```
#include"utilAnsu.h"
#include"lbe2D.h"

main()

{
 FILE *fpt, *fpt1, *fpt2;
 char fileNameOld[40] , fileNameNew[40];
 nodeD2Q9  **latPoint;
 double feq[N_DV];
 double rho, ux, uy;
 double sum, fout;
 double rhoM;
 int nX, nY, time, finalTime;
 double   Re, Ma, Kn, den, u_max;
 double visc,freq, freqC,amp;
 int iX, iY, dv;
 int iXL, iXR, iYU, iYD;
 int lastXM1,lastXP1;
 int lastYM1,lastYP1;
 double vel0,vel1, vel2, vel3, velN;
 double h, tmp1, tmp2, tmp3, tmp4, tmp5;
 double lambda, exph, wtCarl1, wtCarl2, wtCarl3;
 double rhoIn, rhoOut,flowrate_Ref;
 double tau,Kn_M;
 double pIn, pOut;
 nX = 0;
 nY = 0;

/******************Read file par.dat******/

 fpt=fopen("start.dat","r");

 fpt1=fopen("center.dat","w");
 fpt2=fopen("vel.dat","w");

 fprintf(fpt2,"VARIABLES= 'X', 'Y','U', 'V', 'rho' \n");

 if (fpt!=NULL){
   fscanf(fpt,"%d", &nX);
   fscanf(fpt,"%d", &nY);
   fscanf(fpt,"%lf", &Kn);
   fscanf(fpt,"%lf", &rhoIn);
   fscanf(fpt,"%lf", &rhoOut);
   fscanf(fpt,"%lf", &u_max);
   fscanf(fpt,"%lf", &flowrate_Ref);
   fscanf(fpt,"%lf", &tau);
   fscanf(fpt,"%d", &finalTime);

   printf(" nx= %d\n",  nX);
   printf(" nY= %d\n",  nY);
   printf(" Kn= %lf\n", Kn);

    }
   else{
     fprintf(stderr,"\nCheck file par.dat");
     exit(0);

 }

 if (fpt1!=NULL){
   fprintf(fpt1,"# nx= %d\n",  nX);
   fprintf(fpt1,"# nY= %d\n",  nY);
   fprintf(fpt1,"# Kn= %lf\n", Kn);
fprintf(fpt1,"# rhoIn= %lf\n", rhoIn);
   fprintf(fpt1,"# rhoOut= %lf\n", rhoOut);
 }
```



```
latPoint= createNode( nX,  nY);



Twall = T = 1.0/3.0;
rhoM =( rhoIn + rhoOut)/2.0;
Kn_M = Kn ;
visc = tau * rhoM /3.0;
pIn = rhoIn *T;
pOut = rhoOut *T;
printf("\n visc= %lf\n",visc);
printf(" Diff. time scale= %lf\n",rhoOut*  (nY-1.0)*(nY-1.0)/visc);

fprintf(fpt1,"# visc= %lf\n",visc);
fprintf(fpt1,"# Diff. time scale= %lf\n",rhoOut*(nY-1.0)*(nY-1.0)/visc);



printf(" Kn = %lf\n",Kn);
fprintf(fpt1,"# Kn = %lf\n",Kn);


printf(" Max velocity = %lf\n",u_max);
fprintf(fpt1,"# Max velocity = %lf\n",u_max);



rho = rhoM;

ux = 0.0;
uy = 0.0;

printf("\n Before Eq. Cal rho = %.12lf, ux = %.12lf, uy = %.12lf\n",rho,  ux,  uy);

getFeqP(feq,  rho,  ux,  uy);

/*************************************************************
 * Set some dummy variable to  check end points during the   *
 *   simulations                                             *
 *************************************************************/

lastXP1 = nX + 1;
lastXM1 = nX - 1;

lastYP1 = nY + 1;
lastYM1 = nY - 1;

   /* Initialise all population with zero velocity ******/
for ( iY = 1; iY <lastYP1 ; iY++){
   for ( iX = 1; iX < lastXP1; iX++){
      for(dv=0;dv<N_DV; dv++)
       latPoint[iX][iY].f[dv] = feq[dv];
   }
}

printf("\nBefore Main loop...u_max=%lf \n", u_max);

for (time =0; time <finalTime; time++){

   if(time%500==0) printf( "\n Time step = %d ", time);

   if(time==finalTime-1) printPop(latPoint,  nX,  nY, time);



