Titulació:

Grau d'Enginyeria en Tecnologies Aeroespacials

Alumne (nom i cognoms):

Mario Gayete Ibáñez

Enunciat TFG:

Development of CFD codes for the numerical resolution of potential flow and the incompressible form of the Navier-Stokes equations

Contingut:

Annexos

Director TFG:

Carles-David Pérez-Segarra

Codirector del TFG:

Asensio Oliva Llena

Convocatòria del lluirament del TFG:

10/06/2019

Contents

Appendix A

Developed code for Potential Flow (streamline method) imposing the circulation

```
//  Potential flow Numerical solver  (imposed circulation condition) -streamlines
    ↪ method
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <time.h>
using namespace std;
// Constant numbers
const double pi=3.141592;
//************************************************************
// Constants
const double H=5, L=10; //Problem's geometry
const double Rg=287; // gas constant
const double v_in=10, P_in=1.013e5, T_in=288, rho_in=P_in/Rg/T_in; // Inlet flow
    ↪ parameters
const int N=500, M=N; // Mesh density
const double fr=1.8,delta=1e-8; // Program parameters (Gauss_Seidel: fr=1.9 //
    ↪ line-by-line: fr=1.1)
const double Nd=5; // division to estimate the integral
double phi_c=v_in*H/2;
// Cylinder parameters
const double R=0.3, cx=L/2, cy=H/2; // Cylinder's geometry
const double w=10; //Cylinder's rotation
// Desired circulation around the object
double circulacio_final;
const double error_circ=1e-5;
// Characteristic lenght
double D_main;
// Gas cp treatemet: cp(T)=a0*T^0+a1*T^1+a2*T2+a3*T^3+a4*T^4
// Air
const double a0=1034.09, a1=-2.849e-1, a2=7.817e-4, a3=-4.971e-7, a4=1.077e-10;
// Helium
//const double a0=5188, a1=0, a2=0, a3=0, a4=0;
```

```
// Solver type: line_by_line or Gauss-Seidel
bool line_by_line=false;
// Convergence analysis point
double x_conv=5, y_conv=2.9;
int i_conv, j_conv;
//*************************************************************
// Variables declaration
time_t inici,final;
struct info_node{
 double r[2];
 double phi;
 double phi_ant;
 double rho;
 double rho_ant;
 double tau;
 double T;
 double T_ant;
 double P;
 double P_ant;
 double Ax;
 double Ay;
 double cp_barra;
 double cp_barret;
 double gamma_barret;
 double vel[2]; // x and y velocity components
 double v; // velocity modulus
 int mat; // 0 solid, 1 fluid
 double ap;
 double as;
 double an;
 double aw;
 double ae;
 double bp;
 bool frontera; // determines if the node is close to the object
 double t[2]; // vtangential vector
 double n[2]; // normal vector
 double circ; //control volume circulation
};
// Functions
void preprocess(vector< vector<info_node> >& node);
void geometria(vector< vector<info_node> >& node);
int num_material(double x, double y);
void condicions_inlet(vector< vector<info_node> >& node);
void condicions_contorn(vector< vector<info_node> >& node);
void mapa_inicial(vector< vector<info_node> >& node);
void calcul_coefs(vector< vector<info_node> >& node);
double mitjana_harm(double tau1, double tau2,double d1, double d2);
void solver(vector< vector<info_node> >& node);
void solver_phi(vector< vector<info_node> >& node);
void calcul_vel(vector< vector<info_node> >& node);
double calcul_cp_barra(double T2, double T1);
double calcul_cp(double T);
double calcul_gamma(double cp);
double calcul_cp_barret(double T0, double T);
void calcul_prop_term(vector< vector<info_node> >& node);
double calcul_error_max(vector< vector<info_node> >& node);
double v_abs(double a);
void iteracio (vector< vector<info_node> >& node);
void save_phi (vector< vector<info_node> >& node);
void save_rho (vector< vector<info_node> >& node);
void save_T (vector< vector<info_node> >& node);
void save_P (vector< vector<info_node> >& node);
```

```
void save_v (vector< vector<info_node> >& node);
void save_tau (vector< vector<info_node> >& node);
void postprocess(vector< vector<info_node> >& node);
void save_posicio_mat(vector< vector<info_node> >& node);
void calcul_error(vector< vector<info_node> >& node);
void det_frontera(vector< vector<info_node> >& node);
void calcul_circulacio(vector< vector<info_node> >& node,double &circulacio);
void calcul_forces(vector< vector<info_node> >& node,double circulacio);
void guardar_dades(double L,double CL, double D, double CD,double circulacio);
void Newton_Raphson(vector< vector<info_node> >& node, bool &fi,double &phi_prev,
    ↪ double &circulacio_prev);
void canvi_phi_mat(vector< vector<info_node> >& node);
void declaracio_inicial();
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2]);
void find_position(vector< vector<info_node> >& node);
// Code
int main()
{

 time(&inici);
 declaracio_inicial();
 cout<<"Inici"<<endl;
 vector< vector<info_node> > node(N+2, vector<info_node>(M+2));
 bool fi=false;
 double phi_prev=-1;
 double circulacio_prev;
 preprocess(node);
 cout<<"Preprocess"<<endl;
 mapa_inicial(node);
 cout<<"Mapa inicial"<<endl;
 det_frontera(node);
 find_position(node);
 cout<<"Frontera"<<endl;
 calcul_coefs(node);
 cout<<"Coefs"<<endl;
 while (fi==false)
 {
  cout<<"Linia de corrent del solid: "<<phi_c<<endl;
  solver(node);
  cout<<"Solver"<<endl;
  Newton_Raphson(node,fi,phi_prev,circulacio_prev);
  cout<<"-----------------------------------------"<<endl;
  if (fi==false)
  {
   canvi_phi_mat(node);
  }
 }
 postprocess(node);

}
void preprocess(vector< vector<info_node> >& node)
{
 geometria(node);
 condicions_inlet(node);
 condicions_contorn(node);
}
void geometria(vector< vector<info_node> >& node)
{
 int material;
 double Ax=L/N, Ay=H/M;
 // Nodes interiors
```

```
for (int i=1;i<N+2;i++)
{
 for (int j=1;j<M+1;j++)
 {
  if (i==1 || i==N+1)
  {
   node[i][j].r[0]=Ax/2;
  }
  else
  {
   node[i][j].r[0]=node[i-1][j].r[0]+Ax;
  }
  if (j==1)
  {
   node[i][j].r[1]=Ay/2;
  }
  else
  {
   node[i][j].r[1]=node[i][j-1].r[1]+Ay;
  }
  node[i][j].Ax=Ax;
  node[i][j].Ay=Ay;
  material=num_material(node[i][j].r[0],node[i][j].r[1]);
  node[i][j].mat=material;
 }
}
// External nodes
// Right and left nodes
for  (int j=1;j<M+1;j++)
{
 node[0][j].r[0]=0;
 node[0][j].r[1]=node[1][j].r[1];
 node[0][j].Ax=0;
 node[0][j].mat=1;
 node[N+1][j].r[0]=L;
 node[N+1][j].r[1]=node[N][j].r[1];
 node[N+1][j].Ax=0;
 node[N+1][j].mat=1;
}
// Top and bottom nodes
for (int i=1;i<N+1;i++)
{
 node[i][0].r[0]=node[i][2].r[0];
 node[i][0].r[1]=0;
 node[i][0].Ay=0;
 node[i][0].mat=1;
 node[i][M+1].r[0]=node[i][M].r[0];
 node[i][M+1].r[1]=H;
 node[i][M+1].mat=1;
 node[i][M+1].Ay=0;
}
// Vertices
node[0][0].r[0]=0;
node[0][0].r[1]=0;
node[0][0].mat=1;
node[0][M+1].r[0]=0;
node[0][M+1].r[1]=H;
node[0][M+1].mat=1;
node[N+1][0].r[0]=L;
node[N+1][0].r[1]=0;
node[N+1][0].mat=1;
node[N+1][M+1].r[0]=L;
```

```
 node[N+1][M+1].r[1]=H;
 node[N+1][M+1].mat=1;

}
int num_material(double x, double y)
{
 if (sqrt((x-cx)*(x-cx)+(y-cy)*(y-cy))<=R)
 {
  return 0;
 }
 else
 {
  return 1;
 }

}
void condicions_inlet(vector< vector<info_node> >& node)
{
 node[0][0].phi=0;
 node[0][0].phi_ant=0;
 node[0][0].T=T_in;
 node[0][0].P=P_in;
 node[0][0].v=v_in;
 node[0][0].rho=rho_in;
 node[0][0].rho_ant=rho_in;
 node[0][0].tau=1;
 for (int j=1;j<M+2;j++)
 {
  node[0][j].phi=node[0][j-1].phi+v_in*(node[0][j].r[1]-node[0][j-1].r[1]);
  node[0][j].phi_ant=node[0][j-1].phi+v_in*(node[0][j].r[1]-node[0][j-1].r[1]);
  node[0][j].T=T_in;
  node[0][j].P=P_in;
  node[0][j].v=v_in;
  node[0][j].rho=rho_in;
  node[0][j].rho_ant=rho_in;
  node[0][j].tau=1;
 }
}
void condicions_contorn(vector< vector<info_node> >& node) // Boundary conditions
    ↪ (cylinder + walls)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==0)
   {
    node[i][j].phi=phi_c;
    node[i][j].phi_ant=phi_c;
    node[i][j].tau=1e30;
   }
   else if (j==0 || j==M+1)
   {
    node[i][j].phi=node[0][j].phi;
    node[i][j].phi_ant=node[0][j].phi_ant;
   }
  }
 }

}
void mapa_inicial(vector< vector<info_node> >& node)
{
```

```
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].rho_ant=rho_in;
    node[i][j].rho=rho_in;
    node[i][j].phi=node[0][j].phi;
    node[i][j].phi_ant=node[0][j].phi;
    node[i][j].tau=1;
    node[i][j].T=T_in;
    node[i][j].T_ant=T_in;
    node[i][j].P=P_in;
    node[i][j].P_ant=P_in;
   }
  }
 }
}
void calcul_coefs(vector< vector<info_node> >& node)
{
 double dpe,dpw, dps, dpn;
 // Internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
    dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
    dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
    dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
    node[i][j].ae=mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,
        ↪ node[i+1][j].Ax/2)*node[i][j].Ay/dpe;
    node[i][j].aw=mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,
        ↪ node[i-1][j].Ax/2)*node[i][j].Ay/dpw;
    node[i][j].as=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,
        ↪ node[i-1][j].Ay/2)*node[i][j].Ax/dps;
    node[i][j].an=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,
        ↪ node[i+1][j].Ay/2)*node[i][j].Ax/dpn;
    node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an;
    node[i][j].bp=0;
   }
   else
   {
    node[i][j].ae=0;
    node[i][j].aw=0;
    node[i][j].as=0;
    node[i][j].an=0;
    node[i][j].ap=1;
    node[i][j].bp=phi_c;
   }
  }
 }
 // Right nodes
 for (int j=1;j<M+1;j++)
 {
  node[N+1][j].aw=1;
  node[N+1][j].an=0;
  node[N+1][j].as=0;
  node[N+1][j].ae=0;
```

```
  node[N+1][j].ap=1;
  node[N+1][j].bp=0;
 }
}
double mitjana_harm(double tau1, double tau2,double d1, double d2)
{
 return (d1+d2)/(d1/tau1+d2/tau2);
}
void solver(vector< vector<info_node> >& node)
{
 bool trobat=false;
 while (trobat==false)
 {
  calcul_coefs(node);
  solver_phi(node);
  calcul_vel(node);
  calcul_prop_term(node);
  if(calcul_error_max(node)<delta)
  {
   trobat=true;
  }
  else
  {
   iteracio(node);
  }
 }
}
void solver_phi(vector< vector<info_node> >& node)
{
 if (line_by_line==false) // Gauss-Seidel
 {
  for (int i=1;i<N+1;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    if (node[i][j].mat==1)
    {
     node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].aw*node[i-1][j].phi
         ↪ +node[i][j].as*node[i][j-1].phi+node[i][j].an*node[i][j+1].phi)/node[i
         ↪ ][j].ap;
     node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
    }
   }
  }
  for (int j=1;j<M+1;j++)
  {
   if (node[N+1][j].mat==1)
   {
    node[N+1][j].phi=(node[N+1][j].aw*node[N][j].phi+node[N+1][j].as*node[N+1][j
        ↪ -1].phi+node[N+1][j].an*node[N+1][j+1].phi)/node[N+1][j].ap;
    node[N+1][j].phi=node[N+1][j].phi_ant+fr*(node[N+1][j].phi-node[N+1][j].
        ↪ phi_ant);
   }
  }
 }
 else // line_by_line
 {
  double P[N+2],Q[N+2];
  for (int j=1;j<M+1;j++)
  {
   P_Q_calcul(j,node,P,Q);
   for (int i=1;i<N+2;i++)
```

```
   {
    node[i][j].phi=P[i]*node[i-1][j].phi+Q[i];
   }
  }
  // Relaxing factor
  for (int i=1;i<N+2;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    if (node[i][j].mat==1)
    {
     node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
    }
   }
  }
 }
}
void calcul_vel(vector< vector<info_node> >& node)
{
 double vye, vyw, vxn, vxs, vyp, vxp;
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (i<N+1)
   {
    vye=-mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j
        ↪ ].Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
    vyw=-mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j
        ↪ ].Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
    vyp=(vye+vyw)/2;
    node[i][j].vel[1]=vyp;
   }
   else
   {
    vyp=0;
   }

   vxn=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].
       ↪ Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
   vxs=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].
       ↪ Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
   vxp=(vxn+vxs)/2;
   node[i][j].vel[0]=vxp;
   node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
   if (node[i][j].mat==0)
   {
    node[i][j].circ=vye*node[i][j].Ay+vxs*node[i][j].Ax-vyw*node[i][j].Ay-vxn*node
        ↪ [i][j].Ax;
    if (isnan(node[i][j].circ))
    {
     cout<<vye<<"␣"<<vyw<<"␣"<<vxs<<"␣"<<vxn<<endl;
     cout<<i<<"␣"<<j<<endl;
    }
   }
  }
 }
 // Top and bottom nodes
 int j;
 for (int i=1;i<N+2;i++)
 {
  j=0;
```

```
  node[i][j].v=2*mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node
      ↪ [i][j+1].Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j+1].Ay;
  j=1;
  if (i<N+1)
  {
   vye=-mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j].
       ↪ Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
   vyw=-mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j].
       ↪ Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
   vyp=(vye+vyw)/2;
   node[i][j].vel[1]=vyp;
  }
  else
  {
   vyp=0;
  }

  vxn=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].Ay
      ↪ /2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
  vxs=2*mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].
      ↪ Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
  vxp=(vxn+vxs)/2;
  node[i][j].vel[0]=vxp;
  node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
  j=M;
  if (i<N+1)
  {
   vye=-mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j].
       ↪ Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
   vyw=-mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j].
       ↪ Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
   vyp=(vye+vyw)/2;
   node[i][j].vel[1]=vyp;
  }
  else
  {
   vyp=0;
  }
  vxn=2*mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].
      ↪ Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
  vxs=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].Ay
      ↪ /2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
  vxp=(vxn+vxs)/2;
  node[i][j].vel[0]=vxp;
  node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
  j=M+1;
  node[i][j].v=2*mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node
      ↪ [i][j-1].Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j-1].Ay;
 }
}
double calcul_cp_barra(double T2, double T1)
{
 double AT, T_inicial,T_final;
 if (T2==T1)
 {
  T2=T1+0.00000001;
 }
 AT=(T2-T1)/Nd;
 double T_mitjana;
 T_inicial=T1;
 double cp_barr=0,cp;
 for (int i=0;i<Nd-1;i++)
```

```
 {
  T_final=T_inicial+AT;
  T_mitjana=(T_final+T_inicial)/2;
  cp=calcul_cp(T_mitjana);
  cp_barr=cp_barr+cp*AT; // Rectangle integration
  T_inicial=T_final;
 }
 return cp_barr/(T2-T1);
}
double calcul_cp(double T)
{
 return a0+a1*T+a2*pow(T,2)+a3*pow(T,3)+a4*pow(T,4);
}
double calcul_gamma(double cp)
{
 return cp/(cp-Rg);
}
double calcul_cp_barret(double T0, double T)
{
 double AT, T_inicial,T_final;
 if (T0==T)
 {
  T0=T+0.00000001;
 }
 AT=(T0-T)/Nd;
 double T_mitjana;
 T_inicial=T;
 double cp_barr=0,cp;
 for (int i=0;i<Nd-1;i++)
 {
  T_final=T_inicial+AT;
  T_mitjana=(T_final+T_inicial)/2;
  cp=calcul_cp(T_mitjana);
  cp_barr=cp_barr+cp/T_mitjana*AT;
  T_inicial=T_final;
 }
 return cp_barr/(log(T0/T));
}
void calcul_prop_term(vector< vector<info_node> >& node)
{
 double gamma_barret;
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
    node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
    if(node[i][j].T<0)
    {
     node[i][j].T=1;
    }
    node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
    gamma_barret=calcul_gamma(node[i][j].cp_barret);
    node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
    node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
   }
  }

 }
 // Top and bottom nodes
```

```
 int j=0;
 for (int i=1;i<N+2;i++)
 {
  if (node[i][j].mat==1)
  {
   j=0;
   node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
   node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
   node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
   gamma_barret=calcul_gamma(node[i][j].cp_barret);
   node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
   node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
   j=M+1;
   node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
   node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
   node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
   gamma_barret=calcul_gamma(node[i][j].cp_barret);
   node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
   node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
  }
 }
}
double calcul_error_max(vector< vector<info_node> >& node)
{
 double error=0;
 double error_T,error_rho,error_P;
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   error_T=v_abs(node[i][j].T-node[i][j].T_ant);
   error_rho=v_abs(node[i][j].rho-node[i][j].rho_ant);
   error_P=v_abs(node[i][j].P-node[i][j].P_ant);
   if (error_T>error && error_T>error_rho && error_T>error_P)
   {
    error=error_T;
   }
   else if(error_rho>error && error_T<error_rho && error_rho>error_P)
   {
    error=error_rho;
   }
   else if (error_P>error && error_T<error_P && error_rho<error_P)
   {
    error=error_P;
   }
  }
 }
 cout<<delta/error*100<<"%"<<endl;
 return error;
}
double v_abs(double a)
{
 if (a>0)
 {
  return a;
 }
 else
 {
  return -a;
 }
}
void iteracio (vector< vector<info_node> >& node)
```

```
{
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].phi_ant=node[i][j].phi;
    node[i][j].rho_ant=node[i][j].rho;
    node[i][j].tau=rho_in/node[i][j].rho;
    node[i][j].P_ant=node[i][j].P;
    node[i][j].T_ant=node[i][j].T;
   }
  }
 }
}
void save_phi (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Phi");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].phi<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_rho (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Rho");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].rho<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_T (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Temp");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].T<<"\t";
  }
  file<<endl;
 }
 file<<N+2<<endl;
 file<<M+2<<endl;
 file.close();
}
void save_P (vector< vector<info_node> >& node)
{
```

```
 ofstream file;
 file.open("Pres");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].P<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_v (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("vel");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].v<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_tau (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("tau");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].tau<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void postprocess(vector< vector<info_node> >& node)
{
 double circulacio;
 save_phi(node);
 save_rho(node);
 save_T(node);
 save_P(node);
 save_v(node);
 save_posicio_mat(node);
 calcul_error(node);
 calcul_circulacio(node,circulacio);
 calcul_forces(node,circulacio);
 cout<<"--------------------------------"<<endl;
 cout<<"Velocitat␣a␣la␣posicio␣("<<x_conv<<","<<y_conv<<")"<<endl;
 cout<<"v="<<node[i_conv][j_conv].v<<endl;
}
void save_posicio_mat(vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Posicio");
 for (int j=0;j<M+2;j++)
 {
```

```
  file<<node[1][j].r[1]<<endl;
 }
 for (int i=0;i<N+2;i++)
 {
  file<<node[i][1].r[0]<<endl;
 }
 file.close();
 file.open("Material");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].mat<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void calcul_error(vector< vector<info_node> >& node)
{
 double error=0,error2;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    error2=node[i][j].phi*node[i][j].ap-(node[i][j].ae*node[i+1][j].phi+node[i][j
        ↪ ].aw*node[i-1][j].phi+node[i][j].as*node[i][j-1].phi+node[i][j].an*node
        ↪ [i][j+1].phi);
    if (v_abs(error2)>error)
    {
     error=v_abs(error2);
    }
   }
  }
 }
 cout<<"Error comes: "<<error<<endl;
}
void det_frontera(vector< vector<info_node> >& node)
{
 double nx,ny;
 int punts_front=0;
 // Only considered Internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    if(node[i][j-1].mat==0 || node[i][j+1].mat==0|| node[i+1][j].mat==0 || node[i
        ↪ -1][j].mat==0)
    {
     node[i][j].frontera=true;
     nx=(node[i][j].r[0]-cx);
     ny=(node[i][j].r[1]-cy);
     node[i][j].n[0]=(nx)/sqrt(nx*nx+ny*ny);
     node[i][j].n[1]=(ny)/sqrt(nx*nx+ny*ny);
     node[i][j].t[0]=-node[i][j].n[1];
     node[i][j].t[1]=node[i][j].n[0];
     punts_front++;
    }
```

```
    else
    {
     node[i][j].frontera=false;
    }
   }
  }
 }
 double dS=2*pi*R/punts_front;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].frontera==true)
   {
    node[i][j].n[0]=node[i][j].n[0]*dS;
    node[i][j].n[1]=node[i][j].n[1]*dS;
    node[i][j].t[0]=node[i][j].t[0]*dS;
    node[i][j].t[1]=node[i][j].t[1]*dS;
   }
  }
 }

}
void calcul_circulacio(vector< vector<info_node> >& node,double &circulacio)
{
 circulacio=0;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==0)
   {
    circulacio=circulacio+node[i][j].circ;
   }
  }
 }
 cout<<"Circulacio al voltant del cos: "<<circulacio<<endl;
}
void calcul_forces(vector< vector<info_node> >& node,double circulacio)
{
 double Lift=0,D=0;
 double CL,CD;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].frontera==true && node[i][j].mat==1)
   {
    Lift=Lift-node[i][j].P*node[i][j].n[1];
    D=D-node[i][j].P*node[i][j].n[0];
   }
  }
 }
 CL=Lift/(0.5*rho_in*v_in*v_in*D_main); // adimensionalization
 CD=D/(0.5*rho_in*v_in*v_in*D_main);
 cout<<"Lift= "<<Lift<<endl;
 cout<<"Drag ="<<D<<endl;
 cout<<"CL= "<<CL<<endl;
 cout<<"CD= "<<CD<<endl;
 guardar_dades(Lift,CL,D,CD,circulacio);
}
void guardar_dades(double Lift,double CL, double D, double CD,double circulacio)
```

```
{
 ofstream file;
 file.open("Dades_Resultats");
 file<<"DADES␣DEL␣PROBLEMA"<<endl;
 file<<"Altura:␣"<<H<<"␣m"<<endl;
 file<<"Longitud:␣"<<L<<"␣m"<<endl;
 file<<"Radi␣cilindre:␣"<<D_main<<"␣m"<<endl;
 file<<"Posicio␣del␣centre␣del␣cilindre:␣("<<cx<<"␣,␣"<<cy<<")␣m"<<endl;
 file<<"Velocitat␣a␣l'entrada:␣"<<v_in<<"␣m/s"<<endl;
 file<<"Temperatura␣a␣l'entrada:␣"<<T_in-273.15<<"␣ C "<<endl;
 file<<"Pressio␣a␣l'entrada:␣"<<P_in<<"␣Pa"<<endl;
 file<<"Densitat␣a␣l'entrada:␣"<<rho_in<<"␣kg/m^3"<<endl;
 file<<"Linia␣de␣corrent␣de␣l'objecte:␣"<<phi_c<<endl;
 file<<"Densitat␣de␣malla:␣"<<N<<"x"<<M<<endl;
 file<<"RESULTATS␣DEL␣PROBLEMA"<<endl;
 file<<"Lift:␣"<<Lift<<"␣N/m"<<endl;
 file<<"C_L:␣"<<CL<<endl;
 file<<"Drag:␣"<<D<<"␣N/m"<<endl;
 file<<"CD:␣"<<CD<<endl;
 file<<"Circulacio␣al␣voltant␣del␣cos:␣"<<circulacio<<endl;
 time(&final);
 file<<"Temps␣de␣ c lcul :␣"<<difftime(final,inici)<<"␣s"<<endl;
}

void Newton_Raphson(vector< vector<info_node> >& node, bool &fi,double &phi_prev,
    ↪ double &circulacio_prev)
{
 double circulacio;
 calcul_circulacio(node,circulacio);
 if (v_abs(circulacio-circulacio_final)<error_circ)
 {
  fi=true;
 }
 else
 {
  if (phi_prev==-1) // only for the first iteration
  {
   phi_prev=phi_c;
   circulacio_prev=circulacio;
   if(circulacio<circulacio_final) // the streamline has a lower value that the
       ↪ one it should have
   {
    phi_c=v_in*H*1.2;
   }
   else
   {
    phi_c=v_in*H*0.8;
   }
  }
  else
  {
   double pendent=(circulacio-circulacio_prev)/(phi_c-phi_prev);
   phi_prev=phi_c;
   phi_c=phi_c-(circulacio-circulacio_final)/pendent;
   circulacio_prev=circulacio;
  }
 }
}
void canvi_phi_mat(vector< vector<info_node> >& node)
{
 for (int i=1;i<N+1;i++)
 {
```

```
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==0)
   {
    node[i][j].phi=phi_c;
    node[i][j].phi_ant=phi_c;
   }
  }
 }
}
void declaracio_inicial()
{
 D_main=R;
 circulacio_final=-w*R*2*pi*R;
}
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2])
{
 P[N+1]=node[N+1][row].aw/node[N+1][row].ap;
 Q[N+1]=(node[N+1][row].an*node[N+1][row+1].phi+node[N+1][row].as*node[N+1][row
     ↪ -1].phi+node[N+1][row].bp)/node[N+1][row].ap;
 double bp_ast;
 for (int i=N;i>0;i--)
 {
  P[i]=node[i][row].aw/(node[i][row].ap-node[i][row].ae*P[i+1]);
  bp_ast=node[i][row].an*node[i][row+1].phi+node[i][row].as*node[i][row-1].phi+
      ↪ node[i][row].bp;
  Q[i]=(bp_ast+node[i][row].ae*Q[i+1])/(node[i][row].ap-node[i][row].ae*P[i+1]);
 }
}
void find_position(vector< vector<info_node> >& node)
{
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].r[0]-node[i][j].Ax/2 < x_conv && node[i][j].r[0]+node[i][j].Ax/2
       ↪  >= x_conv && node[i][j].r[1]-node[i][j].Ay/2 < y_conv && node[i][j].r
       ↪ [1]+node[i][j].Ay/2 >= y_conv)
   {
    i_conv=i;
    j_conv=j;
    break;
   }
  }
 }
}
```

