A 2D electromagnetic scattering solver for Matlab

Using Method of Moments and Green's tensor technique

by Beat Hangartner, bhangart@ee.ethz.ch

July 1, 2002

1 Introduction and method

The goal of this project was to implement a numerical field solver using one of the methods presented in the lecture *Einführung in numerische Feldberechnungsverfahren*. I chose to solve 2D scattering problems with the Greens tensor technique in Matlab. The Green's tensor is a solution for a point source of the wave equation. In the case for 2D scattering the tensor degrades to a scalar G^B.

Starting with the homogenous wave equation

$$\nabla \times \nabla \times E(r) - k_0^2 \varepsilon(r) E(r) = 0 \tag{1}$$

and using the dielectric contrast

$$\Delta\varepsilon(r) = \varepsilon(r) - \varepsilon_B \tag{2}$$

one can write equation 1 as an inhomogenous one:

$$\nabla \times \nabla \times E(r) - k_0^2 \varepsilon_B E(r) = k_0^2 \Delta\varepsilon(r) E(r) \tag{3}$$

Since in the background, where $\Delta\varepsilon(r) = 0$, the incident plane wave $E^0(r)$ must be a solution of the homogenous equation

$$\nabla \times \nabla \times E^0(r) - k_0^2 \varepsilon_B E^0(r) = 0, \tag{4}$$

therefore $E^0(r)$ results to

$$E^0(r) = E^0 e^{i\mathbf{kr}} \tag{5}$$

in the timeharmonic case where $\mathbf{k}$ denotes the propagation vector satisfying

$$k_0^2 = \mathbf{k} \cdot \mathbf{k}$$

Equation 5 directly gives us the excitation terms in every point in space. In order to compute the scattered field in and outside the scattering object we solve the wave equation with a point source term where the solution as mentioned above is the Green's function G^B.

$$\nabla \times \nabla \times G^B(r, r') - k_0^2 \varepsilon_B G^B(r, r') = \delta(r - r^{'}) \tag{6}$$

Solving equation 6 and combining with equation 1 we find the solution

$$E(r) = E^0(r) + \int_A G^B(r, r') k_0^2 \Delta\varepsilon(r') dr' \tag{7}$$

It shows as well that the field in the background can be computed from the solution inside the scatterer since $\Delta\varepsilon(r) = 0$ is true outside. Splitting up background and scatterer area greatly reduces computation needs since the systems to be solved go with the order of N^2. The fields in the background are a simple linear superposition of the scatterer's field.
The next step is to discretize equation 7 using $E_i = E(r_i)$ and $G^B_{i,j} = G^B(r_i, r'_j)$:

$$E_i = E_i^0 + \sum_{j=1, j\neq i}^{N} G^B_{i,j} k_0^2 \Delta\varepsilon_j E_j V_j + M_i k_0^2 \Delta\varepsilon_i E_i \tag{8}$$

N holds the total number of cell elements of the area to be computed and V_j denotes the cell area (volume) which does not have to be necessarily equal for each cell. Since Green's function has a singularity for $r = r'$ ('in the center of the source'), one has to treat this case separately. This is done using the M_i term. Lastly only Green's function $G^B(r)$ and M_i needs to be defined:

$$G^B(r, r') = \frac{i}{4} H_0(k_\rho \rho) exp(i k_z z) \tag{9}$$

$$\begin{aligned} M_i &= \frac{i\pi}{2} \beta\gamma \\ \beta &= 1 - \frac{k_z^2}{k_B^2} \\ \gamma &= \frac{R_i^{eff}}{k_\rho} H_1(k_\rho R_i^{eff}) + \frac{2i}{\pi k_\rho^2} \end{aligned} \tag{10}$$

$H_\alpha(\rho)$ denotes Hankelfunctions of the first kind. Since only TEM propagation is considered, $k_z = 0$ holds true. Further

$$k_\rho = \sqrt{k_x^2 + k_y^2} \tag{11}$$

$$\rho = \sqrt{(x - x')^2 + (y - y')^2} \tag{12}$$

$$R_i^{eff} = \left(\frac{V_i}{\pi}\right)^{\frac{1}{2}}, \quad V_i = dx_i dy_i \tag{13}$$