   sum =0.0;

   /* Impose peridicity in X dir*/
```

```
    for ( iY = 1; iY <= nY; iY++){
      for (dv=0;dv<N_DV; dv++){
        latPoint[0][iY].f[dv]  = latPoint[lastXM1][iY].f[dv];
        latPoint[lastXP1][iY].f[dv]  = latPoint[2][iY].f[dv];
      }
    }



    /*for Down and Left Down population*/

    /******Lower Wall *****/
    iYU= lastYM1;
    iY =nY;
    for ( iX = 1, iXR=2; iX <= nX; iX++) {
      latPoint[iX][lastYP1].f[ZERO] =latPoint[iX][iY].f[ZERO];

      latPoint[iX][lastYP1].f[D] = 0.5 * ( latPoint[iX][iYU].f[D]\
                                  + latPoint[iX][iY].f[D]);
      latPoint[iX][lastYP1].f[L] = latPoint[iXR][iY].f[L];
      latPoint[iX][lastYP1].f[LD] = 0.5*( latPoint[iXR][iYU].f[LD] + latPoint[iX
][iY].f[LD]);

      iXR++;
    }

    iXL= lastXM1;
    for ( iX = nX; iX > 0; iX--) {
      latPoint[iX][lastYP1].f[RD] = 0.5*(latPoint[iXL][iYU].f[RD]+latPoint[iX][i
Y].f[RD]) ;
      latPoint[iX][lastYP1].f[R] = latPoint[iXL][iY].f[R];

      iXL--;
    }
   for ( iX = 1; iX <= nX; iX++) {
      fout  = (latPoint[iX][lastYP1].f[D]+latPoint[iX][lastYP1].f[RD]+latPoint[i
X][lastYP1].f[LD])  ;
      latPoint[iX][lastYP1].f[U] = 16.0/36.0 * fout ;
      latPoint[iX][lastYP1].f[RU] = 4.0/36.0 * fout ;
      latPoint[iX][lastYP1].f[LU] = 4.0/36.0 * fout ;

   }

  iYU= lastYM1-1;
  for ( iY = lastYM1; iY >1; iY--){
     for ( iX = 1, iXR=2; iX <= nX; iX++) {
        latPoint[iX][iY].f[D] = latPoint[iX][iYU].f[D];
        latPoint[iX][iY].f[L] = latPoint[iXR][iY].f[L];
        latPoint[iX][iY].f[LD] = latPoint[iXR][iYU].f[LD];

        iXR++;
     }
     iXL= lastXM1;
     for ( iX = nX; iX > 0; iX--) {
        latPoint[iX][iY].f[RD] = latPoint[iXL][iYU].f[RD];
        latPoint[iX][iY].f[R] = latPoint[iXL][iY].f[R];

        iXL--;
     }

     iYU--;
  }

/**********************Top Wall***********************/

  iYD= 2;
  iY = 1;

  for (iX = 1, iXR=2; iX <= nX; iX++) {
```



```
     latPoint[iX][0].f[ZERO] =latPoint[iX][iY].f[ZERO];

     latPoint[iX][0].f[U] = 0.5*(latPoint[iX][iYD].f[U]+latPoint[iX][iY].f[U]);
     latPoint[iX][0].f[L] = latPoint[iXR][iY].f[L];
     latPoint[iX][0].f[LU] =0.5*( latPoint[iXR][iYD].f[LU]+latPoint[iX][iY].f[LU
]);

     iXR++;
  }

  iXL= lastXM1;
  for ( iX = nX; iX > 0; iX--) {
     latPoint[iX][0].f[R] = latPoint[iXL][iY].f[R];
     latPoint[iX][0].f[RU] = 0.5*(latPoint[iXL][iYD].f[RU]+ latPoint[iX][iY].f[R
U]);



 iXL--;
  }
  for ( iX = nX; iX > 0; iX--) {
     fout  = latPoint[iX][0].f[U]+latPoint[iX][0].f[RU] + latPoint[iX][0].f[LU]
 ;
     latPoint[iX][0].f[D] = 16.0/36.0 * fout ;
     latPoint[iX][0].f[RD] = 4.0/36.0 * fout ;
     latPoint[iX][0].f[LD] = 4.0/36.0 * fout ;

  }

   /***************Bulk**********/

  iYD= 3;
  for ( iY = 2; iY <nY; iY++){
     for ( iX = 1, iXR=2; iX <= nX; iX++) {
        latPoint[iX][iY].f[U] = latPoint[iX][iYD].f[U];
