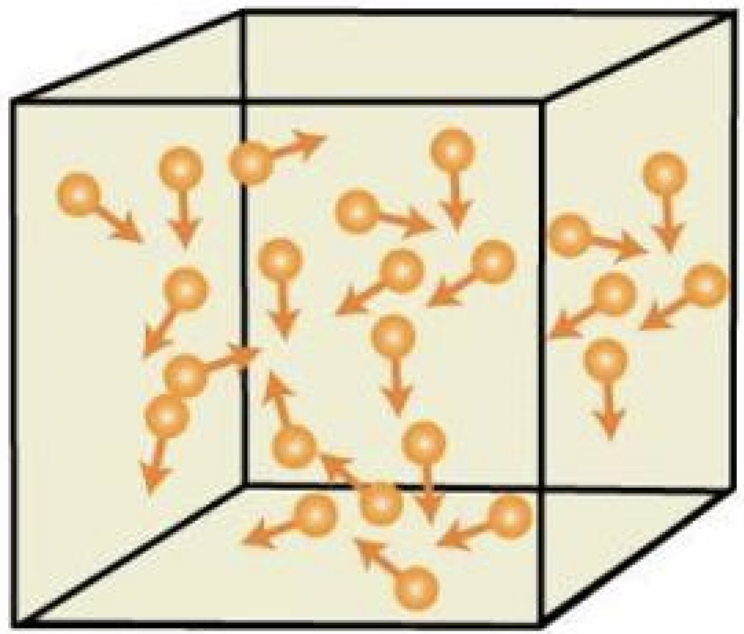
Random Solutions to Deterministic Problems: Brownian Motion and Geodesics

University of California, Riverside

June 2024

Background

- 1827: Botanist Robert Brown observed grains of pollen suspended in water dancing under a microscope. This became known as **Brownian motion.**
- Properties of Brownian motion appear everywhere from the movement of particles in a medium to the modeling stock prices.



Background

On a seemingly unrelated note, the shortest path between two points on Earth is not a straight line on a map. This is due to the curvature of the Earth, which is locally flat but globally curved.

This shortest path is called a **geodesic**, and its length is the **geodesic distance**. Computing geodesic distances is challenging due to their global nature.

Background

In the 1960s, Varadhan demonstrated that geodesic distance can be derived from probabilities associated with Brownian motion. Through the use of his formula:

$$\lim_{t \to 0} -4t \log K(t, x, y) = \|x - y\|^2$$

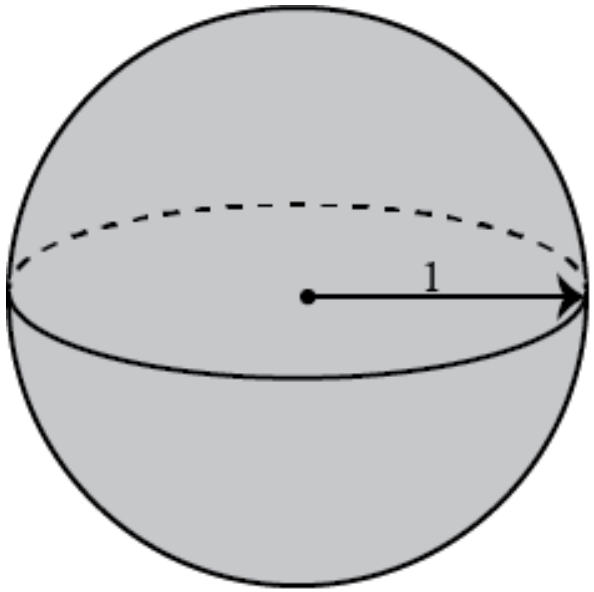
In this project, we aim to apply Varadhan's findings to explore an efficient method for approximating geodesics on manifolds using Varadhan's formula.

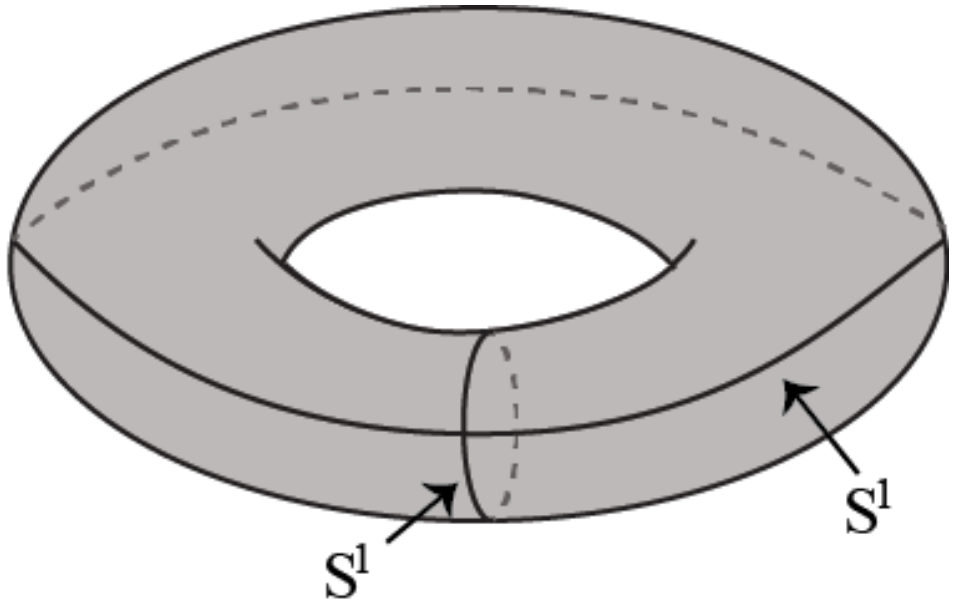
Manifolds

Manifolds are pivotal in our project, providing the geometric foundation within which we define and explore geodesics.

