

Table of Contents

Contents

Table of Contents	1
Chapter 1	2
GIF Animation	2
Chapter 3	3
One-Dimensional Finite-Difference Method	3
Multi-Variable Finite-Difference Method	4
Chapter 4	5
Demonstrate YEEDER2D()	5
YEEDER2D() – Two-Dimensional Derivative Matrices on Yee Grid	5
Demonstrate YEEDER3D()	7
YEEDER3D() – Three-Dimensional Derivative Matrices on Yee Grid	8
Chapter 5	10
Demonstrate ADDUPML2D()	10
ADDUPML2D() – Add Two-Dimensional Uniaxial Perfectly Matched Layer to Grid	11
Demonstrate CALCPML3D()	13
CALCPML3D() – Calculate Three-Dimensional Perfectly Matched Layer	13
Chapter 6	15
Rib Waveguide Analysis	15
Slab Waveguide Analysis	18
Animating the Slab Waveguide Analysis	21
Calculating Surface Plasmon Polaritons (SPPs)	24
Microstrip Transmission Line Analysis	26
Chapter 7	31
Calculating a Two-Dimensional Photonic Band Diagram	31
Calculating a Two-Dimensional Isofrequency Contours	34
Chapter 8	37
Sawtooth Diffraction Grating	37
Self-Collimating Photonic Crystal	41
Integrated Optical Directional Coupler	45
Chapter 9	51
FDFD2D() – Generic Two-Dimensional FDFD for Periodic Structures	51
Main Program to Simulate a Two-Dimensional Guided-Mode Resonance Filter	53
Sweep Depth of Metal Polarizer	56
Sweep Frequency of a Metal Polarizer	59

Chapter 10 63

 FDFD3D() – Generic FDFD Function to Simulate Three-Dimensional Periodic Structures 63

 Main Program to Simulate a Three-Dimensional Guided-Mode Resonance Filter 69

 Main Function to Simulate a Doubly-Periodic Frequency Selective Surface 72

 Main Function to Perform Parameter Retrieval of a Left-Handed Metamaterial..... 75

 Program to Simulate a Two-Dimensional Invisibility Cloak 82

Chapter 1

GIF Animation

```

1  % Chapter1_GIFanimation.m
2
3  % INITIALIZE MATLAB
4  close all;
5  clc;
6  clear all;
7
8  % UNITS
9  degrees = pi/180;
10
11 % DASHBOARD GRID
12 lam0 = 0.1;
13 theta = 30*degrees;
14 Sx = 1;
15 Sy = 1;
16 Nx = 100;
17 Ny = 100;
18
19 NFRAMES = 40;
20 gif_name = 'FDFD_animation.gif';
21
22 % CALCULATE MESHGRID
23 dx = Sx/Nx;
24 xa = [1:Nx]*dx;
25 xa = xa - mean(xa);
26 dy = Sy/Ny;
27 ya = [1:Ny]*dy;
28 ya = ya - mean(ya);
29 [Y,X] = meshgrid(ya,xa);
30
31 % CALCULATE A WAVE (REPLACES FDFD SIMULATION)
32 k0 = 2*pi/lam0;
33 kx = k0*sin(theta);
34 ky = k0*cos(theta);
35 f = exp(-1i*(kx*X + ky*Y));
36
37 % VISUALIZE USING IMAGESC
38 for nframe = 1 : NFRAMES
39
40     % Add Phase
41     phase = 2*pi*nframe/NFRAMES;

```

```
42     fi = f*exp(1i*phase);
43
44     % Draw Field
45     pcolor(xa,ya,real(fi).');
46     shading interp;
47     axis equal tight off;
48     set(gca,'YDir','reverse');
49     drawnow;
50
51     % Capture Frame
52     F = getframe(gca);
53     F = frame2im(F);
54     [ind,cmap] = rgb2ind(F,256,'nodither');
55
56     % Add Frame to GIF
57     if nframe == 1
58         imwrite(ind,cmap,gif_name,'gif',...
59             'DelayTime',0,'Loopcount',inf);
60     else
61         imwrite(ind,cmap,gif_name,'gif',...
62             'DelayTime',0,'WriteMode','append');
63     end
64 end
```

Chapter 3

One-Dimensional Finite-Difference Method

```
1 % Chapter3_fdm1d.m
2
3 % INITIALIZE MATLAB
4 close all;
5 clc;
6 clear all;
7
8 % DEFINE AND CALCULATE GRID
9 a = 0;
10 b = 10;
11 Nx = 100;
12 xa = linspace(a,b,Nx);
13 dx = xa(2) - xa(1);
14
15 % BUILD DERIVATIVE MATRIX DX
16 d = ones(Nx,1)/(2*dx);
17 DX = spdiags(-d,-1,sparse(Nx,Nx));
18 DX = spdiags(+d,+1,DX);
19 DX(Nx,Nx-1:Nx) = [-1 +1]/dx;
20
21 % BUILD MATRIX EQUATION A*f=b
22 A = DX;
23 b = -(1/3)*ones(Nx,1);
24
25 % ADD BOUNDARY VALUE f(0)=1
26 A(1,:) = 0;
27 A(1,1) = 1;
28 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 %
```

```

18 % OUTPUT ARGUMENTS
19 % =====
20 % DEX Derivative Matrix wrt x for Electric Fields
21 % DEY Derivative Matrix wrt to y for Electric Fields
22 % DHX Derivative Matrix wrt to x for Magnetic Fields
23 % DHY Derivative Matrix wrt to y for Magnetic Fields
24
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26 %% HANDLE INPUT ARGUMENTS
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28
29 % EXTRACT GRID PARAMETERS
30 Nx = NS(1); dx = RES(1);
31 Ny = NS(2); dy = RES(2);
32
33 % DEFAULT KINC
34 if ~exist('kinc')
35     kinc = [0 0];
36 end
37
38 % DETERMINE MATRIX SIZE
39 M = Nx*Ny;
40
41 % ZERO MATRIX
42 Z = sparse(M,M);
43
44 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
45 %% BUILD DEX
46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47
48 % HANDLE IF SIZE IS 1 CELL
49 if Nx==1
50     DEX = -1i*kinc(1)*speye(M,M);
51
52 % HANDLE ALL OTHER CASES
53 else
54
55     % Center Diagonal
56     d0 = -ones(M,1);
57
58     % Upper Diagonal
59     d1 = ones(M,1);
60     d1(Nx+1:Nx:M) = 0;
61
62     % Build Derivative Matrix with Dirichlet BCs
63     DEX = spdiags([d0 d1]/dx, [0 1], Z);
64
65     % Incorporate Periodic Boundary Conditions
66     if BC(1)==1
67         d1 = zeros(M,1);
68         d1(1:Nx:M) = exp(-1i*kinc(1)*Nx*dx)/dx;
69         DEX = spdiags(d1,1-Nx,DEX);
70     end
71
72 end
73

```

```

74 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
75 %% BUILD DEY
76 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
77
78 % HANDLE IF SIZE IS 1 CELL
79 if Ny==1
80     DEY = -li*kinc(2)*speye(M,M);
81
82 % HANDLE ALL OTHER CASES
83 else
84
85     % Center Diagonal
86     d0 = -ones(M,1);
87
88     % Upper Diagonal
89     d1 = ones(M,1);
90
91     % Build Derivative Matrix with Dirichlet BCs
92     DEY = spdiags([d0 d1]/dy,[0 Nx],Z);
93
94     % Incorporate Periodic Boundary Conditions
95     if BC(2)==1
96         d1 = (exp(-li*kinc(2)*Ny*dy)/dy)*ones(M,1);
97         DEY = spdiags(d1,Nx-M,DEY);
98     end
99
100 end
101
102 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
103 %% BUILD DHX AND DHY
104 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
105
106 DHX = -DEX';
107 DHY = -DEY';

```

Demonstrate YEEDER3D()

```

1 % Chapter4_yeeder3d_demo.m
2
3 % INITIALIZE MATLAB
4 close all;
5 clc;
6 clear all;
7
8 % DEFINE INPUT ARGUMENTS FOR YEEDER2D
9 NS = [2 2 2];
10 RES = [0.3 0.2 0.1];
11 BC = [0 0 0];
12
13 % CALL YEEDER3D
14 [DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(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('DEZ = ');
24 disp(full(DEZ));
25
26 disp('DHX = ');
27 disp(full(DHX));
28
29 disp('DHY = ');
30 disp(full(DHY));
31
32 disp('DHZ = ');
33 disp(full(DHZ));

```

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

```

1 function [DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(NS,RES,BC,kinc)
2 % YEEDER3D Derivative Matrices on a 3D Yee Grid
3 %
4 % [DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(NS,RES,BC,kinc);
5 %
6 % INPUT ARGUMENTS
7 % =====
8 % NS [Nx Ny Nz] Grid Size
9 % RES [dx dy dz] Grid Resolution
10 % BC [xbc ybc zbc] Boundary Conditions
11 % 0: Dirichlet boundary conditions
12 % 1: Periodic boundary conditions
13 % kinc [kx ky kz] Incident Wave Vector
14 % This argument is only needed for PBCs.
15 %
16 % Note: For normalized grids use k0*RES and kinc/k0
17 %
18 % OUTPUT ARGUMENTS
19 % =====
20 % DEX Derivative Matrix wrt x for Electric Fields
21 % DEY Derivative Matrix wrt y for Electric Fields
22 % DEZ Derivative Matrix wrt z for Electric Fields
23 % DHX Derivative Matrix wrt x for Magnetic Fields
24 % DHY Derivative Matrix wrt y for Magnetic Fields
25 % DHZ Derivative Matrix wrt z for Magnetic Fields
26
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28 %% HANDLE INPUT ARGUMENTS
29 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30
31 % EXTRACT GRID PARAMETERS
32 Nx = NS(1); dx = RES(1);
33 Ny = NS(2); dy = RES(2);
34 Nz = NS(3); dz = RES(3);
35
36 % DEFAULT KINC
37 if ~exist('kinc')
38 kinc = [0 0 0];
39 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]);
105
106        DEY = spdiags(ph*d1,-Nx*(Ny-1),DEY);
107    end
108
109 end
110
111 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
112 %% BUILD DEZ
113 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
114
115 % HANDLE IF SIZE IS 1 CELL
116 if Nz==1
117     DEZ = -1i*kinc(3)*speye(M,M);
118
119 % HANDLE ALL OTHER CASES
120 else
121
122     % Center Diagonal
123     d0 = ones(M,1);
124
125     % Build Derivative Matrix with Dirichlet BCs
126     DEZ = spdiags([-d0 +d0]/dz,[0 Nx*Ny],Z);
127
128     % Incorporate Periodic Boundary Conditions
129     if BC(3)==1
130         d0 = (exp(-1i*kinc(3)*Nz*dz)/dz)*ones(M,1);
131         DEZ = spdiags(d0,-Nx*Ny*(Nz-1),DEZ);
132     end
133
134 end
135
136 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137 %% BUILD DHX, DHY AND DHZ
138 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
139
140 DHX = -DEX';
141 DHY = -DEY';
142 DHZ = -DEZ';

```

Chapter 5

Demonstrate ADDUPML2D()

```

1 % Chapter5_addupml2d_demo.m
2
3 % INITIALIZE MATLAB
4 close all;

```

```
5  clc;
6  clear all;
7
8  % DEFINE GRID
9  Nx = 5;
10 Ny = 5;
11
12 % DEFINE PML
13 NPML = [2 2 2 2];
14
15 % BUILD ER2 AND UR2 ARRAYS
16 ER2 = ones(2*Nx,2*Ny);
17 UR2 = ones(2*Nx,2*Ny);
18
19 % CALL ADDUPML2D
20 [ERxx,ERyy,ERzz,URxx,URyy,URzz] ...
21     = addupml2d(ER2,UR2,NPML);
22
23 % DISPLAY THE RESULTS
24 disp('ERxx =');
25 disp(ERxx);
26
27 disp('ERyy =');
28 disp(ERyy);
29
30 disp('ERzz =');
31 disp(ERzz);
32
33 disp('URxx =');
34 disp(URxx);
35
36 disp('URyy =');
37 disp(URyy);
38
39 disp('URzz =');
40 disp(URzz);
```

