```
ソースファイル名：iblbm.for,　実行ファイル名：iblbm.out
$ cp   ../common/iblbm_ori.for   .    ($ cp   ../common/iblbm_ori.for   .  )
$ cp   iblbm_ori.for   iblbm.for    ($ cp   iblbm_ori.for   iblbm.for )
$ vi   iblbm.for
$ gfortran   -o   iblbm.out   iblbm.for
$ ./ iblbm.out
```

```
      program iblbm

      real*8 f(9,40,40), f0(9,40,40), ftmp(9,40,40)
      real*8 u(40,40), v(40,40), rho(40,40)
      real*8 fx(40,40), fy(40,40), cx(9), cy(9)
      real*8 xp(100), yp(100), fxpe(100), fype(100)
      real*8 upt(100), vpt(100), upe(100), vpe(100)
      real*8 u2, tmp, t, u0, v0, rho0, p0, pi, tau
      real*8 fxp, fyp, fxc, fyc, umax, vmax
      real*8 rp, xpc, ypc, torq, rhop
      real*8 up, vp, up1, vp1, up2, vp2, mp, iip
      real*8 omega, omega1, omega2, theta
      real*8 tmp1, tmp2, tmp3, d, zeta, g
      integer i, j, k, m, nf1, nf2, in, jn, nx, ny, np
      character a(40,40), b(40,40)

      t = 0.; d = 2.; pi = 4.*atan(1.)

      nx = 31; ny = 31;

c.. discrete velocity
      cx(1) =   0.; cy(1) =   0.
      cx(2) =   1.; cy(2) =   0.; cx(3) =   0.; cy(3) =   1.;
      cx(4) = -1.; cy(4) =   0.; cx(5) =   0.; cy(5) = -1.
      cx(6) =   1.; cy(6) =   1.; cx(7) = -1.; cy(7) =   1.
      cx(8) = -1.; cy(8) = -1.; cx(9) =   1.; cy(9) = -1.

c.. initial condition
      u0   = 0.; v0   = 0.0; rho0 = 1.; p0 = rho0/3.

c.. immersed boundary
       np = 80; rhop= 2.;
       rp = 5.; mp = pi*rp**2; iip = pi*rp**4*0.5
        up = 0.; up1 = 0.; up2 = 0.;
```

```
  vp = 0.; vp1 = 0.; vp2 = 0.;
  fxp = 0.; fyp = 0.; torq = 0.;
  omega = 0.; omega1 = 0.; omega2 = 0.; theta = 0.;
  xpc = 8.; ypc = 20.; tau = 0.8; g = -0.0001

 do m = 1, np;
  xp(m) = xpc + rp*cos(2.*pi*(m - 1)/float(np))
  yp(m) = ypc + rp*sin(2.*pi*(m - 1)/float(np))
 enddo

 do m = 1, np;
  fxpe(m) = 0.; fype(m) = 0.;
   upe(m) = 0.;    vpe(m) = 0.;
 enddo

 do i = 1,nx; do j = 1,ny
  u(i,j) = 0.; v(i,j) = 0.; rho(i,j) = rho0;
 enddo; enddo

 do i = 1,nx; do j = 1,ny
  u2 = u(i,j)**2 + v(i,j)**2
  f0(1,i,j) = rho(i,j)*(1. - 3./2.*u2)*4./9.
  do k = 2,5
   tmp = cx(k)*u(i,j) + cy(k)*v(i,j)
   f0(k,i,j) = rho(i,j)*(1. + 3.*tmp + 9./2.*tmp**2 - 3./2.*u2)/9.
  end do;
  do k = 6,9
   tmp = cx(k)*u(i,j) + cy(k)*v(i,j)
   f0(k,i,j) = rho(i,j)*(1. + 3.*tmp + 9./2.*tmp**2 - 3./2.*u2)/36.
  end do;
  do k = 1,9; f(k,i,j) = f0(k,i,j); end do;
 end do; end do


c.. calculation start
 do nf1 = 1, 500; do nf2 = 1,100
   t = t +1

c..    equilibrium distribution function
 do i = 1,nx; do j = 1,ny
```

```
    u2 = u(i,j)**2 + v(i,j)**2
    f0(1,i,j) = rho(i,j)*(1. - 3./2.*u2)*4./9.
    do k = 2,5
     tmp = cx(k)*u(i,j) + cy(k)*v(i,j)
     f0(k,i,j) = rho(i,j)*(1. + 3.*tmp + 9./2.*tmp**2 - 3./2.*u2)/9.
    end do;
    do k = 6,9
     tmp = cx(k)*u(i,j) + cy(k)*v(i,j)
     f0(k,i,j) = rho(i,j)*(1. + 3.*tmp + 9./2.*tmp**2 - 3./2.*u2)/36.
    end do;
   end do; end do

c..   collision
    do i = 1,nx; do j = 1,ny; do k = 1,9;
     f(k,i,j) = f(k,i,j) - (f(k,i,j) - f0(k,i,j))/tau
    end do;end do; end do

c.. forcing
    do i = 1,nx; do j = 1,ny;
     do k = 2,5;
      f(k,i,j) =f(k,i,j)+(fx(i,j)*cx(k)+fy(i,j)*cy(k))/3.
