ME 608 HW2

Yash Ganatra

Comments:

The discretization technique employed is finite volume starting with $\nabla \cdot J = S$ where J is the flux of specific quantity φ and S is the source term at steady state. This is integrated over the control volume and divergence theorem is applied. Subsequently, profile assumptions are made for φ and S. The entire process is shown on paper.

Problem 1(a)

TDMA Routine

```
function u = tdma(a,b,c,d,N)
%TDMA
% a(i) = b(i)T(i+1)+c(i)T(i-1)+d(i)
% a denoted diag index = 0 , b=1 , c=-1, d = RHS , N= gridpoints
% T(i) = P(i)T(i+1)+Q(i) -> Recursion used
%Start with P(1),Q(1)
P = zeros([1,N]); % row vector
Q = zeros([1,N]); % row vector
u = zeros([1,N]); % row vector
P(1) = b(1)./a(1);
Q(1) = d(1)./a(1);

for i = 2:N
    P(i) = b(i)./(a(i) - (c(i).*P(i-1)));
    Q(i) = (d(i)+(c(i).*Q(i-1)))./(a(i) - (c(i).*P(i-1)));
end
u(N) = Q(N);
%Back substitution
for i = N-1:-1:1
    u(i) = (P(i).*u(i+1))+Q(i);
end
end
```

Problem 1(b)

Line by Line TDMA Routine which calls tdma routine (above)

Note: For Problems 1 and 2, no under-relaxation is applied hence it is not an argument of TDMA solver routine. The following routine includes under-relaxation.

```
function [u,iter]=
linebylinetdma_underrelaxation(guess_value,urelax,my_ap,my_an,my_as,my_ae,my_aw,my_b,my_CVx,my_CVy,
my_TOL )
%LINE BY LINE TDMA
error = 1;
iter =0;
u = guess_value;
while error> my_TOL
    iter = iter +1;
    uold = u;

for j = 2: my_CVx+1
```

```
  u(2:my_CVy+1,j) = tdma(my_ap(:,j-1),my_as(:,j-1),my_an(:,j-1),(my_aw(:,j-1).*u(2:my_CVy+1,j-1))+my_b(:,j-1)+(my_ae(:,j-1).*u(2:my_CVy+1,j+1)),my_CVy);
end
%-------------row sweeps from i = 1 to CVy ; top to bottom-----------------
for i = 2:my_CVy+1
 u(i,2:my_CVx+1) = tdma(my_ap(i-1,:),my_ae(i-1,:),my_aw(i-1,:),(my_an(i-1,:).*u(i-1,2:my_CVx+1))+my_b(i-1,:)+ (my_as(i-1,:).*u(i+1,2:my_CVx+1)),my_CVx);
end
% <<<<---------Reverse sweeps------------------>>>>>
% --------column sweeps from right to left-------------
for j = my_CVx+1:-1:2
u(2:my_CVy+1,j) = tdma(my_ap(:,j-1),my_as(:,j-1),my_an(:,j-1),(my_aw(:,j-1).*u(2:my_CVy+1,j-1))+my_b(:,j-1)+(my_ae(:,j-1).*u(2:my_CVy+1,j+1)),my_CVy);
end
% ------------row sweeps from bottom to top--------------
for i = my_CVy+1:-1:2
 u(i,2:my_CVx+1) = tdma(my_ap(i-1,:),my_ae(i-1,:),my_aw(i-1,:),(my_an(i-1,:).*u(i-1,2:my_CVx+1))+my_b(i-1,:)+ (my_as(i-1,:).*u(i+1,2:my_CVx+1)),my_CVx);
end

error= max(max(abs((uold-u)./u)));
u = uold + urelax.*(u-uold);
hold on
end
end
```

Validation case

Rectangular plate with prescribed (Dirichlet BC) on 4 edges

```
close all; clear all

%Solver 2d heat conduction steady state
% flat plate with prescribed temperatures at ends
l = 1; %length in m
w = 1; %width in m
Tb_left = 40; Tb_right = 80;Tb_top = 100; Tb_bottom = 60;

 %Grid props -- uniform grid
CVx = [5]; %number of CV in x
CVy = [5];%number of CV in y
x_increment = l./CVx;
y_increment= w./CVy;

[xface,yface] = meshgrid(0:x_increment:l,0:y_increment:w);%check
for i = 1:(size(xface,2)-1) %Generate cell centroids in x
    xp(i) = (xface(1,i)+xface(1,i+1))./2;
end
for i = 1:(size(yface,1)-1)%Generate cell centroids in y
    yp(i) = (yface(i,1)+yface(i+1,1))./2;
end
[xx,yy] = meshgrid(xp,yp); %Generate 2d gridpoints
x = zeros([1,(CVx+2)]);
y = zeros([(CVy+2),1]);
x(1,2:(CVx+1)) = xp;
y(2:(CVy+1),1) = yp;
x(1,1) = 0.0;
x(1,CVx+2) = l;
```

```
y(1,1) = 0;
y(CVy+2,1) = w;
%Boundary conditions
Tb_left = 40; %in C
Tb_right = 80;
Tb_top = 100;
Tb_bottom = 60;
T_guess = 50;
TOL = 1e-5; %Tolerance criteria
initial_Temperature= ones([CVy+2,CVx+2]).*T_guess;
initial_Temperature(1,:) = Tb_top;
initial_Temperature(CVy+2,:) = Tb_bottom;
initial_Temperature(:,1) = Tb_left;
initial_Temperature(:,CVx+2) = Tb_right;
%Initializing
zone = zeros([CVy,CVx]);
ap = zeros([CVy,CVx]);
aw = zeros([CVy,CVx]);
ae = zeros([CVy,CVx]);
an = zeros([CVy,CVx]);
as = zeros([CVy,CVx]);
ab = zeros([CVy,CVx]);
b = ones([CVy,CVx]).*TOL;

%Identify interior,boundary, edge nodes and label corresponding CV; corner CV denoted by 0
zone(2:(CVy-1),2:(CVx-1)) = 1; %interior nodes
zone(1,2:(CVx-1)) = 2; %top boundary
zone(CVy,2:(CVx-1)) = 3; %bottom boundary
zone(2:(CVy-1),1) = 4; %left boundary
zone(2:(CVy-1),CVx) = 5; %right boundary
%corner nodes are 0
zone;
%Discretize for interior nodes
for i = 1:CVy
    for j = 1:CVx
        if(zone(i,j)==1) %interior
            delta_y = yface(i+1,j)-yface(i,j);
            delta_x = xface(i,j+1)-xface(i,j);
            del_w = xx(i,j)-xx(i,j-1);
            del_e = xx(i,j+1)-xx(i,j);
            del_n = yy(i,j)-yy(i-1,j);
            del_s = yy(i+1,j)-yy(i,j);
            aw(i,j) = k.*delta_y./del_w;
            ae(i,j) = k.*delta_y./del_e;
            an(i,j) = k.*delta_x./del_n;
            as(i,j) = k.*delta_x./del_s;
            ap(i,j) = aw(i,j)+ae(i,j)+an(i,j)+as(i,j);
        end
        if(zone(i,j)==2)% top
            delta_y = yface(i+1,j)-yface(i,j);
            delta_x = xface(i,j+1)-xface(i,j);
            del_w = xx(i,j)-xx(i,j-1);
            del_e = xx(i,j+1)-xx(i,j);
            del_s = yy(i+1,j)-yy(i,j);
            del_b = yy(i,j) - yface(i,j);
            aw(i,j) = k.*delta_y./del_w;
```

```
        ae(i,j) = k.*delta_y./del_e;
        as(i,j) = k.*delta_x./del_s;
        ab(i,j) = k.*delta_x./del_b;
        b(i,j) = ab(i,j).*Tb_top;
        ap(i,j) = aw(i,j)+ae(i,j)+ab(i,j)+as(i,j);
    end
    if(zone(i,j)==3) %bottom
        delta_y = yface(i+1,j)-yface(i,j);
        delta_x = xface(i,j+1)-xface(i,j);
        del_w = xx(i,j)-xx(i,j-1);
        del_e = xx(i,j+1)-xx(i,j);
        del_n = yy(i,j)-yy(i-1,j);
        del_b = yface(i+1,j) - yy(i,j);
        aw(i,j) = k.*delta_y./del_w;
        ae(i,j) = k.*delta_y./del_e;
        an(i,j) = k.*delta_x./del_n;
        ab(i,j) = k.*delta_x./del_b;
        b(i,j) = ab(i,j).*Tb_bottom;
        ap(i,j) = aw(i,j)+ae(i,j)+ab(i,j)+an(i,j);
    end
    if(zone(i,j)==4)%left
        delta_y = yface(i+1,j)-yface(i,j);
        delta_x = xface(i,j+1)-xface(i,j);
        del_e = xx(i,j+1)-xx(i,j);
        del_n = yy(i,j)-yy(i-1,j);
        del_s = yy(i+1,j)-yy(i,j);
        del_b = xx(i,j) - xface(i,j);
        ae(i,j) = k.*delta_y./del_e;
        an(i,j) = k.*delta_x./del_n;
        as(i,j) = k.*delta_x./del_s;
        ab(i,j) = k.*delta_y./del_b;
        b(i,j) = ab(i,j).*Tb_left;
        ap(i,j) = an(i,j)+ae(i,j)+ab(i,j)+as(i,j);
    end
    if(zone(i,j)==5)%right
        delta_y = yface(i+1,j)-yface(i,j);
        delta_x = xface(i,j+1)-xface(i,j);
        del_w = xx(i,j)-xx(i,j-1);
        del_n = yy(i,j)-yy(i-1,j);
        del_s = yy(i+1,j)-yy(i,j);
        del_b = xface(i,j+1)-xx(i,j);
        aw(i,j) = k.*delta_y./del_w;
        an(i,j) = k.*delta_x./del_n;
        as(i,j) = k.*delta_x./del_s;
        ab(i,j) = k.*delta_y./del_b;
        b(i,j) = ab(i,j).*Tb_right;
        ap(i,j) = an(i,j)+aw(i,j)+ab(i,j)+as(i,j);
    end
    %%Corner nodes discretization
    if(zone(i,j)==0 && xx(i,j)<xface(i,2) && yy(i,j)<yface(2,j)) %Left top
       zone(i,j) = 6;
       delta_y = yface(i+1,j)-yface(i,j);
       delta_x = xface(i,j+1)-xface(i,j);
       del_s = yy(i+1,j)-yy(i,j);
       del_e = xx(i,j+1)-xx(i,j);
       as(i,j) = k.*delta_x./del_s;
```

```
    ae(i,j) = k.*delta_y./del_e;
    del_b_w = xx(i,j) - xface(i,j);
    del_b_n = yy(i,j) - yface(i,j);
    ab_w = k.*delta_y./del_b_w; %No west
    ab_n = k.*delta_x./del_b_n; %No north
    ab(i,j) = ab_w + ab_n;
    b(i,j) = (ab_w.*Tb_left)+(ab_n.*Tb_top);
    ap(i,j) = as(i,j)+ae(i,j)+ab(i,j);
  end
   if(zone(i,j)==0 && xx(i,j)<xface(i,CVx+1) && yy(i,j)<yface(2,j)) %Right top
    zone(i,j) = 7;
    delta_y = yface(i+1,j)-yface(i,j);
    delta_x = xface(i,j+1)-xface(i,j);
    del_s = yy(i+1,j)-yy(i,j);
    del_w = xx(i,j)-xx(i,j-1);
    as(i,j) = k.*delta_x./del_s;
    aw(i,j) = k.*delta_y./del_w;
    del_b_e = xface(i,j+1)-xx(i,j);
    del_b_n = yy(i,j) - yface(i,j);
    ab_e = k.*delta_y./del_b_e; %No east-adj
    ab_n = k.*delta_x./del_b_n; %No north
    ab(i,j) = ab_e + ab_n;
    b(i,j) = (ab_e.*Tb_right)+(ab_n.*Tb_top);
    ap(i,j) = aw(i,j)+ab(i,j)+as(i,j);
   end
  if(zone(i,j)==0 && xx(i,j)<xface(i,2) && yy(i,j)<yface(CVy+1,j)) %Left bottom
    zone(i,j) = 8;
    delta_y = yface(i+1,j)-yface(i,j);
    delta_x = xface(i,j+1)-xface(i,j);
    del_n = yy(i,j)-yy(i-1,j);
    del_e = xx(i,j+1)-xx(i,j);
    an(i,j) = k.*delta_x./del_n;
    ae(i,j) = k.*delta_y./del_e;
    del_b_w = xx(i,j) - xface(i,j); %No west
    del_b_s = yface(i+1,j) - yy(i,j); %No south-adj
    ab_w = k.*delta_y./del_b_w;
    ab_s = k.*delta_x./del_b_s;
    ab(i,j) = ab_w + ab_s;
    b(i,j) = (ab_w.*Tb_left)+(ab_s.*Tb_bottom);
    ap(i,j) = an(i,j)+ae(i,j)+ab(i,j);
  end
  if(zone(i,j)==0 && xx(i,j)<xface(i,CVx+1) && yy(i,j)<yface(CVy+1,j)) %Left bottom
    zone(i,j) = 9;
    delta_y = yface(i+1,j)-yface(i,j);
    delta_x = xface(i,j+1)-xface(i,j);
    del_n = yy(i,j)-yy(i-1,j);
    del_w = xx(i,j)-xx(i,j-1);
    an(i,j) = k.*delta_x./del_n;
    aw(i,j) = k.*delta_y./del_w;
    del_b_e = xface(i,j+1)-xx(i,j);
    del_b_s = yface(i+1,j) - yy(i,j); %No south-adj
    ab_e = k.*delta_y./del_b_e; %No east-adj
    ab_s = k.*delta_x./del_b_s;
    ab(i,j) = ab_e + ab_s;
    b(i,j) = (ab_e.*Tb_right)+(ab_s.*Tb_bottom);
    ap(i,j) = an(i,j)+aw(i,j)+ab(i,j);
```

```
        end
    end
end

Temperature = linebylinetdma(initial_Temperature,ap,an,as,ae,aw,b,CVx,CVy,TOL);
%Assign values to plot
% subplot(2,2,1),contour(x,flipud(y),Temperature),
% title('Temperature (Steady State)'),xlabel('x'),ylabel('y'),colorbar
% subplot(2,2,2),pcolor(x,flipud(y),Temperature),shading interp,
% title('Temperature (Steady State)'),xlabel('x'),ylabel('y'),colorbar
% subplot(2,2,3),surf(x,flipud(y),Temperature),
% title('Temperature (Steady State)'),xlabel('x'),ylabel('y'),colorbar,
hold on
plot(flipud(y),Temperature(:,(floor(CVx/2)+1)),'k*-'),xlabel('y'),ylabel('Temperature'),legend(['CV:'
num2str(CVx)]),title('Comparison between finite volume (Line by line TDMA) and finite element discretization
schemes')
```

