Paralelización de la Ecuación de Calor en 2D

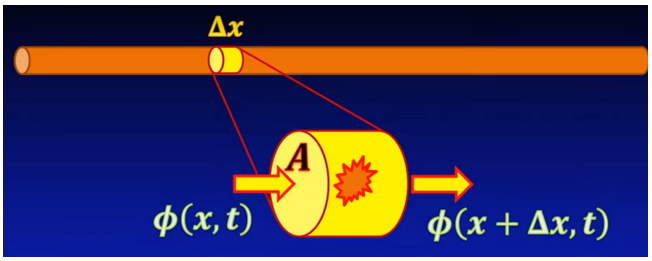
Baruc Samuel Cabrera García

22 de mayo de 2023

1. Deducción de la Ecuación de Calor

Primero, supongamos que tenemos una varilla de metal (esto para ejemplificar el caso unidimensional). En este caso, hay que hacer mención de la función de flujo de energía $\phi(x,t)$, la cual define la cantidad de energía térmica que fluye de un sentido a otro, dependiendo de su posición y tiempo.

Por simplicidad, supondremos que $\phi > 0$ si la energía fluye al extremo derecho de la varilla, y $\phi < 0$ en el otro caso.

Figura 1: 

Luego, definimos también la función $E(x,t)$, la cual es la energía térmica generada por unidad de volumen y tiempo. Esta función se define sobre un pedazo de varilla de longitud $\triangle x$. Así, expresamos la energía térmica por unidad de tiempo como

$$A \cdot \phi(x,t) - A \cdot \phi(x + \triangle x, t) + A \cdot \triangle x \cdot E(x,t),$$

donde A es el área de los extremos de la varilla, y $A \cdot \triangle x$ es el volumen del pedazo de varilla a considerar. Luego, considerando la propiedad

$$\triangle Q = \rho \cdot A \cdot \triangle x \cdot c \cdot \triangle u,$$

donde $\triangle Q$ representa el cambio en la energía térmica, ρ la densidad, c el calor específico y $\triangle u$ el cambio en la temperatura. Entonces al dividir por $\triangle t$ y tender dicho valor a cero, tenemos que

$$\frac{\partial Q}{\partial t} = c\rho A \cdot \triangle x \cdot \frac{\partial u}{\partial t}.$$

Por lo que se tiene que

$$\begin{aligned} c\rho A \cdot \triangle x \cdot \frac{\partial u}{\partial t} &= A \cdot \phi(x,t) - A \cdot \phi(x + \triangle x, t) + A \cdot \triangle x \cdot E(x,t), \\ \Rightarrow c\rho \triangle x \cdot \frac{\partial u}{\partial t} &= \phi(x,t) - \phi(x + \triangle x, t) + \triangle x \cdot E(x,t), \\ \Rightarrow c\rho \triangle x \cdot \frac{\partial u}{\partial t} &= -\frac{\phi(x + \triangle x, t) - \phi(x,t)}{\triangle x} \cdot \triangle x + \triangle x \cdot E(x,t). \end{aligned}$$

Nótese que ahora, el área es irrelevante, por lo que podemos considerar a la varilla infinitamente delgada, es decir, una sola dimensión.

Con base en la expresión anterior, para cualquier intervalo $[a,b]$ se tiene que

$$\int_a^b c\rho \frac{\partial u}{\partial t} dx = -\int_a^b \frac{\partial \phi}{\partial x} dx + \int_a^b E(x,t) dx,$$

$$\Rightarrow \int_a^b \left[c\rho \frac{\partial u}{\partial t} + \frac{\partial \phi}{\partial x} - E(x,t) \right] dx = 0,$$

$$\Rightarrow c\rho \frac{\partial u}{\partial t} + \frac{\partial \phi}{\partial x} - E(x,t) = 0. \qquad \text{Intervalo aleatorio.}$$

Ahora, consideremos las siguientes propiedades de $\phi(x,t)$:

- Si la temperatura es constante, entonces no hay flujo, es decir, $\phi(x,t) = 0$.
- Si hay diferencia de temperatura, entonces la energía térmica fluye de la región caliente a la fría.
- A mayor diferencia de temperatura, mayor es el flujo.