2 Implementation in Matlab

Deriving a linear equation system of equation 8 is easy ($N = 4$, omitting V_i):

$$\begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{pmatrix} = \begin{pmatrix} E_1^0 \\ E_2^0 \\ E_3^0 \\ E_4^0 \end{pmatrix} + k_0^2 \begin{pmatrix} M_1\Delta\varepsilon_1 & G_{12}^B\Delta\varepsilon_2 & G_{13}^B\Delta\varepsilon_3 & G_{14}^B\Delta\varepsilon_4 \\ G_{21}^B\Delta\varepsilon_1 & M_2\Delta\varepsilon_2 & G_{23}^B\Delta\varepsilon_3 & G_{24}^B\Delta\varepsilon_4 \\ G_{31}^B\Delta\varepsilon_1 & G_{32}^B\Delta\varepsilon_2 & M_3\Delta\varepsilon_3 & G_{24}^B\Delta\varepsilon_4 \\ G_{41}^B\Delta\varepsilon_1 & G_{42}^B\Delta\varepsilon_2 & G_{43}^B\Delta\varepsilon_4 & M_4\Delta\varepsilon_4 \end{pmatrix} \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{pmatrix}$$

and it translates to (omitting k_0^2)

$$\begin{pmatrix} 1 - M_1\Delta\varepsilon_1 & -G_{12}^B\Delta\varepsilon_2 & -G_{13}^B\Delta\varepsilon_3 & -G_{14}^B\Delta\varepsilon_4 \\ -G_{21}^B\Delta\varepsilon_1 & 1 - M_2\Delta\varepsilon_2 & -G_{23}^B\Delta\varepsilon_3 & -G_{24}^B\Delta\varepsilon_4 \\ -G_{31}^B\Delta\varepsilon_1 & -G_{32}^B\Delta\varepsilon_2 & 1 - M_3\Delta\varepsilon_3 & -G_{34}^B\Delta\varepsilon_4 \\ -G_{41}^B\Delta\varepsilon_1 & -G_{42}^B\Delta\varepsilon_2 & -G_{43}^B\Delta\varepsilon_3 & 1 - M_4\Delta\varepsilon_4 \end{pmatrix} \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{pmatrix} = \begin{pmatrix} E_1^0 \\ E_2^0 \\ E_3^0 \\ E_4^0 \end{pmatrix}$$

which is of the form $\mathbf{A} * \mathbf{b} = \mathbf{c}$ and can easily be solved in Matlab. Taking into account that the majority of the cells belong to the background and therefore $\Delta\varepsilon$ is zero we gain matrices of the following structure. In this example cell 1 and cell 3 are background, the others belong to the scatterer (again omitting V_i and k_0^2).

$$\begin{pmatrix} 1 & -G_{12}^B\Delta\varepsilon_2 & 0 & -G_{14}^B\Delta\varepsilon_4 \\ 0 & 1 - M_2\Delta\varepsilon_2 & 0 & -G_{24}^B\Delta\varepsilon_4 \\ 0 & -G_{32}^B\Delta\varepsilon_2 & 1 & -G_{34}^B\Delta\varepsilon_4 \\ 0 & -G_{42}^B\Delta\varepsilon_2 & 0 & 1 - M_4\Delta\varepsilon_4 \end{pmatrix} \begin{pmatrix} E_1 \\ E_2 \\ E_3 \\ E_4 \end{pmatrix} = \begin{pmatrix} E_1^0 \\ E_2^0 \\ E_3^0 \\ E_4^0 \end{pmatrix} \tag{14}$$

This leads to the following simplified equation system along with the recipe to compute the background cells:

$$\begin{pmatrix} 1 - M_2\Delta\varepsilon_2 & -G_{24}^B\Delta\varepsilon_4 \\ -G_{42}^B\Delta\varepsilon_2 & 1 - M_4\Delta\varepsilon_4 \end{pmatrix} \begin{pmatrix} E_2 \\ E_4 \end{pmatrix} = \begin{pmatrix} E_2^0 \\ E_4^0 \end{pmatrix} \tag{15}$$

$$\begin{pmatrix} E_1 \\ E_3 \end{pmatrix} = \begin{pmatrix} E_1^0 \\ E_3^0 \end{pmatrix} + \begin{pmatrix} G_{12}^B\Delta\varepsilon_2 & G_{14}^B\Delta\varepsilon_4 \\ G_{32}^B\Delta\varepsilon_2 & G_{34}^B\Delta\varepsilon_4 \end{pmatrix} \begin{pmatrix} E_2 \\ E_4 \end{pmatrix} \tag{16}$$

2.1 Data structures

The geometry information consists in this case only of the dielectric contrast $\Delta\varepsilon_i$ and (x, y)-coordinates of the corresponding cell. All cells have the same shape and size and therefore no other data needs to be stored. A scattering configuration like

1	1	1	1
1	3	3	1
1	3	3	1
1	1	1	1

is mapped to the linear array `g_eometry`:

$$\begin{array}{c} \text{index} \\ \Delta\varepsilon_i \\ x_i \\ y_i \end{array} \left[\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & \dots \\ 0 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & \dots \\ 0.5 & 1.5 & 2.5 & 3.5 & 0.5 & 1.5 & 2.5 & 3.5 & \dots \\ 0.5 & 0.5 & 0.5 & 0.5 & 1.5 & 1.5 & 1.5 & 1.5 & \dots \end{array}\right]$$

