Monte Carlo methods for Solving PDEs
- Report for Computational Physics Final Project

Yucheng Zhang*

Department of Physics, New York University

(Dated: December 21, 2017)

Abstract

In this report, Monte Carlo methods for solving Partial Differential Equations are introduced, including the Simple Random Walk method and the Walk on Spheres method. We mainly focus on the Laplace's Equation with Dirichlet Boundary Conditions. The methods are implemented and tested on two practical bounadry value problems with square and circle boundaries. The properties of the Walk on Spheres method are analyzed.

INTRODUCTION

There are many numerical methods for solving different types of Partial Differential Equations (PDEs) in Boundary Value Problems (BVPs), such as the relaxation method and the Gauss-Seidel method, which we have learnt in the lecture. Obviously, these two methods are both deterministic methods. In this report and the final project, we explore the Monte Carlo methods for solving PDEs with boundary value conditions.

In this report, we mainly focus on Laplace's Equation with Dirichlet Boundary Conditions, i.e.

$$\nabla^2 u = 0 \tag{1}$$

inside the boundary, and

$$u = f(x) \tag{2}$$

given on the boundary. For a two dimensional problem, the Laplace's Equation Eq. 1 can be discreted with finite difference approximation. The centered difference reads,

$$u_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}), \tag{3}$$

as shown in FIG. 1. In this report, we take 2D problem as the example. But we should notice that this discrete form can be easily generalized to higher dimensions.

For the relaxation method, we use this equation, start from the boundary, and iterate until we get the desired accuracy. Here we look at this equation in a different way. We should notice that the value at one point is the statistical average of its neighbours' values. So we can regard the $\frac{1}{4}$ as the statistical weight of each neighbour while calculating the value of the center point. Then we can construct the Monte Carlo method based on this point of view.

Simple Random Walk Method

Given the point P_0 whose value we want to evaluate on the grid,

1. Pick a neighbour point randomly, e.g. P_1, with proper probability distribution, here it's $\frac{1}{4}$ for each point.

