Resolución de la Ecuación del Calor en 2D Aplicando el Método de Diferencias Finitas de Segundo Orden

Daniel Isaías Vásquez Ramos
20161003633
divasquez@unah.hn
Universidad Nacional Autónoma de Honduras

***Resumen*—En este trabajo se presenta la solución de la Ecuación de Calor en dos dimensiones, mediante el método de diferencias finitas. Al reemplazar las derivadas en la ecuación diferencial parcial por su aproximación de cociente de diferencias se obtendrá un sistema de ecuaciones lineales de la forma $Ax = b$. Este sistema de ecuaciones se resolverá implementando métodos de solución directa, así como métodos iterativos clásicos, en el lenguaje de programación C++. Se hará una comparación de su eficiencia a la hora de resolver el problema, se mostrarán estos resultados y la visualización de dicha solución.**

***Index Terms*—Ecuación de Calor, diferencias finitas, solución de sistemas de ecuaciones lineales, C++.**

I. Introducción

EL método de diferencias finitas es muy útil en la resolución de problemas con valores de frontera (PVF). está basado en aproximaciones locales de las derivadas en una ecuación diferencial, la cual se obtiene de expansiones de Taylor de bajo nivel. En dos dimensiones, la idea del método es discretizar la ecuación diferencial parcial sustituyendo las derivadas por su respectiva aproximación de segundo orden. El método resulta especialmente atractivo cuando se definen regiones rectangulares y se utilizan mallas uniformes. La matriz resultante de la discretización suele ser una matriz con unas pocas diagonales distintas de cero, es decir, una matriz rala, lo cual facilita el uso de métodos clásicos para la solución de sistemas de ecuaciones lineales.

II. Metodología

II-A. Descripción del problema

Muchos problemas son modelados a partir de ecuaciones diferenciales parciales, este es el caso de la ecuación del calor que sirve de estudio para encontrar la distribución de temperatura de una superficie. Generalmente estos problemas requieren una discretización en ecuación de diferencias, esto es, reemplazar las derivadas parciales por su aproximación de diferencias. Esta discretización da lugar a un sistema de ecuaciones lineales.

Consideremos el problema de valor en la frontera

$$\begin{cases} -D\left[\dfrac{\partial^2 U}{\partial x^2}+\dfrac{\partial^2 U}{\partial y^2}\right] = f(x,y) & \text{en} \quad \Omega=[a,b]\times[c,d] \\ U(x,y)=g(x,y) & \text{en} \quad \partial\Omega \end{cases}$$

donde D representa la constante de difusión o conductividad térmica del material, $f(x, y)$ es la fuente o sumidero de calor y la función $g(x, y)$ es la condición de frontera que especifica el valor de $U(x, y)$ en el borde $\partial\Omega$.

Este es un problema en condiciones de estado estacionario y sin generación de energía.

II-B. Método de resolución del problema

Dividimos la región bidimensional en pequeñas regiones con incrementos de tamaño $h > 0$ en dirección de x, donde $h = x_{i+1} - x_i$, y $k > 0$ en dirección de y donde $k = y_{j+1} - y_j$, que suelen llamarse "tamaños de paso". Esto nos dará una cantidad N de nodos en x y M nodos en y.

Utilizando la fórmula de diferencias finitas centradas de segundo orden para aproximar las derivadas parciales tenemos

$$\frac{\partial^2 U}{\partial x^2} \approx \frac{U(x_{i-1},y_j)-2U(x_i,y_j)+U(x_{i+1},y_j)}{h^2}$$
$$\frac{\partial^2 U}{\partial y^2} \approx \frac{U(x_i,y_{j-1})-2U(x_i,y_j)+U(x_i,y_{j+1})}{k^2}$$

Simplificando la notación y reemplazando en la ecuación diferencial

$$-D\left[\frac{U_{i-1,j}-2U_{i,j}+U_{i+1,j}}{h^2}+\frac{U_{i,j-1}-2U_{i,j}+U_{i,j+1}}{k^2}\right]=f_{i,j}$$

Luego,

$$-D\left[\frac{U_{i-1,j}+U_{i+1,j}}{h^2}-2\left(\frac{1}{h^2}+\frac{1}{k^2}\right)U_{i,j}+\frac{U_{i,j-1}+U_{i,j+1}}{k^2}\right]=f_{i,j}$$

donde $i = 0, 1, \ldots, N-1$, $j = 0, 1, \ldots, M-1$. Esta ecuación nos genera el sistema de ecuaciones lineales de la forma $Ax = b$, que será construido en el lenguaje C++.