Con base en lo anterior, se tiene que podemos modelar a $\phi(x,t)$ de la forma

$$\phi(x,t) = -k_0 \frac{\partial u}{\partial x}, \quad k_0 > 0.$$

Esta justamente es la Ley de Fourier, y a k_0 se le denomina conductividad térmica.
Así, tenemos que

$$\begin{aligned} c\rho \frac{\partial u}{\partial t} + \frac{\partial \phi}{\partial x} - E(x,t) &= 0 \\ \Rightarrow c\rho \frac{\partial u}{\partial t} - k_0 \frac{\partial^2 u}{\partial x^2} - E(x,t) &= 0 \\ \Rightarrow c\rho \frac{\partial u}{\partial t} &= E(x,t) + k_0 \frac{\partial^2 u}{\partial x^2}. \end{aligned}$$

Ahora, notemos que en nuestro caso de interés, en el cual no hay fuentes adicionales de energía térmica, tenemos la expresión.

$$c\rho \frac{\partial u}{\partial t} = k_0 \frac{\partial^2 u}{\partial x^2} \Rightarrow \frac{\partial u}{\partial t} = \frac{k_0}{c\rho} \cdot \frac{\partial^2 u}{\partial x^2} := k \cdot \frac{\partial^2 u}{\partial x^2}.$$

A la nueva constante $k > 0$, se le denomina coeficiente de difucibidad.
De esta forma tenemos la deducción de la ecuación del calor para una dimensión, y es fácil notar que la ecuación resultante para el caso de dos dimensiones esta dado por

$$\frac{\partial u}{\partial t} = k \cdot \frac{\partial^2 u}{\partial x^2} + k \cdot \frac{\partial^2 u}{\partial y^2}.$$

2. Diferencias Finitas

Ahora, con base en lo anterior, el objetivo es poder calcular el calor de una placa metálica en un momento $t + \triangle t$, conociendo el calor de la placa en un momento t.

Para esto, es necesario poder resolver la ecuación diferencial

$$\frac{\partial u}{\partial t} = k \cdot \frac{\partial^2 u}{\partial x^2} + k \cdot \frac{\partial^2 u}{\partial y^2}.$$

Y para esto emplearemos el método de diferencias finitas para así aproximarnos a un resultado. A continuación definiremos dicho método:

Sea $f : [a, b] \to \mathbb{R}$ una función diferenciable en su dominio. De la definición de derivada, para $x \in [a, b]$ y $h > 0$, se tiene la aproximación

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + ...$$
$$f(x - h) = f(x) - hf'(x) + \frac{h^2}{2} f''(x) + ...$$

Con base en las expresiones anteriores, tenemos las siguientes aproximaciones respectivamente.

- Diferencias finitas hacia adelante

$$f'(x) = \frac{f(x + h) - f(x)}{h} + O(h).$$

- Diferencias finitas hacia atrás

$$f'(x) = \frac{f(x) - f(x - h)}{h} + O(h).$$

Esta aproxiamción viene directamente de la defínción de derivada

$$f'(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h}.$$

Pero ademas, si consideramos una extensión mayor con el polinomio de Taylor, tenemos lo siguiente

$$f(x + h) = f(x) + hf'(x) + \frac{h^2}{2} f''(x) + \frac{h^3}{6} f'''(x) + ...$$
$$f(x - h) = f(x) - hf'(x) + \frac{h^2}{2} f''(x) - \frac{h^3}{6} f'''(x) + ...$$

Al restar dichas expresiones y dividiendo entre h, obtenemos la aproximación por Diferencias finitas centradas:

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

Ahora, sea $\{x_i\}_{i=0}^n$ una sucesión de puntos equidistantes en $[a, b]$ (distancia h). Entonces tenemos que

$$f(x) = \sum_{i=0}^{n} f(x_i) L_{n,i}(x) + \frac{(x - x_0) \cdots (x - x_n)}{(n + 1)!} f^{(n+1)}(\xi_x), \quad \xi_x \in [a, b].$$

Donde

$$L_{n,i}(x) = \prod_{j=0, j \neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)}.$$

Luego al derivar con respecto a x se tiene que

$$f'(x) = \sum_{i=0}^{n} f(x_i)L'_{n,i}(x) + \frac{\frac{\partial}{\partial x}\left[(x-x_0)\cdots(x-x_n)\right]}{(n+1)!} f^{(n+1)}(\xi_x) + \frac{(x-x_0)\cdots(x-x_n)}{(n+1)!}\cdot\frac{\partial}{\partial x}\left[f^{(n+1)}(\xi_x)\right]$$

