Computational fluid dynamics (CFD) simulation of viscid, laminar, external flow over a square block using an explicit MacCormack numerical scheme☆

Paul D. Gessler

Department of Mechanical Engineering, Marquette University, Milwaukee, WI

Abstract

The external fluid flow around a square cylinder is simulated numerically using an explicit MacCormack solution scheme implemented in `FORTRAN 90`. Two flow velocities are considered: the first (Re = 20) produces a steady solution with a pair of vortices attached to the trailing face of the block, while the second (Re = 100) results in an unsteady vortex shedding occurring along a vortex street behind the block.

The steady case results in a drag coefficient $C_D = 5.119$ and minimal lift, while the unsteady case has a mean drag coefficient $C_D = 2.991$ with an amplitude of lift coefficient $C_L = 0.678$. These results differ from literature values by a maximum of 33% under identical conditions. The discrepancy is attributed to an error in the computation of the the drag and lift coefficients of the block.

Keywords: CFD, square block, vortex shedding, explicit MacCormack scheme, finite difference method

1. Introduction

This paper discusses simulations of the viscid, laminar, external flow around a square block at two selected Reynolds numbers (Re = 20 and Re = 100). The boundary conditions are developed using the continuity and momentum equations. An explicit MacCormack scheme is used to compute the numerical solution.

2. Problem Geometry and Flow Conditions

The problem geometry is as specified in Fig. 10.9 of Kundu et al. (2012), with square dimension D, channel height $H = 3D$ and a channel length $L = 35D$. The leading edge of the block is positioned at $x = 15D$, centered vertically within the channel.

The flow conditions, as given by Kundu et al. (2012), are Ma = 0.05 and Re = 20 for the steady case or Re = 100 for the unsteady vortex shedding case. The inflow occurs at the left edge of the domain, and the sides (top and bottom) of the channel move through the domain at the same velocity as the inflow.

3. Boundary and Initial Conditions

The boundary conditions are from Kundu et al. (2012), §10.5. Any modifications to the boundary conditions for the specific geometry and flow conditions considered presently are presented in Sections 3.1 to 3.4.

3.1. Inflow

At the left edge of the domain, $u = U$ and $v = 0$. In terms of the momentum equation, then, we have $\boxed{\rho u = \rho_0 U}$, $\boxed{\rho v = 0}$, and $\boxed{\rho = \rho_0}$ from the continuity equation.

3.2. Outflow

At the right edge of the domain, we require that the momentum gradient is negligible; i.e., $\partial\rho u/\partial x = 0$ and $\partial\rho v/\partial x = 0$. From the continuity equation, the density boundary condition is

$$\rho^*_{i,j} = \rho^n_{i,j} + \frac{\Delta t}{2\Delta x}\left[-(\rho u)^n_{i-2,j} + 4(\rho u)^n_{i-1,j} - 3(\rho u)^n_{i,j}\right], \tag{1}$$

corresponding to Kundu et al. (2012) Eq. 10.141.

For the momentum gradient, consider a second-order accurate backward finite difference approxima-

☆MEEN 6310 Project 1, submitted December 22, 2013 to Dr. J. Borg.

tion of the partial derivative:

$$\frac{\partial \rho u}{\partial x} = 0 \approx \frac{3(\rho u)_i - 4(\rho u)_{i-1} + (\rho u)_{i-2}}{2\Delta x}.$$

Solving for $(\rho u)_i$, we have

$$3(\rho u)_i \approx 4(\rho u)_{i-1} - (\rho u)_{i-2},$$

or

$$(\rho u)_i \approx \Big(4(\rho u)_{i-1} - (\rho u)_{i-2}\Big)/3.$$

Application along all vertical nodes at the outflow gives

$$\boxed{(\rho u)^n_{i,j} = \Big(4(\rho u)^n_{i-1,j} - (\rho u)^n_{i-2,j}\Big)/3}, \tag{2}$$

and, similarly, for the other component of momentum,

$$\boxed{(\rho v)^n_{i,j} = \Big(4(\rho v)^n_{i-1,j} - (\rho v)^n_{i-2,j}\Big)/3}. \tag{3}$$

3.3. Channel Walls

At the top and bottom of the domain (the moving walls), we use slight variations on Kundu et al. (2012) Eq. 10.145 (correcting the indices as appropriate for the boundary) for the density update equations. For the top of the domain, we have

$$\boxed{\begin{aligned}\rho^*_{i,j} = \rho^n_{i,j} &- \frac{\Delta t U}{2\Delta x}\left[\rho^n_{i+1,j} - \rho^n_{i-1,j}\right] \\ &+ \frac{\Delta t}{2\Delta y}\left[-(\rho v)^n_{i,j-2} + 4(\rho v)^n_{i,j-1} - 3(\rho v)^n_{i,j}\right],\end{aligned}} \tag{4}$$

and, for the bottom of the domain,

$$\boxed{\begin{aligned}\rho^*_{i,j} = \rho^n_{i,j} &- \frac{\Delta t U}{2\Delta x}\left[\rho^n_{i+1,j} - \rho^n_{i-1,j}\right] \\ &- \frac{\Delta t}{2\Delta y}\left[-(\rho v)^n_{i,j+2} + 4(\rho v)^n_{i,j+1} - 3(\rho v)^n_{i,j}\right].\end{aligned}} \tag{5}$$

The momentum conditions at the top and bottom are much simpler; as with the inflow condition, we simply require $\boxed{\rho u = \rho_0 U}$ and $\boxed{\rho v = 0}$.

3.4. Block Surfaces

At all block surfaces, the no-slip condition is in effect, so $\boxed{\rho u = \rho v = 0}$. In practice, these conditions are never required because they are imposed with the initial conditions and the solution is only computed away from the block surface.

Following the procedure outlined by Kundu et al. (2012) in Eqs. 10.147–10.154, we derive the density update equations for each face of the block:

$$\boxed{\begin{aligned}\rho_{i,j}\Big|_{\text{front}} = &\frac{4\rho_{i-1,j} - \rho_{i-2,j}}{3} \\ &+ \frac{8\text{Ma}^2}{9\text{Re}\Delta x}\Big(-5u_{i-1,j} + 4u_{i-2,j} - u_{i-3,j}\Big) \\ &- \frac{\text{Ma}^2}{18\text{Re}\Delta y}\left[\begin{aligned}&-(v_{i-2,j+1} - v_{i-2,j-1}) \\ &+ 4(v_{i-1,j+1} - v_{i-1,j-1}) \\ &- 3(v_{i,j+1} - v_{i,j-1})\end{aligned}\right]\end{aligned}} \tag{6}$$

