Computational Fluid Dynamics

Final Project Report

Yutian Pang

1. PDE solved

We solved non-dimensionalized full 2D Navier-Stokes equation for u and v at new time level,

$$\frac{\partial \mathrm{u}}{\partial \mathrm{x}}+\frac{\partial \mathrm{v}}{\partial \mathrm{y}}=0$$

$$\frac{\partial \mathrm{u}}{\partial \mathrm{t}}+\frac{\partial (uu)}{\partial x}+\frac{\partial (uu)}{\partial x}=-\frac{\partial p}{\partial x}+\frac{1}{Re}(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2})$$

$$\frac{\partial \mathrm{v}}{\partial \mathrm{t}}+\frac{\partial (uv)}{\partial x}+\frac{\partial (vv)}{\partial y}=-\frac{\partial p}{\partial y}+\frac{1}{Re}(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2})$$

and use the mass fraction equation below to solve Y,

$$\frac{\partial \mathrm{Y}}{\partial \mathrm{t}}+u\frac{\partial Y}{\partial x}+v\frac{\partial Y}{\partial y}=\alpha(\frac{\partial^2 Y}{\partial x^2}+\frac{\partial^2 Y}{\partial y^2})$$

We use 2D viscous Burgers Equation to find u* and v*,

$$\frac{\partial \mathrm{u}}{\partial \mathrm{t}}+\frac{\partial (uu)}{\partial x}+\frac{\partial (uv)}{\partial y}=\nu(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2})$$

$$\frac{\partial \mathrm{v}}{\partial \mathrm{t}}+\frac{\partial (uv)}{\partial x}+\frac{\partial (vv)}{\partial y}=\nu(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2})$$

However, u* and v* are not the u and v at new time level since we didn't consider the influence of p. Thus after we calculated u* and v* as well as corrected the outlet velocity to satisfies the conservation law, we use Possion equation solver to calculate the Lagrange multiplier φ,

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

Eventually we can apply the influence of p by projecting/correcting velocities using Lagrange multiplierφ. Then we can calculate Y at new time level using the solve Y equation above.

2. Name of method used to solve the final project

2.1 To solve 2D Burgers Equation for u* and v*, I used Adams-Bashforth and Crank-Nicholson solved by ADI method. I can calculate the Adams-Bashforth part at the beginning of each time step and use FTCS for hyperbolic part in first time step to deal with the start up issue.

2.2 To solve the mass fraction equation, I used WENO-5/TVD RK3 for the convective terms and Crank-Nicholson solved by ADI for the diffusive terms.

2.3 To solve the Possion equation, I used a full V-cycle multigrid method and perform one Gauss-Seidel iteration at each mesh level. The coarsest mesh is 25 by 5 in V cycle.

3. Mesh resolutions, CFL numbers, Poisson equation convergence criteria use

3.1 The CFL numbers I used in this whole project is 0.95.

3.2 The finest mesh size I used for GCI analysis is [M,N]=[400,80] and the mesh size for all the plots in Section 5 is [800,160] in order to have a better resolution of the figure. I used the computer in Noble Library to plot these figures.

3.3 The convergence criteria I used for Possion equation is absolute convergence which means the infinity norm of the finest mesh residual below the threshold α where α is 0.1 with reference to the setting of hw5.

4. Value of R at steady state

GCI analysis for Original Geometry:

Table 4.1 Results of R for GCI analysis

[M,N]	C	R
[50,10]	0.95	0.0965618555581060
[100,20]	0.95	0.0824682611956138
[200,40]	0.95	0.0717623146822650
[400,80]	0.95	0.0683107606064166

Table 4.2 GCI results when C=0.95

M	N	p	k	GCI21	GCI23	f0 Results with error bar
[200,100,50]	[40,20,10]	0.3966	1.1492	0.589339370442249	0.675105667208754	0.0379280 +/- 58.9339%
[400,200,100]	[80,40,20]	1.6331	1.0505	0.030050327544301	0.088726281391913	0.0666690 +/- 3.00503%

Thus my predicted steady state value R for the chamber is,

0.0666690 +/- 3.00503%

If we use the finer mesh results, then the real number should be in interval,

[0.064665576549300, 0.068672423450700]

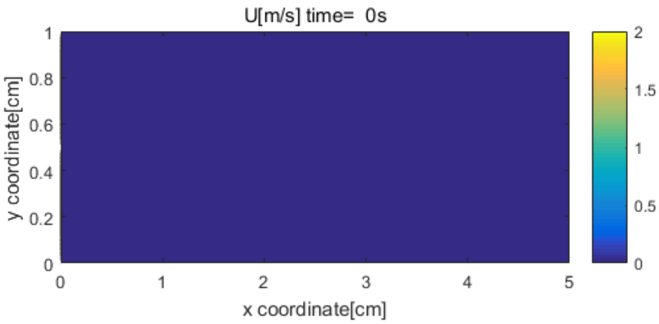
Our goal of assignment 3 is to improve the performance R by 10% percent which is,

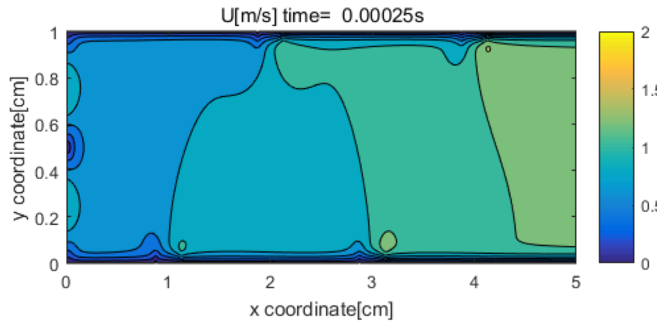
0.068672423450700*1.1=0.075539665795770

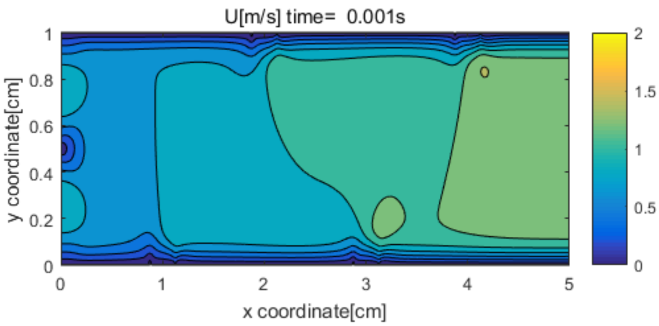
Thus my improved R_{lower} with new geometry should larger than **0.075539665795770**

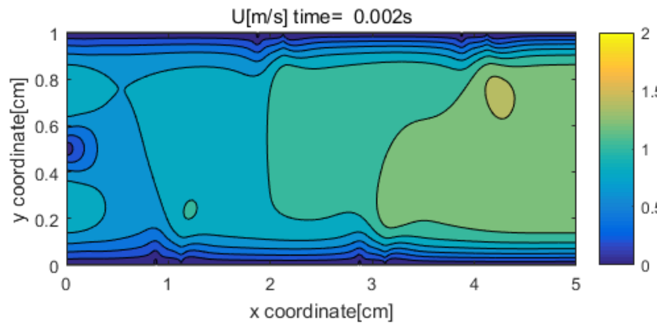
5. Plots of steady flow in the original chamber (M=800,N=160,CFL=0.95)

