```
% AME 535a
% Anthony Medrano
% Boundary Layer FLow


clear all;
close all;

% Initial Parameters

Re=2000;
cfl = .3;
u0 = 1;
L = 1;
H = 3*L;
dx = L/50;
dy = H/100;
t=0;
dt = cfl*dx;
tf = 2.5;
NX = length(0:dx:L);
NY = length(0:dy:H);
[X, Y]=meshgrid(0:dx:L,0:dy:H);
U0 = u0*ones(NY,NX);
V0=zeros(NY,NX);
Fu0=zeros(NY,NX);
Fv0=zeros(NY,NX);

% Dirichlet Boundary Conditions
U0(1,1:NX)=0;
U0(NY,1:NX)=u0;
U0(:,1) = u0;

oo = []; % temporary

% Enter simulation
tic
for t=0:dt:tf
    time = t
    [U1,V1,Fu0,Fv0]=CONVEC(U0,V0,Fu0,Fv0,u0,dx,dy,t,dt); % Convection
    [U2,V2]=PRESS(dt,dx,dy,U1,V1); % Pressure
    [U0,V0]=VISC(dx,dy,u0,U2,V2,Re,dt); % Viscosity

    % velocity profile development visualization at x = L
    plot(U0(:,end),Y(:,end),'b','linewidth',1.5)
    xlim([0 1])
    drawnow
    xlabel('U');ylabel('y')

end
toc
```

```
velocity_profile(X,Y,U0,Re,H)

[Cf,C_Bf,Cf_rms,Cd_num_left,Cd_num_right,Cd_B_left,Cd_B_right] =
 skin_friction_and_drag(U0,Re,u0,NX,X,dy);

[BL] =  boundary_layer_thickness(NX,NY,X,Y,u0,U0,Re);

[Q,Qrms] = flow_rate(U0,Re,NX,NY,X,dy);

function [U1,V1,Fu1,Fv1] = CONVEC(U0,V0,Fu0,Fv0,u0,dx,dy,t,dt)
% calculates velocity due to convection
[NY,NX] = size(U0);
Fv1=zeros(NY,NX);
Fu1=zeros(NY,NX);

% Obtaining nonlinear terms
for i=2:NY-1
    for j=2:NX-1
        Fu1(i,j)= -U0(i,j)*(U0(i,j+1)-U0(i,j-1))/(2*dx) -
 V0(i,j)*(U0(i+1,j)-U0(i-1,j))/(2*dy);
        Fv1(i,j)= -U0(i,j)*(V0(i,j+1)-V0(i,j-1))/(2*dx) -
 V0(i,j)*(V0(i+1,j)-V0(i-1,j))/(2*dy);
    end
    % Right Boundary Conditions - Neumann
    Fu1(i,NX) = 0;
    Fv1(i,NX) = -U0(i,j)*(V0(i,NX)-V0(i,NX-1))/dx;
end

% Time advancement using Adams-Bashforth method
if t == 0
    U1 = U0 + dt*Fu1;
    V1 = V0 + dt*Fv1;
elseif t ~= 0
    U1 = U0 + dt*( (3/2)*Fu1 - (1/2)*Fu0 );
    V1 = V0 + dt*( (3/2)*Fv1 - (1/2)*Fv0 );
end
end

function [U2,V2]=PRESS(dt,dx,dy,U1,V1)
% Calculates velocity due to pressure
% Solves pressure field from poisson's equation


[NY, NX] =size(U1);
Npts = NY*NX;
Pressure=zeros(Npts,Npts);

%  LHS Pressure  Matrix : Boundary Conditions and Corner
 Approximations  %

%BOTTOM LEFT CORNER%

Pressure(1,1)  =  1;
```

```
%TOP LEFT CORNER%

Pressure(NY,NY-1) =  -1; % bottom
Pressure(NY,NY)   =  2;  % center
Pressure(NY,2*NY) =  -1; % right

%TOP RIGHT CORNER

Pressure(NY+(NX-1)*NY,NY-1+(NX-1)*NY)  =  -1; % bottom
Pressure(NY+(NX-1)*NY,NY+(NX-1)*NY)    =  2;  % center
Pressure(NY+(NX-1)*NY,NY+(NX-2)*NY)    =  -1; % left

%BOTTOM RIGHT CORNER

Pressure(1+(NX-1)*NY,2+(NX-1)*NY)  =  -1; % top
Pressure(1+(NX-1)*NY,1+(NX-1)*NY)  =  2;  % center
Pressure(1+(NX-1)*NY,1+(NX-2)*NY)  =  -1; % left

%%%% Internal LHS Pressure Coefficient Matrix %%%%
for j=2:NX-1 % X Direction
    for k=2:NY-1 % Y Direction

        Pressure(k+(j-1)*NY,k+(j-1)*NY)    =  -2*(1/dx^2+1/dy^2); %
 center
        Pressure(k+(j-1)*NY,k+(j-2)*NY)    =  1/dx^2;             %
 left