- A **manifold** is a topological space that is "locally Euclidean" but can have a more complex global structure.
- The key trait of manifolds that is useful to us is that they are **locally homeomorphic** to an open subset of $\mathbb{R}^n$.
- 1-D manifolds include **lines** and **circles**
- 2-D manifolds (surfaces) include the **plane, sphere, torus**, etc.

Manifolds (Cont'd)

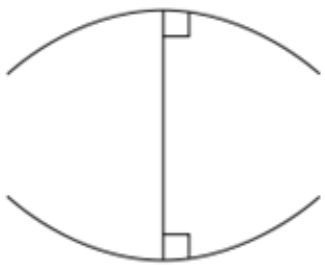




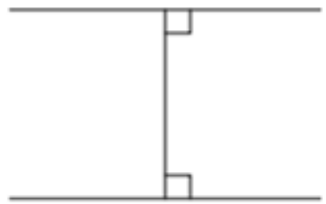
- The local properties of manifolds allowed us to simulate random walks.
- Subsequently, we utilized these simulations to estimate the heat kernel, a process dependent on the manifold's global properties.

Geodesics

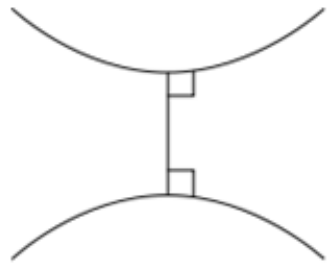
A **geodesic** is a locally length-minimizing curve and the shortest path between two points on a manifold. Geodesics generalize straight lines in Euclidean space to curved surfaces.



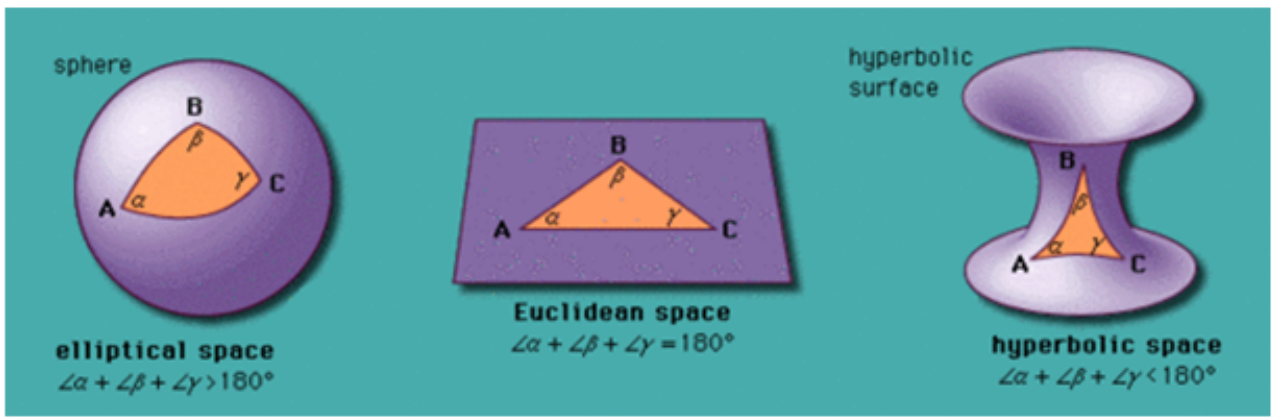
Elliptic



Euclidean



Hyperbolic



Probability, Random Variables, Mean, Variance

Random Variable (Rolling Dice)

- Result that depends on random events e.g. outcome of rolling dice.

$$P(X_i = 6) = \frac{1}{6} \quad \& \quad P(X_i \leq 3) = \frac{1}{2}$$

Expectation (Mean)

- The expected value when measuring a random variable

$$E[X] = \frac{\sum \text{outcomes}}{\#\text{possibilities}} = \frac{1+2+3+4+5+6}{6} = 3.5$$

Variance (Standard Deviation)

- The amount we expect our value to deviate from the mean

$$\sigma^2 = \frac{\sum(\mathit{outcome} - \mathit{expectation})^2}{\#\mathit{possibilities}} = \frac{\sum_i^n (X_i - 3.5)^2}{6} \approx 2.92$$

Then σ gives us the standard deviation.

Probability Distributions

A function that gives us the likelihood of an event occurring. Imagine X is a particular event, Ω be the set of all possible outcomes, and $E \subseteq \Omega$

$$P(X \in E) \geq 0$$
$$P(X \in E) \leq 1$$
$$P(X \in \Omega) = 1$$

For a continuous event, such as the position at time t of a particle undergoing diffusion, given by a function $f(t, x)$ we can take the probability of X lying in a certain interval say $E = [a, b]$ at time t

$$P(X(t) \in E) = P(a \leq X(t) \leq b) = \int_a^b f(t, x)\, dx$$

Random Walks

A random walk is a summation of random variables that are **independent** and **identically distributed**, for example in 1D

$$X_i = \begin{cases} & 1 & P = \frac{1}{2} \\ - & 1 & P = \frac{1}{2} \end{cases}$$

Our random walk would be

$$S_n = \sum_n X_i$$

Giving us the trajectory of a path where at each point, we have equal chance of going up or going down.

