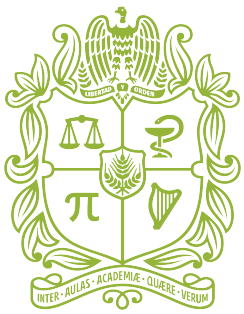

UNIVERSIDAD
NACIONAL
DE COLOMBIA

2023-1

Proyecto final
Solución de la ecuación de difusión con código paralelizado

Autor:

Kevin Cortés - kcortesr@unal.edu.co

Profesor:

William Fernando Oquendo Patiño

Índice

1. Introducción

El proyecto realizado para el segundo módulo del curso **introducción a la computación científica de alto rendimiento** es la implementación del método de volúmenes finitos (FVM por sus siglas en Inglés) para solucionar la ecuación de difusión en un problema de conducción de calor en una placa rectangular en estado estacionario. Adicionalmente, el código de la función de la malla y de la función de procesamiento se paralelizan haciendo uso de **OpenMP**.

Ahora bien, el flujo de trabajo llevado a cabo es:

- Discretización a mano de la ecuación de difusión.
- Programación de la malla.
- Programación de las propiedades físicas y del término fuente.
- Programación de las condiciones de frontera.
- Programación de la etapa de procesamiento: La solución del sistema de ecuaciones se realizó mediante el método iterativo de Gauss - Seidel.
- Programación de la etapa de posprocesamiento: Obtención de las gráficas que muestran los resultados.
- Análisis de los resutados de las métricas paralelas.

Cada vez que se obtuvo avances los archivos y casos fueron subidos a un repositorio de **GitHub** cuyo enlace es: `https://github.com/KevinCortesR/Proyecto_ICCAR.git`

1.1. Objetivos específicos

- Utilizar la herramienta **OpenMP** con el fin de paralelizar las etapas de creación de la malla y solución del problema de interés.
- Obtener gráficas de *speed-up* y de *parallel efficiency* utilizando la librería **chrono** para medir el *wall-clock time* y la función **std::clock_t** para medir el *CPU time*.
- Obtener gráficas del desempeño del código para diferentes grados de optimización.

Por otra parte, para comprobar el funcionamiento de los códigos realizados se hizo la implementación de un caso con solución analítica. Una vez comprobados, se solucionó el problema de interés y se analizaron los resultados.

2. Problema de interés

Se tiene una placa plana como se muestra a continuación:

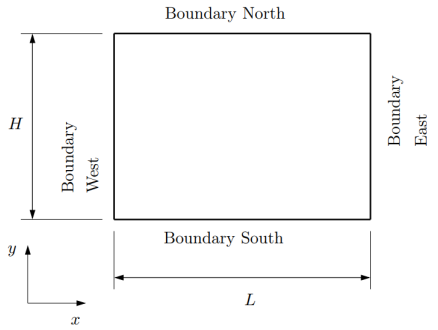

Figura 1: Geometría de la placa: $L = 3$ m y $H = 2$ m

Para esta placa se tienen las siguientes condiciones de frontera, conductividad témica y término fuente:

Frontera	Condición
Norte	$\frac{\partial T}{\partial y} = 0$ (adiabatica)
Sur	$T = -5 + 20\left(\frac{x}{L}\right)^2 + 20\sin\left(\frac{4\pi x}{L}\right)$
Oeste	$T = -5°\text{C}$
Este	$T = 15°\text{C}$
Conduct.	$\kappa = 2\left(2{,}5 - \frac{x}{L}\right)\text{W/m}\cdot°\text{C}$
Fuente	$B = 100$

La ecuación de difusión que rige el problema es la siguiente:

$$0 = \frac{\partial}{\partial x}\left(\kappa\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\kappa\frac{\partial T}{\partial y}\right) + B \tag{1}$$