Y en particular se tiene que

$$f'(x_k) = \sum_{i=0}^{n} f(x_i)L'_{n,i}(x_k) + \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\prod_{j=0,j\neq i}^{n}(x_k - x_j).$$

A esta aproximación se le denomina como aproximación de la derivada con $n+1$ puntos, donde el error esta dado por

$$\frac{f^{(n+1)}(\xi_x)}{(n+1)!}\cdot\prod_{j=0,j\neq i}^{n}(x_k - x_j).$$

Finalmente, si consideremos el caso con $n=2$, usando los puntos $x_k - h, x_k$ y $x_k + h$, se tiene que

$$f'(x_k) = \frac{1}{2h}\left[-2f(x_k) + 4f(x_k+h) - f(x_k+2h)\right] + \frac{h^2}{3}f'''(\xi), \quad \xi \in [x_k - h, x_k + h].$$

Usando un proceso análogo para la segunda derivada, tenemos la aproximación

$$f''(x_k) = \frac{f(x_{k+1}) - 2f(x_k) + f(x_{k-1})}{h^2},$$

con un error de $\frac{h^2}{12}f^{(4)}(\xi)$.

Con base en lo anterior, tenemos las siguientes aproximaciones para el problema de interés:

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \frac{u(x,y,t+\triangle t) - u(x,y,t)}{\triangle t},\\
\frac{\partial^2 u}{\partial x^2} &= \frac{u(x+\triangle x,y,t) + 2u(x,y,t) + u(x-\triangle x,y,t)}{\triangle x^2},\\
\frac{\partial^2 u}{\partial y^2} &= \frac{u(x,y+\triangle y,t) + 2u(x,y,t) + u(x,y-\triangle y,t)}{\triangle y^2}.
\end{aligned}$$

Notemos que lo anterior es válido debido a que unicamente tomamos las derivadas parciales de u. Luego, si consideramos la notación

$$u_{ij}^n := u(x_i, x_j, t_n),$$

y ademas consideramos $\{x_k\}, \{y_k\}, \{t_k\}$ equidistantes, tenemos que

$$\begin{aligned}
\frac{\partial u}{\partial t} &= \frac{u_{ij}^{n+1} - u_{ij}^n}{\triangle t},\\
\frac{\partial^2 u}{\partial x^2} &= \frac{u_{i+1,j}^n + 2u_{i,j}^n + u_{i-1,j}^n}{\triangle x^2},\\
\frac{\partial^2 u}{\partial x^2} &= \frac{u_{i,j+1}^n + 2u_{i,j}^n + u_{i,j-1}^n}{\triangle y^2}.
\end{aligned}$$

Ahora, sustituyendo los valores anteriores en la ecuación diferencial de interés, tenemos la siguiente expresión:

$$\frac{u_{ij}^{n+1} - u_{ij}^{n}}{\triangle t} = k \cdot \frac{u_{i+1,j}^{n} + 2u_{i,j}^{n} + u_{i-1,j}^{n}}{\triangle x^2} + k \cdot \frac{u_{i,j+1}^{n} + 2u_{i,j}^{n} + u_{i,j-1}^{n}}{\triangle y^2}$$

$$\Rightarrow u_{ij}^{n+1} - u_{ij}^{n} = \frac{k\triangle t}{\triangle x^2} \cdot \left(u_{i+1,j}^{n} + 2u_{i,j}^{n} + u_{i-1,j}^{n}\right) + \frac{k\triangle t}{\triangle y^2} \cdot \left(u_{i,j+1}^{n} + 2u_{i,j}^{n} + u_{i,j-1}^{n}\right)$$

$$\Rightarrow u_{ij}^{n+1} - u_{ij}^{n} = r_x \cdot \left(u_{i+1,j}^{n} + 2u_{i,j}^{n} + u_{i-1,j}^{n}\right) + r_y \cdot \left(u_{i,j+1}^{n} + 2u_{i,j}^{n} + u_{i,j-1}^{n}\right)$$

$$\Rightarrow u_{ij}^{n+1} = u_{ij}^{n} + r_x \cdot \left(u_{i+1,j}^{n} + 2u_{i,j}^{n} + u_{i-1,j}^{n}\right) + r_y \cdot \left(u_{i,j+1}^{n} + 2u_{i,j}^{n} + u_{i,j-1}^{n}\right).$$

Dicha expresión nos permite conocer la temperatura en las coordenadas i, j en instante $n+1$, con base en las temperaturas de sus coordenadas vecinas mas ella misma en el instante anterior.

