```
      program ABCflow
c
c ###################################################
c # This program computes particle trajectories in a  #
c # 2D cellular flow field for different particle-to- #
c # fluid density ratios. For further references see  #
c # Maxey, M.R. (1987), Phys. Fluids, 30, 1915-1928.  #
c # The domain sizes are Lx = 4 pi and Ly = 2 pi      #
c ###################################################
c
      implicit real*8(a-h,o-z)
      include 'param.h'
c
c Declaration of variables
c
      COMMON /DATA/npart,STEPS,OLDSTEP,iflag,R,A,W,dump
      COMMON /PART/xdata,ydata,vx,vy,x,y,vvx,vvy,dt

      character (len=80) input,output,output2,restart
      character (len=4) instant

      integer seed,seed1
      integer ISTEP,OLDSTEP
      integer ii
      integer iflag

      real ran

      dimension xdata(nmax),ydata(nmax)
      dimension vx(nmax),vy(nmax)
      dimension vvx(nmax),vvy(nmax),x(nmax),y(nmax)

      real*8 dt,save
c
c Reads input data files from "file.inp"
c
      open(unit=4,file='file.inp',form='formatted',status='OLD')
      read(4,1100)output
      read(4,1100)output2
      read(4,1100)restart
      read(4,1100)input

 1100 format(a80)

c
c Reads input data from "input"
c
      call INPUTDATA(input)
c
      ISTEP=1

c
c      print *,iflag
c
c Sets the integration time-step
c equal to 1/10 of the particle
c relaxation time.
c
      dt = A/10.

c
c Starts time loop (STEPS is
c the number of time-steps)
c
 1    CONTINUE

c
c If the simulation is new, then performs
c particle initialization (at ISTEP=1).
```



```
c Otherwise, it is a restart!
c
      seed=ISTEP

c
c-----*
      if(ISTEP.eq.1)then
c-----*
c
c------*
       if(iflag.eq.1)then
c------*
c
        write (*,*) 'NEW SIMULATION!'
c
c
c 1. Initializes particle position:
c    particles are injected in the
c    flow with  random position.
c
       do i=1,npart
        xdata(i)=(ran(seed))*xmax
         ydata(i)=(ran(seed))*ymax
       enddo
c
       open(unit=1000,file='ISTEP',form='formatted')
       write(1000,91) 0
       close(1000,status='KEEP')

       open(unit=1000,file='ISTEP',form='formatted')
       read(1000,90)instant
       close(1000,status='DELETE')
c
c Opens the particle position data file (pospar).
c This file is opened and saved at the dumping
c frequency given in "input".
c
       open(unit=12,file=TRIM(output)//'_'//
     &  instant,status='NEW')
c
c Opens the particle velocity data file (velpar).
c This file is opened and saved at the dumping
c frequency given in "input".
c
       open(unit=14,file=TRIM(output2)//'_'//
     &  instant,status='NEW')

       do i=1,npart
         write(12,*)x(i),y(i)
         write(14,*)vvx(i),vvy(i)
       enddo

       close(12)
       close(14)

c
c 2. Initializes particle velocity:
c    particles are injected in the
c    flow with velocity equal to the
c    velocity of the fluid at particle
c    position.
c
         call INITVEL

c
c------*
       else
```

```
c------*
c
        write (*,*) 'RESTART SIMULATION!'

        open(unit=15,file=TRIM(restart),status='unknown')
        call RDDATA
c
c------*
       endif
c------*
c-----*
       endif
c-----*
c

       open(unit=1000,file='ISTEP',form='formatted')
       write(1000,91)OLDSTEP+ISTEP
       close(1000,status='KEEP')

       open(unit=1000,file='ISTEP',form='formatted')
       read(1000,90)instant
       close(1000,status='DELETE')

c
c-----*
       if(mod(dfloat(ISTEP),dump).eq.0.)then
c-----*
c
c Opens the particle position data file (pospar).
c This file is opened and saved at the dumping
c frequency given in "input".
c
        open(unit=12,file=TRIM(output)//'_'//
     &  instant,status='NEW')
c
c Opens the particle velocity data file (velpar).
c This file is opened and saved at the dumping
c frequency given in "input".
c
        open(unit=14,file=TRIM(output2)//'_'//
     &  instant,status='NEW')
c
c-----*
       endif
c-----*
c
c Opens restart file at the end of the simulation
c
       if (ISTEP.eq.STEPS) then
       open(unit=15,file=TRIM(restart),status='unknown')
       write (15,*) OLDSTEP+int(STEPS)
       endif
c
c Loops over the particles during the
c current integration time-step.