2.1. Discretización

A continuación, se muestra la discretización la ecuación (1); aunque se deja en la forma genérica, es decir en términos de ϕ y Γ . En primera instancia, al integrar sobre el volumen se obtuvo:

$$0 = \int_V \left[\frac{\partial}{\partial x} \left(\Gamma \frac{\partial \phi}{\partial x} \right) + \frac{\partial}{\partial y} \left(\Gamma \frac{\partial \phi}{\partial y} \right) \right] dV + \int_V B dV \tag{2}$$

Para evitar lidiar con la segunda derivada, se hizo uso del teorema de la divergencia de Gauss para un campo vectorial arbitrario C:

$$\int_V (\nabla \cdot C)\, dV = \int_S (C \cdot \hat{n})\, dS \tag{3}$$

Aplicando el teorema a la ecuación (2) y suponiendo un término fuente B constante se tuvo:

$$0 = \int_S \left[\Gamma \frac{\partial \phi}{\partial x} \hat{n}_x + \Gamma \frac{\partial \phi}{\partial y} \hat{n}_y \right] dS + BV \tag{4}$$

Para una celda interior P, los vectores normales en función de sus celdas vecinas al norte, sur, este y oeste tomaron valores como se muestra en la siguiente figura:

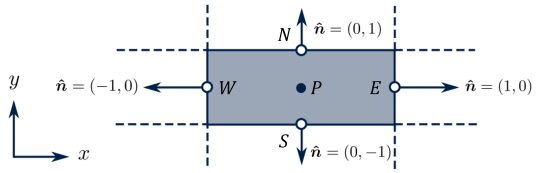

Figura 2: Vectores normales en direcciones norte (N), sur (S), este(E) y oeste (W) para una celda 2D

Teniendo lo anterior en mente y sabiendo que el término difusivo debe ser igual a la suma de los flujos en las fronteras con celdas vecinas ($\int_S \left[\Gamma \frac{\partial \phi}{\partial x_j} \hat{n}_j \right] dS = F_N + F_S + F_E + F_W$), pudo usarse el esquema de diferenciación central (CDS) para obtener:

$$F_e = \Gamma_e \frac{\phi_e - \phi_p}{x_E - x_P} \Delta y \quad ; \quad F_w = -\Gamma_w \frac{\phi_p - \phi_w}{x_P - x_W} \Delta y$$

$$F_n = \Gamma_n \frac{\phi_n - \phi_p}{y_N - y_P} \Delta x \quad ; \quad F_s = -\Gamma_s \frac{\phi_p - \phi_s}{y_P - y_s} \Delta x$$

La ecuación (4) fue reescrita de la siguiente manera:

$$\frac{\Gamma_e \Delta y}{x_E - x_P}\phi_e + \frac{\Gamma_w \Delta y}{x_P - x_W}\phi_w + \frac{\Gamma_n \Delta x}{y_N - y_P}\phi_n + \frac{\Gamma_s \Delta x}{y_P - y_S}\phi_s - \frac{\Gamma_e \Delta y}{x_E - x_P}\phi_p - \frac{\Gamma_w \Delta y}{x_P - x_W}\phi_p - \frac{\Gamma_n \Delta x}{y_N - y_P}\phi_p - \frac{\Gamma_s \Delta x}{y_P - y_S}\phi_p + q_\phi \Delta x \Delta y = 0 \quad (5)$$

Si se agrupa en coeficientes (a_i) para cada celda vecina, se reescribe la ecuación de la siguiente manera:

$$a_E\phi_e + a_W\phi_w + a_N\phi_n + a_S\phi_s - \underbrace{(a_E + a_W + a_N + a_S)}_{a_P}\phi_p = -q_\phi \Delta x \Delta y \quad (6)$$

La ecuación (6) corresponde a la discretización general cuando una celda está rodeada por vecinas en todas las direcciones, es decir que se trata de la discretización para las **celdas internas**.

