Computational Fluid Dynamics

Francisco Castillo
Homework 7

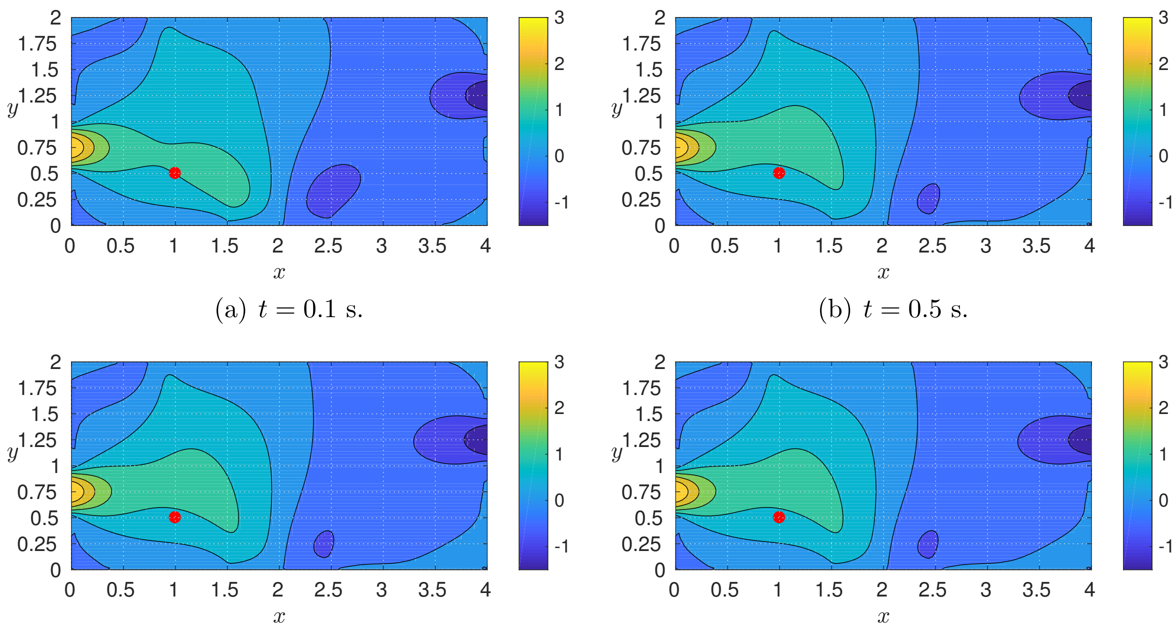
March 20, 2018

1 Introduction

In this assignment we will study a two dimentional mixing chamber. Unlike the previous assignment, in this case we include the continuity equation. Thus, we will have an actual outlet flow through the outlet imposing such equation. In addition, we have included the pressure term as well. First we will show the results requested from the simulation and the code. At the end of the document we will present the equations in sub-index form for the Poisson equation and the projection step of the fractional step method, and the boundary conditions for the Lagrange multiplier. To finish, the calculation of the coefficients that characterize the fully developed velocity profiles given information of the inlets and the average velocities.

2 Results

In figures 1 and 2 we can see the velocity contours at different values of time. In the first figure we can see the horizontal velocity contours. The red dot represents the probe located to meassure such quantity. As we can see we have inputs through inlets 1 (positive u) and 2 (negative u). Figure 2 shows the vertical velocity contours at different values of time. As we can see we only introduce fluid vertically through the inlet 3 and, to satisfy the conservation of mass, there is a fluid going out through the outlet where also have zero Neumann boundary conditions. For both u and v the contours don't change much with time, and their solution is not close to zero (initial condition), indicating that the steady state is reached very quickly.



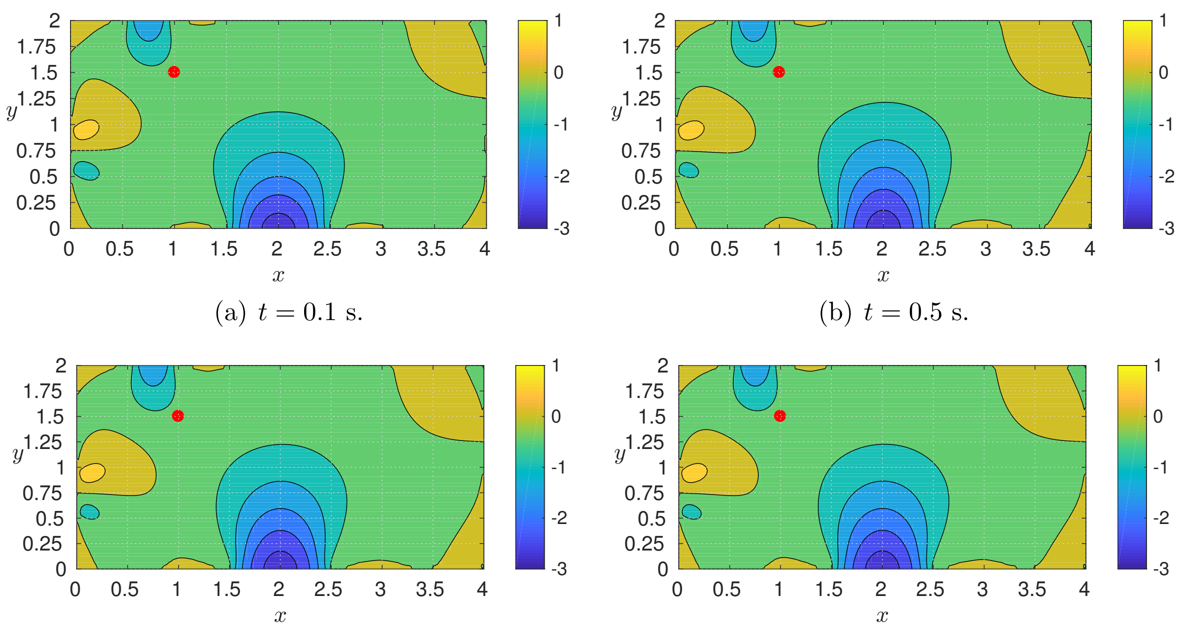
(a) $t = 0.1$ s.

