SSY200 Assignment 2

Set up environment

```
format compact
clear
EM_constants
close all
```

Parameters, as given in §2.4

```
%% Assignment parameters
omega = 3e9;   %
k0 = omega/c0; % Wavenumber for given frequency (in vacuum).
sigm = 0.02;   % conductance sigma for glass [S/m]
er = 2.5;           % Relative permittivity for glass.
eps_glass = e0*er;  % Abs. permittivity for glass.
E0i = 1;            % Magnitude of incident wave. Arbitrary?
```

```
iter = 1
```

```
iter =
     1
```

```
for N=4% = (1:10).*10
```

Generate grid

1. Glass boundaries (-a,+a) fall between grid points.
2. Glass boundaries fall exactly on grid points.

```
%N = 4; % Assert N >= 4 and even.
assert(N >= 4, "Assert failed: N must be >4")
assert(mod(N,2) == 0, "Assert failed: N must be even!")
a = 2e-2; % Half-thickness of glass. 2cm. Given in §2.4

for grid_type=["G1" "G2"]
switch grid_type
    case "G1"
        %% G1: Glass boundaries between grid-points.
        n = -N-1:N;
        h = 2*a/N;
        xn = (n + 1/2)*h; % Grid-points.

        N_gp = length(xn); % Number of grid points.
    case "G2"
        %% G2: Glass boundaries on grid points!
        n = -N:N;
        h = 2*a/N;
        xn = n*h;
```

```
        N_gp = length(xn); % Number of grid points.
end
```

Create system of linear equations for the 1D Helmholtz equation

Create square tri-diagonal A matrix.

Each **row** corresponds to a position along the x-axis.

$$A = \begin{bmatrix} -\frac{1}{h} - \frac{jk_0}{2} & \frac{1}{h} - \frac{jk_0}{2} & 0 & \cdots \\ -\frac{1}{h^2} & \frac{2}{h^2} + \mu_0[\cdots]_1 & -\frac{1}{h^2} & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} + \mu_0[\cdots]_2 & -\frac{1}{h^2} \\ \vdots & 0 & -\frac{1}{h^2} & \frac{2}{h^2} + \mu_0[\cdots]_3 \end{bmatrix}$$, ie. row 1 is the left boundary.

```
A = zeros(N_gp, N_gp); % Initialize

% Helmholtz (non-boundary cases)
% Initialize main diagonal.
for i = 1:N_gp-1
    if (abs( xn(i) ) <= a)  % Inside glass!!
        d = 2/h^2 + mu0*(j*omega*sigm - omega^2*eps_glass);
    else                    % Else in air.
        d = 2/h^2 + mu0*(-omega^2*e0);
    end
    A(i,i) = d;
end

s = -1/h^2;% Upper and lower diagonal entries.
for i = 1:N_gp-1
    A(i,i+1) = s; % Upper diagonal.
    A(i+1,i) = s; % Lower diagonal.
end
```

Insert boundary conditions in A matrix.

```
A(1,1:2) = [-1/h - j*k0/2, 1/h - j*k0/2];                % Left boundary condition.
A(N_gp, end-1:end) = [-1/h + j*k0/2, 1/h + j*k0/2];      % Right boundary condition.
```

Construct b column vector (right hand side at each point in space).

$$b = \begin{bmatrix} -2jk_0 E_z^i \frac{(x_1 + x_0)}{2} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

```
b = [-2*j*k0*(E0i*exp(-j*k0*(xn(2)+xn(1))/2)); zeros(N_gp-1,1)];
```

