Potenciales eléctricos a través de las ecuaciones de Laplace

Kimy Agudelo Jaramillo [1*]
[1] *Instituto de Física, Universidad de Antioquia,*
Calle 70 # 52-21, Apartado Aéreo 1226, Medellín, Colombia

En este trabajo se presenta el estudio del comportamiento que exhibe el potencial eléctrico asociado a todos los puntos dentro de un cuadrado libre de carga, usando el método de monte carlo. Para ello se tenía en consideración que el fondo de dicho cuadrado y los lados están formados por cables que se encuentran conectados a Tierra, es decir, $0V$; mientras que la parte superior se encuentra conectado a un potencial constante de $100V$. Se realiza la implementación de los algoritmos de monte carlo, así como por differencias finitas, y finalmente se realiza una comparación con los resultados obtenidos utilizando multiprocessing. Dichos algoritmos son presentados en el lenguaje de *Python*.

1. INTRODUCCIÓN

El método de Monte Carlo es un método estadítico numérico empleado para aproximar expresiones matemáticas muy complicadas con exactitud, es decir me permite resolver problemas físicos y matemáticos mediante la simulación de variables aleatorias. Aunque originalmente Monte Carlo no era un método para solucionar problemas en física, sino más bien para calcular integrales que no podian ser obtenidas de otra manera, tales como: integrales de funciones pobremente comportadas e integrales en espacios multilineales. Esto se realiza por medio de métodos para la simulación de variables aleatorios.

El método de diferencias finitas (FD) consiste en obtener la solución de un problema diferencial en un conjunto discreto de puntos. Esto se realiza mediante la aproximación de las derivadas parciales que aparecen en las ecuaciones de Lapace por operadores y diferencias finitas. Como resultado se obtiene un sistema de ecuaciones lineales, las cuales permiten una solución más sencilla.

El objetivo de este trabajo es obtener el potencial solucionando la ecuación de Laplace. Para esto se hace necesario la utilización de los paquetes Multiprocessing y numba para Python, los cuales permiten la ejecuación más rápida del código, además permten la obtención de los tiempos de ejecución necesarios para realizar la tarea. Este documento se estructura de la siguiente está compuesto por la siguiente estructura: en la sección 2 se describe la solución a la ecuación de Laplace y se presenta el desarrollo del método computacional utilizado, posteriormente en la sección 3 se muestran y discuten los resultados obtenidos al realizar la simulación. Finalmente, en la sección 4 se presentan las conclusiones del trabajo.

2. MARCO TEÓRICO

A. Ecuación de la Laplace

La ecuación de la Electrostática

$$\nabla \times \vec{E} = 0 \tag{2.1}$$

implica que el campo eléctrico se puede expresar como el gradiente de un potencial escalar (U)

$$\vec{E} = -\nabla U(x, y) \tag{2.2}$$

El cual debe satisfacer también la otra ecuación de la Electrostática,

$$\nabla \cdot \vec{E} = 4\pi\rho \tag{2.3}$$

Por lo tanto, $\nabla^2 U(x, y) = -4\pi\rho$. En sitios donde no hay cargas $(\rho = 0)$, el potencial escalar satisface la ecuación de Laplace

$$\nabla^2 U(x, y) = 0 \tag{2.4}$$

B. Soluciones vía diferencias finitas

Consideremos la ecuación diferencial parcial de Laplace asociada un cuadrado una región del espacio en la que un cable en la parte superior se conecta a una

* kimy.agudelo@udea.edu.co

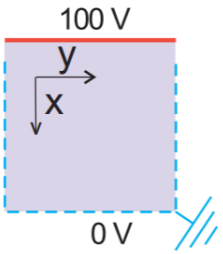

Figura 2.1: Región del espacio en la que un cable en la parte superior se mantiene a 100 V con los lados y la parte inferior conectados a tierra.