(b) $t = 0.5$ s.

(c) $t = 1$ s.

(d) $t = 2$ s.

Figure 1: Horizontal velocity contours.



(a) $t = 0.1$ s.

(b) $t = 0.5$ s.

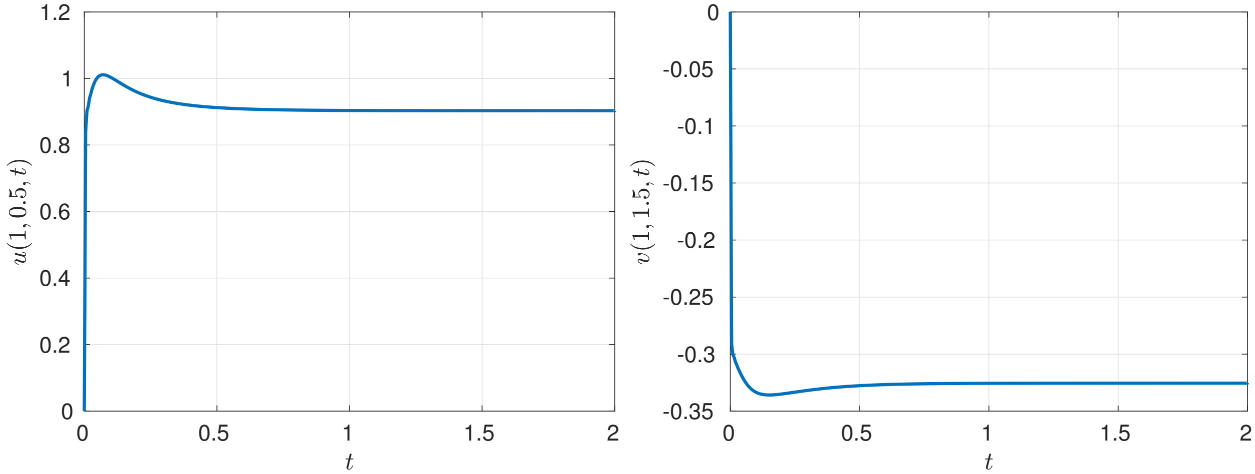
(c) $t = 1$ s.

(d) $t = 2$ s.

Figure 2: Vertical velocity contours.

In the next figure we can see the probes meassurements with time during the first 2 seconds. As commented above, the probes show us how for both u and v the fluid reaches the steady state very quickly.

To finish with the results, we proceed with the GCI analysis. In the following table we have the data needed to perform it obtained by doing simulations with different meshes.



(a) Probe 1. (b) Probe 2.

Figure 3: Meassures of the probes.

Grid	M	N	$u(1, 0.5, 1)$	$v(1, 1.5, 1)$
1	128	64	0.903783280349676	−0.325610242364466
2	64	32	0.908404354402535	−0.326805053418745
3	32	16	0.925282713909409	−0.331680997541251

Table 1: GCI analysis data.

Taking the data from u, we can calculate an order of convergence $p = 1.868874573994991$, close to the theoretical value two. Using Richardson extrapolation with the two finest grids we estimate the solution at $h = 0$,

$$u_{h=0} = 0.902041106237625.$$

We obtain the following GCI values

$$GCI_{21} = 0.002409557343461, \qquad GCI_{32} = 0.008756078906480,$$

which give us the following value

$$\frac{GCI_{21}}{GCI_{32}} r^p = 1.005113033349180,$$

where $r = 2$. The previous value tells us that we are in the asymptotic range of convergence. Thus, we can say that the value meassured by the probe is

$$u(1, 0.5, 1) = 0.902041106237625 \pm 0.2409557343461\%$$

Now doing the same for v, we can calculate an order of convergence $p = 2.028899102510403$, close to the theoretical value two. Using Richardson extrapolation with the two finest grids we estimate the solution at $h = 0$,

$$v_{h=0} = -0.325222434200792.$$

We obtain the following GCI values

$$GCI_{21} = 0.001488774434957, \qquad GCI_{32} = 0.006053376475505,$$

which give us the following value

$$\frac{GCI_{12}}{GCI_{23}} r^p = 1.003669451690472.$$

The previous value tells us that we are in the asymptotic range of convergence. Thus, we can say that the value meassured by the probe is

$$v(1, 1.5, 1) = -0.325222434200792 \pm 0.1488774434957\%$$

These values have been obtained using a tolerance of $eps = 10^{-12}$ for the multigrid function.

