

Tarea 8 - Métodos numéricos
Giovanni Gamaliel López Padilla

Índice

1. Introducción	2
1.1. Métodos iterativos	2
1.1.1. Método de Jacobi y Gauss-Seidel	2
1.2. Ecuaciones diferenciales parciales	2
1.2.1. Condición de frontera de Dirichlet	2
1.2.2. Ecuación de transferencia de calor	2
2. Métodos	3
2.1. Ecuación de calor	3
2.2. Método de Jacobi	3
2.3. Método de Gauss-Seidel	6
3. Resultados	6
3.1. Ecuación de calor	6
3.1.1. Problema 1a	7
3.1.2. Problema 1b	7
3.2. Método de Jacobi	8
3.2.1. Matriz 3x3	8
3.2.2. Matriz 125x125	8
3.3. Métodos Gauss-Seidel	8
3.3.1. Matriz 3x3	8
3.3.2. Matriz 125x125	8
4. Conclusiones	9
5. Compilación y ejecución de los programas	9
5.1. Ecuación de calor	9
5.1.1. Problema 1a	9
5.1.2. Problema 1b	9
5.2. Método de Jacobi	9
5.3. Métodos Gauss-Seidel	9

1. Introducción

1.1. Métodos iterativos

Un método iterativo es un método que va obteniendo aproximaciones a la solución de un problema. En un método iterativo repite un mismo proceso sobre la solución aproximada. Este método puede ser suspendido por medio de parámetros, ya sea un máximo de iteraciones o un factor de convergencia de la solución.

1.1.1. Método de Jácobi y Gauss-Seidel

Los métodos de Jácobi y Gauss-Seidel son métodos iterativos para encontrar la solución de sistemas de ecuaciones lineales. Los dos consisten en obtener una ecuación de recurrencia y proponer un vector solución inicial.

1.2. Ecuaciones diferenciales parciales

Se denomina a las ecuaciones diferenciales parciales (EDP) a aquellas ecuaciones que involucran derivadas parciales de una función desconocida con dos o más variables independientes. La mayoría de problemas físicos están descritos por EDP de segundo orden. El método analítico para resolver este tipo de ecuaciones es el método de variables separables.

En el caso de el método numérico, uno de los métodos para la solución de EDP es elementos finitos, este considera que el continuo se divide en un número finito de partes que se denominan como elementos. Un parámetro del método es el número de puntos característicos llamados nodos. Estos nodos son los puntos de unión de cada elemento adyacente.¹

1.2.1. Condición de frontera de Dirichlet

Las condiciones de frontera de Dirichlet son cantidades definidas en los extremos de una ecuación diferencial para obtener la solución particular.

1.2.2. Ecuación de transferencia de calor

La EDP empleada para modelar la transferencia de calor es la siguiente:

$$k\nabla^2 u - \frac{\partial^2 u}{\partial t^2} = 0$$

para el caso estacionario en una dimensión se tiene:

$$k \frac{\partial^2 u}{\partial x^2} + Q = 0 \tag{1}$$

La ecuación 1 es una ecuación diferencial ordinaria con coeficientes constantes.

2. Métodos

2.1. Ecuación de calor

Aplicando el método de diferencias finitas a la ecuación 1 obtenemos la ecuación 2.

$$\frac{k(u_{i+1} - 2u_i + u_{i-1}))}{\Delta x^2} + Q = 0 \quad (2)$$

Para el caso de cinco nodos se obtiene el sistema de ecuaciones señalado en la ecuación 3.

$$\begin{cases} u_0 + -2u_1 + u_2 = \frac{-Q\Delta x^2}{K} \\ u_1 + -2u_2 + u_3 = \frac{-Q\Delta x^2}{K} \\ u_2 + -2u_3 + u_4 = \frac{-Q\Delta x^2}{K} \end{cases} \quad (3)$$

El sistema de ecuaciones 3 puede ser descrito la ecuación matricial 4.