c
       do i=1,npart
       ii=i
       call INTEGRPART(ii)
       enddo
c
c End of particle loop.
c
c=======================================================
c
c Now writes the output files and sets the
c restart file.
c
```



```
      call SETOUTPUT(ISTEP)

c
c Definition of writing formats
c
 90   format(a4)
 91   format(I4.4)

c
c Closes data files
c
      close(12,status='KEEP')
      close(14,status='KEEP')
      close(15,status='KEEP')

c
c The time-step is incremented.
c
      ISTEP=ISTEP+1

      if(ISTEP.gt.STEPS)then
       write (*,*) 'Time-Step',OLDSTEP+int(ISTEP)-1,
     &             ' of',OLDSTEP+int(STEPS)
       write (*,*) 'SIMULATION DONE!'
       goto 2
      else
       write (*,*) 'Time-Step',OLDSTEP+int(ISTEP)-1,
     &             ' of',OLDSTEP+int(STEPS)
       goto 1
      endif

c
c If the program jumps here, then the
c simulation is finished!
c
 2    CONTINUE

      STOP
      END

ccccccccccccccccccccccccccccccccccccccc
c        END OF MAIN PROGRAM          c
ccccccccccccccccccccccccccccccccccccccc

      SUBROUTINE INPUTDATA(in)
c     ------------------------
c
c This subroutine reads the input data
c

      implicit real*8(a-h,o-z)

      COMMON /DATA/npart,STEPS,OLDSTEP,iflag,R,A,W,dump

      character (len=80) in

      integer OLDSTEP

      open(unit=11,file=in,status='unknown')

      do k=1,3
      read(11,*)
      enddo

c
c Number of particles
c
      read(11,*)npart
```

```
c
c Number of time-steps
c
      read(11,*)STEPS
c
c Flag for random start position
c (new simulation for iflag=1,
c restart simulation otherwise)
c
      read(11,*) iflag
c
c Mass ratio, R (Eq. 13 in Maxey, PoF)
c
      read(11,*) R
c
c Still fluid settling velocity, W (Eq. 15 in Maxey, PoF)
c
      read(11,*) W
c
c Inertia parameter, A (Eq. 5 in Maxey, PoF)
c
      read(11,*) A
c
c Fields dumping frequency
c
      read(11,*) dump

      RETURN
      END

ccccccccccccccccccccccccccccccccccccc

      SUBROUTINE INITVEL
c     ------------------
c
c This subroutine initializes particle velocity.
c

      implicit real*8(a-h,o-z)
      include 'param.h'

      COMMON /DATA/npart,STEPS,OLDSTEP,iflag,R,A,W,dump
      COMMON /PART/xdata,ydata,vx,vy,x,y,vvx,vvy,dt

      dimension xdata(nmax),ydata(nmax)
      dimension vx(nmax),vy(nmax)
      dimension vvx(nmax),vvy(nmax),x(nmax),y(nmax)

      real*8 dt
      real ran

      integer seed
      integer OLDSTEP
      integer STEPS


c
c Solves for Eq. 2 in Maxey, PoF (1987).
c Note that Uo=1 in the present case.
c
       do i=1,npart
        vx(i)=sin(xdata(i))*cos(ydata(i))
        vy(i)=-cos(xdata(i))*sin(ydata(i))
       enddo

      RETURN
      END

ccccccccccccccccccccccccccccccccccccc
```



```
      SUBROUTINE RDDATA
c     -----------------
c
c This subroutine reads particle position/velocity
c from file restart for restart simulations.
c

      implicit real*8(a-h,o-z)
      include 'param.h'

      COMMON /DATA/npart,STEPS,OLDSTEP,iflag,R,A,W,dump
      COMMON /PART/xdata,ydata,vx,vy,x,y,vvx,vvy,dt

      dimension xdata(nmax),ydata(nmax)
      dimension vx(nmax),vy(nmax)
      dimension vvx(nmax),vvy(nmax),x(nmax),y(nmax)

      integer OLDSTEP

      real*8 dt

      read(15,*)OLDSTEP
      do i=1,npart
       read(15,*)xdata(i),ydata(i),vx(i),vy(i)
      enddo

      close(15,status='DELETE')

      RETURN
      END

ccccccccccccccccccccccccccccccccccccc

      SUBROUTINE SETOUTPUT(ITIME)
c     ---------------------------
c
c This subroutine writes output data files
c at the requested dump frequency and saves
c particle position/velocity for subsequent
c restart simulations.
