF_E = zeros(1,NDAT);
80  TRN_H = zeros(1,NDAT);
81  TRN_E = zeros(1,NDAT);
82  CON_H = zeros(1,NDAT);
83  CON_E = zeros(1,NDAT);
84  ER    = zeros(1,NDAT);
85  ER_dB = zeros(1,NDAT);
86
87  %
88  % MAIN LOOP -- ITERATE OVER d
89  %
90  for ndat = 1 : NDAT
91
92      % Get Next Value of d
93      d = d_DAT(ndat);
94
95      % Snap Grid to Critical Dimensions
```

```
96      nx = ceil(a/dx0);
97      dx = a/nx;
98      ny = ceil(d/dy0);
99      dy = d/ny;
100
101     % Calculate Grid Size
102     Sx = a;
103     Nx = ceil(Sx/dx);
104     Sx = Nx*dx;
105     Sy = SPACER(1) + t + d + t + SPACER(2);
106     Ny = DEV.NPML(1) + ceil(Sy/dy) + DEV.NPML(2);
107     Sy = Ny*dy;
108
109     % Calculate 2x Grid Parameters
110     Nx2 = 2*Nx;             dx2 = dx/2;
111     Ny2 = 2*Ny;             dy2 = dy/2;
112
113     % Calculate Axis Vectors
114     xa  = [1:Nx]*dx;
115     ya  = [1:Ny]*dy;
116     xa2 = [1:Nx2]*dx2;
117     ya2 = [1:Ny2]*dy2;
118
119     % Initialize to Air
120     DEV.ER2 = ones(Nx2,Ny2);
121     DEV.UR2 = ones(Nx2,Ny2);
122
123     % Calculate Array Indices
124     nx  = 2*round(w/dx2/2);
125     nx1 = 1 + 2*floor((Nx2 - nx)/4);
126     nx2 = nx1 + nx - 1;
127
128     ny1 = 2*DEV.NPML(1) + 2*round(SPACER(1)/dy2/2) + 1;
129     ny2 = ny1 + 2*round(t/dy2/2);
130     ny3 = ny1 + round((t + d)/dy2);
131     ny4 = ny3 + 2*round(t/dy2/2);
132
133     % Build Polarizer on 2x Grid
134     DEV.ER2(:,ny3:ny4) = erau;
135     DEV.ER2(:,ny4+1:Ny2) = ersi;
136     DEV.ER2(nx1:nx2,ny1:ny2) = erau;
137     DEV.ER2(nx1:nx2,ny2+1:Ny2) = ersi;
138
139     % Finish DEV
140     DEV.RES  = [dx dy];
141
142     % Simulate E Mode
143     SRC.MODE    = 'E';
144     DAT_E       = fdfd2d(DEV,SRC);
145     REF_E(ndat) = DAT_E.REF;
146     TRN_E(ndat) = DAT_E.TRN;
147     CON_E(ndat) = DAT_E.REF + DAT_E.TRN;
148
149     % Simulate H Mode
150     SRC.MODE    = 'H';
151     DAT_H       = fdfd2d(DEV,SRC);
```

```
152      REF_H(ndat) = DAT_H.REF;
153      TRN_H(ndat) = DAT_H.TRN;
154      CON_H(ndat) = DAT_H.REF + DAT_H.TRN;
155
156      % Calculate and Record Extinction Ratio
157      ER(ndat)    = TRN_H(ndat)/TRN_E(ndat);
158      ER_dB(ndat) = 10*log10(ER(ndat));
159
160      % Show E Mode
161      subplot(141);
162      imagesc(xa,ya,real(DAT_E.f).');
163      axis equal tight;
164      colorbar;
165      title('Field');
166
167      % Show H Mode
168      subplot(142);
169      imagesc(xa,ya,real(DAT_H.f).');
170      axis equal tight;
171      colorbar;
172      title('Field');
173
174      subplot(1,4,[3 4]);
175      plot(d_DAT(1:ndat)/micrometers,ER_dB(1:ndat),'-k');
176      xlim([d_DAT(1) d_DAT(NDAT)]/micrometers);
177      ylim([50 70]);
178
179      drawnow;
180  end
```

Sweep Frequency of a Metal Polarizer

```
1    % Chapter9_polarizer_freq.m
2
3    % INITIALIZE MATLAB
4    close all;
5    clc;
6    clear all;
7
8    % UNITS
9    micrometers = 1;
10   nanometers  = 1e-3 * micrometers;
11   millimeters = 1e3 * micrometers;
12   meters      = 1e6 * micrometers;
13   degrees     = pi/180;
14
15   seconds   = 1;
16   hertz     = 1/seconds;
17   kilohertz = 1e3 * hertz;
18   megahertz = 1e6 * hertz;
19   gigahertz = 1e9 * hertz;
20   terahertz = 1e12 * hertz;
21
22   % CONSTANTS
23   c0 = 299279458 * meters/seconds;
24
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE PLANE WAVE SOURCE PARAMETERS
f1        = 0.6 * terahertz;
f2        = 3.0 * terahertz;
NFREQ     = 40;
FREQ      = linspace(f1,f2,NFREQ);
SRC.theta = 0*degrees;

% DEFINE POLARIZER PARAMETERS
a = 4000 * nanometers;
t = 200 * nanometers;
w = 2000 * nanometers;
d = 1500 * nanometers;

ersi = 11.8;
erau = 1e6;

% DEFINE FDFD PARAMETERS
NRES     = 100;
NDIM     = 1;
lam0     = c0/mean(FREQ);
SPACER   = lam0*[0.1 0.1];
DEV.NPML = [10 10];
nmax     = sqrt(ersi);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE OPTIMIZED GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GRID RESOLUTION
lam_min = c0/max(FREQ);
dx      = lam_min/nmax/NRES;
dy      = lam_min/nmax/NRES;

% RESOLVE MINIMUM DIMENIONS
nx = w/dx;
if nx<NDIM
    dx = w/NDIM;
end

ny = t/dy;
if ny<NDIM
    dy = t/NDIM;
end

% SNAP GRID TO CRITICAL DIMENSIONS
nx = ceil(a/dx);
dx = a/nx;
ny = ceil(d/dy);
dy = d/ny;

% GRID SIZE
Sx = a;
```

```
81   Nx = ceil(Sx/dx);
82   Sx = Nx*dx;
83
84   Sy = SPACER(1) + t + d + t + SPACER(2);
85   Ny = DEV.NPML(1) + ceil(Sy/dy) + DEV.NPML(2);
86   Sy = Ny*dy;
87
88   % 2X GRID
89   Nx2 = 2*Nx;             dx2 = dx/2;
90   Ny2 = 2*Ny;             dy2 = dy/2;
91
92   % CALCULATE AXIS VECTORS
93   xa  = [1:Nx]*dx;
94   ya  = [1:Ny]*dy;
95   xa2 = [1:Nx2]*dx2;
96   ya2 = [1:Ny2]*dy2;
97
98   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
99   %% BUILD POLARIZER
100  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
101
102  % INITIALIZE TO AIR
103  DEV.ER2 = ones(Nx2,Ny2);
104  DEV.UR2 = ones(Nx2,Ny2);
105
106  % COMPUTE ARRAY INDICES
107  nx  = 2*round(w/dx2/2);
108  nx1 = 1 + 2*floor((Nx2 - nx)/4);
109  nx2 = nx1 + nx - 1;
110
111  ny1 = 2*DEV.NPML(1) + 2*round(SPACER(1)/dy2/2) + 1;
112  ny2 = ny1 + 2*round(t/dy2/2);
113  ny3 = ny1 + round((t + d)/dy2);
114  ny4 = ny3 + 2*round(t/dy2/2);
115
116  % BUILD DEVICE
117  DEV.ER2(:,ny3:ny4) = erau;
118  DEV.ER2(:,ny4+1:Ny2) = ersi;
119  DEV.ER2(nx1:nx2,ny1:ny2) = erau;
120  DEV.ER2(nx1:nx2,ny2+1:Ny2) = ersi;
121
122  % SHOW DIFFRACTION GRATING
123  subplot(151);
124  imagesc(xa2,ya2,real(DEV.ER2).');
125  axis equal tight off;
126  colorbar;
127  title('ER2');
128
129  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
130  %% SIMULATE DEVICE
131  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
132
133  % FINISH INPUT ARGUMENTS
134  DEV.RES  = [dx dy];
135
136  % INITIALIZE RECORDS
```