Solve

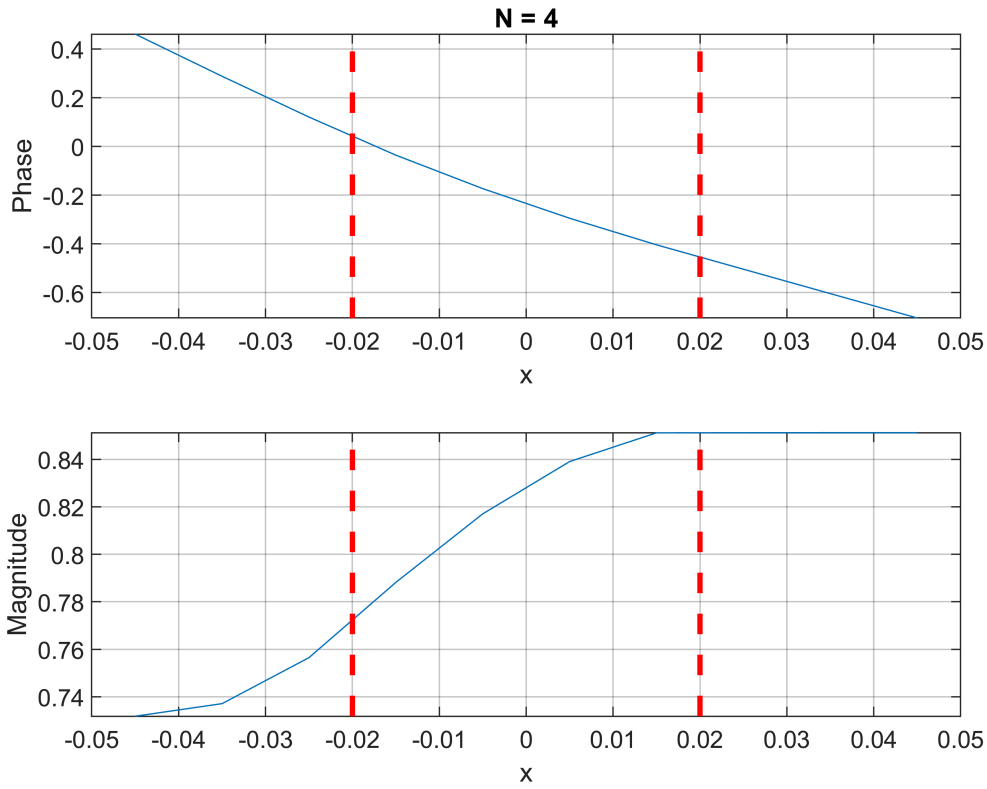
```
z = A\b; % Solve system.
```

Plot total electric field $E_z(x)$

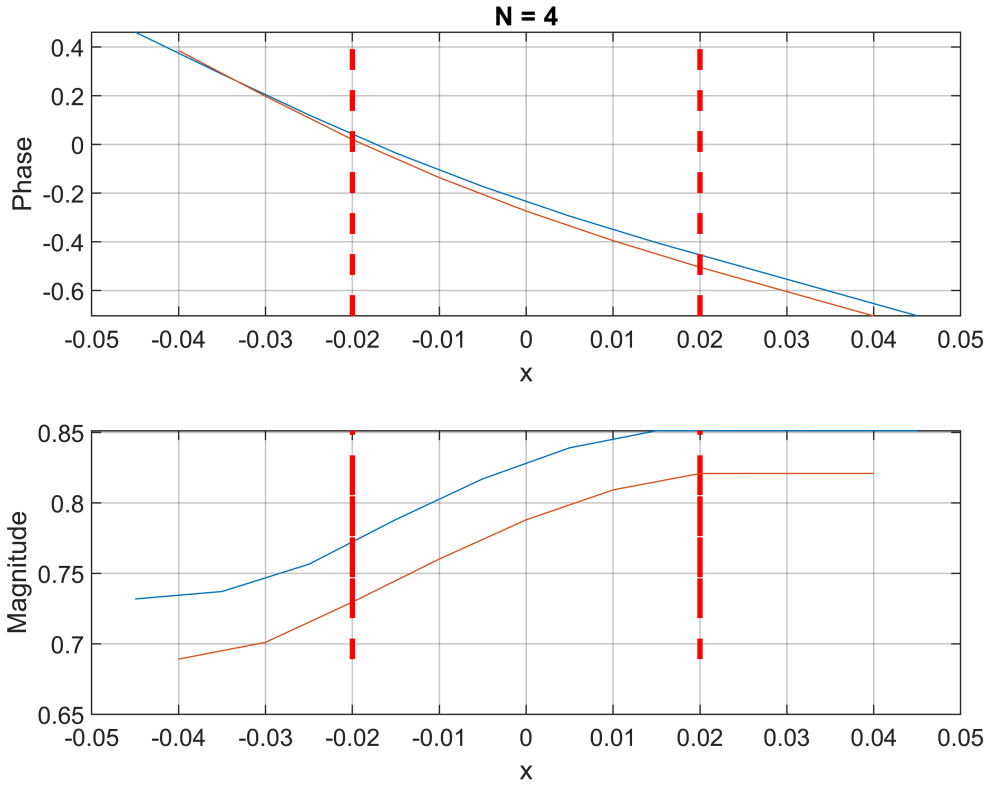
```
subplot(211)
plot(xn, angle(z))
grid on
ylabel('Phase')
xlabel('x')
hold on
plot([-a -a],[min(angle(z)) max(angle(z))],'r--','LineWidth',2.0) % Draw (-a,+a) on plot.
plot([+a +a],[min(angle(z)) max(angle(z))],'r--','LineWidth',2.0) % Draw (-a,+a) on plot.

title(['N = ' num2str(N)])

subplot(212)
plot(xn, abs(z))
grid on
hold on
%legend('G1')
ylabel('Magnitude')
xlabel('x')
plot([-a -a],[min(abs(z)) max(abs(z))],'r--','LineWidth',2.0) % Draw (-a,+a) on plot.
plot([+a +a],[min(abs(z)) max(abs(z))],'r--','LineWidth',2.0) % Draw (-a,+a) on plot.
```

