# Министерство образования и науки Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра теоретической и прикладной информатики

### Лабораторная работа № 5 по дисциплине «Компьютерное моделирование»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| сигма градиент синий1 | Факультет: | ПМИ |  |  |
| Группа: | ПМИ-62 |  |  |
| Студенты: | Ершов П.К., Мамонова Е.В., Цыденов З.Б. |  |  |
| Вариант: | 9 |  |  |
| Преподаватель: | Волкова В.М. |  |  |

Новосибирск

2019

1. **Цель работы**

Научиться моделировать значения непрерывно распределённой случайной величины методом исключений и проводить статистический анализ сгенерированных данных.

1. **Задание**

| Вариант | Распределение | Параметры закона распределения | Непараметрический критерий |
| --- | --- | --- | --- |
| 9 | Бета | μ = 2, ν = 2; μ = 1.5, ν = 2.6; μ = 4.2, ν = 1.5 | Критерий Ω2-Андерсона-Дарлинга |

1. **Описание формата входного файла**

В файле data.txt находятся входные данные для всех тестов.

50 – максимальное количество чисел N в последовательности

200 – максимальное количество чисел N в последовательности

1000 - максимальное количество чисел N в последовательности

2 – μ параметр распределения

1.5 – μ параметр распределения

4.2 – μ параметр распределения

2 – v параметр распределения

2.6 – v параметр распределения

1.5 – v параметр распределения

0.05 – уровень значимости α

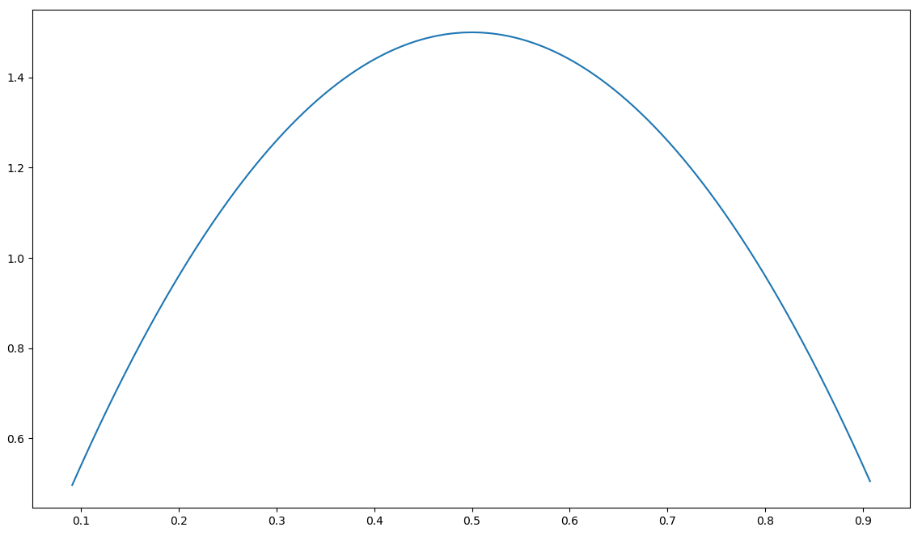
1. **Ход работы**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Длина последовательности** | 50 | | | | | |
| **Параметры распределения** | μ | v | μ | v | μ | v |
| 2 | 2 | 1.5 | 2.6 | 4.2 | 1.5 |
| **Последовательность** | 0.43, 0.51, 0.751, 0.48, 0.308, 0.605, 0.333, 0.476, 0.442, 0.48, 0.289, 0.544, 0.694, 0.787, 0.825, 0.346, 0.228, 0.768, 0.13, 0.508, 0.683, 0.778, 0.578, 0.586, 0.567, 0.226, 0.6, 0.48, 0.536, 0.682, 0.457, 0.76, 0.6, 0.615, 0.274, 0.162, 0.761, 0.779, 0.66, 0.62, 0.568, 0.091, 0.908, 0.168, 0.149, 0.254, 0.368, 0.504, 0.235, 0.87 | | 0.07, 0.234, 0.396, 0.421, 0.504, 0.285, 0.731, 0.496, 0.65, 0.06, 0.225, 0.32, 0.326, 0.558, 0.628, 0.372, 0.022, 0.305, 0.562, 0.372, 0.616, 0.583, 0.399, 0.155, 0.434, 0.44, 0.417, 0.558, 0.362, 0.289, 0.629, 0.104, 0.258, 0.106, 0.357, 0.254, 0.134, 0.573, 0.76, 0.037, 0.598, 0.793, 0.337, 0.197, 0.029, 0.304, 0.63, 0.547, 0.267, 0.486 | | 0.617, 0.649, 0.564, 0.572, 0.923, 0.875, 0.774, 0.913, 0.723, 0.97, 0.324, 0.81, 0.646, 0.624, 0.76, 0.584, 0.776, 0.424, 0.951, 0.976, 0.901, 0.929, 0.631, 0.753, 0.711, 0.916, 0.876, 0.788, 0.324, 0.895, 0.437, 0.512, 0.966, 0.918, 0.685, 0.849, 0.921, 0.658, 0.864, 0.839, 0.844, 0.637, 0.947, 0.867, 0.787, 0.552, 0.612, 0.725, 0.667, 0.711 | |
| **Время моделирования** | 0.126 | | 1.48 | | 1 | |
| **Количество равномерных чисел, которые потребовались для генерации** | 50 | | 30 | | 64 | |
| **Значение статистики S** | 5.19 | | 4.79 | | 2.75 | |
| **Достигнутый уровень значимости** | 0.52 | | 0.57 | | 0.84 | |
| **Результаты проверки критерия *χ*2** | Пройден | | Пройден | | Пройден | |
| **Значение статистики S** | 1.25 | | 0.643 | | 1.196 | |
| **Результаты проверки критерия** **Ω2-Андерсона-Дарлинга** | Не пройден | | Пройден | | Не пройден | |

