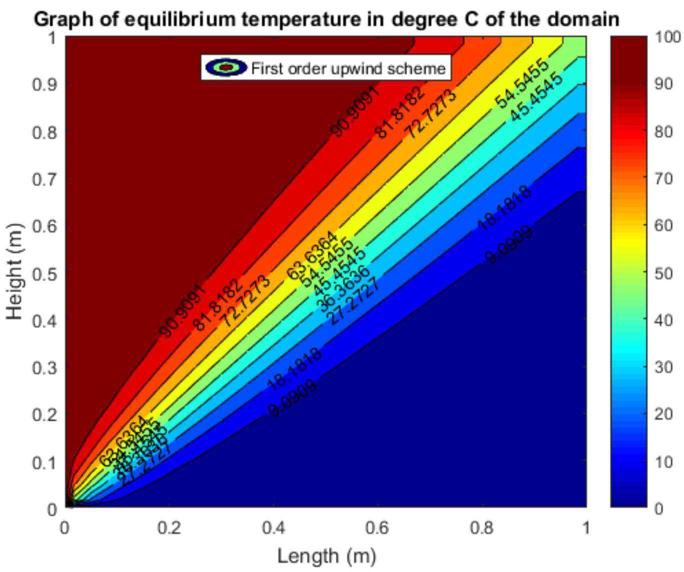
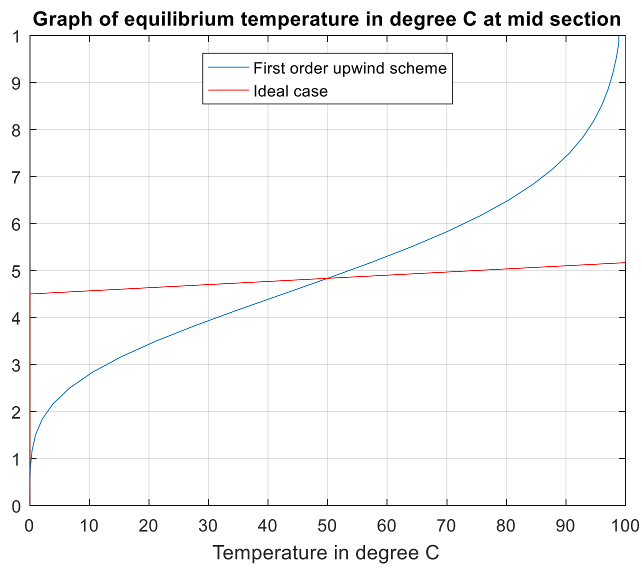
4/9/2018

ASSIGNMENT 3

COMPUTATIONAL FLUID DYNAMICS AND HEAT TRANSFER





173109003 | Sanit Prashant Bhatkar

Computational Fluid Dynamics and Heat Transfer (ME 415), Autumn 2018

Assignment 3: Unsteady Heat Convection

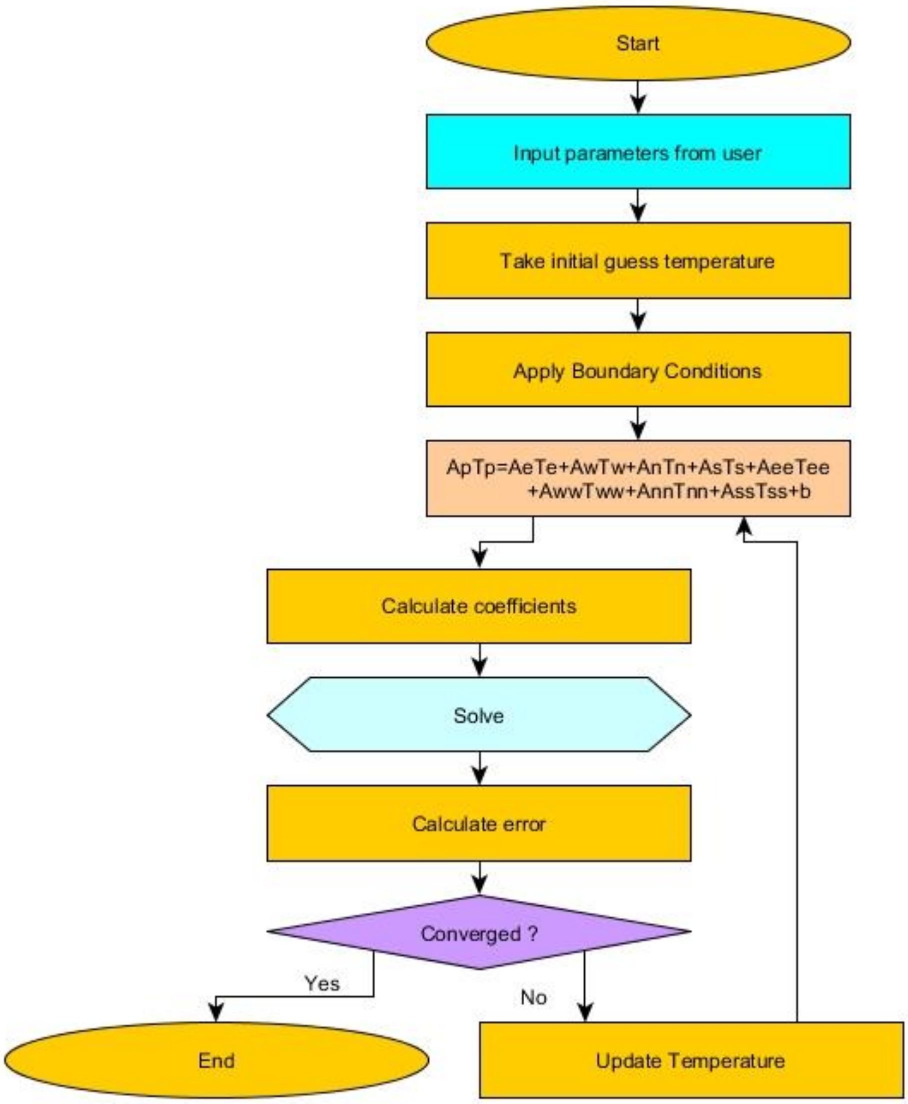
Name: Sanit Prashant Bhatkar

Roll No: 173109003

Problem 1:

Consider a 2D Cartesian (x,y) computational domain of size L=1m and H=1 m, for CHA of a fluid (ρ=1000 kg/m3 and cp=4180 W/m.K) moving with a uniform velocity u=v=1 m/s and an initial temperature of 50^0C. The bottom and left boundary of the domain is subjected to 0^0C and 100^0C, respectively. Run the code for three different advection schemes: (a) FOU, (b) SOU and (c) QUICK. Take the maximum number of grid points in x-and y-direction as imax = jmax=32 and convergence criteria (εst) as 0.000001. Use the stopping criterion for the unsteadiness, with ΔTc = 100^oC.

Algorithm



CFDHT

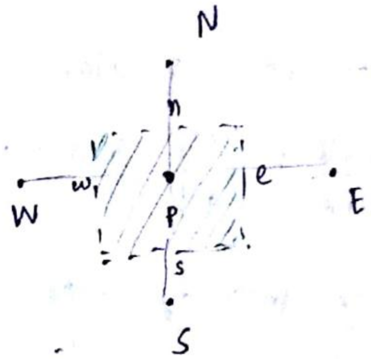
Assignment 3 173109003

Q.1. 2D computational Heat advection

General Equation

$$\frac{\partial}{\partial t}(\rho \phi_i) + \frac{\partial}{\partial x_j}(\rho u_j \phi_i) = \frac{\partial}{\partial x_j}\left(\Gamma \frac{\partial \phi}{\partial x_j}\right) + S$$

Advection Diffusion



$\Gamma = 0 \Rightarrow$ Diffusion is not present.

$S = 0 \Rightarrow$ No heat generation.

$$\boxed{\frac{\partial}{\partial t}(\rho \phi_i) + \frac{\partial}{\partial x_j}(\rho u \phi) = 0} \quad \text{—(A)}$$

Integrating (A) along the domain

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial}{\partial x}(\rho u \phi) + \frac{\partial}{\partial y}(\rho v \phi) = 0.$$

$$\int_t^{t+\Delta t}\int_s^n\int_w^e \frac{\partial}{\partial t}(\rho\phi)\,dx\,dy\,dt + \int_t^{t+\Delta t}\int_s^n\int_w^e \frac{\partial}{\partial x}(\rho u \phi) + \int_t^{t+\Delta t}\int_w^e\int_s^n \frac{\partial}{\partial y}(\rho v \phi) = 0.$$

$$\boxed{\frac{\rho_P \phi_P^n - \rho_P \phi_P^0}{\Delta t}\Delta x \Delta y + [\rho_e u_e \phi_e - \rho_w u_w \phi_w]\Delta y + [\rho_n v_n \phi_n - \rho_s v_s \phi_s]\Delta x = 0.}$$

For making general derivation, ~~to~~ diffusion term ~~to~~ is also included.

$\Rightarrow$ Diffusion term

$$\Delta y\left[\left(\Gamma\frac{d\phi}{dx}\right)_e-\left(\Gamma\frac{d\phi}{dx}\right)_w\right]+\left[\left(\Gamma\frac{d\phi}{dy}\right)_n-\left[\Gamma\frac{d\phi}{dy}\right]_s\right]\Delta x \quad -(2)$$

Say $\boxed{J=\rho u\phi-\Gamma\frac{d\phi}{dx}}$

Continuity equation

$\boxed{\frac{d\rho u}{dx}+\frac{d\rho v}{dy}=0.}$ for steady state.

for unsteady state

$$\boxed{\frac{(\rho_P^n-\rho_P^o)\Delta x\Delta y}{\Delta t}+\left[(\rho u)_e-(\rho u)_w\right]\Delta y+\left[(\rho v)_n-(\rho v)_s\right]\Delta x=0.} \quad -(3)$$

From (1), (2), (3).

$$\left[\phi_P^n-\phi_P^o\right]\frac{\rho_P^o\Delta x\Delta y}{\Delta t}+\left[J_e-F_e\phi_P\right]-\left[J_w-F_w\phi_P\right]$$
$$+\left[J_n-F_n\phi_P\right]-\left[J_s-F_s\phi_P\right]=0.$$

Looking for the form of type.

$$\boxed{a_P\phi_P=a_E\phi_E+a_W\phi_W+a_N\phi_N+a_S\phi_S+b.}$$

Generalized formulation for the scheme of order '1' is made.

