MATLAB EXERCISE 2.1 **GUI – pop-up menu for the permittivity table of materials.** Create a graphical user interface (GUI) in MATLAB to show values of the relative permittivity of selected materials given in Table 2.1 (from the book) as a pop-up menu. *[folder ME2_1(GUI) on IR]*[1]

SOLUTION:

Fig.S2.1 shows the layout of the GUI.

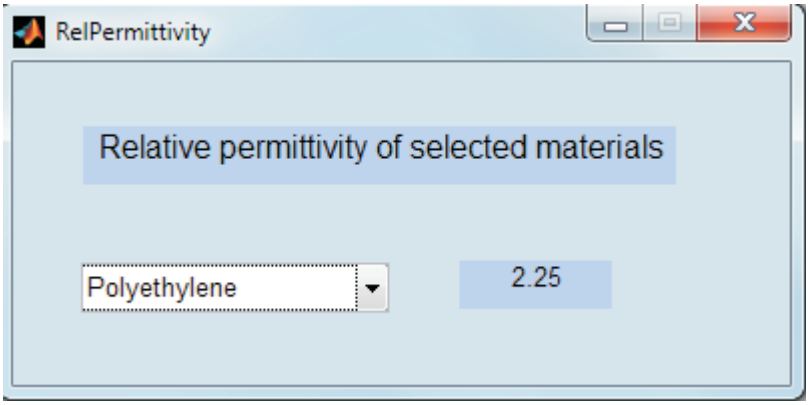

Figure S2.1 Layout of a graphical user interface (GUI) created in MATLAB to show ε_r values of different materials from Table 2.1 (from the book) as a pop-up menu; for MATLAB Exercise 2.1.

[1] *IR* = Instructor Resources (for the book).

```
%
%   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
%
%
%


%   GUI -- pop-up menu for the permittivity table of materials


function varargout = RelPermittivity(varargin)
% RELPERMITTIVITY M-file for RelPermittivity.fig
%      RELPERMITTIVITY, by itself, creates a new RELPERMITTIVITY or raises the existing
%      singleton*.
%
%      H = RELPERMITTIVITY returns the handle to a new RELPERMITTIVITY or the handle to
%      the existing singleton*.
%
%      RELPERMITTIVITY('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in RELPERMITTIVITY.M with the given input arguments.
%
%      RELPERMITTIVITY('Property','Value',...) creates a new RELPERMITTIVITY or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before RelPermittivity_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to RelPermittivity_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help RelPermittivity

% Last Modified by GUIDE v2.5 15-Aug-2012 12:59:27

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @RelPermittivity_OpeningFcn, ...
                   'gui_OutputFcn',  @RelPermittivity_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before RelPermittivity is made visible.
function RelPermittivity_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to RelPermittivity (see VARARGIN)

% Import the adequate relative permittivity
handles.blank = '';
handles.Air = '1.005';
handles.Alcohol = '25';
handles.Alumina = '8.8';
handles.Amber = '2.7';
handles.Ammonia = '22';
handles.Animal = '10';
handles.Bakelite = '4.74';
handles.Barium = '1,200';
handles.Diamond = '5-6';
handles.DrySoil = '2-6';
handles.Freon = '1';
handles.FusedSilica = '3.8';
handles.GalliumArsenide = '13';
handles.Germanium = '16';
handles.Glass = '4-10';
handles.Glycerin = '50';
handles.Human = '10';
handles.Ice = '75';
handles.Marble = '8';
handles.Mica = '5.4';
handles.Neoprene = '6.6';
handles.Nylon = '3.6-4.5';
handles.Oil = '2.3';
handles.Paper = '1.3-3';
handles.Paraffin = '2.1';
handles.Plexiglass = '3.4';
handles.Polyethylene = '2.25';
handles.Polystyrene = '2.56';
handles.Polyurethane = '1.1';
handles.Porcelain = '6';
handles.PVC = '2.7';
handles.Quartz = '5';
```

```
handles.Rubber = '2.4-3';
handles.Rutile = '89-173';
handles.Silicon = '11.9';
handles.SiliconNitride = '7.2';
handles.SodiumChloride = '5.9';
handles.Steatite = '5.8';
handles.Styrofoam = '1.03';
handles.Sulfur = '4';
handles.Tantalum = '25';
handles.Teflon = '2.1';
handles.Vacuum = '1';
handles.Vaseline = '2.16';
handles.Water = '81';
handles.WetSoil = '5-15';
handles.Wood = '2-5';

% Choose default command line output for RelPermittivity
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

set(0,'units','inches');
screenSize = get(0,'ScreenSize');
set(hObject,'Units','inches','Position',[screenSize(3)/2-(3.9375/2),screenSize(4)/2-↙
(1.5729/2),3.9375,1.5729]);

% UIWAIT makes RelPermittivity wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = RelPermittivity_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Determine the selected data set.

% Set current data to the selected data set.
switch get(handles.popupmenu1,'Value')
case 1 %''
```

```
    handles.current_data = handles.blank;
    set(handles.text2,'String',handles.current_data);
case 2 %'Air'
   handles.current_data = handles.Air;
   set(handles.text2,'String',handles.current_data);
case 3 %'Alcohol (ethyl)'
   handles.current_data = handles.Alcohol;
   set(handles.text2,'String',handles.current_data);
case 4 %'Alumina'
   handles.current_data = handles.Alumina;
   set(handles.text2,'String',handles.current_data);
case 5 %'Amber'
   handles.current_data = handles.Amber;
   set(handles.text2,'String',handles.current_data);
case 6 %'Ammonia (liquid)'
   handles.current_data = handles.Ammonia;
   set(handles.text2,'String',handles.current_data);
case 7 %'Animal muscle'
   handles.current_data = handles.Animal;
   set(handles.text2,'String',handles.current_data);
case 8 %'Bakelite'
   handles.current_data = handles.Bakelite;
   set(handles.text2,'String',handles.current_data);
case 9 %'Barium titanate'
   handles.current_data = handles.Barium;
   set(handles.text2,'String',handles.current_data);
case 10 %'Diamond'
   handles.current_data = handles.Diamond;
   set(handles.text2,'String',handles.current_data);
case 11 %'Dry soil'
   handles.current_data = handles.DrySoil;
   set(handles.text2,'String',handles.current_data);
case 12 %'Freon'
   handles.current_data = handles.Freon;
   set(handles.text2,'String',handles.current_data);
case 13 %'Fused silica'
   handles.current_data = handles.FusedSilica;
   set(handles.text2,'String',handles.current_data);
case 14 %'Gallium arsenide'
   handles.current_data = handles.GalliumArsenide;
   set(handles.text2,'String',handles.current_data);
case 15 %'Germanium'
   handles.current_data = handles.Germanium;
   set(handles.text2,'String',handles.current_data);
case 16 %'Glass'
   handles.current_data = handles.Glass;
   set(handles.text2,'String',handles.current_data);
case 17 %'Glycerin'
   handles.current_data = handles.Glycerin;
   set(handles.text2,'String',handles.current_data);
case 18 %'Human muscle'
   handles.current_data = handles.Human;
```

```
   set(handles.text2,'String',handles.current_data);
case 19 %'Ice'
   handles.current_data = handles.Ice;
   set(handles.text2,'String',handles.current_data);
case 20 %'Marble'
   handles.current_data = handles.Marble;
   set(handles.text2,'String',handles.current_data);
case 21 %'Mica (ruby)'
   handles.current_data = handles.Mica;
   set(handles.text2,'String',handles.current_data);
case 22 %'Neoprene'
   handles.current_data = handles.Neoprene;
   set(handles.text2,'String',handles.current_data);
case 23 %'Nylon'
   handles.current_data = handles.Nylon;
   set(handles.text2,'String',handles.current_data);
case 24 %'Oil'
   handles.current_data = handles.Oil;
   set(handles.text2,'String',handles.current_data);
case 25 %'Paper'
   handles.current_data = handles.Paper;
   set(handles.text2,'String',handles.current_data);
case 26 %'Paraffin'
   handles.current_data = handles.Paraffin;
   set(handles.text2,'String',handles.current_data);
case 27 %'Plexiglass'
   handles.current_data = handles.Plexiglass;
   set(handles.text2,'String',handles.current_data);
case 28 %'Polyethylene'
   handles.current_data = handles.Polyethylene;
   set(handles.text2,'String',handles.current_data);
case 29 %'Polystyrene'
   handles.current_data = handles.Polystyrene;
   set(handles.text2,'String',handles.current_data);
case 30 %'Polyurethane foam'
   handles.current_data = handles.Polyurethane;
   set(handles.text2,'String',handles.current_data);
case 31 %'Porcelain'
   handles.current_data = handles.Porcelain;
   set(handles.text2,'String',handles.current_data);
case 32 %'PVC'
   handles.current_data = handles.PVC;
   set(handles.text2,'String',handles.current_data);
case 33 %'Quartz'
   handles.current_data = handles.Quartz;
   set(handles.text2,'String',handles.current_data);
case 34 %'Rubber'
   handles.current_data = handles.Rubber;
   set(handles.text2,'String',handles.current_data);
 case 35 %'Rutile'
   handles.current_data = handles.Rutile;
   set(handles.text2,'String',handles.current_data);
```

```
case 36 %'Silicon'
   handles.current_data = handles.Silicon;
   set(handles.text2,'String',handles.current_data);
case 37 %'Silicon nitride'
   handles.current_data = handles.SiliconNitride;
   set(handles.text2,'String',handles.current_data);
case 38 %'Sodium chloride'
   handles.current_data = handles.SodiumChloride;
   set(handles.text2,'String',handles.current_data);
case 39 %'Steatite'
   handles.current_data = handles.Steatite;
   set(handles.text2,'String',handles.current_data);
case 40 %'Styrofoam'
   handles.current_data = handles.Styrofoam;
   set(handles.text2,'String',handles.current_data);
case 41 %'Sulfur'
   handles.current_data = handles.Sulfur;
   set(handles.text2,'String',handles.current_data);
case 42 %'Tantalum pentoxide'
   handles.current_data = handles.Tantalum;
   set(handles.text2,'String',handles.current_data);
case 43 %'Teflon'
   handles.current_data = handles.Teflon;
   set(handles.text2,'String',handles.current_data);
case 44 %'Vacuum'
   handles.current_data = handles.Vacuum;
   set(handles.text2,'String',handles.current_data);
case 45 %'Vaseline'
   handles.current_data = handles.Vaseline;
   set(handles.text2,'String',handles.current_data);
case 46 %'Water'
   handles.current_data = handles.Water;
   set(handles.text2,'String',handles.current_data);
case 47 %'Wet soil'
   handles.current_data = handles.WetSoil;
   set(handles.text2,'String',handles.current_data);
case 48 %'Wood'
   handles.current_data = handles.Wood;
   set(handles.text2,'String',handles.current_data);

end
% Save the handles structure.
guidata(hObject,handles)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

MATLAB EXERCISE 2.2 **Permittivity tensor of an anisotropic medium.** Based on Eq.(2.3) (from the book), compute in MATLAB the electric flux density vector, **D** (`D`), in an anisotropic dielectric if the electric field intensity vector and the relative-permittivity tensor are given by `E = [1 1 1]` V/m and `epsr = [2.51 0 0; 0 2.99 0; 0 0 4.11]`, respectively. Using MATLAB function `quiver3`, plot vectors **E** and **D** (see MATLAB Exercise 1.3). *(ME2_2.m on IR)*

SOLUTION:

The electric flux density vector is computed to be `D = [22.22 26.47 36.39]` pC/m^2. Fig.S2.2 shows a 3-D plot of normalized vectors **E** and **D**.

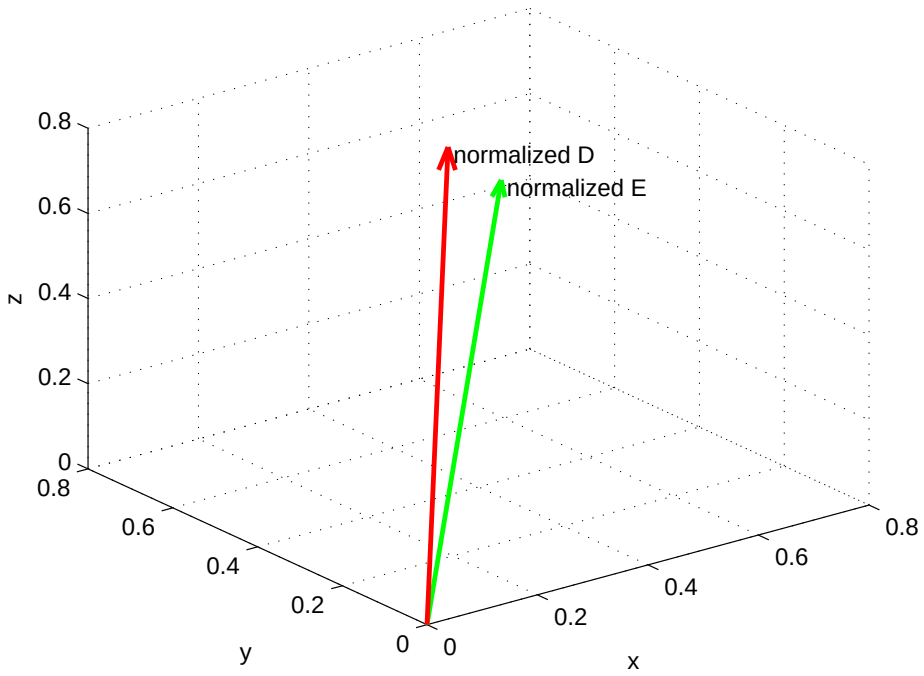

Figure S2.2 3-D plot of normalized vectors **E** and **D** in an anisotropic dielectric – using MATLAB function `quiver3`; for MATLAB Exercise 2.2.