HOMEWORK 7 - FRANCISCO CASTILLO

Contents

Defined functions

```
function u=ADI_u(u,M,N,dt,hx,hy,Re)

%%%%%%%%%% STEP 1 %%%%%%%%%%
[a,b,c,d]=uvectors(u,M,N,dt,hx,hy,1/Re,1); %Obtain tridiagonal vectors
d=GaussTriSol(a,b,c,d); %Gaussian elimination
% Correspondance with u, obtain u(n+1/2)
for j=2:N+1
    for i=2:M
        u(i,j)=d((j-2)*(M-1)+i-1);
    end
end
% Update top and bottom boundaries (ghost cells), they depend on the
% interior
u(:,N+2)=-u(:,N+1);
u(:,1)=-u(:,2);

%%%%%%%%%% STEP 2 %%%%%%%%%%
[a,b,c,d]=uvectors(u,M,N,dt,hx,hy,1/Re,2);
d=GaussTriSol(a,b,c,d);
for i=2:M
    for j=2:N+1
        u(i,j)=d((i-2)*N+j-1);
    end
end
% Update top and bottom boundaries (ghost cells), they depend on the
% interior
u(:,N+2)=-u(:,N+1); % Top
u(:,1)=-u(:,2); % Bottom

end



function v=ADI_v(v,M,N,dt,hx,hy,Re)
xiv=linspace(-hx/2,4+hx/2,M+2);
%%%%%%%%%% STEP 1 %%%%%%%%%%
[a,b,c,d]=vvectors(v,M,N,dt,hx,hy,1/Re,1); %Obtain tridiagonal vectors
% keyboard
d=GaussTriSol(a,b,c,d); %Gaussian elimination
% keyboard
% Correspondance with v, obtain v(n+1/2)
for j=2:N
    for i=2:M+1
        v(i,j)=d((j-2)*M+i-1);
    end
end
% Update left and right boundaries (ghost cells), as well as the outlet
% Neumann BC, they depend on the interior.
v(1,:)=-v(2,:); % Left
v(M+2,:)=-v(M+1,:); % Right
for i=1:M+2
    if (xiv(i)>=1.5 && xiv(i)<=2.5) % Outlet
        v(i,1)=(4/3)*v(i,2)-(1/3)*v(i,3);
    end
end

%%%%%%%%%% STEP 2 %%%%%%%%%%
[a,b,c,d]=vvectors(v,M,N,dt,hx,hy,1/Re,2); %Obtain tridiagonal vectors
% keyboard
```

```matlab
d=GaussTriSol(a,b,c,d); %Gaussian elimination
% keyboard
% Correspondance with v, obtain v(n+1/2)
for i=2:M+1
    for j=2:N
        v(i,j)=d((i-2)*(N-1)+j-1);
    end
end
% Update left and right boundaries (ghost cells), as well as the outlet
% Neumann BC, they depend on the interior.
v(1,:)=-v(2,:); % Left
v(M+2,:)=-v(M+1,:); % Right
for i=1:M+2
    if (xiv(i)>=1.5 && xiv(i)<=2.5) % Outlet
        v(i,1)=(4/3)*v(i,2)-(1/3)*v(i,3);
    end
end
end



function d=GaussTriSol(a,b,c,d)
    N=length(a);
    for i=2:N
        b(i)=b(i)-c(i-1)*a(i)/b(i-1);
        d(i)=d(i)-d(i-1)*a(i)/b(i-1);
    end
    d(N)=d(N)/b(N);
    for i=N-1:-1:1
        d(i)=(d(i)-c(i)*d(i+1))/b(i);
    end
end



function [u,v]=initialization(M,N,hx,hy)
%% Initialize u
u = zeros(M+1,N+2);
% Impose left and right boundaries, only once, since they do not depend on
% the interior nor they are altered.
yju=linspace(-hy/2,2+hy/2,N+2);
for j=1:N+2
    if (yju(j)>=0.5 && yju(j)<=1) % Inlet 1
        u(1,j)=-48*yju(j)^2+72*yju(j)-24;
    elseif (yju(j)>=1 && yju(j)<=1.5) % Inlet 2
        u(M+1,j)=24*yju(j)^2-60*yju(j)+36;
    end
end
%% Initialize v
v = zeros(M+2,N+1);
% Impose top and bottom boundaries, only once, since they do not depend on
% the interior except the outlet Neumann BC, which will be updated after
% iterations since for the initial condition it is satisfied.
xiv=linspace(-hx/2,4+hx/2,M+2);
for i=1:M+2
    if (xiv(i)>=0.5 && xiv(i)<=1) % Inlet 3
        v(i,N+1)=24*xiv(i)^2-36*xiv(i)+12;
    end
end
end



function [a,b,c,d] = uvectors(u,M,N,dt,hx,hy,alpha,step)
dx=alpha*dt/hx^2;
dy=alpha*dt/hy^2;
d1=dx/2;
d2=dy/2;
if step==1
    a = -d1*ones((M-1)*N,1);
    b = (1+2*d1)*ones((M-1)*N,1);
    c = -d1*ones((M-1)*N,1);
    d = zeros((M-1)*N,1);
    for j=2:N+1
        for i=2:M
            if i==2 % Left boundary BCs
                a((j-2)*(M-1)+1)=0;
                d((j-2)*(M-1)+1)=d2*u(2,j-1)+(1-2*d2)*u(2,j)+d2*u(2,j+1)...
                                    +d1*u(1,j); %u(1,j) fixed at initialization
            elseif i==M % Right boundary BCs
                c((j-2)*(M-1)+M-1)=0;
                d((j-2)*(M-1)+M-1)=d2*u(i,j-1)+(1-2*d2)*u(i,j)+d2*u(i,j+1)...
                                    +d1*u(M+1,j); %u(M+1,j) fixed at initialization
            else % The top and bottom boundaries are imposed by updating the ghost
                    % cells accordingly with them.
                d((j-2)*(M-1)+i-1)=d2*u(i,j-1)+(1-2*d2)*u(i,j)+d2*u(i,j+1);
            end
```

```
        end
    end
elseif step==2
    a = -d2*ones((M-1)*N,1);
    b = (1+2*d2)*ones((M-1)*N,1);
    c = -d2*ones((M-1)*N,1);
    d = zeros((M-1)*N,1);
    for i=2:M
        for j=2:N+1
            if j==2 % Bottom boundary BCs
                a((i-2)*N+1)=0;
                b((i-2)*N+1)=1+3*d2;