In a next step all distances ρ_{ij} between each pair of cell are calculated and stored into the matrix `rho`.

```
k=1;
for i=mat_indices
    rho(k,:)=sqrt((g_eometry(3,i) - g_eometry(3,mat_indices)).^2 + ...
              (g_eometry(4,i) - g_eometry(4,mat_indices)).^2);
    k=k+1;
end
```

With the matrix `rho` the Green's function elements for the scatterer cells can easily be calculated like this. (Where `'mat'` stands for material, $\Delta\varepsilon_i \neq 0$; the array `mat_indices` holds the indices of scatterer cells.)

```
G_mat = I/4*V_i*k0^2*besselh(0,1,k0*rho);
```

applying the correct values for the diagonal elements M_i is done with the following statement:

```
G_mat=full(spdiags((I*pi/2*beta*gamma)*k0^2*ones(mat_size,1),0,G_mat));
```

and finally the simplified equation system as introduced before is solved with the excitation term

```
source=(E0*exp(I*kx*g_eometry(3,:)+I*ky*g_eometry(4,:))).';
```

```
% apply epsilon
k=1;
for i=mat_indices
    G_mat(:,k) = G_mat(:,k)* g_eometry(2,i);
    k=k+1;
end

%solve it!
A_mat = eye(mat_size) - G_mat;
solution_mat=(A_mat\source(mat_indices)).';
```

The remaining background fields are calculated like this

```
k=1;
for i=bg_indices
    rho(k,:)=sqrt((g_eometry(3,i) - g_eometry(3,mat_indices)).^2 + ...
             (g_eometry(4,i) - g_eometry(4,mat_indices)).^2);
    k=k+1;
end

G_bg =I/4*V_i*k0^2*besselh(0,1,k0*rho);

solution_bg = source(bg_indices).'  +  ...
    (G_bg * ( solution_mat.* g_eometry(2,mat_indices)).' ) .';
```

At the end the intensity of the solution is plotted using `contourf`.

2.2 How to enter the scatterers shape?

Searching for a handy way of computing different resonators and scatterers I decided to limit myself to integer values for the dielectric contrast $\Delta\varepsilon_i$. This allows me to create geometry input files which 'graphically' represent the shape of the scatterer. The values coded in the files represent the dielectric contrast an the number of lines an columns exactly have to match the dimensions. A typical geometry file looks like this:

```
00000000000000000000
00000000000000000000
00000000000000000000
00000000000222200000
00000000000222200000
00000000000222200000
00000000000222200000
00000000000000000000
00000000000000000000
00000000000000000000
```

The file then is read by Matlab and the values for $\Delta\varepsilon_i$ are stored in the geometry array along with the (x, y)-coordinates which are automatically generated since regular cell sizes and spacing is used.

3 Examples

3.1 A simple rectangular block

This first example shows a quadratic block of 10×10 elements with a dielectricum of $\varepsilon = 3$ and a background of $\varepsilon = 1$. The wave is incident from the left side. The patterns at $\lambda = 1$ and $\lambda = 2$ are maybe a bit confusing since they do not gradually fit into the series of pictures. I assume that since in both cases the width of the block is a integer multiple of the wavelength no resonances occur and the scatterer appears to be invisible. No standing wave settles in front of the block. Only the corners produce irregular fields. Looking at the other wavelengths the pictures show very nice scattering patters, most of them having standing waves in front of the scatterer and two 'jet-streams' of higher field intensity going away towards the upper and lower right corner.