Esto es mas fácil de comprender con la siguiente imagen:

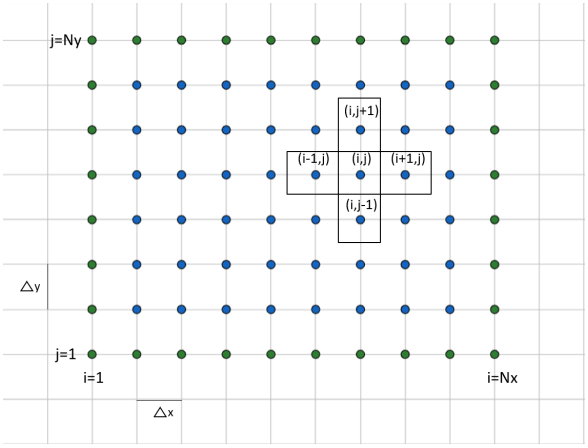


Figura 2:

Donde los puntos verdes son la frontera de la malla, la cual tiene valores conocidos.

3. Estrategias de Cálculo Paralelo

Recordemos el propósito general del código en cuestión, el cual es el aproximar el calor dado por la ecuación de calor en una superficie con una cantidad finita de puntos. Para esto, se realizara el proceso iterativo

$$u_{ij}^{n+1} = u_{ij}^{n} + r_x \cdot \left(u_{i+1,j}^{n} + 2u_{i,j}^{n} + u_{i-1,j}^{n}\right) + r_y \cdot \left(u_{i,j+1}^{n} + 2u_{i,j}^{n} + u_{i,j-1}^{n}\right), \quad r_x = \frac{k_x \triangle t}{\triangle x^2}, \quad r_y = \frac{k_y \triangle t}{\triangle y^2}.$$

Esto de forma secuencia de puede hacer fácilmente usando un ciclo for sobre los tiempos, y dentro dos ciclos for para recorrer el interior de la malla (las fronteras son fijas). De modo simple, dicha estrategia se representa de la siguiente forma:

Algoritmo 1

```
for(int t = 0; t < Dt; t++){
    for(int i = 1; i < Dx-1; x++){
        for(int j = 1; j < Dy-1; y++){
            u_new[i][j] = u_old[i][j];
            u_new[i][j] += rx*(u_old[i+1][j] + 2*u_old[i][j] + u_old[i-1][j]);
            u_new[i][j] += ry*(u_old[i][j+1] + 2*u_old[i][j] + u_old[i][j-1]);
        }
    }
    u_old = u_new;
}
```

A continuación, desarrollaremos la estrategia para realizar dicho cálculo de modo paralelo con la ayuda de la biblioteca MPI.

Primero, lo principal es el dividir el trabajo en varios procesos, los cuales se ejecutaran de forma paralela. Para esto, emplearemos el método de descomposición de dominios, el cual consiste en dividir la malla en bloques, donde cada bloque sera asignado a un procesador. Así, cada procesador calcula las soluciones sobre su sub-dominio, y enviar la información necesaria para los cálculos de las soluciones de bloques vecinos. En este caso, se buscara dividir la malla en bloques rectangulares a lo largo de ambos ejes.

Un ejemplo de como funcionaría dicha descomposición se da en la imagen siguiente:

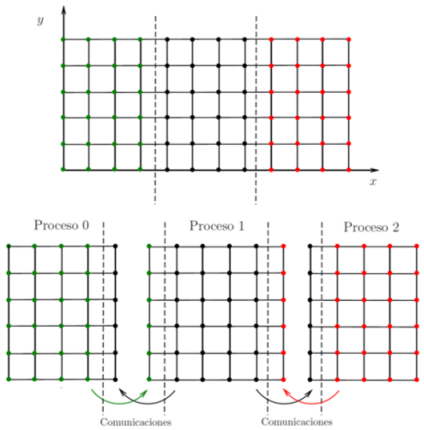


Figura 3:

En este caso dividimos la malla en 3 bloques, lo que da lugar a 3 procesos. Notemos que cada proceso requiere de los bordes del proceso vecino para calcular sus soluciones, esto debido a que para calcular el valor

u_{ij}^{n+1}, se requieren de los valores de sus vecinos sobre los ejes x, y respecto al tiempo anterior. Por lo tanto se agregan dichos bordes a los procesos para los cálculos en cuestión.