```
%
%   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
%
%
%

% Permittivity tensor of an anisotropic medium
% This program calculates electric flux density vector for the given
% electric field intensity vector and permittivity tensor of an anisotropic
% dielectric

clear all;
close all;
EPS0 = 8.8542*10^(-12);

E = [1,1,1];
Emag = sqrt(E(1)^2 + E(2)^2 + E(3)^2);

% Permittivity tensor of an anisotropic medium
EPSR = zeros(3,3);
EPSR (1,1) = 2.51;
EPSR (2,2) = 2.99;
EPSR (3,3) = 4.11;

D = EPS0.*EPSR*E';
Dmag = sqrt(D(1)^2 + D(2)^2 + D(3)^2);

disp(D);

Enorm = E./Emag;  % normalized E
Dnorm = D./Dmag;  % normalized D

figure(1)
quiver3(0,0,0,Enorm(1),Enorm(2),Enorm(3),0,'g','LineWidth',2); hold on;
text(Enorm(1)-0.01,Enorm(2)-0.01,Enorm(3)-0.01,' normalized E');
quiver3(0,0,0,Dnorm(1),Dnorm(2),Dnorm(3),0,'r','LineWidth',2);
text(Dnorm(1)-0.01,Dnorm(2)-0.01,Dnorm(3)-0.01,' normalized D');hold off;
xlabel('x');
ylabel('y');
zlabel('z');
```

MATLAB EXERCISE 2.3 **GUI for the dielectric-strength table of materials.** Repeat MATLAB Exercise 2.1 but for the values of the dielectric strength (E_{cr}) of selected materials given in Table 2.2 (from the book). *[folder ME2_3(GUI) on IR]*

SOLUTION:

Fig.S2.3 shows the layout of the GUI.

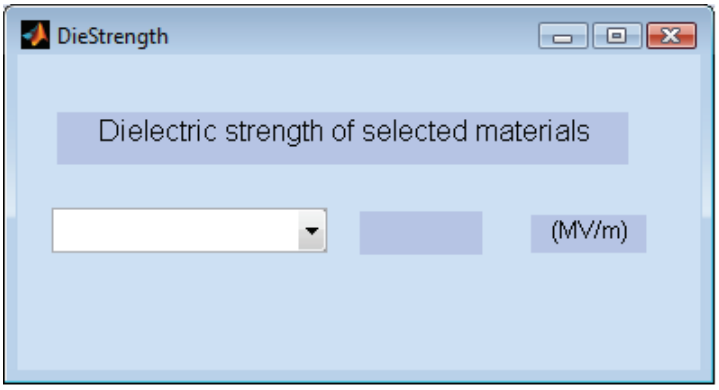

Figure S2.3 Layout of a graphical user interface (GUI) created in MATLAB to show E_{cr} values for different materials from Table 2.2 (from the book) as a pop-up menu; for MATLAB Exercise 2.3.

```
%
%   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
%
%
%


%    GUI for the dielectric-strength table of materials


function varargout = DieStrength(varargin)
% DIESTRENGTH M-file for DieStrength.fig
%      DIESTRENGTH, by itself, creates a new DIESTRENGTH or raises the existing
%      singleton*.
%
%      H = DIESTRENGTH returns the handle to a new DIESTRENGTH or the handle to
%      the existing singleton*.
%
%      DIESTRENGTH('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in DIESTRENGTH.M with the given input arguments.
%
%      DIESTRENGTH('Property','Value',...) creates a new DIESTRENGTH or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before DieStrength_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to DieStrength_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help DieStrength

% Last Modified by GUIDE v2.5 30-May-2010 16:49:28

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @DieStrength_OpeningFcn, ...
                   'gui_OutputFcn',  @DieStrength_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before DieStrength is made visible.
function DieStrength_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to DieStrength (see VARARGIN)
% Import the adequate dielectric strength
handles.blank = '';
handles.Air = '3';
handles.Alumina = '~35';
handles.Bakelite = '25';
handles.Barium = '7.5';
handles.Freon = '~8';
handles.FusedQuartz = '~1000';
handles.GalliumArsenide = '~40';
handles.Germanium = '~10';
handles.Glass = '30';
handles.Mica = '200';
handles.Oil = '15';
handles.Paper = '15';
handles.Paraffin = '~30';
handles.Polyethylene = '47';
handles.Polystyrene = '20';
handles.Porcelain = '11';
handles.Rubber = '25';
handles.Silicon = '~30';
handles.Nitride = '~1000';
handles.Teflon = '20';
handles.Vacuum = 'inf';
handles.Wood = '~10';


% Choose default command line output for DieStrength
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

set(0,'units','inches');
screenSize = get(0,'ScreenSize');
set(hObject,'Units','inches','Position',[screenSize(3)/2-(3.9479/2),screenSize(4)/2-↙
```

```
(1.8229/2),3.9479,1.8229]);

% UIWAIT makes DieStrength wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = DieStrength_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Determine the selected data set.

% Set current data to the selected data set.
switch get(handles.popupmenu1,'Value')
case 1
    handles.current_data = handles.blank;
    set(handles.text3,'String',handles.current_data);
case 2 %'Air'
   handles.current_data = handles.Air;
   set(handles.text3,'String',handles.current_data);
case 3 %'Alumina'
   handles.current_data = handles.Alumina;
   set(handles.text3,'String',handles.current_data);
case 4 %'Bakelite'
   handles.current_data = handles.Bakelite;
   set(handles.text3,'String',handles.current_data);
case 5 %'Barium titanate'
   handles.current_data = handles.Barium;
   set(handles.text3,'String',handles.current_data);
case 6 %'Freon'
   handles.current_data = handles.Freon;
   set(handles.text3,'String',handles.current_data);
case 7 %'Fused quartz'
   handles.current_data = handles.FusedQuartz;
   set(handles.text3,'String',handles.current_data);
case 8 %'Gallium arsenide'
   handles.current_data = handles.GalliumArsenide;
   set(handles.text3,'String',handles.current_data);
case 9 %'Germanium'
   handles.current_data = handles.Germanium;
```

```
   set(handles.text3,'String',handles.current_data);
case 10 %'Glass (plate)'
   handles.current_data = handles.Glass;
   set(handles.text3,'String',handles.current_data);
case 11 %'Mica'
   handles.current_data = handles.Mica;
   set(handles.text3,'String',handles.current_data);
case 12 %'Oil (mineral)'
   handles.current_data = handles.Oil;
   set(handles.text3,'String',handles.current_data);
case 13 %'Paper (impregnated)'
   handles.current_data = handles.Paper;
   set(handles.text3,'String',handles.current_data);
case 14 %'Paraffin'
   handles.current_data = handles.Paraffin;
   set(handles.text3,'String',handles.current_data);
case 15 %'Polyethylene'
   handles.current_data = handles.Polyethylene;
   set(handles.text3,'String',handles.current_data);
case 16 %'Polystyrene'
   handles.current_data = handles.Polystyrene;
   set(handles.text3,'String',handles.current_data);
case 17 %'Porcelain'
   handles.current_data = handles.Porcelain;
   set(handles.text3,'String',handles.current_data);
case 18 %'Rubber (hard)'
   handles.current_data = handles.Rubber;
   set(handles.text3,'String',handles.current_data);
case 19 %'Silicon'
   handles.current_data = handles.Silicon;
   set(handles.text3,'String',handles.current_data);
case 20 %'Silicon nitride'
   handles.current_data = handles.Nitride;
   set(handles.text3,'String',handles.current_data);
case 21 %'Teflon'
   handles.current_data = handles.Teflon;
   set(handles.text3,'String',handles.current_data);
case 22 %'Vacuum'
   handles.current_data = handles.Vacuum;
   set(handles.text3,'String',handles.current_data);
case 23 %'Wood (douglas fir)'
   handles.current_data = handles.Wood;
   set(handles.text3,'String',handles.current_data);

end
% Save the handles structure.
guidata(hObject,handles)


% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1
```

```
% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

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]`.

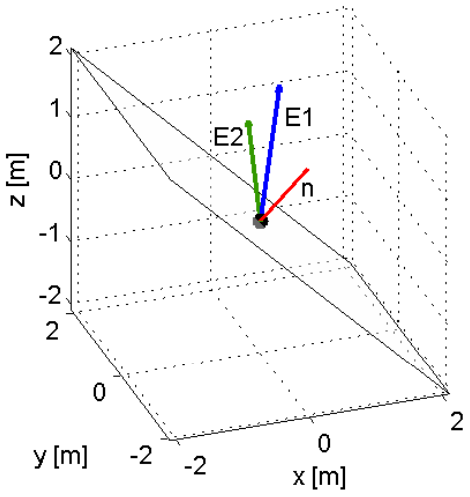

Figure S2.4 MATLAB computation of dielectric–dielectric boundary conditions for an arbitrarily positioned (oblique) boundary plane between media 1 and 2 ($\rho_s = 0$ on the boundary); for MATLAB Exercise 2.4.

```
%
%   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
```

MATLAB EXERCISE 2.5 **Oblique boundary plane with nonzero surface charge.** Repeat the previous MATLAB exercise but for an oblique boundary plane with nonzero surface charge of density ρ_s on it. *(ME2_5.m on IR)*

SOLUTION:

```
%
%   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 - Oblique boundary plane with
% nonzero surface charge

clear all;
close all;
EPS0 = 8.8542*10^(-12);

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: ');

RHOS = input('Enter the surface charge density in pC/m^2: ');
RHOS = RHOS*10^(-12);
% Dielectrics
EPSR1 = input('Enter the relative permittivity of medium 1: ');
EPSR2 = input('Enter the relative permittivity of medium 2: ');

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];

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

alphaAngle = acos((dot(NORMAL,E1'))/Emag);
E1normal = Emag*cos(alphaAngle).*NORMAL;
E1tangential = E1 - E1normal;

E2normal = (E1normal.*EPSR1*EPS0 - RHOS.*NORMAL)/(EPSR2*EPS0);
E2tangential = E1tangential;
E2 = E2normal + E2tangential;

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
```

