MATLAB EXERCISE 2.4 **Dielectric–dielectric boundary conditions, oblique plane.**
Write a program in MATLAB that uses dielectric–dielectric boundary conditions in Eqs.(2.4) and (2.5) (from the book) to find the electric field intensity vector in medium 2 near the boundary ($\mathbf{E}_2$) for a given electric field intensity vector in medium 1 near the boundary ($\mathbf{E}_1$), if no free charge exists on the boundary surface ($\rho_s = 0$). The program should be written for an arbitrarily positioned (oblique) boundary plane between dielectric media 1 and 2, which does not necessarily coincide with one of the coordinate planes of a Cartesian coordinate system; however, the plane contains the coordinate origin. Assume that relative permittivities ε_{r1} and ε_{r2} of the media are also known, as well as the Cartesian components of the normal on the boundary, directed from region 2 to region 1. *(ME2_4.m on IR)*

SOLUTION:
Figure S2.4 shows $\mathbf{E}_1$, $\mathbf{E}_2$, $\hat{\mathbf{n}}$, and the boundary plane for $\varepsilon_{r1} = 2.1$, $\varepsilon_{r2} = 4$, `E1 = [1 2 1]` V/m, and `n = [1 0 1]`.

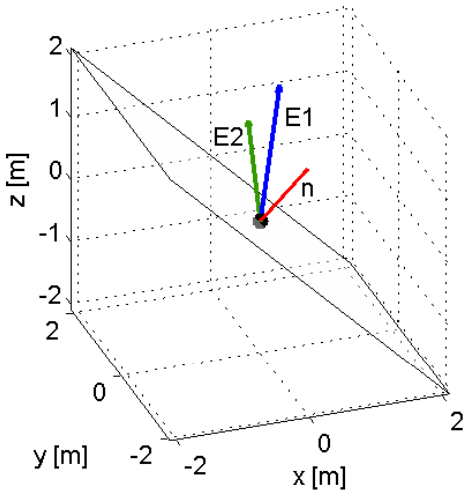

Figure S2.4 MATLAB computation of dielectric–dielectric boundary conditions for an arbitrarily positioned (oblique) boundary plane between media 1 and 2 ($\rho_s = 0$ on the boundary); for MATLAB Exercise 2.4.

```
%
%   Book: MATLAB-Based Electromagnetics (Pearson Prentice Hall)
%   Author: Branislav M. Notaros
%   Instructor Resources
%   (c) 2011
%
%   This MATLAB code or any part of it may be used only for
%   educational purposes associated with the book
%
%
%

% Dielectric - dielectric boundary conditions (rhos = 0), oblique plane

clear all;
close all;

% Vector normal to the boundary
disp('Specify the normal on the boundary.');
disp('If it is not unit vector, it will be normalized in the program.');

NORMALx = input('Enter x-component of the normal: ');
NORMALy = input('Enter y-component of the normal: ');
NORMALz = input('Enter z-component of the normal: ');
% Electric field vector
Ex1 = input('Enter x-component of E-field in medium 1, in V/m: ');
Ey1 = input('Enter y-component of E-field in medium 1, in V/m: ');
Ez1 = input('Enter z-component of E-field in medium 1, in V/m: ');
% Dielectrics
EPSR1 = input('Enter the relative permittivity of medium 1: ');
EPSR2 = input('Enter the relative permittivity of medium 2: ');


% Checking if normal vector is zero, which is unacceptable
if (NORMALx~=0 || NORMALy~=0 || NORMALz~=0)
NORMALmag = sqrt(NORMALx^2 + NORMALy^2 + NORMALz^2);
NORMALx = NORMALx/NORMALmag;
NORMALy = NORMALy/NORMALmag;
NORMALz = NORMALz/NORMALmag;
NORMAL = [NORMALx, NORMALy, NORMALz]; %unit vector

Emag = sqrt(Ex1^2 + Ey1^2 + Ez1^2);
E1 = [Ex1,Ey1,Ez1];

    % Angle between normal vector on the boundary and electric field
    % intensity vector in the first dielectric
    alphaAngle = acos((dot(NORMAL,E1'))/Emag);
    % Normal and tangential components of the electric field intensity vector
    % in the first dielectric
    E1normal = Emag*cos(alphaAngle).*NORMAL;
    E1tangential = E1 - E1normal;
```

```
    % Using boundary conditions -- calculation of electric field vector in
    % the second dielectric

    E2normal = E1normal.*EPSR1/EPSR2;
    E2tangential = E1tangential;
    E2 = E2normal + E2tangential;

    % Display of the results for electric field intensity vector in the
    % second dielectric
    disp('E-field in medium 2, in V/m, is:');
    fprintf('(%.3f)*ux',E2(1));
    fprintf(' + (%.3f)*uy',E2(2));
    fprintf(' + (%.3f)*uz\n',E2(3));

    A = [E1(1),E1(2),E1(3),E2(1),E2(2),E2(3),NORMALx,NORMALy,NORMALz];
    B = max(abs(A)) + 0.1;

    figure(1);
    if NORMALz ~= 0
        [x,y] = meshgrid(-B:2*B:B,-B:2*B:B);
        Bz = -1/NORMALz.*(NORMALx.*x + NORMALy.*y);
        h = surf(x,y,Bz);alpha(0.4);axis equal; hold on;
    elseif NORMALy ~= 0
        [x,z] = meshgrid(-B:2*B:B,-B:2*B:B);
        By = -1/NORMALy.*(NORMALx.*x + NORMALz.*z);
        h = surf (x,By,z);alpha(0.4);axis equal; hold on;
    elseif NORMALx ~= 0
        [y,z] = meshgrid(-B:2*B:B,-B:2*B:B);
        Bx = -1/NORMALx.*(NORMALy.*y + NORMALz.*z);
        h = surf (Bx,y,z);alpha(0.4);axis equal; hold on;
    end
    colormap (white);
    plot3(0,0,0,'ko','MarkerFaceColor','k'); hold on;
    quiver3(0,0,0,NORMALx, NORMALy, NORMALz,0,'r','LineWidth',2);
    text (NORMALx/2, NORMALy/2, NORMALz/2,'n');
    quiver3(0,0,0,E1(1),E1(2),E1(3),0,'b', 'LineWidth',2);
    text (E1(1)/2,E1(2)/2,E1(3)/2,'E1');
    quiver3(0,0,0,E2(1),E2(2),E2(3),0,'g','LineWidth',2);
    text (E2(1)/2,E2(2)/2,E2(3)/2,'E2');
    xlabel('x [m]');
    ylabel('y [m]');
    zlabel('z [m]');
else
    disp('Error - normal cannot be zero');
end
```