$[\![\ ]\!]$ represents maximum of the quantity included in double brackets.

$$a_E = D_e\, A(|P_e|) + [[-F_e, 0]]$$

$$a_W = D_w\, A(|P_w|) + [[F_w, 0]]$$

$$a_N = D_N\, A(|P_N|) + [[-F_N, 0]]$$

$$a_S = D_S\, A(|P_S|) + [[F_S, 0]]$$

$$a_P^0 = \frac{\rho_P^0 \Delta x \Delta y}{\Delta t}$$

$$b = a_P^0 \phi_P^0$$

$$a_P = a_E + a_W + a_N + a_S + a_P^0$$

(H)

Used for first order schemes in the CODE

$$D_e = \frac{\Gamma_e \Delta y}{\delta x_e} \qquad F_e = (\rho u)_e \Delta y$$

③

$$D_w = \frac{\Gamma_w \Delta y}{\delta x_w} \qquad F_w = (\rho u)_w \Delta y$$

$$D_n = \frac{\Gamma_n \Delta x}{\delta y_n} \qquad F_n = (\rho v)_n \Delta x$$

$$D_s = \frac{\Gamma_s \Delta x}{\delta y_s} \qquad F_s = (\rho v)_s \Delta x$$

$\Gamma = \frac{k}{C_p}$; Peclet Number $\boxed{P_e = \frac{F}{D}}$

Q. 1. Formulation for QUICK and SOU

~~Both the~~ QUICK schemes ~~was~~ derived in class and coefficients were known: Based on the coefficients, generalized formulation for higher order scheme is proposed.

$$\left.\begin{aligned}
a_E &= D_e - W_1 [[F_e, 0]] - W_2 [[-F_e, 0]] + W_3 [[-F_w, 0]] \\
a_{EE} &= (W_1 + W_2 + W_3) [[-F_e, 0]] \\
a_W &= D_W + W_2 [[F_w, 0]] - W_3 [[F_e, 0]] + W_1 [[-F_w, 0]] \\
a_{WW} &= W_3 [[F_w, 0]]
\end{aligned}\right\} \quad (G)$$

a_N can be obtained by replacing E of a_E

a_S can be obtained by replacing W of a_W

Weights

W_1	W_2	W_3	Scheme
0	3/2	-1/2	SOU
3/8	6/8	-1/8	QUICK

Grid details and the implemented boundary condition

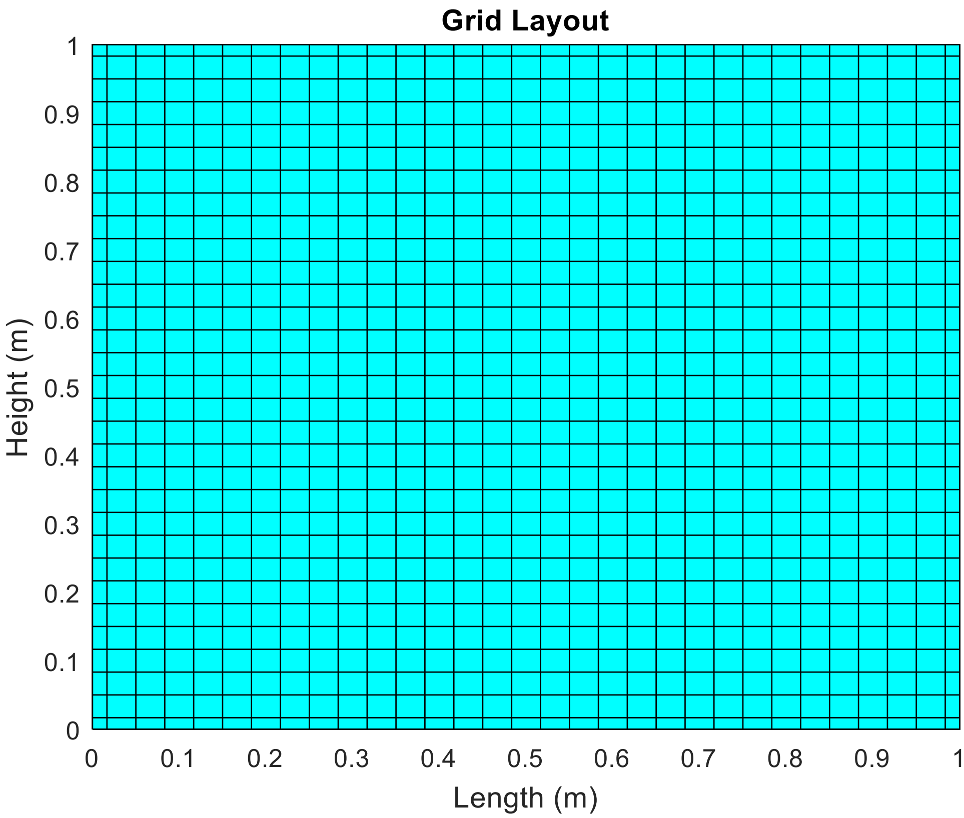


Fig 1. Generated grid for the given Cartesian geometry

Boundary Conditions

- On the left boundary uniform temperature 373K is applied
- On the bottom boundary uniform temperature 273K is applied