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

```
1  function [ERxx,ERyy,ERzz,URxx,URyy,URzz] = addupml2d(ER2,UR2,NPML)
2  % ADDUPML2D      Add UPML to a 2D Yee Grid
3  %
4  % [ERxx,ERyy,ERzz,URxx,URyy,URzz] = addupml2d(ER2,UR2,NPML);
5  %
6  % INPUT ARGUMENTS
7  % =====
8  % ER2          Relative Permittivity on 2x Grid
9  % UR2          Relative Permeability on 2x Grid
10 % NPML         [NXLO NXHI NYLO NYHI] Size of UPML on 1x Grid
11 %
12 % OUTPUT ARGUMENTS
13 % =====
14 % ERxx        xx Tensor Element for Relative Permittivity
15 % ERyy        yy Tensor Element for Relative Permittivity
16 % ERzz        zz Tensor Element for Relative Permittivity
17 % URxx        xx Tensor Element for Relative Permeability
```

```
18 % URyy      yy Tensor Element for Relative Permeability
19 % URzz      zz Tensor Element for Relative Permeability
20
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22 %% INITIALIZE FUNCTION
23 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24
25 % DEFINE PML PARAMETERS
26 amax = 4;
27 cmax = 1;
28 p    = 3;
29
30 % EXTRACT GRID PARAMETERS
31 [Nx2,Ny2] = size(ER2);
32
33 % EXTRACT PML PARAMETERS
34 NXLO = 2*NPML(1);
35 NXHI = 2*NPML(2);
36 NYLO = 2*NPML(3);
37 NYHI = 2*NPML(4);
38
39 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
40 %% CALCULATE PML PARAMETERS
41 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42
43 % INITIALIZE PML PARAMETERS TO PROBLEM SPACE
44 sx = ones(Nx2,Ny2);
45 sy = ones(Nx2,Ny2);
46
47 % ADD XLO PML
48 for nx = 1 : NXLO
49     ax = 1 + (amax - 1)*(nx/NXLO)^p;
50     cx = cmax*sin(0.5*pi*nx/NXLO)^2;
51     sx(NXLO - nx + 1,:) = ax*(1 - 1i*60*cx);
52 end
53
54 % ADD XHI PML
55 for nx = 1 : NXHI
56     ax = 1 + (amax - 1)*(nx/NXHI)^p;
57     cx = cmax*sin(0.5*pi*nx/NXHI)^2;
58     sx(Nx2 - NXHI + nx,:) = ax*(1 - 1i*60*cx);
59 end
60
61 % ADD YLO PML
62 for ny = 1 : NYLO
63     ay = 1 + (amax - 1)*(ny/NYLO)^p;
64     cy = cmax*sin(0.5*pi*ny/NYLO)^2;
65     sy(:,NYLO - ny + 1) = ay*(1 - 1i*60*cy);
66 end
67
68 % ADD YHI PML
69 for ny = 1 : NYHI
70     ay = 1 + (amax - 1)*(ny/NYHI)^p;
71     cy = cmax*sin(0.5*pi*ny/NYHI)^2;
72     sy(:,Ny2 - NYHI + ny) = ay*(1 - 1i*60*cy);
73 end
```

```

74
75 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76 %% INCORPORATE PML
77 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78
79 % CALCULATE TENSOR ELEMENTS WITH UPML
80 ERxx = ER2./sx.*sy;
81 ERYy = ER2.*sx./sy;
82 ERzz = ER2.*sx.*sy;
83
84 URxx = UR2./sx.*sy;
85 URYy = UR2.*sx./sy;
86 URzz = UR2.*sx.*sy;
87
88 % EXTRACT TENSOR ELEMENTS ON YEE GRID
89 ERxx = ERxx(2:2:Nx2,1:2:Ny2);
90 ERYy = ERYy(1:2:Nx2,2:2:Ny2);
91 ERzz = ERzz(1:2:Nx2,1:2:Ny2);
92
93 URxx = URxx(1:2:Nx2,2:2:Ny2);
94 URYy = URYy(2:2:Nx2,1:2:Ny2);
95 URzz = URzz(2:2:Nx2,2:2:Ny2);

```

Demonstrate CALCPML3D()

```

1 % Chapter5_calcpml3d_demo.m
2
3 % INITIALIZE MATLAB
4 close all;
5 clc;
6 clear all;
7
8 % DEFINE INPUT ARGUMENTS
9 NGRID = [6 6 6];
10 NPML = [2 2 2 2 2 2];
11
12 % CALL CALCPML3D
13 [sx,sy,sz] = calcpml3d(NGRID,NPML);
14
15 % DISPLAY THE RESULTS
16 disp('sx =');
17 disp(sx);
18
19 disp('sy =');
20 disp(sy);
21
22 disp('sz =');
23 disp(sz);

```

CALCPML3D() – Calculate Three-Dimensional Perfectly Matched Layer

```

1 function [sx,sy,sz] = calcpml3d(NGRID,NPML)
2 % CALCPML3D Calculate PML Parameters
3 %
4 % [sx,sy,sz] = calcpml3d(NGRID,NPML);
5 %
6 % INPUT ARGUMENTS

```

```

7  % =====
8  % NGRID      [Nx Ny Nz] Number of Cells on Grid
9  % NPML      [NXLO NXHI NYLO NYHI NZLO NZHI] Size of PML
10
11  %%%%%%%%%%%
12  %% HANDLE INPUT AND OUTPUT ARGUMENTS
13  %%%%%%%%%%%
14
15  % DEFINE PML PARAMETERS
16  amax = 4;
17  cmax = 1;
18  p    = 3;
19
20  % EXTRACT GRID SIZE
21  Nx = NGRID(1);
22  Ny = NGRID(2);
23  Nz = NGRID(3);
24
25  % EXTRACT PML SIZE
26  NXLO = NPML(1);
27  NXHI = NPML(2);
28  NYLO = NPML(3);
29  NYHI = NPML(4);
30  NZLO = NPML(5);
31  NZHI = NPML(6);
32
33  %%%%%%%%%%%
34  %% CALCULATE PML PARAMETERS
35  %%%%%%%%%%%
36
37  % INITIALIZE PML PARAMETERS TO PROBLEM SPACE
38  sx = ones(Nx,Ny,Nz);
39  sy = ones(Nx,Ny,Nz);
40  sz = ones(Nx,Ny,Nz);
41
42  % CALCULATE XLO PML
43  for nx = 1 : NXLO
44      ax = 1 + (amax - 1)*(nx/NXLO)^p;
45      cx = cmax*sin(0.5*pi*nx/NXLO)^2;
46      sx(NXLO - nx + 1, :, :) = ax*(1 - 1i*60*cx);
47  end
48
49  % CALCULATE XHI PML
50  for nx = 1 : NXHI
51      ax = 1 + (amax - 1)*(nx/NXHI)^p;
52      cx = cmax*sin(0.5*pi*nx/NXHI)^2;
53      sx(Nx - NXHI + nx, :, :) = ax*(1 - 1i*60*cx);
54  end
55
56  % CALCULATE YLO PML
57  for ny = 1 : NYLO
58      ay = 1 + (amax - 1)*(ny/NYLO)^p;
59      cy = cmax*sin(0.5*pi*ny/NYLO)^2;
60      sy(:, NYLO - ny + 1, :) = ay*(1 - 1i*60*cy);
61  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
32 % NUMBER OF MODES TO CALCULATE
    
```

```

33 NMODES = 3;
34
35 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36 %% COMPUTE OPTIMIZED GRID
37 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
38
39 % FIRST GUESS AT GRID RESOLUTION
40 dx = lam0/nmax/NRES;
41 dy = lam0/nmax/NRES;
42
43 % SNAP GRID TO CRITICAL DIMENSIONS
44 nx = ceil(rib_w/dx);
45 dx = rib_w/nx;
46 ny = ceil(rib_t/dy);
47 dy = rib_t/ny;
48
49 % GRID SIZE
50 Sx = SPACER(1) + rib_w + SPACER(2);
51 Nx = ceil(Sx/dx);
52 Sx = Nx*dx;
53
54 Sy = SPACER(3) + rib_h + rib_t + SPACER(4);
55 Ny = ceil(Sy/dy);
56 Sy = Ny*dy;
57
58 % 2X GRID
59 Nx2 = 2*Nx;          dx2 = dx/2;
60 Ny2 = 2*Ny;          dy2 = dy/2;
61
62 % GRID AXES
63 xa = [1:Nx]*dx;      xa = xa - mean(xa);
64 ya = [1:Ny]*dy;      ya = ya - mean(ya);
65 xa2 = [1:Nx2]*dx2;   xa2 = xa2 - mean(xa2);
66 ya2 = [1:Ny2]*dy2;   ya2 = ya2 - mean(ya2);
67
68 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69 %% BUILD DEVICE
70 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
71
72 % INITIALIZE TO AIR
73 ER2 = rib_n1^2*ones(Nx2,Ny2);
74 UR2 = ones(Nx2,Ny2);
75
76 % CALCULATE POSITION INDICES
77 nx1 = 1 + round(SPACER(1)/dx2);
78 nx2 = nx1 + round(rib_w/dx2) - 1;
79
80 ny1 = 1 + round(SPACER(3)/dy2);
81 ny2 = ny1 + round(rib_h/dy2) - 1;
82 ny3 = ny2 + 1;
83 ny4 = ny3 + round(rib_t/dy2) - 1;
84 ny5 = ny4 + 1;
85
86 % BUILD RIB WAVEGUIDE
87 ER2(nx1:nx2,ny1:ny2) = rib_n2^2;
88 ER2(:,ny3:ny4)       = rib_n2^2;

```

```

89  ER2(:,ny5:Ny2)      = rib_n3^2;
90
91  % EXTRACT MATERIALS TENSORS
92  ERxx = ER2(2:2:Nx2,1:2:Ny2);
93  ERYy = ER2(1:2:Nx2,2:2:Ny2);
94  ERzz = ER2(1:2:Nx2,1:2:Ny2);
95
96  URxx = UR2(1:2:Nx2,2:2:Ny2);
97  URYy = UR2(2:2:Nx2,1:2:Ny2);
98  URzz = UR2(2:2:Nx2,2:2:Ny2);
99
100 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
101 %% PERFORM FINITE-DIFFERENCE ANALYSIS
102 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
103
104 % DIAGONALIZE MATERIAL ARRAYS
105 ERxx = diag(sparse(ERxx(:)));
106 ERYy = diag(sparse(ERYy(:)));
107 ERzz = diag(sparse(ERzz(:)));
108 URxx = diag(sparse(URxx(:)));
109 URYy = diag(sparse(URYy(:)));
110 URzz = diag(sparse(URzz(:)));
111
112 % BUILD DERIVATE MATRICES
113 k0 = 2*pi/lam0;
114 NS = [Nx Ny];
115 RES = [dx dy];
116 BC = [0 0];
117 [DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);
118
119 % BUILD P AND Q MATRICES
120 P = [ DEX/ERzz*DHY , -(DEX/ERzz*DHX + URYy) ; ...
      DEY/ERzz*DHY + URxx , -DEY/ERzz*DHX ];
121 Q = [ DHX/URzz*DEY , -(DHX/URzz*DEX + ERYy) ; ...
      DHY/URzz*DEY + ERxx , -DHY/URzz*DEX ];
122
123 % SOLVE EIGEN-VALUE PROBLEM
124
125 ev = -rib_n2^2;
126 [Exy,D2] = eigs(P*Q,NMODES,ev);
127 D = sqrt(D2);
128 NEFF = -1i*diag(D);
129
130 % CALCULATE OTHER FIELD COMPONENTS
131 M = Nx*Ny;
132 Ex = Exy(1:M,:);
133 Ey = Exy(M+1:2*M,:);
134
135 Hxy = (Q*Exy)/D;
136 Hx = Hxy(1:M,:);
137 Hy = Hxy(M+1:2*M,:);
138
139 Ez = ERzz\(DHX*Hy - DHY*Hx);
140 Hz = URzz\(DEX*Ey - DEY*Ex);
141
142 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
143 %% VISUALIZE THE MODES
144

```

```

145 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
146
147 % VISUALIZE CALCULATED MODES
148 for m = 1 : NMODES
149     % Extract Fields
150     ex = reshape(Ex(:,m),Nx,Ny);
151     ey = reshape(Ey(:,m),Nx,Ny);
152
153     % Scale Fields
154     fmax = max(abs([ ex(:) ; ey(:) ]));
155     ex = ex/fmax;
156     ey = ey/fmax;
157
158     % Plot Modes
159     subplot(NMODES,2,(m - 1)*2 + 1);
160     imagesc(xa,ya,abs(ex).');
161     axis equal tight off;
162     colorbar;
163     caxis([0 1]);
164     title(['Ex, n_{eff} = ' num2str(NEFF(m))]);
165
166     subplot(NMODES,2,(m - 1)*2 + 2);
167     imagesc(xa,ya,abs(ey).');
168     axis equal tight off;
169     colorbar;
170     caxis([0 1]);
171     title('Ey');
172
173 end

```

Slab Waveguide Analysis

```

1 % Chapter6_slabwaveguide.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 OPTIMIZED 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 %% VISUALIZE THE GUIDED MODES
119 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
120
121 % PREPARE THE FIGURE WINDOW
122 clf;
123 hold on;
124
125 % DRAW CORE
126 x = NMODES*[0 1 1 0 0] - 0.5;
127 y = 0.5*a*[-1 -1 +1 +1 -1];
128 fill(x,y,0.8*[1 1 1]);
129
130 % DRAW THE MODES
131 Fy = Fy/max(abs(Fy(:)));
132 for m = 1 : NMODES
133     x0 = m - 1;
134     line(x0+0.4*Fy(:,m),xa,'LineWidth',3,'Color','w');
135     line(x0+0.4*Fy(:,m),xa,'LineWidth',1,'Color','b');
136     T = ['$n_{' num2str(m) ',\text{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 OPTIMIZED GRID
36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37
38 % FIRST GUESS AT GRID RESOLUTION
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 OPTIMIZED 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

