```
//////////////////////////////////////////////////////////////
//おまじない（組込み済み関数の参照用）
//これはそのまま
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
//////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////
//プログラムで扱う変数のうち
//計算中に変更されないパラメータを定義
//以下のプログラム中では、以下の文字は
//それぞれの定義に従い、コンパイル時に
//書きかえられる
//例：NXは51で自動的に書きかえられる
//x方向の格子線の数
#define NX 51
//y方向の格子線の数
#define NY 51
//計算領域のx方向の幅
#define XMAX 1.e-1
//計算領域のy方向の幅
#define YMAX 1.e-1
//クーラン条件により時間刻み幅dtを決定する際の
//安全係数
#define CFL_SAFE_COFF 0.01
//拡散方程式を解く際の安定条件Cdiffに対する
//安全係数
#define VIS_SAFE_COFF 0.1
//////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////
//プログラムで使用する変数の定義（大域変数)
//計算格子に関する変数
//////////////////////////////////////
//格子のインデックス
```

```
int i,j;
//格子線数(nx:x方向,ny:y方向)
int nx,ny;
//計算時に一時的に使用する格子点のインデックス
int iup,jup,idn,jdn;
//x方向の格子線の座標(xx[0]～xx[NX+1]まで使用可能)
double xx[NX+2];
//y方向の格子線の座標(yy[0]～yy[NY+1]まで使用可能)
double yy[NY+2];
//x方向の格子幅（等間隔格子）
double dx;
//y方向の格子幅（等間隔格子）
double dy;
//x方向の計算領域の幅
double xmax;
//y方向の計算領域の幅
double ymax;

////////////////////////////////////////
//Color Functionの値を格納する2次元配列
//値(color_func[0][0]～color_func[NX+1][NY+1]まで使用可能)
double color_func[NX+2][NY+2];
//x方向微分
double color_func_gx[NX+2][NY+2];
//y方向微分
double color_func_gy[NX+2][NY+2];

////////////////////////////////////////
// x方向の流速Uを格納する2次元配列
//値(uu[0][0]～uu[NX+1][NY+1]まで使用可能)
double uu[NX+2][NY+2];
//x方向微分
double uu_gx[NX+2][NY+2];
//y方向微分
double uu_gy[NX+2][NY+2];

////////////////////////////////////////
```

```
// y方向の流速Vを格納する2次元配列
//値(vv[0][0]～vv[NX+1][NY+1]まで使用可能)
double vv[NX+2][NY+2];
//x方向微分
double vv_gx[NX+2][NY+2];
//y方向微分
double vv_gy[NX+2][NY+2];

//////////////////////////////////////
//Color Functionから見積もられる
//実際の密度ρの値を保存する2次元配列
//(density[0][0]～density[NX+1][NY+1]まで使用可能)
//(CIP法により移流は計算しないので微分は無い)
double density[NX+2][NY+2];

//////////////////////////////////////
//Poisson方程式から求める圧力p
//(pressure[0][0]～pressure[NX+1][NY+1]まで使用可能)
//(CIP法により移流は計算しないので微分は無い)
double pressure[NX+2][NY+2];

//////////////////////////////////////
//種種の物理定数を格納する変数
//動粘性係数（気体でも水でも同一）
double viscosity_coff_mol;
//水の部分の密度（一定値とする）
double density_water;
//気体の部分の密度（一定値とする）
double density_air;
//重力加速度
double gravity;

//////////////////////////////////////
//時間に関する変数
//Time Step "n"
int time_step;
//計算を終了する最大のTime Step数
```

```
int end_time_step;
//時刻 "tn"
double time;
//時間刻み幅Δt
double dt;
//time_step=0 -> 1(n=0 -> 1)における時間刻み幅
double dt_initial;
//計算を終了する最大時刻
double end_time;

//////////////////////////////////////
//出力に関する変数
//中間結果を出力するファイルに付ける通し番号
int output_file_number;
//中間結果を出力する時刻を保存する変数
double output_time;
//中間結果を出力する時間間隔
double output_time_dt;
//出力時にファイルを開くための変数
FILE *fp;
//出力ファイル名を一時的に保存する文字配列
char filename[256];

//////////////////////////////////////
//各種計算時に一時的に必要となる変数
//[i+1][j]における値を一時的に保存
double iup_value;
//[i-1][j]における値を一時的に保存
double idn_value;
//[i][j+1]における値を一時的に保存
double jup_value;
//[i][j-1]における値を一時的に保存
double jdn_value;
//時間刻み幅をクーラン数から決定する際に必要となる
//流速Uの絶対値の最大値を保存するのに使用
double umax;
//時間刻み幅をクーラン数から決定する際に必要となる
```

```
//流速Vの絶対値の最大値を保存するのに使用
double vmax;
//時間刻み幅をクーラン数から決定する際に必要となる
//流速Uの最大値を格子幅dxで割った値(umax/dx)を保存するのに使用
double uu_by_dx;
//時間刻み幅をクーラン数から決定する際に必要となる
//流速Vの最大値を格子幅dyで割った値(vmax/dy)を保存するのに使用
double vv_by_dy;
//時間刻み幅をクーラン数から決定する際に必要となる
//流速UがCFL条件を満たすように決定した仮の時間刻み幅
double dt_uu;
//時間刻み幅をクーラン数から決定する際に必要となる
//流速VがCFL条件を満たすように決定した仮の時間刻み幅
double dt_vv;
//移流計算を行う場合に使用
//スタガード格子上における変数夫々の定義位置に応じた
//流速値を一時的に計算するのに使用する
double uu_tmp;
double vv_tmp;

//各種計算の際に計算前の古い値を格納しておくために使用
//Color Functionの値の古い値
double old_color_func[NX+2][NY+2];
//Color Functionのx微分の古い値
double old_color_func_gx[NX+2][NY+2];
//Color Functionのy微分の古い値
double old_color_func_gy[NX+2][NY+2];
//流速Uの値の古い値
double old_uu[NX+2][NY+2];
//流速Uのx微分の古い値
double old_uu_gx[NX+2][NY+2];
//流速Uのy微分の古い値
double old_uu_gy[NX+2][NY+2];
//流速Vの値の古い値
double old_vv[NX+2][NY+2];
//流速Vのx微分の古い値
double old_vv_gx[NX+2][NY+2];
```

```
//流速Vのy微分の古い値
double old_vv_gy[NX+2][NY+2];
//移流計算の際のxiからの上流点へのx方向の距離 UxDt
double udt;
//移流計算の際のyjからの上流点へのy方向の距離 VxDt
double vdt;
//移流計算の際のxiからの上流方向へのx方向格子幅 xiup-xi
double dxiup;
//移流計算の際のyjからの上流方向へのy方向格子幅 yjup-yj
double dyjup;


//CIP法の補間関数の係数
double C00,C10,C20,C30,C01,C02,C03,C31,C21,C11,C12,C13,AA;
//CIP法の微分の修正に用いる速度の勾配
double dudx,dudy,dvdx,dvdy;

//拡散方程式の解法の時間刻み幅の推定
//一時的な時間刻み幅
double dt_tmp;
double dt_old;

//圧力解法に用いる変数群
//速度の発散
double divU[NX+2][NY+2];
//SOR法の緩和ループ（仮想時間ステップ）の最大値
int sor_step_max;
//SOR法の緩和ループ（仮想時間ステップ)
int sor_step;
//SOR法の収束判定に用いる（収束している格子点数）
int sor_conv_num;
//SOR法の収束条件に用いる微少量ε
double sor_epsiron;
//SOR法に用いる修正量δ
double sor_delta;
//SOR法における緩和係数ω
double sor_omega;
```

```
//SOR法の計算における緩和前の古い値を保存する一時変数
double pressure_old;
//SOR法の計算に用いる一時変数
//pressure[i+1][j]の値を一時的に保存
double pressure_iupj;
//pressure[i-1][j]の値を一時的に保存
double pressure_idnj;
//pressure[i][j+1]の値を一時的に保存
double pressure_ijup;
//pressure[i][j-1]の値を一時的に保存
double pressure_ijdn;
//density[i+1/2][j]の値を一時的に保存
double density_iupj;
//density[i-1/2][j]の値を一時的に保存
double density_idnj;
//density[i][j+1/2]の値を一時的に保存
double density_ijup;
//density[i][j-1/2]の値を一時的に保存
double density_ijdn;
//境界y=ymax上での圧力の境界値
double pressure_boundary;
// 収束判定用 圧力の最大最小値
double pressure_min;
double pressure_max;

////////////////////////////////////////////////////////////////
//使用する関数（サブルーチン）の宣言
int boundary();
int boundary_pressure( );

////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
// 実際のプログラム本体
// 実行時にはここから順番に計算される

int main( )
{
```

```
//////////////////////////////////////////////////////////////
// 計算格子の設定
//////////////////////////////////////////////////////////////
//x方向の格子線数
nx=NX;
//y方向の格子線数
ny=NY;
//x方向の計算領域の幅
xmax=XMAX;
//y方向の計算領域の幅
ymax=YMAX;
//等間隔格子の場合のx方向の格子幅Δx
dx=xmax/(nx-1);
//等間隔格子の場合のy方向の格子幅Δy
dy=ymax/(ny-1);
//格子線のx座標 xi=xx[i]
//xx[0]～xx[nx+1] (xx[0],xx[nx+1]:仮想セル)
//実際の計算領域 xx[1]～xx[nx]
//xx[1]=0, xx[nx]=xmaxに一致
for(i=0;i<=nx+1;i++){
  xx[i]=dx*(i-1);
}
//格子線のy座標 yj=yy[j]
//yy[0]～yy[ny+1] (yy[0],yy[ny+1]:仮想セル)
//実際の計算領域 yy[1]～yy[ny]
//yy[1]=0, yy[ny]=ymaxに一致
for(j=0;j<=ny+1;j++){
  yy[j]=dy*(j-1);
}
//////////////////////////////////////////////////////////////
//計算格子の設定:ここまで
//////////////////////////////////////////////////////////////

//////////////////////////////////////////////////////////////
// 計算格子の座標点の出力(確認用)
```

```
