Tarea 5

Física Computacional

Diego Sarceño

201900109

8 de noviembre de 2022

Los códigos tanto de *c++* como de *gnuplot*, se pueden encontrar en la carpeta de Github.

Problema 1

Solución de la ecuación de calor independiente del tiempo (Estado estacionario) por el método SOR

$$\frac{\mathrm{d}^2u}{\mathrm{d}x^2} + \frac{\mathrm{d}^2u}{\mathrm{d}y^2} = 0. \tag{1}$$

Con esto se tiene, que para un "grid" de 9×9 de $1m^2$ y 100^oC en la frontera inferior e izquierda y 0^oC en la frontera superior y derecha, se tiene el siguiente mapa de calor.

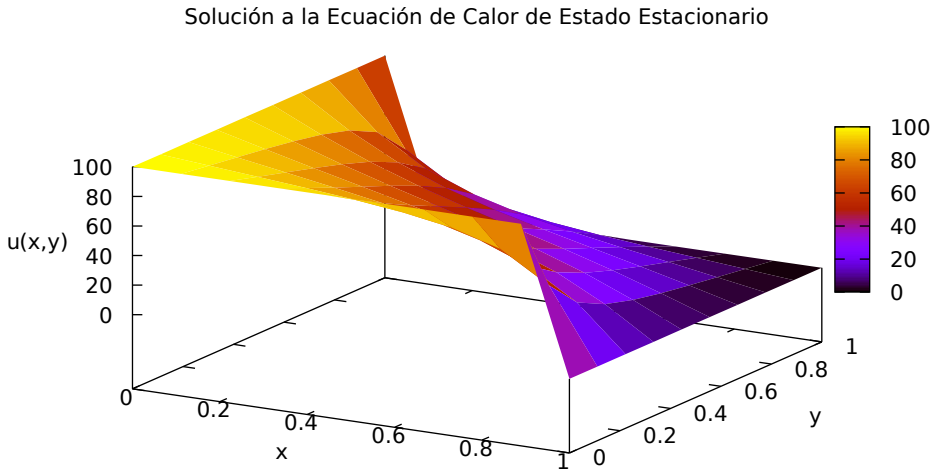

Figura 1: Mapa de calor de la solución a la ecuación (1).

```cpp
// Librerias
#include <cmath>
#include <iostream>
#include <fstream>

using namespace std;


```

```
9
10 void output( ostream &of, double **u, double *x, double *y, int Nx, int
       Ny );
11 double p1( double x );
12 double p2( double x );
13 double q1( double y );
14 double q2( double y );
15
16
17
18
19 int main()
20 {
21   int ITMAX = 10000; // maximo numero de iteraciones
22   double eps = 1e-6; // tolerancia de error
23   int Nx = 9;
24   int Ny = 9; // grid
25   double Lx = 1.0;
26   double Ly = 1.0;
27   double dx = Lx/Nx;
28   double dy = dx;
29   double alpha = 0.2; // factor para acelerar convergencia en SOR
30   ofstream of( "solucion.dat", ios::out);
31
32
33   // reservar memoria
34   double *x        = new double[Nx+1];
35   double *y        = new double[Nx+1];
36   double **u       = new double*[Nx+1];
37   double **u_nueva = new double*[Nx+1];
38
39   for( int i=0; i<Nx+1; i++ ){
40     u_nueva[i] = new double[Ny+1];
41     u[i]       = new double[Ny+1];
42   }
43
44
45   // coordenadas
46   for( int i=0; i<Nx+1; i++ ) x[i] = i*dx;
47   for( int j=0; j<Ny+1; j++ ) y[j] = j*dy;
48
49
50   // inicializar temperatura
51   for( int i=0; i<Nx+1; i++ ){
52     for( int j=0; j<Ny+1; j++ ){
53       u[i][j]       = 0.0;
54       u_nueva[i][j] = 0.0;
55     }
```

```
56  }
57
58
59  // condiciones de frontera
60  // lado inferior
61  for( int i=0; i<Nx+1; i++ ) u[i][0]  = p1( x[i] );
62
63  // lado superior
64  for( int i=0; i<Nx+1; i++ ) u[i][Ny] = p2( x[i] );
65
66  // lado izquierdo
67  for( int j=0; j<Ny+1; j++ ) u[0][j]  = q1( y[j] );
68
69  // lado derecho
70  for( int j=0; j<Ny+1; j++ ) u[Nx][j] = q2( y[j] );
71
72
73  // ciclo principal de SOR
74  bool seguimos = true; // condicion de salida
75  int k = 0; // numero de iteraciones
76
77
78  cout << "Iniciando␣SOR" << endl;
79
80  while( seguimos ){
81    if ( k > ITMAX ){
82      cerr << "Se␣alcanzo␣el␣numero␣maximo␣de␣iteraciones␣para␣SOR" <<
          endl;
83      exit(1);
84    }
85
86    seguimos = false;
87
88    for( int i=1; i<Nx; i++ ){
89      for( int j=1; j<Ny; j++ ){
90        u_nueva[i][j] = 0.25 * ( u[i+1][j] + u[i-1][j] + u[i][j-1] + u[i
             ][j+1] );
91
92        // verificamos si seguimos o no
93        if ( fabs( u_nueva[i][j] - u[i][j] ) > eps )
94          seguimos = true;
95
96        // cambiamos iteraciones
97        u[i][j] = u_nueva[i][j] + alpha*( u_nueva[i][j] - u[i][j] );
98      }
99    }
100
101   // terminamos la iteracion
```

```
102       k++;
103     }
104
105
106     cout << "SOR finalizo en " << k << " iteraciones" << endl;
107
108     // escribir solucion
109     output( of, u, x, y, Nx, Ny );
110
111
112     return 0;
113 }
114
115
116
117
118 void output( ostream &of, double **u, double *x, double *y, int Nx, int
        Ny )
119 {
120     for( int i=0; i<Nx+1; i++ ){
121       for( int j=0; j<Ny+1; j++ )
122         of << x[i] << "\t" << y[j] << "\t" << u[i][j] << endl;
123
124       of << endl;
125     }
126 }
127
128
129
130
131 double p1( double x ){ // Bottom edge
132     return 100.0;
133 }
134
135 double p2( double x ){ // Top edge
136     return 0.0;
137 }
138
139
140 double q1( double y ){ // Left edge
141     return 100.0;
142 }
143
144 double q2( double y){ // Right edge
145     return 0.0;
146 }
```