End of Term Report

Harini Chandramouli [1] , Kiya Holmes [2], Brandon Reeves [3], Nora Stack [4]

Abstract

In this research we are looking at Kakutanis classical result on the connection between Brownian motion, a form of random movement, and harmonic functions, which are solutions to the Laplace equation. Kakutanis theorem is basically a generalization of the mean value property of harmonic functions. We will use this result to solve the Laplace equation in various regions with certain boundary conditions.

Walk on Spheres (WoS) is used to simulate the Brownian motion of a particle suspended in liquid. The average time needed for the particle to hit the boundary of certain regions will be discussed. The distribution of the point of first encounter with the boundary of the region is of interest to us. We will also discuss our use of conformal maps to find probability density functions on certain regions. Additionally, the rate of convergence of the Brownian motion to the boundary as well as the overall computational effort needed to estimate values of the harmonic function using the Monte Carlo algorithm will also be discussed. Lastly, we looked into less expensive real world applications of our research.

Acknowledgments

First, we wish to thank our advisor Dr. Igor Nazarov, for all of his guidance throughout the REU program. Additionally, we would also like to acknowledge Dr. Nicholas Boros for his help with our research during the REU program. We would like to extend our thanks Michigan State University and Lymann Briggs College for being our host institution. Finally, we would like to thank the National Security Agency for funding our research through grant number H98230-11-10222.

[1]University of Pittsburgh, Pittsburgh, PA 15213
[2]Medgar Evers College, Brooklyn, NY 11225
[3]Gonzaga University, Spokane, WA 99207
[4]St. Mary's College of Maryland, St. Mary City, MD 20686

Contents

1 Introduction **4**

2 A Solution to Laplace's Equation: The Half-Plane **5**

3 Walk on Spheres Method **6**

4 Our Programs **6**

4.1 General Process 6

4.2 Various Regions 7

4.2.1 Line 7

4.2.2 Upper Half-Plane 9

4.2.3 Circle 9

4.2.4 Parabola 13

4.2.5 Square 14

4.2.6 Triangle 15

4.2.7 Upper Quarter-Plane 16

4.2.8 Upper Half Space 18

4.2.9 Sphere 19

5 Rates of Convergence **19**

6 Probability Density Functions for Known Regions **21**

6.1 Half-Plane 21

6.1.1 Goodness of Fit 22

6.2 Circle 22

7 Conformal Mappings and Probability Density Functions **23**

7.1 Quarter-Plane to Half-Plane 23

7.2 Showing u is Harmonic 23

7.3 Empirical Probability on the Quarter-Plane 25

7.4 Conformal Mappings on the Quarter-Plane 25

7.5 Finding the Theoretical Probability Density Function for the Quarter-Plane 28

7.6 Conformal Mapping of a Parabolic Region 29

8 Real World Application **30**

8.1 Obtaining Efficient Estimators - Gaussian quadrature 31

8.2 Selecting x_is 32

9 Conclusion **34**

References **35**

1 Introduction

Given a region $\mathcal{R}$ with boundary condition u_0, a problem central to applied mathematics is solving for heat dissipation, population migration, chemical diffusion, etc. for points interior to $\mathcal{R}$. Using Fourier transforms, one can find an expression for the temperatures (population densities, chemical concentrations, etc.) at any point inside of the region $\mathcal{R}$ at a given time t. By letting t tend towards infinity, we obtain a solution that is no longer dependent on time, i.e. a steady-state solution. It is well-known that the functional solution to the steady-state equilibrium, say u, is a harmonic function, that is, $\Delta u = 0$.

In 1944, however, Kakutani (see [1]) showed that one can express the steady-state solution in terms of Brownian motion. He showed that to find the value of the steady-state solution at a particular point,one simply needs to consider a particle undergoing Brownian motion beginning from that point. Under Brownian motion, the particle will travel randomly until it first encounters the boundary. When this happens, one records the value at the boundary at the point of first encounter. After repeating this process a sufficiently large number of times, the mean of the recorded values will converge probabilistically to the value of the harmonic function inside of $\mathcal{R}$.

One efficient way to simulate Brownian motion is the Walk on Spheres method (introduced in [2]). We will use the Walk on Spheres method with discrete time steps in order to simulate Brownian motion. In this process, a walker beginning at an initial point takes a random step to a new point. From this point, the walker again takes a random step to a new point. This process continues until the walker is sufficiently close to the boundary.

In this paper, we discuss the probability density function for the point of first encounter. In essence, we wish to find a function that describes the relative probability that our random walk process will terminate on a segment of our boundary given an initial starting point. Furthermore, we shall explore how altering the lengths of the random walks will influence the rates of convergence to the boundary. Lastly, our paper will consider efficient ways to approximate a solution to the steady-state equilibrium while calculating the fewest number of points along the boundary.

2 A Solution to Laplace's Equation: The Half-Plane

In this section we will solve Laplace's equation on the half-plane, which will be pivotal in our exploration of Laplace's equation in more generalized regions. Find $u(x, y)$ where

$$\begin{aligned} u_{xx} + u_{yy} &= 0 \text{ with} && (1) \\ u(x, 0) &= u_0(x) \text{ for all x} && (2) \end{aligned}$$

Recall, that the Fourier transform of u(x,y) is $\tilde{u}(\omega, y)$ where

$$\tilde{u}(\omega, y) = \int_{-\infty}^{\infty} e^{i\omega x} u(x, y) dx$$

Fourier transforming both (1) and (2) with respect to x we obtain the new system:

$$\begin{aligned} -(\omega)^2 \tilde{u} + \tilde{u_{yy}} &= 0 && (3) \\ \tilde{u}(\omega, 0) &= \tilde{u_0}(\omega) && (4) \end{aligned}$$

Notice, that (3) is simply an ordinary differential equation with characteristic equation that has the solution:

$$\tilde{u}(\omega, y) = e^{-|\omega| y} \cdot C(\omega) \qquad (5)$$

Utilizing (4) we find that

$$\begin{aligned} \tilde{u}(\omega, 0) &= e^{-|\omega| * 0} \cdot C(\omega) \\ &= C(\omega) \\ &= \tilde{u_0}(\omega) \end{aligned}$$

Hence, we know

$$\tilde{u}(\omega, y) = e^{-|\omega| y} \tilde{u_0}(\omega) \qquad (6)$$

By applying the inverse Fourier transform to (6) and recognizing that the inverse Fourier transform of $e^{-|\omega| y}$ is simply the Poisson kernel, we find that

$$u(x, y) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{y}{(x-t)^2 + y^2} u_0(t) dt \qquad (7)$$

From (7) we can find the probability density function for the point of first encounter for a particle undergoing Brownian motion from the point (x,y). Recall, that the expected value of a random variable is defined to be

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx$$

where f(x) is the probability distribution of the random variable X and x is the value of the random variable X.