Para las celdas que tienen una o más fronteras en su vecindad, se realizó la discretización correspondiente; a continuación se muestra para cada celda que tiene condiciones de frontera:

Para la **frontera norte**, en este caso tipo Neumann ($\partial T/\partial y = 0$), el coeficiente a_N no aporta ni en las celdas vecinas ni en el coeficiente a_P y el flujo establecido por la condición de frontera Neumann pasa a contribuir al lado conocido (en este caso es cero).

$$a_E\phi_e + a_W\phi_w + a_S\phi_s - \underbrace{(a_E + a_W + a_S)}_{a_{P1}}\phi_p = -q_\phi \Delta x \Delta y \quad (7)$$

En la **frontera sur** de tipo Dirichlet, el coeficinte a_S no contribuye únicamente en las celdas vecinas pero si contribuye en el lado conocido.

$$a_E\phi_e + a_W\phi_w + a_N\phi_n - \underbrace{(a_E + a_W + a_N + a_S)}_{a_P}\phi_p = -q_\phi \Delta x \Delta y - a_S\phi_s \quad (8)$$

Para la **frontera este**, que también es tipo Dirichlet, ocurre algo análogo al caso de la frontera sur pero con el coeficiente a_E.

$$a_W\phi_w + a_N\phi_n + a_S\phi_s - \underbrace{(a_E + a_W + a_N + a_S)}_{a_P}\phi_p = -q_\phi \Delta x \Delta y - a_E\phi_e \quad (9)$$

Y de igual forma ocurre con la **frontera oeste** y el coeficiente a_W.

$$a_E\phi_e + a_N\phi_n + a_S\phi_s - \underbrace{(a_E + a_W + a_N + a_S)}_{a_P}\phi_p = -q_\phi \Delta x \Delta y - a_W\phi_w \tag{10}$$

En las celdas de las esquinas se tienen dos condiciones de frontera simultáneamente, y es por ello que sus efectos se combinan. Para el caso de la **celda noroeste** se obtuvo:

$$a_E\phi_e + a_W\phi_w + a_S\phi_s - \underbrace{(a_E + a_W + a_S)}_{a_{P1}}\phi_p = -q_\phi \Delta x \Delta y - a_W\phi_w \tag{11}$$

En el caso de la **celda noreste** se logró:

$$a_W\phi_w + a_W\phi_w + a_S\phi_s - \underbrace{(a_E + a_W + a_S)}_{a_{P1}}\phi_p = -q_\phi \Delta x \Delta y - a_E\phi_e \tag{12}$$

En las dos celdas inferiores hay dos condiciones Dirichlet aplicadas. Para el caso de la **celda suroeste** se llegó a:

$$a_E\phi_e + a_N\phi_n - \underbrace{(a_E + a_W + a_N + a_S)}_{a_P}\phi_p = -q_\phi \Delta x \Delta y - a_S\phi_s - a_W\phi_w \tag{13}$$

Y finalmente para la **celda sureste**:

$$a_W\phi_w + a_N\phi_n - \underbrace{(a_E + a_W + a_N + a_S)}_{a_P}\phi_p = -q_\phi \Delta x \Delta y - a_S\phi_s - a_E\phi_e \tag{14}$$

Así pues las ecuaciones (6) a (14) corresponden a las discretizaciones para los nueve tipos de celdas que hay en el dominio computacional (internas, con una y con dos condiciones de frontera).

3. Problema con solución analítica

El problema con solución analítica se lleva a cabo en el mismo dominio, pero con condiciones de frontera diferentes, por lo que el código y la discretización cambia ligeramente.

Frontera	Condición
Norte	$T = 150°\text{C}$
Sur	$T = 15°\text{C}$
Oeste	$T = 15°\text{C}$
Este	$T = 15°\text{C}$
Conduct.	$\kappa = 1$
Fuente	$B = 0$

Este problema tiene la solución analítica dada por:

$$T_{adimensional}(x,y) = \frac{2}{\pi} \cdot \sum_{n_{impar}}^{\infty} \frac{sinh(n\pi x/H)}{sinh(n\pi L/H)} \cdot sin(\frac{n\pi y}{H}) \tag{15}$$

Para darle dimensiones de °C se realizó el siguiente procedimiento:

$$T(x,y) = \frac{T_{max} - T_{min}}{T^{max}_{adimensional}} \cdot T_{adimensional}(x,y) + T_{min} \tag{16}$$

4. Método iterativo Gauss-Seidel

Se escogió el método iterativo de Gauss-Seidel para la implementación en código ya que es intuitivo y fácil de escribir en código.