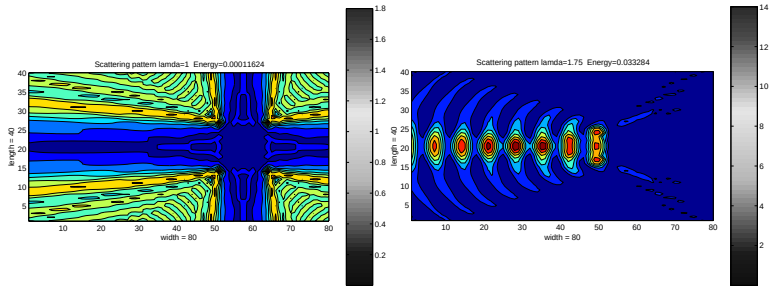

Figure 1: Scattering pattern at $\lambda = 1$ and $\lambda = 1.75$

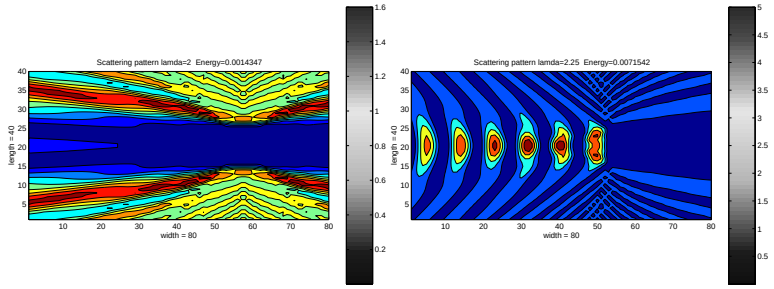

Figure 2: Scattering pattern at $\lambda = 2$ and $\lambda = 2.25$

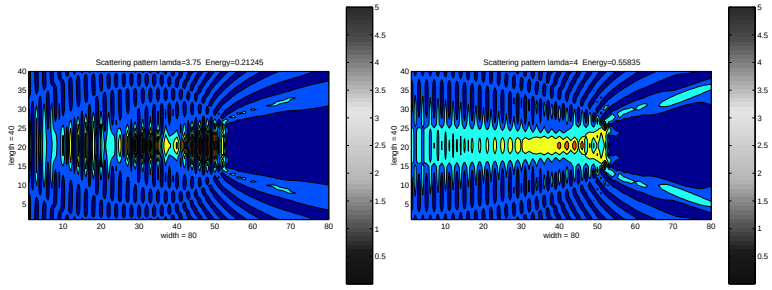

Figure 3: Scattering pattern at $\lambda = 3.75$ and $\lambda = 4$

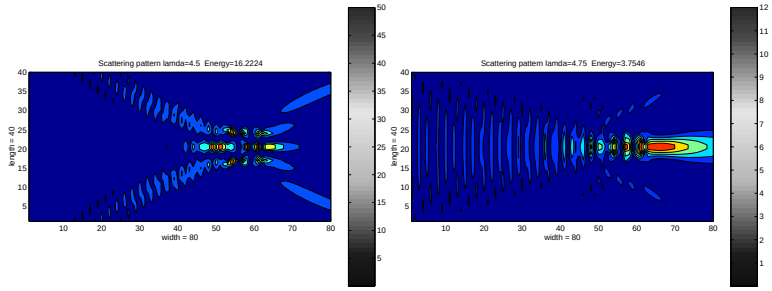

Figure 4: Scattering pattern at $\lambda = 4.5$ and $\lambda = 4.75$

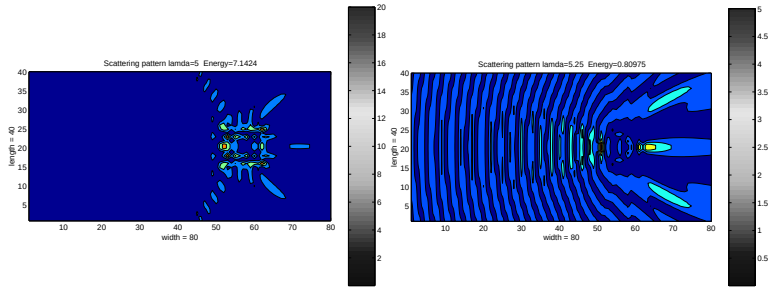

Figure 5: Scattering pattern at $\lambda = 5$ and $\lambda = 5.25$

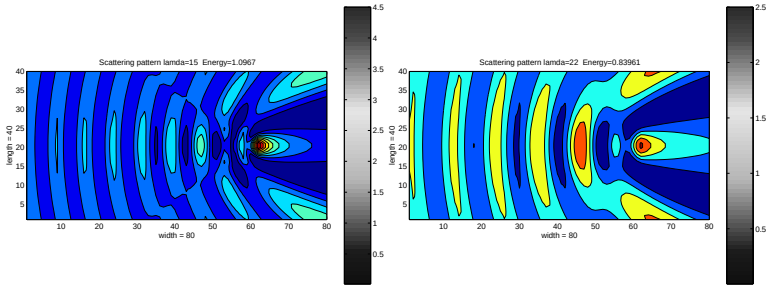

Figure 6: Scattering pattern at $\lambda = 15$ and $\lambda = 22$