Analogously, from (7) we know that the probability density function for the point of first encounter for a particle undergoing Brownian motion on the half-plane beginning from the point (x,y) is:

$$f(t) = \frac{1}{\pi}\frac{y}{(x-t)^2 + y^2} \qquad (8)$$

3 Walk on Spheres Method

Walk on Spheres is a method to simulate Brownian motion. It makes the process go much faster and maintains accuracy. It works as follows:

1. Pick point (x_0, y_0) in the region
2. Create circle with (x_0, y_0) as the center
3. Randomly pick a point (x_1, y_1) on the circle
4. (x_1, y_1) is either on the boundary or is somewhere else in the region
5. If (x_1, y_1) if on the boundary then walk on circles ends
6. If not, create circle with (x_1, y_1) as center
7. Randomly pick a point (x_2, y_2) on the circle
8. Continue process until you hit boundary of region

4 Our Programs

4.1 General Process

Our programs generally worked in this way.

1. Choose starting point (x_0, y_0)

2. Make a circle with (x_0, y_0) as the center and the largest radius possible in the region
3. Using rand function in MATLAB the program choose a random point (x_1, y_1) on the circle
4. (x_1, y_1) is either on the boundary(or are within a certain tolerance of the boundary) of the region or it is inside the region
5. If (x_1, y_1) is on the boundary(or is within a certain tolerance of the boundary) then the program ends
6. If (x_1, y_1) is not on the boundary, then the program uses (x_1, y_1) as the center of a new circle with the largest radius possible in the region
7. Using rand function in MATLAB the program choose a random point (x_2, y_2) on the circle
8. We continue this process until we have reached the boundary(or are within a certain tolerance of the boundary) of the region and then the program ends
9. We then run the program many times to compute an average

4.2 Various Regions

4.2.1 Line

We made a MATLAB program to simulate Brownian Motion on a line from a fixed starting point between 0 and n, where $n \in \mathbb{N}$. We pick an initial starting point, x_0 and then MATLAB will find the closest end point, which will either be 0 or n and make a circle of radius x_0 or $n - x_0$ accordingly. We shall use an example problem to show the accuracy of our program.

There is a rod of length 5.0 m, the left side is constantly at 100 °C **and the right side is constantly at** 1000 °C**. The rod is composed of a uniform heat-conducting material. There is no exchange of heat with the surroundings except at the end points. What is the temperature at the point 3.0 m from the left end of the rod?**
Solving Using Laplace's Equation
The equation in question is as follows,

$$\frac{d^2u(x)}{dx^2} = 0$$

We solve this by integrating both sides twice, we get

$$u(x) = c_2x + c_1$$

Now applying the boundary conditions we get a system of equations

$$\begin{aligned} 100 &= c_1 \\ 1000 &= 5c_2 + c_1 \end{aligned}$$

Plugging in the first equation into the second gives you the following values for the constants

$$u(x) = 180x + 100$$

Plugging in the value of x=3, we will get that the exact value of the temperature at 3.0 m is 640 °C.

Solving Using Brownian Motion and Walk on Spheres
Here we will use the Matlab program that was made. The program will start with the point at $x = 3$ and every time that the particle terminates movement at the left end point, the program will store it as 100. Every time the particle terminates movement at the right end point, the program will store it as 1000. Each run will iterate 1,000,000 times. Recorded below will be the temperature found during each run of the program.

Run Number	Temp. at 3.0 m
1	639.7651
2	639.7003
3	639.8578
4	640.2610
5	639.4429
6	639.7768
7	639.3925
8	640.6273
9	640.4707
10	639.8200

Run Number	Temp. at 3.0 m
11	640.4941
12	639.7633
13	639.6985
14	639.8614
15	640.2628
16	639.4384
17	639.7777
18	639.3952
19	640.6282
20	640.4662

Average=640.192

Thus, the temperature is found to be 640.192 °C

Error Calculation
The percent error is as follows

$$\left(\frac{640.192 - 640}{640}\right)100 \quad = \quad 0.03\%$$

4.2.2 Upper Half-Plane

The first 2D program that we made was for the upper half-plane region, that is, when $y > 0$ on the Cartesian Plane. In this program, an initial point (x, y) is typed in to the program and plotted in cyan. A circle is drawn using this initial (x, y) as the center. The radius of the circle is the value y as that will result in the circle being as large as possible while remaining in the region. A random point was picked on this circle by having MATLAB randomly pick a θ such that 0 . This point was then used as the center of the next circle, the radius being the new point's y-coordinate. This continued until the point got within a certain tolerance of the x-axis. The tolerance was defined to be 0.001 so that as long as the radius of the circle remained larger than this, the program would continue. A for-loop was created to iterate this process n times. Two variables were defined so that we could see how many circles were drawn and what was the last x-coordinate before the process terminated.
Here are some images from the program, simulating the walk on spheres method on the upper half-plane. We can see here that the number of times

the program takes to terminate, or the count, can vary. We can also observe that the point at which the termination occurs changes every time. Doing this same process thousands of times would help estimate the temperature or number of animals at the initial point.

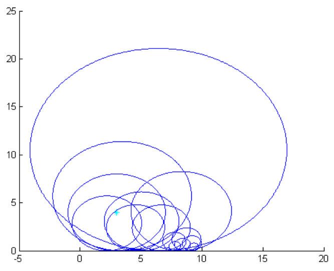


Figure 1: **Initial Point:** (3,4), **Count:** 16, **Tolerance:**0.001, **Ending Point:** (7.1225, 8.2200×10^{-7})

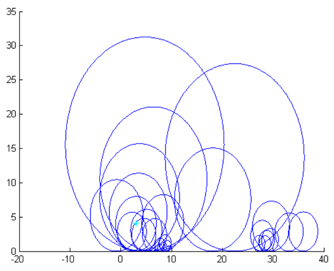


Figure 2: **Initial Point:** (3,4), **Count:** 16, **Tolerance:** 0.001, **Ending Point:** (36.4099, 8.1999×10^{-4})

4.2.3 Circle

This program simulates Brownian Motion on a circular region. To start, we must enter our initial starting point, the desired tolerance level, and the center and radius of the circular region. A circle is drawn with the initial point as the center and the maximum radius such that the circle stays within the region. To find this radius, we had to find the minimum distance from the point we wanted to be the center of the circle to circular region. This would occur where the normal to the tangent of the border points at the

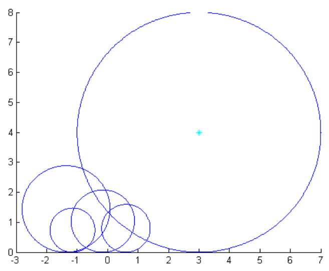


Figure 3: **Initial Point:** (3,4), **Count:** 5, **Tolerance:** 0.001, **Ending Point:** (-1.4108, 8.0219×10^{-4})

point in question. Let us call the the center of the circular region (x_c, y_c), the radius of the circular region R, the point we want to be the center of our circle (x_0, y_0), and the radius of the smaller circle we want to draw r. To find r, we first found the distance between (x_0, y_0) and (x_c, y_c), and then subtracted that away from R.Using this r, a circle is drawn around the initial point and then a random point is chosen on that circle. Using that point as the center, another circle is drawn with that point on as the center and the radius calculated as before. This process is repeated until the radius is within the set tolerance, and then the process stops.
Below are some images from the program. The largest circle is the boundary region. All the following images have the same starting point and tolerance, but in this case we can see how the count and ending point vary from figure to figure.