Code of the problem

```
%==============================================================%
% Assignment 3 CHDHT ME 415
% Problem Number 1. Unsteady Heat Convection
% Progarm uses FVM to solve the problem of 2D convection
% Heat generation is zero
% Designed only for rectangular co-ordinate system
%
% AUTHOR:
% Sanit P. Bhatkar (173109003@iitb.ac.in)
% Roll No: 173109003
% Place: IIT BOMBAY.
%==============================================================%

%% Input of Variables

%===============================INPUT OF VARIABLES==========================
clc
clearvars
fprintf('\n============ Unsteady Heat Convection ============');
fprintf('\n\n---------------------- INPUT -----------------------\n');
l=1;
h=1;
% dl=0.05;
% dh=0.05;
%k is zero as the problem is only of advection
k=0;
eps=1e-6;
u=1;
v=1;
rho=1000;
cp=4180;
fprintf('\nEnter the scheme to be used for the problem \n\n 1) First Order Upwind
Scheme\n 2) Second Order Upwind Scheme\n 3) QUICK Scheme \n');
choice=input('\nEnter scheme number: ');


% k_cond=input('\nInput conductivity of the material: ');
%------------Geometry definition--------------

% fprintf('\nGeometry Parameters -->\n');
% l=input('\nLength of column in m: ');
% h=input('Height of column in m: ');
% dl=input('Grid size along length in m: ');
% dh=input('Grid size along height in m: ');

%------------Boundary condition input---------

% tbn=400;
tbs=273;
% tbe=30;
tbw=373;

% fprintf('\nBoundary Conditions -->\n');
% tbn=input('\nBoundary condition at north boundary in degree C:  ');
% tbs=input('Boundary condition at south boundary in degree C:  ');
% tbe=input('Boundary condition at east boundary in degree C:  ');
% tbw=input('Boundary condition at west boundary in degree C:  ');

%% Grid formation

%============================FINITE VOLUME METHOD==========================
```

```matlab
%STEP 1: Divide domain into finite sized subdomain called control volumes

%Our domain is uniform
%But boundary and the first point grid space is different
%That is why separate calculation is done
%sx represents the number of points in a grid in x direction
%NOTE: dl has to be some multiple of length

sx=32;
dl=l/(sx-2);
x(1,1)=0;
x(sx,1)=l;
x(2,1)=x(1,1)+(dl/2);
for m=3:sx-1
    x(m,1)=x(m-1,1)+dl;
end

%sy represents the number of points in a grid in y direction

sy=32;
dh=h/(sy-2);
y(1,1)=0;
y(sy,1)=h;
y(2,1)=y(1,1)+(dh/2);

for m=3:sy-1
    y(m,1)=y(m-1,1)+dh;
end


%% Boundary Conditions and Initial Condition

T=zeros(sy,sx);
T(1:sy,1:sx)= 50+273;
T_ini=T(1,1);

%Application of boundary conditions
% T(1,1:sx)=tbn;
T(sy,1:sx)=tbs;
% T(1:sy,sx)=tbe;
T(1:sy,1)=tbw;
T(sy,1)=(tbs+tbw)/2;

%Definition of convergence Formula for residual will be used in the code

%eps=input('\nEnter minimum convergence error: ');
%eps=10^-(any value);


%STEP 2: Integrate governing equation over boundaries of control volume

%STEP 3: Profile assumption

%We will get equation of type
%ApTp=AeTe+AwTw+AnTn+AsTs+B
%Assuming isotropic material and uniform control volume with no heat generation

%% Common properties


%Conductivity at each boundary
```

```
ke=k; kw=k; kn=k; ks=k;
%Velocity at each boundary
ue=u; uw=u; vn=v; vs=v;

% Diffusion Coefficient
game=ke/cp;  gamw=kw/cp;
gamn=kn/cp;  gams=ks/cp;

% Advection strength
Fe=rho*ue*dh;
Fw=rho*uw*dh;
Fn=rho*vn*dl;
Fs=rho*vs*dl;

%The syntax is used to calculate rounded time step
dt=round((min((dl/(2*u)),(dh/(2*v))))*1000)/1000;


%% Genrealized formulation for first order Schemes

%% FOU scheme

if choice == 1


To=T;
itr=0;
mxe=10;


while mxe>eps


%% Inner nodes Formulation FOU

for i=sy-1:-1:2

for j=2:sx-1

    %dxe is distance from point P to east point
    %dxw is distance from point P to west point
    %dyn is distance from point P to north point
    %dys is distance from point P to south point

    dxe=x(j+1,1)-x(j,1);
    dxw=x(j,1)-x(j-1,1);
    dyn=y(i,1)-y(i-1,1);
    dys=y(i+1,1)-y(i,1);

    % Diffusion strength
    De=game*dh/dxe;
    Dw=gamw*dh/dxw;
    Dn=gamn*dl/dyn;
    Ds=gams*dl/dys;

    % Peclet Number
    Pe=Fe/De;
    Pw=Fw/Dw;
    Pn=Fn/Dn;
    Ps=Fs/Ds;
```

```matlab
    %% Change the formula for other first order schemes

    % A(|P|)=1+max(-Peclet,0)  for FOU
    % Class notes page 12 (Reference)
    % S.V.Patankar Table 5.1

    % This defined for FOU

    Ae=1+max(-Pe,0);
    Aw=1+max(-Pw,0);
    An=1+max(-Pn,0);
    As=1+max(-Ps,0);



    %% General grid formula FOU

    ae=De*Ae+max(-Fe,0);
    aw=Dw*Aw+max(Fw,0);
    an=Dn*An+max(-Fn,0);
    as=Ds*As+max(Fs,0);
    apo=rho*dl*dh/dt;
    b=apo*To(i,j);

    ap=ae+aw+an+as+apo;
    T(i,j)=(ae*T(i,j+1)+aw*T(i,j-1)+an*T(i-1,j)+as*T(i+1,j)+b)/ap;


end

%% East Boundary FOU

% Calculation for EAST boundary

   T(i,sx)=T(i,sx-1);

end

%% North boundary FOU

%Calculations for North boundary
i=1;

for j=2:sx-1

   T(i,j)=T(i+1,j);

end


%% Stopping Criteria for FOU

%ABSOLUTE ERROR calculation at every point
error=(l*abs(T-To))/(u*dt*100);
mxe=max(max(error));

itr=itr+1;
To=T;


end
```

```
end


%% Genrealized formulation for higher order Schemes

%% SOU scheme

if choice == 2

% SOU scheme

To=T;
itr=0;
mxe=10;
dum=zeros(sy+2,sx+2);
dum(2:sy+1,2:sx+1)=T;
dum(end,1:sx+1)=tbs;


while mxe>eps

dum(2:sy+1,2:sx+1)=T;

for i=sy-1:-1:2

for j=2:sx-1


    %dxe is distance from point P to east point
    %dxw is distance from point P to west point
    %dyn is distance from point P to north point
    %dys is distance from point P to south point

    dxe=x(j+1,1)-x(j,1);
    dxw=x(j,1)-x(j-1,1);
    dyn=y(i,1)-y(i-1,1);
    dys=y(i+1,1)-y(i,1);

    % Diffusion strength
    De=game*dh/dxe;
    Dw=gamw*dh/dxw;
    Dn=gamn*dl/dyn;
    Ds=gams*dl/dys;


    %% Inner nodes formula SOU


    ae=De+(-0)*max(Fe,0)+(-3/2)*max(-Fe,0)+(-1/2)*max(-Fw,0);
    aee=max(-Fe,0);
    aw=Dw+(3/2)*max(Fw,0)+(1/2)*max(Fe,0)+(0)*max(-Fw,0);
    aww=(-1/2)*max(Fw,0);
    an=Dn+(-0)*max(Fn,0)+(-3/2)*max(-Fn,0)+(-1/2)*max(-Fs,0);
    ann=max(-Fn,0);
    as=Ds+(3/2)*max(Fs,0)+(1/2)*max(Fn,0)+(0)*max(-Fs,0);
    ass=(-1/2)*max(Fs,0);
    apo=rho*dl*dh/dt;
    b=apo*To(i,j);
```

```
    % Very long 'if' syntax is used to avoid trouble in code execution at
    % boundaries

    if i==sy-1
        ass=0;
    end

    if i==2
        ann=0;
    end

    if j==2
        aww=0;
    end

    if j==sx-1
        aee=0;
    end

    %% SOU scheme grid formula

    ap=ae+aw+an+as+apo+aee+aww+ann+ass;
    ii=i+1;
    jj=j+1;
    dum(ii,jj)=(ae*dum(ii,jj+1)+aw*dum(ii,jj-1)+an*dum(ii-
1,jj)+as*dum(ii+1,jj)+aee*dum(ii,jj+2)+aww*dum(ii,jj-2)+ann*dum(ii-
2,jj)+ass*dum(ii+2,jj)+b)/ap;
    T=dum(2:sy+1,2:sx+1);
end

end

%% East Boundary SOU

for i=sy-1:-1:2

% Calculation for EAST boundary
    T(i,sx)=T(i,sx-1);

end

%% North boundary SOU

%Calculations for North boundary
i=1;

for j=2:sx-1

     T(i,j)=T(i+1,j);

end


%% Stopping Criteria SOU

%ABSOLUTE ERROR calculation at every point
error=(l*abs(T-To))/(u*dt*100);
mxe=max(max(error));

itr=itr+1;
To=T;
```

```
end

end

%% QUICK scheme

if choice == 3

% QUICK scheme

To=T;
itr=0;
mxe=10;
dum=zeros(sy+2,sx+2);
dum(2:sy+1,2:sx+1)=T;
dum(end,1:sx+1)=tbs;


while mxe>eps

dum(2:sy+1,2:sx+1)=T;

for i=sy-1:-1:2

for j=2:sx-1


    %dxe is distance from point P to east point
    %dxw is distance from point P to west point
    %dyn is distance from point P to north point
    %dys is distance from point P to south point

    dxe=x(j+1,1)-x(j,1);
    dxw=x(j,1)-x(j-1,1);
    dyn=y(i,1)-y(i-1,1);
    dys=y(i+1,1)-y(i,1);

    % Diffusion strength
    De=game*dh/dxe;
    Dw=gamw*dh/dxw;
    Dn=gamn*dl/dyn;
    Ds=gams*dl/dys;


    %% Inner nodes formula QUICK

    %Malasekra page 158

    ae=De+(-3/8)*max(Fe,0)+(-6/8)*max(-Fe,0)+(-1/8)*max(-Fw,0);
    aee=max(-Fe,0);
    aw=Dw+(6/8)*max(Fw,0)+(1/8)*max(Fe,0)+(3/8)*max(-Fw,0);
    aww=(-1/8)*max(Fw,0);
    an=Dn+(-3/8)*max(Fn,0)+(-6/8)*max(-Fn,0)+(-1/8)*max(-Fs,0);
    ann=max(-Fn,0);
    as=Ds+(6/8)*max(Fs,0)+(1/8)*max(Fn,0)+(3/8)*max(-Fs,0);
    ass=(-1/8)*max(Fs,0);
    apo=rho*dl*dh/dt;
    b=apo*To(i,j);


    % Very long if syntax is used to avoid trouble in code execution at
    % boundaries
```

```
    if i==sy-1
        ass=0;
    end

    if i==2
        ann=0;
    end

    if j==2
        aww=0;
    end

    if j==sx-1
        aee=0;
    end

    %% QUICK scheme grid formula

    ap=ae+aw+an+as+apo+aee+aww+ann+ass;
    ii=i+1;
    jj=j+1;
    dum(ii,jj)=(ae*dum(ii,jj+1)+aw*dum(ii,jj-1)+an*dum(ii-
1,jj)+as*dum(ii+1,jj)+aee*dum(ii,jj+2)+aww*dum(ii,jj-2)+ann*dum(ii-
2,jj)+ass*dum(ii+2,jj)+b)/ap;
    T=dum(2:sy+1,2:sx+1);
end

end

%% East Boundary QUICK

for i=sy-1:-1:2

% Calculation for EAST boundary
    T(i,sx)=T(i,sx-1);

end

%% North boundary QUICK

%Calculations for North boundary
i=1;

for j=2:sx-1

     T(i,j)=T(i+1,j);

end


%% Stopping Criteria QUICK

%ABSOLUTE ERROR calculation at every point
error=(l*abs(T-To))/(u*dt*100);
mxe=max(max(error));

itr=itr+1;
To=T;

end
```