Finite Difference Discretization:

```
nref = 7;  %grid has n - 2 interior points per dimension (overlapping)
xref = linspace(0,1,nref); dx = xref(2)-xref(1); yref = xref; dy = dx;
TOLref = 1e-6;
Tref = zeros(nref);
Tref(1,1:nref) = 60; %TOP
Tref(nref,1:nref) = 100;  %BOTTOM
Tref(1:nref,1) = 40;  %LEFT
Tref(1:nref,nref) = 80;  %RIGHT
Tref
dt = dx^2/4;
errorref = 1; kref = 0;
    while errorref > TOLref
        kref = kref+1;
          Toldref = Tref;
          for i = 2:nref-1
              for j = 2:nref-1
              Tref(i,j) = dt*((Toldref(i+1,j)-2*Toldref(i,j)+Toldref(i-1,j))/dx^2 ...
                      + (Toldref(i,j+1)-2*Toldref(i,j)+Toldref(i,j-1))/dy^2) ...
                      + Toldref(i,j);
              end
          end
          errorref = max(max(abs(Toldref-Tref)));
    end
    Tref
% subplot(2,1,1),contour(xref,yref,Tref),
% title('Temperature (Steady State)'),xlabel('x'),ylabel('y'),colorbar
% subplot(2,1,2),pcolor(xref,yref,Tref),shading interp,
% title('Temperature (Steady State)'),xlabel('x'),ylabel('y'),colorbar
plot(yref,Tref(:,(floor(nref/2)+1)),'r+-'),xlabel('y'),ylabel('Temperature')
hold off
```

Result

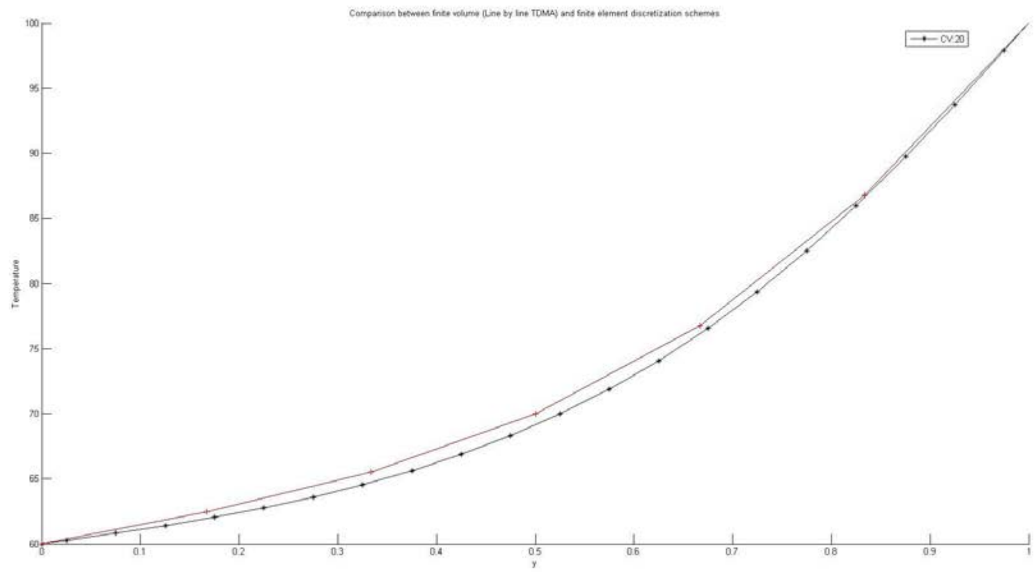

Figure 1. Comparison b/w Finite Difference and Finite volume techniques for 20 control volumes