El método tiene el siguiente flujo para resolver el sistema matricial $[A][\phi] = [Q]$ con n incógnitas y n ecuaciones:

- Hacer una estimación inicial para todas las incógnitas ϕ_i^0.
- Resolver la primera ecuación del sistema de ecuaciones para obtener ϕ_1^1.
- Resolver la segunda ecuación del sistema de ecuaciones teniendo en cuenta a ϕ_1^1 para llegar a un valor de ϕ_2^1
- Resolver la ecuación m-ésima teniendo en cuenta los nuevos valores de las $m-1$ incógnitas anteriores hasta completar las n ecuaciones.
- Calcular el residual de la siguiente forma:

$$R = \sum^{n} |q_\phi - (a_P \phi_P + \sum_{j}^{vecinos} a_j \phi_j)| \tag{17}$$

- Normalizar el residual dividiéndolo entre un flujo característico, así

$$F = \sum^{n} |a_P \phi_P| \tag{18}$$

- Comparando el residual normalizado con el criterio de convergencia ϵ se obtiene:

$$\frac{R}{F} \leq \epsilon \tag{19}$$

  Si esta desigualdad es verdadera no realice más iteraciones pues el método convergió, si es falsa vuelva al segundo item de esta lista ya que el método no ha convergido.

Como métodos de salida del algoritmo se tiene:

- Que el residual normalizado sea menor al criterio de convergencia.
- Que la diferencia entre el residual normalizado de la iteración anterior y el residual normalizado de la iteración actual sea menor que 1×10^{-9} y mayor que 0.
- Que al haber pasado 1000 iteraciones y si el residual normalizado de la iteración anterior es menor que el de la iteración actual.

  Dado que se decidió utilizar los métodos de salida del método anteriormente mencionados no se usó ninguna librería en particular para el mismo.

5. Estado actual de los Códigos

Al momento se tienen completos lo códigos del problema con solución analítica y del problema de interés con solución en serie en los cuales se utiliza la librería **Vector**.

Por otra parte, en el caso del problema de interés paralelizado en las funciones que son objetivo de la paralelización se utiliza la librería **Eigen** y para obtener *CPU time* y *Wall clock time* se utilizan la función **std::clock_t** y la librería **Chrono** respectivamente. No obstante, hace falta implementar las líneas de código que contienen *#pragma*.

Al momento, los objetivos se mantienen puesto que sólo hace falta implementar las líneas de paralelización.

6. Resultados

6.1. Caso con solución analítica

En esta parte se van a mostrar los resultados obtenidos de las soluciones analítica y numérica del problema de conducción de calor en la placa rectangular.

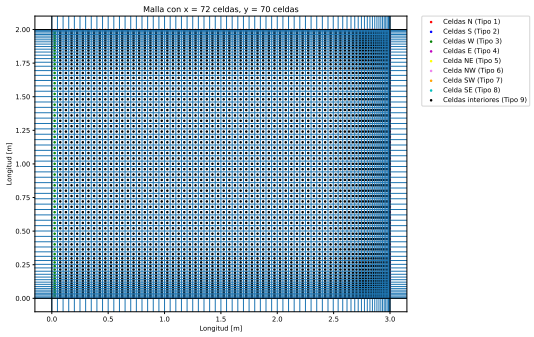

Figura 3: Malla utilizada para el caso con solución analítica

En la figura 3 se puede ver la malla utilizada para resolver este problema, se puede apreciar que la malla tiene un refinamiento hacia la frontera este, la frontera norte y sur, ya que allí en las esquinas nor-este y sur-este son los lugares donde se concentran los mayores gradientes de temperatura y se acumulan las líneas isotermas.