MATLAB EXERCISE 2.6 **Horizontal charge-free boundary plane.** Repeat MATLAB Exercise 2.4 but for a charge-free boundary plane coinciding with the xy-plane ($z = 0$) of the Cartesian coordinate system (in Fig.2.4). In specific, write a separate MATLAB program specialized for this particular boundary plane (and no other plane). Compare the results with those obtained by the general program for an oblique plane run with `n = [0 0 1]`. *(ME2_6.m on IR)*

SOLUTION:

```
%
%   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 - Horizontal charge-free
% boundary plane

clear all;
close all;

NORMAL = [0,0,1];

% 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: ');

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

E1normal = Ez1.*NORMAL;
E1tangential = E1 - E1normal;

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

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

figure(1);
[x,y] = meshgrid(-B:B/4:B,-B:B/4:B);
Bz = NORMAL(1)*x + NORMAL(2)*y;
h = surf(x,y,Bz);axis equal; hold on;
colormap (white);
plot3(0,0,0,'ko','MarkerFaceColor','k'); hold on;
```

```
quiver3(0,0,0,NORMAL(1),NORMAL(2),NORMAL(3),0,'r','LineWidth',2);
text (0,0,1/2,'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]');
```

MATLAB EXERCISE 2.7 **Horizontal boundary plane with surface charge.** Repeat the previous MATLAB exercise but for a horizontal boundary plane with free surface charge ($\rho_s \neq 0$). *(ME2_7.m on IR)*

SOLUTION:

```
%
%   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 - Horizontal boundary plane
% with surface charge

clear all;
close all;
EPS0 = 8.8542*10^(-12);

NORMAL = [0,0,1];

% 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: ');
% Surface charge density
RHOS = input('Enter the surface charge density in pC/m^2: ');
RHOS = RHOS*10^(-12);
% Dielectrics
EPSR1 = input('Enter the relative permittivity of medium 1: ');
EPSR2 = input('Enter the relative permittivity of medium 2: ');

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

E1normal = Ez1.*NORMAL;
E1tangential = E1 - E1normal;

E2normal = (E1normal.*EPSR1*EPS0 - RHOS.*NORMAL)/(EPSR2*EPS0);
E2tangential = E1tangential;
E2 = E2normal + E2tangential;

disp('E-field in medium 2, in V/m, is:');
fprintf('(%f)*ux',E2(1));
fprintf(' + (%f)*uy',E2(2));
fprintf(' + (%f)*uz\n',E2(3));

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

figure(1);
[x,y] = meshgrid(-B:B/4:B,-B:B/4:B);
```

```
Bz = NORMAL(1)*x + NORMAL(2)*y;
h = surf(x,y,Bz);axis equal; hold on;
colormap (white);
plot3(0,0,0,'ko','MarkerFaceColor','k'); hold on;
quiver3(0,0,0,NORMAL(1),NORMAL(2),NORMAL(3),0,'r','LineWidth',2);
text (0,0,1/2,'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]');
```

MATLAB EXERCISE 2.8 **MATLAB computations of boundary conditions.** Assume that the plane $z = 0$ separates medium 1 ($z > 0$) and medium 2 ($z < 0$), with relative permittivities $\varepsilon_{r1} = 4$ and $\varepsilon_{r2} = 2$, respectively. The electric field intensity vector in medium 1 near the boundary (for $z = 0^+$) is $\mathbf{E}_1 = (4\,\hat{\mathbf{x}} - 2\,\hat{\mathbf{y}} + 5\,\hat{\mathbf{z}})$ V/m. In MATLAB, find the electric field intensity vector in medium 2 near the boundary (for $z = 0^-$), $\mathbf{E}_2$, if (a) no free charge exists on the boundary ($\rho_s = 0$) and (b) there is a surface charge of density $\rho_s = 53.12$ pC/m^2 on the boundary.

SOLUTION:

Using MATLAB programs from the previous two MATLAB exercises (for a horizontal boundary plane, $z = 0$) or general programs for an oblique plane (from MATLAB Exercises 2.4 and 2.5), we obtain: (a) $\mathbf{E}_2 = (4\,\hat{\mathbf{x}} - 2\,\hat{\mathbf{y}} + 10\,\hat{\mathbf{z}})$ V/m ($\rho_s = 0$) and (b) $\mathbf{E}_2 = (4\,\hat{\mathbf{x}} - 2\,\hat{\mathbf{y}} + 7\,\hat{\mathbf{z}})$ V/m ($\rho_s \neq 0$).

MATLAB EXERCISE 2.9 **Symbolic Laplacian in different coordinate systems.** By symbolic programming in MATLAB, write functions `LaplaceCar()`, `LaplaceCyl()`, and `LaplaceSph()` that find Laplacian in the Cartesian, cylindrical, and spherical coordinate systems, respectively, based on Eqs.(2.8), (2.10), and (2.11) (from the book) (see MATLAB Exercise 1.33). Test the codes with $f_{\text{Cartesian}} = 3x^2y^3z$, $f_{\text{cylindrical}} = r^2\cos\phi\, z^3$, and $f_{\text{spherical}} = r^2\sin\theta\cos\phi$. *(LaplaceCar.m, LaplaceCyl.m, LaplaceSph.m, and ME2_9.m on IR)*

```
%
%   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
%
%
%


%  Symbolic Laplacian in Cartesian coordinates

% This function computes div(grad(f)) in Cartesian coordinate system and
% returns symbolic expression

function F = LaplaceCar(f)
syms x y z

Fx = diff(f,x,2);
Fy = diff(f,y,2);
Fz = diff(f,z,2);
F = Fx + Fy + Fz;
```

```
%
%   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
%
%
%


%  Symbolic Laplacian in cylindrical coordinates

% This function computes div(grad(f)) in Cylindrical coordinate system and
% returns symbolic expression

function F = LaplaceCyl(f)
syms r phi z

Fr = diff(r*diff(f,r),r);
Fp = diff(f,phi,2);
Fz = diff(f,z,2);
F = 1/r*Fr + 1/r^2*Fp + Fz;
```

```
%
%   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
%
%
%


%     Symbolic Laplacian in spherical coordinates


% This function computes div(grad(f)) in Spherical coordinate system and
% returns symbolic expression

function F = LaplaceSph(f)
syms r phi theta

Fr = diff(r^2*diff(f,r),r);
Fp = diff(f,phi,2);
Ft = diff(sin(theta)*diff(f,theta),theta);
F = 1/r^2*Fr + 1/(r*sin(theta))^2*Fp + 1/(r^2*sin(theta))*Ft;
```

```
%
%   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
%
%
%

clear all;
close all;

syms x y z;

f = 3*x^2*y^3*z;
F = LaplaceCar(f);
pretty(F);
```

```
%
%   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
%
%
%

clear all;
close all;

syms r phi z;

f = r^2*cos(phi)*z^3;
F = LaplaceCyl(f);
pretty(F);
```

```
%
%   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
%
%
%

clear all;
close all;

syms r phi theta;

f = r^2*cos(phi)*sin(theta);
F = LaplaceSph(f);
pretty(F);
```

MATLAB EXERCISE 2.10 **FD-based MATLAB code – iterative solution.** Write a computer program in MATLAB for the iterative finite-difference analysis of a coaxial cable of square cross section, Fig.2.4 (from the book), based on Eq.(2.16) (from the book). Assume that $a = 1$ cm, $b = 3$ cm, $V_a = 1$ V, and $V_b = -1$ V. Plot the results for the distribution of the potential and the electric field intensity in the space between the conductors, and the surface charge density on the surfaces of conductors, taking the grid spacing to be $d = a/10$ and the tolerance of the potential $\delta_V = 10^{-8}$ V. Compute the total charge per unit length of the inner and the outer conductor, taking $d = a/N$ and $N = 2, 3, 5, 7, 9, 10, 12,$ and 25, respectively. *(ME2_10.m on IR)*

SOLUTION:

Simulation results for the distribution of the potential and the electric field intensity in the space between the conductors and for the charge distribution of the conductors of the cable are plotted using MATLAB functions `surf`, `quiver`, and `plot`, respectively, and the plots, for $d = a/10$ and $\delta_V = 10^{-8}$ V, are shown in Figs.S2.5–S2.7. The computed total per-unit-length charges of conductors, taking $d = a/N$ and $N = 2, 3, 5, 7, 9, 10, 12,$ and 25, respectively, are tabulated in Table S2.1.

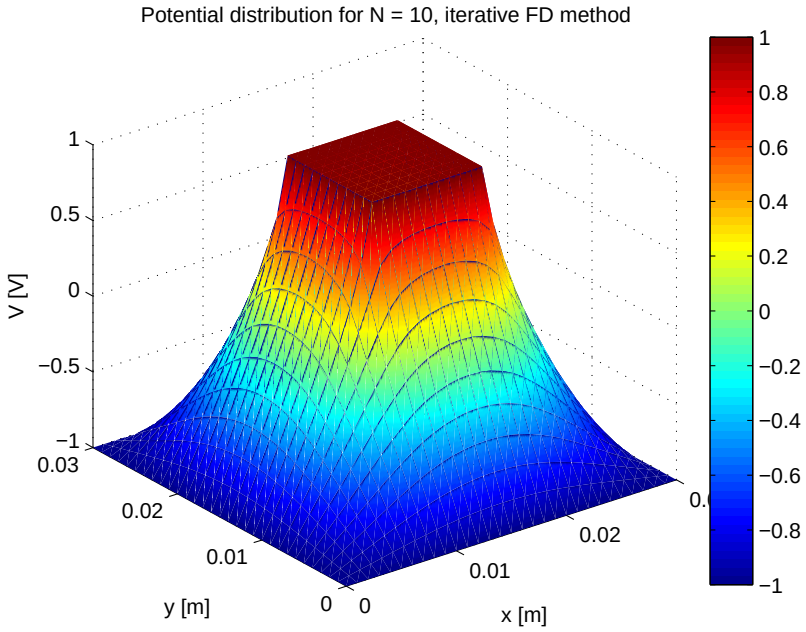

Figure S2.5 Electric potential in the space between the conductors of the coaxial cable of square cross section in Fig.2.4 (from the book) – results by the iterative finite-difference technique implemented in MATLAB; for MATLAB Exercise 2.10.

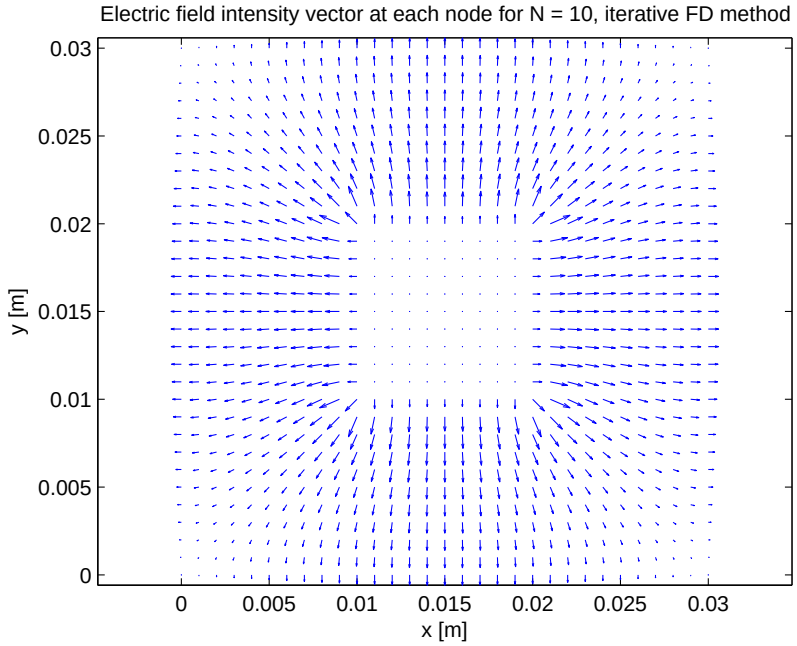

Figure S2.6 Electric field intensity vector corresponding to the potential in Fig.S2.5; for MATLAB Exercise 2.10.

