**Prática 4 – MS**

**Grupo:**

Andressa Alvilino Ferreira Silva

Matheus Cunha Reis

Weuler Borges

**4.1.7)** (a) Generate an Exponential (9) random variate sample of size n = 100 and compute the proportion of points in the sample that fall within the intervals x̄ ± 2s and x̄ ± 3s. Do this for 10 different rngs streams. (b) In each case, compare the results with Chebyshev’s inequality. (c) Comment.

**a)** Ver código.

**b)** Ver código.

**c)** No caso de k = 2, a Desigualdade de Chebyshev nos diz que pk >= 0.75 e conforme o código, o resultado para todas as streams conferem.

No caso de k = 3, a Desigualdade de Chebyshex nos diz que pk>=0.89 e também, conforme o código, o resultado para todas as streams conferem.

**4.1.8)** Generate a plot similar to that in Figure 4.1.2 with calls to Exponential (17), rather than Random to generate the variates. Indicate the values to which the sample mean and sample standard deviation will converge.

Valor converge para µ = 18.

Onde as bolinhas laranjas são os valores de 'x', e as azuis são a média no primeiro gráfico e desvio padrão no segundo.

**4.1.11)** Calculate x̄ and s by hand using the 2-pass algorithm, the 1-pass algorithm, and Welford’s algorithm in the following two cases. (a) The data based on n = 3 observations: x 1 = 1, x 2 = 6, and x 3 = 2. (b) The sample path x(t) = 3 for 0 < t ≤ 2, and x(t) = 8 for 2 < t ≤ 5, over the time interval 0 < t ≤ 5.

**a)**

n = 3;

x1 = 1;

x2 = 6;

x3 = 2;

• 2-pass-algorithm

x¯ = 1 + 6 + 1 / 3 = 3

s = √(1-3)²+(6-3)²+(2-3)/3) = √(4+9+1/3) = √(4,667)

• 1-pass-algorithm

√s² = √(((1²+6²+1² )/ 3) – ((1+6+2)/3)²)

√(41-27)/3 = √(14/3)

x¯ = √9 = 3

• Welford

x¯1 = 0+1 = 1

x¯2 = 1+ (6-1)/2 = 3,5

x¯3 = 3,5 + (2-3,5)/3 = 3

V1 = 0

V2 = 0+(6-1)/2 = 5/2

V3 = 2,5 + 2 \* (2 – 3,5) /3 = 1,5

**b)**

x(t) = 3 0< t <= 2;

x(t) = 8 2< t <=5;

• Two pass

x = 1/5 \* [ 3\* (2-0) + 8 \*(5 -2)] = 30/5 = 6

s² = [ integral de 0 a 2 (3-6) dt + integral de 2 a 5(8-6)² dt] = 1/5 \* (18+12) = 6

s² = 6

√6 = 2,44

• One pass

s² = [1/5 (9\*(2-0) + 64 (5-2)

s² = 1/5 \* (18+192) – 6 = 210/5 – 6 = 42 -36

= √6 = 2,44

X = 6

• Welford

x1 = 0 + 1 \* (3-0)/2 = 3

x2 = 3 + 3\*(8-3)/5 = 3+3 = 6

v1 = 0 + 0\*(3-0)² = 0

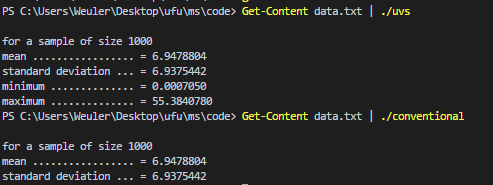
v2 = 0 + (3\*2) /5 \* (8-0)² = 6.64/5 = √6

**4.1.13)** (a) Generate an Exponential(7) random variate sample of size n = 1000 and compute the mean and standard deviation using the Conventional One pass algorithm and the Algorithm 4.1.1. Comment on the results.

**a)** Com o Código generate\_data\_exponential.c nós geramos os dados necessários para este experimento. O output foi jogado para um arquivo “data.txt” que será usado a frente.

Executaremos o código conventional\_pass.c (implementa o Conventional One pass) e o código uvs.c (Algorithm 4.1.1).

