FACULTAD DE INGENIERIA
Universidad de Buenos Aires

75.12 - Análisis Numérico I

Trabajo Práctico Nº1

Resolución de Sistemas de Ecuaciones Lineales

Nombre	**Correo electrónico**	**Padrón**
Gonzalo Ávila Alterach	gonzaloavilaalterach@gmail.com	94950
Nicolás Mariano Fernandez Lema	nicolasfernandezlema@gmail.com	93410

Fecha de entrega: 16 de octubre

2º cuatrimestre de 2013

Resumen

En este problema, el sistema a resolver se trata de $Ax = b$, siendo A una matriz cuadrada tridiagonal, de dimensiones $(n-1)^2$.

El vector incógnita x a averiguar está formado por los elementos $(c_1, c_2, ..., c_{n-1})$, ya que sabiendo su valor se pueden hallar el resto de los coeficientes de todos los polinomios. Debido a las ecuación 4 y la 7, se puede apreciar que la matriz A es tridiagonal, y sus elementos son lo siguientes:

$$A_n = \begin{pmatrix} 2(h_0+h_1) & h_1 & 0 & 0 & \cdots & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \cdots & 0 & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-1}+h_n) \end{pmatrix}$$

Además, debido a que los datos de entrada utilizados la diferencia entre los θ_k consecutivos es constante, los h_k también lo son, y por lo tanto la matriz queda simétrica.

Los programas se desarrollaron para resolver matrices genéricas, sin tener en cuenta que los sistemas del problema a resolver son tridiagonales, por lo que es una resolución ineficiente: se esta utilizando más memoria que la necesaria y también se están multiplicando muchas veces ceros por ceros.

Polinomios encontrados

Las soluciones encontradas mediante Jacobi fueron las siguientes:

Radio espectral de la matriz de Jacobi (ρ_J)

Radio espectral de la matriz de Gauss-Seidel (ρ_{GS})