                d((i-2)*N+1)=d1*u(i-1,2)+(1-2*d1)*u(i,2)+d1*u(i+1,2);
            elseif j==N+1 % Top boundary BCs
                b((i-1)*N)=1+3*d2;
                c((i-1)*N)=0;
                d((i-1)*N)=d1*u(i-1,N+1)+(1-2*d1)*u(i,N+1)+d1*u(i+1,N+1);
            else % The left and right boundaries are imposed by initialization,
                    % they do not change
                d((i-2)*N+j-1)=d1*u(i-1,j)+(1-2*d1)*u(i,j)+d1*u(i+1,j);
            end
        end
    end
end
end


function [a,b,c,d] = vvectors(v,M,N,dt,hx,hy,alpha,step)
xi=linspace(-hx/2,4+hx/2,M+2);
yj=linspace(0,2,N+1);
% keyboard
dx=alpha*dt/hx^2;
dy=alpha*dt/hy^2;
d1=dx/2;
d2=dy/2;
if step==1
    a = -d1*ones(M*(N-1),1);
    b = (1+2*d1)*ones(M*(N-1),1);
    c = -d1*ones(M*(N-1),1);
    d = zeros(M*(N-1),1);
    for j=2:N
        for i=2:M+1
            if i==2
                a((j-2)*M+1)=0;
                b((j-2)*M+1)=1+3*d1;
                d((j-2)*M+1)=d2*v(2,j-1)+(1-2*d2)*v(2,j)+d2*v(2,j+1);
            elseif i==M+1
                b((j-1)*M)=1+3*d1;
                c((j-1)*M)=0;
                d((j-1)*M)=d2*v(M+1,j-1)+(1-2*d2)*v(M+1,j)+d2*v(M+1,j+1);
            elseif (j==2 && xi(i)>=1.5 && xi(i)<=2.5)
                d(i-1)=d2*(4*v(i,2)/3-v(i,3)/3)+(1-2*d2)*v(i,2)+d2*v(i,3);
            else
                d((j-2)*M+i-1)=d2*v(i,j-1)+(1-2*d2)*v(i,j)+d2*v(i,j+1);
            end
        end
    end
elseif step==2
    a = -d2*ones(M*(N-1),1);
    b = (1+2*d2)*ones(M*(N-1),1);
    c = -d2*ones(M*(N-1),1);
    d = zeros(M*(N-1),1);
    for i=2:M+1
        for j=2:N
            if j==2
                a((i-2)*(N-1)+1)=0;
                if (xi(i)>=1.5 && xi(i)<=2.5)
                    b((i-2)*(N-1)+1)=1+(2-4/3)*d2;
                    c((i-2)*(N-1)+1)=-(2/3)*d2;
                end
                d((i-2)*(N-1)+1)=d1*v(i-1,2)+(1-2*d1)*v(i,2)+d1*v(i+1,2);
            elseif j==N
                c((i-1)*(N-1))=0;
                d((i-1)*(N-1))=d1*v(i-1,N)+(1-2*d1)*v(i,N)+d1*v(i+1,N)...
                                +d2*v(i,N+1);
            else
                d((i-2)*(N-1)+j-1)=d1*v(i-1,j)+(1-2*d1)*v(i,j)+d1*v(i+1,j);
            end
        end
    end
end
end



function [a,b,c,d] = vvectors(v,M,N,dt,hx,hy,alpha,step)
```

```matlab
xi=linspace(-hx/2,4+hx/2,M+2);
yj=linspace(0,2,N+1);
% keyboard
dx=alpha*dt/hx^2;
dy=alpha*dt/hy^2;
d1=dx/2;
d2=dy/2;
if step==1
    a = -d1*ones(M*(N-1),1);
    b = (1+2*d1)*ones(M*(N-1),1);
    c = -d1*ones(M*(N-1),1);
    d = zeros(M*(N-1),1);
    for j=2:N
        for i=2:M+1
            if i==2
                a((j-2)*M+1)=0;
                b((j-2)*M+1)=1+3*d1;
                d((j-2)*M+1)=d2*v(2,j-1)+(1-2*d2)*v(2,j)+d2*v(2,j+1);
            elseif i==M+1
                b((j-1)*M)=1+3*d1;
                c((j-1)*M)=0;
                d((j-1)*M)=d2*v(M+1,j-1)+(1-2*d2)*v(M+1,j)+d2*v(M+1,j+1);
            elseif (j==2 && xi(i)>=1.5 && xi(i)<=2.5)
                d(i-1)=d2*(4*v(i,2)/3-v(i,3)/3)+(1-2*d2)*v(i,2)+d2*v(i,3);
            else
                d((j-2)*M+i-1)=d2*v(i,j-1)+(1-2*d2)*v(i,j)+d2*v(i,j+1);
            end
        end
    end
elseif step==2
    a = -d2*ones(M*(N-1),1);
    b = (1+2*d2)*ones(M*(N-1),1);
    c = -d2*ones(M*(N-1),1);
    d = zeros(M*(N-1),1);
    for i=2:M+1
        for j=2:N
            if j==2
                a((i-2)*(N-1)+1)=0;
                if (xi(i)>=1.5 && xi(i)<=2.5)
                    b((i-2)*(N-1)+1)=1+(2-4/3)*d2;
                    c((i-2)*(N-1)+1)=-(2/3)*d2;
                end
                d((i-2)*(N-1)+1)=d1*v(i-1,2)+(1-2*d1)*v(i,2)+d1*v(i+1,2);
            elseif j==N
                c((i-1)*(N-1))=0;
                d((i-1)*(N-1))=d1*v(i-1,N)+(1-2*d1)*v(i,N)+d1*v(i+1,N)...
                                +d2*v(i,N+1);
            else
                d((i-2)*(N-1)+j-1)=d1*v(i-1,j)+(1-2*d1)*v(i,j)+d1*v(i+1,j);
            end
        end
    end
end
end


function [phi,LinfR] = poisson (phi,rhs,hx,hy,nIterMax,tol)
    r = residual(phi,rhs,hx,hy);
    LinfR=zeros(nIterMax,1);
    LinfR(1)=InfNorm(r);
    for n=1:nIterMax
        phi = multigrid(phi,rhs,hx,hy);
        r = residual(phi,rhs,hx,hy);
        LinfR(n+1,1) = InfNorm(r);
        if LinfR(n+1,1)<tol
            break
        end
    end
    LinfR(n+2:nIterMax)=[];
end


function phi = multigrid(phi,rhs,hx,hy)
