

Monte Carlo methods for Solving PDEs

Computational Physics Final Project

Yucheng Zhang

Department of Physics, New York University

yz4035@nyu.edu

December 4, 2017

Overview

1 Introduction

- Laplace's Equation with Dirichlet Boundary Condition
- Simple Random Walk method
- Walk on Spheres method
- Characteristics of Monte Carlo methods

2 Implementation & Analysis

- Parallelization
- Square Boundary
- Circle Boundary
- Analysis of WoS method

3 Future Work

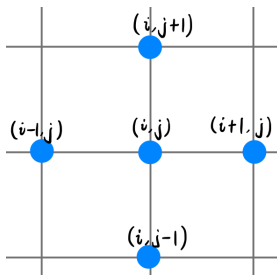
Laplace's Equation with Dirichlet Boundary Condition



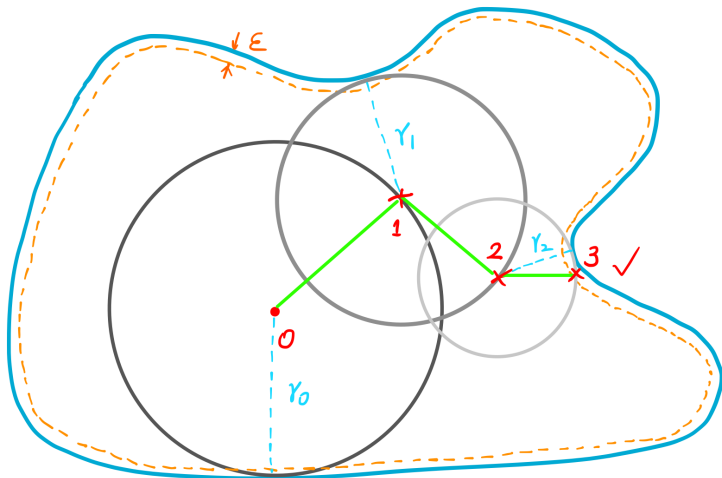
$$\begin{aligned}\nabla^2 u &= 0 && \text{on } G, \\ u &= f(x) && \text{on } \partial G.\end{aligned}$$

- Discrete form with centered finite difference approximation,

$$u_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}).$$



Walk on Spheres method



- Much faster than the Simple Random Walk method.

Characteristics of Monte Carlo methods

- *Independence of points:* **The points to be evaluated are totally independent.** This makes the Monte Carlo methods **very suitable and efficient for evaluating values at certain points.**
- *Boundary & Dimension:* It's easier for the Monte Carlo methods to **handle complex boundaries** and to be **extended to higher dimensions.**
- *Parallel Computing:* Not only different points, but also different estimates for a given point are independent. This makes the Monte Carlo methods **naturally parallel.**

Parallelization

- The two straightforward ways to parallelize the program are,
 - 1 Parallelize the different estimates of one point, then average the results from different processes.
 - 2 Parallelize the set of points to be evaluated.
- For the first way,
 - 1 Allocate the estimates you want to get for every point to all available cores;
 - 2 Initialize the random number generator of different processes with different seeds, with `numpy.random.RandomState(seed)`.
 - 3 Wait for all the processes to finish, then collect and average the data.
- *Python* package *multiprocessing*.

Square Boundary

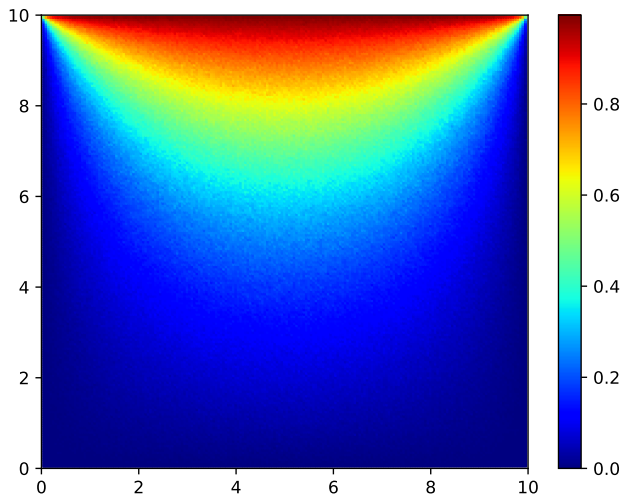


Figure: WoS method on 2D Square Boundary, 200×200 points, 2,000 estimates for each point.

