COMPUTING ASSIGNMENT

Direct Numerical Simulation of a 2D heat exchanger based on a circular cylinder

AEM-ADV19 Computational Fluid Dynamics

Zijun Fang (CID: 01811420)

Songrui Li (CID: 01742372)

Table of Contents

Question 1

Run a first simulation with $Re = 200$;
Time Step $nt = 10000$;
Second-order Adams-Bashforth Scheme: `itemp=1`;
Mesh number in each axis is $n_x * n_y = 129 * 129$;
The below figures are for the four visualisations of the vorticity field under $CFL = 0.25$;

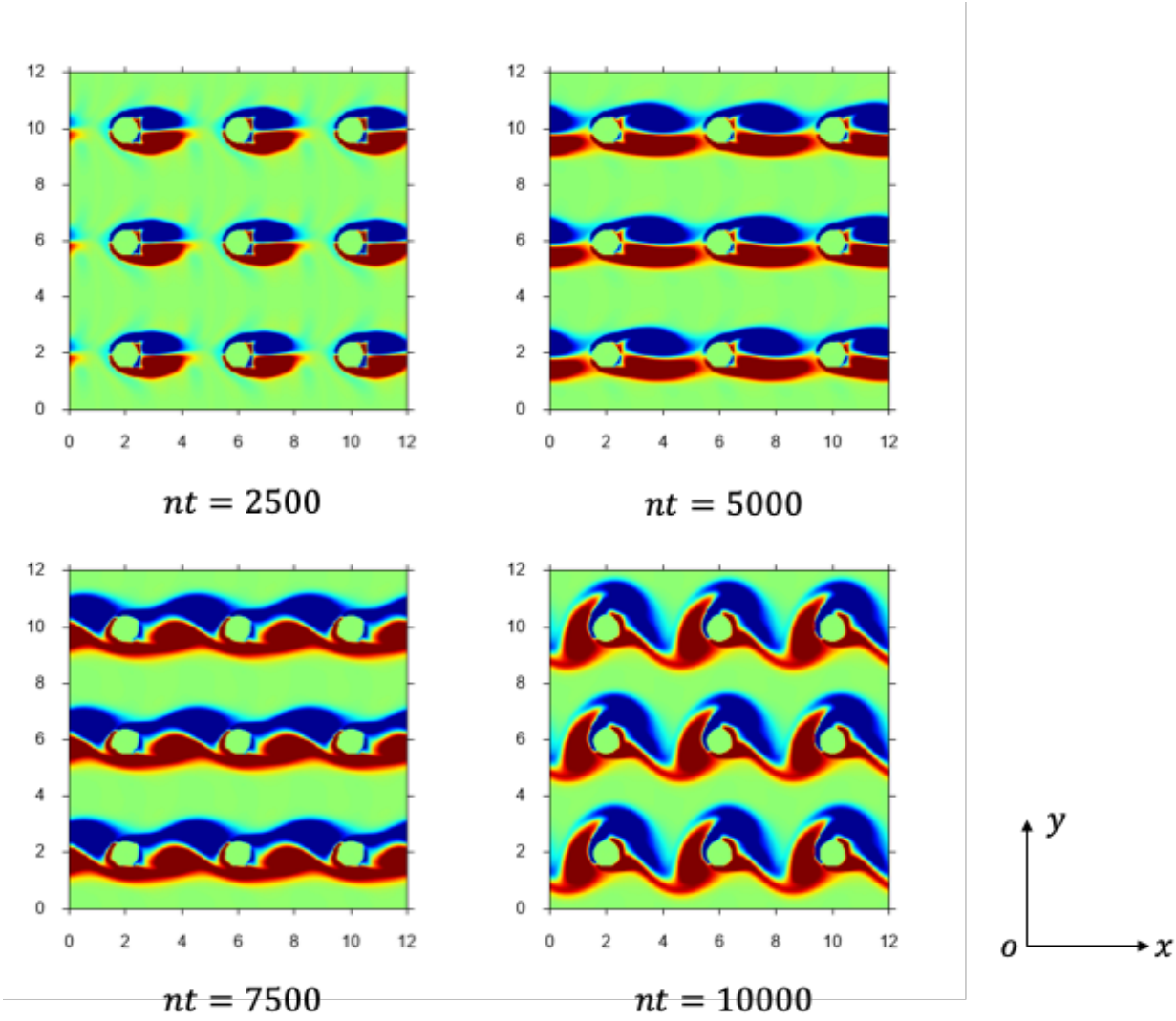


Figure 1.1-1.4: *Displaying the visualisations of the vorticity field in different time steps-in adams subroutine;*

Description:

While for the results of plotting vorticity field in different time steps, which the vortices flows become more non-symmetric, where the clockwise vortices (blue zone) stay at the backward through the structure; also, when the time step increase, flow become more continuous to the next part, the vorticity fields are increasing to connect the next cylindrical cylinders. Not only the phases of different direction vortices have been changed, also the shock of vorticity has been increased through the time step, become stronger when the time passed.

Question 2

Run a first simulation with $Re = 200$;
Time Step $nt = 10000$;
Second-order Adams-Bashforth Scheme: `itemp=1`;
Mesh number in each axis is $n_x * n_y = 129 * 129$;
The below figures are for the four visualisations of the vorticity field under $CFL = 0.75$;

Description:

According to the CFL number has been increased to 0.75, in order to reduce the cost of the simulation, after running the script in 0.75 there is no vorticity picture come out. While calculating the maximum courant number, which has specific this case with a magnitude of 0.5 in steady situation. Due to the value of maximum courant number to solve the discretised equation, if the setting value is larger than the limited constant, there the plot will not come out; for some reasons, the computing time step covers the physical timing generating diagrams, so the data will come into NAN, and the plots are in blank.