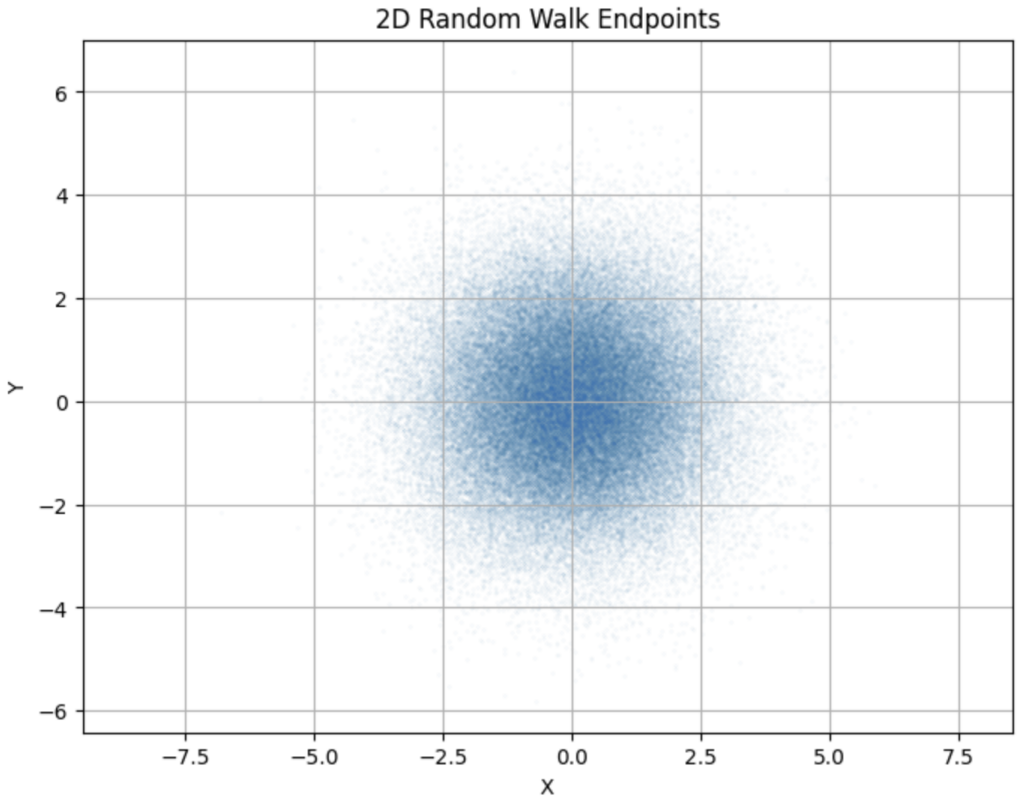
Diffusion

Diffusion describes the process of particles spreading out over time. This is given by:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u(x, t)$$

$u(x, t)$ represents the concentration of particles at position x and time t, ∇^2 is the Laplacian operator (where ∇ is the gradient), and α is the diffusion coefficient.

Diffusion (Cont'd)



The plot on the left shows the endpoints of a large number of random walks. This demonstrates how individual particle movements aggregate to form a smooth diffusion pattern over time.

Law of Large Numbers

Law of Large Numbers states that as the size of a sample increases, the sample mean converges in probability to the population mean.

- As we gather more samples, the distribution will look more and more like the distribution of the whole population.
- As we perform more and more random walks, they will begin to converge to diffusion.

Central Limit Theorem

Central Limit Theorem asserts that the distribution of the sample mean of a sufficiently large sample drawn from any population tends to a normal distribution that is Gaussian distribution, regardless of the shape of the original population distribution, under certain conditions.

- If we take many samples from random variables $X_1, X_2, ...$ such that X_i independent of X_{i+1} and X_i has the same distribution for all i, the distribution of the average of our samples will resemble a 'bell curve'.

Brownian Motion/Wiener Processes

A Wiener process, also known as Brownian motion, is a continuous-time stochastic process W_t defined for $t \geq 0$, possessing the following properties:

1. **Initial Value:** $W_0 = 0$
2. **Independent Increments:** The increments $W_{t+s} - W_t$ are independent of the past values and have the same distribution for all $t \geq 0$ and $s > 0$. (has no memory)
3. **Continuous Paths:** The sample paths of W_t are almost surely continuous functions of t, meaning that W_t is continuous with probability 1.
4. **Gaussian Increments:** The increments $W_{t+u} - W_t$ are normally distributed with mean 0 and variance u.

Invariance Principle/Donsker's Theorem

Let $S_n = \sum_1^n X_i$ where X_i is an independent, identically distributed random variable with mean 0 and variance 1. Then $S = (S_n)_{n \in \mathbb{N}}$ is a random walk.

Rescale the random walk by:

$$W^{(n)}(t) = \frac{S_{\lfloor nt \rfloor}}{\sqrt{n}} \quad t \in [0,1] \quad [1]$$

By the Central Limit Theorem, we know that $W^n(1)$ converges in distribution to the Gaussian random variable. Donsker's theorem proved that the whole function $W^{(n)}(t)$ converges to a Wiener process.

[1] $\lfloor nt \rfloor := \max\{m \in \mathbb{Z} \ : \ m \leq nt\}$

Invariance Principle/Donsker's Theorem (Cont'd)

What does it mean?

- **Brownian Motion:** Brownian motion can be approximated by a random walk, where the steps are independent and identically distributed random variables. By Donsker's Theorem, as the number of steps increases and the step size decreases appropriately, the random walk converges to Brownian motion.
- **Heat Equation:** Solutions to the heat equation can be approximated by considering the diffusion process of particles undergoing random walks. This approximation becomes exact in the limit due to Donsker's Theorem.
- **Random Walks:** Numerical approximations of Brownian motion using random walks rely on Donsker's Theorem. This allows us to use discrete-time random walks to simulate continuous-time Brownian motion.