Appendix B

Developed code for Incompressible Potential Flow (streamline method)

```
//  Incompressible potential flow Numerical solver  (imposed circulation condition
    ↪ ) -streamlines method
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <time.h>
using namespace std;
// Constant numbers
const double pi=3.141592;
//*************************************************************
// Constants
const double H=5, L=10; //Problem's geometry
const double Rg=287; // gas constant
const double v_in=10, P_in=1.013e5, T_in=288, rho_in=P_in/Rg/T_in; // Inlet flow
    ↪ parameters
const int N=500, M=N; // Mesh density
const double fr=1.8,delta=1e-8; // Program parameters (Gauss_Seidel: fr=1.9 //
    ↪ line-by-line: fr=1.1)
const double Nd=5; // division to estimate the integral
double phi_c=v_in*H/2;
// Cylinder parameters
const double R=0.3, cx=L/2, cy=H/2; // Cylinder's geometry
const double w=10; //Cylinder's rotation
// Desired circulation around the object
double circulacio_final;
const double error_circ=1e-5;
// Characteristic lenght
double D_main;
// Gas cp treatemet: cp(T)=a0*T^0+a1*T^1+a2*T2+a3*T^3+a4*T^4
// Air
const double a0=1034.09, a1=-2.849e-1, a2=7.817e-4, a3=-4.971e-7, a4=1.077e-10;
// Helium
//const double a0=5188, a1=0, a2=0, a3=0, a4=0;
// Solver type: line_by_line or Gauss-Seidel
bool line_by_line=false;
// Convergence analysis point
```

```
double x_conv=5, y_conv=2.9;
int i_conv, j_conv;
//************************************************************
// Variables declaration
time_t inici,final;
struct info_node{
 double r[2];
 double phi;
 double phi_ant;
 double rho;
 double rho_ant;
 double tau;
 double T;
 double T_ant;
 double P;
 double P_ant;
 double Ax;
 double Ay;
 double cp_barra;
 double cp_barret;
 double gamma_barret;
 double vel[2]; // x and y velocity components
 double v; // velocity modulus
 int mat; // 0 solid, 1 fluid
 double ap;
 double as;
 double an;
 double aw;
 double ae;
 double bp;
 bool frontera; // determines if the node is close to the object
 double t[2]; // vtangential vector
 double n[2]; // normal vector
 double circ; //control volume circulation
};
// Functions
void preprocess(vector< vector<info_node> >& node);
void geometria(vector< vector<info_node> >& node);
int num_material(double x, double y);
void condicions_inlet(vector< vector<info_node> >& node);
void condicions_contorn(vector< vector<info_node> >& node);
void mapa_inicial(vector< vector<info_node> >& node);
void calcul_coefs(vector< vector<info_node> >& node);
double mitjana_harm(double tau1, double tau2,double d1, double d2);
void solver(vector< vector<info_node> >& node);
void solver_phi(vector< vector<info_node> >& node);
void calcul_vel(vector< vector<info_node> >& node);
double calcul_cp_barra(double T2, double T1);
double calcul_cp(double T);
double calcul_gamma(double cp);
void calcul_prop_term(vector< vector<info_node> >& node);
double calcul_error_max(vector< vector<info_node> >& node);
double v_abs(double a);
void iteracio (vector< vector<info_node> >& node);
void save_phi (vector< vector<info_node> >& node);
void save_rho (vector< vector<info_node> >& node);
void save_T (vector< vector<info_node> >& node);
void save_P (vector< vector<info_node> >& node);
void save_v (vector< vector<info_node> >& node);
void save_tau (vector< vector<info_node> >& node);
void postprocess(vector< vector<info_node> >& node);
void save_posicio_mat(vector< vector<info_node> >& node);
```

```
void calcul_error(vector< vector<info_node> >& node);
void det_frontera(vector< vector<info_node> >& node);
void calcul_circulacio(vector< vector<info_node> >& node,double &circulacio);
void calcul_forces(vector< vector<info_node> >& node,double circulacio);
void guardar_dades(double L,double CL, double D, double CD,double circulacio);
void Newton_Raphson(vector< vector<info_node> >& node, bool &fi,double &phi_prev,
    ↪ double &circulacio_prev);
void canvi_phi_mat(vector< vector<info_node> >& node);
void declaracio_inicial();
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2]);
void find_position(vector< vector<info_node> >& node);
// Code
int main()
{
 time(&inici);
 declaracio_inicial();
 cout<<"Inici"<<endl;
 vector< vector<info_node> > node(N+2, vector<info_node>(M+2));
 bool fi=false;
 double phi_prev=-1;
 double circulacio_prev;
 preprocess(node);
 cout<<"Preprocess"<<endl;
 mapa_inicial(node);
 cout<<"Mapa inicial"<<endl;
 det_frontera(node);
 find_position(node);
 cout<<"Frontera"<<endl;
 calcul_coefs(node);
 cout<<"Coefs"<<endl;
 while (fi==false)
 {
  cout<<"Linia de corrent del solid: "<<phi_c<<endl;
  solver(node);
  cout<<"Solver"<<endl;
  Newton_Raphson(node,fi,phi_prev,circulacio_prev);
  cout<<"-----------------------------------------------"<<endl;
  if (fi==false)
  {
   canvi_phi_mat(node);
  }
 }
 postprocess(node);

}
void preprocess(vector< vector<info_node> >& node)
{
 geometria(node);
 condicions_inlet(node);
 condicions_contorn(node);
}
void geometria(vector< vector<info_node> >& node)
{
 int material;
 double Ax=L/N, Ay=H/M;
 // NInternal nodes
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (i==1 || i==N+1)
```

```
    {
     node[i][j].r[0]=Ax/2;
    }
    else
    {
     node[i][j].r[0]=node[i-1][j].r[0]+Ax;
    }
    if (j==1)
    {
     node[i][j].r[1]=Ay/2;
    }
    else
    {
     node[i][j].r[1]=node[i][j-1].r[1]+Ay;
    }
    node[i][j].Ax=Ax;
    node[i][j].Ay=Ay;
    material=num_material(node[i][j].r[0],node[i][j].r[1]);
    node[i][j].mat=material;
   }
  }
  // NExternal nodes
  // Right and left nodes
  for  (int j=1;j<M+1;j++)
  {
   node[0][j].r[0]=0;
   node[0][j].r[1]=node[1][j].r[1];
   node[0][j].Ax=0;
   node[0][j].mat=1;
   node[N+1][j].r[0]=L;
   node[N+1][j].r[1]=node[N][j].r[1];
   node[N+1][j].Ax=0;
   node[N+1][j].mat=1;
  }
  // Top and bottom nodes
  for (int i=1;i<N+1;i++)
  {
   node[i][0].r[0]=node[i][2].r[0];
   node[i][0].r[1]=0;
   node[i][0].Ay=0;
   node[i][0].mat=1;
   node[i][M+1].r[0]=node[i][M].r[0];
   node[i][M+1].r[1]=H;
   node[i][M+1].mat=1;
   node[i][M+1].Ay=0;
  }
  // Vertices
  node[0][0].r[0]=0;
  node[0][0].r[1]=0;
  node[0][0].mat=1;
  node[0][M+1].r[0]=0;
  node[0][M+1].r[1]=H;
  node[0][M+1].mat=1;
  node[N+1][0].r[0]=L;
  node[N+1][0].r[1]=0;
  node[N+1][0].mat=1;
  node[N+1][M+1].r[0]=L;
  node[N+1][M+1].r[1]=H;
  node[N+1][M+1].mat=1;

}
int num_material(double x, double y)
```

```
{
 if (sqrt((x-cx)*(x-cx)+(y-cy)*(y-cy))<=R)
 {
  return 0;
 }
 else
 {
  return 1;
 }

}
void condicions_inlet(vector< vector<info_node> >& node)
{
 double r, theta;
 r=sqrt((node[0][0].r[0]-cx)*(node[0][0].r[0]-cx)+(node[0][0].r[1]-cy)*(node
    ↪ [0][0].r[1]-cy));
 theta=pi+ atan ((node[0][0].r[1]-cy)/(node[0][0].r[0]-cx));
 node[0][0].phi=v_in*r*sin(theta)*(1-(R*R)/(r*r))+2*pi*w*R*R/(2*pi)*log(r/R);
 node[0][0].phi_ant=node[0][0].phi_ant;
 node[0][0].T=T_in;
 node[0][0].P=P_in;
 node[0][0].v=v_in;
 node[0][0].rho=rho_in;
 node[0][0].rho_ant=rho_in;
 node[0][0].tau=1;
 int i=0;
 for (int j=0;j<M+2;j++)
 {
  r=sqrt((node[i][j].r[0]-cx)*(node[i][j].r[0]-cx)+(node[i][j].r[1]-cy)*(node[i][j
     ↪ ].r[1]-cy));
  theta= atan ((node[i][j].r[1]-cy)/(node[i][j].r[0]-cx));
  if (node[i][j].r[0]<cx)
  {
   theta=theta+pi;
  }
  node[i][j].phi=v_in*r*sin(theta)*(1-(R*R)/(r*r))+2*pi*w*R*R/(2*pi)*log(r/R);
  node[i][j].phi_ant=node[i][j].phi;
  node[0][j].T=T_in;
  node[0][j].T_ant=T_in;
  node[0][j].P=P_in;
  node[0][j].P_ant=P_in;
  node[0][j].v=v_in;
  node[0][j].rho=rho_in;
  node[0][j].rho_ant=rho_in;
  node[0][j].tau=1;
 }
}
void condicions_contorn(vector< vector<info_node> >& node) // boundary conditions
    ↪ (cylinder + wall)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==0)
   {
    node[i][j].phi=phi_c;
    node[i][j].phi_ant=phi_c;
    node[i][j].tau=1e30;
   }
  }
 }
```

```
 int j;
 double r, theta;
 for (int i=1;i<N+2;i++)
 {
  j=0;
  r=sqrt((node[i][j].r[0]-cx)*(node[i][j].r[0]-cx)+(node[i][j].r[1]-cy)*(node[i][j
      ↪ ].r[1]-cy));
  theta= atan ((node[i][j].r[1]-cy)/(node[i][j].r[0]-cx));
  if (node[i][j].r[0]<cx)
  {
   theta=theta+pi;
  }
  node[i][j].phi=v_in*r*sin(theta)*(1-(R*R)/(r*r))+2*pi*w*R*R/(2*pi)*log(r/R);
  node[i][j].phi_ant=node[i][j].phi;
  j=M+1;
  r=sqrt((node[i][j].r[0]-cx)*(node[i][j].r[0]-cx)+(node[i][j].r[1]-cy)*(node[i][j
      ↪ ].r[1]-cy));
  theta= atan ((node[i][j].r[1]-cy)/(node[i][j].r[0]-cx));
  if (node[i][j].r[0]<cx)
  {
   theta=theta+pi;
  }
  node[i][j].phi=v_in*r*sin(theta)*(1-(R*R)/(r*r))+2*pi*w*R*R/(2*pi)*log(r/R);
  node[i][j].phi_ant=node[i][j].phi;
 }

}
void mapa_inicial(vector< vector<info_node> >& node)
{
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].rho_ant=rho_in;
    node[i][j].rho=rho_in;
    node[i][j].tau=1;
    node[i][j].T=T_in;
    node[i][j].T_ant=T_in;
    node[i][j].P=P_in;
    node[i][j].P_ant=P_in;
   }
  }
 }
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].phi=node[0][j].phi;
    node[i][j].phi_ant=node[0][j].phi;
   }
  }
 }
}
void calcul_coefs(vector< vector<info_node> >& node)
{
 double dpe,dpw, dps, dpn;
 // Internal nodes
 for (int i=1;i<N+1;i++)
```

```
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
    dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
    dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
    dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
    node[i][j].ae=mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,
        ↪ node[i+1][j].Ax/2)*node[i][j].Ay/dpe;
    node[i][j].aw=mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,
        ↪ node[i-1][j].Ax/2)*node[i][j].Ay/dpw;
    node[i][j].as=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,
        ↪ node[i-1][j].Ay/2)*node[i][j].Ax/dps;
    node[i][j].an=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,
        ↪ node[i+1][j].Ay/2)*node[i][j].Ax/dpn;
    node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an;
    node[i][j].bp=0;
   }
   else
   {
    node[i][j].ae=0;
    node[i][j].aw=0;
    node[i][j].as=0;
    node[i][j].an=0;
    node[i][j].ap=1;
    node[i][j].bp=phi_c;
   }
  }
 }
 // NRight nodes
 for (int j=1;j<M+1;j++)
 {
  node[N+1][j].aw=1;
  node[N+1][j].an=0;
  node[N+1][j].as=0;
  node[N+1][j].ae=0;
  node[N+1][j].ap=1;
  node[N+1][j].bp=0;
 }
}
double mitjana_harm(double tau1, double tau2,double d1, double d2)
{
 return (d1+d2)/(d1/tau1+d2/tau2);
}
void solver(vector< vector<info_node> >& node)
{
 bool trobat=false;
 while (trobat==false)
 {
  calcul_coefs(node);
  solver_phi(node);
  if(calcul_error_max(node)<delta)
  {
   trobat=true;
   calcul_vel(node);
   calcul_prop_term(node);
  }
  else
  {
   iteracio(node);
```

```
  }
 }
}
void solver_phi(vector< vector<info_node> >& node)
{
 if (line_by_line==false) // Gauss-Seidel
 {
  for (int i=1;i<N+1;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    if (node[i][j].mat==1)
    {
     node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].aw*node[i-1][j].phi
         ↪ +node[i][j].as*node[i][j-1].phi+node[i][j].an*node[i][j+1].phi)/node[i
         ↪ ][j].ap;
     node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
    }
   }
  }
  for (int j=1;j<M+1;j++)
  {
   if (node[N+1][j].mat==1)
   {
    node[N+1][j].phi=(node[N+1][j].aw*node[N][j].phi+node[N+1][j].as*node[N+1][j
        ↪ -1].phi+node[N+1][j].an*node[N+1][j+1].phi)/node[N+1][j].ap;
    node[N+1][j].phi=node[N+1][j].phi_ant+fr*(node[N+1][j].phi-node[N+1][j].
        ↪ phi_ant);
   }
  }
 }
 else // line_by_line
 {
  double P[N+2],Q[N+2];
  for (int j=1;j<M+1;j++)
  {
   P_Q_calcul(j,node,P,Q);
   for (int i=1;i<N+2;i++)
   {
    node[i][j].phi=P[i]*node[i-1][j].phi+Q[i];
   }
  }
  // Relaxing factor
  for (int i=1;i<N+2;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    if (node[i][j].mat==1)
    {
     node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
    }
   }
  }
 }
}
void calcul_vel(vector< vector<info_node> >& node)
{
 double vye, vyw, vxn, vxs, vyp, vxp;
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
```

```
   if (i<N+1)
   {
    vye=-mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j
        ↪ ].Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
    vyw=-mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j
        ↪ ].Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
    vyp=(vye+vyw)/2;
    node[i][j].vel[1]=vyp;

   }
   else
   {
    vyp=0;
   }

   vxn=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].
       ↪ Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
   vxs=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].
       ↪ Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
   vxp=(vxn+vxs)/2;
   node[i][j].vel[0]=vxp;
   node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
   if (node[i][j].mat==0)
   {
    node[i][j].circ=vye*node[i][j].Ay+vxs*node[i][j].Ax-vyw*node[i][j].Ay-vxn*node
        ↪ [i][j].Ax;
    if (isnan(node[i][j].circ))
    {
     cout<<vye<<"␣"<<vyw<<"␣"<<vxs<<"␣"<<vxn<<endl;
     cout<<i<<"␣"<<j<<endl;
    }
   }
  }
 }
 // NTop and bottom nodes
 int j;
 for (int i=1;i<N+2;i++)
 {
  j=0;
  node[i][j].v=2*mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node
      ↪ [i][j+1].Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j+1].Ay;
  j=1;
  if (i<N+1)
  {
   vye=-mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j].
       ↪ Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
   vyw=-mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j].
       ↪ Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
   vyp=(vye+vyw)/2;
   node[i][j].vel[1]=vyp;
  }
  else
  {
   vyp=0;
  }

  vxn=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].Ay
      ↪ /2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
  vxs=2*mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].
      ↪ Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
  vxp=(vxn+vxs)/2;
  node[i][j].vel[0]=vxp;
```

```
  node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
  j=M;
  if (i<N+1)
  {
   vye=-mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j].
       ↪ Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
   vyw=-mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j].
       ↪ Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
   vyp=(vye+vyw)/2;
   node[i][j].vel[1]=vyp;
  }
  else
  {
   vyp=0;
  }
  vxn=2*mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].
      ↪ Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
  vxs=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].Ay
      ↪ /2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
  vxp=(vxn+vxs)/2;
  node[i][j].vel[0]=vxp;
  node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
  j=M+1;
  node[i][j].v=2*mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node
      ↪ [i][j-1].Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j-1].Ay;
 }
}
double calcul_cp_barra(double T2, double T1)
{
 double AT, T_inicial,T_final;
 if (T2==T1)
 {
  T2=T1+0.00000001;
 }
 AT=(T2-T1)/Nd;
 double T_mitjana;
 T_inicial=T1;
 double cp_barr=0,cp;
 for (int i=0;i<Nd-1;i++)
 {
  T_final=T_inicial+AT;
  T_mitjana=(T_final+T_inicial)/2;
  cp=calcul_cp(T_mitjana);
  cp_barr=cp_barr+cp*AT; // Rectangle integration
  T_inicial=T_final;
 }
 return cp_barr/(T2-T1);
}
double calcul_cp(double T)
{
 return a0+a1*T+a2*pow(T,2)+a3*pow(T,3)+a4*pow(T,4);
}
double calcul_gamma(double cp)
{
 return cp/(cp-Rg);
}
void calcul_prop_term(vector< vector<info_node> >& node)
{
 double gamma_barret;
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
```

```
  {
   if (node[i][j].mat==1)
   {
    node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
    node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
    if(node[i][j].T<0)
    {
     node[i][j].T=1;
    }
    node[i][j].P=P_in+0.5*rho_in*(v_in*v_in-node[i][j].v*node[i][j].v);
   }
  }

 }
 // Top and bottom nodes
 int j=0;
 for (int i=1;i<N+2;i++)
 {
  if (node[i][j].mat==1)
  {
   j=0;
   node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
   node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
   node[i][j].P=P_in+0.5*rho_in*(v_in*v_in-node[i][j].v*node[i][j].v);
   j=M+1;
   node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
   node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
   node[i][j].P=P_in+0.5*rho_in*(v_in*v_in-node[i][j].v*node[i][j].v);
  }
 }
}
double calcul_error_max(vector< vector<info_node> >& node)
{
 double error=0;
 double error_phi;
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   error_phi=v_abs(node[i][j].phi-node[i][j].phi_ant);
   if (error_phi>error )
   {
    error=error_phi;
   }
  }
 }
 cout<<delta/error*100<<"%"<<endl;
 return error;
}
double v_abs(double a)
{
 if (a>0)
 {
  return a;
 }
 else
 {
  return -a;
 }
}
void iteracio (vector< vector<info_node> >& node)
{
```

```
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].phi_ant=node[i][j].phi;
   }
  }
 }
}
void save_phi (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Phi");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].phi<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_rho (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Rho");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].rho<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_T (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Temp");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].T<<"\t";
  }
  file<<endl;
 }
 file<<N+2<<endl;
 file<<M+2<<endl;
 file.close();
}
void save_P (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Pres");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
```

```
  {
   file<<node[i][j].P<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_v (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("vel");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].v<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_tau (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("tau");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].tau<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void postprocess(vector< vector<info_node> >& node)
{
 double circulacio;
 save_phi(node);
 save_rho(node);
 save_T(node);
 save_P(node);
 save_v(node);
 save_posicio_mat(node);
 calcul_error(node);
 calcul_circulacio(node,circulacio);
 calcul_forces(node,circulacio);
 cout<<"v="<<node[i_conv][j_conv].v<<endl;
}
void save_posicio_mat(vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Posicio");
 for (int j=0;j<M+2;j++)
 {
  file<<node[1][j].r[1]<<endl;
 }
 for (int i=0;i<N+2;i++)
 {
  file<<node[i][1].r[0]<<endl;
 }
 file.close();
```

```
 file.open("Material");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].mat<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void calcul_error(vector< vector<info_node> >& node)
{
 double error=0,error2;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    error2=node[i][j].phi*node[i][j].ap-(node[i][j].ae*node[i+1][j].phi+node[i][j
        ↪ ].aw*node[i-1][j].phi+node[i][j].as*node[i][j-1].phi+node[i][j].an*node
        ↪ [i][j+1].phi);
    if (v_abs(error2)>error)
    {
     error=v_abs(error2);
    }
   }
  }
 }
 cout<<"Error comes: "<<error<<endl;
}
void det_frontera(vector< vector<info_node> >& node)
{
 double nx,ny;
 int punts_front=0;
 // Only considered internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    if(node[i][j-1].mat==0 || node[i][j+1].mat==0|| node[i+1][j].mat==0 || node[i
        ↪ -1][j].mat==0)
    {
     node[i][j].frontera=true;
     nx=(node[i][j].r[0]-cx);
     ny=(node[i][j].r[1]-cy);
     node[i][j].n[0]=(nx)/sqrt(nx*nx+ny*ny);
     node[i][j].n[1]=(ny)/sqrt(nx*nx+ny*ny);
     node[i][j].t[0]=-node[i][j].n[1];
     node[i][j].t[1]=node[i][j].n[0];
     punts_front++;
    }
    else
    {
     node[i][j].frontera=false;
    }
   }
  }
 }
```

```
 double dS=2*pi*R/punts_front;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].frontera==true)
   {
    node[i][j].n[0]=node[i][j].n[0]*dS;
    node[i][j].n[1]=node[i][j].n[1]*dS;
    node[i][j].t[0]=node[i][j].t[0]*dS;
    node[i][j].t[1]=node[i][j].t[1]*dS;
   }
  }
 }
}
void calcul_circulacio(vector< vector<info_node> >& node,double &circulacio)
{
 circulacio=0;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].frontera==1)
   {
    circulacio=circulacio+node[i][j].vel[0]*node[i][j].t[0]+node[i][j].vel[1]*node
        ↪ [i][j].t[1];
   }
  }
 }
 cout<<"Circulacio al voltant del cos: "<<circulacio<<endl;
}
void calcul_forces(vector< vector<info_node> >& node,double circulacio)
{
 long double Lift=0,D=0;
 double CL,CD;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].frontera==true)
   {
    Lift=Lift-node[i][j].P*node[i][j].n[1];
    D=D-node[i][j].P*node[i][j].n[0];
   }
  }
 }
 CL=Lift/(0.5*rho_in*v_in*v_in*D_main); // adimensionalization
 CD=D/(0.5*rho_in*v_in*v_in*D_main);
 cout<<"Lift= "<<Lift<<endl;
 cout<<"Drag ="<<D<<endl;
 cout<<"CL= "<<CL<<endl;
 cout<<"CD= "<<CD<<endl;
 guardar_dades(Lift,CL,D,CD,circulacio);
}
void guardar_dades(double Lift,double CL, double D, double CD,double circulacio)
{
 ofstream file;
 file.open("Dades_Resultats");
 file<<"DADES DEL PROBLEMA"<<endl;
 file<<"Altura: "<<H<<" m"<<endl;
 file<<"Longitud: "<<L<<" m"<<endl;
 file<<"Radi cilindre: "<<D_main<<" m"<<endl;
```

```
 file<<"Posicio del centre del cilindre: ("<<cx<<" , "<<cy<<") m"<<endl;
 file<<"Velocitat a l'entrada: "<<v_in<<" m/s"<<endl;
 file<<"Temperatura a l'entrada: "<<T_in-273.15<<" C "<<endl;
 file<<"Pressio a l'entrada: "<<P_in<<" Pa"<<endl;
 file<<"Densitat a l'entrada: "<<rho_in<<" kg/m^3"<<endl;
 file<<"Linia de corrent de l'objecte: "<<phi_c<<endl;
 file<<"Densitat de malla: "<<N<<"x"<<M<<endl;
 file<<"RESULTATS DEL PROBLEMA"<<endl;
 file<<"Lift: "<<Lift<<" N/m"<<endl;
 file<<"C_L: "<<CL<<endl;
 file<<"Drag: "<<D<<" N/m"<<endl;
 file<<"CD: "<<CD<<endl;
 file<<"Circulacio al voltant del cos: "<<circulacio<<endl;
 time(&final);
 file<<"Temps de  c lcul : "<<difftime(final,inici)<<" s"<<endl;
}
void Newton_Raphson(vector< vector<info_node> >& node, bool &fi,double &phi_prev,
    ↪ double &circulacio_prev)
{
 double circulacio;
 calcul_circulacio(node,circulacio);
 if (v_abs(circulacio-circulacio_final)<error_circ)
 {
  fi=true;
 }
 else
 {
  if (phi_prev==-1) // first iteration only
  {
   phi_prev=phi_c;
   circulacio_prev=circulacio;
   if(circulacio<circulacio_final) // the streamline has a lower value that the
       ↪ one it should have
   {
    phi_c=v_in*H*3/4;
   }
   else
   {
    phi_c=v_in*H*1/4;
   }
  }
  else
  {
   double pendent=(circulacio-circulacio_prev)/(phi_c-phi_prev);
   phi_prev=phi_c;
   phi_c=phi_c-(circulacio-circulacio_final)/pendent;
   circulacio_prev=circulacio;
  }
 }
}
void canvi_phi_mat(vector< vector<info_node> >& node)
{
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==0)
   {
    node[i][j].phi=phi_c;
    node[i][j].phi_ant=phi_c;
   }
  }
```

```
 }
}
void declaracio_inicial()
{
 D_main=R;
 circulacio_final=-w*R*2*pi*R;
}
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2])
{
 P[N+1]=node[N+1][row].aw/node[N+1][row].ap;
 Q[N+1]=(node[N+1][row].an*node[N+1][row+1].phi+node[N+1][row].as*node[N+1][row
     ↪ -1].phi+node[N+1][row].bp)/node[N+1][row].ap;
 double bp_ast;
 for (int i=N;i>0;i--)
 {
  P[i]=node[i][row].aw/(node[i][row].ap-node[i][row].ae*P[i+1]);
  bp_ast=node[i][row].an*node[i][row+1].phi+node[i][row].as*node[i][row-1].phi+
      ↪ node[i][row].bp;
  Q[i]=(bp_ast+node[i][row].ae*Q[i+1])/(node[i][row].ap-node[i][row].ae*P[i+1]);
 }
}
void find_position(vector< vector<info_node> >& node)
{
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].r[0]-node[i][j].Ax/2 < x_conv && node[i][j].r[0]+node[i][j].Ax/2
       ↪  >= x_conv && node[i][j].r[1]-node[i][j].Ay/2 < y_conv && node[i][j].r
       ↪ [1]+node[i][j].Ay/2 >= y_conv)
   {
    i_conv=i;
    j_conv=j;
    break;
   }
  }
 }
}
```

Appendix C

Developed code for Potential Flow (streamline method) imposing the velocity

```
//  Potential flow Numerical solver  (imposed velocity condition) - streamlines
    ↪ method
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <time.h>
using namespace std;
const double pi=3.141592;
//*************************************************************
// Constants
const double H=5, L=10; //Problem's geometry
const double Rg=287; // Gas constant
const double v_in=10, P_in=1.013e5, T_in=288, rho_in=P_in/Rg/T_in; // Inlet flow
    ↪ parameters
const int N=500, M=N; // Mesh density
const double fr=1.8,delta=1e-6,fr_rho=1; // Program parameters (Gauss_Seidel: fr
    ↪ =1.9 // line-by-line: fr=1.1)
const double Nd=5; // division to estimate the integral
double phi_c=v_in*H/2;
// Cylinder parameters
const double R=0.3, cx=L/2, cy=H/2; // Cylinder's geometry
// Characteristic lenght
double D_main;
// Gas cp treatement: cp(T)=a0*T^0+a1*T^1+a2*T2+a3*T^3+a4*T^4
// Air

const double a0=1034.09, a1=-2.849e-1, a2=7.817e-4, a3=-4.971e-7, a4=1.077e-10;
// Solver type: line_by_line o Gauss-Seidel
bool line_by_line=false;
// Given velocity at one point on the cylinder
const double velocitat_final=3.9248;
const double error_velocitat=1e-2;
```

```
const double theta=pi/180*(15); // 0 is the furthest point respect ot the inlet
    ↪ flow, and theta is measured counterclock wise
int i_cond=27,j_cond=26; // boundary contidion variables
//************************************************************
// Variables declaration
time_t inici,final;
struct info_node{
 double r[2];
 double phi;
 double phi_ant;
 double rho;
 double rho_ant;
 double tau;
 double T;
 double P;
 double Ax;
 double Ay;
 double cp_barra;
 double cp_barret;
 double gamma_barret;
 double vel[2]; // x and y velocity components
 double v; // velocity modulus
 int mat; // 0 solid, 1 fluid
 double ap;
 double as;
 double an;
 double aw;
 double ae;
 double bp;
 bool frontera; // determines if the node is close to the object
 double t[2]; // vtangential vector
 double n[2]; // normal vector
 double circ; //control volume circulation
};
// Functions
void preprocess(vector< vector<info_node> >& node);
void geometria(vector< vector<info_node> >& node);
int num_material(double x, double y);
void condicions_inlet(vector< vector<info_node> >& node);
void condicions_contorn(vector< vector<info_node> >& node);
void mapa_inicial(vector< vector<info_node> >& node);
void calcul_coefs(vector< vector<info_node> >& node);
double mitjana_harm(double tau1, double tau2,double d1, double d2);
void solver(vector< vector<info_node> >& node);
void solver_phi(vector< vector<info_node> >& node);
void calcul_vel(vector< vector<info_node> >& node);
double calcul_cp_barra(double T2, double T1);
double calcul_cp(double T);
double calcul_gamma(double cp);
double calcul_cp_barret(double T0, double T);
void calcul_prop_term(vector< vector<info_node> >& node);
double calcul_error_max(vector< vector<info_node> >& node);
double v_abs(double a);
void iteracio (vector< vector<info_node> >& node);
void save_phi (vector< vector<info_node> >& node);
void save_rho (vector< vector<info_node> >& node);
void save_T (vector< vector<info_node> >& node);
void save_P (vector< vector<info_node> >& node);
void save_v (vector< vector<info_node> >& node);
void save_tau (vector< vector<info_node> >& node);
void postprocess(vector< vector<info_node> >& node);
void save_posicio_mat(vector< vector<info_node> >& node);
```

```
void calcul_error(vector< vector<info_node> >& node);
void det_frontera(vector< vector<info_node> >& node);
void calcul_velocitat(vector< vector<info_node> >& node,double &velocitat);
void calcul_forces(vector< vector<info_node> >& node);
void guardar_dades(double L,double CL, double D, double CD);
void Newton_Raphson(vector< vector<info_node> >& node, bool &fi,double &phi_prev,
    ↪ double &velocitat_prev);
void canvi_phi_mat(vector< vector<info_node> >& node);
void declaracio_inicial();
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2]);
void punt_control (vector< vector<info_node> >& node);
bool condicio_pertinenca (double x,double Ax,double y,double Ay,double x_cond,
    ↪ double y_cond);
// Code
int main()
{
 time(&inici);
 declaracio_inicial();
 cout<<"Inici"<<endl;
 vector< vector<info_node> > node(N+2, vector<info_node>(M+2));
 bool fi=false;
 double phi_prev=-1;
 double velocitat_prev;
 preprocess(node);
 cout<<"Preprocess"<<endl;
 mapa_inicial(node);
 cout<<"Mapa inicial"<<endl;
 det_frontera(node);
 cout<<"Frontera"<<endl;
 calcul_coefs(node);
 cout<<"Coefs"<<endl;
 while (fi==false)
 {
  cout<<"Linia de corrent del solid: "<<phi_c<<endl;
  solver(node);
  cout<<"Solver"<<endl;
  Newton_Raphson(node,fi,phi_prev,velocitat_prev);
  cout<<"-----------------------------------------"<<endl;
  if (fi==false)
  {
   canvi_phi_mat(node);
  }
 }
 postprocess(node);

}

void preprocess(vector< vector<info_node> >& node)
{
 geometria(node);
 condicions_inlet(node);
 condicions_contorn(node);
}
void geometria(vector< vector<info_node> >& node)
{
 int material;
 double Ax=L/N, Ay=H/M;
 // Internal nodes
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
```

```
 {
  if (i==1 || i==N+1)
  {
   node[i][j].r[0]=Ax/2;
  }
  else
  {
   node[i][j].r[0]=node[i-1][j].r[0]+Ax;
  }
  if (j==1)
  {
   node[i][j].r[1]=Ay/2;
  }
  else
  {
   node[i][j].r[1]=node[i][j-1].r[1]+Ay;
  }
  node[i][j].Ax=Ax;
  node[i][j].Ay=Ay;
  material=num_material(node[i][j].r[0],node[i][j].r[1]);
  node[i][j].mat=material;
 }
}
// External nodes
// Right and left nodes
for  (int j=1;j<M+1;j++)
{
 node[0][j].r[0]=0;
 node[0][j].r[1]=node[1][j].r[1];
 node[0][j].Ax=0;
 node[0][j].mat=1;
 node[N+1][j].r[0]=L;
 node[N+1][j].r[1]=node[N][j].r[1];
 node[N+1][j].Ax=0;
 node[N+1][j].mat=1;
}
// Top and bottom nodes
for (int i=1;i<N+1;i++)
{
 node[i][0].r[0]=node[i][2].r[0];
 node[i][0].r[1]=0;
 node[i][0].Ay=0;
 node[i][0].mat=1;
 node[i][M+1].r[0]=node[i][M].r[0];
 node[i][M+1].r[1]=H;
 node[i][M+1].mat=1;
 node[i][M+1].Ay=0;
}
// Vertices
node[0][0].r[0]=0;
node[0][0].r[1]=0;
node[0][0].mat=1;
node[0][M+1].r[0]=0;
node[0][M+1].r[1]=H;
node[0][M+1].mat=1;
node[N+1][0].r[0]=L;
node[N+1][0].r[1]=0;
node[N+1][0].mat=1;
node[N+1][M+1].r[0]=L;
node[N+1][M+1].r[1]=H;
node[N+1][M+1].mat=1;
```

```
}
int num_material(double x, double y)
{
 if (sqrt((x-cx)*(x-cx)+(y-cy)*(y-cy))<=R)
 {
  return 0;
 }
 else
 {
  return 1;
 }

}
void condicions_inlet(vector< vector<info_node> >& node)
{
 node[0][0].phi=0;
 node[0][0].phi_ant=0;
 node[0][0].T=T_in;
 node[0][0].P=P_in;
 node[0][0].v=v_in;
 node[0][0].rho=rho_in;
 node[0][0].rho_ant=rho_in;
 node[0][0].tau=1;
 for (int j=1;j<M+2;j++)
 {
  node[0][j].phi=node[0][j-1].phi+v_in*(node[0][j].r[1]-node[0][j-1].r[1]);
  node[0][j].phi_ant=node[0][j-1].phi+v_in*(node[0][j].r[1]-node[0][j-1].r[1]);
  node[0][j].T=T_in;
  node[0][j].P=P_in;
  node[0][j].v=v_in;
  node[0][j].rho=rho_in;
  node[0][j].rho_ant=rho_in;
  node[0][j].tau=1;
 }
}
void condicions_contorn(vector< vector<info_node> >& node) // Boundary conditions
    ↪ (cylinder + walls)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==0)
   {
    node[i][j].phi=phi_c;
    node[i][j].phi_ant=phi_c;
    node[i][j].tau=1e30;
   }
   else if (j==0 || j==M+1)
   {
    node[i][j].phi=node[0][j].phi;
    node[i][j].phi_ant=node[0][j].phi_ant;
   }
  }
 }

}
void mapa_inicial(vector< vector<info_node> >& node)
{
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
```

```
  {
   if (node[i][j].mat==1)
   {
    node[i][j].rho_ant=rho_in;
    node[i][j].rho=rho_in;
    node[i][j].phi=node[0][j].phi;
    node[i][j].phi_ant=node[0][j].phi;
    node[i][j].tau=1;
    node[i][j].T=T_in;
   }
  }
 }
}
void calcul_coefs(vector< vector<info_node> >& node)
{
 double dpe,dpw, dps, dpn;
 // Internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
    dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
    dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
    dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
    node[i][j].ae=mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,
        ↪ node[i+1][j].Ax/2)*node[i][j].Ay/dpe;
    node[i][j].aw=mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,
        ↪ node[i-1][j].Ax/2)*node[i][j].Ay/dpw;
    node[i][j].as=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,
        ↪ node[i-1][j].Ay/2)*node[i][j].Ax/dps;
    node[i][j].an=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,
        ↪ node[i+1][j].Ay/2)*node[i][j].Ax/dpn;
    node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an;
    node[i][j].bp=0;
   }
   else
   {
    node[i][j].ae=0;
    node[i][j].aw=0;
    node[i][j].as=0;
    node[i][j].an=0;
    node[i][j].ap=1;
    node[i][j].bp=phi_c;
   }
  }
 }
 // Right nodes
 for (int j=1;j<M+1;j++)
 {
  node[N+1][j].aw=1;
  node[N+1][j].an=0;
  node[N+1][j].as=0;
  node[N+1][j].ae=0;
  node[N+1][j].ap=1;
  node[N+1][j].bp=0;
 }
}
double mitjana_harm(double tau1, double tau2,double d1, double d2)
{
```

```
 return (d1+d2)/(d1/tau1+d2/tau2);
}
void solver(vector< vector<info_node> >& node)
{
 bool trobat=false;
 while (trobat==false)
 {
  calcul_coefs(node);
  solver_phi(node);
  calcul_vel(node);
  calcul_prop_term(node);
  if(calcul_error_max(node)<delta)
  {
   trobat=true;
  }
  else
  {
   iteracio(node);
  }
 }
}
void solver_phi(vector< vector<info_node> >& node)
{
 if (line_by_line==false) // Gauss-Seidel
 {
  for (int i=1;i<N+1;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    if (node[i][j].mat==1)
    {
     node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].aw*node[i-1][j].phi
         ↪ +node[i][j].as*node[i][j-1].phi+node[i][j].an*node[i][j+1].phi)/node[i
         ↪ ][j].ap;
     node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
    }
   }
  }
  for (int j=1;j<M+1;j++)
  {
   if (node[N+1][j].mat==1)
   {
    node[N+1][j].phi=(node[N+1][j].aw*node[N][j].phi+node[N+1][j].as*node[N+1][j
        ↪ -1].phi+node[N+1][j].an*node[N+1][j+1].phi)/node[N+1][j].ap;
    node[N+1][j].phi=node[N+1][j].phi_ant+fr*(node[N+1][j].phi-node[N+1][j].
        ↪ phi_ant);
   }
  }
 }
 else // line_by_line
 {
  double P[N+2],Q[N+2];
  for (int j=1;j<M+1;j++)
  {
   P_Q_calcul(j,node,P,Q);
   for (int i=1;i<N+2;i++)
   {
    node[i][j].phi=P[i]*node[i-1][j].phi+Q[i];
   }
  }
  // Relaxing factor
  for (int i=1;i<N+2;i++)
```

```
  {
   for (int j=1;j<M+1;j++)
   {
    if (node[i][j].mat==1)
    {
     node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
    }
   }
  }
 }
}
void calcul_vel(vector< vector<info_node> >& node)
{
 double vye, vyw, vxn, vxs, vyp, vxp;
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (i<N+1)
   {
    vye=mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j].
        ↪ Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
    vyw=mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j].
        ↪ Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
    vyp=(vye+vyw)/2;
    node[i][j].vel[1]=vyp;
   }
   else
   {
    vyp=0;
   }

   vxn=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].
       ↪ Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
   vxs=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].
       ↪ Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
   vxp=(vxn+vxs)/2;
   node[i][j].vel[0]=vxp;
   node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
   if (node[i][j].mat==0)
   {
    node[i][j].circ=vye*node[i][j].Ay+vxs*node[i][j].Ax-vyw*node[i][j].Ay-vxn*node
        ↪ [i][j].Ax;
    if (isnan(node[i][j].circ))
    {
     cout<<vye<<"␣"<<vyw<<"␣"<<vxs<<"␣"<<vxn<<endl;
     cout<<i<<"␣"<<j<<endl;
    }
   }
  }
 }
 // Top and bottom nodes
 int j;
 for (int i=1;i<N+2;i++)
 {
  j=0;
  node[i][j].v=2*mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node
      ↪ [i][j+1].Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j+1].Ay;
  j=1;
  if (i<N+1)
  {
```

```
    vye=mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j].
        ↪ Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
    vyw=mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j].
        ↪ Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
    vyp=(vye+vyw)/2;
    node[i][j].vel[1]=vyp;
   }
   else
   {
    vyp=0;
   }

   vxn=mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].Ay
       ↪ /2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
   vxs=2*mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].
       ↪ Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
   vxp=(vxn+vxs)/2;
   node[i][j].vel[0]=vxp;
   node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
   j=M;
   if (i<N+1)
   {
    vye=mitjana_harm(node[i][j].tau,node[i+1][j].tau,node[i][j].Ax/2,node[i+1][j].
        ↪ Ax/2)*(node[i+1][j].phi-node[i][j].phi)/node[i][j].Ax;
    vyw=mitjana_harm(node[i][j].tau,node[i-1][j].tau,node[i][j].Ax/2,node[i-1][j].
        ↪ Ax/2)*(node[i][j].phi-node[i-1][j].phi)/node[i][j].Ax;
    vyp=(vye+vyw)/2;
    node[i][j].vel[1]=vyp;
   }
   else
   {
    vyp=0;
   }
   vxn=2*mitjana_harm(node[i][j].tau,node[i][j+1].tau,node[i][j].Ay/2,node[i][j+1].
       ↪ Ay/2)*(node[i][j+1].phi-node[i][j].phi)/node[i][j].Ay;
   vxs=mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node[i][j-1].Ay
       ↪ /2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j].Ay;
   vxp=(vxn+vxs)/2;
   node[i][j].vel[0]=vxp;
   node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
   j=M+1;
   node[i][j].v=2*mitjana_harm(node[i][j].tau,node[i][j-1].tau,node[i][j].Ay/2,node
       ↪ [i][j-1].Ay/2)*(node[i][j].phi-node[i][j-1].phi)/node[i][j-1].Ay;
  }
}
double calcul_cp_barra(double T2, double T1)
{
 double AT, T_inicial,T_final;
 if (T2==T1)
 {
  T2=T1+0.00000001;
 }
 AT=(T2-T1)/Nd;
 double T_mitjana;
 T_inicial=T1;
 double cp_barr=0,cp;
 for (int i=0;i<Nd-1;i++)
 {
  T_final=T_inicial+AT;
  T_mitjana=(T_final+T_inicial)/2;
  cp=calcul_cp(T_mitjana);
  cp_barr=cp_barr+cp*AT; // Rectangle integration
```

```
  T_inicial=T_final;
 }
 return cp_barr/(T2-T1);
}
double calcul_cp(double T)
{
 return a0+a1*T+a2*pow(T,2)+a3*pow(T,3)+a4*pow(T,4);
}
double calcul_gamma(double cp)
{
 return cp/(cp-Rg);
}
double calcul_cp_barret(double T0, double T)
{
 double AT, T_inicial,T_final;
 if (T0==T)
 {
  T0=T+0.00000001;
 }
 AT=(T0-T)/Nd;
 double T_mitjana;
 T_inicial=T;
 double cp_barr=0,cp;
 for (int i=0;i<Nd-1;i++)
 {
  T_final=T_inicial+AT;
  T_mitjana=(T_final+T_inicial)/2;
  cp=calcul_cp(T_mitjana);
  cp_barr=cp_barr+cp/T_mitjana*AT;
  T_inicial=T_final;
 }
 return cp_barr/(log(T0/T));
}
void calcul_prop_term(vector< vector<info_node> >& node)
{
 double gamma_barret;
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
    node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
    if(node[i][j].T<0)
    {
     node[i][j].T=1;
    }
    node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
    gamma_barret=calcul_gamma(node[i][j].cp_barret);
    node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
    node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
   }
  }

 }
 // Top and bottom nodes
 int j=0;
 for (int i=1;i<N+2;i++)
 {
  if (node[i][j].mat==1)
  {
```

```
    j=0;
    node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
    node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
    node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
    gamma_barret=calcul_gamma(node[i][j].cp_barret);
    node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
    node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
    j=M+1;
    node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
    node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
    node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
    gamma_barret=calcul_gamma(node[i][j].cp_barret);
    node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
    node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
    node[i][j].rho=node[i][j].rho_ant+fr_rho*(node[i][j].rho-node[i][j].rho_ant);
   }
  }
}
double calcul_error_max(vector< vector<info_node> >& node)
{
 double error=0;
 double error_phi,error_rho;
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   error_phi=v_abs(node[i][j].phi-node[i][j].phi_ant);
   error_rho=v_abs(node[i][j].rho-node[i][j].rho_ant);
   if (error_phi>error && error_phi>error_rho)
   {
    error=error_phi;
   }
   else if(error_rho>error && error_phi<error_rho)
   {
    error=error_rho;
   }
  }
 }
 cout<<delta/error*100<<"%"<<endl;
 return error;
}
double v_abs(double a)
{
 if (a>0)
 {
  return a;
 }
 else
 {
  return -a;
 }
}
void iteracio (vector< vector<info_node> >& node)
{
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].phi_ant=node[i][j].phi;
    node[i][j].rho_ant=node[i][j].rho;
```

```
    node[i][j].tau=rho_in/node[i][j].rho;
   }
  }
 }
}
void save_phi (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Phi");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].phi<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_rho (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Rho");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].rho<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_T (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Temp");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].T<<"\t";
  }
  file<<endl;
 }
 file<<N+2<<endl;
 file<<M+2<<endl;
 file.close();
}
void save_P (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Pres");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].P<<"\t";
  }
  file<<endl;
 }
 file.close();
```

```
}
void save_v (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("vel");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].v<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_tau (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("tau");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].tau<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void postprocess(vector< vector<info_node> >& node)
{
 double circulacio;
 save_phi(node);
 save_rho(node);
 save_T(node);
 save_P(node);
 save_v(node);
 save_posicio_mat(node);
 calcul_error(node);
 calcul_forces(node);
 system("graficar_flux_potencial.m");
}
void save_posicio_mat(vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Posicio");
 for (int j=0;j<M+2;j++)
 {
  file<<node[1][j].r[1]<<endl;
 }
 for (int i=0;i<N+2;i++)
 {
  file<<node[i][1].r[0]<<endl;
 }
 file.close();
 file.open("Material");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].mat<<"\t";
  }
```

```
  file<<endl;
 }
 file.close();
}
void calcul_error(vector< vector<info_node> >& node)
{
 double error=0,error2;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    error2=node[i][j].phi*node[i][j].ap-(node[i][j].ae*node[i+1][j].phi+node[i][j
        ↪ ].aw*node[i-1][j].phi+node[i][j].as*node[i][j-1].phi+node[i][j].an*node
        ↪ [i][j+1].phi);
    if (v_abs(error2)>error)
    {
     error=v_abs(error2);
    }
   }
  }
 }
 cout<<"Error comes: "<<error<<endl;
}
void det_frontera(vector< vector<info_node> >& node)
{
 double nx,ny;
 int punts_front=0;
 // Only considered Internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    if(node[i][j-1].mat==0 || node[i][j+1].mat==0|| node[i+1][j].mat==0 || node[i
        ↪ -1][j].mat==0)
    {
     node[i][j].frontera=true;
     nx=(node[i][j].r[0]-cx);
     ny=(node[i][j].r[1]-cy);
     node[i][j].n[0]=(nx)/sqrt(nx*nx+ny*ny);
     node[i][j].n[1]=(ny)/sqrt(nx*nx+ny*ny);
     node[i][j].t[0]=-node[i][j].n[1];
     node[i][j].t[1]=node[i][j].n[0];
     punts_front++;
    }
    else
    {
     node[i][j].frontera=false;
    }
   }
  }
 }
 double dS=2*pi*R/punts_front;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].frontera==true)
   {
```

```
    node[i][j].n[0]=node[i][j].n[0]*dS;
    node[i][j].n[1]=node[i][j].n[1]*dS;
    node[i][j].t[0]=node[i][j].t[0]*dS;
    node[i][j].t[1]=node[i][j].t[1]*dS;
   }
  }
 }
}
void calcul_velocitat(vector< vector<info_node> >& node,double &velocitat)
{
 velocitat=node[i_cond][j_cond].v;
 cout<<"Velocitat en el punt: "<<velocitat<<endl;
}
void calcul_forces(vector< vector<info_node> >& node)
{
 double Lift=0,D=0;
 double CL,CD;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].frontera==true && node[i][j].mat==1)
   {
    Lift=Lift-node[i][j].P*node[i][j].n[1];
    D=D-node[i][j].P*node[i][j].n[0];
   }
  }
 }
 CL=Lift/(0.5*rho_in*v_in*v_in*D_main); // adimensionalization
 CD=D/(0.5*rho_in*v_in*v_in*D_main);
 cout<<"Lift= "<<Lift<<endl;
 cout<<"Drag ="<<D<<endl;
 cout<<"CL= "<<CL<<endl;
 cout<<"CD= "<<CD<<endl;
 guardar_dades(Lift,CL,D,CD);
}
void guardar_dades(double Lift,double CL, double D, double CD)
{
 ofstream file;
 file.open("Dades_Resultats");
 file<<"DADES DEL PROBLEMA"<<endl;
 file<<"Altura: "<<H<<" m"<<endl;
 file<<"Longitud: "<<L<<" m"<<endl;
 file<<"Radi cilindre: "<<D_main<<" m"<<endl;
 file<<"Posicio del centre del cilindre: ("<<cx<<" , "<<cy<<") m"<<endl;
 file<<"Velocitat a l'entrada: "<<v_in<<" m/s"<<endl;
 file<<"Temperatura a l'entrada: "<<T_in-273.15<<" C "<<endl;
 file<<"Pressio a l'entrada: "<<P_in<<" Pa"<<endl;
 file<<"Densitat a l'entrada: "<<rho_in<<" kg/m^3"<<endl;
 file<<"Linia de corrent de l'objecte: "<<phi_c<<endl;
 file<<"Densitat de malla: "<<N<<"x"<<M<<endl;
 file<<"RESULTATS DEL PROBLEMA"<<endl;
 file<<"Lift: "<<Lift<<" N/m"<<endl;
 file<<"C_L: "<<CL<<endl;
 file<<"Drag: "<<D<<" N/m"<<endl;
 file<<"CD: "<<CD<<endl;
 time(&final);
 file<<"Temps de c lcul : "<<difftime(final,inici)<<" s"<<endl;
}
void Newton_Raphson(vector< vector<info_node> >& node, bool &fi,double &phi_prev,
    ↪ double &velocitat_prev)
{
```

```
 double velocitat;
 calcul_velocitat(node,velocitat);
 if (v_abs(velocitat-velocitat_final)<error_velocitat)
 {
  fi=true;
 }
 else
 {
  if (phi_prev==-1) //first iteration only
  {
   phi_prev=phi_c;
   velocitat_prev=velocitat;
   if(theta<0) // the streamline has a lower value that the one it should have
   {
    phi_c=v_in*H*3/4;
   }
   else
   {
    phi_c=v_in*H*1/4;
   }
  }
  else
  {
   double pendent=(velocitat-velocitat_prev)/(phi_c-phi_prev);
   phi_prev=phi_c;
   phi_c=phi_c-(velocitat-velocitat_final)/pendent;
   velocitat_prev=velocitat;
  }
 }
}
void canvi_phi_mat(vector< vector<info_node> >& node)
{
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==0)
   {
    node[i][j].phi=phi_c;
    node[i][j].phi_ant=phi_c;
   }
  }
 }
}
void declaracio_inicial()
{
 D_main=R;
}
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2])
{
 P[N+1]=node[N+1][row].aw/node[N+1][row].ap;
 Q[N+1]=(node[N+1][row].an*node[N+1][row+1].phi+node[N+1][row].as*node[N+1][row
     ↪ -1].phi+node[N+1][row].bp)/node[N+1][row].ap;
 double bp_ast;
 for (int i=N;i>0;i--)
 {
  P[i]=node[i][row].aw/(node[i][row].ap-node[i][row].ae*P[i+1]);
  bp_ast=node[i][row].an*node[i][row+1].phi+node[i][row].as*node[i][row-1].phi+
      ↪ node[i][row].bp;
  Q[i]=(bp_ast+node[i][row].ae*Q[i+1])/(node[i][row].ap-node[i][row].ae*P[i+1]);
 }
```

```
}
void punt_control (vector< vector<info_node> >& node)
{
 double x_cond, y_cond, AR=0;
 bool finalitzat=false;
 x_cond=cx+R*cos(theta);
 y_cond=cy-R*sin(theta);
 while (finalitzat==false)
 {
  for (int i=1;i<N+1;i++)
  {
   for (int j=1;j<N+1;j++)
   {
    if (condicio_pertinenca(node[i][j].r[0],node[i][j].Ax,node[i][j].r[1],node[i][
        ↪ j].Ay,x_cond,y_cond)==true)
    {
     i_cond=i;
     j_cond=j;
    }
   }
  }
  if (node[i_cond][j_cond].mat==1)
  {
   finalitzat=true;
  }
  else
  {
   AR=AR+node[i_cond][j_cond].Ax*0.51;
   x_cond=cx+(R+AR)*cos(theta);
   y_cond=cy-(R+AR)*sin(theta);
  }
 }
}
bool condicio_pertinenca (double x,double Ax,double y,double Ay,double x_cond,
    ↪ double y_cond)
{
 if (x+Ax/2>x_cond && x-Ax/2<=x_cond && y+Ay/2>y_cond && y-Ay/2<=y_cond)
 {
  return true;
 }
 else
 {
  return false;
 }
}
```