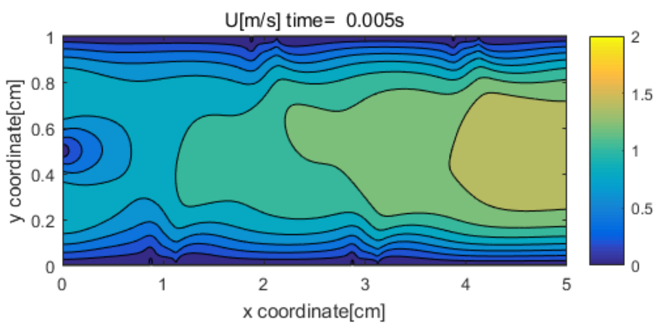
5.1 U velocity plots

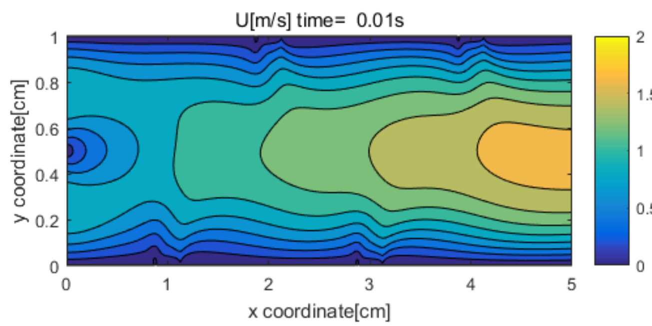


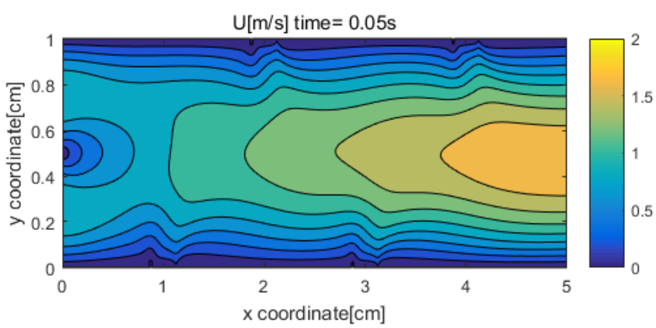


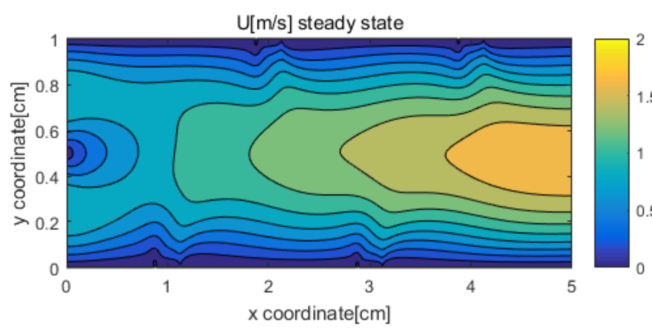




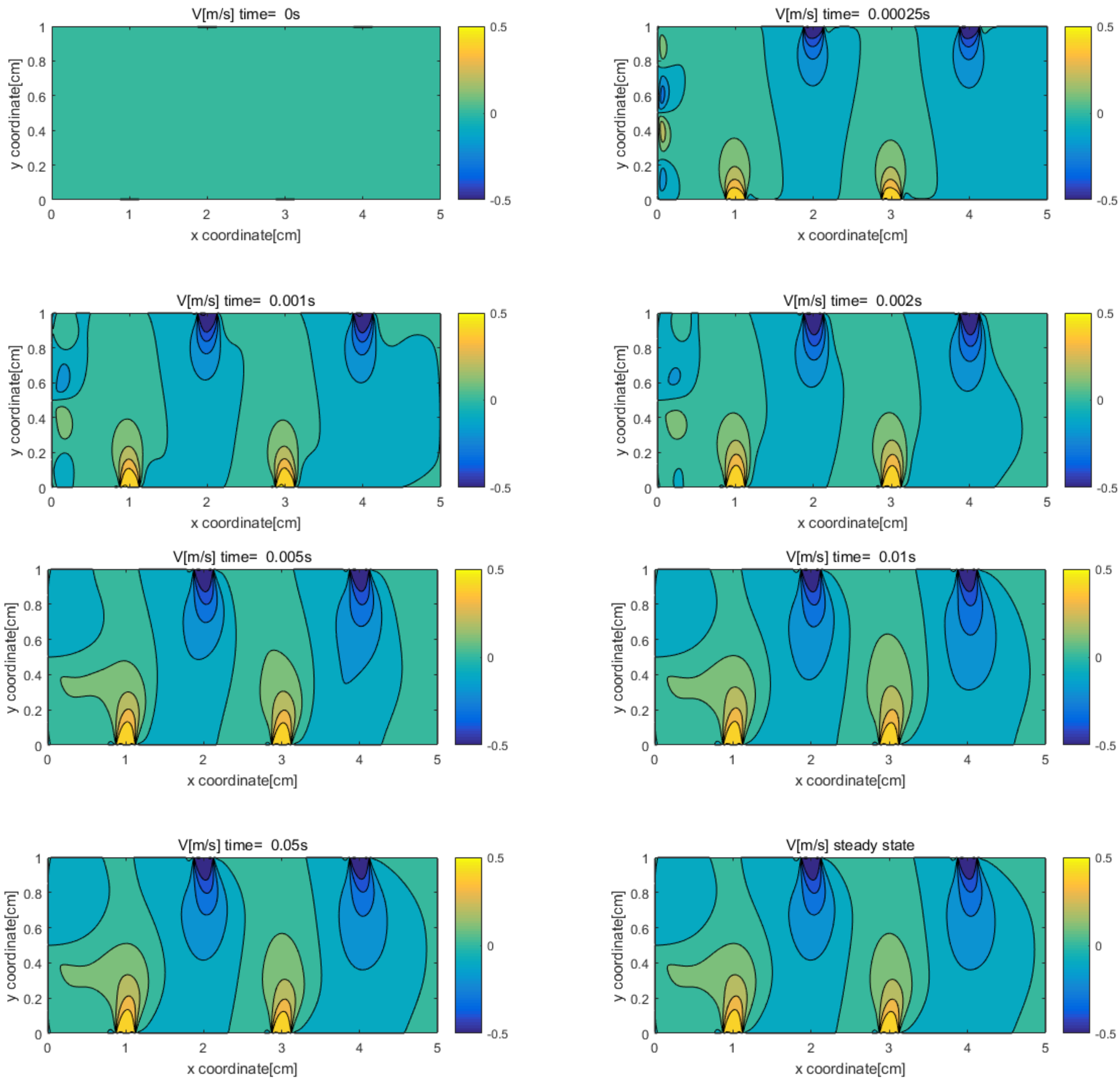




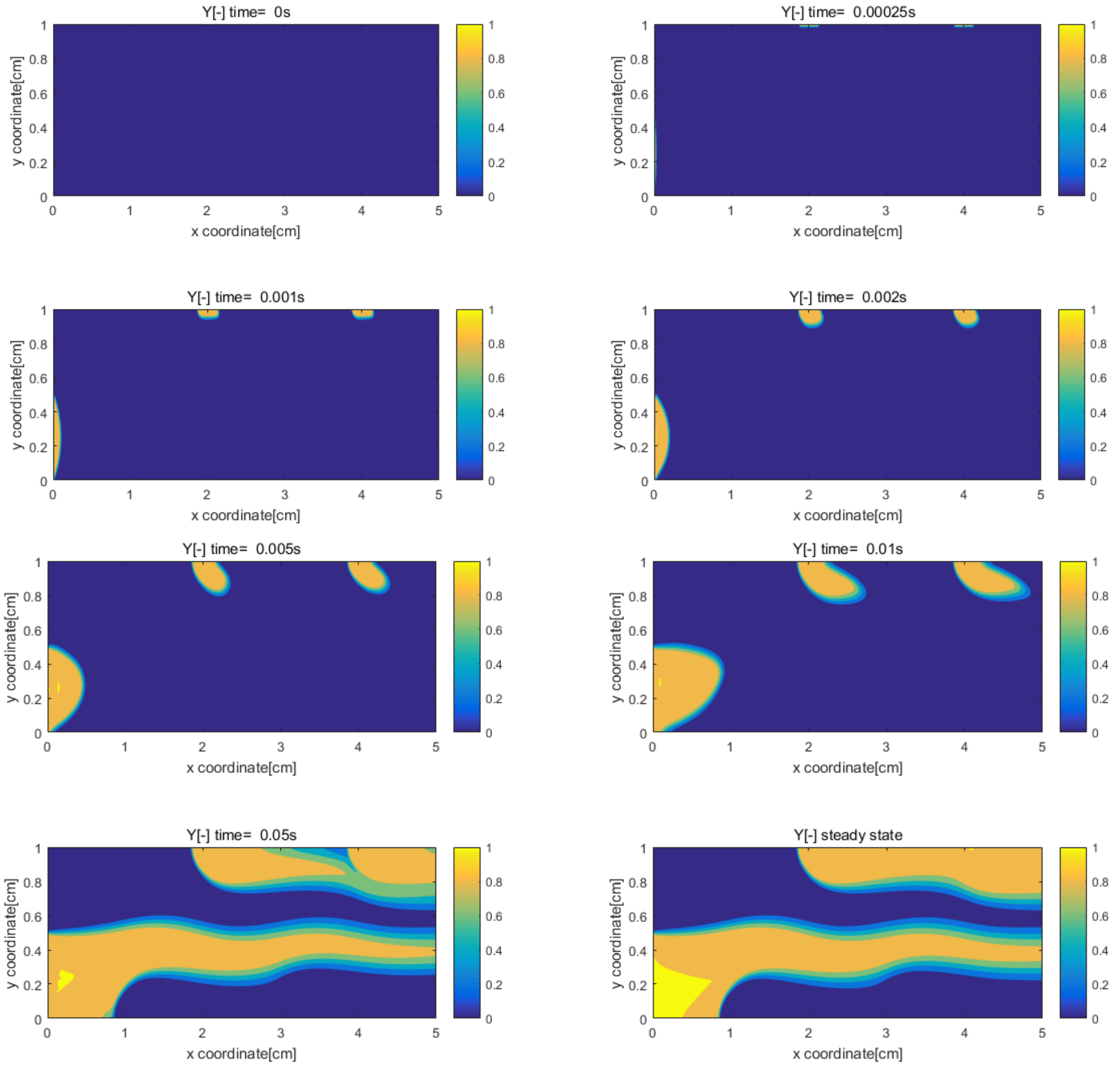




5.2 V velocity plots

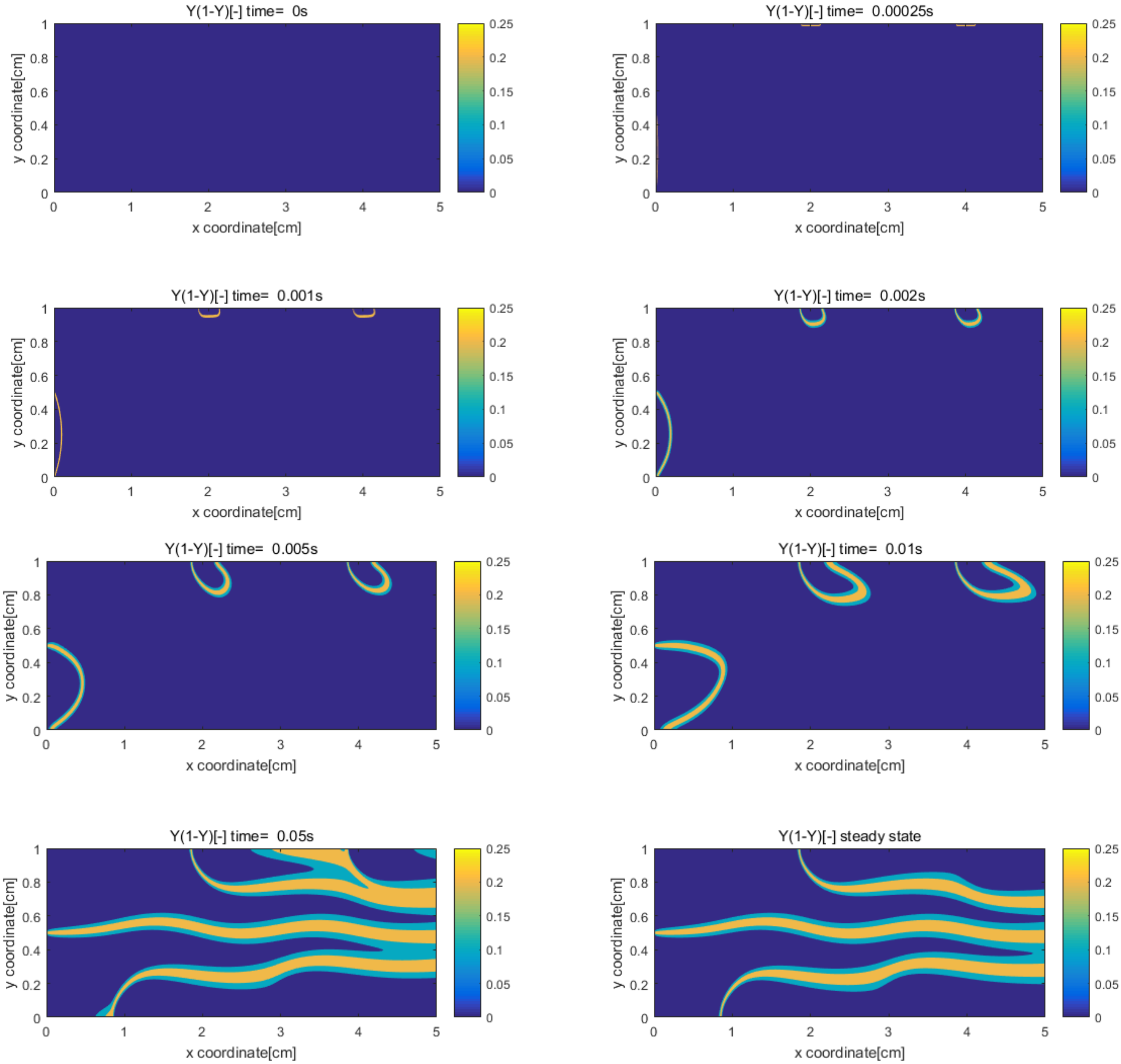


5.3 Y plots



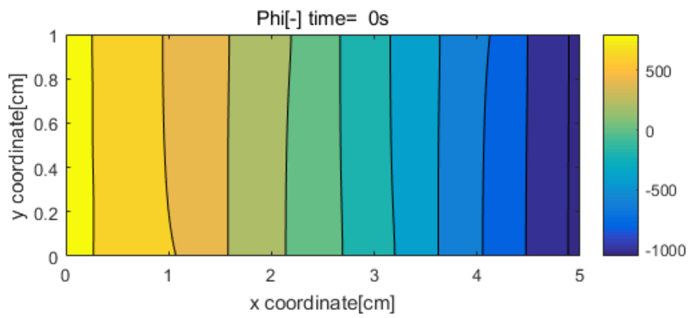
Y[-] time= 0s
Y[-] time= 0.00025s
Y[-] time= 0.001s
Y[-] time= 0.002s
Y[-] time= 0.005s
Y[-] time= 0.01s
Y[-] time= 0.05s
Y[-] steady state
x coordinate[cm]
y coordinate[cm]
0
1
2
3
4
5
0.2
0.4
0.6
0.8

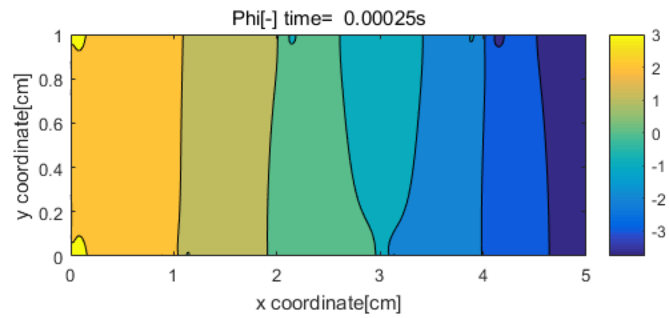
5.4 Q plots

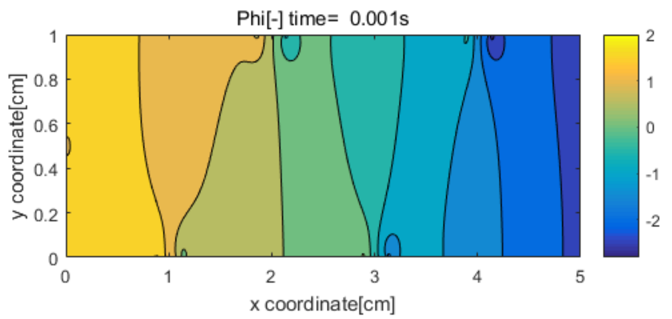


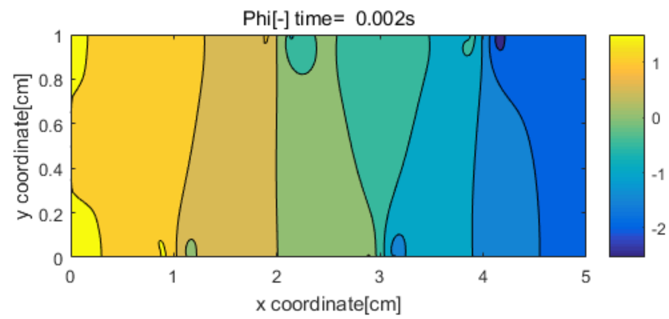
Y(1-Y)[-] time= 0s
Y(1-Y)[-] time= 0.00025s
Y(1-Y)[-] time= 0.001s
Y(1-Y)[-] time= 0.002s
Y(1-Y)[-] time= 0.005s
Y(1-Y)[-] time= 0.01s
Y(1-Y)[-] time= 0.05s
Y(1-Y)[-] steady state
x coordinate[cm]
y coordinate[cm]

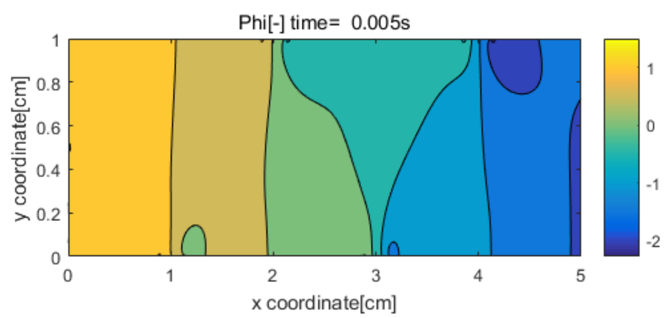
5.5 Phi Plots

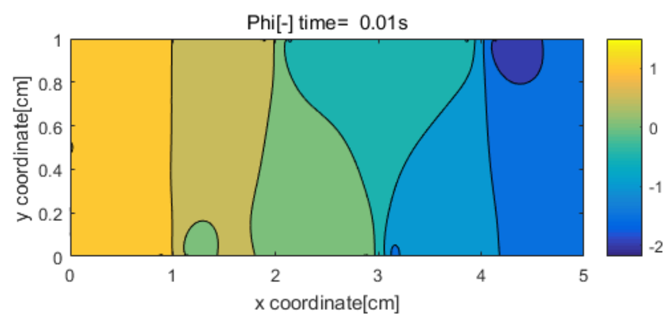

Phi[-] time= 0s
y coordinate[cm]
x coordinate[cm]
500
0
-500
-1000

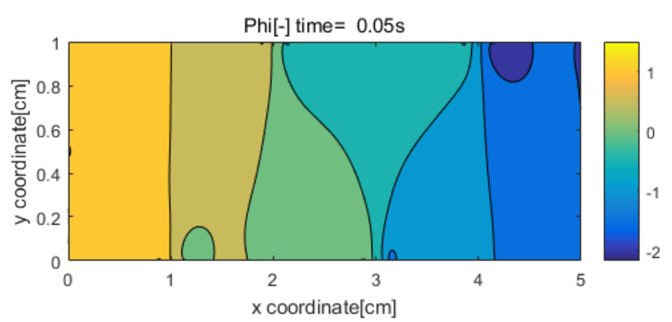

Phi[-] time= 0.00025s
y coordinate[cm]
x coordinate[cm]

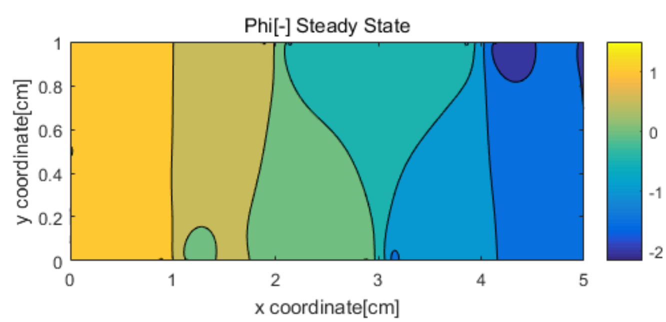

Phi[-] time= 0.001s
y coordinate[cm]
x coordinate[cm]


Phi[-] time= 0.002s
y coordinate[cm]
x coordinate[cm]

