Федеральное Государственное Бюджетное Общеобразовательное
Учреждение Высшего Образования
Московский государственный технический университет
имени Н. Э. Баумана

(Национальный Исследовательский Университет)

Факультет ИУ «Информатика и системы управления»

Кафедра ИУ3 «Информационные системы и телекоммуникации»

Домашняя работа №2 по дисциплине "Теория информационных процессов и систем"

Группа ИУ3-51Б

Выполнил: Магомедов В.О.

Проверил: Бабкин П. С.

Москва, 2020

Задание:

Сгенерируйте случайные числа Гаусса и Рэлея с помощью преобразования Бокса- Мюллера. Преобразование Бокса-Мюллера - это
метод выборки псевдослучайных чисел для генерации пар независимых, стандартных, нормально распределенных (нулевое среднее,

единичная дисперсия) случайных чисел, заданных источником равномерно распределенных случайных чисел. Преобразование Бокса-Мюллера обычно выражается в двух формах. Основная форма, данная Боксом и Мюллером, берет две выборки из равномерного распределения на интервале [0; 1] и сопоставляет их с двумя стандартными, нормально распределенными выборками. Полярная форма берет два образца из другого интервала, [-1; +1] и сопоставляет их с двумя нормально распределенными выборками без использования функций синуса или косинуса.

Напишите компьютерную программу, используя Matlab которая
1. генерирует вектор гауссовских случайных чисел длиной 10^7, используя обе формы преобразования Бокса-Мюллера,
2. генерирует вектор случайных чисел Рэлея длиной 10^7, используя обе формы преобразования Бокса-Мюллера,
3. рисует эмпирическую функцию распределения (гистограмму) каждого случайного вектора (вы можете использовать функцию hist в
Matlab),
4. генерирует вектор гауссовских случайных чисел длиной 10^7, используя функцию randn из Matlab,
5. генерирует вектор случайных чисел Рэлея длиной 10^7, используя raylrnd-функцию Matlab,
6. сравнивает распределение созданных вами случайных чисел с распределением случайных чисел Гаусса и Рэлея сгенерированных с помощью функций randn и raylrnd из Matlab, с использованием метрики расстояния Кульбака-Лейблера (KL).
Расстояние KL между двумя дискретными распределениями p и q определяется выражением:

$$D(p||q) = \sum_x p(X = x) \log\left(\frac{p(X = x)}{q(X = x)}\right) \quad (6)$$

Цель работы:

С помощью преобразования Бокса-Мюллера получить вектора Рэлеевских и Гауссовских случайных величин. Сравнить данные вектора с помощью расстояния Кульбака-Лейблера с векторами, полученными с помощью встроенных функций.

Теоретическая часть:

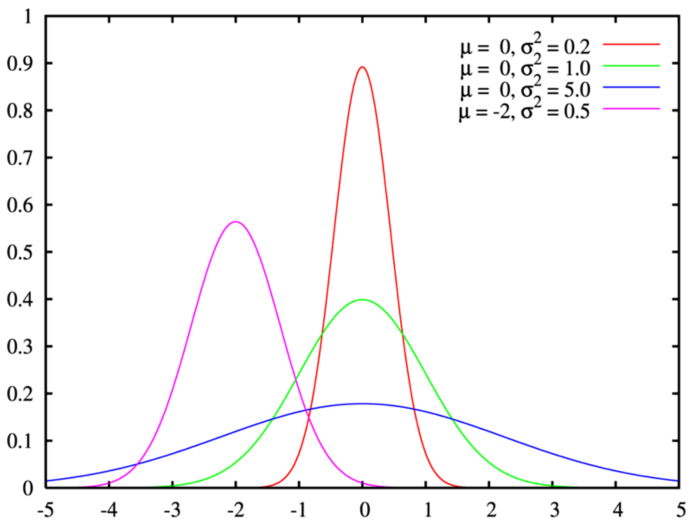
Нормальное распределение (распределение Гаусса) – распределение вероятностей, которое в одномерном случае задается функцией плотности вероятности, совпадающей с функцией Гаусса:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