//////////////////////////////////////////////////////////////
printf("Number of Grid Points: Nx=%d Ny=%d\n",nx,ny);
printf("Domain Size          : Xmax=%e Ymax=%e\n",xmax,ymax);
printf("Location of Grid Points is outputed into \"Grid.dat\"\
// File Prepare
fp=fopen("Grid.dat","w");
fprintf(fp,"i j xx[i] yy[j]\n");
for(i=1;i<=nx;i++){
  for(j=1;j<=ny;j++){
    fprintf(fp,"%d %d %e %e\n",i,j,xx[i],yy[j]);
  }
}
fclose(fp);
//////////////////////////////////////////////////////////////
// 計算格子の座標点の出力:ここまで
//////////////////////////////////////////////////////////////



//////////////////////////////////////////////////////////////
//計算の時間方向に関する設定
//////////////////////////////////////////////////////////////
//計算開始時の時刻
time=0.;
//1ステップ目の時間刻みΔt
dt_initial=1.e-10;
//計算を終了する時刻
end_time=0.5;
//計算を終了するステップ数
end_time_step=10000000;
//////////////////////////////////////////////////////////////
//計算の時間方向に関する設定:ここまで
//////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////
//計算結果の出力に関する設定
//////////////////////////////////////////////////////////////
//途中結果の出力のファイルに付ける通し番号の開始値
```

```
output_file_number=0;
//途中結果を出力する時間間隔
output_time_dt=1.e-2;
//途中結果を出力する次の時刻
output_time=output_time_dt;
//////////////////////////////////////////////////////////////
//計算結果の出力に関する設定:ここまで
//////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////
//初期条件の設定
//////////////////////////////////////////////////////////////
//種々の物理定数
//動粘性係数
viscosity_coff_mol=1.e-6;
//水の密度(Projection法では一定とする)
density_water=1.e+3;
//気体の密度(Projection法では一定とする)
density_air=1.e0;
//重力加速度
gravity=-9.8;

//////////////////////////////////
// Color Functionの設定
//////////////////////////////////
// color_func[i][j]==1:水の領域
// color_func[i][j]==0:空気の領域
// 注意:Color Functionは計算セルの中央に定義するので
//       変数の範囲は, i=1～nx-1, j=1～ny-1
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny-1;j++){
    //水柱の領域の指定
    if((xx[i]<0.3*xmax)&&(yy[j]<0.6*ymax)){
     //水
      color_func[i][j]=1.;
    }else{
```

```
      //空気
       color_func[i][j]=0.;
     }
   }}
////////////////////////////////
// Color Functionの設定:ここまで
////////////////////////////////

////////////////////////////////
// 流速Uの設定
////////////////////////////////
// 注意:Uはxi上に定義するので
//      変数の範囲は, i=1〜nx, j=1〜ny-1
for(i=1;i<=nx;i++){
  for(j=1;j<=ny-1;j++){
      uu[i][j]=0.;
  }}
////////////////////////////////
// 流速Uの設定:ここまで
////////////////////////////////

////////////////////////////////
// 流速Vの設定
////////////////////////////////
// 注意:Vはyj上に定義するので
//      変数の範囲は, i=1〜nx-1, j=1〜ny
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny;j++){
    vv[i][j]=0.;
  }}
////////////////////////////////
// 流速Vの設定:ここまで
////////////////////////////////

printf("Initial Condition:End\n");

//////////////////////////////////////////////////
//境界条件の設定
```

```
//////////////////////////////////////////////////
//以上で設定されたColor Function, U, Vの値を基に
//仮想セルに境界条件に基づく値を設定(代入)する
boundary( );
//////////////////////////////////////////////////
//境界条件の設定:ここまで
//////////////////////////////////////////////////


//////////////////////////////////////////////////
//CIP法で使用する微分(x勾配:gx, y勾配:gy)の
//初期条件を中心差分により設定
//////////////////////////////////////////////////

//////////////////////////////////
// Color Functionの微分の設定
//////////////////////////////////
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny-1;j++){
    iup=i+1;
    idn=i-1;
    jup=j+1;
    jdn=j-1;
    iup_value=color_func[iup][j];
    idn_value=color_func[idn][j];
    jup_value=color_func[i][jup];
    jdn_value=color_func[i][jdn];
    //x方向中心差分
    color_func_gx[i][j]=(iup_value-idn_value)/(2.*dx);
    //y方向中心差分
    color_func_gy[i][j]=(jup_value-jdn_value)/(2.*dy);
  }}
//////////////////////////////////
// Color Functionの微分の設定:ここまで
//////////////////////////////////

//////////////////////////////////
// 流速Uの微分の設定
//////////////////////////////////
```

```
for(i=1;i<=nx;i++){
  for(j=1;j<=ny-1;j++){
    iup=i+1;
    idn=i-1;
    jup=j+1;
    jdn=j-1;
    iup_value=uu[iup][j];
    idn_value=uu[idn][j];
    jup_value=uu[i][jup];
    jdn_value=uu[i][jdn];
    //x方向中心差分
    uu_gx[i][j]=(iup_value-idn_value)/(2.*dx);
    //y方向中心差分
    uu_gy[i][j]=(jup_value-jdn_value)/(2.*dy);
  }}
//////////////////////////////////
// 流速Uの微分の設定:ここまで
//////////////////////////////////

//////////////////////////////////
// 流速Vの微分の設定
//////////////////////////////////
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny;j++){
    iup=i+1;
    idn=i-1;
    jup=j+1;
    jdn=j-1;
    iup_value=vv[iup][j];
    idn_value=vv[idn][j];
    jup_value=vv[i][jup];
    jdn_value=vv[i][jdn];
    //x方向中心差分
    vv_gx[i][j]=(iup_value-idn_value)/(2.*dx);
    //y方向中心差分
    vv_gy[i][j]=(jup_value-jdn_value)/(2.*dy);
  }}
//////////////////////////////////
// 流速Vの微分の設定:ここまで
```

```
  ////////////////////////////////

  ////////////////////////////////////////////////
  //境界条件の設定
  ////////////////////////////////////////////////
  //以上で設定されたColor Function, U, Vの値を基に
  //仮想セルに境界条件に基づく値を設定(代入)する
  //計算領域内部の値が更新されるたびに境界条件を
  //仮想セル上に上書きしなければならない
  boundary( );
  ////////////////////////////////////////////////
  //境界条件の設定:ここまで
  ////////////////////////////////////////////////

  printf("Initial Derivative Estimation:End\n");

  ////////////////////////////////////////////////////////////////
  //初期条件の設定:ここまで
  ////////////////////////////////////////////////////////////////



  ////////////////////////////////////////////////////////////////
  //設定された初期条件を出力(確認用)
  ////////////////////////////////////////////////////////////////

  ////////////////////////////////
  // Color Functionの出力
  ////////////////////////////////
  // Color_func_initial.datというファイル名で出力
  fp=fopen("Color_func_initial.dat","w");
  fprintf(fp,"xx[i] yy[j] color_func[i][j] color_func_gx[i][j] c
\n");
  for(i=1;i<=nx-1;i++){
    for(j=1;j<=ny-1;j++){
      fprintf(fp,"%e %e %e %e %e\n",xx[i],yy[j],
          color_func[i][j],color_func_gx[i][j],color_func_gy[i][
    }
    fprintf(fp,"\n");
```

```
}
fclose(fp);
////////////////////////////////
// Color Function:ここまで
////////////////////////////////

////////////////////////////////
// 流速Uの出力
////////////////////////////////
// UU_initial.datというファイル名で出力
fp=fopen("UU_initial.dat","w");
fprintf(fp,"xx[i] yy[j] uu[i][j] uu_gx[i][j] uu_gy[i][j]\n");