```
1 % Chapter6_microstrip.m
```

```
2
3 % INITIALIZE MATLAB
4 close all;
5 clc;
6 clear all;
7
8 % UNITS
9 meters = 1;
10 centimeters = 1e-2 * meters;
11 millimeters = 1e-3 * meters;
12 micrometers = 1e-6 * meters;
13 seconds = 1;
14 hertz = 1/seconds;
15 kilohertz = 1e3*hertz;
16 megahertz = 1e6*hertz;
17 gigahertz = 1e9*hertz;
18
19 % CONSTANTS
20 c0 = 299792458 * meters/seconds;
21 N0 = 376.73031346177;
22 e0 = 8.8541878176e-12 * 1/meters;
23 u0 = 1.2566370614e-6 * 1/meters;
24
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26 %% DASHBOARD
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28
29 % SOURCE
30 freq = 1.0 * gigahertz;
31 lam0 = c0/freq;
32
33 % MATERIAL PROPERTIES
34 sigma = 5.8e7;
35 erm = 1 + sigma/(1i*2*pi*freq*e0);
36
37 er = 4.4;
38 tand = 0.03;
39 erd = er*(1 - 1i*tand);
40
41 era = 1.0;
42
43 % MICROSTRIP PARAMETERS
44 w = 1 * millimeters;
45 h = 0.5 * millimeters;
46 t = 35 * micrometers;
47
48 % GRID PARAMETERS
49 nmax = sqrt(real(erd));
50 NRES = 40;
51 NDIM = 10;
52 SPACER = w*[3 3 2 0];
53
54 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
55 %% COMPUTE OPTIMIZED GRID
56 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
57
```

```

58 % GRID RESOLUTION
59 dx = lam0/nmax/NRES;
60 dy = lam0/nmax/NRES;
61
62 % RESOLVE MINIMUM DIMENSIONS
63 nx = round(w/dx);
64 if nx<NDIM
65     nx = NDIM;
66     dx = w/nx;
67 end
68
69 ny = round(t/dy);
70 if ny<NDIM
71     ny = NDIM;
72     dy = t/ny;
73 end
74
75 % SNAP GRID TO CRITICAL DIMENSIONS
76 nx = ceil(w/dx);
77 dx = w/nx;
78
79 ny = ceil(h/dy);
80 dy = h/ny;
81
82 % COMPUTE GRID SIZE
83 Sx = SPACER(1) + w + SPACER(2);
84 Nx = ceil(Sx/dx);
85 Sx = Nx*dx;
86
87 Sy = SPACER(3) + t + h + SPACER(4);
88 Ny = ceil(Sy/dy) + 1;
89 Sy = Ny*dy;
90
91 % GRID AXES
92 xa = [1:Nx]*dx;          xa = xa - mean(xa);
93 ya = [0:Ny-1]*dy;
94
95 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
96 %% BUILD DEVICE ON GRID
97 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
98
99 % INITIALIZE MATERIALS TO FREE SPACE
100 URxx = ones(Nx,Ny);
101 URyy = ones(Nx,Ny);
102 URzz = ones(Nx,Ny);
103 ERxx = era*ones(Nx,Ny);
104 ERYy = era*ones(Nx,Ny);
105 ERzz = era*ones(Nx,Ny);
106
107 % COMPUTE POSITION INDICES
108 nx = round(w/dx);
109 nx1 = 1 + floor((Nx - nx)/2);
110 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
44 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
45
46 % BUILD UNIT CELL
```

```

47 ER2 = (X2.^2 + Y2.^2) <= r^2;
48 ER2 = erfill + (erhole - erfill)*ER2;
49 UR2 = ones(Nx2,Ny2);
50
51 % EXTRACT YEE GRID MATERIAL ARRAYS
52 ERxx = ER2(2:2:Nx2,1:2:Ny2);
53 ERyy = ER2(1:2:Nx2,2:2:Ny2);
54 ERzz = ER2(1:2:Nx2,1:2:Ny2);
55 URxx = UR2(1:2:Nx2,2:2:Ny2);
56 URyy = UR2(2:2:Nx2,1:2:Ny2);
57 URzz = UR2(2:2:Nx2,2:2:Ny2);
58
59 % SHOW UNIT CELL
60 subplot(131);
61 imagesc(xa2,ya2,ER2. ');
62 axis equal tight;
63 colorbar;
64 title('Unit Cell');
65
66 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
67 %% CALCUALTE LIST OF BLOCH WAVE VECTORS
68 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69
70 % RECIPROCAL LATTICE VECTORS
71 T1 = (2*pi/a) * [ 1 ; 0 ];
72 T2 = (2*pi/a) * [ 0 ; 1 ];
73
74 % KEY POINTS OF SYMMETRY
75 G = [ 0 ; 0 ];
76 X = 0.5*T1;
77 M = 0.5*T1 + 0.5*T2;
78
79 % CHOOSE PATH AROUND IBZ
80 KP = [ G X M G ];
81 KL = { '\Gamma' 'X' 'M' '\Gamma' };
82
83 % DETERMINE LENGTH OF IBZ PERIMETER
84 NKP = length(KP);
85 LIBZ = 0;
86 for m = 1 : NKP-1
87     LIBZ = LIBZ + norm(KP(:,m+1) - KP(:,m));
88 end
89
90 % GENERATE LIST OF POINTS AROUND IBZ
91 dibz = LIBZ/NBETA;
92 BETA = KP(:,1);
93 KT = 1;
94 NBETA = 1;
95 for m = 1 : NKP-1
96     dK = KP(:,m+1) - KP(:,m);
97     N = ceil(norm(dK)/dibz);
98     BETA = [ BETA , KP(:,m) + dK*[1:N]/N ];
99     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 $\omega_{\text{trm}\{n\}} = a/\lambda_0$',...
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;
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
44 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
45
46 % BUILD UNIT CELL
47 ER2 = (X2.^2 + Y2.^2) <= r^2;
48 ER2 = erfill + (erhole - erfill)*ER2;
49 UR2 = ones(Nx2,Ny2);
50
51 % EXTRACT YEE GRID MATERIAL ARRAYS
52 ERxx = ER2(2:2:Nx2,1:2:Ny2);
53 ERYy = ER2(1:2:Nx2,2:2:Ny2);
54 ERzZ = ER2(1:2:Nx2,1:2:Ny2);
55 URxx = UR2(1:2:Nx2,2:2:Ny2);
56 URYy = UR2(2:2:Nx2,1:2:Ny2);
57 URzZ = UR2(2:2:Nx2,2:2:Ny2);
58
59 % SHOW UNIT CELL
60 subplot(131);
61 imagesc(xa2, ya2, ER2. ');
62 axis equal tight;
63 colorbar;
64 title('Unit Cell');
65
66 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
67 %% CALCUALTE LIST OF BLOCH WAVE VECTORS
68 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69
70 % CREATE BETA MESHGRID
71 bx      = linspace(-pi/a,+pi/a,NBx);
72 by      = linspace(-pi/a,+pi/a,NBY);
73 [BY,BX] = meshgrid(by,bx);
74
75 % EXTRACT IBZ AREA
76 ind_ibz = find(BX>=0 & BY>=0 & BX>=BY);
77 ind_x_ibz = 1 + mod(ind_ibz-1,NBx);
78 ind_y_ibz = 1 + floor((ind_ibz-1)/NBx);
79 NBETA     = length(ind_ibz);
80
81 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82 %% PERFORM FDFD ANALYSIS
83 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
84
85 % DIAGONALIZE MATERIAL TENSORS
86 ERxx = diag(sparse(ERxx(:)));
87 ERYy = diag(sparse(ERYy(:)));
88 ERzZ = diag(sparse(ERzZ(:)));
89 URxx = diag(sparse(URxx(:)));
90 URYy = diag(sparse(URYy(:)));
91 URzZ = diag(sparse(URzZ(:)));
92
93 % INITIALIZE BAND DATA

```

```

94  WNTE = zeros (NBx,NBy,NBANDS) ;
95  WNTM = zeros (NBx,NBy,NBANDS) ;
96
97  %
98  % MAIN LOOP -- ITERATE OVER IBZ
99  %
100 for nbeta = 1 : NBETA
101
102     % Get Next Block Wave Vector
103     beta = [ BX(ind_ibz(nbeta)) ; BY(ind_ibz(nbeta)) ] ;
104
105     % Build Derivative Matrices
106     NS = [Nx Ny];
107     RES = [dx dy];
108     BC = [1 1];
109     [DEX,DEY,DHX,DHY] = yeeder2d(NS,RES,BC,beta) ;
110
111     % TM Mode Analysis
112     A = - DHX/URyy*DEX - DHY/URxx*DEY;
113     B = ERzz;
114     D = eigs (A,B,NBANDS,0) ;
115     D = real (sqrt (D)) ;
116     D = sort (D) ;
117     WNTM(indx_ibz (nbeta),indy_ibz (nbeta),:) ...
118         = a/(2*pi)*D(1:NBANDS) ;
119
120     % TE Mode Analysis
121     A = - DEX/ERYy*DHX - DEY/ERxx*DHY;
122     B = URzz;
123     D = eigs (A,B,NBANDS,0) ;
124     D = real (sqrt (D)) ;
125     D = sort (D) ;
126     WNTE(indx_ibz (nbeta),indy_ibz (nbeta),:) ...
127         = a/(2*pi)*D(1:NBANDS) ;
128
129 end
130
131 % UNFOLD DATA
132 for nb = 1 : NBANDS
133     wn = WNTE (:, :, nb) ;
134     wn = wn + wn.' ;
135     wn = wn - 0.5*diag (diag (wn)) ;
136     wn(1:floor (NBx/2), :) = wn (NBx:-1:2+floor (NBx/2), :) ;
137     wn (:, 1:floor (NBy/2)) = wn (:, NBy:-1:2+floor (NBy/2)) ;
138     WNTE (:, :, nb) = wn ;
139
140     wn = WNTM (:, :, nb) ;
141     wn = wn + wn.' ;
142     wn = wn - 0.5*diag (diag (wn)) ;
143     wn(1:floor (NBx/2), :) = wn (NBx:-1:2+floor (NBx/2), :) ;
144     wn (:, 1:floor (NBy/2)) = wn (:, NBy:-1:2+floor (NBy/2)) ;
145     WNTM (:, :, nb) = wn ;
146 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
4 close all;
5 clc;
6 clear all;
```

```

7
8 % UNITS
9 degrees = pi/180;
10 meters = 1;
11 centimeters = 1e-2 * meters;
12 millimeters = 1e-3 * meters;
13 micrometers = 1e-6 * meters;
14 nanometers = 1e-9 * meters;
15 inches = 2.54 * centimeters;
16 feet = 12 * inches;
17 seconds = 1;
18 hertz = 1/seconds;
19 kilohertz = 1e3 * hertz;
20 megahertz = 1e6 * hertz;
21 gigahertz = 1e9 * hertz;
22 terahertz = 1e12 * hertz;
23 petahertz = 1e15 * hertz;
24
25 % CONSTANTS
26 e0 = 8.85418782e-12 * 1/meters;
27 u0 = 1.25663706e-6 * 1/meters;
28 N0 = sqrt(u0/e0);
29 c0 = 299792458 * meters/seconds;
30
31 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32 %% DASHBOARD
33 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
34
35 % DEFINE PLANE WAVE SOURCE PARAMETERS
36 f0 = 30.0 * gigahertz;
37 theta = 30*degrees;
38 lam0 = c0/f0;
39 k0 = 2*pi/lam0;
40 MODE = 'E';
41
42 % DEFINE DIFFRACTION GRATING PARAMETERS
43 L = 1.5 * centimeters;
44 d = 1.0 * centimeters;
45 er1 = 1.0;
46 er2 = 9.0;
47
48 % DEFINE FDFD PARAMETERS
49 NRES = 100;
50 SPACER = lam0*[1 1];
51 NPML = [20 20];
52 ermax = max([er1 er2]);
53 nmax = sqrt(ermax);
54
55 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56 %% CALCULATE OPTIMIZED GRID
57 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
58
59 % GRID RESOLUTION
60 dx = lam0/nmax/NRES;
61 dy = lam0/nmax/NRES;
62
    