$$\boxed{\begin{aligned}\rho_{i,j}\Big|_{\text{back}} = &\frac{4\rho_{i+1,j} - \rho_{i+2,j}}{3} \\ &- \frac{8\text{Ma}^2}{9\text{Re}\Delta x}\Big(-5u_{i+1,j} + 4u_{i+2,j} - u_{i+3,j}\Big) \\ &- \frac{\text{Ma}^2}{18\text{Re}\Delta y}\left[\begin{aligned}&-(v_{i+2,j+1} - v_{i+2,j-1}) \\ &+ 4(v_{i+1,j+1} - v_{i+1,j-1}) \\ &- 3(v_{i,j+1} - v_{i,j-1})\end{aligned}\right]\end{aligned}} \tag{7}$$

$$\boxed{\begin{aligned}\rho_{i,j}\Big|_{\text{top}} = &\frac{4\rho_{i,j+1} - \rho_{i,j+2}}{3} \\ &- \frac{8\text{Ma}^2}{9\text{Re}\Delta y}\Big(-5v_{i,j+1} + 4v_{i,j+2} - v_{i,j+3}\Big) \\ &- \frac{\text{Ma}^2}{18\text{Re}\Delta x}\left[\begin{aligned}&-(u_{i+1,j+2} - u_{i-1,j+2}) \\ &+ 4(u_{i+1,j+1} - u_{i-1,j+1}) \\ &- 3(u_{i+1,j} - u_{i-1,j})\end{aligned}\right]\end{aligned}} \tag{8}$$

$$\boxed{\begin{aligned}\rho_{i,j}\Big|_{\text{bot}} = &\frac{4\rho_{i,j-1} - \rho_{i,j-2}}{3} \\ &+ \frac{8\text{Ma}^2}{9\text{Re}\Delta y}\Big(-5v_{i,j-1} + 4v_{i,j-2} - v_{i,j-3}\Big) \\ &- \frac{\text{Ma}^2}{18\text{Re}\Delta x}\left[\begin{aligned}&-(u_{i+1,j-2} - u_{i-1,j-2}) \\ &+ 4(u_{i+1,j-1} - u_{i-1,j-1}) \\ &- 3(u_{i+1,j} - u_{i-1,j})\end{aligned}\right]\end{aligned}} \tag{9}$$

3.5. Initial Conditions

Before computation of the solution begins, the domain is initialized as follows:

Inside block ($15 \le x/D \le 16$; $1 \le y/D \le 2$): $u = v = 0$; $\rho = \rho_0$.

Elsewhere: $u = U$; $v = 0$; $\rho = \rho_0$.

This establishes the base flow; the variation around the block is allowed to develop through time as the solution is computed. The reference density, ρ_0, is dimensionless and arbitrary since only deviations in density are relevant.

4. Numerical Approach

4.1. Algorithm

The explicit MacCormack numerical scheme is used, as developed in Kundu et al. (2012) §10.5 for the driven cavity flow but with modified boundary and initial conditions from Section 3. The complete source code was submitted by email and is listed in Appendix A.

4.2. Stability Criterion

A predictor—corrector update scheme is used, and we use only uniform grids with unity aspect ratio; i.e., $\Delta x = \Delta y$. Thus, the stability criterion (Kundu Eq. 10.155) becomes

$$\Delta t \leq \frac{\sigma}{\sqrt{2}} \mathrm{Ma} \Delta x, \tag{10}$$

where σ is a safety factor to prevent marginal stability. For all results presented in Section 5, $\sigma = 0.9$ was used. There is no appreciable difference in results when lower safety factors are selected.

4.3. Drag and Lift Coefficients

The drag and lift coefficients are computed via numerical integration using the trapezoidal rule over all four faces of the block. To develop expressions for computing the drag and lift coefficients, start with the definition of the drag coefficient,

$$C_D = \frac{2F_d}{\rho v^2 A} = C_p + C_f, \tag{11}$$

where C_p and C_f are the pressure and friction (viscous) drag components, respectively. This decomposes to

$$C_D = \frac{1}{\rho U^2 A}\left(\int_S \left(p - p_0\right) \cdot \hat{n} \cdot \hat{\imath} \, \mathrm{d}A + \int_S T_w \cdot \hat{t} \cdot \hat{\imath} \, \mathrm{d}A\right),$$

or for the in-plane two-dimensional equivalent,

$$C_D = \frac{1}{\rho U^2 D}\left(\int \left(p - p_0\right) \cdot \hat{n} \cdot \hat{\imath} \, \mathrm{d}s + \int T_w \cdot \hat{t} \cdot \hat{\imath} \, \mathrm{d}s\right),$$

and in our case, the dot products simplify because of the simple geometry.

The pressure drag will contribute to the drag coefficient on the front and back faces of the block, and to the lift coefficient on the top and bottom faces of the block. The converse is also true; i.e., the viscous drag will contribute to the lift coefficient on the front and back faces of the block and to the drag coefficient on the top and bottom faces of the block.

For the pressure drag integral on a single face (e.g., the front face here),

$$c_p = \int \left(p - p_0\right) \cdot \hat{n} \cdot \hat{\imath} \, \mathrm{d}s = \int \left(p - p_0\right) \mathrm{d}y.$$

Substituting the provided equation of state (Kundu Eq. 10.99), $p = \rho/\mathrm{Ma}^2$, we have

$$= \mathrm{Ma}^{-2} \int \left(\rho - \rho_0\right) \mathrm{d}y.$$

Then approximate the integral using trapezoidal rule:

$$\boxed{\approx \frac{1}{\mathrm{Ma}^2}\left(\sum_{\text{front}} \frac{\rho^n_{i,j} + \rho^n_{i,j+1}}{2} \Delta y - \rho_0 D\right)}. \tag{12}$$

For the viscous drag integral on a single face (e.g., the top face here),

$$c_f = \int T_w \cdot \hat{t} \cdot \hat{\imath} \, \mathrm{d}s = \int T_w \, \mathrm{d}x,$$

but, assuming a Newtonian fluid, $T_w = \mu \partial u / \partial y\big|_{\text{wall}}$, so

$$= \mu_0 \int \frac{\partial u}{\partial y} \, \mathrm{d}x.$$

Now, applying trapezoidal rule, we find

$$\approx \mu_0 \sum_{\text{top}} \frac{\left(\frac{\partial u}{\partial y}\right)^n_{i,j} + \left(\frac{\partial u}{\partial y}\right)^n_{i+1,j}}{2} \Delta x,$$

and, approximating the partial derivative with a first-order accurate forward difference, we have

$$\approx \mu_0 \sum_{\text{top}} \frac{\left(u_{i,j+1} - u_{i,j}\right)^n + \left(u_{i+1,j+1} - u_{i+1,j}\right)^n}{2} \frac{\Delta x}{\Delta y},$$

or

$$\boxed{\approx \frac{\mu_0 \Delta x}{2 \Delta y} \sum_{\text{top}} \left(u^n_{i,j+1} - u^n_{i,j} + u^n_{i+1,j+1} - u^n_{i+1,j}\right)}. \tag{13}$$

Similar expressions follow for the other faces of the block.

