ソースファイル名：fdlbmg.for，実行ファイル名：fdlbmg.out

```
$ cp   ../common/ fdlbmg_ori.for   .
$ cp   fdlbmg_ori.for     fdlbmg.for
$ vi     fdlbmg.for
$ gfortran   -o   fdlbmg.out     fdlbmg.for
$ ./ fdlbmg.out
```

```
      program fdlbm

      real*8   cx(9), cy(9)
      real*8   f(9,35,35), fn(9,35,35), f0(9,35,35)
      real*8   u(35,35), v(35,35), rho(35,35)
      real*8   tau, g, ntime, mu, dx , dy , dt
      integer nx, ny
      character a(40,40), b(40,40)

      tau = 0.6; g = 0.0005; dx = 1.0; dy = 1.0; dt = 0.5
      ntime = 0.; nx = 32; ny = 34; mu = tau/3.

c.. discrete velocity
      cx(1) =   0.0; cy(1) =   0.0
      cx(2) =   1.0; cy(2) =   0.0
      cx(3) =   1.0; cy(3) =   1.0
      cx(4) =   0.0; cy(4) =   1.0
      cx(5) = -1.0; cy(5) =   1.0
      cx(6) = -1.0; cy(6) =   0.0
      cx(7) = -1.0; cy(7) = -1.0
      cx(8) =   0.0; cy(8) = -1.0
      cx(9) =   1.0; cy(9) = -1.0

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

      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 = 1,4
         m = k*2       ; tmp = cx(m)*u(i,j) + cy(m)*v(i,j)
         f0(m,i,j) =
         m = k*2 + 1; tmp = cx(m)*u(i,j) + cy(m)*v(i,j)
```

```
          f0(m,i,j) =rho(i,j)*(1. + 3.*tmp + 9./2.*tmp**2 - 3./2.*u2)/36.
        end do
      end do; end do

      do k = 1,9; do i = 1,nx; do j = 1,ny
        f(k,i,j) = f0(k,i,j)
      end do; end do; end do

100   do n1 = 1,50; do n2 = 1, 100

      do k = 1,9; do i = 1,nx; do j = 1,ny
       fn(k,i,j) =
      end do; end do; end do

      ntime = ntime + 1

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) =
         v(i,j) =
        rho(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.. boundary_mac
      do i = 1,nx
       u(i,2     ) = 0.; v(i,      ) = 0.;
       u(i,ny-1) = 0.; v(i,      ) = 0.;
```

```
      enddo

c..   equiblium
      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 = 1,4
        m = k*2       ; tmp = cx(m)*u(i,j) + cy(m)*v(i,j)
        f0(m,i,j) = [                                    ]
        m = k*2 + 1; tmp = cx(m)*u(i,j) + cy(m)*v(i,j)
        f0(m,i,j)= rho(i,j)*(1. + 3.*tmp + 9./2.*tmp**2 - 3./2.*u2)/36.
       end do
      end do; end do

c..   collide
      do k = 1,9; do i = 1,nx; do j = 1,ny
       fn(k,i,j) = fn(k,i,j) - (f(k,i,j) - [      ])/tau
      end do; end do; end do

c..   force
      do k = 1,9; do i = 1,nx; do j = 1,ny
       fn(k,i,j) = fn(k,i,j) + g*[                      ]
      end do; end do; end do

c.. streaming (centered difference)
      do k = 1,9; do i = 1,nx; do j = 1,ny
       ip = i + 1; im = i - 1
       if(i .eq. nx) ip = 1
       if(i .eq. 1 ) im = nx

       jp = j + 1; jm = j - 1
       if(j .eq. ny) jp = 1
       if(j .eq. 1 ) jm = ny

       fn(k,i,j) = fn(k,i,j)
     .           -( cx(k)*(f(k,ip,j ) - f(k,im,j ))/dx
     .           +[                          ])*.5
      end do; end do; end do

c.. boundary_mic
```

```
      do k = 1,9; do i = 1,nx
        fn(k,i,1)   = 2.*fn(k,i,2)       - fn(k,i,3)
        fn(k,i,ny) =
       enddo; enddo

      do k = 1,9; do i = 1,nx; do j = 1,ny
       f(k,i,j) = f(k,i,j) +
      end do; end do; 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) =
         v(i,j) =
        rho(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

      end do
c.. nf2

      uxmax = 0.
      do i = 1,nx; do j = 1,ny
       if( u(i,j) .ge. uxmax) uxmax = u(i,j)
      end do; end do

      write (*, *) 'Time    = ', ntime*dt,',      ',
     .                    'Error = ',
     . (g*(ny-3)**2*dy**2/8./mu - uxmax)/(g*(ny-3)**2*dy**2/8./mu)
```

```
c
          write (*,'(a30,f15.10)') 'U_max (Numerical    Solution)', uxmax
          write (*,'(a30,f15.10)') 'U_max (Analytical Solution)',
        . g*(ny-3)**2*dy**2/8./mu

2010      format(x, 34e15.6)
2020      format(x, 4e15.6)


c..    graphics
            do i = 1, nx; do j = 1, ny
              a(i,j)= '0'
            end do; end do

            do j = 1, ny
              nb = int(u(nx/2,j)*75)
              do i = 1, nb
                a(i,j)= '-'
              end do
            end do

            do i = 1, nx; do j = 1, ny
              b(i,j)= '0'
            end do; end do
            do j = 1, ny
              nb = int((-g/mu/2.*(j - 2)*(j - ny+1))*75)
              do i = 1, nb
                b(i,j)= '-'
              end do
            end do

            do j = ny,1,-1
              write(*,*) (a(i,j),i = 1,nx),'    ',(b(i,j),i = 1,nx)
            end do
            write(*,*) '*------------------------------*'

          end do
c.. nf1

          end
```

```
*----------------------------*
Time  =    2500.00000000000      ,   Error =   5.002219055760412E-003
  U_max (Numerical  Solution)   0.2988102734
  U_max (Analytical Solution)   0.3003125023
0000000000000000000000000000000000  0000000000000000000000000000000000
0000000000000000000000000000000000  0000000000000000000000000000000000
--00000000000000000000000000000000  --00000000000000000000000000000000
-----000000000000000000000000000000  -----000000000000000000000000000000
-------000000000000000000000000000  -------000000000000000000000000000
----------0000000000000000000000  ----------0000000000000000000000
------------00000000000000000000  ------------00000000000000000000
--------------000000000000000000  --------------000000000000000000
----------------0000000000000000  ----------------0000000000000000
------------------000000000000000  ------------------000000000000000
--------------------00000000000000  --------------------00000000000000
----------------------0000000000000  ----------------------0000000000000
------------------------000000000000  ------------------------000000000000
--------------------------00000000000  --------------------------00000000000
----------------------------0000000000  ----------------------------0000000000
------------------------------000000000  ------------------------------000000000
--------------------------------00000000  --------------------------------00000000
----------------------------------0000000  ----------------------------------0000000
------------------------------------000000  ------------------------------------000000
--------------------------------------00000  --------------------------------------00000
----------------------------------------0000  ----------------------------------------0000
------------------------------------------000  ------------------------------------------000
--------------------------------------------00  --------------------------------------------00
----------------------------------------------0  ----------------------------------------------0
------------------------------------------------  ------------------------------------------------
--00000000000000000000000000000000  --00000000000000000000000000000000
0000000000000000000000000000000000  0000000000000000000000000000000000
0000000000000000000000000000000000  0000000000000000000000000000000000
```