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} \frac{Q\Delta x^2}{k} + u_0 \\ \frac{Q\Delta x^2}{k} \\ \frac{Q\Delta x^2}{k} + u_4 \end{pmatrix} \quad (4)$$

Donde u_0 y u_4 son las condiciones de frontera del problema. Si expandemos el problema a n nodos, obtendremos que para describir el sistema como una ecuación matricial debemos seguir los siguientes parámetros. Para obtener la matriz del sistema se tiene que:

$$A = \begin{cases} 2 & \text{para } i = j \\ -1 & \text{para } |i - j| = 1 \\ 0 & \text{en otro lado} \end{cases} \quad B = \begin{cases} \frac{Q\Delta x^2}{k} + u_0 & \text{para } i = 1 \\ \frac{Q\Delta x^2}{k} + u_n & \text{para } i = n - 1 \\ \frac{Q\Delta x^2}{k} & \text{en otro lado} \end{cases} \quad (5)$$

donde $i, j \in \{1, 2, 3, \dots, n - 1\}$. El sistema de la ecuación 5 describe una matriz tridiagonal, por lo que se obtara por resolver el sistema usando la factorización por Cholesky.

El algoritmo que resuelve esta EDP es el siguiente:

```
1 // inputs: k, Q, L, u_0, u_n, n
2 // output: solutions
3 matrix, results = create_matrix_system(Q, k, l, n)
4 L = Cholesky(matrix)
5 solutions_y = solve_triangular_inferior(L, results)
6 solutions_x = solve_triangular_superior(LT, solutions_y)
```

La funciones `Cholesky`, `solve_triangular_inferior` y `solve_triangular_superior` fueron creadas en tareas anteriores. La función `create_matrix_system` aplica la ecuación 5 para obtener la ecuación matricial.

2.2. Método de Jacobi

Sea el sistema de ecuaciones lineales $Ax = b$, donde A es la matriz de coeficientes, x es el vector de incógnitas y b el vector de términos independientes. Se propone que A puede ser escrito como la suma dos matrices, tales que una contiene ceros en un diagonal y una matriz diagonal. Entonces, la ecuación matricial se convierte en:

$$Dx + Rx = b$$

despejando Dx , se obtiene la ecuación 6:

$$Dx = b - Rx \quad (6)$$

Multiplicando la ecuación 6 por D^{-1} de lado izquierdo la ecuación 7:

$$x = D^{-1}(b - Rx) \quad (7)$$

donde

$$D^{-1} = \begin{pmatrix} \frac{1}{a_{11}} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{a_{33}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{1}{a_{nn}} \end{pmatrix}$$

La cual, escribiendo de forma recursiva resulta en la ecuación 8.

$$x^{(k+1)} = D^{-1}(b - Rx^{(k)}) \quad k = 0, 1, 2, \dots, n \quad (8)$$

Desarrollando la ecuación 8 para obtener una ecuación para obtener la solución $x_i^{(k+1)}$, se obtiene la ecuación 9.

$$x_i^{(k+1)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}}{a_{ii}} \quad (9)$$

La ecuación 9 se implemento en el programa `solution.h`, en la función `solve_jabobi`. El algoritmo es el siguiente:

```

1  // inputs: matrix
2  // output: x
3  for (int i = 0; i < n; i++)
4  {
5      sum = 0;
6      for (int j = 0; j < n; j++)
7      {
8          if (j != i)
9          {
10             sum += m_ij * x_j;
11         }
12     }
13     x_i = (b_i - sum) / m_ii;
14 }
```

La tolerancia de este metodo se definió usando una norma (θ), la cual esta descrita en la ecuación 10.

$$\theta = \sqrt{\sum_{i=1}^{n-1} (x_i^{(k+1)} - x_i^{(k)})^2} \quad (10)$$

El método seguirá hasta que $\theta < 10^{-6}$. El método de Jácobi se asegura que converge cuando la matriz A es diagonal dominante. En el caso que A no sea diagonal dominante no se asegura obtener una solución al sistema. Se dice que una matriz diagonal dominante es aquella que los elementos de su diagonal cumplen la condición de la ecuación

$$|a_{ii}| \geq \sum_{i \neq j} |a_{ij}| \quad (11)$$

Al inicio del programa se comprueba si la matriz introducida es diagonal convergente. Si no lo es intentará ordenar los elementos de tal manera que el elemento con valor absoluto mayor se encuentre en la diagonal de la matriz. Por ejemplo, si se tiene la matriz aumentada:

$$A = \left(\begin{array}{ccc|c} -1 & 3 & 1 & 2 \\ 7 & 1 & 4 & 5 \\ 2 & -5 & 5 & -1 \end{array} \right)$$

Se buscará el valor mas grande en la matriz para organizarlo en la posición a_{11} , esto realizando un intercambio de filas y columnas.