     end do;
     do k = 6,9;
      f(k,i,j) =f(k,i,j)+(fx(i,j)*cx(k)+fy(i,j)*cy(k))/12.
     end do;
    end do; end do

c..   streaming
    do i = 1,nx; do j = 1,ny; do k = 1,9;
     ftmp(k,i,j) = f(k,i,j);
    end do; end do; end do

    do i =    1,nx-1; do j =    1,ny
     f(2,i + 1,j ) = ftmp(2,i,j);
    end do; end do
    do i =    1,nx; do j =    1,ny-1
     f(3,i ,j + 1) = ftmp(3,i,j);
    end do; end do
    do i = 2,nx; do j =    1,ny
     f(4,i - 1,j ) = ftmp(4,i,j);
```

```
        end do; end do
        do i =   1,nx; do j = 2,ny
         f(5,i ,j - 1) = ftmp(5,i,j);
        end do; end do
        do i =   1,nx-1; do j =   1,ny-1
         f(6,i + 1,j + 1) = ftmp(6,i,j);
        end do; end do
        do i = 2,nx; do j = 1,ny-1
         f(7,i - 1,j + 1) = ftmp(7,i,j);
        end do; end do
        do i = 2,nx; do j = 2,ny
         f(8,i - 1,j - 1) = ftmp(8,i,j);
        end do; end do
        do i =   1,nx-1; do j = 2,ny
         f(9,i + 1,j - 1) = ftmp(9,i,j);
        end do; end do

c.. boundary condition
        do i =   1,nx;
         f(3,i, 1) = f(5,i, 1);
         f(6,i, 1) = f(8,i, 1);
         f(7,i, 1) = f(9,i, 1);
         f(5,i,ny) = f(3,i,ny);
         f(8,i,ny) = f(6,i,ny);
         f(9,i,ny) = f(7,i,ny);
        end do;
        do j =   1,ny;
         f(2, 1,j) = f(4, 1,j);
         f(6, 1,j) = f(8, 1,j);
         f(9, 1,j) = f(7, 1,j);
         f(4,nx,j) = f(2,nx,j);
         f(8,nx,j) = f(6,nx,j);
         f(9,nx,j) = f(7,nx,j);
        end do;

c.. physics
        do i = 1,nx; do j = 1,ny
         u(i,j) =0.; v(i,j) =0.; rho(i,j) =f(1,i,j);
         do k = 2,9
            u(i,j) =     u(i,j) + f(k,i,j)*cx(k)
```

```
        v(i,j) =     v(i,j) + f(k,i,j)*cy(k)
      rho(i,j) = rho(i,j) + f(k,i,j)
    end do; end do; end do
    do i = 1,nx; do j = 1,ny
      if (rho(i,j) .ne. 0.) then
        u(i,j) = u(i,j)/rho(i,j)
        v(i,j) = v(i,j)/rho(i,j)
      else
        u(i,j)   = 0.; v(i,j)   = 0.;
      endif
    end do; end do

c.. immersed boundary condition
    do m = 1, np; fxpe(m) = 0.; fype(m) = 0.; enddo

c.. Interpolation: properties (u, v) --> (upt, vpt)
    do m = 1, np;
      upt(m) = 0.; vpt(m) = 0.;
      do i = int(xp(m)) - 3, int(xp(m)) + 3
      do j = int(yp(m)) - 3, int(yp(m)) + 3
        tmp1 = abs(xp(m) - i); tmp2 = abs(yp(m) - j)
        in = i; jn = j;
        if(i .le.    0) in = nx +   i
        if(i .gt.   nx) in = i   - nx
        if(j .le.    0) jn = ny +   j
        if(j .gt.   ny) jn = j   - ny

        if(tmp1 .le. d) then
          tmp3 =
        else
          tmp3 = 0.
        endif

        if(tmp2 .le. d) then
          tmp3 =
        else
          tmp3 = 0.
        endif
        upt(m) = upt(m) + u(in,jn)*tmp3
        vpt(m) =
```

```
    enddo; enddo
  enddo

  do m = 1, np;
    fxpe(m) = upe(m) - upt(m)
    fype(m) = [                    ]
  enddo

c.. Spreading: force (fyib, fyib) --> (fx, fy)
  do i = 1,nx; do j = 1,ny;
    fx(i,j) = 0.; fy(i,j) = 0.;
  enddo; enddo

  do m = 1, np;
    if(xp(m) .gt.   nx) xp(m) = mod(xp(m), float(nx))
    if(yp(m) .gt.   ny) yp(m) = mod(yp(m), float(ny))
    do i = int(xp(m)) - 3, int(xp(m)) + 3
    do j = int(yp(m)) - 3, int(yp(m)) + 3
       tmp1 = abs(xp(m) - i); tmp2 = abs(yp(m) - j)
       in = i; jn = j;
       if(i .le.     0) in = nx +    i
       if(i .gt.   nx) in = mod(i,nx)
       if(j .le.     0) jn = ny +    j
       if(j .gt.   ny) jn = mod(j,ny)

       if(tmp1 .le. d) then
         tmp3 = [                    ].