```

```

63 % SNAP GRID TO CRITICAL DIMENSIONS
64 nx = ceil(L/dx);
65 dx = L/nx;
66
67 ny = ceil(d/dy);
68 dy = d/ny;
69
70 % GRID SIZE
71 Sx = L;
72 Nx = ceil(Sx/dx);
73 Sx = Nx*dx;
74
75 Sy = SPACER(1) + d + SPACER(2);
76 Ny = NPML(1) + ceil(Sy/dy) + NPML(1);
77 Sy = Ny*dy;
78
79 % 2X GRID
80 Nx2 = 2*Nx;           dx2 = dx/2;
81 Ny2 = 2*Ny;           dy2 = dy/2;
82
83 % CALCULATE AXIS VECTORS
84 xa = [1:Nx]*dx;
85 ya = [1:Ny]*dy;
86 [Y,X] = meshgrid(ya,xa);
87 xa2 = [1:Nx2]*dx2;
88 ya2 = [1:Ny2]*dy2;
89
90 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91 %% BUILD DIFFRACTION GRATING
92 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
93
94 % INITIALIZE TO AIR
95 ER2 = er1*ones(Nx2,Ny2);
96 UR2 = ones(Nx2,Ny2);
97
98 % BUILD SAWTOOTH
99 ny1 = 2*NPML(1) + round(SPACER(1)/dy2) + 1;
100 ny2 = ny1 + round(d/dy2) - 1;
101 for ny = ny1 : ny2
102     f = (ny - ny1 + 1)/(ny2 - ny1 + 1);
103     nx = round(f*L/dx2);
104     ER2(Nx2-nx+1:Nx2,ny) = er2;
105 end
106
107 % ADD SUBSTRATE
108 ER2(:,ny2+1:Ny2) = er2;
109
110 % SHOW DIFFRACTION GRATING
111 subplot(121);
112 imagesc(xa2,ya2,ER2. ');
113 axis equal tight off;
114 colorbar;
115 title('ER2');
116
117 % INCORPORATE PML
118 [ERxx,ERyy,ERzz,URxx,URyy,URzz] ...

```

```

119             = addupml2d(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(-li*(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;
172 axis equal tight;
173 set(gca,'YDir','reverse');
174 caxis([-1 1]);
    
```

```

175 colorbar;
176
177 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
178 %% ANALYZE REFLECTION AND TRANSMISSION
179 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
180
181 % EXTRACT MATERIAL PROPERTIES IN EXTERNAL REGIONS
182 urref = UR2(1,1);
183 urtrn = UR2(1,Ny2);
184 erref = ER2(1,1);
185 ertrn = ER2(1,Ny2);
186 nref = sqrt(urref*erref);
187 ntrn = sqrt(urtrn*ertrn);
188
189 % CALCULATE WAVE VECTOR COMPONENTS
190 m = [-floor(Nx/2):+floor((Nx-1)/2)].';
191 kx = kxinc - m*2*pi/Sx;
192 kyref = sqrt((k0*nref)^2 - kx.^2);
193 kytrn = sqrt((k0*ntrn)^2 - kx.^2);
194
195 % EXTRACT REFLECTED AND TRANSMITTED WAVES
196 fref = f(:,NPML(1)+1)./fsrc(:,1);
197 ftrn = f(:,Ny-NPML(2))./fsrc(:,1);
198
199 % CALCULATE AMPLITUDES OF SPATIAL HARMONICS
200 aref = fftshift(fft(fref))/Nx;
201 atrn = fftshift(fft(ftrn))/Nx;
202
203 % CALCULATE DIFFRACTION EFFICIENCIES
204 RDE = abs(aref).^2.*real(kyref/kyinc);
205 if MODE == 'E'
206     TDE = abs(atrn).^2.*real(urref/urtrn*kytrn/kyinc);
207 else
208     TDE = abs(atrn).^2.*real(erref/ertrn*kytrn/kyinc);
209 end
210
211 % CALCULATE OVERALL REFLECTANCE & TRANSMITTANCE
212 REF = sum(RDE(:))
213 TRN = sum(TDE(:))
214 CON = REF + TRN
215
216 % SHOW RESULTS
217 ind = find((RDE(:)+TDE(:))>1e-4);
218 [ m(ind) 100*RDE(ind) 100*TDE(ind) ]

```

Self-Collimating Photonic Crystal

```

1 % Chapter8_photoniccrystal.m
2
3 % INITIALIZE MATLAB
4 close all;
5 clc;
6 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
63 Nx2 = 2*Nx;          dx2 = dx/2;
64 Ny2 = 2*Ny;          dy2 = dy/2;
65
```

```

66 % CALCULATE AXIS VECTORS
67 xa = [1:Nx]*dx;
68 xa = xa - NPML(1)*dx - SPACER(1);
69 ya = [1:Ny]*dy;
70 ya = ya - mean(ya);
71
72 xa2 = [1:Nx2]*dx2;
73 xa2 = xa2 - 2*NPML(1)*dx2 - SPACER(1);
74 ya2 = [1:Ny2]*dy2;
75 ya2 = ya2 - mean(ya2);
76
77 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
78 %% BUILD LATTICE
79 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80
81 % UNIT CELL GRID
82 Nxuc = round(b/dx2);
83 Nyuc = round(b/dy2);
84 xauc = [1:Nxuc]*dx2;      xauc = xauc - mean(xauc);
85 yauc = [1:Nyuc]*dy2;      yauc = yauc - mean(yauc);
86 [Y,X] = meshgrid(yauc,xauc);
87
88 % BUILD UNIT CELL
89 ERuc = (X.^2 + Y.^2) <= r^2;
90 ERuc = ERuc | ((X - b/2).^2 + (Y - b/2).^2) <= r^2;
91 ERuc = ERuc | ((X + b/2).^2 + (Y - b/2).^2) <= r^2;
92 ERuc = ERuc | ((X - b/2).^2 + (Y + b/2).^2) <= r^2;
93 ERuc = ERuc | ((X + b/2).^2 + (Y + b/2).^2) <= r^2;
94 ERuc = er2 + (er1 - er2)*ERuc;
95 URuc = ones(Nxuc,Nyuc);
96
97 % INITIALIZE MATERIALS ARRAYS
98 ER2 = ones(Nx2,Ny2);
99 UR2 = ones(Nx2,Ny2);
100
101 % STACK UNIT CELL TO BUILD LATTICE
102 for nyuc = 1 : round(Ly/b)
103     ny1 = 2*NPML(3) + (nyuc - 1)*Nyuc ...
104         + round(SPACER(3)/dy2) + 1;
105     ny2 = ny1 + Nyuc - 1;
106     for nxuc = 1 : round(Lx/b)
107         nx1 = 2*NPML(1) + (nxuc - 1)*Nxuc ...
108             + round(SPACER(1)/dx2) + 1;
109         nx2 = nx1 + Nxuc - 1;
110         ER2(nx1:nx2,ny1:ny2) = ERuc;
111         UR2(nx1:nx2,ny1:ny2) = URuc;
112     end
113 end
114
115 % SHOW UNIT CELL
116 subplot(131);
117 imagesc(xa2,ya2,ER2. ');
118 axis equal tight;
119 colorbar;
120 title('Lattice');
121

```

```

122 % INCORPORATE PML
123 [ERxx,ERyy,ERzz,URxx,URyy,URzz] = addupml2d(ER2,UR2,NPML);
124
125 % DIAGONALIZE MATERIAL TENSORS
126 ERxx = diag(sparse(ERxx(:)));
127 ERyy = diag(sparse(ERyy(:)));
128 ERzz = diag(sparse(ERzz(:)));
129 URxx = diag(sparse(URxx(:)));
130 URyy = diag(sparse(URyy(:)));
131 URzz = diag(sparse(URzz(:)));
132
133 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
134 %% PERFORM FDFD ANALYSIS
135 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
136
137 % BUILD DERIVATIVE MATRICES
138 NS = [Nx Ny];
139 RES = [dx dy];
140 BC = [0 0];
141 [DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);
142
143 % BUILD WAVE MATRIX
144 A = DHX/URyy*DEX + DHY/URxx*DEY + ERzz;
145
146 % CALCULATE SOURCE FIELD
147 nsrc = sqrt(UR2(1,1)*ER2(1,1));
148 [Y,X] = meshgrid(ya,xa);
149 [TH,R] = cart2pol(X,Y);
150 [XR,YR] = pol2cart(TH-theta,R);
151 fsrc = exp(-(YR/bw).^2).*exp(-1i*k0*nsrc*XR);
152 fsrc(X>0) = 0;
153
154 % CALCULATE SCATTERED-FIELD MASKING MATRIX
155 nx1 = NPML(1) + 2;
156 nx2 = Nx - NPML(2) - 1;
157 ny1 = NPML(3) + 2;
158 ny2 = Ny - NPML(4) - 1;
159 Q = ones(Nx,Ny);
160 Q(nx1:nx2,ny1:ny2) = 0;
161 Q = diag(sparse(Q(:)));
162
163 % CALCULATE SOURCE VECTOR
164 b = (Q*A - A*Q)*fsrc(:);
165
166 % SOLVE FOR FIELD
167 f = A\b;
168 f = reshape(f,Nx,Ny);
169
170 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
171 %% DRAW PRETTY SOLUTION
172 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
173
174 % PREPARE GRAPHICS WINDOW
175 clf;
176 hold on;
177

```

```
178 % DRAW FIELD
179 pcolor(xa,ya,real(f).');
180 shading interp;
181
182 % SUPERIMPOSE LATTICE
183 h = contour(xa2,ya2,ER2.',10,'Color','w','LineWidth',0.5);
184
185 % SET GRAPHICS VIEW
186 hold off;
187 axis equal tight;
188 set(gca,'YDir','reverse');
189 caxis([-1 1]);
190 colorbar;
```

Integrated Optical Directional Coupler

```
1 % Chapter8_directionalcoupler.m
2
3 % INITIALIZE MATLAB
4 close all;
5 clc;
6 clear all;
7
8 % UNITS
9 micrometers = 1;
10
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12 %% DASHBOARD
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14
15 % SOURCE
16 lam0 = 1.55 * micrometers;
17 k0 = 2*pi/lam0;
18
19 % RIB WAVEGUIDE PARAMETERS
20 rib_n1 = 1.0;
21 rib_n2 = 3.5;
22 rib_n3 = 1.5;
23 rib_h = 0.30 * micrometers;
24 rib_t = 0.10 * micrometers;
25 rib_w = 0.40 * micrometers;
26
27 % DIRECTIONAL COUPLER PARAMETERS
28 L1 = 0.5*lam0;
29 L2 = 5.0*lam0;
30 g = 0.1*lam0;
31
32 % FDFD PARAMETERS
33 NRES = 20;
34 SPACER = lam0*[2 2 1 1];
35 NPML = [20 20 20 20];
36 nmax = max([rib_n1 rib_n2 rib_n3]);
37
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
206     % DIAGONALIZE MATERIAL TENSORS
207     ERxx = diag(sparse(ERxx(:)));
208     ERyy = diag(sparse(ERyy(:)));
    
```

```

209 ERzz = diag(sparse(ERzz(:)));
210 URxx = diag(sparse(URxx(:)));
211 URyy = diag(sparse(URyy(:)));
212 URzz = diag(sparse(URzz(:)));
213
214 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
215 %% ANALYZE INPUT WAVEGUIDE
216 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
217
218 % EXTRACT SLAB WAVEGUIDE FROM GRID
219 nx = 2*(NPML(1) + 1);
220 erzz = diag(sparse(ER2(nx,1:2:Ny2)));
221 urxx = diag(sparse(UR2(nx,2:2:Ny2)));
222 uryy = diag(sparse(UR2(nx+1,1:2:Ny2)));
223
224 % BUILD DERIVATIVE MATRICES
225 NS = [1 Ny];
226 RES = [1 dy];
227 BC = [0 0];
228 [~,DEY,~,DHY] = yeeder2d(NS,k0*RES,BC);
229
230 % BUILD EIGEN-VALUE PROBLEM
231 A = -(DHY/urxx*DEY + erzz);
232 B = inv(uryy);
233
234 % SOLVE FULL EIGEN-VALUE PROBLEM
235 [Ez,D2] = eig(full(A),full(B));
236 D = sqrt(D2);
237 NEFF = -1i*diag(D);
238 [NEFF,ind] = sort(real(NEFF),'descend');
239 Ez = Ez(:,ind);
240
241 % GET SOURCE MODE
242 neff = NEFF(1);
243 Ezsrc = Ez(:,1);
244
245 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
246 %% PERFORM FDFD ANALYSIS
247 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
248
249 % BUILD DERIVATIVE MATRICES
250 NS = [Nx Ny];
251 RES = [dx dy];
252 BC = [0 0];
253 [DEX,DEY,DHX,DHY] = yeeder2d(NS,k0*RES,BC);
254
255 % BUILD WAVE MATRIX
256 A = DHX/URyy*DEX + DHY/URxx*DEY + ERzz;
257
258 % CALCULATE SOURCE FIELD
259 fsrc = zeros(Nx,Ny);
260 for nx = 1 : Nx
261     fsrc(nx,:) = Ezsrc*exp(-1i*k0*neff*nx*dx);
262 end
263
264 % CALCULATE SCATTERED-FIELD MASKING MATRIX

```