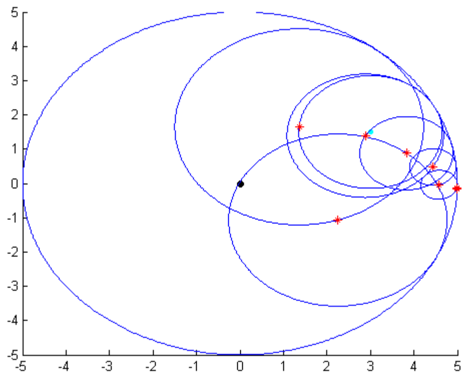


Figure 4: **Boundary Equation:** $x^2 + y^2 = 25$, **Initial Point:** (3,1.5), **Count:** 9, **Tolerance:** 0.001, **Ending Point:** (4.9972, -0.1506)

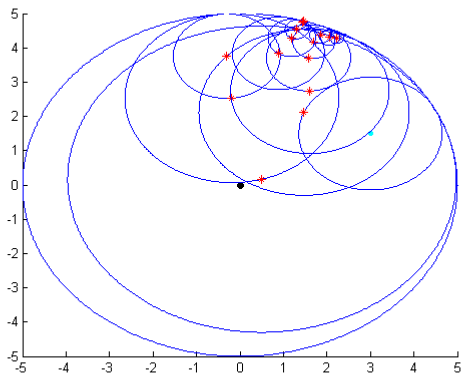


Figure 5: **Boundary Equation:** $x^2 + y^2 = 25$, **Initial Point:** (3,1.5), **Count:** 20, **Tolerance:** 0.001, **Ending Point:** (1.4661,4.7802)

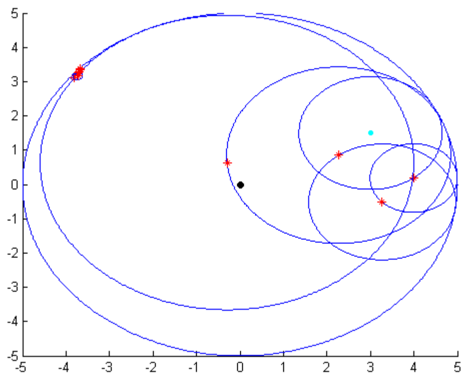


Figure 6: **Boundary Equation:** $x^2 + y^2 = 25$, **Initial Point:** (3,1.5), **Count:** 11, **Tolerance:** 0.001, **Ending Point:** (-3.6827,3.3815)

4.2.4 Parabola

We made a program to simulate Brownian Motion in the parabola $y = ax^2$, without loss of generality. This program is functioning, however it takes a while to run. We wanted to make a more efficient program, so we decided to increase the radius of our circles to be the maximum radius possible inside of the parabola. To do this, we came up with a method to find the smallest distance from our starting point (x_0, y_0) to the boundary. This unknown point on the boundary is (x_1, y_1). To find this radius, we used the distance formula squared.

$$d^2 = (x_1 - x_0)^2 + (y_1 - y_0)^2$$
$$d^2 = (x_1 - x_0)^2 + (ax_1^2 - y_0)^2$$

We then took the derivative of the distance formula squared to minimize the distance.

$$d' = 2(x_1 - x_0) + 2 * 2ax_1(ax_1^2 - y_0)$$
$$d' = 4a^2x_1^3 - 4ax_1y_0 + 2x_1 - 2x_0$$

We then set the derivative equal to zero and divided our equation by two to get these equations.

$$4a^2x_1^3 - 4ax_1y_0 + 2x_1 - 2x_0 = 0$$
$$2a^2x_1^3 - 2ax_1y_0 + x_1 - x_0 = 0$$

MATLAB then solves for the roots of this cubic equation. Our program then disregards any imaginary numbers and picks the smallest number of the remaining numbers. We then have our x_1 value and we get the corresponding y_1. We use the distance formula to calculate the distance between these two points which is used as our radius in the program. The program then performs walk on circles in the parabola.

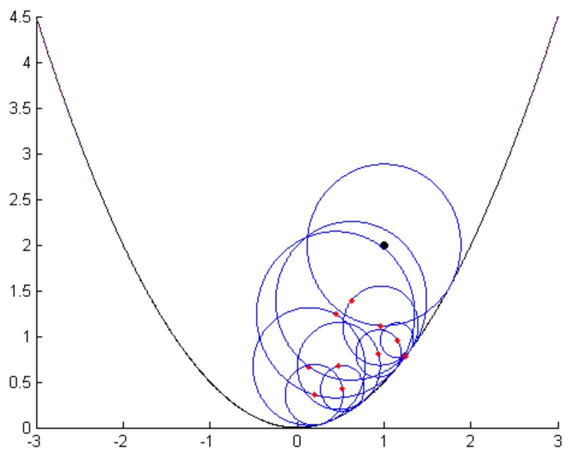
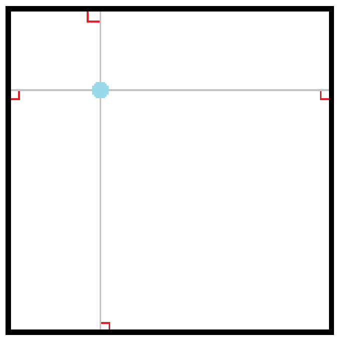


Figure 7: Walk on Circles in a Parabola with a starting point $(1, 2)$ and the maximum radius possible

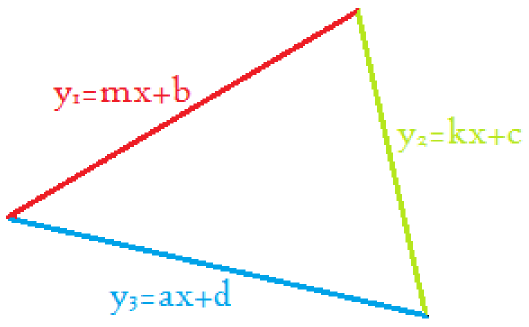
4.2.5 Square

We also created a program that simulates walk on circles in a square. Most of the program is similar to the others, except that we had to find the radius for the circles in walk on circles a little differently. Let our initial point be (x_0, y_0). We had to find the distance form this point to the boundaries of the square. You drop a line down from (x_0, y_0) to each of the lines that make up the square. Each of the lines that you drop down are perpendicular to the lines of the square so that you get the shortest distance from the point to the boundaries of the square. The intersections tell you the points to measure the distance to from (x_0, y_0) using the distance formula. This is further illustrated in the picture below:



The shortest of these distances was then used as the radius for the circles in our walk on circles program.

