Instituto de Matemática e Estatística
da Universidade de São Paulo

EP03 - Integração por Métodos de Monte Carlo com Sequências Quasi-Aleatórias

MAP2212 - Laboratório de Computação e Simulação
Prof. Julio M. Stern

Tiago Yukio Fujii
Nº USP 9350011

25 de abril de 2019

Sumário

Capítulo 1

Introdução

Este relatório detalha as atividades realizadas no EP03 da disciplina MAP2212 - Laboratório de Computação e Simulação. No exercício, devem ser realizadas as mesmas atividades do EP02 - implementar e analisar os resultados de quatro variantes do método de Monte-Carlo para integração -, mas agora usando geradores de números quasi-aleatórios ao invés de pseudo-aleatórios. Os resultados obtidos do EP02 estão enumerados na Tabela 1.1.

Método	$\hat{\gamma}$	n	Erro Relativo (%)
Crude	0.678771	3115	0.2546
Hit or Miss	0.684635	17697	0.6042
Importance Sampling	0.675883	787	0.6830
Control Variates	0.678348	445	0.3171

Tabela 1.1: Resultados obtidos no EP02

Capítulo 2

Implementações

Neste capítulo, são apresentadas as modificações realizadas no projeto do EP02, o qual utilizava quatro métodos de Monte Carlo estudados em aula para integrar a seguinte função no intervalo [0, 1]:

$$f(x) = e^{-0.534548064x} cos(0.9350011x)$$

Para implementar as quatro variantes do método de Monte Carlo, utilizou-se a linguagem de programação Python, com uso das bibliotecas math, scipy, sobol_seq [Con], chaospy [Fei] e matplotlib [Hun07]. O projeto possui seis arquivos:

```
main.py
plot.py
crude.py
hitOrMiss.py
importance.py
controlVariates.py
```

Nas seções a seguir, são detalhadas as modificações realizadas em cada arquivo em relação ao EP02.

2.1 Arquivo main.py

No arquivo main.py, utiliza-se a função sobol_seq.i4_sobol_generate(2, 25000) para gerar os 25000 primeiros números da sequência de Sobol em duas dimensões (A segunda dimensão é necessária para o método Hit or Miss).

A biblioteca chaospy é utilizada para gerar uma sequência de Sobol a partir da distribuição Beta(1, 1.5), como mostra a listagem de código a seguir. O argumento 'S' garante que a lista gerada obedece à sequência de Sobol.

```
a = 1
b = 1.5
distribution = chaospy.Beta(a, b)
betaSeq = distribution.sample(25000, 'S')
```

Em seguida são chamadas as implementações dos quatro métodos, presentes nos demais arquivos, para então apresentar os resultados obtidos: a estimativa $\hat{\gamma}$, o número de iterações, e o erro relativo para cada uma das variantes.

2.2 Arquivos das variantes de Monte Carlo

Os métodos Crude, Hit or Miss, e Control Variates utilizaram a sequência de Sobol com distribuição uniforme em [0, 1] para gera números quasi-aleatórios, enquanto a Importance Sampling usa a sequência de Sobol com distribuição Beta(1, 1.5).

As funções de cada método agora recebem um argumento adicional, a sequência de números quasi-aleatórios a ser utilizada. Dentro de cada função, não é mais gerado um número aleatório por cada iteração, utilizando agora um iterador para a sequência quasi-aleatória que recebe o próximo elemento a cada iteração, como mostra a listagem de código a seguir.

```
def run(alpha, beta, sobolSeq):
    sumFx = 0
    sumFx2 = 0
    n = 0
    errorIsBig = True
    seqIter = iter(sobolSeq)

    while errorIsBig:
        x = next(seqIter)[0]

        ...
```

Os critérios de parada utilizados são os mesmos do EP02, de modo a permitir a comparação do número de iterações realizadas entre os dois exercícios.

2.3 Arquivo plot.py

Em relação ao EP02, o único arquivo novo é o plot.py. Nele, é utilizada a biblioteca matplotlib para gerar gráficos das sequências quasi-aleatórias utilizadas.

A Figura 2.1 mostra a sequência de Sobol gerada a partir da biblioteca sobol_seq.

Para melhor visualização da distribuição, a Figura 2.2 mostra os 4675 primeiros elementos (número de iterações utilizadas pelo método Hit or Miss) da sequência de Sobol.