```
265 nx      = NPML(1) + 2;
266 Q       = zeros(Nx,Ny);
267 Q(1:nx,:) = 1;
268 Q       = diag(sparse(Q(:)));
269
270 % CALCULATE SOURCE VECTOR
271 b = (Q*A - A*Q)*fsrc(:);
272
273 % SOLVE FOR FIELD
274 f = A\b;
275 f = reshape(f,Nx,Ny);
276
277 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
278 %% ANALYZE TRANSMITTED AND REFLECTED
279 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
280
281 % EXTRACT FIELDS
282 nx  = NPML(1) + 1;
283 fref = f(nx,:);
284 nx  = Nx - NPML(2);
285 ftrn = f(nx,:);
286
287 % CALCULATE MODE AMPLITUDES
288 aref = Ez\fref(:);
289 atrn = Ez\ftrn(:);
290
291 % CALCULATE REFLECTANCE AND TRANSMITTANCE
292 R = 100*abs(aref(1))^2;
293 T = 100*abs(atrn(1))^2;
294
295 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
296 %% DRAW PRETTY SOLUTION
297 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
298
299 % PREPARE GRAPHICS WINDOW
300 clf;
301 hold on;
302
303 % DRAW FIELD
304 pcolor(xa,ya,real(f).');
305 shading interp;
306
307 % SUPERIMPOSE LATTICE
308 er0 = (neff1^2 + neff2^2)/2;
309 contour(xa2,ya2,ER2.',er0,'Color','w','LineWidth',0.5);
310
311 % SET GRAPHICS VIEW
312 hold off;
313 axis equal tight;
314 set(gca,'YDir','reverse');
315 caxis(0.25*[-1 1]);
316 colorbar;
317 colormap('gray');
```

Chapter 9
FD2D() – Generic Two-Dimensional FDFD for Periodic Structures

```

1    function DAT = fdfd2d(DEV, SRC)
2    % FDFD2D    Two-Dimensional FDFD for Periodic Structures
3    %
4    % DAT = fdfd2d(DEV, SRC);
5    %
6    % INPUT ARGUMENTS
7    % =====
8    % DEV      Device Parameters
9    %   .ER2    relative permittivity on 2x grid
10   %   .UR2    relative permeability on 2x grid
11   %   .RES    [dx dy] grid resolution on Yee grid
12   %   .NPML   [NYLO NYHI] size of UPML on Yee grid
13   %
14   % SRC      Source Parameters
15   %   .lam0   free space wavelength
16   %   .theta  angle of incidence
17   %   .MODE   'E' or 'H'
18   %
19   % OUTPUT ARGUMENTS
20   % =====
21   % DAT      Output Data
22   %   .f      simulated field
23   %   .m      diffraction order indices
24   %   .RDE    diffraction efficiencies of reflected waves
25   %   .REF    overall reflectance
26   %   .TDE    diffraction efficiencies of transmitted waves
27   %   .TRN    overall transmittance
28
29   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30   %% HANDLE INPUT ARGUMENTS
31   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
32
33   % GET SIZE OF GRID
34   [Nx2, Ny2] = size(DEV.ER2);
35
36   % CALCULATE GRID
37   Nx = Nx2/2;
38   Ny = Ny2/2;
39   dx = DEV.RES(1);
40   dy = DEV.RES(2);
41   Sx = Nx*dx;
42   Sy = Ny*dy;
43   xa = [1:Nx]*dx;
44   ya = [1:Ny]*dy;
45   [Y, X] = meshgrid(ya, xa);
46
47   % WAVE NUMBER
48   k0 = 2*pi/SRC.lam0;
49
50   % EXTRACT MATERIAL PROPERTIES IN EXTERNAL REGIONS
51   urref = DEV.UR2(1,1);
52   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

```

```

109 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110
111 % CALCULATE WAVE VECTOR COMPONENTS
112 DAT.m = [-floor(Nx/2):+floor((Nx-1)/2)].';
113 kx      = kxinc - DAT.m*2*pi/Sx;
114 kyref = sqrt((k0*nref)^2 - kx.^2);
115 kytrn = sqrt((k0*ntrn)^2 - kx.^2);
116
117 % EXTRACT REFELECTED AND TRANSMITTED WAVES
118 fref = DAT.f(:,DEV.NPML(1)+1)./fsrc(:,1);
119 ftrn = DAT.f(:,Ny-DEV.NPML(2))./fsrc(:,1);
120
121 % CALCULATE AMPLITUDES OF SPATIAL HARMONICS
122 aref = fftshift(fft(fref))/Nx;
123 atrn = fftshift(fft(ftrn))/Nx;
124
125 % CALCULATE DIFFRACTION EFFICENCIES
126 DAT.RDE = abs(aref).^2.*real(kyref/kyinc);
127 if SRC.MODE == 'E'
128     DAT.TDE = abs(atrn).^2.*real(urref/urtrn*kytrn/kyinc);
129 else
130     DAT.TDE = abs(atrn).^2.*real(erref/ertrn*kytrn/kyinc);
131 end
132
133 % CALCULATE OVERALL REFLECTANCE & TRANSMITTANCE
134 DAT.REF = sum(DAT.RDE(:));
135 DAT.TRN = sum(DAT.TDE(:));

```

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

```

1 % Chapter9_GMRF.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 degrees     = pi/180;
12
13 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
14 %% DASHBOARD
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17 % DEFINE PLANE WAVE SOURCE PARAMETERS
18 lam1      = 1.4 * micrometers;
19 lam2      = 1.6 * micrometers;
20 NLAM      = 1000;
21 LAMBDA    = linspace(lam1,lam2,NLAM);
22 SRC.theta = 0*degrees;
23 SRC.MODE  = 'E';
24
25 % DEFINE GMRF PARAMETERS
26 f         = 0.5;

```

```

27  n1 = 1.0;
28  n2 = 1.4;
29  nL = 1.9;
30  nH = 2.1;
31  d1 = 265 * nanometers;
32  d2 = 380 * nanometers;
33  L  = 870 * nanometers;
34
35  % DEFINE FDFD PARAMETERS
36  NRES      = 40;
37  SPACER    = max(LAMBDA)*[0.5 0.5];
38  DEV.NPML  = [20 20];
39  nmax      = max([n1 n2 nL nH]);
40
41  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
42  %% CALCULATE OPTIMIZED GRID
43  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
44
45  % GRID RESOLUTION
46  dx = min(LAMBDA)/nmax/NRES;
47  dy = min(LAMBDA)/nmax/NRES;
48
49  % SNAP GRID TO CRITICAL DIMENSIONS
50  nx = ceil(L/dx);
51  dx = L/nx;
52  ny = ceil(d2/dy);
53  dy = d2/ny;
54
55  % GRID SIZE
56  Sx = L;
57  Nx = ceil(Sx/dx);
58  Sx = Nx*dx;
59
60  Sy = SPACER(1) + d1 + d2 + d1 + SPACER(2);
61  Ny = DEV.NPML(1) + ceil(Sy/dy) + DEV.NPML(2);
62  Sy = Ny*dy;
63
64  % 2X GRID
65  Nx2 = 2*Nx;          dx2 = dx/2;
66  Ny2 = 2*Ny;          dy2 = dy/2;
67
68  % CALCULATE AXIS VECTORS
69  xa = [1:Nx]*dx;
70  ya = [1:Ny]*dy;
71  xa2 = [1:Nx2]*dx2;
72  ya2 = [1:Ny2]*dy2;
73
74  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
75  %% BUILD GUIDED-MODE RESONANCE FILTER
76  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
77
78  % INITIALIZE TO AIR
79  DEV.ER2 = n1^2*ones(Nx2,Ny2);
80  DEV.UR2 = ones(Nx2,Ny2);
81
82  % COMPUTE ARRAY INDICES

```

```
83 nx = round(f*L/dx2);
84 nx1 = 1 + floor((Nx2 - nx)/2);
85 nx2 = nx1 + nx - 1;
86
87 ny1 = 2*DEV.NPML(1) + round(SPACER(1)/dy2) + 1;
88 ny2 = ny1 + round(d1/dy2) - 1;
89 ny3 = ny2 + 1;
90 ny4 = ny3 + round(d2/dy2) - 1;
91 ny5 = ny4 + 1;
92 ny6 = ny5 + round(d1/dy2) - 1;
93
94 % BUILD DEVICE
95 DEV.ER2(:,ny1:ny2) = n2^2;
96 DEV.ER2(:,ny3:ny4) = nL^2;
97 DEV.ER2(nx1:nx2,ny3:ny4) = nH^2;
98 DEV.ER2(:,ny5:ny6) = n2^2;
99
100 % SHOW DIFFRACTION GRATING
101 subplot(151);
102 imagesc(xa2, ya2, DEV.ER2. ');
103 axis equal tight off;
104 colorbar;
105 title('ER2');
106
107 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
108 %% SIMULATE DEVICE
109 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110
111 % FINISH INPUT ARGUMENTS
112 DEV.RES = [dx dy];
113
114 % INITIALIZE RECORDS
115 REF = zeros(1,NLAM);
116 TRN = zeros(1,NLAM);
117 CON = zeros(1,NLAM);
118
119 %
120 % WAVELENGTH SWEEP
121 %
122 for nlam = 1 : NLAM
123
124     % Set Wavelength
125     SRC.lam0 = LAMBDA(nlam);
126
127     % Call fdfd2d()
128     DAT = fdfd2d(DEV, SRC);
129
130     % Record Results
131     REF(nlam) = DAT.REF;
132     TRN(nlam) = DAT.TRN;
133     CON(nlam) = DAT.REF + DAT.TRN;
134
135     % Show Field
136     subplot(152);
137     imagesc(xa, ya, real(DAT.f). ');
138     axis equal tight;
```

```

139     colorbar;
140     title('Field');
141
142     % Show Spectra
143     subplot(1,5,[3:5]);
144     plot(LAMBDA(1:nlam)/micrometers,CON(1:nlam),'k');
145     hold on;
146     plot(LAMBDA(1:nlam)/micrometers,TRN(1:nlam),'--b');
147     plot(LAMBDA(1:nlam)/micrometers,REF(1:nlam),'-r');
148     hold off;
149     xlim([LAMBDA(1) LAMBDA(NLAM)]/micrometers);
150     ylim([-0.05 1.05]);
151     drawnow;
152 end
    
```

Sweep Depth of Metal Polarizer

```

1     % Chapter9_polarizer_d.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
25    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26    %% DASHBOARD
27    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28
29    % DEFINE PLANE WAVE SOURCE PARAMETERS
30    f0      = 2.0 * terahertz;
31    SRC.lam0 = c0/f0;
32    SRC.theta = 0*degrees;
33
34    % DEFINE POLARIZER PARAMETERS
35    a = 4000 * nanometers;
36    t = 200 * nanometers;
37    w = 2000 * nanometers;
38
39    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 = fd2d2d(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 = fd2d2d(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
```

```
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26 %% DASHBOARD
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28
29 % DEFINE PLANE WAVE SOURCE PARAMETERS
30 f1      = 0.6 * terahertz;
31 f2      = 3.0 * terahertz;
32 NFREQ   = 40;
33 FREQ    = linspace(f1,f2,NFREQ);
34 SRC.theta = 0*degrees;
35
36 % DEFINE POLARIZER PARAMETERS
37 a = 4000 * nanometers;
38 t = 200 * nanometers;
39 w = 2000 * nanometers;
40 d = 1500 * nanometers;
41
42 ersi = 11.8;
43 erau = 1e6;
44
45 % DEFINE FDFD PARAMETERS
46 NRES   = 100;
47 NDIM   = 1;
48 lam0   = c0/mean(FREQ);
49 SPACER = lam0*[0.1 0.1];
50 DEV.NPML = [10 10];
51 nmax   = sqrt(ersi);
52
53 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54 %% CALCULATE OPTIMIZED GRID
55 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56
57 % GRID RESOLUTION
58 lam_min = c0/max(FREQ);
59 dx      = lam_min/nmax/NRES;
60 dy      = lam_min/nmax/NRES;
61
62 % RESOLVE MINIMUM DIMENIONS
63 nx = w/dx;
64 if nx<NDIM
65     dx = w/NDIM;
66 end
67
68 ny = t/dy;
69 if ny<NDIM
70     dy = t/NDIM;
71 end
72
73 % SNAP GRID TO CRITICAL DIMENSIONS
74 nx = ceil(a/dx);
75 dx = a/nx;
76 ny = ceil(d/dy);
77 dy = d/ny;
78
79 % GRID SIZE
80 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    %
    
```

```

31 % OUTPUT ARGUMENTS
32 % =====
33 % DAT      Output Data
34 %
35 %   .fx   .fy   .fz      simulated field
36 %   .m    .n            diffraction order indices
37 %
38 %   .RDE   diffraction efficiencies of reflected waves
39 %   .REF   overall reflectance
40 %   .TDE   diffraction efficiencies of transmitted waves
41 %   .TRN   overall transmittance
42 %
43 %   .s11   complex ref. coefficients of diffraction orders
44 %   .s21   complex tran. coefficients of diffraction orders
45
46 % DEFINE SOLVER PARAMETERS
47 tol    = 1e-6;
48 maxit  = 15000;
49
50 % ANONYMOUS FUNCTIONS
51 diagonalize = @(x) diag(sparse(x(:)));
52
53 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54 %% HANDLE INPUT ARGUMENTS
55 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
56
57 % GET SIZE OF GRID
58 if isfield(DEV, 'ER2')
59     [Nx2, Ny2, Nz2] = size(DEV.ER2);
60 else
61     [Nx2, Ny2, Nz2] = size(DEV.ER2xx);
62 end
63
64 % CALCULATE GRID
65 Nx = Nx2/2;
66 Ny = Ny2/2;
67 Nz = Nz2/2;
68 dx = DEV.RES(1);
69 dy = DEV.RES(2);
70 dz = DEV.RES(3);
71 Sx = Nx*dx;
72 Sy = Ny*dy;
73 Sz = Nz*dz;
74
75 % GRID AXES
76 xa2 = [1:Nx2]*dx/2;          xa2 = xa2 - mean(xa2);
77 ya2 = [1:Ny2]*dy/2;          ya2 = ya2 - mean(ya2);
78 za2 = [0.5:Nz2-0.5]*dz/2;
79 [Y2, X2, Z2] = meshgrid(ya2, xa2, za2);
80
81 % WAVE NUMBER
82 k0 = 2*pi/SRC.lam0;
83
84 % SPECIAL MATRICES
85 M = Nx*Ny*Nz;
86 ZZ = sparse(M, M);

```