4.2.6 Triangle



Using the equations

$$y_1 = mx + b$$
$$y_2 = kx + c$$
$$y_3 = ax + d$$

we are able to construct a triangle. We then find the minimum radius by taking the derivative of each line. For example, in y_1, the derivative is m. We then have the slope of the tangent line. Using the slope of each, we calculate the slope of the perpendicular line. Using y_1, we have

$$y = \frac{-1}{m}x + b$$

We can then use our point (x_0, y_0) to find b. We then get an equation in the form

$$y = \frac{-1}{m}x + (y_0 + \frac{1}{m}x_0)$$

which we then set equal to y_1

$$mx + b = \frac{-1}{m}x + (y_0 + \frac{1}{m}x_0)$$

and we then find out where these lines intersect. Thus we get the point (x_1, y_1). Once we repeat this process for y_2 and y_3 we use the distance formula to figure out the distance between (x_0, y_0) and each of our new points. Our program then picks the minimum of the three distances as our radius for a circle centered at (x_0, y_0). If the triangle is very obtuse, the distance calculated has the chance to extend outside of the actual triangle. However, this distance will never be shorter than a distance to a point inside the triangle. Thus, it does not create a problem. The program then picks a point on that circle (x_k, y_k). If the point is on the boundary (or within our tolerance), then the program ends. If the point is not, then the program calculates the distance between (x_k, y_k) and our boundary points relative to (x_k, y_k) and finds the minimum distance. This value then becomes our new radius for the circle centered at (x_k, y_k). The program then picks a random point on the circle. We continue this process until we have reached a boundary. We then repeat the process n times to calculate the average.

4.2.7 Upper Quarter-Plane

The program that simulates the walk on spheres method in the upper quarter-plane was coded very similarly to the upper half-plane program, however here the radius of the circle was set to be the minimum of x and y as the program would terminate when the point got within the tolerance of either the $x - axis$ or $y - axis$.

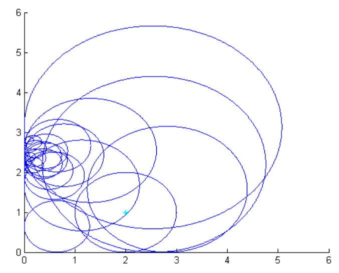


Figure 8: **Initial Point:** (2,1), **Count:** 21, **Tolerance:** 0.01, **Ending Point:** (0.6753,0.0051)

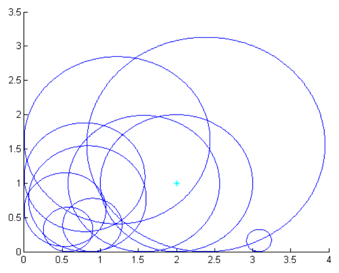


Figure 9: **Initial Point:** (2,1), **Count:** 10, **Tolerance:** 0.01, **Ending Point:** (3.0274,0.0097)

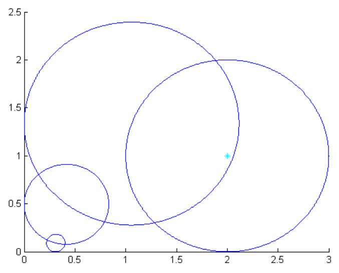


Figure 10: **Initial Point:** (2,1), **Count:** 4, **Tolerance:** 0.01, **Ending Point:** ($0.3139, 3.9515 \times 10^{-5}$)

The counter number can vary highly every time, even though the starting point and tolerance are set to be the same.

4.2.8 Upper Half Space

Next, we moved on to 3D spaces. Previously, we saw in the 2D case that we could randomly generate an point uniformly if we chose a random angle from 0 to 2π. However, in the 3D case, we cannot generate a point so easily. First, we would want to use spherical coordinates to generate a random point on a sphere. The ρ is given to be the shortest distance between the center of our sphere and the end point of the region in question. To randomly choose a point, we can randomly choose an angle from 0 to 2π for θ. However, for ϕ we need to make an adjustment as converting a Cartesian rectangle to a spherical rectangle creates a problem. ϕ goes from 0 to π but is not uniformly distributed. There is a higher chance of picking a point towards the poles than towards the equator. The adjustment is $\phi = \arccos(2u - 1)$ where u is uniformly distributed on (0,1).
We made a program that simulates Walk on Spheres in the upper half-space. We pick an initial point and our program then simulates a random walk on a sphere. We set ρ equal to z. The point that is produced is either on the plane $z = 0$, in which case the program ends, or it is somewhere else on the upper half-space. If it is somewhere else in the upper half-space, then the program then uses that point as the new center for the sphere and our new rho as the radius of the sphere. It then preforms a random walk again. The process continues until the program has the plane $z = 0$. The program then performs Walk on Spheres many times so that we can compute an average.
We also made a program to plot this process. However, the resulting image is not very clear.

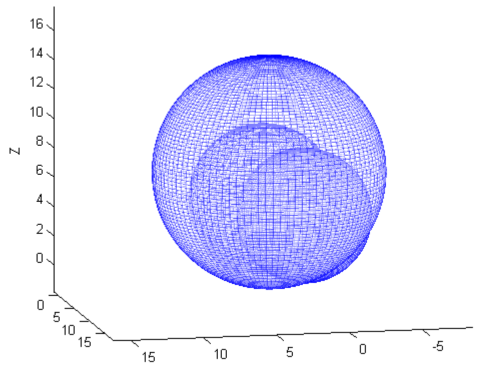


Figure 11: **Initial Point:** $(3, 4, 5)$, **Count:** 17, **Tolerance:**0.001, **Ending Point:** $(3.0538, 1.9212, 7.7757 \times 10^{-7})$

Visualizing 3D Movement in 2D We looked at ways to analyze the distance between points on a 3D plane and found a way that was efficient. We took the 3D plane and made the z-component equal to zero and projected our starting point on the xy plane. This way we can see the distance between the initial point and the terminated point. By doing so, we simplified the dimensions of the original problem. What we also did was apply height to the starting point. We wanted to analyze how the difference in height would affect the distance between the points. So we plotted the distances on histograms to compare. We saw that with more height, the distances were more dispersed and much higher. These results were reasonable and it gave us an idea of what to expect in the theoretical results.

4.2.9 Sphere

We also created a program that simulated Walk on Spheres within a spherical region. The inputs are the center, (x_c, y_c, z_c), and radius of the spherical region in question, along with our initial point, (x_i, y_i, z_i), to start the walk on spheres and the tolerance. The maximum ρ is calculated very similarly to the radius in the 2D Circle program. Let the radius of the spherical region be denoted by R, and let the radius of the sphere for the walk on spheres be denoted by ρ. ρ is calculated as follows:

$$\rho \ = \ R - \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2}$$

The program outputs the (x, y, z) at which the program terminated. However, this program only does the process once and does not create any images.