for(i=1;i<=nx;i++){
  for(j=1;j<=ny-1;j++){
    fprintf(fp,"%e %e %e %e %e\n",xx[i],yy[j],
        uu[i][j],uu_gx[i][j],uu_gy[i][j]);
  }
  fprintf(fp,"\n");
}
fclose(fp);
////////////////////////////////
// 流速U:ここまで
////////////////////////////////

////////////////////////////////
// 流速Vの出力
////////////////////////////////
// VV_initial.datというファイル名で出力
fp=fopen("VV_initial.dat","w");
fprintf(fp,"xx[i] yy[j] vv[i][j] vv_gx[i][j] vv_gy[i][j]\n");
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny;j++){
    fprintf(fp,"%e %e %e %e %e\n",xx[i],yy[j],
        vv[i][j],vv_gx[i][j],vv_gy[i][j]);
  }
  fprintf(fp,"\n");
}
fclose(fp);
////////////////////////////////
```

```
// 流速V:ここまで
//////////////////////////////////

//////////////////////////////////////////////////////////////////
//設定された初期条件を出力(確認用):ここまで
//////////////////////////////////////////////////////////////////


//////////////////////////////////////////////////////////////////
//メインループ:時間発展を計算する本体
//////////////////////////////////////////////////////////////////
// 次のtime_stepを1～ 1づつ増加させて計算を繰り返す
// ステップ数time_stepがend_time_stepより大きくなるか
// 時刻timeがend_timeより大きくなったら終了する
for(time_step=1;
    (time_step<=end_time_step)&&(time<=end_time);
    time_step++){

  ////////////////////////////////////////////////////////////
  //時間刻み幅Δtの決定
  ////////////////////////////////////////////////////////////
  //安定条件を満足するように決定する
  //①:CFL条件 (Udt/dx)<CFL_SAFE_COFF
  //②:拡散方程式に関する条件
  //           (vis_coff_mol*dt/dx/dx)<VIS_SAFE_COFF/2
  ////////////////////////////////////////////////////////////
  dt_old=dt;

  ////////////////////////////////////
  //①:CFL条件に関する条件
  ////////////////////////////////////
  //流速Uの絶対値の最大を求める
  umax=-1.e+10;
  for(i=1;i<=nx;i++){
    for(j=1;j<=ny-1;j++){
     //fabs(XX): XXの絶対値を計算する組込み関数
 if(fabs(uu[i][j])>umax){umax=fabs(uu[i][j]);}
    }}
```

```
//流速Vの絶対値の最大を求める
vmax=-1.e+10;
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny;j++){
if(fabs(vv[i][j])>vmax){vmax=fabs(vv[i][j]);}
  }}
//Umax/dxとVmax/dyを一時的に計算する
uu_by_dx=umax/dx;
vv_by_dy=vmax/dy;
//UmaxからCFL条件を満足するようなΔtを決定(dt_uu)
if(uu_by_dx>0.){
  dt_uu=CFL_SAFE_COFF/uu_by_dx;
}else{
  //Umax=0の場合、即ちすべてのUが0の場合
  //Uからの制限はない
  dt_uu=0.;
}
//VmaxからCFL条件を満足するようなΔtを決定(dt_vv)
if(vv_by_dy>0.){
  dt_vv=CFL_SAFE_COFF/vv_by_dy;
}else{
  //Vmax=0の場合、即ちすべてのVが0の場合
  //Vからの制限はない
  dt_vv=0.;
}
//dt_uuとdt_vvの小さい方を次の時間刻みとする
//UとVともにある流速を持つ場合
if((dt_uu>0.)&&(dt_vv>0.)){
  //dt_uuとdt_vvの小さい方を次の時間刻みとする
  if(dt_uu<dt_vv){
dt=dt_uu;
  }else{
dt=dt_vv;
  }
}else{
  //U=0でVがある流速を持つ場合
  //dt_vvを採用
  if((dt_uu<=0.)&&(dt_vv>0.)){
```

```
dt=dt_vv;
   }
   //V=0でUがある流速を持つ場合
   //dt_uuを採用
   if((dt_uu>0.)&&(dt_vv<=0.)){
dt=dt_uu;
   }
   //U=0かつV=0の場合:CFL条件に対する制限無し
   //dt_initialを採用
   if((dt_uu<=0.)&&(dt_vv<=0.)){
dt=dt_initial;
   }
 }
 ////////////////////////////////
 //①:CFL条件に関する条件:ここまで
 ////////////////////////////////

 ////////////////////////////////
 //②:拡散方程式に関する条件
 ////////////////////////////////
 // ここに②の条件に関するΔtの決定式を書く
 // x方向の制限
 dt_tmp=VIS_SAFE_COFF*(dx*dx/2./viscosity_coff_mol);
 if(dt>dt_tmp){dt=dt_tmp;}
 // y方向の制限
 dt_tmp=VIS_SAFE_COFF*(dy*dy/2./viscosity_coff_mol);
 if(dt>dt_tmp){dt=dt_tmp;}
 ////////////////////////////////
 //②:拡散方程式に関する条件:ここまで
 ////////////////////////////////

 if((time_step>1)&&(dt>1.1*dt_old)){dt=1.1*dt_old;}

 ////////////////////////////////
 //決定された時間刻みだけ時間を進め
 //これから計算する次時刻tn+1の時刻をtimeで計算
 ////////////////////////////////
 time=time+dt;
```

```
//////////////////////////////
//決定された時間間隔等を画面に出力
//////////////////////////////
printf("#######################################\n");
printf("Time Progress: n=%d -> n+1=%d\n",time_step-1,time_st
printf("Time         : time(tn+1)=%e\n",time);
printf("Time Interval: dt=%e\n",dt);

//////////////////////////////////////////////////////////
//時間刻み幅Δtの決定:ここまで
//////////////////////////////////////////////////////////



//////////////////////////////////////////////////////////
//Step1: 移流相(移流方程式の計算)
//////////////////////////////////////////////////////////

//////////////////////////////////////////////////
//計算前の古い値fnをoldに格納
//////////////////////////////////////////////////
//以後の計算にはoldを用いて
//新しい値は上書きで保存する
//////////////////////////////////

//////////////////////////////////
// Color Functionの古い値
//////////////////////////////////
//注意:仮想セルの値もすべてoldに移す
//     i=0〜nx, j=0〜ny
for(i=0;i<=nx;i++){
  for(j=0;j<=ny;j++){
   old_color_func[i][j]=color_func[i][j];
   old_color_func_gx[i][j]=color_func_gx[i][j];
   old_color_func_gy[i][j]=color_func_gy[i][j];
  }}
//////////////////////////////////
// Color Functionの古い値:ここまで
```

```
//////////////////////////////////

//////////////////////////////////
// 流速Uの古い値
//////////////////////////////////
//注意:仮想セルの値もすべてoldに移す
//     i=0～nx+1, j=0～ny
for(i=0;i<=nx+1;i++){
  for(j=0;j<=ny;j++){
   old_uu[i][j]=uu[i][j];
   old_uu_gx[i][j]=uu_gx[i][j];
   old_uu_gy[i][j]=uu_gy[i][j];
  }}
//////////////////////////////////
// 流速Uの古い値:ここまで
//////////////////////////////////

//////////////////////////////////
// 流速Vの古い値
//////////////////////////////////
//注意:仮想セルの値もすべてoldに移す
//     i=0～nx, j=0～ny+1
for(i=0;i<=nx;i++){
  for(j=0;j<=ny+1;j++){
   old_vv[i][j]=vv[i][j];
   old_vv_gx[i][j]=vv_gx[i][j];
   old_vv_gy[i][j]=vv_gy[i][j];
  }}
//////////////////////////////////
// 流速Vの古い値:ここまで
//////////////////////////////////

//////////////////////////////////////////////////
//計算前の古い値fnをoldに格納:ここまで
//////////////////////////////////////////////////


//////////////////////////////////////////////////
//移流計算の本体
```

```
//////////////////////////////////////////////////

////////////////////////////////
// Color Functionの移流
////////////////////////////////

//////////////////////////////////
// 微分の修正
//////////////////////////////////
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny-1;j++){
     old_color_func_gx[i][j]=color_func_gx[i][j];
     old_color_func_gy[i][j]=color_func_gy[i][j];
}}
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny-1;j++){
   //Color Functionは計算セルの中央
   //速度の微分を中心差分で求める
   dudx=(old_uu[i+1][j]-old_uu[i][j])/dx;
   dvdy=(old_vv[i][j+1]-old_vv[i][j])/dy;
   dudy=(0.5*(old_uu[i][j+1]+old_uu[i+1][j+1])
         -0.5*(old_uu[i][j-1]+old_uu[i+1][j-1]))/(2.*dy);
   dvdx=(0.5*(old_vv[i+1][j]+old_vv[i+1][j+1])