, где μ - мат ожидание

σ - среднеквадратичное отклонение

σ^2 - дисперсия



Преобразование Бокса-Мюллера

1 вариант:

$$\left.\begin{aligned} z_0 &= \cos(2\pi\varphi)\sqrt{-2\ln r} \\ z_1 &= \sin(2\pi\varphi)\sqrt{-2\ln r} \end{aligned}\right\}$$

мат ожидание $= 0$; r, φ - равномерно распр $[0;1]$

дисперсия $= 1$

$$\sin(2\pi\varphi) = \sqrt{1-\cos^2(2\pi\varphi)} \Rightarrow \begin{aligned} z_0 &= \cos(2\pi\varphi)\sqrt{-2\ln r} \\ z_1 &= \sqrt{1-\cos^2(2\pi\varphi)}\cdot\sqrt{-2\ln r} \end{aligned}$$

2 вариант

$$z_0 = x\cdot\sqrt{\frac{-2\ln s}{s}}$$

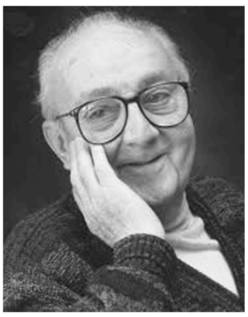
$$z_1 = y\cdot\sqrt{\frac{-2\ln s}{s}}$$

, где x, y независим сл. величины, равномерно распр. на $[-1;1]$;

Условие:

$s = x^2 + y^2$, где $s > 1$ или $s = 0$ то знач-ие x и y следует „выбросить" и сгенер. заново.

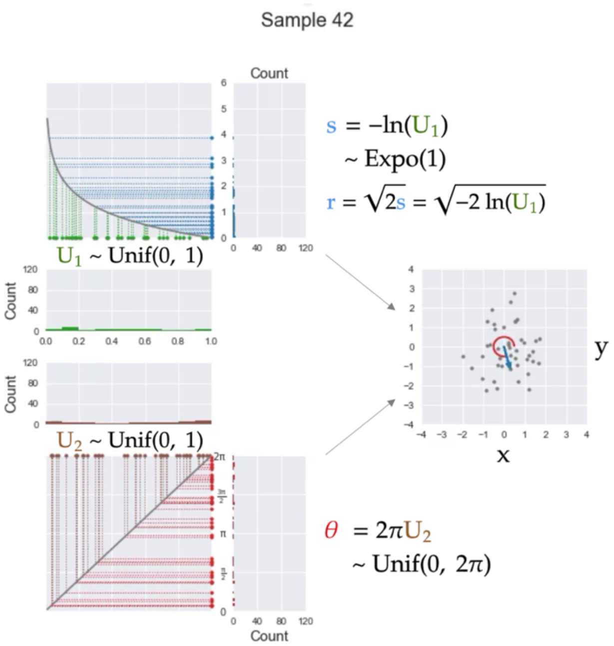
George E. P. Box
(1919–2013)



Mervin E. Muller
(1928–2018)

Box-Muller transform

$$x = r\cos(\theta) = \sqrt{-2\ln(U_1)}\cos(2\pi U_2)$$
$$y = r\sin(\theta) = \sqrt{-2\ln(U_1)}\sin(2\pi U_2)$$