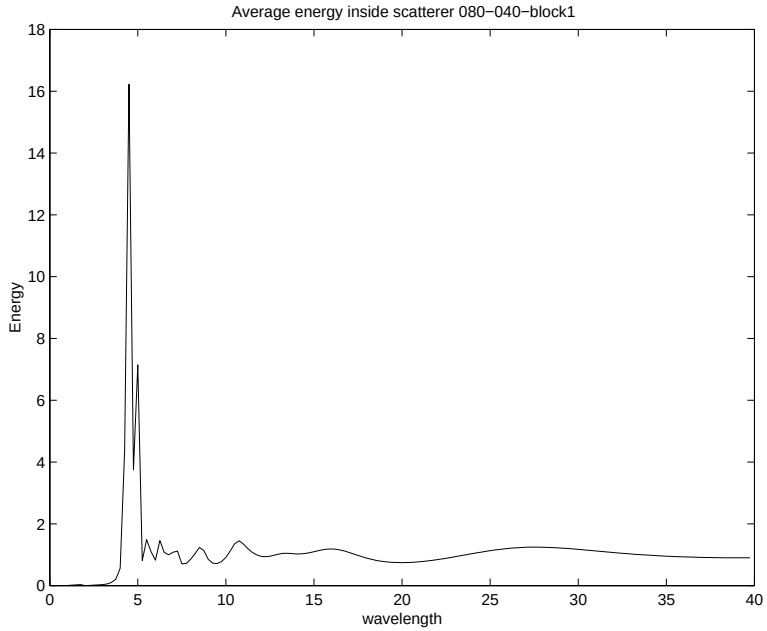

Figure 7: The average Energy inside the scatteres object plotet against the wavelength shows where resonance occurs. Actually I expected stronger resonances above the first peak at about $\lambda = 4.6$

3.2 A regular grid

This second example is a grid of sixteen 4×4 quadratic blocks which are equally spaced. The wave is again incident from the left. This structure has more resonances as the Energy plots show.

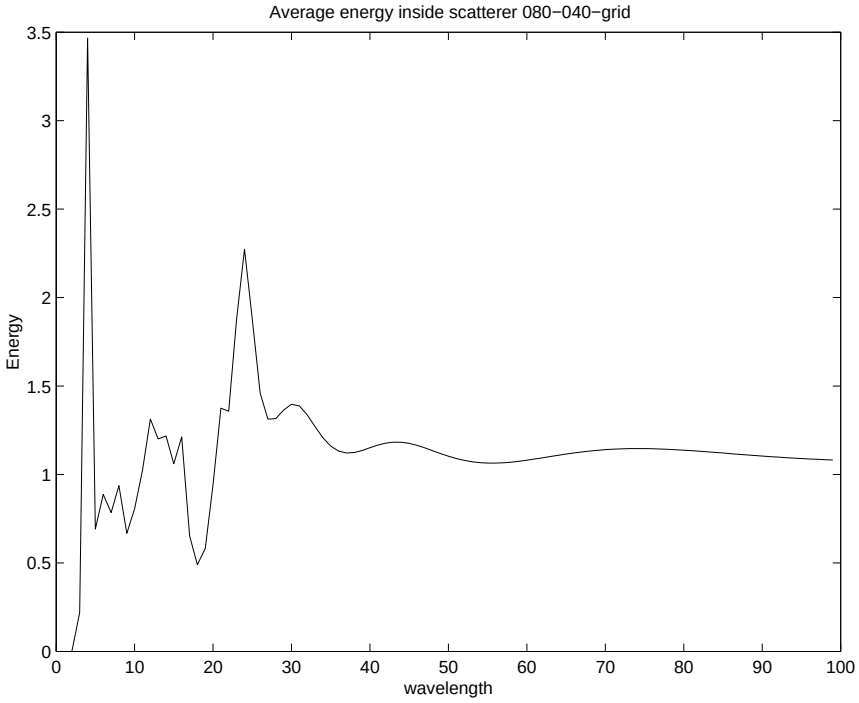

Figure 8: The grid which is a more complex strucuture shows more different resonances over a wider range of wavelengths.