5 Rates of Convergence

Using MATLAB, we created programs to compare the rates of convergences when we changed certain conditions. In this program, we start with an initial point inside a given region, (x_0, y_0). This program is a lot like the walk on spheres programs where it draws a circle around this initial point with a certain radius and then chooses a random point on that circle uniformly. However, here, the radius is set at a very small number, such as 0.01. Also, the circle is not plotted. Thus, the particle jumps from one point to another that is close to the first point. This goes on until the point is within the tolerance of the border of the region in question. Depending on the region, this process can take 700 steps or more. If you increase the radius size, it goes faster (which is how the walk on spheres method works, it optimizes the radius size to make convergence to the border happen quicker). Increasing the radius to 1 can change the number of steps it takes to converge dramatically, taking less than 50 steps sometimes. This can be seen in the graphic, created by a program that we made, below where we compared the rates of convergence of various radii sizes in a circular region. This program finds the average number of iterations needed for convergence for 1000 iterations for each radii. The second graph is especially significant as we see that the

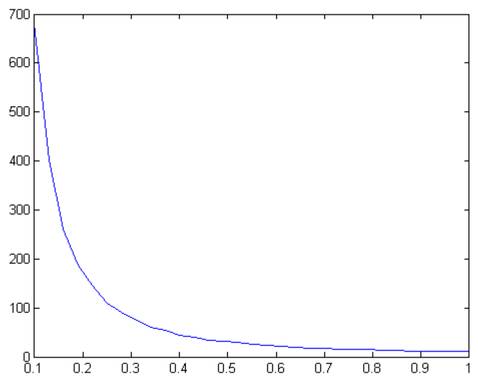


Figure 12: Number of Steps vs. Radius Size

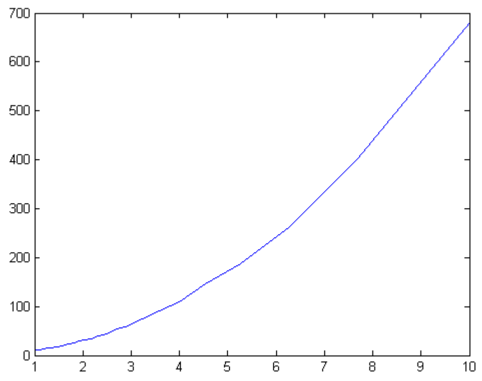


Figure 13: Number of Steps vs. $\frac{1}{RadiusSize}$

Number of Steps and the inverse of the Radius Size has a squared relationship, lining up with Einstein's Theory that distance is related to the square root of time (see [3]).

6 Probability Density Functions for Known Regions

6.1 Half-Plane

One of the main goals of our research this summer is to establish empirically a probability density function for the point of first encounter for a particle in a region $\mathcal{D}$ under Brownian motion. Using the random walk on circles method as a way to simulate Brownian motion, we have constructed an algorithm that will allow us to simulate Brownian motion on the half-plane. In order to find an estimate for the probability density function for the boundary of the half-plane (the x-axis), we can simulate Brownian motion several thousand times (in this case 10,000), record the point of first encounter for each iteration, and then construct a histogram displaying the frequencies of the points of first encounter. Since the number of iterations are large, the shape of the histogram should match the theoretical PDF for the boundary.

Fortunately enough, the theoretical PDF for the half-plane is known to be the Cauchy-distribution of the form : $\frac{1}{\pi} * \frac{a}{x^2+a^2}$ where is the y-coordinate of our initial starting point (i.e. the height of our initial point). When we executed the algorithm described above, we obtained a histogram describing the frequencies of the point of first encounter along the x-axis. We then plotted the known distribution on top of our histogram (with the appropriate scaling measures) and we obtained the following result:

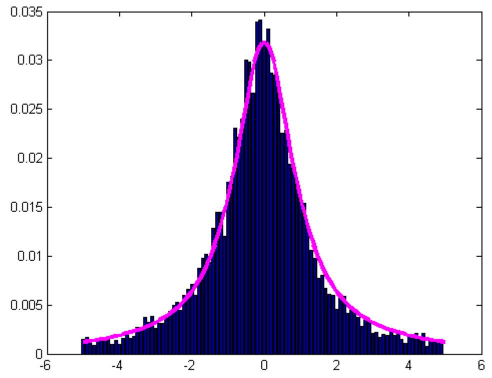


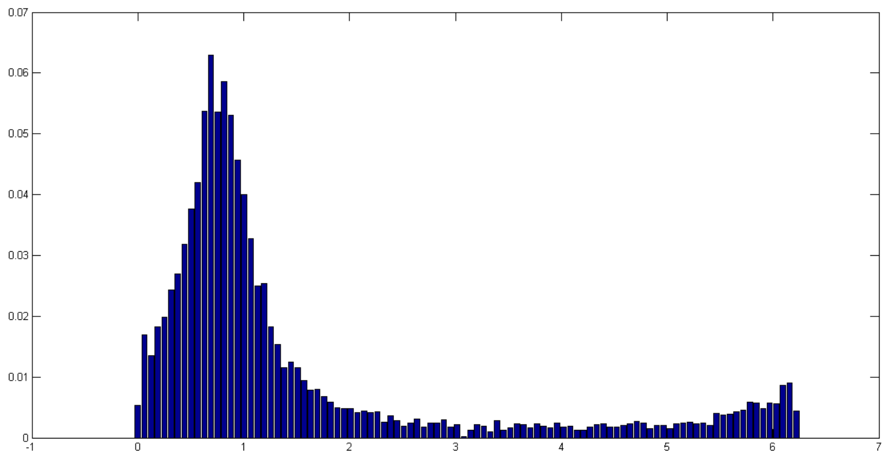
Figure 14: Empirical vs Theoretic and PDFs for half-plane

6.1.1 Goodness of Fit

We can perform Pearson's Chi-Squared Test to check if the events observed in our sample is consistent with the theoretical distribution. The test statistic is $\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$. This test statistic follows a Chi Square distribution with k - 1 degrees of freedom. Our findings last week were significant to the $\alpha = 0.05$ level.

6.2 Circle

As we created a program to make a histogram of the point of first encounter on the half-plane, we also created on to do the same for a circle. We linearized the circle, drawing a line from 0 to 2π. The circle is centered at $(0, 0)$ with radius of 2, and our initial starting point was at $(1, 1)$. Intuition suggests that the highest density will be found around $\frac{\pi}{4}$. The histogram found is below:



The histogram seems to agree with what we believe to be correct, but we still need to do a Goodness of Fit test for this density function also. However, we do not know the theoretical distribution on a circle.

7 Conformal Mappings and Probability Density Functions

To find the theoretical distribution for unknown regions, we can use conformal mapping ([5]). A conformal map maps one region bijectively into another region. It preserves angles between smooth curves going through a point.

7.1 Quarter-Plane to Half-Plane

There is a conformal map that takes the quarter-plane into the half-plane, and this is the map $f(z) = z^2$. We can use this map, in the future, to find the probability density function in the quarter-plane, given the fact that we know the probability density function in the half-plane. But first, let us look more into this map.
Since z is a complex number, we can say $z = x + \imath y$, for some arbitrary x and y such that $x, y \in \mathbb{R}$. By Euler's formula, we see that $z = x + \imath y = r\cos(\theta) + \imath r\sin(\theta) = re^{\imath\theta}$. Plugging in x and y into our map, we get

$$f(z) = z^2 = (x + \imath y)^2 = x^2 - y^2 + 2\imath xy$$

Let

$$g(w) = g(w_1, w_2) = u(w_1, w_2) + \imath v(w_1, w_2)$$

where $w = w_1 + \imath w_2$. Let $\Im(w) > 0$ and $\Re(g(w)) = u(w_1, w_2)$. We will now create the composite function

$$g(f(z)) = g(z^2) = g((x + \imath y)^2) = g(x, y)$$

Since $f(z)$ and $g(w)$ are both analytic functions, we know that $g(f(z))$ is analytic because the composition of analytic functions is analytic. We know that u is harmonic as shown in the next section, *Showing u is Harmonic*. It follows similarly that v is harmonic also, as it is the harmonic conjugate. The functions are harmonic for $x > 0$ and $y > 0$, that is, the quarter-plane.

