Tarea 5 - Diferencias Finitas
Métodos Numéricos

Rafael Alejandro García Ramírez

18 de septiembre de 2023

1. Introducción

La distribución de calor dada dentro de un sistema unidimensional para K la difusividad térmica y Q una fuente de calor puede ser descrita como

$$K\frac{\partial^2 \Phi}{\partial x^2} = -Q \tag{1}$$

Para un sistema discretizado recordemos que

$$f(x+dx) + f(x-dx) = 2f(x) + f^{''}(x)(dx)^2 + \mathcal{O}((dx)^4)$$

Lo cuál es lo mismo que

$$f^{''}(x) = \frac{f(x+dx) - 2f(x) + f(x-dx)}{(dx)^2} + \mathcal{O}((dx)^2)$$

Para algún Δx tomando en cuando que $x \pm dx$ es el equivalente a $i \pm 1$, la ecuación (1) se torna de la forma

$$\Phi_{i-1} - 2\Phi_i + \Phi_{i+1} = \frac{-Q(\Delta x)^2}{K}$$

Tomando esto en cuenta podemos construir el sistema de ecuaciones para $i = 1, 2, \ldots, n$ como

$$\begin{bmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \vdots \\ \Phi_{n-2} \\ \Phi_{n-1} \end{bmatrix} = \begin{bmatrix} \frac{Q(\Delta x)^2}{K} + \Phi_0 \\ \frac{Q(\Delta x)^2}{K} \\ \frac{Q(\Delta x)^2}{K} \\ \vdots \\ \frac{Q(\Delta x)^2}{K} \\ \frac{Q(\Delta x)^2}{K} + \Phi_n \end{bmatrix}$$

Esto se reduce en resolver el los $n-1$ ecuaciones. Para esto haremos el método de Cholesky a continuación, el cuál consiste en tomar el sistema de la forma $\mathbb{A}\mathbf{x} = \mathbf{b}$ y buscar $\mathbb{A} = \mathbb{L}\mathbb{D}\mathbb{L}^T$ o alternativamente $\mathbb{A} = \mathbb{L}\mathbb{L}^T$. Posteriormente mediante sustitución hacia adelante y hacia atrás solucionar los pares de matrices que resultan. Para ello hacemos

$$\mathbb{L}\mathbf{v} = \phi$$

$$\mathbb{D}\mathbf{w} = \mathbf{v}$$

$$\mathbb{L}^T\mathbf{x} = \mathbf{w}$$

Para dos variables vacías $\mathbf{v}$ y $\mathbf{w}$ que nos servirán para dividir el problema original. A continuación de presenta un algoritmo para la descomposición de Cholesky de la forma $\mathbb{L}\mathbb{D}\mathbb{L}^T$.

2. Pseudocódigo

Se relizó un programa en `C` que realiza la descomposición de Cholesky de la forma especificada anteriormente, para ello se toman en consideración las condiciones necesarias tales como primer elemento no nulo $\{\mathbb{A}\}_{0,0} \neq 0$, la simetría del la matriz $\mathbb{A}$ (esto es, para una matriz con entradas reales tal que $\mathbb{A} = \mathbb{A}^T$) y que su diagonal sea dominante, es decir que el elemento de la diagonal sea mayor que la suma de sus demás elementos en la fila.

Cuando ocurre alguno de estos errores se imprime el tipo de error y se devuelve para conveniencia del usuario la entrada que generó este error.

