HOMEWORK 4

YASH GANATRA

Problem 2

Results

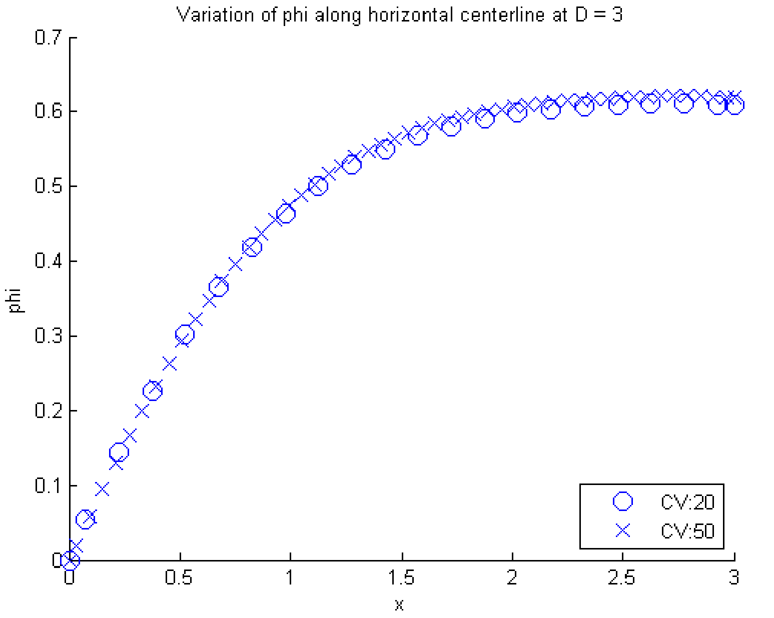

Figure 1 Variation of phi along horizontal centerline for 20, 50 control volumes

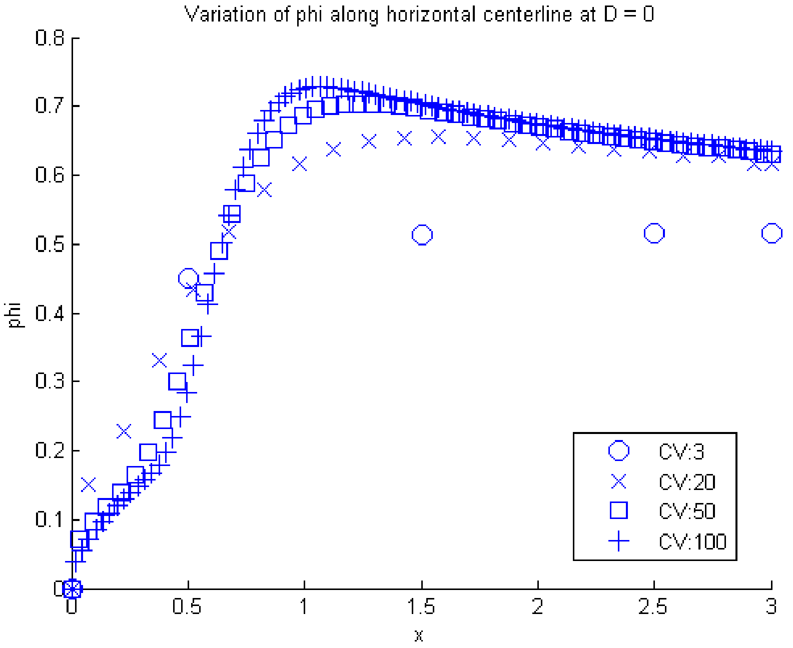

Figure 2 Variation of phi along horizontal centerline for pure convection

Main code

```
%Solver 2D convection-diffusion problem -> QUICK scheme
```

```
clc
close all
clear all
%Geometric properties
l = 3; %length in m
w = 3; %width in m
```

```
%Material Properties
```

```
D = 3; % Diffusion coefficient in kg/ms
rho = 2; %Density in kg/m^3
```

```
%Grid generation -- uniform grid
```

```
i_CVx = [20 50]; %number of CV in x
%CVy = [20];%number of CV in y
```

```
%Plot tools
```

```
plot_style = ['o';'x';'s';'+';'d';'*';'v';'p'];
iter = 0;%for plot
```

```
for CVx = i_CVx
%------Identify interior,boundary nodes and label corresponding CV;
iter = iter+1;
CVy = CVx;
zone = create_zones(CVx,CVy);
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_d,yy_d] = meshgrid(xp,yp); %Generate 2d gridpoints
%FLIPS --> set velocity field according to given cartesian system
yface_flip = flipud(yface); %NOTE v field direction
yy_flip = flipud(yy_d);
yp_flip = flipud(yp);
```

```
%Given Boundary conditions
```

```
phi_s = 0.5;
m_dot = 4;
phi_left = 0.0;
phi_bottom = 1.0;
phi_guess = 0;
phi_inflow_boundary = [phi_left phi_bottom];
```

```
%Convergene criteria for solver
TOL = 1e-5;
```

```
%Dummy rows, columns added to show boundary temperatures