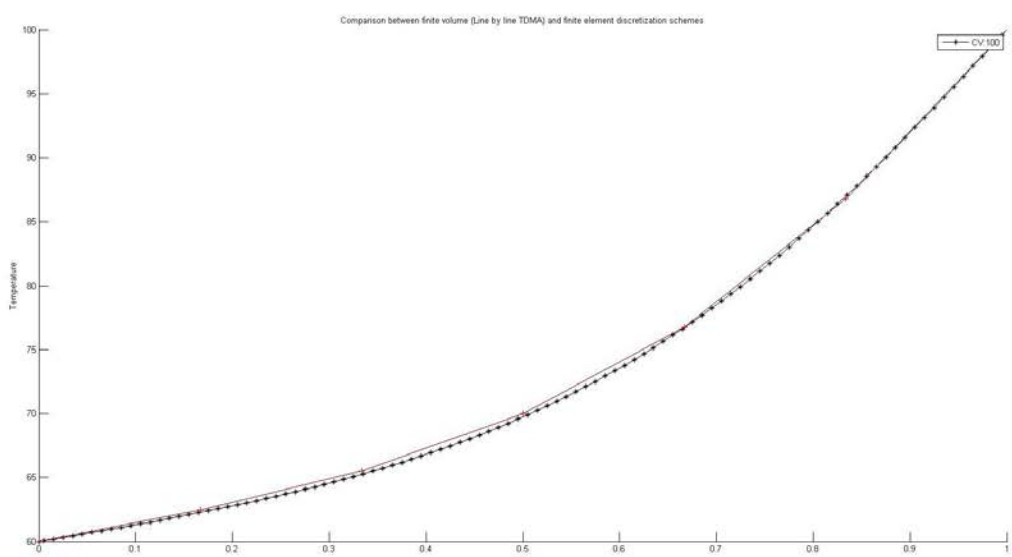

Figure 2: Comparison b/w Finite Difference and Finite volume techniques for 100 control volumes

Problem 2

Length = 0.2 m , Width = 0.2 m ; mu (visocosity) = 8.4e-3 kg/(s.m)

BC: No slip at walls – Dirichlet and Symmetry BC for left and bottom edges

Only a quadrant of square annulus is considered due to symmetry.

The thickness is allocated based upon number of control volumes.