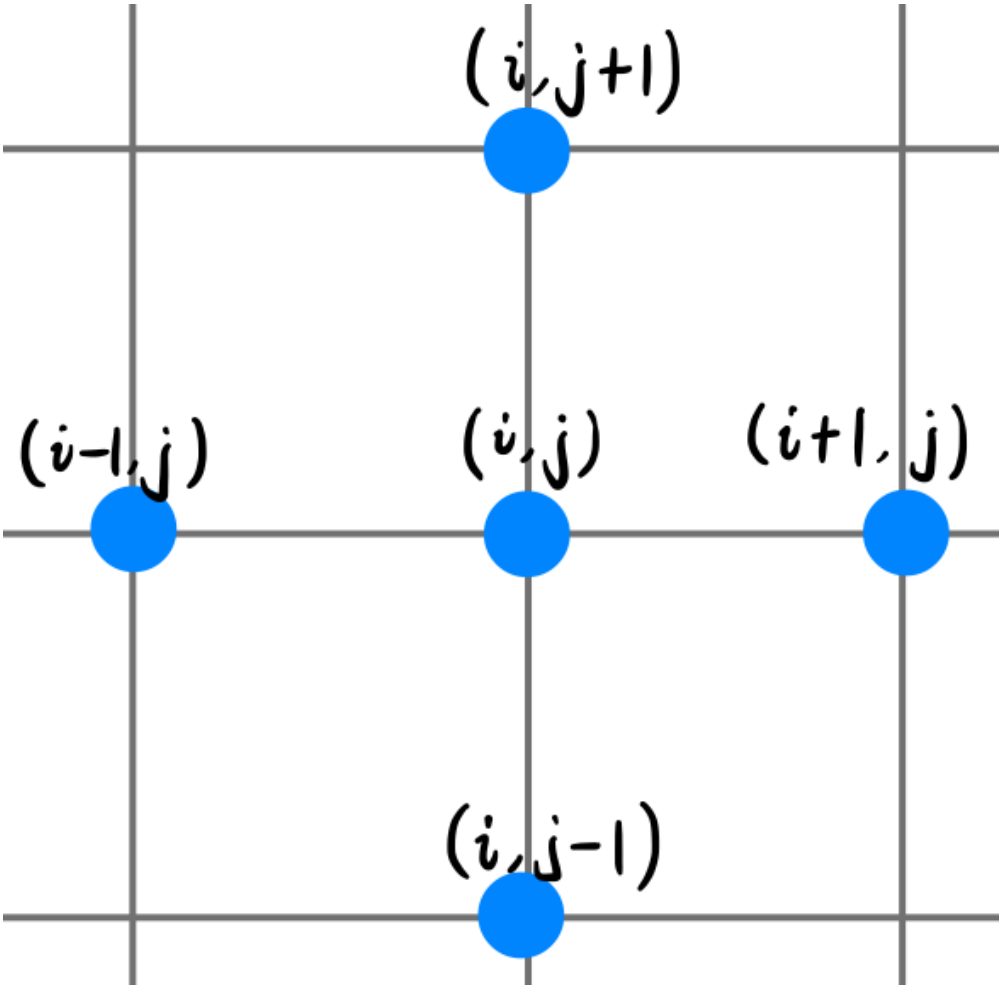

FIG. 1: Finite centered difference on 2D.

2. We don't know the value at P_1, but we know it's the average of its neighbours, so pick one of it's neighbours randomly with proper probability distribution, e.g. P_2. Continue this process, until we reach a point on the boundary, e.g. P_n, whose value is given.

3. We know that P_n is an estimate of P_{n-1}, which is an estimate of P_{n-2}, etc, etc, all the way back to P_0. So after this random walk process, we get the value of P_n as an estimate of P_0.

4. As a Monte Carlo method, we do a large amount of random walks from P_0 to the boundary, then average all the estimates, we can get a good estimate of the value at P_0, whose accuracy depends on the number of random walks, i.e. the number of estimates.

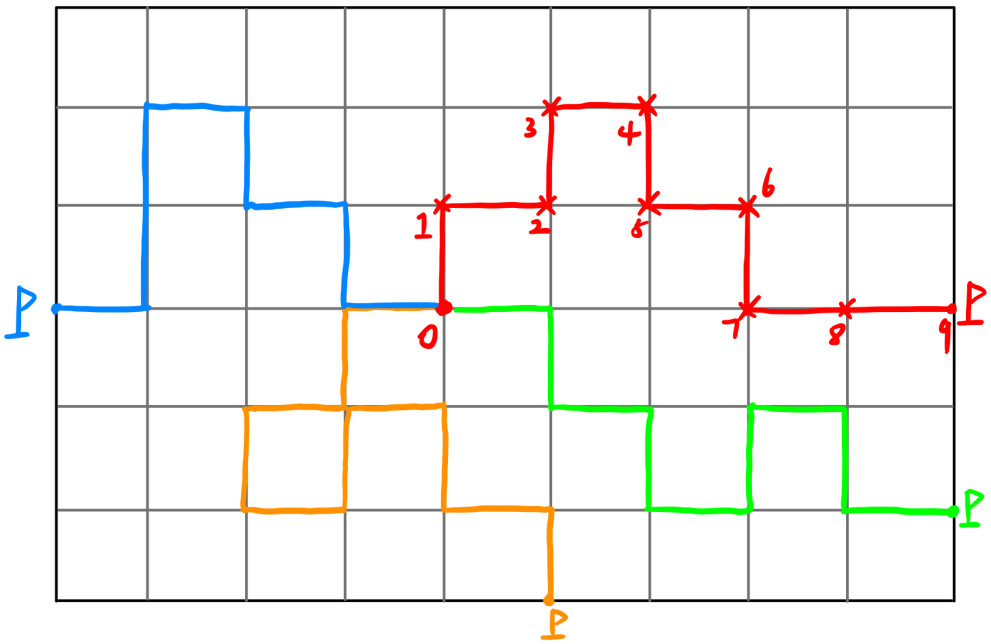

FIG. 2: Simple Random Walk method.

This is the process of the Simple Random Walk method, which is very straightforward based on the discrete form of the Laplace's Equation.