initial_phi= ones([CVy+2,CVx+2]).*phi_guess;
initial_phi(CVy+2,:) = phi_bottom;
initial_phi(:,1) = phi_left;
phi_old = initial_phi(2:CVy+1,2:CVx+1); %intializing phi_old which will carry forward ...
%previous values with initial temperature
```

```
count = 0;
phi_diff = 1;
phi_TOL = 1e-6;
```

```
while phi_diff > phi_TOL
    count = count+1;
    %-----------Discretization------------------
    [ap,an,as,aw,ae,ab,b,b_m,m,F_w,F_e,F_n,F_s,F_b] = discretize2D_cd(phi_old,CVx,CVy,...
        xface,yface_flip,xx_d,yy_flip,...
        D,rho,phi_s,m_dot,phi_inflow_boundary);
    %Solver call
    phi = linebylinetdma(initial_phi,ap,an,as,ae,aw,b,CVx,CVy,TOL);
    phi_current = phi(2:CVy+1,2:CVx+1);
    phi_diff = max(max(abs(phi_old -phi_current)./phi_current));
    phi_old = phi_current;
end
%Using upwinding at boundaries phi_b = phi_p
phi(1,:) = phi(2,:);phi(:,CVx+2)=phi(:,CVx+1);
```

```
%---For Post Processing----%
```

```
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;
figure(1)
hold on
```

```
plot(x,phi((floor(CVy/2)+1),:),plot_style(iter),'MarkerSize',9.5,'DisplayName',(['CV:'
num2str(CVx)]))
xlabel('x'),ylabel('phi')
title('Variation of phi along horizontal centerline at D = 3')
legend('-DynamicLegend','Location','Best')
end
```

Create Zones

```
function [ zone ] = create_zones(CVx,CVy )
zone = zeros([CVy,CVx]);
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
zone(1,1) = 6;
zone(1,CVx) = 7;
zone(CVy,1) = 8;
zone(CVy,CVx) =9 ;
end
```