The coordinate system for programming is Cartesian with geometry on x-y plane

```
clc
close all
clear all
%Solver 2d fluid flow in a square annulus of length L and thickness l
% flat plate with no slip condition at 2 ends + symmetry BC at left ,
% bottom where duct is there -> y needs to be flipped due to array indexing
% in (i,j)
% contstant pressure gradient in z (resembles heat generation inside)
L = 0.1; % quadrant length in m
W = 0.1; %quadrant width in m
mu = 0.00084; % Dynamic viscosity in kg/(s.m)
%Grid props -- uniform grid
CVx = [5]; %number of CV in x
CVy = [5];%number of CV in y
x_increment = L./CVx;
y_increment= W./CVy;

[xface,yface] = meshgrid(0:x_increment:L,0:y_increment:W);
for i = 1:(size(xface,2)-1) %Generate cell centroids in x
   xp(i) = (xface(1,i)+xface(1,i+1))./2;
end
for i = 1:(size(yface,1)-1)%Generate cell centroids in y
   yp(i) = (yface(i,1)+yface(i+1,1))./2;
end
[xx,yy] = meshgrid(xp,yp); %Generate 2d gridpoints
x = zeros([1,(CVx+2)]);
y = zeros([(CVy+2),1]);
x(1,2:(CVx+1)) = xp;
y(2:(CVy+1),1) = yp;
x(1,1) = 0.0;
x(1,CVx+2) = L;
y(1,1) = 0;
y(CVy+2,1) = W;

%Boundary conditions
qub_left = 0; %Symmetry - Neumann
qub_bottom = 0;% Symmetry - Neumann
ub_right = 0;%No slip - Dirichlet
```

```
ub_top = 0; % No slip - Dirichlet
ub_xduct = 0;  % No slip - Dirichlet
ub_yduct = 0; % No slip - Dirichlet
v_guess = 100; %velocity in m/s
%SOURCE TERM THERE
S = 20; %dp/dz = constant
TOL = 1e-6; %Tolerance criteria
initial_velocity= ones([CVy+2,CVx+2]).*v_guess;
initial_velocity(1,:) = ub_top;
initial_velocity(:,CVx+2) = ub_right;
initial_velocity;

%Initializing
zone = zeros([CVy,CVx]);
ap = zeros([CVy,CVx]);
aw = zeros([CVy,CVx]);
ae = zeros([CVy,CVx]);
an = zeros([CVy,CVx]);
as = zeros([CVy,CVx]);
ab = zeros([CVy,CVx]);
b = ones([CVy,CVx]);
%Setting control volumes for duct
CVx_duct = floor(CVx/2);
CVy_duct = floor(CVy/2);
%Hydraulic Diameter
ly = W - abs(yface(CVy,1)-yface((CVy-CVy_duct),1)); %thickness
lx = L - abs(xface(CVy,1)-xface(CVy,CVx_duct+2)); %thickness
Acs = (L*W - (L-2*lx)*(W-2*ly));
Perimeter = 2*(L+W)+2*((L-2*lx)+(W-2*ly));
Dh = 4*Acs/Perimeter;
%Identify interior,boundary, edge nodes and label corresponding CV; corner CV denoted by 0
zone(2:(CVy-1),2:(CVx-1)) = 1; %interior nodes
zone(1,2:(CVx-1)) = 2; %top boundary
zone(CVy,(CVx_duct+3):(CVx-1)) = 3; %bottom boundary
zone(2:(CVy-CVy_duct)-2,1) = 4; %left boundary
zone(2:(CVy-1),CVx) = 5; %right boundary
zone((CVy-CVy_duct):CVy,(1:CVx_duct+1)) = 111; %interior of duct
zone((CVy-CVy_duct)-1,2:CVx_duct+1) = 333; % has a "bottom boundary"
zone((CVy-CVy_duct):(CVy-1),CVx_duct+2) = 444; % has a "left" boundary
%corner nodes are 0
zone;
% Discretize for interior nodes
for i = 1:CVy
   for j = 1:CVx
      if(zone(i,j)==1) %interior
         delta_y = abs(yface(i+1,j)-yface(i,j));
         delta_x = abs(xface(i,j+1)-xface(i,j));
         del_w = abs(xx(i,j)-xx(i,j-1));
         del_e = abs(xx(i,j+1)-xx(i,j));
         del_n = abs(yy(i,j)-yy(i-1,j));
         del_s = abs(yy(i+1,j)-yy(i,j));
         aw(i,j) = mu.*delta_y./del_w;
         ae(i,j) = mu.*delta_y./del_e;
         an(i,j) = mu.*delta_x./del_n;
         as(i,j) = mu.*delta_x./del_s;
         ap(i,j) = aw(i,j)+ae(i,j)+an(i,j)+as(i,j);
```

```
        b(i,j) = (S*delta_y.*delta_x);
    end

    if(zone(i,j)==2)% top - Dirichlet boundary condition -- NO SLIP
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_w = abs(xx(i,j)-xx(i,j-1));
        del_e = abs(xx(i,j+1)-xx(i,j));
        del_s = abs(yy(i+1,j)-yy(i,j));
        del_b = abs(yy(i,j) - yface(i,j));
        aw(i,j) = mu.*delta_y./del_w;
        ae(i,j) = mu.*delta_y./del_e;
        as(i,j) = mu.*delta_x./del_s;
        ab(i,j) = mu.*delta_x./del_b;
        b(i,j) = (ab(i,j).*ub_top) + (S*delta_y.*delta_x);
        ap(i,j) = aw(i,j)+ae(i,j)+ab(i,j)+as(i,j);
    end
    if(zone(i,j)==3) %bottom - Neumann --SYMMETRY
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_w = abs(xx(i,j)-xx(i,j-1));
        del_e = abs(xx(i,j+1)-xx(i,j));
        del_n = abs(yy(i,j)-yy(i-1,j));
        del_b = abs(yface(i+1,j) - yy(i,j));
        aw(i,j) = mu.*delta_y./del_w;
        ae(i,j) = mu.*delta_y./del_e;
        an(i,j) = mu.*delta_x./del_n;
        b(i,j) = (qub_bottom.*delta_x )+(S*delta_y.*delta_x);
        ap(i,j) = aw(i,j)+ae(i,j)+an(i,j);
    end
    if(zone(i,j)==4)%left - Neumann --SYMMETRY
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_e = abs(xx(i,j+1)-xx(i,j));
        del_n = abs(yy(i,j)-yy(i-1,j));
        del_s = abs(yy(i+1,j)-yy(i,j));
        del_b = abs(xx(i,j) - xface(i,j));
        ae(i,j) = mu.*delta_y./del_e;
        an(i,j) = mu.*delta_x./del_n;
        as(i,j) = mu.*delta_x./del_s;
        b(i,j) = (qub_left.*delta_y)+(S*delta_y.*delta_x);
        ap(i,j) = an(i,j)+ae(i,j)+as(i,j);
    end
    if(zone(i,j)==5)%right - Dirichlet -- NO SLIP
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_w = abs(xx(i,j)-xx(i,j-1));
        del_n = abs(yy(i,j)-yy(i-1,j));
        del_s = abs(yy(i+1,j)-yy(i,j));
        del_b = abs(xface(i,j+1)-xx(i,j));
        aw(i,j) = mu.*delta_y./del_w;
        an(i,j) = mu.*delta_x./del_n;
        as(i,j) = mu.*delta_x./del_s;
        ab(i,j) = mu.*delta_y./del_b;
        b(i,j) = (ab(i,j).*ub_right) +(S*delta_y.*delta_x);
        ap(i,j) = an(i,j)+aw(i,j)+ab(i,j)+as(i,j);
```

```
    end

    %%<<<----------Corner nodes discretization----------------->>>
    if(zone(i,j)==0 && xx(i,j)<xface(i,2) && yy(i,j)<yface(2,j)) %Left top -> Neumann+Dirichlet
      zone(i,j) = 6;
      delta_y = abs(yface(i+1,j)-yface(i,j));
      delta_x = abs(xface(i,j+1)-xface(i,j));
      del_s = abs(yy(i+1,j)-yy(i,j));
      del_e = abs(xx(i,j+1)-xx(i,j));
      as(i,j) = mu.*delta_x./del_s;
      ae(i,j) = mu.*delta_y./del_e;
      del_b_w = abs(xx(i,j) - xface(i,j));%No west
      del_b_n = abs(yy(i,j) - yface(i,j));%No north
      ab_n = mu.*delta_x./del_b_n;
      ab(i,j) = ab_n;
      b(i,j) = (qub_left.*delta_y)+(ab(i,j).*ub_top)+(S*delta_y.*delta_x); %took ab instead of ab_n
      ap(i,j) = as(i,j)+ae(i,j)+ab(i,j);
    end
     if(zone(i,j)==0 && xx(i,j)<xface(i,CVx+1) && yy(i,j)<yface(2,j)) %Right top -> Dirichlet+Dirichlet
      zone(i,j) = 7;
      delta_y = abs(yface(i+1,j)-yface(i,j));
      delta_x = abs(xface(i,j+1)-xface(i,j));
      del_s = abs(yy(i+1,j)-yy(i,j));
      del_w = abs(xx(i,j)-xx(i,j-1));
      as(i,j) = mu.*delta_x./del_s;
      aw(i,j) = mu.*delta_y./del_w;
      del_b_e = abs(xface(i,j+1)-xx(i,j));%No east-adj
      del_b_n = abs(yy(i,j) - yface(i,j)); %No north
      ab_e = mu.*delta_y./del_b_e;
      ab_n = mu.*delta_x./del_b_n;
      ab(i,j) = ab_e + ab_n;
      b(i,j) = (ab_e.*ub_right)+(ab_n.*ub_top)+(S*delta_y.*delta_x); %Split effects - note
      ap(i,j) = aw(i,j)+ab(i,j)+as(i,j);
     end
    if(zone(i,j)==0 && xx(i,j)<xface(i,2) && yy(i,j)<yface((CVy-CVy_duct+1),j)) %Left bottom MID corner ->
Neumann+Dirichlet
      zone(i,j) = 8;
      delta_y = abs(yface(i+1,j)-yface(i,j));
      delta_x = abs(xface(i,j+1)-xface(i,j));
      del_n = abs(yy(i,j)-yy(i-1,j));
      del_e = abs(xx(i,j+1)-xx(i,j));
      an(i,j) = mu.*delta_x./del_n;
      ae(i,j) = mu.*delta_y./del_e;
      del_b_w = abs(xx(i,j) - xface(i,j)); %No west
      del_b_s = abs(yface(i+1,j) - yy(i,j)); %No south-adj
      ab_s = mu.*delta_x./del_b_s;
      ab(i,j) = ab_s;
      b(i,j) = (qub_left.*delta_y)+(ab(i,j).*ub_xduct)+(S*delta_y.*delta_x);
      ap(i,j) = an(i,j)+ae(i,j)+ab(i,j);
    end
     if(zone(i,j)==0 && xx(i,j)<xface(i,CVx_duct+3) && yy(i,j)<yface(CVy+1,j)) %Right bottom corner next to
duct -> Neumann+Dirichlet
      zone(i,j) = 9;
      delta_y = abs(yface(i+1,j)-yface(i,j));
      delta_x = abs(xface(i,j+1)-xface(i,j));
      del_n = abs(yy(i,j)-yy(i-1,j));
```

```
        del_e = abs(xx(i,j+1)-xx(i,j));
        an(i,j) = mu.*delta_x./del_n;
        ae(i,j) = mu.*delta_y./del_e;
        del_b_w = abs(xx(i,j) - xface(i,j));%No west-adj
        del_b_s = abs(yface(i+1,j) - yy(i,j)); %No south-adj
        ab_w = mu.*delta_y./del_b_w;
        ab(i,j) = ab_w;
        b(i,j) = (ab_w.*ub_yduct)+(qub_bottom.*delta_x)+(S*delta_y.*delta_x);
        ap(i,j) = an(i,j)+ae(i,j)+ab(i,j);
      end
      if(zone(i,j)==0 && xx(i,j)<xface(i,CVx+1) && yy(i,j)<yface(CVy+1,j)) %Right bottom ->
Neumann+Dirichlet
        zone(i,j) = 10;
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_n = abs(yy(i,j)-yy(i-1,j));
        del_w = abs(xx(i,j)-xx(i,j-1));
        an(i,j) = mu.*delta_x./del_n;
        aw(i,j) = mu.*delta_y./del_w;
        del_b_e = abs(xface(i,j+1)-xx(i,j));%No east-adj
        del_b_s = abs(yface(i+1,j) - yy(i,j)); %No south-adj
        ab_e = mu.*delta_y./del_b_e;
        ab(i,j) = ab_e;
        b(i,j) = (ab_e.*ub_right)+(qub_bottom.*delta_x)+(S*delta_y.*delta_x);
        ap(i,j) = an(i,j)+aw(i,j)+ab(i,j);
      end
      % Discretization for interior duct and cells adjacent to duct other
      % than corner
      if(zone(i,j) == 111) %Interior Duct
         ap(i,j) = 1;b(i,j) = 0;
         an(i,j) = 0;as(i,j) = 0;aw(i,j) = 0;ae(i,j) = 0;
      end
      if(zone(i,j) == 333) % resembles "bottom" boundary ; Dirichlet
         delta_y = abs(yface(i+1,j)-yface(i,j));
         delta_x = abs(xface(i,j+1)-xface(i,j));
         del_w = abs(xx(i,j)-xx(i,j-1));
         del_e = abs(xx(i,j+1)-xx(i,j));
         del_n = abs(yy(i,j)-yy(i-1,j));
         del_b = abs(yface(i+1,j) - yy(i,j));
         aw(i,j) = mu.*delta_y./del_w;
         ae(i,j) = mu.*delta_y./del_e;
         an(i,j) = mu.*delta_x./del_n;
         ab(i,j) = mu.*delta_x./del_b;
         b(i,j) = (ab(i,j).*ub_xduct)+(S*delta_y.*delta_x);
         ap(i,j) = aw(i,j)+ae(i,j)+an(i,j)+ab(i,j);
      end
      if(zone(i,j) == 444) % resembles "Left" boundary ; Dirichlet
         delta_y = abs(yface(i+1,j)-yface(i,j));
         delta_x = abs(xface(i,j+1)-xface(i,j));
         del_e = abs(xx(i,j+1)-xx(i,j));
         del_n = abs(yy(i,j)-yy(i-1,j));
         del_s = abs(yy(i+1,j)-yy(i,j));
         del_b = abs(xx(i,j) - xface(i,j));
         ae(i,j) = mu.*delta_y./del_e;
         an(i,j) = mu.*delta_x./del_n;
         as(i,j) = mu.*delta_x./del_s;
```

```
            ab(i,j) = mu.*delta_y./del_b;
            b(i,j) = (ab(i,j).*ub_yduct)+(S*delta_y.*delta_x);
            ap(i,j) = an(i,j)+ae(i,j)+as(i,j)+ab(i,j);
        end

    end
end
  zone ;
velocity = linebylinetdma(initial_velocity,ap,an,as,ae,aw,b,CVx,CVy,TOL);
del_bw = abs(xx(1,1) - xface(1,1));
del_bs = abs(yface(CVy+1,1) - yy(CVy,1));
velocity(2:CVy+1,1)= (qub_left+ ((mu./del_bw).*velocity(2:CVy+1,2)))./(mu./del_bw);
velocity(CVy+2,1:CVx+1) = (qub_bottom + ((mu./del_bs).*velocity(CVy+2,1:CVx+1)))./(mu./del_bs);
Dim_velocity = mu.*velocity./(power(Dh,2)-S);
%---Post processing----
%Mirror & Concatenate mirrored arrays to map full domain
V1_aux = fliplr(Dim_velocity);
V2_aux = cat(2,V1_aux,Dim_velocity);
V3_aux = flipud(V2_aux);
Velocity_domain = cat(1,V2_aux,V3_aux);
x_aux = -fliplr(x);x_domain = cat(2,x_aux,x);
y_aux = flipud(y); y_aux2= -y; y_domain = cat(1,y_aux,y_aux2);
%Assign values to plot
% subplot(3,2,1),contour(x,flipud(y),velocity),
% title('Velocity'),xlabel('x'),ylabel('y'),colorbar
% subplot(3,2,2),pcolor(x,flipud(y),velocity),shading interp,
% title('Velocity'),xlabel('x'),ylabel('y'),colorbar
% subplot(3,2,3),surf(x,flipud(y),velocity),
% title('Velocity'),xlabel('x'),ylabel('y'),colorbar,
subplot(2,1,2), plot(x_domain,Velocity_domain(CVy,:),'k*-'),axis auto,title('Dimensionless Velocity along center
line'),xlabel('x'),ylabel('y'),legend(['CV:' num2str(CVx),' guess velocity: ' num2str(v_guess)])
% subplot(3,2,6),contour(x_domain,y_domain,Velocity_domain),axis auto
% title('Velocity'),xlabel('x'),ylabel('y'),colorbar
```