A Figura 2.3 mostra o histrograma da sequência quasi-aleatória de distribuição Beta(1, 1.5) gerada pela biblioteca chaospy. Comparando com o gráfico de Beta(1, 1.5) mostrado na Figura 2.4, pode-se observar que a sequência gerada representa fielmente a distribuição.

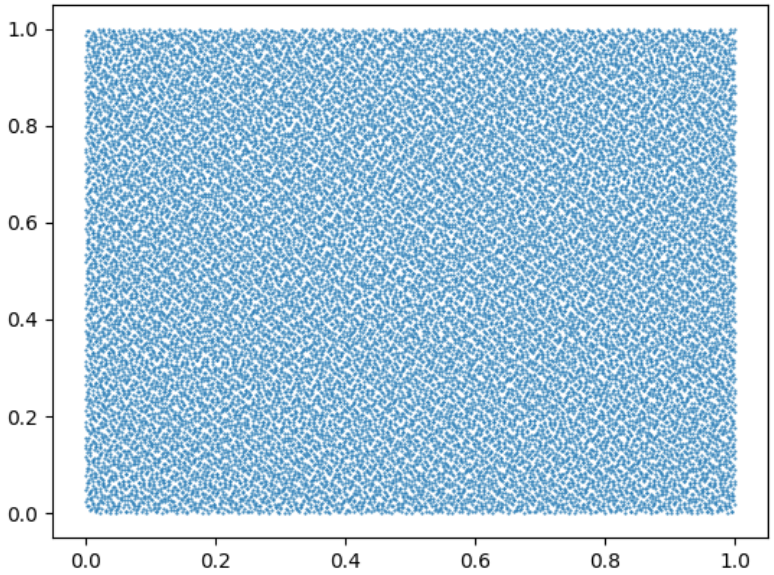

Figura 2.1: 25000 elementos da sequência de Sobol em duas dimensões

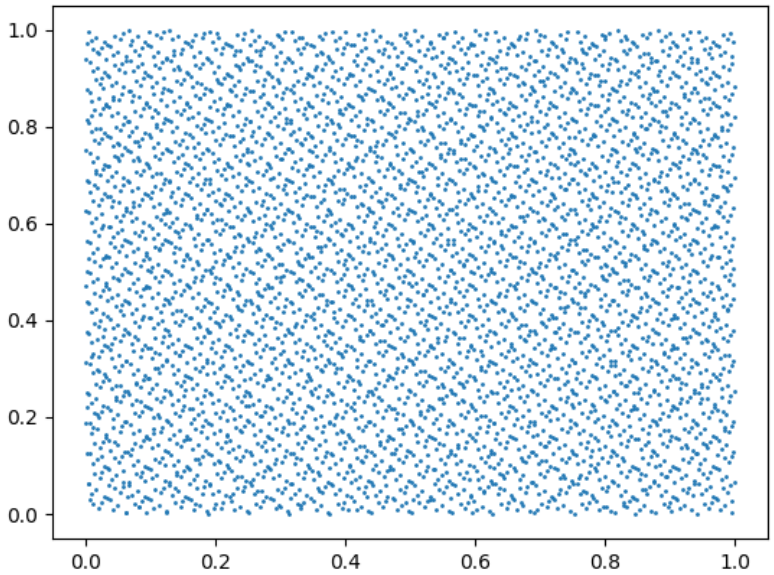

Figura 2.2: 4675 elementos da sequência de Sobol em duas dimensões

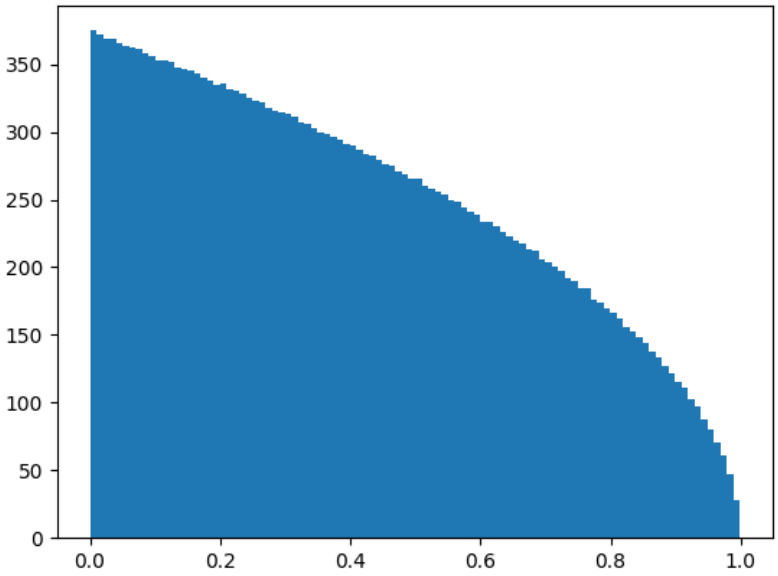

Figura 2.3: Histograma de 25000 elementos da sequência de Sobol com distribuição Beta(1, 1.5)