Discretization

```
%Discretization scheme for Conv-Diff eqn
```

```
function [ap,an,as,aw,ae,ab,b,b_m,m,F_w,F_e,F_n,F_s,F_b] =
discretize2D_cd(phi_old,CVx,CVy,xface,yface,xx,yy,D,rho,phi_s,m_dot,phi_inflow_boundary)
```

```
%Initializing
```

```
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 = zeros([CVy,CVx]);b_m = zeros([CVy,CVx]);
F_e = zeros([CVy,CVx]);F_w = zeros([CVy,CVx]);F_n = zeros([CVy,CVx]);F_s = zeros([CVy,CVx]);
F_b = zeros([CVy,CVx]);
```

```
%-----Variable Terms-----
```

```
m = m_dot.*(xx+yy);%Constant mass source in kg/m^3s
u = @(x)(power(x,2)+1); %calculate "u" velocity (x) component
v = @(y)(power(y,2)+2); %calculate "v" velocity (y) component
```

```
%%INTERIOR CELLS
```

```
for i=2:(CVy-1)
    for j=2:(CVx-1)
        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)); %Note that j-1 = west
        del_e = abs(xx(i,j+1)-xx(i,j)); %Note that j+1 = east
        del_n = abs(yy(i,j)-yy(i-1,j)); %Note that i-1 = north
        del_s = abs(yy(i+1,j)-yy(i,j)); %Note that i+1 = south
        F_e(i,j) = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
        F_w(i,j) = rho*u(xx(i,j)-(delta_x./2))*delta_y;
        F_n(i,j) = rho*v(yy(i,j)+(delta_y./2))*delta_x;
        F_s(i,j) = rho*v(yy(i,j)-(delta_y./2))*delta_x;
        aw(i,j) = max(F_w(i,j),0) + (D.*delta_y./del_w);
        ae(i,j) = max(-F_e(i,j),0) + (D.*delta_y./del_e);
        an(i,j) = max(-F_n(i,j),0) + (D.*delta_x./del_n);
        as(i,j) = max(F_s(i,j),0) + (D.*delta_x./del_s);
        ap(i,j) =  aw(i,j)+ae(i,j)+an(i,j)+as(i,j)+ ...
            F_e(i,j)-F_w(i,j)+F_n(i,j)-F_s(i,j);
        b_m(i,j) = m(i,j)*delta_x*delta_y*phi_s;
```

```
        %Computing corrections to implement QUICK scheme
```

```
        phi_e_correction = ((phi_old(i,j)+ phi_old(i,j+1))./2)-...
            ((phi_old(i,j+1)+phi_old(i,j-1)-2.*phi_old(i,j))./8) - phi_old(i,j);
        phi_w_correction = ((phi_old(i,j)+ phi_old(i,j-1))./2)-...
            ((phi_old(i,j+1)+phi_old(i,j-1)-2.*phi_old(i,j))./8) - phi_old(i,j);
        phi_n_correction = ((phi_old(i,j)+ phi_old(i-1,j))./2)-...
            ((phi_old(i-1,j)+phi_old(i+1,j)-2.*phi_old(i,j))./8) - phi_old(i,j);
        phi_s_correction = ((phi_old(i,j)+ phi_old(i+1,j))./2)-...
            ((phi_old(i-1,j)+phi_old(i+1,j)-2.*phi_old(i,j))./8) - phi_old(i,j);
        b(i,j) = b_m(i,j) + ((F_w(i,j).*phi_w_correction)-(F_e(i,j).*phi_e_correction))+...
            ((F_s(i,j).*phi_s_correction)-(F_n(i,j).*phi_n_correction));
    end
end
```

```
%%LEFT INFLOW BOUNDARY --> UPWIND only
```

```
for i=2:CVy-1
    j=1;
    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));
    F_e(i,j) = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
    F_n(i,j) = rho*v(yy(i,j)+(delta_y./2))*delta_x;
    F_s(i,j) = rho*v(yy(i,j)-(delta_y./2))*delta_x;
    F_b(i,j) = rho*u(xx(i,j)-(delta_x./2))*delta_y;
    ae(i,j) = max(-F_e(i,j),0) + (D.*delta_y./del_e);
    an(i,j) = max(-F_n(i,j),0) + (D.*delta_x./del_n);
