Case Study : Laminar Heated Channel Flow Using FVM and SIMPLE

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

% Finite Volume Method with SIMPLE algorithm

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

clear,clc,close all

global Fw Fe Fs Fn DF aW aE aS aN aP bP dU dV

%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

% CHANNEL FLOW PROBLEM WITH CONSTANT TEMPERATURE OR HEAT FLUX WALLS

% Geometry

H = 0.01; % Height of channel in y-direction(m)

L = 10*H; % Length of cavity in x-direction (m)

% Grid geometry

Nx = 200; % Number of main grid points in x-direction within domain

dx = L/Nx; % Grid spacing in x-direction

Ny = 40; % Number of main grid points in y-direction within domain

dy = H/Ny; % Grid spacing in y-direction

dz = 0.01; % Width in z-direction for flux calculations (m)

x = dx/2:dx:L-dx/2; % x-locations of main grid points (m)

xu = 0:dx:L; % x-locations of u-velocities (m)
```

```
y = dy/2:dy:H-dy/2; % y-locations of main grid points (m)
yv = 0:dy:H; % y-locations of v-velocities (m)
iu = 1:Nx+1; Ju = 2:Ny+1; % Interior node numbers for u
Iv = 2:Nx+1; jv = 1:Ny+1; % Interior node numbers for v
Ip = 2:Nx+1; Jp = 2:Ny+1; % Interior node numbers for p
iF = 2:Nx; jF = 2:Ny; % Node numbers for face flow (or advection)
% Properties (air at STP)
rho = 1.2; % Density (kg/m^3)
mu = 1.8e-5; % Absolute viscosity (N-s/m^2)
nu = mu/rho; % Kinematic viscosity (m^2/s)
kt = 0.025; % Thermal conductivity (W/m-K)
cp = 1006; % Specific heat (J/kg-K)
alpha = kt/(rho*cp); % Thermal diffusivity (m^2/s)
Pr = nu/alpha; % Prandtl number
% Boundary conditions
Re = 100; % Reynolds number
U = Re*nu/(2*H); % Average velocity(m/s)
Ti = 20; % Inlet temperature (deg. C)
Tw = 100; % Wall temperature (deg. C)
qw = 100; % Wall heat flux (W/m^2)
BC_N = 1; % BC_N = 0 for Tw, BC_N = 1 for qw
BC_S = 1; % BC_S = 0 for Tw, BC_S = 1 for symmetry
% Solution controls
alphaU = 0.3; % Velocity relaxation (under)
alphaP = 0.2; % Pressure relaxation (under)
NmaxSIM = 1e+4; % Iteration max for SIMPLE algorithm (-)
NmaxGSI = 1e+1; % Iteration max for numerical method (-)
err = 1e-5; % Convergence criteria (-)
div = 1e+1; % Divergence criteria (-)
% Initialize u, v, p, T, F, a, and residual matrices
u = zeros(Nx+1,Ny+2); v = zeros(Nx+2,Ny+1);
```

```
uStar = zeros(Nx+1,Ny+2); vStar = zeros(Nx+2,Ny+1);

uPrime = zeros(Nx+1,Ny+2); vPrime = zeros(Nx+2,Ny+1);

dU = zeros(Nx+1,Ny+2); dV = zeros(Nx+2,Ny+1);

T = zeros(Nx+2,Ny+2);

p = zeros(Nx+2,Ny+2);

pPrime = zeros(Nx+2,Ny+2);

Fe = zeros(Nx+1,Ny+1); Fw = zeros(Nx+1,Ny+1); % Flow coefficients

Fn = zeros(Nx+1,Ny+1); Fs = zeros(Nx+1,Ny+1);

DF = zeros(Nx+1,Ny+1);

aE = zeros(Nx+1,Ny+1); aW = zeros(Nx+1,Ny+1); % Coefficients for

aN = zeros(Nx+1,Ny+1); aS = zeros(Nx+1,Ny+1); % discretized equations

aP = zeros(Nx+1,Ny+1); bP = zeros(Nx+1,Ny+1);

ures = zeros(NmaxSIM,1); % Residual for u

vres = zeros(NmaxSIM,1); % Residual for v

pres = zeros(NmaxSIM,1); % Residual for pPrime

% Inlet velocity for uniform flow at inlet

u(:,Ju) = U;

% Initialize to inear pressure drop for fully developed flow

p1 = 12*mu*U*L/(2*H)^2;

p(Ip,Jp) = ones(Nx,Ny).*linspace(p1,0,Nx)';

% Initialize temperature to inlet and wall temperatures

T(:,Jp) = Ti; T(:,1) = Tw; T(:,Ny+2) = Tw;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