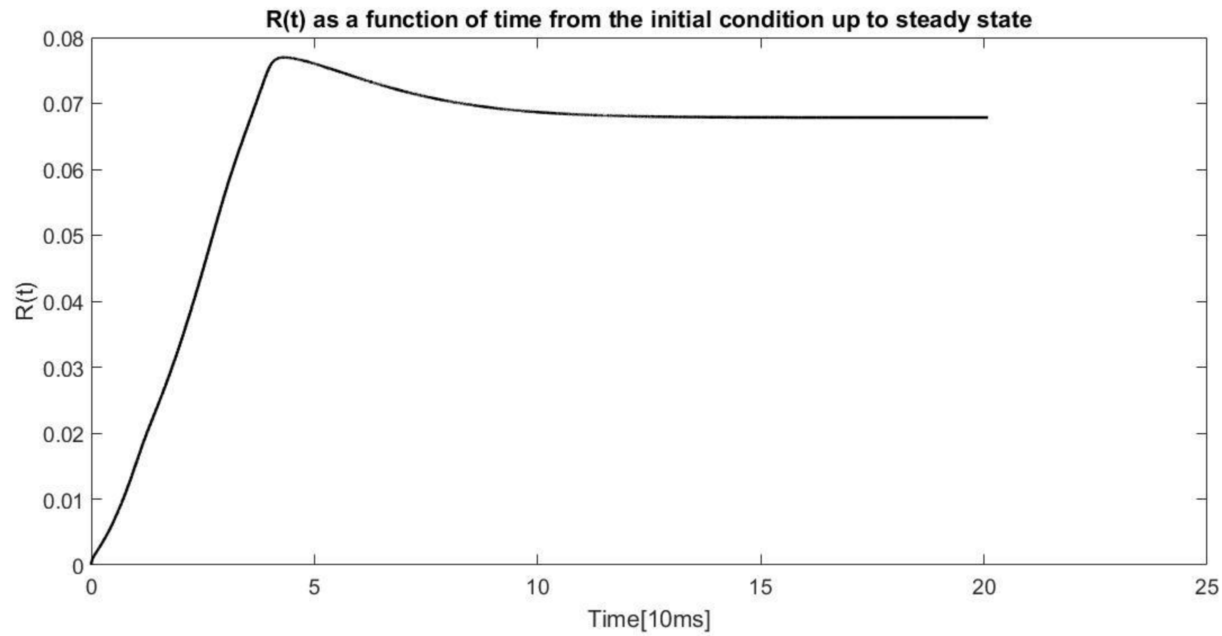

Phi[-] time= 0.005s
y coordinate[cm]
x coordinate[cm]


Phi[-] time= 0.01s
y coordinate[cm]
x coordinate[cm]


Phi[-] time= 0.05s
y coordinate[cm]
x coordinate[cm]


Phi[-] Steady State
y coordinate[cm]
x coordinate[cm]

5.6 R(t) plot (M=800,N=160,C=0.95)



6. Modified Geometry

6.1 Exchange the choice of A and B into the slits

Table 6.1.1 Results of R for GCI analysis

[M,N]	C	Steady State R
[50,10]	0.95	0.0393562588479625
[100,20]	0.95	0.0311436706350090
[200,40]	0.95	0.0278868687365147
[400,80]	0.95	0.0273231020203095

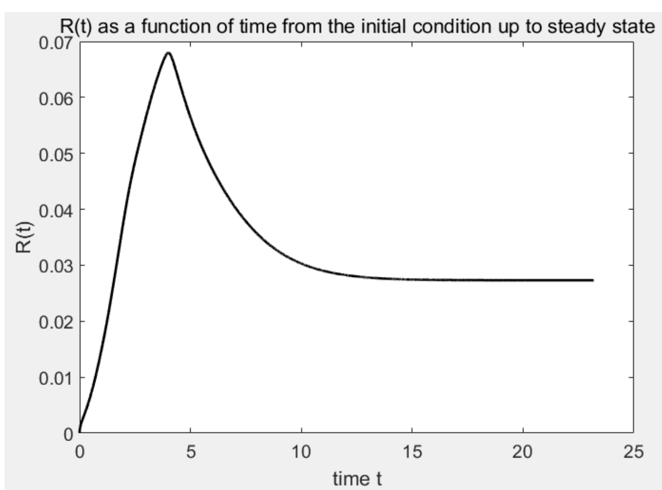


Figure 6.1.1 R(t) when M=400,N=80

Table 6.1.2 GCI results when C=0.95

M	N	p	k	GCI21	GCI23	f0 Results with error bar
[200,100,50]	[40,20,10]	1.3344	1.1168	0.0959357275767616	0.216620240249054	0.0257470 +/- 9.59357%
[400,200,100]	[80,40,20]	2.5303	1.0206	0.00539929239171615	0.0305603981632021	0.0272050 +/- 0.539929%

Comments:

0.0272050 *(1 + 0.539929%) = 0.027351887684450

(0.064665576549300-0.027351887684450)/0.064665576549300=57.70%

The modification of oil A and B into the slits cannot improve the performance. It can decrease R by 57.70% instead. Thus this is an unsuccessful choice.

6.2 All slits oil A (v=0.25m/s)

Table 6.2.1 Results of R for GCI analysis

[M,N]	C	Steady State R
[50,10]	0.95	0.0607254442667625
[100,20]	0.95	0.0571541265276045
[200,40]	0.95	0.0530658609520460
[400,80]	0.95	0.0516978946635019

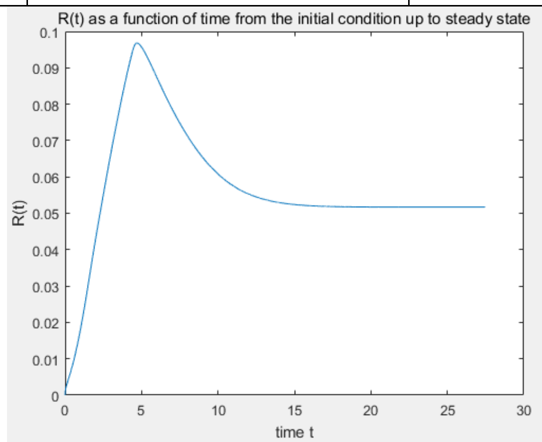


Figure 6.2.1 R(t) when M=400,N=80

Table 6.2.2 GCI results when C=0.95

M	N	p	k	GCI21	GCI23	f0 Results with error bar
[200,100,50]	[40,20,10]	-0.1950	1.0770	-0.761598844233402	-0.617708091170443	0.0853980 +/- 76.1599%
[400,200,100]	[80,40,20]	1.5795	1.0265	0.0166330256775876	0.0484275619765721	0.0510100 +/- 1.6633%

Comments:

0.0510100 *（1-1.6633%）=0.0501615506700000

(0.064665576549300-0.0501615506700000)/0.064665576549300=22.43%

The modification of oil A goes into all the fours slits with velocity of 0.25m/s cannot improve the performance. It can decrease R by 22.43% instead. Thus this is an unsuccessful choice.

6.3 Change the velocity at slits2&4 to 0.25

Table 6.3.1 Results of R for GCI analysis

[M,N]	C	Steady State R
[50,10]	0.95	0.0984091396735665
[100,20]	0.95	0.0865528183785162
[200,40]	0.95	0.0770929122640815
[400,80]	0.95	0.0740170140551861

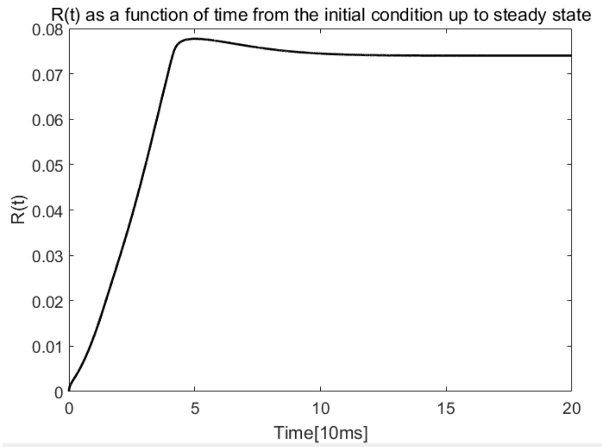


Figure 6.3.1 R(t) when M=400,N=80

Table 6.3.2 GCI results when C=0.95

M	N	p	k	GCI21	GCI23	f0 Results with error bar
[200,100,50]	[40,20,10]	0.3258	1.1227	0.605490246197580	0.675932802257912	0.0397500 +/- 60.549%
[400,200,100]	[80,40,20]	1.6208	1.0416	0.0250281618154092	0.0739028063496876	0.0725350 +/- 2.50282%

Comments:

Minimum improvements:

0.0725350 *(1 - 2.50282%) = 0.070719579513000

(0.070719579513000-0.068672423450700)/ 0.068672423450700=2.98%<10%

Even though this time still now meet with the requirement 10%. I found the correct way to increase R instead of just decrease R.

6.4 Change the velocity at slits2&4 to 0.1m/s

Table 6.4.1 Results of R for GCI analysis

[M,N]	C	Steady State R
[50,10]	0.95	0.0978065649412050
[100,20]	0.95	0.0874608544221607
[200,40]	0.95	0.0795462619544187
[400,80]	0.95	0.0771765157818150

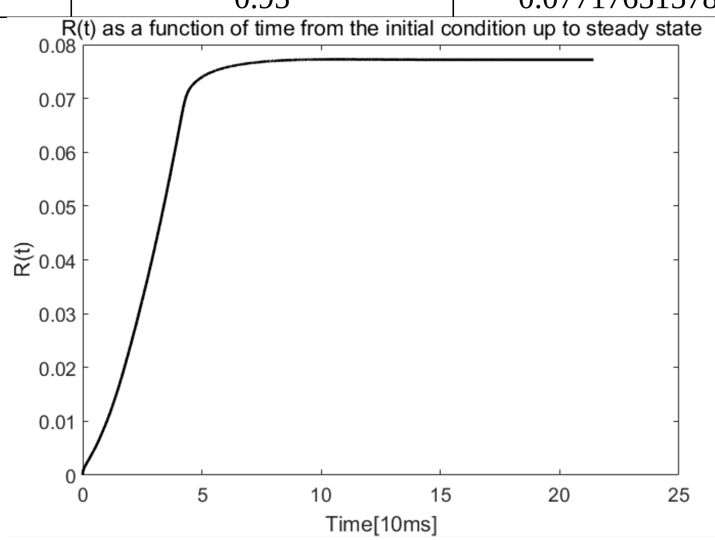


Figure 6.4.1 R(t) when M=400,N=80

Table 6.4.2 GCI results when C=0.95

M	N	p	k	GCI21	GCI23	f0 Results with error bar
[200,100,50]	[40,20,10]	0.3864	1.0995	0.404893965144099	0.481370120129146	0.0537800 +/- 40.4894%
[400,200,100]	[80,40,20]	1.7398	1.0307	0.0164035933391169	0.0531534077414255	0.0761640 +/- 1.64036%

Comments:
Minimum improvements:

0.0761640 *(1 - 1.64036%) = 0.074914636209600

(0.074914636209600-0.068672423450700)/ 0.068672423450700=9.09%<10%

Thus change the velocity at slits 2&4 to 0.1m/s can improve R by 9.09% which is close to the required improvements percentage.

6.5 Change all the velocity to 0.15m/s

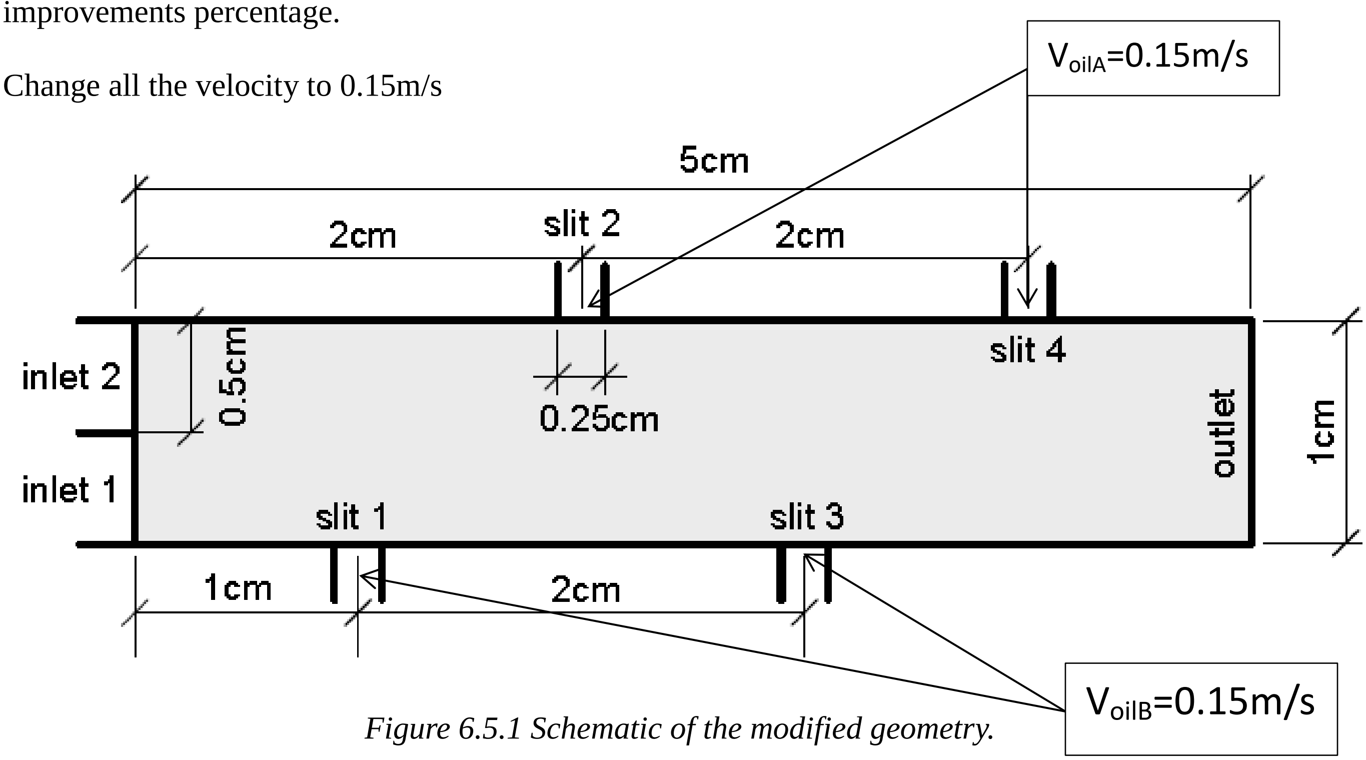


Figure 6.5.1 Schematic of the modified geometry.