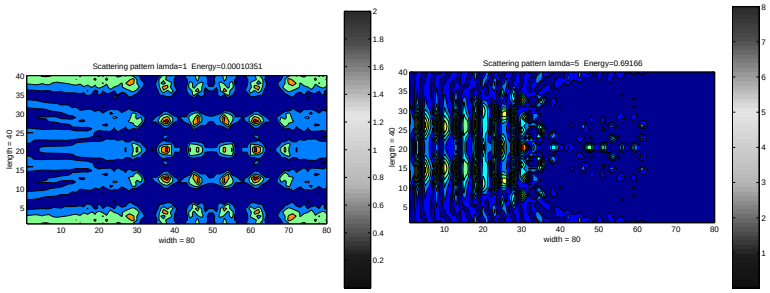

Figure 9: Grid scattering pattern at $\lambda = 1$ and $\lambda = 5$

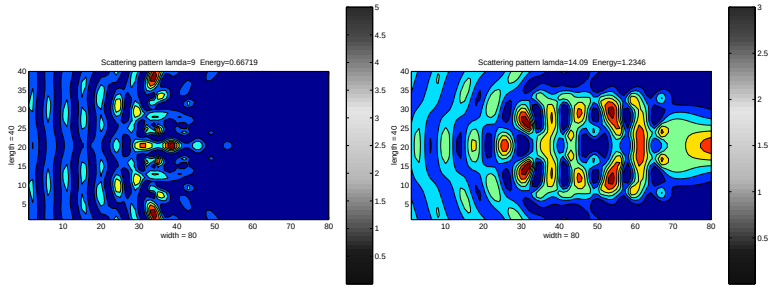

Figure 10: Grid scattering pattern at $\lambda = 9$ and $\lambda = 14.09$

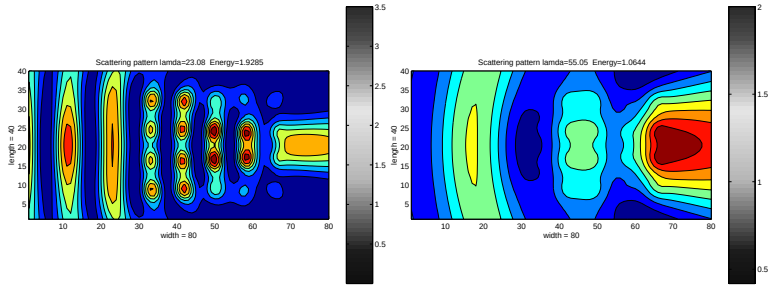

Figure 11: Grid scattering pattern at $\lambda = 23.08$ and $\lambda = 55.05$

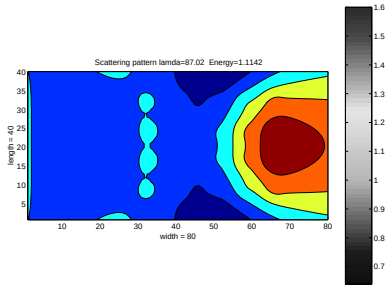

Figure 12: Grid scattering pattern at $\lambda = 87.02$

4 Experiences

Writing this simple solver in Matlab gave me quite a good insight on how numerical field solving prorams can do their work. I also noticed the need to crosscheck the computed results with other existing solvers or methods. This was especially obious as I found a 'simple' mistake in my solver algorithm. Instead of solving the equation system the correct way
`solution_mat=(A_mat\source(mat_indices))';`,
I typed `solution_mat=(source(mat_indices)\(A_mat));`.
Unfortuneatly I recieved a solution shown below, which could be possible too. I also gained a deeper understanding of how electromagnetic waves behave if they hit differnt objects. Last but no least I had great fun using the *condor* tool on the *tardis* student computing cluster which allowed me to set up many processes on different machines which run concurrently.

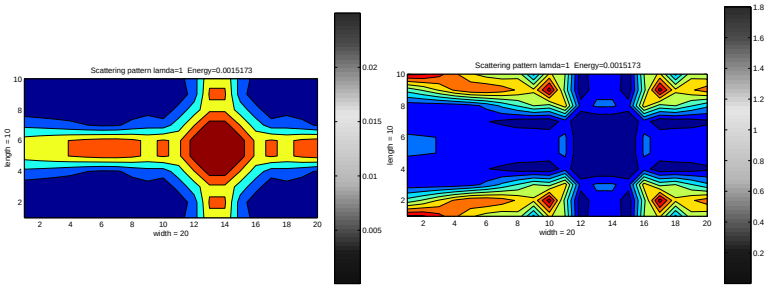

Figure 13: To the left the wrongly solved problem and to the right the (hopefully) more accurate solution. Both at $\lambda = 1$, the scatterer is a quadratic block of 4×4 cells of $\varepsilon = 3$

5 Source code

The whole package consists of the following files

- main.m
- mov.m
- parameters.m
- g_eom.m
- excitation.m
- solver_mat.m
- 020-010-*.m and similar in the corresponding directories (geometry files)

To compute one of geometries defined in the appropriate files simply define `lambda` and call `main` in the Matlab command line. The function `E=mov(start,stop,step)` computes solutions for the specified wavelengths in the range lasting from `start` to `stop`. Return value is a vector of the average energy in the scatterer for each wavelength. Using e.g. `plot(mov(1,100,1))` shows at which wavelenghts resonances within the scatterer occurs.
Changeing the input geometry file is done in the parameters file.

5.1 main.m

```
%   Main routine
%
%   2D Scattering project
%   by Beat Hangartner

tic
parameters
g_eom
excitation
solver_mat2
toc

figure(1)
contourf(S.*conj(S))

title(['Scattering pattern lamda=',num2str(lamda),'  Energy=',num2str(Energy)])
ylabel(['length = ',num2str(len)])
```

```
xlabel(['width = ',num2str(width)])
colorbar
axis image
```

5.2 mov.m

```
%   Movie
%
%   2D Scattering
%   by Beat Hangartner
%

function E2=mov(start,stop,step)
E2=[];
write_jpg=1;

parameters
g_eom

range=start:step:(stop-step);

tic
for lamda=range

    parameters
    excitation
    solver_mat2

    if write_jpg==1
        figure
        contourf(S.*conj(S))
        title(['Scattering pattern lamda=',num2str(lamda),...
                '  Energy=',num2str(Energy)])
        ylabel(['length = ',num2str(len)])
        xlabel(['width = ',num2str(width)])
        axis image
        colorbar
        saveas(gcf,['./',filename,'/' ,int2str(lamda*100), '.eps']);
        clf reset
    end

    E2=[E2,Energy];
end
toc
```

```
clf reset
figure
plot(range,E2);
title(['Average energy inside scatterer ', filename])
xlabel(['wavelength'])
ylabel(['Energy'])

saveas(gcf,['./',filename,'/' ,'Energy-',num2str(start),'-',num2str(stop),...
                '-',num2str(step), '.eps']);
```

