Proyecto 3: Resolución de ecuaciones diferenciales parciales

David Badilla Sánchez y Jairo Méndez Martínez

***Abstract*— En este artículo se explicará la implementación del método de Liebmann para la solución de una ecuación diferencial parcial (EDP) elíptica, para estimar la distribución de temperatura y el flujo calorifíco en una placa cuadrada, con calor aplicado a sus bordes. Además se explica la utilidad de la bibiblioteca OpenMP para la paralelización y así disminuir el tiempo de ejecución del programa implementado.**

I. INTRODUCCIÓN

Las EDP son el corazón de los análisis computacionales o de simulaciones de sistemas físicos continuos; como fluidos, campos electromagneticos, el cuerpo humano y entre otros[1]. Para este proyecto se realizó la implementación de un método para la resolución de ecuaciones diferenciales parciales. El orden de una EDP es el de la derivada parcial de mayor orden en la ecuación; se dice que la EDP es lineal si los coeficientes de las derivadas solo dependen de las variables independientes[2]. Para el caso de este proyecto se trabaja con una ecuación diferencial lineal de segundo orden:

$$A\frac{\partial^2 T}{\partial x^2} + B\frac{\partial^2 T}{\partial xy} + C\frac{\partial^2 T}{\partial y^2} + D = 0 \tag{1}$$

donde A,B,C son funciones de x e y, y D es una función de x, y, T, $\partial d/\partial x$ y $\partial d/\partial y$. En esta ocasión se soluciona una EDP elíptica, y se decriben por la siguiente condición:

$$B^2 - 4AC < 0 \tag{2}$$

por lo que se trabaja con la ecuación de Laplace:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{3}$$

En general las ecuaciones elípticas se utilizan para caracterizar sistemas en estado estacionario (no hay derivadas respecto al tiempo)[2]. Y es por medio de la Eq. 3 que se pueden modelar problemas relacionados con el "potencial" de una variable desconocida, en esta ocasión cómo se distribuye la temperatura en una placa caliente que tiene bordes en contacto con condiciones térmicas dadas.

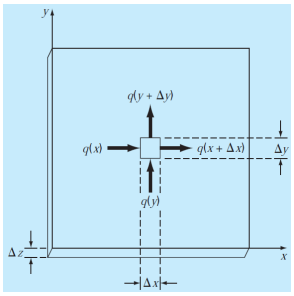

Fig. 1. Diferencial de flujo calorífico en placa recatngular de $\triangle z$[2]

De acuerdo a la figura 1 se obtiene lo que es llamada la ecuación de la conservación de la energía:

$$-\frac{\partial q}{\partial x} - \frac{\partial q}{\partial y} = 0 \tag{4}$$

de la cual con las condiciones fronteras, debe reformularse la ecuación 4, usando la ley de Fourier de conducción de calor:

$$q_i = k^{'}\frac{\partial T}{\partial i} \tag{5}$$

La ecuación de Fourier específica que el flujo de calor perpendicular al eje i es proporcional al gradiente de la temperatura en dirección i[2]. Y sustituyendo esta ley de Fourier en la ecuación diferencial 4 resulta en la ecuación de Laplace.

La implementación de la solución requería utilizar la biblioteca openMP. La API de OpenMP admite la programación paralela de memoria compartida multiplataforma en C/C ++ y Fortran. La API de OpenMP define un modelo portátil y escalable con una interfaz simple y flexible para desarrollar aplicaciones paralelas en plataformas desde el escritorio al superordenador[3].

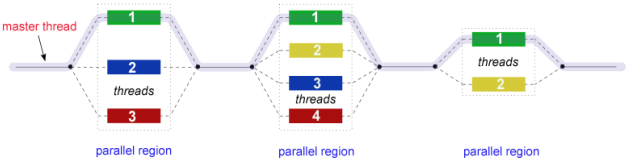

Fig. 2. Modelo Fork-Join de OpenMP[4]

Todos los programas OpenMP comienzan como un solo proceso: el subproceso maestro. El subproceso maestro se ejecuta secuencialmente hasta que se encuentre la primera construcción de la región paralela [4]. En el *FORK* el hilo maestro crea entonces un equipo de hilos paralelos, después, las declaraciones en el programa que están encerradas por la construcción de la región paralela se ejecutan entonces en paralelo entre los diversos hilos de la máquina. Luego, el *JOIN* es cuando los hilos de equipo completan las sentencias en la construcción de la región paralela, se sincronizan y terminan, dejando sólo el subproceso maestro. El número de regiones paralelas y los hilos que las componen son arbitrarios[4].