Circle Boundary

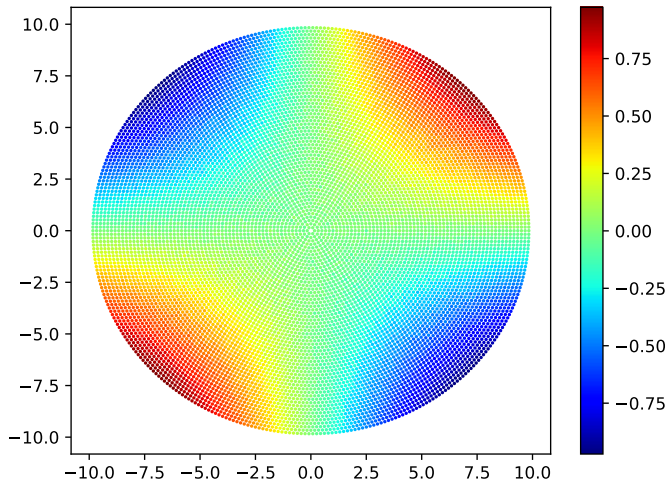


Figure: WoS method on 2D Circle Boundary, 10,620 points, 1,000 estimates for each point.

Analysis of WoS method - Running time vs. ϵ

- $t = -k \log \epsilon + t_0$

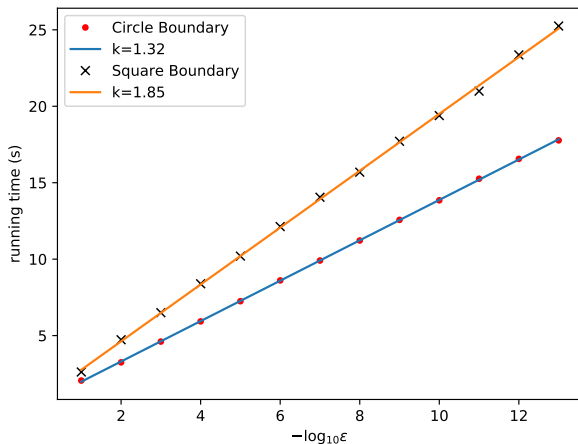


Figure: The running time - ϵ relation on both square and circle boundaries.

Analysis of WoS method - Convergence

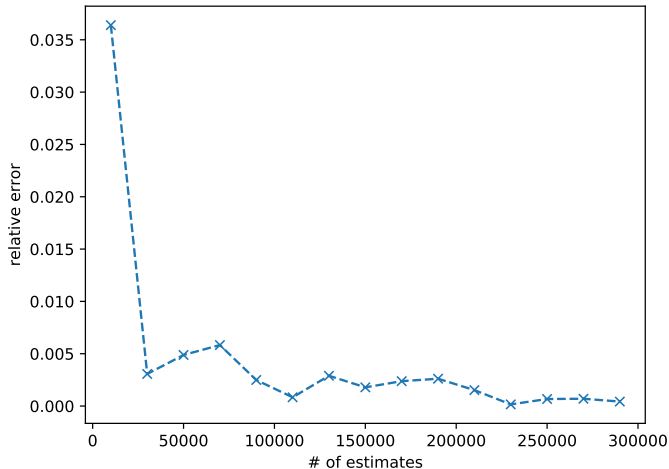


Figure: The convergence of the WoS method at the center point of the square boundary.

Future Work

- *Green's Function First Passage:*

$$p(x_0, x) = \frac{\partial G(x_0, x)}{\partial n}$$

If we know the Green's Function for a domain, we can directly simulate the transition from the starting point to the boundary. Then we can get all the estimates directly.

- *Parallelization on GPU:*

The Monte Carlo methods here are naturally parallel, so they may be accelerated a lot with GPU computing.

- *How things work for other PDEs:*

e.g. Diffusion Equation etc.

Thank you! Questions?