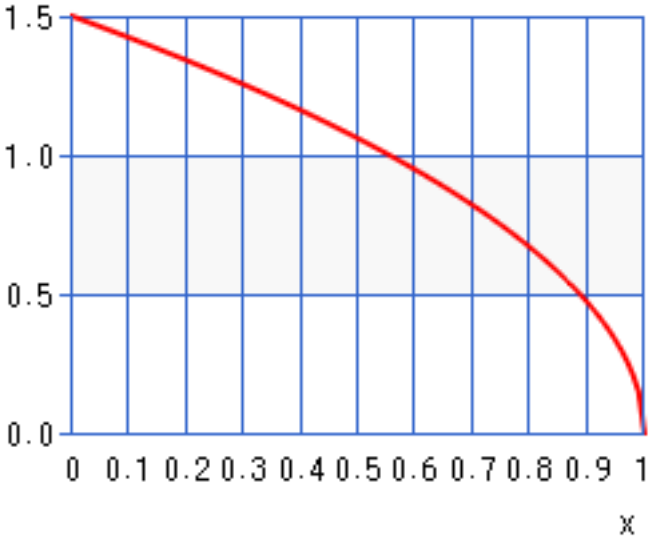

Figura 2.4: Distribuição Beta(1, 1.5)

Capítulo 3

Análise dos Resultados

A Tabela 3.1 mostra a estimativa de γ, o número de iterações realizadas, e o erro relativo dos quatro métodos utilizados.

O erro relativo mostrado refere-se ao seu valor verdadeiro, comparando-se o $\hat{\gamma}$ encontrado pelos métodos executados com o $\gamma = 0.6804991098$ real, calculado numericamente por programas externos. Note que este valor de erro relativo não foi utilizado no critério de parada, mas sim para verificar se os critérios utilizados são condizentes com o esperado pela especificação do exercício.

Para facilitar a leitura desta seção, a Tabela 1.1, a qual informa os resultados obtidos no EP02, está reproduzida nesta seção na Tabela 3.2.

Método	$\hat{\gamma}$	n	Erro Relativo (%)
Crude	0.680208	798	0.0428
Hit or Miss	0.681497	4675	0.1467
Importance Sampling	0.674693	123	0.8533
Control Variates	0.680579	120	0.0118

Tabela 3.1: Resultados da execução com números quasi-aleatórios

Método	$\hat{\gamma}$	n	Erro Relativo (%)
Crude	0.678771	3115	0.2546
Hit or Miss	0.684635	17697	0.6042
Importance Sampling	0.675883	787	0.6830
Control Variates	0.678348	445	0.3171

Tabela 3.2: Resultados obtidos com números pseudo-aleatórios

Em todos os casos, o uso de sequências quasi-aleatórias acelerou a convergência, como pode ser observado na Tabela 3.3. Em geral, ao utilizar sequências de Sobol com distribuições U(0, 1), foram necessárias 75% menos iterações para convergir a um erro relativo inferior a 1%.

O método Importance Sampling, o qual utilizou uma sequência de Sobol com distribuição Beta(1, 1.5), obteve a maior redução relativa, embora tenha um erro relativo maior que as demais.

Método	Número de Iterações		Redução Relativa (%)
	EP02	EP03	
Crude	3115	798	74.38
Hit or Miss	17697	4675	73.58
Importance Sampling	787	123	84.37
Control Variates	445	120	73.03

Tabela 3.3: Redução relativa no número de iterações entre os dois exercícios-programas

Bibliografia

[Hun07] J. D. Hunter. "Matplotlib: A 2D graphics environment". Em: *Computing In Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[Con] Conor.Lawless. *Biblioteca sobol_seq*. URL: https://pypi.org/project/sobol_seq/. Acesso em: 24/04/2019.

[Fei] Jonathan Feinberg. *Biblioteca ChaosPy*. URL: https://chaospy.readthedocs.io/en/master/. Acesso em: 24/04/2019.