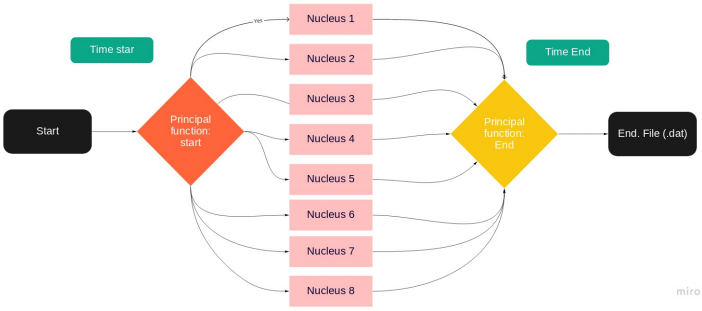

Figura 2.2: Esquema del código implementado para la obtención de potencial asociado a la ecuación de Laplace.

diferencia de potencial de $100V$ con los lados y la parte inferior conectados a tierra, lo cual implica que se encuentra a una diferencia de potencial de $0V$, por lo tanto

$$\frac{\partial^2}{\partial x^2}U(x,y) + \frac{\partial^2}{\partial y^2}U(x,y) = 0, \tag{2.5}$$

La carga eléctrica es la fuente de campos electrostáticos, ya sea directamente en la densidad de carga o indirectamente a través de la imposición de condiciones de contorno. Una condición de contorno especificada $U(x,y) = f(x',y')$. Al usar diferencias finitas obtenemos,

$$\begin{aligned} U(x,y) = \frac{1}{4}(U(x+\Delta,y) + U(x-\Delta,y) \\ +U(x,y+\Delta) + U(x,y-\Delta) \end{aligned} \tag{2.6}$$

Podemos darle a esta ecuación una interpretación probabilística: si consideramos una cuadrícula de puntos en el plano xy con un espaciado de celosía de Δ, entonces la probabilidad de que una caminata aleatoria regrese al punto (x,y) de cualquiera de sus sitios vecinos más cercanos es $1/4$. Por lo tanto, cualquier recorrido aleatorio que comience desde (x,y) terminará en un punto límite (x',y') donde $U(x',y') = f(x',y')$. Luego se obtiene una estimación de la solución ejecutando caminatas aleatorias de N y considerando el promedio

$$U(x,y) = \frac{1}{N}\sum_{i=1}^{N} f(x'_i, y'_i) \tag{2.7}$$

Después de un gran número de caminatas, una buena estimación de Se producirá $U(x,y)$, pero la estimación dependerá tanto del grosor de la cuadrícula como del número de recorridos aleatorios generados.

C. Métodos computacionales

Utlizando el método de monte carlo, diferencias finitas y empleando la librería Multiprocessing de Python para ejecutar cálculos somputacionales en paralelo se puede obtener la solución de la ecuación de Laplace. En la figura 2.2 se presenta el esquema del algoritmo implementado, el cual consiste en distribuir el rango de los valores que se evalúan en el potencial eléctrico. El algoritmo de la ecuación de Laplace en el que el potencial en el punto $(x,y) = (i,j)\triangle$ es igual al promedio de los valores de potencial en los cuatro puntos vecinos más cercanos, por lo tanto, inicialmente se define la función en los cuatro vecinos más cercanos y posteriormente se aplica multiprocessing. Las soluciones a la ecuación 2.4 se van almacenando en las diferentes colas utilizadas, donde luego de terminar con el proceso de computo se obtiene un archibvo .dat que obtiene las soluciones de todos los procesadores utilizados.

3. RESULTADOS Y ANÁLISIS

En la figura 3.1 se presenta el potencial eléctrico obtenido a partir de la solución de la ecuación de Laplace por el método de diferencias finitas, este método permite que el tiempo de cómputo mejore considerablemente con respecto al método de monte carlo, donde se obtuvo un tiempo de compilación de $0{,}99s$. En dicha figura, las proyecciones sobre el plano xy sombreado son líneas equipotenciales (curvas de nivel). En la figura 3.2 se gráfica el potencial eléctrico obtenido a partir de la solución de la ecuación de Laplace por el método de monte car-