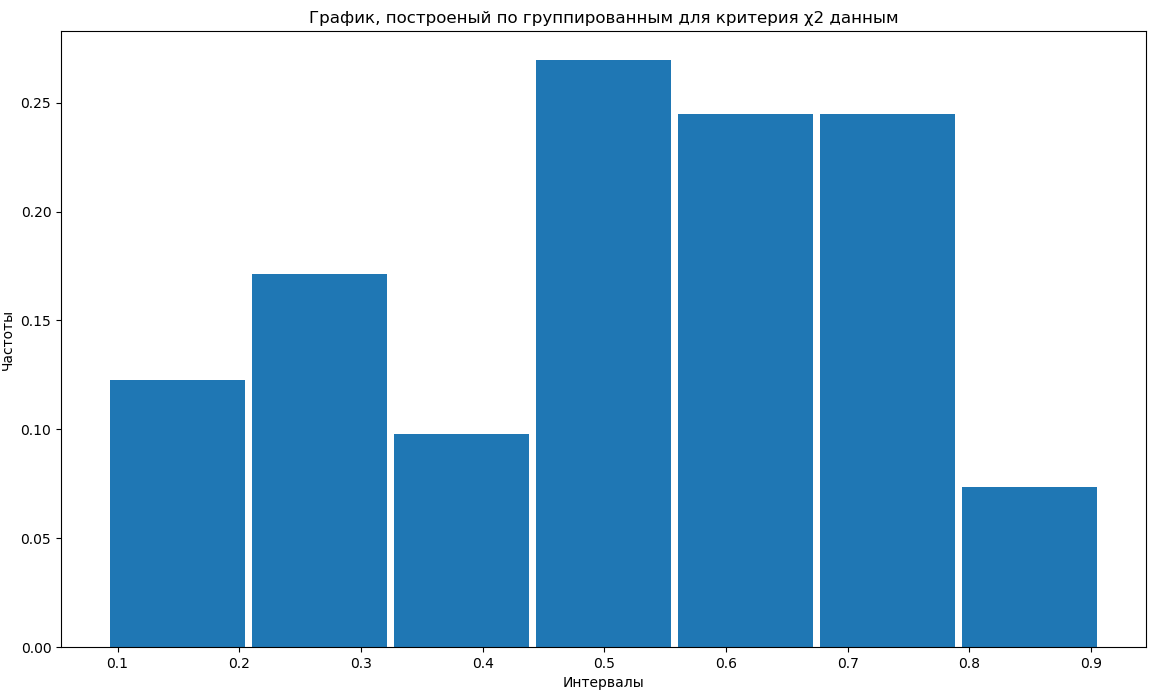
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Длина последовательности** | | 200 | | | | | | | |
| **Параметры распределения** | | μ | v | | μ | v | | μ | v |
| 2 | 2 | | 1.5 | 2.6 | | 4.2 | 1.5 |
| **Последовательность** | | 0.617, 0.608, 0.774, 0.753, 0.75, 0.727, 0.756, 0.488, 0.564, 0.676, 0.329, 0.435, 0.426, 0.742, 0.665, 0.697, 0.121, 0.782, 0.848, 0.24, 0.887, 0.818, 0.158, 0.906, 0.477, 0.621, 0.936, 0.572, 0.769, 0.273, 0.316, 0.1, 0.077, 0.124, 0.545, 0.185, 0.621, 0.452, 0.897, 0.42, 0.702, 0.239, 0.475, 0.762, 0.552, 0.762, 0.411, 0.44, 0.459, 0.185, 0.622, 0.239, 0.676, 0.738, 0.54, 0.683, 0.658, 0.264, 0.095, 0.524, 0.409, 0.282, 0.585, 0.759, 0.359, 0.203, 0.315, 0.566, 0.528, 0.619, 0.317, 0.614, 0.674, 0.581, 0.649, 0.476, 0.48, 0.34, 0.112, 0.717, 0.607, 0.475, 0.665, 0.65, 0.677, 0.259, 0.313, 0.466, 0.208, 0.851, 0.454, 0.522, 0.863, 0.756, 0.771, 0.253, 0.266, 0.224, 0.622, 0.598, 0.681, 0.371, 0.617, 0.572, 0.681, 0.294, 0.203, 0.637, 0.754, 0.359, 0.344, 0.346, 0.244, 0.225, 0.653, 0.335, 0.284, 0.688, 0.884, 0.466, 0.163, 0.425, 0.448, 0.436, 0.791, 0.218, 0.884, 0.443, 0.311, 0.237, 0.449, 0.532, 0.264, 0.751, 0.706, 0.31, 0.641, 0.514, 0.376, 0.68, 0.271, 0.655, 0.614, 0.547, 0.576, 0.174, 0.562, 0.247, 0.371, 0.614, 0.524, 0.461, 0.253, 0.643, 0.336, 0.288, 0.68, 0.752, 0.111, 0.362, 0.7, 0.289, 0.328, 0.535, 0.193, 0.613, 0.099, 0.636, 0.432, 0.25, 0.375, 0.719, 0.42, 0.528, 0.665, 0.42, 0.062, 0.826, 0.706, 0.35, 0.784, 0.663, 0.587, 0.463, 0.591, 0.091, 0.317, 0.268, 0.708, 0.297, 0.406, 0.563, 0.665, 0.696, 0.497, 0.712, 0.66, 0.744, 0.665, 0.526 | | | 0.198, 0.3, 0.321, 0.655, 0.552, 0.858, 0.14, 0.18, 0.168, 0.58, 0.114, 0.661, 0.263, 0.182, 0.657, 0.499, 0.37, 0.703, 0.034, 0.556, 0.442, 0.232, 0.254, 0.807, 0.401, 0.402, 0.235, 0.098, 0.03, 0.667, 0.351, 0.054, 0.725, 0.623, 0.042, 0.463, 0.501, 0.025, 0.115, 0.288, 0.254, 0.526, 0.281, 0.127, 0.429, 0.669, 0.72, 0.492, 0.313, 0.091, 0.711, 0.242, 0.236, 0.153, 0.49, 0.518, 0.714, 0.571, 0.223, 0.156, 0.577, 0.492, 0.717, 0.71, 0.384, 0.404, 0.752, 0.29, 0.643, 0.772, 0.372, 0.285, 0.624, 0.729, 0.067, 0.591, 0.183, 0.324, 0.318, 0.543, 0.033, 0.442, 0.243, 0.102, 0.304, 0.421, 0.264, 0.34, 0.486, 0.169, 0.564, 0.058, 0.26, 0.223, 0.764, 0.5, 0.81, 0.352, 0.869, 0.116, 0.343, 0.303, 0.138, 0.473, 0.367, 0.391, 0.212, 0.568, 0.336, 0.54, 0.222, 0.294, 0.839, 0.765, 0.626, 0.378, 0.354, 0.158, 0.37, 0.314, 0.66, 0.516, 0.559, 0.387, 0.249, 0.315, 0.541, 0.386, 0.086, 0.483, 0.165, 0.215, 0.715, 0.629, 0.311, 0.302, 0.652, 0.315, 0.601, 0.46, 0.57, 0.115, 0.542, 0.694, 0.429, 0.405, 0.236, 0.802, 0.537, 0.621, 0.039, 0.35, 0.809, 0.52, 0.451, 0.428, 0.568, 0.175, 0.156, 0.726, 0.079, 0.493, 0.216, 0.141, 0.139, 0.719, 0.495, 0.312, 0.025, 0.732, 0.211, 0.31, 0.507, 