```
137  REF_H = zeros(1,NFREQ);
138  REF_E = zeros(1,NFREQ);
139  TRN_H = zeros(1,NFREQ);
140  TRN_E = zeros(1,NFREQ);
141  CON_H = zeros(1,NFREQ);
142  CON_E = zeros(1,NFREQ);
143
144  %
145  % WAVELENGTH SWEEP
146  %
147  for nfreq = 1 : NFREQ
148
149      % Set Wavelength
150      SRC.lam0 = c0/FREQ(nfreq);
151
152      % Simulate E Mode
153      SRC.MODE     = 'E';
154      DAT_E        = fdfd2d(DEV,SRC);
155      REF_E(nfreq) = DAT_E.REF;
156      TRN_E(nfreq) = DAT_E.TRN;
157      CON_E(nfreq) = DAT_E.REF + DAT_E.TRN;
158
159      % Simulate H Mode
160      SRC.MODE     = 'H';
161      DAT_H        = fdfd2d(DEV,SRC);
162      REF_H(nfreq) = DAT_H.REF;
163      TRN_H(nfreq) = DAT_H.TRN;
164      CON_H(nfreq) = DAT_H.REF + DAT_H.TRN;
165
166      % Show E Mode
167      subplot(231);
168      imagesc(xa,ya,real(DAT_E.f).');
169      axis equal tight;
170      colorbar;
171      title('Field');
172
173      subplot(2,3,[2 3]);
174      semilogy(FREQ(1:nfreq)/terahertz,CON_E(1:nfreq),':k');
175      hold on;
176      semilogy(FREQ(1:nfreq)/terahertz,TRN_E(1:nfreq),'--b');
177      semilogy(FREQ(1:nfreq)/terahertz,REF_E(1:nfreq),'-r');
178      hold off;
179      xlim([FREQ(1) FREQ(NFREQ)]/terahertz);
180      ylim([-0.05 1.05]);
181
182      % Show H Mode
183      subplot(234);
184      imagesc(xa,ya,real(DAT_H.f).');
185      axis equal tight;
186      colorbar;
187      title('Field');
188
189      subplot(2,3,[5 6]);
190      semilogy(FREQ(1:nfreq)/terahertz,CON_H(1:nfreq),':k');
191      hold on;
192      semilogy(FREQ(1:nfreq)/terahertz,TRN_H(1:nfreq),'--b');
```

```
193      semilogy(FREQ(1:nfreq)/terahertz,REF_H(1:nfreq),'-r');
194      hold off;
195      xlim([FREQ(1) FREQ(NFREQ)]/terahertz);
196      ylim([-0.05 1.05]);
197
198      drawnow;
199  end
200
201  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
202  %% CALCULATE AND PLOT EXTINCTION RATIO
203  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
204
205  % CALCULATE EXTINCTION RATIO
206  ER    = TRN_H./TRN_E;
207  ER_dB = 10*log10(ER);
208
209  % PLOT EXTINCTION RATIO
210  clf;
211  plot(FREQ/terahertz,ER_dB,'-k');
212  xlabel('Frequency (THz)');
213  ylabel('Extinction Ratio');
```

Chapter 10

FDFD3D() – Generic FDFD Function to Simulate Three-Dimensional Periodic Structures

```
1    function DAT = fdfd3d(DEV,SRC)
2    % FDFD3D    Three-Dimensional FDFD for Periodic Structures
3    %
4    % DAT = fdfd3d(DEV,SRC);
5    %
6    % INPUT ARGUMENTS
7    % ================
8    % DEV       Device Parameters
9    %
10   %   .ER2 or
11   %   .ER2xx   .ER2xy   .ER2xz
12   %   .ER2yx   .ER2yy   .ER2yz     relative perm. on 2x grid
13   %   .ER2zx   .ER2zy   .ER2zz
14   %
15   %   .UR2 or
16   %   .UR2xx   .UR2xy   .UR2xz
17   %   .UR2yx   .UR2yy   .UR2yz     relative perm. on 2x grid
18   %   .UR2zx   .UR2zy   .UR2zz
19   %
20   %   .RES     [dx dy] grid resolution on Yee grid
21   %   .NPML    [NZLO NZHI] size of SCPML on Yee grid
22   %
23   % SRC       Source Parameters
24   %
25   %   .lam0    free space wavelength
26   %   .theta   angle of incidence
27   %   .phi     angle of incidence
28   %   .pte     complex amplitude of the TE polarization
29   %   .ptm     complex amplitude of the TM polarization
30   %
```

```
% OUTPUT ARGUMENTS
% ================
% DAT       Output Data
%
%   .fx   .fy   .fz       simulated field
%   .m    .n              diffraction order indices
%
%   .RDE    diffraction efficiencies of reflected waves
%   .REF    overall reflectance
%   .TDE    diffraction efficiencies of transmitted waves
%   .TRN    overall transmittance
%
%   .s11    complex ref. coefficients of diffraction orders
%   .s21    complex tran. coefficients of diffraction orders

% DEFINE SOLVER PARAMETERS
tol   = 1e-6;
maxit = 15000;

% ANONYMOUS FUNCTIONS
diagonalize = @(x) diag(sparse(x(:)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% HANDLE INPUT ARGUMENTS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% GET SIZE OF GRID
if isfield(DEV,'ER2')
    [Nx2,Ny2,Nz2] = size(DEV.ER2);
else
    [Nx2,Ny2,Nz2] = size(DEV.ER2xx);
end

% CALCULATE GRID
Nx = Nx2/2;
Ny = Ny2/2;
Nz = Nz2/2;
dx = DEV.RES(1);
dy = DEV.RES(2);
dz = DEV.RES(3);
Sx = Nx*dx;
Sy = Ny*dy;
Sz = Nz*dz;

% GRID AXES
xa2 = [1:Nx2]*dx/2;          xa2 = xa2 - mean(xa2);
ya2 = [1:Ny2]*dy/2;          ya2 = ya2 - mean(ya2);
za2 = [0.5:Nz2-0.5]*dz/2;
[Y2,X2,Z2] = meshgrid(ya2,xa2,za2);

% WAVE NUMBER
k0 = 2*pi/SRC.lam0;

% SPECIAL MATRICES
M  = Nx*Ny*Nz;
ZZ = sparse(M,M);
```

```
I  = speye(M,M);

% PERMITTIVITY TENSOR ELEMENTS
if isfield(DEV,'ER2')
    ERxx = diagonalize(DEV.ER2(2:2:Nx2,1:2:Ny2,1:2:Nz2));
    ERxy = ZZ;
    ERxz = ZZ;
    ERyx = ZZ;
    ERyy = diagonalize(DEV.ER2(1:2:Nx2,2:2:Ny2,1:2:Nz2));
    ERyz = ZZ;
    ERzx = ZZ;
    ERzy = ZZ;
    ERzz = diagonalize(DEV.ER2(1:2:Nx2,1:2:Ny2,2:2:Nz2));
else
    ERxx = diagonalize(DEV.ER2xx(2:2:Nx2,1:2:Ny2,1:2:Nz2));
    ERxy = diagonalize(DEV.ER2xy(1:2:Nx2,2:2:Ny2,1:2:Nz2));
    ERxz = diagonalize(DEV.ER2xz(1:2:Nx2,1:2:Ny2,2:2:Nz2));
    ERyx = diagonalize(DEV.ER2yx(2:2:Nx2,1:2:Ny2,1:2:Nz2));
    ERyy = diagonalize(DEV.ER2yy(1:2:Nx2,2:2:Ny2,1:2:Nz2));
    ERyz = diagonalize(DEV.ER2yz(1:2:Nx2,1:2:Ny2,2:2:Nz2));
    ERzx = diagonalize(DEV.ER2zx(2:2:Nx2,1:2:Ny2,1:2:Nz2));
    ERzy = diagonalize(DEV.ER2zy(1:2:Nx2,2:2:Ny2,1:2:Nz2));
    ERzz = diagonalize(DEV.ER2zz(1:2:Nx2,1:2:Ny2,2:2:Nz2));
end