5. Results and Discussion

All results presented here use a safety factor σ = 0.9 for the stability criterion and employ uniform grids with unity aspect ratio.

5.1. Grid Independence Study

To confirm that the solution is not influenced by spatial resolution approaching an infinitely fine grid, a convergence study was performed for the Re = 20 case for various numbers of grid points through the y-direction, n_y. The results of the study are shown in Fig. 1. It is clear that the solution converges to stable drag and lift coefficients as the number of grid points increases.

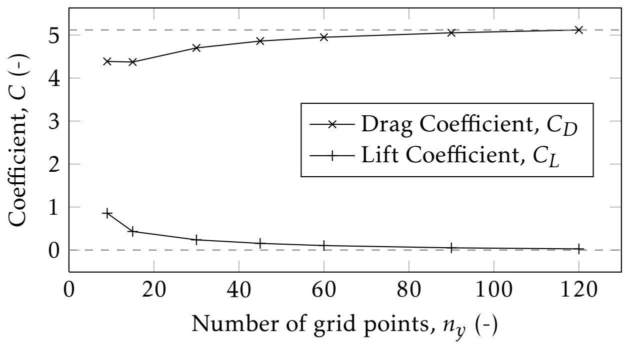


Figure 1: Convergence study on the effect of the number of grid points through the y-direction (n_y) on the drag and lift coefficients.

5.2. Steady Solution—Attached Vortices

For the steady solution (Re = 20), a pair of vortices form at the trailing edge of the block and remain attached throughout the simulation. Figure 2 (pg. 5) shows the streamlines for this solution, clearly showing the near-wake attached vortices.

Dutta et al. (2008) recorded experimental flow fields for square blocks at various angles of incidence. For qualitative validation, a sample flow field at a 0° angle of incidence and Re = 40 is shown in Fig. 3. For this flow field, the recirculation region is significantly larger than the present Re = 20 simulation. This is a result of the increased Reynolds number (Re = 40) in the experimental flow field.

The evolution of the solution through time for n_y = 120 is shown in Fig. 4. A steady solution is achieved by $t \approx 15$, which agrees with $t \approx 20$ shown in Kundu Fig. 10.10(a). The converged results (and comparisons to Kundu et al. (2012)) are listed in Table 1.

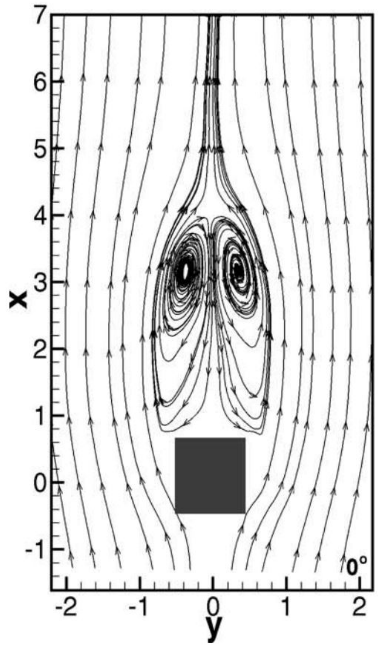


Figure 3: Experimental flow field with 0° angle of incidence and Re = 40. Here, the recirculation zone is longer because of the increased Reynolds number. Adapted from Dutta et al. (2008).

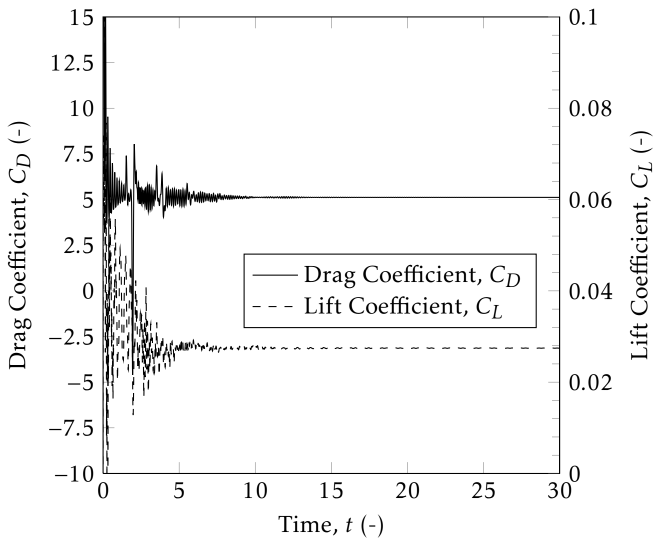


Figure 4: Time evolution of drag and lift coefficients for Re = 20 with n_y = 120. As in Kundu, the solution has reached steady-state by t = 20.

Table 1: Steady solution (Re = 20) results.

Quantity (Units)	Present	Kundu	% Diff.
Drag Coeff., C_D (-)	5.119	7.003	26.9%
Lift Coeff., C_L (-)	0.002	0.003	33.3%

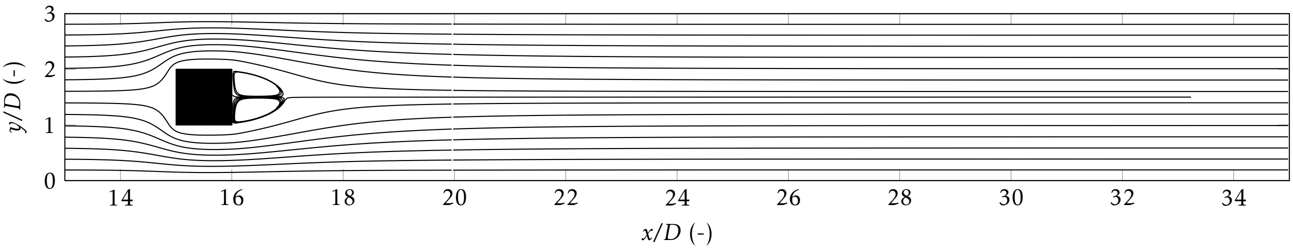


Figure 2: Steady streamlines for flow around a square block at Re = 20, n_y = 120. Flow separation creates two recirculation zones, known as *attached near-wake vortices*.