Question 3

Run a first simulation with $Re = 200$;
Time Step $nt = 10000$;
Third-order **Runge-Kutta** Scheme: `itemp=2`;
Mesh number in each axis is $n_x * n_y = 129 * 129$;
The below figures are for the four visualisations of the vorticity field under $CFL = 0.75$;

* Subroutine **rkutta**

```
!############################################################
!
subroutine rkutta(rho,rou,rov,roe,fro,gro,fru,gru,frv,grv,&
     fre,gre,ftp,gtp,nx,ny,ns,dlt,coef,scp,k)
!
!############################################################

  implicit none
!
  real(8),dimension(nx,ny) :: rho,rou,rov,roe,fro,gro,fru,gru,frv
  real(8),dimension(nx,ny) :: grv,fre,gre,scp,ftp,gtp
  real(8),dimension(2,ns) :: coef
  real(8) :: dlt
  integer :: i,j,nx,ny,ns,k
!
!coefficient for RK sub-time steps
        coef(1,1)=8./15.*dlt
        coef(1,2)=5./12.*dlt
        coef(1,3)=3./4.*dlt
        coef(2,1)=0*dlt
        coef(2,2)=-17./60.*dlt
        coef(2,3)=-5./12.*dlt

  do j=1,ny
     do i=1,nx

        rho(i,j)=rho(i,j)+coef(1,k)*fro(i,j)+coef(2,k)*gro(i,j)
        gro(i,j)=fro(i,j)
        rou(i,j)=rou(i,j)+coef(1,k)*fru(i,j)+coef(2,k)*gru(i,j)
        gru(i,j)=fru(i,j)
        rov(i,j)=rov(i,j)+coef(1,k)*frv(i,j)+coef(2,k)*grv(i,j)
        grv(i,j)=frv(i,j)
        roe(i,j)=roe(i,j)+coef(1,k)*fre(i,j)+coef(2,k)*gre(i,j)
        gre(i,j)=fre(i,j)
        scp(i,j)=scp(i,j)+coef(1,k)*ftp(i,j)+coef(2,k)*gtp(i,j)
        gtp(i,j)=ftp(i,j)

     enddo
  enddo

  return
end subroutine rkutta
!############################################################
```