% PERMEABILITY TENSOR ELEMENTS
if isfield(DEV,'UR2')
    URxx = diagonalize(DEV.UR2(1:2:Nx2,2:2:Ny2,2:2:Nz2));
    URxy = ZZ;
    URxz = ZZ;
    URyx = ZZ;
    URyy = diagonalize(DEV.UR2(2:2:Nx2,1:2:Ny2,2:2:Nz2));
    URyz = ZZ;
    URzx = ZZ;
    URzy = ZZ;
    URzz = diagonalize(DEV.UR2(2:2:Nx2,2:2:Ny2,1:2:Nz2));
else
    URxx = diagonalize(DEV.UR2xx(1:2:Nx2,2:2:Ny2,2:2:Nz2));
    URxy = diagonalize(DEV.UR2xy(2:2:Nx2,1:2:Ny2,2:2:Nz2));
    URxz = diagonalize(DEV.UR2xz(2:2:Nx2,2:2:Ny2,1:2:Nz2));
    URyx = diagonalize(DEV.UR2yx(1:2:Nx2,2:2:Ny2,2:2:Nz2));
    URyy = diagonalize(DEV.UR2yy(2:2:Nx2,1:2:Ny2,2:2:Nz2));
    URyz = diagonalize(DEV.UR2yz(2:2:Nx2,2:2:Ny2,1:2:Nz2));
    URzx = diagonalize(DEV.UR2zx(1:2:Nx2,2:2:Ny2,2:2:Nz2));
    URzy = diagonalize(DEV.UR2zy(2:2:Nx2,1:2:Ny2,2:2:Nz2));
    URzz = diagonalize(DEV.UR2zz(2:2:Nx2,2:2:Ny2,1:2:Nz2));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FDFD ANALYS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% EXTRACT MATERIAL PROPERTIES IN EXTERNAL REGIONS
ur_ref = full(URxx(1,1));
ur_trn = full(URxx(M,M));
er_ref = full(ERxx(1,1));
```

```
er_trn = full(ERxx(M,M));
n_ref  = sqrt(ur_ref*er_ref);
n_trn  = sqrt(ur_trn*er_trn);

% CALCULATE PML PARAMETERS
[sx2,sy2,sz2] = calcpml3d([Nx2 Ny2 Nz2],2*[0 0 0 0 DEV.NPML]);

sx_ey = diagonalize(1./sx2(1:2:Nx2,2:2:Ny2,1:2:Nz2));
sx_ez = diagonalize(1./sx2(1:2:Nx2,1:2:Ny2,2:2:Nz2));
sy_ex = diagonalize(1./sy2(2:2:Nx2,1:2:Ny2,1:2:Nz2));
sy_ez = diagonalize(1./sy2(1:2:Nx2,1:2:Ny2,2:2:Nz2));
sz_ex = diagonalize(1./sz2(2:2:Nx2,1:2:Ny2,1:2:Nz2));
sz_ey = diagonalize(1./sz2(1:2:Nx2,2:2:Ny2,1:2:Nz2));

sx_hy = diagonalize(1./sx2(2:2:Nx2,1:2:Ny2,2:2:Nz2));
sx_hz = diagonalize(1./sx2(2:2:Nx2,2:2:Ny2,1:2:Nz2));
sy_hx = diagonalize(1./sy2(1:2:Nx2,2:2:Ny2,2:2:Nz2));
sy_hz = diagonalize(1./sy2(2:2:Nx2,2:2:Ny2,1:2:Nz2));
sz_hx = diagonalize(1./sz2(1:2:Nx2,2:2:Ny2,2:2:Nz2));
sz_hy = diagonalize(1./sz2(2:2:Nx2,1:2:Ny2,2:2:Nz2));

% CALCULATE WAVE VECTORS
kinc   = k0*n_ref*[ sin(SRC.theta)*cos(SRC.phi) ; ...
                    sin(SRC.theta)*sin(SRC.phi)  ; ...
                    cos(SRC.theta) ];
m      = [-floor(Nx/2):+floor((Nx-1)/2)].';
n      = [-floor(Ny/2):+floor((Ny-1)/2)].';
kx     = kinc(1) - m*2*pi/Sx;
ky     = kinc(2) - n*2*pi/Sy;
kz_ref = sqrt((k0*n_ref)^2 - kx.^2 - ky.^2);
kz_trn = sqrt((k0*n_trn)^2 - kx.^2 - ky.^2);

% BUILD DERIVATIVE MATRICES
NS  = [Nx Ny Nz];
RES = [dx dy dz];
BC  = [1 1 0];
[DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(NS,k0*RES,BC,kinc/k0);

% CALCULATE INTERPOLATION MATRICES (Assumes SRC.theta = 0)
RX = (0.5*k0*dx)*abs(DEX);
RY = (0.5*k0*dy)*abs(DEY);
RZ = (0.5*k0*dz)*abs(DEZ);

% FORM MATERIALS TENSORS
ER = [ ERxx        RX*RY'*ERxy RX*RZ'*ERxz ; ...
       RY*RX'*ERyx ERyy        RY*RZ'*ERyz ; ...
       RZ*RX'*ERzx RZ*RY'*ERzy ERzz ];
UR = [ URxx        RX'*RY*URxy RX'*RZ*URxz ; ...
       RY'*RX*URyx URyy        RY'*RZ*URyz ; ...
       RZ'*RX*URzx RZ'*RY*URzy URzz ];

% BUILD THE WAVE MATRIX A
CE = [ ZZ , -sz_hx*DEZ , sy_hx*DEY ; ...
       sz_hy*DEZ , ZZ , -sx_hy*DEX ; ...
       -sy_hz*DEY , sx_hz*DEX , ZZ ];
CH = [ ZZ , -sz_ex*DHZ , sy_ex*DHY ; ...
```

```
199         sz_ey*DHZ , ZZ , -sx_ey*DHX ; ...
200         -sy_ez*DHY , sx_ez*DHX , ZZ ];
201  A  = CH/UR*CE - ER;
202
203  % CALCULATE POLARIZATION VECTOR P
204  az = [0;0;1];
205  if abs(SRC.theta)<1e-6
206      ate = [0;1;0];
207  else
208      ate = cross(az,kinc);
209      ate = ate/norm(ate);
210  end
211  atm = cross(kinc,ate);
212  atm = atm/norm(atm);
213  P   = SRC.pte*ate + SRC.ptm*atm;
214
215  % CALCULATE SOURCE FIELD fsrc
216  fphase = exp(-1i*(kinc(1)*X2 + kinc(2)*Y2 + kinc(3)*Z2));
217  fx     = P(1)*fphase(2:2:Nx2,1:2:Ny2,1:2:Nz2);
218  fy     = P(2)*fphase(1:2:Nx2,2:2:Ny2,1:2:Nz2);
219  fz     = P(3)*fphase(1:2:Nx2,1:2:Ny2,2:2:Nz2);
220  fsrc   = [ fx(:) ; fy(:) ; fz(:) ];
221
222  % CALCULATE SCATERED FIELD MASKING MATRIX
223  nz = DEV.NPML(1) + 3;
224  Q  = zeros(Nx,Ny,Nz);
225  Q(:,:,1:nz) = 1;
226  Q = diagonalize(Q);
227  Q = [ Q ZZ ZZ ; ZZ Q ZZ ; ZZ ZZ Q ];
228
229  % CALCULATE SOURCE VECTOR B
230  b = (Q*A - A*Q)*fsrc;
231
232  % SOLVE FOR FIELD
233  f = zeros(3*M,1);
234  A = gpuArray(A);
235  b = gpuArray(b);
236  f = bicg(A,b,tol,maxit,[],[],f);
237  f = gather(f);
238
239  % EXTRACT FIELD COMPONENTS
240  DAT.fx = reshape(f(    1:  M),Nx,Ny,Nz);
241  DAT.fy = reshape(f(  M+1:2*M),Nx,Ny,Nz);
242  DAT.fz = reshape(f(2*M+1:3*M),Nx,Ny,Nz);
243
244  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
245  %% ANALYZE REFLECTION AND TRANSMISSION
246  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
247
248  % EXTRACT FIELD FROM RECORD PLANES
249  nz_ref = DEV.NPML(1) + 2;
250  fx_ref = DAT.fx(:,:,nz_ref);
251  fy_ref = DAT.fy(:,:,nz_ref);
252  fz_ref = DAT.fz(:,:,nz_ref-1:nz_ref);
253
254  nz_trn = Nz - DEV.NPML(1);
```

