Mathematical Examples for Monte Carlo and Quasi-Random Sampling

Introduction

This document provides mathematical examples of integration using Monte Carlo sampling and quasi-random sequences. Code snippets in Python are included to demonstrate these techniques, leveraging the matplotlib library for visualization.

1. Monte Carlo Integration

Monte Carlo integration estimates the value of an integral by sampling points randomly and averaging the function values.

Example: Integrating a 1D Function We aim to approximate the integral:

$$I = \int_0^1 x^2 dx.$$

Using Monte Carlo sampling:

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i),$$

where x_i are uniformly sampled points in $[0, 1]$.

Python Code:

Listing 1: Monte Carlo Integration for x^2

```
import numpy as np
import matplotlib.pyplot as plt

# Define the function to integrate
def f(x):
    return x ** 2

# Number of samples
N = 1000

# Monte Carlo sampling
samples = np.random.uniform(0, 1, N)
integral = np.mean(f(samples))

# True value of the integral
true_value = 1 / 3
```

```
print(f"Monte Carlo Estimate: {integral}")
print(f"True Value: {true_value}")

# Visualization
x = np.linspace(0, 1, 100)
plt.plot(x, f(x), label='x^2')
plt.scatter(samples, f(samples), color='red', s=1, label='Samples')
plt.legend()
plt.title('Monte Carlo Sampling')
plt.show()
```

2. Quasi-Random Integration Using Halton Sequence

Quasi-random sequences like Halton reduce variance in integration by distributing samples more evenly.

Example: Integrating the Same Function with Halton We generate a Halton sequence for $[0, 1]$ and compute the integral.

Python Code:

Listing 2: Integration with Halton Sequence

```
from scipy.stats.qmc import Halton

# Number of samples
N = 1000

# Generate Halton sequence
halton = Halton(d=1)  # 1D sequence
samples = halton.random(n=N).flatten()

# Compute the integral
integral = np.mean(f(samples))

print(f"Halton Estimate: {integral}")
print(f"True Value: {true_value}")

# Visualization
plt.plot(x, f(x), label='x^2')
plt.scatter(samples, f(samples), color='red', s=1, label='Halton Samples')
plt.legend()
plt.title('Halton Sampling')
plt.show()
```

3. Comparing Monte Carlo and Quasi-Random

Visualizing Sample Distribution Compare the distribution of Monte Carlo and Halton samples to see the difference in coverage.

Python Code:

Listing 3: Comparing Sample Distributions

```
# Generate random and Halton samples
random_samples = np.random.uniform(0, 1, N)
halton_samples = Halton(d=1).random(n=N).flatten()

# Plot
```

```
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.hist(random_samples, bins=30, color='blue', alpha=0.7, label='Random')
plt.title('Monte Carlo Samples')
plt.legend()

plt.subplot(1, 2, 2)
plt.hist(halton_samples, bins=30, color='orange', alpha=0.7, label='Halton')
plt.title('Halton Samples')
plt.legend()

plt.tight_layout()
plt.show()
```

Conclusion

Monte Carlo integration is versatile and easy to implement, but quasi-random sequences like Halton improve convergence by reducing variance. These examples highlight their mathematical foundation and practical implementation for 1D integrals.