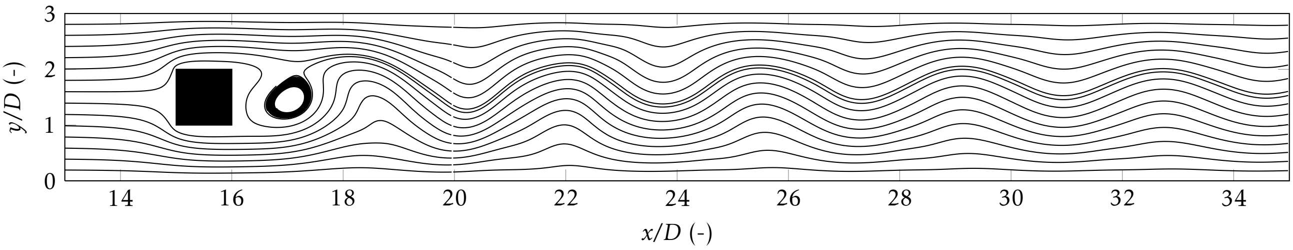


Figure 5: Unsteady streamlines for flow around a square block at Re = 100, n_y = 120, t = 198.87. The near-wake vortices are washed away in an alternating fashion at this Reynolds number.

5.3. *Unsteady Solution—Vortex Shedding*

For the unsteady solution (Re = 100), the vortex pair forms but is swept away as the unsteadiness forms. Upper and lower vortices are washed away in an alternating fashion. Figure 5 shows a typical vortex shedding event, in this case at t = 198.87.

The evolution of the solution through time (after initial transients) for n_y = 120 is shown in Fig. 6. The periodicity of the unsteadiness is clearly apparent, with lift corresponding to a lower vortex shed event and downforce corresponding to an upper vortex shed event. The slight variations in the drag coefficient are results of the changing flow field as vortices are shed.

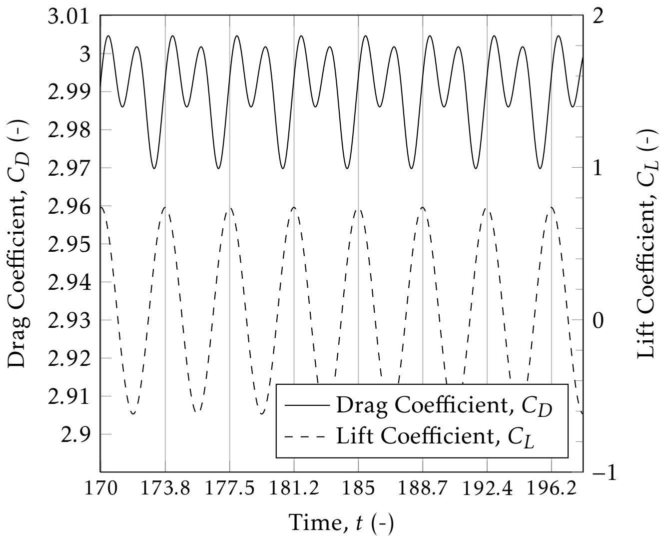


Figure 6: Periodic oscillations of drag and lift coefficients for Re = 100 and n_y = 120. Vortex shedding influences both the drag and lift coefficients on the block.

The streamlines over one full period for Re = 100 and n_y = 120 are shown in Fig. 7 (pg. 6). From Fig. 6, the period of oscillation T = 3.8, yielding a frequency (dimensionless) f = 0.263. We can calculate the Strouhal number for this flow,

$$\mathrm{St} = \frac{fD}{U} = 0.263. \tag{14}$$

The converged results are shown in Table 2.

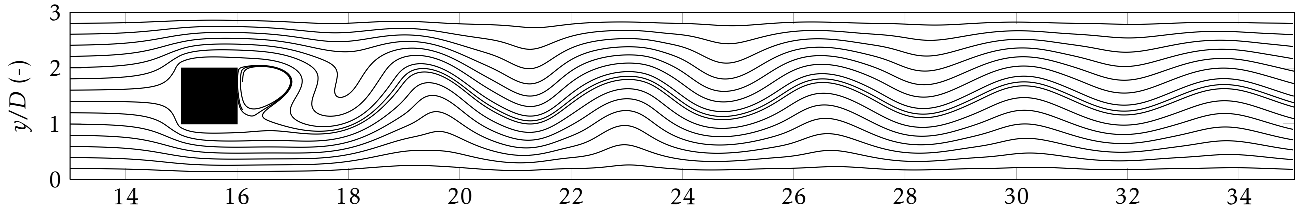
Table 2: Unsteady solution (Re = 100) results.

Quantity (Units)	Present	Kundu	% Diff.
Mean C_D (-)	2.991	3.350	11%
Ampl. C_L (-)	0.678	0.770	12%
Shed Freq., f (-)	0.263	0.282	6.7%

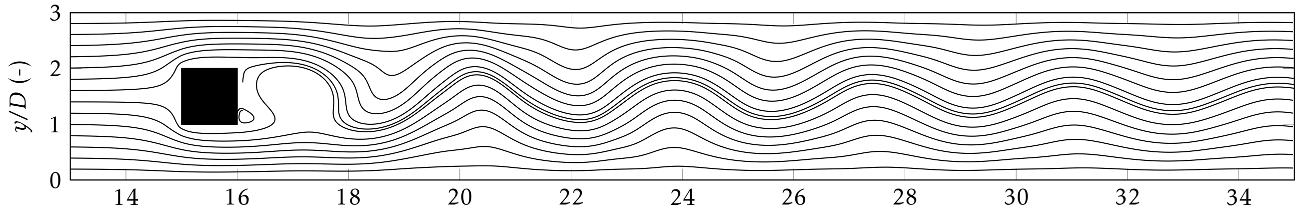
References

Dutta, S., Panigrahi, P. K., Muralidhar, K., September 2008. Experimental investigation of flow past a square cylinder at an angle of incidence. Journal of Engineering Mechanics 134 (9), 788–803.

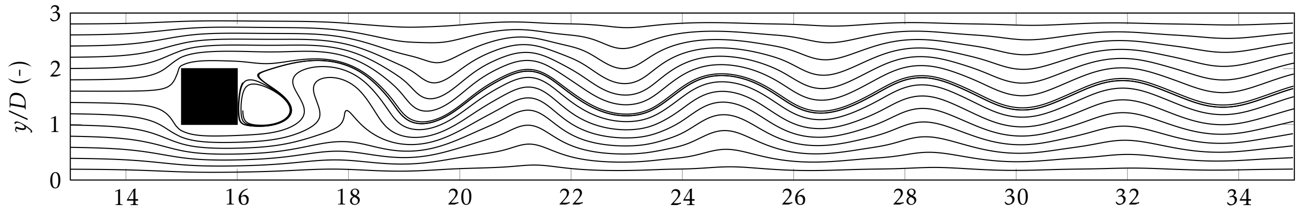
Kundu, P. K., Cohen, I. M., Dowling, D. R., 2012. Fluid Mechanics, 5th Edition. Academic Press, Waltham, MA.