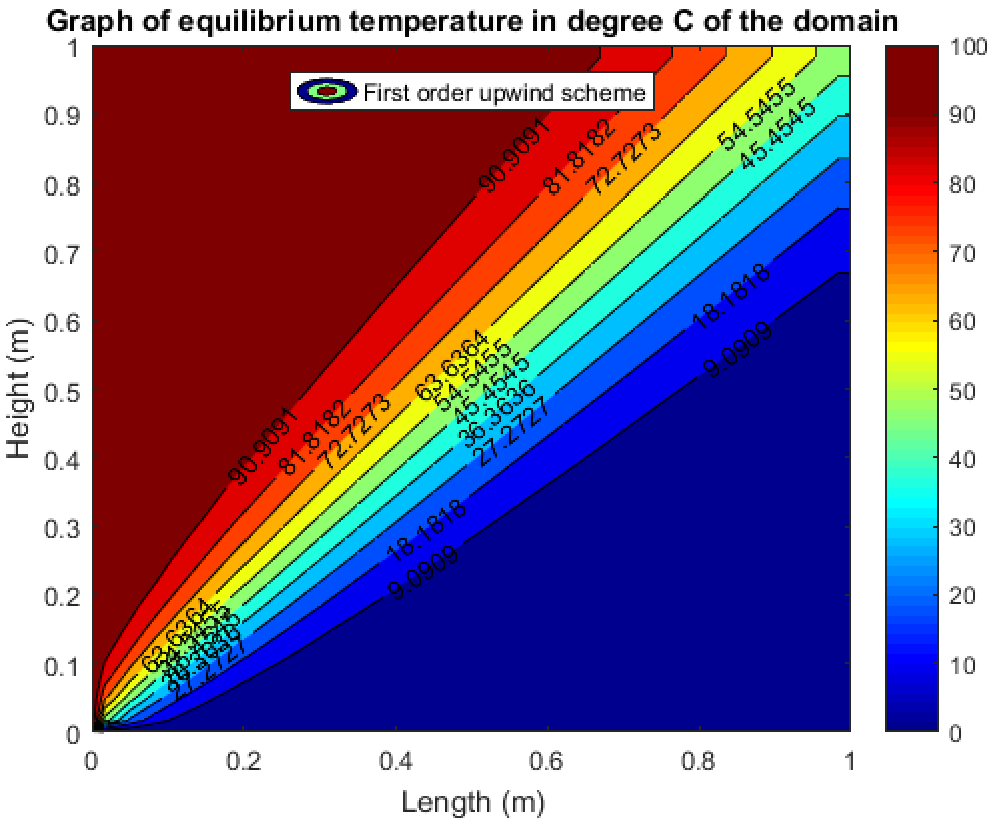
```
end

%% Output %%

%For conversion from kelvin to degree C
T=T-273;

% Calculation for ideal scheme
T_ideal=zeros(sx,sy);
T_ideal(end,1)=50;
r=sx-1;
for i=1:sy
    for j=1:r

        T_ideal(i,j)=100;
        T_ideal(i,r+1)=50;
    end
 r=r-1;
end

% Grid plot
figure (1)
r=ones(sy,sx);
map = [0,1,1];
colormap(map)
pcolor(x,y,r)
title('Grid Layout');
xlabel('Length (m)');
ylabel('Height (m)');


%Plot of the temperature contours as the output.

figure(2)
colormap jet
contourf(x,y,flipud(T),10,'showtext','on')
c=colorbar;
c.Limits= [min(min(T)) max(max(T))];
title('Graph of equilibrium temperature in degree C of the domain')
xlabel('Length (m)');
ylabel('Height (m)');

if choice == 1
    legend('First order upwind scheme','Location','North');
elseif choice == 2
    legend('Second order upwind scheme','Location','North');
else
    legend('QUICK scheme','Location','North');
end

%Plot at middle length of domain

figure (3)
plot(T(sx:-1:1,round(sx/2)),y)
grid on
title('Graph of equilibrium temperature in degree C at mid section')
ylabel('Height in m');
xlabel('Temperature in degree C');
hold on
plot(T_ideal(sx:-1:1,round(sx/2)),y,'r')
hold off
```

```
if choice == 1
    legend('First order upwind scheme','Ideal case','Location','North');
elseif choice == 2
    legend('Second order upwind scheme','Ideal case','Location','North');
else
    legend('QUICK scheme','Ideal case','Location','North');
end

% % Ideal case graph
% figure
% colormap jet
% contourf(x,y,flipud(T_ideal),10,'showtext','on')
% c=colorbar;
% c.Limits= [min(min(T_ideal)) max(max(T_ideal))];
% title('Graph of equilibrium temperature in degree C of the domain')
% xlabel('Length (m)');
% ylabel('Height (m)');
% legend('Ideal scheme','Location','North');


%_______________________________________________________________________________
```

Results and Discussion

A. Steady State Temperature Contours for different Advection Schemes

1. First Order Upwind Scheme



Discussion

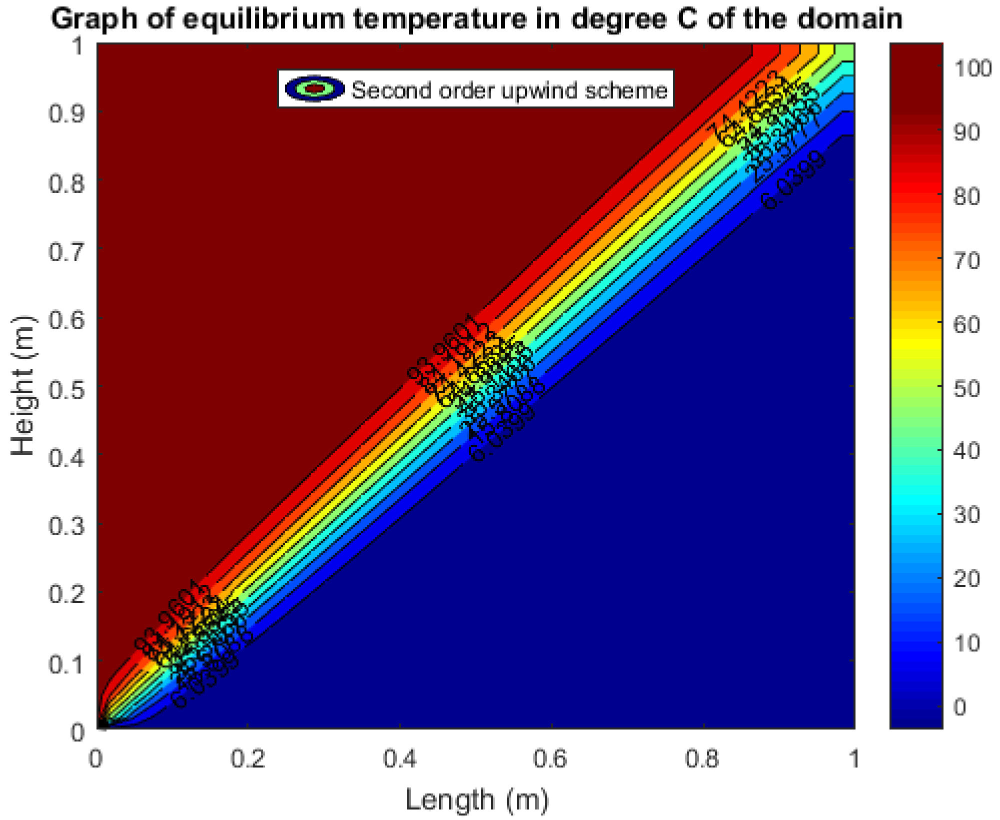
Temperature profile for the mid-section is plotted at steady state.

Order of false diffusion is FOU > SOU >QUICK. The trend with ideal scheme is also plotted and indicated by red line in the graph. QUICK scheme gives closest result.

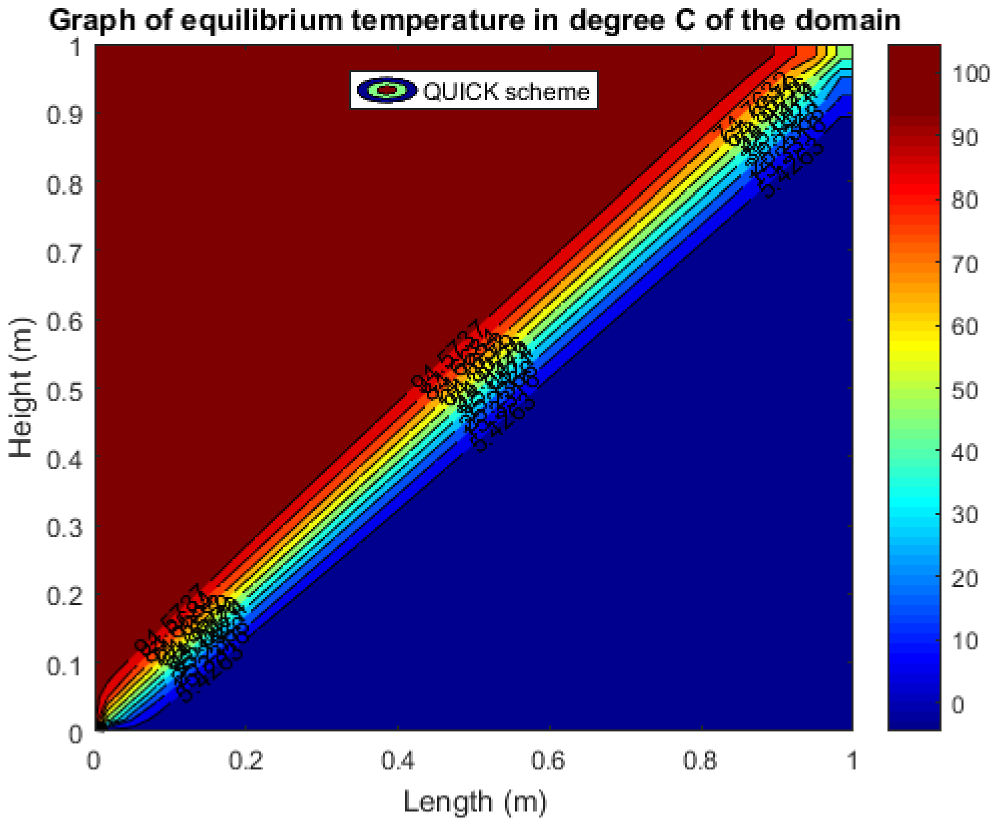
It was also found from temperature values that, FOU gave physically realistic solution. The physically realistic result simply means negative values were not present. For QUICK and SOU at some points, negative values were present which violates the physics of the problem.

Order of accuracy (closeness to the ideal scheme) is QUICK > SOU > FOU.