5.3 parameters.m

```
%   Parameters definition
%
%   2D Scattering
%   by Beat Hangartner
%
%   History
%   26.5.02         created
%   29.5.02         modified    Filname format
%
%   Dimensions follow the SI-System

global I dx dy gamma beta kx ky k0 width len geom_size R_i_eff V_i

% imaginary unit
I = sqrt(-1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   Excitation parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Wavelength
if exist('lamda')==0
    global lamda
    lamda=1;
end;

%Electrical field strength
E0=1;

%Wave propagation direction
kx_=1;
```

```
ky_=0;
kz=0; %(2D)

kx=kx_/sqrt(kx_^2+ky_^2)*2*pi/lamda;
ky=ky_/sqrt(kx_^2+ky_^2)*2*pi/lamda;

k0=2*pi/lamda;


%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Geometry parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Background dielectric constant
epsilon_b=1;

%Grid spaceing
dx = 1;
dy = dx;

%material file
%filename='020-015-block1';
filename='020-010-block1';
%filename='002-002-block1';
%filename='010-010-block1';
%filename='080-040-block1';
%filename='080-040-grid';

%Area
width=str2num(filename(1:3));
len=str2num(filename(5:7));

geom_size = len*width/dx/dy;

% mesh Volume (Area)
 V_i = dx*dy;

% effective radius
R_i_eff = sqrt(V_i/pi);

%%%%%%%%%%%%%%%%%%%%%%%%%%%
% solver parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%
beta=1-kz^2/k0^2;
```

```
gamma=R_i_eff/k0*besselh(1,1,R_i_eff*k0)+I*2/(pi*k0^2);
```

5.4 g_eom.m

```
%   Geometry array definition
%
%   2D Scattering
%   by Beat Hangartner
%
%   History
%   26.5.02         created
%
%
```

```
%%%%%%%%%%%%%%%%%%
% Geometry layout
%%%%%%%%%%%%%%%%%%%%

% Origin
% (0,0)
%   |----------------------------------->  x-axis (width)
%   |  cell 1       |   cell 2        |
%   |  x=0.5*dx     |   x=1.5*dx      |
%   |  y=0.5*dy     |   y=0.5*dy      |
%   |               |                 |
%   |               |                 |
%   |               |                 |
%   |-----------------------------------
%   |  cell 3       |  cell 4         |
%   |  x=0.5*dx     |  x=1.5*dx       |
%   |  y=1.5*dy     |  y=1.5*dy       |
%   |               |                 |
%   |               |                 |
%   |               |                 |
%   |-----------------------------------
%   |
%  \/
% y-axis (length)

global g_eometry
global bg_indices
```

```
global mat_indices


%instantiate geometry cells in a linear array
%structure is as follows:
%   1. row    index
%   2. row    (complex) epsilon
%   3. row    x-position
%   4. row    y-position

g_eometry =zeros(4,geom_size);
g_eometry(1,:)=1:geom_size;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% open material file
% and retrieve scatterer's layout
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fid = fopen(['./',filename,'/',filename,'.m']);
file = fread(fid)';
fclose(fid);

%get rid of carriage return and line feed (PC only)
file=file(find(file~=10));
file=file(find(file~=13));

% impose vlaues coded in the file over the geometry
%('0' is ASCII character 48)
% only integers from 0 to 9 supported
% at the moment for epsilon  -;)
g_eometry(2,:) = file-48;


% generate regular grid position information
for i=1:len/dy
    g_eometry(3,(i-1)*width/dx+1:i*width/dx)=0.5*dx:dx:width;
end

for i=1:width/dx
    g_eometry(4,[i-1+[1:width/dx:geom_size]])=0.5*dy:dy:len;
end

%extract background/material inidces; to be computed separatly
bg_indices = find(g_eometry(2,:)==0);
```

```
mat_indices = find(g_eometry(2,:)~=0);
```

5.5 excitation.m

```
%   Excitation array
%
%   2D Scattering
%   by Beat Hangartner
%
%   History
%   26.5.02         created
%


global source


% compute plane wave source
source=(E0 * exp(I*kx*g_eometry(3,:) + I*ky*g_eometry(4,:))).';
```