Appendix D

Developed code for Potential Flow (velocity potential method)

```
//  Potential flow Numerical solver  (imposed circulation condition) - velocity
    ↪ potential method
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <time.h>
using namespace std;
// Constant numbers
const double pi=3.141592;
//*************************************************************
// Constants
const double H=5, L=10, W=1; //Problem's geometry
const double Rg=287; // gas constant
const double v_in=10, P_in=1.013e5, T_in=288, rho_in=P_in/Rg/T_in; // Inlet flow
    ↪ parameters
const int N=500, M=N; // Mesh density
const double fr=1,delta=1e-8,fr_rho=1; // Program parameters
const double Nd=5; // division to estimate the integral
double phi_c=50;
// Cylinder parameters
const double R=0.3, cx=L/2, cy=H/2; // Cylinder's geometry
const double w=10; //Cylinder's rotation
// Desired circulation around the object
double circulacio_final;
const double error_circ=1e-5;
// Characteristic lenght
double D_main;
// Gas cp treatemet: cp(T)=a0*T^0+a1*T^1+a2*T2+a3*T^3+a4*T^4
// Air
const double a0=1034.09, a1=-2.849e-1, a2=7.817e-4, a3=-4.971e-7, a4=1.077e-10;
// Helium
//const double a0=5188, a1=0, a2=0, a3=0, a4=0;
// Solver type: line_by_line or Gauss-Seidel
bool line_by_line=false;
// Convergence analysis point
double x_conv=5, y_conv=2.9;
```

```
int i_conv, j_conv;
//*************************************************************
// Variable declaration
time_t inici,final;
struct info_node{
 double r[2];
 double phi;
 double phi_ant;
 double phi_sup;
 double rho;
 double rho_ant;
 double rho_sup;
 double tau;
 double T;
 double T_ant;
 double P;
 double P_ant;
 double Ax;
 double Ay;
 double cp_barra;
 double cp_barret;
 double gamma_barret;
 double vel[2]; // x and y velocity components
 double v; // velocity modulus
 int mat; // 0 solid, 1 fluid
 double ap;
 double as;
 double an;
 double aw;
 double ae;
 double bp;
 bool frontera; // determines if the node is close to the object
 double t[2]; // vtangential vector
 double n[2]; // normal vector
};
// Functions
void preprocess(vector< vector<info_node> >& node);
void geometria(vector< vector<info_node> >& node);
int num_material(double x, double y);
void condicions_inlet(vector< vector<info_node> >& node);
void condicions_contorn(vector< vector<info_node> >& node);
void mapa_inicial(vector< vector<info_node> >& node);
void calcul_coefs(vector< vector<info_node> >& node);
double mitjana_harm(double tau1, double tau2,double d1, double d2);
void solver(vector< vector<info_node> >& node);
void solver_phi(vector< vector<info_node> >& node);
void calcul_vel(vector< vector<info_node> >& node);
double calcul_cp_barra(double T2, double T1);
double calcul_cp(double T);
double calcul_gamma(double cp);
double calcul_cp_barret(double T0, double T);
void calcul_prop_term(vector< vector<info_node> >& node);
double calcul_error_max(vector< vector<info_node> >& node);
double v_abs(double a);
void iteracio (vector< vector<info_node> >& node);
void save_phi (vector< vector<info_node> >& node);
void save_rho (vector< vector<info_node> >& node);
void save_T (vector< vector<info_node> >& node);
void save_P (vector< vector<info_node> >& node);
void save_v (vector< vector<info_node> >& node);
void save_tau (vector< vector<info_node> >& node);
void postprocess(vector< vector<info_node> >& node);
```

```
void save_posicio_mat(vector< vector<info_node> >& node);
void calcul_error(vector< vector<info_node> >& node);
void det_frontera(vector< vector<info_node> >& node);
void calcul_forces(vector< vector<info_node> >& node);
void guardar_dades(double L,double CL, double D, double CD);
bool perfil_NACA(double x_global, double y_global);
void gir(double x_global,double y_global,double rotacio[2][2],double &x,double &y)
    ↪ ;
void canvi_temps(vector< vector<info_node> >& node);
void declaracio_inicial();
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2]);
void flux_massic_cos(vector< vector<info_node> >& node);
// Code
int main()
{
 time(&inici);
 declaracio_inicial();
 vector< vector<info_node> > node(N+2, vector<info_node>(M+2));
 bool fi=false;
 preprocess(node);
 mapa_inicial(node);
 det_frontera(node);
 solver(node);
 postprocess(node);

}
void preprocess(vector< vector<info_node> >& node)
{
 geometria(node);
 condicions_inlet(node);
 condicions_contorn(node);
}
void geometria(vector< vector<info_node> >& node)
{
 int material;
 double Ax=L/N, Ay=H/M;
 // Internal nodes
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (i==1 || i==N+1)
   {
    node[i][j].r[0]=Ax/2;
   }
   else
   {
    node[i][j].r[0]=node[i-1][j].r[0]+Ax;
   }
   if (j==1)
   {
    node[i][j].r[1]=Ay/2;
   }
   else
   {
    node[i][j].r[1]=node[i][j-1].r[1]+Ay;
   }
   node[i][j].Ax=Ax;
   node[i][j].Ay=Ay;
   material=num_material(node[i][j].r[0],node[i][j].r[1]);
   node[i][j].mat=material;
```

```
  }
 }
 // External nodes
 // Right and left nodes
 for  (int j=1;j<M+1;j++)
 {
  node[0][j].r[0]=0;
  node[0][j].r[1]=node[1][j].r[1];
  node[0][j].Ax=0;
  node[0][j].mat=1;
  node[N+1][j].r[0]=L;
  node[N+1][j].r[1]=node[N][j].r[1];
  node[N+1][j].Ax=0;
  node[N+1][j].mat=1;
 }
 // Top and bottom nodes
 for (int i=1;i<N+1;i++)
 {
  node[i][0].r[0]=node[i][2].r[0];
  node[i][0].r[1]=0;
  node[i][0].Ay=0;
  node[i][0].mat=1;
  node[i][M+1].r[0]=node[i][M].r[0];
  node[i][M+1].r[1]=H;
  node[i][M+1].mat=1;
  node[i][M+1].Ay=0;
 }
 // Vertices
 node[0][0].r[0]=0;
 node[0][0].r[1]=0;
 node[0][0].mat=1;
 node[0][M+1].r[0]=0;
 node[0][M+1].r[1]=H;
 node[0][M+1].mat=1;
 node[N+1][0].r[0]=L;
 node[N+1][0].r[1]=0;
 node[N+1][0].mat=1;
 node[N+1][M+1].r[0]=L;
 node[N+1][M+1].r[1]=H;
 node[N+1][M+1].mat=1;

}
int num_material(double x, double y)
{
 if (sqrt((x-cx)*(x-cx)+(y-cy)*(y-cy))<=R)
 {
  return 0;
 }
 else
 {
  return 1;
 }

}
void condicions_inlet(vector< vector<info_node> >& node)
{
 node[0][0].phi=0;
 node[0][0].phi_sup=0;
 node[0][0].T=T_in;
 node[0][0].P=P_in;
 node[0][0].v=v_in;
 node[0][0].rho=rho_in;
```

```
 node[0][0].rho_sup=rho_in;
 for (int j=1;j<M+2;j++)
 {
  node[0][j].phi=0;
  node[0][j].phi_sup=0;
  node[0][j].T=T_in;
  node[0][j].P=P_in;
  node[0][j].v=v_in;
  node[0][j].rho=rho_in;
  node[0][j].rho_sup=rho_in;
 }
}
void condicions_contorn(vector< vector<info_node> >& node)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==0)
   {
    node[i][j].phi=phi_c;
    node[i][j].phi_sup=phi_c;
    node[i][j].rho=1e-30;
   }
  }
 }

}
void mapa_inicial(vector< vector<info_node> >& node)
{
 node[0][0].phi=0;
 node[0][0].phi_sup=0;
 node[0][0].rho=rho_in;
 node[0][0].rho_sup=rho_in;
 node[0][0].T=T_in;
 double Ax;
 for (int i=1;i<N+2;i++)
 {
  Ax=node[i][0].r[0]-node[i-1][0].r[0];
  node[i][0].phi=node[i-1][0].phi+v_in*Ax;
  node[i][0].phi_sup=node[i-1][0].phi+v_in*Ax;
  node[i][0].rho=rho_in;
  node[i][0].rho_sup=rho_in;
  node[i][0].T=T_in;
 }
 for (int i=0;i<N+2;i++)
 {
  for (int j=1;j<M+2;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].rho_sup=rho_in;
    node[i][j].rho=rho_in;
    node[i][j].phi=node[i][j-1].phi;
    node[i][j].phi_sup=node[i][j-1].phi_sup;
    node[i][j].rho_ant=rho_in;
    node[i][j].T=T_in;
   }
  }
 }
}
void calcul_coefs(vector< vector<info_node> >& node)
```

```
{
 double dpe,dpw, dps, dpn, Se,Sw,Sn,Ss;
 // Internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
    dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
    dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
    dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
    Se=node[i][j].Ay*W;
    Sw=node[i][j].Ay*W;
    Sn=node[i][j].Ax*W;
    Ss=node[i][j].Ax*W;
    node[i][j].ae=mitjana_harm(node[i][j].rho,node[i+1][j].rho,node[i][j].Ax/2,
        ↪ node[i+1][j].Ax/2)*Se/dpe;
    node[i][j].aw=mitjana_harm(node[i][j].rho,node[i-1][j].rho,node[i][j].Ax/2,
        ↪ node[i-1][j].Ax/2)*Sw/dpw;
    node[i][j].as=mitjana_harm(node[i][j].rho,node[i][j-1].rho,node[i][j].Ay/2,
        ↪ node[i-1][j].Ay/2)*Ss/dps;
    node[i][j].an=mitjana_harm(node[i][j].rho,node[i][j+1].rho,node[i][j].Ay/2,
        ↪ node[i+1][j].Ay/2)*Sn/dpn;
    node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an;
    node[i][j].bp=0;
   }
   else
   {
    node[i][j].ae=0;
    node[i][j].aw=0;
    node[i][j].as=0;
    node[i][j].an=0;
    node[i][j].ap=1;
    node[i][j].bp=phi_c;
   }
  }
 }
 // Right and left nodes
 for (int j=1;j<M+1;j++)
 {
  node[N+1][j].aw=0;
  node[N+1][j].an=0;
  node[N+1][j].as=1;
  node[N+1][j].ae=0;
  node[N+1][j].ap=1;
  node[N+1][j].bp=0;
  node[0][j].aw=0;
  node[0][j].an=0;
  node[0][j].as=0;
  node[0][j].ae=1;
  node[0][j].ap=1;
  node[0][j].bp=-v_in*node[1][j].Ax/2;

 }
 // Top and bottom nodes
 for (int i=1;i<N+1;i++)
 {
  node[i][0].ap=1;
  node[i][0].as=0;
  node[i][0].aw=0;
```

```
  node[i][0].an=1;
  node[i][0].as=0;
  node[i][0].bp=0;
  node[i][M+1].ap=1;
  node[i][M+1].as=1;
  node[i][M+1].aw=0;
  node[i][M+1].an=0;
  node[i][M+1].ae=0;
  node[i][M+1].bp=0;
 }
 // Verctices
 node[0][0].ap=1;
 node[0][0].as=0;
 node[0][0].an=1;
 node[0][0].ae=0;
 node[0][0].aw=0;
 node[0][0].bp=0;
 node[N+1][0].ap=1;
 node[N+1][0].as=0;
 node[N+1][0].an=0;
 node[N+1][0].ae=0;
 node[N+1][0].aw=0;
 node[N+1][0].bp=v_in*L;
 node[0][M+1].ap=1;
 node[0][M+1].as=1;
 node[0][M+1].an=0;
 node[0][M+1].ae=0;
 node[0][M+1].aw=0;
 node[0][M+1].bp=0;
 node[N+1][M+1].ap=1;
 node[N+1][M+1].as=1;
 node[N+1][M+1].an=0;
 node[N+1][M+1].ae=0;
 node[N+1][M+1].aw=0;
 node[N+1][M+1].bp=0;
}
double mitjana_harm(double tau1, double tau2,double d1, double d2)
{

 return (d1+d2)/(d1/tau1+d2/tau2);
}
void solver(vector< vector<info_node> >& node)
{
 bool trobat=false;
 while (trobat==false)
 {
  calcul_coefs(node);
  solver_phi(node);
  calcul_vel(node);
  calcul_prop_term(node);
  if(calcul_error_max(node)<delta)
  {
   trobat=true;
  }
  else
  {
   iteracio(node);
  }
 }
}
void solver_phi(vector< vector<info_node> >& node)
{
```

```
if (line_by_line==false) // Gauss-Seidel
{
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].aw*node[i-1][j].phi
        ↪ +node[i][j].as*node[i][j-1].phi+node[i][j].an*node[i][j+1].phi+node[i
        ↪ ][j].bp)/node[i][j].ap;
    node[i][j].phi=node[i][j].phi_sup+fr*(node[i][j].phi-node[i][j].phi_sup);
   }
  }
 }
 for (int j=1;j<M+1;j++)
 {
  node[0][j].phi=(node[0][j].ae*node[1][j].phi+node[0][j].as*node[0][j-1].phi+
      ↪ node[0][j].an*node[0][j+1].phi+node[0][j].bp)/node[0][j].ap;
  node[0][j].phi=node[0][j].phi_sup+fr*(node[0][j].phi-node[0][j].phi_sup);
  node[N+1][j].phi=(node[N+1][j].aw*node[N][j].phi+node[N+1][j].as*node[N+1][j
      ↪ -1].phi+node[N+1][j].an*node[N+1][j+1].phi+node[N+1][j].bp)/node[N+1][j
      ↪ ].ap;
  node[N+1][j].phi=node[N+1][j].phi_sup+fr*(node[N+1][j].phi-node[N+1][j].phi_sup
      ↪ );
 }
 for (int i=1;i<N+1;i++)
 {
  node[i][0].phi=(node[i][0].ae*node[i+1][0].phi+node[i][0].aw*node[i-1][0].phi+
      ↪ node[i][0].an*node[i][1].phi+node[i][0].bp)/node[i][0].ap;
  node[i][0].phi=node[i][0].phi_sup+fr*(node[i][0].phi-node[i][0].phi_sup);
  node[i][M+1].phi=(node[i][M+1].aw*node[i-1][M+1].phi+node[i][M+1].ae*node[i+1][
      ↪ M+1].phi+node[i][M+1].as*node[i][M].phi+node[i][M+1].bp)/node[i][M+1].ap
      ↪ ;
  node[i][M+1].phi=node[i][M+1].phi_sup+fr*(node[i][M+1].phi-node[i][M+1].phi_sup
      ↪ );
 }
 int i,j;
 i=0;
 j=0;
 node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].an*node[i][j+1].phi+
     ↪ node[i][j].bp)/node[i][j].ap;
 node[i][j].phi=node[i][j].phi_sup+fr*(node[i][j].phi-node[i][j].phi_sup);
 i=0;
 j=M+1;
 node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].as*node[i][j-1].phi+
     ↪ node[i][j].bp)/node[i][j].ap;
 node[i][j].phi=node[i][j].phi_sup+fr*(node[i][j].phi-node[i][j].phi_sup);
 i=N+1;
 j=0;
 node[i][j].phi=(node[i][j].aw*node[i-1][j].phi+node[i][j].an*node[i][j+1].phi+
     ↪ node[i][j].bp)/node[i][j].ap;
 node[i][j].phi=node[i][j].phi_sup+fr*(node[i][j].phi-node[i][j].phi_sup);
 i=N+1;
 j=M+1;
 node[i][j].phi=(node[i][j].aw*node[i-1][j].phi+node[i][j].as*node[i][j-1].phi+
     ↪ node[i][j].bp)/node[i][j].ap;
 node[i][j].phi=node[i][j].phi_sup+fr*(node[i][j].phi-node[i][j].phi_sup);
}
else // line_by_line
{
 double P[N+2],Q[N+2];
```

```
  for (int j=0;j<M+2;j++)
  {
   P_Q_calcul(j,node,P,Q);
   node[N+1][j].phi=Q[N+1];
   for (int i=N;i>=0;i--)
   {
    node[i][j].phi=P[i]*node[i+1][j].phi+Q[i];
   }
  }
  // Relaxing factor
  for (int i=1;i<N+2;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    if (node[i][j].mat==1)
    {
     node[i][j].phi=node[i][j].phi_sup+fr*(node[i][j].phi-node[i][j].phi_sup);
    }
   }
  }
 }
}
void calcul_vel(vector< vector<info_node> >& node)
{
 double vxe, vxw, vyn, vys, vyp, vxp, dpe, dpw, dpn, dps;
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (i<N+1)
   {
    dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
    dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
    dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
    dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
    if (node[i+1][j].mat==0)
    {
     vxe=0;
    }
    else
    {
     vxe=(node[i+1][j].phi-node[i][j].phi)/dpe;
    }
    if (node[i-1][j].mat==0)
    {
     vxw=0;
    }
    else
    {
     vxw=(node[i][j].phi-node[i-1][j].phi)/dpw;
    }
    if (node[i][j+1].mat==0)
    {
     vyn=0;
    }
    else
    {
     vyn=(node[i][j+1].phi-node[i][j].phi)/dpn;
    }
    if (node[i][j-1].mat==0)
    {
     vys=0;
```

```
    }
    else
    {
     vys=(node[i][j].phi-node[i][j-1].phi)/dps;
    }
    vxp=(vxe+vxw)/2;
    node[i][j].vel[0]=vxp;
    vyp=(vyn+vys)/2;
    node[i][j].vel[1]=vyp;
   }
   else
   {
    vxp=2*(node[i][j].phi-node[i-1][j].phi)/node[i-1][j].Ax;
    node[i][j].vel[0]=vxp;
    vyp=0;
    node[i][j].vel[1]=vyp;
   }
   node[i][j].v=sqrt(vyp*vyp+vxp*vxp);
  }
 }
 // Inlet nodes
 for (int j=1;j<M+1;j++)
 {
  node[0][j].v=v_in;
 }
 // Top and bottom nodes
 int j;
 for (int i=1;i<N+2;i++)
 {
  if (i<N+1)
  {
   j=0;
   dpe=node[i][j+1].Ax/2+node[i+1][j+1].Ax/2;
   dpw=node[i][j+1].Ax/2+node[i-1][j+1].Ax/2;
   vxe=(node[i+1][j].phi-node[i][j].phi)/dpe;
   vxw=(node[i][j].phi-node[i-1][j].phi)/dpw;
   node[i][j].v=v_abs(vxe+vxw)/2;
   j=M+1;
   dpe=node[i][j-1].Ax/2+node[i+1][j-1].Ax/2;
   dpw=node[i][j-1].Ax/2+node[i-1][j-1].Ax/2;
   vxe=(node[i+1][j].phi-node[i][j].phi)/dpe;
   vxw=(node[i][j].phi-node[i-1][j].phi)/dpw;
   node[i][j].v=v_abs(vxe+vxw)/2;
  }
  else
  {
   j=0;
   node[i][j].v=(node[i-1][j].v+node[i][j+1].v)/2;
   j=M+1;
   node[i][j].v=(node[i-1][j].v+node[i][j-1].v)/2;
  }
 }
}
double calcul_cp_barra(double T2, double T1)
{
 double AT, T_inicial,T_final;
 if (T2==T1)
 {
  T2=T1+0.00000001;
 }
 AT=(T2-T1)/Nd;
 double T_mitjana;
```

```
 T_inicial=T1;
 double cp_barr=0,cp;
 for (int i=0;i<Nd-1;i++)
 {
  T_final=T_inicial+AT;
  T_mitjana=(T_final+T_inicial)/2;
  cp=calcul_cp(T_mitjana);
  cp_barr=cp_barr+cp*AT; // Rectangle integration
  T_inicial=T_final;
 }
 return cp_barr/(T2-T1);
}
double calcul_cp(double T)
{
 return a0+a1*T+a2*pow(T,2)+a3*pow(T,3)+a4*pow(T,4);
}
double calcul_gamma(double cp)
{
 return cp/(cp-Rg);
}
double calcul_cp_barret(double T0, double T)
{
 double AT, T_inicial,T_final;
 if (T0==T)
 {
  T0=T+0.00000001;
 }
 AT=(T0-T)/Nd;
 double T_mitjana;
 T_inicial=T;
 double cp_barr=0,cp;
 for (int i=0;i<Nd-1;i++)
 {
  T_final=T_inicial+AT;
  T_mitjana=(T_final+T_inicial)/2;
  cp=calcul_cp(T_mitjana);
  cp_barr=cp_barr+cp/T_mitjana*AT;
  T_inicial=T_final;
 }
 return cp_barr/(log(T0/T));
}
void calcul_prop_term(vector< vector<info_node> >& node)
{
 double gamma_barret;
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
    node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
    if(node[i][j].T<0)
    {
     node[i][j].T=1;
    }
    node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
    gamma_barret=calcul_gamma(node[i][j].cp_barret);
    node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
    node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
   }
  }
```

```
 }
 // Top and bottom nodes
 int j=0;
 for (int i=1;i<N+2;i++)
 {
  if (node[i][j].mat==1)
  {
   j=0;
   node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
   node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
   if(node[i][j].T<0)
    {
     node[i][j].T=1;
    }
   node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
   gamma_barret=calcul_gamma(node[i][j].cp_barret);
   node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
   node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
   j=M+1;
   node[i][j].cp_barra=calcul_cp_barra(node[i][j].T,T_in);
   node[i][j].T=T_in+(v_in*v_in-node[i][j].v*node[i][j].v)/2/node[i][j].cp_barra;
   if(node[i][j].T<0)
    {
     node[i][j].T=1;
    }
   node[i][j].cp_barret=calcul_cp_barret(node[i][j].T,T_in);
   gamma_barret=calcul_gamma(node[i][j].cp_barret);
   node[i][j].P=P_in*pow(node[i][j].T/T_in,gamma_barret/(gamma_barret-1));
   node[i][j].rho=node[i][j].P/Rg/node[i][j].T;
   node[i][j].rho=node[i][j].rho_sup+fr_rho*(node[i][j].rho-node[i][j].rho_sup);
  }
 }
}
double calcul_error_max(vector< vector<info_node> >& node)
{
 double error=0;
 double error_phi,error_rho;
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   error_phi=v_abs(node[i][j].phi-node[i][j].phi_sup);
   error_rho=v_abs(node[i][j].rho-node[i][j].rho_sup);
   if (error_phi>error && error_phi>error_rho)
   {
    error=error_phi;
   }
   else if(error_rho>error && error_phi<error_rho)
   {
    error=error_rho;
   }
  }
 }
 cout<<delta/error*100<<"%"<<endl;
 return error;
}
double v_abs(double a)
{
 if (a>0)
 {
  return a;
```

```
 }
 else
 {
  return -a;
 }
}
void iteracio (vector< vector<info_node> >& node)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].phi_sup=node[i][j].phi;
    node[i][j].rho_sup=node[i][j].rho;
   }
  }
 }
}
void save_phi (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Phi");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].phi<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_rho (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Rho");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].rho<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_T (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Temp");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].T<<"\t";
  }
  file<<endl;
 }
 file<<N+2<<endl;
 file<<M+2<<endl;
```

```
 file.close();
}
void save_P (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Pres");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].P<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_v (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("vel");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].v<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_tau (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("tau");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].rho<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void postprocess(vector< vector<info_node> >& node)
{
 double circulacio;
 save_phi(node);
 save_rho(node);
 save_T(node);
 save_P(node);
 save_v(node);
 save_posicio_mat(node);
 calcul_error(node);
 calcul_forces(node);
 flux_massic_cos(node);
 system("graficar_flux_potencial.m");
}
void save_posicio_mat(vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Posicio");
```

```
 for (int j=0;j<M+2;j++)
 {
  file<<node[1][j].r[1]<<endl;
 }
 for (int i=0;i<N+2;i++)
 {
  file<<node[i][1].r[0]<<endl;
 }
 file.close();
 file.open("Material");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].mat<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void calcul_error(vector< vector<info_node> >& node)
{
 double error=0,error2;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    error2=node[i][j].phi*node[i][j].ap-(node[i][j].ae*node[i+1][j].phi+node[i][j
        ↪ ].aw*node[i-1][j].phi+node[i][j].as*node[i][j-1].phi+node[i][j].an*node
        ↪ [i][j+1].phi);
    if (v_abs(error2)>error)
    {
     error=v_abs(error2);
    }
   }
  }
 }
 cout<<"Error comes: "<<error<<endl;
}
void det_frontera(vector< vector<info_node> >& node)
{
 double nx,ny;
 int punts_front=0;
 // Only considered internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].mat==1)
   {
    if(node[i][j-1].mat==0 || node[i][j+1].mat==0|| node[i+1][j].mat==0 || node[i
        ↪ -1][j].mat==0)
    {
     node[i][j].frontera=true;
     nx=(node[i][j].r[0]-cx);
     ny=(node[i][j].r[1]-cy);
     node[i][j].n[0]=(nx)/sqrt(nx*nx+ny*ny);
     node[i][j].n[1]=(ny)/sqrt(nx*nx+ny*ny);
     node[i][j].t[0]=-node[i][j].n[1];
     node[i][j].t[1]=node[i][j].n[0];
```

```
      punts_front++;
     }
     else
     {
      node[i][j].frontera=false;
     }
    }
   }
  }
  double dS=2*pi*R/punts_front;
  for (int i=1;i<N+1;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    if (node[i][j].frontera==true)
    {
     node[i][j].n[0]=node[i][j].n[0]*dS;
     node[i][j].n[1]=node[i][j].n[1]*dS;
     node[i][j].t[0]=node[i][j].t[0]*dS;
     node[i][j].t[1]=node[i][j].t[1]*dS;
    }
   }
  }
}
void calcul_forces(vector< vector<info_node> >& node)
{
 double Lift=0,D=0;
 double CL,CD;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].frontera==true && node[i][j].mat==1)
   {
    Lift=Lift-node[i][j].P*node[i][j].n[1];
    D=D-node[i][j].P*node[i][j].n[0];
   }
  }
 }
 CL=Lift/(0.5*rho_in*v_in*v_in*D_main); // aadimensionalization
 CD=D/(0.5*rho_in*v_in*v_in*D_main);
 cout<<"Lift=␣"<<Lift<<endl;
 cout<<"Drag␣="<<D<<endl;
 cout<<"CL=␣"<<CL<<endl;
 cout<<"CD=␣"<<CD<<endl;
 guardar_dades(Lift,CL,D,CD);
}
void guardar_dades(double Lift,double CL, double D, double CD)
{
 ofstream file;
 file.open("Dades_Resultats_potencial_velocitat");
 file<<"DADES␣DEL␣PROBLEMA"<<endl;
 file<<"Altura:␣"<<H<<"␣m"<<endl;
 file<<"Longitud:␣"<<L<<"␣m"<<endl;
 file<<"Radi␣cilindre:␣"<<D_main<<"␣m"<<endl;
 file<<"Posicio␣del␣centre␣del␣cilindre:␣("<<cx<<"␣,␣"<<cy<<")␣m"<<endl;
 file<<"Velocitat␣a␣l'entrada:␣"<<v_in<<"␣m/s"<<endl;
 file<<"Temperatura␣a␣l'entrada:␣"<<T_in-273.15<<"␣ C "<<endl;
 file<<"Pressio␣a␣l'entrada:␣"<<P_in<<"␣Pa"<<endl;
 file<<"Densitat␣a␣l'entrada:␣"<<rho_in<<"␣kg/m^3"<<endl;
 file<<"Linia␣de␣corrent␣de␣l'objecte:␣"<<phi_c<<endl;
 file<<"Densitat␣de␣malla:␣"<<N<<"x"<<M<<endl;
```

```
 file<<"RESULTATS␣DEL␣PROBLEMA"<<endl;
 file<<"Lift:␣"<<Lift<<"␣N/m"<<endl;
 file<<"C_L:␣"<<CL<<endl;
 file<<"Drag:␣"<<D<<"␣N/m"<<endl;
 file<<"CD:␣"<<CD<<endl;
 time(&final);
 file<<"Temps␣de␣ c lcul :␣"<<difftime(final,inici)<<"␣s"<<endl;
}
void canvi_temps(vector< vector<info_node> >& node)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (node[i][j].mat==1)
   {
    node[i][j].rho_sup=node[i][j].rho;
    node[i][j].rho_ant=node[i][j].rho;
    node[i][j].phi_sup=node[i][j].phi;
   }
  }
 }
}
void declaracio_inicial()
{
 D_main=R;
}
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2])
{
 P[0]=node[0][row].ae/node[0][row].ap;
 if (row==0)
 {
  Q[0]=(node[0][row].an*node[0][row+1].phi+node[0][row].bp)/node[0][row].ap;
 }
 else if (row==M+1)
 {
  Q[0]=(node[0][row].as*node[0][row-1].phi+node[0][row].bp)/node[0][row].ap;
 }
 else
 {
  Q[0]=(node[0][row].an*node[0][row+1].phi+node[0][row].as*node[0][row-1].phi+node
      ↪ [0][row].bp)/node[0][row].ap;
 }
 double bp_ast;
 for (int i=1;i<N+2;i++)
 {
  P[i]=node[i][row].ae/(node[i][row].ap-node[i][row].aw*P[i-1]);
  if (row==0)
  {
   bp_ast=node[i][row].an*node[i][row+1].phi+node[i][row].bp;
  }
  else if (row==M+1)
  {
   bp_ast=node[i][row].as*node[i][row-1].phi+node[i][row].bp;
  }
  else
  {
   bp_ast=node[i][row].an*node[i][row+1].phi+node[i][row].as*node[i][row-1].phi+
       ↪ node[i][row].bp;
  }
  Q[i]=(bp_ast+node[i][row].aw*Q[i-1])/(node[i][row].ap-node[i][row].aw*P[i-1]);
```

```
 }
}
void flux_massic_cos(vector< vector<info_node> >& node)
{
 double entrant=0, sortint=0, vxe,vxw,vys,vyn,dpe,dpw,dps,dpn;
 for (int i=0; i<N+2; i++)
 {
  for (int j=0; j<M+2; j++)
  {
   if (node[i][j].mat==1 && node[i][j].frontera==true)
   {
    if (node[i][j-1].mat==0 && node[i+1][j].mat==0)
    {
     dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
     dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
     vxe=(node[i+1][j].phi-node[i][j].phi)/dpe;
     vys=(node[i][j].phi-node[i][j-1].phi)/dps;
     if (vxe>0)
     {
      sortint=sortint+vxe;
     }
     else
     {
      entrant=entrant-vxe;
     }
     if (vys>0)
     {
      entrant=entrant+vys;
     }
     else
     {
      sortint=sortint-vys;
     }
    }
    else if (node[i][j-1].mat==0 && node[i-1][j].mat==0)
    {
     dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
     dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
     vxw=(node[i][j].phi-node[i-1][j].phi)/dpw;
     vys=(node[i][j].phi-node[i][j-1].phi)/dps;
     if (vxw<0)
     {
      sortint=sortint-vxw;
     }
     else
     {
      entrant=entrant+vxw;
     }
     if (vys>0)
     {
      entrant=entrant+vys;
     }
     else
     {
      sortint=sortint-vys;
     }
    }
    else if (node[i][j+1].mat==0 && node[i+1][j].mat==0)
    {
     dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
     dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
     vxe=(node[i+1][j].phi-node[i][j].phi)/dpe;
```

```
 vyn=(node[i][j+1].phi-node[i][j].phi)/dpn;
 if (vxe>0)
 {
  sortint=sortint+vxe;
 }
 else
 {
  entrant=entrant-vxe;
 }
 if (vyn<0)
 {
  entrant=entrant-vyn;
 }
 else
 {
  sortint=sortint+vyn;
 }
}
else if (node[i][j+1].mat==0 && node[i-1][j].mat==0)
{
 dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
 dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
 vxw=(node[i][j].phi-node[i-1][j].phi)/dpw;
 vyn=(node[i][j+1].phi-node[i][j].phi)/dpn;
 if (vxw<0)
 {
  sortint=sortint-vxw;
 }
 else
 {
  entrant=entrant+vxw;
 }
 if (vyn<0)
 {
  entrant=entrant-vyn;
 }
 else
 {
  sortint=sortint+vyn;
 }
}
else if (node[i][j+1].mat==0)
{
 dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
 vyn=(node[i][j+1].phi-node[i][j].phi)/dpn;
 if (vyn<0)
 {
  entrant=entrant-vyn;
 }
 else
 {
  sortint=sortint+vyn;
 }
}
else if (node[i-1][j].mat==0)
{
 dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
 vxw=(node[i][j].phi-node[i-1][j].phi)/dpw;
 if (vxw<0)
 {
  sortint=sortint-vxw;
 }
```

```
      else
      {
       entrant=entrant+vxw;
      }
     }
     else if (node[i][j-1].mat==0)
     {
      dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
      vys=(node[i][j].phi-node[i][j-1].phi)/dps;
      if (vys>0)
      {
       entrant=entrant+vys;
      }
      else
      {
       sortint=sortint-vys;
      }
     }
     else if (node[i+1][j].mat==0)
     {
      dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
      vxe=(node[i+1][j].phi-node[i][j].phi)/dpe;
      if (vxe>0)
      {
       sortint=sortint+vxe;
      }
      else
      {
       entrant=entrant-vxe;
      }
     }
    }

   }
  }
  cout<<"----------------------------"<<endl;
  cout<<"Flux incident: "<<entrant<<endl;
  cout<<"Flux sortint: "<<sortint<<endl;
}
```