```
fx_trn = DAT.fx(:,:,nz_trn);
fy_trn = DAT.fy(:,:,nz_trn);
fz_trn = DAT.fz(:,:,nz_trn-1:nz_trn);

% INTERPOLATE FIELDS TO ORIGIN OF YEE CELLS
Ex_ref = fx_ref;
Ex_ref(1,:)    = (fx_ref(Nx,:)     + fx_ref(1,:))/2;
Ex_ref(2:Nx,:) = (fx_ref(1:Nx-1,:) + fx_ref(2:Nx,:))/2;

Ey_ref = fy_ref;
Ey_ref(:,1)    = (fy_ref(:,Ny)     + fy_ref(:,1))/2;
Ey_ref(:,2:Ny) = (fy_ref(:,1:Ny-1) + fy_ref(:,2:Ny))/2;

Ez_ref = (fz_ref(:,:,1) + fz_ref(:,:,2))/2;

Ex_trn = fx_trn;
Ex_trn(1,:)    = (fx_trn(Nx,:)     + fx_trn(1,:))/2;
Ex_trn(2:Nx,:) = (fx_trn(1:Nx-1,:) + fx_trn(2:Nx,:))/2;

Ey_trn = fy_trn;
Ey_trn(:,1)    = (fy_trn(:,Ny)     + fy_trn(:,1))/2;
Ey_trn(:,2:Ny) = (fy_trn(:,1:Ny-1) + fy_trn(:,2:Ny))/2;

Ez_trn = (fz_trn(:,:,1) + fz_trn(:,:,2))/2;

% REMOVE PHASE TILT
Ex_ref = Ex_ref./fphase(2:2:Nx2,1:2:Ny2,1);
Ey_ref = Ey_ref./fphase(1:2:Nx2,2:2:Ny2,1);
Ez_ref = Ez_ref./fphase(1:2:Nx2,1:2:Ny2,2);

Ex_trn = Ex_trn./fphase(2:2:Nx2,1:2:Ny2,1);
Ey_trn = Ey_trn./fphase(1:2:Nx2,2:2:Ny2,1);
Ez_trn = Ez_trn./fphase(1:2:Nx2,1:2:Ny2,2);

% CALCULATE AMPLITUDES OF DIFFRACTION ORDERS
ax_ref = fftshift(fft2(Ex_ref))/(Nx*Ny);
ay_ref = fftshift(fft2(Ey_ref))/(Nx*Ny);
az_ref = fftshift(fft2(Ez_ref))/(Nx*Ny);
a_ref  = abs(ax_ref).^2 + abs(ay_ref).^2 + abs(az_ref).^2;
DAT.s11 = sqrt(ax_ref.^2 + ay_ref.^2 + az_ref.^2);

ax_trn = fftshift(fft2(Ex_trn))/(Nx*Ny);
ay_trn = fftshift(fft2(Ey_trn))/(Nx*Ny);
az_trn = fftshift(fft2(Ez_trn))/(Nx*Ny);
a_trn  = abs(ax_trn).^2 + abs(ay_trn).^2 + abs(az_trn).^2;
DAT.s21 = sqrt(ax_trn.^2 + ay_trn.^2 + az_trn.^2);

% CALCULATE DIFFRACTION EFFICIENCIES
DAT.RDE = a_ref.*real(kz_ref/kinc(3));
DAT.TDE = a_trn.*real(ur_ref/ur_trn*kz_trn/kinc(3));

% Calculate Overall Response
DAT.REF = sum(DAT.RDE(:));
DAT.TRN = sum(DAT.TDE(:));
DAT.CON = DAT.REF + DAT.TRN;
```

Main Program to Simulate a Three-Dimensional Guided-Mode Resonance Filter

```
% Chapter10_GMRF.m

% INITIALIZE MATLAB
close all;
clc
clear all;

% DEFINE UNITS
meters      = 1;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
micrometers = 1e-6 * meters;
nanometers  = 1e-9 * meters;
seconds     = 1;
hertz       = 1/seconds;
kilohertz   = 1e3 * hertz;
megahertz   = 1e6 * hertz;
gigahertz   = 1e9 * hertz;
degrees     = pi/180;

% CONSTANTS
c0 = 299792458 * meters/seconds;
e0 = 8.854187812813e-12 * 1/meters;
u0 = 1.256637062121e-6 * 1/meters;
N0 = 376.7303136686;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE SOURCE PARAMETERS
f1        = 4.5 * gigahertz;
f2        = 5.5 * gigahertz;
NFREQ     = 500;
FREQ      = linspace(f1,f2,NFREQ);
SRC.theta = 0 * degrees;
SRC.phi   = 0 * degrees;
SRC.pte   = 1/sqrt(2);
SRC.ptm   = 1i/sqrt(2);

% DEFINE GMRF PARAMETERS
er1 = 1.0;
er2 = 2.0;
erL = 3.0;
erH = 5.0;
a   = 3.7 * centimeters;
t   = 1.5 * centimeters;
f   = 0.5;

% DEFINE GRID PARAMETERS
NRES     = 20;
DEV.NPML = [10 10];
ermax    = max([er1 er2 erL erH]);
nmax     = sqrt(ermax);
```

```
lam_max  = c0/min(FREQ);
SPACER   = lam_max * [1 1];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE OPTIMIZED GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CALCULATE PRELIMINARY RESOLUTION
lam_min = c0/max(FREQ);
dx      = lam_min/nmax/NRES;
dy      = lam_min/nmax/NRES;
dz      = lam_min/nmax/NRES;

% SNAP GRID TO CRITICAL DIMENSIONS
nx = ceil(a/dx);
dx = a/nx;
ny = ceil(a/dy);
dy = a/ny;
nz = ceil(t/dz);
dz = t/nz;

% CALCULATE GRID SIZE
Nx = ceil(a/dx);
Sx = Nx*dx;
Ny = ceil(a/dy);
Sy = Ny*dy;
Sz = SPACER(1) + t + SPACER(2);
Nz = DEV.NPML(1) + ceil(Sz/dz) + DEV.NPML(2);
Sz = Nz*dz;

% 2x GRID
Nx2 = 2*Nx;                 dx2 = dx/2;
Ny2 = 2*Ny;                 dy2 = dy/2;
Nz2 = 2*Nz;                 dz2 = dz/2;

% GRID AXES
xa = [1:Nx]*dx;             xa = xa - mean(xa);
ya = [1:Ny]*dy;             ya = ya - mean(ya);
za = [0.5:Nz-0.5]*dz;

xa2 = [1:Nx2]*dx2;          xa2 = xa2 - mean(xa2);
ya2 = [1:Ny2]*dy2;          ya2 = ya2 - mean(ya2);
za2 = [0.5:Nz2-0.5]*dz2;

% MESHGRIDS
[Y,X,Z]    = meshgrid(ya,xa,za);
[Y2,X2,Z2] = meshgrid(ya2,xa2,za2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD MATERIALS ARRAYS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE TO FREE SPACE
DEV.ER2 = ones(Nx2,Ny2,Nz2);
DEV.UR2 = ones(Nx2,Ny2,Nz2);

```

```
% BUILD GRATING LAYER
r = a*sqrt(f/pi);
DEV.ER2 = (X2.^2 + Y2.^2) <= r^2;
DEV.ER2 = erL + (erH - erL)*DEV.ER2;

% ADD SUPERSTRATE AND SUBSTRATE
nz1 = 2*DEV.NPML(1) + round(SPACER(1)/dz2) + 1;
nz2 = nz1 + round(t/dz2) - 1;
DEV.ER2(:,:,1:nz1-1)  = er1;
DEV.ER2(:,:,nz2+1:Nz2) = er2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FREQUENCY SWEEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEV
DEV.RES = [dx dy dz];

% INITIALIZE RESPONSE
REF = zeros(1,NFREQ);
TRN = zeros(1,NFREQ);
CON = zeros(1,NFREQ);

%
% MAIN LOOP -- ITERATE OVER FREQUENCY
%
for nfreq = 1 : NFREQ

    % Get Nxt Frequency
    f0       = FREQ(nfreq);
    SRC.lam0 = c0/f0;