```

87 I = speye(M,M);
88
89 % PERMITTIVITY TENSOR ELEMENTS
90 if isfield(DEV,'ER2')
91     ERxx = diagonalize(DEV.ER2(2:2:Nx2,1:2:Ny2,1:2:Nz2));
92     ERxy = ZZ;
93     ERxz = ZZ;
94     ERyx = ZZ;
95     ERyy = diagonalize(DEV.ER2(1:2:Nx2,2:2:Ny2,1:2:Nz2));
96     ERyz = ZZ;
97     ERzx = ZZ;
98     ERzy = ZZ;
99     ERzz = diagonalize(DEV.ER2(1:2:Nx2,1:2:Ny2,2:2:Nz2));
100 else
101     ERxx = diagonalize(DEV.ER2xx(2:2:Nx2,1:2:Ny2,1:2:Nz2));
102     ERxy = diagonalize(DEV.ER2xy(1:2:Nx2,2:2:Ny2,1:2:Nz2));
103     ERxz = diagonalize(DEV.ER2xz(1:2:Nx2,1:2:Ny2,2:2:Nz2));
104     ERyx = diagonalize(DEV.ER2yx(2:2:Nx2,1:2:Ny2,1:2:Nz2));
105     ERyy = diagonalize(DEV.ER2yy(1:2:Nx2,2:2:Ny2,1:2:Nz2));
106     ERyz = diagonalize(DEV.ER2yz(1:2:Nx2,1:2:Ny2,2:2:Nz2));
107     ERzx = diagonalize(DEV.ER2zx(2:2:Nx2,1:2:Ny2,1:2:Nz2));
108     ERzy = diagonalize(DEV.ER2zy(1:2:Nx2,2:2:Ny2,1:2:Nz2));
109     ERzz = diagonalize(DEV.ER2zz(1:2:Nx2,1:2:Ny2,2:2:Nz2));
110 end
111
112 % PERMEABILITY TENSOR ELEMENTS
113 if isfield(DEV,'UR2')
114     URxx = diagonalize(DEV.UR2(1:2:Nx2,2:2:Ny2,2:2:Nz2));
115     URxy = ZZ;
116     URxz = ZZ;
117     URYx = ZZ;
118     URYy = diagonalize(DEV.UR2(2:2:Nx2,1:2:Ny2,2:2:Nz2));
119     URYz = ZZ;
120     URzx = ZZ;
121     URzy = ZZ;
122     URzz = diagonalize(DEV.UR2(2:2:Nx2,2:2:Ny2,1:2:Nz2));
123 else
124     URxx = diagonalize(DEV.UR2xx(1:2:Nx2,2:2:Ny2,2:2:Nz2));
125     URxy = diagonalize(DEV.UR2xy(2:2:Nx2,1:2:Ny2,2:2:Nz2));
126     URxz = diagonalize(DEV.UR2xz(2:2:Nx2,2:2:Ny2,1:2:Nz2));
127     URYx = diagonalize(DEV.UR2yx(1:2:Nx2,2:2:Ny2,2:2:Nz2));
128     URYy = diagonalize(DEV.UR2yy(2:2:Nx2,1:2:Ny2,2:2:Nz2));
129     URYz = diagonalize(DEV.UR2yz(2:2:Nx2,2:2:Ny2,1:2:Nz2));
130     URzx = diagonalize(DEV.UR2zx(1:2:Nx2,2:2:Ny2,2:2:Nz2));
131     URzy = diagonalize(DEV.UR2zy(2:2:Nx2,1:2:Ny2,2:2:Nz2));
132     URzz = diagonalize(DEV.UR2zz(2:2:Nx2,2:2:Ny2,1:2:Nz2));
133 end
134
135 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
136 %% PERFORM FDFD ANALYS
137 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
138
139 % EXTRACT MATERIAL PROPERTIES IN EXTERNAL REGIONS
140 ur_ref = full(URxx(1,1));
141 ur_trn = full(URxx(M,M));
142 er_ref = full(ERxx(1,1));

```

```

143 er_trn = full(ERxx(M,M));
144 n_ref  = sqrt(ur_ref*er_ref);
145 n_trn  = sqrt(ur_trn*er_trn);
146
147 % CALCULATE PML PARAMETERS
148 [sx2,sy2,sz2] = calcpml3d([Nx2 Ny2 Nz2],2*[0 0 0 0 DEV.NPML]);
149
150 sx_ey = diagonalize(1./sx2(1:2:Nx2,2:2:Ny2,1:2:Nz2));
151 sx_ez = diagonalize(1./sx2(1:2:Nx2,1:2:Ny2,2:2:Nz2));
152 sy_ex = diagonalize(1./sy2(2:2:Nx2,1:2:Ny2,1:2:Nz2));
153 sy_ez = diagonalize(1./sy2(1:2:Nx2,1:2:Ny2,2:2:Nz2));
154 sz_ex = diagonalize(1./sz2(2:2:Nx2,1:2:Ny2,1:2:Nz2));
155 sz_ey = diagonalize(1./sz2(1:2:Nx2,2:2:Ny2,1:2:Nz2));
156
157 sx_hy = diagonalize(1./sx2(2:2:Nx2,1:2:Ny2,2:2:Nz2));
158 sx_hz = diagonalize(1./sx2(2:2:Nx2,2:2:Ny2,1:2:Nz2));
159 sy_hx = diagonalize(1./sy2(1:2:Nx2,2:2:Ny2,2:2:Nz2));
160 sy_hz = diagonalize(1./sy2(2:2:Nx2,2:2:Ny2,1:2:Nz2));
161 sz_hx = diagonalize(1./sz2(1:2:Nx2,2:2:Ny2,2:2:Nz2));
162 sz_hy = diagonalize(1./sz2(2:2:Nx2,1:2:Ny2,2:2:Nz2));
163
164 % CALCULATE WAVE VECTORS
165 kinc   = k0*n_ref*[ sin(SRC.theta)*cos(SRC.phi) ; ...
166                   sin(SRC.theta)*sin(SRC.phi)  ; ...
167                   cos(SRC.theta)  ];
168 m      = [-floor(Nx/2):+floor((Nx-1)/2)].';
169 n      = [-floor(Ny/2):+floor((Ny-1)/2)].';
170 kx     = kinc(1) - m*2*pi/Sx;
171 ky     = kinc(2) - n*2*pi/Sy;
172 kz_ref = sqrt((k0*n_ref)^2 - kx.^2 - ky.^2);
173 kz_trn = sqrt((k0*n_trn)^2 - kx.^2 - ky.^2);
174
175 % BUILD DERIVATIVE MATRICES
176 NS = [Nx Ny Nz];
177 RES = [dx dy dz];
178 BC = [1 1 0];
179 [DEX,DEY,DEZ,DHX,DHY,DHZ] = yeeder3d(NS,k0*RES,BC,kinc/k0);
180
181 % CALCULATE INTERPOLATION MATRICES (Assumes SRC.theta = 0)
182 RX = (0.5*k0*dx)*abs(DEX);
183 RY = (0.5*k0*dy)*abs(DEY);
184 RZ = (0.5*k0*dz)*abs(DEZ);
185
186 % FORM MATERIALS TENSORS
187 ER = [ ERxx      RX*RY'*ERxy  RX*RZ'*ERxz ; ...
188        RY*RX'*ERyx  ERYy      RY*RZ'*ERyz ; ...
189        RZ*RX'*ERzx  RZ*RY'*ERzy  ERzz ];
190 UR = [ URxx      RX'*RY*URxy  RX'*RZ*URxz ; ...
191        RY'*RX*URyx  URYy      RY'*RZ*URyz ; ...
192        RZ'*RX*URzx  RZ'*RY*URzy  URzz ];
193
194 % BUILD THE WAVE MATRIX A
195 CE = [ ZZ , -sz_hx*DEZ , sy_hx*DEY ; ...
196        sz_hy*DEZ , ZZ , -sx_hy*DEX ; ...
197        -sy_hz*DEY , sx_hz*DEX , ZZ ];
198 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);
    
```

```

255 fx_trn = DAT.fx(:,:,nz_trn);
256 fy_trn = DAT.fy(:,:,nz_trn);
257 fz_trn = DAT.fz(:,:,nz_trn-1:nz_trn);
258
259 % INTERPOLATE FIELDS TO ORIGIN OF YEE CELLS
260 Ex_ref = fx_ref;
261 Ex_ref(1,:) = (fx_ref(Nx,:) + fx_ref(1,))/2;
262 Ex_ref(2:Nx,:) = (fx_ref(1:Nx-1,:) + fx_ref(2:Nx,))/2;
263
264 Ey_ref = fy_ref;
265 Ey_ref(:,1) = (fy_ref(:,Ny) + fy_ref(:,1))/2;
266 Ey_ref(:,2:Ny) = (fy_ref(:,1:Ny-1) + fy_ref(:,2:Ny))/2;
267
268 Ez_ref = (fz_ref(:,,1) + fz_ref(:,,2))/2;
269
270 Ex_trn = fx_trn;
271 Ex_trn(1,:) = (fx_trn(Nx,:) + fx_trn(1,))/2;
272 Ex_trn(2:Nx,:) = (fx_trn(1:Nx-1,:) + fx_trn(2:Nx,))/2;
273
274 Ey_trn = fy_trn;
275 Ey_trn(:,1) = (fy_trn(:,Ny) + fy_trn(:,1))/2;
276 Ey_trn(:,2:Ny) = (fy_trn(:,1:Ny-1) + fy_trn(:,2:Ny))/2;
277
278 Ez_trn = (fz_trn(:,,1) + fz_trn(:,,2))/2;
279
280 % REMOVE PHASE TILT
281 Ex_ref = Ex_ref./fphase(2:2:Nx2,1:2:Ny2,1);
282 Ey_ref = Ey_ref./fphase(1:2:Nx2,2:2:Ny2,1);
283 Ez_ref = Ez_ref./fphase(1:2:Nx2,1:2:Ny2,2);
284
285 Ex_trn = Ex_trn./fphase(2:2:Nx2,1:2:Ny2,1);
286 Ey_trn = Ey_trn./fphase(1:2:Nx2,2:2:Ny2,1);
287 Ez_trn = Ez_trn./fphase(1:2:Nx2,1:2:Ny2,2);
288
289 % CALCULATE AMPLITUDES OF DIFFRACTION ORDERS
290 ax_ref = fftshift(fft2(Ex_ref))/(Nx*Ny);
291 ay_ref = fftshift(fft2(Ey_ref))/(Nx*Ny);
292 az_ref = fftshift(fft2(Ez_ref))/(Nx*Ny);
293 a_ref = abs(ax_ref).^2 + abs(ay_ref).^2 + abs(az_ref).^2;
294 DAT.s11 = sqrt(ax_ref.^2 + ay_ref.^2 + az_ref.^2);
295
296 ax_trn = fftshift(fft2(Ex_trn))/(Nx*Ny);
297 ay_trn = fftshift(fft2(Ey_trn))/(Nx*Ny);
298 az_trn = fftshift(fft2(Ez_trn))/(Nx*Ny);
299 a_trn = abs(ax_trn).^2 + abs(ay_trn).^2 + abs(az_trn).^2;
300 DAT.s21 = sqrt(ax_trn.^2 + ay_trn.^2 + az_trn.^2);
301
302 % CALCULATE DIFFRACTION EFFICIENCIES
303 DAT.RDE = a_ref.*real(kz_ref/kinc(3));
304 DAT.TDE = a_trn.*real(ur_ref/ur_trn*kz_trn/kinc(3));
305
306 % Calculate Overall Response
307 DAT.REF = sum(DAT.RDE(:));
308 DAT.TRN = sum(DAT.TDE(:));
309 DAT.CON = DAT.REF + DAT.TRN;

```

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