0.333, 0.664, 0.326, 0.131, 0.361, 0.42, 0.338, 0.387, 0.361, 0.428, 0.399, 0.228, 0.114, 0.351, 0.501, 0.594, 0.434, 0.601, 0.29, 0.214, 0.055, 0.191, 0.117, 0.33, 0.058, 0.025, 0.19 | | | 0.418, 0.914, 0.587, 0.821, 0.777, 0.758, 0.701, 0.949, 0.718, 0.746, 0.638, 0.611, 0.82, 0.582, 0.615, 0.966, 0.95, 0.849, 0.57, 0.847, 0.908, 0.784, 0.683, 0.591, 0.92, 0.672, 0.89, 0.879, 0.811, 0.778, 0.57, 0.625, 0.659, 0.967, 0.815, 0.807, 0.641, 0.637, 0.863, 0.723, 0.618, 0.795, 0.906, 0.583, 0.848, 0.863, 0.513, 0.694, 0.807, 0.883, 0.866, 0.824, 0.86, 0.485, 0.642, 0.596, 0.655, 0.986, 0.774, 0.871, 0.747, 0.881, 0.836, 0.691, 0.625, 0.647, 0.851, 0.863, 0.96, 0.813, 0.884, 0.454, 0.491, 0.669, 0.86, 0.905, 0.903, 0.979, 0.894, 0.805, 0.839, 0.636, 0.656, 0.367, 0.597, 0.57, 0.912, 0.45, 0.641, 0.594, 0.83, 0.639, 0.799, 0.614, 0.615, 0.852, 0.772, 0.757, 0.877, 0.83, 0.865, 0.965, 0.841, 0.898, 0.409, 0.964, 0.854, 0.981, 0.868, 0.861, 0.632, 0.833, 0.823, 0.655, 0.417, 0.712, 0.889, 0.939, 0.575, 0.717, 0.518, 0.71, 0.914, 0.849, 0.785, 0.577, 0.934, 0.765, 0.827, 0.827, 0.808, 0.765, 0.82, 0.835, 0.639, 0.33, 0.375, 0.907, 0.86, 0.978, 0.777, 0.911, 0.977, 0.872, 0.687, 0.327, 0.869, 0.944, 0.507, 0.688, 0.542, 0.658, 0.58, 0.866, 0.382, 0.679, 0.65, 0.794, 0.856, 0.92, 0.972, 0.641, 0.665, 0.356, 0.95, 0.738, 0.742, 0.908, 0.879, 0.391, 0.802, 0.986, 0.551, 0.902, 0.927, 0.971, 0.973, 0.892, 0.641, 0.539, 0.906, 0.727, 0.668, 0.782, 0.932, 0.769, 0.707, 0.641, 0.706, 0.673, 0.575, 0.898, 0.891, 0.819, 0.674, 0.916, 0.619, 0.758, 0.885, 0.701 | |
| **Время моделирования** | | 0.167 | | | 3.043 | | | 2.31 | |
| **Количество равномерных чисел, которые потребовались для генерации** | | 124 | | | 180 | | | 328 | |
| **Значение статистики S** | | 18.4 | | | 8.344 | | | 25.7 | |
| **Достигнутый уровень значимости** | | 0.14 | | | 0.82 | | | 0.018 | |
| **Результаты проверки критерия *χ*2** | | Пройден | | | Пройден | | | Не пройден | |
| **Значение статистики S** | | 0.374 | | | 1.44 | | | 0.28 | |
| **Результаты проверки критерия** **Ω2-Андерсона-Дарлинга** | | Пройден | | | Не пройден | | | Пройден | |
| **Длина последовательности** | 1000 | | | | | | | | |
| **Параметры распределения** | μ | | v | μ | | v | μ | | v |
| 2 | | 2 | 1.5 | | 2.6 | 4.2 | | 1.5 |
| **Время моделирования** | 0.3 | | | 4 | | | 3.45 | | |
| **Количество равномерных чисел, которые потребовались для генерации** | 710 | | | 976 | | | 1466 | | |
| **Значение статистики S** | 30.54 | | | 39.52 | | | 36.008 | | |
| **Достигнутый уровень значимости** | 0.438 | | | 0.114 | | | 0.207 | | |
| **Результаты проверки критерия *χ*2** | Пройден | | | Пройден | | | Пройден | | |
| **Значение статистики S** | 0.404 | | | 0.6 | | | 0.634 | | |
| **Результаты проверки критерия** **Ω2-Андерсона-Дарлинга** | Пройден | | | Пройден | | | Пройден | | |