    % Call FDFD3D()
    DAT = fdfd3d(DEV,SRC);

    % Calculate Overall Response
    REF(nfreq) = DAT.REF;
    TRN(nfreq) = DAT.TRN;
    CON(nfreq) = DAT.CON;

    % Show Field
    subplot(131);
    slice(Y/centimeters,X/centimeters,Z/centimeters,...
          real(DAT.fy),0,0,0);
    shading interp;
    axis equal tight;
    view(250,20);
    colormap('gray');
    caxis(1.5*[-1 1]);

    % Show Response
    subplot(1,3,[2 3]);
    plot(FREQ(1:nfreq)/gigahertz,CON(1:nfreq),'--k');
    hold on;
    plot(FREQ(1:nfreq)/gigahertz,REF(1:nfreq),'-r');
    plot(FREQ(1:nfreq)/gigahertz,TRN(1:nfreq),'-b');
```

```
    hold off;
    xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
    ylim([-0.05 1.05]);
    title('Frequency Response');
    xlabel('Frequency (GHz)');
    drawnow;

end
```

Main Function to Simulate a Doubly-Periodic Frequency Selective Surface

```
% Chapter10_FSS.m

% INITIALIZE MATLAB
close all;
clc
clear all;

% DEFINE UNITS
meters      = 1;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
micrometers = 1e-6 * meters;
nanometers  = 1e-9 * meters;
seconds     = 1;
hertz       = 1/seconds;
kilohertz   = 1e3 * hertz;
megahertz   = 1e6 * hertz;
gigahertz   = 1e9 * hertz;
degrees     = pi/180;

% CONSTANTS
c0 = 299792458 * meters/seconds;
e0 = 8.854187812813e-12 * 1/meters;
u0 = 1.256637062121e-6 * 1/meters;
N0 = 376.7303136686;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE SOURCE PARAMETERS
f1    = 8 * gigahertz;
f2    = 12 * gigahertz;
NFREQ = 50;
FREQ  = linspace(f1,f2,NFREQ);
SRC.theta = 30 * degrees;
SRC.phi   = 60 * degrees;
SRC.pte   = 1;
SRC.ptm   = 0;

% DEFINE FSS PARAMETERS
a  = 9.95 * millimeters;   % period of FSS
s1 = 1.00 * millimeters;
s2 = 0.90 * millimeters;
w  = 4.50 * millimeters;
```

```
46   d  = 2.32 * millimeters;
47   h  = 3.16 * millimeters;
48
49   erd = 2.5;                  % substrate dielectric
50   erm = 1e6;                  % FSS metal
51
52   % DEFINE GRID PARAMETERS
53   NRESxy   = 100;
54   NRESz    = NRESxy;
55   DEV.NPML = [10 10];
56   nmax     = sqrt(erd);
57   lam_max  = c0/min(FREQ);
58   SPACER   = 0.2*lam_max * [1 1];
59
60   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61   %% CALCULATE OPTIMIZED GRID
62   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63
64   % CALCULATE PRELIMINARY RESOLUTION
65   lam_min = c0/max(FREQ);
66   dx      = lam_min/nmax/NRESxy;
67   dy      = lam_min/nmax/NRESxy;
68   dz      = lam_min/nmax/NRESz;
69
70   % SNAP GRID TO CRITICAL DIMENSIONS
71   nx = ceil(a/dx);
72   dx = a/nx;
73
74   ny = ceil(a/dy);
75   dy = a/ny;
76
77   nz = ceil(h/dz);
78   dz = h/nz;
79
80   % CALCULATE GRID SIZE
81   Nx = ceil(a/dx);
82   Sx = Nx*dx;
83
84   Ny = ceil(a/dy);
85   Sy = Ny*dy;
86
87   Sz = SPACER(1) + h + SPACER(2);
88   Nz = DEV.NPML(1) + ceil(Sz/dz) + DEV.NPML(2);
89   Sz = Nz*dz;
90
91   % 2x GRID
92   Nx2 = 2*Nx;                 dx2 = dx/2;
93   Ny2 = 2*Ny;                 dy2 = dy/2;
94   Nz2 = 2*Nz;                 dz2 = dz/2;
95
96   % GRID AXES
97   xa = [1:Nx]*dx;             xa = xa - mean(xa);
98   ya = [1:Ny]*dy;             ya = ya - mean(ya);
99   za = [0.5:Nz-0.5]*dz;
100
101  xa2 = [1:Nx2]*dx2;          xa2 = xa2 - mean(xa2);
```

```
ya2 = [1:Ny2]*dy2;          ya2 = ya2 - mean(ya2);
za2 = [0.5:Nz2-0.5]*dz2;

% MESHGRIDS
[Y,X,Z]    = meshgrid(ya,xa,za);
[Y2,X2,Z2] = meshgrid(ya2,xa2,za2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BUILD MATERIALS ARRAYS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% INITIALIZE TO FREE SPACE
DEV.ER2 = ones(Nx2,Ny2,Nz2);
DEV.UR2 = ones(Nx2,Ny2,Nz2);

% ADD SUBSTRATE
nz1 = 2*DEV.NPML(1) + 2*round(SPACER(1)/dz2/2) + 1;
nz2 = nz1 + round(h/dz2) - 1;
DEV.ER2(:,:,nz1:nz2) = erd;

% BUILD FSS UNIT CELL
L  = 2*s2 + 2*d + s1;
nx = round(L/dx);
x1 = (a - L)/2;
x2 = x1 + s2;
x3 = (a - w)/2;
x4 = (a - s1)/2;
x5 = (a + s1)/2;
x6 = (a + w)/2;
x8 = (a + L)/2;
x7 = x8 - s2;

nx1 = round(x1/dx2);
nx2 = round(x2/dx2);
nx3 = round(x3/dx2);
nx4 = round(x4/dx2);
nx5 = round(x5/dx2);
nx6 = round(x6/dx2);
nx7 = round(x7/dx2);
nx8 = round(x8/dx2);

ER2 = zeros(Nx2,Ny2);
ER2(nx1:nx8-1,nx4:nx5-1) = 1;
ER2(nx4:nx5-1,nx1:nx8-1) = 1;
ER2(nx1:nx2-1,nx3:nx6-1) = 1;
ER2(nx7:nx8-1,nx3:nx6-1) = 1;
ER2(nx3:nx6-1,nx1:nx2-1) = 1;
ER2(nx3:nx6-1,nx7:nx8-1) = 1;

ER2 = 1 + (erm - 1)*ER2;

% ADD UNIT CELL TO GRID
DEV.ER2(:,:,nz1) = ER2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FREQUENCY SWEEP
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEV
DEV.RES = [dx dy dz];

% INITIALIZE RESPONSE
REF = zeros(1,NFREQ);
TRN = zeros(1,NFREQ);
CON = zeros(1,NFREQ);

%
% MAIN LOOP -- ITERATE OVER FREQUENCY
%
for nfreq = 1 : NFREQ

    % Get Nxt Frequency
    f0       = FREQ(nfreq);
    SRC.lam0 = c0/f0;

    % Call FDFD3D()
    DAT = fdfd3d(DEV,SRC);

    % Calculate Overall Response
    REF(nfreq) = DAT.REF;
    TRN(nfreq) = DAT.TRN;
    CON(nfreq) = DAT.CON;

    % Show Response
    subplot(121);
    plot(FREQ(1:nfreq)/gigahertz,CON(1:nfreq),'--k');
    hold on;
    plot(FREQ(1:nfreq)/gigahertz,REF(1:nfreq),'-r');
    plot(FREQ(1:nfreq)/gigahertz,TRN(1:nfreq),'-b');
    hold off;
    xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
    ylim([-0.05 1.05]);
    title('Frequency Response');
    xlabel('Frequency (GHz)');

    subplot(122);
    TdB = 10*log10(TRN);
    plot(FREQ(1:nfreq)/gigahertz,TdB(1:nfreq),'-k');
    xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
    ylim([-40 4]);
    title('Frequency Response');
    xlabel('Frequency (GHz)');

    drawnow;

end
```

Main Function to Perform Parameter Retrieval of a Left-Handed Metamaterial

```
% Chapter10_metamaterial.m

% INITIALIZE MATLAB
```

```
close all;
clc
clear all;