        Pressure(k+(j-1)*NY,k-1+(j-1)*NY)  =  1/dy^2;             %
 bottom
        Pressure(k+(j-1)*NY,k+(j)*NY)      =  1/dx^2;             %
 right
        Pressure(k+(j-1)*NY,k+1+(j-1)*NY)  =  1/dy^2;             %
 top


    end
end

for j=2:NX-1
    % Bottom Boundary

    Pressure(1+(j-1)*NY,1+(j-1)*NY) =  -2*(1/dx^2+1/dy^2); % center
    Pressure(1+(j-1)*NY,1+(j-2)*NY) =  1/dx^2;             % left
    Pressure(1+(j-1)*NY,2+(j-1)*NY) =  2/dy^2;             % top
    Pressure(1+(j-1)*NY,1+(j)*NY)   =  1/dx^2;             % right
    % Top Boundary
    Pressure(NY+(j-1)*NY,NY+(j-1)*NY)   =  -2*(1/dx^2+1/dy^2);  %
 center
    Pressure(NY+(j-1)*NY,NY+(j-2)*NY)   =  1/dx^2;              % left
    Pressure(NY+(j-1)*NY,NY-1+(j-1)*NY) =  2/dy^2;              %
 bottom
    Pressure(NY+(j-1)*NY,NY+(j)*NY)     =  1/dx^2;              %
 right

end
```

```
for k=2:NY-1
    % Right Boundary
    Pressure(k+(NX-1)*NY,k+(NX-1)*NY)   =  -2*(1/dx^2+1/dy^2); %
 center
    Pressure(k+(NX-1)*NY,k-1+(NX-1)*NY) =  1/dy^2;             %
 bottom
    Pressure(k+(NX-1)*NY,k+(NX-2)*NY)   =  2/dx^2;             % left
    Pressure(k+(NX-1)*NY,k+1+(NX-1)*NY) =  1/dy^2;             % top

    % Left Boundary
    Pressure(k,k)       =   -2*(1/dx^2+1/dy^2); % center
    Pressure(k,k-1)     =   1/dy^2;             % bottom
    Pressure(k,k+1)     =   1/dy^2;             % top
    Pressure(k,k+NY)    =   2/dx^2;             % right

end


% RHS Velocity Coefficient Matrix

sol=zeros(Npts,1);

for j=1:NX
% Bottom Boundary
    sol(1+(j-1)*NY)=0;
% Top Boundary
    sol(NY+(j-1)*NY)=0;
end

for k=1:NY
% Left Boundary
    sol(k)=0;
% Right Boundary
    sol(k+(NX-1)*NY)=0;
end

% Interior points
for j=2:NX-1
    for k=2:NY-1
        sol(k+(j-1)*NY) = (1/dt)*((U1(k,j+1)-U1(k,j))/dx+(V1(k+1,j)-
V1(k,j))/dy);
    end
end


sol(1)=1;

% FACT
%LU factorization of pressure matrix
[Pressure,mat]=fact(Npts,Pressure,zeros(Npts,1));


% SOLVE
```

```
%back substitution to solve for pressure
[pField]=solve(Npts,Pressure,mat,sol);
dpdx = zeros(NY,NX);
dpdy = zeros(NY,NX);
P = zeros(NY,NX);

for j=1:NX
    for k=1:NY
        P(k,j)=pField(k+(j-1)*NY);
    end
end


% constructing velocity
u0  = 1;
U2=zeros(NY,NX);
V2=zeros(NY,NX);
for k=2:NY-1
    for j=2:NX-1
        % discretized pressure-induced velocity
        dpdy(k,j)  =  0.5*(P(k+1,j) - P(k-1,j))/dy;
        dpdx(k,j)  =  0.5*(P(k,j+1) - P(k,j-1))/dx;
        V2(k,j)    =  V1(k,j) - dt*dpdy(k,j);
        U2(k,j)    =  U1(k,j) - dt*dpdx(k,j);

        % BOTTOM BOUNDARY / DIRICHLET
        U2(1,j) = U1(1,j);
        % TOP BOUNDARY / DIRICHLET
        U2(NY,j) = U1(NY,j);

    end

    % RIGHT BOUNDARY / NEUMANN
    U2(k,NX)=U1(k,NX);
%     U2(k,NX) = U2(k,NX-1);


end
 % LEFT BOUNDARY / DIRICHLET
U2(2:NY,1) = u0; % U1(k,1);
U2(1,:) = 0;
U2(NY,:) = u0;
end

function [U,V]=VISC(dx,dy,u0,U2,V2,Re,dt)
% Calculates velocity due to viscosity using the Crank-Nicolson scheme