2. Second Order Upwind Scheme

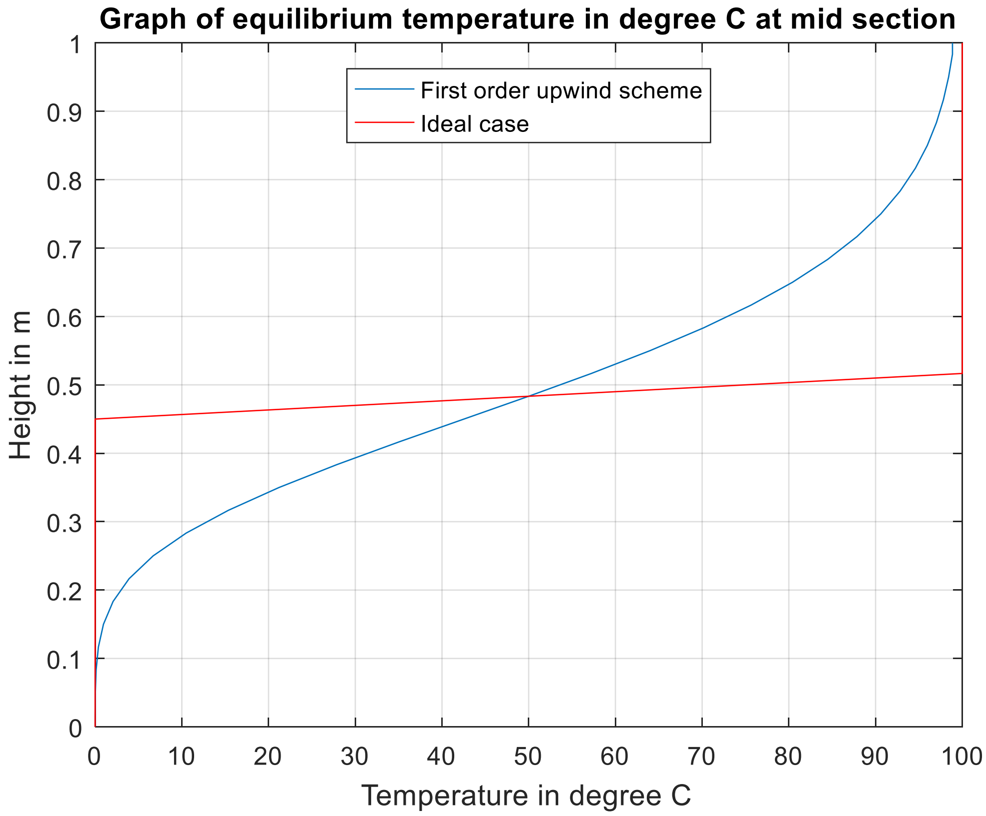


3. QUICK scheme

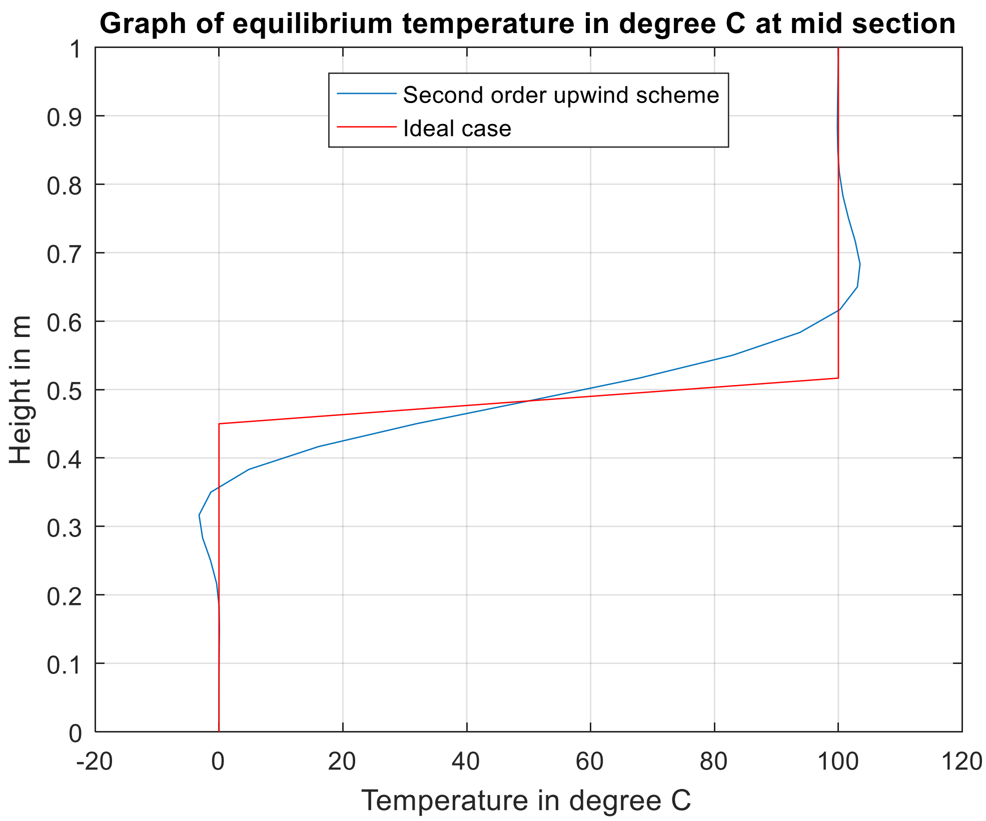


B. Temperature profile at the vertical centreline (x=0.5) for the different advection schemes

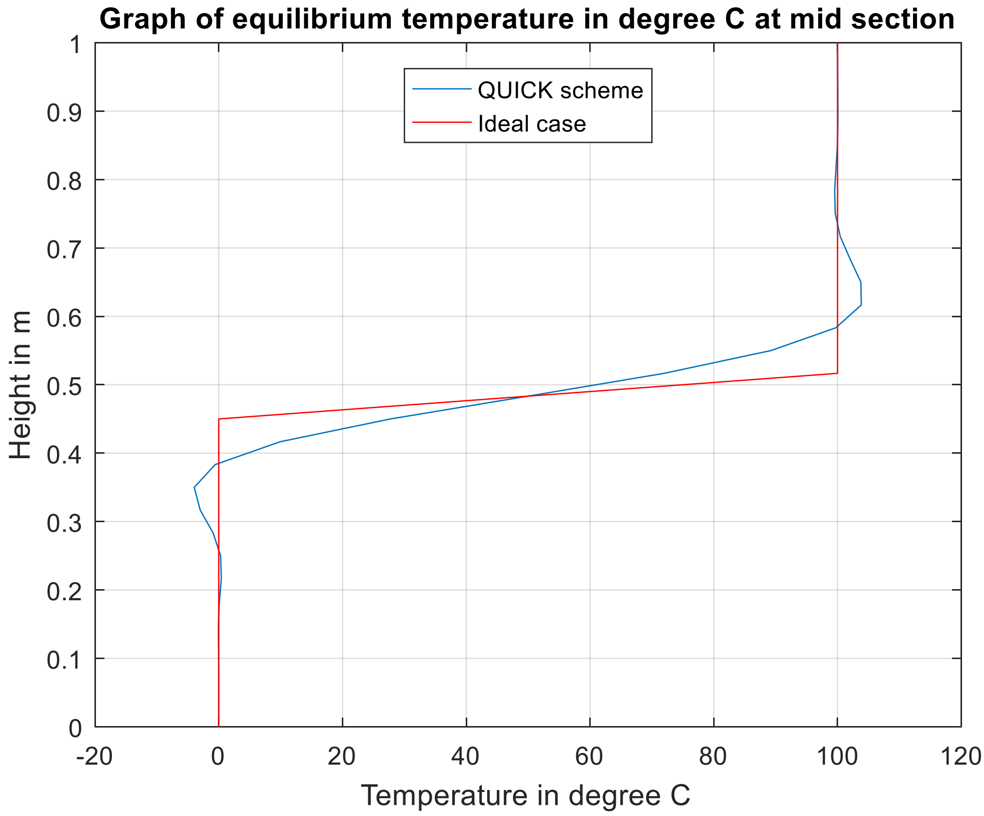
1. First Order Upwind Scheme



2. Second Order Upwind Scheme



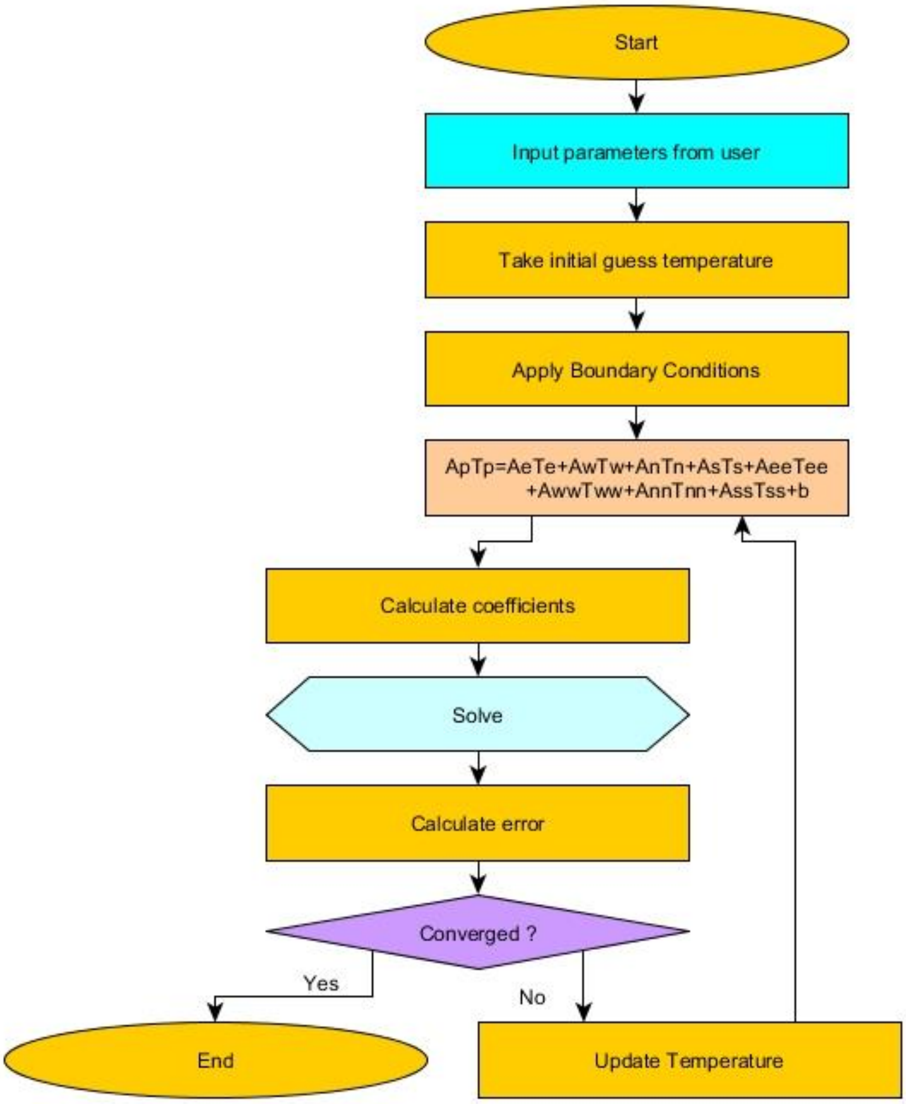
3. QUICK scheme



Problem 2:

Consider a 2D Cartesian computational x-y domain of non-dimensional size L=6 unit and H=1 unit, for CHC with a prescribed velocity field. This corresponds to a slug flow (u=1, v=0) of a fluid in a channel; subjected to a non-dimensional temperature of θ=1 at the inlet and θ=0 at the walls. At the outlet, fully developed Neumann BC is used. The initial condition for non-dimensional temperature of the fluid is θ=0. Run the code for two different advection schemes: (a) FOU and (b) QUICK; at Re=10 and Pr=1 (you can take any value of thermo-physical properties to obtain the given Re and Pr). Take the maximum number of grid points in x-and y-direction as imax=42 and jmax=22, respectively; and convergence criteria (εst) as 0.0001.