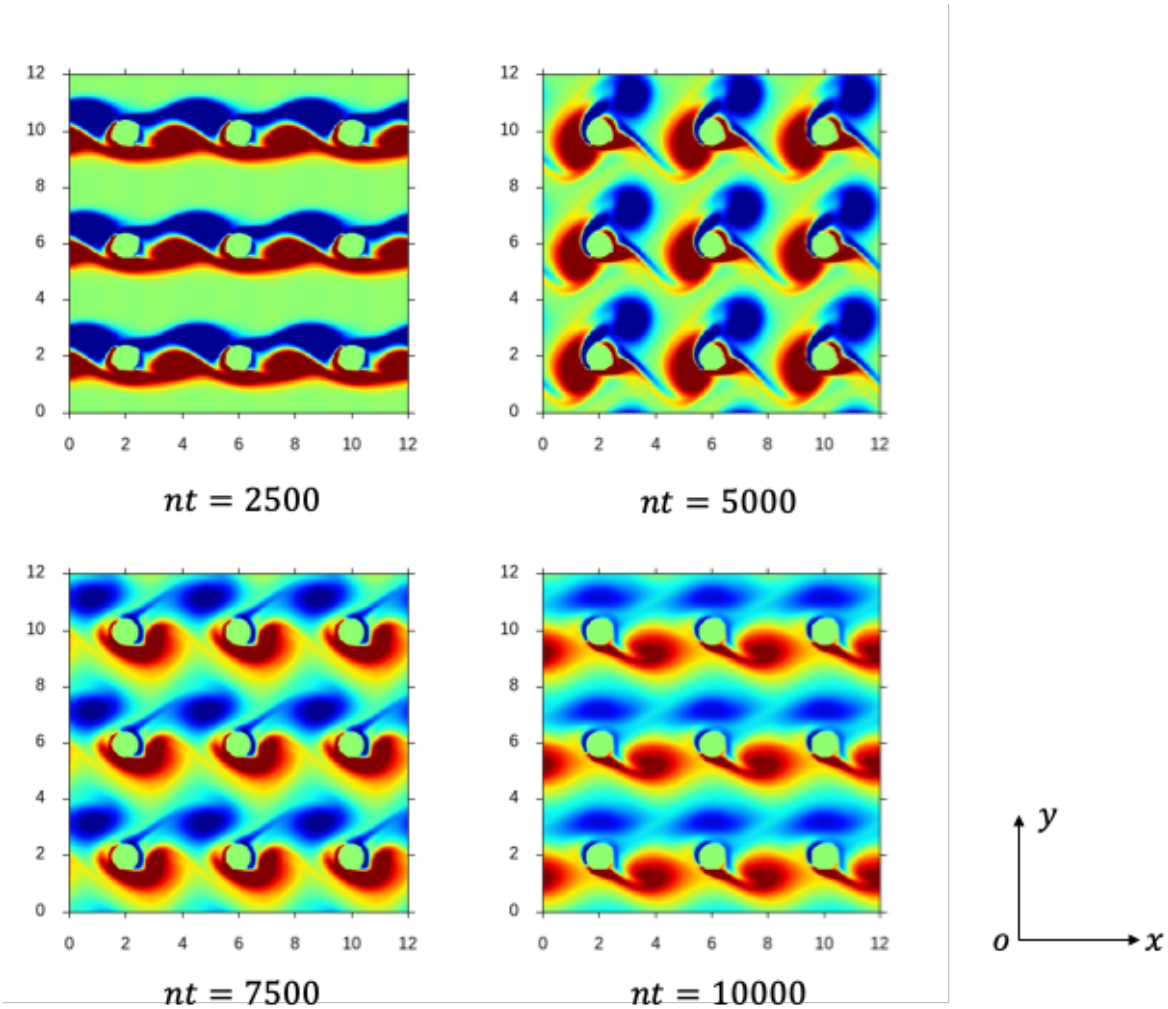


Figure 3.1-3.4: *Displaying the visualisations of the vorticity field in different time steps-in rkutta subroutine;*

Description:

By using the Runge-Kutta scheme, the figures show the continuous flow condition in a higher order, while the second-order function could not reach. And the previous diagram in nt=2500 and 5000 are all presenting the last two diagrams in previous condition outcome, therefore, this means the higher order differential equation in the same time step will present more flow shoots content than the lower one, actually, these four figures present more flows after the 5000 time steps and the vorticity from upper and below are in more backwards.

Question 4

The fourth-order schemes for the first and second derivatives:

$$f'(x) = \frac{-f(x+2\Delta x) + 8f(x+\Delta x) - 8f(x-\Delta x) + f(x-2\Delta x)}{12\Delta x}$$

$$f''(x) = \frac{-f(x+2\Delta x) + 16f(x+\Delta x) - 30f(x) + 16f(x-\Delta x) - f(x-2\Delta x)}{12\Delta x^2}$$

These formulation present in Fortran language subroutines are:

* Subroutine **derix4**

```
!##########################################
!
subroutine derix4(phi,nx,ny,dfi,xlx)
!
!Fourth-order first derivative in the x direction
!##########################################

  implicit none

  real(8),dimension(nx,ny) :: phi,dfi
  real(8) :: dlx,xlx,udx
  integer :: i,j,nx,ny

  dlx=xlx/nx
  udx=1./(12*dlx)
  do j=1,ny
     dfi(1,j)=udx*(-phi(3,j)+8*phi(2,j)-8*phi(nx,j)+phi(nx-1,j))
     dfi(2,j)=udx*(-phi(4,j)+8*phi(3,j)-8*phi(1,j)+phi(nx,j))
     do i=3,nx-2
        dfi(i,j)=udx*(-phi(i+2,j)+8*phi(i+1,j)-8*phi(i-1,j)+phi(i-2,j))
     enddo
     dfi(nx-1,j)=udx*(-phi(1,j)+8*phi(nx,j)-8*phi(nx-2,j)+phi(nx-3,j))
     dfi(nx,j)=udx*(-phi(2,j)+8*phi(1,j)-8*phi(nx-1,j)+phi(nx-2,j))
  enddo

  return
end subroutine derix4
!##########################################
```

* Subroutine **deriy4**

```
!###########################################
!
subroutine deriy4(phi,nx,ny,dfi,yly)
!
!Fourth-order first derivative in the y direction
!###########################################

  implicit none

  real(8),dimension(nx,ny) ::  phi,dfi
  real(8) :: dly,yly,udy
  integer :: i,j,nx,ny

  dly=yly/ny
  udy=1./(12*dly)

  do j=3,ny-2
     do i=1,nx
     dfi(i,j)=udy*(-phi(i,j+2)+8*phi(i,j+1)-8*phi(i,j-1)+phi(i,j-2))
     enddo
  enddo

  do i=1,nx
     dfi(i,1)=udy*(-phi(i,3)+8*phi(i,2)-8*phi(i,ny)+phi(i,ny-1))
     dfi(i,2)=udy*(-phi(i,4)+8*phi(i,3)-8*phi(i,1)+phi(i,ny))
     dfi(i,ny-1)=udy*(-phi(i,1)+8*phi(i,ny)-8*phi(i,ny-2)+phi(i,ny-3))
     dfi(i,ny)=udy*(-phi(i,2)+8*phi(i,1)-8*phi(i,ny-1)+phi(i,ny-2))
  enddo

  return
end subroutine deriy4
!###########################################
```

* Subroutine **derixx4**

```
!###########################################
!
subroutine derxx4(phi,nx,ny,dfi,xlx)
!
!Fourth-order second derivative in y direction
!###########################################

  implicit none

  real(8),dimension(nx,ny) ::  phi,dfi
  real(8) :: dlx,xlx,udx
  integer :: i,j,nx,ny

  dlx=xlx/nx
  udx=1./(12.*dlx*dlx)
  do j=1,ny
     dfi(1,j)=udx*(-phi(3,j)+16*phi(2,j)-30*phi(1,j)+16*phi(nx,j)-phi(nx-
1,j))
     dfi(2,j)=udx*(-phi(4,j)+16*phi(3,j)-30*phi(2,j)+16*phi(1,j)-phi(nx,j))
     do i=3,nx-2
```

```
        dfi(i,j)=udx*(-phi(i+2,j)+16*phi(i+1,j)-30*phi(i,j)+16*phi(i-1,j)-
phi(i-2,j))
     enddo
     dfi(nx-1,j)=udx*(-phi(1,j)+16*phi(nx,j)-30*phi(nx-1,j)+16*phi(nx-2,j)-
phi(nx-3,j))
     dfi(nx,j)=udx*(-phi(2,j)+16*phi(1,j)-30*phi(nx,j)+16*phi(nx-1,j)-
phi(nx-2,j))
  enddo


  return
end subroutine derxx4
!###########################################
```