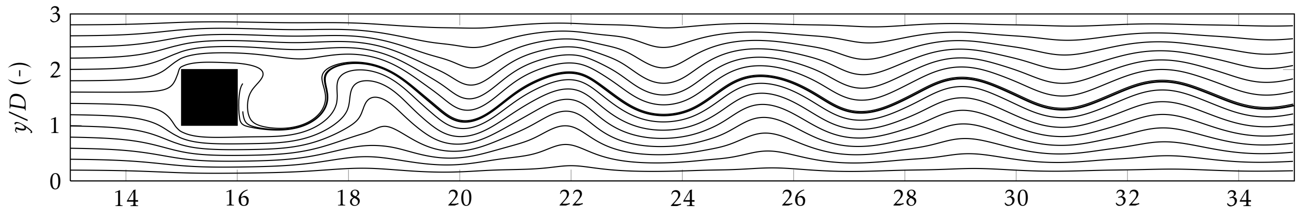
(a) $t = 39.65$



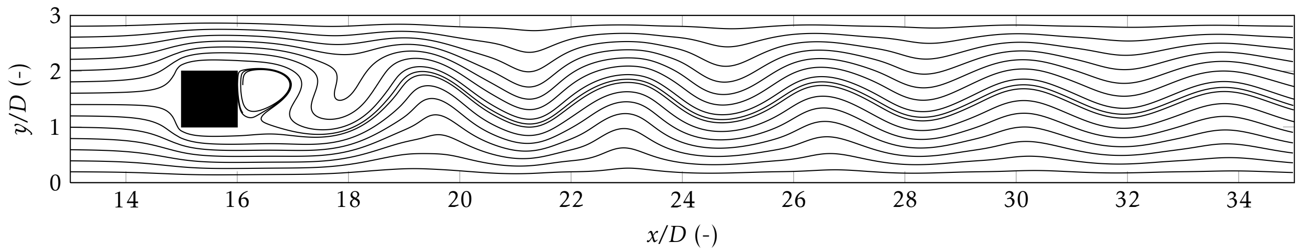
(b) $t = 40.57$



(c) $t = 41.49$



(d) $t = 42.44$



(e) $t = 43.39$

Figure 7: A sequence of streamlines for flow around a square block at Re = 100 for one full period of vortex shedding.

Appendix A. FORTRAN 90 Code Listing

Listing 1 presents the implementation of the numerical solution. This code was also submitted via email.