7.2 Showing u is Harmonic

Given that $f(z)$ and $g(w)$ are analytic functions, $g(f(z))$ is analytic because the compositions of analytic functions are analytic. Let the real part of g be represented by u and the imaginary part of g be represented by v such that

$g = u + \imath v$.
Let $z = x + \imath y$ and $w = w_1 + \imath w_2$.

$$\begin{aligned} f(z) &= z^2 = (x+\imath y)^2 = x^2 - y^2 + 2\imath xy \\ g(f(z)) &= g(z^2) \end{aligned}$$

since $g(w) = g(w_1, w_2)$, where w_1 is the real and w_2 is imaginary

$$g(w_1, w_2) = g(x^2 - y^2, 2xy)$$

Thus, g is a function of x and y.
So, g is an analytic function in the region R, where R is the Quarter Plane, where $x, y \geq 0$. Since it is analytic in R, then the Cauchy-Riemann equations:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \tag{9}$$

$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y} \tag{10}$$

are satisfied. Assuming that u and v have continuous second partial derivatives, we can differentiate (9) with respect to x and differentiate (10) with respect to y.

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 v}{\partial x \partial y} \tag{11}$$

$$\frac{\partial^2 u}{\partial y^2} = -\frac{\partial^2 v}{\partial y \partial x} \tag{12}$$

We can add (11) and (12) together to get the following equation:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial^2 v}{\partial x \partial y} - \frac{\partial^2 v}{\partial y \partial x} \tag{13}$$

By Clairaut's Theorem on the symmetry of second derivatives, we have

$$\frac{\partial^2 v}{\partial x \partial y} = \frac{\partial^2 v}{\partial y \partial x}$$

thus, we can use this fact with equation (13)

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{\partial^2 v}{\partial x \partial y} - \frac{\partial^2 v}{\partial x \partial y} \tag{14}$$

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{15}$$

Thus, u is harmonic because it satisfies Laplace's Equation, $u_{xx} + u_{yy} = 0$.

7.3 Empirical Probability on the Quarter-Plane

This week we made a program that performs walk on circles n number of times and stores the points where the program terminates. We then made a program that linearizes this data so we can make a histogram. If the program terminated at a point on the y-axis, then the program linearizes the point to be on the negative x-axis. If the program terminated at a point on the x-axis, then the program linearizes the point to be on the positive x-axis. We then used our histogram plotter to make a histogram of this data. The resulting

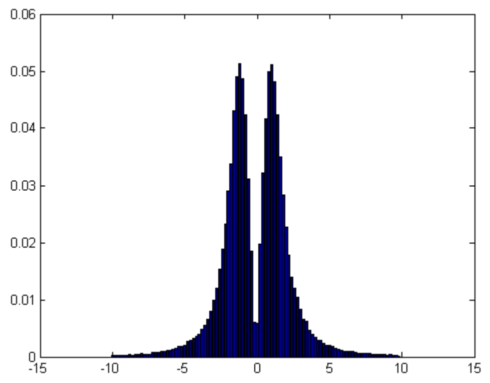


Figure 15: Approximate probability density of the quarter-plane with an **initial point** $(1, 1)$, **tolerance**=0.01, 100,000 **iterations** of Walk on Circles, and 100 **bins**.

histogram was in line with our expectations. The two biggest peaks were at 1 on the x-axis and 1 on the y-axis (which corresponds to -1 on the x-axis in histogram above). There is such a drastic dip at 0 because there is no way that the point could actually reach 0, it would reach the tolerance before it ever hit 0.

7.4 Conformal Mappings on the Quarter-Plane

Assume $u(x, y)$ satisfies Laplace's Equation in the half-plane,

$$\begin{aligned} f(z) &= z^2 \\ &= (x^2 - y^2) + (2xy)\imath \end{aligned}$$

We want to apply this conformal mapping to find h(x,y) in the quarter plane.

$$\begin{aligned} h(x,y) &= h(z) = u(f(z)) = u(x^2 - y^2, 2xy) \\ h_0(x) &= h(x,0) = u(x^2, 0) \\ h_1(y) &= h(0,y) = u(-y^2, 0) \end{aligned}$$

$$u_0(\xi) = \begin{cases} h_0(\sqrt{\xi}) \text{ if } \xi > 0 \\ h_1(\sqrt{-\xi}) \text{ if } \xi < 0 \end{cases}$$

Given the boundary condition $u_0(t)$ for $-\infty < t < \infty$, we know that our solution is

$$\begin{aligned} u(x_0, y_0) &= \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{y_0}{(x_0 - t)^2 + {y_0}^2} u_0(t)dt \\ &= \frac{1}{\pi}\int_{-\infty}^{0} \frac{y_0}{(x_0 - t)^2 + {y_0}^2} h_1(\sqrt{-t})dt \\ &+ \frac{1}{\pi}\int_{0}^{\infty} \frac{y_0}{(x_0 - t)^2 + {y_0}^2} h_0(\sqrt{t})dt \end{aligned}$$

Let t = -τ^2 for the first and t = τ^2 for the second integral.

$$\begin{aligned} u(x_0, y_0) = &\frac{1}{\pi}\int_{0}^{\infty} \frac{2\tau y_0}{(x_0 + \tau^2)^2 + {y_0}^2} h_1(\tau)d\tau \\ &+\frac{1}{\pi}\int_{0}^{\infty} \frac{2y_0\tau}{(x_0 - \tau^2)^2 + {y_0}^2} h_1(\tau)d\tau \end{aligned}$$

Recall that $u({x_0}^2 - {y_0}^2, 2x_0y_0) = h(x_0, y_0)$, so we need to substitute

$$\begin{aligned} h(x_0, y_0) &= u({x_0}^2 - {y_0}^2, 2x_0y_0) \\ &= \frac{1}{\pi}\int_{0}^{\infty} \frac{4x_0y_0\tau}{({x_0}^2 - {y_0}^2 + \tau^2)^2 + (2x_0y_0)^2} h_1(\tau)d\tau \\ &+ \frac{1}{\pi}\int_{0}^{\infty} \frac{4x_0y_0\tau}{({x_0}^2 - {y_0}^2 - \tau^2)^2 + (2x_0y_0)^2} h_0(\tau)(\tau)d\tau \end{aligned}$$

Thus, we know that the PDF for the quarter-plane is:

$$\frac{1}{\pi}\frac{4x_0y_0\tau}{({x_0}^2 - {y_0}^2 + \tau^2)^2 + (2x_0y_0)^2}$$

on the y-axis and

$$\frac{1}{\pi}\frac{4x_0y_0\tau}{({x_0}^2 - {y_0}^2 - \tau^2)^2 + (2x_0y_0)^2}$$

on the x-axis.

