Imperial College London

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

Supervisor: Prof Sylvain Laizet
Department: Department of Aeronautics
Course: AERO70008
Author: Jason Zhao
Date: 30/06/2025

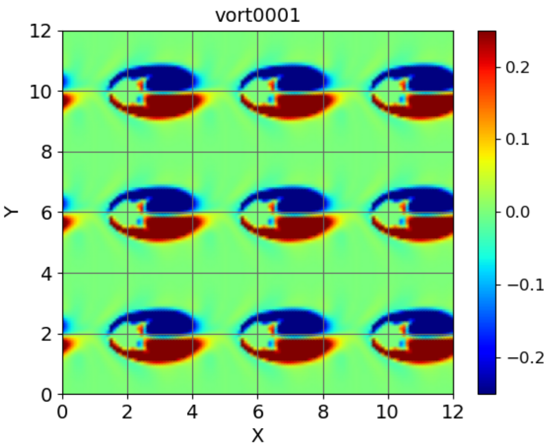
Department of Aeronautics
South Kensington Campus
Imperial College London
London SW7 2AZ
U.K.

Contents

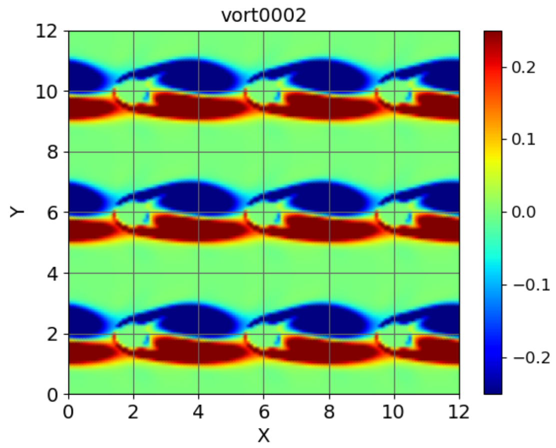
1 Baseline Simulation with Second-Order Adams-Bashforth

The results demonstrate numerical stability under given perturbation, and shows typical pattern under given Reynolds number. The vortices formed behind the cylinder with maximum range around 0.22. The upper vortices are clockwise and lowers are anticlockwise.

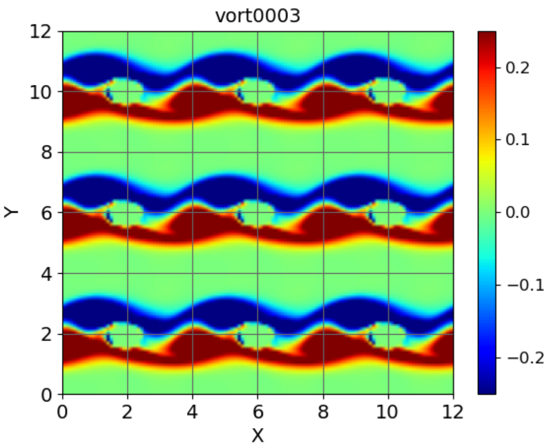
From $nt = 7500$ onward, alternative periodic vortex sheddings develop downstream the cylinders, and they influence each other and formed typical Karman vortex street [1]. Hence, the simulation demonstrates expected flow features for a subcritical Reynolds number regime.