Brownian Motion and the Heat Equation

The connection between Brownian motion and the heat equation lies in the probabilistic interpretation of the heat equation. If we consider a large number of particles undergoing Brownian motion, the probability density function $p(x, t)$ of finding a particle at position x at time t satisfies the heat equation:

$$\frac{\partial p}{\partial t} = \frac{\sigma^2}{2} \nabla^2 p$$

Here, $\sigma^2/2$ plays the role of the thermal diffusivity α. This relationship shows that the macroscopic behavior of heat diffusion can be understood in terms of the microscopic random motion of particles.

Heat Kernel

K is a transition probability. The heat kernel is the solution to the heat equation, which explains the diffusion of heat along a surface given by

$$K(t, x, y) = \frac{1}{(4\pi t)^{d/2}} \exp\left(-\frac{\|x - y\|^2}{4t}\right)$$

This formula shows how heat diffuses from point x to point y over time t.

- With fixed t, kernel values decrease as they move away from the source
- As t increases, the kernel values become more uniform through the surface

Varadhan's Formula

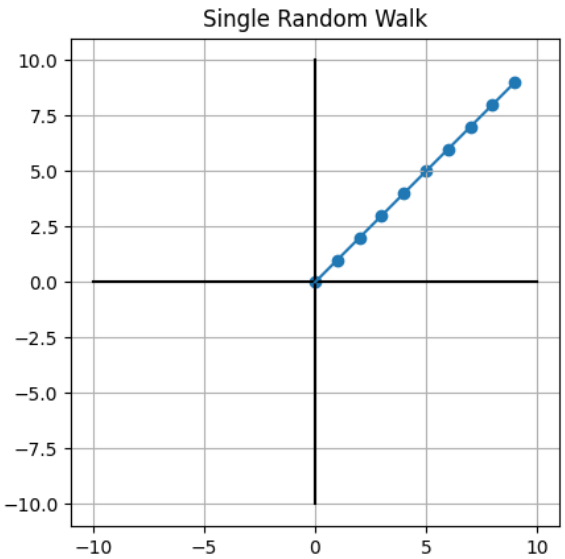
For the diffusion process Varadhan's formula is used to approximate the relationship within the particles. The formula is given by:

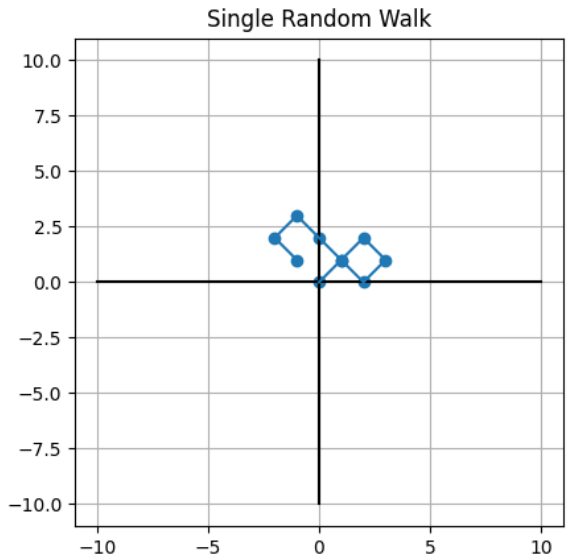
$$d(x,y)^2 = \lim_{t \to 0} 4t \log k(t,x,y)$$

where $d(x,y)$ represents the geodesic distance between points x and y in a Riemannian manifold, and $k(t,x,y)$ is the heat kernel, which represents the fundamental solution to the heat equation.

Intuition

Individual particles undergoing heat diffusion will follow Brownian motion. As t decreases, the likelihood of a particle being far from the heat source will be very low, and the only way it could have traveled far is if it took an efficient path to get there.

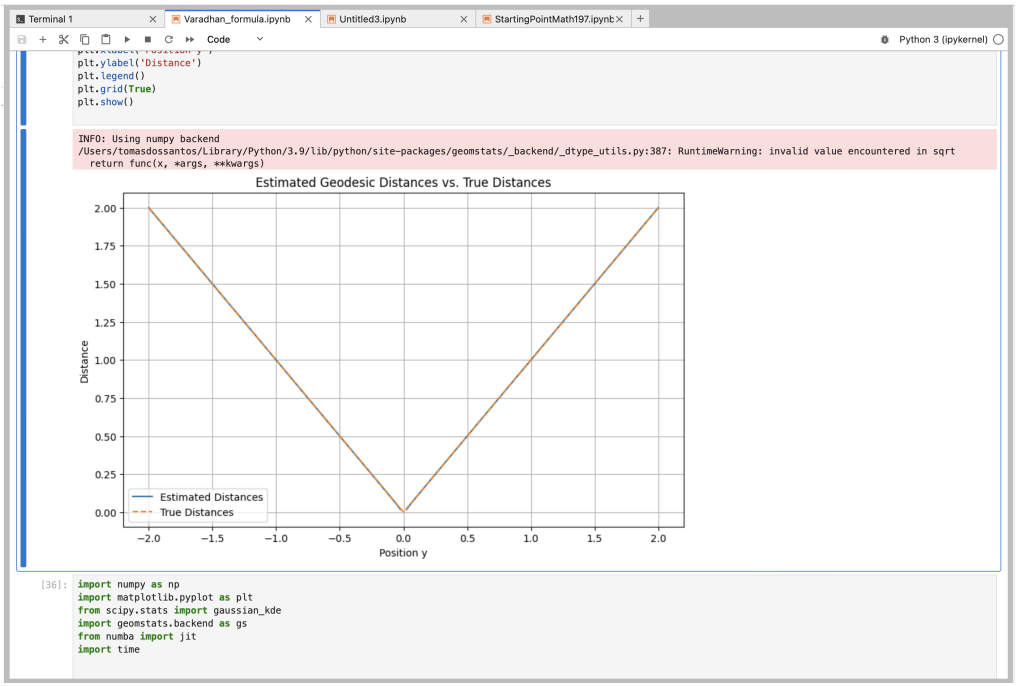




Above you can see two random walks with fixed number of steps (simulating fixed t), one takes a straight path, which would be akin to a geodesic on the manifold, and the other meanders and doesn't make it very far.