% SIMPLE algorithm

% Constant diffusion coefficients

Dx = (mu/dx)*dy*dz;

Dy = (mu/dy)*dx*dz;

for n = 1:NmaxSIM
```

```
% Initial guess

uOld = u;

vOld = v;

pStar = p;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

% STEP 1a: solve x-momentum as uStar

% Setup coefficients

FVM_u(Nx,Ny,dx,dy,dz,rho,Dx,Dy,iF,Ju,alphaU,uOld,vOld,pStar,BC_S);

% Use previous calculation as initial guess in numerical method

[uStar,ures(n)] = FVM_GS_ext_mesh(Nx,Ny+1,alphaU,NmaxGSI,err,uOld);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

% STEP 1b: solve y-momentum as vStar

% Setup coefficients

FVM_v(Nx,Ny,dx,dy,dz,rho,Dx,Dy,Iv,jF,alphaU,u,v,pStar)

% Use previous calculation as initial guess in numerical method

[vStar,vres(n)] = FVM_GS_ext_mesh(Nx+1,Ny,alphaU,NmaxGSI,err,vOld);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

% STEP 2: Solve pressure correction equation (PCE)

% Setup coefficients

FVM_pcorr(Nx,Ny,dx,dy,dz,rho,Ip,Jp,uStar,vStar)

% Use numerical method to calculate pressure correction

pPrime(:,:) = 0;

[pPrime,pres(n)] = FVM_GS_ext_mesh(Nx+1,Ny+1,1,NmaxGSI,err,pPrime);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

% STEP 3: calculate corrected pressure and velocity

% p corrections with under-relaxation

p(Ip,Jp) = pStar(Ip,Jp) + pPrime(Ip,Jp)*alphaP;

% u corrections
```

```
uPrime(iF,Ju) = dU(iF,Ju).*(pPrime(iF,Ju) - pPrime(iF+1,Ju));

u(iF,Ju) = uStar(iF,Ju) + uPrime(iF,Ju);

% v corrections

vPrime(Iv,jF) = dV(Iv,jF).*(pPrime(Iv,jF) - pPrime(Iv,jF+1));

v(Iv,jF) = vStar(Iv,jF) + vPrime(Iv,jF);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

% STEP 4: Check for convergence or divergence

if n > 10

fprintf('n = %5.0f, u = %6.2e, v = %6.2e, p = %6.2e \n',...

n,ures(n),vres(n),pres(n))

cTest = max([ures(n),vres(n)]);

if cTest < err

break;

elseif cTest > div || isnan(cTest)

fprintf('Residuals are too high.')

break;

end

end

% Apply right boundary condition (outlet, du/dx = dv/dx = 0)

u(Nx+1,:) = u(Nx,:);

v(Nx+2,:) = v(Nx+1,:);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

% STEP 5: solve for temperature distribution

% Setup coefficients

FVM_phi(Nx,Ny,dx,dy,dz,rho,kt/cp,qw/cp,Ip,Jp,u,v,BC_S,BC_N)

% Use numerical method to calculate temperature

[T,Tres] = FVM_GS_ext_mesh(Nx+1,Ny+1,1.0,1e4,1e-8,T);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
```