    as(i,j) = max(F_s(i,j),0) + (D.*delta_x./del_s);
    ab(i,j) = max(F_b(i,j),0) + (D.*delta_y./del_b);
    ap(i,j) =  ae(i,j)+an(i,j)+as(i,j)+ab(i,j)+F_e(i,j)-F_b(i,j)+F_n(i,j)-F_s(i,j);
    b(i,j) = m(i,j)*delta_x*delta_y*phi_s + (ab(i,j).*phi_inflow_boundary(1));
end
```

```
%%BOTTOM INFLOW BOUNDARY ->UPWIND ONLY
```

```
for j=2:CVx-1
    i=CVy;
    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));
    F_e(i,j) = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
    F_w(i,j) = rho*u(xx(i,j)-(delta_x./2))*delta_y;
    F_n(i,j) = rho*v(yy(i,j)+(delta_y./2))*delta_x;
    F_b(i,j) = rho*v(yy(i,j)-(delta_y./2))*delta_x;
    aw(i,j) = max(F_w(i,j),0) + (D.*delta_y./del_w);
    ae(i,j) = max(-F_e(i,j),0) + (D.*delta_y./del_e);
    an(i,j) = max(-F_n(i,j),0) + (D.*delta_x./del_n);
    ab(i,j) = max(F_b(i,j),0) + (D.*delta_x./del_b);
    ap(i,j) =  aw(i,j)+ae(i,j)+an(i,j)+ab(i,j)+F_e(i,j)-F_w(i,j)+F_n(i,j)-F_b(i,j);
    b(i,j) = m(i,j)*delta_x*delta_y*phi_s + (ab(i,j).*phi_inflow_boundary(2));
end
```

```
%%RIGHT OUTFLOW BOUNDARY ->UPWIND ONLY
```

```
for i=2:CVy-1
    j=CVx;
    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));
    F_b(i,j) = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
    F_w(i,j) = rho*u(xx(i,j)-(delta_x./2))*delta_y;
    F_n(i,j) = rho*v(yy(i,j)+(delta_y./2))*delta_x;
    F_s(i,j) = rho*v(yy(i,j)-(delta_y./2))*delta_x;
    aw(i,j) = max(F_w(i,j),0) + (D.*delta_y./del_w);
    an(i,j) = max(-F_n(i,j),0) + (D.*delta_x./del_n);
    as(i,j) = max(F_s(i,j),0) + (D.*delta_x./del_s);
    ap(i,j) = aw(i,j)+an(i,j)+as(i,j)+F_b(i,j)-F_w(i,j)+F_n(i,j)-F_s(i,j);
    b(i,j) = m(i,j)*delta_x*delta_y*phi_s;
end
```

```
%%TOP OUTFLOW BOUNDARY ->UPWIND ONLY
```

```
for j=2:CVx-1
    i=1;
    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));
    F_e(i,j) = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
    F_w(i,j) = rho*u(xx(i,j)-(delta_x./2))*delta_y;
    F_b(i,j) = rho*v(yy(i,j)+(delta_y./2))*delta_x;
    F_s(i,j) = rho*v(yy(i,j)-(delta_y./2))*delta_x;
    aw(i,j) = max(F_w(i,j),0) + (D.*delta_y./del_w);
    ae(i,j) = max(-F_e(i,j),0) + (D.*delta_y./del_e);
    as(i,j) = max(F_s(i,j),0) + (D.*delta_x./del_s);
    ap(i,j) = aw(i,j)+ae(i,j)+as(i,j)+F_b(i,j)-F_s(i,j)+F_e(i,j)-F_w(i,j);
    b(i,j) = m(i,j)*delta_x*delta_y*phi_s;
end
```

```
%%-----------------------CORNER NODES-----------------------------%%
```

```
%%BOTTOM LEFT CORNER -> both inflow -> UPWIND ONLY ; no W-S
```

```
i=CVy;j=1;
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_b_left = abs(xx(i,j) - xface(i,j));
del_b_bottom = abs(yface(i+1,j) - yy(i,j));
F_e(i,j) = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
F_n(i,j) = rho*v(yy(i,j)+(delta_y./2))*delta_x;
F_b_left = rho*u(xx(i,j)-(delta_x./2))*delta_y;
F_b_bottom = rho*v(yy(i,j)-(delta_y./2))*delta_x;
F_b(i,j) = F_b_left + F_b_bottom;
ae(i,j) = max(-F_e(i,j),0) + (D.*delta_y./del_e);
an(i,j) = max(-F_n(i,j),0) + (D.*delta_x./del_n);