         -0.5*(old_vv[i-1][j]+old_vv[i-1][j+1]))/(2.*dx);
   color_func_gx[i][j]=old_color_func_gx[i][j]
                       -dudx*old_color_func_gx[i][j]*dt
                       -dvdx*old_color_func_gy[i][j]*dt;
   color_func_gy[i][j]=old_color_func_gy[i][j]
                       -dudy*old_color_func_gx[i][j]*dt
                       -dvdy*old_color_func_gy[i][j]*dt;
}}
//////////////////////////////////////////////////
//境界条件の設定
//////////////////////////////////////////////////
boundary( );
//////////////////////////////////////////////////
//境界条件の設定:ここまで
//////////////////////////////////////////////////
```

```
//注意:仮想セルの値以外を計算し、
//      計算結果をold_colorに上書きする
//      i=1〜nx-1, j=1〜ny-1
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny-1;j++){
   //Color Functionは計算セルの中央
   //中央での流速値を計算セルの境界面上に位置する
   //流速uu[i][j],uu[i+1][j],vv[i][j],vv[i][j+1]
   //から平均操作で推定
  uu_tmp=0.5*(old_uu[i][j]+old_uu[i+1][j]);
  vv_tmp=0.5*(old_vv[i][j]+old_vv[i+1][j]);
   //上流側の格子点の決定
  if(uu_tmp>0.){iup=i-1;}else{iup=i+1;}
  if(vv_tmp>0.){jup=j-1;}else{jup=j+1;}
   //移流する距離(上流点への距離)の計算
   udt=-uu_tmp*dt;
  vdt=-vv_tmp*dt;
   //上流側の格子点への距離
   //Δxi->iup=dxiup
  dxiup=xx[iup]-xx[i];
   //Δyj->jup=dyjup
  dyjup=yy[jup]-yy[j];
   //CIP法の計算
   C00=old_color_func[i][j];
   C10=old_color_func_gx[i][j];
   C01=old_color_func_gy[i][j];
   C20=3.*(-old_color_func[i][j]+old_color_func[iup][j])/dxi
       -(2.*old_color_func_gx[i][j]+old_color_func_gx[iup][j
   C02=3.*(-old_color_func[i][j]+old_color_func[i][jup])/dyj
       -(2.*old_color_func_gy[i][j]+old_color_func_gy[i][jup
   C30=2.*(old_color_func[i][j]-old_color_func[iup][j])/dxiu
       +(old_color_func_gx[i][j]+old_color_func_gx[iup][j])/
   C03=2.*(old_color_func[i][j]-old_color_func[i][jup])/dyju
       +(old_color_func_gy[i][j]+old_color_func_gy[i][jup])/
   AA=old_color_func[i][j]-old_color_func[iup][j]-old_color_
olor_func[iup][jup];
   C31=-2.*AA/dxiup/dxiup/dxiup/dyjup
       +(-old_color_func_gx[i][j]-old_color_func_gx[iup][j]
```

```
              +old_color_func_gx[i][jup]+old_color_func_gx[iup][j
yjup;
       C13=-2.*AA/dxiup/dyjup/dyjup/dyjup
           +(-old_color_func_gy[i][j]-old_color_func_gy[i][jup]
              +old_color_func_gy[iup][j]+old_color_func_gy[iup][j
yjup;
       C21=3.*AA/dxiup/dxiup/dyjup
           +(2.*old_color_func_gx[i][j]+old_color_func_gx[iup][j
              -2.*old_color_func_gx[i][jup]-old_color_func_gx[iup
p;
       C12=3.*AA/dxiup/dyjup/dyjup
           +(2.*old_color_func_gy[i][j]+old_color_func_gy[i][jup
              -2.*old_color_func_gy[iup][j]-old_color_func_gy[iup
p;
       C11=-AA/dxiup/dyjup
           +(-old_color_func_gx[i][j]+old_color_func_gx[i][jup])
           +(-old_color_func_gy[i][j]+old_color_func_gy[iup][j])

       color_func[i][j]=C00
                        +C10*udt+C20*udt*udt+C30*udt*udt*udt
                        +C01*vdt+C02*vdt*vdt+C03*vdt*vdt*vdt
                        +C31*udt*udt*udt*vdt+C21*udt*udt*vdt+C11*
t*vdt+C13*udt*vdt*vdt*vdt;
       color_func_gx[i][j]=
                        C10+2.*C20*udt+3.*C30*udt*udt
                        +3.*C31*udt*udt*vdt+2.*C21*udt*vdt+C11*vd
vdt*vdt*vdt;
       color_func_gy[i][j]=
                        C01+2.*C02*vdt+3.*C03*vdt*vdt
                        +C31*udt*udt*udt+C21*udt*udt+C11*udt+2.*C
udt*vdt*vdt;
      }}


    ////////////////////////////////
    // Color Functionの移流:ここまで
    ////////////////////////////////

    ////////////////////////////////////////////////
    //境界条件の設定
```

```
//////////////////////////////////////////////////
boundary( );
//////////////////////////////////////////////////
//境界条件の設定:ここまで
//////////////////////////////////////////////////



////////////////////////////////
// 流速Uの移流
////////////////////////////////

//////////////////////////////////
// 微分の修正
//////////////////////////////////
for(i=1;i<=nx;i++){
  for(j=1;j<=ny-1;j++){
     old_uu_gx[i][j]=uu_gx[i][j];
     old_uu_gy[i][j]=uu_gy[i][j];
}}
for(i=1;i<=nx;i++){
  for(j=1;j<=ny-1;j++){
   //速度の微分を中心差分で求める
   dudx=(old_uu[i+1][j]-old_uu[i-1][j])/(2.*dx);
   dvdy=(0.5*(old_vv[i][j+1]+old_vv[i-1][j+1])
         -0.5*(old_vv[i][j]+old_vv[i-1][j]))/dy;
   dudy=(old_uu[i][j+1]-old_uu[i][j-1])/(2.*dy);
   dvdx=(0.5*(old_vv[i][j]+old_vv[i][j+1])
        -0.5*(old_vv[i-1][j]+old_vv[i-1][j+1]))/(dx);
   uu_gx[i][j]=old_uu_gx[i][j]
               -dudx*old_uu_gx[i][j]*dt
               -dvdx*old_uu_gy[i][j]*dt;
   uu_gy[i][j]=old_uu_gy[i][j]
               -dudy*old_uu_gx[i][j]*dt
               -dvdy*old_uu_gy[i][j]*dt;
}}

//////////////////////////////////////////////////
```

```
//境界条件の設定
//////////////////////////////////////////////////
boundary( );
//////////////////////////////////////////////////
//境界条件の設定:ここまで
//////////////////////////////////////////////////

//注意:仮想セルの値以外を計算し、
//      計算結果をuuに上書きする
//      i=1〜nx, j=1〜ny-1
for(i=1;i<=nx;i++){
  for(j=1;j<=ny-1;j++){
   //uu[i][j]はxi軸の上
   //Vの流速値を付近の４点のVから平均して求める
   //vv[i][j],vv[i][j+1],vv[i-1][j],vv[i-1][j+1]
  uu_tmp=old_uu[i][j];
  vv_tmp=0.25*(old_vv[i][j]+old_vv[i][j+1]
       +old_vv[i-1][j]+old_vv[i-1][j+1]);
   //上流側の格子点の決定
  if(uu_tmp>0.){iup=i-1;}else{iup=i+1;}
  if(vv_tmp>0.){jup=j-1;}else{jup=j+1;}
   //移流する距離(上流点への距離)の計算
  udt=-uu_tmp*dt;
  vdt=-vv_tmp*dt;
   //上流側の格子点への距離
   //Δxi->iup=dxiup
  dxiup=xx[iup]-xx[i];
   //Δyj->jup=dyjup
  dyjup=yy[jup]-yy[j];
   //CIP法の計算
   C00=old_uu[i][j];
   C10=old_uu_gx[i][j];
   C01=old_uu_gy[i][j];
   C20=3.*(-old_uu[i][j]+old_uu[iup][j])/dxiup/dxiup
       -(2.*old_uu_gx[i][j]+old_uu_gx[iup][j])/dxiup;
   C02=3.*(-old_uu[i][j]+old_uu[i][jup])/dyjup/dyjup
       -(2.*old_uu_gy[i][j]+old_uu_gy[i][jup])/dyjup;
   C30=2.*(old_uu[i][j]-old_uu[iup][j])/dxiup/dxiup/dxiup
```

```
            +(old_uu_gx[i][j]+old_uu_gx[iup][j])/dxiup/dxiup;
        C03=2.*(old_uu[i][j]-old_uu[i][jup])/dyjup/dyjup/dyjup
            +(old_uu_gy[i][j]+old_uu_gy[i][jup])/dyjup/dyjup;
        AA=old_uu[i][j]-old_uu[iup][j]-old_uu[i][jup]+old_uu[iup]
        C31=-2.*AA/dxiup/dxiup/dxiup/dyjup
            +(-old_uu_gx[i][j]-old_uu_gx[iup][j]
              +old_uu_gx[i][jup]+old_uu_gx[iup][jup])/dxiup/dxiup
        C13=-2.*AA/dxiup/dyjup/dyjup/dyjup
            +(-old_uu_gy[i][j]-old_uu_gy[i][jup]
              +old_uu_gy[iup][j]+old_uu_gy[iup][jup])/dxiup/dyjup