Table 6.5.1 Results of R for GCI analysis

[M,N]	C	Steady State R
[100,20]	0.95	0.096404229738922
[200,40]	0.95	0.092441647479612
[400,80]	0.95	0.090933840746061

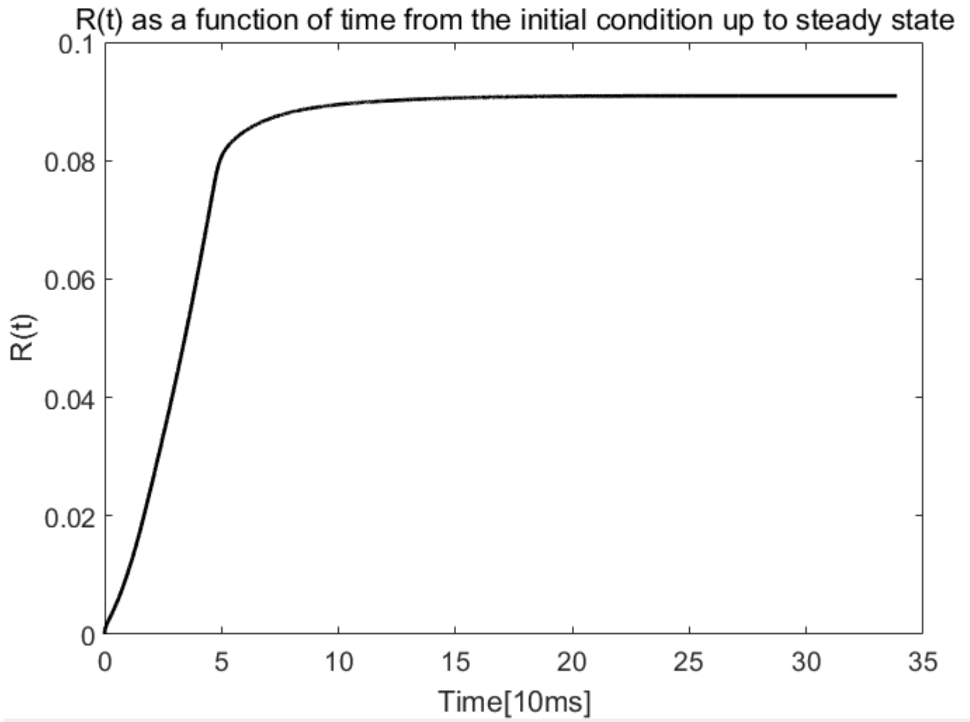


Figure 6.5.2 R(t) when M=400,N=80

Table 6.5.2 GCI results when C=0.95

M	N	p	k	GCI21	GCI23	f0 Results with error bar
[400,200,100]	[80,40,20]	1.3940	1.0166	0.012731044 9219705	0.032912018 6772436	0.0900080 +/- 1.2731%

Comments:
Minimum improvements:

0.0900080*(1 - 1.2731%) = 0.088862108152000

(0.088862108152000-0.068672423450700)/0.068672423450700 = 29.40%>10%

Maximum improvements:

0.0900080*(1 + 1.2731%) = 0.091153891848000

(0.091153891848000-0.064665576549300)/0.064665576549300 = 40.96%>10%

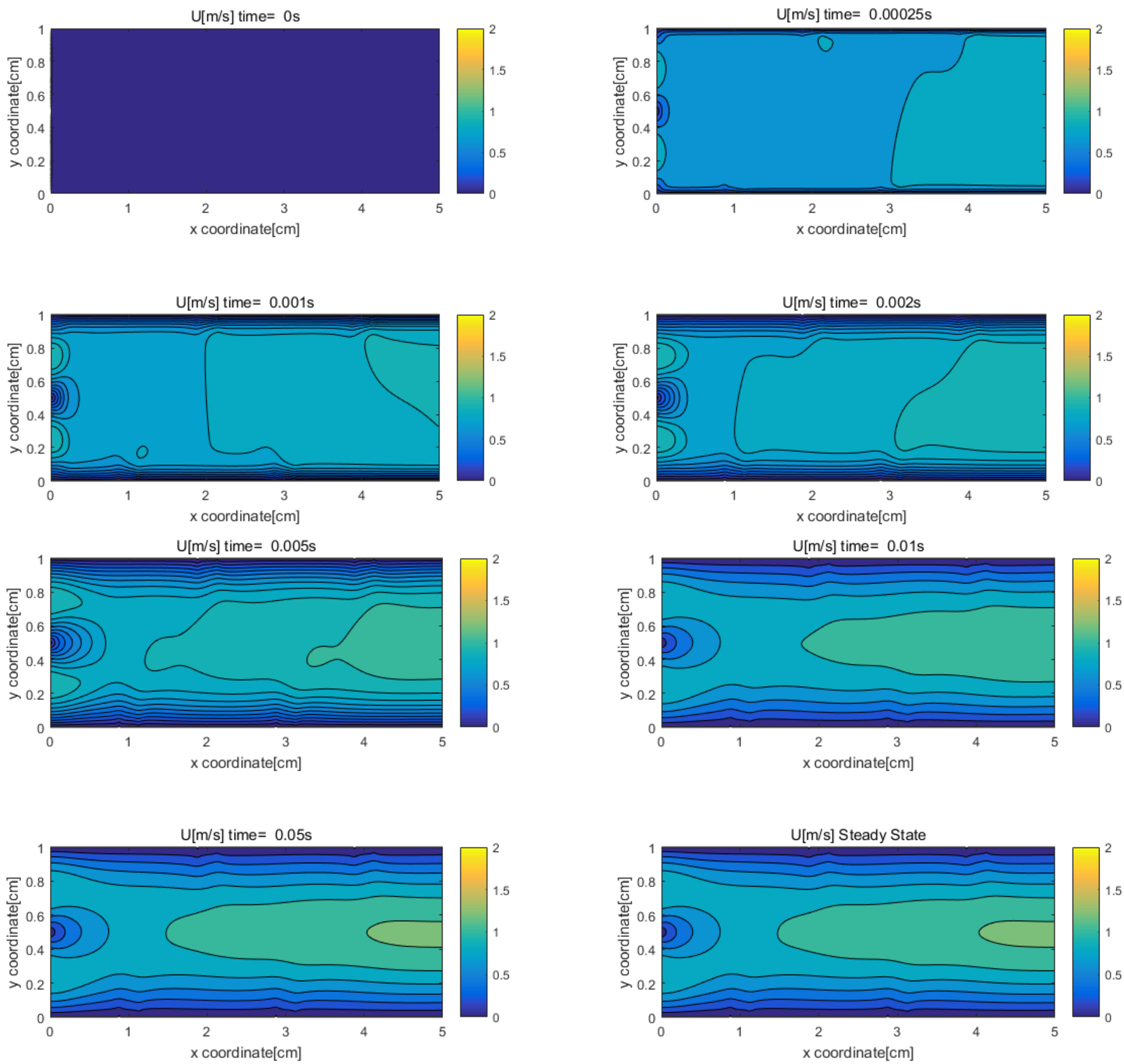
Thus this method can improve R by at least 29.40% and at most 40.96%
The final improvement represented with error bar is

(35.18% +/- 5.78%)

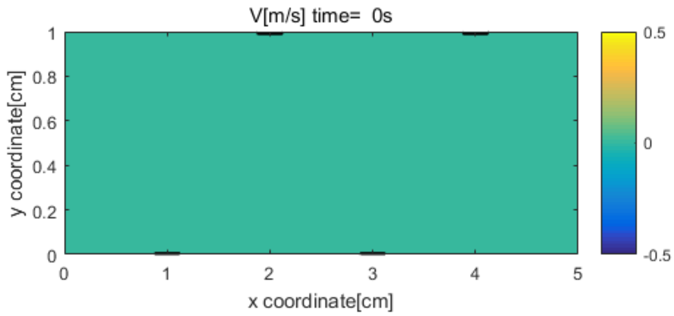
Plots of U, V, Y, Y(1-Y), Phi after modification 6.5 for over 10% improvement

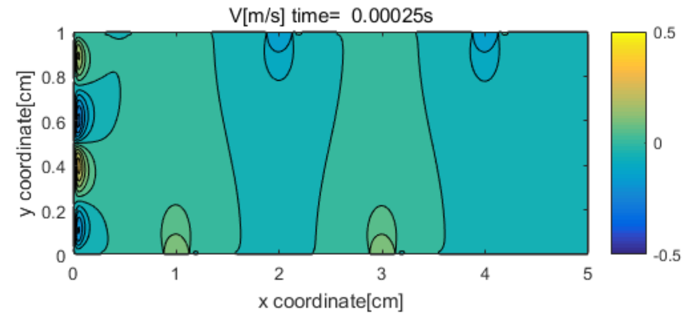
M=400,N=80,C=0.95

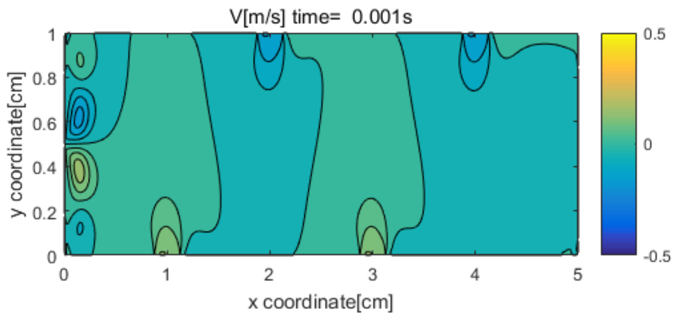
6.5.1 U Plots

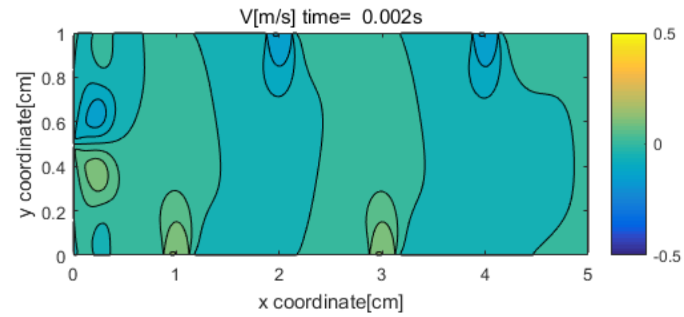


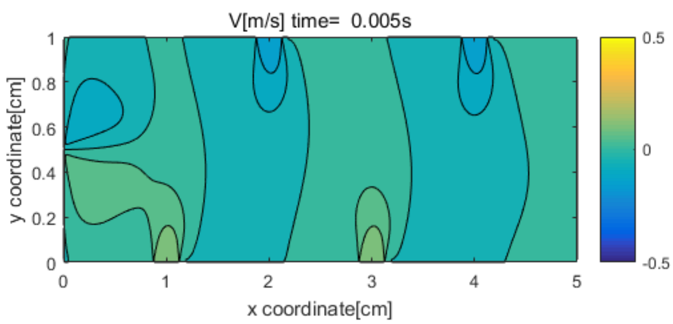
6.5.2 V Plots

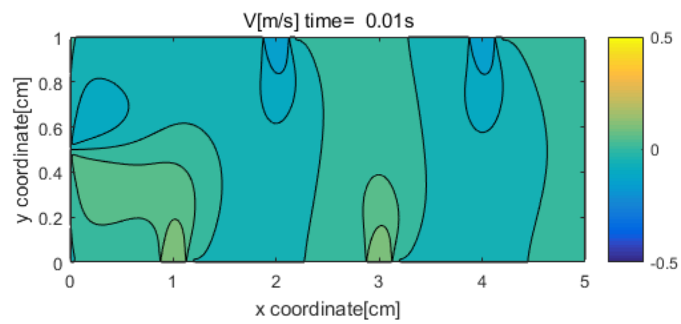

V[m/s] time= 0s
y coordinate[cm]
x coordinate[cm]
0 0.2 0.4 0.6 0.8 1
0 1 2 3 4 5
0.5 0 -0.5

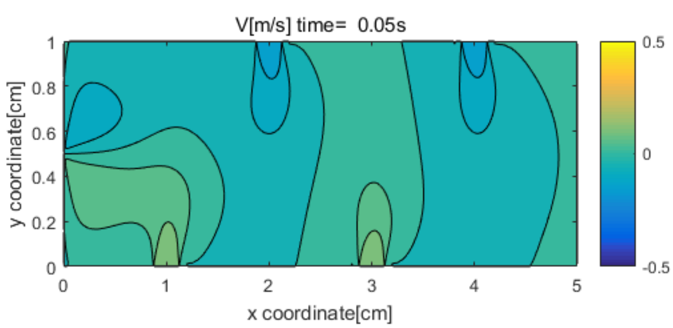

V[m/s] time= 0.00025s
y coordinate[cm]
x coordinate[cm]
0 0.2 0.4 0.6 0.8 1
0 1 2 3 4 5
0.5 0 -0.5

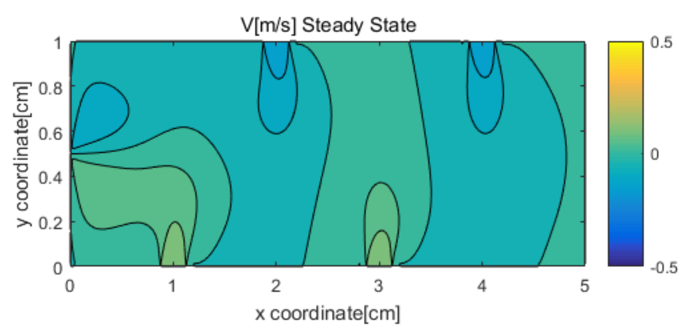

V[m/s] time= 0.001s
y coordinate[cm]
x coordinate[cm]
0 0.2 0.4 0.6 0.8 1
0 1 2 3 4 5
0.5 0 -0.5


V[m/s] time= 0.002s
y coordinate[cm]
x coordinate[cm]
0 0.2 0.4 0.6 0.8 1
0 1 2 3 4 5
0.5 0 -0.5