% DEFINE UNITS
meters      = 1;
centimeters = 1e-2 * meters;
millimeters = 1e-3 * meters;
micrometers = 1e-6 * meters;
nanometers  = 1e-9 * meters;
seconds     = 1;
hertz       = 1/seconds;
kilohertz   = 1e3 * hertz;
megahertz   = 1e6 * hertz;
gigahertz   = 1e9 * hertz;
degrees     = pi/180;

% CONSTANTS
c0 = 299792458 * meters/seconds;
e0 = 8.854187812813e-12 * 1/meters;
u0 = 1.256637062121e-6 * 1/meters;
N0 = 376.7303136686;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINE SOURCE PARAMETERS
f1        = 9 * gigahertz;
f2        = 14 * gigahertz;
NFREQ     = 200;
FREQ      = linspace(f1,f2,NFREQ);
SRC.theta = 0 * degrees;
SRC.phi   = 0 * degrees;
SRC.pte   = 0;
SRC.ptm   = 1;

% DEFINE METAMATERIAL PARAMETERS
a  = 2.500 * millimeters;   % period of metamaterial
h  = 0.250 * millimeters;   % substrate thickness
w1 = 0.140 * millimeters;   % width of wire
w2 = 0.200 * millimeters;   % linewidth of ring
g  = 0.300 * millimeters;   % gap in ring
L2 = 2.200 * millimeters;   % length of outer ring
L1 = 1.500 * millimeters;   % length of inner ring

er   = 4.4;
tand = 0.02;
erd  = er*(1 - 1i*tand);

f0    = 10 * gigahertz;
sigma = 5.8e7;
erm   = 1 + sigma/(1i*2*pi*f0*e0);

% DEFINE GRID PARAMETERS
NRES     = 120;
```

```
60   DEV.NPML = [10 10];
61   nmax     = sqrt(erd);
62   lam0     = c0/mean(FREQ);
63   SPACER   = 0.1*lam0 * [1 1];
64
65   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
66   %% CALCULATE OPTIMIZED GRID
67   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
68
69   % CALCULATE PRELIMINARY RESOLUTION
70   lam_min = c0/max(FREQ);
71   dx      = lam_min/nmax/NRES;
72   dy      = lam_min/nmax/NRES;
73   dz      = lam_min/nmax/NRES;
74
75   % SNAP GRID TO CRITICAL DIMENSIONS
76   nx = ceil(a/dx);
77   dx = a/nx;
78   ny = ceil(a/dy);
79   dy = a/ny;
80   nz = ceil(a/dz);
81   dz = a/nz;
82
83   % CALCULATE GRID SIZE
84   Nx = ceil(a/dx);
85   Sx = Nx*dx;
86
87   Ny = ceil(a/dy);
88   Sy = Ny*dy;
89
90   Sz = SPACER(1) + a + SPACER(2);
91   Nz = DEV.NPML(1) + ceil(Sz/dz) + DEV.NPML(2);
92   Sz = Nz*dz;
93
94   % 2x GRID
95   Nx2 = 2*Nx;                 dx2 = dx/2;
96   Ny2 = 2*Ny;                 dy2 = dy/2;
97   Nz2 = 2*Nz;                 dz2 = dz/2;
98
99   % GRID AXES
100  xa = [1:Nx]*dx;             xa = xa - mean(xa);
101  ya = [1:Ny]*dy;             ya = ya - mean(ya);
102  za = [0.5:Nz-0.5]*dz;
103
104  xa2 = [1:Nx2]*dx2;          xa2 = xa2 - mean(xa2);
105  ya2 = [1:Ny2]*dy2;          ya2 = ya2 - mean(ya2);
106  za2 = [0.5:Nz2-0.5]*dz2;
107
108  % MESHGRIDS
109  [Y,X,Z]    = meshgrid(ya,xa,za);
110  [Y2,X2,Z2] = meshgrid(ya2,xa2,za2);
111
112  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
113  %% BUILD MATERIALS ARRAYS
114  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
115
```

```
% INITIALIZE TO FREE SPACE
DEV.ER2 = ones(Nx2,Ny2,Nz2);
DEV.UR2 = ones(Nx2,Ny2,Nz2);

% ADD SUBSTRATE
ny  = 2*round(h/dy2/2);
ny1 = 1 + 2*floor((Ny2 - ny)/4);
ny2 = ny1 + ny - 1;
nz  = round(a/dz2);
nz1 = 1 + 2*floor((Nz2 - nz)/2/2);
nz2 = nz1 + nz - 1;
DEV.ER2(:,ny1:ny2,nz1:nz2) = erd;

% ADD WIRE
nza = nz1 + 2*round((a-w1)/2/dz2/2);
nzb = nza + round(w1/dz2) - 1;
DEV.ER2(:,ny1,nza:nzb) = erm;

% ADD OUTER RING
ny  = ny2 + 1;
nz  = 2*round(L2/dz2/2);
nz1 = 1 + 2*floor((Nz2 - nz)/4);
nz2 = nz1 + nz - 1;
nx  = round(L2/dz2);
nx1 = 1 + 2*floor((Nx2 - nx)/4);
nx2 = nx1 + nx - 1;
DEV.ER2(nx1:nx2,ny,nz1:nz2) = erm;
nxa = nx1 + round(w2/dz2) - 1;
nxb = nx2 - round(w2/dz2) + 1;
nz1 = nz1 + round(w2/dz2) - 1;
nz2 = nz2 - round(w2/dz2) + 1;
DEV.ER2(nxa:nxb,ny,nz1:nz2) = 1;
nz  = 2*round(g/dz2/2);
nz1 = 2*floor((Nz2 - nz)/4);
nz2 = nz1 + nz - 1;
DEV.ER2(nx1:nxb,ny,nz1:nz2) = 1;