        C21=3.*AA/dxiup/dxiup/dyjup
            +(2.*old_uu_gx[i][j]+old_uu_gx[iup][j]
              -2.*old_uu_gx[i][jup]-old_uu_gx[iup][jup])/dxiup/dy
        C12=3.*AA/dxiup/dyjup/dyjup
            +(2.*old_uu_gy[i][j]+old_uu_gy[i][jup]
              -2.*old_uu_gy[iup][j]-old_uu_gy[iup][jup])/dxiup/dy
        C11=-AA/dxiup/dyjup
            +(-old_uu_gx[i][j]+old_uu_gx[i][jup])/dyjup
            +(-old_uu_gy[i][j]+old_uu_gy[iup][j])/dxiup;

        uu[i][j]=C00
                +C10*udt+C20*udt*udt+C30*udt*udt*udt
                +C01*vdt+C02*vdt*vdt+C03*vdt*vdt*vdt
                +C31*udt*udt*udt*vdt+C21*udt*udt*vdt+C11*udt*vdt+
3*udt*vdt*vdt*vdt;
        uu_gx[i][j]=
                 C10+2.*C20*udt+3.*C30*udt*udt
                 +3.*C31*udt*udt*vdt+2.*C21*udt*vdt+C11*vdt+C12*v
*vdt;
        uu_gy[i][j]=
                 C01+2.*C02*vdt+3.*C03*vdt*vdt
                 +C31*udt*udt*udt+C21*udt*udt+C11*udt+2.*C12*udt*
*vdt;
      }}



    ////////////////////////////////
    // 流速Uの移流:ここまで
    ////////////////////////////////
```

```
//////////////////////////////////
// 流速Vの移流
//////////////////////////////////

////////////////////////////////////
// 微分の修正
////////////////////////////////////
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny;j++){
     old_vv_gx[i][j]=vv_gx[i][j];
     old_vv_gy[i][j]=vv_gy[i][j];
}}
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny;j++){
   //速度の微分を中心差分で求める
   dudx=(0.5*(old_uu[i+1][j]+old_uu[i+1][j-1])
        -0.5*(old_uu[i][j]+old_uu[i][j-1]))/dx;
   dvdy=(old_vv[i][j+1]-old_vv[i][j-1])/(2.*dx);
   dudy=(0.5*(old_uu[i][j+1]+old_uu[i+1][j+1])
        -0.5*(old_uu[i][j]+old_uu[i+1][j]))/dy;
   dvdx=(old_vv[i+1][j]-old_vv[i-1][j])/(2.*dx);
   vv_gx[i][j]=old_vv_gx[i][j]
               -dudx*old_vv_gx[i][j]*dt
               -dvdx*old_vv_gy[i][j]*dt;
   vv_gy[i][j]=old_vv_gy[i][j]
               -dudy*old_vv_gx[i][j]*dt
               -dvdy*old_vv_gy[i][j]*dt;
}}

//////////////////////////////////////////////////
//境界条件の設定
//////////////////////////////////////////////////
boundary( );
//////////////////////////////////////////////////
//境界条件の設定:ここまで
//////////////////////////////////////////////////
```

```
//注意:仮想セルの値以外を計算し、
//      計算結果をvvに上書きする
//      i=1～nx-1, j=1～ny
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny;j++){
   //vv[i][j]はyj軸の上
   //Uの流速値を付近の４点のUから平均して求める
   //uu[i][j],uu[i+1][j],uu[i][j-1],uu[i+1][j-1]
  uu_tmp=0.25*(old_uu[i][j]+old_uu[i+1][j]
       +old_uu[i][j-1]+old_uu[i+1][j-1]);
  vv_tmp=old_vv[i][j];
   //上流側の格子点の決定
  if(uu_tmp>0.){iup=i-1;}else{iup=i+1;}
  if(vv_tmp>0.){jup=j-1;}else{jup=j+1;}
   //移流する距離(上流点への距離)の計算
  udt=-uu_tmp*dt;
  vdt=-vv_tmp*dt;
   //上流側の格子点への距離
   //Δxi->iup=dxiup
  dxiup=xx[iup]-xx[i];
   //Δyj->jup=dyjup
  dyjup=yy[jup]-yy[j];
   //CIP法の計算
   C00=old_vv[i][j];
   C10=old_vv_gx[i][j];
   C01=old_vv_gy[i][j];
   C20=3.*(-old_vv[i][j]+old_vv[iup][j])/dxiup/dxiup
       -(2.*old_vv_gx[i][j]+old_vv_gx[iup][j])/dxiup;
   C02=3.*(-old_vv[i][j]+old_vv[i][jup])/dyjup/dyjup
       -(2.*old_vv_gy[i][j]+old_vv_gy[i][jup])/dyjup;
   C30=2.*(old_vv[i][j]-old_vv[iup][j])/dxiup/dxiup/dxiup
       +(old_vv_gx[i][j]+old_vv_gx[iup][j])/dxiup/dxiup;
   C03=2.*(old_vv[i][j]-old_vv[i][jup])/dyjup/dyjup/dyjup
       +(old_vv_gy[i][j]+old_vv_gy[i][jup])/dyjup/dyjup;
   AA=old_vv[i][j]-old_vv[iup][j]-old_vv[i][jup]+old_vv[iup]
   C31=-2.*AA/dxiup/dxiup/dxiup/dyjup
       +(-old_vv_gx[i][j]-old_vv_gx[iup][j]
         +old_vv_gx[i][jup]+old_vv_gx[iup][jup])/dxiup/dxiup
   C13=-2.*AA/dxiup/dyjup/dyjup/dyjup
```

```
            +(-old_vv_gy[i][j]-old_vv_gy[i][jup]
              +old_vv_gy[iup][j]+old_vv_gy[iup][jup])/dxiup/dyjup
        C21=3.*AA/dxiup/dxiup/dyjup
            +(2.*old_vv_gx[i][j]+old_vv_gx[iup][j]
              -2.*old_vv_gx[i][jup]-old_vv_gx[iup][jup])/dxiup/dy
        C12=3.*AA/dxiup/dyjup/dyjup
            +(2.*old_vv_gy[i][j]+old_vv_gy[i][jup]
              -2.*old_vv_gy[iup][j]-old_vv_gy[iup][jup])/dxiup/dy
        C11=-AA/dxiup/dyjup
            +(-old_vv_gx[i][j]+old_vv_gx[i][jup])/dyjup
            +(-old_vv_gy[i][j]+old_vv_gy[iup][j])/dxiup;

        vv[i][j]=C00
                +C10*udt+C20*udt*udt+C30*udt*udt*udt
                +C01*vdt+C02*vdt*vdt+C03*vdt*vdt*vdt
                +C31*udt*udt*udt*vdt+C21*udt*udt*vdt+C11*udt*vdt+
3*udt*vdt*vdt*vdt;
        vv_gx[i][j]=
                 C10+2.*C20*udt+3.*C30*udt*udt
                 +3.*C31*udt*udt*vdt+2.*C21*udt*vdt+C11*vdt+C12*v
*vdt;
        vv_gy[i][j]=
                 C01+2.*C02*vdt+3.*C03*vdt*vdt
                 +C31*udt*udt*udt+C21*udt*udt+C11*udt+2.*C12*udt*
*vdt;
      }}

    ////////////////////////////////
    // 流速Vの移流:ここまで
    ////////////////////////////////

    ////////////////////////////////////////////////
    //境界条件の設定
    ////////////////////////////////////////////////
    boundary( );
    ////////////////////////////////////////////////
    //境界条件の設定:ここまで
    ////////////////////////////////////////////////
```

```
//////////////////////////////////////////////////////////
//Step 2 : 拡散(粘性)の計算
//////////////////////////////////////////////////////////
//////////////////////////////////////////////////
//計算前の古い値fnをoldに格納
//////////////////////////////////////////////////
//以後の計算にはoldを用いて
//新しい値は上書きで保存する
//////////////////////////////////
//////////////////////////////////
// 流速Uの古い値
//////////////////////////////////
//注意:仮想セルの値もすべてoldに移す
//     i=0〜nx+1, j=0〜ny
for(i=0;i<=nx+1;i++){
  for(j=0;j<=ny;j++){
   old_uu[i][j]=uu[i][j];
   old_uu_gx[i][j]=uu_gx[i][j];
   old_uu_gy[i][j]=uu_gy[i][j];
  }}
//////////////////////////////////
// 流速Uの古い値:ここまで
//////////////////////////////////
//////////////////////////////////
// 流速Vの古い値
//////////////////////////////////
//注意:仮想セルの値もすべてoldに移す
//     i=0〜nx, j=0〜ny+1
for(i=0;i<=nx;i++){
  for(j=0;j<=ny+1;j++){
   old_vv[i][j]=vv[i][j];
   old_vv_gx[i][j]=vv_gx[i][j];
   old_vv_gy[i][j]=vv_gy[i][j];
  }}
//////////////////////////////////
// 流速Vの古い値:ここまで
//////////////////////////////////
```

```
    ////////////////////////////////////////////////
    //計算前の古い値fnをoldに格納:ここまで
    ////////////////////////////////////////////////


    ////////////////////////////////////////////////
    //陽解法+中心差分で解く
    ////////////////////////////////////////////////