One more thing worth being mentioned here is that this method can be easily generalized for the Poisson's Equation,

$$\nabla^2 u = q(x, y) \tag{4}$$

with boundary value condition. The discrete form of Poisson's Equation reads,

$$u_{i,j} = \frac{1}{4}(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}) - \frac{1}{4}q_{i,j}. \tag{5}$$

Then after one random walk process with n steps, the estimate is given by,

$$u_n - \frac{1}{4}\sum_{k=0}^{n-1} q_k, \tag{6}$$

where u_n is the value on the boundary and q_k's are the values on the path. More details can be found in references, in the following discussion, we still keep concentration on the Laplace's Equation.

Walk on Spheres

As we should notice, for the Laplace Equation, one estimate given by one random walk is just the value at the point on the boundary where the random walk ends. So the estimate doesn't really depend on the path of the random walk. If we let the grid spacing go to zero, which obviously will increase the accuracy of the estimate, then the random walk here becomes simple Brownian motion.

Consider a sphere centered at $\boldsymbol{r}$, for a simple Brownian motion starting from $\boldsymbol{r}$, the possibility for the motion to reach any point on the sphere is the same. So since the path doesn't matter here for the Laplace equation, why do we simulate the Brownian motion step by step? We can jump to one point on the sphere directly! This will lead us to the Walk on Spheres method.

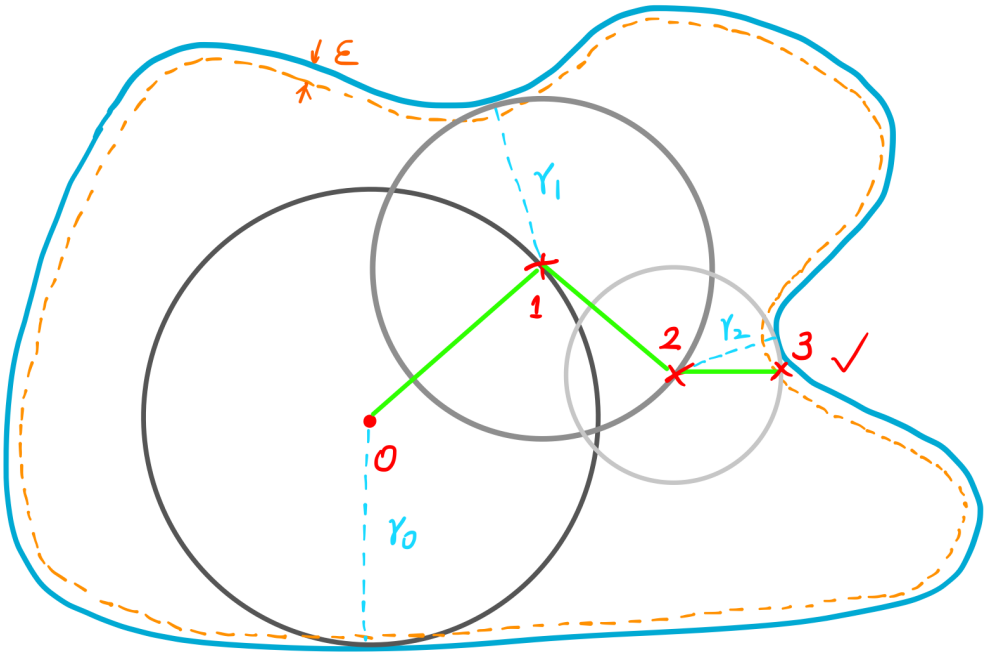

FIG. 3: Walk on Spheres method.

The process of the Walk on Spheres (WoS) method is shown in FIG. 3. Here are some explainations.

- The radius of the spheres (circles in 2D) are determined by the shortest distance from the point to the boundary, e.g. r_0, r_1, and r_2 in FIG. 3.
- The next point is chosen randomly on the sphere, e.g. points $1, 2, 3$ in FIG. 3.

- Since the WoS method is grid-free, we use the shell with thickness ϵ to judge whether the random motion reaches the boundary, as shown in FIG. 3.

Since most of the time in the Simple Random Walk method is wasted on the random walk, we can expect the WoS method to be much faster. Also, the WoS method is grid free, so there is no discretization errors. The error for one estimate is caused by the shell thickness ϵ, which will be analyzed in the implementation section.