II-B1. Discretización del dominio Ω: Se implementa la función "double **getMesh", esta recibe los parámetros, a, c que son los límites inferiores del dominio en x y y respectivamente, los tamaños de paso h y k, y la cantidad de nodos N y M. La función retorna un arreglo bidimensional con los valores del mallado.

Por conveniencia se toma la cantidad de nodos como $N = (b-a)/h + 2$ y $M = (d-c)/k + 1$. Creamos un arreglo bidimensional con M filas y N columnas, se almacena en la

primer posición de cada fila el valor $y_j = c + i * k$, el resto de valores en la fila son los $x_i = a + (j-1) * h$ de tal forma que si deseamos acceder al punto (x_i, y_j) este será

$$(x_i, y_j) = (\text{malla}[j][i+1]\ ,\ \text{malla}[j][0]).$$

II-B2. Matriz $\boldsymbol{A}$ de coeficientes: Es una matriz por bloques, donde los bloques centrales son tridiagonales, los bloques adyacentes son diagonales y el resto son bloques de ceros. Para implementarlo, primero, se consideran dos funciones, "double **getTridiag" para obtener los bloques tridiagonales y "double **getDiag" para los bloques diagonales.

Según el sistema planteado, se consideran a $i = 0, 1 \ldots, N-1$ y $j = i = 0, 1 \ldots, M-1$, sin embargo, anteriormente asignamos $N = (b-a)/h+2$ y $M = (d-c)/k+1$ por conveniencia a la hora de estructurar el mallado, por lo que los $N-1$ y $M-1$ nodos, para nosotros serán $N-3$ y $M-2$ nodos en x y y respectivamente. Dejando claro esto, podemos decir entonces que la matriz de coeficientes tendrá dimensión $(N-3)*(M-2) \times (N-3)*(M-2)$,

$$\begin{bmatrix} \boldsymbol{A}_{0,0} & \boldsymbol{A}_{0,1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \boldsymbol{A}_{1,0} & \boldsymbol{A}_{1,1} & \boldsymbol{A}_{1,2} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{A}_{2,2} & \boldsymbol{A}_{2,3} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \cdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{A}_{M-3,M-4} & \boldsymbol{A}_{M-3,M-3} & \boldsymbol{A}_{M-3,M-2} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{A}_{M-2,M-3} & \boldsymbol{A}_{M-2,M-2} \end{bmatrix}$$

como resultado, tenemos $(M-2) \times (M-2)$ bloques de tamaño $(N-3) \times (N-3)$ cada bloque.

Para el bloque tridiagonal, la diagonal principal es $a_{i,i} = 2D(\frac{1}{h^2} + \frac{1}{k^2})$ y las diagonales adyacentes, es decir, cuando $j = i+1$ y $j = i-1$ son $a_{i,j} = -\frac{D}{h^2}$. En el bloque diagonal, $d_{i,i} = -\frac{D}{k^2}$, el resto de elementos en los bloques son ceros.

A partir de estas dos funciones, se crea la matriz de coeficientes con la función "double **getAmatrix" que retornará la matriz del sistema. Se utilizan las operaciones de división entera y módulo para posicionarnos en un determinado bloque y en cada fila de dicho bloque.

Al tener bloques de $(N-3) \times (N-3)$, los bloques están alineados con los múltiplos de $N-3$, entonces, al dividir (división entera) el índice de fila o columna por $N-3$ se obtiene el número de bloque al que pertenece el elemento. Por ejemplo, si $N = 6$ y $i = 5$, entonces $i/(N-3) = 5/3 = 1$ lo que significa que el elemento está en el segundo bloque de la fila. El procedimiento es análogo para las columnas.

El módulo sirve para saber la posición dentro del bloque, si $i = 5$ y $N = 6$, $i\,\%(N-3) = 5\,\%3 = 2$, entonces el elemento esta en la tercer fila del bloque, si $j = 8$, $j\,\%(N-3) = 8\,\%3 = 2$, entonces estamos en la tercer columna del bloque.

Cuando $i/(N-3) = j/(N-3)$, se rellenan los bloques tridiagonales, cuando $i/(N-3) = j/(N-3)+1$ o $i/(N-3) = j/(N-3)-1$ se rellenan los bloques diagonales, el resto de bloques son ceros.

