Laboratorio # 3
Computación Científica II

Alumno: Diego Villouta Fredes
Rol: 2773019-1

23 de noviembre de 2012

Índice

1. Introducción

- En el área de métodos numéricos y de la computación científica en general, es común trabajar con ecuaciones diferenciales, debido a las aplicaciones que pueden tener estas al usarlas para modelar fenomenos de la física, como en este caso sería una ecuación de calor.

- Mediante el uso de la poderosa herramienta matemática MatLab, se busca resolver una ecuación de este tipo aplicando el método de Forward Difference para ecuaciones parabólicas.

2. Objetivos

- Implementar la resolución de la ecuación de calor en una barra, de modo de obtener su aproximación mediante ecuaciones diferenciales parciales bajo diferentes condiciones de borde, en este caso corresponde a una ecuación parabólica.

- Generar una solución para una ecuación diferencial utilizando el método de Forward Difference.

- Utilizar la herramienta MatLab u Octave para desarrollar el laboratorio.

- Realizar gráficos comparativos de como varían las soluciones con respecto al tiempo.

3. Desarrollo

3.1. Ejercicio 1

- Se pide programar la función eqHeatFD.m, la cual fue implementada siguiendo la idea del Pseudo-código adjunto, de la siguiente forma:

```
%Recibe como inputs el largo L de la barra, el intervalo de distancia entre puntos h, intervalo de tiempo k,el factor
%alfa y el numero maximo de iteraciones a considerar como condicion de termino para el problema.

function [result] = eqHeatFD(L, h, k, alfa, MAX_ITER)
%Valores iniciales eqHeatFD(1,0.01,0.00000001,1,5000)
  result = [];
  m = L/h;

%Se genera la particion de m intervalos con m+1 valores, correspondientes a los X_i a evaluar.
  X = linspace(0,L,m+1);
  u_0 = zeros(1,m+1);
  u_j = [];

%Se evaluan los X_i en la funcion de la condicion inicial para generar el vector U_0 inicial.
%De este vector, por condicion de borde, el primer y ultimo valor deben ser cero, por lo que no se evaluan
%en la funcion.
  for i = 2:m,
    u_0(i) = (exp(2*X(i)))*sin(X(i))*cos(X(i));
  end

%Se crea la matriz tridiagonal A.
  A = zeros(m+1,m+1);
  lambda = ((alfa^2)*k)/(h^2);
  dif = 1-(2*lambda);

  for i = 1:m+1,
    if i == 1
      A(1,i) = dif;
      A(1,i+1) = lambda;
    elseif i == m+1
      A(i,i) = dif;
      A(i,i-1) = lambda;
    else
      A(i,i) = dif;
      A(i,i-1) = lambda;
      A(i,i+1) = lambda;
  end

  u_ant = u_0;

%Se ejecuta lo que aparece en el pseudo-codigo desde la linea 5 hasta la linea 12.
  for iter = 1:MAX_ITER,
    u_j = A*u_ant';

%Se aplica la condicion de borde donde el primer y ultimo elemento de u_j deben ser cero.
    u_j(1) = 0;
    u_j(m+1) = 0;

    u_j2 = u_j';
```

```
    diferencia  = u_j' - u_ant;

  %Se agregan vectores u_j generados a una matriz de resultados, para ser usada al momento de graficar.
    result (:, iter ) = u_ant;

    u_ant = u_j2;

  %Condicion de termino del algoritmo. Si la diferencia entre dos vectores U_j consecutivos es menor
  %a cierta cota, se almacena el ultimo vector generado y se detiene el algoritmo.
    if max(abs(diferencia)) <= (1.1*10^(-4))
      result (:, iter ) = u_ant;
      break
    end
  end
end
```