Gráfico de ω en función de iteraciones hasta convergencia

Gráfico de splines obtenidos

Máxima velocidad de avance

Viviendas alcanzadas por el fuego

Conclusiones

1. Código (C++)

```
1  #include <iostream>
2  #include <fstream>
3  #include <cmath>
4  #include <locale>
5  using namespace std;
6
7  #define PADRON1 94950
8  #define PADRON2 93410
9
10 double normaDos(double *x, int n){
11     double suma = 0.0;
12     for(int i = 0; i < n; i++){
13         suma += (x[i] * x[i]);
14     }
15     return sqrt(suma);
16 }
17 double normaInf(double *x, int n){
18     double maximo = 0.0;
19     for(int i = 0; i < n; i++){
20         maximo = max(maximo, abs(x[i]));
21     }
22     return maximo;
23 }
24
25 int cargarDatos(const char *archivo, double *&x, double *&y){
26     int n;
27     ifstream in(archivo);
28     if(!in.is_open()) return -1;
29
30     //Primera línea: cantidad de puntos
31     in >> n;
32     x = new double[n];
33     y = new double[n];
34
35     for(int i=0;i<n;i++){
36         in >> x[i] >> y[i]; //Cada línea tiene un par (x,y)
37         y[i] *= (PADRON1+PADRON2)/(2.0*100000.0);
38     }
39
40     return n;
41 }
42
43 void prepararMatriz(double *a, double *h, int n){
44     for(int y=0;y<n;y++)
45     for(int x=0;x<n;x++){
46         int valor = 0;
47
48         if(x == y){
49             valor = 2.0*(h[y]+h[y+1]);  //Diagonal
50         }else if(x == y+1){
51             valor = h[y+1];          //Arriba de la diagonal
52         }else if(x == y-1){
53             valor = h[y];            //Abajo de la diagonal
54         }
55         a[x+y*n] = valor;
56     }
57 }
58
```

```
59  //Resuelve Ax = b por SOR, Siendo n la dimensión
60  //w la constante de relajación (?) y rtol el error mínimo deseado
61  int sor(double *a, double *x, double *b, int n, double w, double rtol){
62      int it=0;
63      bool termino = false;
64      double *xError = new double[n];
65      while(!termino){
66          for(int j=0;j<n;j++){
67              double suma = 0.0;
68
69              for(int k=0;k<n;k++)
70                  if(j!=k)
71                      suma += a[j*n+k] * x[k];
72
73              xError[j] = x[j];
74              //Supongo que la diagonal no es cero
75              x[j] = w*(b[j]-suma)/a[j*n+j] + (1.0-w)*x[j];
76              xError[j] = x[j] - xError[j];
77          }
78          termino = normaInf(xError,n)/normaInf(x,n) <= rtol;
79          it++;
80      }
81      delete []xError;
82      return it;
83  }
84
85  int jacobi(double *a, double *x, double *b, int n, double rtol){
86      int it=0;
87      bool termino = false;
88      double *xAnterior = x;
89      double *xActual = new double[n]; // Para cambiar los punteros entre si
90      double *xError = new double[n];
91      while(!termino){
92          for(int nFila=0; nFila<n ;nFila++){
93              double suma = 0.0;
94              // sum desde j = 1 hasta n-1 de anj * xj
95              for(int j = 0; j < nFila; j++)
96                  suma += a[nFila*n+j] * xAnterior[j];
97
98              //Supongo que la diagonal no es cero
99              xActual[nFila] = (b[nFila] - suma)/a[nFila*n+nFila];
100             xError[nFila] = xActual[nFila] - xAnterior[nFila];
101         }
102         // Calculo el error para saber si se termino de iterar
103         termino = normaInf(xError,n)/normaInf(xActual,n) < rtol;
104         // Cambio los punteros de lugar para la proxima iteracion,
105         // sino x queda con el contenido correcto
106         x = xActual;
107         xActual = xAnterior;
108         xAnterior = x;
109         it++;
110     }
111     delete []xError;
112     delete []xActual;
113     return it;
114 }
115
116 //Dados los Ck, hk los pares de puntos y los h,
117 //imprime todos los polinomios con sus intervalos
118 void poly(double *c, double *x, double *y, double *h, int n){
```

```
119     ofstream outCSV("salida.csv");
120 
121     for(int k=0;k<n;k++){
122         double a = y[k];
123         double b = (y[k+1]-y[k])/h[k] - (h[k]/3.0) * (2.0*c[k]+c[k+1]);
124         double d = (c[k+1]-c[k])/(3.0*h[k]);
125 
126         cout << "y = "
127              << a << "+"
128              << b << "*(x-"<<x[k]<<")+"
129              << c[k] << "*(x-"<<x[k]<<")^2+"
130              << d << "*(x-"<<x[k]<<")^3"
131              << " {" << x[k] << ";" << x[k+1] << "}" << endl;
132 
133         for(int dx=0;dx<h[k];dx++){
134             //Imprimo los valores de los polinomios cada 1º
135             outCSV << dx+x[k] << "\t" << (a + b*dx + c[k]*dx*dx + d*dx*dx*dx) << endl;
136         }
137     }
138 }
139 
140 int main(int argc, char **args){
141     double *x=NULL, *y=NULL;
142     int n = cargarDatos("d1.csv", x, y)-1;
143     if(n<0){
144         cerr << "No se pudo abrir el archivo." << endl;
145         return 1;
146     }
147 
148     //N es la cantidad de polinomios, no de puntos totales!
149     cout << "N = " << n << endl;
150 
151     double *h = new double[n];
152     for(int k=0;k<n;k++)
153         h[k] = x[k+1] - x[k];
154 
155     //Creo el sistema a resolver
156     double *aSist = new double[(n-1)*(n-1)];
157     double *xSist = new double[n+1];
158     double *bSist = new double[n-1];
159 
160     //Semilla inicial para el SOR
161     for(int i=0;i<=n;i++)
162         xSist[i] = 0;
163 
164     prepararMatriz(aSist, h, n-1);
165 
166     //Preparo el B del sistema
167     for(int k=0;k<n-1;k++)
168         bSist[k] = (3.0/h[k+1]) * (y[k+2]-y[k+1]) - (3.0/h[k+1]) * (y[k+1]-y[k]);
169 
170     //Resuelvo el sistema, teniendo en cuenta que xSist son los Ck, por lo que 'salteo'
        el primer elemento
171     sor(aSist,xSist+1,bSist,n-1,1.0,0.001);
172     poly(xSist, x, y, h, n);
173 
174     delete []aSist;
175     delete []xSist;
176     delete []bSist;
177     delete []x;
```

```
178     delete []y;
179     return 0;
180 }
```