Algorithm 1 Descomposición de Cholesky $\mathbb{LDL}^T$

```
procedure DescomponerCholesky(N, A, L, D)
                ▷ Descompone una matriz A en L (triangular inferior) y D (diagonal) usando el método de Cholesky.
                ▷ N: Tamaño de la matriz A y las matrices L y D.
                ▷ A: Matriz de entrada A de tamaño N × N.
                ▷ L: Matriz triangular inferior L de tamaño N × N.
                ▷ D: Matriz diagonal D de tamaño N × N.
    if |A[0][0]| ≤ 1 × 10^-16 then
        Imprimir(.Error: Matriz imposible. Error Cholesky 0: A[0][0] = 0”)
        Salir(1)
                ▷ Revisar si la matriz es simétrica
    A^T ← Transponer(A)
    for i ← 0 hasta N − 1 do
        for j ← 0 hasta N − 1 do
            if |A[i][j] − A^T[i][j]| > 1 × 10^-16 then
                Imprimir(Error Cholesky: Matriz no simétrica: A[i][j]! = A[j][i])
                Salir(1)
                ▷ Liberar memoria para A^T (Para C)
    for i ← 0 hasta N − 1 do
        LiberarMemoria(A^T[i])
    LiberarMemoria(A^T)
                ▷ Inicializar las matrices L y D con ceros
    for i ← 0 hasta N − 1 do
        for j ← 0 hasta N − 1 do
            L[i][j] ← 0.0
            D[i][j] ← 0.0
                ▷ Iniciar descomposición de Cholesky
    for i ← 0 hasta N − 1 do
        for j ← 0 hasta i do
            suma ← 0.0
            if i == j then
                for k ← 0 hasta j − 1 do
                    suma ← suma + L[j][k] · D[k][k] · L[j][k]
                if (temp ← A[j][j] − suma) < 0 then
                    Imprimir(Error: Matriz no definida positiva. Error Cholesky 1: A[j][j] − suma < 0)
                    Salir(1)
                D[j][j] ← temp
                L[j][j] ← 1.0
            else
                for k ← 0 hasta j − 1 do
                    suma ← suma + L[i][k] · D[k][k] · L[j][k]
                if |D[j][j]| ≤ 1 × 10^-16 then
                    Imprimir(Error: Matriz imposible. Error Cholesky 2: D[j][j] = 0)
                    Salir(1)
                L[i][j] ← (A[i][j] − suma)/(D[j][j] · L[j][j])
                D[i][j] ← 0.0
```

Después de realizar esta descomposición, de debe realizar sustitución hacia adelante y hacia atrás, así como resolver un sistema de matriz diagonal; estos pseudocódigos junto con aquellos para la generación de la matriz tridiagonal de tamaño $n-1$ y el vector ϕ se pueden encontrar anexos al final de este trabajo.

3. Resultados

El pseudocódigo anterior se implementó para $n \in \{4, 10, 15, 20, 25, 30, 35, \cdots, 90, 95, 100\}$ guardando en cada caso el vector ϕ, del cual después se sustrae el valor de su nodo central. Asimismo, se muestra como referencia la solución analítica de la ecuación que tiene como expresión para las condiciones de $Q = 1000$, $K = 1$, $\Phi_0 = 0$, $\Phi_n = 100$ y $L = 10$

$$\Phi(x) = 10x(501 - 50x) \tag{2}$$

Que para un $x = 5$ tenemos que $\Phi(x = 5) = 12550$. El comportamiento de esta solución por diferencias finitas se puede apreciar dentro de la figura 1.