Characteristics of Monte Carlo method

Both the Simple Random Walk (SRW) and Walk on Spheres (WoS) methods are Monte Carlo methods. The Monte Carlo methods here have some characteristics and advantages that normal deterministic methods don't have.

Independence of points

The first thing to notice is that unlike the relaxation or Gauss-Seidel methods, the points on the grid for the SRW or on the grid-free WoS are totally independent. So it's very fast to estimate some point values instead of the whole space.

Boundary & Dimension

For the Monte Carlo methods here, the boundary doesn't matter a lot. The only thing is to decide whether the random walk reaches the boundary, and at which point. So it's relatively easy for these methods to handle some problems with complex boundaries. Also, it's easier for these methods to be extended to higher dimensions.

Parallel Computing

Not only different points, as mentioned above, but also different estimates for one point are independent. So these Monte Carlo methods are naturally parallel. The two most straightforward ways to parallelize the program are,

- Parallelize the different estimates for one point, then average the results from different processes.

- Parallelize the set of points to be evaluated.

In our simulation, we mainly take the first way.

IMPLEMENTATION & ANALYSIS

In this section, we first introduce how the programs are implemented. Then we use the methods to study two practical problems and analyze the properties of these methods.

Parallelization in *Python*

In *Python*, there are many packages for parallel computing. In this project, we use the *multiprocessing* package. For a given set of points to be evaluated, we allocate the total number of independent estimates of every point to all available cores. Then we initialize the random number generators in all processes with different seeds. Here we use the *numpy.random* module to produce random numbers, which can be initialized with *numpy.random.RandomState(seed)*.

Square Boundary Problem

Assume we have a 2D Laplace problem with the boundary given as a square. The results are shown in FIG. 4 and FIG. 5.

Circle Boundary Problem

Here we study the 2D Laplace problem with a circle boundary. The result is shown in FIG. 6.

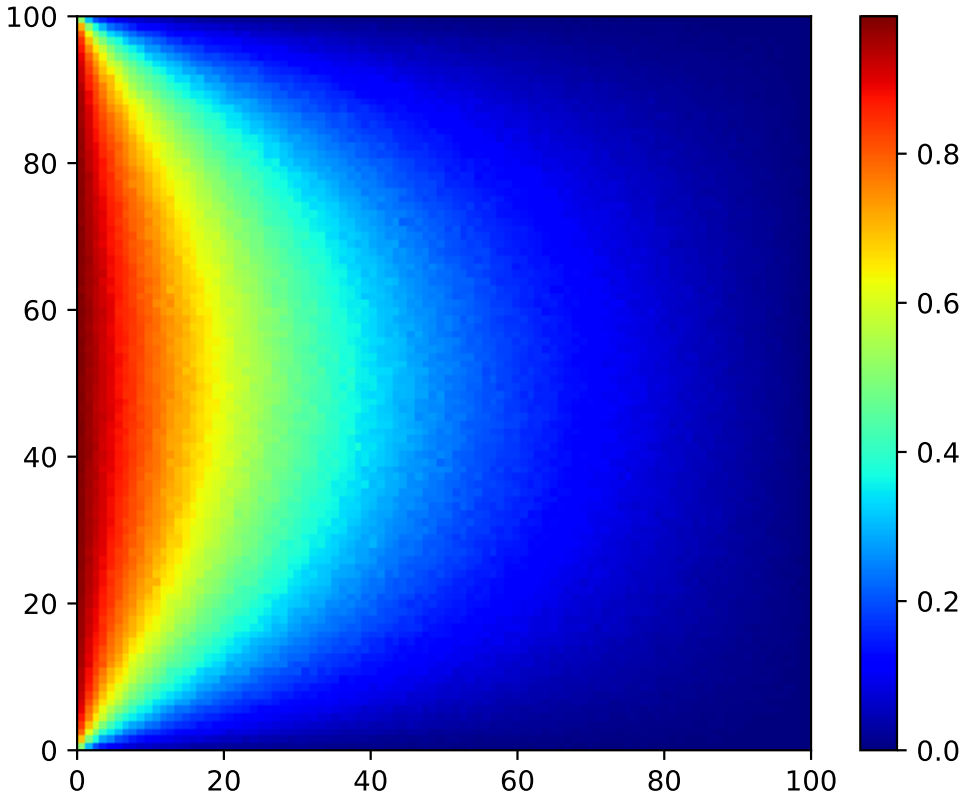

FIG. 4: SRW method on 2D Square Boundary with Square Lattice.