Формирование величин, распределенных по закону Релея

Распределение неотрицательной случайной величины ρ с плотностью вероятностей

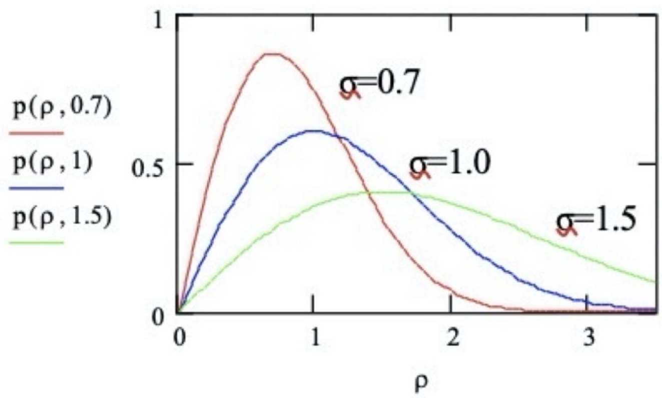
$$P(\rho) = \begin{cases} 0 & \text{при } \rho < 0 \\ \frac{\rho}{\sigma^2} \cdot \exp\left(-\frac{\rho^2}{2 \cdot \sigma^2}\right) & \text{при } \rho > 0 \end{cases}$$

называется распределением Релея

Математическое ожидание для распределения Релея $M\rho = 1{,}253\sigma$

Дисперсия $D\rho = 0{,}43\sigma$

(Здесь σ – параметр распределения)

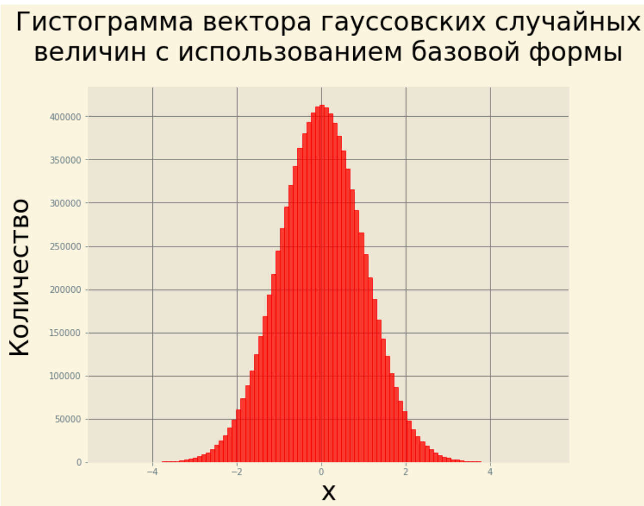


Практическая часть:

In [46]:
```
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import uniform as U, normal as N, rayleigh as R
plt.style.use('Solarize_Light2')
def BM_rayl_p(N):
    u1, u2 = U(-1, 1, N*2),U(-1, 1, N*2)
    u = u1**2 + u2**2
    cond = (u!=0)&(u<=1)
    u = u[cond][:N]
    a = np.sqrt(-2*np.log(u))
    return a

def BM_norm(N):
    return np.cos(2*np.pi*U(0, 1, N))*np.sqrt(-2*np.log(U(0, 1, N) ))

def BM_norm_p(N):
    u1, u2 = U(-1, 1, N*2), U(-1, 1, N*2)
    u = u1**2 + u2**2
    cond = (u!=0)&(u<=1)
    u, u1 = u[cond][:N], u1[cond][:N]
    a = u1 * np.sqrt(-2*np.log(u)/u)
    return a

def BM_rayl(N):
    return np.sqrt(-2*np.log(U(0, 1, N)))

def D_KL(p, q):
    N = len(p)
    p = np.sort(p)
    q = np.sort(q)
    c = int(np.floor(1+np.log2(N)))
    maximum = max(np.max(p), np.max(q))
    minimum = min(np.min(p), np.min(q))
    height = (maximum - minimum)/c
    l = np.arange(minimum, maximum, height)
    r = l + height
    l[0] = l[0] - 0.1
    r[-1] = r[-1] + 0.1
    intervals = [(left, right) for left, right in zip(l, r)]
    probs = np.zeros((len(intervals), 2))
    for num, d in enumerate((p, q)):
        k = 0
        for X in d:
            if X>=intervals[k][0] and X<intervals[k][1]:
                probs[k][num] += 1
            else:
                probs[k+1][num] += 1
                k += 1
    probs /= N
    eps = 1e-10
    Dkl = sum([p*np.log((p+eps)/(q+eps)) for p, q in zip(probs[:, 0], probs[:, 1])])
    return Dkl
print(D_KL(norm, norm_b), D_KL(norm, norm_p), D_KL(rayl, rayl_b), D_KL(rayl, rayl_p))
```

```
3.839385778553463e-06 1.3079146721819004e-06 2.04645338960247l3e-06 1.3132745965034285
e-06
```