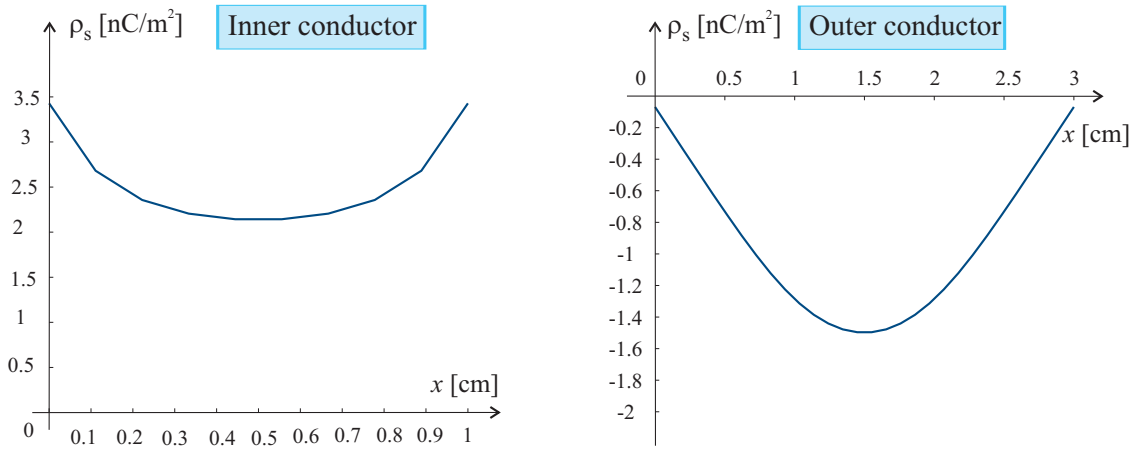

Figure S2.7 Computed surface charge density on the surfaces of conductors of the square coaxial cable in Fig.2.4 (from the book); for MATLAB Exercise 2.10.

Table S2.1 MATLAB FD results for conductor p.u.l. charges; for MATLAB Exercise 2.10.

N	Q'_{inner}[C/m]	Q'_{outer}[C/m]
2	0.8854×10^{-10}	-1.0625×10^{-10}
3	0.9391×10^{-10}	-1.0983×10^{-10}
5	0.9847×10^{-10}	-1.1084×10^{-10}
7	1.0066×10^{-10}	-1.1083×10^{-10}
9	1.0201×10^{-10}	-1.1074×10^{-10}
10	1.0252×10^{-10}	-1.1069×10^{-10}
12	1.0332×10^{-10}	-1.1060×10^{-10}
25	1.0579×10^{-10}	-1.1032×10^{-10}

```
%
%   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
%
%
%

%  FD computer program - iterative solution

clear all;
close all;

format long;
a = 0.01;
b = 0.03;
Va = 1;
Vb = -1;
deltaV = 10^(-8);
EPS0 = 8.8542*10^(-12);
maxIter = 500000;

N = [2 3 5 7 9 10 12 25];
%N = 10;
%m = 1;
for m = 1 : length(N)
    d = a/N(m);
   %number of inner nodes
    N1 = N(m) + 1;
    %number of outer nodes
    N2 = b/a *N(m) + 1;
    V = ones(N2,N2)*(Va+Vb)/2;
    %outer boundary
    V(1,:) = Vb; V(:,1) = Vb; V(:,N2)=Vb; V(N2,:) = Vb;
    %inner boundary
    V((N2-N1)/2+1:(N2+N1)/2,(N2-N1)/2+1:(N2+N1)/2) = Va;


    iterationCounter = 0;
    maxError = 2*deltaV;
    while (maxError > deltaV)&&(iterationCounter < maxIter)
        Vprev = V;
        for i = 2 : N2-1
            for j = 2 : N2-1
                if V(i,j)~=Va
                     V(i,j)=(Vprev(i-1,j)+ Vprev(i,j-1)+Vprev(i+1,j)+Vprev(i,j+1))/4;
                end;
            end;
```

```
        end;
        difference = max(abs(V-Vprev));
        maxError = max(difference);
       iterationCounter = iterationCounter + 1;
     end;

   [x,y]= meshgrid(0:d:b);
   [Ex,Ey] = gradient(-V,d,d);

sigmaOut = zeros(1,N2-1);
sigmaIn = zeros(1,N1-1);
for i = 1:N2-1
sigmaOut(i) = EPS0/2/d*(3/2*V(1,i)-2*V(2,i)+1/2*V(3,i)+...
    3/2*V(1,i+1)-2*V(2,i+1)+1/2*V(3,i+1));
end;
k = (N2-N1)/2 + 1;

for i = k:(N2 + N1)/2 -1
sigmaIn(i-k+1)=EPS0/2/d*(3/2*V(k,i)-2*V(k-1,i)+1/2*V(k-2,i)+...
    3/2*V(k,i+1)-2*V(k-1,i+1) + 1/2*V(k-2,i+1));
end;

Qouter(m) = 4*d*sum(sigmaOut);
Qinner(m) = 4*d*sum(sigmaIn);


Cout(m)= Qouter(m)/(Vb - Va);
Cin(m) = Qinner(m)/(Va - Vb);

 figure(4*m - 3);
 quiver (x,y,Ex,Ey); xlabel('x [m]'); ylabel('y [m]');
 title(['Electric field intensity vector at each node for N = '...
     ,num2str(N(m)),', iterative FD method']);axis equal;

   figure(4*m - 2);
   surf(x,y,V); shading interp; colorbar;
   xlabel('x [m]'); ylabel('y [m]'); zlabel('V [V]');
   title(['Potential distribution for N = ', num2str(N(m)),...
       ', iterative FD method']);

   figure(4*m-1);
   dinner = a/(length(sigmaIn)-1);
   innerCond = 0:dinner:a;
   plot(innerCond, sigmaIn);
   xlabel('x [m]'); ylabel('\rho_{in} [C/m^2]');

   figure(4*m);
   douter = b/(length(sigmaOut)-1);
   outerCond = 0:douter:b;
   plot(outerCond,sigmaOut);
   xlabel('x [m]'); ylabel('\rho_{out} [C/m^2]');
```

```
    clear Vstart V ;
   error(m) = (Qinner(m) + Qouter(m))/Qouter(m)*100;
end;
```

MATLAB EXERCISE 2.11 **Computation of matrices for a direct FD method.** As an alternative to the iterative technique based on Eq.(2.16) (from the book), the finite-difference analysis of a square coaxial cable, in Fig.2.4(a) (from the book), can be carried out by directly solving the system of linear algebraic equations with the potentials at interior grid nodes in Fig.2.4(b) as unknowns [applying Eq.(2.15) (from the book) to each interior grid node, we get a set of simultaneous equations the number of which equals the number of unknown potentials]. The system of equations, in which known potentials at nodes on the surface of conductors appear on the right-hand side of equations, is solved by the Gaussian elimination method (or by matrix inversion). In this MATLAB exercise, write a function `mACfd()` that establishes the system of equations, i.e., that computes matrices $[A]$ and $[C]$ of the matrix equation, for the direct FD analysis of the cable. *(mACfd.m on IR)*

SOLUTION:

```
%
%   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
%
%
%


%   Computation of matrices for a direct FD method


function[A,C]= mACfd(Vstart,N2)

C = zeros(N2^2,1);
for i=1:N2
    for j = 1:N2
        k = (i-1)*N2 + j;

        A(k,k)= 1;

        if (Vstart(i,j)==0)
            A(k,k-N2)= -1/4; %up
            A(k,k+N2)= -1/4; %down
            A(k,k+1)=-1/4;   %right
            A(k,k-1)= -1/4;  %left
        else C(k)=Vstart(i,j);
        end;

    end;
end;
```

MATLAB EXERCISE 2.12 **FD-based MATLAB code – direct solution.** Write the main MATLAB code for the direct finite-difference analysis of a square coaxial cable, Fig.2.4, based on Eq.(2.15) (from the book) – see the previous MATLAB exercise. Compute and plot the same quantities as in MATLAB Exercise 2.10, and compare the results obtained by the two programs. *(ME2_12.m on IR)*

SOLUTION:

Plots of the computed potential, field, and charge distributions, using the direct FD technique, look identical to those in Figs.S2.5–S2.7, and the total p.u.l. charges of conductors are given in Table S2.2, so an excellent agreement with the results by the iterative FD technique (MATLAB Exercise 2.10) is observed.

Table S2.2 Conductor p.u.l. charges obtained by the direct FD code; for MATLAB Exercise 2.12.

N	Q'_{inner}[C/m]	Q'_{outer}[C/m]
2	0.8854×10^{-10}	-1.0625×10^{-10}
3	0.9391×10^{-10}	-1.0983×10^{-10}
5	0.9847×10^{-10}	-1.1084×10^{-10}
7	1.0066×10^{-10}	-1.1083×10^{-10}
9	1.0201×10^{-10}	-1.1074×10^{-10}
10	1.0252×10^{-10}	-1.1069×10^{-10}
12	1.0332×10^{-10}	-1.1060×10^{-10}
25	1.0580×10^{-10}	-1.1032×10^{-10}

```
%
%   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
%
%
%

%   FD computer program - direct solution

clear all;
close all;

a = 0.01;
b = 0.03;
Va = 1;
Vb = -1;
EPS0=8.8542*10^(-12);
N = [2 3 5 7 9 10 12 25];
%N = 10;
%m = 1;
for m = 1 : length(N)
    d = a/N(m);
    %number of inner nodes
    N1 = N(m) + 1;
    %number of outer node
    N2 = b/a *N(m) + 1;
    %K = N2^2; % total nodes

    Vstart = zeros(N2,N2);
    Vstart(1,:) = Vb;
    Vstart(:,1) = Vb;
    Vstart(N2,:) = Vb;
    Vstart(:,N2) = Vb;

    lim1 = (N2-N1)/2 + 1;
    lim2 = (N2+N1)/2;
    Vstart(lim1:lim2,lim1:lim2)=Va;

    % A,C matrix
    [A,C]= mACfd(Vstart,N2);
    V = inv(A)*C;

   %V2D
    for i = 1:N2
    V2D(i,:) = V((i-1)*N2+1:i*N2);
    end;
```

```
   [x,y]= meshgrid(0:d:b);
   [Ex,Ey] = gradient(-V2D,d,d);

   sigmaOut = zeros(1,N2-1);
   sigmaIn = zeros(1,N1-1);

for i = 1:N2-1
sigmaOut(i) = EPS0/2/d*(3/2*V2D(1,i)-2*V2D(2,i)+1/2*V2D(3,i)+...
    3/2*V2D(1,i+1)-2*V2D(2,i+1)+1/2*V2D(3,i+1));
end;
k = (N2-N1)/2 + 1;

for i = k:(N2 + N1)/2 -1
sigmaIn(i-k+1)=EPS0/2/d*(3/2*V2D(k,i)-2*V2D(k-1,i)+1/2*V2D(k-2,i)+...
    3/2*V2D(k,i+1)-2*V2D(k-1,i+1) + 1/2*V2D(k-2,i+1));
end;

Qouter(m) = 4*d*sum(sigmaOut);
Qinner(m) = 4*d*sum(sigmaIn);


Cout(m)= Qouter(m)/(Vb - Va);
Cin(m) = Qinner(m)/(Va - Vb);

figure(4*m - 3);
 quiver (x,y,Ex,Ey); xlabel('x [m]'); ylabel('y [m]');
 title(['Electric field intensity vector at each node for N = '...
     ,num2str(N(m)),', direct FD method']);axis equal;

   figure(4*m - 2);
   surf(x,y,V2D); shading interp; colorbar;
   xlabel('x [m]'); ylabel('y [m]'); zlabel('V [V]');
   title(['Potential distribution for N = ', num2str(N(m))...
       ', direct FD method']);

   figure(4*m - 1);
   dinner = a/(length(sigmaIn)-1);
   innerCond = 0:dinner:a;
   plot(innerCond, sigmaIn);
   xlabel('x [m]'); ylabel('\rho_{in} [C/m^2]');

   figure(4*m);
   douter = b/(length(sigmaOut)-1);
   outerCond = 0:douter:b;
   plot(outerCond,sigmaOut);
   xlabel('x [m]'); ylabel('\rho_{out} [C/m^2]');

error(m) = (Qinner(m)+Qouter(m))/Qouter(m)*100;
clear Vstart V V2D sigmaIn sigmaOut;
end;
```

MATLAB EXERCISE 2.13 **Capacitance calculator and GUI for multiple structures.**
Create a capacitance calculator in the form of a graphical user interface (GUI) in MATLAB to calculate and show the capacitance or per-unit-length capacitance of a coaxial cable [Fig.2.9(b) (from the book)], microstrip transmission line [Fig.2.9(e)], parallel-plate capacitor [Fig.2.9(d)], spherical capacitor [Fig.2.9(a)], and strip transmission line [Fig.2.9(f)], respectively, with the names of structures appearing in a pop-up menu. *[folder ME2_13(GUI) on IR]*

SOLUTION:
Figure S2.8 shows the GUI if, for instance, a strip line is selected in the pop-up menu.