    ////////////////////////////////
    // 流速Uの拡散(粘性)
    ////////////////////////////////
    //注意:仮想セルの値以外を計算し、
    //      計算結果をuuに上書きする
    //      i=1〜nx, j=1〜ny-1
    for(i=1;i<=nx;i++){
      for(j=1;j<=ny-1;j++){
         uu[i][j]=old_uu[i][j]
          +viscosity_coff_mol*(old_uu[i+1][j]-2.*old_uu[i][j]+ol
x*dt
          +viscosity_coff_mol*(old_uu[i][j+1]-2.*old_uu[i][j]+ol
y*dt;
    }}
    ////////////////////////////////////////////////
    //境界条件の設定
    ////////////////////////////////////////////////
    boundary( );
    ////////////////////////////////////////////////
    //境界条件の設定:ここまで
    ////////////////////////////////////////////////
    ////////////////////////////////
    // 流速Uの微分の修正
    ////////////////////////////////
    for(i=1;i<=nx;i++){
      for(j=1;j<=ny-1;j++){
         uu_gx[i][j]=old_uu_gx[i][j]
          +(uu[i+1][j]-uu[i-1][j]-old_uu[i+1][j]+old_uu[i-1][j])
         uu_gy[i][j]=old_uu_gy[i][j]
          +(uu[i][j+1]-uu[i][j-1]-old_uu[i][j+1]+old_uu[i][j-1])
    }}
```

```
    ////////////////////////////////////////////////
    //境界条件の設定
    ////////////////////////////////////////////////
    boundary( );
    ////////////////////////////////////////////////
    //境界条件の設定:ここまで
    ////////////////////////////////////////////////


    ////////////////////////////////
    // 流速Vの拡散(粘性)
    ////////////////////////////////
    //注意:仮想セルの値以外を計算し、
    //      計算結果をvvに上書きする
    //      i=1〜nx-1, j=1〜ny
    for(i=1;i<=nx-1;i++){
      for(j=1;j<=ny;j++){
         vv[i][j]=old_vv[i][j]
          +viscosity_coff_mol*(old_vv[i+1][j]-2.*old_vv[i][j]+ol
x*dt
          +viscosity_coff_mol*(old_vv[i][j+1]-2.*old_vv[i][j]+ol
y*dt
          +gravity*dt;
    }}

    ////////////////////////////////////////////////
    //境界条件の設定
    ////////////////////////////////////////////////
    boundary( );
    ////////////////////////////////////////////////
    //境界条件の設定:ここまで
    ////////////////////////////////////////////////
    ////////////////////////////////
    // 流速Vの微分の修正
    ////////////////////////////////
    for(i=1;i<=nx-1;i++){
      for(j=1;j<=ny;j++){
         vv_gx[i][j]=old_vv_gx[i][j]
          +(vv[i+1][j]-vv[i-1][j]-old_vv[i+1][j]+old_vv[i-1][j])
```

```
     vv_gy[i][j]=old_vv_gy[i][j]
      +(vv[i][j+1]-vv[i][j-1]-old_vv[i][j+1]+old_vv[i][j-1])
}}
//////////////////////////////////////////////////
//境界条件の設定
//////////////////////////////////////////////////
boundary( );
//////////////////////////////////////////////////
//境界条件の設定:ここまで
//////////////////////////////////////////////////



////////////////////////////////////////////////////////////
// Step3:圧力修正相
////////////////////////////////////////////////////////////
// 密度の分布を推定
// color_func[i][j]>0.5: 水
// color_func[i][j]<0.5:空気
// 仮想セルでの値も推定
for(i=0;i<=nx;i++){
  for(j=0;j<=ny;j++){
     if(color_func[i][j]>0.5){density[i][j]=density_water;}
     else{density[i][j]=density_air;}

}}
//////////////////////////////////////
// 密度の設定ここまで
//////////////////////////////////////

//////////////////////////////////////
// 速度の発散を計算
// divU[i][j]=(du[i][j]/dx+dv[i][j]/dy)
//////////////////////////////////////
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny-1;j++){
     divU[i][j]=(uu[i+1][j]-uu[i][j])/dx+(vv[i][j+1]-vv[i][j
}}
//////////////////////////////////////
```

```
// 速度の発散ここまで
//////////////////////////////////////

//////////////////////////////////////////////////
//SOR法による圧力の決定
//////////////////////////////////////////////////
//////////////////////////////////
// 圧力の初期値の設定
//////////////////////////////////
if(time_step==1){
  for(i=1;i<=nx-1;i++){
    for(j=1;j<=ny-1;j++){
       pressure[i][j]=1.;
  }}
}
//////////////////////////////////
// 圧力の初期値の設定:ここまで
//////////////////////////////////

//////////////////////////////////
// SOR の本体
//////////////////////////////////
// SORの緩和計算のループの最大値
// (最大反復数)
sor_step_max=2000;
// SORの緩和計算の収束判定条件
sor_epsiron=1.e-5;
// SORの緩和係数ω
sor_omega=1.981;

for(sor_step=1;sor_step<=sor_step_max;sor_step++){

  ////////////////////////////////////////
  // 境界条件
  ////////////////////////////////////////
  boundary_pressure( );


  ////////////////////////////////////////
```

```
// 緩和計算の本体
////////////////////////////////////////

//収束条件を満たしている格子点の数をリセット
sor_conv_num=0;

pressure_min=1.e+32;
pressure_max=-1.e+32;
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny-1;j++){
     if(pressure[i][j]>pressure_max){pressure_max=pressure
     if(pressure[i][j]<pressure_min){pressure_min=pressure
}}

for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny-1;j++){
     // 圧力の古い値を保存
     pressure_old=pressure[i][j];
     // 隣接４点の圧力の値
     pressure_iupj=pressure[i+1][j];
     pressure_idnj=pressure[i-1][j];
     pressure_ijup=pressure[i][j+1];
     pressure_ijdn=pressure[i][j-1];
     // 格子セルの境界上の密度
     // density[i+1/2][j]
     density_iupj=0.5*(density[i+1][j]+density[i][j]);
     // density[i-1/2][j]
     density_idnj=0.5*(density[i-1][j]+density[i][j]);
     // density[i][j+1/2]
     density_ijup=0.5*(density[i][j+1]+density[i][j]);
     // density[i][j-1/2]
     density_ijdn=0.5*(density[i][j-1]+density[i][j]);
     // 修正量δ

     sor_delta=(divU[i][j]/dt
               -pressure_iupj/dx/dx/density_iupj
               -pressure_idnj/dx/dx/density_idnj
               -pressure_ijup/dy/dy/density_ijup
               -pressure_ijdn/dy/dy/density_ijdn
```

```
                    )
                    /
                    (-1./dx/dx/density_iupj-1./dx/dx/density_ic
                     -1./dy/dy/density_ijup-1./dy/dy/density_ij
                    -pressure_old;
            // ω倍して修正
            pressure[i][j]=pressure_old+sor_omega*sor_delta;

            // 収束判定
            if(fabs(pressure[i][j]-pressure_old)<=sor_epsiron*fc
essure_min)){
              sor_conv_num=sor_conv_num+1;
             }
      }}
      ////////////////////////////////////////
      // 緩和計算の本体(1ステップ):ここまで
      ////////////////////////////////////////

      ////////////////////////////////////////
      // 境界条件
      ////////////////////////////////////////
      boundary_pressure( );


      ////////////////////////////////////////
      // 収束判定
      ////////////////////////////////////////
      // 全ての格子点上の圧力が収束条件