**Графики**

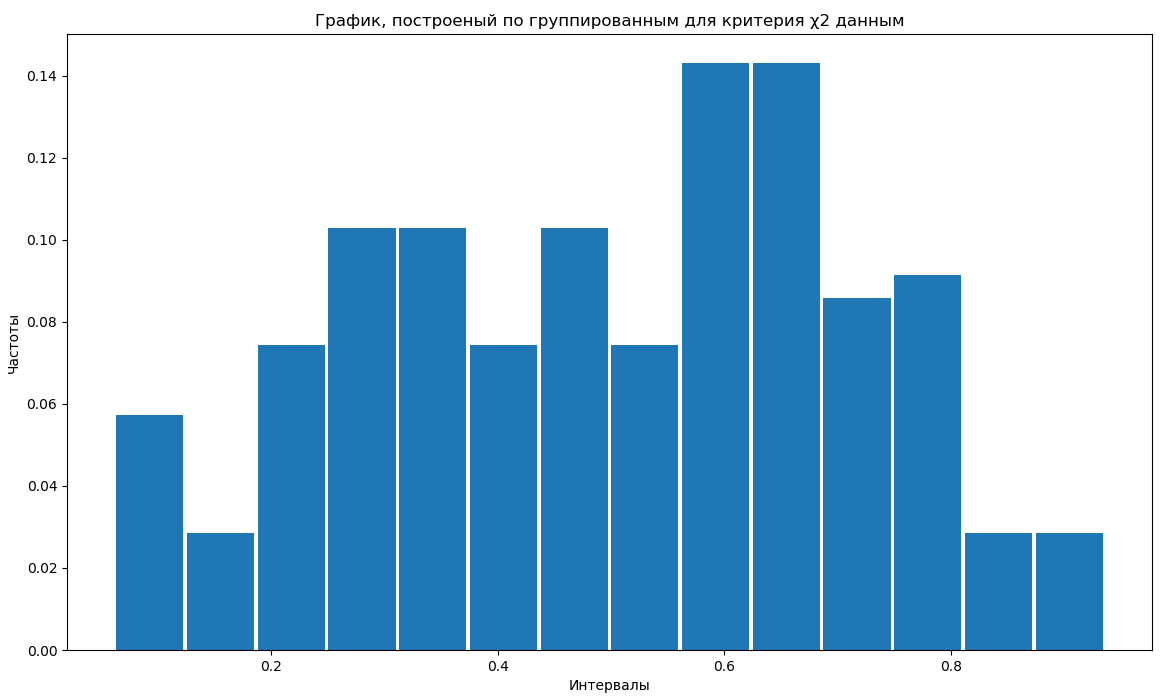
μ = 2 v = 2



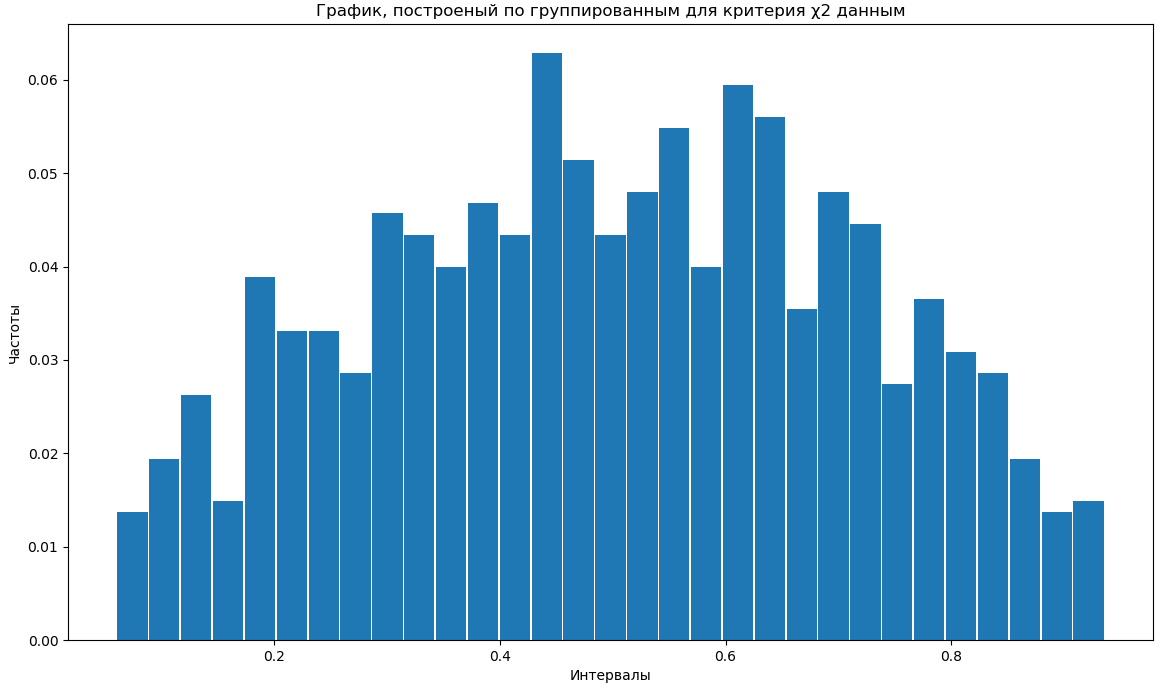
N=50



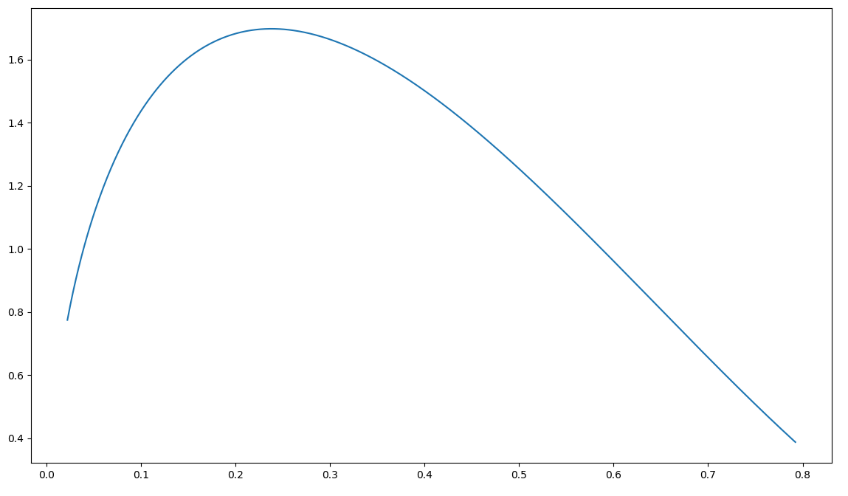
N=200



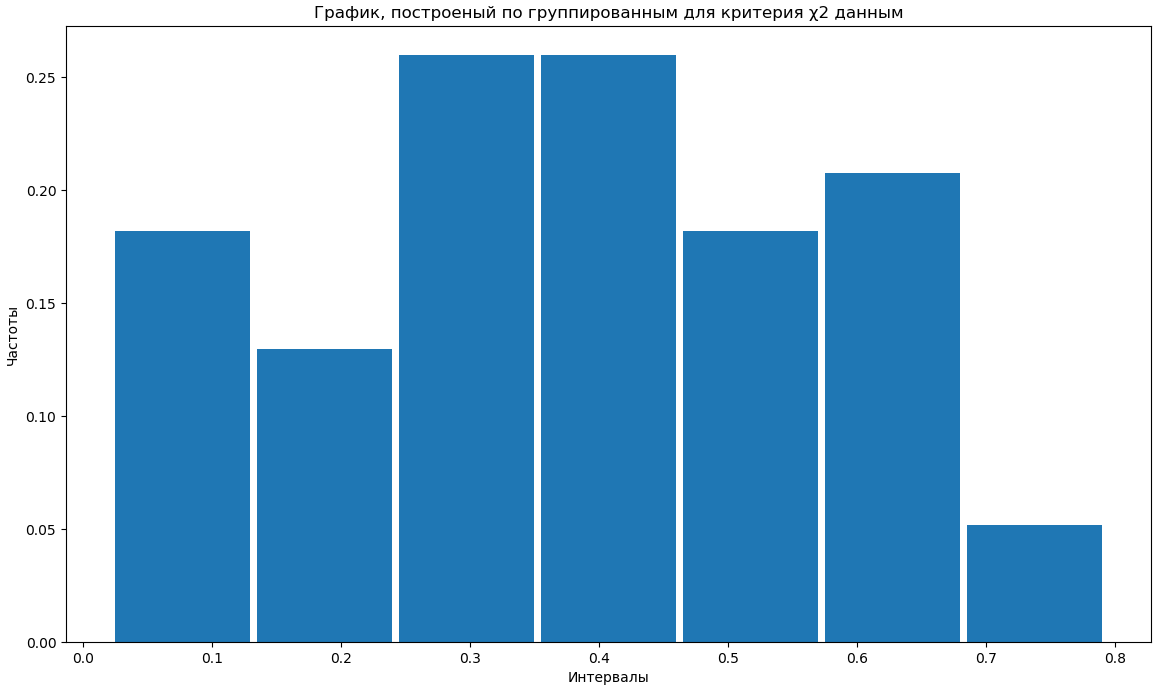
N=1000



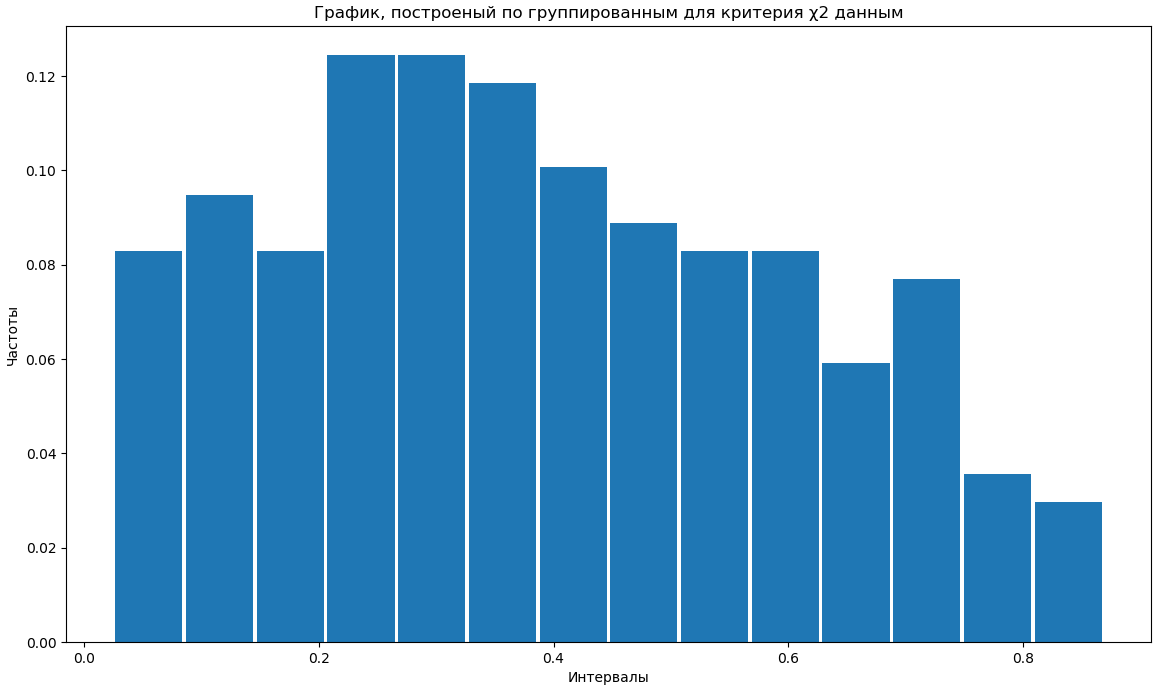
μ = 1.5 v = 2.6



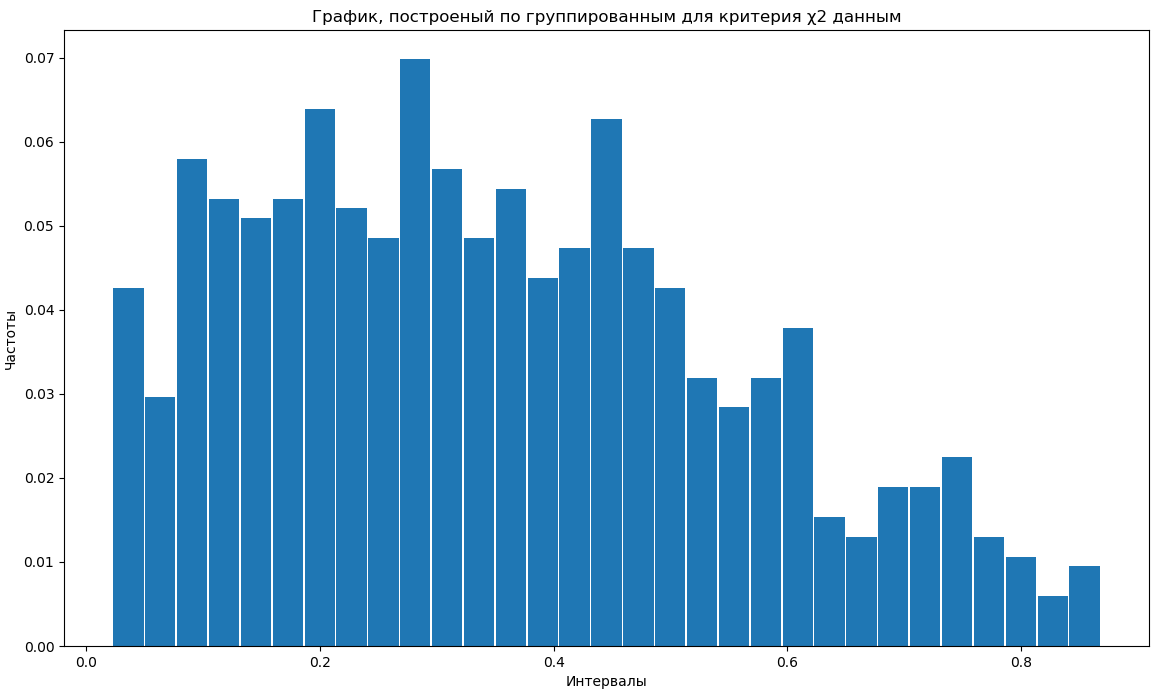
N= 50



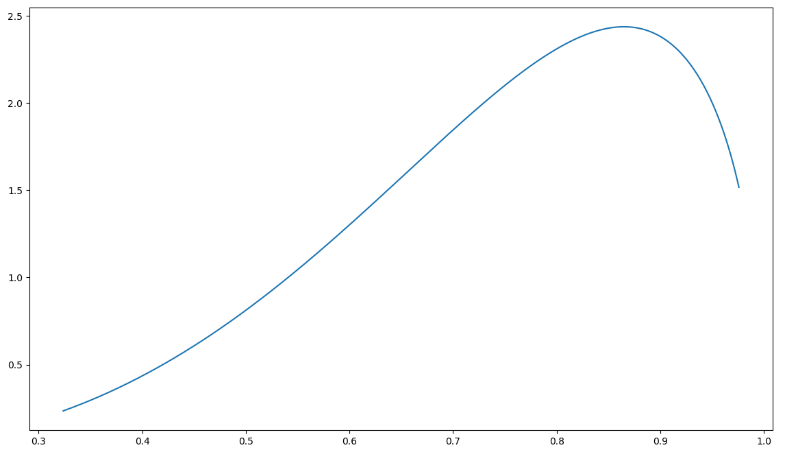
N = 200



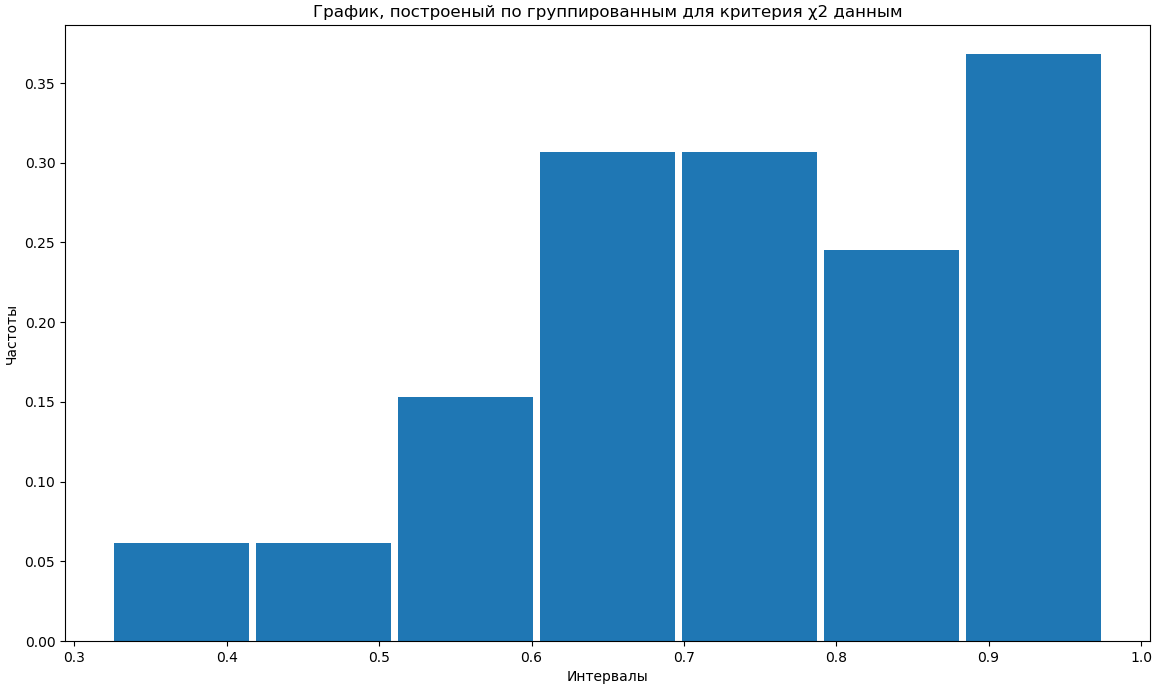
N = 1000



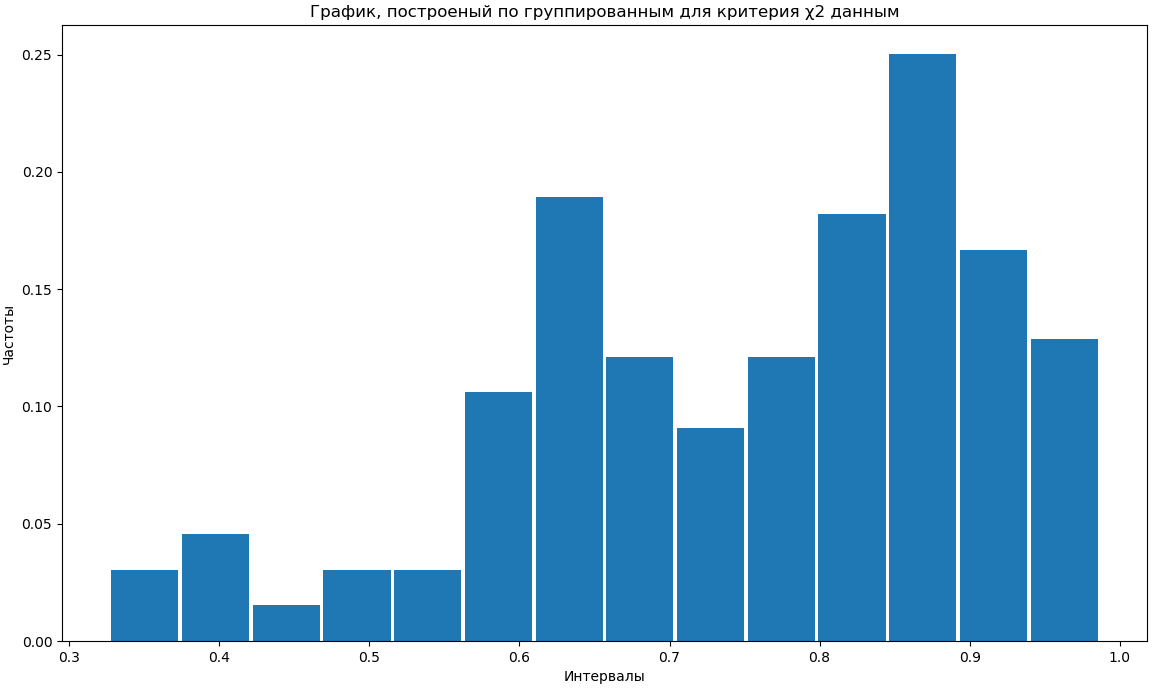
μ = 4.2 v = 1.5



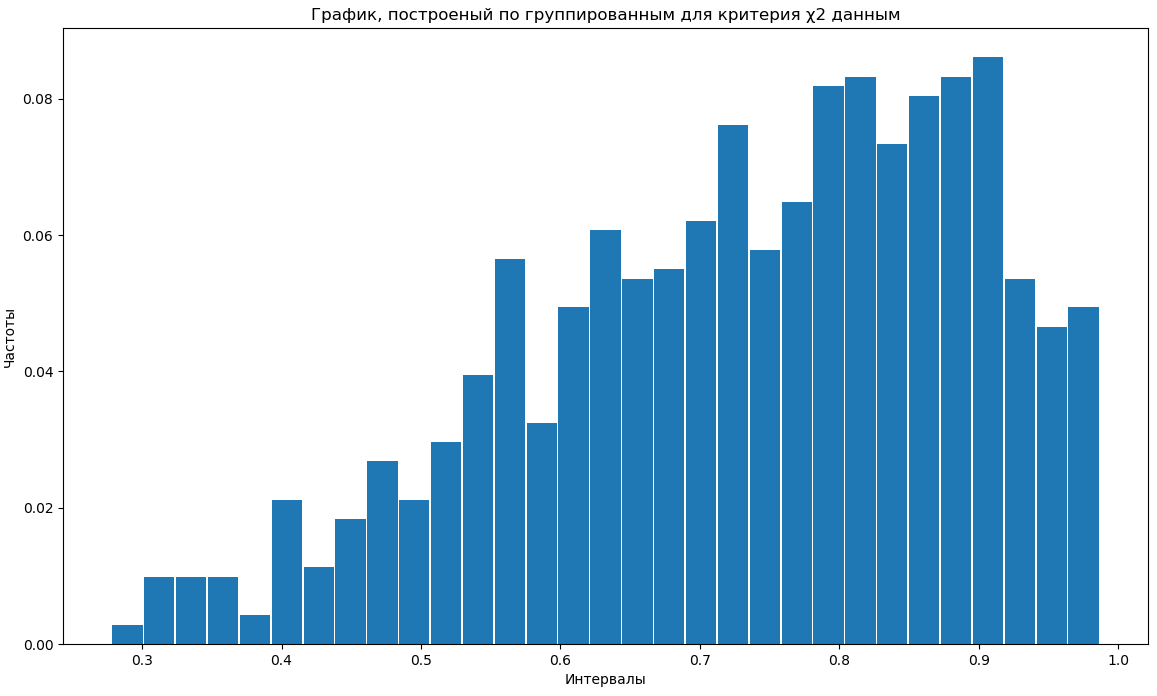
N = 50



N = 200



N = 1000



1. **Код программы**

**import** math **as** mh  
**import** pandas **as** pd  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** pylab **import** \*  
**import** scipy  
**import** scipy.stats  
**from** scipy.stats **import** anderson  
**from** scipy **import** integrate  
**from** scipy.stats **import** gamma  
**import** time  
**from** datetime **import** datetime  
**import** timeit  
**import** random  
  
random.seed(0)  
pd.set\_option(**'display.max\_columns'**, **None**)  
  
  