Appendix E

Developed code for the convection-diffusion equation

```
// Numerical program to solve the convection-diffusion equation
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <time.h>
using namespace std;
// Constants de numeros
const double pi=3.141592;
//
    ↪ ********************************************************************************
    ↪
// Constants
const double H=1, L=2; //Geometry
const int N=500, M=500;
const double v0=1,P_in=1.013e5, rho_in=1; // Inlet fow parameters
const double fr=0.4,delta=1e-8; // Program parameters (Gauss_Seidel: fr=1.9 //
    ↪ line-by-line: fr=1.1)
int SCHEME=0; // type of convective scheme (0=UDS, 1=CDS, 2=SUDS, 3=QUICK, 4=SMART
    ↪ )
const int Nd=20;
const int exercise=3;
double phi_L,phi_R,phi_B,phi_T;
double jL,jR,jB,jT;
 // Exercise 1
 const double phi_in=288,phi_out=300;
 // Exercise 2
 const double phi_low=288,phi_high=400;
 // Exercise 3
 const double j_out=0,phi_bound= 1 -tanh(10);
 double phi_left[N];
 // Exercise 4 --> vertical 1
// Physical parameters
const double Pe=1e3;
const double gamma=rho_in*v0*L/Pe;

// Type of solver: line_by_line o Gauss-Seidel
```

```
bool line_by_line=false;
// Convergence analysis' point
double x_conv=5, y_conv=2.9;
int i_conv, j_conv;
//
    ↪ ********************************************************************************
    ↪
// Variables
time_t inici,final;
struct info_node{
 double r[2];
 double phi;
 double phi_ant;
 double Ax;
 double Ay;
 double vel[2]; // x and y velocity components
 double v; // velocity modulus
 double ap;
 double as;
 double an;
 double aw;
 double ae;
 double bp;
 double me;
 double mw;
 double ms;
 double mn;
 double De;
 double Dw;
 double Dn;
 double Ds;
 double tau;
 double taup;
 double Spc;
 double Spp;
 int BC; //boundary condition (0=dirichlet; 1 = Neumann)
};
// Functions declaration
void preprocess(vector< vector<info_node> >& node);
void geometria(vector< vector<info_node> >& node);
void condicions_contorn(vector< vector<info_node> >& node);
void mapa_inicial(vector< vector<info_node> >& node);
void calcul_coefs(vector< vector<info_node> >& node);
void solver(vector< vector<info_node> >& node);
void solver_phi(vector< vector<info_node> >& node);
double calcul_error_max(vector< vector<info_node> >& node);
double v_abs(double a);
void iteracio (vector< vector<info_node> >& node);
void save_phi (vector< vector<info_node> >& node);
void save_P (vector< vector<info_node> >& node);
void postprocess(vector< vector<info_node> >& node);
void save_posicio_mat(vector< vector<info_node> >& node);
void calcul_error(vector< vector<info_node> >& node);
void guardar_dades();
void Newton_Raphson(vector< vector<info_node> >& node, bool &fi,double &phi_prev,
    ↪ double &circulacio_prev);
void canvi_phi_mat(vector< vector<info_node> >& node);
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2]);
void find_position(vector< vector<info_node> >& node);
double phi_face_pos(double m_dot, double xe, double xP, double phiP, double xE,
    ↪ double phiE, double xW,double phiW, double xEE, double phiEE); // for faces
```

```
    ↪  e, n
double phi_face_neg (double m_dot, double xw, double xP, double phiP, double xW,
    ↪ double phiW, double xE,double phiE, double xWW, double phiWW); // for faces
    ↪  w, s
double UDS_pos(double m_dot, double phiP,double phiE); // for faces e, n
double UDS_neg(double m_dot, double phiP,double phiW); // for faces w, s
void def_vel_mdot(vector< vector<info_node> >& node);
void verif_coefs(vector< vector<info_node> >& node);
void save_coefs (vector< vector<info_node> >& node);
void save_outgoing_phi(vector< vector<info_node> >& node);
void compute_bp(vector< vector<info_node> >& node);
void sol_analytic (vector< vector<info_node> >& node);
void save_error_analytic(double error);
// Codi
int main()
{

 time(&inici);
 cout<<"Inici"<<endl;
 vector< vector<info_node> > node(N+2, vector<info_node>(M+2));
 preprocess(node);
 cout<<"Preprocess"<<endl;
 mapa_inicial(node);
 cout<<"Mapa inicial"<<endl;
 find_position(node);
 cout<<"Posicio de convergencia trobada"<<endl;
 calcul_coefs(node);
 verif_coefs(node);
 cout<<"Coefs"<<endl;
 solver(node);
 postprocess(node);
 if (exercise==1)
 {
  sol_analytic(node);
 }

}
void preprocess(vector< vector<info_node> >& node)
{
 geometria(node);
 condicions_contorn(node);
 def_vel_mdot(node);
 mapa_inicial(node);
}
void geometria(vector< vector<info_node> >& node)
{
 int material;
 double Ax=L/N, Ay=H/M;
 // Internal nodes
 for (int i=1;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (i==1 || i==N+1)
   {
    node[i][j].r[0]=Ax/2;
   }
   else
   {
    node[i][j].r[0]=node[i-1][j].r[0]+Ax;
   }
   if (j==1)
```

```
    {
     node[i][j].r[1]=Ay/2;
    }
    else
    {
     node[i][j].r[1]=node[i][j-1].r[1]+Ay;
    }
    node[i][j].Ax=Ax;
    node[i][j].Ay=Ay;
   }
  }
  // External nodes
  // Right and left nodes
  for  (int j=1;j<M+1;j++)
  {
   node[0][j].r[0]=0;
   node[0][j].r[1]=node[1][j].r[1];
   node[0][j].Ax=0;
   node[N+1][j].r[0]=L;
   node[N+1][j].r[1]=node[N][j].r[1];
   node[N+1][j].Ax=0;
  }
  // Top and bottom nodes
  for (int i=1;i<N+1;i++)
  {
   node[i][0].r[0]=node[i][2].r[0];
   node[i][0].r[1]=0;
   node[i][0].Ay=0;
   node[i][M+1].r[0]=node[i][M].r[0];
   node[i][M+1].r[1]=H;
   node[i][M+1].Ay=0;
  }
  // Vertices
  node[0][0].r[0]=0;
  node[0][0].r[1]=0;
  node[0][M+1].r[0]=0;
  node[0][M+1].r[1]=H;
  node[N+1][0].r[0]=L;
  node[N+1][0].r[1]=0;
  node[N+1][M+1].r[0]=L;
  node[N+1][M+1].r[1]=H;
}
void condicions_contorn(vector< vector<info_node> >& node) // boundary conditions
{
 if (exercise==1)
 {
  int i;
  for (int j=0;j<M+2;j++)
  {
   i=0;
   node[i][j].BC=0;
   phi_L=phi_in;
   i=N+1;
   node[i][j].BC=0;
   phi_R=phi_out;
  }
  int j;
  for (int i=1;i<N+1;i++)
  {
   j=0;
   node[i][j].BC=1;
   jB=0;
```

```
   j=M+1;
   node[i][j].BC=1;
   jT=0;
  }
 }
 else if (exercise==2)
 {
  for (int i=0;i<N+2;i++)
  {
   node[i][0].BC=0;
   node[i][M+1].BC=0;
  }
  phi_B=phi_low;
  phi_T=phi_high;
  for (int j=1;j<M+1;j++)
  {
   node[0][j].BC=0;
   node[N+1][j].BC=0;
  }
  phi_L=phi_high;
  phi_R=phi_low;
 }
 else if (exercise==3)
 {
  bool inner=true;
  int i=1;
  while (inner==true)
  {
   node[i][0].BC=0;
   phi_left[i]=1+ tanh(10*(2*node[i][0].r[0]-1));
   i++;
   if (node[i][0].r[0]>L/2)
   {
    inner=false;
   }
  }
  int inici=i;
  for (int i=inici;i<N+2;i++)
  {
   node[i][0].BC=1;
  }
  for (int i=1;i<N+1;i++)
  {
   node[i][M+1].BC=0;
  }
  phi_T=phi_bound;
  for (int j=1;j<M+1;j++)
  {
   node[0][j].BC=0;
   node[N+1][j].BC=0;
  }
  phi_L=phi_bound;
  phi_R=phi_bound;
 }
 else if (exercise==4)
 {
  int j;
  for (int i=0;i<N+2;i++)
  {
   j=0;
   node[i][j].BC=0;
   phi_B=phi_in;
```

```
    j=M+1;
    node[i][j].BC=0;
    phi_T=phi_out;
   }
   int i;
   for (int j=1;j<M+1;j++)
   {
    i=0;
    node[i][j].BC=1;
    jL=0;
    i=N+1;
    node[i][j].BC=1;
    jR=0;
   }
  }


}
void mapa_inicial(vector< vector<info_node> >& node)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   node[i][j].phi=288;//(phi_R-phi_L)*node[i][j].r[0]/L+phi_L;
   node[i][j].phi_ant=node[i][j].phi;
  }
 }
}
void calcul_coefs(vector< vector<info_node> >& node)
{
 double dpe,dpw, dps, dpn;
 double phie,phin,phis,phiw;
 // Most internal nodes (not considering i,j=1 i N,M)
 for (int i=2;i<N;i++)
 {
  for (int j=2;j<M;j++)
  {
   dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
   dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
   dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
   dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
   node[i][j].De=gamma*node[i][j].Ay/dpe;
   node[i][j].Dw=gamma*node[i][j].Ay/dpw;
   node[i][j].Ds=gamma*node[i][j].Ax/dps;
   node[i][j].Dn=gamma*node[i][j].Ax/dpn;
   phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r
       ↪ [0],node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],
       ↪ node[i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
   phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r
       ↪ [0],node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],
       ↪ node[i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
   phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r
       ↪ [1],node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],
       ↪ node[i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
   phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r
       ↪ [1],node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],
       ↪ node[i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
   if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
   {
    cout<<phis<<"␣␣"<<phin<<"␣␣"<<phie<<"␣␣"<<phiw<<endl;
    system("pause");
```

```
   }
   node[i][j].ae=node[i][j].De;
   node[i][j].aw=node[i][j].Dw;
   node[i][j].an=node[i][j].Dn;
   node[i][j].as=node[i][j].Ds;
   node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
       ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j
       ↪ ].mn+node[i][j].ms;
   node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
       ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
  }
 }
 // Internal nodes, the closest to the boundary conditions
 int j;
  // Top and bottom nodes
  for (int i=2;i<N;i++)
  {
   // Top
   j=M;
   dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
   dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
   dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
   dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
   node[i][j].De=gamma*node[i][j].Ay/dpe;
   node[i][j].Dw=gamma*node[i][j].Ay/dpw;
   node[i][j].Ds=gamma*node[i][j].Ax/dps;
   node[i][j].Dn=gamma*node[i][j].Ax/dpn;
   phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r
       ↪ [0],node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],
       ↪ node[i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
   phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r
       ↪ [0],node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],
       ↪ node[i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
   phin=node[i][j+1].phi;
   phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r
       ↪ [1],node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],
       ↪ node[i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
   if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
   {
    cout<<i<<"␣␣"<<j<<endl;
    system("pause");
   }
   node[i][j].ae=node[i][j].De;
   node[i][j].aw=node[i][j].Dw;
   node[i][j].an=node[i][j].Dn;
   node[i][j].as=node[i][j].Ds;
   node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
       ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j
       ↪ ].mn+node[i][j].ms;
   node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
       ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
   // Bottom
   j=1;
   dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
   dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
   dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
   dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
   node[i][j].De=gamma*node[i][j].Ay/dpe;
   node[i][j].Dw=gamma*node[i][j].Ay/dpw;
   node[i][j].Ds=gamma*node[i][j].Ax/dps;
   node[i][j].Dn=gamma*node[i][j].Ax/dpn;
```

```
  phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r
      ↪ [0],node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],
      ↪ node[i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
  phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r
      ↪ [0],node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],
      ↪ node[i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
  phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r
      ↪ [1],node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],
      ↪ node[i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
  phis=node[i][j-1].phi;
  if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
  {
   cout<<i<<"␣␣"<<j<<endl;
   system("pause");
  }
  node[i][j].ae=node[i][j].De;
  node[i][j].aw=node[i][j].Dw;
  node[i][j].an=node[i][j].Dn;
  node[i][j].as=node[i][j].Ds;
  node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
      ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j
      ↪ ].mn+node[i][j].ms;
  node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
      ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
 }
int i;
 // Right and left nodes
 for (int j=2;j<M;j++)
 {
  // Right
  i=N;
  dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
  dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
  dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
  dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
  node[i][j].De=gamma*node[i][j].Ay/dpe;
  node[i][j].Dw=gamma*node[i][j].Ay/dpw;
  node[i][j].Ds=gamma*node[i][j].Ax/dps;
  node[i][j].Dn=gamma*node[i][j].Ax/dpn;
  phie=node[i+1][j].phi;
  phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r
      ↪ [0],node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],
      ↪ node[i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
  phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r
      ↪ [1],node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],
      ↪ node[i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
  phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r
      ↪ [1],node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],
      ↪ node[i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
  if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
  {
   cout<<i<<"␣␣"<<j<<endl;
   system("pause");
  }
  node[i][j].ae=node[i][j].De;
  node[i][j].aw=node[i][j].Dw;
  node[i][j].an=node[i][j].Dn;
  node[i][j].as=node[i][j].Ds;
  node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
      ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j
      ↪ ].mn+node[i][j].ms;
```

```
 node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
     ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
 // Left
 i=1;
 dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
 dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
 dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
 dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
 node[i][j].De=gamma*node[i][j].Ay/dpe;
 node[i][j].Dw=gamma*node[i][j].Ay/dpw;
 node[i][j].Ds=gamma*node[i][j].Ax/dps;
 node[i][j].Dn=gamma*node[i][j].Ax/dpn;
 phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r
     ↪ [0],node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],
     ↪ node[i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
 phiw=node[i-1][j].phi;
 phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r
     ↪ [1],node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],
     ↪ node[i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
 phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r
     ↪ [1],node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],
     ↪ node[i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
 if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
 {
  cout<<i<<"␣␣"<<j<<endl;
  system("pause");
 }

 node[i][j].ae=node[i][j].De;
 node[i][j].aw=node[i][j].Dw;
 node[i][j].an=node[i][j].Dn;
 node[i][j].as=node[i][j].Ds;
 node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
     ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j
     ↪ ].mn+node[i][j].ms;
 node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
     ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
}
// The 4 remaining nodes
i=1;
j=1;
dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
node[i][j].De=gamma*node[i][j].Ay/dpe;
node[i][j].Dw=gamma*node[i][j].Ay/dpw;
node[i][j].Ds=gamma*node[i][j].Ax/dps;
node[i][j].Dn=gamma*node[i][j].Ax/dpn;
phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r[0],
    ↪ node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],node[
    ↪ i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
phiw=node[i-1][j].phi;
phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r[1],
    ↪ node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],node[
    ↪ i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
phis=node[i][j-1].phi;
node[i][j].ae=node[i][j].De;
node[i][j].aw=node[i][j].Dw;
node[i][j].an=node[i][j].Dn;
node[i][j].as=node[i][j].Ds;
```

```
node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
    ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j].
    ↪ mn+node[i][j].ms;
node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+node
    ↪ [i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
j=M;
dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
node[i][j].De=gamma*node[i][j].Ay/dpe;
node[i][j].Dw=gamma*node[i][j].Ay/dpw;
node[i][j].Ds=gamma*node[i][j].Ax/dps;
node[i][j].Dn=gamma*node[i][j].Ax/dpn;
phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r[0],
    ↪ node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],node[
    ↪ i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
phiw=node[i-1][j].phi;
phin=node[i][j+1].phi;
phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r[1],
    ↪ node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],node[
    ↪ i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
node[i][j].ae=node[i][j].De;
node[i][j].aw=node[i][j].Dw;
node[i][j].an=node[i][j].Dn;
node[i][j].as=node[i][j].Ds;
node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
    ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j].
    ↪ mn+node[i][j].ms;
node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+node
    ↪ [i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
i=N;
j=1;
dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
node[i][j].De=gamma*node[i][j].Ay/dpe;
node[i][j].Dw=gamma*node[i][j].Ay/dpw;
node[i][j].Ds=gamma*node[i][j].Ax/dps;
node[i][j].Dn=gamma*node[i][j].Ax/dpn;
phie=node[i+1][j].phi;
phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r[0],
    ↪ node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],node[
    ↪ i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r[1],
    ↪ node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],node[
    ↪ i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
phis=node[i][j-1].phi;
node[i][j].ae=node[i][j].De;
node[i][j].aw=node[i][j].Dw;
node[i][j].an=node[i][j].Dn;
node[i][j].as=node[i][j].Ds;
node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
    ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j].
    ↪ mn+node[i][j].ms;
node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+node
    ↪ [i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;

j=M;
dpe=node[i][j].Ax/2+node[i+1][j].Ax/2;
dpw=node[i][j].Ax/2+node[i-1][j].Ax/2;
```

```
dpn=node[i][j].Ay/2+node[i][j+1].Ay/2;
dps=node[i][j].Ay/2+node[i][j-1].Ay/2;
node[i][j].De=gamma*node[i][j].Ay/dpe;
node[i][j].Dw=gamma*node[i][j].Ay/dpw;
node[i][j].Ds=gamma*node[i][j].Ax/dps;
node[i][j].Dn=gamma*node[i][j].Ax/dpn;
phie=node[i+1][j].phi;
phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r[0],
    ↪ node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],node[
    ↪ i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
phin=node[i][j+1].phi;
phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r[1],
    ↪ node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],node[
    ↪ i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
node[i][j].ae=node[i][j].De;
node[i][j].aw=node[i][j].Dw;
node[i][j].an=node[i][j].Dn;
node[i][j].as=node[i][j].Ds;
node[i][j].ap=node[i][j].ae+node[i][j].aw+node[i][j].as+node[i][j].an-node[i][j
    ↪ ].Spp*node[i][j].Ax*node[i][j].Ay-node[i][j].me+node[i][j].mw-node[i][j].
    ↪ mn+node[i][j].ms;
node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+node
    ↪ [i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;


// Right wall nodes
if (node[N+1][1].BC==0) // Dirichlet
{
 for (int j=1;j<M+1;j++)
 {
  node[N+1][j].aw=0;
  node[N+1][j].an=0;
  node[N+1][j].as=0;
  node[N+1][j].ae=0;
  node[N+1][j].ap=1;
  node[N+1][j].bp=phi_R;
 }
}
else // Neumann
{
 for (int j=1;j<M+1;j++)
 {
  node[N+1][j].aw=1;
  node[N+1][j].an=0;
  node[N+1][j].as=0;
  node[N+1][j].ae=0;
  node[N+1][j].ap=1;
  node[N+1][j].bp=-jR*node[N][j].Ax/2/gamma;
 }
}
// Left wall nodes
if (node[0][1].BC==0) // Dirichlet
{
 for (int j=1;j<M+1;j++)
 {
  node[0][j].aw=0;
  node[0][j].an=0;
  node[0][j].as=0;
  node[0][j].ae=0;
  node[0][j].ap=1;
  node[0][j].bp=phi_L;
 }
```

```
 }
 else //Neumann
 {
  for (int j=1;j<M+1;j++)
  {
   node[0][j].aw=0;
   node[0][j].an=0;
   node[0][j].as=0;
   node[0][j].ae=1;
   node[0][j].ap=1;
   node[0][j].bp=-jL*node[1][j].Ax/2/gamma;;
  }
 }
 // Top wall nodes
 if (node[1][M+1].BC==0) // Dirichlet
 {
  for (int i=1;i<N+1;i++)
  {
   node[i][M+1].aw=0;
   node[i][M+1].an=0;
   node[i][M+1].as=0;
   node[i][M+1].ae=0;
   node[i][M+1].ap=1;
   node[i][M+1].bp=phi_T;
  }
 }
 else //Neumann
 {
  for (int i=1;i<N+1;i++)
  {
   node[i][M+1].aw=0;
   node[i][M+1].an=0;
   node[i][M+1].as=1;
   node[i][M+1].ae=0;
   node[i][M+1].ap=1;
   node[i][M+1].bp=-jT*node[i][M].Ay/2/gamma;
  }
 }

if (exercise==1 || exercise==2 || exercise==4)
{
 // Bottom wall nodes
 if (node[1][0].BC==0) // Dirichlet
 {
  for (int i=1;i<N+1;i++)
  {
   node[i][0].aw=0;
   node[i][0].an=0;
   node[i][0].as=0;
   node[i][0].ae=0;
   node[i][0].ap=1;
   node[i][0].bp=phi_B;
  }
 }
 else //Neumann
 {
  for (int i=1;i<N+1;i++)
  {
   node[i][0].aw=0;
   node[i][0].an=1;
   node[i][0].as=0;
   node[i][0].ae=0;
```

```
    node[i][0].ap=1;
    node[i][0].bp=-jB*node[i][M].Ay/2/gamma;
   }
  }
 }
 else
 {
  for (i=1;i<N+1;i++)
  {
   if(node[i][0].BC==0)
   {
    node[i][0].aw=0;
    node[i][0].an=0;
    node[i][0].as=0;
    node[i][0].ae=0;
    node[i][0].ap=1;
    node[i][0].bp=phi_left[i];
   }
   else
   {
    node[i][0].aw=0;
    node[i][0].an=1;
    node[i][0].as=0;
    node[i][0].ae=0;
    node[i][0].ap=1;
    node[i][0].bp=-jB*node[i][M].Ay/2/gamma;
   }
  }
 }
 // Corner nodes
 i=0;
 j=0;
 node[i][j].aw=0;
 node[i][j].an=1;
 node[i][j].as=0;
 node[i][j].ae=0;
 node[i][j].ap=1;
 node[i][j].bp=0;
 i=N+1;
 j=0;
 node[i][j].aw=0;
 node[i][j].an=1;
 node[i][j].as=0;
 node[i][j].ae=0;
 node[i][j].ap=1;
 node[i][j].bp=0;
 i=0;
 j=M+1;
 node[i][j].aw=0;
 node[i][j].an=0;
 node[i][j].as=1;
 node[i][j].ae=0;
 node[i][j].ap=1;
 node[i][j].bp=0;
 i=N+1;
 j=M+1;
 node[i][j].aw=0;
 node[i][j].an=0;
 node[i][j].as=1;
 node[i][j].ae=0;
 node[i][j].ap=1;
 node[i][j].bp=0;
```

```
}
void solver(vector< vector<info_node> >& node)
{
 bool trobat=false;
 while (trobat==false)
 {
  compute_bp(node);
  solver_phi(node);

  if(calcul_error_max(node)<delta)
  {
   trobat=true;
  }
  else
  {
   iteracio(node);
  }
 }
}
void solver_phi(vector< vector<info_node> >& node)
{
 if (line_by_line==false) // Gauss-Seidel
 {
  for (int i=1;i<N+1;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].aw*node[i-1][j].phi+
        ↪ node[i][j].as*node[i][j-1].phi+node[i][j].an*node[i][j+1].phi+node[i][j
        ↪ ].bp)/node[i][j].ap;
    node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
   }
  }
  for (int j=1;j<M+1;j++)
  {
   node[0][j].phi=(node[0][j].ae*node[1][j].phi+node[0][j].as*node[0][j-1].phi+
       ↪ node[0][j].an*node[0][j+1].phi+node[0][j].bp)/node[0][j].ap;
   node[0][j].phi=node[0][j].phi_ant+fr*(node[0][j].phi-node[0][j].phi_ant);
   node[N+1][j].phi=(node[N+1][j].aw*node[N][j].phi+node[N+1][j].as*node[N+1][j
       ↪ -1].phi+node[N+1][j].an*node[N+1][j+1].phi+node[N+1][j].bp)/node[N+1][j
       ↪ ].ap;
   node[N+1][j].phi=node[N+1][j].phi_ant+fr*(node[N+1][j].phi-node[N+1][j].phi_ant
       ↪ );
  }
  for (int i=1;i<N+1;i++)
  {
   node[i][0].phi=(node[i][0].ae*node[i+1][0].phi+node[i][0].aw*node[i-1][0].phi+
       ↪ node[i][0].an*node[i][1].phi+node[i][0].bp)/node[i][0].ap;
   node[i][0].phi=node[i][0].phi_ant+fr*(node[i][0].phi-node[i][0].phi_ant);
   node[i][M+1].phi=(node[i][M+1].ae*node[i+1][M+1].phi+node[i][M+1].aw*node[i-1][
       ↪ M+1].phi+node[i][M+1].as*node[i][M].phi+node[i][M+1].bp)/node[i][M+1].ap
       ↪ ;
   node[i][M+1].phi=node[i][M+1].phi_ant+fr*(node[i][M+1].phi-node[i][M+1].phi_ant
       ↪ );
  }
  int i=0,j=0;
  node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].an*node[i][j+1].phi+
      ↪ node[i][j].bp)/node[i][j].ap;
  node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
  j=M+1;
  node[i][j].phi=(node[i][j].ae*node[i+1][j].phi+node[i][j].as*node[i][j-1].phi+
      ↪ node[i][j].bp)/node[i][j].ap;
```

```
  node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
  i=N+1; j=0;
  node[i][j].phi=(node[i][j].aw*node[i-1][j].phi+node[i][j].an*node[i][j+1].phi+
      ↪ node[i][j].bp)/node[i][j].ap;
  node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
  j=M+1;
  node[i][j].phi=(node[i][j].aw*node[i-1][j].phi+node[i][j].as*node[i][j-1].phi+
      ↪ node[i][j].bp)/node[i][j].ap;
  node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
 }
 else // line_by_line
 {
  double P[N+2],Q[N+2];
  for (int j=1;j<M+1;j++)
  {
   P_Q_calcul(j,node,P,Q);
   for (int i=1;i<N+2;i++)
   {
    node[i][j].phi=P[i]*node[i-1][j].phi+Q[i];
   }
   //system("pause");
  }
  // relaxing factor
  for (int i=1;i<N+2;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    node[i][j].phi=node[i][j].phi_ant+fr*(node[i][j].phi-node[i][j].phi_ant);
   }
  }
 }
}
double calcul_error_max(vector< vector<info_node> >& node)
{
 double error=0,error_phi;
 for (int i=1;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   error_phi=v_abs(node[i][j].phi-node[i][j].phi_ant);
   if (error_phi>error)
   {
    error=error_phi;
   }
  }
 }
 cout<<delta/error*100<<"%"<<endl;
 return error;
}
double v_abs(double a)
{
 if (a>0)
 {
  return a;
 }
 else
 {
  return -a;
 }
}
void iteracio (vector< vector<info_node> >& node)
{
```

```
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   node[i][j].phi_ant=node[i][j].phi;
  }
 }
}
void save_phi (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Phi");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].phi<<"\t";
  }
  file<<endl;
 }
 file<<N+2<<endl;
 file<<M+2;
 file.close();
}
void postprocess(vector< vector<info_node> >& node)
{
 double circulacio;
 save_phi(node);
 save_posicio_mat(node);
 calcul_error(node);
 cout<<"-----------------------------"<<endl;
 save_outgoing_phi(node);
}
void save_posicio_mat(vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Posicio");
 for (int j=0;j<M+2;j++)
 {
  file<<node[1][j].r[1]<<endl;
 }
 for (int i=0;i<N+2;i++)
 {
  file<<node[i][1].r[0]<<endl;
 }
 file.close();
}
void calcul_error(vector< vector<info_node> >& node)
{
 double error=0,error2;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   error2=node[i][j].phi*node[i][j].ap-(node[i][j].ae*node[i+1][j].phi+node[i][j].
       ↪ aw*node[i-1][j].phi+node[i][j].as*node[i][j-1].phi+node[i][j].an*node[i
       ↪ ][j+1].phi+node[i][j].bp);
   if (v_abs(error2)>error)
   {
    error=v_abs(error2);
   }
  }
```

```
 }
 cout<<"Error␣comes:␣"<<error<<endl;
}

void guardar_dades()
{
 ofstream file;
 file.open("Dades_Resultats");
 file<<"DADES␣DEL␣PROBLEMA"<<endl;
 file<<"Altura:␣"<<H<<"␣m"<<endl;
 file<<"Longitud:␣"<<L<<"␣m"<<endl;
 file<<"Densitat␣de␣malla:␣"<<N<<"x"<<M<<endl;
 file<<"RESULTATS␣DEL␣PROBLEMA"<<endl;
 time(&final);
 file<<"Temps␣de␣ c lcul :␣"<<difftime(final,inici)<<"␣s"<<endl;
}
void P_Q_calcul(int row, vector< vector<info_node> >& node,double P[N+2],double Q[
    ↪ N+2])
{
 P[N+1]=node[N+1][row].aw/node[N+1][row].ap;
 Q[N+1]=(node[N+1][row].an*node[N+1][row+1].phi+node[N+1][row].as*node[N+1][row
     ↪ -1].phi+node[N+1][row].bp)/node[N+1][row].ap;
 double bp_ast;
 for (int i=N;i>0;i--)
 {
  P[i]=node[i][row].aw/(node[i][row].ap-node[i][row].ae*P[i+1]);
  bp_ast=node[i][row].an*node[i][row+1].phi+node[i][row].as*node[i][row-1].phi+
      ↪ node[i][row].bp;
  Q[i]=(bp_ast+node[i][row].ae*Q[i+1])/(node[i][row].ap-node[i][row].ae*P[i+1]);
 }
}
void find_position(vector< vector<info_node> >& node)
{
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (node[i][j].r[0]-node[i][j].Ax/2 < x_conv && node[i][j].r[0]+node[i][j].Ax/2
       ↪  >= x_conv && node[i][j].r[1]-node[i][j].Ay/2 < y_conv && node[i][j].r
       ↪ [1]+node[i][j].Ay/2 >= y_conv)
   {
    i_conv=i;
    j_conv=j;
    break;
   }
  }
 }
}
double phi_face_pos(double m_dot, double xe, double xP, double phiP, double xE,
    ↪ double phiE, double xW,double phiW, double xEE, double phiEE)
{
 double xD, phiD,xC,phiC,xU,phiU;
 if (m_dot>0)
 {
  xD=xE;
  phiD=phiE;
  xC=xP;
  phiC=phiP;
  xU=xW;
  phiU=phiW;
 }
 else
```

```
 {
  xD=xP;
  phiD=phiP;
  xC=xE;
  phiC=phiE;
  xU=xEE;
  phiU=phiEE;
 }

 double norm_phic,norm_xe, norm_phie, norm_xc;
 if (phiD==phiU)
 {
  return phiD;
 }
 else
 {
  norm_phic=(phiC-phiU)/(phiD-phiU); // vigilar que phiD no debe ser igual a phiU
  norm_xe=(xe-xU)/(xD-xU);
  norm_xc=(xC-xU)/(xD-xU);
  if (SCHEME==0) //UDS (ot FUDS)
  {
   norm_phie=norm_phic;
  }
  else if (SCHEME==1) //CDS
  {
   norm_phie=(norm_xe-norm_xc)/(1-norm_xc)+(norm_xe-1)/(norm_xc-1)*norm_phic;
  }
  else if (SCHEME==2) //SUDS
  {
   norm_phie=norm_xe/norm_xc*norm_phic;
  }
  else if (SCHEME==3) //QUICK
  {
   norm_phie=norm_xe+(norm_xe*(norm_xe-1))/(norm_xc*(norm_xc-1))*(norm_phic-
       ↪ norm_xc);
  }
  else if (SCHEME==4) //SMART
  {
   if (norm_phic>0 && norm_phic<norm_xc/3)
   {
    norm_phie=-(norm_xe*(1-3*norm_xc+2*norm_xe))/(norm_xc*(norm_xc-1))*norm_phic;
   }
   else if(norm_phic>norm_xc/6 && norm_phic<norm_xc/norm_xe*(1+norm_xe-norm_xc))
   {
    norm_phie=norm_xe*(norm_xe-norm_xc)/(1-norm_xc)+norm_xe*(norm_xe-1)/(norm_xc*(
        ↪ norm_xc-1))*norm_phic;
   }
   else if (norm_phic>norm_xc/norm_xe*(1+norm_xe-norm_xc) && norm_phic<1)
   {
    norm_phie=1;
   }
   else
   {
    norm_phie=norm_phic;
   }
  }
  return phiU+(phiD-phiU)*norm_phie;
 }

}
double phi_face_neg (double m_dot, double xw, double xP, double phiP, double xW,
    ↪ double phiW, double xE,double phiE, double xWW, double phiWW)
```

```
{
 double xD, phiD,xC,phiC,xU,phiU;
 if (m_dot>0)
 {
  xD=xP;
  phiD=phiP;
  xC=xW;
  phiC=phiW;
  xU=xWW;
  phiU=phiWW;
 }
 else
 {
  xD=xW;
  phiD=phiW;
  xC=xP;
  phiC=phiP;
  xU=xE;
  phiU=phiE;
 }
 double norm_phic,norm_xe, norm_phie, norm_xc;
 if (phiD==phiU)
 {
  return phiD;
 }
 norm_phic=(phiC-phiU)/(phiD-phiU);
 norm_xe=(xw-xU)/(xD-xU);
 norm_xc=(xC-xU)/(xD-xU);
 if (SCHEME==0) //UDS (ot FUDS)
 {
  norm_phie=norm_phic;
 }
 else if (SCHEME==1) //CDS
 {
  norm_phie=(norm_xe-norm_xc)/(1-norm_xc)+(norm_xe-1)/(norm_xc-1)*norm_phic;
 }
 else if (SCHEME==2) //SUDS
 {
  norm_phie=norm_xe/norm_xc*norm_phic;
 }
 else if (SCHEME==3) //QUICK
 {
  norm_phie=norm_xe+(norm_xe*(norm_xe-1))/(norm_xc*(norm_xc-1))*(norm_phic-norm_xc
      ↪ );
 }
 else if (SCHEME==4) //SMART
 {
  if (norm_phic>0 && norm_phic<norm_xc/3)
  {
   norm_phie=-(norm_xe*(1-3*norm_xc+2*norm_xe))/(norm_xc*(norm_xc-1))*norm_phic;
  }
  else if(norm_phic>norm_xc/6 && norm_phic<norm_xc/norm_xe*(1+norm_xe-norm_xc))
  {
   norm_phie=norm_xe*(norm_xe-norm_xc)/(1-norm_xc)+norm_xe*(norm_xe-1)/(norm_xc*(
       ↪ norm_xc-1))*norm_phic;
  }
  else if (norm_phic>norm_xc/norm_xe*(1+norm_xe-norm_xc) && norm_phic<1)
  {
   norm_phie=1;
  }
  else
  {
```

```
    norm_phie=norm_phic;
   }
  }
  if (norm_phic==0)
  {
   norm_phie=0;
  }
  else if(norm_phic==1)
  {
   norm_phie=1;
  }


  return phiU+(phiD-phiU)*norm_phie;

}
double UDS_pos(double m_dot, double phiP,double phiE)
{
 if (m_dot>0)
 {
  return phiP;
 }
 else
 {
  return phiE;
 }
}
double UDS_neg(double m_dot, double phiP,double phiW)
{
 if (m_dot>0)
 {
  return phiW;
 }
 else
 {
  return phiP;
 }
}
void def_vel_mdot(vector< vector<info_node> >& node)
{
 double alpha,x,y,xb,yb,v;
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (exercise==1)
   {
    node[i][j].vel[0]=v0;
    node[i][j].vel[1]=0;
    node[i][j].me=v0*rho_in*node[i][j].Ay;
    node[i][j].mw=v0*rho_in*node[i][j].Ay;
    node[i][j].ms=0;
    node[i][j].mn=0;
    node[i][j].Spp=0;
    node[i][j].Spc=0;
   }
   else if (exercise==2)
   {
    alpha=pi/4;
    node[i][j].vel[0]=v0*cos(alpha);
    node[i][j].vel[1]=v0*sin(alpha);
    node[i][j].me=node[i][j].vel[0]*rho_in*node[i][j].Ay;
```

```
    node[i][j].mw=node[i][j].vel[0]*rho_in*node[i][j].Ay;
    node[i][j].ms=node[i][j].vel[1]*rho_in*node[i][j].Ax;
    node[i][j].mn=node[i][j].vel[1]*rho_in*node[i][j].Ax;
    node[i][j].Spp=0;
    node[i][j].Spc=0;
   }
   else if (exercise==3)
   {
    x=node[i][j].r[0]-1;
    y=node[i][j].r[1];
    node[i][j].vel[0]=2*y*(1-x*x);
    node[i][j].vel[1]=-2*x*(1-y*y);
    node[i][j].Spp=0;
    node[i][j].Spc=0;
    if (i<N+1 && i>0 && j<M+1 && j>0)
    {
     xb=node[i+1][j].r[0]-1;
     yb=node[i+1][j].r[1];
     xb=(x+xb)/2;
     yb=(y+yb)/2;
     v=2*yb*(1-xb*xb);
     node[i][j].me=v*rho_in*node[i][j].Ay;
     xb=node[i-1][j].r[0]-1;
     yb=node[i-1][j].r[1];
     xb=(x+xb)/2;
     yb=(y+yb)/2;
     v=2*yb*(1-xb*xb);
     node[i][j].mw=v*rho_in*node[i][j].Ay;
     xb=node[i][j-1].r[0]-1;
     yb=node[i][j-1].r[1];
     xb=(x+xb)/2;
     yb=(y+yb)/2;
     v=-2*xb*(1-yb*yb);
     node[i][j].ms=v*rho_in*node[i][j].Ax;
     xb=node[i][j+1].r[0]-1;
     yb=node[i][j+1].r[1];
     xb=(x+xb)/2;
     yb=(y+yb)/2;
     v=-2*xb*(1-yb*yb);
     node[i][j].mn=v*rho_in*node[i][j].Ax;
    }
    else
    {
     node[i][j].me=0;
     node[i][j].mw=0;
     node[i][j].ms=0;
     node[i][j].mn=0;
    }
   }
   else if (exercise==4)
   {
    node[i][j].vel[0]=0;
    node[i][j].vel[1]=v0;
    node[i][j].me=0;
    node[i][j].mw=0;
    node[i][j].ms=v0*rho_in*node[i][j].Ax;
    node[i][j].mn=v0*rho_in*node[i][j].Ax;
    node[i][j].Spp=0;
    node[i][j].Spc=0;
    }
  }
 }
```

```
}
void verif_coefs(vector< vector<info_node> >& node)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (isnan(node[i][j].bp))
   {
    cout<<i<<"␣␣"<<j<<endl;
    system("pause");
   }
  }
 }
}
void save_coefs (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Coefs");
 file<<"AP"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].ap<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AE"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].ae<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AW"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].aw<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AS"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].as<<"\t";
  }
  file<<endl;
 }
```

```
 file<<endl;
 file<<"AN"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].an<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"BP"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<node[i][j].bp<<"\t";
  }
  file<<endl;
 }
 file.close();
 system("pause");
}
void save_outgoing_phi(vector< vector<info_node> >& node)
{
 std::ofstream ofs;
 ofs.open("Outlet_phi_values.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("Outlet_phi_values.txt",std::ios_base::app);
 int i_slected;
 double x_selected=1,Ax=0.1;
 int j=0;
 file<<"In␣this␣document␣it␣will␣be␣represented␣the␣phi␣value␣for␣the␣outlet␣
     ↪ section␣"<<endl;
 file<<"
     ↪ ********************************************************************************
     ↪ "<<endl;
 file<<"Case␣of␣rho/gamma=␣"<<rho_in/gamma<<endl;
 file<<"x-position"<<"\t"<<"Phi"<<endl;
 file<<"--------------------"<<endl;
 while (x_selected<2)
 {
  for (int i=N/2-2;i<N+2;i++)
  {
   if (node[i][j+1].r[0]-node[i][j+1].Ax/2 < x_selected && node[i][j+1].r[0]+node[
       ↪ i][j+1].Ax/2 >= x_selected)
   {
    file<<x_selected-1<<"\t"<<"\t"<<node[i][j].phi<<endl;
    break;
   }
  }
  x_selected=x_selected+Ax;
 }
 file<<1<<"\t"<<"\t"<<node[N+1][j].phi<<endl;

 for (int i=1;i<5;i++)
 {
  file<<endl;
 }
```

```
 file<<"N␣=␣"<<N<<endl;
 file<<"M␣=␣"<<M<<endl;
 file.close();
}
void compute_bp(vector< vector<info_node> >& node)
{
 double dpe,dpw, dps, dpn;
 double phie,phin,phis,phiw;
 // Internal nodes (snot considering i,j=1 i N,M)
 for (int i=2;i<N;i++)
 {
  for (int j=2;j<M;j++)
  {
   phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r
       ↪ [0],node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],
       ↪ node[i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
   phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r
       ↪ [0],node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],
       ↪ node[i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
   phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r
       ↪ [1],node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],
       ↪ node[i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
   phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r
       ↪ [1],node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],
       ↪ node[i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
   if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
   {
    cout<<phis<<"␣␣"<<phin<<"␣␣"<<phie<<"␣␣"<<phiw<<endl;
    system("pause");
   }

   node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
       ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
  }
 }
 // internal nodes closest to the boundary nodes
 int j;
  // Top and bottom nodes
  for (int i=2;i<N;i++)
  {
   // Top
   j=M;
   phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r
       ↪ [0],node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],
       ↪ node[i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
   phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r
       ↪ [0],node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],
       ↪ node[i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
   phin=node[i][j+1].phi;
   phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r
       ↪ [1],node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],
       ↪ node[i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
   if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
   {
    cout<<i<<"␣␣"<<j<<endl;
    system("pause");
   }
   node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
       ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
   // Bottom
   j=1;
```

```
  phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r
      ↪ [0],node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],
      ↪ node[i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
  phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r
      ↪ [0],node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],
      ↪ node[i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
  phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r
      ↪ [1],node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],
      ↪ node[i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
  phis=node[i][j-1].phi;
  if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
  {
   cout<<i<<"␣␣"<<j<<endl;
   system("pause");
  }
  node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
      ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
 }
int i;
 // Right and left nodes
 for (int j=2;j<M;j++)
 {
  // Right
  i=N;
  phie=node[i+1][j].phi;
  phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r
      ↪ [0],node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],
      ↪ node[i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
  phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r
      ↪ [1],node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],
      ↪ node[i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
  phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r
      ↪ [1],node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],
      ↪ node[i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
  if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
  {
   cout<<i<<"␣␣"<<j<<endl;
   system("pause");
  }
  node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
      ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
  // Left
  i=1;
  phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r
      ↪ [0],node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],
      ↪ node[i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
  phiw=node[i-1][j].phi;
  phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r
      ↪ [1],node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],
      ↪ node[i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
  phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r
      ↪ [1],node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],
      ↪ node[i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
  if (isnan(phis) || isnan(phin) || isnan(phie) || isnan(phiw))
  {
   cout<<i<<"␣␣"<<j<<endl;
   system("pause");
  }

  node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+
      ↪ node[i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
 }
```

```
  // Last 4 remaining nodes
  i=1;
  j=1;
  phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r[0],
      ↪ node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],node[
      ↪ i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
  phiw=node[i-1][j].phi;
  phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r[1],
      ↪ node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],node[
      ↪ i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
  phis=node[i][j-1].phi;
  node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+node
      ↪ [i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
  j=M;
  phie=phi_face_pos(node[i][j].me,node[i][j].r[0]+node[i][j].Ax/2,node[i][j].r[0],
      ↪ node[i][j].phi,node[i+1][j].r[0],node[i+1][j].phi,node[i-1][j].r[0],node[
      ↪ i-1][j].phi,node[i+2][j].r[0],node[i+2][j].phi);
  phiw=node[i-1][j].phi;
  phin=node[i][j+1].phi;
  phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r[1],
      ↪ node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],node[
      ↪ i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
  node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+node
      ↪ [i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;
  i=N;
  j=1;
  phie=node[i+1][j].phi;
  phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r[0],
      ↪ node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],node[
      ↪ i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
  phin=phi_face_pos(node[i][j].mn,node[i][j].r[1]+node[i][j].Ay/2,node[i][j].r[1],
      ↪ node[i][j].phi,node[i][j+1].r[1],node[i][j+1].phi,node[i][j-1].r[1],node[
      ↪ i][j-1].phi,node[i][j+2].r[1],node[i][j+2].phi);
  phis=node[i][j-1].phi;
  node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+node
      ↪ [i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;

  j=M;
  phie=node[i+1][j].phi;
  phiw=phi_face_neg(node[i][j].mw,node[i][j].r[0]-node[i][j].Ax/2,node[i][j].r[0],
      ↪ node[i][j].phi,node[i-1][j].r[0],node[i-1][j].phi,node[i+1][j].r[0],node[
      ↪ i+1][j].phi,node[i-2][j].r[0],node[i-2][j].phi);
  phin=node[i][j+1].phi;
  phis=phi_face_neg(node[i][j].ms,node[i][j].r[1]-node[i][j].Ay/2,node[i][j].r[1],
      ↪ node[i][j].phi,node[i][j-1].r[1],node[i][j-1].phi,node[i][j+1].r[1],node[
      ↪ i][j+1].phi,node[i][j-2].r[1],node[i][j-2].phi);
  node[i][j].bp=node[i][j].Spc*node[i][j].Ax*node[i][j].Ay-node[i][j].me*phie+node
      ↪ [i][j].mw*phiw-node[i][j].mn*phin+node[i][j].ms*phis;

}
void sol_analytic (vector< vector<info_node> >& node)
{
 ofstream file;
 file.open("Phi_analytic");
 double phi_a,error=0.000000000;
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   phi_a=phi_in+(phi_out-phi_in)*(exp(node[i][j].r[0]*Pe/L)-1)/(exp(Pe)-1);
   file<<phi_a<<"\t";
   if (v_abs(node[i][j].phi-phi_a)>error)
```

```
    {
     error=v_abs(node[i][j].phi-phi_a);
    }
   }
   file<<endl;
  }
  file<<N+2<<endl;
  file<<M+2;
  file.close();
  save_error_analytic(error);
  cout<<"Error␣analitic:␣"<<error;
}
void save_error_analytic(double error)
{
  std::ofstream ofs;
  ofstream file;
  file.open("Error_analytic.txt",std::ios_base::app);
  if (SCHEME==0)
  {
   file<<"UDS"<<"\t";
  }
  else if (SCHEME==1)
  {
   file<<"CDS"<<"\t";
  }
  else if (SCHEME==2)
  {
   file<<"SUDS"<<"\t";
  }
  else if (SCHEME==3)
  {
   file<<"QUICK"<<"\t";
  }
  else if (SCHEME==4)
  {
   file<<"SMART"<<"\t";
  }
  file<<Pe<<"\t"<<N+2<<"\t";
  file<<error/v_abs(phi_out-phi_in)<<endl;

}
```