WoS method - The Influence of ϵ

For the Walk on Spheres method, here we study the influence of the shell thickness ϵ on the running time of the program. We evaluate a certain set of points with a fixed number of estimates for each point. The relation between the running time and ϵ is shown in FIG. 7. As we can see, for the two boundaries we simulate here, the square boundary and the circle boundary, the relation between the running time t and the shell thickness ϵ goes like,

$$t = -k \log \epsilon + t_0, \tag{7}$$

where k and t_0 depend on the shape of the boundary and the set of points to be evaluated. This logarithmic relationship here means that we can decrease the order of magnitude of ϵ to increase the accuracy of the estimates with an acceptable cost of time.

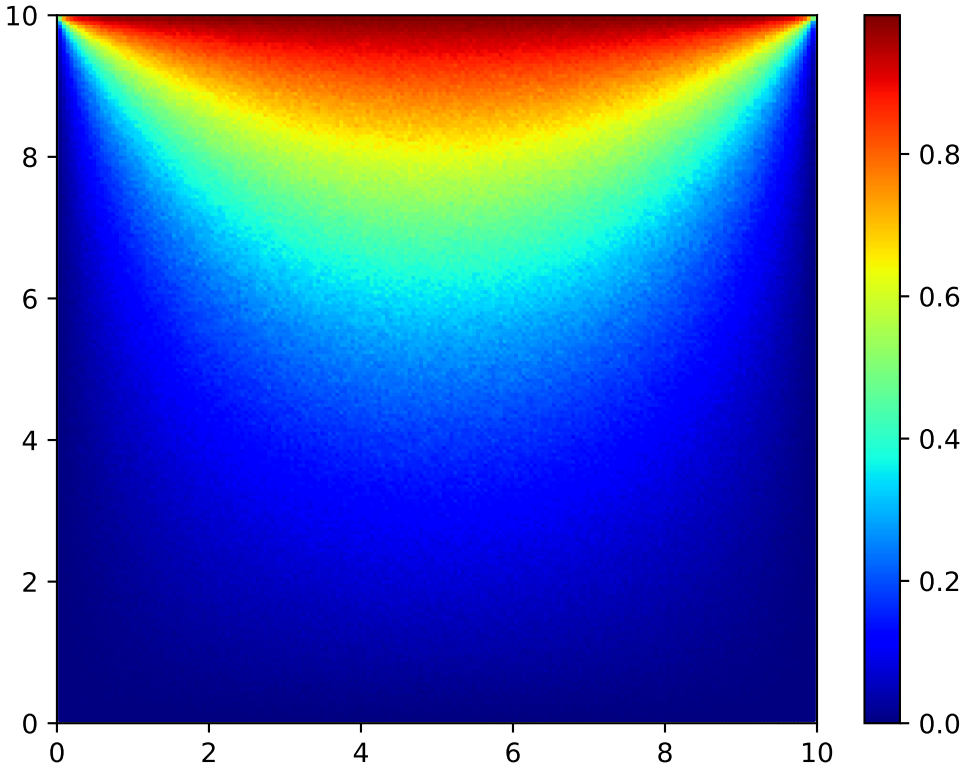

FIG. 5: WoS method on 2D Square Boundary.