    M=size(phi,1)-2;
    N=size(phi,2)-2;
    phi = GaussSeidel(phi,rhs,hx,hy,1);
    if (M>2 || N>2)
        rh = residual(phi,rhs,hx,hy);
        r2h = restrict(rh);
        e2h = zeros(M/2+2,N/2+2);
        e2h = multigrid(e2h,r2h,2*hx,2*hy);
        eh = prolong(e2h);
        phi = phi+eh;
        phi = GaussSeidel(phi,rhs,hx,hy,1);
    end
```

```matlab
end


function r2h = restrict(rh)
    M=size(rh,1)-2;
    N=size(rh,2)-2;
    M2h=M/2;
    N2h=N/2;
    r2h=zeros(M2h+2,N2h+2);
    rh(:,[1 end])=[];
    rh([1 end],:)=[];
    for i=1:M2h
        for j=1:N2h
            r2h(i+1,j+1) = 0.25*sum(sum(rh(2*i-1:2*i,2*j-1:2*j)));
        end
    end
    % Update the ghost cells
        r2h(1,:) = r2h(2,:);
        r2h(M2h+2,:) = r2h(M2h+1,:);
    r2h(:,1) = r2h(:,2);
        r2h(:,N2h+2) = r2h(:,N2h+1);
end


function eh = prolong(e2h)
    M2h=size(e2h,1)-2;
    N2h=size(e2h,2)-2;
    Mh=2*M2h;
    Nh=2*N2h;
    eh=zeros(Mh+2,Nh+2);
    e2h(:,[1 end])=[];
    e2h([1 end],:)=[];
    % Interior
    for i=1:M2h
        for j=1:N2h
            eh(2*i-1+1:2*i+1,2*j-1+1:2*j+1) = e2h(i,j);
        end
    end
    % Calculate the ghost cells
        eh(1,:) = eh(2,:);
        eh(Mh+2,:) = eh(Mh+1,:);
    eh(:,1) = eh(:,2);
        eh(:,Nh+2) = eh(:,Nh+1);
end


function phi = GaussSeidel (phi,rhs,hx,hy,n)
    for k=1:n
        M=size(phi,1)-2;
        N=size(phi,2)-2;
        h1=hy^2/(2*(hx^2+hy^2));
        h2=hx^2/(2*(hx^2+hy^2));
        h3=hx^2*hy^2/(2*(hx^2+hy^2));
        % Update the interior cells
        for i=2:M+1
            for j=2:N+1
                phi(i,j)=h1*(phi(i+1,j)+phi(i-1,j))...
                    +h2*(phi(i,j+1)+phi(i,j-1))-h3*rhs(i,j);
            end
        end
        % Update the ghost cells
        phi(1,:) = phi(2,:);
        phi(M+2,:) = phi(M+1,:);
        phi(:,1) = phi(:,2);
        phi(:,N+2) = phi(:,N+1);
    end
end


function r = residual(phi,rhs,hx,hy)
    M = size(phi,1)-2;
    N = size(phi,2)-2;
    r= zeros(size(phi));
    % Calculate the interior cells
    for i=2:M+1
        for j=2:N+1
            r(i,j)=rhs(i,j)-(phi(i+1,j)-2*phi(i,j)+phi(i-1,j))/hx^2 ...
            -(phi(i,j+1)-2*phi(i,j)+phi(i,j-1))/hy^2;
        end
    end
    % calculate the ghost cells
    r(1,:) = r(2,:);
    r(M+2,:) = r(M+1,:);
    r(:,1) = r(:,2);
```

```
    r(:,N+2) = r(:,N+1);
end


function Linf = InfNorm(x)
    Linf= max(max(abs(x)));
end


function rhs=divV(u,v,M,N,hx,hy,dt)
rhs=zeros(M+2,N+2);
for i=2:M+1
    for j=2:N+1
        rhs(i,j)=(u(i,j)-u(i-1,j))/(hx*dt)+(v(i,j)-v(i,j-1))/(hy*dt);
    end
end
end


function v=outletCorrection(u,v,xv,hx,hy)
s=-sum(u(1,:))*hy+sum(u(end,:))*hy-sum(v(:,1))*hx+sum(v(:,end))*hx;
num=find(xv<=2.5,1,'last')-find(xv>=1.5,1)+1;
vcorr=s/(num*hx);
for i=find(xv>=1.5,1):find(xv<=2.5,1,'last')
    v(i,1)=v(i,1)+vcorr;
end
end


function [u,v]=LagProjection(u,v,phi,dt,hx,hy)
M=size(phi,1)-2;
N=size(phi,2)-2;
for i=1:M+1
    for j=1:N+2
        phix(i,j)=(phi(i+1,j)-phi(i,j))/hx;
    end
end
for i=1:M+2
    for j=1:N+1
        phiy(i,j)=(phi(i,j+1)-phi(i,j))/hy;
    end
end
u=u-dt*phix;
v=v-dt*phiy;
% Apply boundary conditions to ghost cells only
u(:,1)=-u(:,2);
u(:,N+2)=-u(:,N+1);
v(1,:)=-v(2,:);
v(M+2,:)=-v(M+1,:);
end
```

```
clear all; close all; format long; clc
axisSize=14;
markersize=16;
linewidth=3.5;
Lx = 4;
Ly = 2;
Mv = [1 0.5 0.25]*128;
Nv = [1 0.5 0.25]*64;
CFL = 1;
Re = 2;
nIterMax=50;
tol=1e-12;
p1(1)=0;
p2(1)=0;
p3(1)=0;
for i=1:length(Mv)
```

```
    M=Mv(i);
    N=Nv(i);
        if (M==128 && N==64)
        outputTime=[0.1 0.5 1 2];
    else
        outputTime=[0.1 0.5 1];
    end
    endtime=outputTime(end);
    hx = Lx/M;
    hy = Ly/N;
    if hx~=hy
        error('Cells not square')
    end
```

```
    time=0;
%     dt=CFL*0.25*hx^2*Re;
    dt=5e-3;
    n=1;
    % Define the points of the different meshes
    xu=linspace(0,4,M+1);
    yu=linspace(-hy/2,2+hy/2,N+2);
    xv=linspace(-hx/2,4+hx/2,M+2);
    yv=linspace(0,2,N+1);
    % Define initial values
    phi=zeros(M+2,N+2);
    [u,v]=initialization(M,N,hx,hy);
    iter=1;
    t=0;
    while time < endtime
```

```
        if (time < outputTime(n) && time+dt >= outputTime(n))
            dt=outputTime(n)-time;
            n=n+1;
        else
%             dt=CFL*0.25*hx^2*Re;
            dt=5e-3;
        end
```