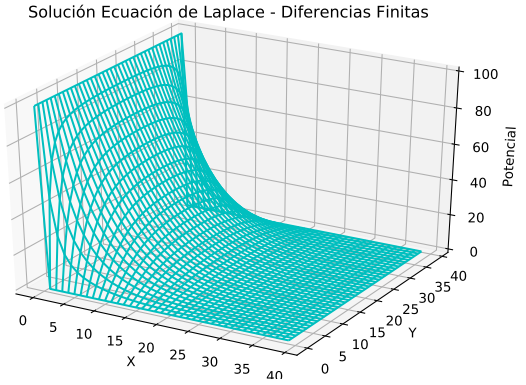

Figura 3.1: Potencial eléctrico calculado en función de las coordenadas xy

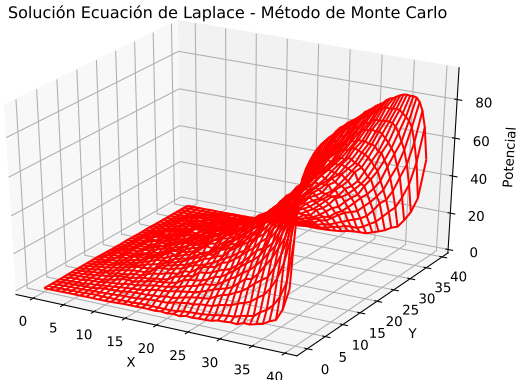

Figura 3.2: Potencial eléctrico calculado en función de las coordenadas xy. Realizado a partir del método de monte carlo, utilizando 10^8 iteraciones

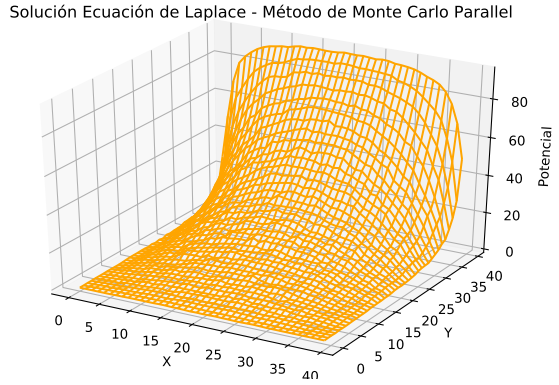

Figura 3.3: Potencial eléctrico calculado en función de las coordenadas xy. Realizado a partir del método de monte carlo y Multiprocessing, utilizando 10^8 iteraciones

Núcleos	Tiempo(s)	Porcentaje (%)
1	9.1791	100
2	8.5592	93.2
3	8.5353	92.98
4	6.1750	67.27
5	6.0320	65.71
6	5.6743	61.81
7	5.6266	61.29
8	5.6266	61.29

Cuadro I: Porcentaje del tiempo de cómputo en paralelo.

lo, donde es importante resaltar que se utilizaron 10^8 iteraciones. Para este cálculo se obtuvo un tiempo de cómputo de $21{,}2s$, lo cual sigue siendo un buen tiempo sin embargo el mejor tiempo de compilación fue el obtenido a partir del método de diferencias finitas. Es importante considerar los valores que alcanzó el potencial eléctrico $U(x, y)$ con diferencias finitas y su error con el método de monte carlo.

finalmente, se gráfico (ver figura 3.3) el potencial eléctrico utilizando la librería *Multiprocessing* de Python, la cual permitía que el código ejecutará más rápido las tareas y creará un archivo .dat, el cual en este caso consistio de una matriz de 40 columnas y 40 filas. En la figura 3.4 se presenta el tiempo de ejecución necesario para obtener el potencial eléctrico mostrado en la figura 3.3, dicho tiempo se obtuvo considerando los diferentes procesadores utilizados (ver I), el cual inicia a contar con la asignación de las funciones y finaliza cuando arroja el archivo .dat, el cual es necesario para obtener dichas figuras. Como se puede observar en dicha figura el tiempo de cómputo mejoró considerablemente a medida que se incrementaba los procesadores, con un procesador se obtuvo un tiempo de $9{,}1791s$ correspondiente a un porcentaje de 100 %, mientras que cuando se utilizaron los 8 procesadores se obtuvo un tiempo de cómputo de $5{,}622s$, correspodiente a un porcentaje de 61,29 % con respecto al primer núcleo. Es muy importante tener presente que desde el quinto procesador los cambios fueron pocos, de aproximandamente un 4 % con respecto al anterior.