Pero hay una cuestión importante, la cual es la actualización. Como se puede apreciar, calculamos los valores de la malla para cada proceso, pero los bordes de cada uno no cuentan con sus nuevos valores, lo cual es un problema para la iteración siguiente. Pero notemos que los valores de dichos bordes se encuentran dentro de los valores calculados para los procesos vecinos, por lo que se necesita emplear comunicación entre procesadores.

Para realizar la comunicación entre procesadores, se hace uso de los comandos de MPI; MPI_Send y MPI_Recv. Dichos comandos permiten enviar los valores de los bordes desde los procesos vecinos.

Con base en lo dicho anteriormente, tenemos que el método de descomposición de dominios es una técnica eficiente para poder paralelizar el cálculo de la ecuación de calor en dos dimensiones sobre una malla. Donde dicha malla se divide en varios sub-dominios, y a cada procesador se le asigna un bloque para procesarlo de forma paralela. Además, también es requerida la comunicación entre procesos con bloques vecinos para garantizar que la solución en los bordes de cada sub-dominio coincidan con los valores de los sub-dominios vecinos. Finalmente, estos valores se combinan para formar la solución final de cada iteración.

Una visualización de como es la comunicación entre procesos es la que se ve a continuación:

Figura 4:

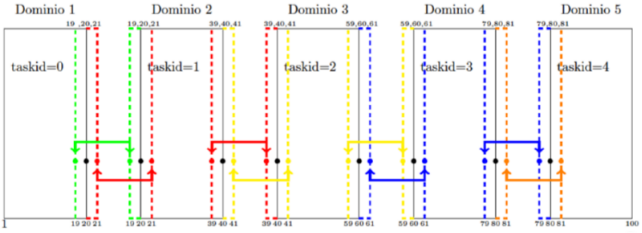
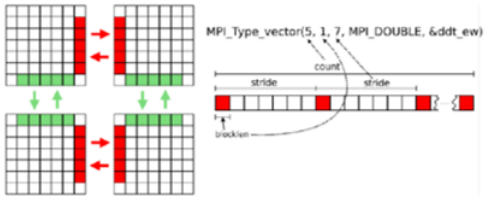


Figura 5:



Una vez que hemos explicado como se llevara a cabo la paralelización, es necesario mencionar los comandos en MPI a utilizar en el código en cuestión:

- **MPI_Send:** Manda información de un proceso a otro.
- **MPI_Recv:** Recibe información de otro proceso.
- **MPI_Init:** Inicializa el proceso en MPI, bloquea la ejecución hasta que todos los procesos hayan sido inicializados.
- **MPI_Comm_size:** Obtiene el número de procesos ejecutados.
- **MPI_Comm_rank :** Obtiene el número de rango del proceso actual.
- **MPI_Finalize:** Termina la conexión MPI, bloquea la ejecución hasta que todos los procesos terminen.

4. Desarrollo del código de forma secuencial y Paralela

Para la implementación del código, se utilizaron dos implementaciones, una secuencial y otra paralela, las cuales se encuentran en los archivos "Calor_2D_sec.cpp" y "Calor_2D_par_2.cpp" respectivamente.

A continuación, explicaremos el funcionamiento del código secuencial.

1. Inicialización de valores.

Inicializamos las variables que se necesitaran a lo largo del código. Entre dichas variables, se encuentran los dominios de la malla, $[xI, xF] \times [yI, yF] = [0,1] \times [0,1]$, los términos difusivos $kx = ky = 1$, el tiempo total $[tI, tF] = [0,1]$, la cantidad de puntos $NxG = 200, NyG = 200$, y la cantidad de tiempos $Nt = 200000$.

2. Cálculo de Constantes

Se calculan las distancias entre puntos sobre ambos ejes, distancia entre tiempos, y las constantes rx, ry:

$$Dx = \frac{xF - xI}{NxG - 1}, \quad Dy = \frac{yF - yI}{NyG - 1}, \quad Dt = \frac{tF - tI}{Nt - 1},$$
$$rx = kx \cdot \frac{Dt}{Dx^2}, \qquad ry = ky \cdot \frac{Dt}{Dy^2}.$$

3. Condición de error.

Una condición para determinar si el sistema es estable, es el hecho de que se cumpla que $rx + ry < 1/2$. Por lo que se realiza un if para determinar si el ciclo sera estable o no.