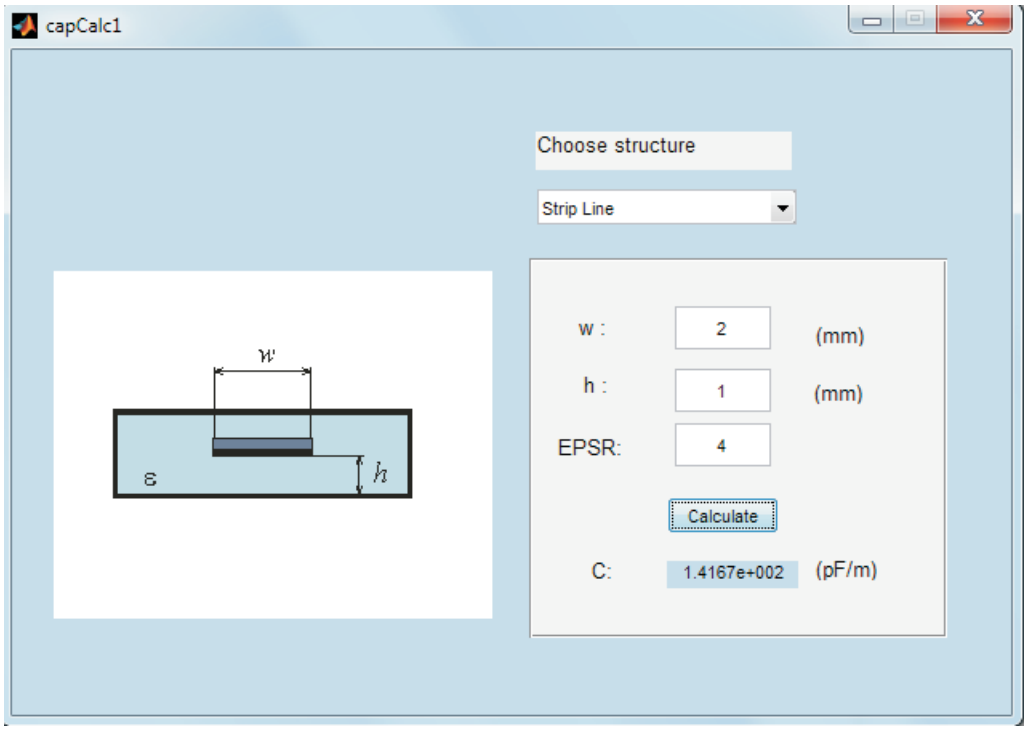

Figure S2.8 MATLAB capacitance calculator and graphical user interface for multiple structures: GUI in the case a strip line is selected in the pop-up menu; for MATLAB Exercise 2.13.

```
%
%   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
%
%
%


%    Capacitance calculator and GUI for multiple structures


function varargout = capCalc1(varargin)
% CAPCALC1 M-file for capCalc1.fig
%      CAPCALC1, by itself, creates a new CAPCALC1 or raises the existing
%      singleton*.
%
%      H = CAPCALC1 returns the handle to a new CAPCALC1 or the handle to
%      the existing singleton*.
%
%      CAPCALC1('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CAPCALC1.M with the given input arguments.
%
%      CAPCALC1('Property','Value',...) creates a new CAPCALC1 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before capCalc1_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to capCalc1_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help capCalc1

% Last Modified by GUIDE v2.5 01-Jun-2010 21:53:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @capCalc1_OpeningFcn, ...
                   'gui_OutputFcn',  @capCalc1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before capCalc1 is made visible.
function capCalc1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to capCalc1 (see VARARGIN)

% Import the figure of the current structure

handles.coaxcable = imread('coaxcable.png');
handles.microstrip = imread('microstrip.png');
handles.ppcap = imread('ppcap.png');
handles.sphcap = imread('sphcap.png');
handles.stripline = imread('stripline.png');

% Set the current data value.

set(handles.uipanel1,'Visible','off');
set(handles.axes2,'Visible','off');

% Choose default command line output for capCalc1
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

set(0,'units','inches');
screenSize = get(0,'ScreenSize');
set(hObject,'Units','inches','Position',[screenSize(3)/2-(6.5/2),screenSize(4)/2-(4.3229↙
/2),6.5,4.3229]);

% UIWAIT makes capCalc1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = capCalc1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Determine the selected data set.

global i;
% Set current data to the selected data set.
switch get(handles.popupmenu1,'Value')
case 1
   set(handles.uipanel1,'Visible','off');
   cla reset; axis off;
case 2
   handles.current_data = handles.coaxcable;
   imshow(handles.current_data);
   set(handles.text1,'String','a :');
   set(handles.text4,'String','b :');
   set(handles.text6,'String','(mm)');
   set(handles.text11,'String','(pF/m)');
   set(handles.uipanel1,'Visible','on');
   i = 1;
case 3
   handles.current_data = handles.microstrip;
   imshow(handles.current_data);
   set(handles.text1,'String','w :');
   set(handles.text4,'String','h :');
   set(handles.text6,'String','(mm)');
   set(handles.text11,'String','(pF/m)');
   set(handles.uipanel1,'Visible','on');
   i = 2;
case  4
   handles.current_data = handles.ppcap;
   imshow(handles.current_data);
   set(handles.text1,'String','S :');
   set(handles.text4,'String','d :');
   set(handles.text6,'String','(mm^2) :');
   set(handles.text11,'String','(pF)');
   set(handles.uipanel1,'Visible','on');
   i = 3;
case 5
   handles.current_data = handles.sphcap;
   imshow(handles.current_data);
   set(handles.text1,'String','a :');
   set(handles.text4,'String','b :');
   set(handles.text6,'String','(mm)');
   set(handles.text11,'String','(pF)');
```

```
   set(handles.uipanel1,'Visible','on');
   i = 4;
case 6
   handles.current_data = handles.stripline;
   imshow(handles.current_data);
   set(handles.text1,'String','w :');
   set(handles.text4,'String','h :');
   set(handles.text6,'String','(mm)');
   set(handles.text11,'String','(pF/m)');
   set(handles.uipanel1,'Visible','on');
   i = 5;

end
 global ready;
 global var;
 ready = [0 0 0];
 var = [0 0 0];
 set(handles.pushbutton1,'Enable','off');
 set(handles.edit1,'String','');
 set(handles.edit2,'String','');
 set(handles.edit3,'String','');
 set(handles.text10,'String','');

 %Save the handles structure.
  guidata(hObject,handles)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

 global ready;
 ready = [0 0 0];
 global var;
 var = [0 0 0];



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.

handles.edit1 = str2double(get(hObject,'String'));
 global ready;
 global var;
if (isnan(handles.edit1));
    msgbox('Invalid input','Error');
    ready(1)= 0;
else
    ready(1)= 1;
    var(1) = handles.edit1;
end;
if (ready == [1 1 1])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

% --- Executes during object creation, after setting all properties.

handles.edit2 = str2double(get(hObject,'String'));
global ready;
global var;
```

```
if (isnan(handles.edit2));
    msgbox('Invalid input','Error');
    ready(2)= 0;
else
    ready(2)= 1;
    var(2) = handles.edit2;
end;
if (ready == [1 1 1])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double

% --- Executes during object creation, after setting all properties.

handles.edit3 = str2double(get(hObject,'String'));
global ready;
global var;
if (isnan(handles.edit3));
    msgbox('Invalid input','Error');
    ready(3)= 0;
else
    ready(3)= 1;
    var(3) = handles.edit3;
end;
if (ready == [1 1 1])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
```

```
end;

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes during object creation, after setting all properties.
global var;
global i;
EPS0 = 8.8542*10^(-12);
mm2m = 10^(-3);
mmsq2msq = 10^(-6);

if  i == 1;
    a = var(1)*mm2m;
    b = var(2)*mm2m;
    EPSR = var(3);
    EPS = EPS0*EPSR;
    C = capacitanceCoaxCable(EPS,a,b);
else if  i == 2;
        w = var(1)*mm2m;
        h = var(2)*mm2m;
        EPSR = var(3);
        EPS = EPS0*EPSR;
        C = capacitanceMicrostrip(EPS,w,h);
    else if i == 3;
            S = var(1)*mmsq2msq;
            d = var(2)*mm2m;
            EPSR = var(3);
            EPS = EPS0*EPSR;
            C = capacitancePPCapacitor(EPS,S,d);
        else if  i == 4;
                a = var(1)*mm2m;
                b = var(2)*mm2m;
                EPSR = var(3);
                EPS = EPS0*EPSR;
```

```
                C = capacitanceSphCapacitor(EPS,a,b);
            else
                w = var(1)*mm2m;
                h = var(2)*mm2m;
                EPSR = var(3);
                EPS = EPS0*EPSR;
                C = capacitanceStripline(EPS,w,h);
            end;
        end;
    end;
end;
C = C*10^12;
set(handles.text10,'String',num2str(C,'%.4e'));


% --- Executes during object deletion, before destroying properties.
function text6_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to text6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
%
%   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
%
%
%

% Capacitance per unit length of coaxial cable

function C = capacitanceCoaxCable(EPS,a,b)

C = 2*pi*EPS/(log(b/a));

return
```

```
%
%   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
%
%
%

% Capacitance per unit length of microstrip

function C = capacitanceMicrostrip(EPS,w,h)

C = EPS*w/h;

return
```

```
%
%   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
%
%
%

% Capacitance of parallel plate capacitor

function C = capacitancePPCapacitor(EPS,S,d)

C = EPS*S/d;

return
```

```
%
%   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
%
%
%

% Capacitance of spherical capacitor

function C = capacitanceSphCapacitor(EPS,a,b)

C = 4*pi*EPS*a*b/(b-a);

return
```

```
%
%   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
%
%
%

% Capacitance of stripline

function C = capacitanceStripline(EPS,w,h)

C = 2*EPS*w/h;

return
```

MATLAB EXERCISE 2.14 **RG-55/U coaxial cable.** An RG-55/U coaxial cable has conductor radii $a = 0.5$ mm and $b = 3$ mm. The dielectric is polyethylene ($\varepsilon_r = 2.25$). Determine the capacitance per unit length of the cable using the capacitance calculator from the previous MATLAB exercise.

SOLUTION:

The capacitance per unit length of the RG-55/U coaxial cable amounts to $C' = 70$ pF/m.

MATLAB EXERCISE 2.15 **Parallel-plate capacitor model of a thundercloud.** A typical thundercloud can be approximately represented, as far as its electrical properties are concerned, as a parallel-plate capacitor with horizontal plates of area $S = 15$ km^2 and vertical separation $d = 1$ km. Neglecting the fringing effects, find the capacitance of this capacitor by the capacitance calculator from MATLAB Exercise 2.13.

