PROYECTO SEMESTRAL
EDPN

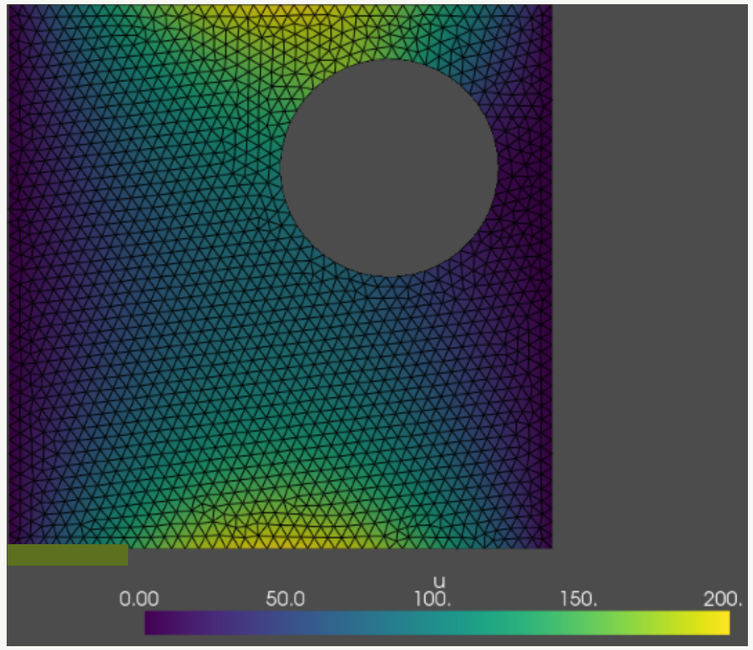
COMPARACIÓN DE UN MÉTODO DE TIPO MONTECARLO Y RESOLUCIÓN DIRECTA DE UNA EDP

Presentación de Johnny Godoy
Javier Santidrián
Patricio Yáñez

INTRODUCCIÓN

TEMAS PARA DEBATIR

Durante el curso hemos resuelto EDPs transformándolas en ecuaciones lineales del tipo Ah*uh = bh. Aumentar la precisión de esta aproximación (afinando la discretrización) requiere solucionar un sistema con más variables, aumentando el costo computacional.



Este método lo hemos aplicado a EDPs del estilo Poisson sobre distintos dominios (cuadrados perforados y no perforados) obteniendo múltiples resultados.

¿QUÉ BUSCAMOS?

Buscamos estudiar otra forma de resolver estos problemas mediante un método de Markov Chain Monte Carlo (MCMC).

Esto se puede realizar gracias a la ecuación de Feynman-Kac, que escribe la solución de EDPs como una esperanza condicional, permitiendo el uso de algoritmos estocásticos.

$$\frac{\partial u}{\partial t}(x,t) + \mu(x,t)\frac{\partial u}{\partial x}(x,t) + \tfrac{1}{2}\sigma^2(x,t)\frac{\partial^2 u}{\partial x^2}(x,t) - V(x,t)u(x,t) + f(x,t) = 0,$$

$$u(x,T) = \psi(x),$$

$$u(x,t) = E^Q\left[\int_t^T e^{-\int_t^r V(X_\tau,\tau)\,d\tau} f(X_r,r)dr + e^{-\int_t^T V(X_\tau,\tau)\,d\tau}\psi(X_T)\,\middle|\, X_t = x\right]$$

PROBLEMA A ESTUDIAR

$$\triangle u = f \text{ en } (0,1)^2$$
$$u = g \text{ en } \partial[0,1]^2$$

Por simplicidad, iniciamos con un problema de Poisson con condiciones de borde Dirichlet en dominio cuadrado, con una grilla cuadrada de tamaño N.

Explicaremos ahora el algoritmo para calcular u(x) en un punto interno al dominio. Esto se parametriza por un número de paseos W. Aumentar esta cantidad mejora la aproximación, pero tiene mayor costo computacional.