Analytical values for k_n, R, T

```
k1 = sqrt(er*k0^2 - j*omega*mu0*sigm)
Delta =  (k0 + k1)^2*exp(j*4*a*k1) - (k0 - k1)^2; % Intermediate value (not used elsewhere)

R = (k0^2 - k1^2)/Delta * exp(j*2*a*k0) * (exp(j*4*a*k1)-1);
T = (k0*k1)/Delta * 4 *exp(j*2*a*(k0+k1));



a_idx = (ceil(N_gp*1/4)); % Index corresponding to coordinate of -a in grid. Rounded down.
Rnum = (z(a_idx)-exp(-j*k0*xn(a_idx)))/exp(-j*k0*xn(a_idx));
a_idx = (floor(N_gp*3/4)); % Index corresponding to coordinate of +a in grid. Rounded down.
Tnum = z(a_idx)/exp(-j*k0*xn(a_idx));

RnumVec(iter) = Rnum;
TnumVec(iter) = Tnum;
iter = iter + 1;
```




```
k1 =
 15.996876574092919 - 2.356654542433087i
```




```
k1 =
 15.996876574092919 - 2.356654542433087i
```

```
end % Grid selection loop
end % end grid refinement loop
```

Fit curve

Reflection coeff. R

```
figure
h = 2*a./(10:10:100);
hcont = linspace(0, 2*a/10, 1000);
RVec = repmat(abs(R),length(hcont),1);% For illustrating analytical solution
plot(hcont.^2, RVec, 'R--')
hold on
plot(h.^2, abs(RnumVec), 'b-+')

p = polyfit(h,abs(RnumVec),2);
fitted_poly = polyval(p,hcont);
plot(hcont.^2, fitted_poly)
legend('Analytical (fixed)', 'Numerical', 'Extrapolated')

hold off

title('|R| vs h^2')
grid on
xlabel('h^2');ylabel('|R|')
```

Transmission coeff. T

```
figure
h = 2*a./(10:10:100);
hcont = linspace(0, 2*a/10, 1000);
TVec = repmat(abs(T),length(hcont),1);% For illustrating analytical solution
plot(hcont.^2, TVec, 'R--')
hold on
plot(h.^2, abs(TnumVec), 'b-+')

p = polyfit(h,abs(TnumVec),2);
fitted_poly = polyval(p,hcont);
plot(hcont.^2, fitted_poly)
legend('Analytical (fixed)', 'Numerical', 'Extrapolated')

hold off

title('|T| vs h^2')
grid on
```

Transmission coeff phase

```
figure
h = 2*a./(10:10:100);
hcont = linspace(0, 2*a/10, 1000);
TVec = repmat(angle(T),length(hcont),1);% For illustrating analytical solution
plot(hcont.^2, TVec, 'R--')
hold on
plot(h.^2, angle(TnumVec), 'b-+')

p = polyfit(h,angle(TnumVec),2);
fitted_poly = polyval(p,hcont);
plot(hcont.^2, fitted_poly)
legend('Analytical (fixed)', 'Numerical', 'Extrapolated')

hold off

title('angle(T) vs h^2')
grid on

xlabel('h^2');ylabel('angle(T)')
```