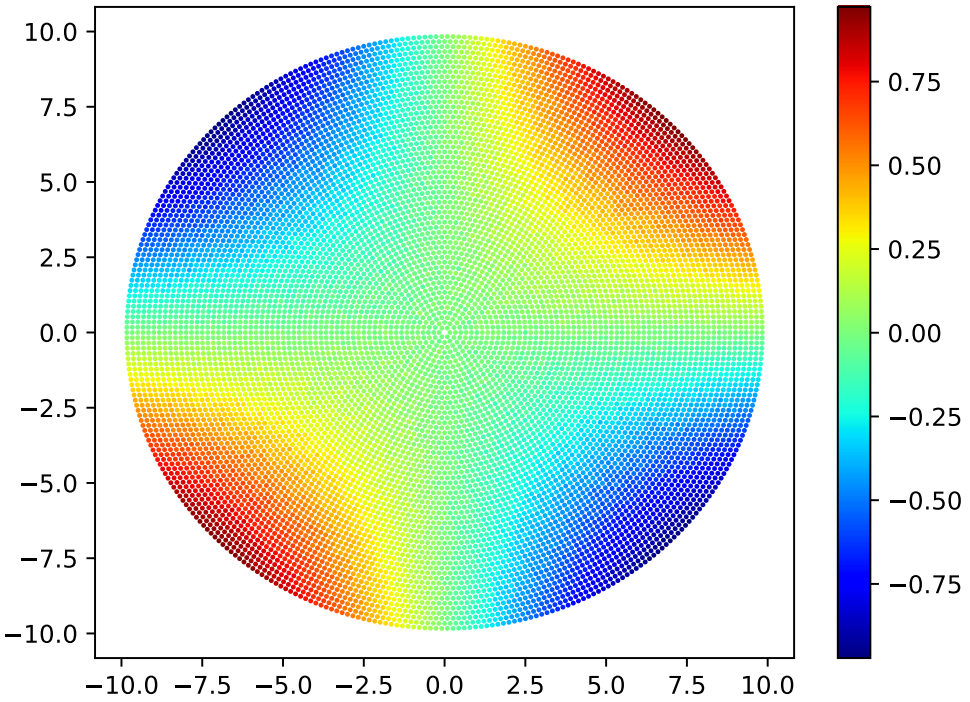

FIG. 6: WoS method on 2D Circle Boundary.

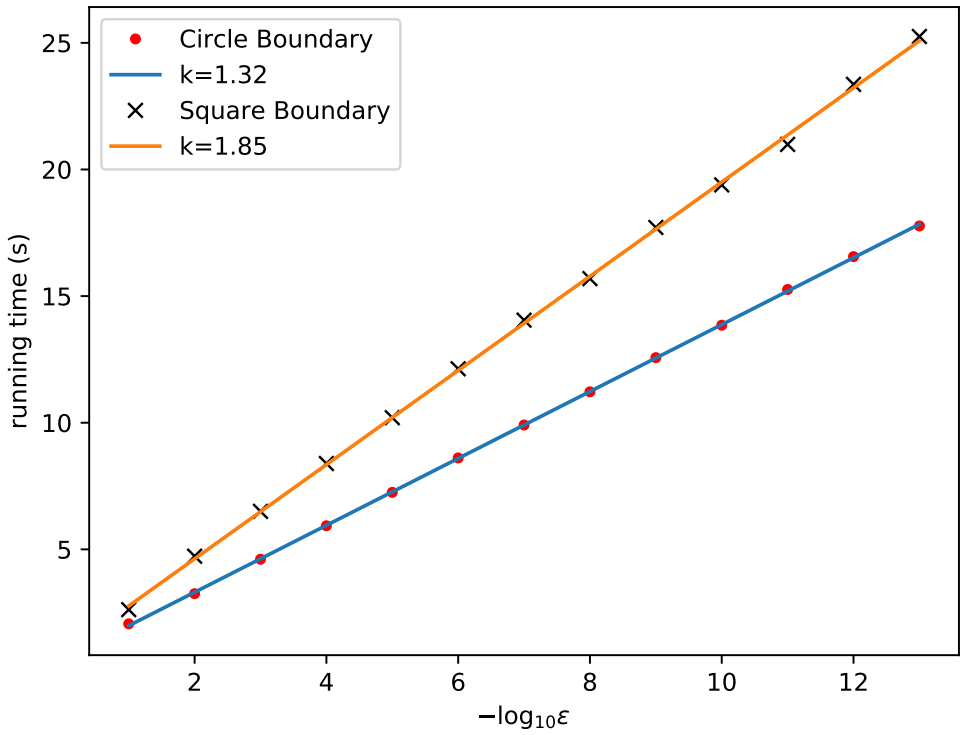

FIG. 7: The running time - ϵ relation on both square and circle boundary.