        latPoint[iX][iY].f[LU] = latPoint[iXR][iYD].f[LU];

        iXR++;
     }
     iXL= lastXM1;
     for ( iX = nX; iX > 0; iX--) {
        latPoint[iX][iY].f[RU] = latPoint[iXL][iYD].f[RU];

        iXL--;
     }
     iYD++;
  }

/* *****Copy value from dummy node to real boundary node*****/
  if (time%20000==0)
           fprintf(fpt2,"ZONE T='%d ', I= %d , J= %d,F=POINT\n", time, nX, nY);

  for ( iX = 1; iX <= nX; iX++) {
     for (dv =0;dv< N_DV; dv++){
        latPoint[iX][nY].f[dv] = latPoint[iX][lastYP1].f[dv] ;
        latPoint[iX][1].f[dv]  = latPoint[iX][0].f[dv] ;

     }
  }

  for ( iY = 1; iY <=lastYP1 ; iY++){
     for ( iX = 1; iX < lastXP1; iX++){
        getMoment( latPoint[iX][iY].f, &rho, &ux, &uy);
        sum +=rho;

   /*ux = ux + 0.5 *  forceX/ rho;*/

        if (iX==nX/2 && iY==nY/2 && time%1000==0)
          printf("\ncenterline profile at t= %d  is  rho = %lf,  ux=%lf,  uy= %lf ",
```

```
                time,rho, ux/u_max, uy);
        if (iX==nX/2 && iY==nY/2) fprintf(fpt1," %d %.10lf %.10lf %.10lf \n",time,ux/u_max,u
y);

/*   Dump velocity and density fields   */
        if (time%20000==0) fprintf(fpt2, "%f %f %002.16lf %002.16lf %002.16lf \n",
                    (float)iX, (float)iY, ux, uy, rho );

        getFeqP(feq,  rho,  ux,  uy);
        if(T<0.1) {
printf( "\n....time =%d %f %f %002.16lf %002.16lf %002.16lf  %002.16lf \n",
                   time, (float)iX, (float)iY, ux, uy, rho, T );
 exit(0);
        }
        freq = tau * rhoM/(rho);

        freq =  2.0/(2.0* freq + 1.0);
 /*     freq = 1- exp(-1.0/freq);*/

        freqC = (1.0 -  freq);
        for (dv =0;dv< N_DV; dv++){
           latPoint[iX][iY].f[dv] = latPoint[iX][iY].f[dv] * freqC + freq * feq[
dv];
      /* Add force term
        force =  (DVX[dv] - (ux) + 3.0 * DVX[dv]* ( ux*DVX[dv] + uy*DVY[dv] ) )
;
        force  *=Wt[dv] *amp;
        latPoint[iX][iY].f[dv] += force ; */
        }
     }
  }
/*if(time>0) printf("\n at time (%d )total density = %lf", time, sum);*/
Kn = Kn_M;
 if (time%1000==0)
   printMomC(latPoint, u_max,flowrate_Ref, Kn, visc, nX, nY, time);

}


/* Now print center line profile and analytical solution using Navier-Stokes */

 printMomC(latPoint,  u_max,flowrate_Ref, Kn, visc,  nX,  nY, finalTime);

 /*****************************************************************
  *  In order to plot the data using matlab datafile is renamed as *
  *   profileSteady.dat                                            *
  *
  *****************************************************************/

 sprintf(fileNameOld,"momC%d.dat", finalTime);
 sprintf(fileNameNew,"profileSteady.dat");
 rename(fileNameOld, fileNameNew);
 free_lattice(latPoint);

 return;
}


/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~ */

void printPop(nodeD2Q9 **point, int nX, int nY, int label)
{
 FILE *fpt;
 char *fileName;
 int iX, iY, lastYP1, lastXP1;
```



```
