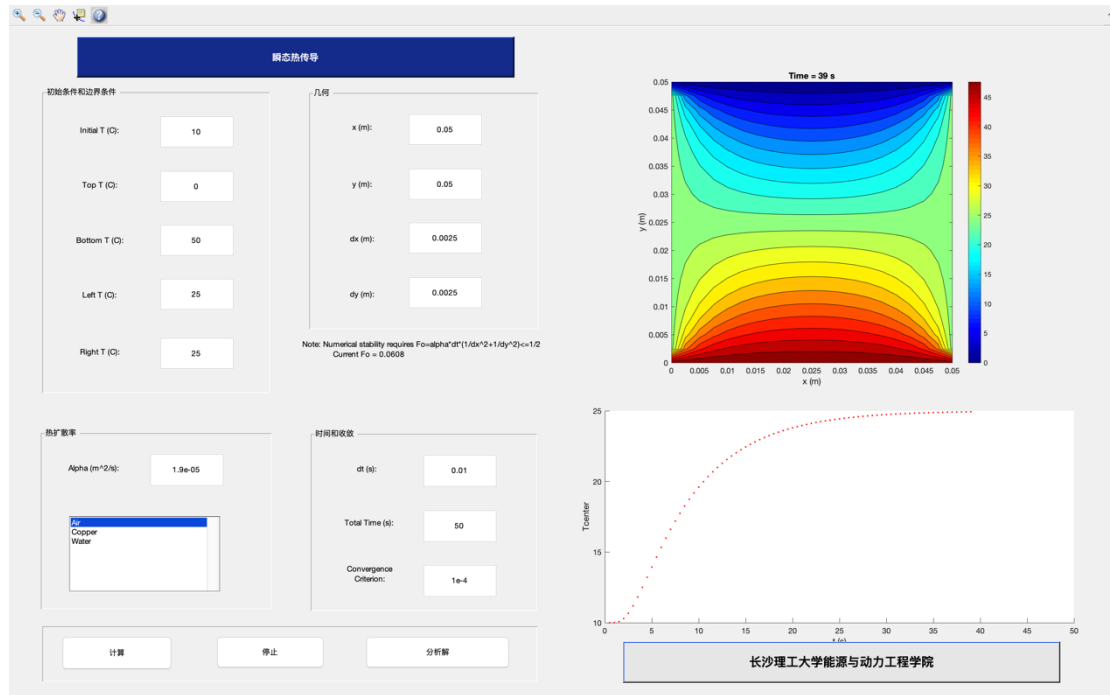


高等传热学教学 APP

目录

- 一、瞬态热传导教学 APP
- 二、边界层流动教学 APP
- 三、辐射角系数计算教学 APP
- 四、竖直平壁自然对流教学 APP
- 五、管道保温计算教学 APP
- 六、螺旋式静态混合器教学 APP
- 七、管状反应器教学 APP
- 八、同心圆管换热器教学 APP
- 九、换热片强制冷却教学 APP
- 十、无粘性通道流教学 APP

一、瞬态热传导教学 APP



部分程序代码：

```
function varargout = transientConductionGUI(varargin)
%
%
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',   gui_Singleton, ...
    'gui_OpeningFcn', @transientConductionGUI_OpeningFcn, ...
    'gui_OutputFcn',  @transientConductionGUI_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 transientConductionGUI is made visible.
```

```

function transientConductionGUI_OpeningFcn(hObject, eventdata, handles, varargin)

% Create the help button on toolbar
[X, map] = imread(fullfile(...
    matlabroot,'toolbox','matlab','icons','csh_icon.gif'));
icon = ind2rgb(X,map);
uipushtool(handles.uitoolbar1,'CData',icon,...
    'TooltipString','Help',...
    'ClickedCallback',@AppHelp);

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

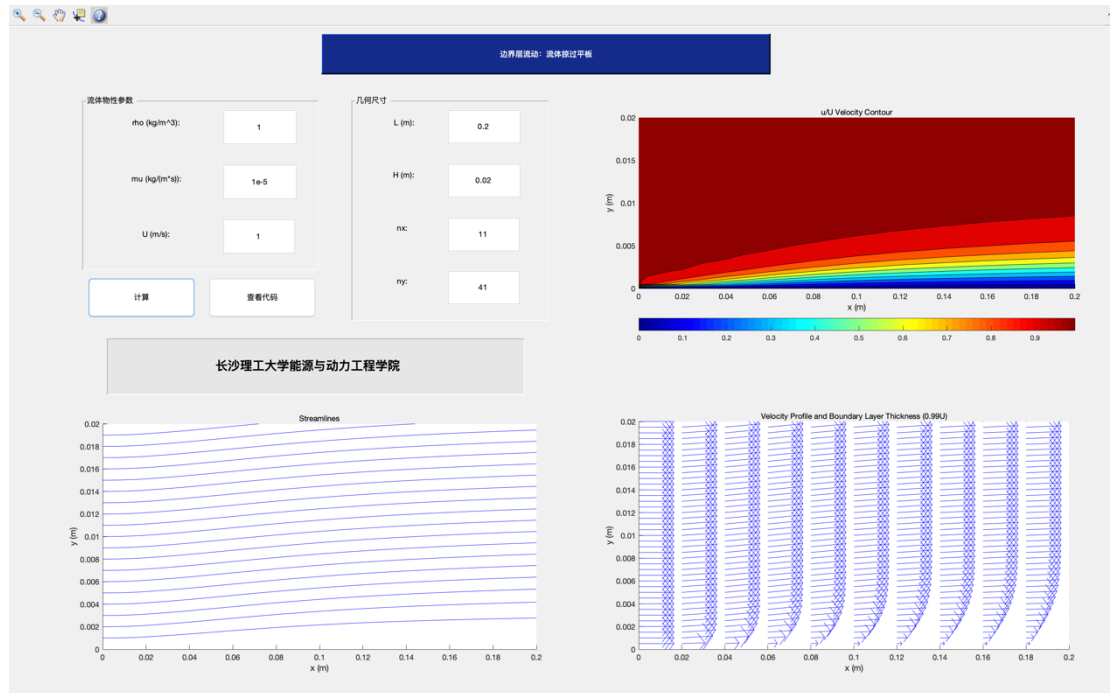
% Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = transientConductionGUI_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function handles = getpara(handles)
% Obtain the input to show the current Fo Number
handles.data.T_int = str2double(get(handles.T_int,'String'));
handles.data.T_top = str2double(get(handles.T_top,'String'));
handles.data.T_btm = str2double(get(handles.T_btm,'String'));
handles.data.T_lft = str2double(get(handles.T_lft,'String'));
handles.data.T_rht = str2double(get(handles.T_rht,'String'));
handles.data.L = str2double(get(handles.L,'String'));
handles.data.H = str2double(get(handles.H,'String'));
handles.data.dx = str2double(get(handles.dx,'String'));
handles.data.dy = str2double(get(handles.dy,'String'));
handles.data.tmax = str2double(get(handles.tmax,'String'));
handles.data.dt = str2double(get(handles.dt,'String'));
handles.data.epsilon = str2double(get(handles.epsilon,'String'));
% Preset the thermal diffusivity of the medium chosen from list
listStrings = get(handles.listalp,'String');
domaintype = listStrings{get(handles.listalp,'Value')};
switch domaintype
    case 'Air'
        domainalp = 1.9e-5;
    case 'Water'
        domainalp = 1.43e-7;
    case 'Copper'
        domainalp = 1.11e-4;