c

      implicit real*8(a-h,o-z)
      include 'param.h'

      COMMON /DATA/npart,STEPS,OLDSTEP,iflag,R,A,W,dump
      COMMON /PART/xdata,ydata,vx,vy,x,y,vvx,vvy,dt

      dimension xdata(nmax),ydata(nmax)
      dimension vx(nmax),vy(nmax)
      dimension vvx(nmax),vvy(nmax),x(nmax),y(nmax)

      real*8 dt

      integer ITIME
      integer OLDSTEP
c
c Writes pospar and velpar at the requested
c dump frequency.
c
      if(mod(dfloat(ITIME),dump).eq.0)then
       do i=1,npart
        write(12,*)x(i),y(i)
        write(14,*)vvx(i),vvy(i)
       enddo
      endif
c
c Writes restart at the end of the simulation
c for subsequent restarts.
```

```
c
      if(int(ITIME).eq.int(STEPS))then
c       write(15,*)OLDSTEP+STEPS
       do i=1,npart
        write(15,107)x(i),y(i),vvx(i),vvy(i)
       enddo
      endif

 107  format(4(1x,f15.9))

      RETURN
      END

ccccccccccccccccccccccccccccccccccccccc

      SUBROUTINE INTEGRPART(J)
c     ------------------------
c
c This subroutine integrates the particle motion
c equation using a 4th-order Runge-Kutta scheme.
c

      implicit real*8(a-h,o-z)
      include 'param.h'

      COMMON /DATA/npart,STEPS,OLDSTEP,iflag,R,A,W,dump
      COMMON /PART/xdata,ydata,vx,vy,x,y,vvx,vvy,dt

      dimension xdata(nmax),ydata(nmax)
      dimension vx(nmax),vy(nmax)
      dimension vvx(nmax),vvy(nmax),x(nmax),y(nmax)

      real*8 dt,ux,duxdy
      real ran

      integer J,J1
      integer OLDSTEP

      dimension der(4),YYY(4),YOUT(4)

c
c Initializes vectors to be used in RK4
c
      do k=1,4
       der(k)=0.
       YYY(k)=0.
       YOUT(k)=0.
      enddo

      J1=J

c
c Starts the time-integration routine (see Eqs. 2,
c 25a and 25b in Maxey, Phys. Fluids, 1987).
c
       der(1)=vx(J)
       der(2)=vy(J)
       der(3)=A*(-vx(J)+sin(xdata(J))*cos(ydata(J)) +
     &        0.5*R/A*(vx(J)*cos(xdata(J))*cos(ydata(J)) -
     &                 vy(J)*sin(xdata(J))*sin(ydata(J))) +
     &                 R/A*sin(xdata(J))*cos(xdata(J)))
       der(4)=A*(-vy(J)-cos(xdata(J))*sin(ydata(J)) + W +
     &        0.5*R/A*(vx(J)*sin(xdata(J))*sin(ydata(J)) -
     &                 vy(J)*cos(xdata(J))*cos(ydata(J))) +
     &                 R/A*sin(ydata(J))*cos(ydata(J)))

c
c Input values for particle position (xdata and ydata)
c and for particle velocity (vx and vy) respectively.
```



```
c
       YYY(1)=xdata(J)
       YYY(2)=ydata(J)
       YYY(3)=vx(J)
       YYY(4)=vy(J)

       call rk4(YYY,der,4,dt,YOUT,ux,duxdy,J)

c
c Output values for particle position (x and y) and
c for particle velocity (vvx and vvy) respectively.
c
       x(J)  =YOUT(1)
       y(J)  =YOUT(2)
       vvx(J)=YOUT(3)
       vvy(J)=YOUT(4)

c
c Checks if the new particle position is inside the domain
c and checks for particles sticking to the wall.
c
       if(y(J).le.0.)then
        temp=y(J)
        y(J)=ymax+temp
       endif

       if(y(J).gt.ymax)then
        temp=0.
        temp=y(J)
        y(J)=temp - ymax
       endif

       if(x(J).gt.xmax)then
        temp=0.
        temp=x(J)-xmax
        x(J)=temp
       endif

       if(x(J).lt.0.)then
        temp=0.
        temp=abs(x(J))
        x(J)=xmax-temp
       endif

c
c Updates particle position and velocity before
c returning to the main program.