* Subroutine **deriyy4**

```
!###########################################
!
subroutine deryy4(phi,nx,ny,dfi,yly)
!
!Fourth-order second derivative in the y direction
!###########################################

  implicit none

  real(8),dimension(nx,ny) ::  phi,dfi
  real(8) :: dly,yly,udy
  integer :: i,j,nx,ny

  dly=yly/ny
  udy=1./(12.*dly*dly)
  do i=1,nx
     dfi(i,1)=udy*(-phi(i,3)+16*phi(i,2)-30*phi(i,1)+16*phi(i,ny)-phi(i,ny-
1))
     dfi(i,2)=udy*(-phi(i,4)+16*phi(i,3)-30*phi(i,2)+16*phi(i,1)-phi(i,ny))
     do j=3,ny-2
        dfi(i,j)=udy*(-phi(i,j+2)+16*phi(i,j+1)-30*phi(i,j)+16*phi(i,j-1)-
phi(i,j-2))
     enddo
     dfi(i,ny-1)=udy*(-phi(i,1)+16*phi(i,ny)-30*phi(i,ny-1)+16*phi(i,ny-2)-
phi(i,ny-3))
     dfi(i,ny)=udy*(-phi(i,2)+16*phi(i,1)-30*phi(i,ny)+16*phi(i,ny-1)-
phi(i,ny-2))
  enddo

  return
end subroutine deryy4
!###########################################
```

Run a first simulation with $Re = 200$;
Time Step $nt = 10000$;
Fourth-order Runge-Kutta Scheme;
Mesh number in each axis is $n_x * n_y = 129 * 129$;
The below figures are for the four visualisations of the vorticity field under $CFL = 0.75$;

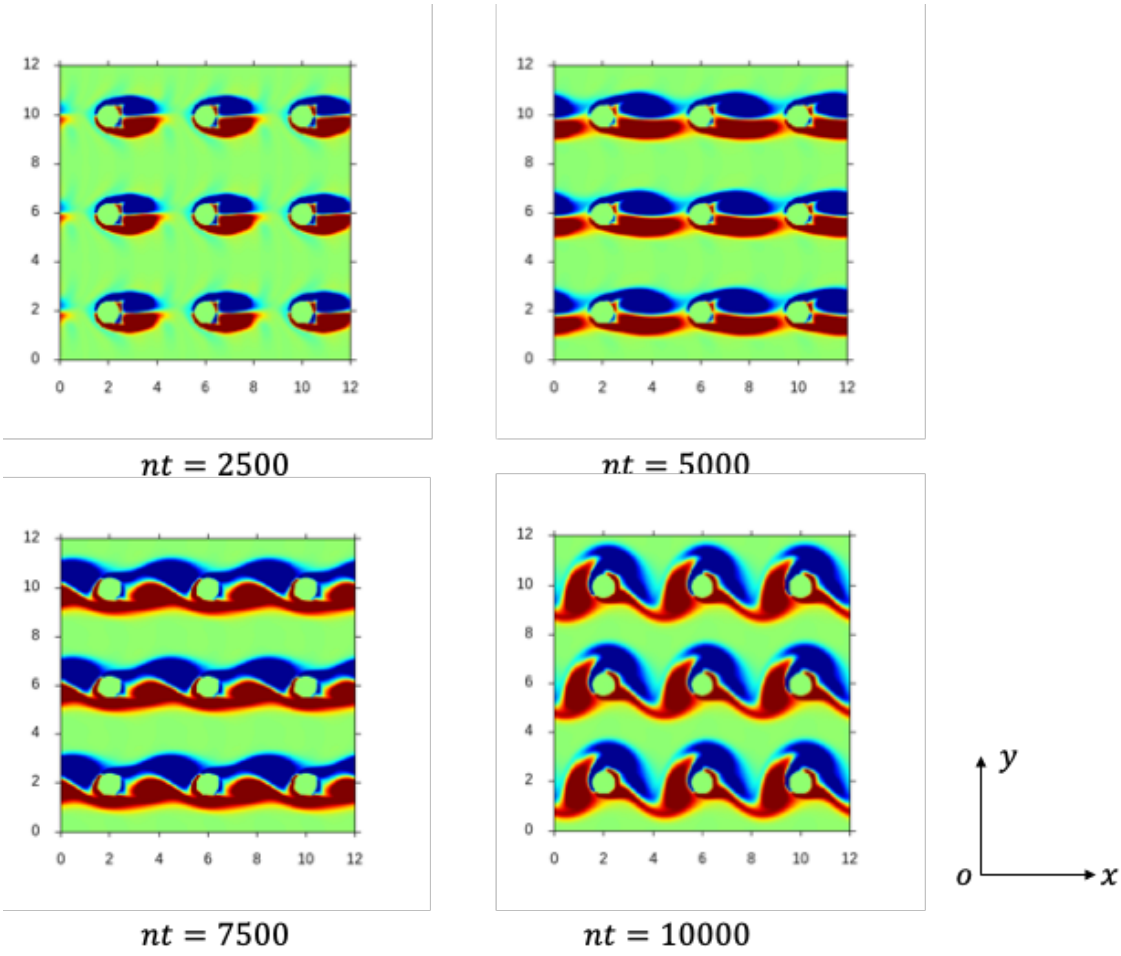


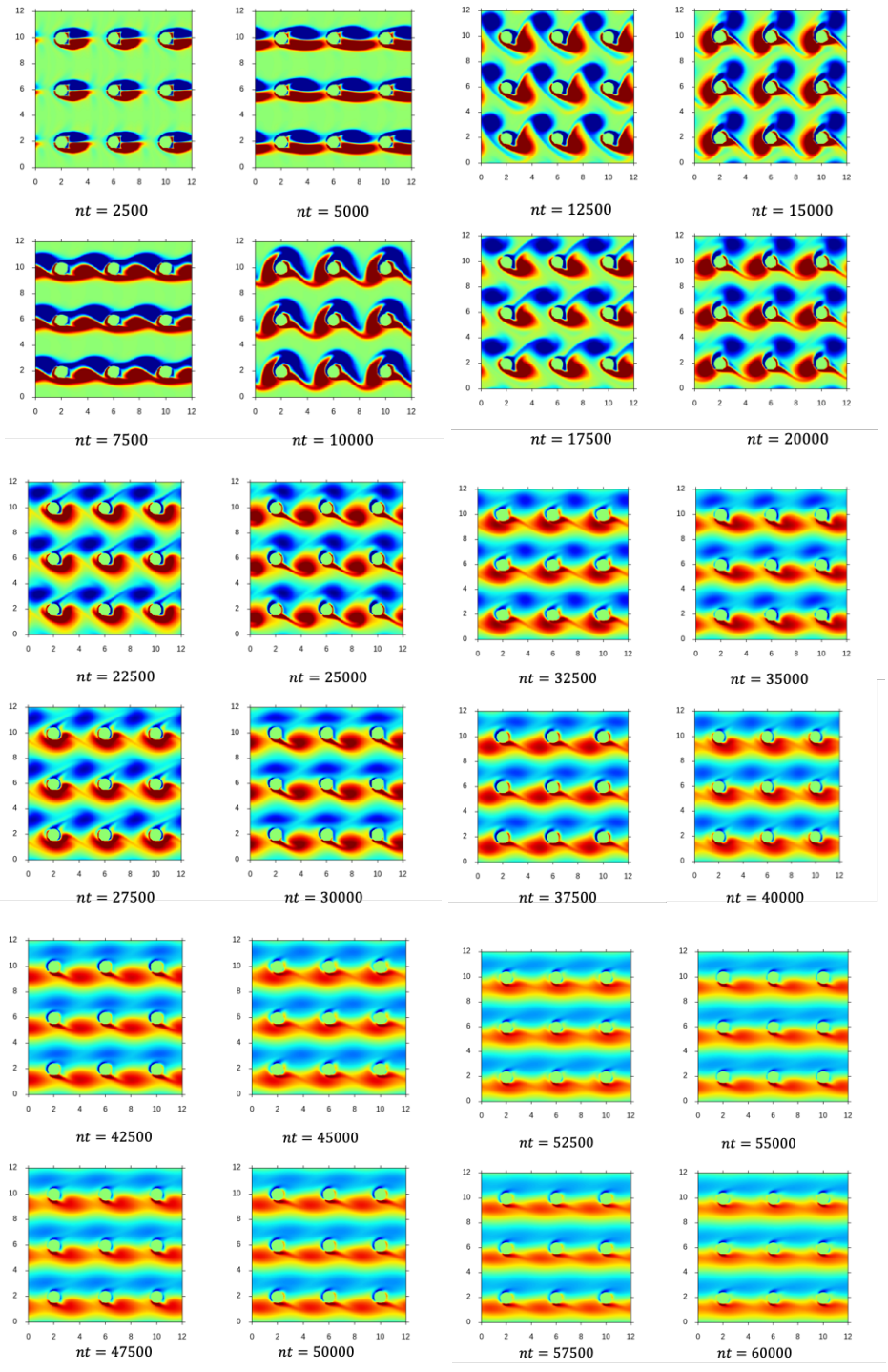
Figure 4.1-4.4: *Displaying the visualisations of the vorticity field in different time steps-in fourth-order rkutta subroutine;*

Description:

After generating the fourth order centred scheme, comparing to the first question results, first, there is no such huge difference between them, four diagrams present in a high quality of vorticities profile in system when constant flow happens; also, the vorticity flow (or the vorticity region), where the upper one stays more on the backwards of the flow, and the below in forwards. In fact, there is no such specific differences need to compare to the first one, due to the higher order in the same time step unit for present the stage of vorticity profile figure shootings are the same, only the cost of this one will be more expensive than the first one.

Question 5

To generate a very long time, this section has been set in $nt = 100000$;
For each time step in 2500 to generate a simulation screen shoot;
For the results, there are 40 pictures have been come out in the result, which listed below:



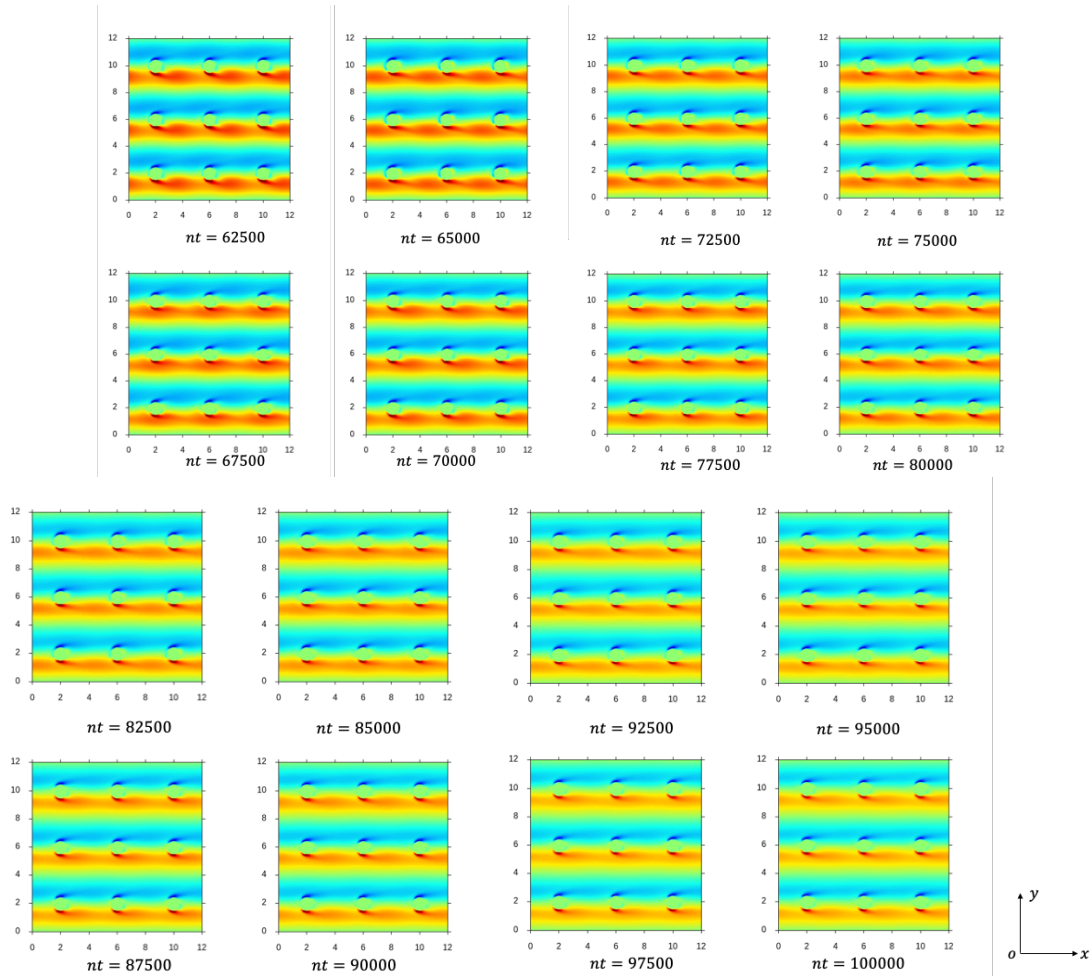


Figure 5.1-5.40: *Displaying the visualisations of the vorticity field in different time steps-run the script in long time;*

Description:

By increasing the time step to generate the plot, firstly the curve of vorticity edge become smoother than in the low time duration; the next change in diagram is that the vorticity boundaries are not in sharply separated from the uniform flow stream. However, it can be seen that when the time step larger than 8000, there is no such hugh difference between each plot outcome. When the time duration increases, the velocity profile is integrating with high time range magnitude, the edge and boundary will become smoother and without oscillation happen, when the time step increases, flow separate edge will be connected by the smoother calcultation functions in continuous way.

Question 6

Dimensions of the domain: $L_x * L_y = 20d * 12d$;
Mesh nodes number and size in decision: $n_x * n_y = 513 * 257$;
Run a first simulation with $Re = 200$;
Time Step $nt = 10000$;
Third-order **Runge-Kutta** Scheme: `itemp=2`;
The below figures are for the four visualisations of the vorticity field under $CFL = 0.25$;

* Setting the new boundary conditions for the new inlet/outlet situation, which the periodic conditions are not relevant anymore, for this question the use of non-centred schemes for the first and last points.)

$$\frac{\partial \phi}{\partial t} + U_c \frac{\partial \phi}{\partial x} = 0$$

* Subroutine **boundary**

```
!############################################################
!
subroutine boundary(uuu,vvv,rho,eee,pre,tmp,rou,rov,roe,nx,ny,&
     xlx,yly,xmu,xba,gma,chp,dlx,dlt,eps,scp,xkt,uu0)
!
!############################################################

  implicit none

  real(8),dimension(nx,ny) :: uuu,vvv,rho,eee,pre,tmp,rou,rov,roe,eps,scp
  real(8) :: xlx,yly,xmu,xba,gma,chp,roi,cci,d,tpi,chv,uu0
  real(8) :: epsi,pi,dlx,dly,dlt,ct1,ct2,ct3
  real(8) :: xkt
  integer :: nx,ny,i,j,ic

  call param(xlx,yly,xmu,xba,gma,chp,roi,cci,d,tpi,chv,uu0)

  ct1=uu0*dlt/dlx
  ct2=1-ct1

!#########################################################################
!inlet boundary condition
!#########################################################################
!

  do j=1,ny

        i=1
        rho(i,j)=roi
        rou(i,j)=rho(i,j)*uuu(i,j)
```

```
        rov(i,j)=rho(i,j)*vvv(i,j)
        roe(i,j)=rho(i,j)*eee(i,j)
        scp(i,j)=1.

!######################################################################
!outlet boundary condition
!######################################################################
!

       i=nx
        rho(i,j)=ct2*rho(i,j)+ct1*rho(i-1,j)
        rou(i,j)=ct2*rou(i,j)+ct1*rou(i-1,j)
        rov(i,j)=ct2*rov(i,j)+ct1*rov(i-1,j)
        roe(i,j)=ct2*roe(i,j)+ct1*roe(i-1,j)
        scp(i,j)=ct2*scp(i,j)+ct1*scp(i-1,j)

enddo

  return
end subroutine boundary
```

* Subroutine **derix**

```
!############################################
!
subroutine derix(phi,nx,ny,dfi,xlx)
!
!First derivative in the x direction
!############################################

  implicit none

  real(8),dimension(nx,ny) :: phi,dfi
  real(8) :: dlx,xlx,udx
  integer :: i,j,nx,ny

  dlx=xlx/nx
  udx=1./(dlx+dlx)
  do j=1,ny
     dfi(1,j)=2.*udx*(phi(2,j)-phi(1,j))
     do i=2,nx-1
        dfi(i,j)=udx*(phi(i+1,j)-phi(i-1,j))
     enddo
     dfi(nx,j)=2.*udx*(phi(nx,j)-phi(nx-1,j))
  enddo

  return
end subroutine derix
!############################################
```

* Subroutine **derixx**

```
!#############################################
!
subroutine derxx(phi,nx,ny,dfi,xlx)
!
!Second derivative in y direction
!#############################################

  implicit none

  real(8),dimension(nx,ny) ::  phi,dfi
  real(8) :: dlx,xlx,udx
  integer :: i,j,nx,ny

  dlx=xlx/nx
  udx=1./(dlx*dlx)
  do j=1,ny
     dfi(1,j)=udx*(phi(3,j)-(phi(2,j)+phi(2,j))+phi(1,j))
     do i=2,nx-1
        dfi(i,j)=udx*(phi(i+1,j)-(phi(i,j)+phi(i,j))&
             +phi(i-1,j))
     enddo
     dfi(nx,j)=udx*(phi(nx,j)-(phi(nx-1,j)+phi(nx-1,j))+phi(nx-2,j))
  enddo

  return
end subroutine derxx
!#############################################
```

* Subroutine **initl**

```
!#############################################################
!
subroutine initl(uuu,vvv,rho,eee,pre,tmp,rou,rov,roe,nx,ny,&
     xlx,yly,xmu,xba,gma,chp,dlx,eta,eps,scp,xkt,uu0)
!
!#############################################################

  implicit none

  real(8),dimension(nx,ny) :: uuu,vvv,rho,eee,pre,tmp,rou,rov,roe,eps,scp
  real(8) :: xlx,yly,xmu,xba,gma,chp,roi,cci,d,tpi,chv,uu0
  real(8) :: epsi,pi,dlx,dly,ct3,ct4,ct5,ct6,y,x,eta,radius
  real(8) :: xkt
  integer :: nx,ny,i,j,ic,jc,imin,imax,jmin,jmax

  call param(xlx,yly,xmu,xba,gma,chp,roi,cci,d,tpi,chv,uu0)

  epsi=0.1
  dlx=xlx/nx
  dly=yly/ny
  ct3=log(2.)
  ct4=yly/2.
  ct5=xlx/2.
  ct6=(gma-1.)/gma
```

```
  y=-ct4
  x=0.
  eta=0.1
  eta=eta/2.
  radius=d/2.
  xkt=xba/(chp*roi)
  pi=acos(-1.)

!######for the square cylinder############################################
      ic=nint((xlx/2./dlx)+1) !X coordinate center of square
      jc=nint((yly/2./dly)+1) !Y coordinate center of square
!      imax=XXX
!      imin=XXX
!      jmax=XXX
!      jmin=XXX
!########################################################################



!##########CYLINDER DEFINITION#############################################
  do j=1,ny
     do i=1,nx
        if (((i*dlx-xlx/4.)**2+(j*dly-yly/2.)**2).lt.radius**2) then
           eps(i,j)=1.
        else
           eps(i,j)=0.
        end if
     enddo
  enddo
!########################################################################
  do j=1,ny
     do i=1,nx
        uuu(i,j)=uu0
        vvv(i,j)=0.01*(sin(4.*pi*i*dlx/xlx)&
             +sin(7.*pi*i*dlx/xlx))*&
             exp(-(j*dly-yly/2.)**2)
        tmp(i,j)=tpi
        eee(i,j)=chv*tmp(i,j)+0.5*(uuu(i,j)*uuu(i,j)+vvv(i,j)*vvv(i,j))
        rho(i,j)=roi
        pre(i,j)=rho(i,j)*ct6*chp*tmp(i,j)
        rou(i,j)=rho(i,j)*uuu(i,j)
        rov(i,j)=rho(i,j)*vvv(i,j)
        roe(i,j)=rho(i,j)*eee(i,j)
        scp(i,j)=1.
        x=x+dlx
     enddo
     y=y+dly
  enddo

  return
end subroutine initl
!############################################################
```

* For the simulating screen shoots:

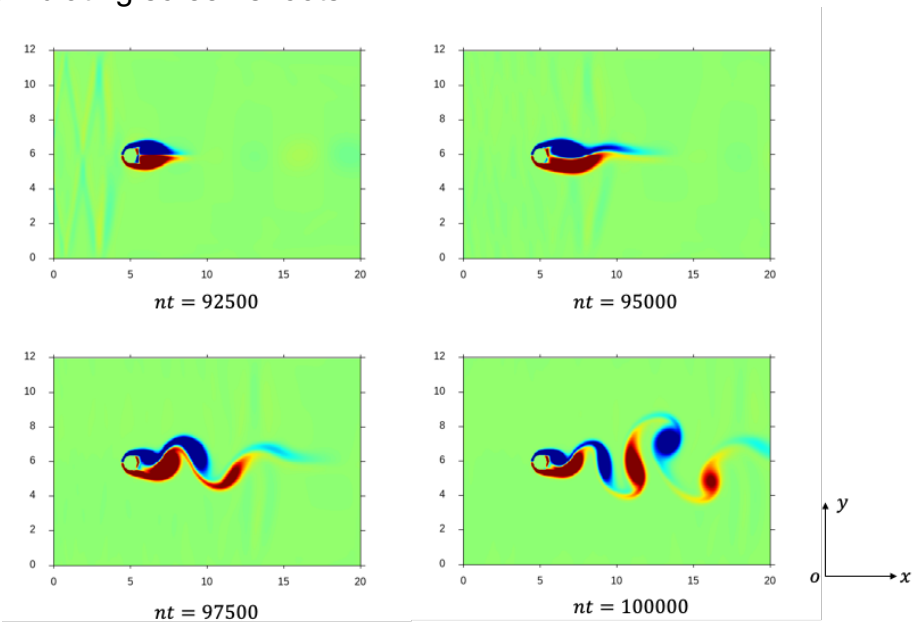


***Figure 6.1-6.4**: Displaying the visualisations of the vorticity field in different time steps-by generating non periodic BCs;*

* For the monitoring script plotting results:

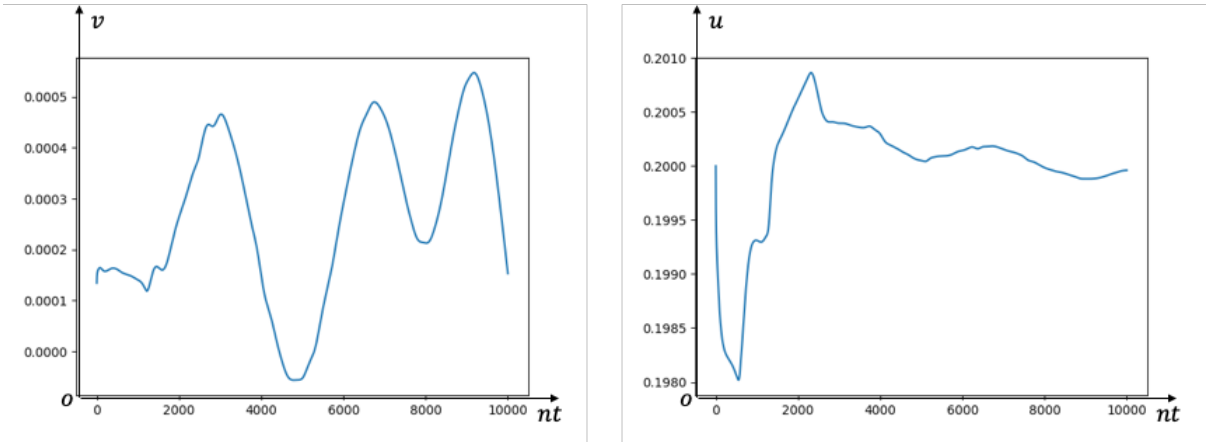


***Figure 6.5-6.6**: Plotting the monitoring results about the velocity magnitude in different time step;*

Appendix

* Complete Script of Monitoring Script: (Python Language)

```
#! /usr/bin/env python
#import matplotlib as mpl #use for remote server
#mpl.use('AGG')           #use for remote server
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import matplotlib.animation as animation
import sys
#########################################################################
#
#  $Name: PLOT_MONITOR
#  $Description: PYTHON SCRIPT FOR PLOT THE MONITOR.DAT FILE AS INTERATING.
#  $Author: Songrui LI
#  $Date: March 2019
#
#########################################################################
def open_point(filename):
        """
        Open an monitor file.
        The header is extracted with the variables names, and the numerical data is
transformed from string (text) to float numbers and stored into a NUMPY array.
        """
        with open(filename,'r') as f:
                data = f.readlines()
        header = data[1].split('" "')
        data = data[2:]
        data = [line.split() for line in data]
        data = [map(float,line) for line in data]
        return header, np.array(data)
def set_axis(header):
        print '\nthe varibles are:'
        print header
        print '\nx axis is pre-set as timestep'
        num_x = 1               #x axis variable number: time
        print '\ninput y axis variable number (started from 1)'
        num_y = input()
        return num_x, num_y
def animate(i):
        header1, data1 = open_point(filename)
        ax1.clear()
        ax1.plot(data1[:,num_x-1], data1[:,num_y-1])
filename = sys.argv[1]
#filename = 'M09_10.0.monitor.tmp.dat'
header, data = open_point(filename)
num_x, num_y = set_axis(header)
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ani = animation.FuncAnimation(fig, animate, interval=10)
plt.show()
```