EP3 - Método de Monte Carlo

George Othon
NUSP 103xxxxx

April 2020

1 Introdução

Este relatório tem como objetivo analisar os resultados obtidos pelo Método de Monte Carlo nas suas 4 variações ao integrar f(x) utilizando um gerador de quasi-aleatórios. Para gerar os números Quasi-aleatórios foi empregada a biblioteca ChaosPy.

$$f(x) = e^{-ax} \cos bx$$

Com a = 0.RG e b = 0.NUSP, no intervalo [0,1]. (Com RG = 39xxxxxxx e NUSP = 103xxxxx, logo, a = 0.39xxxxxxx e b = 103xxxxx).

2 Bibliotecas

Para nos axiliar no manuseio das listas, gerar os números aleatórios e para utilzar funções matemáticas e estatísticas que não temos nas built-in do Python, foi necessário importar as seguintes bibliotecas com as seguintes funções:

1. Numpy
   (a) exp() - Função exponecial
   (b) cos() - Cosseno
   (c) std() - Desvio padrão
   (d) sqrt() - Raíz quadrada
   (e) mean() - Média
   (f) cov() - Covariância
   (g) var() - Variância
2. ChaosPy
   (a) Uniform() - Distribuição uniforme
   (b) J() - Joint operator

(c) sample() - Amostra

(d) Beta() - Distribuição de probabilidade beta

3. Scipy

(a) stats.beta.pdf() - Função de densidade de probabilidade

3 Gerador quasi-aleatório

A biblioteca ChaosPy foi a escolhida para gerar os números quasi-aleatórios, retornando sempre a mesma lista de números respeitando os requisitos necessários para um gerador quasi-aleatório. Para testar com outros dados geramos a lista de quasi-aleatório e escolhemos um número ao acaso.

4 Critério de parada

Como critério de parada utilizamos o erro padrão, onde a cada iteração verificamos se o erro é menor que 1%, e assim que atendesse o critério, ele calcula a média das iterações e retorna como resultado final.

5 Método Crud

Este foi o primeiro método que testamos e o mais simples onde apenas calculamos $\frac{1}{n}\sum_{i=1}^{n} f(x)$, que rendeu bons resultados, e teve desvio padrão de $\sigma = 0.0170$

6 Método Hit or miss

Ao tentar aproximar a integral de f(x) no intervalo [0,1] geramos diversos pares ordenados (x,y) e verificamos se y $\leq$ f(x), ou seja, se o par ordenado tem imagem acima ou abaixo da função f, e usamos essa proporção para aproximar $\int_0^1 f(x)dx$. Apresentou desvio padrão $\sigma = 0.0100$

7 Método Importance Sampling

No método em questão, utilizamos um gerador de números pseudo-aleatórios com distribuição beta. Após diversos testes tivemos os melhores resultados com os parâmetros $\alpha = 0.9$ e $\beta = 1.0$. Nessas condições tivemos boas aproximações, e com diversos testes o desvio padrão foi de $\sigma = 0.0046$

8 Método Control Variate

No Control Variate, após algumas análises escolhemos como aproximação para f(x), a função $g(x) = e^{-ax}$, com a = 0.399104525. Como $g(x) > f(x)$, $\forall x \in [0, 1]$, e g(x) é extremamente próxima à f(x).

Dividimos este método em duas partes. Na primeira calculamos o resultado da integral pelo método Crud para f(x) e para g(x). Na segunda parte, calculamos o fator **c** dado por

$$c = \frac{-Cov[f(x), g(x)]}{Var[g(x)]}$$

e em seguida calculamos

$$Crud(f(x)) + c * (Crud(g(x) - \int_0^1 g(x)dx)$$

onde $\int_0^1 g(x)dx = 0.824$, para obter a aproximação final, que teve excelentes resultados apresentando desvio padrão $\sigma = 0.0007$

9 Comparando os métodos

Rodamos cada método 100 vezes e fizemos o seguinte resumo estatístico com o auxílio da biblioteca pandas.

Utilizamos o jupiter notebook para gerar a tabela a seguir, e incluimos como uma imagem no artigo. Para a criação da tabela foram usadas as funções DataFrame() e describe().

	Crud	Hit or miss	Importance Sampling	Control Variate
count	100.000000	100.000000	100.000000	100.000000
mean	0.822025	0.831569	0.824188	0.822796
std	0.017094	0.010063	0.004650	0.000743
min	0.729217	0.814527	0.814230	0.821169
25%	0.817982	0.824658	0.820679	0.822208
50%	0.824705	0.829681	0.823678	0.822891
75%	0.830435	0.836953	0.827799	0.823350
max	0.846274	0.860668	0.834564	0.824207

Figure 1: Análise Descritiva (n = 100)

Ao análisar a tabela acima percebemos que para todos os métodos a média e a mediana (50%) fica próximo à 0.82. O método Crud foi o que teve a maior variação e o Control Variate teve a menor. Mas escolhemos o desvio padrão para

analisar o desempenho de cada método, e com base nesse parametro fizemos a seguinte classificação dos métodos.

	Método	Desvio padrão
1	Control Variate	0.0007
2	Importance Sampling	0.0046
3	Hit or miss	0.0100
4	Crud	0.0170

Table 1: Classificação dos métodos

Portanto, dentre os quatro métodos que implementamos, o que retornou os melhores resultados foi o Método Control Variate utilizando a função g como aproximação para f.

9.1 Comparando os geradores

Ao comparar as aproximações com os geradores pseudo e quasi-aleatórios tivemos melhor resultados com o gerador quasi, exceto no método Control Variate que foi o melhor dos métodos com utilizando o gerador de pseudo-aleatório. Classificamos os métodos com cada gerador pelo desvio padrão apresentado e obtivemos a seguinte tabela.

	Método	Gerador	Desvio padrão
1	Control Variate	Pseudo	0.0006
2	Control Variate	Quasi	0.0007
3	Importance Sampling	Quasi	0.0046
4	Importance Sampling	Pseudo	0.0048
5	Hit or miss	Quasi	0.0100
6	Hit or miss	Pseudo	0.0108
7	Crud	Quasi	0.0170
8	Crud	Pseudo	0.0203

Table 2: Comparação métodos e geradores

Desse modo, para uma melhor aproximação para a nossa integral no intervalo [0,1] utilizamos o método Control Variate, que nos proporcionou as duas melhores aproximações alterando entre os geradores pseudo e quasi-aleatório, e apesar da pequena diferença, tivemos melhores resultados quando usamos o gerador pseudo-aleatório.

References

[1] https://numpy.org/doc/

[2] https://chaospy.readthedocs.io/en/master/tutorial.html

[3] https://docs.scipy.org/doc/scipy/reference/stats.html

[4] Cognitive Constructivism and the Epistemic Significance of Sharp Statistical Hypotheses in Natural Sciences - Julio Stern