Result:

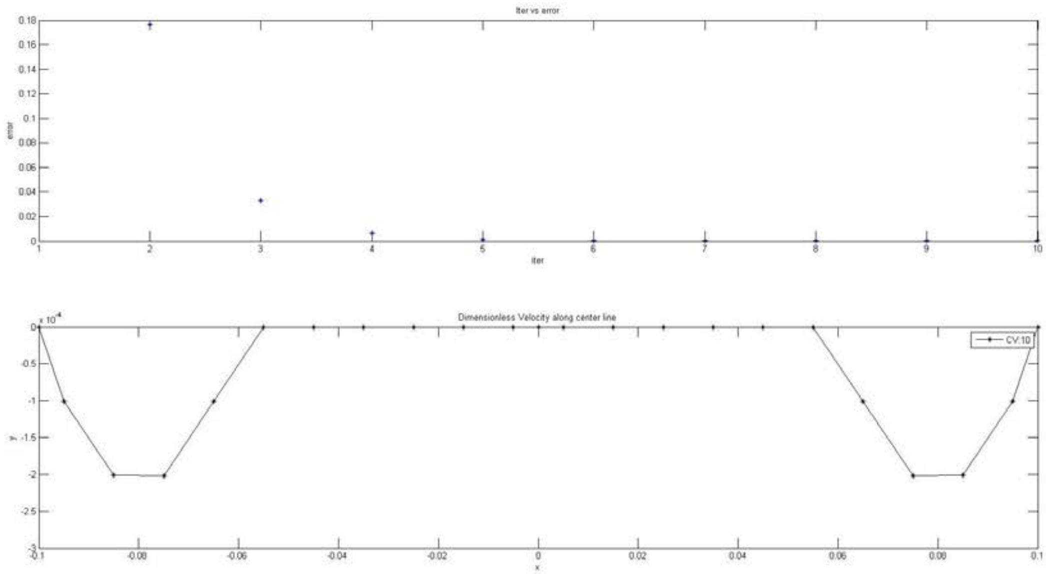

Figure 3(a) Iteration vs Error plot (b) Dimensionless velocity along for 10 control volumes

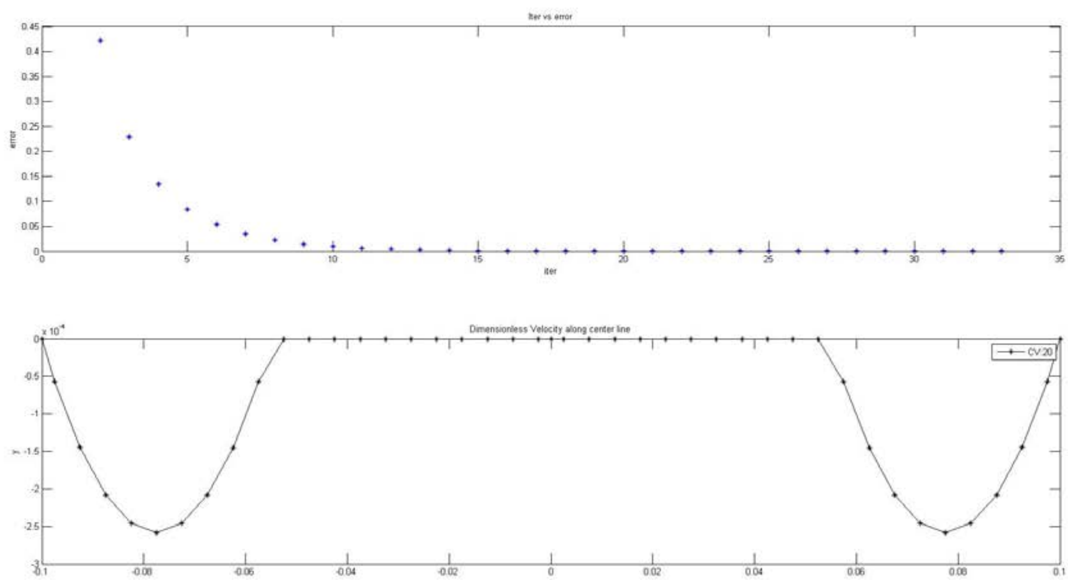

Figure 4 (a) Iteration vs Error plot (b) Dimensionless velocity along for 20 control volumes

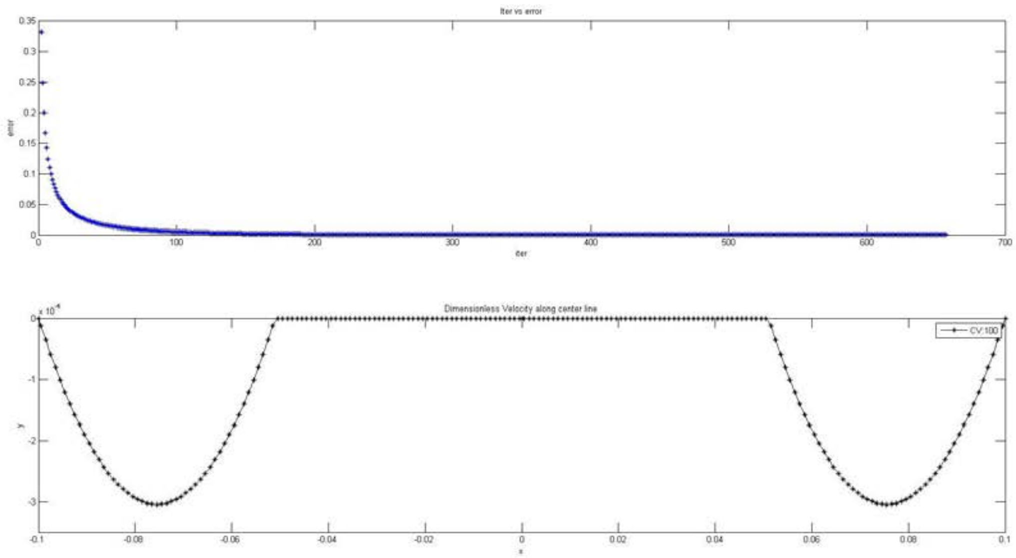

Figure 5 Iteration vs Error plot (b) Dimensionless velocity along for 100 control volumes

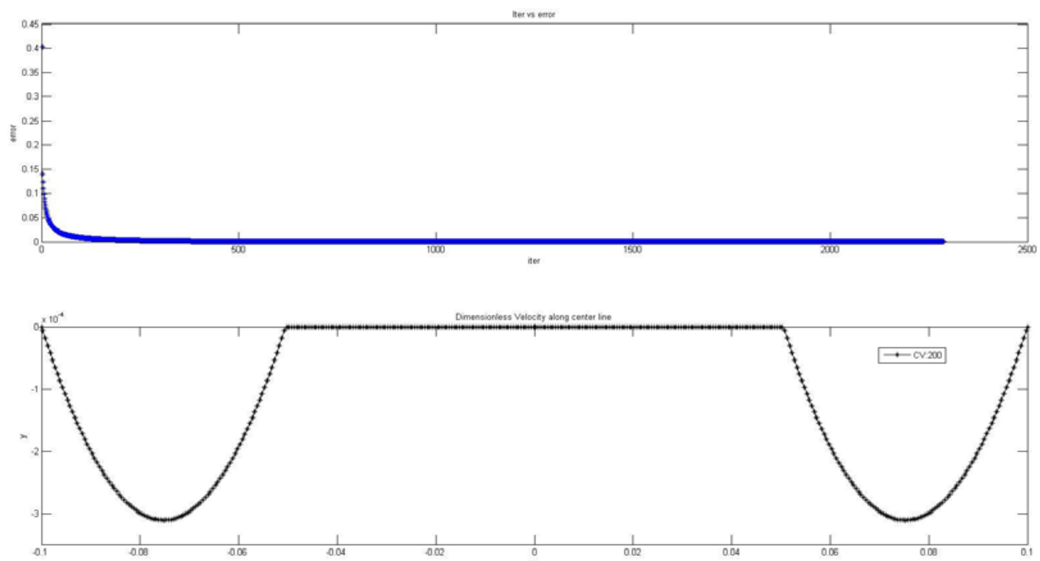

Figure 6 Iteration vs Error plot (b) Dimensionless velocity along for 200 control volumes

Comments:

As number of control volumes increases, the shape of dimensionless velocity changes as shown in Figure 3,4,5. With increasing cell centroids, the piecewise linear profile assumption at face of φ captures its' true variation. The results do not change with guess values.

Problem 3

$\partial T/\partial x$ is derived from an energy balance on control volume of thickness dx and neglecting diffusion. (Shown on paper)

Only a quadrant of square annulus is considered due to symmetry. U is assumed constant – plug flow. A square plate is assumed (0.1 x 0.1 m) and the thickness is allocated based upon number of control volumes.