Algoritmo MCMC

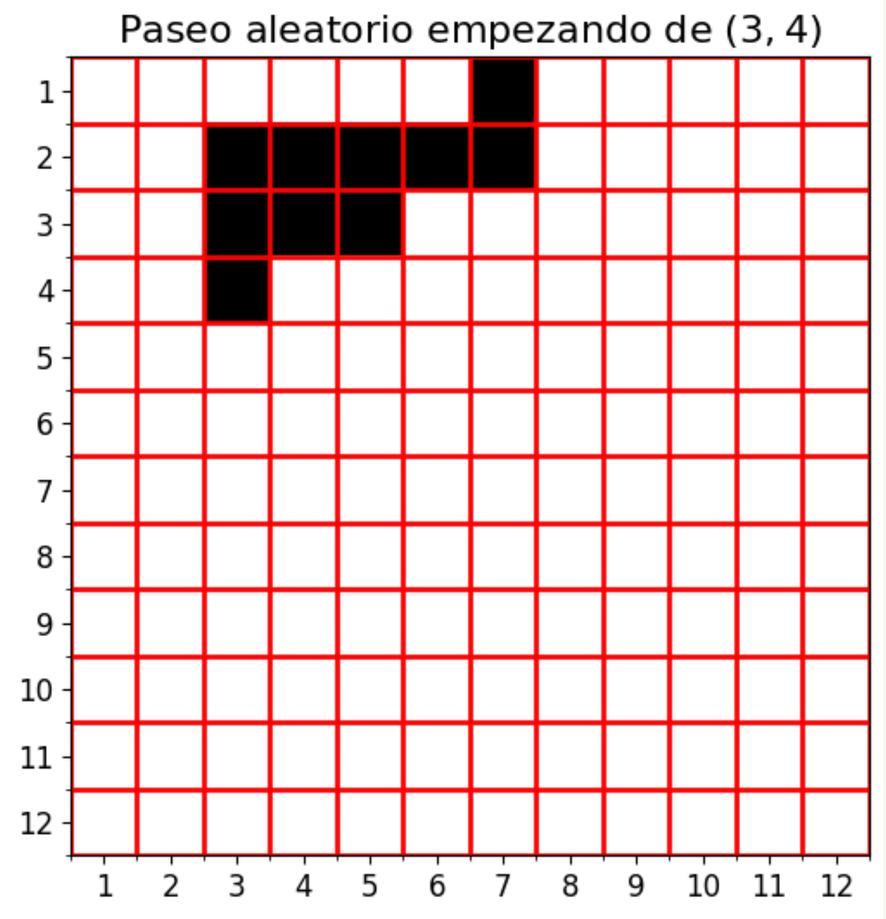
Paso 0: Inicialización

Inicializamos un diccionario *C* cuyas llaves son puntos del borde de la grilla, asociándolos al valor 0.

Este diccionario busca contar cuantos paseos pararon en cada punto de borde.

Paso 1: Simulación del paseo

Simulamos un paseo aleatorio simple en la grilla empezando desde *x* hasta llegar a un punto del borde.
Sea *w* este paseo.



Paso 2: Procesamiento del paseo

Calculamos y almacenamos el siguiente valor en función de los puntos interiores del paseo *w*

$$F(w) = \sum_{i \in w[:-1]} f(i)h^2$$

Donde *h* es el tamaño de la discretización. Más aún, si ξ es el punto final del paseo, se realiza la actualización C[ξ] += 1

Paso 3: Estimación de probabilidades

Repetimos los pasos 1 y 2 *W* veces. Esto nos permite estimar la probabilidad que un paseo que empieza de *x* termine en un punto de borde ξ como tal:

$$P(\xi) := \mathbb{P}_x(\exists n \in \mathbb{N} : X_n = \xi) \approx \frac{C[\xi]}{W}$$

Paso 4: Cálculo de *u(x)*

Retornamos la siguiente aproximación

$$u(x) = \sum_{\xi \in \partial\Omega} P(\xi) g(\xi) - \sum_{w} F(w)$$

Para calcular *u* en cada punto interior del dominio se itera este algoritmo en cada valor de *x*

Desventajas del algoritmo

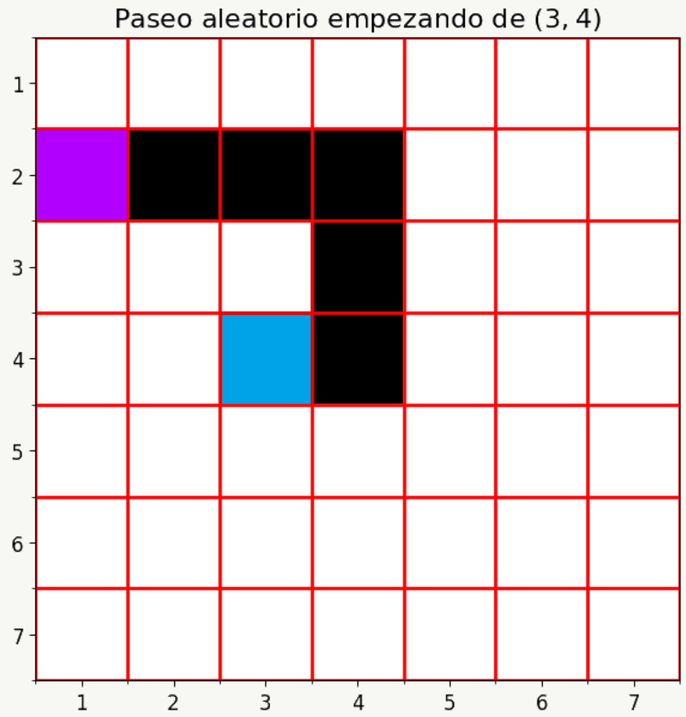
- El costo computacional para calcular u en todo el dominio es Ω(W*n^3) en el mejor de los casos. Esto es peor que los algoritmos de solución de sistemas lineales que lo logran en O(n^2.8).
- La información de un paseo solamente se utiliza para la estimación de u en un punto y luego se descarta.
- La estimación de P(ξ) puede dar 0 cuando la probabilidad que describe no es nula. De hecho, es condición necesaria que W = O(n) para que esto no ocurra (consideramos tantos paseos como puntos del borde), empeorando aún más la complejidad temporal.
- Cómo es estocástico, existe varianza asociada a la aproximación