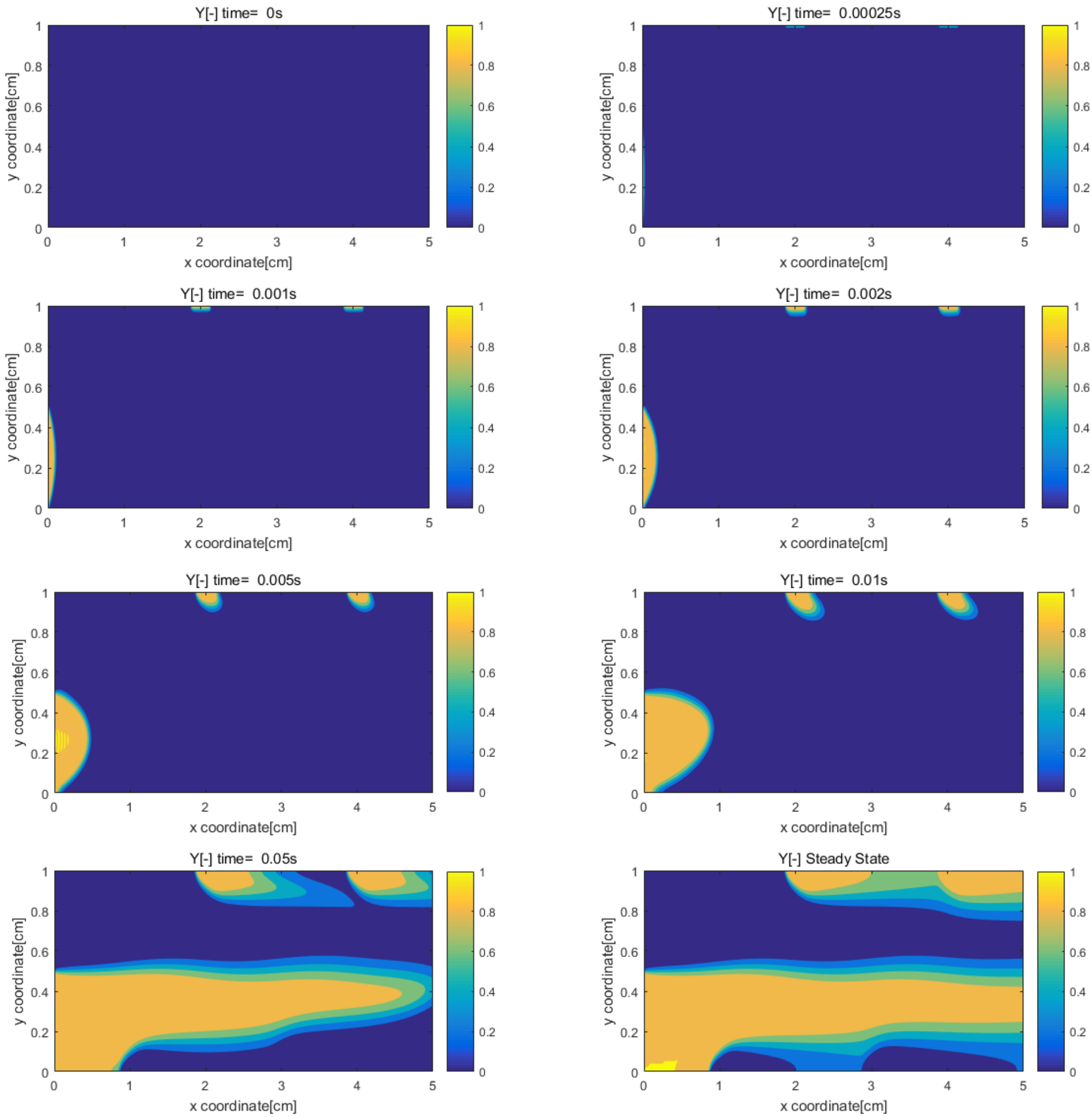

V[m/s] time= 0.005s
y coordinate[cm]
x coordinate[cm]
0 0.2 0.4 0.6 0.8 1
0 1 2 3 4 5
0.5 0 -0.5


V[m/s] time= 0.01s
y coordinate[cm]
x coordinate[cm]
0 0.2 0.4 0.6 0.8 1
0 1 2 3 4 5
0.5 0 -0.5

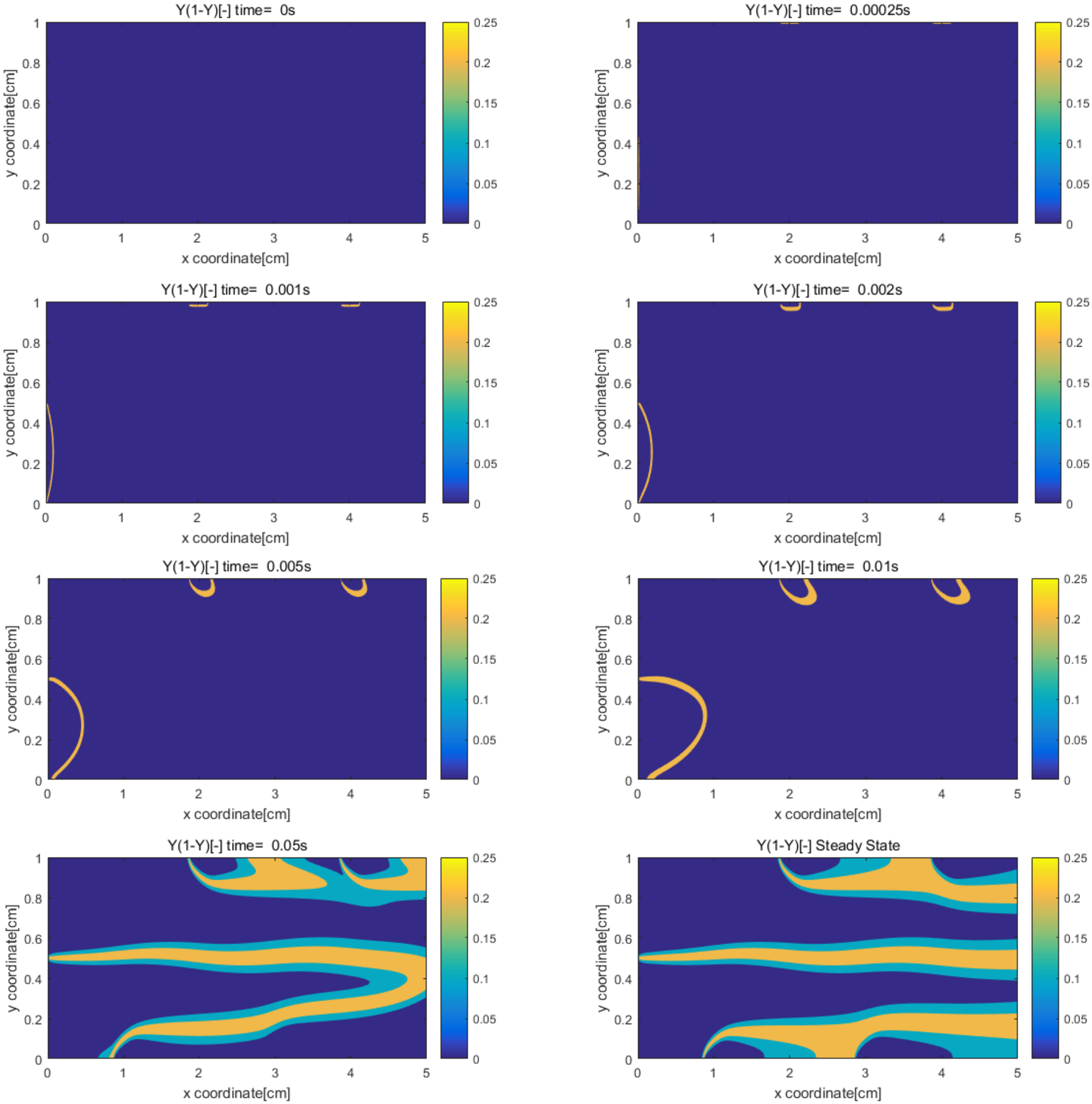

V[m/s] time= 0.05s
y coordinate[cm]
x coordinate[cm]
0 0.2 0.4 0.6 0.8 1
0 1 2 3 4 5
0.5 0 -0.5


V[m/s] Steady State
y coordinate[cm]
x coordinate[cm]
0 0.2 0.4 0.6 0.8 1
0 1 2 3 4 5
0.5 0 -0.5

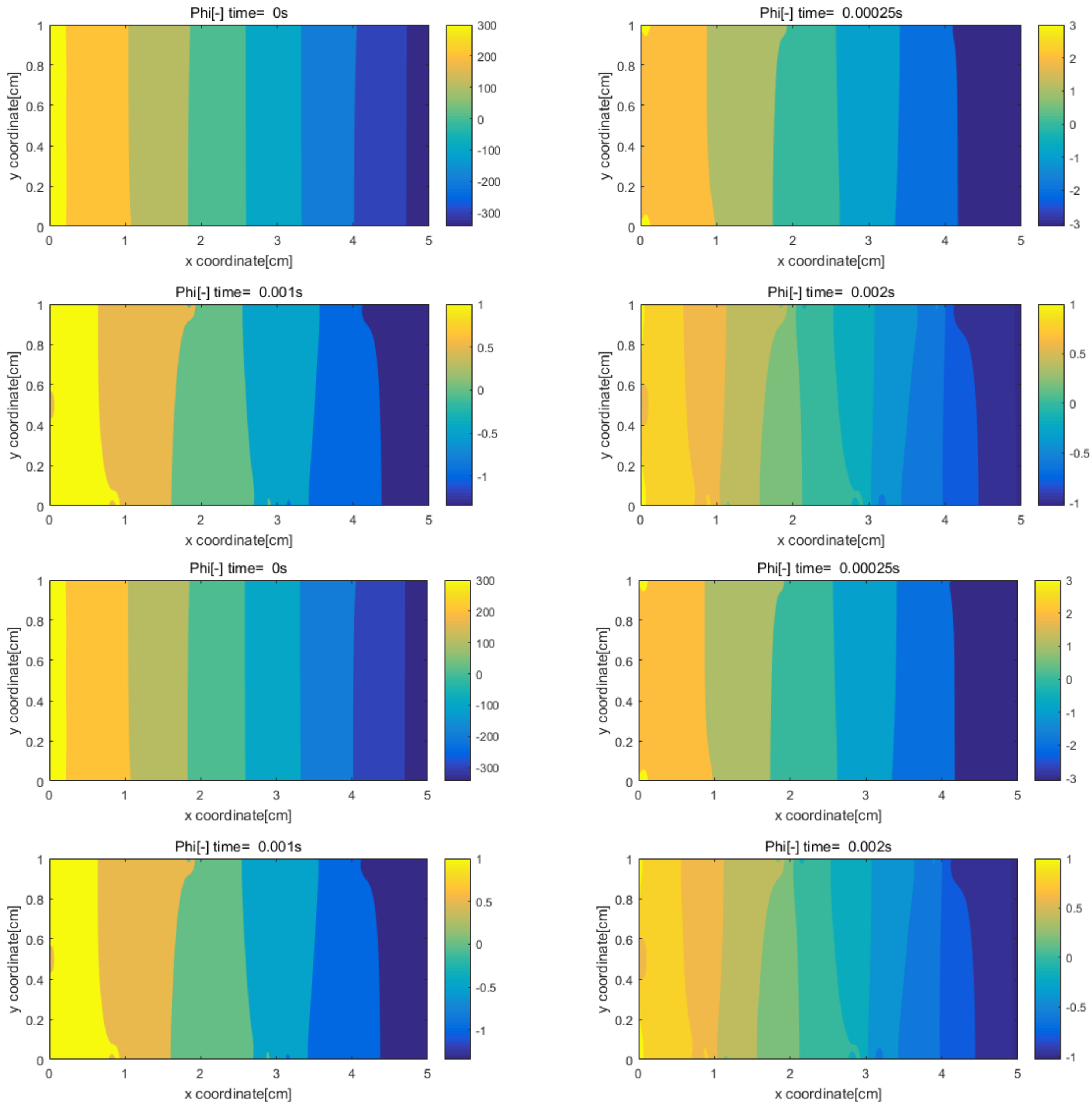
6.5.3 Y Plots


Y[-] time= 0s
Y[-] time= 0.00025s
Y[-] time= 0.001s
Y[-] time= 0.002s
Y[-] time= 0.005s
Y[-] time= 0.01s
Y[-] time= 0.05s
Y[-] Steady State
y coordinate[cm]
x coordinate[cm]