% ADD INNER RING
ny  = ny2 + 1;
nz  = 2*round(L1/dz2/2);
nz1 = 1 + 2*floor((Nz2 - nz)/4);
nz2 = nz1 + nz - 1;
nx  = round(L1/dz2);
nx1 = 1 + 2*floor((Nx2 - nx)/4);
nx2 = nx1 + nx - 1;
DEV.ER2(nx1:nx2,ny,nz1:nz2) = erm;
nxa = nx1 + round(w2/dz2) - 1;
nxb = nx2 - round(w2/dz2) + 1;
nz1 = nz1 + round(w2/dz2) - 1;
nz2 = nz2 - round(w2/dz2) + 1;
DEV.ER2(nxa:nxb,ny,nz1:nz2) = 1;
nz  = 2*round(g/dz2/2);
nz1 = 2*floor((Nz2 - nz)/4);
nz2 = nz1 + nz - 1;
DEV.ER2(nxa:nx2,ny,nz1:nz2) = 1;

```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FREQUENCY SWEEP
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEV
DEV.RES = [dx dy dz];

% INITIALIZE RESPONSE
S11 = zeros(1,NFREQ);
S21 = zeros(1,NFREQ);
REF = zeros(1,NFREQ);
TRN = zeros(1,NFREQ);
CON = zeros(1,NFREQ);

%
% MAIN LOOP -- ITERATE OVER FREQUENCY
%
for nfreq = 1 : NFREQ

    % Get Nxt Frequency
    f0       = FREQ(nfreq);
    SRC.lam0 = c0/f0;
    k0       = 2*pi/SRC.lam0;

    % Call FDFD3D()
    DAT = fdfd3d(DEV,SRC);

    % Record Response
    dsrc = SPACER(1) - 3*dz;
    dref = SPACER(1) - 1*dz;
    dtrn = SPACER(2) - 1*dz;
    nxc  = 1 + floor(Nx/2);
    nyc  = 1 + floor(Nx/2);
    S11(nfreq) = DAT.s11(nxc,nyc) ./ exp(-1i*k0*(dsrc + dref));
    S21(nfreq) = DAT.s21(nxc,nyc) ./ exp(-1i*k0*(dsrc + dtrn));
    REF(nfreq) = DAT.REF;
    TRN(nfreq) = DAT.TRN;
    CON(nfreq) = DAT.CON;

    % Show Response
    subplot(2,3,[2 3]);
    plot(FREQ(1:nfreq)/gigahertz,CON(1:nfreq),'--k');
    hold on;
    plot(FREQ(1:nfreq)/gigahertz,REF(1:nfreq),'-r');
    plot(FREQ(1:nfreq)/gigahertz,TRN(1:nfreq),'-b');
    hold off;
    xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
    ylim([-0.05 1.05]);
    title('Power Response');
    xlabel('Frequency (GHz)');

    subplot(2,3,[5 6]);
    plot(FREQ(1:nfreq)/gigahertz,angle(S11(1:nfreq)),'-r');
    hold on;
    plot(FREQ(1:nfreq)/gigahertz,angle(S21(1:nfreq)),'-b');
    hold off;
```

```
228      xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
229      title('Phase Response');
230      xlabel('Frequency (GHz)');
231
232      drawnow;
233
234  end
235
236  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
237  %% PERFORM PARAMETER RETRIEVAL
238  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
239
240  % CHANGE SIGN CONVENTION
241  S11 = conj(S11);
242  S21 = conj(S21);
243
244  % UNWRAP PHASE
245  phi = angle(S11);
246  for n = 1 : NFREQ-1
247      dphi = phi(n+1) - phi(n);
248      if abs(dphi) > 0.5*pi
249          phi(n+1:NFREQ) = phi(n+1:NFREQ) - dphi;
250      end
251  end
252  S11 = abs(S11).*exp(1i*phi);
253  phi11 = phi;
254
255  phi = angle(S21);
256  for n = 1 : NFREQ-1
257      dphi = phi(n+1) - phi(n);
258      if abs(dphi) > 0.5*pi
259          phi(n+1:NFREQ) = phi(n+1:NFREQ) - dphi;
260      end
261  end
262  S21 = abs(S21).*exp(1i*phi);
263  phi21 = phi;
264
265  % RETRIEVE IMPEDANCE AND REFRACTIVE INDEX
266  X = (1 - S21.^2 + S11.^2)./(2*S11);
267
268  s      = ones(1,NFREQ);
269  r      = X + s.*sqrt(X.^2 - 1);
270  ind    = find(abs(r)>1);
271  s(ind) = -1;
272  r      = X + s.*sqrt(X.^2 - 1);
273
274  t = (S11 + S21 - r)./(1 - (S11 + S21).*r);
275
276  k0   = 2*pi*FREQ/c0;
277  eta  = N0*(1 + r)./(1 - r);
278  neff = log(t)./(1i*k0*a);
279
280  % RETRIEVE PERMITTIVITY AND PERMEABILITY
281  er = neff*N0./eta;
282  ur = neff.*eta./N0;
283
```

```
% SHOW RESPONSE
subplot(321);
plot(FREQ/gigahertz,CON,'--k','LineWidth',2);
hold on;
plot(FREQ/gigahertz,REF,':k','LineWidth',2);
plot(FREQ/gigahertz,TRN,'-k','LineWidth',2);
hold off;
xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
ylim([-0.05 1.05]);
xlabel('Frequency (GHz)');
title('Reflectance & Transmittance');

subplot(322);
plot(FREQ/gigahertz,phi11,':k','LineWidth',2);
hold on;
plot(FREQ/gigahertz,phi21,'-k','LineWidth',2);
hold off;
xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
ylim([-2.25 2.25]);
xlabel('Frequency (GHz)');
title('Phase Response');

% SHOW REFRACTIVE INDEX AND IMPEDANCE
subplot(323);
plot(FREQ/gigahertz,real(neff),'-k','LineWidth',2);
hold on;
plot(FREQ/gigahertz,imag(neff),'--k','LineWidth',2);
hold off;
xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
ylim([-5 +7]);
xlabel('Frequency (GHz)');
title('Refractive Index $n$','Interpreter','LaTex');

subplot(324);
plot(FREQ/gigahertz,real(eta)/N0,'-k','LineWidth',2);
hold on;
plot(FREQ/gigahertz,imag(eta)/N0,'--k','LineWidth',2);
hold off;
xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
ylim([-3 +3]);
xlabel('Frequency (GHz)');
title('Impedance $\eta/\eta_0$','Interpreter','LaTex');

% SHOW RELATIVE PERMITTIVITY AND PERMEABILITY
subplot(325);
plot(FREQ/gigahertz,real(er),'-k','LineWidth',2);
hold on;
plot(FREQ/gigahertz,imag(er),'--k','LineWidth',2);
hold off;
xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
ylim([-5 2]);
xlabel('Frequency (GHz)');
title('Relative Permittivity $\varepsilon_r$',...
      'Interpreter','LaTex');

subplot(326);
```

```
plot(FREQ/gigahertz,real(ur),'-k','LineWidth',2);
hold on;
plot(FREQ/gigahertz,imag(ur),'--k','LineWidth',2);
hold off;
xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
ylim([-7 +14]);
xlabel('Frequency (GHz)');
title('Relative Permeability $\mu_r$','Interpreter','LaTex');
```