Appendix F

Developed code for incompressible Navier-Stokes (general code without the energy equation)

```
// Navier-Stokes solver for incompressible form
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <time.h>
using namespace std;
// Constant numbers
const double C1=0.35*0.5;
const double C2=0.2*0.5;
int n_it=0;
const double t_show=0.5;
//************************************************************
// Problem geometry
const double H=1;
const double L=1;
// Initial conditions
const double P0=0;
const double v0=0;
// Boundary conditions for veocity
const double u_T=1,v_T=0;
const double u_B=0,v_B=0;
const double u_R=0,v_R=0;
const double u_L=0,v_L=0;
// Numerical parameters
const int N=60;
const int M=N;
const int VC=(N+2)*(M+2);
const double fr=0.8;
const double delta=1e-9;
const double delta_v=1e-6;
const double delta_P=1e-6;
const double k=1.5;
double At;
```

```
double t_lim=700;
// Non-dimensional numbers
const double Re=1e3;
// Type of convective scheme
 /*
  0-UDS
  1-CDS
  2-SUDS
  3-QUICK
  4-SMART
 */
const int SCHEME=0;
// Solver type
bool line_by_line=false; // If Gauss-Seidel-->true; if line-by-line--> false
//************************************************************
// Time variables for the computatinal cost
time_t inici,final;
// Time variable for the physical phnomenon
double t=0;
// Main mesh variable
struct main_mesh{
 double r[2];
 double Ax;
 double Ay;
 double P_sup;
 double P;
 double P_ant;
 double ap;
 double as;
 double an;
 double aw;
 double ae;
 double bp;
 int mat; // 0  for solid, 1 for fluid
};
// Stagg-x mesh
struct stg_x_mesh{
 double r[2];
 double xw;
 double xe;
 double yn;
 double ys;
 double Ax;
 double Ay;
 double u;
 double u_ant; // n
 double u_P;
 double R;
 double R2;
};
// Stagg-y mesh
struct stg_y_mesh{
 double r[2];
 double xw;
 double xe;
 double yn;
 double ys;
 double Ax;
 double Ay;
 double v;
 double v_ant; // n
 double v_P;
```

```
 double R;
 double R2;
};
// Connectivities matrix (declares which nodes have solid neighbourds and which
    ↪ not) - based on the main mesh
 // The first element ([0][0] gives the information of how much nodes are treated)
int Ts[2*N][2]; // south nodes are solid
int Tn[2*N][2]; // north nodes are solid
int Te[2*M][2]; // east nodes are solid
int Tw[2*M][2]; // west nodes are solid
int T[VC+1][2]; // all neighbourd nodes are fluid
int Tw_u[2*(N+50)][2]; // nodes are solid
int T_u[VC+1][2]; // nodes are flow
int Tw_v[2*(M+50)][2]; // west nodes are solid
int T_v[VC+1][2]; // all neighbourd nodes are solid
//
    ↪ ********************************************************************************************************
    ↪
// Function declaration
void preprocess(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void geometry(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >& stgx
    ↪ ,vector< vector<stg_y_mesh> >& stgy);
void connectivity (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪  stgx,vector< vector<stg_y_mesh> >& stgy);
void initial_map (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void coeffs_a(vector< vector<main_mesh> >& main);
void coeffs_bp(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void time_solver (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
double time_step_choice (vector< vector<main_mesh> >& main,vector< vector<
    ↪ stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy);
void time_step (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
double find_Ax_min(vector< vector<main_mesh> >& main);
double find_v_max(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪  stgy);
void compute_R (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void compute_Ru_ant (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh>
    ↪  >& stgy);
void compute_Rv_ant (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh>
    ↪  >& stgy);
double phi_face_pos(double m_dot, double xe, double xP, double phiP, double xE,
    ↪ double phiE, double xW,double phiW, double xEE, double phiEE);
double phi_face_neg (double m_dot, double xw, double xP, double phiP, double xW,
    ↪ double phiW, double xE,double phiE, double xWW, double phiWW);
double UDS (double v_inf, double v_sup, double F);
void compute_velp(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪  stgy);
void instant_solver(vector< vector<main_mesh> >& main);
void iterative_solver(vector< vector<main_mesh> >& main);
double error_max (vector< vector<main_mesh> >& main);
void iteration_step(vector< vector<main_mesh> >& main);
void compute_vel_instant (vector< vector<main_mesh> >& main,vector< vector<
    ↪ stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy);
void save_P(vector< vector<main_mesh> >& main);
void save_vel(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void save_pos(vector< vector<main_mesh> >& main);
```

```
void save_lengths();
void comprovation(vector< vector<main_mesh> >& main);
void save_vel_gnu (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪  stgx,vector< vector<stg_y_mesh> >& stgy);
bool stationary (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void verif_NS1 (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void error_P(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >& stgx,
    ↪ vector< vector<stg_y_mesh> >& stgy);
void save_Bench(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void save_u_v(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void save_R (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy
    ↪ );
void save_R2 (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void save_parameters ();

int main()
{
 save_lengths();
 time(&inici);
 cout<<"Start..."<<endl;
 vector< vector<main_mesh> > main(N+2, vector<main_mesh>(M+2));
 vector< vector<stg_x_mesh> > stgx(N+1, vector<stg_x_mesh>(M+2));
 vector< vector<stg_y_mesh> > stgy(N+2, vector<stg_y_mesh>(M+1));
 cout<<"Preprocess..."<<endl;
 preprocess(main,stgx,stgy);
 cout<<"Discretization␣coeffieciets␣(ap,ae,aw,an,as)..."<<endl;
 coeffs_a(main);
 cout<<"Temporal␣solver..."<<endl;
 time_solver(main,stgx,stgy);
 save_vel(stgx,stgy);
 save_P(main);
 save_vel_gnu(main,stgx,stgy);
 save_Bench(stgx,stgy);
 save_u_v(stgx,stgy);
 save_parameters();
}
void preprocess(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 geometry(main, stgx,stgy);
 connectivity(main,stgx,stgy);
 initial_map(main,stgx,stgy);
 save_pos(main);
}
void geometry(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >& stgx
    ↪ ,vector< vector<stg_y_mesh> >& stgy)
{
 double x_lim[N+1], y_lim[M+1];
 x_lim[0]=0;
 y_lim[0]=0;
 x_lim[N]=L;
 y_lim[M]=H;
 double si;
 double Ax=L/N;
 double Ay=H/M;
 double x1,x2,d;
 int N2,N_ant;
```

```
// Concentrated mesh
 // x-direction
if (N%2!=0) // even number of horizontal nodes
{
 x_lim[(N-1)/2]=L/2-Ax/2;
 x_lim[(N+1)/2]=L/2+Ax/2;
 x1=x_lim[0];
 x2=x_lim[(N-1)/2];
 N2=(N-1)/2;
 for (int i=1;i<N2;i++)
 {
  d=1.0*i/N2-1.0;
  si=1+tanh(k*(d))/tanh(k);
  x_lim[i]=x1+si*(x2-x1);
 }
 N_ant=N2+1;
 N2=N-(N+1)/2;
 x1=x_lim[N];
 x2=x_lim[(N+1)/2];
 for (int i=1;i<N2;i++)
 {
  d=1.0*(N2-i)/N2-1.0;
  si=1+tanh(k*(d))/tanh(k);
  x_lim[i+N_ant]=x1+si*(x2-x1);
 }
}
else // pair number of horizontal nodes
{
 x_lim[(N)/2]=L/2;
 x1=x_lim[0];
 x2=x_lim[N/2];
 N2=N/2;
 for (int i=1;i<N2;i++)
 {
  d=1.0*i/N2-1.0;
  si=1+tanh(k*(d))/tanh(k);
  x_lim[i]=x1+si*(x2-x1);
 }
 N_ant=N2;
 x1=x_lim[N];
 x2=x_lim[N/2];
 for (int i=1;i<N2;i++)
 {
  d=1.0*(N2-i)/N2-1.0;
  si=1+tanh(k*(d))/tanh(k);
  x_lim[i+N_ant]=x1+si*(x2-x1);
 }
}
// y-direction
double y1,y2;
int M2,M_ant;
if (M%2!=0) // even number of horizontal nodes
{
 y_lim[(M-1)/2]=H/2-Ay/2;
 y_lim[(M+1)/2]=H/2+Ay/2;
 y1=y_lim[0];
 y2=y_lim[(M-1)/2];
 M2=(M-1)/2;
 for (int i=1;i<M2;i++)
 {
  d=1.0*i/M2-1.0;
  si=1+tanh(k*(d))/tanh(k);
```

```
   y_lim[i]=y1+si*(y2-y1);
  }
  M_ant=M2+1;
  M2=M-(M+1)/2;
  y1=y_lim[M];
  y2=y_lim[(M+1)/2];
  for (int i=1;i<M2;i++)
  {
   d=1.0*(M2-i)/M2-1.0;
   si=1+tanh(k*(d))/tanh(k);
   y_lim[i+M_ant]=y1+si*(y2-y1);
  }
 }
 else // pair number of horizontal nodes
 {
  y_lim[(M)/2]=H/2;
  y1=y_lim[0];
  y2=y_lim[M/2];
  M2=M/2;
  for (int i=1;i<M2;i++)
  {
   d=1.0*i/M2-1.0;
   si=1+tanh(k*(d))/tanh(k);
   y_lim[i]=y1+si*(y2-y1);
  }
  M_ant=M2;
  y1=y_lim[M];
  y2=y_lim[M/2];
  for (int i=1;i<M2;i++)
  {
   d=1.0*(M2-i)/M2-1.0;
   si=1+tanh(k*(d))/tanh(k);
   y_lim[i+M_ant]=y1+si*(y2-y1);
  }
 }
 // Main mesh
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   main[i][j].r[0]=(x_lim[i-1]+x_lim[i])/2;
   main[i][j].r[1]=(y_lim[j-1]+y_lim[j])/2;
   main[i][j].Ax=x_lim[i]-x_lim[i-1];
   main[i][j].Ay=y_lim[j]-y_lim[j-1];

  }
 }
 for  (int j=1;j<M+1;j++)
 {
  main[0][j].r[0]=0;
  main[0][j].r[1]=main[1][j].r[1];
  main[N+1][j].r[0]=L;
  main[N+1][j].r[1]=main[N][j].r[1];

 }
 for (int i=1;i<N+1;i++)
 {
  main[i][0].r[0]=main[i][2].r[0];
  main[i][0].r[1]=0;
  main[i][M+1].r[0]=main[i][M].r[0];
  main[i][M+1].r[1]=H;
 }
```

```
main[0][0].r[0]=0;
main[0][0].r[1]=0;
main[0][M+1].r[0]=0;
main[0][M+1].r[1]=H;
main[N+1][0].r[0]=L;
main[N+1][0].r[1]=0;
main[N+1][M+1].r[0]=L;
main[N+1][M+1].r[1]=H;
// Stagg-x mesh
for (int i=0;i<N+1;i++)
{
 for (int j=0;j<M+2;j++)
 {
  stgx[i][j].r[0]=x_lim[i];
  stgx[i][j].r[1]=main[i][j].r[1];
  //stgx[i][j].r[1]=(y_lim[j-1]+y_lim[j])/2;


 }
 /*
 stgx[i][0].r[0]=x_lim[i];
 stgx[i][0].r[1]=0;
 stgx[i][M+1].r[0]=x_lim[i];
 stgx[i][M+1].r[1]=H;
 */
}
// Stagg-y mesh
for (int j=0;j<M+1;j++)
{
 for (int i=0;i<N+2;i++)
 {
  stgy[i][j].r[0]=main[i][j].r[0];
  //stgy[i][j].r[0]=(x_lim[i-1]+x_lim[i])/2;
  stgy[i][j].r[1]=y_lim[j];
 }
 /*
 stgy[0][j].r[0]=0;
 stgy[0][j].r[1]=y_lim[j];
 stgy[N+1][j].r[0]=L;
 stgy[N+1][j].r[1]=y_lim[j];
 */
}

double xw,xe,yn,ys;
// Ax and Ay for Stagg-x mesh
for (int i=1; i<N;i++)
{
 for (int j=1; j<M+1;j++)
 {
  xw=main[i][j].r[0];
  xe=main[i+1][j].r[0];
  yn=y_lim[j];
  ys=y_lim[j-1];
  stgx[i][j].xe=xe;
  stgx[i][j].xw=xw;
  stgx[i][j].yn=yn;
  stgx[i][j].ys=ys;
  stgx[i][j].Ax=xe-xw;
  stgx[i][j].Ay=yn-ys;
 }
}
// Ax and Ay for Stagg-y mesh
```

```
 for (int i=1; i<N+1;i++)
 {
  for (int j=1; j<M;j++)
  {
   yn=main[i][j+1].r[1];
   ys=main[i][j].r[1];
   xw=x_lim[i-1];
   xe=x_lim[i];
   stgy[i][j].xe=xe;
   stgy[i][j].xw=xw;
   stgy[i][j].yn=yn;
   stgy[i][j].ys=ys;
   stgy[i][j].Ax=xe-xw;
   stgy[i][j].Ay=yn-ys;

  }
 }

 // Definition of the node's material
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   main[i][j].mat=1;
  }
 }

}
void connectivity (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪  stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // Main mesh
 int cont=0;
 T[0][0]=cont;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   cont++;
   T[0][0]=cont;
   T[cont][0]=i;
   T[cont][1]=j;
  }
 }
 T[0][0]=cont+1;
 Ts[0][0]=0;
 Tn[0][0]=0;
 Tw[0][0]=0;
 Te[0][0]=0;
 // Stagg-x
 cont=0;
 T_u[0][0]=cont;
 for (int i=1;i<N;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   cont++;
   T_u[0][0]=cont;
   T_u[cont][0]=i;
   T_u[cont][1]=j;
  }
 }
```

```
 T_u[0][0]=cont+1;
 Tw_u[0][0]=0;
 // Stagg-y
 cont=0;
 T_v[0][0]=cont;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M;j++)
  {
   cont++;
   T_v[0][0]=cont;
   T_v[cont][0]=i;
   T_v[cont][1]=j;
  }
 }
 T_v[0][0]=cont+1;
 Tw_v[0][0]=0;
}
void initial_map (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // Main mesh
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   main[i][j].P=P0;
   main[i][j].P_ant=P0;
   main[i][j].P_sup=P0;
  }
 }
 // Stagg-x mesh
 for (int i=0;i<N+1;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   stgx[i][j].u=v0;
   stgx[i][j].u_ant=v0;
   if (j==M+1)
   {
    stgx[i][j].u=1;
    stgx[i][j].u_ant=0;
   }
  }
 }
 // Stagg-y mesh
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+1;j++)
  {
   stgy[i][j].v=v0;
   stgy[i][j].v_ant=v0;
  }
 }
 compute_R(stgx,stgy);
}
void coeffs_a(vector< vector<main_mesh> >& main)
{
 // Internal nodes
 double dPE,dPW,dPN,dPS;
 int cont,i,j;
 cont = T[0][0];
```

```
for (int k=1;k<cont;k++)
{

 i=T[k][0];
 j=T[k][1];
 dPE=main[i+1][j].r[0]-main[i][j].r[0];
 dPW=main[i][j].r[0]-main[i-1][j].r[0];
 dPN=main[i][j+1].r[1]-main[i][j].r[1];
 dPS=main[i][j].r[1]-main[i][j-1].r[1];
 main[i][j].ae=main[i][j].Ay/dPE;
 main[i][j].aw=main[i][j].Ay/dPW;
 main[i][j].an=main[i][j].Ax/dPN;
 main[i][j].as=main[i][j].Ax/dPS;
 main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
}
cont=Te[0][0];
for (int k=1;k<cont;k++)
{
 i=Te[k][0];
 j=Te[k][1];
 dPE=main[i+1][j].r[0]-main[i][j].r[0];
 dPW=main[i][j].r[0]-main[i-1][j].r[0];
 dPN=main[i][j+1].r[1]-main[i][j].r[1];
 dPS=main[i][j].r[1]-main[i][j-1].r[1];
 main[i][j].ae=0;
 main[i][j].aw=main[i][j].Ay/dPW;
 main[i][j].an=main[i][j].Ax/dPN;
 main[i][j].as=main[i][j].Ax/dPS;
 main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
}
cont = Tw[0][0];
for (int k=1;k<cont;k++)
{
 i=Tw[k][0];
 j=Tw[k][1];
 dPE=main[i+1][j].r[0]-main[i][j].r[0];
 dPW=main[i][j].r[0]-main[i-1][j].r[0];
 dPN=main[i][j+1].r[1]-main[i][j].r[1];
 dPS=main[i][j].r[1]-main[i][j-1].r[1];
 main[i][j].ae=main[i][j].Ay/dPE;
 main[i][j].aw=0;
 main[i][j].an=main[i][j].Ax/dPN;
 main[i][j].as=main[i][j].Ax/dPS;
 main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
}
cont = Ts[0][0];
for (int k=1;k<cont;k++)
{
 i=Ts[k][0];
 j=Ts[k][1];
 dPE=main[i+1][j].r[0]-main[i][j].r[0];
 dPW=main[i][j].r[0]-main[i-1][j].r[0];
 dPN=main[i][j+1].r[1]-main[i][j].r[1];
 dPS=main[i][j].r[1]-main[i][j-1].r[1];
 main[i][j].ae=main[i][j].Ay/dPE;
 main[i][j].aw=main[i][j].Ay/dPW;
 main[i][j].an=main[i][j].Ax/dPN;
 main[i][j].as=0;
 main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
}
cont = Tn[0][0];
for (int k=1;k<cont;k++)
```

```
 {
  i=Tn[k][0];
  j=Tn[k][1];
  dPE=main[i+1][j].r[0]-main[i][j].r[0];
  dPW=main[i][j].r[0]-main[i-1][j].r[0];
  dPN=main[i][j+1].r[1]-main[i][j].r[1];
  dPS=main[i][j].r[1]-main[i][j-1].r[1];
  main[i][j].ae=main[i][j].Ay/dPE;
  main[i][j].aw=main[i][j].Ay/dPW;
  main[i][j].an=0;
  main[i][j].as=main[i][j].Ax/dPS;
  main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
 }
 // Boundary nodes
 // Left and right nodes
 for (int j=0;j<M+2;j++)
 {
  // Left nodes
  main[0][j].ae=1;
  main[0][j].aw=0;
  main[0][j].an=0;
  main[0][j].as=0;
  main[0][j].ap=1;
  // Right nodes
  main[N+1][j].ae=0;
  main[N+1][j].aw=1;
  main[N+1][j].an=0;
  main[N+1][j].as=0;
  main[N+1][j].ap=1;
 }
 // Top and bottom nodes
 for (int i=1;i<N+1;i++)
 {
  // Bottom nodes
  main[i][0].ae=0;
  main[i][0].aw=0;
  main[i][0].an=1;
  main[i][0].as=0;
  main[i][0].ap=1;
  // Top nodes
  main[i][M+1].ae=0;
  main[i][M+1].aw=0;
  main[i][M+1].an=0;
  main[i][M+1].as=1;
  main[i][M+1].ap=1;
 }
}
void coeffs_bp(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // Internal nodes
 double ue,uw,vn,vs,Ax,Ay;
 // Internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   ue=stgx[i][j].u_P;
   uw=stgx[i-1][j].u_P;
   vn=stgy[i][j].v_P;
   vs=stgy[i][j-1].v_P;
   Ax=main[i][j].Ax;
```

```
   Ay=main[i][j].Ay;
   main[i][j].bp=-1/At*((ue-uw)*Ay+(vn-vs)*Ax);
  }
 }
 // Boundary nodes
 // Left and right nodes
 for (int j=0;j<M+2;j++)
 {
  // Left nodes
  main[0][j].bp=0;
  // Right nodes
  main[N+1][j].bp=0;
 }

 // Top and bottom nodes
 for (int i=1;i<N+1;i++)
 {
  // Bottom nodes
  main[i][0].bp=0;
  // Top nodes
  main[i][M+1].bp=0;

 }
}
void time_solver (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 bool end_t=false;
 while (end_t==false)
 {
  At=time_step_choice(main,stgx,stgy);
  time_step(main,stgx,stgy);
  compute_R(stgx,stgy);
  compute_velp(stgx,stgy);
  coeffs_bp(main,stgx,stgy);
  instant_solver(main);
  compute_vel_instant(main,stgx,stgy);
  if ((stationary(main,stgx,stgy)==true || t>t_lim))
  {
   end_t=true;
  }
 }
}
double time_step_choice (vector< vector<main_mesh> >& main,vector< vector<
    ↪ stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy)
{
 double Ax;
 double v_max;
 Ax=find_Ax_min(main);
 v_max=find_v_max(stgx,stgy);
 double tc,td;
 tc=C1*Ax/v_max;
 td=C2*Re*Ax*Ax;
 return min(tc,td);
}
double find_Ax_min(vector< vector<main_mesh> >& main)
{
 double min=1e30;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
```

```
   if (main[i][j].Ax<main[i][j].Ay && main[i][j].Ax<min)
   {
    min=main[i][j].Ax;
   }
   else if (main[i][j].Ax>=main[i][j].Ay && main[i][j].Ay<min)
   {
    min=main[i][j].Ay;
   }
  }
 }
 return min;
}
double find_v_max(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪  stgy)
{
 // The velocities are evaluated on the pressure nodes
 double max=0;
 double u,v,mod_v;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   u=(stgx[i-1][j].u+stgx[i][j].u)/2;
   v=(stgy[i][j].v+stgy[i][j-1].v)/2;
   mod_v=sqrt(u*u+v*v);
   if (mod_v>max)
   {
    max=mod_v;
   }
  }
 }
 for (int i=1;i<N+1;i++)
 {
  u=(stgx[i-1][M+1].u+stgx[i][M+1].u)/2;
  v=stgy[i][M+1].v;
  mod_v=sqrt(u*u+v*v);
   if (mod_v>max)
   {
    max=mod_v;
   }
 }
 if (max==0)
 {
  return 1e-30;
 }
 else
 {
  return max;
 }
}
void time_step (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 t=t+At;
 // Main mesh
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   main[i][j].P_ant=main[i][j].P;
   main[i][j].P_sup=main[i][j].P;
  }
```

```
 }
 // Staggered-x mesh
 for (int i=0; i<N+1; i++)
 {
  for (int j=0; j<M+2;j++)
  {
   stgx[i][j].u_ant=stgx[i][j].u;
   stgx[i][j].R2=stgx[i][j].R;
  }
 }
 // Staggered-x mesh
 for (int i=0; i<N+2; i++)
 {
  for (int j=0; j<M+1;j++)
  {
   stgy[i][j].v_ant=stgy[i][j].v;
   stgy[i][j].R2=stgy[i][j].R;
  }
 }
}
void compute_R (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
 compute_Ru_ant(stgx,stgy);
 compute_Rv_ant(stgx,stgy);
}
void compute_Ru_ant (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh>
    ↪  >& stgy)
{
 // No contact solid nodes
 int cont,i,j;
 cont=T_u[0][0];
 double vA,vB,uP,uE,uW,uS,uN,ue,uw,us,un,dpe,dpw,dps,dpn;
 double Aa,Ab,Fe,Fw,Fn,Fs,Ax,Ay;
 for (int k=1;k<cont;k++)
 {
  i=T_u[k][0];
  j=T_u[k][1];
  uP=stgx[i][j].u_ant;
  uE=stgx[i+1][j].u_ant;
  uW=stgx[i-1][j].u_ant;
  uN=stgx[i][j+1].u_ant;
  uS=stgx[i][j-1].u_ant;
  dpe=stgx[i+1][j].r[0]-stgx[i][j].r[0];
  dpw=stgx[i][j].r[0]-stgx[i-1][j].r[0];
  dpn=stgx[i][j+1].r[1]-stgx[i][j].r[1];
  dps=stgx[i][j].r[1]-stgx[i][j-1].r[1];
  Ax=stgx[i][j].Ax;
  Ay=stgx[i][j].Ay;
  Aa=stgx[i][j].r[0]-stgx[i][j].xw;
  Ab=stgx[i][j].xe-stgx[i][j].r[0];
  Fe=0.5*Ay*(uP+uE);
  Fw=0.5*Ay*(uP+uW);
  vA=stgy[i][j].v_ant;
  vB=stgy[i+1][j].v_ant;
  Fn=Aa*vA+Ab*vB;
  vA=stgy[i][j-1].v_ant;
  vB=stgy[i+1][j-1].v_ant;
  Fs=Aa*vA+Ab*vB;
  if (i==N-1) // UDS
  {
   ue=UDS(uP,uE,Fe);
```

```
  }
  else
  {
   ue=phi_face_pos(Fe,stgx[i][j].xe,stgx[i][j].r[0],uP,stgx[i+1][j].r[0],uE,stgx[i
       ↪ -1][j].r[0],uW,stgx[i+2][j].r[0],stgx[i+2][j].u_ant);
  }
  if (i==1) // UDS
  {
   uw=UDS(uW,uP,Fw);
  }
  else
  {
   uw=phi_face_neg(Fw,stgx[i][j].xw,stgx[i][j].r[0],uP,stgx[i-1][j].r[0],uW,stgx[i
       ↪ +1][j].r[0],uE,stgx[i-2][j].r[0],stgx[i-2][j].u_ant);
  }
  if (j==M)
  {
   un=uN;
  }
  else
  {
   un=phi_face_pos(Fn,stgx[i][j].yn,stgx[i][j].r[1],uP,stgx[i][j+1].r[1],uN,stgx[i
       ↪ ][j-1].r[1],uS,stgx[i][j+2].r[1],stgx[i][j+2].u_ant);
  }
  if (j==1)
  {
   us=uS;
  }
  else
  {
   us=phi_face_neg(Fs,stgx[i][j].ys,stgx[i][j].r[1],uP,stgx[i][j-1].r[1],uS,stgx[i
       ↪ ][j+1].r[1],uN,stgx[i][j-2].r[1],stgx[i][j-2].u_ant);
  }
  stgx[i][j].R=1/(Ax*Ay)*(-(Fe*ue-Fw*uw+Fn*un-Fs*us)+1/Re*(((uE-uP)/dpe-(uP-uW)/
      ↪ dpw)*Ay+((uN-uP)/dpn-(uP-uS)/dps)*Ax));
 }
 cont = Tw_u[0][0];
 for (int k=1;k<cont;k++)
 {
  i=Tw_u[k][0];
  j=Tw_u[k][1];
  stgx[i][j].R=0;
 }
 for (int i=0;i<N+1;i++)
 {
  stgx[i][0].R=0;
  stgx[i][M+1].R=0;
 }
 for (int j=1;j<M+1;j++)
 {
  stgx[0][j].R=0;
  stgx[N][j].R=0;
 }
}
void compute_Rv_ant (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh>
    ↪  >& stgy)
{
 // No contact solid nodes
 int cont,i,j;
 cont=T_v[0][0];
 double uA,uB,vP,vE,vW,vS,vN,ve,vw,vs,vn,dpe,dpw,dps,dpn;
 double Aa,Ab,Fe,Fw,Fn,Fs,Ax,Ay;
```

```
for (int k=1;k<cont;k++)
{
 i=T_v[k][0];
 j=T_v[k][1];
 vP=stgy[i][j].v_ant;
 vE=stgy[i+1][j].v_ant;
 vW=stgy[i-1][j].v_ant;
 vN=stgy[i][j+1].v_ant;
 vS=stgy[i][j-1].v_ant;
 dpe=stgy[i+1][j].r[0]-stgy[i][j].r[0];