 fileName = ( char  * ) malloc( (size_t) (45 * sizeof( char ) ) );

 if (!fileName){
   fprintf(stderr, "Memory allocation failure");
   exit(0);
 }

 sprintf(fileName,"pop%d.dat", label);

 fpt = fopen(fileName,"w");

 lastYP1 = nY +1;
 lastXP1 = nX +1;

 for ( iY = 1; iY <lastYP1 ; iY++){
    for ( iX = 1; iX < lastXP1; iX++){
       fprintf(fpt,"%f %f %002.16lf %002.16lf %002.16lf %002.16lf %002.16lf %002.16lf %00
2.16lf %002.16lf \n",
                   (float)iX, (float)iY, point[iX][iY] );
    }
  }
}

/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~ */

void printMomC(nodeD2Q9 **point, double v_max, double ref_Flowrate, double Kn, d
ouble visc, int nX, int nY, int label)
{

 FILE *fpt;
 char *fileName;
 int iX, iY, lastYP1, lastXP1;
 double rho, ux, uy, y, ux_anal,slip;
 double forceX;
 double flowrate, flowrate_anal;
 double mass, T;
 forceX = v_max * 8. * visc/((nY-1.0) * (nY-1.0));
 fileName = ( char  * ) malloc( (size_t) (45 * sizeof( char ) ) );

 if (!fileName){
   fprintf(stderr, "Memory allocation failure");
   exit(0);
 }

 sprintf(fileName,"momC%d.dat", label);

 fpt = fopen(fileName,"w");
 fprintf(fpt, "# Velocity profile at x=L/2\n");
 fprintf(fpt, "# Kn = %lf, nX = %d, nY= %d, visc= %lf \n", Kn, nX, nY, visc);
 fprintf(fpt, "#  y    U     U_anal    V     rho\n");


 lastYP1 = nY +1;
 lastXP1 = nX +1;

 flowrate = 0.0;
 flowrate_anal = 0.0;
 mass =0.0;
 for ( iY = 1; iY <lastYP1 ; iY++){
    iX = nX /2 ;
    getMoment(  point[iX][iY].f,  &rho,  &ux, & uy);
    /*  ux = ux + 0.5 *  forceX/ rho;*/
    if (nY%2==0)
      y = nY/2   - iY+1 - 0.5;
    else
      y = nY/2  - iY +1;
```

```
    y = y/ ( (nY-1.0)*1.0);

    slip =  4.0* ( Kn * 1.01615 + Kn*Kn * (0.5 + 1.01615*1.01615  ));

/* Analytical velocity profile */


    ux_anal = (1. - 4.*y*y + slip );

    ux = ux/v_max ;
/* Simpson's rule int_{x1}^{x2} f(x)dx = dx*(f1+4*(f2+f4+f6+...)+2*(f3+f5+f7+...
)+fN)/3  */

    if (iY == 1 || iY == lastYP1-1){
      mass += rho ;
      flowrate += rho *ux;
      flowrate_anal +=rho * ux_anal;}
    else if ( iY%2==0 ) {
      mass += 4.0 * rho;
      flowrate += 4.0*rho * ux;
      flowrate_anal += 4.0*rho * ux_anal;}
    else {
      mass += 2.0 * rho;
      flowrate += 2.0 * rho * ux ;
      flowrate_anal += 2.0*rho * ux_anal;}

    fprintf(fpt, "%002.08lf %002.08lf %002.08lf %002.08lf %002.08lf\n",
           y,  ux, ux_anal, uy , rho);
    }
 mass = mass/(3.0 * nY -3.0);
    flowrate = flowrate/(mass *3.0);
 /* flowrate = flowrate/(mass);
 mass = mass/(3.0 * nY -3.0);*/
      flowrate_anal = flowrate_anal/(mass *3.0);
/*      flowrate = flowrate/ ref_Flowrate;*/
   flowrate_anal = 0.25 * flowrate_anal/(nY-1.0);
   flowrate_anal =  flowrate_anal/Kn;
flowrate = 0.25 * flowrate/(nY-1.0);
   flowrate =  flowrate/Kn;
    printf ("\n At Kn=%.08lf  Comp. Flowrate = %002.08lf Anal. Flowrate =%002.08lf\n",Kn,  flowrate
, flowrate_anal);


}
```