$$A = \left(\begin{array}{ccc|c} 7 & 1 & 4 & 5 \\ -1 & 3 & 1 & 2 \\ 2 & 2 & 5 & -1 \end{array} \right)$$

Para elemento a_{22} , se buscara el elemento con valor absoluto mayor que tal que su número de fila y columna es mayor a 2. En este ejemplo, la siguiente organización sería la siguiente:

$$A = \left(\begin{array}{ccc|c} 7 & 4 & 1 & 5 \\ 2 & 5 & 2 & -1 \\ -1 & 1 & 3 & 2 \end{array} \right)$$

Al llegar al ultimo del elemento de la diagonal de la matriz no se realizará ningún cambio. Este procedimiento puede no obtener una matriz diagonal dominante. Sin embargo, se comprueba que algunos sistemas que no convergían, al realizar este proceso se llega a su solución. Este proceso de ordenamiento se encuentra contenido en el archivo [dominant_diagonal.h](#).

El algoritmo que sigue es el siguiente:

```

1  // inputs: matrix
2  // output: matrix_ordenada
3  for (int i = 0; i < n; i++)
4  {
5      // Se supone que el maximo ya se encuentra en la diagonal
6      max = fabs(m_ii);
7      i_max = i;
8      j_max = i;
9      for (int j = i; j < n; j++)
10     {
11         for (int k = i; k < n; k++)
12         {
13             if (fabs(m_ij) > max)
14             {
15                 max = fabs(m_ij);
16                 i_max = j;
17                 j_max = k;
18             }
19         }
20     }
21 }
```

```

19         }
20     }
21     if (i != i_max)
22     {
23         change_columns(matrix, max_position);
24     }
25     if (i != j_max)
26     {
27         change_rows(matrix, results);
28     }
29 }
30 is_diagonal_dominant_matrix(matrix);

```

2.3. Método de Gauss-Seidel

El método de Gauss-Seidel es semejante al método de Jácobi. El método de Jácobi usa el valor de las incógnitas para determinar una nueva aproximación en cada iteración, en cambio, el método de Gauss-Seidel calcula las nuevas aproximaciones con los valores calculados en la misma iteración. Entonces, tomando esto en cuenta, la ecuación recursiva para el método de Gauss-Seidel es la siguiente:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij}x_j^{(k)}}{a_{ii}} \quad (12)$$

La convergencia de la ecuación 12 es definida con la ecuación usada para el método de Jacobi (ecuación 10). El algoritmo implementado para este método se encuentra en el archivo [solution.h](#) en la función [solve_seidel](#), el cual es el siguiente:

```

1  // inputs: matrix
2  // output: x
3  for (int i = 0; i < n; i++)
4  {
5      sum = 0;
6      for (int j = 0; j < i; j++)
7      {
8          sum += m_ij * x_j;
9      }
10     x_i = (b_i - sum) / m_ii;
11     sum = 0;
12     for (int j = i + 1; j < n; j++)
13     {
14         sum += m_ij * x_j;
15     }
16     x_i = x_i - sum / m_ii;
17 }

```

3. Resultados

3.1. Ecuación de calor

Los programas y resultados del problema 1a y 1b se encuentran en la carpeta [Problema_1](#). Cada inciso tiene su propia carpeta llamadas [Problema_1a](#) y [Problema_1b](#).

3.1.1. Problema 1a

- Resolver la ecuación de calor considerando $\{Q = 3, K = 5, u_0 = 10, u_n = 20, n = 4, L = 1\}$, L es la longitud de la barra.

Para este caso se puede visualizar el sistema de ecuaciones, el cual es el siguiente:

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 10.037500 \\ 0.037500 \\ 20.037500 \end{pmatrix} \quad (13)$$

La solución del sistema 13 es el siguiente:

$$\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 12.556250 \\ 15.075000 \\ 17.556250 \\ 20 \end{pmatrix} \quad (14)$$

- Resolver la ecuación de calor considerando $\{Q = 3, K = 5, u_0 = 10, u_n = 20, n = 100, L = 1\}$.

El sistema de ecuaciones para el problema es el siguiente:

$$\begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ 0 & 0 & -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{98} \\ u_{99} \end{pmatrix} = \begin{pmatrix} 10.000060 \\ 0.000060 \\ 0.000060 \\ 0.000060 \\ 0.000060 \\ 0.000060 \\ 20.000060 \end{pmatrix} \quad (15)$$

La solución del sistema de ecuaciones 15 se encuentra en el archivo [Solution_100.txt](#).