4. Declaración de "matrices" para las iteraciones.

Declaramos vectores de tamaño $NxG \cdot NyG$, los cuales representan una matriz de dimensiones $NxG \times NyG$.

Así, al punto (i, j), lo conoceremos como la entrada $(i \cdot NyG + j)$ del vector.

Nótese que debido a que este es un proceso iterativo, declararemos dos vectores, uno para el e tiempo n (mu_old), y otro para el tiempo $n + 1$ (mu_new)

5. Inicialización de la malla vieja.

Inicializamos el vector mu_old sobre todas su entradas de la forma

$$mu\_old[i \cdot NyG + j] = \sin\left(xI + i \cdot Dx + yI + j \cdot Dy\right)^2.$$

6. Condiciones de frontera de la malla nueva.

Inicializamos las fronteras de mu_new con el valor 1.

7. Iteraciones.

Se implementa un ciclo for para recorrer los Nt tiempos, y en cada iteración de cada tiempo, se busca calcular mu_new con base en mu_old.

Tras terminar de calcular mu_new para cada tiempo, actualiza el valor de mu_old con el de mu_new.

8. Cálculo de tiempo final.

Se calcula el tiempo en el que se ejecuto todo el código.

A continuación, explicaremos el funcionamiento del código paralelo.

1. Inicialización de valores.

Inicializamos las variables que se necesitaran a lo largo del código. Entre dichas variables, se encuentran los dominios de la malla, $[xI, xF] \times [yI, yF] = [0,1] \times [0,1]$, los términos difusivos $kx = ky = 1$, el tiempo total $[tI, tF] = [0,1]$, la cantidad de puntos $NxG = 200, NyG = 200$, y la cantidad de tiempos $Nt = 200000$.

2. Cálculo de Constantes

Se calculan las distancias entre puntos sobre ambos ejes, distancia entre tiempos, y las constantes rx, ry:

$$Dx = \frac{xF - xI}{NxG - 1}, \quad Dy = \frac{yF - yI}{NyG - 1}, \quad Dt = \frac{tF - tI}{Nt - 1},$$
$$rx = kx \cdot \frac{Dt}{Dx^2}, \qquad ry = ky \cdot \frac{Dt}{Dy^2}.$$

3. Condición de error.

Una condición para determinar si el sistema es estable, es el hecho de que se cumpla que $rx + ry < 1/2$. Por lo que se realiza un if para determinar si el ciclo sera estable o no.

4. Cálculo y Declaración de variables para procesos.

Se realiza el proceso de división de la malla en pase a la cantidad de procesos. Además, se tomaran los números de procesos vecinos (si es que existen).

5. Declaración de "matrices" para las iteraciones.

Declaramos vectores de tamaño $Nx \cdot Ny$, los cuales representan una matriz de dimensiones $Nx \times Ny$. Notemos que se añadieron dos columnas extra al proceso para tener en cuenta los posibles bordes que tendrán.

6. Inicialización de la malla vieja.

Inicializamos el vector mu_old sobre todas su entradas de la forma

$$mu\_old[i \cdot Ny + j] = \sin\left(xI + i \cdot Dx + yI + j \cdot Dy\right)^2.$$

7. Condiciones de frontera de la malla nueva.

Inicializamos las fronteras de mu_new con el valor 1. En este caso, es necesario ver si el proceso en cuestión tiene fronteras este u oeste. Ya que sino es así, no es necesario inicializar las susodichas.

8. Iteraciones.

Se implementa un ciclo for para recorrer los Nt tiempos, y en cada iteración de cada tiempo, se busca calcular mu_new con base en mu_old.

En este proceso, por cada tiempo se envía y se recibe información de los nodos vecinos (si es que existen).

Tras terminar de calcular mu_new para cada tiempo, actualiza el valor de mu_old con el de mu_new.

9. Cálculo de tiempo final.

Se calcula el tiempo en el que se ejecuto todo el código.

5. Simulaciones numéricas

Usaremos el grupo "C1Mitad1"

Procesos\Iteraciones	1	2	3	4	5	Promedio	Speedup	Eficiencia
0	129.748	129.526	129.609	129.799	129.849	129.7062		
2	69.1564	69.2815	69.1114	69.2983	69.2624	69.222	1.8738	0.9369
4	34.5762	34.5928	34.4713	34.5486	34.4168	34.52114	3.7573	0.9393
6	23.5874	23.6612	23.5446	23.5948	23.5	23.5776	5.5012	0.9168
8	17.8833	17.8794	17.9833	17.8758	17.9808	17.92052	7.2378	0.9047
10	14.365	14.3948	14.4071	14.3869	14.3816	14.38705	9.0154	0.9015

Nótese que hacemos la notaremos como la ejecución de 0 procesos como la ejecución secuencial.