In [47]:
```
norm_b = BM_norm(10**7)
norm_p = BM_norm_p(10**7)
plt.figure(figsize=(10, 8))
plt.hist(norm_b, bins=100, facecolor='red', alpha=0.75, edgecolor='red')
plt.xlabel('x', fontsize=30,color = 'black')
plt.ylabel('Количество', fontsize=30, labelpad=20,color = 'black')
plt.title('''Гистограмма вектора гауссовских случайных
величин с использованием базовой формы''',
          fontsize=30, pad=30)
plt.grid(color = 'grey')
```



In [48]:

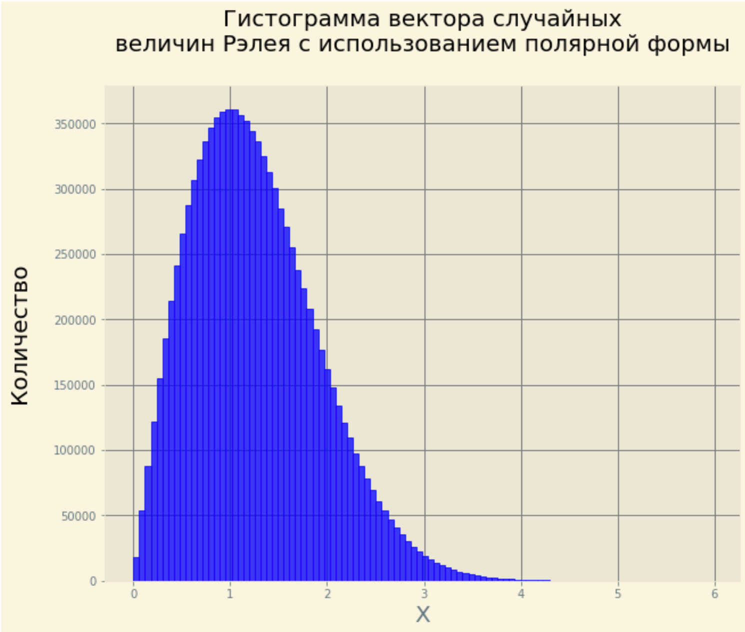
```
plt.figure(figsize=(10, 8))
plt.hist(norm_b, bins=100, facecolor='blue', alpha=0.75, edgecolor='blue')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', fontsize=20, labelpad=20,color = 'black')
plt.title('''Гистограмма вектора гауссовских случайных
величин с использованием полярной формы''',
          fontsize=20, pad=30,color = 'black')
plt.grid(color = 'grey')
```

In [49]:

```
rayl_b = BM_rayl(10**7)
rayl_p = BM_rayl_p(10**7)
plt.figure(figsize=(10, 8))
plt.hist(rayl_b, bins=100, facecolor='red', alpha=0.75, edgecolor='red')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', fontsize=20, labelpad=20,color = 'black')
plt.title('''Гистограмма вектора случайных
величин Рэлея с использованием базовой формы''',
          fontsize=20, pad=30,color = 'black')
plt.grid(color = 'grey')
```

In [50]:

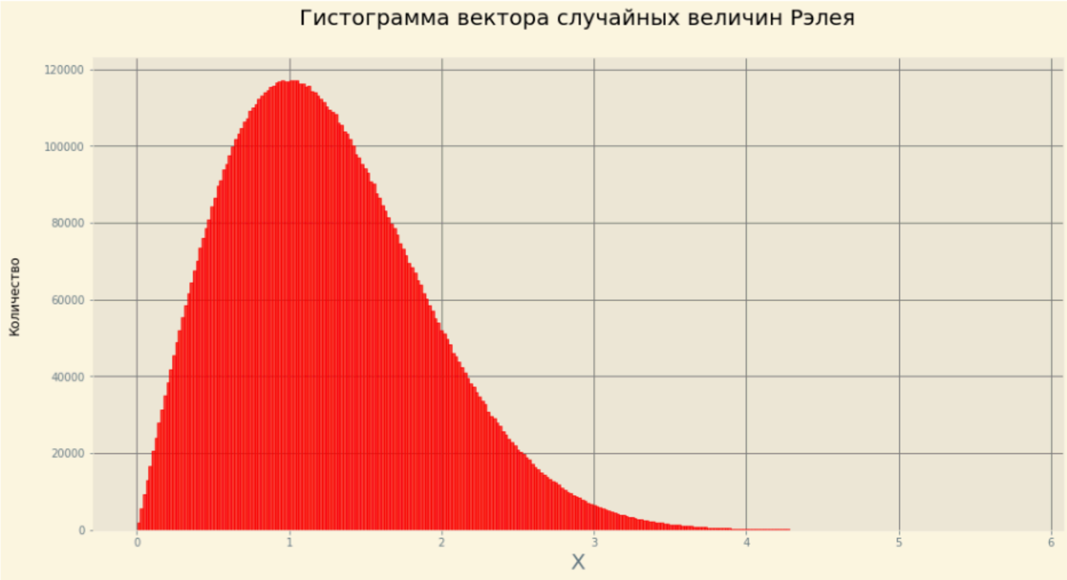
```
plt.figure(figsize=(10, 8))
plt.hist(rayl_b, bins=100, facecolor='blue', alpha=0.75, edgecolor='blue')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', fontsize=20, labelpad=20,color = 'black')
plt.title('''Гистограмма вектора случайных
величин Рэлея с использованием полярной формы''',
          fontsize=20, pad=30,color = 'black')
plt.grid(color = 'grey')
```



In [51]:

```
norm = N(size=10**7)
rayl = R(size=10**7)
plt.figure(figsize=(16, 8))
plt.hist(norm, bins=300, facecolor='blue', alpha=0.75, edgecolor='blue')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', labelpad=20,color = 'black')
plt.title('''Гистограмма вектора гауссовских случайных величин''',
          fontsize=20, pad=30,color = 'black')
plt.grid(color = 'grey')
```

In [52]:
```
plt.figure(figsize=(16, 8))
plt.hist(rayl, bins=300, facecolor='red', alpha=0.75, edgecolor='red')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', labelpad=20,color = 'black')
plt.title('''Гистограмма вектора случайных величин Рэлея''',
          fontsize=20, pad=30,color = 'black')
plt.grid(color = 'grey')
```



Сравним распределение созданных случайных чисел с распределением случайных чисел Гаусса и Рэлея сгенерированных с помощью функций, с использованием метрики расстояния Кульбака-Лейблера.

	Вектор гауссовских величин	Вектор величин Релея
Базовая форма	0,0000038393857785	0,000001307914672
Полярная форма	0,000002046453389	0,0000013132745965

Значение метрики расстояния Кульбака-Лейблера

Вывод

Были сгенерированы с помощью преобразования Бокса-Мюллера вектора Гауссовских и Рэлеевских случайных величин. Мы сравнили вектора с полученными с помощью встроенных функций векторами, с помощью расстояния Кульбака-Лейблера.

Листинг программы:

jupyter tipis_dz2.py несколько секунд назад

Разлогиниться

Файл Редактировать Вид Язык Python

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import uniform as U, normal as N, rayleigh as R
plt.style.use('Solarize_Light2')
def BM_rayl_p(N):
    u1, u2 = U(-1, 1, N*2),U(-1, 1, N*2)
    u = u1**2 + u2**2
    cond = (u!=0)&(u<=1)
    u = u[cond][:N]
    a = np.sqrt(-2*np.log(u))
    return a

def BM_norm(N):
    return np.cos(2*np.pi*U(0, 1, N))*np.sqrt(-2*np.log(U(0, 1, N) ))