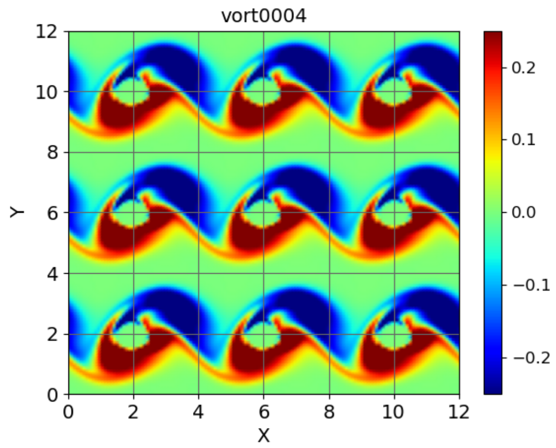
(a) Vorticity field for $nt = 2500$



(b) Vorticity field for $nt = 5000$



(c) Vorticity field for $nt = 7500$



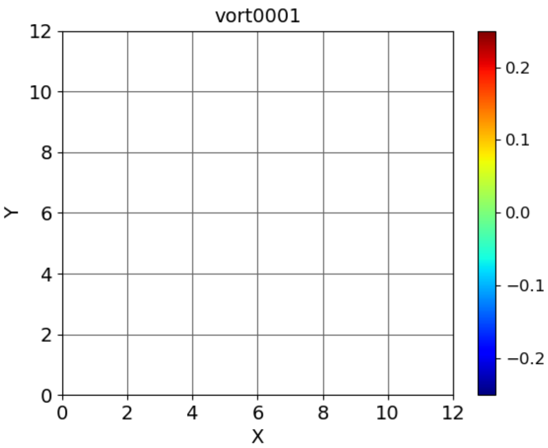
(d) Vorticity field for $nt = 10000$

Figure 1. *Time evolution of the vorticity field for different time steps (nt = 2500, 5000, 7500, 10000) using the second-order Adams-Bashforth scheme with Re = 200 and CFL = 0.25.*

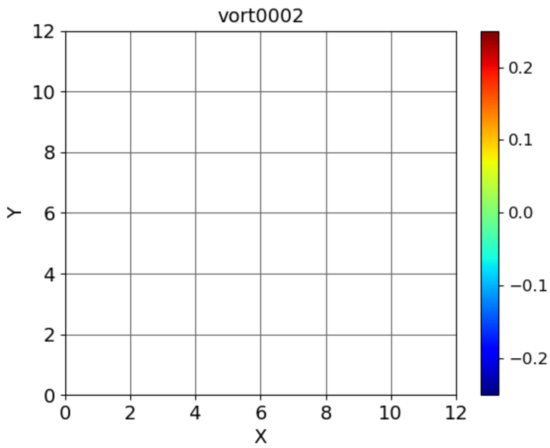
2 Stability Analysis with Increased CFL

There is no meaningful result generated. The numerical values grew unbounded and the scheme becomes numerically unstable. Adams-Bashforth is an explicit scheme, thus sensitive to time step, which is determined by CFL condition (1). Large time step causing oscillation and eventually grow unbounded. $CFL = 0.75$ is above the threshold value and caused unstable

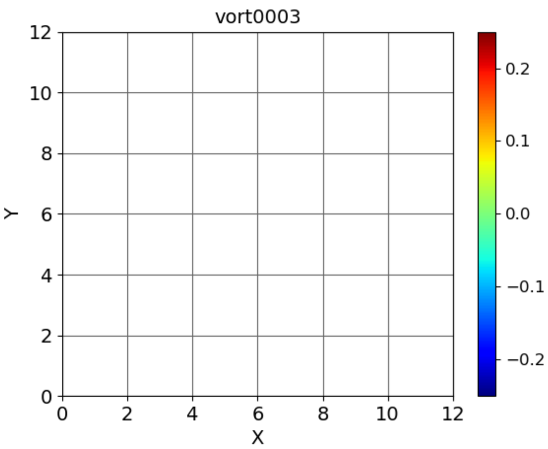
$$\text{CFL} = \frac{u\,\Delta t}{\Delta x} \;\leq\; \text{(Threshold Value)}. \tag{1}$$