6.5.4 Y(1-Y) Plots



6.5.5 Phi Plots



Appendix： (I just put the code for original geometry here)

```
% set parameters
clear all;
close all;
m=800;
n=160;
k=100000;
C=0.95;
eps=1e-7;
tmax=[0.025 0.1 0.2 0.5 1 5 100] ;
options=2;
nmax=1500;
alp=0.1;
xe=5;
ye=1;
hx=xe/m;
hy=ye/n;
% run the solver
Y=zeros(m+6,n+6,8);
u=zeros(m+1,n+2,8);
v=zeros(m+2,n+1,8);
phi=zeros(m+2,n+2,8);
phi0=zeros(m+2,n+2);
for i=1:7
[Y(:,:,i+1),u(:,:,i+1),v(:,:,i+1),phi(:,:,i+1),phi0,~,~]=solver(m,n,k,C,tmax(i),options,nmax,alp,eps);
end
% restore initial guess Y,U,V,PHI
% build x array for u xu and yu
xu=linspace(0,xe,m+1);
yu=linspace(-0.5*hy,ye+0.5*hy,n+2);
% build initial u array u0 no bc
u0nobc=initialu(xu,yu);
% add bc condition
u(:,:,1)=ubc(u0nobc,yu);

% build initial v array v0 no bc
% build x array for u xu and yu
xv=linspace(-0.5*hx,xe+0.5*hx,m+2);
yv=linspace(0,ye,n+1);
% build initial v array v0 no bc
v0nobc=initialv(xv,yv);
% add bc condition
v(:,:,1)=vbc(v0nobc,xv);

% build initial Y array
xy=linspace(0-2.5*hx,xe+2.5*hx,m+6);
yy=linspace(0-2.5*hy,ye+2.5*hy,n+6);
% build initial Y array v0 no bc
Y0nobc=initialY(xy,yy);
% apply bc to Y
Y(:,:,1)=Ybc(Y0nobc,m,n,xy,yy);

% build initial phi array(m+2,n+2)
phix=xy(3:end-2);
phiy=yy(3:end-2);
phi(:,:,1)=phi0;

% Y(1-Y) in Q
```

```
Q=zeros(m+6,n+6,5);
for i=1:8
Q(:,:,i)=Y(:,:,i).*(1-Y(:,:,i));
end

% plot figure
% plot u
figure(1)
for i=1:4
subplot(2,2,i)
contourf(xu,yu,u(:,:,i)');
axis([0 5 0 1]);
caxis([0 2]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('U[m/s] time=    s')
end
figure(2)
for i=5:8
subplot(2,2,i-4)
contourf(xu,yu,u(:,:,i)');
axis([0 5 0 1]);
caxis([0 2]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('U[m/s] time=    s')
end
% plot v
figure(3)
for i=1:4
subplot(2,2,i)
contourf(xv,yv,v(:,:,i)');
axis([0 5 0 1]);
caxis([-0.5 0.5]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('V[m/s] time=    s')
end
figure(4)
for i=5:8
subplot(2,2,i-4)
contourf(xv,yv,v(:,:,i)');
axis([0 5 0 1]);
caxis([-0.5 0.5]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('V[m/s] time=    s')
end
% plot Y
figure(5)
for i=1:4
subplot(2,2,i)
contourf(xy,yy,Y(:,:,i)');
axis([0 5 0 1]);
```

```
caxis([0 1]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('Y[-] time=    s')
end
figure(6)
for i=5:8
subplot(2,2,i-4)
contourf(xy,yy,Y(:,:,i)');
axis([0 5 0 1]);
caxis([0 1]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('Y[-] time=    s')
end
% Plot Q
figure(7)
for i=1:4
subplot(2,2,i)
contourf(xy,yy,Q(:,:,i)');
axis([0 5 0 1]);
caxis([0 0.25]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('Y(1-Y)[-] time=    s')
end
figure(8)
for i=5:8
subplot(2,2,i-4)
contourf(xy,yy,Q(:,:,i)');
axis([0 5 0 1]);
caxis([0 0.25]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('Y(1-Y)[-] time=    s')
end
% plot phi
figure(9)
for i=1:4
subplot(2,2,i)
contourf(phix,phiy,phi(:,:,i)');
axis([0 5 0 1]);
colorbar
xlabel('x coordinate[cm]')
ylabel('y coordinate[cm]')
title('Phi[-] time=    s')
end
figure(10)
for i=5:8
subplot(2,2,i-4)
contourf(phix,phiy,phi(:,:,i)');
axis([0 5 0 1]);
colorbar
xlabel('x coordinate[cm]')
```

```
ylabel('y coordinate[cm]')
title('Phi[-] time=    s')
end

% R(t)plot
clear all;
close all;
% set parametres
m=400;
n=80;
k=10000;
C=0.95;
tmax=100;
options=2;
nmax=1000;
alp=0.1;
eps=1e-7;
[~,~,~,~,~,~,Rall]=solver(m,n,k,C,tmax,options,nmax,alp,eps);
% plot Rall(R;t)
plot(Rall(2,:),Rall(1,:));
xlabel('time t')
ylabel('R(t)')
title('R(t) as a function of time from the initial condition up to steady state')

% GCI analysis
R=[0.0717623146822650,0.0824682611956138,0.0965618555581060];
% GCI analysis
% initial processing
Fsec=1.25;
r=2;
% calculate order of convergence p
p=log(((R(3)-R(2))/(R(2)-R(1))))/log(r);
% determine estimate for exact solution using Richardson extrapolation
f0=R(1)+(R(1)-R(2))/(r^p-1);
% Grid Convergence Index
e12=abs((R(1)-R(2))/R(1));
GCI21=Fsec*e12/(r^p-1);
e23=abs((R(2)-R(3))/R(2));
GCI23=Fsec*e23/(r^p-1);
% Asymptotic range of convergence
k=GCI21/GCI23*r^p;
% final answer output in command window
per=abs(GCI21*100);
formatSpec = 'The value of R at t=0.03s is %f0 +/- %G%%';
GCI_analysis_result = sprintf(formatSpec,f0,per)

% apply bc to Y input 106x26 output 106x26
function Y=Ybc(Y,m,n,x,y)
% outlet bc
Y(m+4,:)=Y(m+3,:);
Y(m+5,:)=Y(m+2,:);
Y(m+6,:)=Y(m+1,:);
% inlet bc
for i=1:length(y)
if y(i)<0.5
    Y(3,i)=2-Y(4,i);
```

```
        Y(2,i)=2-Y(5,i);
        Y(1,i)=2-Y(6,i);
else
        Y(3,i)=-Y(4,i);
        Y(2,i)=-Y(5,i);
        Y(1,i)=-Y(6,i);
end
end
% wall bc
Y(:,3)=Y(:,4);
Y(:,2)=Y(:,5);
Y(:,1)=Y(:,6);
Y(:,n+4)=Y(:,n+3);
Y(:,n+5)=Y(:,n+2);
Y(:,n+6)=Y(:,n+1);
% slits
for i=1:length(x)
        if x(i)<=1.125&&x(i)>=0.875
                Y(i,3)=-Y(i,4);
                Y(i,2)=-Y(i,5);
                Y(i,1)=-Y(i,6);
        else if x(i)<=3.125&&x(i)>=2.875
                  Y(i,3)=-Y(i,4);
                  Y(i,2)=-Y(i,5);
                  Y(i,1)=-Y(i,6);
                else if x(i)>=1.875&&x(i)<=2.125
                                Y(i,n+6)=2-Y(i,n+1);
                                Y(i,n+5)=2-Y(i,n+2);
                                Y(i,n+4)=2-Y(i,n+3);
                        else if x(i)>=3.875&&x(i)<=4.125
                                Y(i,n+6)=2-Y(i,n+1);
                                Y(i,n+5)=2-Y(i,n+2);
                                Y(i,n+4)=2-Y(i,n+3);
                                end
                        end
                end
        end
end
end

% add bc to v input(m+2,n+1) output(m+2,n+1)
function m=vbc(m,x)
% upper and lower wall bc
m(:,1)=vslitlow(x);
m(:,end)=vslitup(x);
% inlet bc (Drichilet bc)
m(1,:)=-m(2,:);
% outlet bc (Drichilet bc)
m(end,:)=-m(end-1,:);
end

%    add v velocity to lower slits
function m=vslitlow(x)
m=zeros(1,length(x));
for i=1:length(x)
        if x(i)>=0.875&&x(i)<=1.125
```

```
            m(i)=0.5;
        else if x(i)>=2.875&&x(i)<=3.125
            m(i)=0.5;
            end
        end
end
end

%    add v velocity to upper slits
function m=vslitup(x)
m=zeros(1,length(x));
for i=1:length(x)
        if x(i)>=1.875&&x(i)<=2.125
            m(i)=-0.5;
        else if x(i)>=3.875&&x(i)<=4.125
            m(i)=-0.5;
            end
        end
end
end

% add bc to u    input(m+1,n+2) output(m+1,n+2)
function m=ubc(m,y)
% left bc (inlet source A,B)
m(1,:)=uinlet(y);
% right bc(outlet source 0st order in dubug)
m(end,:)=m(end-1,:);
% upper and lower wall bc
m(:,1)=-m(:,2);
m(:,end)=-m(:,end-1);
end

% inlet of u for initial condition
function uy=uinlet(y)
uy=zeros(1,length(y));
for i=1:length(y)
if y(i)<=0.5
        uy(i)=-16*(y(i)-0.25)^2+1;
else
        uy(i)=-16*(y(i)-0.75)^2+1;
end
end
end

function [phi,err]=Vcycle(phi,f,m,h,options,nmax,alpha)
% set initial parameters
p=log2(m/25)+1;
% judge m
if rem(p,1)~=0
        err=2;
        warning('the number of elements is not a correct value')
        return
end
% err is 1 or 0 need to determine later
if rem(p,1)==0
% err=1;
```

```
% err=0;

M=zeros(1,p);
M(1)=m;
% build Mesh space array M
for i=2:p
M(i)=M(i-1)/2;
end
% build N array
N=M/5;
% build h array
H=zeros(1,p);
H(1)=h;
for i=2:p
H(i)=2*H(i-1);
end



% option 1
if options==1
% initial processing
r=zeros(M(1)+2,N(1)+2,length(M));
phi(:,:,1)=phi;
% loop
for j=1:nmax
eps=zeros(M(1)+2,N(1)+2,length(M));
phi(:,:,1)=GaussSeidel(phi(:,:,1),f,H(1),M(1),N(1));
r(:,:,1)=calcResidual(phi(:,:,1),f,H(1),M(1),N(1));
for q=2:p
    rhs(1:M(q)+2,1:N(q)+2,q)=restrict(r(:,:,q-1),M(q),N(q));
    eps(:,:,q)=GaussSeidel(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
    r(1:M(q)+2,1:N(q)+2,q)=calcResidual(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
end
epsc=zeros(M(1)+2,N(1)+2,p);
for q=p-1:-1:2
    epsc(1:M(q)+2,1:N(q)+2,q)=prolong(eps(1:M(q+1)+2,1:N(q+1)+2,q+1),M(q+1),N(q+1));
    eps(:,:,q)=correct(eps(:,:,q),epsc(:,:,q));
    eps(:,:,q)=GaussSeidel(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
end
epsc(:,:,1)=prolong(eps(:,:,2),M(2),N(2));
phi(:,:,1)=correct(phi(:,:,1),epsc(:,:,1));
end
err=0;
end

% option 2
if options==2
% initial processing
r=zeros(M(1)+2,N(1)+2,length(M));
error=zeros(M(1)+2,N(1)+2);
Linf=zeros(nmax,1);
phi(:,:,1)=phi;
% loop
for j=1:nmax
eps=zeros(M(1)+2,N(1)+2,length(M));
phi(:,:,1)=GaussSeidel(phi(:,:,1),f,H(1),M(1),N(1));
```

```
r(:,:,1)=calcResidual(phi(:,:,1),f,H(1),M(1),N(1));
for q=2:p
        rhs(1:M(q)+2,1:N(q)+2,q)=restrict(r(:,:,q-1),M(q),N(q));
        eps(:,:,q)=GaussSeidel(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
        r(1:M(q)+2,1:N(q)+2,q)=calcResidual(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
end
epsc=zeros(M(1)+2,N(1)+2,p);
for q=p-1:-1:2
        epsc(1:M(q)+2,1:N(q)+2,q)=prolong(eps(1:M(q+1)+2,1:N(q+1)+2,q+1),M(q+1),N(q+1));
        eps(:,:,q)=correct(eps(:,:,q),epsc(:,:,q));
        eps(:,:,q)=GaussSeidel(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
end
epsc(:,:,1)=prolong(eps(:,:,2),M(2),N(2));
phi(:,:,1)=correct(phi(:,:,1),epsc(:,:,1));
% calculate error
for jj=2:length(phi(1,:))-1
for i=2:length(phi(:,1))-1
error(i,jj)=f(i,jj)-1/H(1)^2*(phi(i-1,jj)+phi(i+1,jj)+phi(i,jj+1)+phi(i,jj-1)-4*phi(i,jj));
end
end
% calculate infinity norm of error
Linf(j)=max(max(abs(error)));
% absolute convergence determine below max n
if Linf(j)<alpha
err=0;
return
end
end
% absolute convergence determine after max n
if Linf(end)>alpha
        err=1;
        warning ('set iteration times did not satisfying criterium 2');
end
if Linf(end)==alpha
        err=1;
        warning ('set iteration times did not satisfying criterium 2');
end
end


% option 3
if options==3
% initial processing
r=zeros(M(1)+2,N(1)+2,length(M));
error=zeros(M(1)+2,N(1)+2);