       else
         tmp3 = 0.
       endif
       if(tmp2 .le. d) then
         tmp3 = [                    ]
       else
         tmp3 = 0.
       endif
      fx(in,jn) = fx(in,jn) + fxpe(m)*tmp3*2.*pi*rp/float(np)
      fy(in,jn) = fy(in,jn) + [                          ]
    enddo; enddo
  enddo
```

```
c.. particle-wall collision
            zeta= rp/8
c.. left wall
            tmp1 = abs(xpc - 1. + rp)
            if(tmp1 .lt. 2.*rp + zeta)then
              fxc = fxc + (xpc - 1. + rp)/abs(xpc-1.+rp)
     .              *((2.*rp - tmp1 + zeta)/zeta)**2*0.00001
            endif
c.. bottom wall
            tmp1 = abs(ypc - 1. + rp)
            if(tmp1 .lt. 2.*rp + zeta)then
              fyc = fyc + (ypc - 1. + rp)/abs(ypc-1.+rp)
     .              *((2.*rp - tmp1 + zeta)/zeta)**2*0.00001
            endif
c.. right wall
            tmp1 = abs(xpc - nx - rp)
            if(tmp1 .lt. 2.*rp + zeta)then
              fxc = fxc + (xpc - nx - rp)/abs(xpc-nx-rp)
     .              *((2.*rp - tmp1 + zeta)/zeta)**2*0.00001
            endif
c.. top wall
            tmp1 = abs(ypc - ny - rp)
            if(tmp1 .lt. 2.*rp + zeta)then
              fyc = fyc + (ypc - ny - rp)/abs(ypc-ny-rp)
     .              *((2.*rp - tmp1 + zeta)/zeta)**2*0.00001
            endif

c.. particle motion
            fxp = 0.; fyp = 0.; torq = 0.
            do m = 1, np;
              fxp = fxp + fxpe(m);
              fyp = fyp + fype(m)
              torq=torq +    fype(m)*(xp(m) - xpc)
     .                  -    fxpe(m)*(yp(m) - ypc)
            enddo
            fxp = -fxp*2.*pi*rp/float(np)
            fyp = -fyp*2.*pi*rp/float(np)
            torq= torq*2.*pi*rp/float(np)

            dx     =(fxp + fxc)/rhop/mp    + (1. - 1./rhop)*g
```

```
          dy    =(fyp + fyc)/rhop/mp   + (1. - 1./rhop)*g
          up    =( 1 + 1./rhop)*up1     - 1./rhop*up2 + dx
          vp    =
          omega=( 1 + 1./rhop)*omega1 - 1./rhop*omega2
     .          - torq/rhop/iip
          xpc = xpc + (up + up1)*0.5
          ypc =
          theta = theta + (omega + omega1)*0.5

          up2 =    up1;       up1 =     up;
          vp2 =    vp1;       vp1 =     vp;
          omega2 =omega1;   omega1 = omega;

          do m = 1, np;
           xp(m) = xpc + rp*cos(2.*pi*(m - 1)/float(np))
           yp(m) = ypc + rp*sin(2.*pi*(m - 1)/float(np))
          enddo;

          do m = 1, np;
           upe(m) = up   - omega*(yp(m) - ypc);
           vpe(m) = vp   + omega*(xp(m) - xpc);
          enddo

c..loop nf2
          enddo
c.. graphics
          do i = 1, nx; do j = 1, ny
           a(i,j)= '+'; b(i,j)= '9';
          end do; end do

          do i = 1, nx; do j = 1, ny
           if((i-xpc)**2 + (j-ypc)**2 .le. rp**2) a(i,j)= '*'
          end do; end do

          umax = 0.; umin = 100.;
          do i = 1, nx; do j = 1, ny
           u2 =u(i,j)**2 + v(i,j)**2
           if(u2 .gt. umax) umax = u2
           if(u2 .lt. umin) umin = u2
          end do; end do
```

```
      do i = 1, nx; do j = 1, ny
        u2 =u(i,j)**2 + v(i,j)**2
        if(u2 .le. umax*0.9 + umin*0.1) b(i,j)= '8'
        if(u2 .le. umax*0.8 + umin*0.2) b(i,j)= '7'
        if(u2 .le. umax*0.7 + umin*0.3) b(i,j)= '6'
        if(u2 .le. umax*0.6 + umin*0.4) b(i,j)= '5'
        if(u2 .le. umax*0.5 + umin*0.5) b(i,j)= '4'
        if(u2 .le. umax*0.4 + umin*0.6) b(i,j)= '3'
        if(u2 .le. umax*0.3 + umin*0.7) b(i,j)= '2'
        if(u2 .le. umax*0.2 + umin*0.8) b(i,j)= '1'
        if(u2 .le. umax*0.1 + umin*0.9) b(i,j)= '0'
      end do; end do

      write(*,*) t, xpc,ypc
      do j = ny,1,-1
        write(*,*) (a(i,j),i = 1,nx),'    ', (b(i,j),i = 1,nx)
      end do

c.. loop nf1
     end do

     stop
     end
```