```

二、边界层流动教学 APP



部分程序代码：

```
function varargout = boundaryLayerGUI(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @boundaryLayerGUI_OpeningFcn, ...
    'gui_OutputFcn',  @boundaryLayerGUI_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 boundaryLayerGUI is made visible.
```

```

function boundaryLayerGUI_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 boundaryLayerGUI (see VARARGIN)

% Create the help button on toolbar
[X, map] = imread(fullfile(
    matlabroot,'toolbox','matlab','icons','csh_icon.gif'));
icon = ind2rgb(X,map);
uipushtool(handles.uitoolbar1,'CData',icon,...
    'TooltipString','Help',...
    'ClickedCallback',@AppHelp);

% Set up the label and range of axes
H = str2double(get(handles.H,'String'));
L = str2double(get(handles.L,'String'));
set(handles.contour,'XLim',[0 L],'YLim',[0 H])
title(handles.contour,'u/U Velocity Contour')
xlabel(handles.contour,'x (m)'),ylabel(handles.contour,'y (m)')
set(handles.vector,'XLim',[0 L],'YLim',[0 H])
title(handles.vector,'Velocity Profile and Boundary Layer Thickness (0.99U)')
xlabel(handles.vector,'x (m)'),ylabel(handles.vector,'y (m)')
set(handles.strline,'XLim',[0 L],'YLim',[0 H])
title(handles.strline,'Streamlines')
xlabel(handles.strline,'x (m)'),ylabel(handles.strline,'y (m)')
annotation(gcf,'textbox',get(handles.Re,'Position'),...
    'String',{'      Re_x = \rho U x / \mu      \delta_x \sim 5x / {\surd} Re_x'},...
    'LineStyle','none')

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

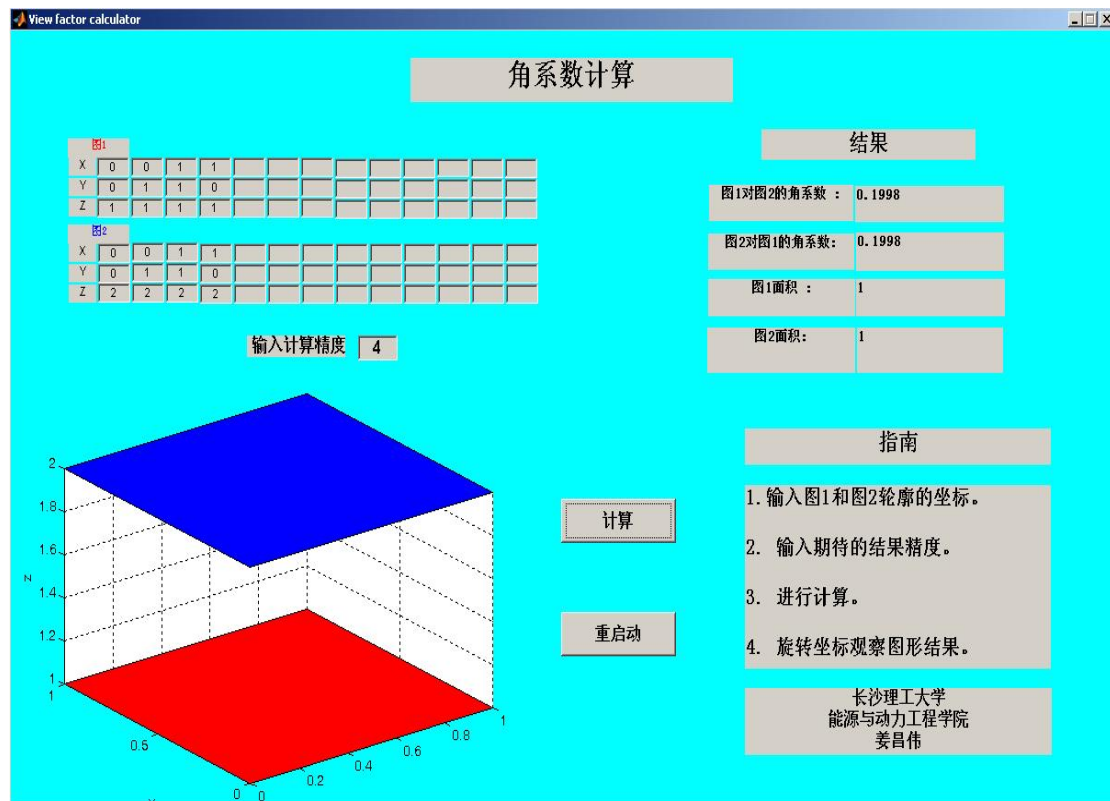
% Update handles structure
guidata(hObject, handles);

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

% --- Outputs from this function are returned to the command line.
function varargout = boundaryLayerGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure

```

三、辐射角系数计算教学 APP



部分程序代码：

```
function varargout = vfact(varargin)
```

```
%Procedure to launch the graphical user interface
```

```
if nargin == 0 % LAUNCH GUI
```

```
    fig = openfig(mfilename,'reuse');
```

```
    % Generate a structure of handles to pass to callbacks, and store it.
```

```
    handles = guihandles(fig);
```

```
    guidata(fig, handles);
```

```
    if nargin > 0
```

```
        varargout{1} = fig;
```

```
    end
```

```
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
```

```
    try
```

```
        [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
```

```
    catch
```

```

        disp(lasterr);
    end

end

% -----
function varargout = Calcul_Callback(h, eventdata, handles, varargin)
%Function to execute when the user press the button "Calculate"

%Labels show 'Wait a moment please' when the computer calculates.
set(handles.resultats,'string','Wait a moment please');
set(handles.resultats2,'string','Wait a moment please');
set(handles.aire1,'string','Wait a moment please');
set(handles.aire2,'string','Wait a moment please');

%Look for the coordinates of figure 1
figure1(1,1)=str2num(get(handles.figure1X1,'string'));
figure1(1,2)=str2num(get(handles.figure1Y1,'string'));
figure1(1,3)=str2num(get(handles.figure1Z1,'string'));

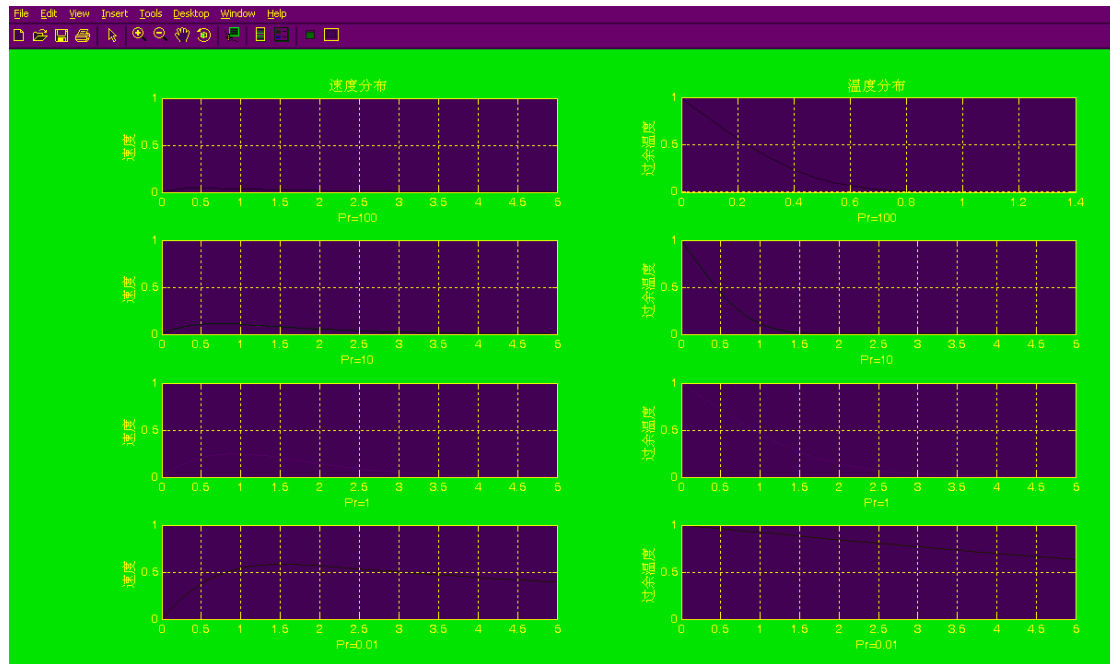
figure1(2,1)=str2num(get(handles.figure1X2,'string'));
figure1(2,2)=str2num(get(handles.figure1Y2,'string'));
figure1(2,3)=str2num(get(handles.figure1Z2,'string'));

figure1(3,1)=str2num(get(handles.figure1X3,'string'));
figure1(3,2)=str2num(get(handles.figure1Y3,'string'));
figure1(3,3)=str2num(get(handles.figure1Z3,'string'));

%When we reach the fourth point, we look if the box is empty or filled before we continue.
if isempty(get(handles.figure1X4,'string'))==0 & isempty(get(handles.figure1Y4,'string'))==0 &
isempty(get(handles.figure1Z4,'string'))==0
figure1(4,1)=str2num(get(handles.figure1X4,'string'));
figure1(4,2)=str2num(get(handles.figure1Y4,'string'));
figure1(4,3)=str2num(get(handles.figure1Z4,'string'));
end
if isempty(get(handles.figure1X5,'string'))==0 & isempty(get(handles.figure1Y5,'string'))==0 &
isempty(get(handles.figure1Z5,'string'))==0
figure1(5,1)=str2num(get(handles.figure1X5,'string'));
figure1(5,2)=str2num(get(handles.figure1Y5,'string'));
figure1(5,3)=str2num(get(handles.figure1Z5,'string'));
end
if isempty(get(handles.figure1X6,'string'))==0 & isempty(get(handles.figure1Y6,'string'))==0 &
isempty(get(handles.figure1Z6,'string'))==0

```

四、竖直平壁自然对流教学 APP



部分程序代码：

```
global lamda n nu0 nuI rend

% Parameters for Carreau Fluid

title1 = 'Parameters of Carreau Fluid';

ind1 = {'n (0.5)', 'lamda (0.2)', 'nu0 (1.72775e-3)', 'nuI (0)'};

setpar = {'0.5', '0.2', '1.72775e-3', '0'};

options.resize='on';

prompt = ind1;
def = setpar;
dlgTitle = title1;
lineNo = 1;

Mz = inputdlg(prompt, dlgTitle, lineNo, def, options);

if isempty(Mz)==0
    PAR = str2num(char(Mz));
else
    PAR = [1e-6, 2];
end
```

```

disp('Parameters of Carreau Fluid:')
disp(' ')
disp(PAR);

n=PAR(1);
lamda=PAR(2);
nu0=PAR(3);
nuI=PAR(4);

% pipe radius is 0.02m

rend=0.02;

% finding the velocity at the center of the pipe (we use the shooting
% method)

vo=fsolve(@zero,0.2);

X0f=[0,vo,nu0];

% Solving the governing equations to get the velocity profile

opts = odeset('Mass','M','MassSingular','yes');

[r,X]=ode15s('goveq',[1e-5 rend],X0f,opts);

% plotting the shear stress and the velocity in the pipe

figure(1);
plot(r,X(:,2),'b');
axis tight
xlabel('radial position')
ylabel('velocity')

figure(2);
plot(r,X(:,1),'r');
axis tight
xlabel('radial position')
ylabel('shear stress')
function xdot=goveq(r,x,flag)

global lamda n nu0 nuI rend

if isempty(flag)

```

五、管道保温计算教学 APP

部分程序代码：

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 0;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @heat_conduction_OpeningFcn, ...
                  'gui_OutputFcn',  @heat_conduction_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 heat_conduction is made visible.
function heat_conduction_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 heat_conduction (see VARARGIN)

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

% Update handles structure
guidata(hObject, handles);

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

% --- Outputs from this function are returned to the command line.
function varargout = heat_conduction_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;