ab_left = max(F_b_left,0) + (D.*delta_y./del_b_left);
ab_bottom = max(F_b_bottom,0) + (D.*delta_x./del_b_bottom);
ab(i,j) = ab_left+ab_bottom;
ap(i,j) =  ae(i,j)+an(i,j)+ab(i,j)+F_e(i,j)-F_b_left-F_b_bottom+F_n(i,j);
b(i,j) = m(i,j)*delta_x*delta_y*phi_s +
(ab_left.*phi_inflow_boundary(1))+(ab_bottom.*phi_inflow_boundary(2));
```

```
%%TOP LEFT CORNER -> inflow (left) and outflow(top)-> UPWIND ONLY ; no W-N
```

```
i=1;j=1;
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_s = abs(yy(i+1,j)-yy(i,j));
del_b_left = abs(xx(i,j) - xface(i,j));
F_e(i,j) = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
F_s(i,j) = rho*v(yy(i,j)-(delta_y./2))*delta_x;
F_b_left = rho*u(xx(i,j)-(delta_x./2))*delta_y;
F_b_top = rho*v(yy(i,j)+(delta_y./2))*delta_x;
F_b(i,j) = F_b_left + F_b_top;
ae(i,j) = max(-F_e(i,j),0) + (D.*delta_y./del_e);
as(i,j) = max(F_s(i,j),0) + (D.*delta_x./del_s);
ab(i,j) = max(F_b_left,0) + (D.*delta_y./del_b_left);
ap(i,j) =  ae(i,j)+as(i,j)+ab(i,j)+F_e(i,j)-F_b_left+F_b_top(i,j)-F_s(i,j);
b(i,j) = m(i,j)*delta_x*delta_y*phi_s + (ab(i,j).*phi_inflow_boundary(1));
```

```
%%TOP RIGHT CORNER ->  outflow (top) and outflow(right)-> UPWIND ONLY ;
%no N-E
```

```
i=1;j=CVx;
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_s = abs(yy(i+1,j)-yy(i,j));
F_w(i,j) = rho*u(xx(i,j)-(delta_x./2))*delta_y;
F_s(i,j) = rho*v(yy(i,j)-(delta_y./2))*delta_x;
F_b_top = rho*v(yy(i,j)+(delta_y./2))*delta_x;
F_b_right = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
F_b(i,j) = F_b_top + F_b_right;
aw(i,j) = max(F_w(i,j),0) + (D.*delta_y./del_w);
as(i,j) = max(F_s(i,j),0) + (D.*delta_x./del_s);
ap(i,j) = aw(i,j)+as(i,j)+F_b_top-F_s(i,j)+F_b_right-F_w(i,j);
b(i,j) = m(i,j)*delta_x*delta_y*phi_s;
```

```
% BOTTOM RIGHT CORNER -> inflow (bottom) and outflow(right)-> UPWIND ONLY ;
% no E-S
```

```
i=CVy;j=CVx;
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_b_bottom = abs(yface(i+1,j) - yy(i,j));
F_w(i,j) = rho*u(xx(i,j)-(delta_x./2))*delta_y;
F_n(i,j) = rho*v(yy(i,j)+(delta_y./2))*delta_x;
F_b_bottom = rho*v(yy(i,j)-(delta_y./2))*delta_x; %Check
F_b_right = rho.*u(xx(i,j)+(delta_x./2)).*delta_y;
F_b(i,j) = F_b_bottom + F_b_right; %NEED?
aw(i,j) = max(F_w(i,j),0) + (D.*delta_y./del_w);
an(i,j) = max(-F_n(i,j),0) + (D.*delta_x./del_n);
ab(i,j) = max(F_b_bottom,0) + (D.*delta_x./del_b_bottom); %ab_bottom
```

```
ap(i,j) =  aw(i,j)+an(i,j)+ab(i,j)+F_b_right-F_w(i,j)+F_n(i,j)-F_b_bottom;
b(i,j) = m(i,j)*delta_x*delta_y*phi_s + (ab(i,j).*phi_inflow_boundary(2));
end
```

Appendix – Solving Routines

Line by Line TDMA

```
function u=
linebylinetdma(guess_temperature,my_ap,my_an,my_as,my_ae,my_aw,my_b,my_CVx,my_CVy,my_TOL )
%LINE BY LINE TDMA
%Solving by line by line TDMA
error = 1;
iter =0;
u = guess_temperature;
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)));
end
end
```

TDMA

```
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 %Check
    u(i) = (P(i).*u(i+1))+Q(i);
end
end
```