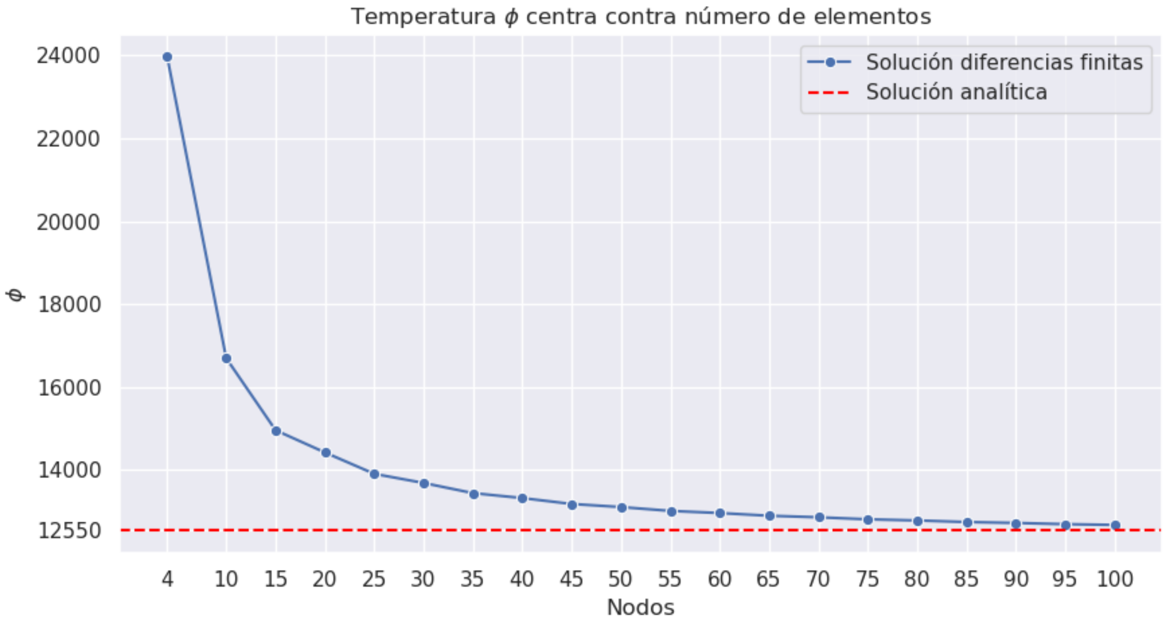

Figura 1: Gráfico conversión de método de diferencias finitas.

Podemos observar que para una cantidad de nodos pequeña nuestra solución es bastante diferente a la solución analítica a la cuál, mientras más nodos se le agregan mejores resultados se tienen.

4. Conclusiones

Dentro de las diversas soluciones posibles para abordar este problema, el método de diferencias finitas emerge como una opción altamente confiable desde el punto de vista computacional. Esto se debe a su notable nivel de convergencia, incluso con tan solo 100 divisiones. Sin embargo, es importante tener en cuenta que, dado su carácter exacto, este método presenta ciertas limitaciones generales.

Una de estas limitaciones radica en que al aumentar el número de divisiones, la matriz involucrada en el cálculo se vuelve más grande. Este aumento en la dimensionalidad de la matriz conlleva a una acumulación de errores durante el proceso de resolución. En consecuencia, llegará un punto en el que, al trabajar con un número significativo de nodos o incluso más, el método de diferencias finitas comenzará a experimentar dificultades en términos de convergencia.

En tales circunstancias, será necesario recurrir a otros métodos para resolver el sistema de ecuaciones. Esto se vuelve esencial para verificar si la solución cumple con la condición deseada, expresada como $\mathbb{A}\mathbf{x} - \mathbf{b} = 0$ en caso de no contar con una solución analítica para verificar el resultado, como lo sería por ejemplo para algún $Q = Q(x)$.

5. Referencias

1. Richard. L. Burden y J. Douglas Faires, Análisis Numérico, 7a Edición, Edito-rial Thomson Learning, 2002.

2. Samuel S M Wong, Computational Methods in Physics and Enginering, Ed. World Scientific, 3rd Edition, 1997.

3. Teukolsky, S. A., Flannery, B. P., Press, W. H., & Vetterling, W. T. (1992). Numerical recipes in C. SMR, 693(1), 59-70.

4. William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, Numerical Recipes: The Art of Scientific Computing, 3rd Edition, Cambridge University Press, 2007

Anexo: otros pseudocódigos

A continuación se presentan los pseudocógios que complementan la sección 2 de este documento que brindan ayuda para la resolución del problema completo.