def BM_norm_p(N):
    u1, u2 = U(-1, 1, N*2), U(-1, 1, N*2)
    u = u1**2 + u2**2
    cond = (u!=0)&(u<=1)
    u, u1 = u[cond][:N], u1[cond][:N]
    a = u1 * np.sqrt(-2*np.log(u)/u)
    return a

def BM_rayl(N):
    return np.sqrt(-2*np.log(U(0, 1, N)))

norm_b = BM_norm(10**7)
norm_p = BM_norm_p(10**7)
plt.figure(figsize=(10, 8))
plt.hist(norm_b, bins=100, facecolor='green', alpha=0.75, edgecolor='#000000')
plt.xlabel('x', fontsize=20)
plt.ylabel('Количество', fontsize=20, labelpad=20)
plt.title('''Гистограмма вектора гауссовских случайных
величин с использованием базовой формы''',
          fontsize=20, pad=30)
plt.grid(True)
plt.figure(figsize=(10, 8))
plt.hist(norm_b, bins=100, facecolor='#2132bF', alpha=0.75, edgecolor='#000000')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', fontsize=20, labelpad=20)
plt.title('''Гистограмма вектора гауссовских случайных
величин с использованием полярной формы''',
          fontsize=20, pad=30)
plt.grid(True)
rayl_b = BM_rayl(10**7)
rayl_p = BM_rayl_p(10**7)
plt.figure(figsize=(10, 8))
plt.hist(rayl_b, bins=100, facecolor='#5F320F', alpha=0.75, edgecolor='#000000')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', fontsize=20, labelpad=20)
plt.title('''Гистограмма вектора случайных
величин Рэлея с использованием базовой формы''',
          fontsize=20, pad=30)
plt.grid(True)
plt.figure(figsize=(10, 8))
plt.hist(rayl_b, bins=100, facecolor='#A3B4DF', alpha=0.75, edgecolor='#000000')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', fontsize=20, labelpad=20)
plt.title('''Гистограмма вектора случайных
величин Рэлея с использованием полярной формы''',
          fontsize=20, pad=30)
plt.grid(True)
norm = N(size=10**7)
rayl = R(size=10**7)
plt.figure(figsize=(16, 8))
plt.hist(norm, bins=300, facecolor='#43AFA3', alpha=0.75, edgecolor='#000000')
plt.xlabel('X', fontsize=20)
plt.ylabel('Количество', labelpad=20)
plt.title('''Гистограмма вектора гауссовских случайных величин''',
          fontsize=20, pad=30)
plt.grid(True)
plt.figure(figsize=(16, 8))
plt.hist(rayl, bins=300, facecolor='#635FA3', alpha=0.75, edgecolor='#000000')
plt.xlabel('X', fontsize=20)
```

```
plt.ylabel('Количество', labelpad=20)
plt.title('''Гистограмма вектора случайных величин Рэлея''',
          fontsize=20, pad=30)
plt.grid(True)
def D_KL(p, q):
    N = len(p)
    p = np.sort(p)
    q = np.sort(q)
    c = int(np.floor(1+np.log2(N)))
    maximum = max(np.max(p), np.max(q))
    minimum = min(np.min(p), np.min(q))
    height = (maximum - minimum)/c
    l = np.arange(minimum, maximum, height)
    r = l + height
    l[0] = l[0] - 0.1
    r[-1] = r[-1] + 0.1
    intervals = [(left, right) for left, right in zip(l, r)]
    probs = np.zeros((len(intervals), 2))
    for num, d in enumerate((p, q)):
        k = 0
        for X in d:
            if X>=intervals[k][0] and X<intervals[k][1]:
                probs[k][num] += 1
            else:
                probs[k+1][num] += 1
                k += 1
    probs /= N
    eps = 1e-10
    Dkl = sum([p*np.log((p+eps)/(q+eps)) for p, q in zip(probs[:, 0], probs[:, 1])])
    return Dkl
print(D_KL(norm, norm_b), D_KL(norm, norm_p), D_KL(rayl, rayl_b), D_KL(rayl, rayl_p))
```