```
% POST PROCESSING
figure('Name','Convergence Plot for Scaled Residuals',...
'Position',[100 100 500 300])
% Convergence plot
nlist = 10:n;
semilogy(nlist,ures(nlist),'-b',nlist,vres(nlist),'-r',...
nlist,pres(nlist),'-g')
legend('u residual','v residual','p residual')
xlabel('Iteration')
ylabel('Scaled Residual')
FVM_Vplot(Nx,Ny,x,xu,y,H,u(iu,Ju),v(Iv,jv),p(Ip,Jp),U)
FVM_Tplot(Nx,Ny,x,y,L,H,rho,cp,u(iu,Ju),T(Ip,Jp),U,Ti,Tw,qw,BC_N)
```

Figure 1 convergence plot for scaled residuals

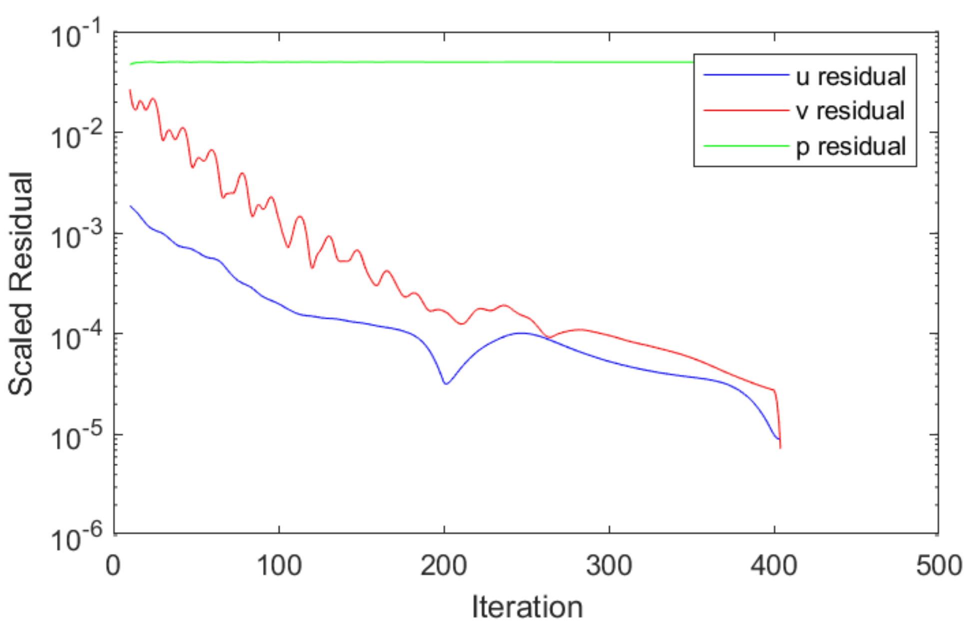