c
       xdata(J)=x(J)
       ydata(J)=y(J)
       vx(J)=vvx(J)
       vy(J)=vvy(J)

      RETURN
      END
```

```
/newhome/cris/ABCFLOW/OUTPUT/pospar
/newhome/cris/ABCFLOW/OUTPUT/velpar
/newhome/cris/ABCFLOW/OUTPUT/restart
/newhome/cris/ABCFLOW/input
```



```
#################################################################
### INPUT FILE FOR PARTICLE TRACKING IN CELLULAR FLOW FIELD ###
#################################################################
50000       ----- number of particles tracked
50          ----- number of time-steps computed
2           ----- random generation of particle position at start
2.          ----- mass ratio R
0.5         ----- non dimensional still-fluid settling velocity W
10.         ----- inertia parameter A (equivalent to 1/St)
10.         ----- fields dumping frequency
```

```
c
c Maximum number of particles
c
      parameter(nmax=50000)
c
c Dimension of the computational domain
c
      parameter(xmin=0.,xmax=12.56637,ymin=0.,ymax=6.283185)
```



```
      real function ran(idum)
c     ----------------------
c
c Random number generation function
c
      integer idum,IA,IM,IQ,IR,MASK
      real AM
      parameter (IA=16807,IM=2147483647,AM=1./IM,
     &           IQ=127773,IR=2836,MASK=123459876)
      integer k
c
      idum=ieor(idum,MASK)
      k=idum/IQ
      idum=IA*(idum-k*IQ)-IR*k
      if (idum.lt.0) idum=idum+IM
      ran=AM*idum
      idum=ieor(idum,MASK)
c
      return
      end
```

```
      SUBROUTINE rk4(y,dydx,n,h,yout,UX,dux,JJ)
c     ----------------------------------------
c
c This subroutine performs time-integration
c of a system of first-order ODEs
c
      implicit real*8(a-h,o-z)
      INTEGER n,NMAX
      REAL*8 h,x,dydx(n),y(n),yout(n),dux,UX

      PARAMETER (NMAX=50)
      INTEGER i,JJ
      REAL h6,hh,xh,dym(NMAX),dyt(NMAX),yt(NMAX)

      hh=h*0.5
      h6=h/6.
      xh=x+hh
      do 11 i=1,n
       yt(i)=y(i)+hh*dydx(i)
 11   continue
      call DERIVS(yt,dyt,UX,dux,JJ)
      do 12 i=1,n
        yt(i)=y(i)+hh*dyt(i)
 12   continue
      call DERIVS(yt,dym,UX,dux,JJ)
      do 13 i=1,n
        yt(i)=y(i)+h*dym(i)
        dym(i)=dyt(i)+dym(i)
 13   continue
      call DERIVS(yt,dyt,UX,dux,JJ)
      do 14 i=1,n
        yout(i)=y(i)+h6*(dydx(i)+dyt(i)+2.*dym(i))
 14   continue
      return
      END

c
c  (C) Copr. 1986-92 Numerical Recipes Software 0)001.
c

ccccccccccccccccccccccccccccccccccccc

      SUBROUTINE DERIVS(YY,DERIV,UUX,DUUXDY,II)
c     ----------------------------------------

      implicit real*8(a-h,o-z)
      include 'param.h'

      COMMON /DATA/npart,STEPS,OLDSTEP,iflag,R,A,W,dump
      COMMON /PART/xdata,ydata,vx,vy,x,y,vvx,vvy,dt

      dimension xdata(nmax),ydata(nmax)
      dimension vx(nmax),vy(nmax)
      dimension vvx(nmax),vvy(nmax),x(nmax),y(nmax)

      real*8 dt,UUX,DUUXDY

      integer II
      integer OLDSTEP

      dimension DERIV(4)
      dimension YY(4)
c     write(*,*)ii

      DERIV(1)=YY(3)
      DERIV(2)=YY(4)
      DERIV(3)=A*(-YY(3)+sin(xdata(II))*cos(ydata(II)) +
     &        0.5*R/A*(YY(3)*cos(xdata(II))*cos(ydata(II)) -
     &                 YY(4)*sin(xdata(II))*sin(ydata(II))) +
```



```
     &                 R/A*sin(xdata(II))*cos(xdata(II)))
      DERIV(4)=A*(-YY(4)-cos(xdata(II))*sin(ydata(II)) + W +
     &        0.5*R/A*(YY(3)*sin(xdata(II))*sin(ydata(II)) -
     &                 YY(4)*cos(xdata(II))*cos(ydata(II))) +
     &                 R/A*sin(ydata(II))*cos(ydata(II)))

      RETURN
      END
```