EP 05 - MAP2212

Gustavo Nunes - 13685534
Rennisson Davi Alves - 13687175

1 O Programa

O programa é o mesmo do EP04, porém com uma mudança no gerador de números de acordo com a distribuição Dirichlet, sem usar o método rvs da função Dirichlet da biblioteca Scipy.

A ideia para o gerador é usar o algorítmo de Metropolis-Hastings, que para o nosso caso pode ser escrito em pseudocódigo:

Algorithm 1 Algorítmo de Metropolis-Hastings para gerador de números aleatórios de acordo com a distribuição Diricihlet

```
Theta1 = ponto arbitrário
f = array de resultados
while c ≤ N do
    Theta2 gerado de Normal Multivariada de média Theta1
    Alpha = Dirichlet(Theta2) / Dirichlet(Theta1)
    if alpha ≥ 1 then
        f[c] = Theta2                                    ▷ Aceita Theta2
    else if 0 ≤ alpha ≤ 1 then
        f[c] = Theta2 com probabilidade alpha
        f[c] = Theta1 com probabiliade (1-alpha)
    end if
    Theta1 = f[c]
end while
return f
```

Para a geração, foram usadas na geração de números de acordo com a normal e com a Dirichlet a biblioteca "Scipy". Na normal, foi usado o parâmetro "cov = 0.6", que termina a matriz de covariância que empíricamente rejeita $\approx 30\%$ dos Thetas gerados. Na Dirichlet, o parâmetro usado é gerado aleatóriamente na função "main", assim como no EP04.

Estão abaixo o resultado de alguns testes com o gerador modificado:

N	v	U(v)	Tempo de execução
5000	0	0.0010	0
5000	4	0.1150	0.0002
5000	8	0.2560	0.0005
5000	12	0.4030	0.0006
5000	16	0.5750	0.0009
5000	20	0.7150	0.0014
5000	24	0.8890	0.0016
5000	27	1	0.0.0016

Para medida de precisão, foram gerados 100 amostras diferentes e medido quantas ficaram dentro do intervalo de confiança pedido, com a referência sendo um N = 10^6 arbitráriamente grande e $k = 10^4$ suficiente para garantir a precisão dado N grande suficiente.

2 Conclusão

De acordo com os testes realizados e mostrados na tabela da seção anterior, os resultados obtidos são extremamente satisfatórios para o objetivo do que foi proposto. Para cada corte v, o algoritmo foi capaz de se aproximar do valor real da massa da função W(v) com precisão acima de 99.95%.

Vale lembrar que para gerar todos os arrays necessários, foi utilizado o NUSP 13687175 como seed. Desse modo, fica mais fácil reproduzir os testes e verificar seus resultados.

Não obstante, os tempos de execução de cada experimento após a geração dos números foram excelentes, não ultrapassando o tempo de 1 segundo. Claro que o valor máximo de corte para analise da função é relativamente pequeno (26.95 para ser mais exato), tornando o número de comparações também pequeno.

O problema do gerador é que como cada número precisa ser gerado com um loop independentemente, não sendo possível o uso de métodos mais rápidos de vetores do Numpy e do Scipy, o tempo de geração dos números é consideravelmente maior, oque é de ser esperado dado um algoritmo para situações em que não conseguimos gerar os números diretamente com a distribuição desejada.