**def** read\_data(): *# считывание данных из файла* data = []  
 **with** open(**"data.txt"**) **as** f:  
 **for** line **in** f:  
 data.append([float(x) **for** x **in** line.split()])  
 **return** data  
  
  
**def** Bess(miu, v):*#функция Бесселя* func = **lambda** x: np.power(x, miu - 1) \* np.power(1 - x, v - 1)  
 P = integrate.quad(func, 0, 1)  
 **return** P[0]  
  
  
**def** beta(miu, v, x):*#функция плотности бета распределения* f = 0  
 **if** x >= 0 **and** x <= 1:  
 f = mh.pow(x, miu - 1) \* mh.pow(1 - x, v - 1) / Bess(miu, v)  
 **return** f  
  
  
**def** dens\_beta(miu, v, x):*#функция бета распределения* f = 0  
 func = **lambda** y: np.power(y, miu - 1) \* np.power(1 - y, v - 1) / Bess(miu, v)  
 f = integrate.quad(func, 0, x)  
 **return** f[0]  
  
  
**def** intervals(miu, v):*#функция вычисления границ, в которых содержиться* a = 0  
 b = 0  
 f = **lambda** x: np.power(x, miu - 1) \* np.power(1 - x, v - 1) / Bess(miu, v)*#функция плотности распределения* i = 0  
 **while** 1:*#поиск левой границы* buf = integrate.quad(f, 0, i)*#высичляем интегла до тех пор, пока он не достигнет 0.01* **if** buf[0] >= 0.01:  
 a = i  
 **break  
 else**:  
 i += 0.001  
 i = 0  