% It is calculated twice, once in each direction

NY = size(V2,1);
NX = size(V2,2);
Npoints = NX*NY;
for direction = 1:2
    % x direction
    s=0.5*dt/Re/dx^2; % discretization parameter
```

```
    leftX =zeros(Npoints,Npoints);
    leftY = leftX;
    u3 = zeros(1,Npoints);
    v3 = u3;

    if direction == 2
        % y direction
        U2 = U;
        V2 = V;
        s=0.5*dt/Re/dy^2; % discretization parameter
    end

%%%%% Left Term Matrix
% Interior values, coefficients of unknown values
for j=2:NX-1
    for i=2:NY-1
        if direction == 1
            leftX((j-1)*NY+i,(j-1)*NY+i)   =  (1+2*s); % center
            leftX((j-1)*NY+i,j*NY+i)       =  -s; % top/ right
            leftX((j-1)*NY+i,(j-2)*NY+i)   =  -s; % bottom / left
        elseif direction == 2
            leftX((j-1)*NY+i,(j-1)*NY+i)   =  (1+2*s); % center
            leftX((j-1)*NY+i,(j-1)*NY+i+1) =  -s; % top/ right
            leftX((j-1)*NY+i,(j-1)*NY+i-1) =  -s; % bottom / left
        end
    end
end

for j=1:NX
    %Bottom Boundary
    leftX((j-1)*NY+1,(j-1)*NY+1)=1;
    %Top Boundary
    leftX(NY+(j-1)*NY,(j-1)*NY+NY)=1;
end

for i=2:NY-1
    %Left Boundary
    leftX(i,i)=1;
    %RightBoundary
    if direction==1
        leftX(i+(NX-1)*NY,i+(NX-1)*NY)   =  1+2*s;
        leftX(i+(NX-1)*NY,i+(NX-2)*NY)   =  -2*s;
    elseif direction == 2
        leftX(i+(NX-1)*NY,i+1+(NX-1)*NY) =  -s;
        leftX(i+(NX-1)*NY,i+(NX-1)*NY)   =  1+2*s;
        leftX(i+(NX-1)*NY,i-1+(NX-1)*NY) =  -s;
    end
end

size(leftY);

leftY;
leftY=leftX;
%Right Boundary
```

```
for k=2:NY-1
    leftY(k+(NX-1)*NY,k+(NX-1)*NY)=1;
end

% Right hand side construction
%Internal Points, known values
for j=2:NX-1
    for i=2:NY-1
        if direction == 2 % Y
            v3(i+(j-1)*NY)=s*V2(i+1,j)+(1-2*s)*V2(i,j)+s*V2(i-1,j);
            u3(i+(j-1)*NY)=s*U2(i+1,j)+(1-2*s)*U2(i,j)+s*U2(i-1,j);
        elseif direction == 1 % X
            v3(i+(j-1)*NY)=s*V2(i,j+1)+(1-2*s)*V2(i,j)+s*V2(i,j-1);
            u3(i+(j-1)*NY)=s*U2(i,j+1)+(1-2*s)*U2(i,j)+s*U2(i,j-1);

        end
    end
end

%Left Boundary
for i=2:NY-1
    u3(i)=u0;
    v3(i)=0;
%Right Boundary
    if direction==1
        u3(i+(NX-1)*NY)=(1-2*s)*U2(i,NX)+2*s*U2(i,NX-1);
        v3(i+(NX-1)*NY)=0;
    elseif direction == 2
        u3(i+(NX-1)*NY)=s*U2(i+1,NX)+(1-2*s)*U2(i,NX)+s*U2(i-1,NX);
        v3(i+(NX-1)*NY)=s*V2(i+1,NX)+(1-2*s)*V2(i,NX)+s*V2(i-1,NX);
    end
end

for j=1:NX
    %Top Boundary
    u3((j-1)*NY+NY)=u0;
    v3(NY+(j-1)*NY)=0;
    %Bottom Boundary
    u3(1+NY*(j-1))=0;
    v3(1+NY*(j-1))=0;
end

% FACT
% LU factorization of LHS Matrix
[leftX,matx]=fact(Npoints,leftX,zeros(Npoints,1));
[leftY,maty]=fact(Npoints,leftY,zeros(Npoints,1));


% SOLVE
% back substitution to solve for velocity field
[u_field]=solve(Npoints,leftX,matx,u3);
[v_field]=solve(Npoints,leftY,maty,v3);

% Constructs Flow Field
```

```
U=zeros(NY,NX);

for j=1:NX
    for i=2:NY-1