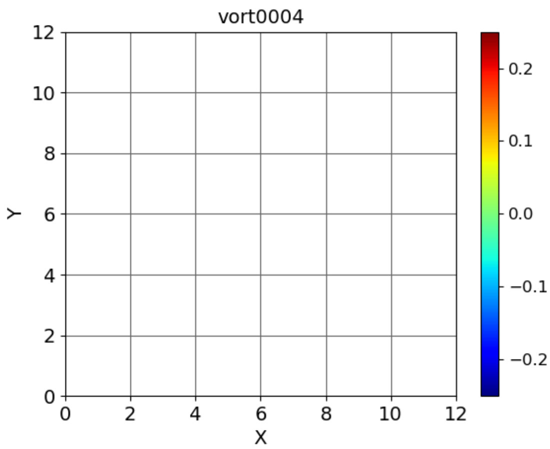
(a) Vorticity field for $nt = 2500$



(b) Vorticity field for $nt = 5000$



(c) Vorticity field for $nt = 7500$

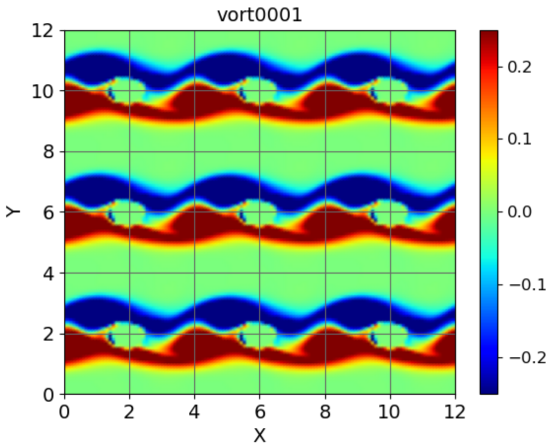


(d) Vorticity field for $nt = 10000$

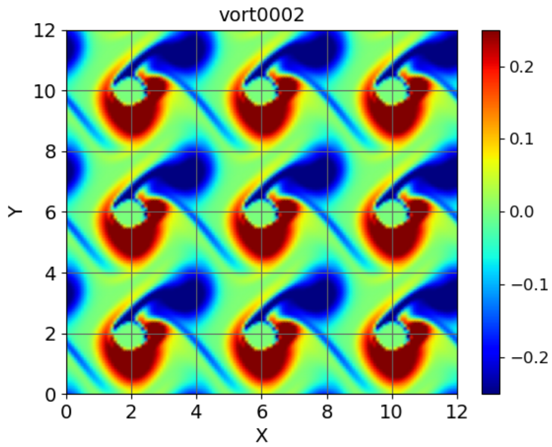
Figure 2. *Time evolution of the vorticity field for different time steps (nt = 2500, 5000, 7500, 10000) using the second-order Adams-Bashforth scheme with Re = 200 and CFL = 0.75. No meaningful results generated.*

3 Implementing a Third-Order Runge-Kutta Scheme

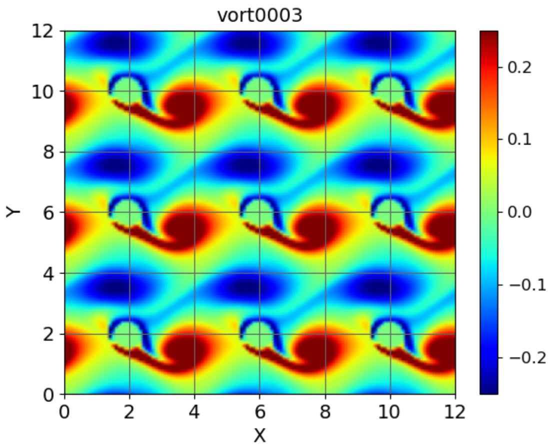
The third order Runge-Kutta scheme remains stable and confirming robust robustness at $CFL = 0.75$. Compared with second-order Adams-Bashforth, the higher-order Runge-Kutta scheme captures finer details, and has a larger stability region Compared to Second-Order Adams-Bashforth scheme, Runge-Kutta has more accuracy at high frequency and phase error, allowing it remains stable at a larger time step. Meanwhile, larger time step reduces the computational cost. Figure 3a and Figure 1c are the same time second simulation and shows the same pattern vortex sheddings.



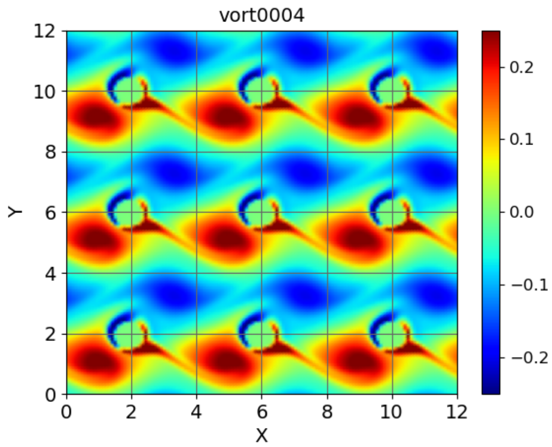
(a) Vorticity field for $nt = 2500$



(b) Vorticity field for $nt = 5000$



(c) Vorticity field for $nt = 7500$



(d) Vorticity field for $nt = 10000$

Figure 3. *Time evolution of the vorticity field for different time steps (nt = 2500, 5000, 7500, 10000) using the third-order Runge-kutta scheme with Re = 200 and CFL = 0.75.*

- Subroutine **rkutta()**:

```
subroutine rkutta(rho,rou,rov,roe,fro,gro,fru,gru,frv,grv,&
     fre,gre,nx,ny,ns,dlt,coef,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
  real(8),dimension(2,ns) :: coef
```

