Estimação da área debaixo de um gráfico atrvés de Monte Carlo

Rennisson Davi D. Alves - 13687175

Maio de 2023

1 Introdução

Neste trabalho estamos interessados em estimar novamente a área debaixo de um gráfico, mas agora ao invés de usar geradores de números quase aleatórios, vamos utilizar geradores quasi-random.

Os métodos utilizados aqui serão os mesmos do EP2: Crude, Hit or Miss, Importance Sampling e Control Variate. Todos são variantes do método de Monte Carlo.

Como ferramentas, utilizamos a linguagem Python e suas bibliotecas científicas, Numpy e Scipy.

2 Quasi-random

Os geradores quasi-random têm um método diferente de sortear um número aleatório. Diferente de bibliotecas padrões, o gerador quasi tem uma "memória", e sorteia um número qualquer a partir do numero sorteado anteriormente. Fazendo dessa maneira, a discrepância entre os numeros sorteados é baixa. Assim, a variância de uma dada amostragem retirada a partir de geradores quasi é menor que a variância de uma amostragem retirada a partir de geradores quase aleatórios.

3 Implementação

A estrutura do código foi repetida neste EP, com algumas alterações para lidar com o comportamento de halton (gerador quasi random utilizado neste EP). As estruturas de classes foram mantidas, adicionando-se poucos métodos, são eles:

experimento(): função específica para fazer os experimentos necessários, de acordo com o algoritmo de estimação de cada um dos métodos. Retorna a estimativa do experimento.

media_estimativa(): Retorna a média das estimativas adquiridas em cada um dos experimentos realizados.

var_estimativa(): Retorna a variância das estimativas adquiridas em cada um dos experimentos realizados.

media_tempo(): Retorna a média de tempo de execução de cada um dos experimentos realizados.

precisão(): Retorna a precisão de acerto de cada um dos experimentos realizados. Aqui neste trabalho, como foi definido no EP2, estamos interessados em um intervalo de confiança de 95%, portanto queremos que o retorno dessa função seja maior que 0.95.

halton(): Retorna um array com os números sorteados através da função qmc.Halton(), que é o gerador quasi random da biblioteca Scipy.

halton_beta(): Retorna um array com os números sorteados através da função qmc.Halton(), depois de transformá-los em pontos de uma distribuição $beta(\alpha, \beta)$.

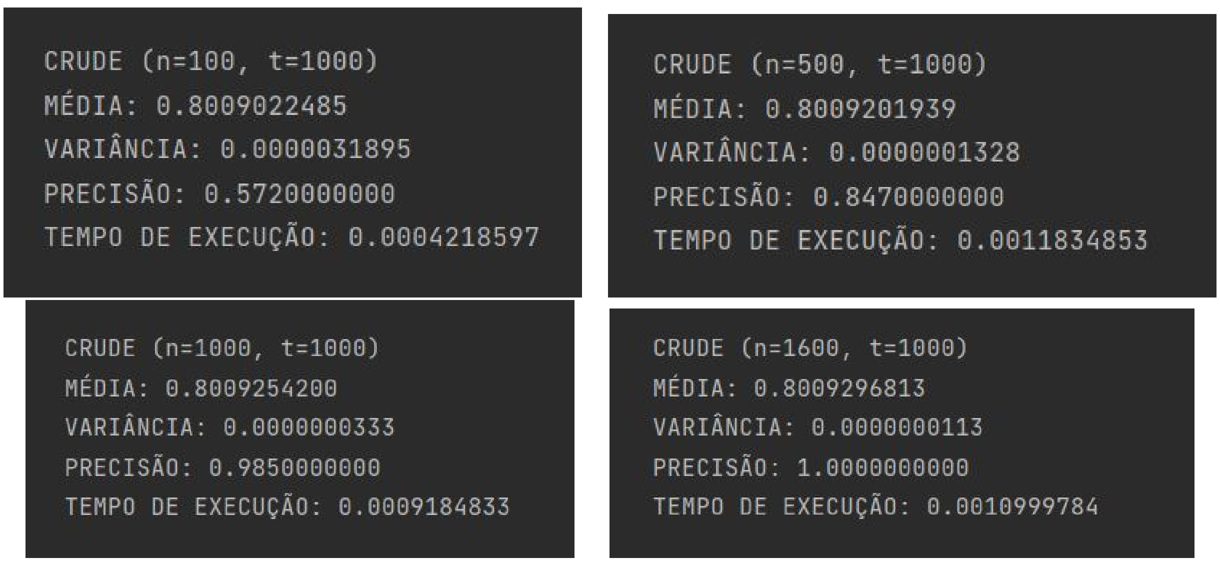
4 Experimentos

Para encontrarmos o tamanha da amostra n que melhor aproxima a área abaixo do gráfico de $f(x) = e^{-0.386617636x}\cos(0.47950399848x)$, optamos por testes empíricos. Observando a média, variância e precisão de cada experimento, seremos capazes de observar e perceber a partir de qual n nós nos aproximamos do valor real da integral de f(x).

Para cada um dos métodos, a função experimento() fará T experimentos para cada n selecionado arbitrariamente. Por definição, é esperado que esse n convirja rapidamente para o valor esperado, visto que a variância da amostra retirada por quasi-random é menor.