Solve for u(x,y,t)

```
        u=ADI_u(u,M,N,dt,hx,hy,Re);
```

Solve for v(x,y,t)

```
        v=ADI_v(v,M,N,dt,hx,hy,Re);
```

Outlet correction

```
        v=outletCorrection(u,v,xv,hx,hy);
```

Calculate rhs

```
        rhs=divV(u,v,M,N,hx,hy,dt);
```

Solve Poisson equation, V-cycle multigrid

```
        phi = poisson(phi,rhs,hx,hy,nIterMax,tol);
```

Project velocities using Lagrange multiplier

```
        [u,v]=LagProjection(u,v,phi,dt,hx,hy);
```

Next time step

```
        time=time+dt;
```

Filled contour plots, only for M=128 and N=64

```
        if (M==128 && N==64 && ismember(time,outputTime))
            % Plot u
            figure(n-1)
            contourf(xu,yu,u')
            hold on
            plot(1,0.5,'r.','markersize',markersize,'linewidth',linewidth)
            axis([0 4 0 2])
            caxis([-1.5 3])
            colorbar
            xlabel('$x$','Interpreter','latex')
            ylabel('$y$','Interpreter','latex')
            yticks([0 0.25 0.50 0.75 1.0 1.25 1.50 1.75 2.0]);
            xticks([0 0.50 1.0 1.50 2.0 2.50 3.0 3.50 4.0]);
            set(get(gca,'ylabel'),'rotation',0)
            set(gca,'fontsize',axisSize)
            pbaspect([2 1 1])
            grid on
            txt=['Latex/FIGURES/u_' num2str(n-1)];
            saveas(gcf,txt,'epsc')
```

```
            % Plot v
            figure(n+3)
            contourf(xv,yv,v')
            hold on
            plot(1,1.5,'r.','markersize',markersize,'linewidth',linewidth)
            axis([0 4 0 2])
            caxis([-3 1])
            colorbar
            xlabel('$x$','Interpreter','latex')
            ylabel('$y$','Interpreter','latex')
            yticks([0 0.25 0.50 0.75 1.0 1.25 1.50 1.75 2.0]);
            xticks([0 0.50 1.0 1.50 2.0 2.50 3.0 3.50 4.0]);
            set(get(gca,'ylabel'),'rotation',0)
            set(gca,'fontsize',axisSize)
            pbaspect([2 1 1])
            grid on
            txt=['Latex/FIGURES/v_' num2str(n-1)];
            saveas(gcf,txt,'epsc')
        end
```