Segue print das execuções:



É interessante notar dessas execuções que apesar de o Algoritmo 4.1.1 calcular de forma mais robusta e com menos erros de precisão o desvio padrão, em situações onde o desvio é próximo da média erros não devem ocorrer frequentemente. Por isso, ambos algoritmos calcularam exatamente o mesmo valor de desvio padrão.

**4.2.2)** (a) Generate the 2000-ball histogram in Example 4.2.2. (b) Verify that the resulting relative frequencies f(x) satisfy the equation

f(x) ∼ (2^x \* exp(−2))/x! x = 0, 1, 2, . . .

(c) Then generate the corresponding histogram if 10 000 balls are placed, at random, in 1000 boxes. (d) Find an equation that seems to fit the resulting relative frequencies well and illustrate the quality of the fit.

**a)** Com o programa **generate\_ball.c** geramos usando a função Equilikely(1,n) lançamentos conforme o exemplo 4.2.2. Alem disso, nesse mesmo código, calculamos as frequências em cada caixa e com essas frequências, calculamos as frequências relativas. Usamos um valor ate 25 intencionalmente alto para servir também para o próximo item.

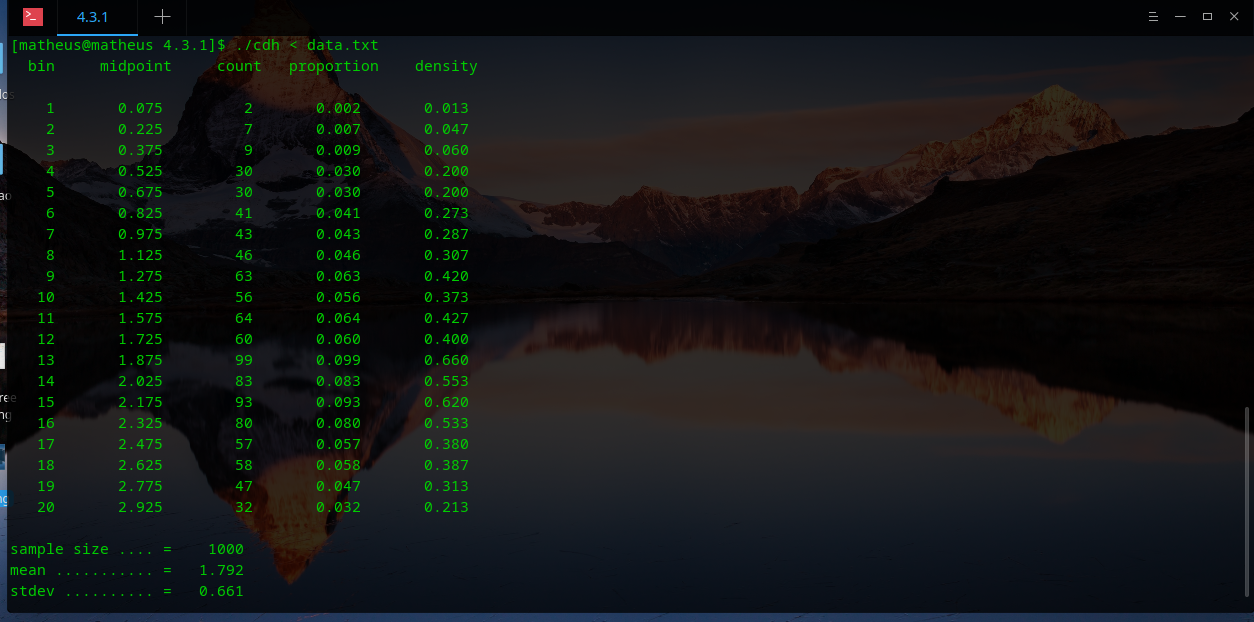
**b)** Ainda no código **generate\_ball.c** calculamos a função f(x) e printamos os valores para calcular o seguinte histograma. Como esperado, ele tem o mesmo formato e valores próximos do resultado aleatório.