II. IMPLEMENTACIÓN DE LA SOLUCIÓN

II-A. *Método de Liebmann*

Para la resolución del método de Liebmann se resuelve la EDP elítica de acuerdo a la siguiente figura:

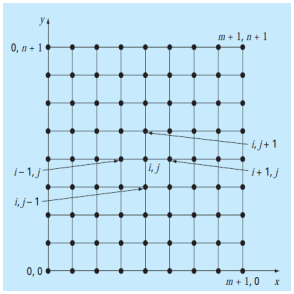

Fig. 3. Red de solución de diferencias finitas para una PDE elíptica[2]

A esta figura se le aplica la condición de frontera de Dirichlet, la cual consiste en tomar el calor aplicado en los bordes, y darle este valor a los (0,0...n), (m,0...n), (0...m,0) y (0...m,n). Cabe recalcar que la matriz utilizada está invertida según la figura 3, es decir, (0,0)..(0,n-1) es arriba y no abajo El método de Liebmann se caracteriza por la siguiente ecuación:

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4} \tag{6}$$

Puesto que las matrices obtenidas son diagonalmente dominantes, el procedimiento converge a una solución estable[2]. La ecuación 6 se realiza para i = 1,...,n y para j = 1,...,m. Y se itera este método hasta alcanzar un error deseado indicado por el usuario, y se calcula mediante la siguiente fórmula:

$$|e_{i,j}| = \left(\frac{{T_{i,j}}^{actual} - {T_{i,j}}^{anterior}}{{T_{i,j}}^{actual}}\right) * 100\,\% \tag{7}$$

El error también se puede obtener de la diferencia entre la norma de la matriz actual y la anterior. Cuando una Para el calculo de temperaturas se define un rango de 0 a 400 grados Celcius. Las ecuaciones utilizadas para partes aisladas fueron las indicadas en [1]. Para la parte de abajo por ejemplo:

$$T_{n-1,j} = \frac{(T_{n-1,j-1} + T_{n-1,j+1} + 2 * T_{n-2,j})}{4} \tag{8}$$

Para indicar que una pared es aislada se le indica -1 en su valor

II-B. *Flujo calorífico*

El flujo se calculó por medio de las siguientes ecuaciones:

$$q_x = k' \frac{T_{i+1,j} - T_{i-1,j}}{2dx} \tag{9}$$

$$q_y = k' \frac{T_{i,j+1} - T_{i,j-1}}{2dy} \tag{10}$$

en donde se asume que k' es igual a 1, y que dx = dy = 1.

II-C. *OpenMP*

Básicamente para la parelilización se realizó el mismo algoritmo para el método de Liebman, solo que en su calculo principal (Eq. 6), se realiza con #pragma omp parallel { #pragma omp for } , en donde el #pragma omp for es utilizado para recorrer la matriz con los *"fors"* requeridos.

II-D. *Medición de tiempos*

Para la medición de tiempo se usó la librería de OpenMP omp.h, y se mide el tiempo en segundos con el método omp_get_wtime(). Para cada una de las formas (serial o paralelo), se obtiene el tiempo con el que se inicia la ejecución, luego se ejecuta 50 veces, y se mide el tiempo final; (final-inicial)/50 da el resultado que duró ejecutandose el método. La medición de tiempo se realizó antes de la implementación de evaluar el error con un umbral. ¿Por qué?, bueno básicamente porque en implementación para verificar el error se crea un loop infinito (while(true)), y se sale del loop hasta que el error alcanze el umbral indicado. En OpenMP esta implementación se complicó, por lo cual la información en la figura 4 se realizó con un $O(n^3)$, con tres *"fors"* anidados, esto obviamente deja un error en la matriz de solución.

II-E. *Graficación*

Para realizar la graficación de el mapa de calor y flujo se utilizó la biblioteca Matplotlib de Python. Mediante el uso del módulo de *"Histogram2D"* y *"Quiver"*, fue posible implementar fácilmente el mapa de calor y flujo respectivamente. La comunicación entre nuestra aplicación de c++ y python se realizó mediante Python.h, que permite uitlizar código de python de manera embebida en c++.

III. RESULTADOS

. Para la sección de resultados se decidió hacer dos análisis, la diferencia de tiempos entre la ejecución serial y paralela, asimismo como averiguar la ecuación que describe al tiempo de acuerdo al tamaño de la matrix NxN. Además se pretende observar la distribución de temperatura y el flujo calorífico aplicando distintas condiciones a los bordes.