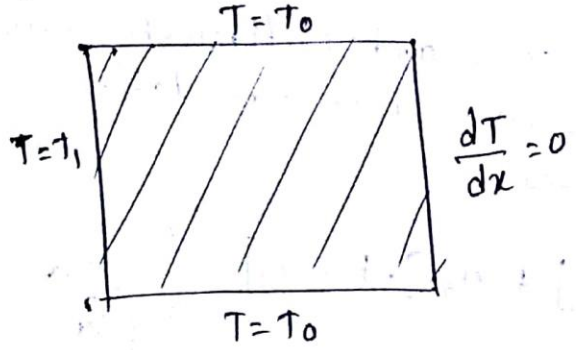
Algorithm



Q.2. General Convection diffusion equation

$$\boxed{\rho \frac{\partial T_i}{\partial t} + \rho u_j \frac{\partial T_i}{\partial x_j} = \frac{k_i}{C_p} \frac{\partial^2 T_i}{\partial x_j \partial x_j}}$$

Defining non-dimensional parameters.



⇐ Given.

$$\theta = \frac{T - T_0}{T_1 - T_0} \; ; \quad x = \frac{X}{h}$$

$$u = \frac{u^*}{U_\infty} \; ; \quad \tau = \frac{t}{t_0}$$

$$\left[\frac{(T_1 - T_0)}{t_0} \cdot \frac{\partial \theta}{\partial \tau} + \frac{(T_1 - T_0) U_\infty u^*}{h} \frac{\partial \theta}{\partial x} = (T_1 - T_0) \cdot \frac{k}{\rho C_p} \cdot \frac{1}{h^2} \cdot \frac{\partial^2 \theta}{\partial x^2} \right]$$

The general formulation is same for the problem 2. Only the value of diffusion coefficient is changed in the formula. It has to be noted that the time step 'dt' is calculated assuming the Strouhal Number 'Sr' equal to one. Strouhal number is associated with wake formation. Since it is not the part of the study for this code, effect of Strouhal number is neglected. This can be justified by the low Reynolds's Number flow with Re=10.

Multiplying by $\frac{h}{U_\infty}$.

$$\frac{h}{U_\infty t_0}\frac{\partial \theta}{\partial \psi} + u^* \frac{\partial \theta}{\partial x} = \left(\frac{K}{\rho C_p}\cdot\frac{\mu}{\mu}\cdot\frac{1}{U_\infty h}\right)\frac{\partial \theta}{\partial x}$$

We know that

$$Re = \frac{\rho U_\infty h}{\mu} \; ; \; Pr = \frac{\mu C_p}{K} \; ; \; Sr = \frac{h}{U_\infty t_0} \quad — (D)$$

Reynold's number   Prandtl number   Strouhal number

$$\boxed{Sr\frac{\partial \theta}{\partial \psi} + U_\infty \frac{\partial \theta}{\partial x} = \frac{1}{Re \cdot Pr}\cdot\frac{\partial \theta}{\partial x}} \quad — (A)$$

As thermophysipcal properties are to be adjusted according to Re, Pr, taking $\boxed{Sr = 1}$

We have

$$\boxed{\frac{\partial \theta}{\partial \psi} + U_\infty \frac{\partial \theta}{\partial x} = \frac{1}{Re \cdot Pr}\frac{\partial \theta}{\partial x}} \quad — (B)$$

from Neumann boundary condition.

$\boxed{\theta_{end} = \theta_{end-1}}$ as $\frac{\partial \theta}{\partial x} = 0$ .. fully developed flow

from (B) we have $\boxed{\Gamma = \frac{1}{Re \cdot Pr}}$

The final form (G) is valid for this question as well. Eqn (H) from question (1) is also valid for ~~the~~ Q.2.

Definition of time step $\Rightarrow$ $\boxed{dt = \frac{dh}{U_\infty \cdot Sr}}$ $\rightarrow$ from (D)

Grid details and the implemented boundary condition

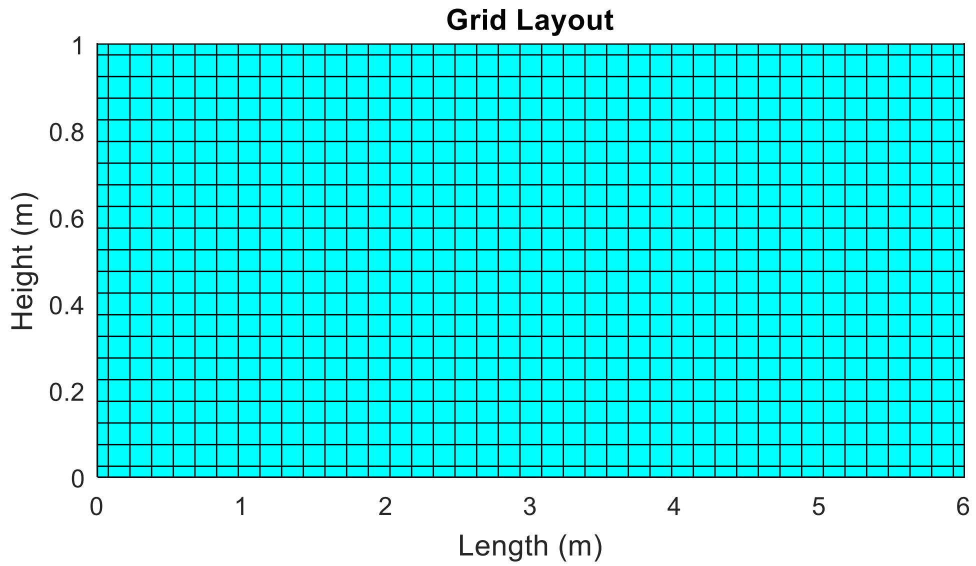


Fig 2. Generated uniform grid for given Cartesian geometry

Grid Specifications

A uniform grid was generated. Black lines are passing through the actual points on the domain. Control volume surfaces lie at half the unit distance from them.

Boundary Conditions

- On the left boundary uniform temperature $1^{o}C$ is applied
- On the right boundary Fully developed Neumann boundary condition is applied
- On the top boundary uniform temperature $0^{o}C$ is applied
- On the bottom boundary uniform temperature $0^{o}C$ is applied