 dpw=stgy[i][j].r[0]-stgy[i-1][j].r[0];
 dpn=stgy[i][j+1].r[1]-stgy[i][j].r[1];
 dps=stgy[i][j].r[1]-stgy[i][j-1].r[1];
 Ax=stgy[i][j].Ax;
 Ay=stgy[i][j].Ay;
 Fn=0.5*Ax*(vN+vP);
 Fs=0.5*Ax*(vS+vP);
 Aa=stgy[i][j].r[1]-stgy[i][j].ys;
 Ab=stgy[i][j].yn-stgy[i][j].r[1];
 uA=stgx[i][j].u_ant;
 uB=stgx[i][j+1].u_ant;
 Fe=uA*Aa+uB*Ab;
 uA=stgx[i-1][j].u_ant;
 uB=stgx[i-1][j+1].u_ant;
 Fw=uA*Aa+uB*Ab;
 if (i==N)
 {
  ve=vE;
 }
 else
 {
  ve=phi_face_pos(Fe,stgy[i][j].xe,stgy[i][j].r[0],vP,stgy[i+1][j].r[0],vE,stgy[i
      ↪ -1][j].r[0],vW,stgy[i+2][j].r[0],stgy[i+2][j].v_ant);
 }
 if (i==1)
 {
  vw=vW;
 }
 else
 {
  vw=phi_face_neg(Fw,stgy[i][j].xw,stgy[i][j].r[0],vP,stgy[i-1][j].r[0],vW,stgy[i
      ↪ +1][j].r[0],vE,stgy[i-2][j].r[0],stgy[i-2][j].v_ant);
 }
 if (j==M-1) // UDS
 {
  vn=UDS(vP,vN,Fn);
 }
 else
 {
  vn=phi_face_pos(Fn,stgy[i][j].yn,stgy[i][j].r[1],vP,stgy[i][j+1].r[1],vN,stgy[i
      ↪ ][j-1].r[1],vS,stgy[i][j+2].r[1],stgy[i][j+2].v_ant);
 }
 if (j==1)// UDS
 {
  vs=UDS(vS,vP,Fs);
 }
 else
 {
  vs=phi_face_neg(Fs,stgy[i][j].ys,stgy[i][j].r[1],vP,stgy[i][j-1].r[1],vS,stgy[i
      ↪ ][j+1].r[1],vN,stgy[i][j-2].r[1],stgy[i][j-2].v_ant);
 }
```

```
  stgy[i][j].R=1/(Ax*Ay)*(-(Fe*ve-Fw*vw+Fn*vn-Fs*vs)+1/Re*(((vE-vP)/dpe-(vP-vW)/
      ↪ dpw)*Ay+((vN-vP)/dpn-(vP-vS)/dps)*Ax));
 }
 cont = Tw_v[0][0];
 for (int k=1;k<cont;k++)
 {
  i=Tw_v[k][0];
  j=Tw_v[k][1];
  stgy[i][j].v_P=0;
 }
 for (int i=0;i<N+2;i++)
 {
  stgy[i][0].R=0;
  stgy[i][M].R=0;
 }
 for (int j=1;j<M;j++)
 {
  stgy[0][j].R=0;
  stgy[N+1][j].R=0;
 }
}
double phi_face_pos(double m_dot, double xe, double xP, double phiP, double xE,
    ↪ double phiE, double xW,double phiW, double xEE, double phiEE)
{
 double xD, phiD,xC,phiC,xU,phiU;
 if (m_dot>0)
 {
  xD=xE;
  phiD=phiE;
  xC=xP;
  phiC=phiP;
  xU=xW;
  phiU=phiW;
 }
 else
 {
  xD=xP;
  phiD=phiP;
  xC=xE;
  phiC=phiE;
  xU=xEE;
  phiU=phiEE;
 }

 double norm_phic,norm_xe, norm_phie, norm_xc;
 if (phiD==phiU)
 {
  return phiD;
 }
 else
 {
  norm_phic=(phiC-phiU)/(phiD-phiU); // vigilar que phiD no debe ser igual a phiU
  norm_xe=(xe-xU)/(xD-xU);
  norm_xc=(xC-xU)/(xD-xU);
  if (SCHEME==0) //UDS (ot FUDS)
  {
   norm_phie=norm_phic;
  }
  else if (SCHEME==1) //CDS
  {
   norm_phie=(norm_xe-norm_xc)/(1-norm_xc)+(norm_xe-1)/(norm_xc-1)*norm_phic;
  }
```

```
  else if (SCHEME==2) //SUDS
  {
   norm_phie=norm_xe/norm_xc*norm_phic;
  }
  else if (SCHEME==3) //QUICK
  {
   norm_phie=norm_xe+(norm_xe*(norm_xe-1))/(norm_xc*(norm_xc-1))*(norm_phic-
       ↪ norm_xc);
  }
  else if (SCHEME==4) //SMART
  {
   if (norm_phic>0 && norm_phic<norm_xc/3)
   {
    norm_phie=-(norm_xe*(1-3*norm_xc+2*norm_xe))/(norm_xc*(norm_xc-1))*norm_phic;
   }
   else if(norm_phic>norm_xc/6 && norm_phic<norm_xc/norm_xe*(1+norm_xe-norm_xc))
   {
    norm_phie=norm_xe*(norm_xe-norm_xc)/(1-norm_xc)+norm_xe*(norm_xe-1)/(norm_xc*(
        ↪ norm_xc-1))*norm_phic;
   }
   else if (norm_phic>norm_xc/norm_xe*(1+norm_xe-norm_xc) && norm_phic<1)
   {
    norm_phie=1;
   }
   else
   {
    norm_phie=norm_phic;
   }
  }
  return phiU+(phiD-phiU)*norm_phie;
 }

}
double phi_face_neg (double m_dot, double xw, double xP, double phiP, double xW,
    ↪ double phiW, double xE,double phiE, double xWW, double phiWW)
{
 double xD, phiD,xC,phiC,xU,phiU;
 if (m_dot>0)
 {
  xD=xP;
  phiD=phiP;
  xC=xW;
  phiC=phiW;
  xU=xWW;
  phiU=phiWW;
 }
 else
 {
  xD=xW;
  phiD=phiW;
  xC=xP;
  phiC=phiP;
  xU=xE;
  phiU=phiE;
 }
 double norm_phic,norm_xe, norm_phie, norm_xc;
 if (phiD==phiU)
 {
  return phiD;
 }
 norm_phic=(phiC-phiU)/(phiD-phiU);
 norm_xe=(xw-xU)/(xD-xU);
```

```
 norm_xc=(xC-xU)/(xD-xU);
 if (SCHEME==0) //UDS (ot FUDS)
 {
  norm_phie=norm_phic;
 }
 else if (SCHEME==1) //CDS
 {
  norm_phie=(norm_xe-norm_xc)/(1-norm_xc)+(norm_xe-1)/(norm_xc-1)*norm_phic;
 }
 else if (SCHEME==2) //SUDS
 {
  norm_phie=norm_xe/norm_xc*norm_phic;
 }
 else if (SCHEME==3) //QUICK
 {
  norm_phie=norm_xe+(norm_xe*(norm_xe-1))/(norm_xc*(norm_xc-1))*(norm_phic-norm_xc
     ↪ );
 }
 else if (SCHEME==4) //SMART
 {
  if (norm_phic>0 && norm_phic<norm_xc/3)
  {
   norm_phie=-(norm_xe*(1-3*norm_xc+2*norm_xe))/(norm_xc*(norm_xc-1))*norm_phic;
  }
  else if(norm_phic>norm_xc/6 && norm_phic<norm_xc/norm_xe*(1+norm_xe-norm_xc))
  {
   norm_phie=norm_xe*(norm_xe-norm_xc)/(1-norm_xc)+norm_xe*(norm_xe-1)/(norm_xc*(
       ↪ norm_xc-1))*norm_phic;
  }
  else if (norm_phic>norm_xc/norm_xe*(1+norm_xe-norm_xc) && norm_phic<1)
  {
   norm_phie=1;
  }
  else
  {
   norm_phie=norm_phic;
  }
 }
 if (norm_phic==0)
 {
  norm_phie=0;
 }
 else if(norm_phic==1)
 {
  norm_phie=1;
 }
 return phiU+(phiD-phiU)*norm_phie;
}
double UDS (double v_inf, double v_sup, double F)
{
 if (F>=0)
 {
  return v_inf;
 }
 else
 {
  return v_sup;
 }
}
void compute_velp(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪  stgy)
{
```

```
 // Staggered-x
 for (int i=0;i<N+1;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   stgx[i][j].u_P=stgx[i][j].u_ant+At*(1.5*stgx[i][j].R-0.5*stgx[i][j].R2);
  }
 }
 // Staggered-y
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+1;j++)
  {
   stgy[i][j].v_P=stgy[i][j].v_ant+At*(1.5*stgy[i][j].R-0.5*stgy[i][j].R2);
  }
 }
}
void instant_solver(vector< vector<main_mesh> >& main)
{
 bool convergence=false;
 while (convergence==false)
 {
  iterative_solver(main);
  if (error_max(main)<delta)
  {
   convergence=true;
  }
  else
  {
   iteration_step(main);
  }
 }
}
void iterative_solver(vector< vector<main_mesh> >& main)
{
 if (line_by_line==false)
 {
  int cont,i,j;
  // Inner nodes
  cont=T[0][0];
  for (int k=1;k<cont;k++)
  {
   i=T[k][0];
   j=T[k][1];
   main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
       ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
       ↪ i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  }
  cont=Te[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Te[k][0];
   j=Te[k][1];
   main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
       ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
       ↪ i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  }
  cont=Tw[0][0];
  for (int k=1;k<cont;k++)
  {
```

```
 i=Tw[k][0];
 j=Tw[k][1];
 main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
     ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
     ↪ i][j].bp)/main[i][j].ap;
 main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
}
cont=Ts[0][0];
for (int k=1;k<cont;k++)
{
 i=Ts[k][0];
 j=Ts[k][1];
 main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
     ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
     ↪ i][j].bp)/main[i][j].ap;
 main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
}
cont=Tn[0][0];
for (int k=1;k<cont;k++)
{
 i=Tn[k][0];
 j=Tn[k][1];
 main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
     ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
     ↪ i][j].bp)/main[i][j].ap;
 main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
}
// Boundary nodes
// Top and bottom nodes
for (int i=1;i<N+1;i++)
{
 main[i][0].P=(main[i][0].ae*main[i+1][0].P_sup+main[i][0].aw*main[i-1][0].P_sup
     ↪ +main[i][0].an*main[i][1].P_sup+main[i][0].bp)/main[i][0].ap;
 main[i][0].P=main[i][0].P_sup+fr*(main[i][0].P-main[i][0].P_sup);
 main[i][M+1].P=(main[i][M+1].ae*main[i+1][M+1].P_sup+main[i][M+1].aw*main[i-1][
     ↪ M+1].P_sup+main[i][M+1].as*main[i][M].P_sup+main[i][M+1].bp)/main[i][M
     ↪ +1].ap;
 main[i][M+1].P=main[i][M+1].P_sup+fr*(main[i][M+1].P-main[i][M+1].P_sup);
}
// Right and left nodes
for (int j=1;j<M+1;j++)
{
 i=0;
 main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].an*main[i][j+1].P_sup
     ↪ +main[i][j].as*main[i][j-1].P_sup+main[i][j].bp)/main[i][j].ap;
 main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
 i=N+1;
 main[i][j].P=(main[i][j].aw*main[i-1][j].P_sup+main[i][j].an*main[i][j+1].P_sup
     ↪ +main[i][j].as*main[i][j-1].P_sup+main[i][j].bp)/main[i][j].ap;
 main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
}
// Corner nodes
i=0;
j=0;
main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].an*main[i][j+1].P_sup+
    ↪ main[i][j].bp)/main[i][j].ap;
main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
i=0;
j=M+1;
main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].as*main[i][j-1].P_sup+
    ↪ main[i][j].bp)/main[i][j].ap;
main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
```

```
  i=N+1;
  j=0;
  main[i][j].P=(main[i][j].aw*main[i-1][j].P_sup+main[i][j].an*main[i][j+1].P_sup+
      ↪ main[i][j].bp)/main[i][j].ap;
  main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  i=N+1;
  j=M+1;
  main[i][j].P=(main[i][j].aw*main[i-1][j].P_sup+main[i][j].as*main[i][j-1].P_sup+
      ↪ main[i][j].bp)/main[i][j].ap;
  main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
 }

}
double error_max (vector< vector<main_mesh> >& main)
{
 double error=0;
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (fabs(main[i][j].P-main[i][j].P_sup)>error && main[i][j].mat!=0)
   {
    error=fabs(main[i][j].P-main[i][j].P_sup);
   }
  }
 }
 return error;
}
void iteration_step(vector< vector<main_mesh> >& main)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (main[i][j].mat!=0)
   {
    main[i][j].P_sup=main[i][j].P;
   }
  }
 }
}
void compute_vel_instant (vector< vector<main_mesh> >& main,vector< vector<
    ↪ stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // Stagg-x mesh
 int cont,i,j;
 cont=T_u[0][0];
 for (int k=1;k<cont;k++)
 {
  i=T_u[k][0];
  j=T_u[k][1];
  stgx[i][j].u=stgx[i][j].u_P-At*(main[i+1][j].P-main[i][j].P)/stgx[i][j].Ax;
 }

 for (int i=0; i<N+1;i++)
 {
  stgx[i][0].u=u_B;
  stgx[i][M+1].u=u_T;
 }
 // Right & Left nodes velocities
 for (int j=1;j<M+1;j++)
 {
```

```
  stgx[0][j].u=u_L;
  stgx[N][j].u=u_R;
 }
  // Solid nodes
 if (Tw_u[0][0]>0)
 {
  cont = Tw_u[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Tw_u[k][0];
   j=Tw_u[k][1];
   stgx[i][j].u=0;
  }
 }
 // Stagg-y-mesh
 cont=T_v[0][0];
 for (int k=1;k<cont;k++)
 {
  i=T_v[k][0];
  j=T_v[k][1];
  stgy[i][j].v=stgy[i][j].v_P-At*(main[i][j+1].P-main[i][j].P)/stgy[i][j].Ay;
 }
  // Top & Bottom nodes velocities
 for (int i=0; i<N+2;i++)
 {
  stgy[i][0].v=v_B;
  stgy[i][M].v=v_T;
 }
 // Right & Left nodes velocities
 for (int j=1;j<M;j++)
 {
  stgy[0][j].v=v_L;
  stgy[N+1][j].v=v_R;
 }
  // Solid nodes
 if (Tw_v[0][0]>0)
 {
  cont = Tw_v[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Tw_v[k][0];
   j=Tw_v[k][1];
   stgy[i][j].v=0;
  }
 }
}
void save_vel(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
 ofstream file;
 double u,v;
 file.open("Nodal_u");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   if (i==0)
   {
    u=stgx[i][j].u;
    file<<u<<endl;
   }
   else if (i==N+1)
```

```
   {
    u=stgx[i-1][j].u;
    file<<u<<endl;
   }
   else
   {
    u=(stgx[i-1][j].u+stgx[i][j].u)/2;
    file<<u<<endl;
   }
  }
 }
 file.close();
 file.open("Nodal_v");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   if (j==0)
   {
    v=stgy[i][j].v;
    file<<v<<endl;
   }
   else if (j==M+1)
   {
    v=stgy[i][j-1].v;
    file<<v<<endl;
   }
   else
   {
    v=(stgy[i][j].v+stgy[i][j-1].v)/2;
    file<<v<<endl;
   }
  }
 }
 file.close();
}
void save_pos(vector< vector<main_mesh> >& main)
{
 ofstream file;
 file.open("Position");
 for (int j=0;j<M+2;j++)
 {
  file<<main[1][j].r[1]<<endl;
 }
 for (int i=0;i<N+2;i++)
 {
  file<<main[i][1].r[0]<<endl;
 }
 file.close();
}
void save_lengths()
{
 ofstream file;
 file.open("Lengths");
 file<<N+2<<endl;
 file<<M+2<<endl;
 file.close();
}
void save_P(vector< vector<main_mesh> >& main)
{
 ofstream file;
 double u,v;
```

```
 file.open("Pressure");
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   file<<main[i][j].P<<endl;
  }
 }
 file.close();
}
void comprovation (vector< vector<main_mesh> >& main)
{
 std::ofstream ofs;
 ofs.open("Coefs.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("Coefs.txt",std::ios_base::app);
 file<<"AP"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].ap<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AE"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].ae<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AW"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].aw<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AS"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].as<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AN"<<endl;
```

```
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].an<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"BP"<<endl;
 file<<endl;
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].bp<<"\t";
  }
  file<<endl;
 }
 file.close();
 system("pause");
}
void save_vel_gnu (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪  stgx,vector< vector<stg_y_mesh> >& stgy)
{
 double u,v;
 double scale=0.01;
 std::ofstream ofs;
 ofs.open("Velocity_field.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("Velocity_field.txt",std::ios_base::app);
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   // x-position
   file<<main[i][j].r[0]<<"␣";
   // y_position
   file<<main[i][j].r[1]<<"␣";
   // x-velocity
   if (i==0)
   {
    u=stgx[i][j].u;
   }
   else if (i==N+1)
   {
    u=stgx[i-1][j].u;
   }
   else
   {
    u=(stgx[i-1][j].u+stgx[i][j].u)/2;
   }
   // y-velocity
   if (j==0)
   {
    v=stgy[i][j].v;
   }
   else if (j==M+1)
   {
    v=stgy[i][j-1].v;
```

```
    }
    else
    {
     v=(stgy[i][j].v+stgy[i][j-1].v)/2;
    }
    file<<scale*u/sqrt(u*u+v*v)<<" "<<scale*v/sqrt(v*v+u*u)<<" "<<sqrt(u*u+v*v)<<
        ↪ endl;
   }

 }
 //gnuplot representation > plot 'Velocity_field.txt' with vectors filled lc
     ↪ palette
}
bool stationary (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 double maxP=0,max_vel=0;
 int im,jm;
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (fabs(main[i][j].P-main[i][j].P_ant)>maxP)
   {
    maxP=fabs(main[i][j].P-main[i][j].P_ant)/main[i][j].P;
    im=i;
    jm=j;
   }
  }
 }
 for (int i=1;i<N;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (fabs(stgx[i][j].u-stgx[i][j].u_ant)>max_vel)
   {
    max_vel=fabs(stgx[i][j].u-stgx[i][j].u_ant);
   }
  }
 }
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M;j++)
  {
   if (fabs(stgy[i][j].v-stgy[i][j].v_ant)>max_vel)
   {
    max_vel=fabs(stgy[i][j].v-stgy[i][j].v_ant);
   }
  }
 }
 if (max_vel>delta_v || maxP>(delta_P))
 {
  if (t>n_it*t_show)
  {
   cout<<"Time instant: "<<t<<" s"<<endl;
   if (max_vel/delta_v > maxP/(delta_P))
   {
    cout<<"Stationary status (vel): "<< delta_v/max_vel*100<<"%"<<endl;
   }
   else
   {
    cout<<"Stationary status (P): "<< delta_P/maxP*100<<"%"<<endl;
```

```
    }
    n_it++;
    cout<<"----------------------------------------"<<endl;
   }

   return false;
 }
 else
 {

   return true;
 }
}
void verif_NS1 (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 double me,mw,ms,mn,maxm=0,maxx=0,maxy=0,err;
 double uP,uE,uW,uN,uS,vAn,vBn,vAs,vBs,vP,vE,vW,vN,vS,uAe,uBe,uAw,uBw,Aa,Ab;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   mn=stgy[i][j].v*main[i][j].Ax;
   ms=stgy[i][j-1].v*main[i][j].Ax;
   me=stgx[i][j].u*main[i][j].Ay;
   mw=stgx[i-1][j].u*main[i][j].Ay;
   err=mw+ms-me-mn;
   if (fabs(err)>maxm)
   {
    maxm=fabs(err);
   }
  }
 }
 int cont;
 cont=T_u[0][0];
 int i,j;
 for (int k=1;k<cont;k++)
 {
  i=T_u[k][0];
  j=T_u[k][1];
  uP=stgx[i][j].u;
   uE=stgx[i+1][j].u;
   uW=stgx[i-1][j].u;
   uN=stgx[i][j+1].u;
   uS=stgx[i][j-1].u;
   vAn=stgy[i][j].v;
   vBn=stgy[i+1][j].v;
   vAs=stgy[i][j-1].v;
   vBs=stgy[i+1][j-1].v;
   Aa=stgx[i][j].r[0]-stgx[i][j].xw;
   Ab=stgx[i][j].xe-stgx[i][j].r[0];
   mn=(vAn*Aa+vBn*Ab);
   ms=(vAs*Aa+vBs*Ab);
   me=0.5*stgx[i][j].Ay*(uE+uP);
   mw=0.5*stgx[i][j].Ay*(uW+uP);
   err=mw+ms-me-mn;
   if (fabs(err)>maxx)
   {
    maxx=fabs(err);
   }

 }
```

```
 cont=T_v[0][0];
 for (int k=1;k<cont;k++)
 {
  i=T_v[k][0];
  j=T_v[k][1];
  vP=stgy[i][j].v;
   vE=stgy[i+1][j].v;
   vW=stgy[i-1][j].v;
   vN=stgy[i][j+1].v;
   vS=stgy[i][j-1].v;
   uAe=stgx[i][j].u;
   uBe=stgx[i][j+1].u;
   uAw=stgx[i-1][j].u;
   uBw=stgx[i-1][j+1].u;
   Aa=stgy[i][j].r[1]-stgy[i][j].ys;
   Ab=stgy[i][j].yn-stgy[i][j].r[1];
   mn=0.5*stgy[i][j].Ax*(vN+vP);
   ms=0.5*stgy[i][j].Ax*(vS+vP);
   me=(uAe*Aa+uBe*Ab);
   mw=(uAw*Aa+uBw*Ab);
   err=mw+ms-me-mn;
   if (fabs(err)>maxy)
   {
    maxy=fabs(err);
   }
 }
 cout<<"Main:␣"<<maxm<<endl;
 cout<<"Stagg-x:␣"<<maxx<<endl;
 cout<<"Stagg-y:␣"<<maxy<<endl;

}
void error_P(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >& stgx,
    ↪ vector< vector<stg_y_mesh> >& stgy)
{
 double maxP=0,maxP2=0;
 double pP,pE,pW,pS,pN,Ax,Ay,uE,uW,vS,vN;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   pP=main[i][j].P;
   pE=main[i+1][j].P;
   pW=main[i-1][j].P;
   pS=main[i][j-1].P;
   pN=main[i][j+1].P;
   Ax=main[i][j].Ax;
   Ay=main[i][j].Ay;
   uE=stgx[i][j].u_P;
   uW=stgx[i-1][j].u_P;
   vN=stgy[i][j].v_P;
   vS=stgy[i][j-1].v_P;
   maxP=max(maxP,fabs((pE-pP)/Ax*Ay-(pP-pW)/Ax*Ay+(pN-pP)/Ay*Ax-(pP-pS)/Ay*Ax-1/At
       ↪ *(uE*Ay-uW*Ay+vN*Ax-vS*Ax)));
   maxP2=max(maxP2,fabs(main[i][j].ap*pP-main[i][j].ae*pE-main[i][j].aw*pW-main[i
       ↪ ][j].an*pN-main[i][j].as*pS-main[i][j].bp));
  }
 }
 cout<<"Pressure␣errror␣discretization:␣"<<maxP<<"␣␣␣"<<maxP2<<endl;
}
void save_Bench(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
```

```
 std::ofstream ofs;
 ofs.open("u_60x60_UDS_Re100.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("u_60x60_UDS_Re100.txt",std::ios_base::app);
 int i=(N+1)/2;
 for (int j=0;j<M+2;j++)
 {
  file<<stgx[i][j].r[1]<<"␣"<<stgx[i][j].u<<endl;
 }
 file.close();
 ofs.open("v_60x60_UDS_Re100.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 file.open("v_60x60_UDS_Re100.txt",std::ios_base::app);
 int j=(M+1)/2;
 for (i=0;i<N+2;i++)
 {
  file<<stgy[i][j].r[0]<<"␣"<<stgy[i][j].v<<endl;
 }
 file.close();
}
void save_u_v(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
 std::ofstream ofs;
 ofs.open("u.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("u.txt",std::ios_base::app);
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+1;i++)
  {
   file<<stgx[i][j].u<<"\t";
  }
  file<<endl;
 }
 file.close();
 ofs.open("v.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 file.open("v.txt",std::ios_base::app);
 for (int j=0;j<M+1;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<stgy[i][j].v<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_R (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy
    ↪ )
{
 std::ofstream ofs;
 ofs.open("R.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("R.txt",std::ios_base::app);
 file<<"Ru:␣"<<endl;
 for (int j=0;j<M+2;j++)
 {
```

```
  for (int i=0;i<N+1;i++)
  {
   file<<stgx[i][j].R<<"\t";
  }
  file<<endl;
 }
 file<<"Rv: "<<endl;
 for (int j=0;j<M+1;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<stgy[i][j].R<<"\t";
  }
  file<<endl;
 }
 file.close();

}
void save_R2 (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
 std::ofstream ofs;
 ofs.open("R2.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("R2.txt",std::ios_base::app);
 file<<"Ru2: "<<endl;
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+1;i++)
  {
   file<<stgx[i][j].R2<<"\t";
  }
  file<<endl;
 }
 file<<"Rv2: "<<endl;
 for (int j=0;j<M+1;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<stgy[i][j].R2<<"\t";
  }
  file<<endl;
 }
 file.close();

}
void save_parameters ()
{
 std::ofstream ofs;
 ofs.open("Inputa_Data_time.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("Inputa_Data_Results.txt",std::ios_base::app);
 file<<"Re: "<<Re<<endl;
 file<<"Mesh density: "<<N+2<<"x"<<M+2<<endl;
 file<<"k factor: "<<k<<endl;
 file<<"delta_it: "<<delta<<endl;
 file<<"delta_v: "<<delta_v<<endl;
 file<<"delta_P: "<<delta_P<<endl;
 file<<"Scheme: ";
 if (SCHEME==0)
```

```
 {
  file<<"UDS"<<endl;
 }
 else if (SCHEME==1)
 {
  file<<"CDS"<<endl;
 }
 else if (SCHEME==2)
 {
  file<<"SUDS"<<endl;
 }
 else if (SCHEME==3)
 {
   file<<"QUICK"<<endl;
 }
 else if (SCHEME==4)
 {
   file<<"SMART"<<endl;
 }
 time(&final);
 file<<"Computation␣time:␣"<<difftime(final,inici)<<"␣s"<<endl;
}
```