Listing 1: FORTRAN 90 implementation of explicit MacCormack scheme and application to square block channel flow.

```
!****************************************************************!
! File:      squareblock.f90                                     !
! Synopsis: Explicit MacCormack method                           !
!            Solution of viscid laminar flow around a square block !
!            MEEN 6310 Project 1                                 !
! Author:   Paul Gessler <paul.gessler@mu.edu>                   !
! Date:      19 December 2013                                    !
!****************************************************************!

PROGRAM SquareBlock
INTEGER istep,n,nn,ii,jj,itermax,nskip,ishow,jshow
PARAMETER (nn=2,jj=60,ii=int(jj*35.d0/3.d0),itermax=250000,nskip=10)
DOUBLE PRECISION              x(0:ii),              y(0:jj)
DOUBLE PRECISION     u(nn,0:ii,0:jj),     v(nn,0:ii,0:jj)
DOUBLE PRECISION    us( 1,0:ii,0:jj),    vs( 1,0:ii,0:jj)
DOUBLE PRECISION   rho(nn,0:ii,0:jj),  rhos(nn,0:ii,0:jj)
DOUBLE PRECISION  rhou(nn,0:ii,0:jj),  rhov(nn,0:ii,0:jj)
DOUBLE PRECISION rhous( 1,0:ii,0:jj), rhovs( 1,0:ii,0:jj)
DOUBLE PRECISION cdcl(3,0:int(itermax/nskip)), t(0:itermax)
DOUBLE PRECISION UU,rho0,VV,Mach,Masq,Re,dx,dy,dt,D,resid,Cd,Cl
DOUBLE PRECISION a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11

Mach = 0.05d0
Masq = Mach**2
UU   = 1.00d0
VV   = 0.00d0
Re   = 20.0d0
D    = 1.00d0
rho0 = 1.00d0
dx   = 35.0d0*D/dfloat(ii)
dy   = 3.00d0*D/dfloat(jj)
dt   = 0.90d0*Mach*dx/dsqrt(2.d0)

ishow = -1
jshow = -1

print *,'Reynolds␣number:␣',Re
print *,'Time␣step,␣dt:␣␣␣',dt

do i=0,ii
  x(i) = dx*dfloat(i)
enddo
do j=0,jj
  y(j) = dy*dfloat(j)
enddo
open(unit=31,file='x.dat',form='formatted',status='unknown')
open(unit=32,file='y.dat',form='formatted',status='unknown')
write(31,'(E12.4)') (x(i), i=0,ii)
write(32,'(E12.4)') (y(j), j=0,jj)
close(31)
close(32)

a1  = dt/dx
a2  = dt/dy
a3  = dt/(dx*Masq)
a4  = dt/(dy*Masq)
a5  = 4.d0*dt/(3.d0*Re*dx**2)
a6  = dt/(Re*dy**2)
a7  = dt/(Re*dx**2)
a8  = 4.d0*dt/(3.d0*Re*dy**2)
a9  = dt/(12.d0*Re*dx*dy)
a10 = 2.d0*(a5+a6)
a11 = 2.d0*(a7+a8)

! initialize
do i=0,ii
```

```
do j=0,jj
  n = 1
  if ( x(i) .ge. 15.d0*D .and. x(i) .le. 16.d0*D .and. &
       y(j) .ge.  1.d0*D .and. y(j) .le.  2.d0*D ) then
    u(n,i,j) = 0.0d0
    v(n,i,j) = 0.0d0
  else
    u(n,i,j) = UU
    v(n,i,j) = VV
  endif
    rho(n,i,j) = rho0
   rhos(n,i,j) = rho0
   rhou(n,i,j) = rho(n,i,j)*u(n,i,j)
   rhov(n,i,j) = rho(n,i,j)*v(n,i,j)
  rhous(n,i,j) = rhou(n,i,j)
  rhovs(n,i,j) = rhov(n,i,j)
enddo
enddo

do istep=0,itermax
  n = 1
  t(istep) = istep*dt

  ! step 1 done below (update solution)

  ! step 2
  do i=1,ii-1
  do j=1,jj-1
    rhos(n,i,j) = rho(n,i,j) - a1*(rhou(n,i+1,  j) - rhou(n,i,j)) &
                             - a2*(rhov(n,  i,j+1) - rhov(n,i,j))
    if ( x(i) .ge. 15.d0*D .and. x(i) .le. 16.d0*D .and. &
         y(j) .ge.  1.d0*D .and. y(j) .le.  2.d0*D ) then
      rhous(n,i,j) = 0.0d0
      rhovs(n,i,j) = 0.0d0
    else
      rhous(n,i,j) = rhou(n,i,j) - a3*(rho(n,i+1,j) - rho(n,i,j)) &
       - a1*(rho(n,i+1,j)*u(n,i+1,j)**2 - rho(n,i,j)*u(n,i,j)**2) &
       - a2*(rho(n,i,j+1)*u(n,i,j+1)*v(n,i,j+1)  &
           - rho(n,i,  j)*u(n,i,  j)*v(n,i,  j)) &
       - a10*u(n,i,j) + a5*(u(n,i+1,  j) + u(n,i-1,  j)) &
                      + a6*(u(n,  i,j+1) + u(n,  i,j-1)) &
       + a9*(v(n,i+1,j+1) + v(n,i-1,j-1) - v(n,i+1,j-1) - v(n,i-1,j+1))
      rhovs(n,i,j) = rhov(n,i,j) - a4*(rho(n,i,j+1) - rho(n,i,j)) &
       - a2*(rho(n,i,j+1)*v(n,i,j+1)**2 - rho(n,i,j)*v(n,i,j)**2) &
       - a1*(rho(n,i+1,j)*u(n,i+1,j)*v(n,i+1,j)  &
           - rho(n,i,  j)*u(n,i,  j)*v(n,i,  j)) &
       - a11*v(n,i,j) + a7*(v(n,i+1,  j) + v(n,i-1,  j)) &
                      + a8*(v(n,  i,j+1) + v(n,  i,j-1)) &
       + a9*(u(n,i+1,j+1) + u(n,i-1,j-1) - u(n,i+1,j-1) - u(n,i-1,j+1))
    endif
  enddo
  enddo

  ! step 3 - impose boundary conditions
  do i=1,ii ! since rho(n,i-1,j) is not defined at i=0
    j = 0 ! bottom
    rhos(n,i,j) = rho(n,i,j) &
     - dt*(4.d0*rhov(n,i,j+1) - rhov(n,i,j+2) &
         - 3.d0*rhov(n,i,  j))/(2.d0*dy)
    rhous(n,i,j) = rhos(n,i,j)*UU
    rhovs(n,i,j) = 0.d0
    j = jj ! top
    rhos(n,i,j) = rho(n,i,j) &
     + dt*(4.d0*rhov(n,i,j-1) - rhov(n,i,j-2) &
         - 3.d0*rhov(n,i,  j))/(2.d0*dy)
    rhous(n,i,j) = rhos(n,i,j)*UU
    rhovs(n,i,j) = 0.d0
    if (x(i) .ge. 15.d0*D .and. x(i) .le. 16.d0*D) then
      j = int(2.d0*D/dy) ! top of block
      rhos(n,i,j) = (4.d0*rho(n,i,j+1) - rho(n,i,j+2))/3.d0 &
       - 8.d0*Masq*(-5.d0*v(n,i,j+1) + 4.d0*v(n,i,j+2) - v(n,i,j+3)) &
```

```
     /(9.d0*dy*Re) &
     - Masq*(-(u(n,i+1,j+2) - u(n,i-1,j+2)) + 4.d0*(u(n,i+1,j+1) &
     - u(n,i-1,j+1)) - 3.d0*(u(n,i+1,j) - u(n,i-1,j)))/(18.d0*dx*Re)