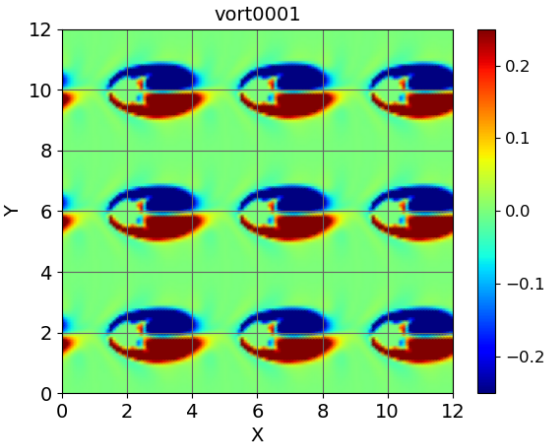
```
  real(8) :: dlt
  integer :: i,j,nx,ny,ns,k
!
!coefficient for RK sub-time steps
  coef(1,1)=8./15.
  coef(1,2)=5./12.
  coef(1,3)=3./4.
  coef(2,1)=0.
  coef(2,2)=-17./60.
  coef(2,3)=-5./12.

  do j=1,ny
     do i=1,nx
!!
         rho(i,j)=rho(i,j)+dlt*(coef(1,k)*fro(i,j)+coef(2,k)*gro(i,j))
         gro(i,j)=fro(i,j)!g is k-1
         rou(i,j)=rou(i,j)+dlt*(coef(1,k)*fru(i,j)+coef(2,k)*gru(i,j))
         gru(i,j)=fru(i,j)
         rov(i,j)=rov(i,j)+dlt*(coef(1,k)*frv(i,j)+coef(2,k)*grv(i,j))
         grv(i,j)=frv(i,j)
         roe(i,j)=roe(i,j)+dlt*(coef(1,k)*fre(i,j)+coef(2,k)*gre(i,j))
         gre(i,j)=fre(i,j)

     enddo
  enddo

  return
end subroutine rkutta
```

4 Implementing Fourth-Order Centered Differences

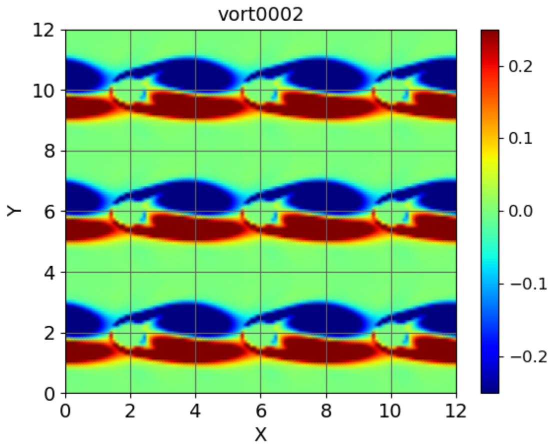
The centered fourth-order schemes are computed from Taylor's expansion as equations (2) and (3). Compared to task 1, a longer computational time is seen as more points involved in computations. It is expected that centered fourth-order schemes should have less dissipation and dispersion error, capturing finer details such as turbulence, small vortices, and shear layer [2]. However, no significant difference is seen. This might due to the large grid size which also causes the error, or due to time step Δt too small, making the advantage of higher order scheme difficult to be seen.

$$\left.\frac{d\phi}{dx}\right|_i \approx \frac{-\phi_{i+2} + 8\phi_{i+1} - 8\phi_{i-1} + \phi_{i-2}}{12\Delta x} \quad (2)$$

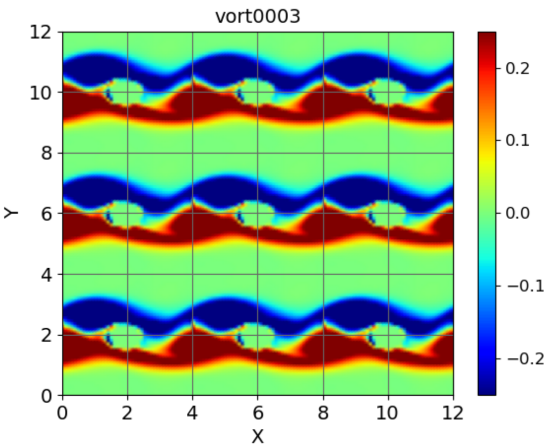
$$\left.\frac{d^2\phi}{dx^2}\right|_i \approx \frac{-\phi_{i+2} + 16\phi_{i+1} - 30\phi_i + 16\phi_{i-1} - \phi_{i-2}}{12(\Delta x)^2} \quad (3)$$