5.6 solver_mat2.m

```
%   Solver
%
%   2D Scattering
%   by Beat Hangartner
%
%   History
%   26.5.02         created
%   29.5.02         modified    Algorithm corrections (epsilons)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%     compute elements with contrast other than zero  %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global Energy

%number of material elements
mat_size = size(mat_indices,2);
bg_size = size(bg_indices,2);

rho=zeros(mat_size);
k=1;
for i=mat_indices
```

```
    rho(k,:)=sqrt((g_eometry(3,i) - g_eometry(3,mat_indices)).^2 +...
             (g_eometry(4,i) - g_eometry(4,mat_indices)).^2);
    k=k+1;
end

% put them into the hankelfunction
G_mat = I/4*V_i*k0^2*besselh(0,1,k0*rho);

%free memory
clear rho;

% add the correct diagonal elements
G_mat=full(spdiags((I*pi/2*beta*gamma)*k0^2*ones(mat_size,1),0,G_mat));

% apply epsilon
k=1;
for i=mat_indices
    G_mat(:,k) = G_mat(:,k)* g_eometry(2,i);
    k=k+1;
end

A_mat = eye(mat_size) - G_mat;

%free memory
clear G_mat;

solution_mat = (A_mat\source(mat_indices)).';

%free memory
clear A_mat;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%     compute background solutions    %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%compute rho's between background and material
rho=zeros(bg_size,mat_size);
k=1;
for i=bg_indices
    rho(k,:)=sqrt((g_eometry(3,i) - g_eometry(3,mat_indices)).^2 + ...
             (g_eometry(4,i) - g_eometry(4,mat_indices)).^2);
    k=k+1;
end
```

```
% put them into the hankelfunction
G_bg = I/4* V_i*k0^2*besselh(0,1,k0*rho);

%free memory
clear rho;

solution_bg = source(bg_indices).'  +  ...
              (G_bg * ( solution_mat .* g_eometry(2, mat_indices)).' ) .';

%free memory
clear G_bg;

solution = zeros(1,geom_size);
solution(mat_indices)=solution_mat;
solution(bg_indices)=solution_bg;

% concatenate solution to matrix S
S=solution(1:width/dx);
for i=2:len/dy
    S = [solution((i-1)*width/dx+1:i*width/dx);S];
end

Energy = sum(solution(mat_indices).*conj(solution(mat_indices)))/mat_size;
```

5.7 020-010-block1.m

```
00000000000000000000
00000000000000000000
00000000000000000000
00000000000222200000
00000000000222200000
00000000000222200000
00000000000222200000
00000000000000000000
00000000000000000000
00000000000000000000
```

5.8 Condor files

To send a number of processes to the cluster for computing simply type 'condor_submit feldber.condor'. In the file feldber.condor the number of processes and the wavelength range is specified and then passed to the function mov(start,stop,step) within Matlab. The EPS files of the computed images

are afterwards stored to the directory belonging to the geometry file.

5.8.1 feldber.condor

```
######################################################################
##
##  Scattering movies with MATLAB/Feldber and Condor
##
##  compiled by Beat Hangartner
##
######################################################################

#
# For this example, variables not used by Condor are marked my_*
#

universe = vanilla
getenv = True # MATLAB needs local environment

initialdir = /home/bhangart/feldber/
my_prefix = feldber

#
# Seek max floating point performance
#
Rank = Kflops

#
# Feldber Constants
#
my_procs =   1
my_start_wl = 2
my_stop_wl = 100
my_step_wl      = 1

#
# MATLAB does all the math there,
# Condor just does string substitution
#
my_start_wl_p = ($(my_start_wl) + ...
                  ($(my_stop_wl)-$(my_start_wl))*$(Process)/$(my_procs))
my_stop_wl_p = ($(my_start_wl) + ...
               ($(my_stop_wl)-$(my_start_wl))*($(Process)+1)/$(my_procs))
```

```
#
# For MATLAB and other SEPP packages, the executable must be a script wrapper.
#
executable = feldber.sh
arguments = mov($(my_start_wl_p),$(my_stop_wl_p),$(my_step_wl));

#
# To redirect stdout and/or stderr to /dev/null, comment these out.
#
log = $(my_prefix).log
#output = $(my_prefix).$(Process).out
#error = $(my_prefix).$(Process).err

#
# Lastly, tell condor how many jobs to queue up.
#
queue $(my_procs)
```

5.8.2 feldber.sh

```
#!/bin/sh
#
# Filename: mandelbrot.sh

#
# We use a shell wrapper for two reasons:
#
#   1) By using "$*" we ensure that the matlab command string is
# passed as a single argument even if it contains spaces.
#
#   2) Condor changes argv[0], which causes problems for SEPP.  Hence,
# whenever we run a program from /usr/sepp/bin/* we must use a
# shell script wrapper.
#
exec /usr/sepp/bin/matlab  -nodisplay -nosplash -nojvm -r "$*"
```