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

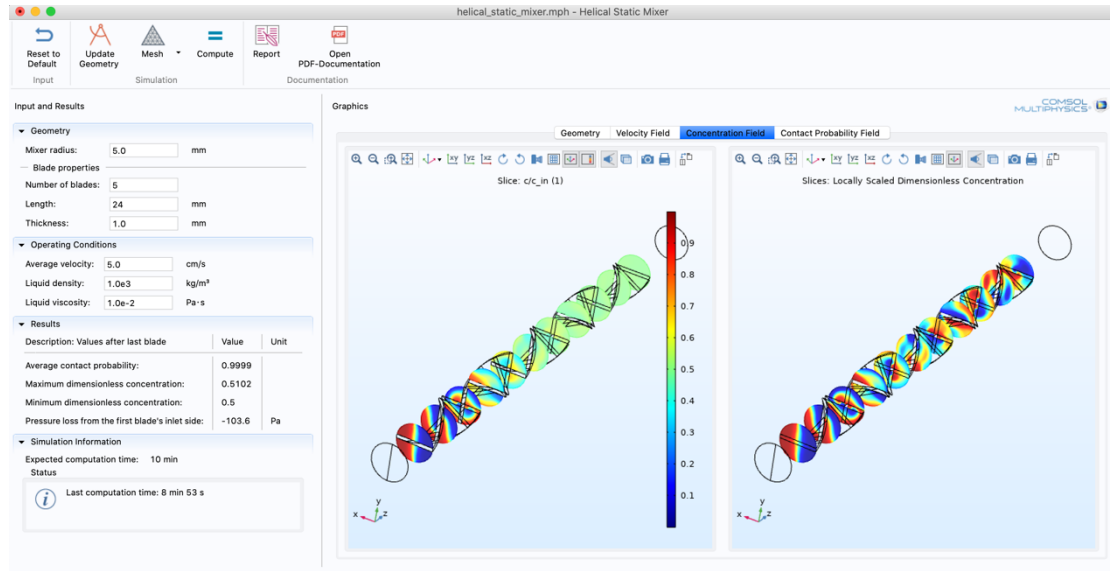
% Hints: get(hObject,'String') returns contents of lambda as text
%         str2double(get(hObject,'String')) returns contents of lambda as a double
lambda = str2double(get(hObject, 'String'));
if isnan(lambda)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end

% --- Executes during object creation, after setting all properties.
function lambda_CreateFcn(hObject, eventdata, handles)
% hObject      handle to lambda (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'))

```

六、螺旋式静态混合器教学 APP



部分程序代码：

```
import com.comsol.model.*
import com.comsol.model.util.*

model = ModelUtil.create('Model');

model.component.create('comp1', true);

model.component('comp1').geom.create('geom1', 3);

model.component('comp1').mesh.create('mesh1');

model.component('comp1').physics.create('spf', 'LaminarFlow', 'geom1');
model.component('comp1').physics.create('tds', 'DilutedSpecies', {'c'});

model.study.create('std1');
model.study('std1').setGenConv(true);
model.study('std1').create('stat', 'Stationary');
model.study('std1').feature('stat').activate('spf', true);
model.study('std1').feature('stat').activate('tds', true);

model.setExpectedComputationTime('10 min');

model.param.set('th_bl_i', '1.0[mm]', 'Blade thickness');
model.param.set('l_bl_i', '24[mm]', 'Blade length');
model.param.set('r_mx_i', '5.0[mm]', 'Mixer radius');
model.param.set('n_bl', '5', 'Number of blades');
```

```

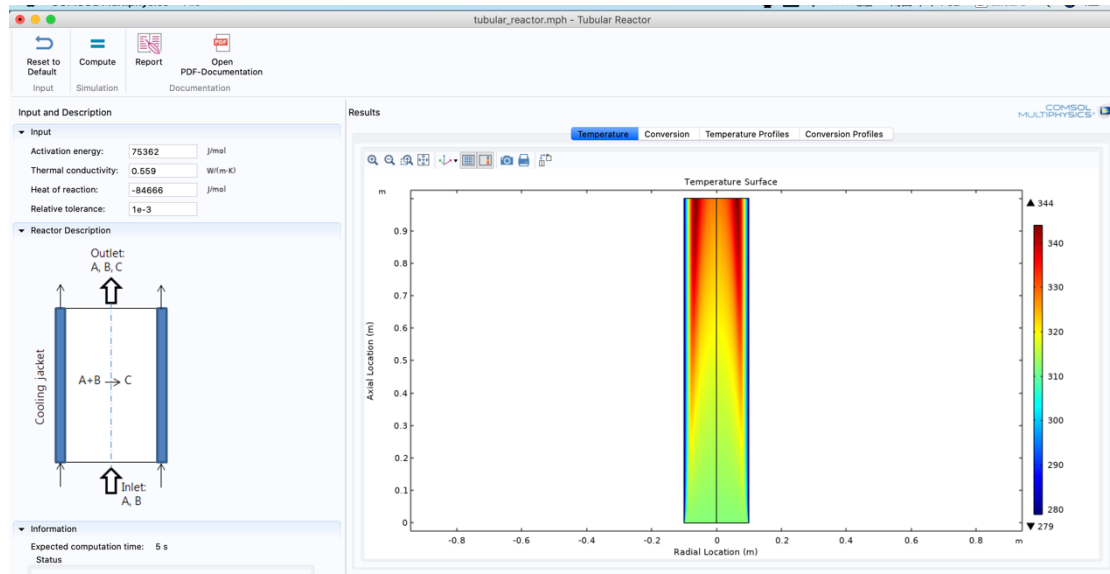
model.param.set('l_mx', 'n_bl*l_bl_i+1.5*l_bl_i', 'Mixer length');
model.param.set('il_mx', '0.5*l_bl_i', 'Mixer inlet length');
model.param.set('rho_l', '1.0e3[kg/m^3]', 'Liquid density');
model.param.set('visc_l', '1.0e-2[Pa*s]', 'Liquid viscosity');
model.param.set('u_av', '5.0[cm/s]', 'Average velocity');
model.param.set('c_in', '1.0[mole/m^3]', 'Inlet concentration');
model.param.set('D1', '1.0e-10[m^2/s]', 'Diffusivity');

model.func.create('step1', 'Step');
model.func('step1').set('smooth', 'r_mx_i/100');