Code of the problem

```
%============================================================%
% Assignment 3 CHDHT ME 415
% Problem Number 1. Unsteady Heat Convection
% Progarm uses FVM to solve the problem of 2D convection
% Heat generation is zero
% Designed only for rectangular co-ordinate system
%
% AUTHOR:
% Sanit P. Bhatkar (173109003@iitb.ac.in)
% Roll No: 173109003
% Place: IIT BOMBAY.
%============================================================%

%% Input of Variables

%==============================INPUT OF VARIABLES=========================
clc
clearvars
fprintf('\n============ Unsteady Heat Convection ============');
fprintf('\n\n---------------------- INPUT -----------------------\n');
l=6;
h=1;
l=l/h;
h=h/h;
% dl=0.05;
% dh=0.05;
Re=10;
Pr=1;
Sr=1;
k=1/(Re*Pr);
eps=1e-4;
u=1;
v=0;
rho=1;
cp=1;
fprintf('\nEnter the scheme to be used for the problem \n\n 1) First Order Upwind
Scheme\n 2) QUICK Scheme \n');
choice=input('\nEnter scheme number: ');


% k_cond=input('\nInput conductivity of the material: ');
%------------Geometry definition--------------

% fprintf('\nGeometry Parameters -->\n');
% l=input('\nLength of column in m: ');
% h=input('Height of column in m: ');
% dl=input('Grid size along length in m: ');
% dh=input('Grid size along height in m: ');

%------------Boundary condition input---------

tbn=0;
tbs=0;
% tbe=30;
tbw=1;

% fprintf('\nBoundary Conditions -->\n');
% tbn=input('\nBoundary condition at north boundary in degree C:  ');
% tbs=input('Boundary condition at south boundary in degree C:  ');
% tbe=input('Boundary condition at east boundary in degree C:  ');
% tbw=input('Boundary condition at west boundary in degree C:  ');
```

```
%% Grid formation

%============================FINITE VOLUME METHOD==========================

%STEP 1: Divide domain into finite sized subdomain called control volumes

%Our domain is uniform
%But boundary and the first point grid space is different
%That is why separate calculation is done
%sx represents the number of points in a grid in x direction
%NOTE: dl has to be some multiple of length

sx=42;
dl=l/(sx-2);
x(1,1)=0;
x(sx,1)=l;
x(2,1)=x(1,1)+(dl/2);
for m=3:sx-1
    x(m,1)=x(m-1,1)+dl;
end

%sy represents the number of points in a grid in y direction

sy=22;
dh=h/(sy-2);
y(1,1)=0;
y(sy,1)=h;
y(2,1)=y(1,1)+(dh/2);

for m=3:sy-1
    y(m,1)=y(m-1,1)+dh;
end


%% Boundary Conditions and Initial Condition

T=zeros(sy,sx);
T(1:sy,1:sx)= 0;
T_ini=T(1,1);

%Application of boundary conditions
% T(1,1:sx)=tbn;
T(sy,1:sx)=tbs;
% T(1:sy,sx)=tbe;
T(1:sy,1)=tbw;
T(sy,1)=(tbs+tbw)/2;

%Definition of convergence Formula for residual will be used in the code

%eps=input('\nEnter minimum convergence error: ');
%eps=10^-(any value);


%STEP 2: Integrate governing equation over boundaries of control volume

%STEP 3: Profile assumption

%We will get equation of type
%ApTp=AeTe+AwTw+AnTn+AsTs+B
%Assuming isotropic material and uniform control volume with no heat generation

%% Common properties
```

```
%Conductivity at each boundary
ke=k; kw=k; kn=k; ks=k;
%Velocity at each boundary
ue=u; uw=u; vn=v; vs=v;

% Diffusion Coefficient
game=ke/cp;  gamw=kw/cp;
gamn=kn/cp;  gams=ks/cp;

% Advection strength
Fe=rho*ue*dh;
Fw=rho*uw*dh;
Fn=rho*vn*dl;
Fs=rho*vs*dl;

%The syntax is used to calculate rounded time step
dt=round((min((dl/(2*u)),(dh/(2*v))))*1000)/1000;
dt=(dh*u*Sr);


%% Genrealized formulation for first order Schemes

%% FOU scheme

if choice == 1


To=T;
itr=0;
mxe=10;


while mxe>eps


%% Inner nodes Formulation FOU

for i=sy-1:-1:2

for j=2:sx-1

    %dxe is distance from point P to east point
    %dxw is distance from point P to west point
    %dyn is distance from point P to north point
    %dys is distance from point P to south point

    dxe=x(j+1,1)-x(j,1);
    dxw=x(j,1)-x(j-1,1);
    dyn=y(i,1)-y(i-1,1);
    dys=y(i+1,1)-y(i,1);

    % Diffusion strength
    De=game*dh/dxe;
    Dw=gamw*dh/dxw;
    Dn=gamn*dl/dyn;
    Ds=gams*dl/dys;

    % Peclet Number
    Pe=Fe/De;
    Pw=Fw/Dw;
```

```
    Pn=Fn/Dn;
    Ps=Fs/Ds;


    %% Change the formula for other first order schemes

    % A(|P|)=1+max(-Peclet,0) for FOU
    % Class notes page 12 (Reference)
    % S.V.Patankar Table 5.1

    % This defined for FOU

    Ae=1+max(-Pe,0);
    Aw=1+max(-Pw,0);
    An=1+max(-Pn,0);
    As=1+max(-Ps,0);



    %% General grid formula FOU

    ae=De*Ae+max(-Fe,0);
    aw=Dw*Aw+max(Fw,0);
    an=Dn*An+max(-Fn,0);
    as=Ds*As+max(Fs,0);
    apo=rho*dl*dh/dt;
    b=apo*To(i,j);

    ap=ae+aw+an+as+apo;
    T(i,j)=(ae*T(i,j+1)+aw*T(i,j-1)+an*T(i-1,j)+as*T(i+1,j)+b)/ap;


end

%% East Boundary FOU

% Calculation for EAST boundary

  T(i,sx)=T(i,sx-1); %Fully Developed Neumann boundary condition

end


%% Stopping Criteria for FOU

%ABSOLUTE ERROR calculation at every point
error=(l*abs(T-To))/(u*dt*100);
mxe=max(max(error));

itr=itr+1;
To=T;


end

end


%% Genrealized formulation for higher order Schemes

%% QUICK scheme
```

```
if choice == 2

% QUICK scheme

To=T;
itr=0;
mxe=10;
dum=zeros(sy+2,sx+2);
dum(2:sy+1,2:sx+1)=T;
dum(end,1:sx+1)=tbs;


while mxe>eps

dum(2:sy+1,2:sx+1)=T;

for i=sy-1:-1:2

for j=2:sx-1


    %dxe is distance from point P to east point
    %dxw is distance from point P to west point
    %dyn is distance from point P to north point
    %dys is distance from point P to south point

    dxe=x(j+1,1)-x(j,1);
    dxw=x(j,1)-x(j-1,1);
    dyn=y(i,1)-y(i-1,1);
    dys=y(i+1,1)-y(i,1);

    % Diffusion strength
    De=game*dh/dxe;
    Dw=gamw*dh/dxw;
    Dn=gamn*dl/dyn;
    Ds=gams*dl/dys;


    %% Inner nodes formula QUICK

    %Malasekra page 158

    ae=De+(-3/8)*max(Fe,0)+(-6/8)*max(-Fe,0)+(-1/8)*max(-Fw,0);
    aee=max(-Fe,0);
    aw=Dw+(6/8)*max(Fw,0)+(1/8)*max(Fe,0)+(3/8)*max(-Fw,0);
    aww=(-1/8)*max(Fw,0);
    an=Dn+(-3/8)*max(Fn,0)+(-6/8)*max(-Fn,0)+(-1/8)*max(-Fs,0);
    ann=max(-Fn,0);
    as=Ds+(6/8)*max(Fs,0)+(1/8)*max(Fn,0)+(3/8)*max(-Fs,0);
    ass=(-1/8)*max(Fs,0);
    apo=rho*dl*dh/dt;
    b=apo*To(i,j);


    % Very long if syntax is used to avoid trouble in code execution at
    % boundaries

    if i==sy-1
        ass=0;
    end
```

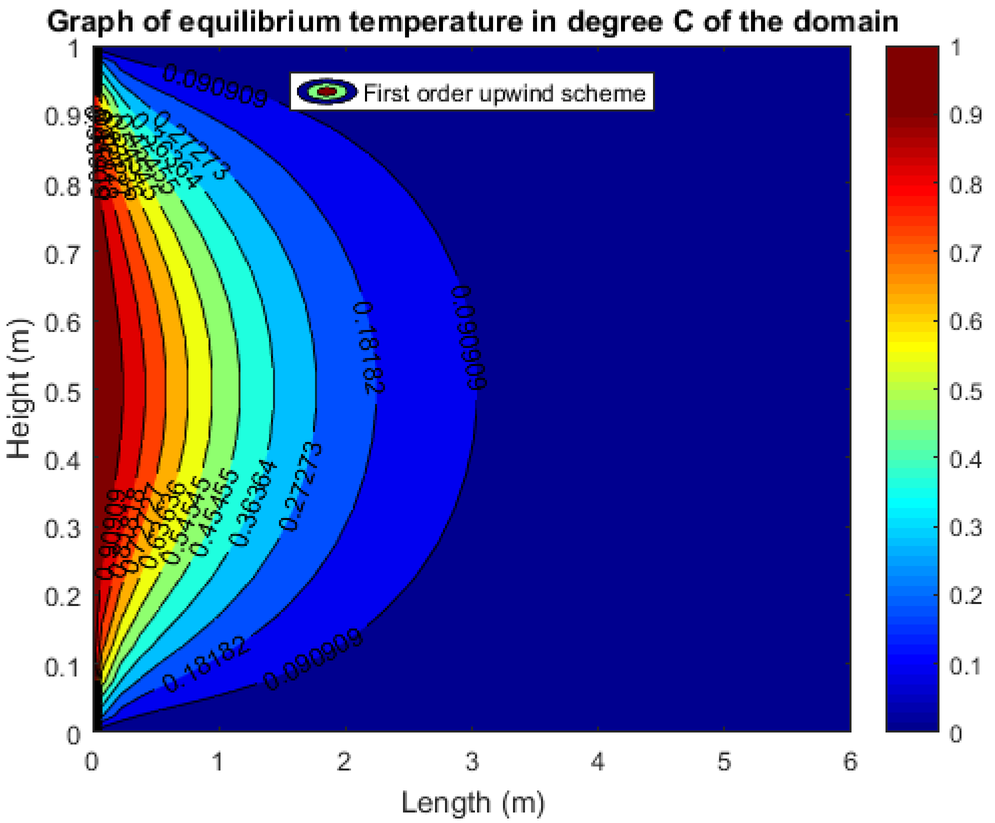
```
    if i==2
        ann=0;
    end

    if j==2
        aww=0;
    end

    if j==sx-1
        aee=0;
    end

    %% QUICK scheme grid formula

    ap=ae+aw+an+as+apo+aee+aww+ann+ass;
    ii=i+1;
    jj=j+1;
    dum(ii,jj)=(ae*dum(ii,jj+1)+aw*dum(ii,jj-1)+an*dum(ii-
1,jj)+as*dum(ii+1,jj)+aee*dum(ii,jj+2)+aww*dum(ii,jj-2)+ann*dum(ii-
2,jj)+ass*dum(ii+2,jj)+b)/ap;
    T=dum(2:sy+1,2:sx+1);
end

end

%% East Boundary QUICK

for i=sy-1:-1:2

% Calculation for EAST boundary
    T(i,sx)=T(i,sx-1);

end


%% Stopping Criteria QUICK

%ABSOLUTE ERROR calculation at every point
error=(l*abs(T-To))/(u*dt*100);
mxe=max(max(error));

itr=itr+1;
To=T;

end

end

%% Output %%

% Grid plot
figure (1)
r=ones(sy,sx);
map = [0,1,1];
colormap(map)
pcolor(x,y,r)
title('Grid Layout');
xlabel('Length (m)');
ylabel('Height (m)');
pbaspect([l/h 3 1])
```

```
%Plot of the temperature contours as the output.

figure(2)
colormap jet
contourf(x,y,flipud(T),10,'showtext','on')
c=colorbar;
c.Limits= [min(min(T)) max(max(T))];
title('Graph of equilibrium temperature in degree C of the domain')
xlabel('Length (m)');
ylabel('Height (m)');

if choice == 1
    legend('First order upwind scheme','Location','North');
else
    legend('QUICK scheme','Location','North');
end

r=x/h;

figure(3)

idx=find(round(r,1,'Significant')==1,1);
plot(T(sy:-1:1,idx),y)
grid on
title('Graph of equilibrium temperature in degree C at different locations')
ylabel('Height in m');
xlabel('Non-Dimensional Temperature in degree C');
hold on
idx=find(round(r,1,'Significant')==2,1);
plot(T(sy:-1:1,idx),y)
hold on
idx=find(round(r,1,'Significant')==3,1);
plot(T(sy:-1:1,idx),y)
hold on
idx=find(round(r,1,'Significant')==5,1);;
plot(T(sy:-1:1,idx),y)
hold off
legend('x/h = 1','x/h = 2','x/h = 3','x/h = 5','Location','NorthEast');



%_____________________________________________________________________________
```