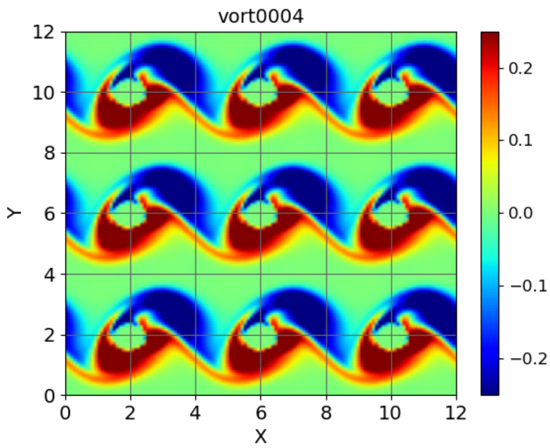
(a) Vorticity field for $nt = 2500$



(b) Vorticity field for $nt = 5000$



(c) Vorticity field for $nt = 7500$



(d) Vorticity field for $nt = 10000$

Figure 4. *Time evolution of the vorticity field for different time steps (nt = 2500, 5000, 7500, 10000) using the second-order Adams-Bashforth scheme with Re = 200 and CFL = 0.25.*

- Subroutine **derix4()**, **deriy4()**, **derxx4()**, and **deryy4()**:

```
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)   !updated multiplier
  do j=1,ny
      dfi(1,j)=udx*(-phi(3,j)+8.*phi(2,j)-8.*phi(nx,j)+phi(nx-1,j))  !Boundary Condition: i=1
      dfi(2,j)=udx*(-phi(4,j)+8.*phi(3,j)-8.*phi(1,j)+phi(nx,j))  !Boundary Condition: i=2
     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))  !i=3 to nx-2
     enddo
     dfi(nx-1,j)=udx*(-phi(1,j)+8.*phi(nx,j)-8.*phi(nx-2,j)+phi(nx-3,j))  !Boundary Condition: i=nx-1
     dfi(nx,j)=udx*(-phi(2,j)+8.*phi(1,j)-8.*phi(nx-1,j)+phi(nx-2,j))  !Boundary Condition: i=nx
  enddo

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

!#############################################
!
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 !dly:grid size; yly: total length
  udy=1./(12.*dly)   !updated multiplier
  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))  !j=3 to ny-2
     enddo
  enddo
  do i=1,nx !Boundary Condition
     dfi(i,1)=udy*(-phi(i,3)+8.*phi(i,2)-8.*phi(i,ny)+phi(i,ny-1))   !Boundary Condition: j=1
     dfi(i,2)=udy*(-phi(i,4)+8.*phi(i,3)-8.*phi(i,1)+phi(i,ny))   !Boundary Condition: j=2
     dfi(i,ny-1)=udy*(-phi(i,1)+8.*phi(i,ny)-8.*phi(i,ny-2)+phi(i,ny-3))  !Boundary Condition: j=ny-1
     dfi(i,ny)=udy*(-phi(i,2)+8.*phi(i,1)-8.*phi(i,ny-1)+phi(i,ny-2))  !Boundary Condition: j=ny
  enddo


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

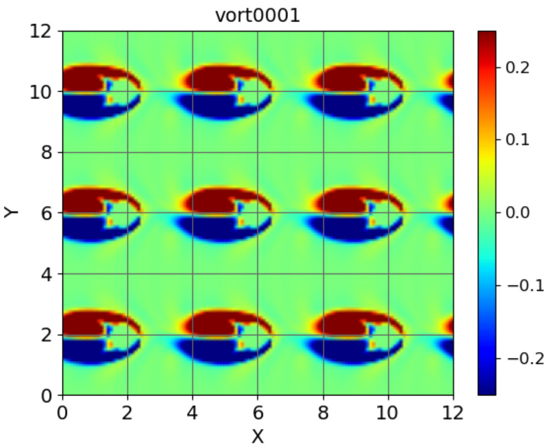
```fortran