Figure 2 velocity and contour plots

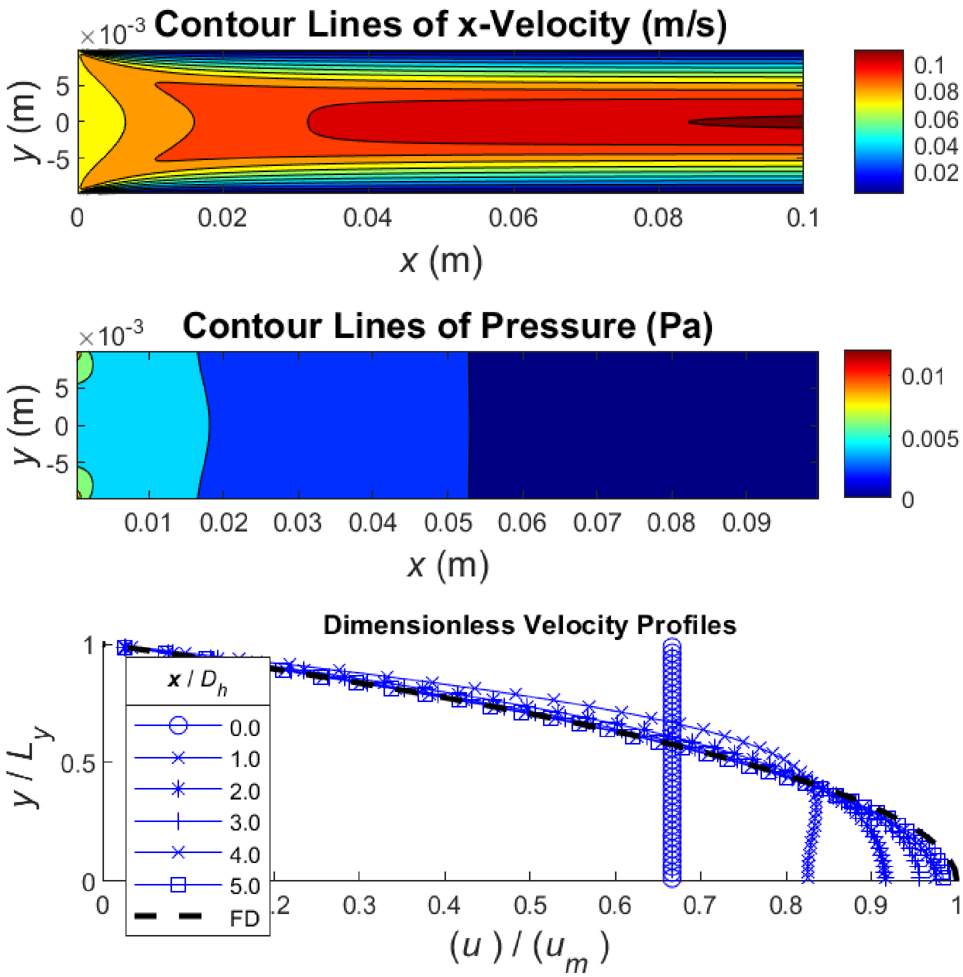
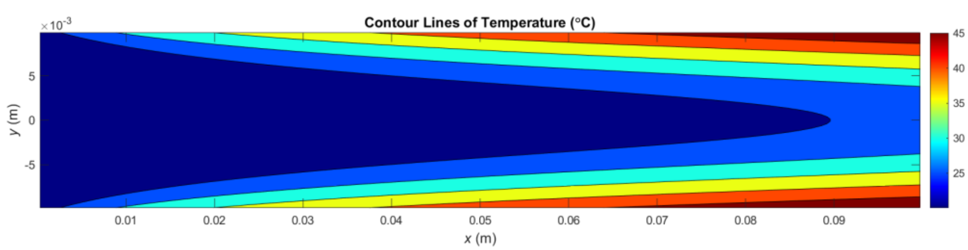
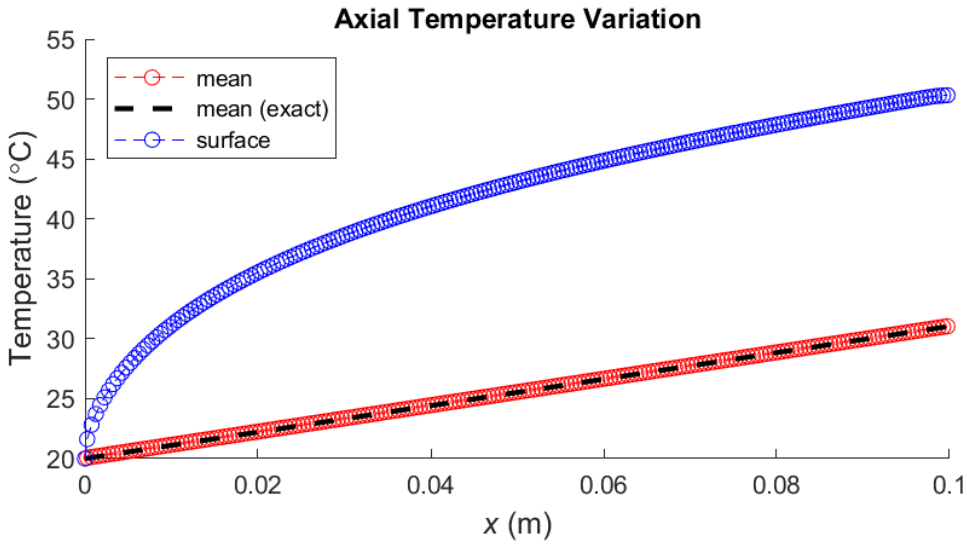
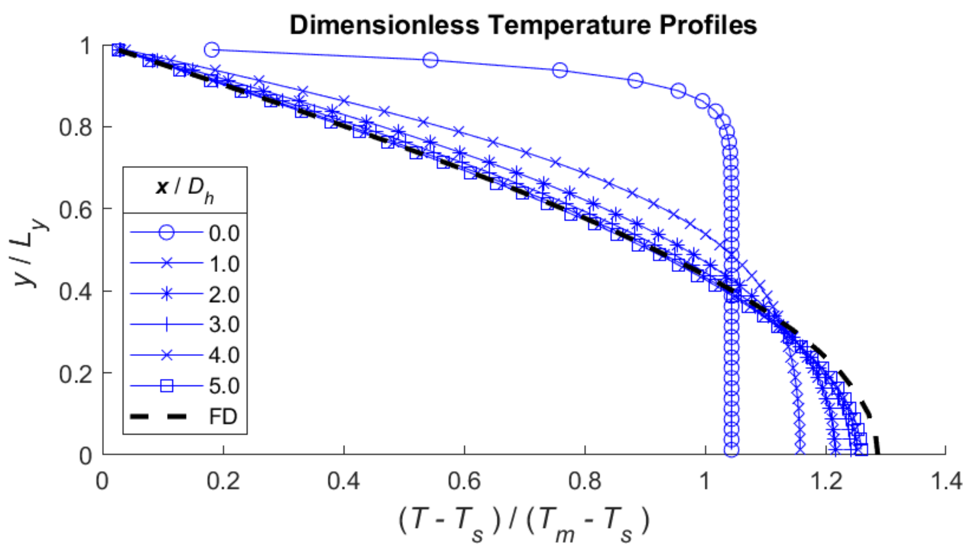


Figure 3 constant surface heat flux




Axial Temperature Variation
mean
mean (exact)
surface
Temperature (°C)
55
50
45
40
35
30
25
20
0
0.02
0.04
0.06
0.08
0.1
x (m)


Dimensionless Temperature Profiles
x / D_h
0.0
1.0
2.0
3.0
4.0
5.0
FD
y / L_y
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
(T - T_s) / (T_m - T_s)