    j = int(1.d0*D/dy) ! bottom of block
    rhos(n,i,j) = (4.d0*rho(n,i,j-1) - rho(n,i,j-2))/3.d0 &
     + 8.d0*Masq*(-5.d0*v(n,i,j-1) + 4.d0*v(n,i,j-2) - v(n,i,j-3)) &
     /(9.d0*dy*Re) &
     - Masq*(-(u(n,i+1,j-2) - u(n,i-1,j-2)) + 4.d0*(u(n,i+1,j-1) &
     - u(n,i-1,j-1)) - 3.d0*(u(n,i+1,j) - u(n,i-1,j)))/(18.d0*dx*Re)
  endif
enddo
do j=0,jj
  i = 0 ! left
  rhos(n,i,j) = rho0
  rhous(n,i,j) = rhos(n,i,j)*UU
  rhovs(n,i,j) = rhos(n,i,j)*VV
  i = ii ! right
  rhos(n,i,j) = rho(n,i,j) &
   + dt*(4.d0*rhou(n,i-1,j) - rhou(n,i-2,j) &
      - 3.d0*rhou(n,  i,j))/(2.d0*dx)
  rhous(n,i,j) = (4.d0*rhous(n,i-1,j) - rhous(n,i-2,j))/3.d0
  rhovs(n,i,j) = (4.d0*rhovs(n,i-1,j) - rhovs(n,i-2,j))/3.d0
  if (y(j) .ge. 1.d0*D .and. y(j) .le. 2.d0*D) then
    i = int(15.d0*D/dx) ! front
    rhos(n,i,j) = (4.d0*rho(n,i-1,j) - rho(n,i-2,j))/3.d0 &
     + 8.d0*Masq*(-5.d0*u(n,i-1,j) + 4.d0*u(n,i-2,j) - u(n,i-3,j)) &
     /(9.d0*dx*Re) - Masq*(4.d0*(v(n,i-1,j+1) - v(n,i-1,j-1)) &
     - (v(n,i-2,j+1) - v(n,i-2,j-1)) - 3.d0*(v(n,i,j+1) - v(n,i,j-1))) &
     /(18.d0*dy*Re)
    if (j .eq. int(1.d0*D/dy)) then ! bottom front corner
      rhos(n,i,j) = (rhos(n,i,j) + (4.d0*rho(n,i,j-1) - rho(n,i,j-2))/3.d0 &
       + 8.d0*Masq*(-5.d0*v(n,i,j-1) + 4.d0*v(n,i,j-2) - v(n,i,j-3)) &
       /(9.d0*dy*Re) &
       - Masq*(-(u(n,i+1,j-2) - u(n,i-1,j-2)) + 4.d0*(u(n,i+1,j-1) &
       - u(n,i-1,j-1)) - 3.d0*(u(n,i+1,j) - u(n,i-1,j)))/(18.d0*dx*Re))/2.d0
    elseif (j .eq. int(2.d0*D/dy)) then ! top front corner
      rhos(n,i,j) = (rhos(n,i,j) + (4.d0*rho(n,i,j+1) - rho(n,i,j+2))/3.d0 &
       - 8.d0*Masq*(-5.d0*v(n,i,j+1) + 4.d0*v(n,i,j+2) - v(n,i,j+3)) &
       /(9.d0*dy*Re) &
       - Masq*(-(u(n,i+1,j+2) - u(n,i-1,j+2)) + 4.d0*(u(n,i+1,j+1) &
       - u(n,i-1,j+1)) - 3.d0*(u(n,i+1,j) - u(n,i-1,j)))/(18.d0*dx*Re))/2.d0
    endif
    i = int(16.d0*D/dx) ! back
    rhos(n,i,j) = (4.d0*rho(n,i+1,j) - rho(n,i+2,j))/3.d0 &
     - 8.d0*Masq*(-5.d0*u(n,i+1,j) + 4.d0*u(n,i+2,j) - u(n,i+3,j)) &
     /(9.d0*dx*Re) - Masq*(4.d0*(v(n,i+1,j+1) - v(n,i+1,j-1)) &
     - (v(n,i+2,j+1) - v(n,i+2,j-1)) - 3.d0*(v(n,i,j+1) - v(n,i,j-1))) &
     /(18.d0*dy*Re)
  endif
enddo

! step 4
do i=0,ii
do j=0,jj
  us(n,i,j) = rhous(n,i,j)/rhos(n,i,j)
  vs(n,i,j) = rhovs(n,i,j)/rhos(n,i,j)
enddo
enddo

! step 5
do i=1,ii-1
do j=1,jj-1
  rho(n+1,i,j) = ((rho(n,i,j) + rhos(n,i,j)) &
   - a1*(rhous(n,i,j) - rhous(n,i-1,  j)) &
   - a2*(rhovs(n,i,j) - rhovs(n,  i,j-1)))/2.d0
  if ( x(i) .ge. 15.d0*D .and. x(i) .le. 16.d0*D .and. &
       y(j) .ge.  1.d0*D .and. y(j) .le.  2.d0*D ) then
    rhou(n+1,i,j) = 0
    rhov(n+1,i,j) = 0
  else
    rhou(n+1,i,j) = (rhou(n,i,j) + rhous(n,i,j) &
```

```fortran
         - a3*(rhos(n,i,j) - rhos(n,i-1,j)) &
         - a1*(rhos(n,  i,j)*us(n,  i,j)**2 &
             - rhos(n,i-1,j)*us(n,i-1,j)**2) &
         - a2*(rhos(n,i,  j)*us(n,i,  j)*vs(n,i,  j) &
             - rhos(n,i,j-1)*us(n,i,j-1)*vs(n,i,j-1)) &
         - a10*us(n,i,j) + a5*(us(n,i+1,  j) + us(n,i-1,  j)) &
                         + a6*(us(n,  i,j+1) + us(n,  i,j-1)) &
         + a9*(vs(n,i+1,j+1) + vs(n,i-1,j-1) &
             - vs(n,i+1,j-1) - vs(n,i-1,j+1)))/2.d0
         rhov(n+1,i,j) = (rhov(n,i,j) + rhovs(n,i,j) &
          - a4*(rhos(n,i,j) - rhos(n,i,j-1)) &
          - a1*(rhos(n,  i,j)*us(n,  i,j)*vs(n,  i,j)  &
              - rhos(n,i-1,j)*us(n,i-1,j)*vs(n,i-1,j)) &
          - a2*(rhos(n,i,  j)*vs(n,i,  j)**2 &
              - rhos(n,i,j-1)*vs(n,i,j-1)**2) &
          - a11*vs(n,i,j) + a7*(vs(n,i+1,  j) + vs(n,i-1,  j)) &
                          + a8*(vs(n,  i,j+1) + vs(n,  i,j-1)) &
          + a9*(us(n,i+1,j+1) + us(n,i-1,j-1) &
              - us(n,i+1,j-1) - us(n,i-1,j+1)))/2.d0
      endif
    enddo
    enddo

    ! step 6 - impose boundary conditions
    do j = 0,jj
      i = 0 ! left
      rho(n+1,i,j) = (rho(n,i,j) + rhos(n,i,j) &
       - dt*(4.d0*rhous(n,i+1,j) - rhous(n,i+2,j) &
           - 3.d0*rhous(n,  i,j))/(2.d0*dx))/2.d0
      rhou(n+1,i,j) = rho(n+1,i,j)*UU
      rhov(n+1,i,j) = rho(n+1,i,j)*VV
      i = ii ! right
      rho(n+1,i,j) = (rho(n,i,j) + rhos(n,i,j) &
       + dt*(4.d0*rhous(n,i-1,j) - rhous(n,i-2,j) &
           - 3.d0*rhous(n,i,j))/(2.d0*dx))/2.d0
      rhou(n+1,i,j) = (4.d0*rhou(n+1,i-1,j) - rhou(n+1,i-2,j))/3.d0
      rhov(n+1,i,j) = (4.d0*rhov(n+1,i-1,j) - rhov(n+1,i-2,j))/3.d0
      if (y(j) .ge. 1.d0*D .and. y(j) .le. 2.d0*D) then
        i = int(15.d0*D/dx) ! front
        rho(n+1,i,j) = (rhos(n,i,j) &
         + (4.d0*rhos(n,i-1,j) - rhos(n,i-2,j))/3.d0 &
         + 8.d0*Masq*(-5.d0*us(n,i-1,j) + 4.d0*us(n,i-2,j) &
                          - us(n,i-3,j))/(9.d0*dx*Re) &
         - Masq*(4.d0*(vs(n,i-1,j+1) - vs(n,i-1,j-1)) &
                     - (vs(n,i-2,j+1) - vs(n,i-2,j-1)) &