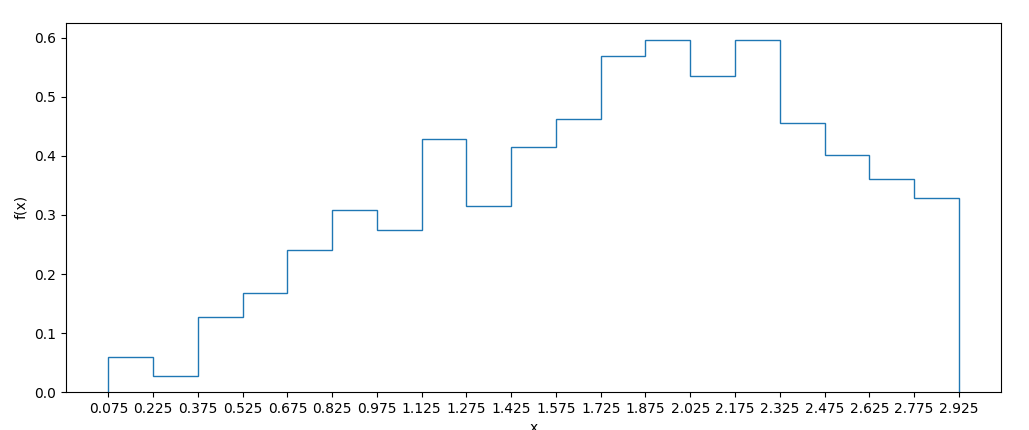
**c)** Histograma gerado também usando o generate\_ball.c modificando apenas o numero de lançamentos feitos de 2000 para 10000.

**d)** A função que fit esses dados é similar a f(x) do item b, com a diferença que usamos 10 como a base da exponenciação e -10 em exp. Fazemos isso pois 10 é a média esperada para este exemplo. O fit é de alta qualidade pois o formato da função experimental é bastante parecido com a esperada, o que demonstra que encontramos provavelmente a melhor função.

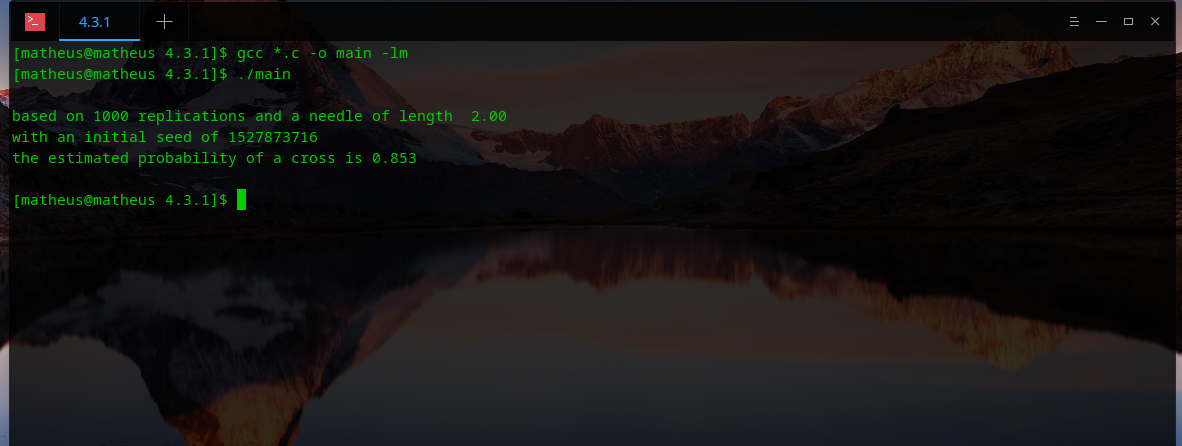
**4.3.1)** (a) Use program cdh to construct a continuous-data histogram like the one on the left in Example 4.3.1, but corresponding to a needle of length r = 2. (b) Based on this histogram what is the probability that the needle will cross at least one line. (c) What is the corresponding axiomatic probability that a needle of length r = 2 will cross at least one line?