```
1 % Chapter10_GMRF.m
2
3 % INITIALIZE MATLAB
4 close all;
5 clc
6 clear all;
7
8 % DEFINE UNITS
9 meters = 1;
10 centimeters = 1e-2 * meters;
11 millimeters = 1e-3 * meters;
12 micrometers = 1e-6 * meters;
13 nanometers = 1e-9 * meters;
14 seconds = 1;
15 hertz = 1/seconds;
16 kilohertz = 1e3 * hertz;
17 megahertz = 1e6 * hertz;
18 gigahertz = 1e9 * hertz;
19 degrees = pi/180;
20
21 % CONSTANTS
22 c0 = 299792458 * meters/seconds;
23 e0 = 8.854187812813e-12 * 1/meters;
24 u0 = 1.256637062121e-6 * 1/meters;
25 N0 = 376.7303136686;
26
27 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28 %% DASHBOARD
29 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30
31 % DEFINE SOURCE PARAMETERS
32 f1 = 4.5 * gigahertz;
33 f2 = 5.5 * gigahertz;
34 NFREQ = 500;
35 FREQ = linspace(f1, f2, NFREQ);
36 SRC.theta = 0 * degrees;
37 SRC.phi = 0 * degrees;
38 SRC.ptc = 1/sqrt(2);
39 SRC.ptm = 1i/sqrt(2);
40
41 % DEFINE GMRF PARAMETERS
42 er1 = 1.0;
43 er2 = 2.0;
44 erL = 3.0;
45 erH = 5.0;
46 a = 3.7 * centimeters;
47 t = 1.5 * centimeters;
48 f = 0.5;
49
50 % DEFINE GRID PARAMETERS
51 NRES = 20;
52 DEV.NPML = [10 10];
53 ermax = max([er1 er2 erL erH]);
54 nmax = sqrt(ermax);
```

```

55 lam_max = c0/min(FREQ);
56 SPACER = lam_max * [1 1];
57
58 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
59 %% CALCULATE OPTIMIZED GRID
60 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61
62 % CALCULATE PRELIMINARY RESOLUTION
63 lam_min = c0/max(FREQ);
64 dx = lam_min/nmax/NRES;
65 dy = lam_min/nmax/NRES;
66 dz = lam_min/nmax/NRES;
67
68 % SNAP GRID TO CRITICAL DIMENSIONS
69 nx = ceil(a/dx);
70 dx = a/nx;
71 ny = ceil(a/dy);
72 dy = a/ny;
73 nz = ceil(t/dz);
74 dz = t/nz;
75
76 % CALCULATE GRID SIZE
77 Nx = ceil(a/dx);
78 Sx = Nx*dx;
79 Ny = ceil(a/dy);
80 Sy = Ny*dy;
81 Sz = SPACER(1) + t + SPACER(2);
82 Nz = DEV.NPML(1) + ceil(Sz/dz) + DEV.NPML(2);
83 Sz = Nz*dz;
84
85 % 2x GRID
86 Nx2 = 2*Nx;          dx2 = dx/2;
87 Ny2 = 2*Ny;          dy2 = dy/2;
88 Nz2 = 2*Nz;          dz2 = dz/2;
89
90 % GRID AXES
91 xa = [1:Nx]*dx;      xa = xa - mean(xa);
92 ya = [1:Ny]*dy;      ya = ya - mean(ya);
93 za = [0.5:Nz-0.5]*dz;
94
95 xa2 = [1:Nx2]*dx2;   xa2 = xa2 - mean(xa2);
96 ya2 = [1:Ny2]*dy2;   ya2 = ya2 - mean(ya2);
97 za2 = [0.5:Nz2-0.5]*dz2;
98
99 % MESHGRIDS
100 [Y,X,Z] = meshgrid(ya, xa, za);
101 [Y2,X2,Z2] = meshgrid(ya2, xa2, za2);
102
103 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
104 %% BUILD MATERIALS ARRAYS
105 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
106
107 % INITIALIZE TO FREE SPACE
108 DEV.ER2 = ones(Nx2,Ny2,Nz2);
109 DEV.UR2 = ones(Nx2,Ny2,Nz2);
110

```

```

111 % BUILD GRATING LAYER
112 r = a*sqrt(f/pi);
113 DEV.ER2 = (X2.^2 + Y2.^2) <= r^2;
114 DEV.ER2 = erL + (erH - erL)*DEV.ER2;
115
116 % ADD SUPERSTRATE AND SUBSTRATE
117 nz1 = 2*DEV.NPML(1) + round(SPACER(1)/dz2) + 1;
118 nz2 = nz1 + round(t/dz2) - 1;
119 DEV.ER2(:, :, 1:nz1-1) = er1;
120 DEV.ER2(:, :, nz2+1:Nz2) = er2;
121
122 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
123 %% PERFORM FREQUENCY SWEEP
124 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
125
126 % DEV
127 DEV.RES = [dx dy dz];
128
129 % INITIALIZE RESPONSE
130 REF = zeros(1,NFREQ);
131 TRN = zeros(1,NFREQ);
132 CON = zeros(1,NFREQ);
133
134 %
135 % MAIN LOOP -- ITERATE OVER FREQUENCY
136 %
137 for nfreq = 1 : NFREQ
138
139     % Get Nxt Frequency
140     f0      = FREQ(nfreq);
141     SRC.lam0 = c0/f0;
142
143     % Call FDFD3D()
144     DAT = fdfd3d(DEV, SRC);
145
146     % Calculate Overall Response
147     REF(nfreq) = DAT.REF;
148     TRN(nfreq) = DAT.TRN;
149     CON(nfreq) = DAT.CON;
150
151     % Show Field
152     subplot(131);
153     slice(Y/centimeters,X/centimeters,Z/centimeters,...
154         real(DAT.fy),0,0,0);
155     shading interp;
156     axis equal tight;
157     view(250,20);
158     colormap('gray');
159     caxis(1.5*[-1 1]);
160
161     % Show Response
162     subplot(1,3,[2 3]);
163     plot(FREQ(1:nfreq)/gigahertz,CON(1:nfreq),'--k');
164     hold on;
165     plot(FREQ(1:nfreq)/gigahertz,REF(1:nfreq),'-r');
166     plot(FREQ(1:nfreq)/gigahertz,TRN(1:nfreq),'-b');

```

```
167     hold off;
168     xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
169     ylim([-0.05 1.05]);
170     title('Frequency Response');
171     xlabel('Frequency (GHz)');
172     drawnow;
173
174 end
```

Main Function to Simulate a Doubly-Periodic Frequency Selective Surface

```
1     % Chapter10_FSS.m
2
3     % INITIALIZE MATLAB
4     close all;
5     clc
6     clear all;
7
8     % DEFINE UNITS
9     meters      = 1;
10    centimeters = 1e-2 * meters;
11    millimeters = 1e-3 * meters;
12    micrometers = 1e-6 * meters;
13    nanometers  = 1e-9 * meters;
14    seconds     = 1;
15    hertz       = 1/seconds;
16    kilohertz   = 1e3 * hertz;
17    megahertz   = 1e6 * hertz;
18    gigahertz   = 1e9 * hertz;
19    degrees     = pi/180;
20
21    % CONSTANTS
22    c0 = 299792458 * meters/seconds;
23    e0 = 8.854187812813e-12 * 1/meters;
24    u0 = 1.256637062121e-6 * 1/meters;
25    N0 = 376.7303136686;
26
27    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28    %% DASHBOARD
29    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30
31    % DEFINE SOURCE PARAMETERS
32    f1 = 8 * gigahertz;
33    f2 = 12 * gigahertz;
34    NFREQ = 50;
35    FREQ = linspace(f1,f2,NFREQ);
36    SRC.theta = 30 * degrees;
37    SRC.phi = 60 * degrees;
38    SRC.pte = 1;
39    SRC.ptm = 0;
40
41    % DEFINE FSS PARAMETERS
42    a = 9.95 * millimeters; % period of FSS
43    s1 = 1.00 * millimeters;
44    s2 = 0.90 * millimeters;
45    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);

```

```
102 ya2 = [1:Ny2]*dy2;          ya2 = ya2 - mean(ya2);
103 za2 = [0.5:Nz2-0.5]*dz2;
104
105 % MESHGRIDS
106 [Y,X,Z] = meshgrid(ya, xa, za);
107 [Y2,X2,Z2] = meshgrid(ya2, xa2, za2);
108
109 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110 %% BUILD MATERIALS ARRAYS
111 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
112
113 % INITIALIZE TO FREE SPACE
114 DEV.ER2 = ones (Nx2,Ny2,Nz2);
115 DEV.UR2 = ones (Nx2,Ny2,Nz2);
116
117 % ADD SUBSTRATE
118 nz1 = 2*DEV.NPML(1) + 2*round(SPACER(1)/dz2/2) + 1;
119 nz2 = nz1 + round(h/dz2) - 1;
120 DEV.ER2(:, :, nz1:nz2) = erd;
121
122 % BUILD FSS UNIT CELL
123 L = 2*s2 + 2*d + s1;
124 nx = round(L/dx);
125 x1 = (a - L)/2;
126 x2 = x1 + s2;
127 x3 = (a - w)/2;
128 x4 = (a - s1)/2;
129 x5 = (a + s1)/2;
130 x6 = (a + w)/2;
131 x8 = (a + L)/2;
132 x7 = x8 - s2;
133
134 nx1 = round(x1/dx2);
135 nx2 = round(x2/dx2);
136 nx3 = round(x3/dx2);
137 nx4 = round(x4/dx2);
138 nx5 = round(x5/dx2);
139 nx6 = round(x6/dx2);
140 nx7 = round(x7/dx2);
141 nx8 = round(x8/dx2);
142
143 ER2 = zeros (Nx2,Ny2);
144 ER2(nx1:nx8-1, nx4:nx5-1) = 1;
145 ER2(nx4:nx5-1, nx1:nx8-1) = 1;
146 ER2(nx1:nx2-1, nx3:nx6-1) = 1;
147 ER2(nx7:nx8-1, nx3:nx6-1) = 1;
148 ER2(nx3:nx6-1, nx1:nx2-1) = 1;
149 ER2(nx3:nx6-1, nx7:nx8-1) = 1;
150
151 ER2 = 1 + (erm - 1)*ER2;
152
153 % ADD UNIT CELL TO GRID
154 DEV.ER2(:, :, nz1) = ER2;
155
156 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
157 %% PERFORM FREQUENCY SWEEP
```

```

158 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
159
160 % DEV
161 DEV.RES = [dx dy dz];
162
163 % INITIALIZE RESPONSE
164 REF = zeros(1,NFREQ);
165 TRN = zeros(1,NFREQ);
166 CON = zeros(1,NFREQ);
167
168 %
169 % MAIN LOOP -- ITERATE OVER FREQUENCY
170 %
171 for nfreq = 1 : NFREQ
172
173     % Get Nxt Frequency
174     f0      = FREQ(nfreq);
175     SRC.lam0 = c0/f0;
176
177     % Call FDFD3D()
178     DAT = fdfd3d(DEV, SRC);
179
180     % Calculate Overall Response
181     REF(nfreq) = DAT.REF;
182     TRN(nfreq) = DAT.TRN;
183     CON(nfreq) = DAT.CON;
184
185     % Show Response
186     subplot(121);
187     plot(FREQ(1:nfreq)/gigahertz, CON(1:nfreq), '--k');
188     hold on;
189     plot(FREQ(1:nfreq)/gigahertz, REF(1:nfreq), '-r');
190     plot(FREQ(1:nfreq)/gigahertz, TRN(1:nfreq), '-b');
191     hold off;
192     xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
193     ylim([-0.05 1.05]);
194     title('Frequency Response');
195     xlabel('Frequency (GHz)');
196
197     subplot(122);
198     TdB = 10*log10(TRN);
199     plot(FREQ(1:nfreq)/gigahertz, TdB(1:nfreq), '-k');
200     xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
201     ylim([-40 4]);
202     title('Frequency Response');
203     xlabel('Frequency (GHz)');
204
205     drawnow;
206
207 end

```

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

```

1 % Chapter10_metamaterial.m
2
3 % INITIALIZE MATLAB

```

```
4   close all;
5   clc
6   clear all;
7
8   % DEFINE UNITS
9   meters      = 1;
10  centimeters = 1e-2 * meters;
11  millimeters = 1e-3 * meters;
12  micrometers = 1e-6 * meters;
13  nanometers  = 1e-9 * meters;
14  seconds     = 1;
15  hertz       = 1/seconds;
16  kilohertz   = 1e3 * hertz;
17  megahertz   = 1e6 * hertz;
18  gigahertz   = 1e9 * hertz;
19  degrees     = pi/180;
20
21  % CONSTANTS
22  c0 = 299792458 * meters/seconds;
23  e0 = 8.854187812813e-12 * 1/meters;
24  u0 = 1.256637062121e-6 * 1/meters;
25  N0 = 376.7303136686;
26
27  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
28  %% DASHBOARD
29  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
30
31  % DEFINE SOURCE PARAMETERS
32  f1      = 9 * gigahertz;
33  f2      = 14 * gigahertz;
34  NFREQ   = 200;
35  FREQ    = linspace(f1,f2,NFREQ);
36  SRC.theta = 0 * degrees;
37  SRC.phi   = 0 * degrees;
38  SRC.pte   = 0;
39  SRC.ptm   = 1;
40
41  % DEFINE METAMATERIAL PARAMETERS
42  a = 2.500 * millimeters; % period of metamaterial
43  h = 0.250 * millimeters; % substrate thickness
44  w1 = 0.140 * millimeters; % width of wire
45  w2 = 0.200 * millimeters; % linewidth of ring
46  g = 0.300 * millimeters; % gap in ring
47  L2 = 2.200 * millimeters; % length of outer ring
48  L1 = 1.500 * millimeters; % length of inner ring
49
50  er = 4.4;
51  tand = 0.02;
52  erd = er*(1 - li*tand);
53
54  f0 = 10 * gigahertz;
55  sigma = 5.8e7;
56  erm = 1 + sigma/(li*2*pi*f0*e0);
57
58  % DEFINE GRID PARAMETERS
59  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

```