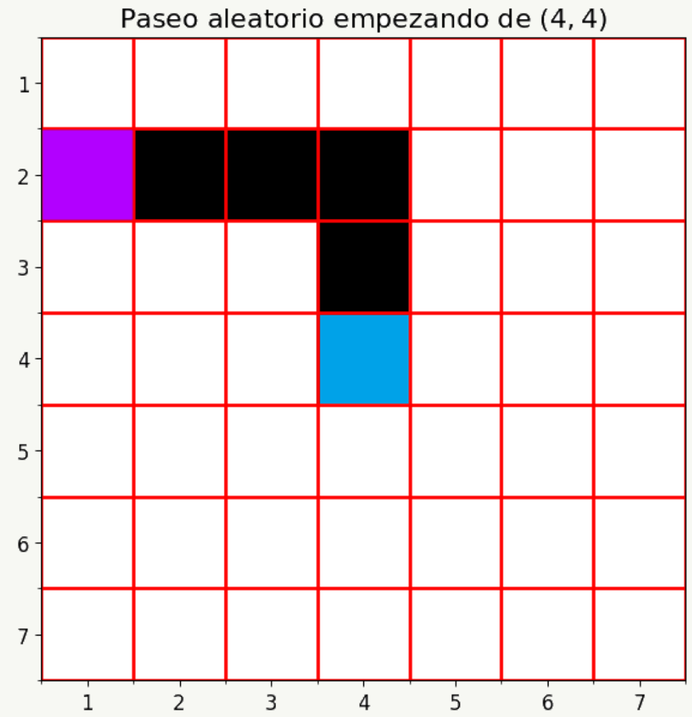
Soluciones a la Problemática

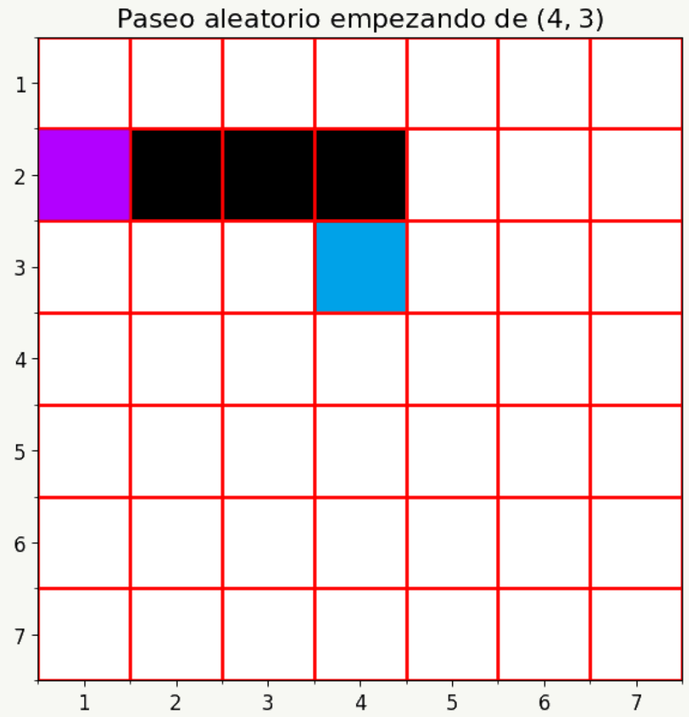
- Ideamos un algoritmo alternativo que mejora la complejidad temporal reutilizando la información de los paseos visitados
- Ambos algoritmos son fácilmente paralelizables, mitigando problemas de tiempo
- Nuestro algoritmo admite fácilmente técnicas de reducción de varianza conocidas del curso de Simulación Estocástica
- Consideramos utilizar una estimación bayesiana de $P(\xi)$ para mejorar el desempeño

Algoritmo alternativo

Observación clave: Al simular un paseo w = x_0x_1...x_n obtenemos información para calcular u(x_0). Por tanto, el subpaseo w' = x_1...x_n nos da información sobre u(x_1).







Así, en este algoritmo comenzamos con un punto inicial x_0 aleatorio y generamos un paseo tal como antes.

Luego, se iteran los subpaseos x_i...x_n para actualizar u(x_i).

Métricas de Comparación entre Métodos

-Escalabilidad (tiempo, memoria, código-qué tan utilizable es al cambiar parámetros o dominios)
-Limitaciones
-Error en la solución
-Simplicidad código

Reflexiones Finales

- Buscamos analizar, comparar y evaluar de buena forma el desempeño de los algoritmos en base a las métricas señaladas y a nuestras herramientas actuales.
- Deseamos expandir nuestros conocimientos sobre análisis numérico y sus aplicaciones.

Bibliografía

- Partial differential equations and Diffusion Processes - James Nolen
- https://github.com/s-ankur/montecarlo-pde/blob/master/Monte%20Carlo%20Methods%20Report.pdf