3.1.2. Problema 1b

Graficar la variación de temperatura u del nodo central contra el número de elementos $n \in \{10, 30, 50, 70, 100\}$ que equivale a $n + 1$ nodos.

Para este problema se creo un ciclo que repetirá el proceso del problema 1a. En este ciclo, el valor de n cambiará para obtener la solución del problema. Los archivos que contienen el resultado del nodo central y anterior al central de cada iteración son [results_1.csv](#) y [results_2.csv](#) respectivamente. Se creo un programa en python, el cual lee el archivo de resultados y realiza las gráficas 1 y 2.

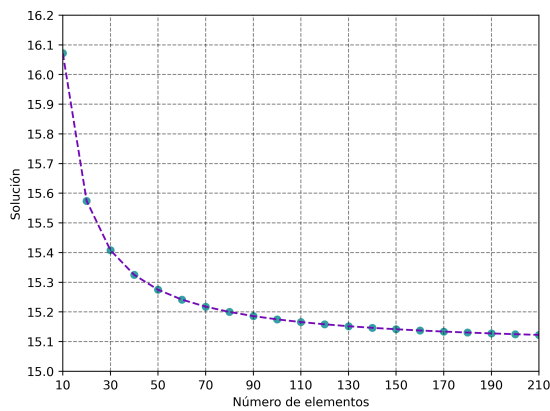


Figura 1: Resultado del nodo central $\left(\frac{n}{2}\right)$ para cada valor de n .

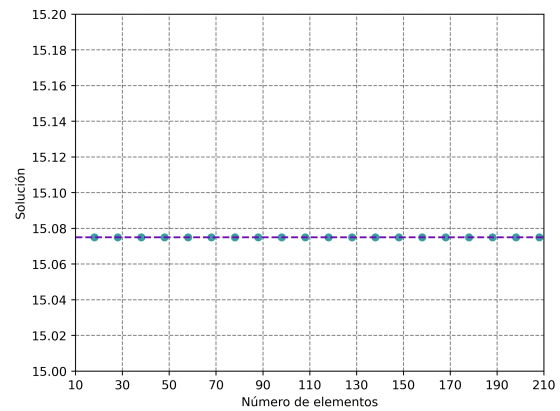


Figura 2: Resultado del nodo anterior al central $\left(\frac{n}{2} - 1\right)$ para cada valor de n .

3.2. Método de Jacobi

3.2.1. Matriz 3x3

La matriz introducida para este caso es la contenida en los archivos [M_sys_3x3.txt](#) y [V_sys_3x1.txt](#). El programa arroja la siguiente solución con siete iteraciones:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ -2.5 \\ 7 \end{pmatrix}$$

3.2.2. Matriz 125x125

La matriz introducida para este caso es la contenida en los archivos [M_sys_125x125.txt](#) y [V_sys_125x1.txt](#). El programa obtiene una solución con 1874 iteraciones. El output del programa se encuentra en el archivo [Solution_125.txt](#).

3.3. Métodos Gauss-Seidel

3.3.1. Matriz 3x3

La matriz introducida para este caso es la contenida en los archivos [M_sys_3x3.txt](#) y [V_sys_3x1.txt](#). El programa arroja la siguiente solución con cinco iteraciones:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3 \\ -2.5 \\ 7 \end{pmatrix}$$

3.3.2. Matriz 125x125

La matriz introducida para este caso es la contenida en los archivos [M_sys_125x125.txt](#) y [V_sys_125x1.txt](#). El programa obtiene una solución con 991 iteraciones. El output del programa se

encuentra en el archivo [Solution_125.txt](#).

4. Conclusiones

5. Compilación y ejecución de los programas

5.1. Ecuación de calor

5.1.1. Problema 1a

5.1.2. Problema 1b

5.2. Método de Jacobi

5.3. Métodos Gauss-Seidel

Referencias

¹ M. Acosta, C. R.de Coss. Solución numérica de ecuaciones diferenciales unidimensionales por el método de diferencias finitas. *Ingeniería*, 2016.

² Curtis Gerald. *Applied numerical analysis*. Pearson/Addison-Wesley, Boston, 2004.