Coding it Up

In practice, our goal was to apply what we learned about manifolds, brownian motion, and Varadhan's formula to approximate geodesics in a real world scenario. We used python for our computational tasks maintaining our code in a Jupyter notebook.



Steps to Estimate Geodesics

Now that we have the background and definitions, we would like to apply what we have to find actual geodesics.

1. Recognize that calculating geodesics analytically is often impossible.
2. Use Varadhan's formula to relate the heat equation to geodesic distance.
3. Approximate the heat kernel on certain manifolds via particle paths (Brownian Motion).
4. Use scaled random walks to approximate Brownian motion (Donsker's Theorem).
5. Apply Monte Carlo methods to estimate the heat kernel.
6. Estimate geodesics from the approximated heat kernel.

Coding Random Walks in Euclidean Space

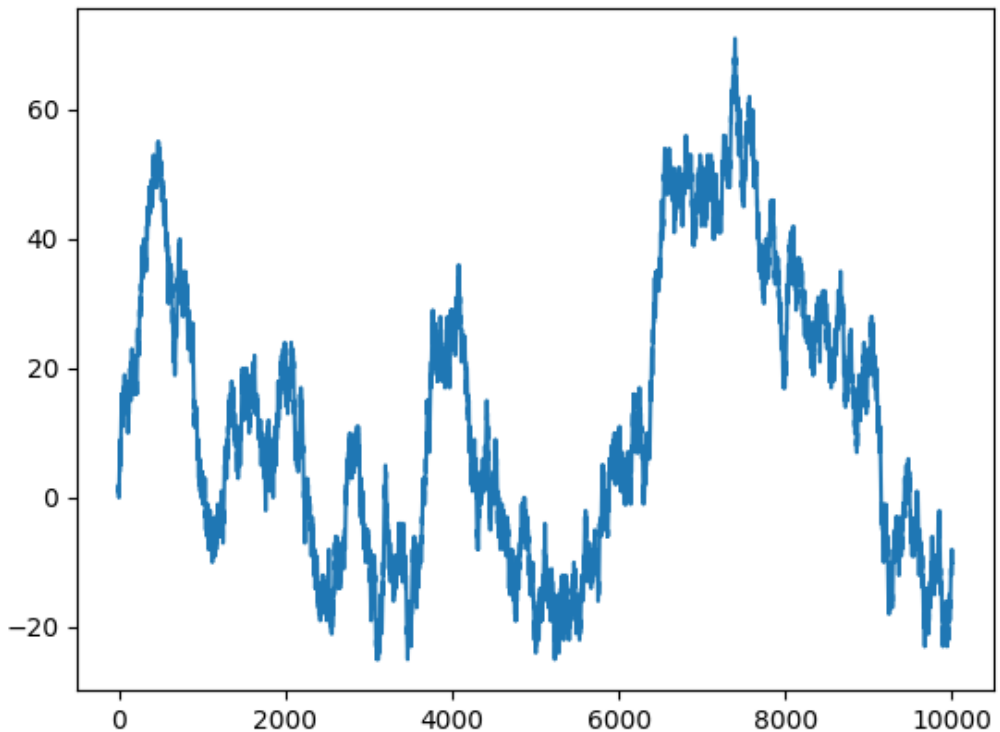
In our project, we implemented random walks to approximate geodesic distances. We used two different approaches depending on the space we were working in.

Algorithm 1: Euclidean Space Random Walks

1. Initialize the starting position x_0.
2. For each step:
   - 2.1 Randomly sample a direction from the basis vectors of the Euclidean space.
   - 2.2 Move to the new position by adding the step size times the sampled direction to the current position.
3. Repeat until the desired number of steps is reached.

Example Euclidean Random Walk

Here is an example of the simplest random walk with 10,000 steps and a step size of 1. This is done simply by sampling from going up or going down with equal probability.



Proper Scaling of Random Walks

To approximate Brownian motion we need scaled steps given by,

1. For Brownian motion, the expected mean squared deviation over time t is:
$$(\Delta x)^2 = 2D\Delta t$$
2. For a random walk with N steps over time t, the time per step is:
$$\Delta t = \frac{t}{N}$$
3. To match the mean squared deviation per step:
$$\Delta x = \sqrt{\frac{2Dt}{N}}$$

We leave $D = 1$ for simplicity leaving us with $\Delta x = \sqrt{\frac{2t}{N}}$.

Monte Carlo Simulations

Monte Carlo simulations are a technique used to evaluate integrals numerically. We use this since we cannot find an explicit value for the heat kernel in certain cases

1. **Generate random samples:** Create a large number of random points within the domain of integration (random walks)
2. **Evaluate the integrand:** Compute the value of the integrand at each random point (# of walks in interval)
3. **Compute the average:** Calculate the average value of the integrand across all sampled points (divde by length of interval)