      // |fnew-fold|/|fold|<ε
      // を満たしていればSOR法の緩和ループを抜ける
      ////////////////////////////////////////
      if(sor_conv_num==(nx-1)*(ny-1)){
         printf("SOR : Converging\n");
         printf("SOR : End Step =%d\n",sor_step);
         goto mark_sor_end;
      }

    }
    // SOR法で収束しない場合に以下の一文が実行される。
```

```
printf("SOR : Not Converging\n");

mark_sor_end:;
//////////////////////////////////
// SOR の本体:ここまで
//////////////////////////////////



//////////////////////////////////////////////////
// SOR法で決定された圧力を用いて流速を更新する
//////////////////////////////////////////////////

//////////////////////////////////////////////////
//計算前の古い値fnをoldに格納
//////////////////////////////////////////////////
//以後の計算にはoldを用いて
//新しい値は上書きで保存する
//////////////////////////////////
//////////////////////////////////
// 流速Uの古い値
//////////////////////////////////
//注意:仮想セルの値もすべてoldに移す
//     i=0〜nx+1, j=0〜ny
for(i=0;i<=nx+1;i++){
  for(j=0;j<=ny;j++){
   old_uu[i][j]=uu[i][j];
   old_uu_gx[i][j]=uu_gx[i][j];
   old_uu_gy[i][j]=uu_gy[i][j];
  }}
//////////////////////////////////
// 流速Uの古い値:ここまで
//////////////////////////////////
//////////////////////////////////
// 流速Vの古い値
//////////////////////////////////
//注意:仮想セルの値もすべてoldに移す
//     i=0〜nx, j=0〜ny+1
```

```
for(i=0;i<=nx;i++){
  for(j=0;j<=ny+1;j++){
   old_vv[i][j]=vv[i][j];
   old_vv_gx[i][j]=vv_gx[i][j];
   old_vv_gy[i][j]=vv_gy[i][j];
  }}
//////////////////////////////////
// 流速Vの古い値:ここまで
//////////////////////////////////
////////////////////////////////////////////////////
//計算前の古い値fnをoldに格納:ここまで
////////////////////////////////////////////////////


////////////////////////////////////////////////////
//中心差分で解く
////////////////////////////////////////////////////
//////////////////////////////////
// 流速Uの加速
//////////////////////////////////
//注意:仮想セルの値以外を計算し、
//      計算結果をuuに上書きする
//      i=1〜nx, j=1〜ny-1
for(i=1;i<=nx;i++){
  for(j=1;j<=ny-1;j++){
     density_idnj=0.5*(density[i-1][j]+density[i][j]);
     uu[i][j]=old_uu[i][j]
      -(pressure[i][j]-pressure[i-1][j])/dx/density_idnj*dt;
}}
////////////////////////////////////////////////////
//境界条件の設定
////////////////////////////////////////////////////
boundary( );
////////////////////////////////////////////////////
//境界条件の設定:ここまで
////////////////////////////////////////////////////
//////////////////////////////////
// 流速Uの微分の修正
//////////////////////////////////
```

```
for(i=1;i<=nx;i++){
  for(j=1;j<=ny-1;j++){
     uu_gx[i][j]=old_uu_gx[i][j]
      +(uu[i+1][j]-uu[i-1][j]-old_uu[i+1][j]+old_uu[i-1][j])
     uu_gy[i][j]=old_uu_gy[i][j]
      +(uu[i][j+1]-uu[i][j-1]-old_uu[i][j+1]+old_uu[i][j-1])
}}
//////////////////////////////////////////////////
//境界条件の設定
//////////////////////////////////////////////////
boundary( );
//////////////////////////////////////////////////
//境界条件の設定:ここまで
//////////////////////////////////////////////////


//////////////////////////////////
// 流速Vの加速
//////////////////////////////////
//注意:仮想セルの値以外を計算し、
//      計算結果をvvに上書きする
//      i=1～nx-1, j=1～ny
for(i=1;i<=nx-1;i++){
  for(j=1;j<=ny;j++){
     density_ijdn=0.5*(density[i][j-1]+density[i][j]);
     vv[i][j]=old_vv[i][j]
      -(pressure[i][j]-pressure[i][j-1])/dy/density_ijdn*dt;
}}
//////////////////////////////////////////////////
//境界条件の設定
//////////////////////////////////////////////////
boundary( );
//////////////////////////////////////////////////
//境界条件の設定:ここまで
//////////////////////////////////////////////////
//////////////////////////////////
// 流速Vの微分の修正
//////////////////////////////////
for(i=1;i<=nx-1;i++){
```

```
    for(j=1;j<=ny;j++){
       vv_gx[i][j]=old_vv_gx[i][j]
        +(vv[i+1][j]-vv[i-1][j]-old_vv[i+1][j]+old_vv[i-1][j])
       vv_gy[i][j]=old_vv_gy[i][j]
        +(vv[i][j+1]-vv[i][j-1]-old_vv[i][j+1]+old_vv[i][j-1])
  }}
  ////////////////////////////////////////////////
  //境界条件の設定
  ////////////////////////////////////////////////
  boundary( );
  ////////////////////////////////////////////////
  //境界条件の設定:ここまで
  ////////////////////////////////////////////////



  ////////////////////////////////////////////////////////////
  //途中結果の出力
  ////////////////////////////////////////////////////////////
  //時刻が次に出力すべき時刻output_timeを超えたら出力
  if(time>=output_time){
    //出力ファイルの通し番号を+1する
    output_file_number=output_file_number+1;
    //次に出力する時刻に更新
    output_time=output_time+output_time_dt;

    ////////////////////////////////
    // Color Function 出力
    ////////////////////////////////
    // Color_func_xxxx.datというファイル名で出力
    // xxxxの部分は通し番号がつく
    sprintf(filename,"Color_func_%4.4d.dat",output_file_number
    fp=fopen(filename,"w");
    fprintf(fp,"xx[i] yy[j] color_func[i][j] color_func_gx[i][
][j]\n");
    for(i=1;i<=nx-1;i++){
     for(j=1;j<=ny-1;j++){
       fprintf(fp,"%e %e %e %e %e\n",xx[i],yy[j],
       color_func[i][j],color_func_gx[i][j],color_func_gy[i][j
```

```
 }
 fprintf(fp,"\n");
}
fclose(fp);
//////////////////////////////////
// Color Function 出力:ここまで
//////////////////////////////////

//////////////////////////////////
// 流速U 出力
//////////////////////////////////
// UU_xxxx.datというファイル名で出力
// xxxxの部分は通し番号がつく
sprintf(filename,"UU_%4.4d.dat",output_file_number);
fp=fopen(filename,"w");
fprintf(fp,"xx[i] yy[j] uu[i][j] uu_gx[i][j] uu_gy[i][j]\r
for(i=1;i<=nx;i++){
 for(j=1;j<=ny-1;j++){
   fprintf(fp,"%e %e %e %e %e\n",xx[i],yy[j],
       uu[i][j],uu_gx[i][j],uu_gy[i][j]);
 }
 fprintf(fp,"\n");
}
fclose(fp);

//////////////////////////////////
// 流速U 出力:ここまで
//////////////////////////////////

//////////////////////////////////
// 流速V 出力
//////////////////////////////////
// VV_xxxx.datというファイル名で出力
// xxxxの部分は通し番号がつく
sprintf(filename,"VV_%4.4d.dat",output_file_number);
fp=fopen(filename,"w");
fprintf(fp,"xx[i] yy[j] vv[i][j] vv_gx[i][j] vv_gy[i][j]\r
for(i=1;i<=nx-1;i++){
 for(j=1;j<=ny;j++){
```

```
   fprintf(fp,"%e %e %e %e %e\n",xx[i],yy[j],
      vv[i][j],vv_gx[i][j],vv_gy[i][j]);
 }
 fprintf(fp,"\n");
}
fclose(fp);
//////////////////////////////////
// 流速V 出力:ここまで
//////////////////////////////////


//////////////////////////////////
// 密度 出力
//////////////////////////////////
// DENSITY_xxxx.datというファイル名で出力
// xxxxの部分は通し番号がつく
sprintf(filename,"DENSITY_%4.4d.dat",output_file_number);
fp=fopen(filename,"w");
fprintf(fp,"xx[i] yy[j] density[i][j]\n");