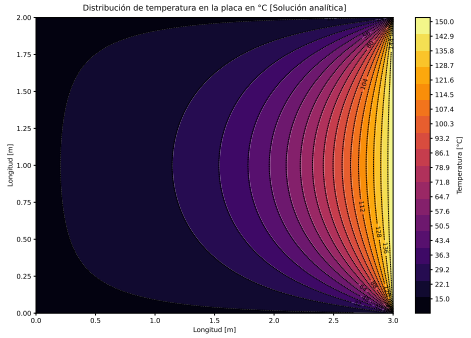

Figura 4: Distribución de temperatura con la solución analítica

Ahora bien, en la imagen 4 se ve la distribución de temperatura en la placa con la solución analítica; no obstante, hay que ser consiente de que esta tiene el error de truncamiento debido a que no se podía tener los infinitos términos de la serie, por lo que sólo se usaron 75 términos.

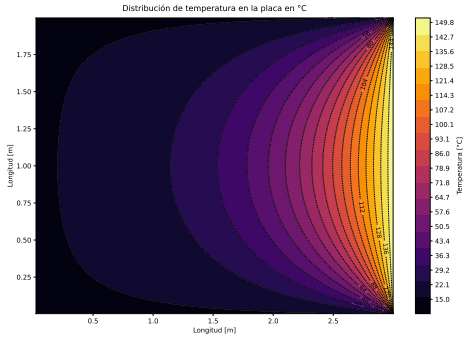

Figura 5: Distribución de temperatura con la solución numérica

Por otra parte, en la figura 5 se puede observar la distribución de temperatura en la placa con la solución numérica.

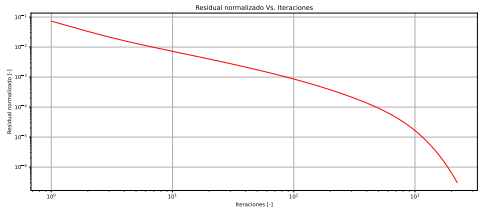

Figura 6: Residual normalizado de la solución numérica del caso con solución analítica

En este problema, el residual normalizado tuvo el comportamiento expuesto en la figura 6, alcanzando las 2226 iteraciones con un residual normalizado de $3{,}081 \cdot 10^{-7}$. Por lo tanto, el problema no alcanzó el criterio de convergencia, pero la curva llegó a cambiar menos que $1 \cdot 10^{-9}$ y salió del método iterativo.

6.2. Caso de interés

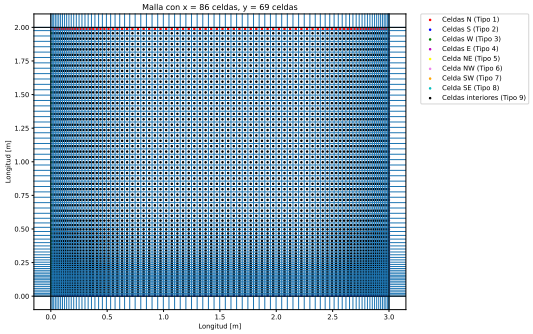

Figura 7: Malla utilizada para el caso de interés

En la figura 7 se puede apreciar la malla utilizada para resolver este caso. Es posible ver el refinamiento que hay hacia las esquinas inferiores y en general en la frontera sur, ya que en esa región es donde está la frontera de tipo Dirchlet que cambia con la función $sin(x)$.

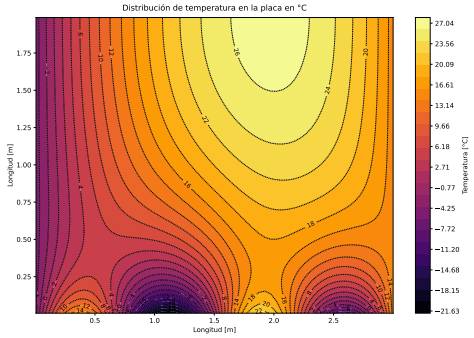

Figura 8: Distribución de temperatura con la solución numérica

Así pues, en la figura 8 se puede observar la distribución de temperatura en la placa del problema de interés. Se aprecia cómo se aglomeran las isotermas en la frontera sur y las esquinas sur-este y sur-oeste.