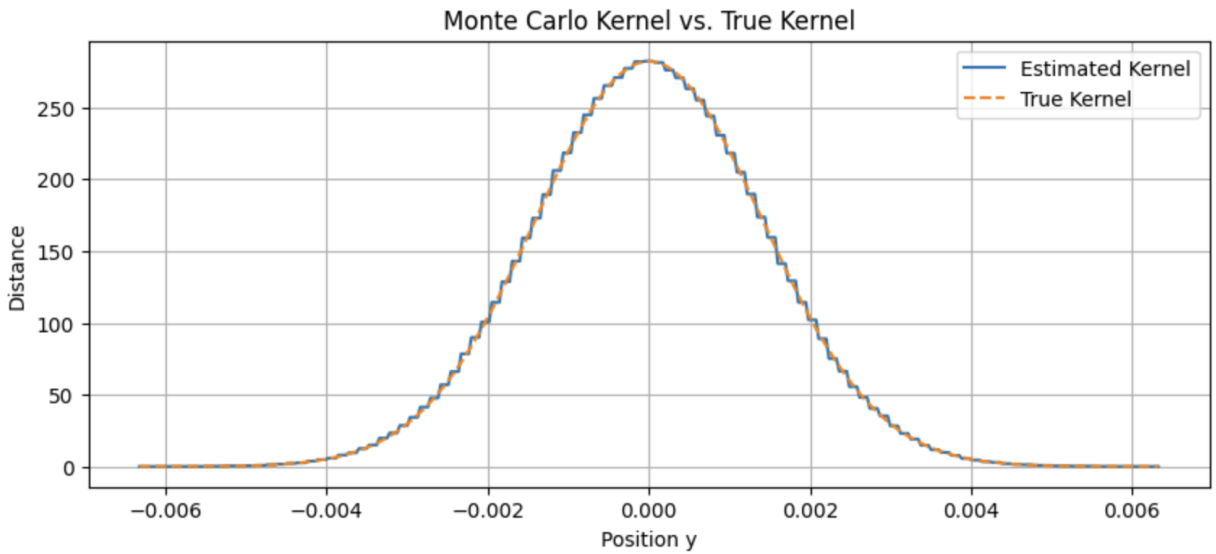
This approach allows us to estimate integrals, using the law of large numbers to converge to the correct value as the number of samples increases.

Monte Carlo Methods

To find the heat kernel numerically, notice:

$$K(t,x) \approx \frac{1}{2\varepsilon}\int_{x-\varepsilon}^{x+\varepsilon} K(t,y)\,dy = \frac{1}{2\varepsilon}P(W(t)\in B_\varepsilon(x))$$
$$\approx \frac{1}{2\varepsilon}P(\text{rw}\in(x-\varepsilon,x+\varepsilon))$$
$$\approx \frac{\text{num\_walks in } (x-\varepsilon,x+\varepsilon)}{\text{num\_walks}\cdot 2\varepsilon}$$

Here, $K(t,x)$ is the heat kernel, ε is a small positive value.



Applying to Varadhan's Formula

Now for approximating our values we must put the random walks, Monte Carlo simulations, and Vardhan's Formula together.

Approximating Distances

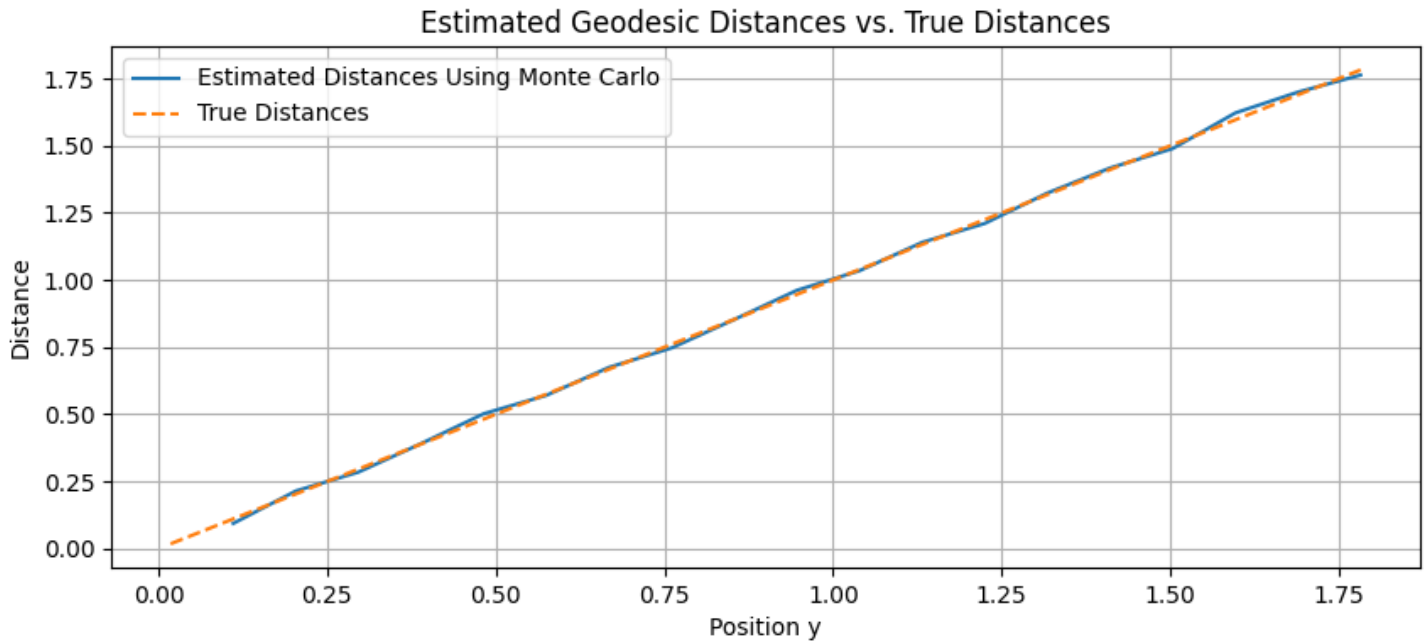
1. Choose a test point.
2. Simulate large number of random walks on our surface.
3. Apply Monte Carlo to estimate the heat kernel at test point.
4. Use Varadhan's formula to extract the distance.