4.1 Crude

A seguir estão os experimentos feitos para o método Crude:


CRUDE (n=100, t=1000)
MÉDIA: 0.8009022485
VARIÂNCIA: 0.0000031895
PRECISÃO: 0.5720000000
TEMPO DE EXECUÇÃO: 0.0004218597

CRUDE (n=500, t=1000)
MÉDIA: 0.8009201939
VARIÂNCIA: 0.0000001328
PRECISÃO: 0.8470000000
TEMPO DE EXECUÇÃO: 0.0011834853

CRUDE (n=1000, t=1000)
MÉDIA: 0.8009254200
VARIÂNCIA: 0.0000000333
PRECISÃO: 0.9850000000
TEMPO DE EXECUÇÃO: 0.0009184833

CRUDE (n=1600, t=1000)
MÉDIA: 0.8009296813
VARIÂNCIA: 0.0000000113
PRECISÃO: 1.0000000000
TEMPO DE EXECUÇÃO: 0.0010999784

Figure 1: Testes do método Crude

Como podemos ver, para n=100, a precisão das estimativas é de 57.2%. Conforme vamos aumentando o valor de n, vamos nos aproximando da precisão pedida. Para n=500, nós acertamos em 84.7% o IC. Quando selecionamos n=1000, ultrapassamos os 95% de precisão que queríamos e chegamos em 98.5% de precisão. Se quisermos atingir 100%, precisamos de um n maior ou igual a 1600.

4.2 Hit or Miss

Como podemos ver, para n=1000, a precisão das estimativas fica próxima de 55%. Conforme vamos aumentando o valor de n, vamos nos aproximando da precisão pedida. Para n=10000, a precisão aumenta para 81%. Quando selecionamos n=30000, ultrapassamos os 95% de precisão que queríamos. E se desejarmos atingir 100%, é necessário um n maior que 50000.

```
HIT OR MISS (n=1000, t=1000)
ESTIMATIVA: 0.8008600000
VARIÂNCIA: 0.0000057264
TEMPO: 0.0014659433
PRECISÃO: 0.5410000000
```

```
HIT OR MISS (n=10000, t=1000)
ESTIMATIVA: 0.8009156000
VARIÂNCIA: 0.0000001856
TEMPO: 0.0074668434
PRECISÃO: 0.8100000000
```

```
HIT OR MISS (n=30000, t=1000)
ESTIMATIVA: 0.8009321000
VARIÂNCIA: 0.0000000351
TEMPO: 0.0208480420
PRECISÃO: 0.9870000000
```

```
HIT OR MISS (n=50000, t=1000)
ESTIMATIVA: 0.8009295200
VARIÂNCIA: 0.0000000177
TEMPO: 0.0350477064
PRECISÃO: 0.9990000000
```

Figure 2: Testes do método Hit or Miss

4.3 Importance Sampling

Para n=10, a precisão atingida foi só 50%. Quando aumentamos o valor de n para n=2000, a precisão aumenta para 83%. E para n=30000, ultrapassamos os 95% de precisão que queríamos. E para atingir 100%, é necessário um n maior que 8000.

```
IMPORTANCE SAMPLING (n=10, t=1000)
ESTIMATIVA: 0.8024488532
VARIÂNCIA: 0.0048334148
TEMPO: 0.0009843409
PRECISÃO: 0.5000000000
```

```
IMPORTANCE SAMPLING (n=2000, t=1000)
ESTIMATIVA: 0.8009391555
VARIÂNCIA: 0.0000001562
TEMPO: 0.0025940630
PRECISÃO: 0.8310000000
```

```
IMPORTANCE SAMPLING (n=5000, t=1000)
ESTIMATIVA: 0.8009375628
VARIÂNCIA: 0.0000000356
TEMPO: 0.0052816248
PRECISÃO: 0.9890000000
```

```
IMPORTANCE SAMPLING (n=8000, t=1000)
ESTIMATIVA: 0.8009341149
VARIÂNCIA: 0.0000000093
TEMPO: 0.0079443035
PRECISÃO: 1.0000000000
```

Figure 3: Testes do método Importance Sampling

4.4 Control Variate

Como esperado, conforme os resultados do EP2, o método Control Variate é o que retorna melhor resposta, isto é, menor tamanho de amostra populacional para convergir para o valor de real da integral de f(x). Com n=5, já obtemos precisão de 100%. E isso é incrível, atingir o resultado necessário com tão poucos pontos.

```
CONTROL VARIATE (n=5, t=1000)
ESTIMATIVA: 0.8009333596
VARIÂNCIA: 0.0000000315
TEMPO: 0.0004062359
PRECISÃO: 1.0000000000
```

```
CONTROL VARIATE (n=10, t=1000)
ESTIMATIVA: 0.8009309641
VARIÂNCIA: 0.0000000005
TEMPO: 0.0005312297
PRECISÃO: 1.0000000000
```

Figure 4: Testes do método Control Variate

5 Tempo de execução

Analisando o tempo de execução, a média de tempo gasto por experimento realizado foi menor que 0.5 segundos. O método que mais gastou tempo executando seus experimentos foi o Hit or Miss, que gastou quase 1 segundo para realizar cada experimento. E isso se deve ao fato de ele ser o método com maior n, necessitando então de mais interações para cumprir o experimento. Não só isso, precisamos também converter os pontos obtidos pelo quasi-random em pontos advindos de uma distribuição beta, por isso o código demora um pouco mais para ser executado.

De todo modo, o gerador quasi-random se provou muito útil e poderoso, visto que todos os tamanhos de amostra necessários para aproximação de um certo valor diminuíram substancialmente.

O menor n encontrado no EP2 foi perto de $3 \cdot 10^6$ e o maior n encontrado foi da ordem de $6 \cdot 10^7$. Agora, o maior n não passa de 50000. E isso provocou uma melhora significativa nos tempos de execução de cada método, fazendo-os serem mais rápidos e menos custosos computacionalmente.