Appendix G

Developed code for the Differentially Heated Cavity problem

```
// Navier-Stokes solver for incompressible form (concrete for the Differentially
    ↪ heated cavity problem)
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <vector>
#include <time.h>
using namespace std;
// Constant numbers
const double C1=0.35*0.4;
const double C2=0.2*0.4;
const double C3=0.2*0.3;
int n_it=0;
const double t_show=0.5;
//*************************************************************
// Problem geometry
const double H=1;
const double L=1;
// Initial conditions
const double P0=0;
const double v0=0;
const double sigma0=-0.04;
// Boundary conditions for veocity
const double u_T=0,v_T=0;
const double u_B=0,v_B=0;
const double u_R=0,v_R=0;
const double u_L=0,v_L=0;
// Numerical parameters
const int N=20;
const int M=N;
const int VC=(N+2)*(M+2);
const double fr=0.8;
const double delta=1e-9;
const double delta_v=1e-4;
const double delta_P=1e-4;
const double delta_T=1e-3;
const double k=1e-6;
```

```
double At;
double t_lim=700;
// Non-dimensional numbers
const double Fr=9.81;
const double Pr=1;
const double Re=1e3;
// Type of convective scheme
 /*
  0-UDS
  1-CDS
  2-SUDS
  3-QUICK
  4-SMART
 */
const int SCHEME=0;
// Solver type
bool line_by_line=false; // If Gauss-Seidel-->true; if line-by-line--> false
//************************************************************
// Time variables for the computatinal cost
time_t inici,final;
// Time variable for the physical phnomenon
double t=0;
// Main mesh variable
struct main_mesh{
 double r[2];
 double Ax;
 double Ay;
 double P_sup;
 double P;
 double P_ant;
 double ap;
 double as;
 double an;
 double aw;
 double ae;
 double bp;
 double sigma;
 double sigma_ant;
 double Rt;
 double Rt2;
 int mat; // 0  for solid, 1 for fluid
};
// Stagg-x mesh
struct stg_x_mesh{
 double r[2];
 double xw;
 double xe;
 double yn;
 double ys;
 double Ax;
 double Ay;
 double u;
 double u_ant; // n
 double u_P;
 double R;
 double R2;
};
// Stagg-y mesh
struct stg_y_mesh{
 double r[2];
 double xw;
 double xe;
```

```
 double yn;
 double ys;
 double Ax;
 double Ay;
 double v;
 double v_ant; // n
 double v_P;
 double R;
 double R2;
};
// Connectivities matrix (declares which nodes have solid neighbourds and which
    ↪ not) - based on the main mesh
 // The first element ([0][0] gives the information of how much nodes are treated)
int Ts[2*N][2]; // south nodes are solid
int Tn[2*N][2]; // north nodes are solid
int Te[2*M][2]; // east nodes are solid
int Tw[2*M][2]; // west nodes are solid
int T[VC+1][2]; // all neighbourd nodes are fluid
int Tw_u[2*(N+50)][2]; // nodes are solid
int T_u[VC+1][2]; // nodes are flow
int Tw_v[2*(M+50)][2]; // west nodes are solid
int T_v[VC+1][2]; // all neighbourd nodes are solid
//
    ↪ ********************************************************************************************
    ↪
// Function declaration
void preprocess(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void geometry(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >& stgx
    ↪ ,vector< vector<stg_y_mesh> >& stgy);
void connectivity (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪  stgx,vector< vector<stg_y_mesh> >& stgy);
void initial_map (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void coeffs_a(vector< vector<main_mesh> >& main);
void coeffs_bp(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void time_solver (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
double time_step_choice (vector< vector<main_mesh> >& main,vector< vector<
    ↪ stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy);
void time_step (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
double find_Ax_min(vector< vector<main_mesh> >& main);
double find_v_max(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪  stgy);
void compute_R (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void compute_Ru_ant (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh>
    ↪  >& stgy);
void compute_Rv_ant (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh>
    ↪ >& stgx,vector< vector<stg_y_mesh> >& stgy);
double phi_face_pos(double m_dot, double xe, double xP, double phiP, double xE,
    ↪ double phiE, double xW,double phiW, double xEE, double phiEE);
double phi_face_neg (double m_dot, double xw, double xP, double phiP, double xW,
    ↪ double phiW, double xE,double phiE, double xWW, double phiWW);
double UDS (double v_inf, double v_sup, double F);
void compute_velp(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪  stgy);
void instant_solver(vector< vector<main_mesh> >& main);
void iterative_solver(vector< vector<main_mesh> >& main);
double error_max (vector< vector<main_mesh> >& main);
```

```
void iteration_step(vector< vector<main_mesh> >& main);
void compute_vel_instant (vector< vector<main_mesh> >& main,vector< vector<
    ↪ stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy);
void save_P(vector< vector<main_mesh> >& main);
void save_vel(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void save_pos(vector< vector<main_mesh> >& main);
void save_lengths();
void comprovation(vector< vector<main_mesh> >& main);
void save_vel_gnu (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪  stgx,vector< vector<stg_y_mesh> >& stgy);
bool stationary (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void verif_NS1 (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void error_P(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >& stgx,
    ↪ vector< vector<stg_y_mesh> >& stgy);
void save_Bench(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void save_u_v(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void save_R (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy
    ↪ );
void save_R2 (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy);
void save_parameters ();
void compute_Rt (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy);
void compute_T(vector< vector<main_mesh> >& main);
void save_T(vector< vector<main_mesh> >& main);

int main()
{
 save_lengths();
 time(&inici);
 cout<<"Start..."<<endl;
 vector< vector<main_mesh> > main(N+2, vector<main_mesh>(M+2));
 vector< vector<stg_x_mesh> > stgx(N+1, vector<stg_x_mesh>(M+2));
 vector< vector<stg_y_mesh> > stgy(N+2, vector<stg_y_mesh>(M+1));
 cout<<"Preprocess..."<<endl;
 preprocess(main,stgx,stgy);
 cout<<"Discretization␣coeffieciets␣(ap,ae,aw,an,as)..."<<endl;
 coeffs_a(main);
 cout<<"Temporal␣solver..."<<endl;
 time_solver(main,stgx,stgy);
 save_vel(stgx,stgy);
 save_P(main);
 save_vel_gnu(main,stgx,stgy);
 save_Bench(stgx,stgy);
 save_u_v(stgx,stgy);
 save_parameters();
 save_T(main);
}
void preprocess(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 geometry(main, stgx,stgy);
 connectivity(main,stgx,stgy);
 initial_map(main,stgx,stgy);
 save_pos(main);
}
```

```
void geometry(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >& stgx
    ↪ ,vector< vector<stg_y_mesh> >& stgy)
{
 double x_lim[N+1], y_lim[M+1];
 x_lim[0]=0;
 y_lim[0]=0;
 x_lim[N]=L;
 y_lim[M]=H;
 double si;
 double Ax=L/N;
 double Ay=H/M;
 double x1,x2,d;
 int N2,N_ant;
 // Concentrated mesh
  // x-direction
 if (N%2!=0) // even number of horizontal nodes
 {
  x_lim[(N-1)/2]=L/2-Ax/2;
  x_lim[(N+1)/2]=L/2+Ax/2;
  x1=x_lim[0];
  x2=x_lim[(N-1)/2];
  N2=(N-1)/2;
  for (int i=1;i<N2;i++)
  {
   d=1.0*i/N2-1.0;
   si=1+tanh(k*(d))/tanh(k);
   x_lim[i]=x1+si*(x2-x1);
  }
  N_ant=N2+1;
  N2=N-(N+1)/2;
  x1=x_lim[N];
  x2=x_lim[(N+1)/2];
  for (int i=1;i<N2;i++)
  {
   d=1.0*(N2-i)/N2-1.0;
   si=1+tanh(k*(d))/tanh(k);
   x_lim[i+N_ant]=x1+si*(x2-x1);
  }
 }
 else // pair number of horizontal nodes
 {
  x_lim[(N)/2]=L/2;
  x1=x_lim[0];
  x2=x_lim[N/2];
  N2=N/2;
  for (int i=1;i<N2;i++)
  {
   d=1.0*i/N2-1.0;
   si=1+tanh(k*(d))/tanh(k);
   x_lim[i]=x1+si*(x2-x1);
  }
  N_ant=N2;
  x1=x_lim[N];
  x2=x_lim[N/2];
  for (int i=1;i<N2;i++)
  {
   d=1.0*(N2-i)/N2-1.0;
   si=1+tanh(k*(d))/tanh(k);
   x_lim[i+N_ant]=x1+si*(x2-x1);
  }
 }
 // y-direction
```

```
double y1,y2;
int M2,M_ant;
if (M%2!=0) // even number of horizontal nodes
{
 y_lim[(M-1)/2]=H/2-Ay/2;
 y_lim[(M+1)/2]=H/2+Ay/2;
 y1=y_lim[0];
 y2=y_lim[(M-1)/2];
 M2=(M-1)/2;
 for (int i=1;i<M2;i++)
 {
  d=1.0*i/M2-1.0;
  si=1+tanh(k*(d))/tanh(k);
  y_lim[i]=y1+si*(y2-y1);
 }
 M_ant=M2+1;
 M2=M-(M+1)/2;
 y1=y_lim[M];
 y2=y_lim[(M+1)/2];
 for (int i=1;i<M2;i++)
 {
  d=1.0*(M2-i)/M2-1.0;
  si=1+tanh(k*(d))/tanh(k);
  y_lim[i+M_ant]=y1+si*(y2-y1);
 }
}
else // pair number of horizontal nodes
{
 y_lim[(M)/2]=H/2;
 y1=y_lim[0];
 y2=y_lim[M/2];
 M2=M/2;
 for (int i=1;i<M2;i++)
 {
  d=1.0*i/M2-1.0;
  si=1+tanh(k*(d))/tanh(k);
  y_lim[i]=y1+si*(y2-y1);
 }
 M_ant=M2;
 y1=y_lim[M];
 y2=y_lim[M/2];
 for (int i=1;i<M2;i++)
 {
  d=1.0*(M2-i)/M2-1.0;
  si=1+tanh(k*(d))/tanh(k);
  y_lim[i+M_ant]=y1+si*(y2-y1);
 }
}
// Main mesh
for (int i=1;i<N+1;i++)
{
 for (int j=1;j<M+1;j++)
 {
  main[i][j].r[0]=(x_lim[i-1]+x_lim[i])/2;
  main[i][j].r[1]=(y_lim[j-1]+y_lim[j])/2;
  main[i][j].Ax=x_lim[i]-x_lim[i-1];
  main[i][j].Ay=y_lim[j]-y_lim[j-1];

 }
}
for  (int j=1;j<M+1;j++)
{
```

```
 main[0][j].r[0]=0;
 main[0][j].r[1]=main[1][j].r[1];
 main[N+1][j].r[0]=L;
 main[N+1][j].r[1]=main[N][j].r[1];

}
for (int i=1;i<N+1;i++)
{
 main[i][0].r[0]=main[i][2].r[0];
 main[i][0].r[1]=0;
 main[i][M+1].r[0]=main[i][M].r[0];
 main[i][M+1].r[1]=H;
}
main[0][0].r[0]=0;
main[0][0].r[1]=0;
main[0][M+1].r[0]=0;
main[0][M+1].r[1]=H;
main[N+1][0].r[0]=L;
main[N+1][0].r[1]=0;
main[N+1][M+1].r[0]=L;
main[N+1][M+1].r[1]=H;
// Stagg-x mesh
for (int i=0;i<N+1;i++)
{
 for (int j=0;j<M+2;j++)
 {
  stgx[i][j].r[0]=x_lim[i];
  stgx[i][j].r[1]=main[i][j].r[1];
  //stgx[i][j].r[1]=(y_lim[j-1]+y_lim[j])/2;


 }
 /*
 stgx[i][0].r[0]=x_lim[i];
 stgx[i][0].r[1]=0;
 stgx[i][M+1].r[0]=x_lim[i];
 stgx[i][M+1].r[1]=H;
 */
}
// Stagg-y mesh
for (int j=0;j<M+1;j++)
{
 for (int i=0;i<N+2;i++)
 {
  stgy[i][j].r[0]=main[i][j].r[0];
  //stgy[i][j].r[0]=(x_lim[i-1]+x_lim[i])/2;
  stgy[i][j].r[1]=y_lim[j];
 }
 /*
 stgy[0][j].r[0]=0;
 stgy[0][j].r[1]=y_lim[j];
 stgy[N+1][j].r[0]=L;
 stgy[N+1][j].r[1]=y_lim[j];
 */
}

double xw,xe,yn,ys;
// Ax and Ay for Stagg-x mesh
for (int i=1; i<N;i++)
{
 for (int j=1; j<M+1;j++)
 {
```

```
    xw=main[i][j].r[0];
    xe=main[i+1][j].r[0];
    yn=y_lim[j];
    ys=y_lim[j-1];
    stgx[i][j].xe=xe;
    stgx[i][j].xw=xw;
    stgx[i][j].yn=yn;
    stgx[i][j].ys=ys;
    stgx[i][j].Ax=xe-xw;
    stgx[i][j].Ay=yn-ys;
   }
  }
  // Ax and Ay for Stagg-y mesh
  for (int i=1; i<N+1;i++)
  {
   for (int j=1; j<M;j++)
   {
    yn=main[i][j+1].r[1];
    ys=main[i][j].r[1];
    xw=x_lim[i-1];
    xe=x_lim[i];
    stgy[i][j].xe=xe;
    stgy[i][j].xw=xw;
    stgy[i][j].yn=yn;
    stgy[i][j].ys=ys;
    stgy[i][j].Ax=xe-xw;
    stgy[i][j].Ay=yn-ys;

   }
  }

  // Definition of the node's material
  for (int i=0;i<N+2;i++)
  {
   for (int j=0;j<M+2;j++)
   {
    main[i][j].mat=1;
   }
  }

}
void connectivity (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪  stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // Main mesh
 int cont=0;
 T[0][0]=cont;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   cont++;
   T[0][0]=cont;
   T[cont][0]=i;
   T[cont][1]=j;
  }
 }
 T[0][0]=cont+1;
 Ts[0][0]=0;
 Tn[0][0]=0;
 Tw[0][0]=0;
 Te[0][0]=0;
```

```
 // Stagg-x
 cont=0;
 T_u[0][0]=cont;
 for (int i=1;i<N;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   cont++;
   T_u[0][0]=cont;
   T_u[cont][0]=i;
   T_u[cont][1]=j;
  }
 }
 T_u[0][0]=cont+1;
 Tw_u[0][0]=0;
 // Stagg-y
 cont=0;
 T_v[0][0]=cont;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M;j++)
  {
   cont++;
   T_v[0][0]=cont;
   T_v[cont][0]=i;
   T_v[cont][1]=j;
  }
 }
 T_v[0][0]=cont+1;
 Tw_v[0][0]=0;
}
void initial_map (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // Main mesh
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   main[i][j].P=P0;
   main[i][j].P_ant=P0;
   main[i][j].P_sup=P0;
   if (i==N+1)
   {
    main[i][j].sigma=sigma0;
    main[i][j].sigma_ant=0;
   }
   else
   {
    main[i][j].sigma=0;
    main[i][j].sigma_ant=0;
   }
  }
 }
 // Stagg-x mesh
 for (int i=0;i<N+1;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   stgx[i][j].u=0;
   stgx[i][j].u_ant=0;
   if (j==M+1)
```

```
   {
    stgx[i][j].u=u_T;
    stgx[i][j].u_ant=0;
   }
  }
 }
 // Stagg-y mesh
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+1;j++)
  {
   stgy[i][j].v=v0;
   stgy[i][j].v_ant=v0;
  }
 }
 compute_R(main,stgx,stgy);
 compute_Rt(main,stgx,stgy);
}
void coeffs_a(vector< vector<main_mesh> >& main)
{
 // Internal nodes
 double dPE,dPW,dPN,dPS;
 int cont,i,j;
 cont = T[0][0];
 for (int k=1;k<cont;k++)
 {

  i=T[k][0];
  j=T[k][1];
  dPE=main[i+1][j].r[0]-main[i][j].r[0];
  dPW=main[i][j].r[0]-main[i-1][j].r[0];
  dPN=main[i][j+1].r[1]-main[i][j].r[1];
  dPS=main[i][j].r[1]-main[i][j-1].r[1];
  main[i][j].ae=main[i][j].Ay/dPE;
  main[i][j].aw=main[i][j].Ay/dPW;
  main[i][j].an=main[i][j].Ax/dPN;
  main[i][j].as=main[i][j].Ax/dPS;
  main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
 }
 cont=Te[0][0];
 for (int k=1;k<cont;k++)
 {
  i=Te[k][0];
  j=Te[k][1];
  dPE=main[i+1][j].r[0]-main[i][j].r[0];
  dPW=main[i][j].r[0]-main[i-1][j].r[0];
  dPN=main[i][j+1].r[1]-main[i][j].r[1];
  dPS=main[i][j].r[1]-main[i][j-1].r[1];
  main[i][j].ae=0;
  main[i][j].aw=main[i][j].Ay/dPW;
  main[i][j].an=main[i][j].Ax/dPN;
  main[i][j].as=main[i][j].Ax/dPS;
  main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
 }
 cont = Tw[0][0];
 for (int k=1;k<cont;k++)
 {
  i=Tw[k][0];
  j=Tw[k][1];
  dPE=main[i+1][j].r[0]-main[i][j].r[0];
  dPW=main[i][j].r[0]-main[i-1][j].r[0];
  dPN=main[i][j+1].r[1]-main[i][j].r[1];
```

```
 dPS=main[i][j].r[1]-main[i][j-1].r[1];
 main[i][j].ae=main[i][j].Ay/dPE;
 main[i][j].aw=0;
 main[i][j].an=main[i][j].Ax/dPN;
 main[i][j].as=main[i][j].Ax/dPS;
 main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
}
cont = Ts[0][0];
for (int k=1;k<cont;k++)
{
 i=Ts[k][0];
 j=Ts[k][1];
 dPE=main[i+1][j].r[0]-main[i][j].r[0];
 dPW=main[i][j].r[0]-main[i-1][j].r[0];
 dPN=main[i][j+1].r[1]-main[i][j].r[1];
 dPS=main[i][j].r[1]-main[i][j-1].r[1];
 main[i][j].ae=main[i][j].Ay/dPE;
 main[i][j].aw=main[i][j].Ay/dPW;
 main[i][j].an=main[i][j].Ax/dPN;
 main[i][j].as=0;
 main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
}
cont = Tn[0][0];
for (int k=1;k<cont;k++)
{
 i=Tn[k][0];
 j=Tn[k][1];
 dPE=main[i+1][j].r[0]-main[i][j].r[0];
 dPW=main[i][j].r[0]-main[i-1][j].r[0];
 dPN=main[i][j+1].r[1]-main[i][j].r[1];
 dPS=main[i][j].r[1]-main[i][j-1].r[1];
 main[i][j].ae=main[i][j].Ay/dPE;
 main[i][j].aw=main[i][j].Ay/dPW;
 main[i][j].an=0;
 main[i][j].as=main[i][j].Ax/dPS;
 main[i][j].ap=main[i][j].ae+main[i][j].aw+main[i][j].an+main[i][j].as;
}
// Boundary nodes
// Left and right nodes
for (int j=0;j<M+2;j++)
{
 // Left nodes
 main[0][j].ae=1;
 main[0][j].aw=0;
 main[0][j].an=0;
 main[0][j].as=0;
 main[0][j].ap=1;
 // Right nodes
 main[N+1][j].ae=0;
 main[N+1][j].aw=1;
 main[N+1][j].an=0;
 main[N+1][j].as=0;
 main[N+1][j].ap=1;
}
// Top and bottom nodes
for (int i=1;i<N+1;i++)
{
 // Bottom nodes
 main[i][0].ae=0;
 main[i][0].aw=0;
 main[i][0].an=1;
 main[i][0].as=0;
```

```
  main[i][0].ap=1;
  // Top nodes
  main[i][M+1].ae=0;
  main[i][M+1].aw=0;
  main[i][M+1].an=0;
  main[i][M+1].as=1;
  main[i][M+1].ap=1;
 }
}
void coeffs_bp(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // Internal nodes
 double ue,uw,vn,vs,Ax,Ay;
 // Internal nodes
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   ue=stgx[i][j].u_P;
   uw=stgx[i-1][j].u_P;
   vn=stgy[i][j].v_P;
   vs=stgy[i][j-1].v_P;
   Ax=main[i][j].Ax;
   Ay=main[i][j].Ay;
   main[i][j].bp=-1/At*((ue-uw)*Ay+(vn-vs)*Ax);
  }
 }
 // Boundary nodes
 // Left and right nodes
 for (int j=0;j<M+2;j++)
 {
  // Left nodes
  main[0][j].bp=0;
  // Right nodes
  main[N+1][j].bp=0;
 }

 // Top and bottom nodes
 for (int i=1;i<N+1;i++)
 {
  // Bottom nodes
  main[i][0].bp=0;
  // Top nodes
  main[i][M+1].bp=0;

 }
}
void time_solver (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 bool end_t=false;
 while (end_t==false)
 {
  At=time_step_choice(main,stgx,stgy);
  time_step(main,stgx,stgy);
  compute_R(main,stgx,stgy);
  compute_Rt(main,stgx,stgy);
  compute_T(main);
  compute_velp(stgx,stgy);
  coeffs_bp(main,stgx,stgy);
  instant_solver(main);
```

```
  compute_vel_instant(main,stgx,stgy);
  if ((stationary(main,stgx,stgy)==true || t>t_lim))
  {
   end_t=true;
  }
 }
}
double time_step_choice (vector< vector<main_mesh> >& main,vector< vector<
    ↪ stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy)
{
 double Ax;
 double v_max;
 Ax=find_Ax_min(main);
 v_max=find_v_max(stgx,stgy);
 double tc,td,tt;
 tc=C1*Ax/v_max;
 td=C2*Re*Ax*Ax;
 tt=C3*Pr*Re*Ax*Ax;
 return min(tc,min(td,tt));
}
double find_Ax_min(vector< vector<main_mesh> >& main)
{
 double min=1e30;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   if (main[i][j].Ax<main[i][j].Ay && main[i][j].Ax<min)
   {
    min=main[i][j].Ax;
   }
   else if (main[i][j].Ax>=main[i][j].Ay && main[i][j].Ay<min)
   {
    min=main[i][j].Ay;
   }
  }
 }
 return min;
}
double find_v_max(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪  stgy)
{
 // The velocities are evaluated on the pressure nodes
 double max=0;
 double u,v,mod_v;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   u=(stgx[i-1][j].u+stgx[i][j].u)/2;
   v=(stgy[i][j].v+stgy[i][j-1].v)/2;
   mod_v=sqrt(u*u+v*v);
   if (mod_v>max)
   {
    max=mod_v;
   }
  }
 }
 for (int i=1;i<N+1;i++)
 {
  u=(stgx[i-1][M+1].u+stgx[i][M+1].u)/2;
  v=stgy[i][M+1].v;
```

```
  mod_v=sqrt(u*u+v*v);
   if (mod_v>max)
   {
    max=mod_v;
   }
 }
 if (max==0)
 {
  return 1e-30;
 }
 else
 {
  return max;
 }
}
void time_step (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 t=t+At;
 // Main mesh
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   main[i][j].P_ant=main[i][j].P;
   main[i][j].P_sup=main[i][j].P;
   main[i][j].sigma_ant=main[i][j].sigma;
   main[i][j].Rt2=main[i][j].Rt;
  }
 }
 // Staggered-x mesh
 for (int i=0; i<N+1; i++)
 {
  for (int j=0; j<M+2;j++)
  {
   stgx[i][j].u_ant=stgx[i][j].u;
   stgx[i][j].R2=stgx[i][j].R;
  }
 }
 // Staggered-x mesh
 for (int i=0; i<N+2; i++)
 {
  for (int j=0; j<M+1;j++)
  {
   stgy[i][j].v_ant=stgy[i][j].v;
   stgy[i][j].R2=stgy[i][j].R;
  }
 }
}
void compute_R (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 compute_Ru_ant(stgx,stgy);
 compute_Rv_ant(main,stgx,stgy);
}
void compute_Ru_ant (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh>
    ↪  >& stgy)
{
 // No contact solid nodes
 int cont,i,j;
 cont=T_u[0][0];
 double vA,vB,uP,uE,uW,uS,uN,ue,uw,us,un,dpe,dpw,dps,dpn;
```

```
double Aa,Ab,Fe,Fw,Fn,Fs,Ax,Ay;
for (int k=1;k<cont;k++)
{
 i=T_u[k][0];
 j=T_u[k][1];
 uP=stgx[i][j].u_ant;
 uE=stgx[i+1][j].u_ant;
 uW=stgx[i-1][j].u_ant;
 uN=stgx[i][j+1].u_ant;
 uS=stgx[i][j-1].u_ant;
 dpe=stgx[i+1][j].r[0]-stgx[i][j].r[0];
 dpw=stgx[i][j].r[0]-stgx[i-1][j].r[0];
 dpn=stgx[i][j+1].r[1]-stgx[i][j].r[1];
 dps=stgx[i][j].r[1]-stgx[i][j-1].r[1];
 Ax=stgx[i][j].Ax;
 Ay=stgx[i][j].Ay;
 Aa=stgx[i][j].r[0]-stgx[i][j].xw;
 Ab=stgx[i][j].xe-stgx[i][j].r[0];
 Fe=0.5*Ay*(uP+uE);
 Fw=0.5*Ay*(uP+uW);
 vA=stgy[i][j].v_ant;
 vB=stgy[i+1][j].v_ant;
 Fn=Aa*vA+Ab*vB;
 vA=stgy[i][j-1].v_ant;
 vB=stgy[i+1][j-1].v_ant;
 Fs=Aa*vA+Ab*vB;
 if (i==N-1) // UDS
 {
  ue=UDS(uP,uE,Fe);
 }
 else
 {
  ue=phi_face_pos(Fe,stgx[i][j].xe,stgx[i][j].r[0],uP,stgx[i+1][j].r[0],uE,stgx[i
      ↪ -1][j].r[0],uW,stgx[i+2][j].r[0],stgx[i+2][j].u_ant);
 }
 if (i==1) // UDS
 {
  uw=UDS(uW,uP,Fw);
 }
 else
 {
  uw=phi_face_neg(Fw,stgx[i][j].xw,stgx[i][j].r[0],uP,stgx[i-1][j].r[0],uW,stgx[i
      ↪ +1][j].r[0],uE,stgx[i-2][j].r[0],stgx[i-2][j].u_ant);
 }
 if (j==M)
 {
  un=uN;
 }
 else
 {
  un=phi_face_pos(Fn,stgx[i][j].yn,stgx[i][j].r[1],uP,stgx[i][j+1].r[1],uN,stgx[i
      ↪ ][j-1].r[1],uS,stgx[i][j+2].r[1],stgx[i][j+2].u_ant);
 }
 if (j==1)
 {
  us=uS;
 }
 else
 {
  us=phi_face_neg(Fs,stgx[i][j].ys,stgx[i][j].r[1],uP,stgx[i][j-1].r[1],uS,stgx[i
      ↪ ][j+1].r[1],uN,stgx[i][j-2].r[1],stgx[i][j-2].u_ant);
 }
```

```
  stgx[i][j].R=1/(Ax*Ay)*(-(Fe*ue-Fw*uw+Fn*un-Fs*us)+1/Re*(((uE-uP)/dpe-(uP-uW)/
      ↪ dpw)*Ay+((uN-uP)/dpn-(uP-uS)/dps)*Ax));
 }
 cont = Tw_u[0][0];
 for (int k=1;k<cont;k++)
 {
  i=Tw_u[k][0];
  j=Tw_u[k][1];
  stgx[i][j].R=0;
 }
 for (int i=0;i<N+1;i++)
 {
  stgx[i][0].R=0;
  stgx[i][M+1].R=0;
 }
 for (int j=1;j<M+1;j++)
 {
  stgx[0][j].R=0;
  stgx[N][j].R=0;
 }
}
void compute_Rv_ant (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh>
    ↪ >& stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // No contact solid nodes
 int cont,i,j;
 cont=T_v[0][0];
 double uA,uB,vP,vE,vW,vS,vN,ve,vw,vs,vn,dpe,dpw,dps,dpn;
 double Aa,Ab,Fe,Fw,Fn,Fs,Ax,Ay;
 double sigma_P;
 for (int k=1;k<cont;k++)
 {
  i=T_v[k][0];
  j=T_v[k][1];
  vP=stgy[i][j].v_ant;
  vE=stgy[i+1][j].v_ant;
  vW=stgy[i-1][j].v_ant;
  vN=stgy[i][j+1].v_ant;
  vS=stgy[i][j-1].v_ant;
  dpe=stgy[i+1][j].r[0]-stgy[i][j].r[0];
  dpw=stgy[i][j].r[0]-stgy[i-1][j].r[0];
  dpn=stgy[i][j+1].r[1]-stgy[i][j].r[1];
  dps=stgy[i][j].r[1]-stgy[i][j-1].r[1];
  Ax=stgy[i][j].Ax;
  Ay=stgy[i][j].Ay;
  Fn=0.5*Ax*(vN+vP);
  Fs=0.5*Ax*(vS+vP);
  Aa=stgy[i][j].r[1]-stgy[i][j].ys;
  Ab=stgy[i][j].yn-stgy[i][j].r[1];
  uA=stgx[i][j].u_ant;
  uB=stgx[i][j+1].u_ant;
  Fe=uA*Aa+uB*Ab;
  uA=stgx[i-1][j].u_ant;
  uB=stgx[i-1][j+1].u_ant;
  Fw=uA*Aa+uB*Ab;
  sigma_P=(main[i][j].sigma_ant+main[i][j+1].sigma_ant)/2;
  if (i==N)
  {
   ve=vE;
  }
  else
  {
```

```
    ve=phi_face_pos(Fe,stgy[i][j].xe,stgy[i][j].r[0],vP,stgy[i+1][j].r[0],vE,stgy[i
        ↪ -1][j].r[0],vW,stgy[i+2][j].r[0],stgy[i+2][j].v_ant);
   }
   if (i==1)
   {
    vw=vW;
   }
   else
   {
    vw=phi_face_neg(Fw,stgy[i][j].xw,stgy[i][j].r[0],vP,stgy[i-1][j].r[0],vW,stgy[i
        ↪ +1][j].r[0],vE,stgy[i-2][j].r[0],stgy[i-2][j].v_ant);
   }
   if (j==M-1) // UDS
   {
    vn=UDS(vP,vN,Fn);
   }
   else
   {
    vn=phi_face_pos(Fn,stgy[i][j].yn,stgy[i][j].r[1],vP,stgy[i][j+1].r[1],vN,stgy[i
        ↪ ][j-1].r[1],vS,stgy[i][j+2].r[1],stgy[i][j+2].v_ant);
   }
   if (j==1)// UDS
   {
    vs=UDS(vS,vP,Fs);
   }
   else
   {
    vs=phi_face_neg(Fs,stgy[i][j].ys,stgy[i][j].r[1],vP,stgy[i][j-1].r[1],vS,stgy[i
        ↪ ][j+1].r[1],vN,stgy[i][j-2].r[1],stgy[i][j-2].v_ant);
   }
   stgy[i][j].R=1/(Ax*Ay)*(-(Fe*ve-Fw*vw+Fn*vn-Fs*vs)+1/Re*(((vE-vP)/dpe-(vP-vW)/
       ↪ dpw)*Ay+((vN-vP)/dpn-(vP-vS)/dps)*Ax))+1/Fr*sigma_P;
  }
  cont = Tw_v[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Tw_v[k][0];
   j=Tw_v[k][1];
   stgy[i][j].v_P=0;
  }
  for (int i=0;i<N+2;i++)
  {
   stgy[i][0].R=0;
   stgy[i][M].R=0;
  }
  for (int j=1;j<M;j++)
  {
   stgy[0][j].R=0;
   stgy[N+1][j].R=0;
  }
}
double phi_face_pos(double m_dot, double xe, double xP, double phiP, double xE,
    ↪ double phiE, double xW,double phiW, double xEE, double phiEE)
{
 double xD, phiD,xC,phiC,xU,phiU;
 if (m_dot>0)
 {
  xD=xE;
  phiD=phiE;
  xC=xP;
  phiC=phiP;
  xU=xW;
```

```
 phiU=phiW;
}
else
{
 xD=xP;
 phiD=phiP;
 xC=xE;
 phiC=phiE;
 xU=xEE;
 phiU=phiEE;
}

double norm_phic,norm_xe, norm_phie, norm_xc;
if (phiD==phiU)
{
 return phiD;
}
else
{
 norm_phic=(phiC-phiU)/(phiD-phiU); // vigilar que phiD no debe ser igual a phiU
 norm_xe=(xe-xU)/(xD-xU);
 norm_xc=(xC-xU)/(xD-xU);
 if (SCHEME==0) //UDS (ot FUDS)
 {
  norm_phie=norm_phic;
 }
 else if (SCHEME==1) //CDS
 {
  norm_phie=(norm_xe-norm_xc)/(1-norm_xc)+(norm_xe-1)/(norm_xc-1)*norm_phic;
 }
 else if (SCHEME==2) //SUDS
 {
  norm_phie=norm_xe/norm_xc*norm_phic;
 }
 else if (SCHEME==3) //QUICK
 {
  norm_phie=norm_xe+(norm_xe*(norm_xe-1))/(norm_xc*(norm_xc-1))*(norm_phic-
      ↪ norm_xc);
 }
 else if (SCHEME==4) //SMART
 {
  if (norm_phic>0 && norm_phic<norm_xc/3)
  {
   norm_phie=-(norm_xe*(1-3*norm_xc+2*norm_xe))/(norm_xc*(norm_xc-1))*norm_phic;
  }
  else if(norm_phic>norm_xc/6 && norm_phic<norm_xc/norm_xe*(1+norm_xe-norm_xc))
  {
   norm_phie=norm_xe*(norm_xe-norm_xc)/(1-norm_xc)+norm_xe*(norm_xe-1)/(norm_xc*(
       ↪ norm_xc-1))*norm_phic;
  }
  else if (norm_phic>norm_xc/norm_xe*(1+norm_xe-norm_xc) && norm_phic<1)
  {
   norm_phie=1;
  }
  else
  {
   norm_phie=norm_phic;
  }
 }
 return phiU+(phiD-phiU)*norm_phie;
}
```

```
}
double phi_face_neg (double m_dot, double xw, double xP, double phiP, double xW,
    ↪ double phiW, double xE,double phiE, double xWW, double phiWW)
{
 double xD, phiD,xC,phiC,xU,phiU;
 if (m_dot>0)
 {
  xD=xP;
  phiD=phiP;
  xC=xW;
  phiC=phiW;
  xU=xWW;
  phiU=phiWW;
 }
 else
 {
  xD=xW;
  phiD=phiW;
  xC=xP;
  phiC=phiP;
  xU=xE;
  phiU=phiE;
 }
 double norm_phic,norm_xe, norm_phie, norm_xc;
 if (phiD==phiU)
 {
  return phiD;
 }
 norm_phic=(phiC-phiU)/(phiD-phiU);
 norm_xe=(xw-xU)/(xD-xU);
 norm_xc=(xC-xU)/(xD-xU);
 if (SCHEME==0) //UDS (ot FUDS)
 {
  norm_phie=norm_phic;
 }
 else if (SCHEME==1) //CDS
 {
  norm_phie=(norm_xe-norm_xc)/(1-norm_xc)+(norm_xe-1)/(norm_xc-1)*norm_phic;
 }
 else if (SCHEME==2) //SUDS
 {
  norm_phie=norm_xe/norm_xc*norm_phic;
 }
 else if (SCHEME==3) //QUICK
 {
  norm_phie=norm_xe+(norm_xe*(norm_xe-1))/(norm_xc*(norm_xc-1))*(norm_phic-norm_xc
      ↪ );
 }
 else if (SCHEME==4) //SMART
 {
  if (norm_phic>0 && norm_phic<norm_xc/3)
  {
   norm_phie=-(norm_xe*(1-3*norm_xc+2*norm_xe))/(norm_xc*(norm_xc-1))*norm_phic;
  }
  else if(norm_phic>norm_xc/6 && norm_phic<norm_xc/norm_xe*(1+norm_xe-norm_xc))
  {
   norm_phie=norm_xe*(norm_xe-norm_xc)/(1-norm_xc)+norm_xe*(norm_xe-1)/(norm_xc*(
       ↪ norm_xc-1))*norm_phic;
  }
  else if (norm_phic>norm_xc/norm_xe*(1+norm_xe-norm_xc) && norm_phic<1)
  {
   norm_phie=1;
```

```
  }
  else
  {
   norm_phie=norm_phic;
  }
 }
 if (norm_phic==0)
 {
  norm_phie=0;
 }
 else if(norm_phic==1)
 {
  norm_phie=1;
 }
 return phiU+(phiD-phiU)*norm_phie;
}
double UDS (double v_inf, double v_sup, double F)
{
 if (F>=0)
 {
  return v_inf;
 }
 else
 {
  return v_sup;
 }
}
void compute_velp(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪  stgy)
{
 // Staggered-x
 for (int i=0;i<N+1;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   stgx[i][j].u_P=stgx[i][j].u_ant+At*(1.5*stgx[i][j].R-0.5*stgx[i][j].R2);
  }
 }
 // Staggered-y
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+1;j++)
  {
   stgy[i][j].v_P=stgy[i][j].v_ant+At*(1.5*stgy[i][j].R-0.5*stgy[i][j].R2);
  }
 }
}
void instant_solver(vector< vector<main_mesh> >& main)
{
 bool convergence=false;
 while (convergence==false)
 {
  iterative_solver(main);
  if (error_max(main)<delta)
  {
   convergence=true;
  }
  else
  {
   iteration_step(main);
  }
 }
```

```
}
void iterative_solver(vector< vector<main_mesh> >& main)
{
 if (line_by_line==false)
 {
  int cont,i,j;
  // Inner nodes
  cont=T[0][0];
  for (int k=1;k<cont;k++)
  {
   i=T[k][0];
   j=T[k][1];
   main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
       ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
       ↪ i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  }
  cont=Te[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Te[k][0];
   j=Te[k][1];
   main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
       ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
       ↪ i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  }
  cont=Tw[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Tw[k][0];
   j=Tw[k][1];
   main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
       ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
       ↪ i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  }
  cont=Ts[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Ts[k][0];
   j=Ts[k][1];
   main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
       ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
       ↪ i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  }
  cont=Tn[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Tn[k][0];
   j=Tn[k][1];
   main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].aw*main[i-1][j].P_sup
       ↪ +main[i][j].an*main[i][j+1].P_sup+main[i][j].as*main[i][j-1].P_sup+main[
       ↪ i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  }
  // Boundary nodes
  // Top and bottom nodes
  for (int i=1;i<N+1;i++)
  {
```

```
   main[i][0].P=(main[i][0].ae*main[i+1][0].P_sup+main[i][0].aw*main[i-1][0].P_sup
       ↪ +main[i][0].an*main[i][1].P_sup+main[i][0].bp)/main[i][0].ap;
   main[i][0].P=main[i][0].P_sup+fr*(main[i][0].P-main[i][0].P_sup);
   main[i][M+1].P=(main[i][M+1].ae*main[i+1][M+1].P_sup+main[i][M+1].aw*main[i-1][
       ↪ M+1].P_sup+main[i][M+1].as*main[i][M].P_sup+main[i][M+1].bp)/main[i][M
       ↪ +1].ap;
   main[i][M+1].P=main[i][M+1].P_sup+fr*(main[i][M+1].P-main[i][M+1].P_sup);
  }
  // Right and left nodes
  for (int j=1;j<M+1;j++)
  {
   i=0;
   main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].an*main[i][j+1].P_sup
       ↪ +main[i][j].as*main[i][j-1].P_sup+main[i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
   i=N+1;
   main[i][j].P=(main[i][j].aw*main[i-1][j].P_sup+main[i][j].an*main[i][j+1].P_sup
       ↪ +main[i][j].as*main[i][j-1].P_sup+main[i][j].bp)/main[i][j].ap;
   main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  }
  // Corner nodes
  i=0;
  j=0;
  main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].an*main[i][j+1].P_sup+
      ↪ main[i][j].bp)/main[i][j].ap;
  main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  i=0;
  j=M+1;
  main[i][j].P=(main[i][j].ae*main[i+1][j].P_sup+main[i][j].as*main[i][j-1].P_sup+
      ↪ main[i][j].bp)/main[i][j].ap;
  main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  i=N+1;
  j=0;
  main[i][j].P=(main[i][j].aw*main[i-1][j].P_sup+main[i][j].an*main[i][j+1].P_sup+
      ↪ main[i][j].bp)/main[i][j].ap;
  main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
  i=N+1;
  j=M+1;
  main[i][j].P=(main[i][j].aw*main[i-1][j].P_sup+main[i][j].as*main[i][j-1].P_sup+
      ↪ main[i][j].bp)/main[i][j].ap;
  main[i][j].P=main[i][j].P_sup+fr*(main[i][j].P-main[i][j].P_sup);
 }

}
double error_max (vector< vector<main_mesh> >& main)
{
 double error=0;
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (fabs(main[i][j].P-main[i][j].P_sup)>error && main[i][j].mat!=0)
   {
    error=fabs(main[i][j].P-main[i][j].P_sup);
   }
  }
 }
 return error;
}
void iteration_step(vector< vector<main_mesh> >& main)
{
 for (int i=0;i<N+2;i++)
```

```
 {
  for (int j=0;j<M+2;j++)
  {
   if (main[i][j].mat!=0)
   {
    main[i][j].P_sup=main[i][j].P;
   }
  }
 }
}
void compute_vel_instant (vector< vector<main_mesh> >& main,vector< vector<
    ↪ stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy)
{
 // Stagg-x mesh
 int cont,i,j;
 cont=T_u[0][0];
 for (int k=1;k<cont;k++)
 {
  i=T_u[k][0];
  j=T_u[k][1];
  stgx[i][j].u=stgx[i][j].u_P-At*(main[i+1][j].P-main[i][j].P)/stgx[i][j].Ax;
 }

 for (int i=0; i<N+1;i++)
 {
  stgx[i][0].u=u_B;
  stgx[i][M+1].u=u_T;
 }
 // Right & Left nodes velocities
 for (int j=1;j<M+1;j++)
 {
  stgx[0][j].u=u_L;
  stgx[N][j].u=u_R;
 }
  // Solid nodes
 if (Tw_u[0][0]>0)
 {
  cont = Tw_u[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Tw_u[k][0];
   j=Tw_u[k][1];
   stgx[i][j].u=0;
  }
 }
 // Stagg-y-mesh
 cont=T_v[0][0];
 for (int k=1;k<cont;k++)
 {
  i=T_v[k][0];
  j=T_v[k][1];
  stgy[i][j].v=stgy[i][j].v_P-At*(main[i][j+1].P-main[i][j].P)/stgy[i][j].Ay;
 }
  // Top & Bottom nodes velocities
 for (int i=0; i<N+2;i++)
 {
  stgy[i][0].v=v_B;
  stgy[i][M].v=v_T;
 }
 // Right & Left nodes velocities
 for (int j=1;j<M;j++)
 {
```

```
  stgy[0][j].v=v_L;
  stgy[N+1][j].v=v_R;
 }
  // Solid nodes
 if (Tw_v[0][0]>0)
 {
  cont = Tw_v[0][0];
  for (int k=1;k<cont;k++)
  {
   i=Tw_v[k][0];
   j=Tw_v[k][1];
   stgy[i][j].v=0;
  }
 }
}
void save_vel(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
 ofstream file;
 double u,v;
 file.open("Nodal_u");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   if (i==0)
   {
    u=stgx[i][j].u;
    file<<u<<endl;
   }
   else if (i==N+1)
   {
    u=stgx[i-1][j].u;
    file<<u<<endl;
   }
   else
   {
    u=(stgx[i-1][j].u+stgx[i][j].u)/2;
    file<<u<<endl;
   }
  }
 }
 file.close();
 file.open("Nodal_v");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   if (j==0)
   {
    v=stgy[i][j].v;
    file<<v<<endl;
   }
   else if (j==M+1)
   {
    v=stgy[i][j-1].v;
    file<<v<<endl;
   }
   else
   {
    v=(stgy[i][j].v+stgy[i][j-1].v)/2;
    file<<v<<endl;
```

```
    }
   }
  }
  file.close();
}
void save_pos(vector< vector<main_mesh> >& main)
{
 ofstream file;
 file.open("Position");
 for (int j=0;j<M+2;j++)
 {
  file<<main[1][j].r[1]<<endl;
 }
 for (int i=0;i<N+2;i++)
 {
  file<<main[i][1].r[0]<<endl;
 }
 file.close();
}
void save_lengths()
{
 ofstream file;
 file.open("Lengths");
 file<<N+2<<endl;
 file<<M+2<<endl;
 file.close();
}
void save_P(vector< vector<main_mesh> >& main)
{
 ofstream file;
 double u,v;
 file.open("Pressure");
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   file<<main[i][j].P<<endl;
  }
 }
 file.close();
}
void comprovation (vector< vector<main_mesh> >& main)
{
 std::ofstream ofs;
 ofs.open("Coefs.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("Coefs.txt",std::ios_base::app);
 file<<"AP"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].ap<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AE"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
```

```
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].ae<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AW"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].aw<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AS"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].as<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"AN"<<endl;
 file<<endl;
 for (int j=M+1;j>=0;j--)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].an<<"\t";
  }
  file<<endl;
 }
 file<<endl;
 file<<"BP"<<endl;
 file<<endl;
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].bp<<"\t";
  }
  file<<endl;
 }
 file.close();
 system("pause");
}
void save_vel_gnu (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪  stgx,vector< vector<stg_y_mesh> >& stgy)
{
 double u,v;
 double scale=0.01;
 std::ofstream ofs;
 ofs.open("Velocity_field.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
```

```
 ofstream file;
 file.open("Velocity_field.txt",std::ios_base::app);
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   // x-position
   file<<main[i][j].r[0]<<"␣";
   // y_position
   file<<main[i][j].r[1]<<"␣";
   // x-velocity
   if (i==0)
   {
    u=stgx[i][j].u;
   }
   else if (i==N+1)
   {
    u=stgx[i-1][j].u;
   }
   else
   {
    u=(stgx[i-1][j].u+stgx[i][j].u)/2;
   }
   // y-velocity
   if (j==0)
   {
    v=stgy[i][j].v;
   }
   else if (j==M+1)
   {
    v=stgy[i][j-1].v;
   }
   else
   {
    v=(stgy[i][j].v+stgy[i][j-1].v)/2;
   }
   file<<scale*u/sqrt(u*u+v*v)<<"␣"<<scale*v/sqrt(v*v+u*u)<<"␣"<<sqrt(u*u+v*v)<<
       ↪ endl;
  }

 }
 //gnuplot representation > plot 'Velocity_field.txt' with vectors filled lc
     ↪ palette
}
bool stationary (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 double maxP=0,max_vel=0,maxT=0;
 int im,jm;
 for (int i=0;i<N+2;i++)
 {
  for (int j=0;j<M+2;j++)
  {
   if (fabs(main[i][j].P-main[i][j].P_ant)>maxP)
   {
    maxP=fabs(main[i][j].P-main[i][j].P_ant)/main[i][j].P;
    im=i;
    jm=j;
   }
   if (fabs(main[i][j].sigma-main[i][j].sigma_ant)>maxT)
   {
    maxT=fabs(main[i][j].sigma-main[i][j].sigma_ant);
```

```
    }

   }
  }
  for (int i=1;i<N;i++)
  {
   for (int j=1;j<M+1;j++)
   {
    if (fabs(stgx[i][j].u-stgx[i][j].u_ant)>max_vel)
    {
     max_vel=fabs(stgx[i][j].u-stgx[i][j].u_ant);
    }
   }
  }
  for (int i=1;i<N+1;i++)
  {
   for (int j=1;j<M;j++)
   {
    if (fabs(stgy[i][j].v-stgy[i][j].v_ant)>max_vel)
    {
     max_vel=fabs(stgy[i][j].v-stgy[i][j].v_ant);
    }
   }
  }
  if (max_vel>delta_v || maxP>(delta_P) || maxT > delta_T)
  {
   if (t>n_it*t_show)
   {
    cout<<"Time instant: "<<t<<" s"<<endl;
    if (max_vel/delta_v > maxP/(delta_P) && max_vel/delta_v > maxT/(delta_T)  )
    {
     cout<<"Stationary status (vel): "<< delta_v/max_vel*100<<"%"<<endl;
    }
    else if (maxP/delta_P > maxT/(delta_T))
    {
     cout<<"Stationary status (P): "<< delta_P/maxP*100<<"%"<<endl;
    }
    else
    {
     cout<<"Stationary status (T): "<< delta_T/maxT*100<<"%"<<endl;
    }
    n_it++;
    cout<<"----------------------------------------"<<endl;
   }

   return false;
  }
  else
  {

   return true;
  }
}
void verif_NS1 (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
{
 double me,mw,ms,mn,maxm=0,maxx=0,maxy=0,err;
 double uP,uE,uW,uN,uS,vAn,vBn,vAs,vBs,vP,vE,vW,vN,vS,uAe,uBe,uAw,uBw,Aa,Ab;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
```

```
   mn=stgy[i][j].v*main[i][j].Ax;
   ms=stgy[i][j-1].v*main[i][j].Ax;
   me=stgx[i][j].u*main[i][j].Ay;
   mw=stgx[i-1][j].u*main[i][j].Ay;
   err=mw+ms-me-mn;
   if (fabs(err)>maxm)
   {
    maxm=fabs(err);
   }
  }
}
int cont;
cont=T_u[0][0];
int i,j;
for (int k=1;k<cont;k++)
{
 i=T_u[k][0];
 j=T_u[k][1];
 uP=stgx[i][j].u;
  uE=stgx[i+1][j].u;
  uW=stgx[i-1][j].u;
  uN=stgx[i][j+1].u;
  uS=stgx[i][j-1].u;
  vAn=stgy[i][j].v;
  vBn=stgy[i+1][j].v;
  vAs=stgy[i][j-1].v;
  vBs=stgy[i+1][j-1].v;
  Aa=stgx[i][j].r[0]-stgx[i][j].xw;
  Ab=stgx[i][j].xe-stgx[i][j].r[0];
  mn=(vAn*Aa+vBn*Ab);
  ms=(vAs*Aa+vBs*Ab);
  me=0.5*stgx[i][j].Ay*(uE+uP);
  mw=0.5*stgx[i][j].Ay*(uW+uP);
  err=mw+ms-me-mn;
  if (fabs(err)>maxx)
  {
   maxx=fabs(err);
  }

}
cont=T_v[0][0];
for (int k=1;k<cont;k++)
{
 i=T_v[k][0];
 j=T_v[k][1];
 vP=stgy[i][j].v;
  vE=stgy[i+1][j].v;
  vW=stgy[i-1][j].v;
  vN=stgy[i][j+1].v;
  vS=stgy[i][j-1].v;
  uAe=stgx[i][j].u;
  uBe=stgx[i][j+1].u;
  uAw=stgx[i-1][j].u;
  uBw=stgx[i-1][j+1].u;
  Aa=stgy[i][j].r[1]-stgy[i][j].ys;
  Ab=stgy[i][j].yn-stgy[i][j].r[1];
  mn=0.5*stgy[i][j].Ax*(vN+vP);
  ms=0.5*stgy[i][j].Ax*(vS+vP);
  me=(uAe*Aa+uBe*Ab);
  mw=(uAw*Aa+uBw*Ab);
  err=mw+ms-me-mn;
  if (fabs(err)>maxy)
```

```
    {
     maxy=fabs(err);
    }
  }
  cout<<"Main: "<<maxm<<endl;
  cout<<"Stagg-x: "<<maxx<<endl;
  cout<<"Stagg-y: "<<maxy<<endl;

}
void error_P(vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >& stgx,
    ↪ vector< vector<stg_y_mesh> >& stgy)
{
 double maxP=0,maxP2=0;
 double pP,pE,pW,pS,pN,Ax,Ay,uE,uW,vS,vN;
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   pP=main[i][j].P;
   pE=main[i+1][j].P;
   pW=main[i-1][j].P;
   pS=main[i][j-1].P;
   pN=main[i][j+1].P;
   Ax=main[i][j].Ax;
   Ay=main[i][j].Ay;
   uE=stgx[i][j].u_P;
   uW=stgx[i-1][j].u_P;
   vN=stgy[i][j].v_P;
   vS=stgy[i][j-1].v_P;
   maxP=max(maxP,fabs((pE-pP)/Ax*Ay-(pP-pW)/Ax*Ay+(pN-pP)/Ay*Ax-(pP-pS)/Ay*Ax-1/At
       ↪ *(uE*Ay-uW*Ay+vN*Ax-vS*Ax)));
   maxP2=max(maxP2,fabs(main[i][j].ap*pP-main[i][j].ae*pE-main[i][j].aw*pW-main[i
       ↪ ][j].an*pN-main[i][j].as*pS-main[i][j].bp));
  }
 }
 cout<<"Pressure errror discretization: "<<maxP<<"   "<<maxP2<<endl;
}
void save_Bench(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
 std::ofstream ofs;
 ofs.open("u_60x60_UDS_Re100.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("u_60x60_UDS_Re100.txt",std::ios_base::app);
 int i=(N+1)/2;
 for (int j=0;j<M+2;j++)
 {
  file<<stgx[i][j].r[1]<<" "<<stgx[i][j].u<<endl;
 }
 ofs.open("v_60x60_UDS_Re100.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 file.open("v_60x60_UDS_Re100.txt",std::ios_base::app);
 int j=(M+1)/2;
 for (i=0;i<N+2;i++)
 {
  file<<stgy[i][j].r[0]<<" "<<stgy[i][j].v<<endl;
 }
}
void save_u_v(vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
```

```
 std::ofstream ofs;
 ofs.open("u.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("u.txt",std::ios_base::app);
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+1;i++)
  {
   file<<stgx[i][j].u<<"\t";
  }
  file<<endl;
 }
 file.close();
 ofs.open("v.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 file.open("v.txt",std::ios_base::app);
 for (int j=0;j<M+1;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<stgy[i][j].v<<"\t";
  }
  file<<endl;
 }
 file.close();
}
void save_R (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >& stgy
    ↪ )
{
 std::ofstream ofs;
 ofs.open("R.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("R.txt",std::ios_base::app);
 file<<"Ru:␣"<<endl;
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+1;i++)
  {
   file<<stgx[i][j].R<<"\t";
  }
  file<<endl;
 }
 file<<"Rv:␣"<<endl;
 for (int j=0;j<M+1;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<stgy[i][j].R<<"\t";
  }
  file<<endl;
 }
 file.close();

}
void save_R2 (vector< vector<stg_x_mesh> >& stgx,vector< vector<stg_y_mesh> >&
    ↪ stgy)
{
 std::ofstream ofs;
 ofs.open("R2.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
```

```
 ofstream file;
 file.open("R2.txt",std::ios_base::app);
 file<<"Ru2:␣"<<endl;
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+1;i++)
  {
   file<<stgx[i][j].R2<<"\t";
  }
  file<<endl;
 }
 file<<"Rv2:␣"<<endl;
 for (int j=0;j<M+1;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<stgy[i][j].R2<<"\t";
  }
  file<<endl;
 }
 file.close();

}
void save_parameters ()
{
 std::ofstream ofs;
 ofs.open("Inputa_Data_time.txt", std::ofstream::out | std::ofstream::trunc);
 ofs.close();
 ofstream file;
 file.open("Inputa_Data_time.txt",std::ios_base::app);
 file<<"Re:␣"<<Re<<endl;
 file<<"Mesh␣density:␣"<<N+2<<"x"<<M+2<<endl;
 file<<"k␣factor:␣"<<k<<endl;
 file<<"delta_it:␣"<<delta<<endl;
 file<<"delta_v:␣"<<delta_v<<endl;
 file<<"delta_P:␣"<<delta_P<<endl;
 file<<"Scheme:␣";
 if (SCHEME==0)
 {
  file<<"UDS"<<endl;
 }
 else if (SCHEME==1)
 {
  file<<"CDS"<<endl;
 }
 else if (SCHEME==2)
 {
  file<<"SUDS"<<endl;
 }
 else if (SCHEME==3)
 {
   file<<"QUICK"<<endl;
 }
 else if (SCHEME==4)
 {
   file<<"SMART"<<endl;
 }
 time(&final);
 file<<"Computation␣time:␣"<<difftime(final,inici)<<"␣s"<<endl;
}
void compute_Rt (vector< vector<main_mesh> >& main,vector< vector<stg_x_mesh> >&
    ↪ stgx,vector< vector<stg_y_mesh> >& stgy)
```

```
{
 double ue,uw,vn,vs,Ax,Ay,Fe,Fw,Fn,Fs,oe,ow,on,os,oE,oW,oN,oS,oP,xw,xe,yn,ys,dpe,
     ↪ dpw,dpn,dps; // o represent the greek letter sigma
 for (int i=1;i<N+1;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   Ax=main[i][j].Ax;
   Ay=main[i][j].Ay;
   oP=main[i][j].sigma_ant;
   oE=main[i+1][j].sigma_ant;
   oW=main[i-1][j].sigma_ant;
   oN=main[i][j+1].sigma_ant;
   oS=main[i][j-1].sigma_ant;
   xw=stgx[i-1][j].r[0];
   xe=stgx[i][j].r[0];
   yn=stgy[i][j].r[1];
   ys=stgy[i][j-1].r[1];
   dpe=main[i+1][j].r[0]-main[i][j].r[0];
   dpw=main[i][j].r[0]-main[i-1][j].r[0];
   dpn=main[i][j+1].r[1]-main[i][j].r[1];
   dps=main[i][j].r[1]-main[i][j-1].r[1];
   ue=stgx[i][j].u;
   uw=stgx[i-1][j].u;
   vn=stgy[i][j].v;
   vs=stgy[i][j-1].v;
   Fe=ue*Ay;
   Fw=uw*Ay;
   Fn=vn*Ax;
   Fs=vs*Ax;
   if (i==1)
   {
    ow=UDS(oW,oP,Fw);
   }
   else
   {
    ow=phi_face_neg(Fw,xw,main[i][j].r[0],oP,main[i-1][j].r[0],oW,main[i+1][j].r
        ↪ [0],oE,main[i-2][j].r[0],main[i-2][j].sigma_ant);
   }
   if (i==N)
   {
    oe=UDS(oP,oE,Fe);
   }
   else
   {
    oe=phi_face_pos(Fe,xe,main[i][j].r[0],oP,main[i+1][j].r[0],oE,main[i-1][j].r
        ↪ [0],oW,main[i+2][j].r[0],main[i+2][j].sigma_ant);
   }
   if (j==1)
   {
    os=UDS(oS,oP,Fs);
   }
   else
   {
    os=phi_face_neg(Fs,ys,main[i][j].r[1],oP,main[i][j-1].r[1],oS,main[i][j+1].r
        ↪ [1],oN,main[i][j-2].r[1],main[i][j-2].sigma_ant);
   }
   if (j==M)
   {
    on=UDS(oP,oN,Fn);
   }
   else
```

```
   {
    on=phi_face_pos(Fn,yn,main[i][j].r[1],oP,main[i][j+1].r[1],oN,main[i][j-1].r
        ↪ [1],oS,main[i][j+2].r[1],main[i][j+2].sigma_ant);
   }
   main[i][j].Rt=1/(Ax*Ay)*(-(oe*Fe-ow*Fw+on*Fn-os*Fs)+1/(Re*Pr)*((oE-oP)/dpe*Ay-(
       ↪ oP-oW)*Ay/dpw+(oN-oP)*Ax/dpn-(oP-oS)*Ax/dps));
  }
 }
 for (int i=0;i<N+2;i++)
 {
  main[i][0].Rt=0;
  main[i][M+1].Rt=0;
 }
 for (int j=1;j<M+1;j++)
 {
  main[0][j].Rt=0;
  main[N+1][j].Rt=0;
 }
}
void compute_T(vector< vector<main_mesh> >& main)
{
 for (int i=0;i<N+2;i++)
 {
  for (int j=1;j<M+1;j++)
  {
   main[i][j].sigma=main[i][j].sigma_ant+At*(1.5*main[i][j].Rt-0.5*main[i][j].Rt2)
       ↪ ;
  }
  main[i][0].sigma=main[i][1].sigma;
  main[i][M+1].sigma=main[i][M].sigma;
 }
}
void save_T(vector< vector<main_mesh> >& main)
{
 ofstream file;
 file.open("Temperature");
 for (int j=0;j<M+2;j++)
 {
  for (int i=0;i<N+2;i++)
  {
   file<<main[i][j].sigma<<"\t";
  }
  file<<endl;
 }
 file.close();
}
```