Application in Euclidean Space Code

```python
def simulate_random_walks(start, num_walks, num_steps, step_size):
    steps = np.random.choice([-step_size, step_size], size=(num_walks, num_steps))
    walks = np.sum(steps, axis=1)
    endpoints = walks + start
    return endpoints


def monte_carlo_heat_kernel(endpoints, start, end, epsilon, num_walks):
    numerator = np.sum(np.abs(endpoints - end) <= epsilon)
    denominator = num_walks * epsilon * 2
    return numerator / denominator


def analytical_heat_kernel(start, end, t):
    return 1 / np.sqrt(4 * np.pi * t) * np.exp(-((end - start) ** 2) / (4 * t))


def varadhan_formula(kernel_estimate, t):
    if kernel_estimate > 0:
        estimated_distance = np.sqrt(np.abs(-4 * t * np.log(kernel_estimate)))
    else:
        estimated_distance = np.inf
    return estimated_distance
```

Application in Euclidean Space Results

We first applied our program to Euclidean space because it was much faster and easier to compute, which gave us good results.



Remarks and Limitations on Euclidean Space

Performing our estimates on Euclidean space was a good place to make observations on potential problems when applying to manifolds

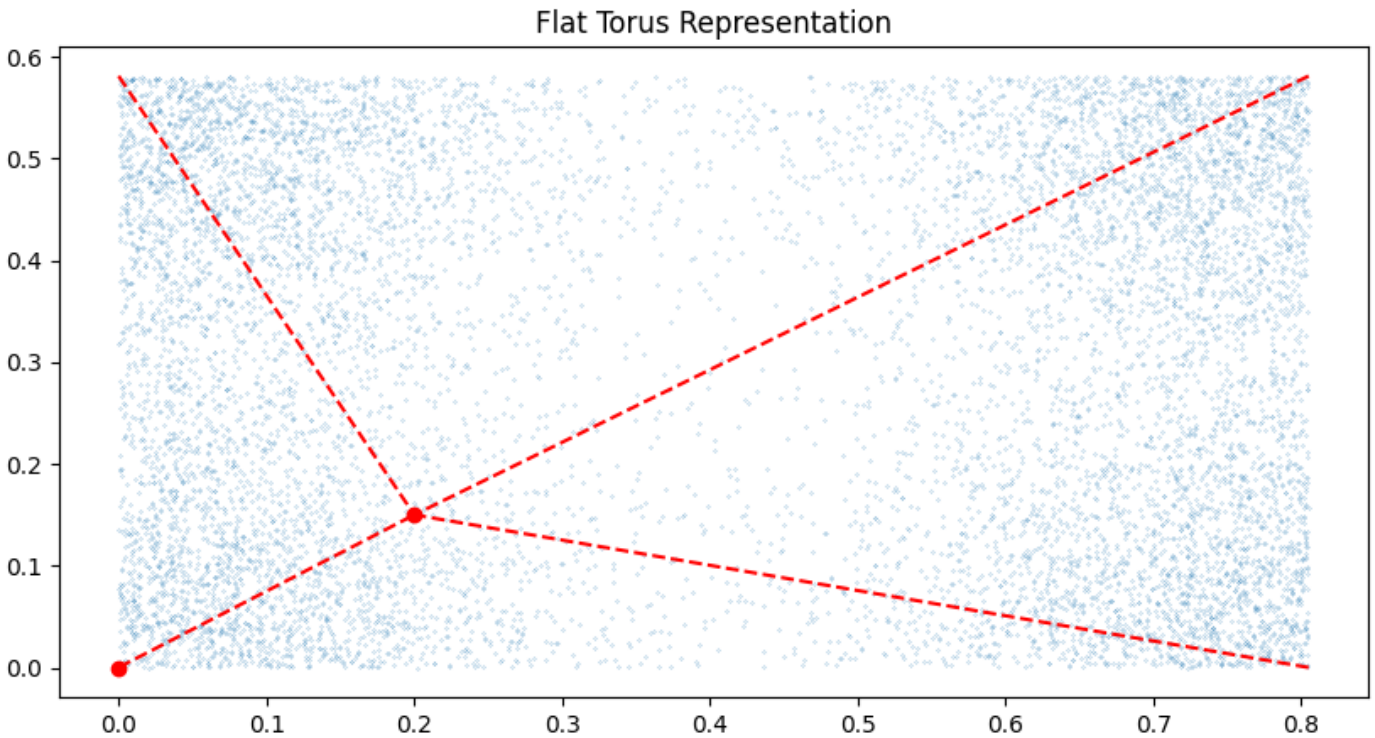
- Our approximation was highly dependent on the accuracy of the heat kernel
- We were limited by the number of steps, since the furthest our random walk could reach would be #steps * # walks (technically there should be no 0 values in the heat kernel, but since we only have a finite number of steps and walks, after a certain point our approximation would just be 0)

Application to 'Flat Torus'

To simplify our application, we can apply our code for Euclidean space to find geodesics of a flat torus based on the equivalence relation:

$$(x, y) \sim (x + 1, y) \sim (x, y + 1),$$

This will not be completely accurate as it doesn't account for curvature, but is a good starting point.