7.5 Finding the Theoretical Probability Density Function for the Quarter-Plane

We will first look at the equation we derived from the previous section.

$$\frac{1}{\pi}\int_0^\infty \frac{4x_0y_0\tau}{\left({x_0}^2-{y_0}^2-\tau^2\right)^2+\left(2x_0y_0\right)^2}d\tau$$

To solve this integral, we will first do a u-substitution. As this is a function of τ, we will take the derivative with respect to τ

$$\begin{aligned} u &= {x_0}^2-{y_0}^2-\tau^2 \\ du &= -2\tau d\tau \end{aligned}$$

Plugging this into the integral, we see

$$\begin{aligned} &\frac{1}{\pi}\int_{-\infty}^{{x_0}^2-{y_0}^2} \frac{2x_0y_0}{u^2+\left(2x_0y_0\right)^2}du \\ &\left[\frac{1}{\pi}\arctan\left(\frac{u}{2x_0y_0}\right)\right]_{-\infty}^{{x_0}^2-{y_0}^2} \\ &\frac{1}{\pi}\arctan\left(\frac{{x_0}^2-{y_0}^2}{2x_0y_0}\right)+\frac{1}{2} \end{aligned}$$

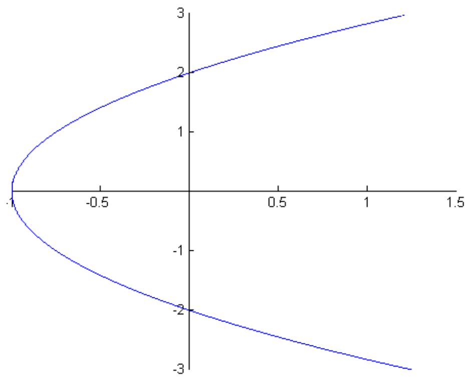
This is the formula we found for the probability density function on the x-axis. To find the probability density function on the y-axis, just take 1 minus the function on the x-axis, which is

$$\frac{1}{2}-\frac{1}{\pi}\arctan\left(\frac{{x_0}^2-{y_0}^2}{2x_0y_0}\right)$$

A way to test to see if this is accurate is to plug in a point on the line $y = x$. Intuition suggests that any point on this line would have a $\frac{1}{2}$ chance of landing on either the x or y-axis. Plugging $y = x$ into this formula gives us $\frac{1}{2}$ for both the probability of landing on the x and y-axis.

7.6 Conformal Mapping of a Parabolic Region

Given a parabola x = -1 + $\frac{y^2}{4}$,



f(x,y) = $i\cosh(\frac{\pi}{2}\sqrt{x+iy}) = u+iv$ is the map that takes this parabola to the half-plane. This map can be rewritten as $f(x,y) = \frac{1}{2}(e^{i\frac{\pi}{2}+\sqrt{x+iy}} + e^{i\frac{\pi}{2}-\sqrt{x+iy}})$. After much simplification, we came up with the following formulas:

$$u(x,y) = \frac{1}{2}e^{\frac{\pi}{2}\left(\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}+1\right)}\cos\left(\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}+1\right)$$

$$+\frac{1}{2}e^{\frac{-\pi}{2}\left(1-\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}\right)}\cos\left(\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}+1\right)$$

$$v(x,y) = \frac{1}{2}e^{\frac{\pi}{2}\left(\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}+1\right)}\sin\left(\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}+1\right)$$

$$+\frac{1}{2}e^{\frac{-\pi}{2}\left(1-\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}\right)}\sin\left(\sqrt{\frac{x+\sqrt{x^2+y^2}}{2}}+1\right)$$

From our parabola, we have the boundaries (x,2$\sqrt{x+1}$) and (x,-2$\sqrt{x+1}$). If we plug y = $\pm 2\sqrt{x+1}$, we find out where our boundary is 'sent' to:

$$h_0(x) : f(x, 2\sqrt{x+1}) = (-\sinh(\frac{\pi}{2}\sqrt{x+1}), 0)$$

$$h_1(x) : f(x, -2\sqrt{x+1}) = (\sinh(\frac{\pi}{2}\sqrt{x+1}), 0)$$

We can describe our boundary in the half-plane in terms of the boundary of the parabola under the conformal mapping:

$$u_0(\tau) \quad = \quad \begin{cases} h_0\left((\frac{2}{\pi}\sinh^{-1}(-\tau))^2 - 1 \right) \text{if} \tau < 0 \\ h_1\left((\frac{2}{\pi}\sinh -1(\tau))^2 - 1 \right) \text{if} \tau > 0 \end{cases}$$

The Cauchy distribution states

$$u(x_0, y_0) = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{y_0}{(x_0 - \tau)^2 + {y_0}^2} u_0(t) d\tau$$

Thus our solution in the parabolic region is

$$h(x_0, y_0) = h(s_0, t_0) \text{ with } s_0(x_0, y_0) \text{ and } t_0(x_0, y_0)$$
$$f(x_0, y_0) = \imath\cosh(\tfrac{\pi}{2}\sqrt{x_0 + iy_0}) = s_0 + t_0\imath$$

$$\begin{aligned} h(s_0, t_0) \quad &= \quad \int_{-1}^{\infty} \frac{\cosh(\frac{\pi}{2}\sqrt{\tau+1})t_0}{4\sqrt{\tau+1}\left(\left(s_0 + \sinh(\frac{\pi}{2}\sqrt{\tau+1})\right)^2 + {t_0}^2\right)} h_0(\tau) d\tau \\ &+ \quad \int_{-1}^{\infty} \frac{\cosh(\frac{\pi}{2}\sqrt{\tau+1})t_0}{4\sqrt{\tau+1}\left(\left(s_0 - \sinh(\frac{\pi}{2}\sqrt{\tau+1})\right)^2 + {t_0}^2\right)} h_1(\tau) d\tau \end{aligned}$$

So what have we done so far? We have efficiently simulated Brownian motion and made our point in question converge quickly to the boundary by using the walk on spheres method. What if the boundary is unknown but computable? These computations can be very expensive. We are looking for a way to maintain accuracy whilst limiting the number of boundary points that are computed, and find the best way to do this. The use of Gaussian Quadratures and polynomial spaces is what leads to the solution, assuming that we have a nice, smooth boundary function. So, let $p(x)$ be the linear space of polynomials up to degree n with inner product defined as Similarly, for p_2 we can write a similar set of equations. which reduce to

8 Real World Application

Using the random walks method as we described before to solve Laplace's equation in a region $\mathcal{R}$, we must calculate the boundary value for each iteration of the random walk method. As a result, we must calculate values on the boundary a significant number of times. Although this presents no problem if the boundary condition is known *a priori*, the shear number of calculations this process requires is impractical if an expression for the boundary condition is not known. Rather, it is reasonable to assume that we do not have an expression for the boundary values, but we ascertain boundary values through observations methods. For instance, we could survey population densities or temperatures at a particular point on the boundary of the region. Determining this boundary value might be expensive. In this case, it is natural to try and restrict the number of times one must determine the boundary value.

One such method to limit the number of times one must determine the boundary value is to choose n distinct points along the boundary. At each of these points, the researcher observes and records the boundary values. Then, using computer simulations, the researcher utilizes the random walks method to obtain a point on the boundary, say p_0. Typically, the researcher would calculate the boundary value at p_0. In this method, however, the researcher finds the closest of the n pre-selected points and instead uses the boundary value at this point as an approximation to the boundary value at p_0. Therefore, the researcher is able to limit the number of times he must compute the boundary. How do we know, however, that this method is able to retain a sufficient amount of accuracy?