```
116 % INITIALIZE TO FREE SPACE
117 DEV.ER2 = ones (Nx2,Ny2,Nz2);
118 DEV.UR2 = ones (Nx2,Ny2,Nz2);
119
120 % ADD SUBSTRATE
121 ny = 2*round(h/dy2/2);
122 ny1 = 1 + 2*floor((Ny2 - ny)/4);
123 ny2 = ny1 + ny - 1;
124 nz = round(a/dz2);
125 nz1 = 1 + 2*floor((Nz2 - nz)/2/2);
126 nz2 = nz1 + nz - 1;
127 DEV.ER2(:,ny1:ny2,nz1:nz2) = erd;
128
129 % ADD WIRE
130 nza = nz1 + 2*round((a-w1)/2/dz2/2);
131 nzb = nza + round(w1/dz2) - 1;
132 DEV.ER2(:,ny1,nza:nzb) = erm;
133
134 % ADD OUTER RING
135 ny = ny2 + 1;
136 nz = 2*round(L2/dz2/2);
137 nz1 = 1 + 2*floor((Nz2 - nz)/4);
138 nz2 = nz1 + nz - 1;
139 nx = round(L2/dz2);
140 nx1 = 1 + 2*floor((Nx2 - nx)/4);
141 nx2 = nx1 + nx - 1;
142 DEV.ER2(nx1:nx2,ny,nz1:nz2) = erm;
143 nxa = nx1 + round(w2/dz2) - 1;
144 nxb = nx2 - round(w2/dz2) + 1;
145 nz1 = nz1 + round(w2/dz2) - 1;
146 nz2 = nz2 - round(w2/dz2) + 1;
147 DEV.ER2(nxa:nxb,ny,nz1:nz2) = 1;
148 nz = 2*round(g/dz2/2);
149 nz1 = 2*floor((Nz2 - nz)/4);
150 nz2 = nz1 + nz - 1;
151 DEV.ER2(nx1:nxb,ny,nz1:nz2) = 1;
152
153 % ADD INNER RING
154 ny = ny2 + 1;
155 nz = 2*round(L1/dz2/2);
156 nz1 = 1 + 2*floor((Nz2 - nz)/4);
157 nz2 = nz1 + nz - 1;
158 nx = round(L1/dz2);
159 nx1 = 1 + 2*floor((Nx2 - nx)/4);
160 nx2 = nx1 + nx - 1;
161 DEV.ER2(nx1:nx2,ny,nz1:nz2) = erm;
162 nxa = nx1 + round(w2/dz2) - 1;
163 nxb = nx2 - round(w2/dz2) + 1;
164 nz1 = nz1 + round(w2/dz2) - 1;
165 nz2 = nz2 - round(w2/dz2) + 1;
166 DEV.ER2(nxa:nxb,ny,nz1:nz2) = 1;
167 nz = 2*round(g/dz2/2);
168 nz1 = 2*floor((Nz2 - nz)/4);
169 nz2 = nz1 + nz - 1;
170 DEV.ER2(nxa:nx2,ny,nz1:nz2) = 1;
171
```

```

172 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
173 %% PERFORM FREQUENCY SWEEP
174 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
175
176 % DEV
177 DEV.RES = [dx dy dz];
178
179 % INITIALIZE RESPONSE
180 S11 = zeros(1,NFREQ);
181 S21 = zeros(1,NFREQ);
182 REF = zeros(1,NFREQ);
183 TRN = zeros(1,NFREQ);
184 CON = zeros(1,NFREQ);
185
186 %
187 % MAIN LOOP -- ITERATE OVER FREQUENCY
188 %
189 for nfreq = 1 : NFREQ
190
191     % Get Nxt Frequency
192     f0      = FREQ(nfreq);
193     SRC.lam0 = c0/f0;
194     k0      = 2*pi/SRC.lam0;
195
196     % Call FDFD3D()
197     DAT = fd3d(DEV, SRC);
198
199     % Record Response
200     dsrc = SPACER(1) - 3*dz;
201     dref = SPACER(1) - 1*dz;
202     dtrn = SPACER(2) - 1*dz;
203     nxc  = 1 + floor(Nx/2);
204     nyc  = 1 + floor(Ny/2);
205     S11(nfreq) = DAT.s11(nxc,nyc) ./ exp(-1i*k0*(dsrc + dref));
206     S21(nfreq) = DAT.s21(nxc,nyc) ./ exp(-1i*k0*(dsrc + dtrn));
207     REF(nfreq) = DAT.REF;
208     TRN(nfreq) = DAT.TRN;
209     CON(nfreq) = DAT.CON;
210
211     % Show Response
212     subplot(2,3,[2 3]);
213     plot(FREQ(1:nfreq)/gigahertz,CON(1:nfreq),'--k');
214     hold on;
215     plot(FREQ(1:nfreq)/gigahertz,REF(1:nfreq),'-r');
216     plot(FREQ(1:nfreq)/gigahertz,TRN(1:nfreq),'-b');
217     hold off;
218     xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
219     ylim([-0.05 1.05]);
220     title('Power Response');
221     xlabel('Frequency (GHz)');
222
223     subplot(2,3,[5 6]);
224     plot(FREQ(1:nfreq)/gigahertz,angle(S11(1:nfreq)),'-r');
225     hold on;
226     plot(FREQ(1:nfreq)/gigahertz,angle(S21(1:nfreq)),'-b');
227     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

```

```
284 % SHOW RESPONSE
285 subplot(321);
286 plot(FREQ/gigahertz,CON,'--k','LineWidth',2);
287 hold on;
288 plot(FREQ/gigahertz,REF,':k','LineWidth',2);
289 plot(FREQ/gigahertz,TRN,'-k','LineWidth',2);
290 hold off;
291 xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
292 ylim([-0.05 1.05]);
293 xlabel('Frequency (GHz)');
294 title('Reflectance & Transmittance');
295
296 subplot(322);
297 plot(FREQ/gigahertz,phi11,':k','LineWidth',2);
298 hold on;
299 plot(FREQ/gigahertz,phi21,'-k','LineWidth',2);
300 hold off;
301 xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
302 ylim([-2.25 2.25]);
303 xlabel('Frequency (GHz)');
304 title('Phase Response');
305
306 % SHOW REFRACTIVE INDEX AND IMPEDANCE
307 subplot(323);
308 plot(FREQ/gigahertz,real(neff),'-k','LineWidth',2);
309 hold on;
310 plot(FREQ/gigahertz,imag(neff),'--k','LineWidth',2);
311 hold off;
312 xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
313 ylim([-5 +7]);
314 xlabel('Frequency (GHz)');
315 title('Refractive Index  $n$ ','Interpreter','LaTeX');
316
317 subplot(324);
318 plot(FREQ/gigahertz,real(eta)/N0,'-k','LineWidth',2);
319 hold on;
320 plot(FREQ/gigahertz,imag(eta)/N0,'--k','LineWidth',2);
321 hold off;
322 xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
323 ylim([-3 +3]);
324 xlabel('Frequency (GHz)');
325 title('Impedance  $\eta/\eta_0$ ','Interpreter','LaTeX');
326
327 % SHOW RELATIVE PERMITTIVITY AND PERMEABILITY
328 subplot(325);
329 plot(FREQ/gigahertz,real(er),'-k','LineWidth',2);
330 hold on;
331 plot(FREQ/gigahertz,imag(er),'--k','LineWidth',2);
332 hold off;
333 xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
334 ylim([-5 2]);
335 xlabel('Frequency (GHz)');
336 title('Relative Permittivity  $\varepsilon_r$ ,...
337     'Interpreter','LaTeX');
338
339 subplot(326);
```

```

340 plot(FREQ/gigahertz,real(ur),'-k','LineWidth',2);
341 hold on;
342 plot(FREQ/gigahertz,imag(ur),'--k','LineWidth',2);
343 hold off;
344 xlim([FREQ(1) FREQ(NFREQ)]/gigahertz);
345 ylim([-7 +14]);
346 xlabel('Frequency (GHz)');
347 title('Relative Permeability  $\mu_r$ ','Interpreter','LaTeX');
    
```

Program to Simulate a Two-Dimensional Invisibility Cloak

```

1   % Chapter10_cloak.m
2
3   % INITIALIZE MATLAB
4   close all;
5   clc;
6   clear all;
7
8   % UNITS
9   degrees = pi/180;
10
11  % ANONYMOUS FUNCTIONS
12  diagonalize = @(x) diag(sparse(x(:)));
13
14  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15  %% DASHBOARD
16  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
17
18  % SOURCE
19  lam0 = 1;
20  theta = 30*degrees;
21  P = [0;0;1];
22
23  % CLOAK
24  R1 = 1.0*lam0;
25  R2 = 3.5*lam0;
26
27  % GRID
28  Sx = 10*lam0;
29  Sy = Sx;
30  NRES = 60;
31  NPML = [20 20 20 20];
32
33  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
34  %% CALCULATE GRID
35  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
36
37  % RESOLUTION
38  dx = lam0/NRES;
39  dy = lam0/NRES;
40
41  % GRID SIZE
42  Nx = ceil(Sx/dx);
43  dx = Sx/Nx;
44  Ny = ceil(Sy/dy);
45  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);

```

```

102             ERzy(nx,ny) = ER(3,2);
103             ERzz(nx,ny) = ER(3,3);
104         end
105     end
106 end
107
108 % CALCULATE IMPERMEABILITY
109 D = ERxx.*ERyy.*ERzz - ERxx.*ERyz.*ERzy ...
110     - ERxy.*ERyx.*ERzz + ERxy.*ERyz.*ERzx ...
111     + ERxz.*ERyx.*ERzy - ERxz.*ERyy.*ERzx;
112 YRxx = (ERyy.*ERzz - ERyz.*ERzy)./D;
113 YRxy = (ERxz.*ERzy - ERxy.*ERzz)./D;
114 YRxz = (ERxy.*ERyz - ERxz.*ERyy)./D;
115 YRyx = (ERyz.*ERzx - ERyx.*ERzz)./D;
116 YRyy = (ERxx.*ERzz - ERxz.*ERzx)./D;
117 YRyz = (ERxz.*ERyx - ERxx.*ERyz)./D;
118 YRzx = (ERyx.*ERzy - ERyy.*ERzx)./D;
119 YRzy = (ERxy.*ERzx - ERxx.*ERzy)./D;
120 YRzz = (ERxx.*ERyy - ERxy.*ERyx)./D;
121
122 % FORM DIAGONAL MATERIALS MATRICES
123 ERxx = diagonalize(ERxx(2:2:Nx2,1:2:Ny2));
124 ERxy = diagonalize(ERxy(1:2:Nx2,2:2:Ny2));
125 ERxz = diagonalize(ERxz(1:2:Nx2,1:2:Ny2));
126 ERyx = diagonalize(ERYx(2:2:Nx2,1:2:Ny2));
127 ERYy = diagonalize(ERYy(1:2:Nx2,2:2:Ny2));
128 ERYz = diagonalize(ERYz(1:2:Nx2,1:2:Ny2));
129 ERzx = diagonalize(ERzx(2:2:Nx2,1:2:Ny2));
130 ERzy = diagonalize(ERzy(1:2:Nx2,2:2:Ny2));
131 ERzz = diagonalize(ERzz(1:2:Nx2,1:2:Ny2));
132
133 YRxx = diagonalize(YRxx(1:2:Nx2,2:2:Ny2));
134 YRxy = diagonalize(YRxy(2:2:Nx2,1:2:Ny2));
135 YRxz = diagonalize(YRxz(2:2:Nx2,2:2:Ny2));
136 YRyx = diagonalize(YRyx(1:2:Nx2,2:2:Ny2));
137 YRyy = diagonalize(YRyy(2:2:Nx2,1:2:Ny2));
138 YRyz = diagonalize(YRyz(2:2:Nx2,2:2:Ny2));
139 YRzx = diagonalize(YRzx(1:2:Nx2,2:2:Ny2));
140 YRzy = diagonalize(YRzy(2:2:Nx2,1:2:Ny2));
141 YRzz = diagonalize(YRzz(2:2:Nx2,2:2:Ny2));
142
143 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
144 %% PERFORM FDFD SIMULATION
145 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
146
147 % CALCULATE SOURCE
148 k0 = 2*pi/lam0;
149 ersrc = ERxx(1,1);
150 nsrc = sqrt(ersrc);
151 kinc = k0*nsrc*[sin(theta);cos(theta);0];
152
153 E = exp(-1i*(kinc(1)*X2 + kinc(2)*Y2));
154 Ex = P(1)*E(2:2:Nx2,1:2:Ny2);
155 Ey = P(2)*E(1:2:Nx2,2:2:Ny2);
156 Ez = P(3)*E(1:2:Nx2,1:2:Ny2);
157 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');
```