SOLUTION:

The capacitance of the parallel-plate capacitor approximating a thundercloud, is computed to be $C = 132.8$ nF.

MATLAB EXERCISE 2.16 **Capacitance of a metallic cube, using MoM MATLAB code.** Find the capacitance of the metallic cube numerically analyzed by the method of moments in MATLAB Exercise 1.41, and compare the result with capacitances of the following metallic spheres, respectively: (a) the sphere inscribed in the cube, (b) the sphere overscribed about the cube, (c) the sphere whose radius is the arithmetic mean of the radii of spheres in (a) and (b), (d) the sphere having the same surface as the cube, and (e) the sphere with the same volume as the cube.

SOLUTION:

The results are provided in the TUTORIAL (in the book) for this MATLAB Exercise.

MATLAB EXERCISE 2.17 **Capacitance computation using FD MATLAB codes.** Compute the capacitance per unit length of the coaxial cable of square cross section numerically analyzed by a finite-difference technique in MATLAB Exercises 2.10 and 2.12, respectively.

SOLUTION:

The results are provided in the TUTORIAL for this MATLAB Exercise.

MATLAB EXERCISE 2.18 **Main MoM matrix for a parallel-plate capacitor.** Consider the parallel-plate capacitor shown in Fig.S2.9, and write a function `matrixACap()` in MATLAB that computes the matrix $[A]$ in Eq.(1.58) (from the book) for the method-of-moments analysis of the structure, assuming that the upper and lower plates are at potentials V and $-V$, respectively. *(matrixACap.m on IR)*

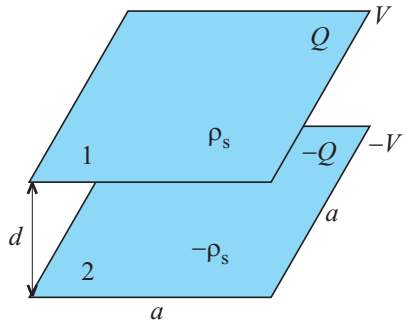

Figure S2.9 Air-filled parallel-plate capacitor with square plates; for MATLAB Exercise 2.18.

SOLUTION:

```
%
%   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
%
%
%


%    Main MoM matrix for a parallel-plate capacitor


function A = matrixACap (EPS,dS,x,y,d)
N = length(x);

if (length(dS)== N)
    for i = 1 : N
        for j = 1 : N
            r1 = sqrt ((x(j)-x(i))^2 + (y(j)-y(i))^2 );
            r2 = sqrt ((x(j)-x(i))^2 + (y(j)-y(i))^2 + d^2);
            if (i==j)
                A(i,j) = sqrt(dS(j))/(2*sqrt(pi)*EPS)- dS(j)/(4*pi*EPS*r2);
            else
                A(i,j) = dS(j)/(4*pi*EPS*r1) - dS(j)/(4*pi*EPS*r2);
            end;
        end;
    end;
else
    A = 0;
    disp ('Incorrect input data in function matrixACap');
end;
```

MATLAB EXERCISE 2.19 **MoM analysis of a parallel-plate capacitor in MATLAB.**
Write the main MATLAB program based on the method of moments to evaluate the capacitance (C) of the parallel-plate capacitor in Fig.2.9, using function `matrixACap`, developed in the previous MATLAB exercise. Assume that $a = 1$ m, $V = 1$ V ($V_1 = 1$ V and $V_2 = -1$ V), and $N = 100$ (each plate is subdivided into $N = 10 \times 10 = 100$ patches). By means of this MoM program, find C for the following d/a ratios: (i) 0.1, (ii) 0.5, (iii) 1, (iv) 2, and (v) 10. *(ME2_19.m on IR)*

SOLUTION:
The computed surface charge distribution over the upper plate is shown in Fig.S2.10.

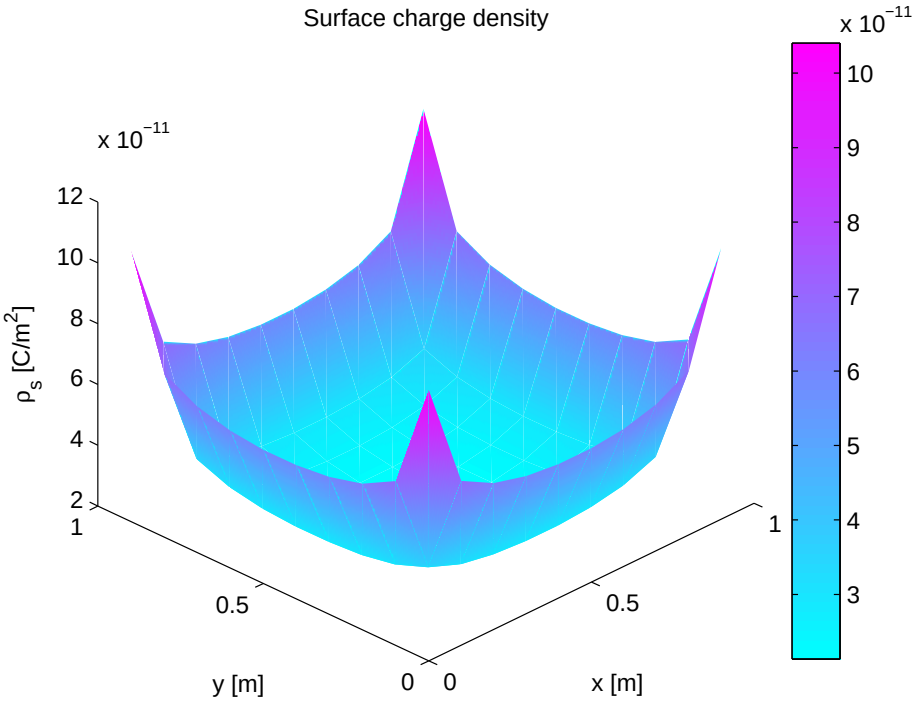

Figure S2.10 MATLAB display of the charge distribution of the upper plate of the parallel-plate capacitor in Fig.2.9 computed by a method-of-moments MATLAB code; for MATLAB Exercise 2.19.

For d/a ratios of 0.1, 0.5, 1, 2, and 10, C turns out to be 117 pF, 38.3 pF, 28.7 pF, 24 pF, and 20.6 pF, respectively. Note that the corresponding C values obtained from Eq.(2.28) (from the book), which neglects the fringing effects, are 88.5 pF, 17.7 pF, 8.85 pF, 4.43 pF, and 0.885 pF.

```
%
%   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
%
%
%


%    MoM analysis of a parallel-plate capacitor in MATLAB


% k = i) 0.1,(ii) 0.5, (iii) 1, (iv) 2, and (v) 10,


clear all;
close all;
V = 1;
N = 100;
n = sqrt(N);
a = 1;
k = [0.1 0.5 1 2 10]; % given ratio d/a
EPS0 = 8.8542*10^(-12);

[x,y,S] = localCoordinates(n,n,a,a);

C = zeros(1,length(k));
Ca = zeros(1,length(k));

for t = 1:length(k)
d = k(t)*a;
A = matrixACap(EPS0,S,x,y,d);
B = V*ones(N,1);
rhos = inv(A)*B;

for i = 1:n;
    rhos2D(i,:) = rhos((i-1)*n+1:i*n);
end;

X = 0.05:0.1:0.95;
Y = 0.05:0.1:0.95;

figure(1);
surf(X,Y,rhos2D);
title('Surface charge density');
xlabel('x [m]');
ylabel('y [m]');
zlabel('\rho_s [C/m^2]');
```

```
colorbar; colormap 'cool'; shading interp; grid off;

Qtot = totalCharge(S,rhos);
C(t) = Qtot/(2*V);

Ca(t) = EPS0*a^2/d;
end;

figure(2);
plot(k , C*10^12 ,'r','linewidth',2);
hold on;
plot(k, Ca*10^12,'b','linewidth',2);
hold off;
xlabel('ratio d/a');
ylabel('C [pF]');
legend('Numerical','Analytical');
```

MATLAB EXERCISE 2.20 **GUI for capacitors with inhomogeneous dielectrics.** Repeat MATLAB Exercise 2.13 but for the five types of capacitors with inhomogeneous dielectrics in Fig.2.13 (from the book). *[folder ME2_20(GUI) on IR]*

SOLUTION:

Figure S2.11 shows the GUI if, for instance, a two-wire line with dielectrically coated conductors is selected in the pop-up menu.