Linf=zeros(nmax,1);

% calculate infinity norm of initial guess Linf0
residual0=zeros(M(1)+2,N(1)+2);
for i=2:length(phi(:,1))-1
for j=2:length(phi(1,:))-1
residual0(i,j)=f(i,j)-1/H(1)^2*(phi(i-1,j)+phi(i,j+1)+phi(i+1,j)+phi(i,j-1)-4*phi(i,j));
end
end
Linf0=max(max(abs(residual0)));
```

```
phi(:,:,1)=phi;
% loop
for j=1:nmax
eps=zeros(M(1)+2,N(1)+2,length(M));
phi(:,:,1)=GaussSeidel(phi(:,:,1),f,H(1),M(1),N(1));
r(:,:,1)=calcResidual(phi(:,:,1),f,H(1),M(1),N(1));
for q=2:p
    rhs(1:M(q)+2,1:N(q)+2,q)=restrict(r(:,:,q-1),M(q),N(q));
    eps(:,:,q)=GaussSeidel(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
    r(1:M(q)+2,1:N(q)+2,q)=calcResidual(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
end
epsc=zeros(M(1)+2,N(1)+2,p);
for q=p-1:-1:2
    epsc(1:M(q)+2,1:N(q)+2,q)=prolong(eps(1:M(q+1)+2,1:N(q+1)+2,q+1),M(q+1),N(q+1));
    eps(:,:,q)=correct(eps(:,:,q),epsc(:,:,q));
    eps(:,:,q)=GaussSeidel(eps(:,:,q),rhs(:,:,q),H(q),M(q),N(q));
end
epsc(:,:,1)=prolong(eps(:,:,2),M(2),N(2));
phi(:,:,1)=correct(phi(:,:,1),epsc(:,:,1));
% calculate error
for jj=2:length(phi(1,:))-1
for i=2:length(phi(:,1))-1
error(i,jj)=f(i,jj)-1/H(1)^2*(phi(i-1,jj)+phi(i+1,jj)+phi(i,jj+1)+phi(i,jj-1)-4*phi(i,jj));
end
end
% calculate infinity norm of error
Linf(j)=max(max(abs(error)));
% calculate the ratio
ratio(j)=Linf(j)/Linf0;
% relative convergence in the middle iterations
if ratio(j)<alpha
    err=0;
    return
end
end
% judge the last time iteration
if ratio(end)>alpha
    err=1;
  warning ('set iteration times did not satisfying criterium 3');
end
if ratio(end)==alpha
    err=1;
  warning ('set iteration times did not satisfying criterium 3');
end
end

end
end

function [Ynew]=solveY2D(Y0,u,v,alpha,hx,hy,dt,m,n,xy,yy)
% move u and v to the cell centers
u=reformu(u,m,n);
v=reformv(v,m,n);
% RK start
% Y0=Y0
% Y1
```

```
Y1=zeros(m+6,n+6);
% dYdx (no ghost cell)
dYdx0=dYdx(u,hx,Y0,m,n);
% dYdy
dYdy0=dYdy(v,hy,Y0,m,n);

for j=4:n+3
for i=4:m+3
Y1(i,j)=Y0(i,j)-1*(u(i-3,j-3)*dt*dYdx0(i-3,j-3))-1*(v(i-3,j-3)*dt*dYdy0(i-3,j-3));
end
end
% add ghost cell values
Y1=Ybc(Y1,m,n,xy,yy);

% Y2
Y2=zeros(m+6,n+6);
dYdx1=dYdx(u,hx,Y1,m,n);
dYdy1=dYdy(v,hy,Y1,m,n);

for j=4:n+3
for i=4:m+3
Y2(i,j)=Y1(i,j)+0.75*(u(i-3,j-3)*dt*dYdx0(i-3,j-3)+v(i-3,j-3)*dt*dYdy0(i-3,j-3))-0.25*(u(i-3,j-3)*dt*dYdx1(i-3,j-3)+v(i-3,j-3)*dt*dYd
y1(i-3,j-3));
end
end
% add ghost cell values
Y2=Ybc(Y2,m,n,xy,yy);

% Y3
Y3=zeros(m+6,n+6);
dYdx2=dYdx(u,hx,Y2,m,n);
dYdy2=dYdy(v,hy,Y2,m,n);

for j=4:n+3
for i=4:m+3
Y3(i,j)=Y2(i,j)+1/12*(u(i-3,j-3)*dt*dYdx0(i-3,j-3)+v(i-3,j-3)*dt*dYdy0(i-3,j-3))+1/12*(u(i-3,j-3)*dt*dYdx1(i-3,j-3)+v(i-3,j-3)*dt*dY
dy1(i-3,j-3))-2/3*(u(i-3,j-3)*dt*dYdx2(i-3,j-3)+v(i-3,j-3)*dt*dYdy2(i-3,j-3));
end
end
% add ghost cell values
Y3=Ybc(Y3,m,n,xy,yy);

%HY (HY 106x26) as a source to the parabolic solver
HY=(Y3-Y0)/dt;

% parabolic part
% get rid of bc of Ybc(100X20) and HY(100X20)
Y=Y0(3:end-2,3:end-2);
S=HY(3:end-2,3:end-2);
% set d1 and d2
d1=0.5*alpha*dt/hx^2;
d2=0.5*alpha*dt/hy^2;
% build A1,B1,C1 array when implicit x only
A1=-d1*ones(m,n);
B1=(1+2*d1)*ones(m,n);
C1=-d1*ones(m,n);
```

```
% build D1 array when implicit x only
D1=zeros(m,n);
for j=2:n+1
        for i=2:m+1
                D1(i-1,j-1)=d2*Y(i,j+1)+d2*Y(i,j-1)+(1-2*d2)*Y(i,j)+0.5*dt*S(i,j);
        end
end
% apply left boundary to A1,B1,C1,D1
A1(1,:)=0;
B1(1,:)=1+3*d1;
C1(1,:)=-d1;
for j=2:n+1
        if yy(j+2)<0.5
D1(1,j-1)=d2*Y(2,j+1)+(1-2*d2)*Y(2,j)+d2*Y(2,j-1)+0.5*dt*S(2,j)+2*d1;
        else
D1(1,j-1)=d2*Y(2,j+1)+(1-2*d2)*Y(2,j)+d2*Y(2,j-1)+0.5*dt*S(2,j);
        end
end
% apply right boundary to A1,B1,C1,D1
A1(end,:)=-d1;
B1(end,:)=1+d1;
C1(end,:)=0;
for j=2:n+1
D1(end,j-1)=d2*Y(end-1,j+1)+(1-2*d2)*Y(end-1,j)+d2*Y(end-1,j-1)+0.5*dt*S(end-1,j);
end
% calculate Y^(n+0.5) restore into Y
for j=1:n
        Y(2:end-1,j+1)=GE(A1(:,j),B1(:,j),C1(:,j),D1(:,j));
end
% add ghost cell values
Yhalf=zeros(m+6,n+6);
Yhalf(3:end-2,3:end-2)=Y;
Y=Ybc(Yhalf,m,n,xy,yy);

% step 2 initial
Y=Y(3:end-2,3:end-2);
% build A2,B2,C2 array when implicit x only
A2=-d2*ones(n,m);
B2=(1+2*d2)*ones(n,m);
C2=-d2*ones(n,m);
% build D2 array when implicit x only
D2=zeros(n,m);
for j=2:n+1
        for i=2:m+1
                D2(j-1,i-1)=d1*Y(i+1,j)+d1*Y(i-1,j)+(1-2*d1)*Y(i,j)+0.5*dt*S(i,j);
        end
end
% apply left boundary to A2,B2,C2,D2
A2(1,:)=0;
B2(1,:)=1+d2;
C2(1,:)=-d2;
for i=2:m+1
        if xy(i+2)>=0.875&&xy(i+2)<=1.125
                        B2(1,i-1)=1+3*d2;
                        D2(1,i-1)=d1*Y(i+1,2)+(1-2*d1)*Y(i,2)+d1*Y(i-1,2)+0.5*dt*S(i,2);
        else if xy(i+2)>=2.875&&xy(i+2)<=3.125
```

```
            B2(1,i-1)=1+3*d2;
            D2(1,i-1)=d1*Y(i+1,2)+(1-2*d1)*Y(i,2)+d1*Y(i-1,2)+0.5*dt*S(i,2);
        end
    end
end
% apply right boundary to A2,B2,C2,D2
A2(end,:)=-d2;
B2(end,:)=1+d2;
C2(end,:)=0;
for i=2:m+1
    if xy(i+2)>=1.875&&xy(i+2)<=2.125
                B2(end,i-1)=1+3*d2;
                D2(end,i-1)=d1*Y(i+1,end-1)+(1-2*d1)*Y(i,end-1)+d1*Y(i-1,end-1)+0.5*dt*S(i,end-1)+2*d2;
    else if xy(i+2)>=3.875&&xy(i+2)<=4.125
                B2(end,i-1)=1+3*d2;
                D2(end,i-1)=d1*Y(i+1,end-1)+(1-2*d1)*Y(i,end-1)+d1*Y(i-1,end-1)+0.5*dt*S(i,end-1)+2*d2;
        end
    end
end
% calculate Y^(n+1) restore into Y
for i=1:m
    Y(i+1,2:end-1)=GE(A2(:,i),B2(:,i),C2(:,i),D2(:,i));
end
% add bc values    to Y
Ynew=zeros(m+6,n+6);
Ynew(3:end-2,3:end-2)=Y;
Ynew=Ybc(Ynew,m,n,xy,yy);
end

function [unew,vnew,Huold,Hvold]=solveBurgers2D(u,v,dx,dy,dt,m,n,yu,xv,miu,q,Huold,Hvold)
% for u
duudx=zeros(m+1,n+2);
duvdy=zeros(m+1,n+2);
for j=2:n+1
    for i=2:m
        duudx(i,j)=((u(i+1,j)+u(i,j))^2-(u(i,j)+u(i-1,j))^2)/(4*dx);
        duvdy(i,j)=((u(i,j)+u(i,j+1))*(v(i+1,j)+v(i,j))-(u(i,j-1)+u(i,j))*(v(i,j-1)+v(i+1,j-1)))/(4*dy);
    end
end
% add boundary
duudx=ubc(duudx,yu);
duvdy=ubc(duvdy,yu);
% Hu
Hu=zeros(m+1,n+2);
for j=1:n+2
    for i=1:m+1
      Hu(i,j)=-duudx(i,j)-duvdy(i,j);
    end
end

% for v
dvvdy=zeros(m+2,n+1);
duvdx=zeros(m+2,n+1);
for j=2:n
    for i=2:m+1
        dvvdy(i,j)=((v(i,j+1)+v(i,j))^2-(v(i,j)+v(i,j-1))^2)/(4*dy);
```

```
        duvdx(i,j)=((u(i,j)+u(i,j+1))*(v(i+1,j)+v(i,j))-(u(i-1,j)+u(i-1,j+1))*(v(i-1,j)+v(i,j)))/(4*dx);
    end
end
% add boundary
dvvdy=vbc(dvvdy,xv);
duvdx=vbc(duvdx,xv);
% Hv
Hv=zeros(m+2,n+1);
for j=1:n+1
    for i=1:m+2
      Hv(i,j)=-dvvdy(i,j)-duvdx(i,j);
    end
end

% Huold and Hvold
if q==1
Huold=Hu;
Hvold=Hv;
end
% Source array Su and Sv
Su=1.5*Hu-0.5*Huold;
Sv=1.5*Hv-0.5*Hvold;

% hyperbolic part
% set d1 and d2
d1=0.5*miu*dt/dx^2;
d2=0.5*miu*dt/dy^2;

% ADI for u
% u step1
% build Au1,Bu1,Cu1 array when implicit x only
Au1=-d1*ones(m-1,n);
Bu1=(1+2*d1)*ones(m-1,n);
Cu1=-d1*ones(m-1,n);
% build Du1 array when implicit x only
Du1=zeros(m-1,n);
for j=2:n+1
    for i=2:m
        Du1(i-1,j-1)=d2*u(i,j+1)+d2*u(i,j-1)+(1-2*d2)*u(i,j)+0.5*dt*Su(i,j);
    end
end
% apply left boundary to Au1,Bu1,Cu1,Du1
Au1(1,:)=0;
Bu1(1,:)=1+2*d1;
Cu1(1,:)=-d1;
for j=2:n+1
Du1(1,j-1)=d2*u(2,j+1)+(1-2*d2)*u(2,j)+d2*u(2,j-1)+0.5*dt*Su(2,j)+d1*u(1,j);
end
% apply right boundary to Au1,Bu1,Cu1,Du1
Au1(end,:)=-d1;
Bu1(end,:)=1+d1;
Cu1(end,:)=0;
for j=2:n+1
Du1(end,j-1)=d2*u(end-1,j+1)+(1-2*d2)*u(end-1,j)+d2*u(end-1,j-1)+0.5*dt*Su(end-1,j);
end
% calculate u^(n+0.5) restore into u
```

```
for j=1:n
        u(2:end-1,j+1)=GE(Au1(:,j),Bu1(:,j),Cu1(:,j),Du1(:,j));
end
% add bc to u^(n+0.5)
us=ubc(u,yu);

% u step 2
% build Au2,Bu2,Cu2 array when implicit x only
Au2=-d2*ones(n,m-1);
Bu2=(1+2*d2)*ones(n,m-1);
Cu2=-d2*ones(n,m-1);
% build Du2 array when implicit x only
Du2=zeros(n,m-1);
for j=2:n+1
        for i=2:m
                Du2(j-1,i-1)=d1*us(i+1,j)+d1*us(i-1,j)+(1-2*d1)*us(i,j)+0.5*dt*Su(i,j);
        end
end
% apply left boundary to Au2,Bu2,Cu2,Du2
Au2(1,:)=0;
Bu2(1,:)=1+3*d2;
Cu2(1,:)=-d2;

% apply right boundary to Au2,Bu2,Cu2,Du2
Au2(end,:)=-d2;
Bu2(end,:)=1+3*d2;
Cu2(end,:)=0;

% calculate u^(n+1) restore into u
unew=zeros(m+1,n+2);
for i=1:m-1
        unew(i+1,2:end-1)=GE(Au2(:,i),Bu2(:,i),Cu2(:,i),Du2(:,i));
end
% add bc to unew
unew=ubc(unew,yu);

% ADI for v(m+2,n+1)
% v step1
% build Av1,Bv1,Cv1 array when implicit x only
Av1=-d1*ones(m,n-1);
Bv1=(1+2*d1)*ones(m,n-1);
Cv1=-d1*ones(m,n-1);
% build Dv1 array when implicit x only
Dv1=zeros(m,n-1);
for j=2:n
        for i=2:m+1
                Dv1(i-1,j-1)=d2*v(i,j+1)+d2*v(i,j-1)+(1-2*d2)*v(i,j)+0.5*dt*Sv(i,j);
        end
end
% apply left boundary to Av1,Bv1,Cv1,Dv1
Av1(1,:)=0;
Bv1(1,:)=1+3*d1;
Cv1(1,:)=-d1;

% apply right boundary to Av1,Bv1,Cv1,Dv1
Av1(end,:)=-d1;
```

```
Bv1(end,:)=1+3*d1;
Cv1(end,:)=0;

% calculate v^(n+0.5) restore into vs
for j=1:n-1
        v(2:end-1,j+1)=GE(Av1(:,j),Bv1(:,j),Cv1(:,j),Dv1(:,j));
end
% add bc to v^(n+0.5) vs
vs=vbc(v,xv);

% v step2
% build Av2,Bv2,Cv2 array when implicit x only
Av2=-d2*ones(n-1,m);
Bv2=(1+2*d2)*ones(n-1,m);
Cv2=-d2*ones(n-1,m);
% build Dv2 array when implicit x only
Dv2=zeros(n-1,m);
for j=2:n
        for i=2:m+1
                Dv2(j-1,i-1)=d1*vs(i+1,j)+d1*vs(i-1,j)+(1-2*d1)*vs(i,j)+0.5*dt*Sv(i,j);
        end
end

% apply left boundary to Av2,Bv2,Cv2,Dv2
Av2(1,:)=0;
Bv2(1,:)=1+2*d2;
Cv2(1,:)=-d2;
for i=2:m+1
        if xv(i)>=0.875&&xv(i)<=1.125
                        Dv2(1,i-1)=d1*vs(i+1,2)+(1-2*d1)*vs(i,2)+d1*vs(i-1,2)+0.5*dt*Sv(i,2)+0.5*d2;
        else if xv(i)>=2.875&&xv(i)<=3.125
                        Dv2(1,i-1)=d1*vs(i+1,2)+(1-2*d1)*vs(i,2)+d1*vs(i-1,2)+0.5*dt*Sv(i,2)+0.5*d2;
                end
        end
end
% apply right boundary to A2,B2,C2,D2
Av2(end,:)=-d2;
Bv2(end,:)=1+2*d2;
Cv2(end,:)=0;
for i=2:m+1
        if xv(i)>=1.875&&xv(i)<=2.125