        U(i,j)=u_field(i+(j-1)*NY);
    end

%     U(NY,j) = u0;%
%     U(NY,j) = u_new(end);%0;%u0; % TOP BC
end
% Left BC
U(2:NY,1) = u0;
% Bottom BC
U(1,:) = 0;
% Top BC
U(NY,:) = u0;

V=zeros(NY,NX);
for j=2:NX-1
    for i=2:NY-1
        V(i,j)=v_field(i+(j-1)*NY);
    end
end


end
end

function [] = velocity_profile(X,Y,U0,Re,H)

ymax = H;

figure, set(gcf, 'color', 'w'), box on, hold on
for i = 1:51

    if mod(X(1,i),0.2) == 0

        for j = 1:101
            if Y(j,end) > 0.1*ymax
                break
            end
            plot(X(1:j,i)+ 0.2*U0(1:j,i),Y(1:j,i),'linewidth',1.5)
            xlim([0 1.2]); ylim([0 0.1*ymax])
        end
    end
end
xlabel('x');ylabel('y')
title(strcat('Velocity profiles at different x (Re =
 ',num2str(Re),')'))

end
```

```
function
 [Cf,C_Bf,Cf_rms,Cd_num_left,Cd_num_right,Cd_B_left,Cd_B_right] =
 skin_friction_and_drag(U0,Re,u0,NX,X,dy)

nu = mean(u0*X(1,:)/Re);

tau0 = zeros(1,NX);

for i = 1:NX
    tau0(i) = nu*(U0(2,i)-U0(1,i))/dy;
end

Cf = tau0/(0.5*u0^2);
Cf_rms = rms(Cf); % computing rms error
Re_x = u0*X(1,:)/nu;
C_Bf = 0.664*Re_x.^(-0.5);

figure, set(gcf,'color','w'), box on, hold on
plot(X(1,:),Cf,'b','linewidth',1.5)
plot(X(1,:),C_Bf,'r','linewidth',1.5)
xlabel('x'); ylabel('Skin friction coefficient C_f')
legend('Numerical Solution','Blasius Similarity Solution')
title(strcat('Skin friction coefficient (Re = ',num2str(Re),')'))

%compute rms error

% 0 < x < 0.5
Cd_num_left = sum(Cf(2:round(end/2)));
Cd_B_left = sum(C_Bf(2:round(end/2)));

% 0.5 < x < 1
Cd_num_right = sum(Cf(round(end/2):end));
Cd_B_right = sum(C_Bf(round(end/2):end));

end

function [BL] = boundary_layer_thickness(NX,NY,X,Y,u0,U0,Re)

nu = zeros(1,size(X,2));
BL = zeros(1,NX); %boundary layer thickness
for i = 1:NX
    for j = NY:-1:1%1:NY
        if U0(j,i) < 0.99*u0
            BL(i) = Y(j,i);
            nu(i) = u0*X(j,i)/Re;
            break
        end
    end
end
nu = mean(nu);

figure, set(gcf,'color','w'), hold on,box on
plot(X(1,:),BL,'b','linewidth',1.5)
plot(X(1,:),4.9*sqrt(nu*X(1,:)/u0),'r','linewidth',1.5)
```

```
legend('numerical solution','blasius solution','location','nw')
xlabel('x'),ylabel('\delta_9_9')
title(strcat('Boundary Layer Thickness (Re = ',num2str(Re),')'))

end

function [Q,Qrms] = flow_rate(U0,Re,NX,NY,X,dy)



Q=zeros(NX,1);
for i=1:NX
    for j=2:NY-1
        Q(i)=U0(j,i)*dy+Q(i); % flow rate
    end
end

Qrms = rms(Q); % computing rms error


% end

figure, set(gcf,'color','w'),box on, hold on
plot(X(1,:),Q,'b','linewidth',1.5)
plot(X(1,:),Q(1)*ones(size(Q)),'r','linewidth',1.5)
xlabel('x');ylabel('Flow rate Q')
title(strcat('Flow rate (Re = ',num2str(Re),')'))
legend('Q(x)','Q(x=0)','location','best')

end
```

```
Error using dbstatus
Error: File: /Volumes/Lexar/AME535Anthony/superscript.m Line: 40
 Column: 30
Function definitions are not permitted in this context.
```

Published with MATLAB® R2016a