Algorithm 2 Resolver sistema de ecuaciones: Sustitución hacia adelante

procedure Lx_b(N, L, b, x)
▷ N: Tamaño de la matriz triangular inferior L y los vectores b y x.
▷ L: Matriz triangular inferior de tamaño $N \times N$.
▷ b: Vector de términos independientes de tamaño N.
▷ x: Vector de solución, se actualizará con la solución después de la llamada.
  for $i \leftarrow 0$ **hasta** $N - 1$ **do**
    $suma \leftarrow 0$
    for $k \leftarrow 0$ **hasta** i **do**
      $suma \leftarrow suma + L[i][k] \cdot x[k]$
    if $|L[i][i]| \leq 1 \times 10^{-16}$ **then**
      Imprimir(Error: Matriz imposible para $Lx = b : L[i][i] = 0$)
      Salir(1)
    $x[i] \leftarrow (b[i] - suma)/L[i][i]$

Algorithm 3 Resolver sistema de ecuaciones: Matriz Diagonal

procedure Dx_b(N, D, b, x)
▷ N: Tamaño de la matriz diagonal D y los vectores b y x.
▷ D: Matriz diagonal de tamaño $N \times N$.
▷ b: Vector de términos independientes de tamaño N.
▷ x: Vector de solución, se actualizará con la solución después de la llamada.
  for $i \leftarrow 0$ **hasta** $N - 1$ **do**
    if $|D[i][i]| \leq 1 \times 10^{-16}$ **then**
      Imprimir(Error: Entrada nula para $Dx = b : D[i][i] = 0$)
      Salir(1)
    $x[i] \leftarrow b[i]/D[i][i]$

Algorithm 4 Resolver sistema de ecuaciones: Sustitución hacia atrás

procedure Ux_b(N, U, b, x)
▷ N: Tamaño de la matriz triangular superior U y los vectores b y x.
▷ U: Matriz triangular superior de tamaño $N \times N$.
▷ b: Vector de términos independientes de tamaño N.
▷ x: Vector de solución, se actualizará con la solución después de la llamada.
  for $i \leftarrow N - 1$ **hasta** 0 **decremento** 1 **do**
    $suma \leftarrow 0$
    for $k \leftarrow N - 1$ **hasta** $i + 1$ **decremento** 1 **do**
      $suma \leftarrow suma + U[i][k] \cdot x[k]$
    if $|U[i][i]| \leq 1 \times 10^{-16}$ **then**
      Imprimir(Error: Matriz imposible para $Ux = b$: $U[i][i] = 0$)
      Salir(1)
    $x[i] \leftarrow (b[i] - suma)/U[i][i]$

Algorithm 5 Construir vector ϕ para el sistema de calor

procedure construir_vector_phi(nodos, Q, k, phi_0, phi_N, L, b)
▷ Construye un vector ϕ para el sistema de calor.

  $\Delta x \leftarrow L/nodos$
  $dato \leftarrow Q \cdot \Delta x^2/k$
  for $i \leftarrow 0$ **hasta** $nodos - 2$ **do**
    $b[i] \leftarrow dato$
  $b[0] \leftarrow b[0] + phi_0$
  $b[nodos - 2] \leftarrow b[nodos - 2] + phi_N$

Algorithm 6 Construir matriz para el sistema de calor 1D

```
procedure CONSTRUIR_MATRIZ_CALOR(nodos, matriz)
        ▷ Construye una matriz que describe mediante diferencias finitas el sistema de una barra 1D siendo calentada.
                                                                        ▷ nodos: El número de nodos en la barra.
                                                        ▷ matriz: La matriz de entrada A de tamaño N × N.

    for i ← 0 hasta nodos − 1 do                                        ▷ Llenamos la matriz con ceros
        for j ← 0 hasta nodos − 2 do
            matriz[i][j] ← 0.0


    for i ← 0 hasta nodos − 1 do                 ▷ Llenamos la matriz con los valores correspondientes
        for j ← 0 hasta nodos − 1 do
            if i == j then
                matriz[i][j] ← 2.0
            else
                matriz[i][j] ← 0.0
            if i == j + 1 || i == j − 1 then
                matriz[i][j] ← −1.0
```