model.geom.create('part1', 'Part', 3);
model.geom('part1').label('Blade');
model.geom('part1').inputParam.set('th_bl', '1[mm]');
model.geom('part1').inputParam.descr('th_bl', 'Blade thickness');
model.geom('part1').inputParam.set('l_bl', '24[mm]');
model.geom('part1').inputParam.descr('l_bl', 'Blade length');
model.geom('part1').inputParam.set('r_mx', '5[mm]');
model.geom('part1').inputParam.descr('r_mx', 'Mixer radius');
model.geom('part1').inputParam.set('nr_bl', '1');
model.geom('part1').inputParam.descr('nr_bl', 'Blade number (Note: not number of blades!)');
model.geom('part1').repairTolType('relative');
model.geom('part1').localParam.set('w_bl', '2.5*r_mx');
model.geom('part1').localParam.set('sl_bl', 'l_bl/6');
model.geom('part1').localParam.set('an_bl', '30[deg]');
model.geom('part1').feature.create('wp1', 'WorkPlane');
model.geom('part1').feature('wp1').set('unite', true);
model.geom('part1').feature('wp1').geom.create('r1', 'Rectangle');
model.geom('part1').feature('wp1').geom.feature('r1').set('size', {'w_bl' 'th_bl'});
model.geom('part1').feature('wp1').geom.feature('r1').set('base', 'center');
model.geom('part1').feature('wp1').geom.run('r1');
model.geom('part1').run('wp1');
model.geom('part1').feature.create('ext1', 'Extrude');
model.geom('part1').feature('ext1').setIndex('distance', 'sl_bl', 0);
model.geom('part1').feature('ext1').set('crossfaces', false);
model.geom('part1').feature('ext1').setIndex('twist', 'an_bl', 0);
model.geom('part1').run('ext1');
model.geom('part1').run('ext1');
model.geom('part1').create('boxsel1', 'BoxSelection');
model.geom('part1').feature('boxsel1').label('Sweeping Surface 1');
model.geom('part1').feature('boxsel1').set('entitydim', 2);
model.geom('part1').feature('boxsel1').set('zmin', 'il_mx+sl_bl-sl_bl/50');
model.geom('part1').feature('boxsel1').set('zmax', 'il_mx+sl_bl+sl_bl/50');
model.geom('part1').feature('boxsel1').set('condition', 'inside');

```

七、管状反应器教学 APP



部分程序代码：

```
model.param.set('A', '16.96e12[1/h]', 'Frequency factor');
model.param.set('cA0', 'n_po0/v0', 'Propylene oxide concentration, inlet');
model.param.set('cB0', 'n_w0/v0', 'Water concentration, inlet');
model.param.set('cMe0', 'n_m0/v0', 'Methanol concentration, inlet');
model.param.set('Cp_m', '81.095[J/mol/K]', 'Specific heat, m');
model.param.set('Cp_pg', '192.59[J/mol/K]', 'Specific heat, pg');
model.param.set('Cp_po', '146.54[J/mol/K]', 'Specific heat, po');
model.param.set('Cp_w', '75.36[J/mol/K]', 'Specific heat, w');
model.param.set('Cp0', '(Cp_po*cA0+Cp_m*cMe0+Cp_w*cB0)/rho0', 'Heat capacity at inlet');
model.param.set('Cpc', '4180[J/(kg*K)]', 'Heat capacity, cooling fluid');
model.param.set('dHrx', '-84666[J/mol]', 'Heat of reaction');
model.param.set('Diff', '1e-9[m^2/s]', 'Diffusion coefficient');
model.param.set('E', '75362[J/mol]', 'Activation energy');
model.param.set('ke', '0.559[W/m/K]', 'Thermal conductivity');
model.param.set('L', '1[m]', 'Reactor length');
model.param.set('M_m', '32.042[g/mol]', 'Molar weight, methanol');
model.param.set('M_po', '58.095[g/mol]', 'Molar weight, propylene oxide');
model.param.set('M_w', '18[g/mol]', 'Molar weight, water');
model.param.set('mc', '0.1[kg/s]', 'Mass flow rate, cooling fluid');
model.param.set('n_m0', 'v_m0*rho_m_p/M_m', 'Molar flow rate m in');
model.param.set('n_po0', '0.1[mol/s]', 'Molar flow rate po in');
model.param.set('n_w0', 'rho_w_p*v_w0/M_w', 'Molar flow rate w in');
model.param.set('Ra', '0.1[m]', 'Reactor radius');
model.param.set('rho_m_p', '791.3[kg/m^3]', 'Density, methanol');
model.param.set('rho_po_p', '830[kg/m^3]', 'Density, propylene oxide');
model.param.set('rho_w_p', '1000[kg/m^3]', 'Density, water');
```

```

model.param.set('rho0', '(cA0*M_po+cB0*M_w+cMe0*M_m)', 'Density at inlet');
model.param.set('T0', '312[K]', 'Inlet temperature');
model.param.set('Ta0', '277[K]', 'Inlet temperature of the coolant');
model.param.set('tolerance', '1e-3', 'Relative tolerance of the nonlinear solver');
model.param.set('Uk', '1300[W/m^2/K]', 'Overall heat transfer coefficient');
model.param.set('v_m0', 'v_po0', 'Volumetric flow rate m in');
model.param.set('v_po0', 'n_po0*M_po/rho_po_p', 'Volumetric flow rate po in');
model.param.set('v_ratio', '3.5', 'Ratio water flow rate / (prop+me) flow rate');
model.param.set('v_w0', 'v_ratio*(v_po0+v_m0)', 'Volumetric flow rate w in');
model.param.set('v0', 'v_w0+v_po0+v_m0', 'Total flow rate');

model.component('comp1').variable.create('var1');

model.component('comp1').geom('geom1').run;

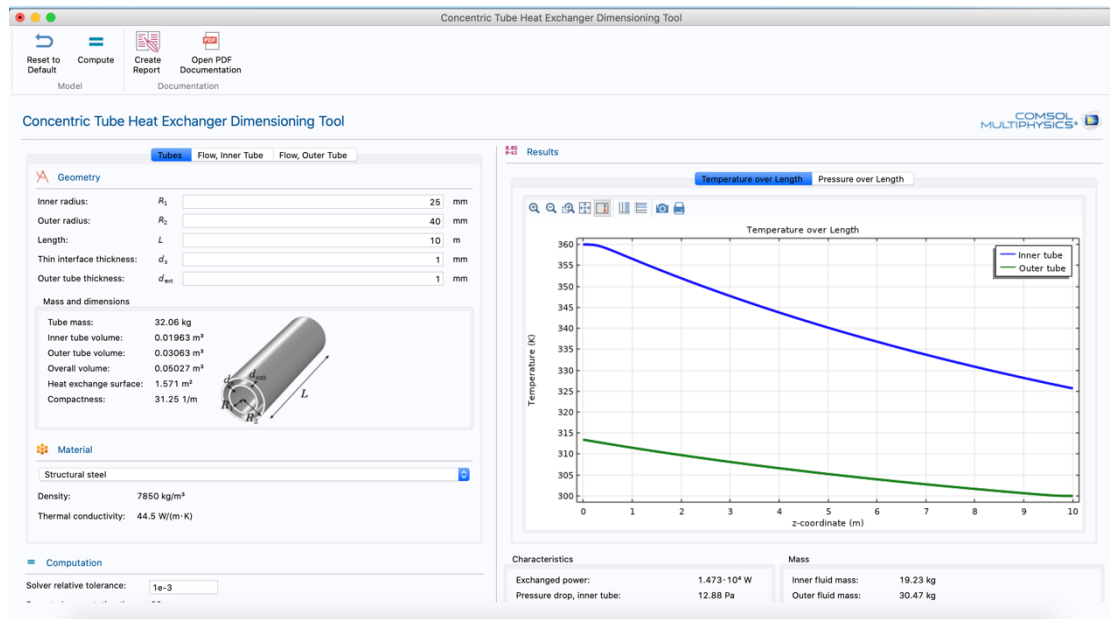
model.component('comp1').variable('var1').set('u0', 'v0/(pi*Ra^2)', 'Average flow rate');
model.component('comp1').variable('var1').set('uz', '2*u0*(1-(r/Ra)^2)', 'Laminar flow profile');
model.component('comp1').variable('var1').set('xA', '(cA0-cA)/cA0', 'Conversion species A');
model.component('comp1').variable('var1').set('cB', 'cB0-cA0*xA', 'Concentration species B');
model.component('comp1').variable('var1').set('cC', 'cA0*xA', 'Concentration species C');
model.component('comp1').variable('var1').set('rA', '-A*exp(-E/R_const/T)*cA', 'Reaction rate');
model.component('comp1').variable('var1').set('Q', '(-rA)*(-dHrx)', 'Heat production term');
model.component('comp1').variable('var1').set('cpm', '(Cp_po*cA+Cp_m*cMe0+Cp_w*cB)/rho0',
'Mixture specific heat');

model.component('comp1').geom('geom1').repairTolType('relative');
model.component('comp1').geom('geom1').create('r1', 'Rectangle');
model.component('comp1').geom('geom1').feature('r1').set('size', {'Ra' 'L'});
model.component('comp1').geom('geom1').feature('fin').set('repairtoltype', 'relative');
model.component('comp1').geom('geom1').runPre('fin');

model.component('comp1').material.create('mat1', 'Common');
model.component('comp1').material('mat1').propertyGroup('def').func.create('eta', 'Piecewise');
model.component('comp1').material('mat1').propertyGroup('def').func.create('Cp', 'Piecewise');
model.component('comp1').material('mat1').propertyGroup('def').func.create('rho', 'Piecewise');
model.component('comp1').material('mat1').propertyGroup('def').func.create('k', 'Piecewise');
model.component('comp1').material('mat1').propertyGroup('def').func.create('cs', 'Interpolation');
model.component('comp1').material('mat1').propertyGroup('def').func.create('an1', 'Analytic');
model.component('comp1').material('mat1').propertyGroup('def').func.create('an2', 'Analytic');
model.component('comp1').material('mat1').propertyGroup('def').func.create('an3', 'Analytic');
model.component('comp1').material('mat1').label('Water');
model.component('comp1').material('mat1').set('family', 'water');
model.component('comp1').material('mat1').set('groups', {'liquids' 'Liquids'});
model.component('comp1').material('mat1').propertyGroup('def').func('eta').set('arg', 'T');

```

八、同心圆管换热器教学 APP



部分程序代码：

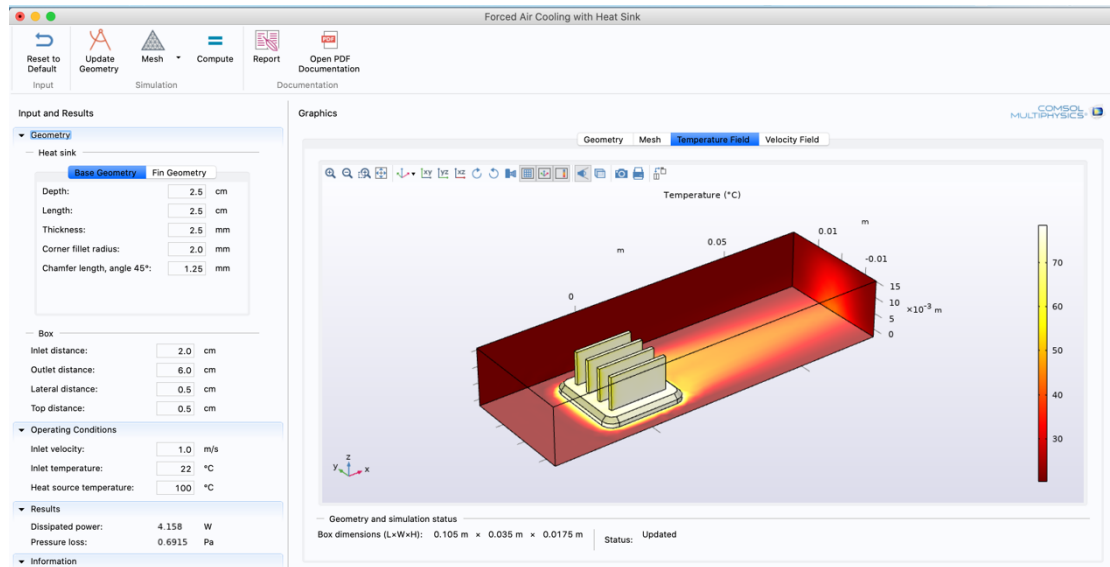
```
model.param.set('R1', '25[mm]');
model.param.descr('R1', 'Inner radius');
model.param.set('R2', '40[mm]');
model.param.descr('R2', 'Outer radius');
model.param.set('L', '10[m]');
model.param.descr('L', 'Length');
model.param.set('ds', '1[mm]');
model.param.descr('ds', 'Thin interface thickness');
model.param.set('dext', '1[mm]');
model.param.descr('dext', 'Outer tube thickness');
model.param.set('T1', '360[K]');
model.param.descr('T1', 'Inner inlet temperature');
model.param.set('T2', '300[K]');
model.param.descr('T2', 'Outer inlet temperature');
model.param.set('mfr1', '0.1[kg/s]');
model.param.descr('mfr1', 'Inner mass flow rate');
model.param.set('mfr2', '0.25[kg/s]');
model.param.descr('mfr2', 'Outer mass flow rate');
model.param.set('pA1', '2[bar]');
model.param.descr('pA1', 'Inner outlet absolute pressure');
model.param.set('pA2', '2[bar]');
model.param.descr('pA2', 'Outer outlet absolute pressure');
model.param.set('vol1', 'L*pi*R1^2');
model.param.descr('vol1', 'Inner tube volume');
model.param.set('vol2', 'L*pi*(R2^2-R1^2)');
```

```

model.param.descr('vol2', 'Outer tube volume');
model.param.set('vol0', 'vol1+vol2');
model.param.descr('vol0', 'Overall volume');
model.param.set('S', '2*pi*R1*L');
model.param.descr('S', 'Heat exchange surface');
model.param.set('beta', 'S/vol0');
model.param.descr('beta', 'Compactness');
model.param.set('rho0_tube', '7850[kg/m^3]');
model.param.descr('rho0_tube', 'Tube density');
model.param.set('k0_tube', '44.5[W/(m*K)]');
model.param.descr('k0_tube', 'Tube thermal conductivity');
model.param.set('rho0_inner', '1000[kg/m^3]');
model.param.descr('rho0_inner', 'Density, inner tube');
model.param.set('Cp0_inner', '4200[J/(kg*K)]');
model.param.descr('Cp0_inner', 'Heat capacity at constant pressure, inner tube');
model.param.set('k0_inner', '0.6[W/(m*K)]');
model.param.descr('k0_inner', 'Thermal conductivity, inner tube');
rtyGroup('def').set('ratioofspecificeat', {'gamma0_inner'});
model.material.create('mat5', 'Common', '');
model.material('mat5').label('User defined, outer fluid');
model.material('mat5').propertyGroup('def').set('dynamicviscosity', '');
model.material('mat5').propertyGroup('def').set('heatcapacity', '');
model.material('mat5').propertyGroup('def').set('density', '');
model.material('mat5').propertyGroup('def').set('thermalconductivity', '');
model.material('mat5').propertyGroup('def').set('ratioofspecificeat', '');
model.material('mat5').propertyGroup('def').set('dynamicviscosity', {'mu0_outer'});
model.material('mat5').propertyGroup('def').set('heatcapacity', {'Cp0_outer'});
model.material('mat5').propertyGroup('def').set('density', {'rho0_outer'});
model.material('mat5').propertyGroup('def').set('thermalconductivity', {'k0_outer'});
model.material('mat5').propertyGroup('def').set('ratioofspecificeat', {'gamma0_outer'});
model.material.create('mat6', 'Common', '');
model.material('mat6').propertyGroup.create('Enu', 'Young"s modulus and Poisson"s ratio');
model.material('mat6').propertyGroup.create('Murnaghan', 'Murnaghan');
model.material('mat6').propertyGroup.create('Lame', ['Lam' native2unicode(hex2dec({'00' 'e9'}),
'unicode') ' parameters']);
model.material('mat6').label('Structural steel');
model.material('mat6').set('family', 'custom');
model.material('mat6').set('specular', 'custom');
model.material('mat6').set('customspecular', [0.7843137254901961 0.7843137254901961
0.7843137254901961]);
model.material('mat6').set('diffuse', 'custom');
model.material('mat6').set('customdiffuse', [0.6666666666666666 0.6666666666666666
0.6666666666666666]);
model.material('mat6').set('ambient', 'custom');

```

九、换热片强制冷却教学 APP



部分程序代码：

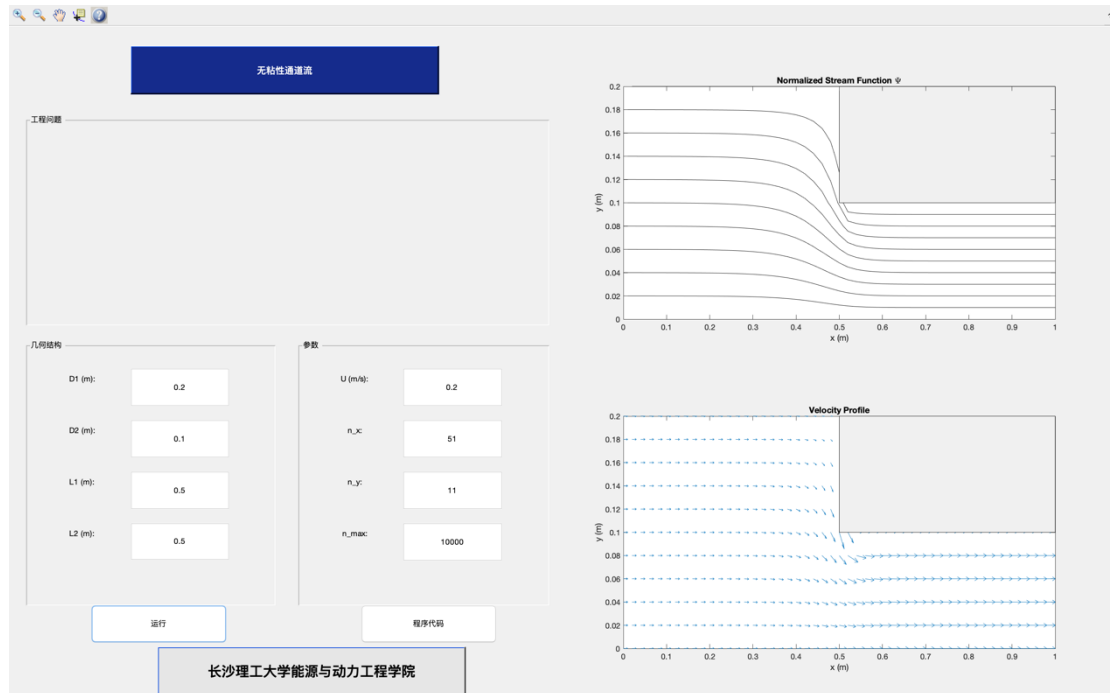
```
model.geom('part4').feature('arr1').selection('input').set({'cyl1'});
model.geom('part4').feature('arr1').set('fullsize', {'pin_n_width' '1' '1'});
model.geom('part4').feature('arr1').setIndex('fullsize', 'pin_n_depth', 1);
model.geom('part4').feature('arr1').set('displ', {'spacing_width' '0' '0'});
model.geom('part4').feature('arr1').setIndex('displ', 'spacing_depth', 1);
model.geom('part4').run('arr1');
model.geom('part4').run('arr1');
model.geom('part4').create('uni1', 'Union');
model.geom('part4').feature('uni1').selection('input').set({'arr1'});
model.geom('part4').run('uni1');
model.geom.create('part5', 'Part', 3);
model.geom('part5').inputParam.set('surf_width', '2.25[cm]');
model.geom('part5').inputParam.descr('surf_width', 'Upper surface width, heat sink');
model.geom('part5').inputParam.set('surf_depth', '2.25[cm]');
model.geom('part5').inputParam.descr('surf_depth', 'Upper surface depth, heat sink');
model.geom('part5').inputParam.set('fillet_r', '2.0[mm]');
model.geom('part5').inputParam.descr('fillet_r', 'Fillet radius');
model.geom('part5').inputParam.set('fin_h', '1.0[cm]');
model.geom('part5').inputParam.descr('fin_h', 'Fin height');
model.geom('part5').inputParam.set('pin_th', '2.0[mm]');
model.geom('part5').inputParam.descr('pin_th', 'Pin thickness');
model.geom('part5').inputParam.set('pin_n_width', '5');
model.geom('part5').inputParam.descr('pin_n_width', 'Number of pins in width');
model.geom('part5').inputParam.set('pin_n_depth', '5');
model.geom('part5').inputParam.descr('pin_n_depth', 'Number of pins in depth');
model.geom('part5').inputParam.set('pin_rot', '1');
```

```

model.component('comp1').geom('geom1').feature.create('wp1', 'WorkPlane');
model.component('comp1').geom('geom1').feature('wp1').set('unite', true);
model.component('comp1').geom('geom1').feature('wp1').geom.create('r1', 'Rectangle');
model.component('comp1').geom('geom1').feature('rot4').set('keep', true);
model.component('comp1').geom('geom1').feature('rot4').set('rot', 180);
model.component('comp1').geom('geom1').run('rot4');
model.material.create('mat1', 'Common', "");
model.material('mat1').propertyGroup.create('Enu', 'Young"s modulus and Poisson"s ratio');
model.material('mat1').propertyGroup.create('Murnaghan', 'Murnaghan');
model.material('mat1').propertyGroup.create('Lame', ['Lam' native2unicode(hex2dec({'00' 'e9'}),
'unicode') ' parameters']);
model.material('mat1').label('Aluminum');
model.material('mat1').set('family', 'aluminum');
model.material('mat1').propertyGroup('def').set('relpermeability', {'1' '0' '0' '0' '1' '0' '0' '0' '1'});
model.material('mat1').propertyGroup('def').set('heatcapacity', '900[J/(kg*K)]');
model.component('comp1').material('mat3').propertyGroup('def').set('relpermittivity', {'1' '0' '0' '0' '1' '0'
'0' '0' '1'});
model.component('comp1').material('mat3').propertyGroup('def').set('dynamicviscosity', 'eta(T)');
model.component('comp1').material('mat3').propertyGroup('def').set('ratioofspecificeat', '1.4');
model.component('comp1').material('mat3').propertyGroup('def').set('electricconductivity', {'0[S/m]' '0'
'0' '0' '0[S/m]' '0' '0' '0' '0[S/m]'});
model.component('comp1').material('mat3').propertyGroup('def').set('heatcapacity', 'Cp(T)');
model.component('comp1').material('mat3').propertyGroup('def').set('density', 'rho(pA,T)');
model.component('comp1').material('mat3').propertyGroup('def').set('thermalconductivity', {'k(T)' '0' '0'
'0' 'k(T)' '0' '0' '0' 'k(T)'});
model.component('comp1').material('mat3').propertyGroup('def').set('soundspeed', 'cs(T)');
model.component('comp1').material('mat3').propertyGroup('def').set('thermalexpansioncoefficient',
{'alpha_p(pA,T)' '0' '0' '0' 'alpha_p(pA,T)' '0' '0' '0' 'alpha_p(pA,T)'});
model.component('comp1').material('mat3').propertyGroup('def').set('molarmass', '0.02897');
model.component('comp1').material('mat3').propertyGroup('def').set('bulkviscosity', 'muB(T)');
model.component('comp1').material('mat3').propertyGroup('def').addInput('temperature');
model.component('comp1').material('mat3').propertyGroup('def').addInput('pressure');
model.component('comp1').material('mat3').propertyGroup('RefractiveIndex').set('n', "");
model.component('comp1').material('mat3').propertyGroup('RefractiveIndex').set('ki', "");
model.component('comp1').material('mat3').propertyGroup('RefractiveIndex').set('n', {'1' '0' '0' '0' '1' '0'
'0' '0' '1'});
model.component('comp1').material('mat3').propertyGroup('RefractiveIndex').set('ki', {'0' '0' '0' '0' '0'
'0' '0' '0'});
model.component('comp1').material('mat3').propertyGroup('NonlinearModel').set('BA',
'(def.gamma+1)/2');
model.component('comp1').material('mat3').materialType('nonSolid');
model.component('comp1').material('mat3').set('family', 'air');
model.component('comp1').material('mat3').selection.named('dif1');

```

十、无粘性通道流教学 APP



部分程序代码：

```
function varargout = inviscidChannelGUI(varargin)
```

```
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @inviscidChannelGUI_OpeningFcn, ...
    'gui_OutputFcn',  @inviscidChannelGUI_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 inviscidChannelGUI is made visible.
```

```
function inviscidChannelGUI_OpeningFcn(hObject, eventdata, handles, varargin)
```

```

% Initialize the GUI layout
sketchplot(gcf);
set(get(handles.plotstreamline,'XLabel'),'String','x (m)')
set(get(handles.plotstreamline,'YLabel'),'String','y (m)')
set(get(handles.plotstreamline,'Title'),'String','Normalized Stream Function \Psi')
set(get(handles.plotvector,'XLabel'),'String','x (m)')
set(get(handles.plotvector,'YLabel'),'String','y (m)')
set(get(handles.plotvector,'Title'),'String','Velocity Profile')
% Create the help button on toolbar
[X, map] = imread(fullfile(...
    matlabroot,'toolbox','matlab','icons','csh_icon.gif'));
icon = ind2rgb(X,map);
uipushtool(handles.uitoolbar1,'CData',icon,...
    'TooltipString','Help',...
    'ClickedCallback',@AppHelp);

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

% Update handles structure
guidata(hObject, handles);

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

% --- Outputs from this function are returned to the command line.
function varargout = inviscidChannelGUI_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 button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject      handle to start (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
inviscidchannel(handles);

```

```

function handles = getpara(handles)
% Get user input parameters from the GUI
handles para.D1 = str2double(get(handles.D1,'String'));
handles para.D2 = str2double(get(handles.D2,'String'));
handles para.L1 = str2double(get(handles.L1,'String'));
handles para.L2 = str2double(get(handles.L2,'String'));
handles para.U1 = str2double(get(handles.U1,'String'));
handles para.nx = str2double(get(handles.nx,'String'));
handles para.ny = str2double(get(handles.ny,'String'));
handles para.nmax = str2double(get(handles.nmax,'String'));
cla(handles.plotstreamline),cla(handles.plotvector),
axis([handles para.L1 handles para.L2 0
max(handles para.D1,handles para.D2)]),
% Draw the channel shape according to input
if handles para.D1 < handles para.D2 % converging channel
    cornerposition = [0 handles para.D1 handles para.L1 handles para.D2-handles para.D1];
elseif handles para.D1 > handles para.D2 % diverging channel
    cornerposition = [handles para.L1 handles para.D2 handles para.L2 handles para.D1-
handles para.D2];
else
    cornerposition = [0 handles para.D1 1 1]; % straight channel
end
rectangle('Parent',handles.plotstreamline,'Position',cornerposition,'FaceColor',[0.94 0.94 0.94])
rectangle('Parent',handles.plotvector,'Position',cornerposition,'FaceColor',[0.94 0.94 0.94])

```