Program to Simulate a Two-Dimensional Invisibility Cloak

```
% Chapter10_cloak.m

% INITIALIZE MATLAB
close all;
clc;
clear all;

% UNITS
degrees = pi/180;

% ANONYMOUS FUNCTIONS
diagonalize = @(x) diag(sparse(x(:)));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% DASHBOARD
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% SOURCE
lam0  = 1;
theta = 30*degrees;
P     = [0;0;1];

% CLOAK
R1 = 1.0*lam0;
R2 = 3.5*lam0;

% GRID
Sx   = 10*lam0;
Sy   = Sx;
NRES = 60;
NPML = [20 20 20 20];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% CALCULATE GRID
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% RESOLUTION
dx = lam0/NRES;
dy = lam0/NRES;

% GRID SIZE
Nx = ceil(Sx/dx);
dx = Sx/Nx;
Ny = ceil(Sy/dy);
dy = Sy/Ny;
```

```
46
47  % 2X GRID
48  Nx2 = 2*Nx;          dx2 = dx/2;
49  Ny2 = 2*Ny;          dy2 = dy/2;
50
51  % GRID AXES
52  xa = [1:Nx]*dx;      xa = xa - mean(xa);
53  ya = [1:Ny]*dy;      ya = ya - mean(ya);
54  xa2 = [1:Nx2]*dx2;   xa2 = xa2 - mean(xa2);
55  ya2 = [1:Ny2]*dy2;   ya2 = ya2 - mean(ya2);
56
57  % MESHGRID
58  [Y2,X2] = meshgrid(ya2,xa2);
59
60  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61  %% BUILD CLOAK
62  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63
64  % INITIALIZE TO FREE SPACE
65  ERxx = ones(Nx2,Ny2);
66  ERxy = zeros(Nx2,Ny2);
67  ERxz = zeros(Nx2,Ny2);
68  ERyx = zeros(Nx2,Ny2);
69  ERyy = ones(Nx2,Ny2);
70  ERyz = zeros(Nx2,Ny2);
71  ERzx = zeros(Nx2,Ny2);
72  ERzy = zeros(Nx2,Ny2);
73  ERzz = ones(Nx2,Ny2);
74
75  % CALCULATE PERMITTIVITY TENSORS
76  for ny = 1 : Ny2
77      for nx = 1 : Nx2
78          % Calculate Polar Coordinates
79          r   = sqrt(xa2(nx)^2 + ya2(ny)^2);
80          phi = atan2(ya2(ny),xa2(nx));
81
82          % Populate Tensor
83          if r>=R1 && r<=R2
84              % Calculate Polar Tensor
85              ERr = (r - R1)/r;
86              ERp = r/(r - R1);
87              ERz = (R2/(R2-R1))^2*ERr;
88              ER  = [ ERr 0 0 ; 0 ERp 0 ; 0 0 ERz ];
89              % Calculate Cartesian Tensor
90              R  = [ cos(phi) -sin(phi) 0 ; ...
91                     sin(phi)  cos(phi) 0 ; ...
92                     0 0 1 ];
93              ER = R*ER/R;
94              % Populate Grid
95              ERxx(nx,ny) = ER(1,1);
96              ERxy(nx,ny) = ER(1,2);
97              ERxz(nx,ny) = ER(1,3);
98              ERyx(nx,ny) = ER(2,1);
99              ERyy(nx,ny) = ER(2,2);
100             ERyz(nx,ny) = ER(2,3);
101             ERzx(nx,ny) = ER(3,1);
```

```
            ERzy(nx,ny) = ER(3,2);
            ERzz(nx,ny) = ER(3,3);
        end
    end
end

% CALCULATE IMPERMEABILITY
D = ERxx.*ERyy.*ERzz - ERxx.*ERyz.*ERzy ...
  - ERxy.*ERyx.*ERzz + ERxy.*ERyz.*ERzx ...
  + ERxz.*ERyx.*ERzy - ERxz.*ERyy.*ERzx;
YRxx = (ERyy.*ERzz - ERyz.*ERzy)./D;
YRxy = (ERxz.*ERzy - ERxy.*ERzz)./D;
YRxz = (ERxy.*ERyz - ERxz.*ERyy)./D;
YRyx = (ERyz.*ERzx - ERyx.*ERzz)./D;
YRyy = (ERxx.*ERzz - ERxz.*ERzx)./D;
YRyz = (ERxz.*ERyx - ERxx.*ERyz)./D;
YRzx = (ERyx.*ERzy - ERyy.*ERzx)./D;
YRzy = (ERxy.*ERzx - ERxx.*ERzy)./D;
YRzz = (ERxx.*ERyy - ERxy.*ERyx)./D;

% FORM DIAGONAL MATERIALS MATRICES
ERxx = diagonalize(ERxx(2:2:Nx2,1:2:Ny2));
ERxy = diagonalize(ERxy(1:2:Nx2,2:2:Ny2));
ERxz = diagonalize(ERxz(1:2:Nx2,1:2:Ny2));
ERyx = diagonalize(ERyx(2:2:Nx2,1:2:Ny2));
ERyy = diagonalize(ERyy(1:2:Nx2,2:2:Ny2));
ERyz = diagonalize(ERyz(1:2:Nx2,1:2:Ny2));
ERzx = diagonalize(ERzx(2:2:Nx2,1:2:Ny2));
ERzy = diagonalize(ERzy(1:2:Nx2,2:2:Ny2));
ERzz = diagonalize(ERzz(1:2:Nx2,1:2:Ny2));

YRxx = diagonalize(YRxx(1:2:Nx2,2:2:Ny2));
YRxy = diagonalize(YRxy(2:2:Nx2,1:2:Ny2));
YRxz = diagonalize(YRxz(2:2:Nx2,2:2:Ny2));
YRyx = diagonalize(YRyx(1:2:Nx2,2:2:Ny2));
YRyy = diagonalize(YRyy(2:2:Nx2,1:2:Ny2));
YRyz = diagonalize(YRyz(2:2:Nx2,2:2:Ny2));
YRzx = diagonalize(YRzx(1:2:Nx2,2:2:Ny2));
YRzy = diagonalize(YRzy(2:2:Nx2,1:2:Ny2));
YRzz = diagonalize(YRzz(2:2:Nx2,2:2:Ny2));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% PERFORM FDFD SIMULATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% CALCULATE SOURCE
k0    = 2*pi/lam0;
ersrc = ERxx(1,1);
nsrc  = sqrt(ersrc);
kinc  = k0*nsrc*[sin(theta);cos(theta);0];

E     = exp(-1i*(kinc(1)*X2 + kinc(2)*Y2));
Ex    = P(1)*E(2:2:Nx2,1:2:Ny2);
Ey    = P(2)*E(1:2:Nx2,2:2:Ny2);
Ez    = P(3)*E(1:2:Nx2,1:2:Ny2);
fsrc  = [ Ex(:) ; Ey(:) ; Ez(:) ];
```

```
158
159  % BUILD DERIVATIVE MATRICES
160  NS  = [Nx Ny 1];
161  RES = [dx dy 1];
162  BC  = [0 0 0];
163  [DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(NS,k0*RES,BC,kinc/k0);
164
165  % COMPUTE INTEPOLATION MATRICES
166  RX = (k0*dx)*abs(DEX)/2;
167  RY = (k0*dy)*abs(DEY)/2;
168  RZ = (k0*1)*abs(DEZ)/2;
169
170  % CALCULATE SCPML MATRICES
171  [sx2,sy2,sz2] = calcpml3d([Nx2 Ny2 2],2*[NPML 0 0]);
172
173  sx_ey = diagonalize(1./sx2(1:2:Nx2,2:2:Ny2,1));
174  sx_ez = diagonalize(1./sx2(1:2:Nx2,1:2:Ny2,2));
175  sy_ex = diagonalize(1./sy2(2:2:Nx2,1:2:Ny2,1));
176  sy_ez = diagonalize(1./sy2(1:2:Nx2,1:2:Ny2,2));
177  sz_ex = diagonalize(1./sz2(2:2:Nx2,1:2:Ny2,1));
178  sz_ey = diagonalize(1./sz2(1:2:Nx2,2:2:Ny2,1));
179
180  sx_hy = diagonalize(1./sx2(2:2:Nx2,1:2:Ny2,2));
181  sx_hz = diagonalize(1./sx2(2:2:Nx2,2:2:Ny2,1));
182  sy_hx = diagonalize(1./sy2(1:2:Nx2,2:2:Ny2,2));
183  sy_hz = diagonalize(1./sy2(2:2:Nx2,2:2:Ny2,1));
184  sz_hx = diagonalize(1./sz2(1:2:Nx2,2:2:Ny2,2));
185  sz_hy = diagonalize(1./sz2(2:2:Nx2,1:2:Ny2,2));
186
187  % BUILD MATERIAL TENSORS
188  ER = [ ERxx , RX*RY'*ERxy , RX*RZ'*ERxz ; ...
189         RY*RX'*ERyx , ERyy , RY*RZ'*ERyz ; ...
190         RZ*RX'*ERzx , RZ*RY'*ERzy , ERzz ];
191  YR = [ YRxx , RX'*RY*YRxy , RX'*RZ*YRxz ; ...
192         RY'*RX*YRyx , YRyy , RY'*RZ*YRyz ; ...
193         RZ'*RX*YRzx , RZ'*RY*YRzy , YRzz ];
194
195  % CALCULATE CURL MATRICES WITH SCPML
196  M  = Nx*Ny;
197  ZZ = sparse(M,M);
198  CE = [ ZZ , -sz_hx*DEZ , sy_hx*DEY ; ...
199         sz_hy*DEZ , ZZ , -sx_hy*DEX ; ...
200         -sy_hz*DEY , sx_hz*DEX , ZZ ];
201  CH = [ ZZ , -sz_ex*DHZ , sy_ex*DHY ; ...
202         sz_ey*DHZ , ZZ , -sx_ey*DHX ; ...
203         -sy_ez*DHY , sx_ez*DHX , ZZ ];
204
205  % BUILD SCATTERED-FIELD MASKING MATRIX
206  nx1 = NPML(1) + 2;
207  nx2 = Nx - NPML(2) - 1;
208  ny1 = NPML(3) + 2;
209  ny2 = Ny - NPML(4) - 1;
210  Q   = ones(Nx,Ny);
211  Q(nx1:nx2,ny1:ny2) = 0;
212  Q   = diagonalize(Q);
213  Q   = [ Q ZZ ZZ ; ZZ Q ZZ ; ZZ ZZ Q ];
```

```
214
215  % CALCULATE WAVE MATRIX
216  A = CH*YR*CE - ER;
217
218  % CALCULATE SOURCE VECTOR
219  b = (Q*A - A*Q)*fsrc;
220
221  % Solve for Field
222  disp('Solving...'); drawnow;
223  f = A\b;
224
225  % EXTRACT FIELD COMPONENTS
226  fx = f(1:M);
227  fy = f(M+1:2*M);
228  fz = f(2*M+1:3*M);
229
230  % RESHAPE BACK TO TWO-DIMENSIONAL GRID
231  fx = reshape(fx,Nx,Ny);
232  fy = reshape(fy,Nx,Ny);
233  fz = reshape(fz,Nx,Ny);
234
235  % VISUALIZE RESULT
236  clf;
237  pcolor(xa,ya,real(fz).');
238  shading interp;
239
240  hold on;
241  phi = linspace(0,2*pi,100);
242  x = cos(phi);
243  y = sin(phi);
244  line(R1*x,R1*y,'Color','w');
245  line(R2*x,R2*y,'Color','w');
246  hold off;
247
248  axis equal tight;
249  colorbar;
250  set(gca,'YDir','reverse');
```