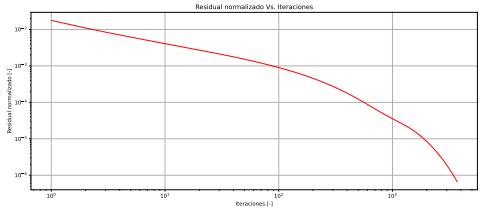

Figura 9: Residual normalizado de la solución numérica del caso de interés

En este caso, el residual normalizado se comportó de la forma en que muestra la figura 9, alcanzando las 3701 iteraciones con un residual normalizado de $6{,}667 \cdot 10^{-7}$. Entonces, la curva llegó a tener una variación menor que $1 \cdot 10^{-9}$, por lo que salió del ciclo *while* de la función del método iterativo de Gauss-Seidel.

6.3. Caso de interés paralelizado

Las pruebas en este caso se han llevado a cabo con una malla de 25 celdas en x y 20 celdas en y.

Tiempos obtenidos para la función Malla.cpp		
Num Threads	Wall clock time [s]	CPU time [s]
1	0.000298705	0.000298
2	0.000283928	0.000287
3	0.000289217	0.000292
4	0.00027976	0.000282
5	0.000278377	0.000282
6	0.000275903	0.000279
7	0.000277606	0.00028
8	0.000272146	0.000275
9	0.000288416	0.000291
10	0.000276424	0.000279
11	0.000305148	0.00031
12	0.000282675	0.000285
13	0.000277786	0.00028
14	0.000291212	0.000295
15	0.000325246	0.000324
16	0.000289238	0.000292

Tiempos obtenidos para la función Procesamiento.cpp		
Num Threads	Wall clock time [s]	CPU time [s]
1	2.19908	2.19757
2	2.20235	2.20111
3	2.15537	2.1542
4	2.16737	2.16602
5	2.22489	2.22368
6	2.18364	2.18222
7	2.19653	2.19536
8	2.20293	2.20141
9	2.19433	2.193
10	2.22594	2.22479
11	2.21002	2.20844
12	2.18639	2.18527
13	2.20686	2.2055
14	2.21453	2.21278
15	2.16336	2.16197
16	2.16896	2.16782

En las tablas anteriores se puede apreciar que el proceso de paralelización no funciona en el momento, lo cual muestra que faltan los comandos *#pragma*.

Referencias

[1] Benavides, A. *CFD Finite Volume Method*, Computational Fluid Dynamics, 2023.

[2] Versteeg, H & Malalasekera, W. *An Introduction to Computational Fluid Dynamics*, Second edition, 2007.

[3] Wimshurst, A. *Lecture Notes, Computational Fluid Dynamics*, Fundamentals Course 2, 2020.

[4] Pletcher, R. H., Tannehill, J. C., & Anderson, D. *Computational fluid mechanics and heat transfer*, 2012.

[5] Minkowycz, W. J., Sparrow, E. M., Schneider, G. E., & Pletcher, R. H. *Handbook of numerical heat transfer*, 1988.

[6] "C++ variables de tipo vector," Inicio. [Online]. Available: `https://aprende.olimpiada-informatica.org/cpp-vector`.

[7] "2D vector in C++ with user defined size," GeeksforGeeks, 10-Jan-2023. [Online]. Available: `https://www.geeksforgeeks.org/2d-vector-in-cpp-with-user-defined-size/`.

[8] D. Krishna, "Different ways to remove elements from vector in C++ STL," OpenGenus IQ: Computing Expertise & Legacy, 26-Oct-2019. [Online]. Available: `https://iq.opengenus.org/ways-to-remove-elements-from-vector-cpp/`.

[9] "Reading in .DAT file, line by line," C++ forum. [Online]. Available: `https://cplusplus.com/forum/beginner/117331/`.

[10] "Modules and header files," Eigen. [Online]. Available: `https://eigen.tuxfamily.org/dox/group__QuickRefPage.html`.

[11] Barney, B, "OpenMP," LLNL HPC Tutorials. [Online]. Available: `https://hpc-tutorials.llnl.gov/openmp/`.