**a)** Código: generateRandomNumbers.c cdh.c generateHistogram.py

****

****

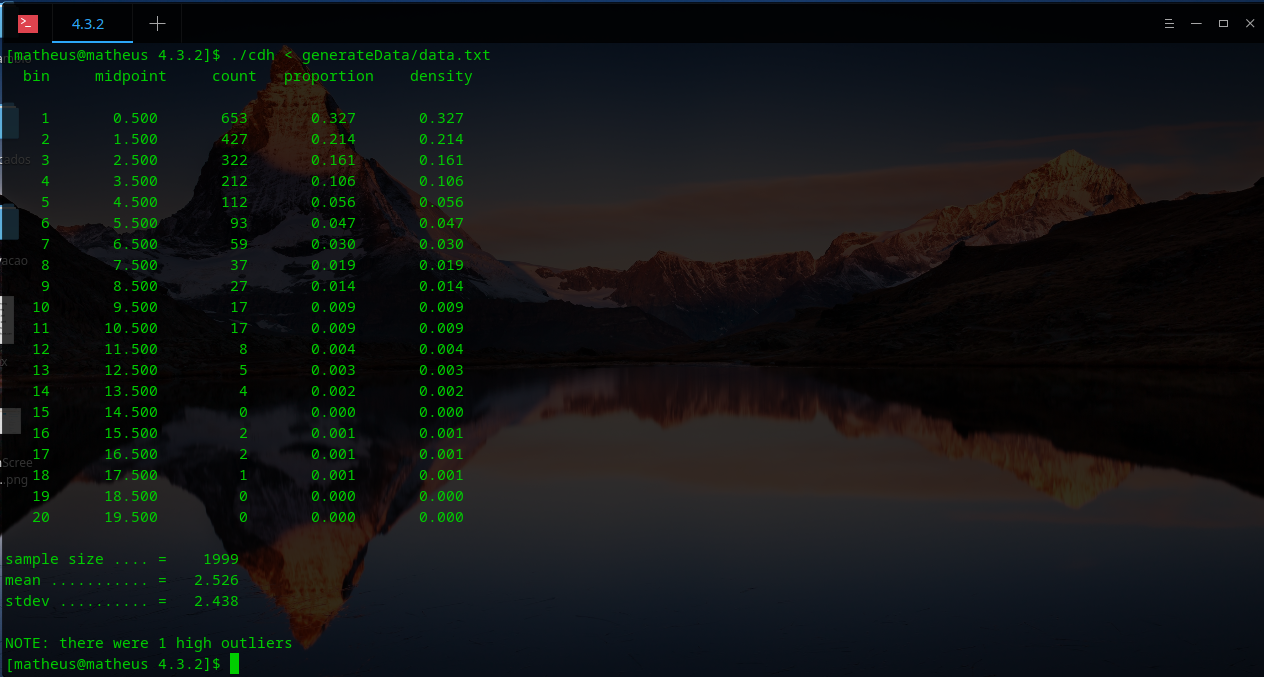
**b)** A probabilidade é de 85,3 %

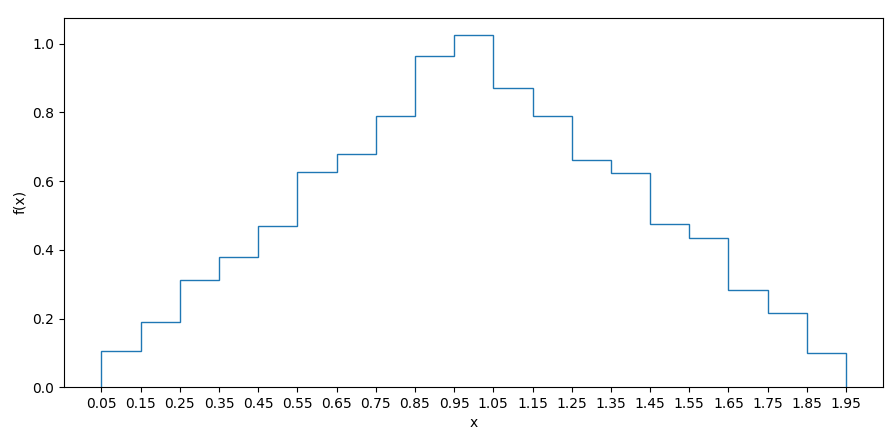


**c)**

**4.3.2)** Repeat the experiment in Example 4.3.6 with t = 5000 and n = 2000. Do not use a bubble sort.

Código: generateData.cpp cdh.c generateHistogram.py



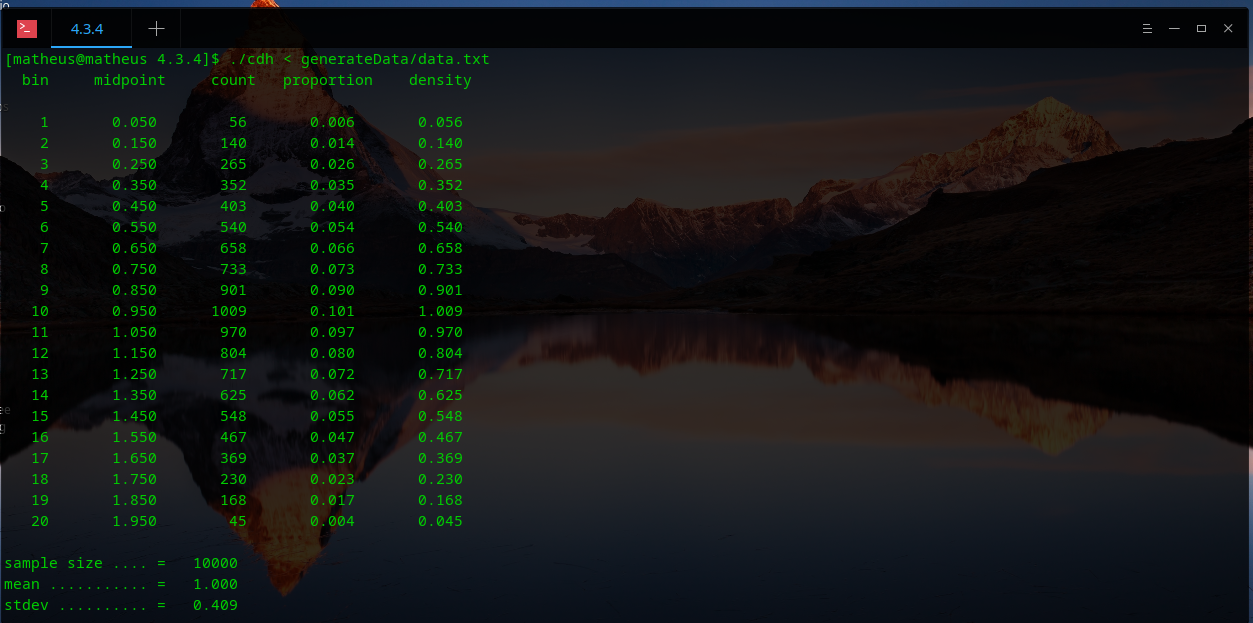


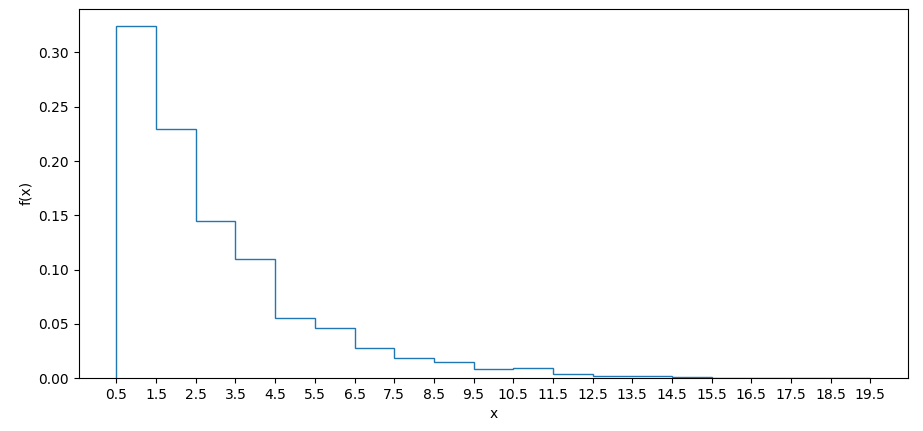
**4.3.4)** Generate a random variate sample x1 , x 2 , . . . , x n of size n = 10000 as follows:

for (i = 1; i <= n; i++) X[i] = Random() + Random();

(a) Use program cdh to construct a 20-bin continuous-data histogram. (b) Can you find an equation that seems to fit the histogram density well?

**a)** Código: generateRandomNumbers.c cdh.c





**b)** Uma equação que se encaixa razoavelmente bem ao histograma é:

