UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Estudi de la resolució computacional de les equacions de Navier-Stokes aplicades a fluxos laminars i introducció a la turbulència

Universitat Politècnica de Catalunya

Annexos

ESEIAAT

Treball de Fi de Grau
Grau en Enginyeria Mecànica

Autor: Oriol Sanmarti Perona
Director: Carlos David Perez Segarra
Co-director: Jordi Vera Fernández
22 de juny de 2021

Índex

1. 4 Materials

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
using namespace std;

const double q = 60;              //W/m
const double cp1 = 750;           //J/(KgK)
const double cp2 = 770;
const double cp3 = 810;
const double cp4 = 930;
const double densitat1 = 1500;    //Kg/m^3
const double densitat2 = 1600;
const double densitat3 = 1900;
const double densitat4 = 2500;
const double lambda1 = 170;       //W/(mK)
const double lambda2 = 140;
const double lambda3 = 200;
const double lambda4 = 140;
const double lambda42 = 2/((1/lambda2)+(1/lambda4));
const double lambda13 = 2/((1/lambda1)+(1/lambda3));
const double lambda34 = 2/((1/lambda3)+(1/lambda4));
const double lambda32 = 2/((1/lambda3)+(1/lambda2));
const double lambda12 = 2/((1/lambda1)+(1/lambda2));
const double L = 1.1;
const double H = 0.8;
const int Nx = 55;                //NODES x
const int Ny = 40;                //NODES Y
const double epsilon = 0.000001;  //ERROR
const double inct = 1;            //CAMBIAR INCREMENT DE TEMPS
const double tempsmaxim = 5000;   //CAMBIAR TEMPS
const double temps = tempsmaxim/inct;  //NO CAMBIAR
const double incx = L/(Nx);
const double incy = H/(Ny);
const double gamma1 = (densitat1*cp1*incx*incy)/(inct);   //450
const double gamma2 = (densitat2*cp2*incx*incy)/(inct);   //492.8
const double gamma3 = (densitat3*cp3*incx*incy)/(inct);   //615.6
const double gamma4 = (densitat4*cp4*incx*incy)/(inct);   //930
const double alpha = 9;   //W/m^2K
const double Tf = 33;     //C
const double archius = 500;
const int iteprint = tempsmaxim/archius;

double CalculTiM2(double TE, double TW, double TN, double TS, double Tp){
   double r;
   double ae = (lambda2*incy)/(gamma2*incx);
   double aw = (lambda2*incy)/(gamma2*incx);
   double an = (lambda2*incx)/(gamma2*incy);
   double as = (lambda2*incx)/(gamma2*incy);
   double b  = Tp;
   double ap = 1 + ae + aw + an + as;
   r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
   return r;
}
double CalculTiM1(double TE, double TW, double TN, double TS, double Tp){
   double r;
   double ae = (lambda1*incy)/(gamma1*incx);
   double aw = (lambda1*incy)/(gamma1*incx);
   double an = (lambda1*incx)/(gamma1*incy);
   double as = (lambda1*incx)/(gamma1*incy);
   double b  = Tp;
   double ap = 1 + ae + aw + an + as;
```

```
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM3(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda3*incy)/(gamma3*incx);
    double aw = (lambda3*incy)/(gamma3*incx);
    double an = (lambda3*incx)/(gamma3*incy);
    double as = (lambda3*incx)/(gamma3*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM4(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda4*incy)/(gamma4*incx);
    double aw = (lambda4*incy)/(gamma4*incx);
    double an = (lambda4*incx)/(gamma4*incy);
    double as = (lambda4*incx)/(gamma4*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM24(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda2*incy)/(((gamma2))*incx);
    double aw = (lambda2*incy)/(((gamma2))*incx);
    double an = (lambda42*incx)/(((gamma2))*incy);
    double as = (lambda2*incx)/(((gamma2))*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM42(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda4*incy)/(((gamma4))*incx);
    double aw = (lambda4*incy)/(((gamma4))*incx);
    double an = (lambda4*incx)/(((gamma4))*incy);
    double as = (lambda42*incx)/(((gamma4))*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM13(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda1*incy)/((gamma1)*incx);
    double aw = (lambda1*incy)/((gamma1)*incx);
    double an = (lambda13*incx)/((gamma1)*incy);
    double as = (lambda1*incx)/((gamma1)*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM31(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda3*incy)/(((gamma3))*incx);
    double aw = (lambda3*incy)/(((gamma3))*incx);
    double an = (lambda3*incx)/(((gamma3))*incy);
    double as = (lambda13*incx)/(((gamma3))*incy);
```

```
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM34(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda34*incy)/(((gamma3))*incx);
    double aw = (lambda3*incy)/(((gamma3))*incx);
    double an = (lambda3*incx)/(((gamma3))*incy);
    double as = (lambda3*incx)/(((gamma3))*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM43(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda4*incy)/(((gamma4))*incx);
    double aw = (lambda34*incy)/(((gamma4))*incx);
    double an = (lambda4*incx)/(((gamma4))*incy);
    double as = (lambda4*incx)/(((gamma4))*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM32(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda32*incy)/(((gamma3))*incx);
    double aw = (lambda3*incy)/(((gamma3))*incx);
    double an = (lambda3*incx)/(((gamma3))*incy);
    double as = (lambda3*incx)/(((gamma3))*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM23(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda2*incy)/(((gamma2))*incx);
    double aw = (lambda32*incy)/(((gamma2))*incx);
    double an = (lambda2*incx)/(((gamma2))*incy);
    double as = (lambda2*incx)/(((gamma2))*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM12(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda12*incy)/(((gamma1))*incx);
    double aw = (lambda1*incy)/(((gamma1))*incx);
    double an = (lambda1*incx)/(((gamma1))*incy);
    double as = (lambda1*incx)/(((gamma1))*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculTiM21(double TE, double TW, double TN, double TS, double Tp){
    double r;
    double ae = (lambda2*incy)/(((gamma2))*incx);
    double aw = (lambda12*incy)/(((gamma2))*incx);
```

```
    double an = (lambda2*incx)/(((gamma2))*incy);
    double as = (lambda2*incx)/(((gamma2))*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + an + as;
    r = ((ae*TE+aw*TW+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculNeumannesquerra(double TE, double TW,double TS, double Tp){
    double r;
    double ae = (lambda3*incy)/(gamma3*incx);
    double aw = (lambda3*incy)/(gamma3*incx);
    double as = (lambda3*incx)/(gamma3*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + as;
    r = ((ae*TE+aw*TW+((q*incx)/gamma3)+as*TS)+b)/(ap);
    return r;
}
double CalculNeumanndreta(double TE, double TW,double TS, double Tp){
    double r;
    double ae = (lambda4*incy)/(gamma4*incx);
    double aw = (lambda4*incy)/(gamma4*incx);
    double as = (lambda4*incx)/(gamma4*incy);
    double b  = Tp;
    double ap = 1 + ae + aw + as;
    r = ((ae*TE+aw*TW+((q*incx)/gamma4)+as*TS)+b)/(ap);
    return r;
}
double CalculConvectionabaix(double TN, double TE,double TS, double Tp){
    double r;
    double an = (lambda1*incx)/(gamma1*incy);
    double ae = (lambda1*incy)/(gamma1*incx);
    double as = (lambda1*incx)/(gamma1*incy);
    double b  = Tp + ((Tf*alpha*incy)/gamma1);
    double ap = 1  + ae + as + an +((alpha*incy)/gamma1);
    r = ((ae*TE+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculConvectionadalt(double TN, double TE,double TS, double Tp){
    double r;
    double an = (lambda3*incx)/(gamma3*incy);
    double ae = (lambda3*incy)/(gamma3*incx);
    double as = (lambda3*incx)/(gamma3*incy);
    double b  = Tp + ((Tf*alpha*incy)/gamma3);
    double ap = 1  + ae + as + an +((alpha*incy)/gamma3);
    r = ((ae*TE+an*TN+as*TS)+b)/(ap);
    return r;
}
double CalculAdaltEsquerra(double TE, double TS, double Tp){
    double r;
    double ae = (lambda3*incy)/(gamma3*incx);
    double as = (lambda3*incx)/(gamma3*incy);
    double aw = (alpha*incy)/gamma3;
    double b  = Tp + Tf*aw + (q*incx)/gamma3;
    double ap = 1  + ae + as + aw;
    r = ((ae*TE+as*TS)+b)/(ap);
    return r;
}
double CalculAbaixDreta(double TW, double TN, double Tp){
    double r;
    double aw = (lambda2*incy)/(gamma2*incx);
    double ae = (lambda2*incy)/(gamma2*(incx/2));
    double an = (lambda2*incx)/(gamma2*incy);
    double as = (lambda2*(incx/2))/(gamma2*(incy/2));
    double b  = Tp;
```

```
    double ap = 1 + aw + an + as +ae;
    r = ((aw*TW+an*TN+as*23+ae*23)+b)/(ap);
    return r;
}
double CalculAbaixEsquerra( double TN, double Tp){
    double r;
    double ae = (lambda1*incy)/(gamma1*incx);
    double an = (lambda1*incx)/(gamma1*incy);
    double as = (lambda1*(incx))/+(gamma1*(incy/2));
    double b  = Tp + (alpha*incy*Tf)/gamma1;
    double ap = 1 + ae + an + as + (alpha*incy)/gamma1;
    r = ((ae*23 + an*TN + as*23 )+b)/(ap);
    return r;
}
double CalculAdaltDreta(double TE, double TS, double TW, double Tp){
    double r;
    double ae = (lambda3*incy)/(gamma3*(incx/2));
    double as = (lambda3*incx)/(gamma3*incy);
    double aw = (lambda3*incy)/(gamma3*incx);
    double b  = Tp + (q*incy)/gamma3;
    double ap = 1  + ae + as + aw + (alpha*(incy))/gamma3;
    r = ((ae*TE+as*TS+aw*TW)+b)/(ap);
    return r;
}

int main(){

    double T[Nx][Ny];
    double Tant[Nx][Ny];
    double Tguarda[Nx][Ny];
    double x = 0;
    double y = 0;

    for(int j = 0; j<Ny; j++){
        for(int i = 0; i<Nx; i++){
            T[i][j] = 8;
            Tguarda[i][j]=20;
            Tant[i][j] = 8;
            if(j==Ny-1){
                T[i][j] = 23;
            }
        }
    }

    ofstream Archivo;
    char nombre[500];
    Archivo.open("Temp_0.txt");
    Archivo << "t" << "\t" << "T" << "\t" << "x" << "\t" << "y" << "\t" << "z" "\n";
    for(int j = 0; j<Ny; j++){
        for(int i = 0; i<Nx; i++){
                x = incx*i + incx/2;
                y = incy*j + incy/2;
        Archivo << 0 << "\t" << T[i][j] << "\t" << x << "\t" << y << "\t" << 0<< "\n";
        }
    }

    Archivo.close();
    ofstream Archivo2;
    Archivo2.open("Apartat_b.txt");
    Archivo2 << "t" <<"\t"<< "T(0.65,0.56)" <<"\t"<< "T(0.74,0.72)"<<"\n";
    Archivo2 << 0 <<"\t"<< 8 <<"\t"<< 8 <<"\n";


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

```
bool itefin = false;
while(!itefin){
itefin = true;
for(int j = 0; j<Ny; j++){
for(int i = 0; i<Nx; i++){
   x = incx*i + incx/2;
   y = incy*j + incy/2;
   //PARET DRETA:
   if(x==L-(incx/2) and y < H - (incy/2) and y > incy/2){
      Tguarda[i][j] = T[i][j];
      T[i][j] = 8 + (0.005 * (inct * k));
   }
   //PARET ABAIX:
   if(y == H - (incy/2) and x < L-(incx/2) and x > incx/2){
      Tguarda[i][j] = T[i][j];
      T[i][j] = 23;
   }
   //PARET ADALT Neumann
   if(y == incy/2 and x > incx/2 and x <= 0.5){
      Tguarda[i][j] = T[i][j];
      T[i][j] = CalculNeumannesquerra(T[i+1][j], T[i-1][j],T[i][j+1], Tant[i][j]);

   }
   if(y == incy/2 and x > 0.5 and x < L-(incx/2)){
      Tguarda[i][j] = T[i][j];
      T[i][j] = CalculNeumanndreta(T[i+1][j], T[i-1][j],T[i][j+1], Tant[i][j]);

   }
   //PARET ESQUERRA CONVECCIO

   if(x == incx/2 and y >= 0.4 and y < H-(incy/2)){
      Tguarda[i][j] = T[i][j];
      T[i][j] = CalculConvectionabaix(T[i][j-1], T[i+1][j], T[i][j+1], Tant[i][j]);
   }

   if(x == incx/2 and y > incy/2 and y < 0.4){
      Tguarda[i][j] = T[i][j];
      T[i][j] = CalculConvectionadalt(T[i][j-1], T[i+1][j], T[i][j+1], Tant[i][j]);
   }

   //M2
   if(x>=0.5 and x < L-(incx/2) and y >= 0.1 and y < H-(incy/2)){
      //M21
      if(x-incx < 0.5 and y > 0.4){
      Tguarda[i][j] = T[i][j];
      T[i][j] = CalculTiM21(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
      }
      //M24
      else if (y-incy < 0.1){
      Tguarda[i][j] = T[i][j];
      T[i][j] = CalculTiM24(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
      }
      //M23
      else if(x-incx < 0.5 and y < 0.4){
      Tguarda[i][j] = T[i][j];
      T[i][j] = CalculTiM23(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
      }
      else {
      Tguarda[i][j] = T[i][j];
      T[i][j] = CalculTiM2(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
      }
   }

   //M1
```

```
if(x > incx/2 and x < 0.5 and y >= 0.4 and y < H-(incy/2)){
  //M13
  if(y-incy < 0.4){
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM13(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
  }
  //M12
  else if(x+incx > 0.5){
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM12(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);

  }
  else{
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM1(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
}
}
//M3
if(x > incx/2 and x < 0.5 and y > incy/2 and y < 0.4){
  //M32
  if(x+incx > 0.5 and y > 0.1 and y <= 0.4){
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM32(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
  }
  //M31
  else if(x > incx/2 and x < 0.5 and y+incy > 0.4){
    Tguarda[i][j] = T[i][j];
    T[i][j] = CalculTiM31(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
  }
  //M34
  else if(x+incx > 0.5 and y > incy/2 and y < 0.1){
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM34(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
  }
  else{
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM3(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
}
}
//M4
if(x >= 0.5 and x < L-(incx/2) and y > incy/2 and y < 0.1){
  //M43
  if(x-incx < 0.5){
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM43(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
  }
  //M42

  else if(y+incy > 0.1 ){
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM42(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
  }

  else {
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculTiM4(T[i+1][j], T[i-1][j], T[i][j-1], T[i][j+1], Tant[i][j]);
}
}
//NODE ADALT ESQUERRA

if(x == incx/2 and y == incy/2 ){
  Tguarda[i][j] = T[i][j];
  T[i][j] = CalculAdaltEsquerra(T[i+1][j],T[i][j+1], Tant[i][j]);
}
```

```
        //NODE ABAIX DRETA
        if(x == L-(incx/2) and y == H-(incy/2)){
            Tguarda[i][j] = T[i][j];
            T[i][j] = CalculAbaixDreta(T[i-1][j], T[i][j-1], Tant[i][j]);

        }
        //NODE ABAIX ESQUERRA
        if(x == incx/2 and y == H-(incy/2)){
            Tguarda[i][j] = T[i][j];
            T[i][j] = CalculAbaixEsquerra( T[i][j-1], Tant[i][j]);

        }
        //NODE ADALT DRETA
        if(x == L-(incx/2) and y == incy/2){
            Tguarda[i][j] = T[i][j];
            T[i][j] = CalculAdaltDreta(T[i][j+1], T[i][j+1], T[i-1][j], Tant[i][j]);
        }
        }
    }
    for(int j = 1; j<Ny-1; j++){
    for(int i = 1; i<Nx-1; i++){
    double error = 0;
    error = abs(Tguarda[i][j] - T[i][j]);
    if(error > epsilon) {itefin = false;
        //cout<<"la i es: " << i << " la j es : "<<j<<" La t guarda es: "<<Tguarda[i][j]<<" La T es : " <<T[i][j]<<" El
     temps es : "<<k*inct<<endl;
        //cout<<error<<endl;
        }

    //cout<<"L'error es: "<< itefin<<endl;
    }
}
//cout<<endl;
    }
    //cout<<inct*k<<endl;
    int num = k;
    if(k%iteprint == 0){
    sprintf(nombre, "Temp_%05d.txt", num);
    Archivo.open(nombre, ios::trunc);
    Archivo << "t" << "\t" << "T" << "\t" << "x" << "\t" << "y" << "\t" << "z" "\n";
    }

    for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){
        Tant[i][j] = T[i][j];
        if(k%iteprint == 0){

                x = incx*i + incx/2;
                y = incy*j + incy/2;
                Archivo << k*inct << "\t" << T[i][j] << "\t" << x << "\t" << y << "\t" << 0<< "\n";


        }
    }
}
x = 0;
y = 0;
Archivo.close();

//APARTAT B
int punt1 = 0.64/incx;
int punt1prima = 0.66/incx;
int punt2 = 0.44/incy;
int punt3 = 0.74/incx;
```

```
    int punt4 = 0.28/incy;
    double Tpunt1 = ((T[32][12])+(T[32][11]))/2;
    double Tpunt2 = ((T[37][4])+(T[36][4])+(T[37][3])+(T[36][3]))/4;
    Archivo2 << k*inct <<"\t"<< Tpunt1 <<"\t"<< Tpunt2 <<"\n";
}
    Archivo2.close();
    for(int j = 0; j<Ny; j++){
        for(int i = 0; i<Nx; i++){
            cout<<T[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

2. Smith Hutton CDS

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
using namespace std;

double const L = 2;
double const H = 1;
int const Nx = 100;
int const Ny = 50;
double const inct = 0.001;
double const densitat = 1000000;
double const gamma = 1;
double const incx = L/Nx;
double const incy = H/Ny;
//double const temps = 200;
double const alpha = 10;
double const epsilon = 0.0001;

double Calculvelu(double x, double y){
    double u = 2*y*(1-pow(x,2));
    return u;
}
double Calculvelv(double x, double y){
    double v = -2*x*(1-pow(y,2));
    return v;
}
double Calculinlet(double PHI, double x){
    double r = 1 + tanh((2*x+1)*alpha);
    return r;
}

int main(){

  double PHI[Nx][Ny];
  double PHIant[Nx][Ny];
  double u[Nx][Ny];
  double v[Nx][Ny];
  double xi = -(L/2) + (incx/2);
  double yi = H - (incy/2);
  double x;
  double y;
  ofstream Archivo1;
  ofstream Archivo2;
  ofstream Archivo3;
```

```
Archivo1.open("Mapa de velocitats.txt");
Archivo1<<"x"<<"\t"<<"y"<<"\t"<<"z"<<"\t"<<"u"<<"\t"<<"v"<<"\n";

Archivo2.open("Mapa Propietat.txt");
Archivo2<<"x"<<"\t"<<"y"<<"\t"<<"z"<<"\t"<<"PHI"<<"\n";

Archivo3.open("Outlet propietat.txt");
Archivo3<<"x"<<"\t"<<"PHI"<<"\n";

//DECLAREM MAPA DE VELOCITATS
for(int j = 0; j<Ny; j++){
  for(int i = 0; i<Nx; i++){
     x = xi + i*incx;
     y = yi - j*incy;
     u[i][j] = Calculvelu(x,y);
     v[i][j] = Calculvelv(x,y);
     PHI[i][j] = 0;
     PHIant[i][j] = PHI[i][j];
     //cout<<" ("<<Vu[i][j]<<","<<Vv[i][j]<<") ";
     cout<<PHI[i][j]<<" ";
     Archivo1<<x<<"\t"<<y<<"\t"<<0<<"\t"<<u[i][j]<<"\t"<<v[i][j]<<"\n";
  }
  cout<<endl;
}
Archivo1.close();

//    double tempsmaxim = temps/inct;

   bool conv = false;
   int k = 0;
   while(!conv){
      conv = true;
      for(int j = 0; j<Ny; j++){
         for(int i = 0; i<Nx; i++){
            x = xi + i*incx;
            y = yi - j*incy;
         //CALCUL NODES CENTRALS
         if(x > xi and y < yi and x < -xi and j<Ny-1){
            PHIant[i][j] = PHI[i][j];
            double uw = u[i-1][j];
            double ue = u[i+1][j];
            double vn = v[i][j-1];
            double vs = v[i][j+1];
            double ae = (gamma*incy)/incx;
            double aw = (gamma*incy)/incx;
            double an = (gamma*incx)/incy;
            double as = (gamma*incx)/incy;
            double PHIp = PHI[i][j];
            double PHIe = (PHI[i+1][j] + PHIp)/2;
            double PHIw = (PHI[i-1][j] + PHIp)/2;
            double PHIn = (PHI[i][j-1] + PHIp)/2;
            double PHIs = (PHI[i][j+1] + PHIp)/2;
            double PHIE2 = PHI[i+1][j];
            double PHIW2 = PHI[i-1][j];
            double PHIN2 = PHI[i][j-1];
            double PHIS2 = PHI[i][j+1];
            double conveccio = densitat*(PHIe*incy*ue - PHIw*incy*uw + PHIn*incx*vn - PHIs*incx*vs);
            double diffusio = ae*PHIE2 + aw*PHIW2 + an*PHIN2 + as*PHIS2 - PHIp*(an+aw+as+ae);
            PHI[i][j] = (inct/(incx*incy*densitat)) * (-conveccio + diffusio) + PHIp;

            if(abs(PHIant[i][j]-PHI[i][j]) > epsilon){
               conv = false;
               //cout<<PHIant[i][j]<<"--"<<PHI[i][j]<<endl;
```

```
            }
        }
        //INLET
        else if(j == Ny-1 and x < 0 and x > xi){
            PHIant[i][j] = PHI[i][j];
            PHI[i][j] = Calculinlet(PHI[i][j], x);
        }
        //OUTLET
        else if(j == Ny-1 and x > 0 and x < -xi){
            PHIant[i][j] = PHI[i][j];
            PHI[i][j] = PHI[i][j-1];
            if(abs(PHIant[i][j]-PHI[i][j]) > epsilon){
                conv = false;
                //cout<<PHIant[i][j]<<"--"<<PHI[i][j]<<endl;
            }
        }
        //REST OF WALLS
        else if(abs(x-xi)<0.0001 or abs(x+xi)<0.0001 or abs(y-yi)<0.0001){
            PHIant[i][j] = PHI[i][j];
            PHI[i][j] = 1-tanh(10);
        }
        }
    }

  k++;
  if(k < 15000){
     conv = false;
  }
  }
  cout<<"------------------------------------------------------------"<<endl;
 for(int j = 0; j<Ny; j++){
        for(int i = 0; i<Nx; i++){
     x = xi + i*incx;
     y = yi - j*incy;
     cout<<PHI[i][j]<<" ";
     Archivo2<<x<<"\t"<<y<<"\t"<<0<<"\t"<<PHI[i][j]<<"\n";
     if(j==Ny-1){
        Archivo3<<x<<"\t"<<PHI[i][j]<<"\n";
     }
  }
  cout<<endl;
 }
 Archivo2.close();
 Archivo3.close();
}
```

3. Smith Hutton UDS

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
using namespace std;

double const L = 2;
double const H = 1;
int const Nx = 100;
int const Ny = 50;
double const inct = 0.0001;
double const densitat = 1000000;
double const gamma = 1;
```

```
double const incx = L/Nx;
double const incy = H/Ny;
//double const temps = 200;
double const alpha = 10;
double const epsilon = 0.000001;


double Calculvelu(double x, double y){
   double u = 2*y*(1-pow(x,2));
   return u;
}
double Calculvelv(double x, double y){
   double v = -2*x*(1-pow(y,2));
   return v;
}
double Calculinlet(double PHI, double x){
   double r = 1 + tanh((2*x+1)*alpha);
   return r;
}
double CalculUDS(double PHIee, double PHIww, double PHInn, double PHIss, double PHI, double uw, double ue, double vn
      , double vs){
   double ae = (gamma*incy)/incx;
   double aw = (gamma*incy)/incx;
   double an = (gamma*incx)/incy;
   double as = (gamma*incx)/incy;
   double PHIe, PHIw, PHIn, PHIs;
   if(ue > 0){
      PHIe = PHI;
   }else if( ue <= 0 ){
      PHIe = PHIee;
   }
   if(uw > 0){
      PHIw = PHIww;
   }else if( uw <= 0 ){
      PHIw = PHI;
   }
   if(vn > 0){
      PHIn = PHI;
   }else if( vn <= 0 ){
      PHIn = PHInn;
   }
   if(vs > 0){
      PHIs = PHIss;
   }else if( vs <= 0 ){
      PHIs = PHI;
   }

   double conveccio = densitat*(PHIe*incy*ue - PHIw*incy*uw + PHIn*incx*vn - PHIs*incx*vs);
   double diffusio = ae*PHIee + aw*PHIww + an*PHInn + as*PHIss - PHI*(an+aw+as+ae);
   double r = (inct/(incx*incy*densitat)) * (-conveccio + diffusio) + PHI;
   return r;

}

int main(){

  double PHI[Nx][Ny];
  double PHIant[Nx][Ny];
  double Vu[Nx][Ny];
  double Vv[Nx][Ny];
  double xi = -(L/2) + (incx/2);
  double yi = H - (incy/2);
  double x;
  double y;
```

```
double v;
double u;
ofstream Archivo1;
ofstream Archivo2;
ofstream Archivo3;
Archivo1.open("Mapa de velocitats.txt");
Archivo1<<"x"<<"\t"<<"y"<<"\t"<<"z"<<"\t"<<"u"<<"\t"<<"v"<<"\n";

Archivo2.open("Mapa Propietat.txt");
Archivo2<<"x"<<"\t"<<"y"<<"\t"<<"z"<<"\t"<<"PHI"<<"\n";

Archivo3.open("Outlet propietat.txt");
Archivo3<<"x"<<"\t"<<"PHI"<<"\n";

//DECLAREM MAPA DE VELOCITATS
for(int j = 0; j<Ny; j++){
 for(int i = 0; i<Nx; i++){
    x = xi + i*incx;
    y = yi - j*incy;
    u = Calculvelu(x,y);
    v = Calculvelv(x,y);
    Vu[i][j] = u;
    Vv[i][j] = v;
    PHI[i][j] = 0;
    PHIant[i][j] = PHI[i][j];
    //cout<<" ("<<Vu[i][j]<<","<<Vv[i][j]<<") ";
    cout<<PHI[i][j]<<" ";
    Archivo1<<x<<"\t"<<y<<"\t"<<0<<"\t"<<Vu[i][j]<<"\t"<<Vv[i][j]<<"\n";
 }
 cout<<endl;
}
Archivo1.close();

//   double tempsmaxim = temps/inct;

  bool conv = false;
  int k = 0;
  while(!conv){


    conv = true;

    for(int j = 0; j<Ny; j++){
      for(int i = 0; i<Nx; i++){
        x = xi + i*incx;
        y = yi - j*incy;
      //CALCUL NODES CENTRALS
      if(x > xi and y < yi and x < -xi and j<Ny-1){
        PHIant[i][j] = PHI[i][j];
        PHI[i][j] = CalculUDS(PHI[i+1][j], PHI[i-1][j], PHI[i][j-1], PHI[i][j+1], PHI[i][j], Vu[i-1][j], Vu[i+1][j], Vv[i][j
    -1], Vv[i][j+1]);
        if(abs(PHIant[i][j]-PHI[i][j]) > epsilon){
          conv = false;
          //cout<<PHIant[i][j]<<"--"<<PHI[i][j]<<endl;
        }
      }
      //INLET
      else if(j == Ny-1 and x < 0 and x > xi){
        PHIant[i][j] = PHI[i][j];
        PHI[i][j] = Calculinlet(PHI[i][j], x);
      }
      //OUTLET
      else if(j == Ny-1 and x > 0 and x < -xi){
        PHIant[i][j] = PHI[i][j];
```

```
            PHI[i][j] = PHI[i][j-1];
            if(abs(PHIant[i][j]-PHI[i][j]) > epsilon){
               conv = false;
               //cout<<PHIant[i][j]<<"--"<<PHI[i][j]<<endl;
            }
         }
         //REST OF WALLS
         else if(abs(x-xi)<0.0001 or abs(x+xi)<0.0001 or abs(y-yi)<0.0001){
            PHIant[i][j] = PHI[i][j];
            PHI[i][j] = 1-tanh(10);
         }

         }
      }


   k++;
   if(k < 150000){
      conv = false;
   }

   }




   cout<<"------------------------------------------------------------"<<endl;
  for(int j = 0; j<Ny; j++){    for(int i = 0; i<Nx; i++){
      x = xi + i*incx;
      y = yi - j*incy;
      cout<<PHI[i][j]<<" ";
      Archivo2<<x<<"\t"<<y<<"\t"<<0<<"\t"<<PHI[i][j]<<"\n";
       if(j==Ny-1){
          Archivo3<<x<<"\t"<<PHI[i][j]<<"\n";
       }

   }
   cout<<endl;
  }
  Archivo2.close();
  Archivo3.close();


}
```

4. Lid Driven Cavity

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
#include <ctime>
using namespace std;

int main(){
unsigned t0, t1;

t0=clock();
double L = 1;
int Ny = 128;
int Nx = 128;
```

```
double incx = L/Nx;
double incy = L/Ny;
double Uref = 1;
double temps = 45;
double inct = 0.001;
double tempsmaxim = temps/inct;
double densitat = 100;
double nyu = 1;
double Re = (densitat*Uref*L)/(nyu);
double Ru = 0;
double Rv = 0;
double itefin = false;
double epsilon = 0.00001;

double Vpu[Nx+1][Ny];
double Vu[Nx+1][Ny];
double Vpv[Nx][Ny+1];
double Vv[Nx][Ny+1];
double P[Nx][Ny];
double Ruant[Nx+1][Ny];
double Rvant[Nx][Ny+1];

//INTRODUIM MALLA Vpu i Vu
for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx+1; i++){
      if(j == Ny-1){
         Vpu[i][j] = Uref;
         Vu[i][j] = Uref;
      }
      else{
         Vpu[i][j] = 0;
         Vu[i][j] = 0;
      }
      Ruant[i][j] = 0;
   }
}
//INTRODUIM MALLA Vpv i Vv
for(int j = 0; j<Ny+1; j++){
   for(int i = 0; i<Nx; i++){
      Vpv[i][j] = 0;
      Vv[i][j] = 0;
      Rvant[i][j] = 0;
   }
}
//INTRODUIM MALLA P
for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx; i++){
      P[i][j] = 0;
   }
}

for(int k = 0; k<tempsmaxim; k++){

   if(k%1000==0){
      cout<<k/1000<<endl;
   }

   //*****************************************************************************
   //                                STEP 1
   //*****************************************************************************
   //Calcul Velocitat predictoria u

   for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx+1; i++){
```

```
      if(i>0 and i <Nx and j>0 and j<Ny-1){
         double Vup = Vu[i][j];
         double Vvn = 0.5*(Vv[i-1][j+1]+Vv[i][j+1]);
         double Vvs = 0.5*(Vv[i-1][j]+Vv[i][j]);
         double Vue = 0.5*(Vu[i+1][j]+Vu[i][j]);
         double Vuw = 0.5*(Vu[i-1][j]+Vu[i][j]);
         double Vun = 0.5*(Vu[i][j+1]+Vu[i][j]);
         double Vus = 0.5*(Vu[i][j-1]+Vu[i][j]);
         double Un = Vu[i][j+1];
         double Us = Vu[i][j-1];
         double Ue = Vu[i+1][j];
         double Uw = Vu[i-1][j];
         //Calcul up
         Ru = -((densitat*incx)*(Vvn*Vun-Vvs*Vus+Vue*Vue-Vuw*Vuw)) + nyu*(Un+Us+Ue+Uw-4*Vup);
         //Vpu[i][j] = Vu[i][j] + ((inct/(densitat*incx*incx))*(1.5*Ru-0.5*Ruant[i][j]));
         Vpu[i][j] = Vu[i][j] + ((inct/(densitat*incx*incx))*(1*Ru));
         Ruant[i][j] = Ru;
      }
      else if(j == Ny-1){
         Vpu[i][j] = Uref;
      }
      else {
         Vpu[i][j] = 0;
      }
   }
}
   //Calcul velocitat predictoria v

   for(int j = 0; j<Ny+1; j++){
   for(int i = 0; i<Nx; i++){
      if(i>0 and i <Nx-1 and j>0 and j<Ny){
         double Vvp = Vv[i][j];
         double Vvn = 0.5*(Vv[i][j+1]+Vv[i][j]);
         double Vvs = 0.5*(Vv[i][j-1]+Vv[i][j]);
         double Vve = 0.5*(Vv[i+1][j]+Vv[i][j]);
         double Vvw = 0.5*(Vv[i-1][j]+Vv[i][j]);
         double Vue = 0.5*(Vu[i+1][j]+Vu[i+1][j-1]);
         double Vuw = 0.5*(Vu[i][j]+Vu[i][j-1]);
         double Vn = Vv[i][j+1];
         double Vs = Vv[i][j-1];
         double Ve = Vv[i+1][j];
         double Vw = Vv[i-1][j];
         //Calcul vp
         Rv = -(densitat*incx)*(Vvn*Vvn-Vvs*Vvs+Vue*Vve-Vuw*Vvw) + nyu*(Vn+Vs+Ve+Vw-4*Vvp);
         //Vpv[i][j] = Vv[i][j] + (inct/(densitat*incx*incx))*(1.5*Rv - 0.5*Rvant[i][j]);
         Vpv[i][j] = Vv[i][j] + ((inct/(densitat*incx*incx))*(1*Rv));
         Rvant[i][j] = Rv;
      }
      else{
         Vpv[i][j] = 0;
      }
   }
}

   //*********************************************************************
   //                              STEP 2
   //*********************************************************************

   //Calcul camp de Pressio
   itefin = false;
   while(!itefin){
   itefin = true;
   for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx; i++){
```

```
      double Pant = P[i][j];
      double Pn = 0;
      double Ps = 0;
      double Pe = 0;
      double Pw = 0;
      double vpn = Vpv[i][j+1];
      double vps = Vpv[i][j];
      double upe = Vpu[i+1][j];
      double upw = Vpu[i][j];
      double an = 0;
      double as = 0;
      double ae = 0;
      double aw = 0;

      if(i != 0){
         aw = 1;
         Pw = P[i-1][j];
      }
      if(i != Nx-1){
         ae = 1;
         Pe = P[i+1][j];
      }
      if(j != 0){
         as = 1;
         Ps = P[i][j-1];
      }
      if(j != Ny-1){
         an = 1;
         Pn = P[i][j+1];
      }
      double ap = an+as+ae+aw;
      P[i][j] = (1/ap)*(an*Pn+as*Ps+ae*Pe+aw*Pw - ((densitat*incx)/(inct))*(vpn-vps+upe-upw));
      double error = fabs(Pant - P[i][j]);
      if(error > epsilon) itefin = false;
   }
}
   }

   //*********************************************************************************
   //                                STEP 3
   //*********************************************************************************

   //Calcul velocitat u

   for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx+1; i++){
      if(i>0 and i <Nx and j>0 and j<Ny-1){
         Vu[i][j] = Vpu[i][j] - (inct/densitat)*((P[i][j]-P[i-1][j])/incx);
      }
      else if(j == Ny-1){
         Vu[i][j] = Uref;
      }
      else {
         Vu[i][j] = 0;
      }

   }
}
   //Calcul velocitat v

   for(int j = 0; j<Ny+1; j++){
   for(int i = 0; i<Nx; i++){
      if(i>0 and i <Nx-1 and j>0 and j<Ny){
         Vv[i][j] = Vpv[i][j] - (inct/densitat)*((P[i][j]-P[i][j-1])/incy);
```

```
        }
        else{
            Vv[i][j] = 0;
        }
    }
}
}
ofstream Archivo1;
Archivo1.open("Velocitat_u.txt");
Archivo1<<"Y"<<"\t"<<"u"<<"\n";
ofstream Archivo2;
Archivo2.open("Velocitat_v.txt");
Archivo2<<"X"<<"\t"<<"v"<<"\n";

cout<<"--------------------MAPA Vpu--------------------"<<endl;

for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<=Nx; i++){
        cout<<Vpu[i][j]<<" ";
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA Vu--------------------"<<endl;
cout<<endl;
for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<=Nx; i++){
        cout<<Vu[i][j]<<" ";
        if(i == Nx/2){
            Archivo1<<(incy/2) + j*incy<<"\t"<<Vu[i][j]<<"\n";
        }
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA Vpv--------------------"<<endl;
cout<<endl;
for(int j = Ny; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<Vpv[i][j]<<" ";
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA Vv--------------------"<<endl;
cout<<endl;
for(int j = Ny; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<Vv[i][j]<<" ";
        if(j == Nx/2){
            Archivo2<<(incx/2) + i*incx<<"\t"<<Vv[i][j]<<"\n";
        }
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA P--------------------"<<endl;
cout<<endl;
for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<P[i][j]<<" ";
    }
    cout<<endl;
}
```

```
ofstream Archivo3;
Archivo3.open("Mapa_velocitat.txt");
Archivo3<<"X"<<"\t"<<"Y"<<"\t"<<"Z"<<"\t"<<"u"<<"\t"<<"v"<<"\t"<<"P"<<"\n";
double u = 0;
double v = 0;
for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx; i++){
      u = 0.5*(Vu[i][j]+Vu[i+1][j]);
      v = 0.5*(Vv[i][j]+Vv[i][j+1]);
      Archivo3<<(incx/2)+i*incx<<"\t"<<(incy/2)+j*incy<<"\t"<<0<<"\t"<<u<<"\t"<<v<<"\t"<<P[i][j]<<"\n
       ";
   }
}
t1 = clock();

double time = (double(t1-t0)/CLOCKS_PER_SEC);
cout << "Execution Time: " << time << endl;
ofstream Archivo6;
Archivo6.open("Execution time.txt");
Archivo6<<time;
}
```

5. Differentially Heated Cavity

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
#include <ctime>
using namespace std;


int main(){
unsigned t0, t1;

t0=clock();
double L = 1;
int Ny = 81;
int Nx = 81;
double incx = L/Nx;
double incy = L/Ny;
double Uref = 0;
double temps = 30;
double inct = 0.00001;
double tempsmaxim = temps/inct;
double densitat = 1;
double nyu = 1;
double Re = (densitat*Uref*L)/(nyu);
double Ru = 0;
double Rv = 0;
double itefin = false;
double epsilon = 0.00001;

double Thot = 1;
double Tcold = 0;
double cp = 0.71;
double beta = 1;
double lambda = 1;
double Ra = 1E6;
double gravetat = (Ra * nyu * lambda) / (cp*beta*pow(densitat,2)*(Thot-Tcold)*pow(L,3));
```

```
double gamma = lambda/cp;
 Ra = (cp*gravetat*beta*pow(densitat,2)*(Thot-Tcold)*pow(L,3))/(nyu*lambda);
double Tfluid = 0.5;
cout<<"La gravetat es: "<<gravetat<<endl;
cout<<"El nombre Ra es: "<<Ra<<endl;

string esquema = "CDS";        //UDS or CDS
cout<<"L'esquema seleccionat es: "<<esquema<<endl;
double up[Nx+1][Ny];
double u[Nx+1][Ny];
double vp[Nx][Ny+1];
double v[Nx][Ny+1];
double P[Nx][Ny];
double T[Nx][Ny];

//INTRODUIM MALLA Vpu i Vu
for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx+1; i++){
      if(j == Ny-1){
         up[i][j] = Uref;
         u[i][j] = Uref;
      }
      else{
         up[i][j] = 0;
         u[i][j] = 0;
      }
   }
}
//INTRODUIM MALLA Vpv i Vv
for(int j = 0; j<Ny+1; j++){
   for(int i = 0; i<Nx; i++){
      vp[i][j] = 0;
      v[i][j] = 0;
   }
}
//INTRODUIM MALLA P i T
for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx; i++){
      if(i==0 and j<Ny-1 and j>0){
      T[i][j] = Thot;
   }
      else if(i==Nx-1 and j<Ny-1 and j>0){
      T[i][j] = Tcold;
   }
   else{
      T[i][j] = 0;
   }
      P[i][j] = 0;
   }
}

for(int k = 0; k<tempsmaxim; k++){

int iteprint = 10000;
   if(k%iteprint==0){
      cout<<k*inct<<endl;
   }

   //*********************************************************************************
   //                                STEP 1
   //*********************************************************************************
   //Calcul Velocitat predictoria u

   for(int j = 0; j<Ny; j++){
```

```
    for(int i = 0; i<Nx+1; i++){
        if(i>0 and i <Nx and j>0 and j<Ny-1){
            double uP = u[i][j];
            double vn = 0.5*(v[i-1][j+1]+v[i][j+1]);
            double vs = 0.5*(v[i-1][j]+v[i][j]);
            double ue = 0.5*(u[i+1][j]+u[i][j]);
            double uw = 0.5*(u[i-1][j]+u[i][j]);
            double un = 0.5*(u[i][j+1]+u[i][j]);
            double us = 0.5*(u[i][j-1]+u[i][j]);
            double Un = u[i][j+1];
            double Us = u[i][j-1];
            double Ue = u[i+1][j];
            double Uw = u[i-1][j];
            //Calcul up
            Ru = -((densitat*incx)*(vn*un-vs*us+ue*ue-uw*uw)) + nyu*(Un+Us+Ue+Uw-4*uP); // + inct*beta*(T[i][j]-Tfluid)*
        gravetat;
            up[i][j] = u[i][j] + ((inct/(densitat * incx * incx)) * Ru);

        }
        else if(j == Ny-1){
            up[i][j] = Uref;
        }
        else {
            up[i][j] = 0;
        }
    }
}
    //Calcul velocitat predictoria v

    for(int j = 0; j<Ny+1; j++){
    for(int i = 0; i<Nx; i++){
        if(i>0 and i <Nx-1 and j>0 and j<Ny){
            double vP = v[i][j];
            double vn = 0.5*(v[i][j+1]+v[i][j]);
            double vs = 0.5*(v[i][j-1]+v[i][j]);
            double ve = 0.5*(v[i+1][j]+v[i][j]);
            double vw = 0.5*(v[i-1][j]+v[i][j]);
            double ue = 0.5*(u[i+1][j]+u[i+1][j-1]);
            double uw = 0.5*(u[i][j]+u[i][j-1]);
            double Vn = v[i][j+1];
            double Vs = v[i][j-1];
            double Ve = v[i+1][j];
            double Vw = v[i-1][j];
            //Calcul vp
            Rv = -(densitat*incx)*(vn*vn-vs*vs+ue*ve-uw*vw) + nyu*(Vn+Vs+Ve+Vw-4*vP) ;
            vp[i][j] = v[i][j] + ((inct/(densitat*incx*incx))*(1*Rv)) + inct*beta*(T[i][j]-Tfluid)*gravetat;
        }
        else{
            vp[i][j] = 0;
        }
    }
}
    //*****************************************************************************
    //                              STEP 2
    //*****************************************************************************

    //Calcul camp de Pressio
    itefin = false;
    while(!itefin){
    itefin = true;
    for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){
        double Pant = P[i][j];
        double Pn = 0;
```

```
      double Ps = 0;
      double Pe = 0;
      double Pw = 0;
      double vpn = vp[i][j+1];
      double vps = vp[i][j];
      double upe = up[i+1][j];
      double upw = up[i][j];
      double an = 0;
      double as = 0;
      double ae = 0;
      double aw = 0;

      if(i != 0){
         aw = 1;
         Pw = P[i-1][j];
      }
      if(i != Nx-1){
         ae = 1;
         Pe = P[i+1][j];
      }
      if(j != 0){
         as = 1;
         Ps = P[i][j-1];
      }
      if(j != Ny-1){
         an = 1;
         Pn = P[i][j+1];
      }
      double ap = an+as+ae+aw;
      P[i][j] = (1/ap)*(an*Pn+as*Ps+ae*Pe+aw*Pw - ((densitat*incx)/(inct))*(vpn-vps+upe-upw));
      double error = fabs(Pant - P[i][j]);
      if(error > epsilon) itefin = false;
   }
}
   }

   //*********************************************************************************
   //                                STEP 3
   //*********************************************************************************

   //Calcul velocitat u

   for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx+1; i++){
      if(i>0 and i <Nx and j>0 and j<Ny-1){
         u[i][j] = up[i][j] - (inct/densitat)*((P[i][j]-P[i-1][j])/incx);
      }
      else if(j == Ny-1){
         u[i][j] = Uref;
      }
      else {
         u[i][j] = 0;
      }
   }
}
   //Calcul velocitat v

   for(int j = 0; j<Ny+1; j++){
   for(int i = 0; i<Nx; i++){
      if(i>0 and i <Nx-1 and j>0 and j<Ny){
         v[i][j] = vp[i][j] - (inct/densitat)*((P[i][j]-P[i][j-1])/incy);
      }
      else{
         v[i][j] = 0;
```

```
    }
  }
}
  //*********************************************************************************
  //                                STEP 4
  //*********************************************************************************

  if(esquema == "CDS"){
  //Calcul camp de Temperatura CDS

  for(int j = 0; j<Ny; j++){
  for(int i = 0; i<Nx; i++){

    if(i>0 and j>0 and i<Nx-1 and j<Ny-1){
          double uw = u[i][j];
          double ue = u[i+1][j];
          double vn = v[i][j+1];
          double vs = v[i][j];
          double ae = (gamma*incy)/incx;
          double aw = (gamma*incy)/incx;
          double an = (gamma*incx)/incy;
          double as = (gamma*incx)/incy;
          double Tp = T[i][j];
          double Te = (T[i+1][j] + Tp)/2.0;
          double Tw = (T[i-1][j] + Tp)/2.0;
          double Tn = (T[i][j+1] + Tp)/2.0;
          double Ts = (T[i][j-1] + Tp)/2.0;
          double TE2 = T[i+1][j];
          double TW2 = T[i-1][j];
          double TN2 = T[i][j+1];
          double TS2 = T[i][j-1];
          double conveccio = densitat*(Te*incy*ue - Tw*incy*uw + Tn*incx*vn - Ts*incx*vs);
          double diffusio = ae*TE2 + aw*TW2 + an*TN2 + as*TS2 - Tp*(an+aw+as+ae);
          T[i][j] = (inct/(incx*incy*densitat)) * (-conveccio + diffusio) + Tp;
  }  else if(i==0 and j<Ny-1 and j>0){
    T[i][j] = Thot;
  }
    else if(i==Nx-1 and j<Ny-1 and j>0){
    T[i][j] = Tcold;
  }
    else if(j == 0){
    T[i][j] = T[i][j+1];
  }
  else if(j == Ny-1){
    T[i][j] = T[i][j-1];
  }

  }
}
  }else if(esquema == "UDS"){

  //Calcul camp de Temperatura UDS

  for(int j = 0; j<Ny; j++){
  for(int i = 0; i<Nx; i++){

  if(i>0 and j>0 and i<Nx-1 and j<Ny-1){
        double uw = u[i][j];
        double ue = u[i+1][j];
        double vn = v[i][j+1];
        double vs = v[i][j];
        double ae = (gamma*incy)/incx;
        double aw = (gamma*incy)/incx;
        double an = (gamma*incx)/incy;
```

```
        double as = (gamma*incx)/incy;
        double Tp = T[i][j];
        double TE2 = T[i+1][j];
        double TW2 = T[i-1][j];
        double TN2 = T[i][j+1];
        double TS2 = T[i][j-1];
        double Te, Tw, Tn, Ts;
   if(ue > 0){
      Te = Tp;
   }else if( ue <= 0 ){
      Te = TE2;
   }
   if(uw > 0){
      Tw = TW2;
   }else if( uw <= 0 ){
      Tw = Tp;
   }
   if(vn > 0){
      Tn = Tp;
   }else if( vn <= 0 ){
      Tn = TN2;
   }
   if(vs > 0){
      Ts = TS2;
   }else if( vs <= 0 ){
      Ts = Tp;
   }
   double conveccio = densitat*(Te*incy*ue - Tw*incy*uw + Tn*incx*vn - Ts*incx*vs);
   double diffusio = ae*TE2 + aw*TW2 + an*TN2 + as*TS2 - Tp*(an+aw+as+ae);
   T[i][j] = (inct/(incx*incy*densitat)) * (-conveccio + diffusio) + Tp;
   }
   else if(i==0 and j<Ny-1 and j>0){
      T[i][j] = Thot;
   }
      else if(i==Nx-1 and j<Ny-1 and j>0){
      T[i][j] = Tcold;
   }
      else if(j == 0){
      T[i][j] = T[i][j+1];
   }
   else if(j == Ny-1){
      T[i][j] = T[i][j-1];
   }

    }
}
   }
}
   //*****************************************************************************
   //                                CALCUL NUSSELT
   //*****************************************************************************
   ofstream Archivo11;
   Archivo11.open("Nusseltstotals.txt");
   Archivo11<<"Y"<<"\t"<<"Nu"<<"\n";

   ofstream Archivo12;
   Archivo12.open("Nusseltcomponentx.txt");
   Archivo12<<"x"<<"\t"<<"Nu"<<"\n";
//Es parteix dels punts seguents:
double alpha = lambda/(densitat*cp);
double Nu = 0;
for (int i = 1; i<Nx; i++){
   double Nux = 0;
for(int j = 0; j<Ny; j++){
```

```
        double uprima = (u[i][j] * L)/(alpha);
        double Tprima = (T[i][j] - Tcold)/(Thot-Tcold);
        double Tprimaesquerra = (T[i-1][j] - Tcold)/(Thot-Tcold);
    Nux = Nux +  (uprima*Tprima - ((Tprima-Tprimaesquerra)/incx)) * incy;
    Archivo11<<j*incy<<"\t"<<Nux<<"\n";
}
    Archivo12<<i*incx<<"\t"<<Nux<<"\n";
    Nu = Nu + (Nux) * L/(Nx-1);
cout<<i<<": "<<Nux<<endl;

}

ofstream Archivo1;
Archivo1.open("Velocitat_u.txt");
Archivo1<<"Y"<<"\t"<<"u"<<"\n";
ofstream Archivo2;
Archivo2.open("Velocitat_v.txt");
Archivo2<<"X"<<"\t"<<"v"<<"\n";

cout<<"--------------------MAPA Vpu--------------------"<<endl;

for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<=Nx; i++){
        cout<<up[i][j]<<" ";
    }
    cout<<endl;
}

double Umax = 0;
double yumax = 0;

cout<<endl;
cout<<"--------------------MAPA Vu--------------------"<<endl;
cout<<endl;
for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<=Nx; i++){
        cout<<u[i][j]<<" ";
        if(i == Nx/2){
            if(u[i][j] > Umax){
                Umax = u[i][j];
                yumax = (incy/2) + j*incy;
            }
            Archivo1<<(incy/2) + j*incy<<"\t"<<u[i][j]<<"\n";
        }
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA Vpv--------------------"<<endl;
cout<<endl;
for(int j = Ny; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<vp[i][j]<<" ";
    }
    cout<<endl;
}
double Vmax = 0;
double xvmax = 0;
cout<<endl;
cout<<"--------------------MAPA Vv--------------------"<<endl;
cout<<endl;
for(int j = Ny; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<v[i][j]<<" ";
```

```
        if(j == Nx/2){
            if(v[i][j] > Vmax){
                Vmax = v[i][j];
                xvmax = (incx/2) + i*incx;
            }
            Archivo2<<(incx/2) + i*incx<<"\t"<<v[i][j]<<"\n";
        }
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA P--------------------"<<endl;
cout<<endl;
for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<P[i][j]<<" ";
    }
    cout<<endl;
}
cout<<"--------------------MAPA T--------------------"<<endl;
cout<<endl;
for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<T[i][j]<<" ";
    }
    cout<<endl;
}
cout<<"--------------------Nusselt--------------------"<<endl;
ofstream Archivo5;
Archivo5.open("Nusselt.txt");
Archivo5<<"El nombre de Nusselt es:"<<"\t"<<Nu<<"\n";
Archivo5<<"La velocitat mes gran u es: "<<"\t"<<Umax/(1/cp)<<"\t"<<"En la posicio y: "<<"\t"<<yumax<<"\n";
Archivo5<<"La velocitat mes gran v es: "<<"\t"<<Vmax/(1/cp)<<"\t"<<"En la posicio x: "<<"\t"<<xvmax<<"\n";
cout<<Nu<<endl;

ofstream Archivo3;
Archivo3.open("Mapa_velocitat.txt");
Archivo3<<"X"<<"\t"<<"Y"<<"\t"<<"Z"<<"\t"<<"u"<<"\t"<<"v"<<"\t"<<"P"<<"\n";
double U = 0;
double V = 0;
for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){
        U = 0.5*(u[i][j]+u[i+1][j]);
        V = 0.5*(v[i][j]+v[i][j+1]);
        Archivo3<<(incx/2)+i*incx<<"\t"<<(incy/2)+j*incy<<"\t"<<0<<"\t"<<U<<"\t"<<V<<"\t"<<T[i][j]<<"\
        n";
    }
}

ofstream Archivo4;
Archivo4.open("Mapa_Temperatura.txt");
Archivo4<<"X"<<"\t"<<"Y"<<"\t"<<"Z"<<"\t"<<"T"<<"\n";
for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){
        Archivo4<<(incx/2)+i*incx<<"\t"<<(incy/2)+j*incy<<"\t"<<0<<"\t"<<T[i][j]<<"\n";
    }
}
t1 = clock();

double time = (double(t1-t0)/CLOCKS_PER_SEC);
cout << "Execution Time: " << time << endl;
ofstream Archivo6;
Archivo6.open("Execution time.txt");
Archivo6<<time;
```

```
}
```

6. Square Cylinder

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
#include <ctime>
using namespace std;

int main(){
unsigned t0, t1;
t0=clock();
double D = 1;
double Lxx = 26;
double Lyy = 20;
int Nx = 150;
int Ny = 120;
double Uref2= 1;
double temps = 200;
double inct = 0.0001;
double tempsmaxim = temps/inct;
double nyu = 1;
double Reynolds = 100;
double densitat = Reynolds*nyu/(Uref2*D);
double Re = (densitat*Uref2*D)/(nyu);
double Ru = 0;
double Rv = 0;
double itefin = false;
double epsilon = 0.00001;

cout<<"El Reynolds es: "<<Re<<endl;
cout<<"La densitat es "<<densitat<<endl;

vector< vector<double> > up(Nx+1, vector<double> (Ny, 0));
vector< vector<double> > Ruant(Nx+1, vector<double> (Ny, 0));
vector< vector<double> > u(Nx+1, vector<double> (Ny, 0));
vector< vector<double> > vp(Nx, vector<double> (Ny+1, 0));
vector< vector<double> > v(Nx, vector<double> (Ny+1, 0));
vector< vector<double> > Rvant(Nx, vector<double> (Ny+1, 0));
vector< vector<double> > P(Nx, vector<double> (Ny, 0));

vector< vector<double> > x(Nx, vector<double> (Ny, 0));
vector< vector<double> > y(Nx, vector<double> (Ny, 0));
vector< vector<double> > incx(Nx, vector<double> (Ny, 0));
vector< vector<double> > incy(Nx, vector<double> (Ny, 0));

//INTRODUIM MALLA

double La = 8;
double Lb = 2;
double Lc = 16;
double Lu = 9;
double Lv = 2;
double Lw = 9;

int N1 = Nx/5;
int N2 = Nx/5;
int N3 = 3.0*Nx/5.0;
int N4 = 2*Ny/5.0;
```

```
int N5 = Ny/5.0;
int N6 = 2*Ny/5.0;


int Na = N1 + 0.5/(Lb/N2);
int Nb = Na + 1/(Lb/N2);
int Nu = N4 + 0.5/(Lv/N5);
int Nv = Nu + 1/(Lv/N5);

double Lx, Ly, Nxx, Nyy, A, B, c1, c2, ii, jj, xo, yo, incxx, incyy;

for(int j = 0; j<Ny; j++){
   for(int i = 0; i<Nx; i++){
      if(j<N4){

      if(i<N1){
         incx[i][j] = La/N1;
         x[i][j] = (La/N1)/2 + i*La/N1;
      }
      if(i>=N1 and i<=N2+N1){
         incx[i][j] = Lb/N2;
         x[i][j] = x[i-1][j] + incx[i-1][j]/2 + incx[i][j]/2;
      }
      if(i>N2+N1){
         incx[i][j] = Lc/N3;
         x[i][j] = x[i-1][j] + incx[i-1][j]/2 + incx[i][j]/2;
      }

      incy[i][j] = Lu/N4;
      y[i][j] = (Lu/N4)/2 + j*Lu/N4;
   }
   if(j>=N4 and j<=N5+N4){

      if(i<N1){
         incx[i][j] = La/N1;
         x[i][j] = (La/N1)/2 + i*La/N1;
      }
      if(i>=N1 and i<=N2+N1){
         incx[i][j] = Lb/N2;
         x[i][j] = x[i-1][j] + incx[i-1][j]/2 + incx[i][j]/2;
      }
      if(i>N2+N1){
         incx[i][j] = Lc/N3;
         x[i][j] = x[i-1][j] + incx[i-1][j]/2 + incx[i][j]/2;
      }

      incy[i][j] = Lv/N5;
      y[i][j] = y[i][j-1] + incy[i][j-1]/2 + incy[i][j]/2;
   }
   if(j>N5+N4){

      if(i<N1){
         incx[i][j] = La/N1;
         x[i][j] = (La/N1)/2 + i*La/N1;
      }
      if(i>=N1 and i<=N2+N1){
         incx[i][j] = Lb/N2;
         x[i][j] = x[i-1][j] + incx[i-1][j]/2 + incx[i][j]/2;
      }
      if(i>N2+N1){
         incx[i][j] = Lc/N3;
         x[i][j] = x[i-1][j] + incx[i-1][j]/2 + incx[i][j]/2;
      }
```

```
        incy[i][j] = Lw/N6;
        y[i][j] = y[i][j-1] + incy[i][j-1]/2 + incy[i][j]/2;
    }
  }
}

//INTRODUIM MALLA Vpu i Vu
for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx+1; i++){
        if(j == Ny-1){
            up[i][j] = 0;
            u[i][j] = 0;
        }else if(i == 0){
            up[i][j] = Uref2;
            u[i][j] = Uref2;
        }else if(i == Nx){
            up[i][j] = 0;
            u[i][j] = 0;
        }else if(j == 0){
            up[i][j] = 0;
            u[i][j] = 0;
        }else{
            up[i][j] = 0;
            u[i][j] = 0;
        }
        Ruant[i][j] = 0;
    }
}
//INTRODUIM MALLA Vpv i Vv
for(int j = 0; j<Ny+1; j++){
    for(int i = 0; i<Nx; i++){
        vp[i][j] = 0;
        v[i][j] = 0;
        Rvant[i][j] = 0;
    }
}
//INTRODUIM MALLA P
for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){
        P[i][j] = 0;
    }
}

for(int k = 0; k<tempsmaxim; k++){


    if(k%1000==0){
        cout<<k/1000<<endl;
    }

    //*************************************************************************
    //                                STEP 1
    //*************************************************************************
    //Calcul Velocitat predictoria u

    for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx+1; i++){

        if(i>0 and i <Nx and j>0 and j<Ny-1){
            if(i>=Na and i<=Nb and j>=Nu and j<=Nv-1){
                up[i][j] = 0;
            }else{
            double uP = u[i][j];
            double vn = 0.5*(v[i-1][j+1]+v[i][j+1]);
```

```
        double vs = 0.5*(v[i-1][j]+v[i][j]);
        double ue = 0.5*(u[i+1][j]+u[i][j]);
        double uw = 0.5*(u[i-1][j]+u[i][j]);
        double un = 0.5*(u[i][j+1]+u[i][j]);
        double us = 0.5*(u[i][j-1]+u[i][j]);
        double Un = u[i][j+1];
        double Us = u[i][j-1];
        double Ue = u[i+1][j];
        double Uw = u[i-1][j];

        double dN = y[i][j+1]-y[i][j];
        double dS = y[i][j]-y[i][j-1];
        double dE = incx[i-1][j];
        double dW = incx[i][j];
        double Sn = x[i][j]-x[i-1][j];
        double Ss = x[i][j]-x[i-1][j];
        double Se = incy[i][j];
        double Sw = incy[i][j];

        //Calcul up
        Ru = -((densitat)*(vn*un*Sn-vs*us*Ss+ue*ue*Se-uw*uw*Sw)) + nyu*(((Un-uP)*Sn/dN)+((Us-uP)*Ss/dS)+((Ue-uP)*Se
    /dE)+((Uw-uP)*Sw/dW));
        //up[i][j] = u[i][j] + ((inct/(densitat * incx[i][j] * incy[i][j])) * (1.5 * Ru - 0.5 * Ruant[i][j]));
        up[i][j] = u[i][j] + ((inct/(densitat * Sn * Se)) * Ru);
        Ruant[i][j] = Ru;
      }
      }
      else if(j == Ny-1){
        up[i][j] = up[i][j-1];
      }else if(i == 0){
        up[i][j] = Uref2;
      }else if(i == Nx){
        up[i][j] = up[i-1][j];
      }else if(j == 0){
        up[i][j] = up[i][j+1];
      }
    }
}
  //Calcul velocitat predictoria v

  for(int j = 0; j<Ny+1; j++){
  for(int i = 0; i<Nx; i++){
    if(i>0 and i <Nx-1 and j>0 and j<Ny){
      if(i>=Na and i<=Nb-1 and j>=Nu and j<=Nv){
        vp[i][j] = 0;
      }else{
      double vP = v[i][j];
      double vn = 0.5*(v[i][j+1]+v[i][j]);
      double vs = 0.5*(v[i][j-1]+v[i][j]);
      double ve = 0.5*(v[i+1][j]+v[i][j]);
      double vw = 0.5*(v[i-1][j]+v[i][j]);
      double ue = 0.5*(u[i+1][j]+u[i+1][j-1]);
      double uw = 0.5*(u[i][j]+u[i][j-1]);
      double Vn = v[i][j+1];
      double Vs = v[i][j-1];
      double Ve = v[i+1][j];
      double Vw = v[i-1][j];

      double dN = incy[i][j];
      double dS = incy[i][j-1];
      double dE = x[i+1][j]-x[i][j];
      double dW = x[i][j]-x[i-1][j];
      double Sn = incx[i][j];
      double Ss = incx[i][j];
```

```
            double Se = y[i][j]-y[i][j-1];
            double Sw = y[i][j]-y[i][j-1];
            //Calcul vp
            Rv = -(densitat)*(vn*vn*Sn-vs*vs*Ss+ue*ve*Se-uw*vw*Sw) + nyu*(((Vn-vP)*Sn/dN)+((Vs-vP)*Ss/dS)+((Ve-vP)*Se/
        dE)+((Vw-vP)*Sw/dW));
            //vp[i][j] = v[i][j] + (inct/(densitat*incx[i][j]*incy[i][j]))*(1.5*Rv - 0.5*Rvant[i][j]);
            vp[i][j] = v[i][j] + ((inct/(densitat*Sn*Se))*(1*Rv));
            Rvant[i][j] = Rv;
        }
    }
        else if(j == Ny){
            vp[i][j] = 0;
        }else if(i == 0){
            vp[i][j] = 0;
        }else if(i == Nx-1){
            vp[i][j] = vp[i-1][j];
        }else if(j == 0){
            vp[i][j] = 0;
        }
    }
}

    //*********************************************************************************
    //                                  STEP 2
    //*********************************************************************************

    //Calcul camp de Pressio
    itefin = false;
    double contpres = 0;
    while(!itefin){
    itefin = true;
    for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){
        double Pant = P[i][j];
        double Pn = 0;
        double Ps = 0;
        double Pe = 0;
        double Pw = 0;
        double vpn = vp[i][j+1];
        double vps = vp[i][j];
        double upe = up[i+1][j];
        double upw = up[i][j];
        double an = 0;
        double as = 0;
        double ae = 0;
        double aw = 0;

        if(i != 0){
            aw = incy[i][j]/(x[i][j]-x[i-1][j]);
            Pw = P[i-1][j];
        }
        if(i != Nx-1){
            ae = incy[i][j]/(x[i+1][j]-x[i][j]);
            Pe = P[i+1][j];
        }
        if(j != 0){
            as = incx[i][j]/(y[i][j]-y[i][j-1]);
            Ps = P[i][j-1];
        }
        if(j != Ny-1){
            an = incx[i][j]/(y[i][j+1]-y[i][j]);
            Pn = P[i][j+1];
        }
```

```
    if(i==Na-1 and j>=Nu and j<=Nv-1){      //Paret esquerra cuadrat
      ae = 0;
    }

    if(i==Nb and j>=Nu and j<=Nv-1){       //Paret dreta cuadrat
      aw = 0;
    }

    if(j==Nv and i>=Na and i<=Nb-1){       //Paret adalt cuadrat
      as = 0;
    }

    if(j==Nu-1 and i>=Na and i<=Nb-1){     //Paret abaix cuadrat
      an = 0;
    }


    double ap = an + as + ae + aw;
    if(i == Nx-1){      //Sortida
      P[i][j] = 0;
    }else if(i>=Na and i<=Nb-1 and j>=Nu and j<=Nv-1){      //Cuadrat
      P[i][j] = 0;
    }
    else{
      P[i][j] = (1/ap)*(an*Pn+as*Ps+ae*Pe+aw*Pw - ((densitat*incx[i][j])/(inct))*(vpn-vps) - ((densitat*incy[i][j])/(inct))*(
     upe-upw) );
    }
    double error = fabs(Pant - P[i][j]);
    if(error > epsilon) itefin = false;
    //cout<<error<<endl;
    //cout<<P[i][j]<<endl;
  }
}     if(contpres >= 1000000){
      cout<<"Error calculant Pressio, No convergencia"<<endl;
      return 0;
    }
    contpres = contpres + 1;

    if(isnan(P[Nx/4][Ny/4])){
      cout<<"Error NaN pressio"<<endl;
      ofstream Archivo7;
      Archivo7.open("Informe_Errors.txt");
      Archivo7<<"NaN Pressio";
      return 0;
    }
  }

  //*************************************************************************
  //                              STEP 3
  //*************************************************************************

  //Calcul velocitat u

  for(int j = 0; j<Ny; j++){
  for(int i = 0; i<Nx+1; i++){
    if(i>0 and i <Nx and j>0 and j<Ny-1){
      if(i>=Na and i<=Nb-1 and j>=Nu and j<=Nv-1){
        u[i][j] = 0;
      }else{
      u[i][j] = up[i][j] - (inct/densitat)*((P[i][j]-P[i-1][j])/(x[i][j]-x[i-1][j]));
    }
    }
    else if(j == Ny-1){
```

```
            u[i][j] = u[i][j-1];
        }else if(i == 0){
            u[i][j] = Uref2;
        }else if(i == Nx){
            u[i][j] = u[i-1][j];
        }else if(j == 0){
            u[i][j] = u[i][j+1];
        }
    }
}
    //Calcul velocitat v

    for(int j = 0; j<Ny+1; j++){
    for(int i = 0; i<Nx; i++){
        if(i>0 and i <Nx-1 and j>0 and j<Ny){
            if(i>=Na and i<=Nb-1 and j>=Nu and j<=Nv-1){
                v[i][j] = 0;
            }else{
            v[i][j] = vp[i][j] - (inct/densitat)*((P[i][j]-P[i][j-1])/(y[i][j]-y[i][j-1]));
        }
        }
        else if(j == Ny){
            v[i][j] = 0;
        }else if(i == 0){
            v[i][j] = 0;
        }else if(i == Nx-1){
            v[i][j] = v[i-1][j];
        }else if(j == 0){
            v[i][j] = 0;
        }
    }
}
        int num = k;
        ofstream Archivo;
        char nombre[500];
        double U = 0;
        double V = 0;
        if(k%1000 == 0){
        sprintf(nombre, "Vel_%05d.txt", num);
        Archivo.open(nombre, ios::trunc);
        Archivo<<"X"<<"\t"<<"Y"<<"\t"<<"Z"<<"\t"<<"u"<<"\t"<<"v"<<"\t"<<"P"<<"\t"<<"incx"<<"\t"
         <<"incy"<<"\n";
        for(int j = 0; j<Ny; j++){
            for(int i = 0; i<Nx; i++){
                U = 0.5*(u[i][j]+u[i+1][j]);
                V = 0.5*(v[i][j]+v[i][j+1]);
                Archivo<<x[i][j]<<"\t"<<y[i][j]<<"\t"<<0<<"\t"<<U<<"\t"<<V<<"\t"<<P[i][j]<<"\t"<<incx[i][j
        ]<<"\t"<<incy[i][j]<<"\n";
                }
            }
    }
}

ofstream Archivo1;
Archivo1.open("Velocitat_u.txt");
Archivo1<<"Y"<<"\t"<<"u"<<"\n";
ofstream Archivo2;
Archivo2.open("Velocitat_v.txt");
Archivo2<<"X"<<"\t"<<"v"<<"\n";

cout<<"--------------------MAPA Vpu--------------------"<<endl;

for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<=Nx; i++){
```

```
        cout<<up[i][j]<<" ";
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA Vu--------------------"<<endl;
cout<<endl;
for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<=Nx; i++){
        cout<<u[i][j]<<" ";
        if(i == Nx/2){
            Archivo1<<y[i][j]<<"\t"<<u[i][j]<<"\n";
        }
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA Vpv--------------------"<<endl;
cout<<endl;
for(int j = Ny; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<vp[i][j]<<" ";
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA Vv--------------------"<<endl;
cout<<endl;
for(int j = Ny; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<v[i][j]<<" ";
        if(j == Nx/2){
            Archivo2<<x[i][j]<<"\t"<<v[i][j]<<"\n";
        }
    }
    cout<<endl;
}
cout<<endl;
cout<<"--------------------MAPA P--------------------"<<endl;
cout<<endl;
for(int j = Ny-1; j>=0; j--){
    for(int i = 0; i<Nx; i++){
        cout<<P[i][j]<<" ";
    }
    cout<<endl;
}

ofstream Archivo3;
Archivo3.open("Mapa_velocitat.txt");
Archivo3<<"X"<<"\t"<<"Y"<<"\t"<<"Z"<<"\t"<<"u"<<"\t"<<"v"<<"\t"<<"P"<<"\t"<<"incx"<<"\t"<<"
    incy"<<"\n";
double U = 0;
double V = 0;
for(int j = 0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){
        U = 0.5*(u[i][j]+u[i+1][j]);
        V = 0.5*(v[i][j]+v[i][j+1]);
        if(i>=Na and i<=Nb-1 and j>=Nu and j<=Nv-1){        //Cuadrat
            U = 0;
            V = 0;
        }
        Archivo3<<x[i][j]<<"\t"<<y[i][j]<<"\t"<<0<<"\t"<<U<<"\t"<<V<<"\t"<<P[i][j]<<"\t"<<incx[i][j]<<
        "\t"<<incy[i][j]<<"\n";
    }
```

```
}

// Code to execute
t1 = clock();

double time = (double(t1-t0)/CLOCKS_PER_SEC);
cout << "Execution Time: " << time << endl;
ofstream Archivo6;
Archivo6.open("Execution time.txt");
Archivo6<<time;
}
```

7. Square Cylinder, coeficients

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
#include <ctime>
using namespace std;

//i>=Na and i<=Nb-1 and j>=Nu and j<=Nv-1
int main(){
   double densitat = 100;
   double temps = 200;
   double inct = 0.0001;
   double tempsmaxim = temps/inct;
   int Nx = 150;
   int Ny = 120;
   vector< vector<double> > u(Nx, vector<double> (Ny, 0));
   vector< vector<double> > v(Nx, vector<double> (Ny, 0));
   vector< vector<double> > P(Nx, vector<double> (Ny, 0));
   vector< vector<double> > incx(Nx, vector<double> (Ny, 0));
   vector< vector<double> > incy(Nx, vector<double> (Ny, 0));
   vector< vector<double> > x(Nx, vector<double> (Ny, 0));
   vector< vector<double> > y(Nx, vector<double> (Ny, 0));
   int N1 = Nx/3;
   int N2 = Nx/3;
   int N3 = Nx/3;
   int N4 = Ny/3;
   int N5 = Ny/3;
   int N6 = Ny/3;
   double La = 8;
   double Lb = 2;
   double Lc = 16;
   double Lu = 9;
   double Lv = 2;
   double Lw = 9;int Na = Nx/3 + 0.5/(Lb/N2);
   int Nb = Na + 1/(Lb/N2);
   int Nu = Ny/3 + 0.5/(Lv/N5);
   int Nv = Nu + 1/(Lv/N5);

   double cdpaverage = 0;
   double cdaverage = 0;
   double claverage = 0;
   double total = 0;

   ofstream Archivo4;
   Archivo4.open("Coeficientsglobals.txt");
   Archivo4<<"t"<<"\t"<<"Cdp"<<"\t"<<"Cd"<<"\t"<<"Cl"<<"\n";
```

```
for(int k = 0; k<1999; k++){
   ifstream Archivo;
   char nombre[500000];
   int num = (k+1)*1000;

   sprintf(nombre, "Vel_%05d.txt", num);
   Archivo.open(nombre);

   vector<double> col1V;
   vector<double> col2V;
   vector<double> col3V;
   vector<double> col4V;
   vector<double> col5V;
   vector<double> col6V;
   vector<double> col7V;
   vector<double> col8V;

   cout<<nombre<<endl;
   Archivo.seekg(0);



   while(!Archivo.eof()) {

      string col1,col2,col3,col4,col5,col6,col7,col8;
      getline(Archivo,col1,'\t'); //Separation of a tabulation
      getline(Archivo,col2,'\t'); //Separation of a tabulation
      getline(Archivo,col3,'\t'); //Separation of a tabulation
      getline(Archivo,col4,'\t'); //Separation of a tabulation
      getline(Archivo,col5,'\t'); //Separation of a tabulation
      getline(Archivo,col6,'\t'); //Separation of a tabulation
      getline(Archivo,col7,'\t'); //Separation of a tabulation
      getline(Archivo,col8,'\n'); //Separation of a new line
      if (Archivo.eof()) break;
      double a = atof(col1.c_str()); // Transforms the string to a floating point number
      double b = atof(col2.c_str());
      double c = atof(col3.c_str());
      double d = atof(col4.c_str());
      double e = atof(col5.c_str());
      double f = atof(col6.c_str());
      double g = atof(col7.c_str());
      double h = atof(col8.c_str());
      col1V.push_back(a);
      col2V.push_back(b);
      col3V.push_back(c);
      col4V.push_back(d);
      col5V.push_back(e);
      col6V.push_back(f);
      col7V.push_back(g);
      col8V.push_back(h);
      //cout << a << " " << b << endl;
   }


   double cont = 1;
   for(int j=0; j<Ny; j++){
      for(int i = 0; i<Nx; i++){

         x[i][j] = col1V[cont];
         y[i][j] = col2V[cont];
         u[i][j] = col4V[cont];
         v[i][j] = col5V[cont];
         P[i][j] = col6V[cont];
         incx[i][j] = col7V[cont];
```

```
        incy[i][j] = col8V[cont];
    cont++;
    }
}
Archivo.close();
//cout<<"hola"<<endl;
/*
ofstream Archivo3;
Archivo3.open("Mapa_velocitat.txt");
Archivo3<<"X"<<"\t"<<"Y"<<"\t"<<"Z"<<"\t"<<"u"<<"\t"<<"v"<<"\t"<<"P"<<"\t"<<"incx"<<"\t"<<"
    incy"<<"\n";
for(int j=0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){

        Archivo3<<x[i][j]<<"\t"<<y[i][j]<<"\t"<<0<<"\t"<<u[i][j]<<"\t"<<v[i][j]<<"\t"<<P[i][j]<<"\t"<<
    incx[i][j]<<"\t"<<incy[i][j]<<"\n";

    }
}
*/


double dragvn = 0;
double liftpn = 0;
double dragvs = 0;
double liftps = 0;
double dragpw = 0;
double liftvw = 0;
double dragpe = 0;
double liftve = 0;
for(int j=0; j<Ny; j++){
    for(int i = 0; i<Nx; i++){
        if(i>=Na and i<=Nb-1 and j == Nv){
            dragvn = dragvn + ((u[i][j]-u[i][j-1])/incy[i][j])*incx[i][j];
            liftpn = liftpn + P[i][j]*incx[i][j];
        }
        if(i>=Na and i<=Nb-1 and j ==Nu-1){
            dragvs = dragvs + ((-u[i][j+1]+u[i][j])/incy[i][j])*incx[i][j];
            liftps = liftps + P[i][j]*incx[i][j];
        }
        if(j>=Nu and j<=Nv-1 and i == Na-1){
            dragpw = dragpw + P[i][j]*incy[i][j];
            liftvw = liftvw + ((v[i][j]-v[i+1][j])/incx[i][j])*incy[i][j];
        }
        if(j>=Nu and j<=Nv-1 and i == Nb){
            dragpe = dragpe + P[i][j]*incy[i][j];
            liftve = liftve + ((v[i][j]-v[i-1][j])/incx[i][j])*incy[i][j];
        }
    }
}

double dragp = dragpw - dragpe;
double dragv = dragvn + dragvs;

double drag = dragvn + dragvs + dragpw - dragpe;
double lift = liftpn - liftps + liftvw + liftve;

Archivo4<<1000*inct*(k+1)<<"\t"<<fabs(dragp/(0.5*densitat))<<"\t"<<fabs(drag/(0.5*densitat))<<"\t"<<fabs(lift
    /(0.5*densitat))<<"\n";


cdpaverage = cdpaverage + fabs(dragp/(0.5*densitat));
cdaverage = cdaverage + fabs(drag/(0.5*densitat));
claverage = claverage + fabs(lift/(0.5*densitat));
```

```
    total = k;

    cout<<inct*k*1000<<endl;

}
    ofstream Archivo7;
    Archivo7.open("CoeficientsAverage.txt");
    Archivo7<<"Cdp average : "<<cdpaverage/total<<"\n";
    Archivo7<<"Cd average : "<<cdaverage/total<<"\n";
    Archivo7<<"Cl average : "<<claverage/total<<"\n";

/*
    ofstream Archivo2;
    Archivo2.open("Coeficients.txt");
    Archivo2<<"dragvn = "<<dragvn<<"\n";
    Archivo2<<"liftpn = "<<liftpn<<"\n";
    Archivo2<<"dragvs = "<<dragvs<<"\n";
    Archivo2<<"liftps = "<<liftps<<"\n";
    Archivo2<<"dragpw = "<<dragpw<<"\n";
    Archivo2<<"liftvw = "<<liftvw<<"\n";
    Archivo2<<"dragpe = "<<dragpe<<"\n";
    Archivo2<<"liftve = "<<liftve<<"\n";
    Archivo2<<"El coeficient de drag de Pressio es : "<<dragp/(0.5*densitat)<<"\n";
    Archivo2<<"El coeficient de drag de Viscositat es : "<<dragv/(0.5*densitat)<<"\n";
    Archivo2<<"El coeficient de drag es : "<<drag/(0.5*densitat)<<"\n";
    Archivo2<<"El coeficient de lift es : "<<lift/(0.5*densitat)<<"\n";
    Archivo2.close();
    */


}
```

8. Burgers equation

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
#include <ctime>
using namespace std;


class ComplexNum{
    public:
        double Re;
        double Im;

        ComplexNum(){
        }

        ComplexNum(double Re_, double Im_){
            Re = Re_;
            Im = Im_;
    }

};

static ComplexNum operator * (const ComplexNum& l, const ComplexNum& r){
    ComplexNum result;
    result.Re = (r.Re * l.Re) - (l.Im * r.Im);
    result.Im = (l.Re * r.Im) + (l.Im * r.Re);
```

```
    return result;
};
static ComplexNum operator * (const double& l, const ComplexNum& r){
    ComplexNum result;
    result.Re = r.Re * l;
    result.Im = r.Im * l;
    return result;
};
static ComplexNum operator + (const ComplexNum& l, const ComplexNum& r){
    ComplexNum result;
    result.Re = l.Re + r.Re;
    result.Im = l.Im + r.Im;
    return result;
};
static ComplexNum operator - (const ComplexNum& l, const ComplexNum& r){
    ComplexNum result;
    result.Re = l.Re - r.Re;
    result.Im = l.Im - r.Im;
    return result;
};

int main(){
    int N = 21;
    double Re = 40;
    double inct = 0.0001;
    double temps = 10;
    double instant = temps/inct;
    ComplexNum i(0,1);

    ComplexNum uri(1,1);
    ComplexNum uriol(3,3);
    uri = operator+(operator*(3,uriol) , uri);
    uri.Re = uri.Re;
    uri.Im = -uri.Im;
    cout<<uri.Re<<" "<<uri.Im<<endl;

    vector<ComplexNum> u(N);

     //Introduim vector velocitat

     for(int k = 0; k<N; k++){
        if(k==0){
            u[k].Re = 0;
            u[k].Im= 0;
         }
         else{
            u[k].Re = double(1)/k;
            u[k].Im= 0;
         }

     }

    //bucle temps

    for(int t = 0; t<instant; t++){
        //Calcul velocitat
        for(int k = 0; k<N; k++){
            if(k>1){

            //Terme Reynolds
            ComplexNum termeRe = operator*(-(pow(k,2)/Re),u[k]);
            //Terme sumatori
            ComplexNum termeSum(0,0);
            ComplexNum up(0,0);
```

```
        ComplexNum uq(0,0);
        for(int p=-N; p<=N; p++){
          for(int q=-N; q<=N; q++){
            if(q+p==k){
              if(p<0){
                up.Re = u[-p].Re;
                up.Im = -(u[-p].Im);
              }
              if(p>=0){
                up = u[p];
              }
              if(q<0){
                uq.Re = u[-q].Re;
                uq.Im = -(u[-q].Im);
              }
              if(q>=0){
                uq = u[q];
              }
              termeSum = operator+(termeSum , (operator*(operator*(up,i),operator*(q,uq)) ));

            }
          }

        }

        //cout<<"("<<termeRe.Re<<" "<<termeRe.Im<<endl;
        ComplexNum claudator = operator-(termeRe, termeSum);
        u[k] = u[k] + operator*(inct, claudator);
      }

        else{
          u[k] = u[k];
        }
      }

  }
  //Mostrar per pantalla els vectors
  ofstream Archivo1;
  Archivo1.open("Energia20.txt");
  Archivo1<<"k"<<"\t"<<"E"<<"\n";
  //velocitat
   for(int k = 1; k<N; k++){
     ComplexNum uk(u[k].Re, -(u[k].Im));
     ComplexNum Energia = operator*(u[k],uk);
     cout<<"("<<u[k].Re<<" + "<<u[k].Im<<")"<<" Energia:"<<Energia.Re<<endl;
     Archivo1<<k<<"\t"<<Energia.Re<<"\n";

   }

}
```

9. Burgers equation LES

```
#include <iostream>
#include <fstream>
#include <math.h> // To use pow
#include <vector> // To use vectors
#include <ctime>
using namespace std;
```

```
class ComplexNum{
   public:
      double Re;
      double Im;

      ComplexNum(){
      }

      ComplexNum(double Re_, double Im_){
         Re = Re_;
         Im = Im_;
   }

};

static ComplexNum operator * (const ComplexNum& l, const ComplexNum& r){
   ComplexNum result;
   result.Re = (r.Re * l.Re) - (l.Im * r.Im);
   result.Im = (l.Re * r.Im) + (l.Im * r.Re);
   return result;
};
static ComplexNum operator * (const double& l, const ComplexNum& r){
   ComplexNum result;
   result.Re = r.Re * l;
   result.Im = r.Im * l;
   return result;
};
static ComplexNum operator + (const ComplexNum& l, const ComplexNum& r){
   ComplexNum result;
   result.Re = l.Re + r.Re;
   result.Im = l.Im + r.Im;
   return result;
};
static ComplexNum operator - (const ComplexNum& l, const ComplexNum& r){
   ComplexNum result;
   result.Re = l.Re - r.Re;
   result.Im = l.Im - r.Im;
   return result;
};


int main(){

   int N = 21;
   double Re = 40;
   double inct = 0.001;
   double temps = 10;
   double instant = temps/inct;
   ComplexNum i(0,1);

   ComplexNum uri(1,1);
   ComplexNum uriol(3,3);
   uri = operator+(operator*(3,uriol) , uri);
   uri.Re = uri.Re;
   uri.Im = -uri.Im;
   cout<<uri.Re<<" "<<uri.Im<<endl;

   vector<ComplexNum> u(N);

    //Introduim vector velocitat

    for(int k = 0; k<N; k++){
      if(k==0){
         u[k].Re = 0;
```

```
        u[k].Im= 0;
      }
      else{
        u[k].Re = double(1)/k;
        u[k].Im= 0;
      }
  }

 //bucle temps
 ComplexNum Energiaultima(0,0);
 for(int t = 0; t<instant; t++){
    //Calcul velocitat
    for(int k = 0; k<N; k++){

        if(k>1){

        //Terme Reynolds
        double m = 2.0;
        double ck = 0.4523;
        double vtinfinf = 0.31 * (5.0-m)/(m+1.0) * pow((3.0-m),0.5) * pow(ck,(-3.0/2.0));

        if(k==N-1){
            ComplexNum uk(u[k].Re, -(u[k].Im));
            Energiaultima = operator*(u[k],uk);
        }

        double vtinf = (vtinfinf) * pow((Energiaultima.Re/double(N)),0.5);
        double vtasterisc = 1 + 34.5*exp(-3.03*(double(N)/double(k)));
        double vtk = vtinf * vtasterisc;
        double viscositat = 1.0/Re + vtk;
        ComplexNum termeRe = operator*(viscositat, operator*(-(pow(k,2.0)),u[k]) );
        //Terme sumatori
        ComplexNum termeSum(0,0);
        ComplexNum up(0,0);
        ComplexNum uq(0,0);
        for(int p=-N; p<=N; p++){
            for(int q=-N; q<=N; q++){
                if(q+p==k){
                    if(p<0){
                        up.Re = u[-p].Re;
                        up.Im = -(u[-p].Im);
                    }
                    if(p>=0){
                        up = u[p];
                    }
                    if(q<0){
                        uq.Re = u[-q].Re;
                        uq.Im = -(u[-q].Im);
                    }
                    if(q>=0){
                        uq = u[q];
                    }
                    termeSum = operator+(termeSum , (operator*(operator*(up,i),operator*(q,uq)) ));

                }
            }
        }
        //cout<<"("<<termeRe.Re<<" "<<termeRe.Im<<endl;
        ComplexNum claudator = operator-(termeRe, termeSum);
        u[k] = u[k] + operator*(inct, claudator);
    }

        else{
            u[k] = u[k];
```

```
        }
      }
    }

    //Mostrar per pantalla els vectors
    ofstream Archivo1;
    Archivo1.open("Energia20LES.txt");
    Archivo1<<"k"<<"\t"<<"E"<<"\n";
    //velocitat
     for(int k = 1; k<N; k++){
       ComplexNum uk(u[k].Re, -(u[k].Im));
       ComplexNum Energia = operator*(u[k],uk);
       cout<<"("<<u[k].Re<<" + "<<u[k].Im<<")"<<" Energia:"<<Energia.Re<<endl;
       Archivo1<<k<<"\t"<<Energia.Re<<"\n";

     }

}
```