II-B3. Vector $\boldsymbol{b}$ de términos independientes : Se implementa la función "double *getBvector", el vector a retornar es de tamaño $(N-3)*(M-2)$.

Para rellenar los valores de cada componente del vector, necesitamos una serie de condiciones:

- Si $i = j = 1$: $f_{1,1} + D(U_{0,1}/h^2 + U_{1,0}/k^2)$
- Si $i = 1,\ j = 2, \ldots, M-2$: $f_{1,j} + D(U_{0,j})/h^2$
- Si $i = 1,\ j = M-1$: $f_{1,M-1} + D(U_{0,M-1}/h^2 + U_{1,M}/k^2)$
- Si $i = 2, \ldots, N-2,\ j = 1$: $f_{i,1} + D(U_{i,0}/k^2)$
- Si $i = 2, \ldots, N-2,\ j = M-1$: $f_{i,M-1} + D(U_{i,M}/k^2)$
- Si $i = 2, \ldots, N-2,\ j = 2, \ldots, M-2$: $f_{i,j}$
- Si $i = N-1,\ j = 1$: $f_{N-1,1} + D(U_{N,1}/h^2 + U_{N-1,0}/k^2)$
- Si $i = N-1,\ j = 2, \ldots, M-2$: $f_{N-1,j} + D(U_{N,j}/h^2)$
- Si $i = N-1,\ j = M-1$: $f_{N-1,M-1} + D(U_{N,M-1}/h^2 + U_{N-1,M}/k^2)$

Recordando que $N-1$ y $M-1$ para la implementación son $N-3$ y $M-2$, además, los valores $U_{0,j}$, $U_{i,0}$, $U_{i,M}$ y $U_{N,j}$ con $i = 0, \ldots, N$ y $j = 0, \ldots, M$ son las condiciones de frontera.

La función getBvector además de los parámetros de los que hemos hablado a lo largo del trabajo, recibe también la función $f(x,y)$ donde se evalua cada punto de la discretización, y la función $g(x,y)$ de la que se generan las condiciones de frontera mencionadas anteriormente.

II-C. Descripción de los experimentos

Se ha construido el sistema de ecuaciones lineales de la forma $Au = b$ donde A es la matriz de coeficientes, b el vector independiente y el vector u es la solución que estamos buscando. Con lo anterior, estamos listos para realizar el experimento. Resolveremos el sistema implementando en C++ los métodos directos:

- Eliminación Gaussiana
- Factorización LU
- Factorización de Cholesky
- Factorización QR

Además se implementarán los métodos iterativos:

- Método de Jacobi
- Método de Gauss-Seidel
- Gradiente Conjugado

Si el método resuelve el sistema, se registrará el tiempo que toma en resolverlo, y el registro del número de iteraciones para los métodos iterativos, esto nos servirá para comparar la eficiencia y exactitud entre cada método. Para esto, tomaremos el siguiente ejemplo:

Considere el problema de determinar la distribución de calor en estado estable en una placa cuadrada de acero delgada, con dimensiones 1 m por 1 m. Conservamos dos fronteras adyacentes a $0\ ^\circ C$, las otras dos fronteras aumenta linealmente de $0\ ^\circ C$ en una esquina hasta $200\ ^\circ C$ en el sitio donde ambos lados se encuentran.

Se utilizará los parámetros $D = 45$, $h = k = 0{,}025$ dando como resultado $N = M = 40$ nodos en x y y respectivamente y la función $f(x,y) = e^{-x-y}$. Además, la función $g(x,y)$ para generar las condiciones de frontera.

II-D. Resultados

Al resolver el problema con los distintos métodos antes mencionados, obtenemos los siguientes resultados:

Cuadro I
RESULTADOS MÉTODOS DIRECTOS

Método	Tiempo de ejecución	¿Resolvió el sistema?
Eliminación Gaussiana	4.894352 s	Sí
Factorización LU	4.75842 s	Sí
Factorización de Cholesky	3.322786 s	Sí
Factorización QR	14.19802 s	Sí

En el tiempo de ejecución para los métodos directos se hizo un promedio de cinco ejecuciones por cada método, obteniendo así los valores que se muestran en el cuadro I.

Cuadro II
RESULTADOS MÉTODOS ITERATIVOS

Método	Tiempo de ejecución	Iteraciones	¿Resolvió el sistema?
Jacobi	237.824 s	4998	Sí
Gauss-Seidel	118.897 s	2624	Sí
Grad. Conjugado	320.883 s	6934	Sí

En el caso de los métodos iterativos debido a su extenso tiempo de ejecución, se hizo un promedio de tres ejecuciones para cada método, obteniendo los resultados del cuadro II.