 **while** 1:*#поиск правой границы* buf = integrate.quad(f, 0, i)*#высичляем интегла до тех пор, пока он не достигнет 0.99* **if** buf[0] >= 0.99:  
 b = i  
 **break  
 else**:  
 i += 0.001  
 **return** a, b  
  
**def** f\_max(a, b, miu, v):*#фнукция поиска значения x при котором функция плотности максимальна* Mx = 0  
 I = []  
 y = []  
 x = np.arange(a, b, 0.001)  
 **for** i **in** x:  
 y.append(beta(miu, v, i))  
 I.append(i)  
 m = max(y)  
 **for** i **in** range(len(y)):  
 **if** y[i] == m:  
 Mx = I[i]  
 **break  
 return** Mx  
  
**def** cont\_seq(left, right, miu, v, N):*# функция генератор псевдослучайной последовательности* seq = []  
 MAX = f\_max(left, right, miu, v) *#ищем x при котором функцяи плотности максимальна на отрезке [a, b]* M = beta(miu, v, MAX) *#ищем коэффициент М, который используется своместно с y0* j = 0 *#число равномерных псевдослучайных чисел, которые потребовались для генерации посдедовательности* **for** i **in** range(N):  
 x = []  
 **while** 1:  
 **for** k **in** range(2):  
 x\_n = random.random()  
 x.append(x\_n)  
 x0 = left + x[0] \* (right - left)  
 y0 = x[1] \* M  