El speedup se emplea para medir la eficiencia de la paralelización, y se define como t_{seq}/t_{par}. Se espera que el speedup funciona de forma lineal, esto es, que si se paraleliza q veces, entonces el código es q veces mas rápido. Pero esto no siempre sucede debido a factores externos. A continuación, veremos una gráfica del speedup evaluado anteriormente.

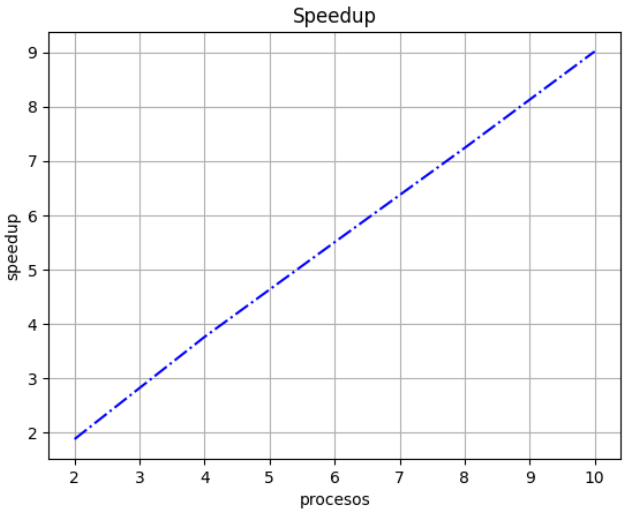


Figura 6:

Como podemos apreciar, se cumple en este caso que la eficiencia se asemeja bastante a una función lineal.

Luego, la eficiencia, como su nombre lo indica, nos ayuda a medir que tan eficiente es una ejecución. Esto debido a que dependiendo de cual sea la diferencia de speedup, puede no ser tan conveniente el ejecutar tantos procesos.

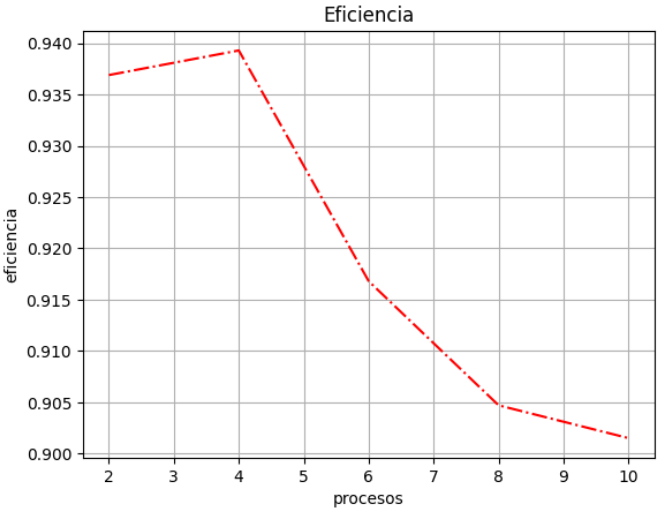


Figura 7:

En este caso, podemos ver como la eficiencia se reduce considerablemente a partir de los cuatro procesos.

6. Conclusiones y Bibliografía

Al notar la gráfica de speedup, podemos apreciar como tenemos un comportamiento muy lineal. Conforme el número de número de procesos aumenta, el speedup aumenta de igual forma. Ahora, esto no implica que entre mas procesos mejor, esto porque se puede llegar a un punto en que se tengan mas procesos de los necesarios, haciendo que el tiempo de ejecución se estanque, o incluso crezca. Por lo tanto, para los procesos utilizados, podemos afirmar que se presenta una buena capacidad de paralelización, pero no sepuede garantizar que dicha propiedad se mantenga para algún número de procesos mas altos.

Bibliografía.

1. Peter V. O'Neil , Matemáticas avanzadas para ingeniería.
2. Peter J. Olver, Introduction to Partial Differential Equations.
3. Richard Haberman, Ecuaciones en Derivadas Parciales.
4. Clarke, L., Glendinning, I., and Hempel, R. The mpi message passing interface standard.