!#############################################
!
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)   !updated multiplier
  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))    !Boundary Condition: i=1
     dfi(2,j)=udx*(-phi(4,j)+16.*phi(2,j)-30.*phi(2,j)+16.*phi(1,j)-phi(nx,j))    !Boundary Condition: i=2
     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))    !i=3 to nx-2
     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))    !Boundary Condition: i=nx-1
     dfi(nx,j)=udx*(-phi(2,j)+16.*phi(1,j)-30.*phi(nx,j)+16.*phi(nx-1,j)-phi(nx-2,j))    !Boundary Condition: i=nx
   enddo

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

!#############################################
!
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)   !updated multiplier
  do j=3,ny-2
     do i=1,nx
        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))    !j=3 to ny-2
     enddo
  enddo
  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))    !Boundary Condition: j=1
   dfi(i,2)=udy*(-phi(i,4)+16.*phi(i,3)-30.*phi(i,2)+16.*phi(i,1)-phi(i,ny))  !Boundary Condition: j=2
   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))    !Boundary Condition: j=ny-1
   dfi(i,ny)=udy*(-phi(i,2)+16.*phi(i,1)-30.*phi(i,ny)+16.*phi(i,ny-1)-phi(i,ny-2))
```

```
      !Boundary Condition: j=ny
  enddo


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

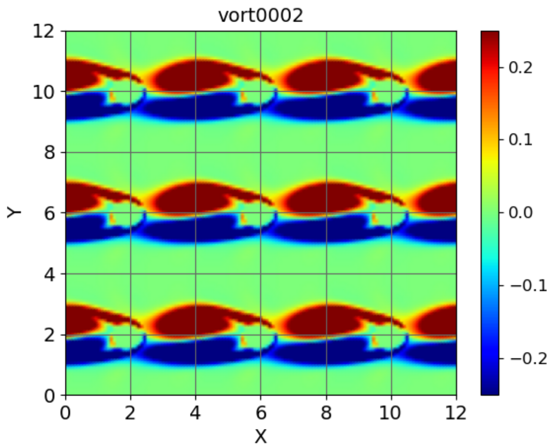
5 Long-Term Flow Behavior Analysis

When the simulation continues, the vortex shedding begin to vanish, and the flow becomes uniformed. The numerical value of the simulation gradually dissipates, and the outline of the vortices becomes blurry. This is due to the dissipation of the second-order Adams-Bashforth scheme and finite difference, and the boundary condition which smooth out the small-scale vortices.

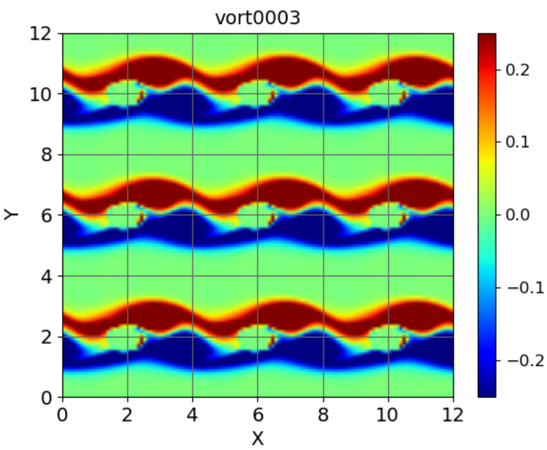
6 Reverse Flow Direction Simulation



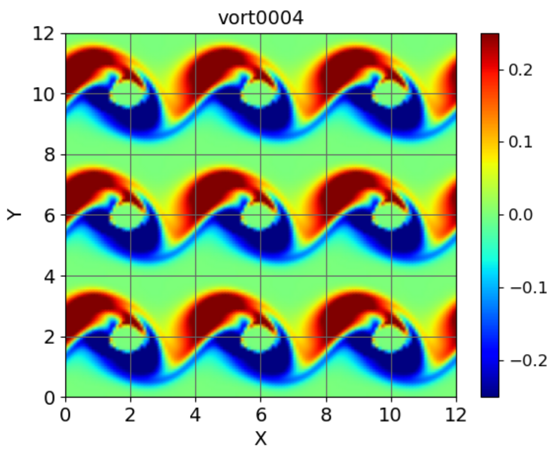
(a) Vorticity field for $nt = 2500$



(b) Vorticity field for $nt = 5000$



(c) Vorticity field for $nt = 7500$



(d) Vorticity field for $nt = 10000$

Figure 5. *Time evolution of the vorticity field for different time steps (nt = 2500, 5000, 7500, 10000) using the second-order Adams-Bashforth scheme with Re = 200 and CFL = 0.25, with flow coming from right to left.*