Probes, only for M=128 and N=64

```
        if (M==128 && N==64 && time<=2+dt)
            iter=iter+1;
            t=[t;time];
            p1(iter) = (u (find(xu==1),find(yu<=0.5,1,'last'))...
                +u (find(xu==1),find(yu>=0.5,1)))/2;
            p2(iter) = (v (find(xv<=1,1,'last'),find(yv==1.5))...
                +v (find(xv>=1,1),find(yv==1.5)))/2;
        end
```

Probes value for u, v and Y at t=1

```
        if time==1
            p11(i)= (u (find(xu==1),find(yu<=0.5,1,'last'))...
                    +u (find(xu==1),find(yu>=0.5,1)))/2;
            p21(i)= (v (find(xv<=1,1,'last'),find(yv==1.5))...
                    +v (find(xv>=1,1),find(yv==1.5)))/2;
        end
```

```
    end
```

Plot the probes, only for M=128 and N=64

```
    if (M==128 && N==64)
        figure
        plot(t,p1,'linewidth',2)
        xlim([0 2])
        xlabel('$t$','Interpreter','latex')
        ylabel('$u(1,0.5,t)$','Interpreter','latex')
        set(gca,'fontsize',axisSize)
        grid on
        txt='Latex/FIGURES/probe1';
        saveas(gcf,txt,'epsc')
        figure
        plot(t,p2,'linewidth',2)
        xlim([0 2])
        xlabel('$t$','Interpreter','latex')
        ylabel('$v(1,1.5,t)$','Interpreter','latex')
        set(gca,'fontsize',axisSize)
        grid on
        txt='Latex/FIGURES/probe2';
        saveas(gcf,txt,'epsc')
    end
```

```
end
```

GCI analysis

```
clc
r=2
Fsec=1.25
% Probe 1
p_u=log((p11(3)-p11(2))/(p11(2)-p11(1)))/log(r)
u_h0=p11(1)+(p11(1)-p11(2))/(r^p_u-1)
GCI21_u=Fsec*(p11(2)-p11(1))/(p11(1)*(r^p_u-1))
```

```
GCI32_u=Fsec*(p11(3)-p11(2))/(p11(2)*(r^p_u-1))
coeff_u=GCI21_u*r^p_u/GCI32_u
percent_u=GCI21_u*100
pause

% Probe 2
p_v=log((p21(3)-p21(2))/(p21(2)-p21(1)))/log(r)
v_h0=p21(1)+(p21(1)-p21(2))/(r^p_v-1)
GCI21_v=Fsec*(p21(2)-p21(1))/(p21(1)*(r^p_v-1))
GCI32_v=Fsec*(p21(3)-p21(2))/(p21(2)*(r^p_v-1))
coeff_v=GCI21_v*r^p_v/GCI32_v
percent_v=GCI21_v*100
% pause
```

POISSON EQUATION

$$\nabla \cdot (\nabla \varphi^{n+1}) = \frac{1}{\Delta t} \nabla \cdot \vec{v}^{*}$$

$$\nabla^2 \varphi^{n+1} = \frac{1}{\Delta t} \nabla \cdot \vec{v}^{*}$$

$$\frac{\partial^2 \varphi^{n+1}}{\partial x^2} + \frac{\partial^2 \varphi^{n+1}}{\partial y^2} = \frac{1}{\Delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right)$$

Since we are trying to obtain φ^{n+1}, we will discretize the derivatives at the cell centers:

$$\left.\frac{\partial^2 \varphi^{n+1}}{\partial x^2}\right|_{i,j} = \frac{\varphi_{i+1,j}^{n+1} - 2\varphi_{i,j}^{n+1} + \varphi_{i-1,j}^{n+1}}{\Delta x^2} + O(\Delta x^2)$$

$$\left.\frac{\partial^2 \varphi^{n+1}}{\partial y^2}\right|_{i,j} = \frac{\varphi_{i,j+1}^{n+1} - 2\varphi_{i,j}^{n+1} + \varphi_{i,j-1}^{n+1}}{\Delta y^2} + O(\Delta y^2)$$

$$\left.\frac{\partial u^*}{\partial x}\right|_{i,j} = \frac{u^*_{i+\frac{1}{2},j} - u^*_{i-\frac{1}{2},j}}{\cancel{2}\frac{\Delta x}{\cancel{2}}} + O(\Delta x^2)$$

$$\left.\frac{\partial v^*}{\partial y}\right|_{i,j} = \frac{v^*_{i,j+\frac{1}{2}} - v^*_{i,j-\frac{1}{2}}}{\Delta y} + O(\Delta y^2)$$

Thus, we will solve

$$\frac{\varphi_{i+1,j}^{n+1} - 2\varphi_{i,j}^{n+1} + \varphi_{i-1,j}^{n+1}}{\Delta x^2} + \frac{\varphi_{i,j+1}^{n+1} - 2\varphi_{i,j}^{n+1} + \varphi_{i,j-1}^{n+1}}{\Delta y^2} = \frac{1}{\Delta t}\left(\frac{u^*_{i+\frac{1}{2},j} - u^*_{i-\frac{1}{2},j}}{\Delta x} + \frac{v^*_{i,j+\frac{1}{2}} - v^*_{i,j-\frac{1}{2}}}{\Delta y}\right)$$