for(i=1;i<=nx-1;i++){
 for(j=1;j<=ny-1;j++){
   fprintf(fp,"%e %e %e\n",xx[i],yy[j],
      density[i][j]);
 }
 fprintf(fp,"\n");
}
fclose(fp);
//////////////////////////////////
// 密度 出力:ここまで
//////////////////////////////////

//////////////////////////////////
// 圧力 出力
//////////////////////////////////
// PRESSURE_xxxx.datというファイル名で出力
// xxxxの部分は通し番号がつく
sprintf(filename,"PRESSURE_%4.4d.dat",output_file_number);
fp=fopen(filename,"w");
fprintf(fp,"xx[i] yy[j] pressure[i][j]\n");
```

```
  for(i=1;i<=nx-1;i++){
   for(j=1;j<=ny-1;j++){
     fprintf(fp,"%e %e %e\n",xx[i],yy[j],
     pressure[i][j]);
   }
   fprintf(fp,"\n");
  }
  fclose(fp);
  ////////////////////////////////
  // 圧力 出力:ここまで
  ////////////////////////////////

  ////////////////////////////////
  // 流速ベクトル 出力
  ////////////////////////////////
  // VECTOR_xxxx.datというファイル名で出力
  // xxxxの部分は通し番号がつく
  sprintf(filename,"VECTOR_%4.4d.dat",output_file_number);
  fp=fopen(filename,"w");
  fprintf(fp,"xx[i] yy[j] uu[i][j] vv[i][j] [/10]\n");
  for(i=1;i<=nx-1;i++){
   for(j=1;j<=ny-1;j++){
     fprintf(fp,"%e %e %e %e\n",0.5*(xx[i]+xx[i+1]),0.5*(yy[
        0.5*(uu[i][j]+uu[i+1][j])/100.,0.5*(vv[i][j]+vv[i][
   }
   fprintf(fp,"\n");
  }
  fclose(fp);

  ////////////////////////////////
  // 流速U 出力:ここまで
  ////////////////////////////////

}
//////////////////////////////////////////////////////////
//途中結果の出力:ここまで
//////////////////////////////////////////////////////////
```

```
  }
  ////////////////////////////////////////////////////////////////
  //メインループ:時間発展を計算する本体:ここまで
  ////////////////////////////////////////////////////////////////



  ////////////////////////////////////////////////////////////////
  //プログラムの終了
  ////////////////////////////////////////////////////////////////

  return 0;
}




////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//サブルーチンboundaryの定義
////////////////////////////////////////////////////////////////

int boundary( )
{

  ////////////////////////////////////////////
  // 境界I(Y=Ymax)の仮想セルの設定
  // 自由境界
  ////////////////////////////////////////////
  for(i=0;i<=nx;i++){
    color_func[i][ny]=color_func[i][ny-1];
    color_func_gx[i][ny]=color_func_gx[i][ny-1];
    color_func_gy[i][ny]=0.;
  }
  for(i=0;i<=nx+1;i++){
```

```
  uu[i][ny]=uu[i][ny-1];
  uu_gx[i][ny]=uu_gx[i][ny-1];
  uu_gy[i][ny]=0.;
}
for(i=0;i<=nx;i++){
  vv[i][ny+1]=vv[i][ny];
  vv_gx[i][ny+1]=vv_gx[i][ny];
  vv_gy[i][ny+1]=0.;
}
//////////////////////////////////////////
// 境界I(Y=Ymax)の仮想セルの設定:ここまで
//////////////////////////////////////////

//////////////////////////////////////////
// 境界II(X=0)の仮想セルの設定
// 壁面境界
//////////////////////////////////////////
for(j=0;j<=ny;j++){
  color_func[0][j]=color_func[1][j];
  color_func_gx[0][j]=-color_func_gx[1][j];
  color_func_gy[0][j]=color_func_gy[1][j];
}
for(j=0;j<=ny;j++){
  //壁面上でUは0
  uu[1][j]=0.;
  uu[0][j]=-uu[2][j];
  uu_gx[0][j]=uu_gx[2][j];
  uu_gy[0][j]=-uu_gy[2][j];
}
//すべり無し境界
for(j=0;j<=ny+1;j++){
  vv[0][j]=-vv[1][j];
  vv_gx[0][j]=vv_gx[1][j];
  vv_gy[0][j]=-vv_gy[1][j];
}
//////////////////////////////////////////
// 境界II(X=0)の仮想セルの設定:ここまで
//////////////////////////////////////////
```

```
//////////////////////////////////////////
// 境界III(Y=0)の仮想セルの設定
// 壁面境界
//////////////////////////////////////////
for(i=0;i<=nx;i++){
  color_func[i][0]=color_func[i][1];
  color_func_gx[i][0]=color_func_gx[i][1];
  color_func_gy[i][0]=0.;
}
//すべり無し境界
for(i=0;i<=nx+1;i++){
  uu[i][0]=-uu[i][1];
  uu_gx[i][0]=-uu_gx[i][1];
  uu_gy[i][0]=uu_gy[i][1];
}
for(i=0;i<=nx;i++){
  //壁面上でV=0
  vv[i][1]=0.;
  vv[i][0]=-vv[i][2];
  vv_gx[i][0]=-vv_gx[i][2];
  vv_gy[i][0]=vv_gy[i][2];
}
//////////////////////////////////////////
// 境界III(Y=0)の仮想セルの設定:ここまで
//////////////////////////////////////////

//////////////////////////////////////////
// 境界IV(X=Xmax)の仮想セルの設定
// 壁面境界
//////////////////////////////////////////
for(j=0;j<=ny;j++){
  color_func[nx][j]=color_func[nx-1][j];
  color_func_gx[nx][j]=-color_func_gx[nx-1][j];
  color_func_gy[nx][j]=color_func_gy[nx-1][j];
}
for(j=0;j<=ny;j++){
  //壁面上でU=0
  uu[nx][j]=0.;
  uu[nx+1][j]=-uu[nx-1][j];
```

```
    uu_gx[nx+1][j]=uu_gx[nx-1][j];
    uu_gy[nx+1][j]=-uu_gy[nx-1][j];
  }
  //すべり無し境界
  for(j=0;j<=ny+1;j++){
    vv[nx][j]=-vv[nx-1][j];
    vv_gx[nx][j]=vv_gx[nx-1][j];
    vv_gy[nx][j]=-vv_gy[nx-1][j];
  }
  ////////////////////////////////////////
  // 境界IV(X=Xmax)の仮想セルの設定:ここまで
  ////////////////////////////////////////


  return 0;


}


////////////////////////////////////////////////////////////////
//サブルーチンboundaryの定義:ここまで
////////////////////////////////////////////////////////////////





////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
//サブルーチンboundary_pressureの定義
////////////////////////////////////////////////////////////////

int boundary_pressure( )
{

  // 圧力の境界値(Y=Ymax)の設定
```

```
pressure_boundary=1.;

//////////////////////////////////////////
// 境界I(Y=Ymax)の仮想セルの設定
// 圧力はpressure_boundaryに一致
//////////////////////////////////////////
for(i=0;i<=nx;i++){
  pressure[i][ny]=pressure_boundary;
}
//////////////////////////////////////////
// 境界I(Y=Ymax)の仮想セルの設定:ここまで
//////////////////////////////////////////

//////////////////////////////////////////
// 境界II(X=0)の仮想セルの設定
// 壁面境界
//////////////////////////////////////////
for(j=0;j<=ny;j++){
  pressure[0][j]=pressure[1][j];
}
//////////////////////////////////////////
// 境界II(X=0)の仮想セルの設定:ここまで
//////////////////////////////////////////

//////////////////////////////////////////
// 境界III(Y=0)の仮想セルの設定
// 壁面境界
//////////////////////////////////////////
for(i=0;i<=nx;i++){
  pressure[i][0]=pressure[i][1];
}
//////////////////////////////////////////
// 境界III(Y=0)の仮想セルの設定:ここまで
//////////////////////////////////////////

//////////////////////////////////////////
// 境界IV(X=Xmax)の仮想セルの設定
// 壁面境界
```

```
  ////////////////////////////////////////////
  for(j=0;j<=ny;j++){
    pressure[nx][j]=pressure[nx-1][j];
  }
  ////////////////////////////////////////////
  // 境界IV(X=Xmax)の仮想セルの設定:ここまで
  ////////////////////////////////////////////


  return 0;


}


////////////////////////////////////////////////////////////////////
//サブルーチンboundary_pressureの定義:ここまで
////////////////////////////////////////////////////////////////////
```