III. CONCLUSIONES

- Se ha observado que tanto los métodos directos como los iterativos son capaces de resolver el sistema de ecuaciones lineales, siendo muy exactos entre sí, sin embargo, en términos de eficiencia los métodos directos son más rápidos para resolver el problema planteado.

- Entre los métodos directos el más eficiente es la Factorización de Cholesky, esto se debe a que la matriz del sistema es una matriz simétrica definida positiva lo cual le da una ventaja ya que necesita menos operaciones comparado con la Eliminación Gaussiana y la Factorización LU. Detrás de estos está la Factorización QR siendo el menos eficiente para resolver nuestro problema con un tiempo de ejecución alejado del resto de métodos directos.

- Los métodos iterativos resultan ser poco eficientes para resolver el sistema con un tiempo de ejecución bastante grande comparado con los métodos directos y una cantidad de iteraciones muy elevadas tratándose de un sistema de ecuaciones que para el ejemplo resuelto, es relativamente pequeño. Esto se debe a las características del mismo, pues debemos tomar en cuenta el condicionamiento de la matriz del sistema lineal.

- Se ha concluido que el método más rápido para resolver el problema que planteamos es la Factorización de Cholesky, con un tiempo de ejecución aproximado de 3.32 segundos. El método de Gauss-Seidel a pesar de ser casi el doble de eficiente que Jacobi y casi el triple que el Gradiente Conjugado sigue estando por debajo de los métodos directos.

IV. ANEXOS

```
     0     0   0.25    0.5   0.75     1
  0.25     0   0.25    0.5   0.75     1
   0.5     0   0.25    0.5   0.75     1
  0.75     0   0.25    0.5   0.75     1
     1     0   0.25    0.5   0.75     1
```

Figura 1. Ejemplo de la malla con $\Omega = [0,1] \times [0,1]$, $h = k = 0{,}25$, $N = 4+2$, $M = 4+1$ en C++

```
2880  -720     0  -720     0     0     0     0     0
-720  2880  -720     0  -720     0     0     0     0
   0  -720  2880     0     0  -720     0     0     0
-720     0     0  2880  -720     0  -720     0     0
   0  -720     0  -720  2880  -720     0  -720     0
   0     0  -720     0  -720  2880     0     0  -720
   0     0     0  -720     0     0  2880  -720     0
   0     0     0     0  -720     0  -720  2880  -720
   0     0     0     0     0  -720     0  -720  2880
```

Figura 2. Ejemplo matriz A, con $D = 45$, $h = k = 0{,}25$ $N = 4{+}2$, $M = 4+1$ en C++

Utilizamos la herramienta Gnuplot para graficar la solución $U(x, y)$:

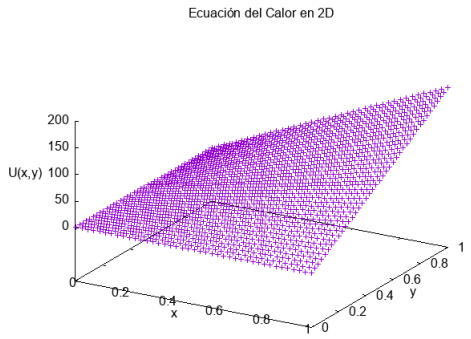

Figura 3. Representación gráfica de $U(x, y)$ en Gnuplot

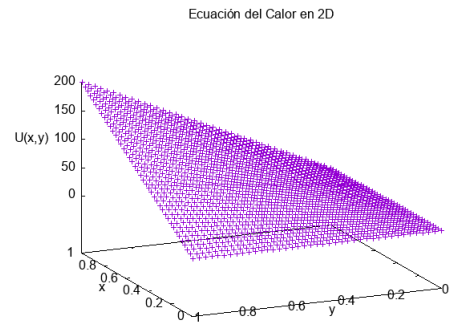

Figura 4. $U(x, y)$ con Gnuplot

REFERENCIAS

[1] Yousef Saad, *Iterative Methods for Sparse Linear Systems*, 3rd ed. 2000.
[2] Biswa Nath Datta, *Numerical Linear Algebra and Applications*, 2nd ed. 2010.
[3] Richard L. Burden, J. Douglas Faires, *Análisis Numérico*, 7th ed. 2001.