                                Dv2(end,i-1)=d1*vs(i+1,end-1)+(1-2*d1)*vs(i,end-1)+d1*vs(i-1,end-1)+0.5*dt*Sv(i,end-1)-0.5*d2;
        else if xv(i)>=3.875&&xv(i)<=4.125
                                Dv2(end,i-1)=d1*vs(i+1,end-1)+(1-2*d1)*vs(i,end-1)+d1*vs(i-1,end-1)+0.5*dt*Sv(i,end-1)-0.5*d2;
                end
        end
end

% calculate v^(n+1) restore into vs
for j=1:m
        vs(j+1,2:end-1)=GE(Av2(:,j),Bv2(:,j),Cv2(:,j),Dv2(:,j));
end
% add bc to v^(n+1) restore in vnew
vnew=vbc(vs,xv);
end
```

```
function [Y,u,v,phi,phi0,Rout,Rall]=solver(m,n,k,C,maxtime,options,nmax,alp,eps)
% set parameters
xe=5;
ye=1;
hx=xe/m;
hy=ye/n;
miu=2e-2;
alpha=1e-3;
% build x array for u xu and yu
xu=linspace(0,xe,m+1);
yu=linspace(-0.5*hy,ye+0.5*hy,n+2);
% build initial u array u0 no bc
u0nobc=initialu(xu,yu);
% add bc condition
u=ubc(u0nobc,yu);

% build initial v array v0 no bc
% build x array for u xu and yu
xv=linspace(-0.5*hx,xe+0.5*hx,m+2);
yv=linspace(0,ye,n+1);
% build initial v array v0 no bc
v0nobc=initialv(xv,yv);
% add bc condition
v=vbc(v0nobc,xv);

% build initial Y array
xy=linspace(0-2.5*hx,xe+2.5*hx,m+6);
yy=linspace(0-2.5*hy,ye+2.5*hy,n+6);
% build initial Y array v0 no bc
Y0nobc=initialY(xy,yy);
% apply bc to Y
Y=Ybc(Y0nobc,m,n,xy,yy);

% calculate R
R0=R(Y,m+6,n+6,hx,hy);
% initial time
t=0;
% initial Huold and Hvold
Huold=0;
Hvold=0;
Rall(1,1)=R0;
% set initial guess phi
phi=zeros(m+2,n+2);
% time loop start
for q=1:k
% calculate dt
dtu=min(hx,hy)/(2*max(max(abs(u)))+max(max(abs(v))));
dtv=min(hx,hy)/(2*max(max(abs(v)))+max(max(abs(u))));
% dty= inifinity
dt=C*min(dtu,dtv);
Rall(2,q)=t;
% maxtime determination
if t<maxtime&&t+dt>maxtime
    dt=maxtime-t;
    Rall(2,q)=maxtime;
    t=t+dt
```

```
        % RK3/WENO5 + CN/ADI for Ynew
        [Y]=solveY2D(Y,u,v,alpha,hx,hy,dt,m,n,xy,yy);
        % Rnew
        Rout=R(Y,m+6,n+6,hx,hy);
        Rall(1,end)=Rout;
        break;
end

% steady state R determination
if q>20&&abs((Rall(1,q)-Rall(1,q-20))/(Rall(2,q)-Rall(2,q-20)))<eps
        % Rnew
        Rout=R(Y,m+6,n+6,hx,hy);
        Rall(1,end)=Rout;
        break;
end

% accumulate time t
t=t+dt

% RK3/WENO5 + CN/ADI for Ynew
[Y]=solveY2D(Y,u,v,alpha,hx,hy,dt,m,n,xy,yy);
% Rnew
Rnew=R(Y,m+6,n+6,hx,hy);
Rall(1,q+1)=Rnew;
% Adams-Bashforth/Crank-Nicholson for unew and vnew
[u,v,Huold,Hvold]=solveBurgers2D(u,v,hx,hy,dt,m,n,yu,xv,miu,q,Huold,Hvold);
% outlet correction of u (convective)
[u]=correction(u,v,hx,hy,m,n);
% rhs of possion equation as a source f(m+2,n+2)
[f]=possionrhs(u,v,hx,hy,dt,m,n);
% solve possion equation with f as a source phi(102x22)
[phi,~]=Vcycle(phi,f,m,hx,options,nmax,alp);
if q==1
        phi0=phi;
end
% project/correct velocities using Lagrange multiplier
[u,v]=projection(u,v,phi,dt,hx,hy,yu,xv,m,n);
end
end

% calculate the coefficient in WENO-5
function phi=weno(a,b,c,d)
e=1e-6;
IS0=13*(a-b)^2+3*(a-3*b)^2;
IS1=13*(b-c)^2+3*(b+c)^2;
IS2=13*(c-d)^2+3*(3*c-d)^2;
a0=1/(e+IS0)^2;
a1=6/(e+IS1)^2;
a2=3/(e+IS2)^2;
w0=a0/(a0+a1+a2);
w2=a2/(a0+a1+a2);
phi=1/3*w0*(a-2*b+c)+1/6*(w2-0.5)*(b-2*c+d);
end

function rcoarse=restrict(rfine,mcoarse,ncoarse)
% restrict the residual to coarse mesh
```

```
rcoarse=zeros(mcoarse+2,ncoarse+2);
for j=2:ncoarse+1
for i=2:mcoarse+1
    rcoarse(i,j)=0.25*(rfine(2*i-2,2*j-2)+rfine(2*i-2,2*j-1)+rfine(2*i-1,2*j-2)+rfine(2*i-1,2*j-1));
end
end
% add ghost cell values
rcoarse(1,:)=rcoarse(2,:);
rcoarse(end,:)=rcoarse(end-1,:);
rcoarse(:,1)=rcoarse(:,2);
rcoarse(:,end)=rcoarse(:,end-1);
end

% move v to the cell centers
function vnew=reformv(v,m,n)
vnew=zeros(m,n);
for j=1:n
    for i=1:m
        vnew(i,j)=0.5*(v(i+1,j)+v(i+1,j+1));
    end
end
end

% move u to the cell centers
function unew=reformu(u,m,n)
unew=zeros(m,n);
for j=1:n
    for i=1:m
        unew(i,j)=0.5*(u(i,j+1)+u(i+1,j+1));
    end
end
end

% calculate R(t)
function M=R(Y,m,n,dx,dy)
r=0;
for   j=4:n-3
    for i=4:m-3
        r=r+Y(i,j)*(1-Y(i,j))*dx*dy;
    end
end
M=r/5;

function rfine=prolong(rcoarse,mcoarse,ncoarse)
% prolong the residual on coarse mesh
rfine=zeros(2*mcoarse+2,2*ncoarse+2);
for j=2:ncoarse+1
for i=2:mcoarse+1
    rfine(2*i-1,2*j-1)=rcoarse(i,j);
    rfine(2*i-1,2*j-2)=rcoarse(i,j);
    rfine(2*i-2,2*j-1)=rcoarse(i,j);
    rfine(2*i-2,2*j-2)=rcoarse(i,j);
end
end
% add ghost cell values
rfine(1,:)=rfine(2,:);
```

```
rfine(end,:)=rfine(end-1,:);
rfine(:,1)=rfine(:,2);
rfine(:,end)=rfine(:,end-1);
end

% project/correct velocities using Lagrange multiplier
function [unew,vnew]=projection(u,v,phi,dt,dx,dy,yu,xv,m,n)
unew=zeros(m+1,n+2);
vnew=zeros(m+2,n+1);
for j=2:n+1
    for i=2:m
unew(i,j)=u(i,j)-dt/dx*(phi(i+1,j)-phi(i,j));
    end
end
for j=2:n
    for i=2:m+1
   vnew(i,j)=v(i,j)-dt/dy*(phi(i,j+1)-phi(i,j));
    end
end
% apply boundary
unew=ubc(unew,yu);
vnew=vbc(vnew,xv);
end

%calculate the rhs for possion equation f(m+2,n+2)
function [f]=possionrhs(u,v,dx,dy,dt,m,n)
f=zeros(m+2,n+2);
for j=2:n+1
    for i=2:m+1
   f(i,j)=1/dt*((u(i,j)-u(i-1,j))/dx+(v(i,j)-v(i,j-1))/dy);
    end
end
end

% build 2D initial Y array
function Y=initialY(x,y)
Y=zeros(length(x),length(y));
for j=1:length(y)
    for i=1:length(x)
%           Y(i,j)=sin(0.7*x(i))+cos(1.2*y(j));
              Y(i,j)=0;
    end
end
end

% build 2D initial v array
function v=initialv(x,y)
v=zeros(length(x),length(y));
for j=1:length(y)
    for i=1:length(x)
%           v(i,j)=cos(1.4*y(j))+cos(1.3*x(i));
        v(i,j)=0;
    end
end
end
```

```
% build 2D initial u array
function u=initialu(x,y)
u=zeros(length(x),length(y));
for j=1:length(y)
    for i=1:length(x)
%           u(i,j)=sin(1.2*x(i)+0.1)+sin(1.4*y(j));
        u(i,j)=0;
    end
end
end

function d = GE(a,b,c,d)
% This function is to solve the tri-diagonal matrix using Gauss
% eliminatioon method.
% The inputs are 4 arrays, a,b,c,d which can stand for the whole
% tri-diagonal matrix.
% b is the vector of diagonal entries, a is the array of numbers below
% diagonal entries and c is the numbers above the diagonal entries.
% The output is the vector of the correct solution of this given equation
% restored in array d.
% However,this method will destory the original data vector b and d.
format long;
% In order to have more precision.
n=length(b);
% This is just the exactly derivation of Gauss elimination.
% Step 1: elimination
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
% Step 2: back substitution
d(n)=d(n)/b(n);
for j=n-1:-1:1
    d(j)=(d(j)-c(j)*d(j+1))/b(j);
end
end

function y=GaussSeidel(phi,rhs,h,m,n)
% GS method
for j=2:m+1
    for i=2:n+1
      phi(j,i)=0.25*(phi(j-1,i)+phi(j+1,i)+phi(j,i+1)+phi(j,i-1))-0.25*h^2*rhs(j,i);
    end
end
% apply ghost cell values
phi(1,:)=phi(2,:);
phi(m+2,:)=phi(m+1,:);
phi(:,1)=phi(:,2);
phi(:,n+2)=phi(:,n+1);
y=phi;

% calculate dY/dy
function w=dYdy(v,hy,Y,m,n)
A=zeros(m,n);
B=zeros(m,n);
C=zeros(m,n);
```

```
D=zeros(m,n);
w=zeros(m,n);

for j=4:n+3
for i=4:m+3
% v>0
if v(i-3,j-3)>0
	A(i-3,j-3)=1/hy*((Y(i,j-1)-Y(i,j-2))-(Y(i,j-2)-Y(i,j-3)));
	B(i-3,j-3)=1/hy*((Y(i,j)-Y(i,j-1))-(Y(i,j-1)-Y(i,j-2)));
	C(i-3,j-3)=1/hy*((Y(i,j+1)-Y(i,j))-(Y(i,j)-Y(i,j-1)));
	D(i-3,j-3)=1/hy*((Y(i,j+2)-Y(i,j+1))-(Y(i,j+1)-Y(i,j)));

w(i-3,j-3)=1/(12*hy)*(-(Y(i,j-1)-Y(i,j-2))+7*(Y(i,j)-Y(i,j-1))+7*(Y(i,j+1)-Y(i,j))-(Y(i,j+2)-Y(i,j+1)))-weno(A(i-3,j-3),B(i-3,j-3),C(i-3,j-3),D
(i-3,j-3));
end
% v<0
if v(i-3,j-3)<0
	A(i-3,j-3)=1/hy*((Y(i,j+3)-Y(i,j+2))-(Y(i,j+2)-Y(i,j+1)));
	B(i-3,j-3)=1/hy*((Y(i,j+2)-Y(i,j+1))-(Y(i,j+1)-Y(i,j)));
	C(i-3,j-3)=1/hy*((Y(i,j+1)-Y(i,j))-(Y(i,j)-Y(i,j-1)));
	D(i-3,j-3)=1/hy*((Y(i,j)-Y(i,j-1))-(Y(i,j-1)-Y(i,j-2)));

w(i-3,j-3)=1/(12*hy)*(-(Y(i,j-1)-Y(i,j-2))+7*(Y(i,j)-Y(i,j-1))+7*(Y(i,j+1)-Y(i,j))-(Y(i,j+2)-Y(i,j+1)))+weno(A(i-3,j-3),B(i-3,j-3),C(i-3,j-3),D
(i-3,j-3));
end
end
end
end

% calculate dY/dx
function w=dYdx(u,hx,Y,m,n)

A=zeros(m,n);
B=zeros(m,n);
C=zeros(m,n);
D=zeros(m,n);
w=zeros(m,n);

for j=4:n+3
for i=4:m+3
% u>0
if u(i-3,j-3)>0
	A(i-3,j-3)=1/hx*((Y(i-1,j)-Y(i-2,j))-(Y(i-2,j)-Y(i-3,j)));
	B(i-3,j-3)=1/hx*((Y(i,j)-Y(i-1,j))-(Y(i-1,j)-Y(i-2,j)));
	C(i-3,j-3)=1/hx*((Y(i+1,j)-Y(i,j))-(Y(i,j)-Y(i-1,j)));
	D(i-3,j-3)=1/hx*((Y(i+2,j)-Y(i+1,j))-(Y(i+1,j)-Y(i,j)));

w(i-3,j-3)=1/(12*hx)*(-(Y(i-1,j)-Y(i-2,j))+7*(Y(i,j)-Y(i-1,j))+7*(Y(i+1,j)-Y(i,j))-(Y(i+2,j)-Y(i+1,j)))-weno(A(i-3,j-3),B(i-3,j-3),C(i-3,j-3),D
(i-3,j-3));
end
% u<0
if u(i-3,j-3)<0
	A(i-3,j-3)=1/hx*((Y(i+3,j)-Y(i+2,j))-(Y(i+2,j)-Y(i+1,j)));
	B(i-3,j-3)=1/hx*((Y(i+2,j)-Y(i+1,j))-(Y(i+1,j)-Y(i,j)));
	C(i-3,j-3)=1/hx*((Y(i+1,j)-Y(i,j))-(Y(i,j)-Y(i-1,j)));
	D(i-3,j-3)=1/hx*((Y(i,j)-Y(i-1,j))-(Y(i-1,j)-Y(i-2,j)));
```

```
w(i-3,j-3)=1/(12*hx)*(-(Y(i-1,j)-Y(i-2,j))+7*(Y(i,j)-Y(i-1,j))+7*(Y(i+1,j)-Y(i,j))-(Y(i+2,j)-Y(i+1,j)))+weno(A(i-3,j-3),B(i-3,j-3),C(i-3,j-3),D
(i-3,j-3));
end
end
end
end

% Outlet Correction
function [u]=correction(u,v,dx,dy,m,n)
ly=1;
A=0;
B=0;
C=0;
D=0;
for j=1:n
A=A+(max(0,u(1,j+1))+max(0,-u(m+1,j+1)))*dy;
C=C+(max(0,-u(1,j+1))+max(0,u(m+1,j+1)))*dy;
end
for i=1:m
B=B+(max(0,v(i+1,1))+max(0,-v(i+1,n+1)))*dx;
D=D+(max(0,-v(i+1,1))+max(0,v(i+1,n+1)))*dx;
end
qin=A+B;
qout=C+D;
qcorr=qin-qout;
ucorr=qcorr/ly;
u(m+1,:)=u(m+1,:)+ucorr;
end

% add the residual to phi
function y=correct(eps,epsc)
y=eps+epsc;
end

function y=calcResidual(phi,rhs,h,m,n)
% calculate residual
y=zeros(m+2,n+2);
for j=2:n+1
for i=2:m+1
    y(i,j)=rhs(i,j)-1/h^2*(phi(i+1,j)+phi(i-1,j)+phi(i,j-1)+phi(i,j+1)-4*phi(i,j));
end
end
% set ghost cell values
y(1,:)=y(2,:);
y(end,:)=y(end-1,:);
y(:,1)=y(:,2);
y(:,end)=y(:,end-1);
end
% this function is to add ghost cell values to T
function [T]=addghost(T)
T(1,:)=T(2,:);
T(end,:)=T(end-1,:);
T(:,1)=T(:,2);
T(:,end)=T(:,end-1);
end
```