8.1 Obtaining Efficient Estimators - Gaussian quadrature

Given that we want to limit the number of times we find the boundary values to n times, we want to find a way to approximate our solution. In essence, we want to find D_i and x_i for i=1,2,...,n such that

$$\int_{-\infty}^{\infty} D(x)u_0(x)dx \;\approx\; \sum_{i=1}^{n} D_i u(x_i) \tag{16}$$

where D(x) is the probability density function for the point of first encounter and $u_0(x)$ is the boundary-value condition.

Given any collection of n distinct points along the boundary, we can select our weights D_i such that the approximation is exact in (16) so long as u_0(x) is a polynomial with degree < n. In fact, it can be shown that if we pick our weights to satisfy the system of n equations given by:

$$\int_{-\infty}^{\infty} x^k D(x)dx \;=\; \sum_{i=1}^{n} {x_i}^k D_i \tag{17}$$

for k = 0, 1, 2, . . . , n -1

To show that (17) is exact for deg($u_0(x)$)<n let $u_0(x)$ be a polynomial of degree n-1. Hence,

$$u_0(x) = \sum_{j=0}^{n-1} \alpha_j x^j$$

Therefore,

$$\begin{aligned}
\int_{-\infty}^{\infty} D(x)u_0(x)dx &= \int_{-\infty}^{\infty} D(x) \sum_{j=0}^{n-1} \alpha_j x^j dx \\
&= \sum_{j=0}^{n-1} \alpha_j \int_{-\infty}^{\infty} x^j D(x)dx \\
&= \sum_{j=0}^{n-1} \alpha_j \sum_{i=1}^{n} {x_i}^j D_i \\
&= \sum_{i=1}^{n} \sum_{j=0}^{n-1} \alpha_j {x_i}^j D_i \\
&= \sum_{i=1}^{n} u_0(x_i) D_i
\end{aligned}$$

We can, however, obtain more accuracy through wise choices for our x_i terms.

8.2 Selecting x_is

One way to pick our x_is in an efficient manner is to pick our n x_is as the n real roots of the nth degree polynomial, say $p_n(x)$, orthogonal to all polynomials of lesser degree with respect to the distribution D(x). We define the inner product on pairs of polynomials p and q as

$$(p, q) = \int_{-\infty}^{\infty} p(x)q(x)D(x)dx$$

Since two polynomials are orthogonal if and only if their inner product is zero, we want to find p_n(x) such that (p_n(x),f(x)) = 0 for all f(x) such that deg(f(x)) < n. One way to find such p_n(x) is to solve the following system of equations:

$$\begin{aligned}
(p_n, 1) &= \int_{-\infty}^{\infty} p_n(x)D(x)dx = 0 \\
(p_n, x) &= \int_{-\infty}^{\infty} p_n(x)xD(x)dx = 0 \\
(p_n, x^2) &= \int_{-\infty}^{\infty} p_n(x)x^2D(x)dx = 0 \\
&\vdots \\
(p_n, x^{n-1}) &= \int_{-\infty}^{\infty} p_n(x)x^{n-1}D(x)dx = 0
\end{aligned}$$

where $p_n(x) = \sum_{i=0}^{n} \alpha_i x^i$.
Notice, however, that this system consists of n+1 unknowns but only n equations. As a result, in order to determine p_n(x) uniquely, we will choose p_n(x) to be a monic polynomial.

Now, we shall show that choosing our x_is and D_is in such a manner will result in an exact answer for (XXX) given u_0(x) is a polynomial of degree 2n-1. Let u_0(x) be a polynomial of degree up to 2n - 1.

By the division algorithm, we know there exists polynomials $\alpha(x)$ and r(x) such that $u_0(x) = \alpha(x)p_n(x) + r(x)$ with deg(α(x)) = deg(u_0(x)) - deg(p_n(x)) $\leq$ 2n -1 - n = n - 1 and deg(r(x))<deg(p_n(x))=n.

Hence,

$$\int_{-\infty}^{\infty} u_0(x)D(x)dx = \int_{-\infty}^{\infty} \alpha(x)p_n(x)D(x)dx + \int_{-\infty}^{\infty} r(x)D(x)dx$$

Since we constructed $p_n(x)$ to be orthogonal to all polynomials of lesser degree with respect to the weight D(x) and since α(x) has degree less than n, this reduces to

$$= \int_{-\infty}^{\infty} r(x)D(x)dx$$

Recall that we selected our weights, D_i in such a way that this integral would match the summation provided our degree was n-1 or less. Furthermore, we know that the degree of r(x) must be less than n. Hence, we have

$$= \sum_{i=1}^{n} r(x_i)D_i$$

Since we selected x_is to be the roots of p_n(x) we have that $\sum_{i=1}^{n} \alpha(x)p_n(x_i)D_i = 0$. Hence,

$$\begin{aligned}
&= \sum_{i=1}^{n} \alpha(x) p_n(x_i) D_i + \sum_{i=1}^{n} r(x_i) D_i \\
&= \sum_{i=1}^{n} u_o(x_i) D_i \text{ and hence,} \\
\int_{-\infty}^{\infty} u_0(x) D(x) dx &= \sum_{i=1}^{n} u_o(x_i) D_i
\end{aligned}$$

Thus, we know that we can select our x_is and our D_is in such a way to ensure that our new method is exact provided that $u_0(x)$ is a $2n-1^{th}$ or less degree polynomial. Furthermore, if u_0(x) is sufficiently smooth (i.e. the 2n-1 derivatives exist) then we know our new method will be accurate provided u_0(x) is well approximated by a polynomial.

Since any distribution can be transformed to an uniform distribution on an interval, we can simplify the calculations of the abscissae x_i and weights w_i by implemeting the Legendre integration (a version of the Gaussian quadrature for a uniform weight function). The points and weights are available in Table 25.4 in [4] for high order of Legendre polynomials. They can be translated to the border of the upper half-plane by applying the tangent function (the inverse of the Cauchy cdf).

9 Conclusion

References

[1] Shizuo Kakutani. Two-dimensional Brownian motion and harmonic functions. *Proc. Imp. Acad. Tokyo*, 20:706–714, 1944.

[2] M. E. Muller. Some continuous Monte Carlo methods for the Dirichlet problem. *Ann. Math. Statist.*, 27:569589, 1956.

[3] Albert Einstein, Investigation on the Theory of the Brownian Movement, New York: Dover 1956.

[4] Milton Abramowitz, Irene Stegun, eds. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, New York: Dover Publications, 1972.

[5] Murray Spiegel, Seymour Lipschutz, John Schiller, Dennis Spellman. *Schaum's Outline of Complex Variables, 2ed (Schaum's Outline Series).* New York: McGraw-Hill, 2009

[6] David Poole. *Linear Algebra, A Modern Introduction, 2ed.* Thompson Brooks/Cole 2006

[7] Pierre Picco, Jaime San Martin. From Classical to Modern Probability: Cimpa Summer School 2001. Birkhäuser Bäsel 2004.