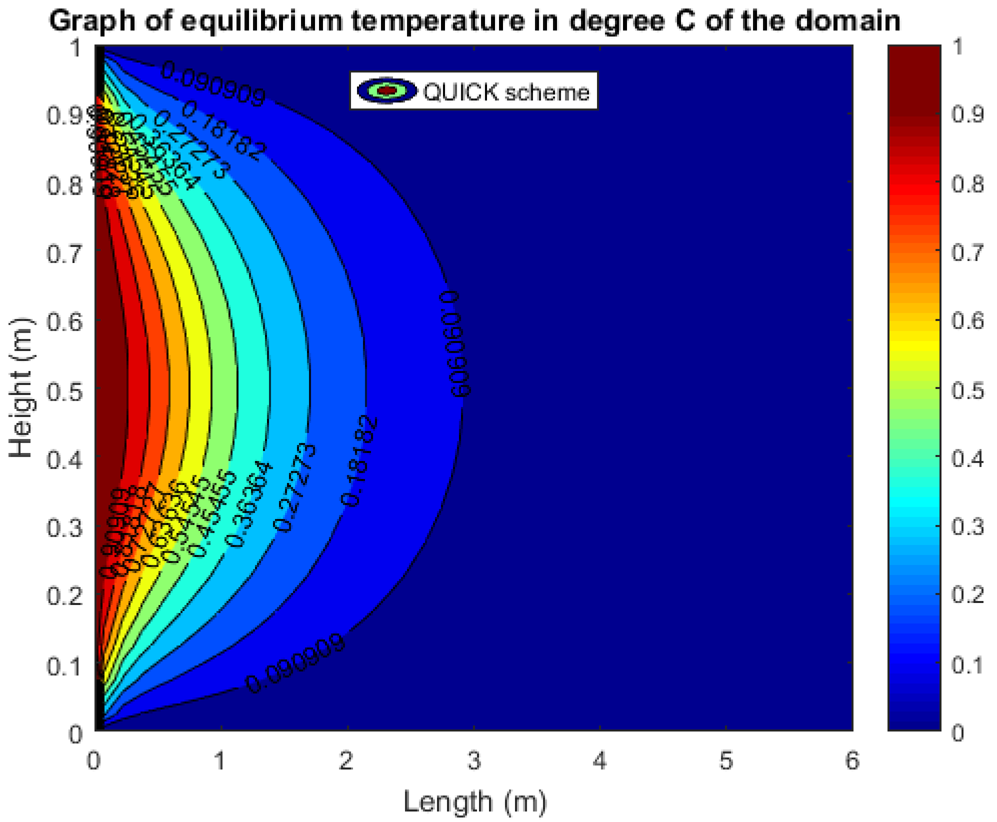
Results and Discussion

A. Steady State Temperature Contours for different Advection Schemes

1. First Order Upwind Scheme

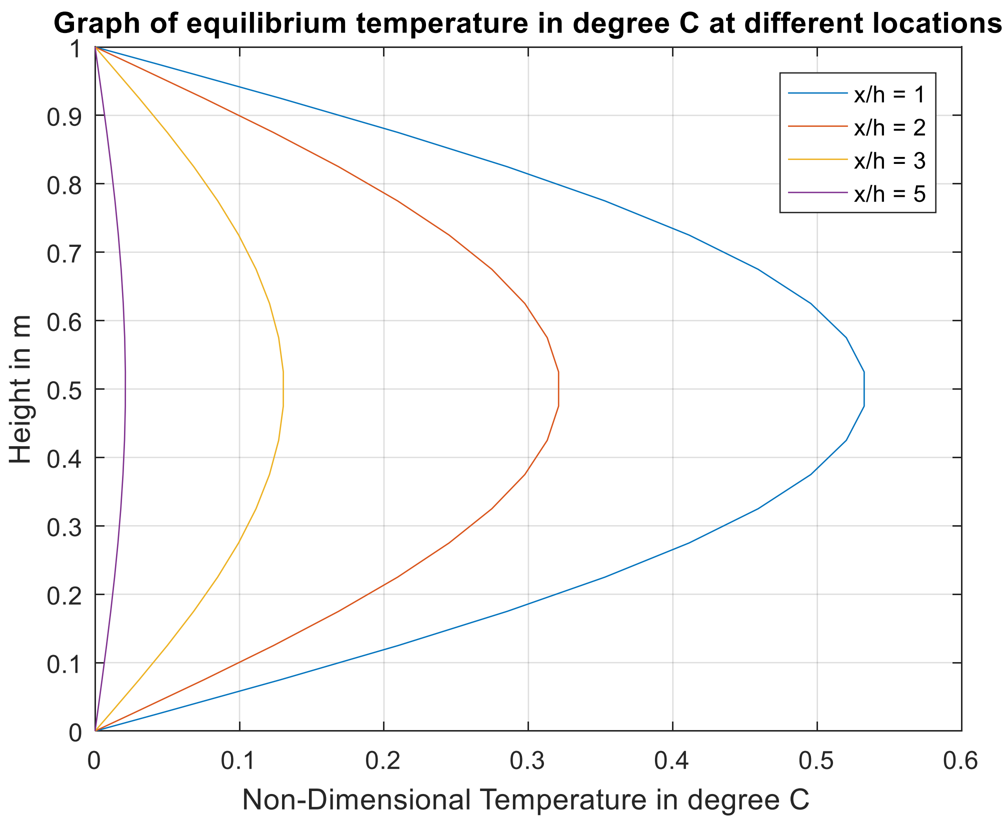


2. **QUICK scheme**

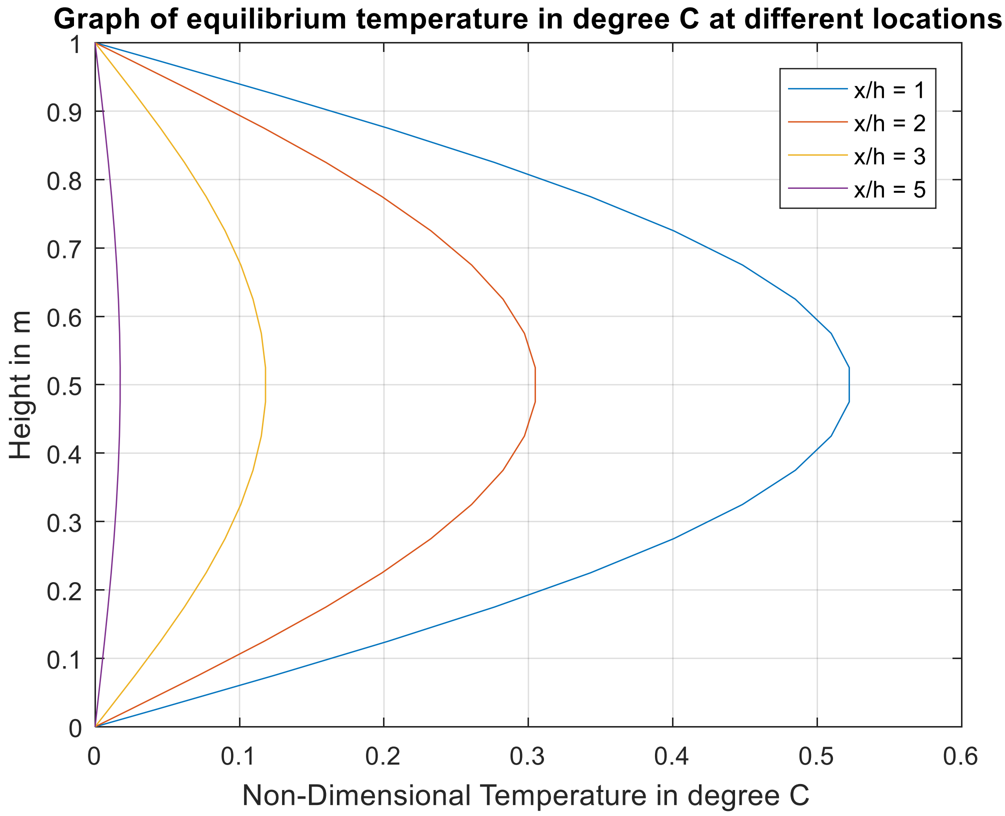


B. **Temperature profile at the vertical line (x/h = 1 , 2, 3, 5) for the different advection schemes**

1. **First Order Upwind Scheme**



2. QUICK scheme



Discussion

Temperature profile for the various x/h values are plotted at steady state.

Order of false diffusion is FOU >QUICK. This can be seen the graph that for same x/h value (red curve) the diffusion by FOU scheme is more than QUICK scheme.

Unlike the case where angle of attack was 45^{o}, negative values for QUICK were not present. It can be concluded that the higher order schemes can be accurate but sometimes not physically realistic. Thus one must always run first order upwind scheme before any higher order upwind scheme to check the trend of variation of quantity.

Order of accuracy (closeness to the ideal scheme) is QUICK > FOU.