Con el fin de obtener mayor información de los errores computacionales a la hora de utilizar el método de monte carlo y el método de diferencias finitas se obtuvo una gráfica que presennta este error. donde, como se observa en las figuras 3.3) y 3.2) se presenta una diferencia de los valores alcanzados por el potencial eléctrico ($U(x, y)$), lo cual se confirma en la figura 3.5).

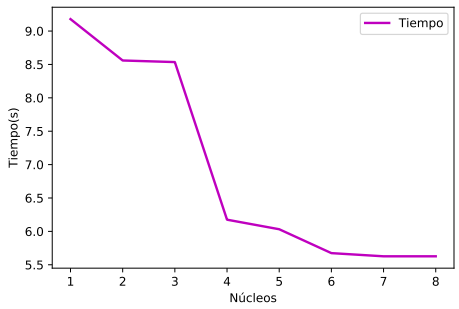

Figura 3.4: Tiempos de cálculo requeridos para obtener el potencial

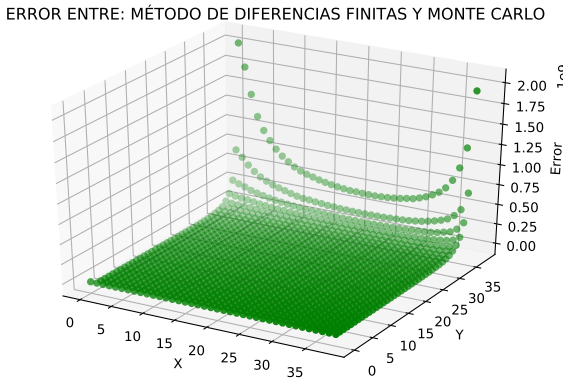

Figura 3.5: Error computacional entre el método de monte carlo y diferencias finitas

4. CONCLUSIONES

A través de métodos de cómputo en paralelo se obtuvieron resultados numéricos para el potencial eléctrico como solución de la ecuación de Laplace, el cual fue graficado y se corroboró el comportamiento teórico del potencial. Se encontró que el tiempo de compilación necesario para realizar las tareas disminuye al utilizar más núcleos del procesador, e este caso se utilizaron 8 núcleos, sin embargo cae resaltar que apartir del sexto núcleo los cambios presentados en el tiempo de compilación fueron pequeños, alrededor de un 4 % con respecto al obtenido con el quinto núcleo. Esto se obtuvo con dos librerías de Python, *Multiprocessing* y *Numba*

Se encontró que cuando se aplica el método de diferencias finitas para hallar el potencial eléctico asociado a una barra sometida a un potencial y formando un cuadrado el cual se encuentra conectado a Tierra es mucho más eficiente que cuando se utiliza el método de mote carlo, aproximadamente un 95 % de mejoría e el tiempo de compilación.

finalmente, se obtuvo una gráfica donde se muestra el error computacional presentado cuando se utiliza el método de monte carlo y el método de diferencias finitas, en el cual se encontró que para el caso de diferencias finitas se obtienen valores mayores del potencial eléctrico, mientras que por monte carlo se alcanzaron valores menores.

[1] R. H. Landau, J.M. Páez, C. C. Bordeianu, Computational Physics. Weinheim, Germany , 2015.

[2] Luis de la Peña, "Introducción a la mecánica cuántica," Universidad Autónoma de México, México, (2014).

[3] Chung, K.L. (1982): Lectures from Markov Processes to Brownian Motion. Springer-Verlag

[4] Matter Paul; Tso Brandt, "Classification Methods for Remotely Sensed Data 2 ed.", 2009.

[5] Charles J. Geyer, Introduction to Markov Chain Monte Carlo. Statistical Science, 7:473–511.

[6] Abhirag Awasthi, Thinking Recursively in Python, (2021), https://realpython.com/python-thinking-recursively/

[7] IPython Cookbook, Second Edition (2018), https://ipython-books.github.io/

[8] [Alonso, Finn(67)] Alonso, M. and E. J. Finn (1967), Fundamental University Physics, Addison-Wesley, Reading.