- In the subroutine **initl**, change *uuu* to the negative direction.

```
    uuu(i,j)=-uu0    !negtive
```

- Additional: change the index of y-direction turbulence to *nx + 1 - i* to get a more accurate mirrored pattern the original case.

```
    vvv(i,j)=0.01*(sin(4.*pi*(nx+1-i)*dlx/xlx)&
         +sin(7.*pi*(nx+1-i)*dlx/xlx))*&
         exp(-(j*dly-yly/2.)**2)
```

7 Simulation of a Single Cylinder with Inflow/Outflow Boundary Conditions

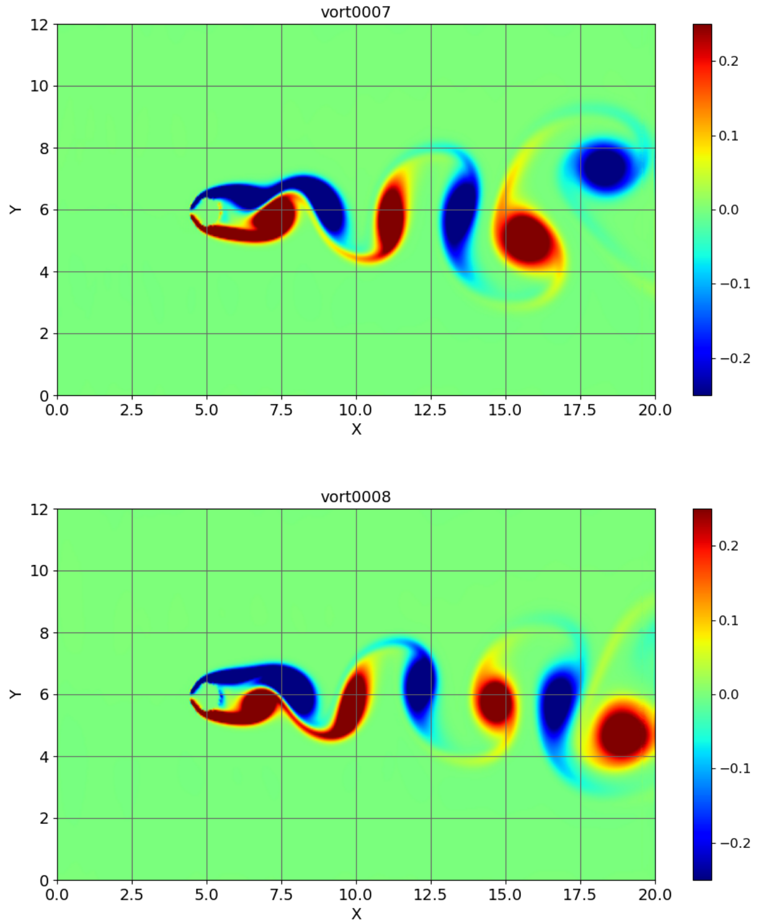


Figure 6. *Instantaneous visualisations of the flow around a single cylinder with inflow/outflow boundary conditions in the streamwise direction.*

1. New subroutine **boundary()**:

```
subroutine boundary(uuu,vvv,rho,eee,tmp,rou,rov,roe,nx,ny,&
xlx,yly,xmu,xba,gma,dlx,dlt,uu0)

   implicit none

   real(8),dimension(nx,ny) :: uuu,vvv,rho,eee,tmp,rou,rov,roe
   real(8) :: xlx,yly,xmu,xba,gma,roi,cci,d,tpi,chv,uu0
   real(8) :: pi,dlx,dly,dlt,ct6
   integer :: nx,ny,j

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

   ct6=(gma-1.)/gma
   pi=acos(-1.)

   !#########Inlet Condition################################
   do j=1,ny
      uuu(1,j)=uu0
      vvv(1,j)=0.01*(sin(4.*pi*(1)*dlx/xlx)&
```

```
         +sin(7.*pi*(1)*dlx/xlx))*&
         exp(-(j*dly-yly/2.)**2)
      tmp(1,j)=tpi
      eee(1,j)=chv*tmp(1,j)+0.5*(uuu(1,j)*uuu(1,j)+vvv(1,j)*vvv(1,j))
      rho(1,j)=roi
      rou(1,j)=rho(1,j)*uuu(1,j)
      rov(1,j)=rho(1,j)*vvv(1,j)
      roe(1,j)=rho(1,j)*eee(1,j)
   enddo

   !#########Outlet Condition################################
   do j=1,ny
      rho(nx,j)=rho(nx,j)-uu0*dlt/dlx*(rho(nx,j)-rho(nx-1,j))
      rou(nx,j)=rou(nx,j)-uu0*dlt/dlx*(rou(nx,j)-rou(nx-1,j))
      rov(nx,j)=rov(nx,j)-uu0*dlt/dlx*(rov(nx,j)-rov(nx-1,j))
      roe(nx,j)=roe(nx,j)-uu0*dlt/dlx*(roe(nx,j)-roe(nx-1,j))

   enddo

   return
end subroutine boundary
```