The coordinate system for programming is Cartesian with geometry on x-y plane.

```
clc
close all
clear all
%Solver 2d fluid flow (constant prescribed velocity - plug flow) in a square annulus of length L and thickness l
% flat plate with adiabatic BC (Neumann BC) on all edges
%--> in discretization U gets cancelled
% bottom where duct is there -> y needs to be flipped due to array indexing
% in (i,j)
%Geometry and Material Properties
L = 0.1; %length in m
W = 0.1; %width in m
k = 50; % W/mK
%Grid props -- uniform grid
CVx = [20]; %number of CV in x
CVy = [20];%number of CV in y
x_increment = L./CVx;
y_increment= W./CVy;
[xface,yface] = meshgrid(0:x_increment:L,0:y_increment:W);
for i = 1:(size(xface,2)-1) %Generate cell centroids in x
   xp(i) = (xface(1,i)+xface(1,i+1))./2;
end
for i = 1:(size(yface,1)-1)%Generate cell centroids in y
   yp(i) = (yface(i,1)+yface(i+1,1))./2;
end
[xx,yy] = meshgrid(xp,yp); %Generate 2d gridpoints
%Setting control volumes for duct
CVx_duct = floor(CVx/2);
CVy_duct = floor(CVy/2);
ly = W - abs(yface(CVy,1)-yface((CVy-CVy_duct),1));
lx = L - abs(xface(CVy,1)-xface(CVy,CVx_duct+2));
Acs = (L*W - (L-2*lx)*(W-2*ly));
x(1,2:(CVx+1)) = xp;
y(1,2:(CVy+1)) = yp;
x(1,1) = 0.0;
x(1,CVx+2) = L;
y(1,1) = 0;
y(1,CVy+2) = W;
insert = @(a, x, n)cat(2,  x(1:n), a, x(n+1:end));
x = insert(xface(1,CVx_duct+2),x,CVx_duct+2); %insert an additional face point as temp
y = insert(yface(CVy-CVy_duct,1),y,CVy-CVy_duct)';%insert an additional face point as temp and take transpose
%Hydraulic Diameter
ly = W - abs(yface(CVy,1)-yface((CVy-CVy_duct),1));
lx = L - abs(xface(CVy,1)-xface(CVy,CVx_duct+2));
Acs = (L*W - (L-2*lx)*(W-2*ly));
Perimeter = 2*(L+W)+2*((L-2*lx)+(W-2*ly));
Dh = 4*Acs/Perimeter;
%Boundary conditions
```

```
qb_left = 0; %Symmetry - Neumann
qb_bottom = 0;% Symmetry - Neumann
qb_right = 0;%Neumann
qb_top = 0; %Neumann
qb_xduct = 0; %Neumann along x axis of duct
qb_yduct = 0; %Neumann along y axis of duct
T_guess = 1; %temperature in K
q0 = 2;
q_gen = q0.*(1+ sin(pi.*xx./L)+sin(pi.*yy./W)).*(Acs-1);%SOURCE TERM
%----------------Solver setup----------
TOL = 7e-4; %Tolerance criteria
alpha = 0.8; %Under-relaxation factor
initial_temperature= ones([CVy+2,CVx+2]).*T_guess;
%Initializing
zone = zeros([CVy,CVx]);
ap = zeros([CVy,CVx]);
aw = zeros([CVy,CVx]);
ae = zeros([CVy,CVx]);
an = zeros([CVy,CVx]);
as = zeros([CVy,CVx]);
b = ones([CVy,CVx]);

%Identify interior,boundary, edge nodes and label corresponding CV; corner CV denoted by 0
zone(2:(CVy-1),2:(CVx-1)) = 1; %interior nodes
zone(1,2:(CVx-1)) = 2; %top boundary
zone(CVy,(CVx_duct+3):(CVx-1)) = 3; %bottom boundary
zone(2:(CVy-CVy_duct)-2,1) = 4; %left boundary
zone(2:(CVy-1),CVx) = 5; %right boundary
zone((CVy-CVy_duct):CVy,(1:CVx_duct+1)) = 111; %interior of duct
zone((CVy-CVy_duct)-1,2:CVx_duct+1) = 333; % has a "bottom boundary"
zone((CVy-CVy_duct):(CVy-1),CVx_duct+2) = 444; % has a "left" boundary
%corner nodes are 0
% Discretize for interior nodes
for i = 1:CVy
    for j = 1:CVx
        if(zone(i,j)==1) %interior
            delta_y = abs(yface(i+1,j)-yface(i,j));
            delta_x = abs(xface(i,j+1)-xface(i,j));
            del_w = abs(xx(i,j)-xx(i,j-1));
            del_e = abs(xx(i,j+1)-xx(i,j));
            del_n = abs(yy(i,j)-yy(i-1,j));
            del_s = abs(yy(i+1,j)-yy(i,j));
            aw(i,j) = k.*delta_y./del_w;
            ae(i,j) = k.*delta_y./del_e;
            an(i,j) = k.*delta_x./del_n;
            as(i,j) = k.*delta_x./del_s;
            ap(i,j) = aw(i,j)+ae(i,j)+an(i,j)+as(i,j);
            b(i,j) = (q_gen(i,j)*delta_y.*delta_x);
        end

        if(zone(i,j)==2)% top - ADIABATIC
            delta_y = abs(yface(i+1,j)-yface(i,j));
            delta_x = abs(xface(i,j+1)-xface(i,j));
            del_w = abs(xx(i,j)-xx(i,j-1));
            del_e = abs(xx(i,j+1)-xx(i,j));
            del_s = abs(yy(i+1,j)-yy(i,j));
```

```
        del_b = abs(yy(i,j) - yface(i,j));
        aw(i,j) = k.*delta_y./del_w;
        ae(i,j) = k.*delta_y./del_e;
        as(i,j) = k.*delta_x./del_s;
        b(i,j) = (qb_top*delta_x) + (q_gen(i,j).*delta_y.*delta_x);
        ap(i,j) = aw(i,j)+ae(i,j)+as(i,j);
    end
    if(zone(i,j)==3) %bottom - Neumann --SYMMETRY
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_w = abs(xx(i,j)-xx(i,j-1));
        del_e = abs(xx(i,j+1)-xx(i,j));
        del_n = abs(yy(i,j)-yy(i-1,j));
        aw(i,j) = k.*delta_y./del_w;
        ae(i,j) = k.*delta_y./del_e;
        an(i,j) = k.*delta_x./del_n;
        b(i,j) = (qb_bottom.*delta_x )+(q_gen(i,j).*delta_y.*delta_x);
        ap(i,j) = aw(i,j)+ae(i,j)+an(i,j);
    end
    if(zone(i,j)==4)%left - Neumann --SYMMETRY
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_e = abs(xx(i,j+1)-xx(i,j));
        del_n = abs(yy(i,j)-yy(i-1,j));
        del_s = abs(yy(i+1,j)-yy(i,j));
        ae(i,j) = k.*delta_y./del_e;
        an(i,j) = k.*delta_x./del_n;
        as(i,j) = k.*delta_x./del_s;
        b(i,j) = (qb_left.*delta_y)+(q_gen(i,j).*delta_y.*delta_x);
        ap(i,j) = an(i,j)+ae(i,j)+as(i,j);
    end
    if(zone(i,j)==5)%right - ADIABATIC
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_w = abs(xx(i,j)-xx(i,j-1));
        del_n = abs(yy(i,j)-yy(i-1,j));
        del_s = abs(yy(i+1,j)-yy(i,j));
        aw(i,j) = k.*delta_y./del_w;
        an(i,j) = k.*delta_x./del_n;
        as(i,j) = k.*delta_x./del_s;
        b(i,j) = (qb_right*delta_y) +(q_gen(i,j).*delta_y.*delta_x);
        ap(i,j) = an(i,j)+aw(i,j)+as(i,j);
    end
    %%<<<----------Corner nodes discretization----------------->>>
    if(zone(i,j)==0 && xx(i,j)<xface(i,2) && yy(i,j)<yface(2,j)) %Left top -> SYMMETRY+CHF
       zone(i,j) = 6;
       delta_y = abs(yface(i+1,j)-yface(i,j));
       delta_x = abs(xface(i,j+1)-xface(i,j));
       del_s = abs(yy(i+1,j)-yy(i,j)); %No west
       del_e = abs(xx(i,j+1)-xx(i,j));%No north
       as(i,j) = k.*delta_x./del_s;
       ae(i,j) = k.*delta_y./del_e;
       b(i,j) = (qb_left.*delta_y)+(qb_top*delta_x)+(q_gen(i,j).*delta_y.*delta_x);
       ap(i,j) = as(i,j)+ae(i,j);
    end
     if(zone(i,j)==0 && xx(i,j)<xface(i,CVx+1) && yy(i,j)<yface(2,j)) %Right top -> CHF+CHF
```

```
            zone(i,j) = 7;
            delta_y = abs(yface(i+1,j)-yface(i,j));
            delta_x = abs(xface(i,j+1)-xface(i,j));
            del_s = abs(yy(i+1,j)-yy(i,j));%No north