With zero Neumann Boundary Conditions:

$$\varphi_{i,1}^{n+1} = \varphi_{i,2}^{n+1} \quad \text{(Bottom)}$$

$$\varphi_{i,N+2}^{n+1} = \varphi_{i,N+1}^{n+1} \quad \text{(Top)}$$

NOTE: Matlab indices used.

$$\varphi_{1,j}^{n+1} = \varphi_{2,j}^{n+1} \quad \text{(Left)}$$

$$\varphi_{M+2,j}^{n+1} = \varphi_{M+1,j}^{n+1} \quad \text{(Right)}$$

PROJECTION STEP

$$\frac{\partial \vec{v}}{\partial t} = -\nabla \varphi^{n+1} \Rightarrow \frac{\vec{v}^{n+1} - \vec{v}^*}{\Delta t} = -\nabla \varphi^{n+1}$$

$$\vec{v}^{n+1} = \vec{v}^* - \Delta t \nabla \varphi^{n+1} \quad \begin{cases} u^{n+1} = u^* - \Delta t \dfrac{\partial \varphi}{\partial x} \\ v^{n+1} = v^* - \Delta t \dfrac{\partial \varphi}{\partial y} \end{cases}$$

Since we want to obtain velocities, we will discretize the derivatives centering at the staggered mesh.

$$\left.\frac{\partial \varphi}{\partial x}\right|_{i+\frac{1}{2},j} = \frac{\varphi_{i+1,j}^{n+1} - \varphi_{i,j}^{n+1}}{\Delta x} + O(\Delta x^2)$$

$$\left.\frac{\partial \varphi}{\partial y}\right|_{i,j+\frac{1}{2}} = \frac{\varphi_{i,j+1}^{n+1} - \varphi_{i,j}^{n+1}}{\Delta y} + O(\Delta y^2)$$

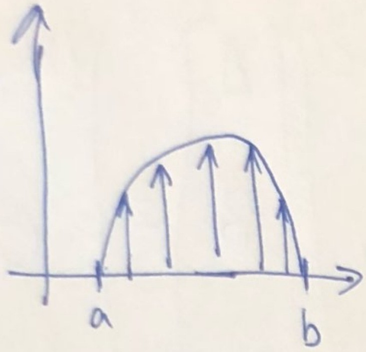
No BCs are needed since the zero Neumann BCs. for φ leaves the boundary velocities uncorrected.

Thus, we will solve:

$$u_{i+\frac{1}{2},j}^{n+1} = u_{i+\frac{1}{2},j}^{*} - \Delta t \frac{\varphi_{i+1,j}^{n+1} - \varphi_{i,j}^{n+1}}{\Delta x}$$

$$v_{i,j+\frac{1}{2}}^{n+1} = v_{i,j+\frac{1}{2}}^{*} - \Delta t \frac{\varphi_{i,j+1}^{n+1} - \varphi_{i,j}^{n+1}}{\Delta y}$$

INLETS COEFFICIENTS



$$v(x) = Ax^2 + Bx + C$$

$$v(a) = Aa^2 + Ba + C = 0 \quad (1)$$

$$v(b) = Ab^2 + Bb + C = 0 \quad (2)$$

$$v_{avg} = \frac{1}{b-a}\int_a^b v(x)\,dx = \frac{1}{b-a}\left[\frac{1}{3}A(b^3-a^3) + \frac{1}{2}B(b^2-a^2) + C(b-a)\right]$$

Hence,

$$\frac{1}{3}(b^3-a^3)A + \frac{1}{2}(b^2-a^2)B + (b-a)C = (b-a)\,v_{avg} \quad (3)$$

The system of equations (1), (2), (3) can be expressed as

$$\begin{bmatrix} \frac{1}{3}(b^3-a^3) & \frac{1}{2}(b^2-a^2) & (b-a) \\ a^2 & a & 1 \\ b^2 & b & 1 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} (b-a)\,v_{avg} \\ 0 \\ 0 \end{bmatrix}$$

$r_3 - \frac{b^2}{a^2} r_2$:

$$\begin{bmatrix} \frac{1}{3}(b^3-a^3) & \frac{1}{2}(b^2-a^2) & (b-a) \\ a^2 & a & 1 \\ 0 & b - \frac{b^2}{a} & 1 - \frac{b^2}{a^2} \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} (b-a)\,v_{avg} \\ 0 \\ 0 \end{bmatrix}$$

$r_1-(b-a)r_2$:

$$\begin{bmatrix} \frac{1}{3}(b^3-a^3)-(b-a)a^2 & \frac{1}{2}(b^2-a^2)-(b-a)a & 0 \\ a^2 & a & 1 \\ 0 & b-\frac{b^2}{a} & 1-\frac{b^2}{a^2} \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \begin{bmatrix} (b-a)\,v_{avg} \\ 0 \\ 0 \end{bmatrix}$$

We can obtain A, B, C using our tridiagonal solver. It also works for horizontal inlets.

I have obtained:

Inlet 1 $\begin{cases} A=-48 \\ B=72 \\ C=-24 \end{cases}$

Inlet 2 $\begin{cases} A=24 \\ B=-60 \\ C=36 \end{cases}$

Inlet 3 $\begin{cases} A=24 \\ B=-36 \\ C=12 \end{cases}$