2. Modified 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
    !Inlet Condition, First order forward difference scheme
     dfi(1,j)=(1./dlx)*(phi(2,j)-phi(1,j))
     do i=2,nx-1
        dfi(i,j)=udx*(phi(i+1,j)-phi(i-1,j))
     enddo
    !Outlet Condition, First order backward difference scheme
     dfi(nx,j)=(1./dlx)*(phi(nx,j)-phi(nx-1,j))
  enddo

  return
end subroutine derix
```

3. Modified subroutine **derxx()**:

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

  implicit none

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

```

```fortran
  dlx=xlx/nx
  udx=1./(dlx*dlx)
  do j=1,ny
     !Inlet Boundary Condition, Second-order one-sided difference scheme
     dfi(1,j)=udx*((phi(1,j)+phi(1,j))-5.*phi(2,j)+4.*phi(3,j)-phi(4,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
     !Outlet Boundary Condition, Second-order one-sided difference scheme
     dfi(nx,j)=udx*((phi(nx,j)+phi(nx,j))-5.*phi(nx-1,j)+4.*phi(nx-2,j)-phi(nx-3,j))
  enddo

  return
end subroutine derxx
```

4. Additional. Implementing subroutine **boundary** in the time loop:

```fortran
    call adams(rho,rou,rov,roe,fro,gro,fru,gru,frv,grv,&
             fre,gre,nx,ny,dlt)

    !Imposing Boundary Condition
    call boundary(uuu,vvv,rho,eee,tmp,rou,rov,roe,nx,ny,&
    xlx,yly,xmu,xba,gma,dlx,dlt,uu0)

    call state(uuu,vvv,rho,pre,tmp,rou,rov,roe,nx,ny,gma)
```

5. Additional. Layout amendments in the **2D_compressible.f90** file:

```fortran
  !nx->513, ny->257, nf->1
  integer,parameter :: nx=513,ny=257,nt=15000,ns=3,nf=1,mx=nf*nx,my=nf*ny
```

```fortran
!##########CIRCULAR CYLINDER DEFINITION#################################
  do j=1,ny
     do i=1,nx
         !Centre of cylinder located at (5d,6d)
        if (((i*dlx-5.d0)**2+(j*dly-6.d0)**2).lt.radius**2) then
           eps(i,j)=1.
        else
           eps(i,j)=0.
        end if
     enddo
  enddo
```

```fortran
  xlx=20.*d     !DOMAIN SIZE X DIRECTION
  yly=12.*d     !DOMAIN SIZE Y DIRECTION
```

6. Additional. Layout amendments in the **plot_py** file:

```python
#nx->513, ny->257, lx->20.0, ly->12.0
nx=513
ny=257
lx=20.0
ly=12.0

#12->lx
X=np.linspace(0,lx,num=nx)
```

```
#plot
fig,ax = plt.subplots(figsize=(12,6))
plt.figure((i+1),figsize=(20,12), edgecolor='none')
```

References

[1] Cooper JE. AEROELASTIC RESPONSE. In: Braun S, editor. Encyclopedia of Vibration. Oxford: Elsevier; 2001. p. 87-97. Available from: https://www.sciencedirect.com/science/article/pii/B0122270851001259.

[2] Wang ZJ, Fidkowski K, Abgrall R, Bassi F, Caraeni D, Cary A, et al. High-Order CFD Methods: Current Status and Perspective. International Journal for Numerical Methods in Fluids. 2013;72(8):811-45. Available from: https://www.researchgate.net/publication/236950745_High-Order_CFD_Methods_Current_Status_and_Perspective.