                - 3.d0*(vs(n,  i,j+1) - vs(n,  i,j-1)))/(18.d0*dy*Re))/2.d0
        i = int(16.d0*D/dx) ! back
        rho(n+1,i,j) = (rhos(n,i,j) &
         + (4.d0*rhos(n,i+1,j) - rhos(n,i+2,j))/3.d0 &
         - 8.d0*Masq*(-5.d0*us(n,i+1,j) + 4.d0*us(n,i+2,j) &
                          - us(n,i+3,j))/(9.d0*dx*Re) &
         - Masq*(4.d0*(vs(n,i+1,j+1) - vs(n,i+1,j-1)) &
                     - (vs(n,i+2,j+1) - vs(n,i+2,j-1)) &
                - 3.d0*(vs(n,  i,j+1) - vs(n,  i,j-1)))/(18.d0*dy*Re))/2.d0
      endif
    enddo
    do i = 1,ii-1
      j = 0 ! bottom
      rho(n+1,i,j) = (rho(n,i,j) + rhos(n,i,j) &
       - dt*(4.d0*rhovs(n,i,j+1) - rhovs(n,i,j+2) &
           - 3.d0*rhovs(n,i,  j))/(2.d0*dy))/2.d0
      rhou(n+1,i,j) = rho(n+1,i,j)*UU
      rhov(n+1,i,j) = rho(n+1,i,j)*0.d0
      j = jj ! top
      rho(n+1,i,j) = (rho(n,i,j) + rhos(n,i,j) &
       + dt*(4.d0*rhovs(n,i,j-1) - rhovs(n,i,j-2) &
           - 3.d0*rhovs(n,i,  j))/(2.d0*dy))/2.d0
      rhou(n+1,i,j) = rho(n+1,i,j)*UU
      rhov(n+1,i,j) = rho(n+1,i,j)*0.d0
      if (x(i) .ge. 15.d0*D .and. x(i) .le. 16.d0*D) then
        j = int(2.d0*D/dy) ! top of block
```

```
      rho(n+1,i,j) = (rhos(n,i,j) &
       + (4.d0*rhos(n,i,j+1) - rhos(n,i,j+2))/3.d0 &
       - 8.d0*Masq*(-5.d0*vs(n,i,j+1) + 4.d0*vs(n,i,j+2) &
                         - vs(n,i,j+3))/(9.d0*dy*Re) &
       - Masq*(-(us(n,i+1,j+2) - us(n,i-1,j+2)) &
         + 4.d0*(us(n,i+1,j+1) - us(n,i-1,j+1)) &
         - 3.d0*(us(n,i+1,  j) - us(n,i-1,  j)))/(18.d0*dx*Re))/2.d0
      j = int(1.d0*D/dy) ! bottom of block
      rho(n+1,i,j) = (rhos(n,i,j) &
       + (4.d0*rhos(n,i,j-1) - rhos(n,i,j-2))/3.d0 &
       + 8.d0*Masq*(-5.d0*vs(n,i,j-1) + 4.d0*vs(n,i,j-2) &
                         - vs(n,i,j-3))/(9.d0*dy*Re) &
       - Masq*(-(us(n,i+1,j-2) - us(n,i-1,j-2)) &
         + 4.d0*(us(n,i+1,j-1) - us(n,i-1,j-1)) &
         - 3.d0*(us(n,i+1,  j) - us(n,i-1,  j)))/(18.d0*dx*Re))/2.d0
    endif
  enddo

  ! update solution
  resid = 0.d0
  Cd = 0.d0
  Cl = 0.d0
  do i = 0,ii
  do j = 0,jj
    ! debug output
    if (j .eq. jshow) then
      print '("(n,i,j):␣(",I9.9,",",I4.4,",",I4.4,")␣&
␣␣␣␣␣␣␣&rho:␣",F12.8,"␣u:␣",F12.8,"␣v:␣",F12.8)', &
       istep,i,j,rho(n,i,j),u(n,i,j),v(n,i,j)
    endif
    if (i .eq. ishow) then
      print '("(n,i,j):␣(",I9.9,",",I4.4,",",I4.4,")␣&
␣␣␣␣␣␣␣&rho:␣",F12.8,"␣u:␣",F12.8,"␣v:␣",F12.8)', &
       istep,i,j,rho(n,i,j),u(n,i,j),v(n,i,j)
    endif

    ! move to next timestep
          resid = resid + dabs(rho(n,i,j) - rho(n+1,i,j))
     rho(n,i,j) =  rho(n+1,i,j)
    rhou(n,i,j) = rhou(n+1,i,j)
    rhov(n,i,j) = rhov(n+1,i,j)
       u(n,i,j) = rhou(n+1,i,j)/rho(n+1,i,j)
       v(n,i,j) = rhov(n+1,i,j)/rho(n+1,i,j)

    ! compute drag and lift coefficients
    if (i .eq. int(15.d0*D/dx) .and. &
        y(j) .ge.  1.d0*D .and. y(j) < 2.d0*D) then
      Cd = Cd + ((rho(n,i,j) + rho(n,i,j+1))*dy/Masq/2.d0)/(rho0*UU**2*D)
      Cl = Cl - (dy/(2.d0*dx*Re*UU))* &
        (-v(n,i,j) + v(n,i-1,j) - v(n,i,j+1) + v(n,i-1,j+1))
    endif
    if (i .eq. int(16.d0*D/dx) .and. &
        y(j) .ge.  1.d0*D .and. y(j) < 2.d0*D) then
      Cd = Cd - ((rho(n,i,j) + rho(n,i,j+1))*dy/Masq/2.d0)/(rho0*UU**2*D)
      Cl = Cl + (dy/(2.d0*dx*Re*UU))* &
        (-v(n,i,j) + v(n,i+1,j) - v(n,i,j+1) + v(n,i+1,j+1))
    endif
    if (j .eq.  int(1.d0*D/dy) .and. &
        x(i) .ge. 15.d0*D .and. x(i) < 16.d0*D) then
      Cl = Cl + ((rho(n,i,j) + rho(n,i+1,j))*dy/Masq/2.d0)/(rho0*UU**2*D)
      Cd = Cd - (dx/(2.d0*dy*Re*UU))* &
        (-u(n,i,j) + u(n,i,j-1) - u(n,i+1,j) + u(n,i+1,j-1))
    endif
    if (j .eq.  int(2.d0*D/dy) .and. &
        x(i) .ge. 15.d0*D .and. x(i) < 16.d0*D) then
      Cl = Cl - ((rho(n,i,j) + rho(n,i+1,j))*dy/Masq/2.d0)/(rho0*UU**2*D)
      Cd = Cd + (dx/(2.d0*dy*Re*UU))* &
        (-u(n,i,j) + u(n,i,j+1) - u(n,i+1,j) + u(n,i+1,j+1))
    endif
  enddo
  enddo
```

```

  ! output solution
  if (MOD(istep,nskip) .eq. 0) then
    print '("␣Iter:␣",I9.9,"␣␣Time:␣",F12.8,␣&
␣␣␣␣␣&␣"␣␣Res:␣",E16.9E3,"␣␣Cd:␣",F12.8,"␣␣Cl:␣",F11.8)', &
     istep,t(istep),resid,Cd,Cl
    cdcl(1,int(istep/nskip)) = t(istep)
    cdcl(2,int(istep/nskip)) = Cd
    cdcl(3,int(istep/nskip)) = Cl
  endif
enddo ! istep loop

! output
open(unit=34,file=   'u.dat',form='formatted',status='unknown')
open(unit=35,file=   'v.dat',form='formatted',status='unknown')
open(unit=36,file= 'rho.dat',form='formatted',status='unknown')
open(unit=37,file='cdcl.dat',form='formatted',status='unknown')
do j=0,jj
  write(34,1)  (  u(n,i,j),i=0,ii)
  write(35,1)  (  v(n,i,j),i=0,ii)
  write(36,1)  (rho(n,i,j),i=0,ii)
enddo
do istep=0,int(itermax/nskip)
  write(37,1)  (cdcl(n,istep),n=1,3)
enddo
close(34)
close(35)
close(36)
close(37)

1 Format(2400001E18.8E3)
END
```