МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

ТЕОРИЯ ПСЕВДОСЛУЧАЙНЫХ ГЕНЕРАТОРОВ

ЛАБОРАТОРНАЯ РАБОТА

студента 4 курса 431 группы
специальности 10.05.01 — Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Гущина Андрея Юрьевича

Проверил
доцент _____ И. И. Слеповичев

Саратов 2023

СОДЕРЖАНИЕ

1 Постановка задачи

1.1 Цель

1. Сгенерировать псевдослучайную последовательность заданным методом.
2. Исследовать полученную псевдослучайную последовательность на случайность.

1.2 Исходные данные

Исходными данными для лабораторных занятий являются метод генерации псевдослучайных чисел, диапазон генерации случайных чисел, функция распределения, которой должны подчиняться случайные числа, количество генерируемых чисел.

В данной работе используются ППСЧ, сгенерированные с помощью программы, разработанной в практическом задании №1. Для генераторов указаны следующие параметры (эти параметры также можно найти в файле `lab/generate.sh`):

— `/g:lc /f:rnd-lc.dat /i:31104,625,6571,23`

— `/g:add /f:rnd-add.dat /i:30000,24,55,79,134,213,`
`347,560,907,1467,2374,3841,6215,10056,16271,26327,`
`12598,8925,21523,448,21971,22419,14390,6809,21199,`
`28008,19207,17215,6422,23637,59,23696,23755,17451,`
`11206,28657,9863,8520,18383,26903,15286,12189,27475,`
`9664,7139,16803,23942,10745,4687,15432,20119,5551,`
`25670,1221,26891,28112,23779,17506`

— `/g:5p /f:rnd-5p.dat /i:19,7,13,17,19,`
`1,0,1,0,1,1,0,0,1,0,1,0,1,1,0,0,1,1,1`

— `/g:lfsr /f:rnd-lfsr.dat /i:0,1,1,0,1,0,1,`
`0,1,1,0,1,0,0,1,0,1,1,0,1,0,1,1,0,1,0,0,1,0,1,0,0,1,0,1,0`

— `/g:nfsr /f:rnd-nfsr.dat /i:0,1,1,0,1,0,1,0,`
`1,1,0,1,0,0,1,0,1,1,0,1,0,1,1,0,1,0,0,1,0,1,0,0,1,0,1,0,`
`19,2144512,412454,3124551`

— `/g:mt /f:rnd-mt.dat /i:1024,1234`

— `/g:rc4 /f:rnd-rc4.dat /i:1,2,3,4,5,6,7,8,`
`1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,`
`1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,`
`1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,`

```
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,
1,2,3,4,5,6,7,8
```