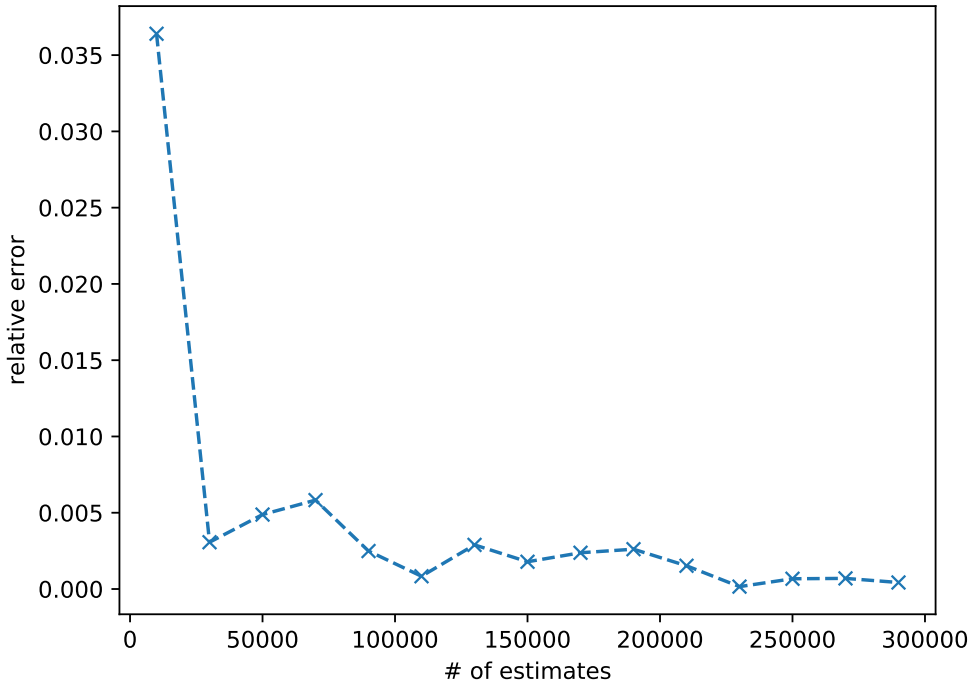

FIG. 8: The convergence of the WoS method at the center point of the square boundary.

WoS method - Convergence

In this subsection we study the convergence of the Walk on Spheres method. Here we pick the point at the center of the square bounadry, and study how it's value changes with the number of estimates.

FUTURE WORK

Some future work directions include,

- *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 accelarated a lot with GPU computing.

- *How things work for other PDEs*:
  e.g. Diffusion Equation etc.

ACKNOWLEDGEMENT

I'd like to thank Prof. Andrew MacFadyen for his instructive lectures during the class. I would also like to thank TA Marc Williamson for helpful discussions.

REFERENCES

- Gregory F. Lawler. *Random Walk and the Heat Equation*
  `http://www.math.uchicago.edu/~lawler/reu.pdf`

- Kevin Hu. *Partial Differential Equations and Random Walks*
  `http://www.math.toronto.edu/courses/mat394h1/20139/Random_Walk_and_PDE_1.pdf`

- Eric Guan. *Random Walks in Z^d and the Dirichlet Problem*
  http://www.math.uchicago.edu/~may/VIGRE/VIGRE2011/REUPapers/Guan.pdf
- *Monte Carlo Methods in Partial Differential Equations*
  http://www.hep.fsu.edu/~berg/teach/mcmc05/homework/Fleming_PDEs.ppt
- Michael Mascagni. *Monte Carlo Methods and Partial Differential Equations: Algorithms and Implications for High-Performance Computing*
  https://engineering.jhu.edu/ams/wp-content/uploads/sites/44/2015/04/Exascale_Mascagni.pdf
- K.K.Sabelfeld and N.A.Simonov. *Random Walks on Boundary for Solving PDEs*
  http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.7107&rep=rep1&type=pdf

---

* yz4035@nyu.edu