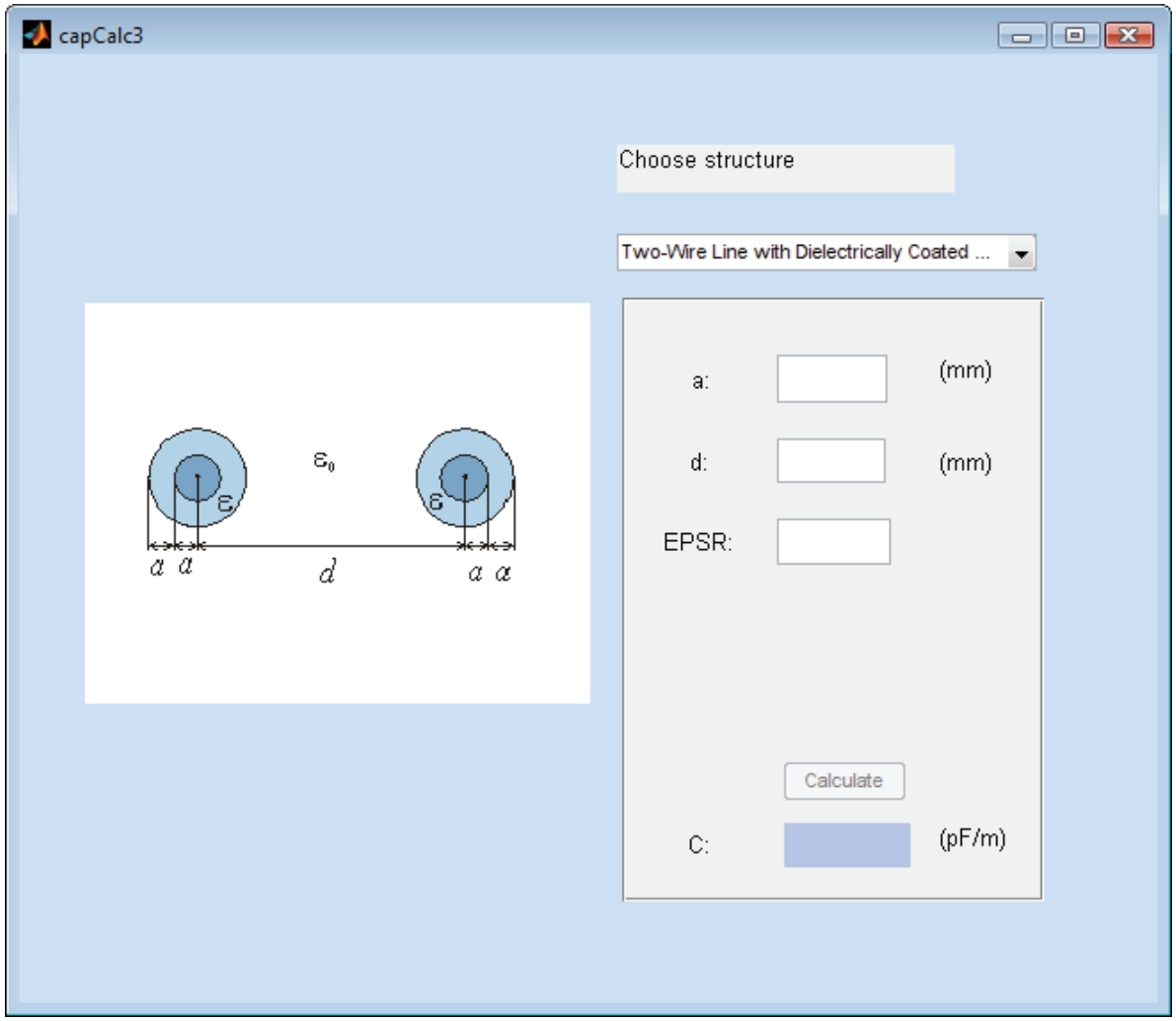

Figure S2.11 MATLAB capacitance calculator and graphical user interface for multiple structures with inhomogeneous dielectrics: GUI in the case a two-wire line with dielectrically coated conductors is selected in the pop-up menu; for MATLAB Exercise 2.20.

```
%
%   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
%
%
%


%   GUI for capacitors with inhomogeneous dielectrics


function varargout = capCalc3(varargin)
% CAPCALC3 M-file for capCalc3.fig
%      CAPCALC3, by itself, creates a new CAPCALC3 or raises the existing
%      singleton*.
%
%      H = CAPCALC3 returns the handle to a new CAPCALC3 or the handle to
%      the existing singleton*.
%
%      CAPCALC3('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CAPCALC3.M with the given input arguments.
%
%      CAPCALC3('Property','Value',...) creates a new CAPCALC3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before capCalc3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to capCalc3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help capCalc3

% Last Modified by GUIDE v2.5 03-Jun-2010 15:25:14

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @capCalc3_OpeningFcn, ...
                   'gui_OutputFcn',  @capCalc3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before capCalc3 is made visible.
function capCalc3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to capCalc3 (see VARARGIN)
% Import the figure of the current structure

handles.ppCapD = imread('ppCapD.png');
handles.ppCapE = imread('ppCapE.png');
handles.sphCapCon = imread('sphCapCon.png');
handles.sphCapHalf = imread('sphCapHalf.png');
handles.twlCoat = imread('twlCoat.png');
% Set the current data value.

set(handles.uipanel1,'Visible','off');
set(handles.axes1,'Visible','off');
% Choose default command line output for capCalc3
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

set(0,'units','inches');
screenSize = get(0,'ScreenSize');
set(hObject,'Units','inches','Position',[screenSize(3)/2-(6.6563/2),screenSize(4)/2-↙
(5.5417/2),6.6563,5.5417]);

% UIWAIT makes capCalc3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = capCalc3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global i;
% Set current data to the selected data set.
switch get(handles.popupmenu1,'Value')
case 1
   set(handles.uipanel1,'Visible','off');
   cla reset; axis off;
case 2
   handles.current_data = handles.ppCapD;
   imshow(handles.current_data);
   set(handles.edit4,'Visible','on');
   set(handles.edit5,'Visible','on');
   set(handles.text2,'String','d1:');
   set(handles.text3,'String','d2:');
   set(handles.text4,'String','S:');
   set(handles.text5,'Visible','on','String','EPSR1:');
   set(handles.text6,'Visible','on','String','EPSR2:');
   set(handles.text7,'String','(mm)');
   set(handles.text8,'String','(mm)');
   set(handles.text9,'Visible','on','String','(mm^2)');
   set(handles.text12,'String','(pF)');
   set(handles.uipanel1,'Visible','on');
   i = 1;
case 3
   handles.current_data = handles.ppCapE;
   imshow(handles.current_data);
   set(handles.edit4,'Visible','on');
   set(handles.edit5,'Visible','on');
   set(handles.text2,'String','S1:');
   set(handles.text3,'String','S2:');
   set(handles.text4,'String','d:');
   set(handles.text5,'Visible','on','String','EPSR1:');
   set(handles.text6,'Visible','on','String','EPSR2:');
   set(handles.text7,'String','(mm^2)');
   set(handles.text8,'String','(mm^2)');
   set(handles.text9,'Visible','on','String','(mm)');
   set(handles.text12,'String','(pF)');
   set(handles.uipanel1,'Visible','on');
   i = 2;
case  4
   handles.current_data = handles.sphCapCon;
   imshow(handles.current_data);
   set(handles.edit4,'Visible','on');
   set(handles.edit5,'Visible','on');
   set(handles.text2,'String','a:');
   set(handles.text3,'String','b:');
```

```
    set(handles.text4,'String','c:');
    set(handles.text5,'Visible','on','String','EPSR1:');
    set(handles.text6,'Visible','on','String','EPSR2:');
    set(handles.text7,'String','(mm)');
    set(handles.text8,'String','(mm)');
    set(handles.text9,'Visible','on','String','(mm)');
    set(handles.text12,'String','(pF)');
    set(handles.uipanel1,'Visible','on');
    i = 3;
case 5
    handles.current_data = handles.sphCapHalf;
    imshow(handles.current_data);
    set(handles.edit4,'Visible','on');
    set(handles.text5,'Visible','on');
    set(handles.text2,'String','a:');
    set(handles.text3,'String','b:');
    set(handles.text4,'String','EPSR1:');
    set(handles.text5,'String','EPSR2:');
    set(handles.text6,'Visible','off');
    set(handles.edit5,'Visible','off');
    set(handles.text7,'String','(mm)');
    set(handles.text8,'String','(mm)');
    set(handles.text9,'Visible','off');
    set(handles.text12,'String','(pF)');
    set(handles.uipanel1,'Visible','on');
    i = 4;
case 6
    handles.current_data = handles.twlCoat;
    imshow(handles.current_data);
    set(handles.text2,'String','a:');
    set(handles.text3,'String','d:');
    set(handles.text4,'String','EPSR:');
    set(handles.text5,'Visible','off');
    set(handles.text6,'Visible','off');
    set(handles.edit4,'Visible','off');
    set(handles.edit5,'Visible','off');
    set(handles.text7,'String','(mm)');
    set(handles.text8,'String','(mm)');
    set(handles.text9,'Visible','off');
    set(handles.text12,'String','(pF/m)');
    set(handles.uipanel1,'Visible','on');
    i = 5;

end
global ready;
global var;
 ready = [0 0 0 0 0];
 var = [0 0 0 0 0];
 set(handles.pushbutton1,'Enable','off');
 set(handles.edit1,'String','');
 set(handles.edit2,'String','');
 set(handles.edit3,'String','');
```

```
 set(handles.edit4,'String','');
 set(handles.edit5,'String','');
 set(handles.text11,'String','');

% Hints: contents = get(hObject,'String') returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
global ready;
 ready = [0 0 0 0 0];
global var;
 var = [0 0 0 0 0];



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double

% --- Executes during object creation, after setting all properties.
handles.edit1 = str2double(get(hObject,'String'));
 global ready;
 global var;
 global i;
 if i == 4;
     if (isnan(handles.edit1));
     msgbox('Invalid input','Error');
     ready(1)= 0;
else
    ready(1)= 1;
    var(1) = handles.edit1;
end;
if (ready == [1 1 1 1 0])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
```

```
end;
 else if i == 5;
         if (isnan(handles.edit1));
     msgbox('Invalid input','Error');
     ready(1)= 0;
else
    ready(1)= 1;
    var(1) = handles.edit1;
end;
if (ready == [1 1 1 0 0])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
     else
         if (isnan(handles.edit1));
     msgbox('Invalid input','Error');
     ready(1)= 0;
else
    ready(1)= 1;
    var(1) = handles.edit1;
end;
if (ready == [1 1 1 1 1])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
     end;
 end




% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double

% --- Executes during object creation, after setting all properties.
handles.edit2 = str2double(get(hObject,'String'));
 global ready;
 global var;
 global i;
 if i == 4;
     if (isnan(handles.edit2));
     msgbox('Invalid input','Error');
     ready(2)= 0;
else
    ready(2)= 1;
    var(2) = handles.edit2;
end;
if (ready == [1 1 1 1 0])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
 else if i == 5;
         if (isnan(handles.edit2));
     msgbox('Invalid input','Error');
     ready(2)= 0;
else
    ready(2)= 1;
    var(2) = handles.edit2;
end;
if (ready == [1 1 1 0 0])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
     else
         if (isnan(handles.edit2));
     msgbox('Invalid input','Error');
     ready(2)= 0;
else
    ready(2)= 1;
    var(2) = handles.edit2;
end;
if (ready == [1 1 1 1 1])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
     end;
 end

% --- Executes during object creation, after setting all properties.
```

```
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
% --- Executes during object creation, after setting all properties.
handles.edit3 = str2double(get(hObject,'String'));
 global ready;
 global var;
 global i;
 if i == 4;
     if (isnan(handles.edit3));
     msgbox('Invalid input','Error');
     ready(3)= 0;
else
    ready(3)= 1;
    var(3) = handles.edit3;
end;
if (ready == [1 1 1 1 0])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
 else if i == 5;
         if (isnan(handles.edit3));
     msgbox('Invalid input','Error');
     ready(3)= 0;
else
    ready(3)= 1;
    var(3) = handles.edit3;
end;
if (ready == [1 1 1 0 0])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
```

```
     else
         if (isnan(handles.edit3));
     msgbox('Invalid input','Error');
     ready(3)= 0;
else
    ready(3)= 1;
    var(3) = handles.edit3;
end;
if (ready == [1 1 1 1 1])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
     end;
 end


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double

% --- Executes during object creation, after setting all properties.
handles.edit4 = str2double(get(hObject,'String'));
 global ready;
 global var;
 global i;
 if i == 4;
     if (isnan(handles.edit4));
     msgbox('Invalid input','Error');
     ready(4)= 0;
else
    ready(4)= 1;
    var(4) = handles.edit4;
```

```
end;
if (ready == [1 1 1 1 0])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
 else
         if (isnan(handles.edit4));
     msgbox('Invalid input','Error');
     ready(4)= 0;
else
    ready(4)= 1;
    var(4) = handles.edit4;
end;
if (ready == [1 1 1 1 1])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;
 end

% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double

% --- Executes during object creation, after setting all properties.
handles.edit5 = str2double(get(hObject,'String'));
 global ready;
 global var;

 if (isnan(handles.edit5));
     msgbox('Invalid input','Error');
     ready(5)= 0;
```

```
else
    ready(5)= 1;
    var(5) = handles.edit5;
end;
if (ready == [1 1 1 1 1])
     set(handles.pushbutton1,'Enable','on');
 else
    set(handles.pushbutton1,'Enable','off');
end;

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get↙
(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes during object creation, after setting all properties.
global var;
global i;
EPS0 = 8.8542*10^(-12);
mm2m = 10^(-3);
mmsq2msq = 10^(-6);
if  i == 1;
    d1 = var(1)*mm2m;
    d2 = var(2)*mm2m;
    S = var(3)*mmsq2msq;
    EPSR1 = var(4);
    EPSR2 = var(5);
    C = EPS0*EPSR1*EPSR2*S/(EPSR2*d1 + EPSR1*d2);
else if  i == 2;
        S1 = var(1)*mmsq2msq;
        S2 = var(2)*mmsq2msq;
        d = var(3)*mm2m;
        EPSR1 = var(4);
        EPSR2 = var(5);
        C = EPS0*(EPSR1*S1 + EPSR2*S2)/d;
    else if i == 3;
            a = var(1)*mm2m;
            b = var(2)*mm2m;
```

```
            c = var(3)*mm2m;
            EPSR1 = var(4);
            EPSR2 = var(5);
            C = 4*pi*EPS0*((b-a)/(EPSR1*a*b) + (c-b)/(EPSR2*b*c))^(-1);
        else if  i == 4;
                a = var(1)*mm2m;
                b = var(2)*mm2m;
                EPSR1 = var(3);
                EPSR2 = var(4);
                C = 2*pi*EPS0*(EPSR1 + EPSR2)*a*b/(b-a);
            else
                a = var(1)*mm2m;
                d = var(2)*mm2m;
                EPSR = var(3);
                C = pi*(log(2)/(EPSR*EPS0) + log(d/(2*a))/EPS0)^(-1);
            end;
        end
    end
end
C = C*10^12;
set(handles.text11,'String',num2str(C,'%.4e'));
```