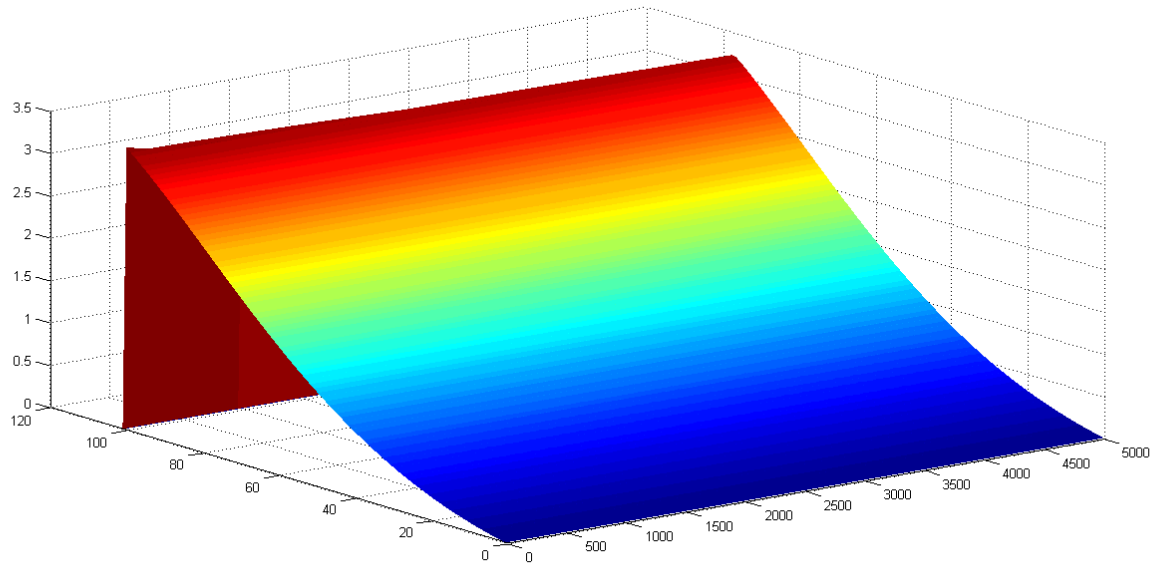
3.2. Ejercicio 2

- Se pide graficar el comportamiento de la función U para los puntos obtenidos anteriormente. El código a continuación, cabe destacar lo breve que es ya que solo recibe datos y los grafica:

```
%Recibe el parametro MAX_ITER el cual segun el laboratorio debe ser 5000, y corresponde al numero
%maximo de iteraciones del algoritmo antes de detenerse.

function [] = surfaceDataTime(MAX_ITER)
  mesh(eqHeatFD(1,0.01,0.00000001,1,MAX_ITER))
end
```

- A continuación, el gráfico:



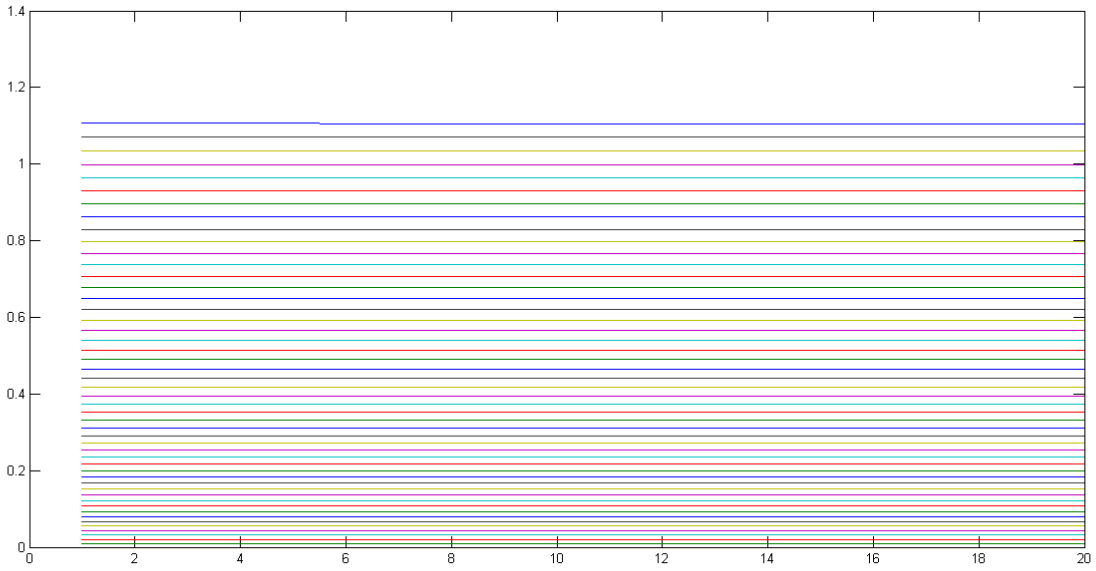
3.3. Ejercicio 3

- Se pide graficar el comportamiento de la función U a lo largo de un numero de iteraciones $t = 20$ dentro de la porción de la barra $[0; 0, 5][m]$. El código a continuación, es bastante breve ya que solo recibe datos y los grafica:

```
%Recibe como parametros el largo de la barra a utilizar y el numero de iteraciones.
%Para este caso son L = 0.5, MAX_ITER = 20.

function [] = surfaceDataInterval(L, MAX_ITER)
  result = eqHeatFD(L, 0.01, 0.00000001, 1, MAX_ITER);
  plot(result')
end
```

- A continuación el gráfico:



4. Conclusiones

- Se puede observar que el comportamiento de las temperaturas a lo largo de la barra, se mantienen constantes a travez del tiempo. Lo más probable es que al agrandar los valores de h y k, se vean más variaciones en las temperaturas, o al ejecutar un número mayor de iteraciones.
- El pseudo-código entregado, se comprobó que funciona correctamente, sin embargo se le podrían agregar detalles de manera de mejorar su funcionamiento, tales como generar una matriz que devuelva todos los vectores generados por iteración, más detalle sobre como crear el vector u_0 y sobre el porque de las dimensiones de la matriz A.

5. Anexos

- Cualquier input necesario para el laboratorio, se incluye dentro de las mismas funciones, y se encuentran ademas en los enunciados del mismo.