Monte Carlo Approximation of Pi in N-Dimensional Space

Introduction

In this report, we use the Monte Carlo method to approximate the value of π by generating random points in a unit n-dimensional cube and checking how many fall inside an n -dimensional sphere of radius 1, centered at the origin.

The basic idea of the Monte Carlo method is to use randomness to solve problems that may be deterministic in principle. By randomly generating points, we can estimate the proportion of points that fall inside the n -dimensional sphere, and from this, we can derive an approximation of π.

Method

Mathematical Foundation

The volume of an n-dimensional unit sphere is given by the formula:

$$V_{\text{sphere}}(n) = \frac{\pi^{n/2}}{\Gamma(n/2+1)}$$

Where Γ is the Gamma function. The volume of an n-dimensional cube is 2^n.

The ratio of the volume of the sphere to the volume of the cube is:

$$\text{Ratio} = \frac{V_{\text{sphere}}(n)}{V_{\text{cube}}(n)} = \frac{\pi^{n/2}}{2^n\Gamma(n/2+1)}$$

Using Monte Carlo sampling, we estimate the ratio by generating random points in the cube and counting how many fall inside the sphere. From this ratio, we can derive an approximation of π by rearranging the equation to solve for π:

$$\pi \approx \left(\frac{\text{Ratio} \cdot 2^n \cdot \Gamma(n/2+1)}{1} \right)^{2/n}$$

R Code

The following R code implements the Monte Carlo method for estimating ( ):

```r
# Function to check if a point is inside the n-dimensional sphere
is_point_inside <- function(n, point) {
  r_centre <- numeric(length = length(point))
  # Initialize the center of the sphere
  dist <- norm(point - r_centre, "2")
  # Calculate the Euclidean distance from the origin
  if (dist <= 1) {
    return(1)  # Point is inside the sphere
  } else {
    return(0)  # Point is outside the sphere
  }
}

# Function to generate a random point in n-dimensional space within the cube [-1, 1]
generate_point_ref <- function(n) {
  points <- numeric(n)
  for (i in 1:n) {
    points[i] <- runif(1, min = -1, max = 1)
    # Generate random coordinates in the cube
  }
  return(points)
}

# Function to estimate Pi using the Monte Carlo method
value_pi_individual <- function(n) {
  obs <- numeric(10000)  # Number of samples
  for (i in 1:10000) {
    point <- generate_point_ref(n)
    obs[i] <- is_point_inside(n, point)
    # Check if the point is inside the sphere
  }
  sum_obs <- sum(obs)
  ratio <- sum_obs / 10000  # Ratio of points inside the sphere
  value_pi <- ((ratio * (2^n) * gamma((n / 2) + 1)))^(2 / n)
```

```r
  # Estimate Pi using the ratio
  return(value_pi)
}

# Estimate Pi for 5-dimensional space
dimen <- 5
samples_by_diff <- numeric(100)  # Collect multiple estimates to improve accuracy
for (i in 1:100) {
  samples_by_diff[i] <- value_pi_individual(dimen)
}
most_apprx_pi <- mean(samples_by_diff)  # Take the mean of the estimates

# Calculate the error in the approximation
error <- (abs(most_apprx_pi - pi) / pi) * 100
most_apprx_pi  # Display the estimated value of Pi
```

```
[1] 3.138206
```

```r
error  # Display the error percentage
```

```
[1] 0.1078122
```