 **if** y0 < beta(miu, v, x0):  
 seq.append(x0)  
 **break** j += 2  
 x.clear()*#всегда очищаем пару равномерно распределённхы чисел* **return** seq, j  
  
  
**def** P\_crit(Sk, r): *# Вычисление значения статистики* coef = 1 / (np.power(2, (r / 2)) \* mh.gamma(r / 2))  
 func = **lambda** S: np.power(S, (r / 2 - 1)) \* (np.power(mh.e, (- S / 2)))  
 P = integrate.quad(func, Sk, np.inf)  
 **return** P[0] \* coef  
  
  
**def** hi\_2(x, alpha, V, miu):  
 print(**'Проверка критерия χ2:\n'**)  
 Max = np.max(x) *# максимально возможный элемент в последовательности длиной N* Min = np.min(x) *# минимально возможный элемент в последовательности длиной N* h = Max - Min *# длина отрезка интервалов* sum = 0  
 v = []  
 interval = [] *# список граничных точек интервала  
  
 # определяем число интервалов K* K = int(mh.sqrt(len(x)))  
  
 *# заполняем список нулями* **for** i **in** range(K + 1):  
 interval.append(0)  
 P = []  
 *# заполняем список граничными точками* u = Min  
 **for** i **in** range(K + 1):  
 interval[i] = u  
 u = interval[i] + h / K  
 *# заполняем список частот нулями* **for** i **in** range(K):  
 v.append(0)  
  