            del_w = abs(xx(i,j)-xx(i,j-1));%No east-adj
            as(i,j) = k.*delta_x./del_s;
            aw(i,j) = k.*delta_y./del_w;
            b(i,j) = (qb_right*delta_y)+(qb_top*delta_x)+(q_gen(i,j).*delta_y.*delta_x);
            ap(i,j) = aw(i,j)+as(i,j);
         end
        if(zone(i,j)==0 && xx(i,j)<xface(i,2) && yy(i,j)<yface((CVy-CVy_duct+1),j)) %Left bottom MID corner ->
Symmetry+CHF
            zone(i,j) = 8;
            delta_y = abs(yface(i+1,j)-yface(i,j));
            delta_x = abs(xface(i,j+1)-xface(i,j));
            del_n = abs(yy(i,j)-yy(i-1,j));
            del_e = abs(xx(i,j+1)-xx(i,j));
            an(i,j) = k.*delta_x./del_n;%No south-adj
            ae(i,j) = k.*delta_y./del_e;%No west
            b(i,j) = (qb_left.*delta_y)+(qb_xduct*delta_x)+(q_gen(i,j).*delta_y.*delta_x);
            ap(i,j) = an(i,j)+ae(i,j);
        end
         if(zone(i,j)==0 && xx(i,j)<xface(i,CVx_duct+3) && yy(i,j)<yface(CVy+1,j)) %Right bottom corner next to
duct -> Neumann+Dirichlet
            zone(i,j) = 9;
            delta_y = abs(yface(i+1,j)-yface(i,j));
            delta_x = abs(xface(i,j+1)-xface(i,j));
            del_n = abs(yy(i,j)-yy(i-1,j));
            del_e = abs(xx(i,j+1)-xx(i,j));
            an(i,j) = k.*delta_x./del_n;%No south-adj
            ae(i,j) = k.*delta_y./del_e;%No west-adj
            b(i,j) = (qb_yduct*delta_y)+(qb_bottom.*delta_x)+(q_gen(i,j).*delta_y.*delta_x);
            ap(i,j) = an(i,j)+ae(i,j);
        end
        if(zone(i,j)==0 && xx(i,j)<xface(i,CVx+1) && yy(i,j)<yface(CVy+1,j)) %Right bottom -> Symmetry+CHF
            zone(i,j) = 10;
            delta_y = abs(yface(i+1,j)-yface(i,j));
            delta_x = abs(xface(i,j+1)-xface(i,j));
            del_n = abs(yy(i,j)-yy(i-1,j));
            del_w = abs(xx(i,j)-xx(i,j-1));
            an(i,j) = k.*delta_x./del_n;%No south-adj
            aw(i,j) = k.*delta_y./del_w;%No east-adj
            b(i,j) = (qb_right*delta_y)+(qb_bottom.*delta_x)+(q_gen(i,j).*delta_y.*delta_x);
            ap(i,j) = an(i,j)+aw(i,j);
        end
        % Discretization for interior duct and cells adjacent to duct other
        % than corners
        if(zone(i,j) == 111) %Interior Duct
             ap(i,j) = 1;b(i,j) = 0;
             an(i,j) = 0;as(i,j) = 0;aw(i,j) = 0;ae(i,j) = 0;
        end
        if(zone(i,j) == 333) % resembles "bottom" boundary ; CHF
             delta_y = abs(yface(i+1,j)-yface(i,j));
             delta_x = abs(xface(i,j+1)-xface(i,j));
             del_w = abs(xx(i,j)-xx(i,j-1));
             del_e = abs(xx(i,j+1)-xx(i,j));
```

```
        del_n = abs(yy(i,j)-yy(i-1,j));
        aw(i,j) = k.*delta_y./del_w;
        ae(i,j) = k.*delta_y./del_e;
        an(i,j) = k.*delta_x./del_n;
        b(i,j) = (qb_xduct*delta_x)+(q_gen(i,j).*delta_y.*delta_x);
        ap(i,j) = aw(i,j)+ae(i,j)+an(i,j);
      end
      if(zone(i,j) == 444) % resembles "Left" boundary ; CHF
        delta_y = abs(yface(i+1,j)-yface(i,j));
        delta_x = abs(xface(i,j+1)-xface(i,j));
        del_e = abs(xx(i,j+1)-xx(i,j));
        del_n = abs(yy(i,j)-yy(i-1,j));
        del_s = abs(yy(i+1,j)-yy(i,j));
        ae(i,j) = k.*delta_y./del_e;
        an(i,j) = k.*delta_x./del_n;
        as(i,j) = k.*delta_x./del_s;
        b(i,j) = (qb_yduct*delta_y)+(q_gen(i,j).*delta_y.*delta_x);
        ap(i,j) = an(i,j)+ae(i,j)+as(i,j);
      end

  end
end
[Temp, iteration] = linebylinetdma_underrelaxation(initial_temperature,alpha,ap,an,as,ae,aw,b,CVx,CVy,TOL);

%-----Post Processing------
%Compute temperature at boundary faces from adjacent cell centroids for
%domain
del_bw = abs(xx(1,1) - xface(1,1)); % for left boundary
del_be = abs(xface(1,CVx+1)-xx(1,CVx)); %for right boundary
del_bn = abs(yface(1,1) - yy(1,1)); %for top boundary
del_bs = abs(yface(CVy+1,1) - yy(CVy,1)); % for bottom boundary
Temp(2:CVy+1,1)= (qb_left+ ((k./del_bw).*Temp(2:CVy+1,2)))./(k./del_bw);  %Left boundary face
Temp(2:CVy+1,CVx+2)= (qb_right + ((k./del_be).*Temp(2:CVy+1,CVx+1)))./(k./del_be); %Right boundary face
Temp(1,1:CVx+1) = (qb_top + ((k./del_bn).*Temp(2,1:CVx+1)))./(k./del_bn); % Top boundary face
Temp(CVy+2,1:CVx+1) = (qb_bottom + ((k./del_bs).*Temp(CVy+1,1:CVx+1)))./(k./del_bs); % Bottom boundary
face

%Compute temperature at boundary faces from adjacent cell centroids for
% square duct
Temperature = insertrows(Temp,zeros([1,CVx+2]),CVy - CVy_duct);
Temperature = insertrows(Temperature.',zeros([CVy+3,1]).',CVx_duct+2).';
del_b_duct_s = abs(yface((CVy-CVy_duct),1)-yy((CVy-CVy_duct-1),1));
del_b_duct_w = abs(xface((CVy-CVy_duct),CVx_duct)- xx(CVy-CVy_duct,CVx_duct+2));
Temperature(CVy-CVy_duct+1,1:CVx_duct+2) = (qb_xduct + ((k./del_b_duct_s).*Temperature(CVy-
CVy_duct,1:CVx_duct+2)))./(k./del_b_duct_s); % Bottom duct face
Temperature((CVy-CVy_duct+2):CVy+3,CVx_duct+3)= (qb_yduct  + ((k./del_b_duct_w).*Temperature((CVy-
CVy_duct+2):CVy+3,CVx_duct+4)))./(k./del_b_duct_w); %Left duct face
%Average between temps - at faces
Temperature((CVy-CVy_duct+1),(CVx_duct+4):end) = (Temperature((CVy-
CVy_duct),(CVx_duct+4):end)+Temperature((CVy-CVy_duct+2),(CVx_duct+4):end))./2;
Temperature(1:(CVy-CVy_duct+1),CVx_duct+3) = (Temperature(1:(CVy-
CVy_duct+1),CVx_duct+2)+Temperature(1:(CVy-CVy_duct+1),CVx_duct+4))./2;
% Dimensionless Temperature
T_slice = Temperature(1:(CVy+2),2:end);% Excluding temperatures in first column and last row due to symmetry
delta_y = abs(yface(2,1)-yface(1,1)); %Uniform Mesh throughout domain
delta_x = abs(xface(1,2)-xface(1,1));
```

```
Tsum = sum(T_slice(:)).*4; %Taking entire duct
Tb = Tsum.*delta_x.*delta_y./Acs;
theta = (Temperature(1:(CVy+2),2:end) - Tb)./(q0.*power(Dh,2)./k);
%Assign values to plot
subplot(2,2,1),contour(x(2:end),flipud(y(1:(CVy+2))),T_slice),
title('Temperature (Steady State)'),xlabel('x'),ylabel('y'),colorbar
subplot(3,2,2),pcolor(x(2:end),flipud(y(1:(CVy+2))),T_slice),shading interp,
title('Temperature (Steady State)'),xlabel('x'),ylabel('y'),colorbar
subplot(3,2,3),surf(x(2:end),flipud(y(1:(CVy+2))),T_slice),
title('Temperature (Steady State)'),xlabel('x'),ylabel('y'),colorbar,
% subplot(1,1,1),plot(x(1,(CVx-CVx_duct+1):end),theta((CVx-CVx_duct):end,CVy+2),'k*-'),title('Dimensionless
temperture'),xlabel('x'),ylabel('theta'),legend(['CV:' num2str(CVx), ' under-relaxation: ' num2str(alpha)])
```

Result:

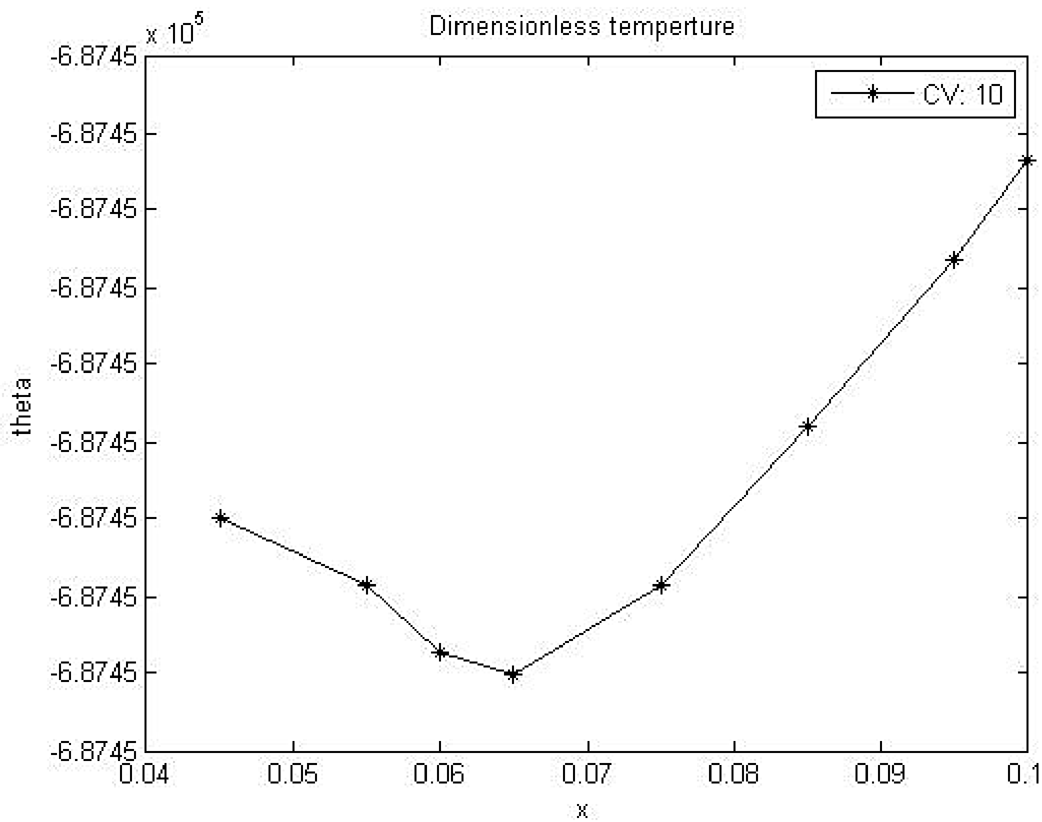

Figure 7 Dimensionless temperature along annulus thickness along horizontal centerline for 10 control volumes

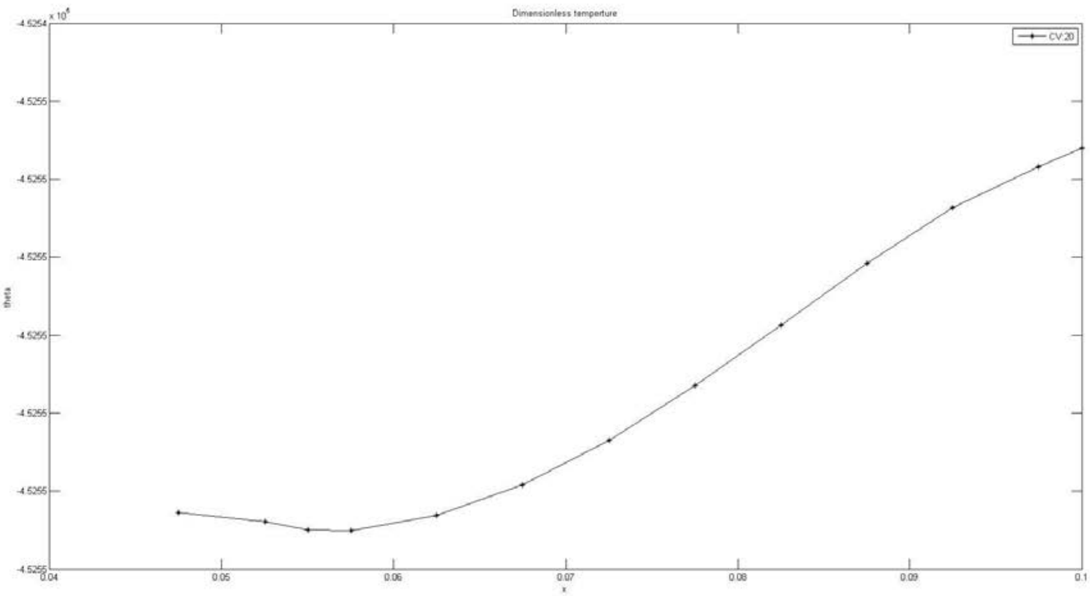

Figure 8 Dimensionless temperature along annulus thickness along horizontal centerline for 20 control volumes

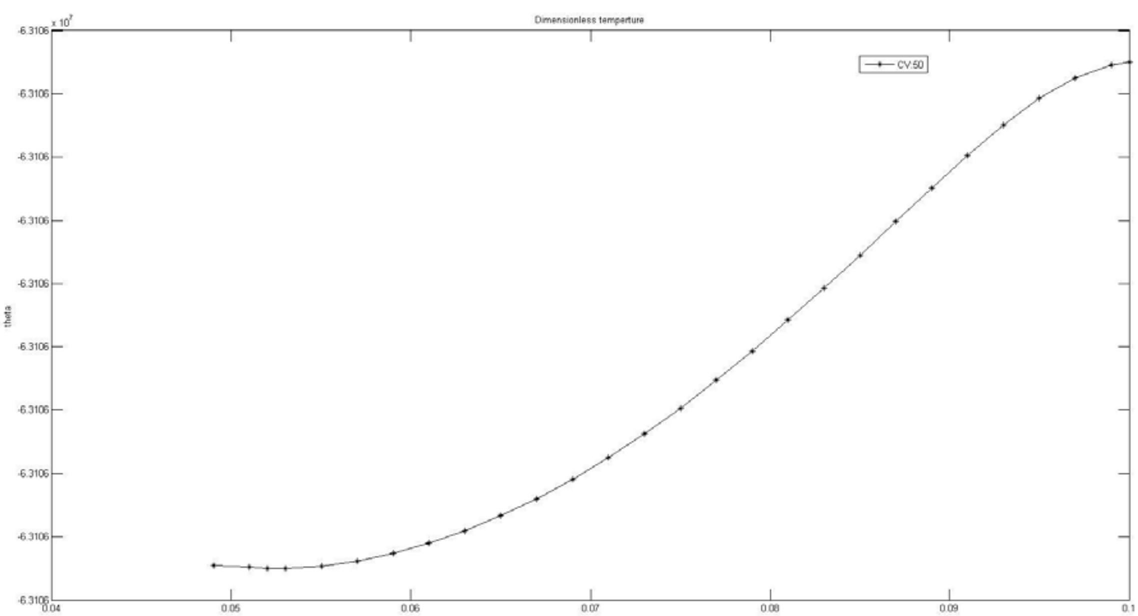

Figure 9 Dimensionless temperature along annulus thickness along horizontal centerline for 50 control volumes

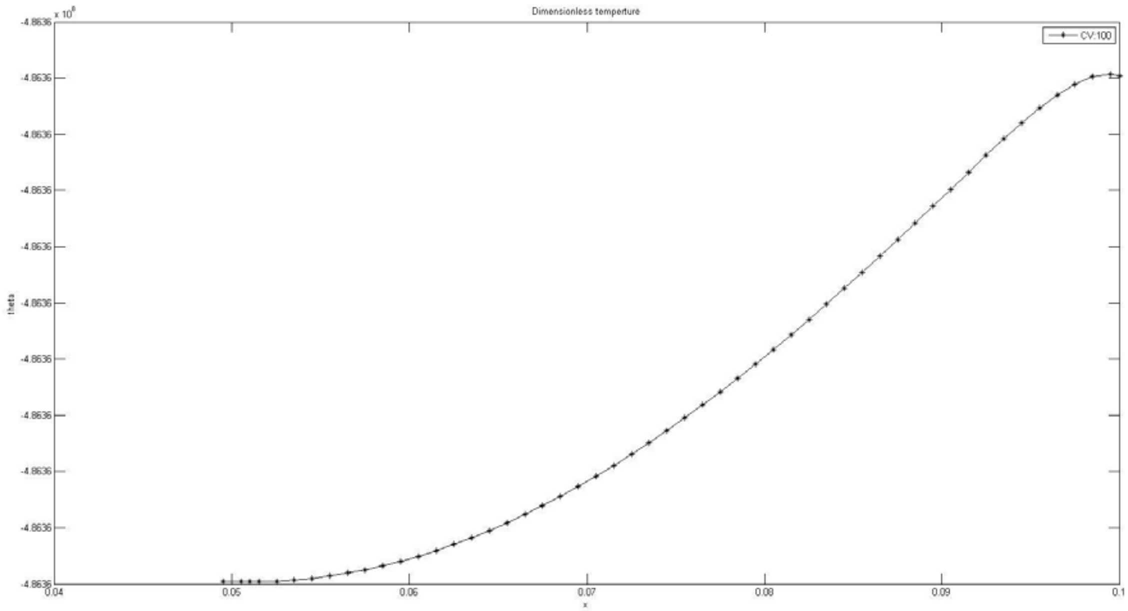

Figure 10 Dimensionless temperature along annulus thickness along horizontal centerline for 100 control volumes

Comments:

It is observed that the temperatures depend on the guess value and the number of nodes. However, the shape of temperature profile remains the same qualitatively. At the ends, the effect of adiabatic boundary condition can be seen with zero slope of the dimensionless temperature profile.

The results don’t change with under-relaxation factors indicating correctness of the solution scheme.