III-A. *Tiempos*

Tipo de Ejecución	n	Tiempo (ms)				T promedio(ms)
		t1	t2	t3	t4	
Serie	20	0,733	0,738	0,516	0,790	0,69
	40	3,288	3,135	3,112	3,307	3,21
	80	25,816	26,051	26,253	25,450	25,89
	160	220,772	215,910	211,084	230,065	219,46
Paralelo	20	0,358	0,360	0,532	0,282	0,38
	40	2,061	2,114	2,105	2,176	2,11
	80	16,696	16,242	16,847	16,441	16,56
	160	148,498	138,872	144,743	148,498	145,15

Fig. 4. Pruebas de tiempo de ejecución

La función del tiempo con respecto a n para una ejecución serial se define por la siguiente ecuación:

$$T(n) = 0{,}0141n^2 - 0{,}9976n + 16{,}996[ms] \tag{11}$$

con un R^2 = 0.9994

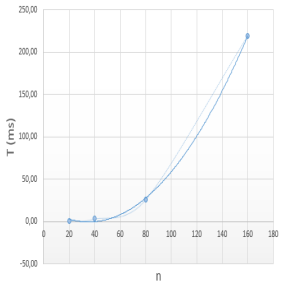

Fig. 5. T vs N en ejecución serial

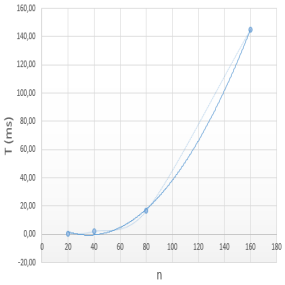

Fig. 6. T vs N en ejecución en paralelo

La función del tiempo con respecto a n para una ejecución en paralelo se define por la siguiente ecuación:

$$T(n) = 0{,}0094n^2 - 0{,}6784n + 11{,}61[ms] \qquad (12)$$

con un R^2 = 0.9993.

III-B. Distribución de temperatura y flujo calorífico

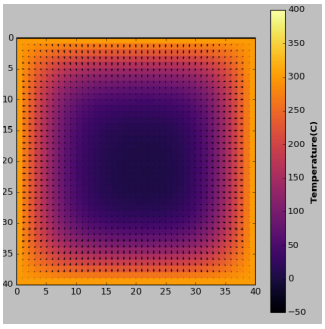

Fig. 7. Distribución de temperatura con error de 50 300x300x300x300

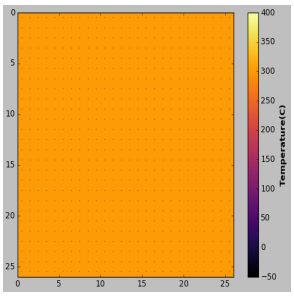

Fig. 8. Distribución de temperatura con error de 0.00001 300x300x0x0

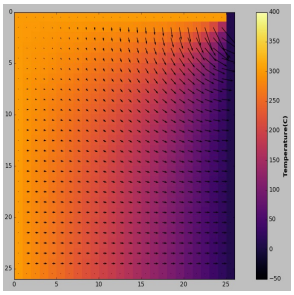

Fig. 9. Distribución de temperatura con pared aislada -1x300x300x10

IV. ANÁLISIS DE RESULTADOS

En al figura 7,8 y 9, podemos encontrar dos ejemplos generados por nuestra aplicación. En ellos se puede apreciar el comportamiento del flujo del calor mediante flechas que nos indican la dirección del mismo. El comportamiento normal es que el flujo tenga una dirección que se dirija de la zona más caliente a la de menor calor. Es posible escoger dos tipos de escala para representar el calor, una la cual tiene valores fijos que van desde los 0 hasta los 400 grados, y otra que se adapta del menor al mayor punto del mapa de calor.

Con respecto a la implementación de la paralelización, según las ecuaciones 11 y 12, se va a obtener un menor tiempo para el algoritmo paralelizado que al de la implmentación normal. Para valores de n muy pequeños, la diferencia va a ser casi nula, pero si usamos valores de n muy grandes vamos a encontrar una gran reducción en el tiempo del algortimo paralelizado con respecto a la implementación normal.

V. CONCLUSIONES

Mediante el método de Liebman, fue posible la implmentación de un mapa de calor la cual se comporta de manera correcta según lo observado en las figuras 7 y 8, al igual que se flujo de calor el cual debe tener una dirección de propagación de una zona donde hay mayor calor a una donde hay menor. Se comprueba, mediante los resultados obtenidos de la figura 5 y 6, que fue posible disminuir el tiempo de ejecución de el algoritmo mediante el uso de OpenMP.

REFERENCES

[1] William H. Press, Saul A. Teukolsky, William T. Vetterling y Brian P. Flannery. Numerical recipes. the art of scientific computing. Cambridge University Press, Segunda edición, 2002.

[2] Steven C. Chapra y Razmond P. Canale. Métodos numéricos para ingenieros. McGraw Hill, Mexico, sexta edición, 2011.

[3] R. Friedman, "Home - OpenMP", OpenMP, 2017. [Online]. Disponible en : http://www.openmp.org/. [Accedido: 30- Mayo- 2017].

[4] [2].ºpenMP", Computing.llnl.gov, 2017. [Online]. Disponible en: https://computing.llnl.gov/tutorials/openMP/. [Accedido: 30- Mayo- 2017].