 **for** i **in** range(1, len(interval)):  
 buf = dens\_beta(miu, V, interval[i]) - dens\_beta(miu, V, interval[i - 1])  
 P.append(buf)  
 *# подсчет частоты попаданий элементов выборки в интервалы* **for** i **in** range(len(x)):  
 **for** j **in** range(K):  
 **if** x[i] >= interval[j] **and** x[i] <= interval[j + 1]:  
 v[j] += 1  
  
 print(**"V = "**, interval)  
 print(**"P"**, P)  
 **for** i **in** range(len(v)):  
 v[i] /= len(x) *# отностельные частоты элементов* print(**"V = "**, v)  
 vt = []*# построение нормированной гистограммы* **for** i **in** v:  
 vt.append(i / (h / K)) *#делим относительные частоты  
 # график, построенный по группированным для критерия χ2 данным* gridsize = (1, 1)  
 fig = plt.figure(figsize=(14, 8))  
 ax1 = fig.add\_subplot()  
 plt.xlabel(**'Интервалы'**)  
 plt.ylabel(**'Частоты'**)  
 ax1.hist(interval[:-1], bins=interval, weights=vt, rwidth=0.95)  
 ax1.set\_title(**'График, построеный по группированным для критерия χ2 данным'**)  
 plt.show()  
 *# вычисляем сумму в формуле S(hi2)* **for** i **in** range(K):  
 sum += mh.pow((v[i] - P[i]), 2) / P[i]  
 S = len(x) \* sum *# умножаем сумму на объем выборки n* r = K - 1 *# число степеней свободы* Sk = scipy.stats.chi2.ppf(1 - alpha, r)  
 print(S, r)  
 S\_star = P\_crit(S, r)  
 **if** S\_star > alpha:  
 print(**'P {Sχ2 > Sχ2\*}(достигнутый уровень значимости) = '**, S\_star)  
 print(**'Проверка критерия χ2 окончена.\n'**)  
 **return** 1, S, S\_star *# гипотеза отвергается* **else**:  
 print(**'P {Sχ2 > Sχ2\*}(достигнутый уровень значимости) = '**, S\_star)  
 print(**'Проверка критерия χ2 окончена.\n'**)  
 **return** 0, S, S\_star *# гипотеза не отвергается*data = read\_data()*#считывание данных из файла*alpha = data[9][0]*#выделение уровня значимости*N = []*#Список длин последовательностей*MIU = []*#Список значений параметра мю*V = []*#Список значений параметра ню*LT = []*#Список значений левой границы*RT = []*#Список значений правой границы***for** i **in** range(0, 3):  
 N.append(int(data[i][0]))  
  
**for** i **in** range(3, 6):  
 MIU.append(data[i][0])  
  
**for** i **in** range(6, 9):  
 V.append(data[i][0])  
  
**for** i **in** range(len(V)):*#вычисление левых и правых границ для всех вариантов входых данных* l, r = intervals(MIU[i], V[i])  
 LT.append(l)  
 RT.append(r)  
  
**for** i **in** range(len(N)):  
 **for** j **in** range(len(V)):  
 fl = 0 *#флаг, обозначающий выполнение или не выполнение теста Андерсона-Дарлинга* start = timeit.default\_timer()*#таймер для подсчёта времени генерации* S, Y = cont\_seq(LT[j], RT[j], MIU[j], V[j], N[i])  
 end = timeit.default\_timer() - start  
 fig = plt.figure(figsize=(14, 8))  
 ax0 = fig.add\_subplot()  
 y2 = []  
 print(**"Количество псевдослучайных чисел, потребовавшихся для генерации последовательности"**, Y)  
 x2 = np.arange(min(S), max(S), 0.001)  
 **for** o **in** x2:  
 y2.append(beta(MIU[j], V[j], o))  
 ax0.plot(x2, y2) *# График* plt.show()  
 print(**"Для моделирования выборки с параметрами: u ="**, MIU[j], **" v ="**, V[j], **" потребовалось "**, end, **" секунд"**)  
 h, s, St = hi\_2(S, alpha, V[j], MIU[j])  
 result = anderson(S)  
 print(result)  
 **for** r **in** range(len(result.critical\_values)):  
 sl, cv = result.significance\_level[r], result.critical\_values[r]  
 **if** result.statistic < result.critical\_values[r]:  
 fl = 1  
 **if** i == 0:  
 **if** j == 0:  
 f = open(**"out\_seq\_50\_1.txt"**, **"w"**)  
 **if** j == 1:  
 f = open(**"out\_seq\_50\_2.txt"**, **"w"**)  
 **if** j == 2:  
 f = open(**"out\_seq\_50\_3.txt"**, **"w"**)  
 **if** i == 1:  
 **if** j == 0:  
 f = open(**"out\_seq\_200\_1.txt"**, **"w"**)  
 **if** j == 1:  
 f = open(**"out\_seq\_200\_2.txt"**, **"w"**)  
 **if** j == 2:  
 f = open(**"out\_seq\_200\_3.txt"**, **"w"**)  
 **if** i == 2:  
 **if** j == 0:  
 f = open(**"out\_seq\_1000\_1.txt"**, **"w"**)  
 **if** j == 1:  
 f = open(**"out\_seq\_1000\_2.txt"**, **"w"**)  
 **if** j == 2:  
 f = open(**"out\_seq\_1000\_3.txt"**, **"w"**)  
 p = **'Последовательность: '** f.write(p)  
 **for** u **in** S:  
 p = str(round(u, 3)) + **', '** f.write(p)  
 p = **'\nВремея моделирования: '** + str(end) + **' секунд\n'** f.write(p)  
 p = **'Параметры последовательности: \n'** p = **'n = '** + str(N[i]) + **'miu = '** + str(MIU[j]) + **'v = '** + str(V[j]) + **'\n'** f.write(p)  
  
 p = **'Значение статистики S: '** + str(result.statistic ) + **'\n'** f.write(p)  
 **if** fl:  
 p = **'Критери омега^2-Андерсона-Дарлинга пройден, следовательно не причин для отклонения гипотезы о согласии распределения сгенерированной последовательности с бета распределением'** + **'\n'** f.write(p)  
 **else**:  
 p = **'Критери омега^2-Андерсона-Дарлинга не пройден, следовательно гипотеза о согласии распределения сгенерированной последовательности с бета распределением отвергается'** + **'\n'** f.write(p)  
   
 p = **'Значение статистики S: '** + str(s) + **'\n'** f.write(p)  
 p = **'Достигнутый уровень значимости: '** + str(St) + **'\n'** f.write(p)  
 **if** h:  
 p = **'Критери Xi^2 пройден, следовательно не причин для отклонения гипотезы о согласии распределения сгенерированной последовательности с бета распределением'** + **'\n'** f.write(p)  
 **else**:  
 p = **'Критери Xi^2 не пройден, следовательно гипотеза о согласии распределения сгенерированной последовательности с бета распределением отвергается'** + **'\n'** f.write(p)  
 f.close()

1. **Выводы**

В ходе выполненной работы было разработано программное средство для генерации непрерывных псевдослучайных чисел методом исключений. Так же был проведён статистический анализ сгенерированных последовательностей путём проверки критериев Ω2-Андерсона-Дарлинга и *χ*2.