MATLAB EXERCISE 2.21 **Breakdown in a spherical capacitor with a multilayer dielectric.** Consider a spherical capacitor with N concentric dielectric layers. The inner radius, relative permittivity, and the dielectric strength of the ith layer are a_i, ε_{ri}, and E_{cri}, respectively ($i = 1, 2, \ldots, N$). The inner radius of the outer conductor of the capacitor is b. Write a MATLAB program to find which dielectric layer would break down first after a voltage of critical value is applied across the capacitor electrodes and to compute the breakdown voltage of the capacitor. Test the program for $N = 3$, $a_1 = 2.5$ cm, $a_2 = 5$ cm, $a_3 = 7$ cm, and $b = 9$ cm, if the dielectrics constituting layers 1, 2, and 3 are polystyrene, quartz, and silicon, respectively (use GUI's from MATLAB Exercises 2.1 and 2.3 to get the values of material parameters). *(ME2_21.m on IR)*

SOLUTION:

For the test example ($N = 3$, $a_1 = 2.5$ cm, $a_2 = 5$ cm, $a_3 = 7$ cm, and $b = 9$ cm), the material parameters for layers 1, 2, and 3 are, respectively: polystyrene ($\varepsilon_r = 2.56$, $E_{cr} = 20$ MV/m), quartz ($\varepsilon_r = 5$, $E_{cr} = 1000$ MV/m), and silicon ($\varepsilon_r = 11.9$, $E_{cr} = 30$ MV/m). The breakdown occurs in layer 1. The critical value of the capacitor charge for breakdown amounts to $Q_{cr} = 3.5605$ μC and the breakdown voltage of the capacitor is $V_{cr} = 295.108$ kV.

```
%
%   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
%
%
%

% This program calculates breakdown voltage in spherical capacitor with
% arbitrary number of concentric dielectric layers.

clear all;
close all;
EPS0 = 8.8542*10^(-12);

% Number of dielectric layers
N = input('Enter the number of concentric dielectric layers: ');

% Dielectric permittivity constant, inner radius of dielectric layer and
% dielectric strength for each dielectric layer.
for i = 1:N
    EPSR(i) = input(['Enter dielectric rel. permittivity in ',int2str(i),'. layer: ']);
    a(i) = input(['Enter inner radius (in mm) of ',int2str(i),'. layer: ']);
    a(i) = a(i)/1000;
    Ecr(i) = input(['Enter dielectric strength (in MV/m) for ',int2str(i),'. layer: ']);
    Ecr(i) = Ecr(i)*10^6;
end;
b = input(['Enter outer radius (in mm) of ',int2str(N),'. layer: ']);
b = b/1000;

% Maximum of line charge before breakdown - calculation for each layer
Q = 4*pi*EPS0.*EPSR.*Ecr.*a.^2;
% In which layer will happen breakdown - minimum of the line charge
[Qmin,index] = min(Q);
fprintf(['Breakdown will happen in ',int2str(index),'. dielectric layer for Q = ']);
fprintf('%f nC/m.\n',Qmin*10^9);

% Breakdown voltage
Vcr = 0;   % Initialization
if N>1
    for i=1:N-1  % Integral E*dl in the first N-1 layers
        Vcr = Vcr + Qmin/4/pi/EPS0/EPSR(i)*(a(i+1)-a(i))/a(i+1)/a(i);
    end
end
% Addition of integral in the last layer
Vcr = Vcr + Qmin/4/pi/EPS0/EPSR(N)*(b-a(N))/b/a(N);
fprintf('Breakdown voltage is: %f V.\n',Vcr);
```

MATLAB EXERCISE 2.22 **Breakdown in a coaxial cable with a multilayer dielectric.** Repeat the previous MATLAB exercise but for a coaxial cable with N coaxial dielectric layers. *(ME2_22.m on IR)*

SOLUTION:

Similarly as in the previous MATLAB exercise, we test the program for $N = 3$, $a_1 = 2.5$ cm, $a_2 = 5$ cm, $a_3 = 7$ cm, and $b = 9$ cm, and the material parameters for coaxial layers 1, 2, and 3 given by: polystyrene ($\varepsilon_\mathrm{r} = 2.56$, $E_\mathrm{cr} = 20$ MV/m), quartz ($\varepsilon_\mathrm{r} = 5$, $E_\mathrm{cr} = 1000$ MV/m), and silicon ($\varepsilon_\mathrm{r} = 11.9$, $E_\mathrm{cr} = 30$ MV/m). The breakdown occurs in layer 1. The critical value of the charge per unit length of the cable for breakdown amounts to $Q'_\mathrm{cr} = 71.21$ μC/m and the breakdown voltage of the cable is $V_\mathrm{cr} = 459.742$ kV.

```
%
%   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
%
%
%

% This program calculates breakdown voltage in coaxial cable with arbitrary
% number of coaxial dielectric layers.

clear all;
close all;
EPS0 = 8.8542*10^(-12);

% Number of dielectric layers
N = input('Enter the number of coaxial dielectric layers: ');

% Dielectric permittivity constant, smaller radius of dielectric layer and
% dielectric strength for each dielectric layer.
for i = 1:N
    EPSR(i) = input(['Enter dielectric rel. permittivity in ',int2str(i),'. layer: ']);
    a(i) = input(['Enter inner radius (in mm) of ',int2str(i),'. layer: ']);
    a(i) = a(i)/1000;
    Ecr(i) = input(['Enter dielectric strength (in MV/m) for ',int2str(i),'. layer: ']);
    Ecr(i) = Ecr(i)*10^6;
end;
b = input(['Enter outer radius (in mm) of ',int2str(N),'. layer: ']);
b = b/1000;
% Maximum of line charge before breakdown - calculation for each layer
Qprim = 2*pi*EPS0.*EPSR.*a.*Ecr;
% In which layer will happen breakdown - minimum of the line charge
[Qprimmin,index] = min(Qprim);
fprintf(['Breakdown will happen in ',int2str(index),'. dielectric layer for Q` = ']);
fprintf('%f nC/m.\n',Qprimmin*10^9);

% Breakdown voltage
Vcr = 0;   % Initialization
if N>1
    for i=1:N-1  % Integral E*dl in the first N-1 layers
        Vcr = Vcr + Qprimmin/2/pi/EPS0/EPSR(i)*log(a(i+1)/a(i));
    end
end
% Addition of integral in the last layer
Vcr = Vcr + Qprimmin/2/pi/EPS0/EPSR(N)*log(b/a(N));
fprintf('Breakdown voltage is: %f V.\n',Vcr);
```

MATLAB EXERCISE 2.23 **Parallel-plate capacitor with multiple layers.** Repeat MATLAB Exercise 2.21 but for a parallel-plate capacitor with N dielectric layers placed like in Fig.2.13(a) (from the book). The thicknesses of layers are d_i $(i = 1, 2, \ldots, N)$ and fringing effects can be neglected. *(ME2_23.m on IR)*

SOLUTION:

We test the program for $N = 3$, thicknesses of layers 1, 2, and 3 amounting to $d_1 = 1$ cm, $d_2 = 2$ cm, and $d_3 = 3$ cm, and the material parameters for the layers given by polystyrene ($\varepsilon_{\rm r} = 2.56$, $E_{\rm cr} = 20$ MV/m), quartz ($\varepsilon_{\rm r} = 5$, $E_{\rm cr} = 1000$ MV/m), and silicon ($\varepsilon_{\rm r} = 11.9$, $E_{\rm cr} = 30$ MV/m). The breakdown occurs in layer 1. The critical value of the electric flux density vector for breakdown is $D_{\rm cr} = 453.335$ μC/m^2 and the breakdown voltage of the capacitor is $V_{\rm cr} = 533.875$ kV.

```
%
%   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
%
%
%

% Parallel-plate capacitor with N dielectric layers.
% This program calculates breakdown voltage in parallel plate capacitor
% with arbitrary number of dielectric layers parallel to the capacitor
% plates

clear all;
close all;
EPS0 = 8.8542*10^(-12);

% Number of dielectric layers
N = input('Enter the number of layers parallel to the plates of capacitor: ');

% Dielectric permittivity constant and dielectric strength for each
% dielectric layer
for i = 1:N
    EPSR(i) = input(['Enter dielectric rel. permittivity in ',int2str(i),'. layer: ']);
    Ecr(i) = input(['Enter dielectric strength (in MV/m) for ',int2str(i),'. layer: ']);
    Ecr(i) = Ecr(i)*10^6;
    d(i) = input(['Enter the width (in mm) for ',int2str(i),'. layer: ']);
    d(i) = d(i)/1000;
end;

% Maximum of electric flux density vector in each layer - before breakdown
D = EPS0.*EPSR.*Ecr;

% In which layer will happen breakdown
[Dmin,index] = min(D);
fprintf(['Breakdown will happen in ',int2str(index),'. dielectric layer for D = ']);
fprintf('%f uC/m^2.\n',Dmin*10^6);

% Electric field intesity vector in each dielectric
E = Ecr(index)*EPSR(index)./EPSR;
% Breakdown voltage
Vcr = dot(E,d);
fprintf('Breakdown voltage is: %f V.\n',Vcr)
```

MATLAB EXERCISE 2.24 **Parallel-plate capacitor with multiple sectors.** Repeat the previous MATLAB exercise but for a parallel-plate capacitor with N dielectric sectors placed like in Fig.2.13(b) (from the book). *(ME2_24.m on IR)*

SOLUTION:

We test the program for $N = 3$, the separation between the plates $d = 1$ cm, and the material parameters for the dielectric sectors 1, 2, and 3 given by polystyrene ($\varepsilon_r = 2.56$, $E_{cr} = 20$ MV/m), quartz ($\varepsilon_r = 5$, $E_{cr} = 1000$ MV/m), and silicon ($\varepsilon_r = 11.9$, $E_{cr} = 30$ MV/m). The breakdown occurs in dielectric sector 1. The breakdown voltage of the capacitor is $V_{cr} = 200$ kV.

```
%
%   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
%
%
%

% Parallel-plate capacitor with N dielectric sectors.
% This program calculates breakdown voltage in parallel plate capacitor
% with arbitrary number of dielectric sectors normal to the capacitor
% plates.

clear all;
close all;

% Number of dielectric sectors
N = input('Enter the number of dielectric sectors normal to capacitor plates: ');

% Dielectric strength for each dielectric sector
for i = 1:N
    Ecr(i) = input(['Enter dielectric strength (in MV/m) for ',int2str(i),'. sector: ']);
    Ecr(i) = Ecr(i)*10^6;
end;
% The plate separation
d = input('Enter the plate separation (in mm) of the capacitor: ');
d = d/1000;

% Maximum of electric field intensity vector in each sector - before breakdown
[Emin,index] = min(Ecr);
fprintf(['Breakdown occurs in ',int2str(index),'. dielectric sector']);
% Breakdown voltage
Vcr = Emin*d;
fprintf('Breakdown voltage is: %f V.\n',Vcr);
```

MATLAB®-Based Electromagnetics

Instructor's Solutions Manual

for Chapter 2 Electrostatic Field in Dielectrics

Branislav M. Notaroš

Department of Electrical and Computer Engineering
Colorado State University

This *Instructor's Solutions Manual* contains multiple PDF files provided in 12 folders for 12 chapters of the book, *MATLAB®-Based Electromagnetics*, and is available for download on Instructor Resources for the book.

The author and publisher of this *Manual* have used their best efforts in preparing this Manual. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this *Manual*. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

The material on Instructor Resources for this book, *MATLAB®-Based Electromagnetics*, including this *Instructor's Solutions Manual*, is meant *only for instructors adopting this book*.

Any of the provided m files on Instructor Resources and any of the included MATLAB codes or any part of a code may be used only for educational purposes associated with the book, MATLAB®-Based Electromagnetics.