Code for Torus

```
def monte_carlo_heat_kernel(walks, end, epsilon, num_walks):
    numerator = np.sum((walks[:,0] >= end[0] - epsilon) &
                       (walks[:,0] <= end[0] + epsilon) &
                       (walks[:,1] >= end[1] - epsilon) &
                       (walks[:,1] <= end[1] + epsilon))
    denominator = num_walks * (2 * epsilon)**2
    return numerator / denominator


def heat_kernel(x, y, t, num_terms):

    kernel = 0.0
    for i in range(-num_terms, num_terms + 1):
        for j in range(-num_terms, num_terms + 1):
            x = x + (i * torus_length_x)
            y = y + (j * torus_length_y)
            kernel += ((1 / (np.pi * 4 * t)) * np.exp(-(x**2 + y**2) / 4 * t))

    return kernel
```

Retraction-based Random Walks

For larger manifolds, we employ a more sophisticated method using retraction-based random walks as outlined by Schwarz.

Algorithm 2: Retraction-based Random Walks

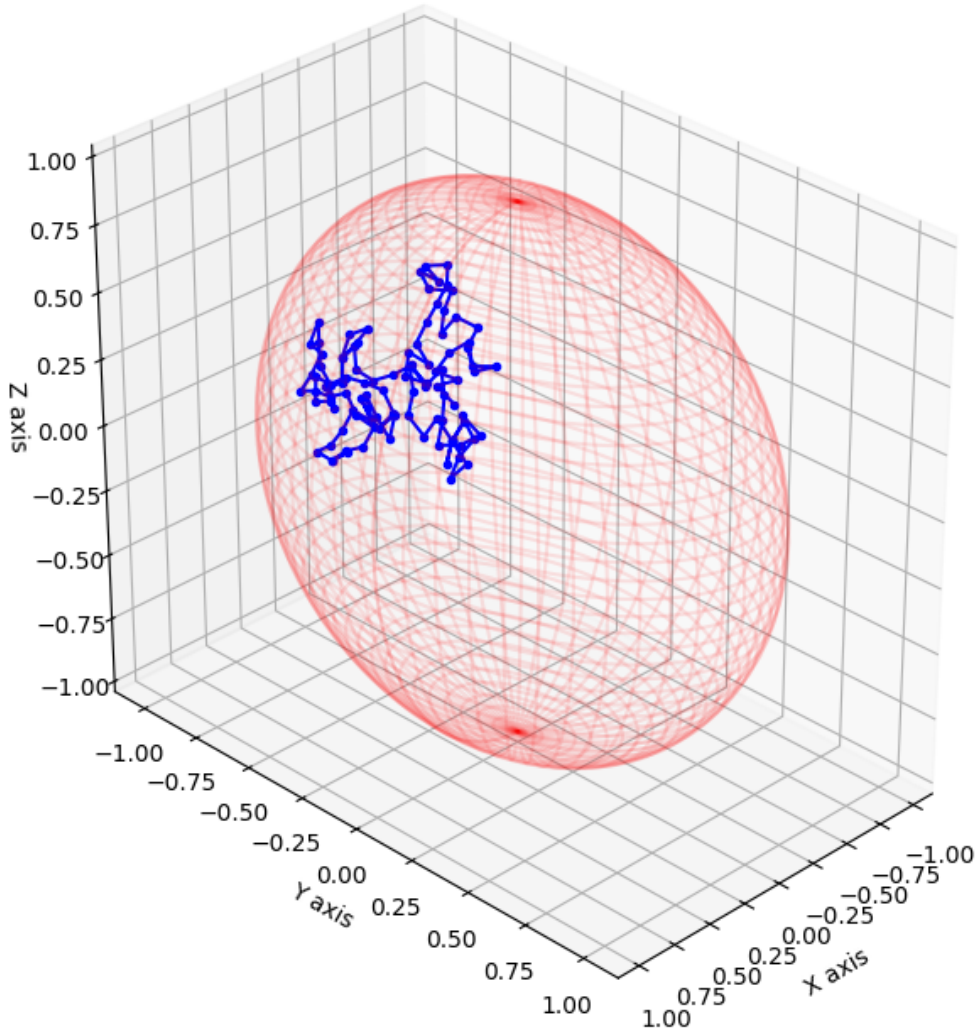
1. Initialize the starting position x_0.
2. For each iteration:
   - 2.1 Sample $\bar{v}$ uniformly from the unit tangent sphere at the current position x_{i-1}^{ϵ}.
   - 2.2 Scale $\bar{v}$ to $v = \sqrt{m} \cdot \bar{v}$.
   - 2.3 Update the position to $x_i^{\epsilon} = \mathrm{Ret}_{x_{i-1}^{\epsilon}}(\epsilon v)$ using the retraction map.
3. Repeat for N iterations.

This algorithm allows us to generalized what we worked on in this quarter to curved surfaces.

Example Implementation

```
dimension = 3

sphere = Hypersphere(dim=dimension - 1)


def random_tangent_vector(x, epsilon, n):

    v = np.random.randn(n)
    v -= v.dot(x) * x
    v /= np.linalg.norm(v)
    return v * epsilon

def retraction(x, v):
    return (x + v) / np.linalg.norm(x + v)

def retraction_based_random_walk(N, epsilon, n, x0):
    x_current = x0
    walk = [x_current]

    for i in range(N):
        v_tilde = random_tangent_vector(x_current, epsilon, n)
        v = np.sqrt(n) * v_tilde
        x_current = retraction(x_current, v)
        walk.append(x_current)

    return walk


N = 100
epsilon = 0.05
n = dimension
x0 = sphere.random_uniform(n_samples = 1)


rw_2_sphere = retraction_based_random_walk(N, epsilon, n, x0)
```

Example Random Walk on Sphere



Questions

Q & A

Acknowledgements

We would like to thank Professor Costello and Professor Gonzalez as well as the math department for hosting the Math 197 Program. We would also like to thank our mentor Rahul for guiding us through this subject.