— /g:rsa /f:rnd-rsa.dat /i:39203,1024,17,14523

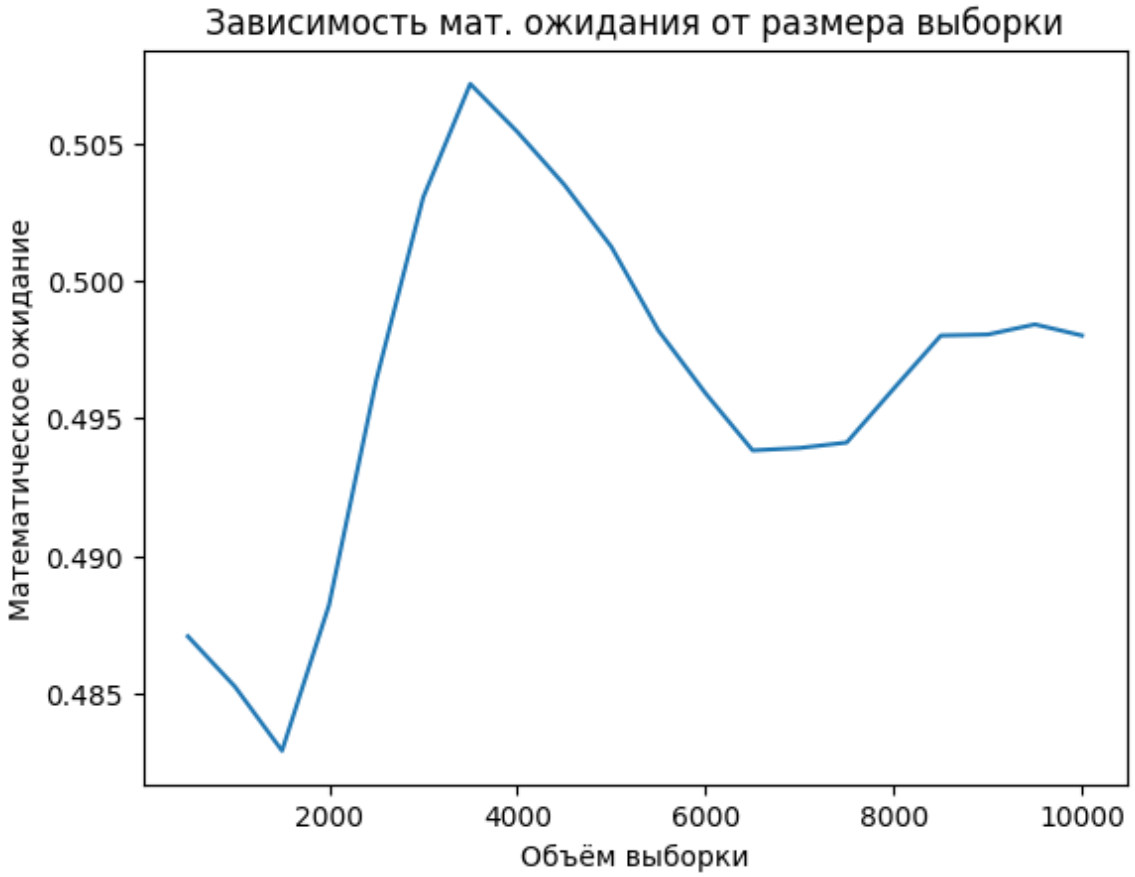
— /g:bbs /f:rnd-bbs.dat /i:312

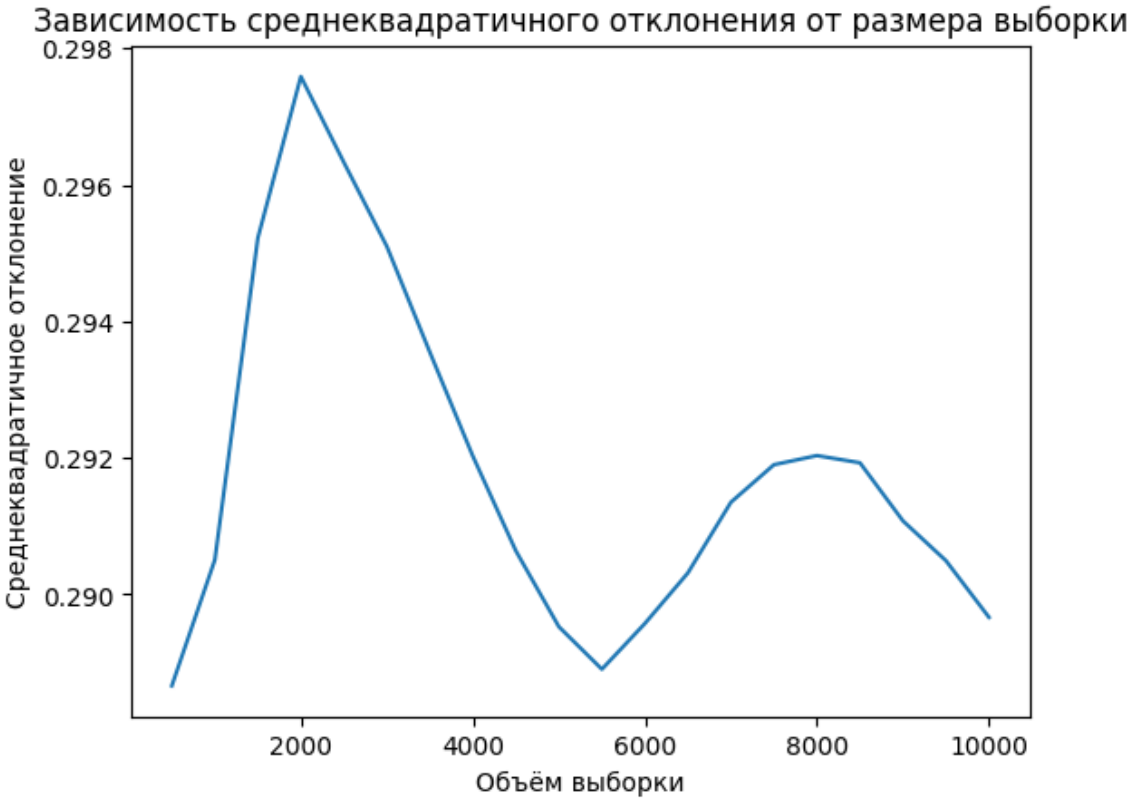
2 Статистические свойства последовательности псевдослучайных чисел

2.1 Вычисление оценок

2.1.1 Линейный конгруэнтный метод

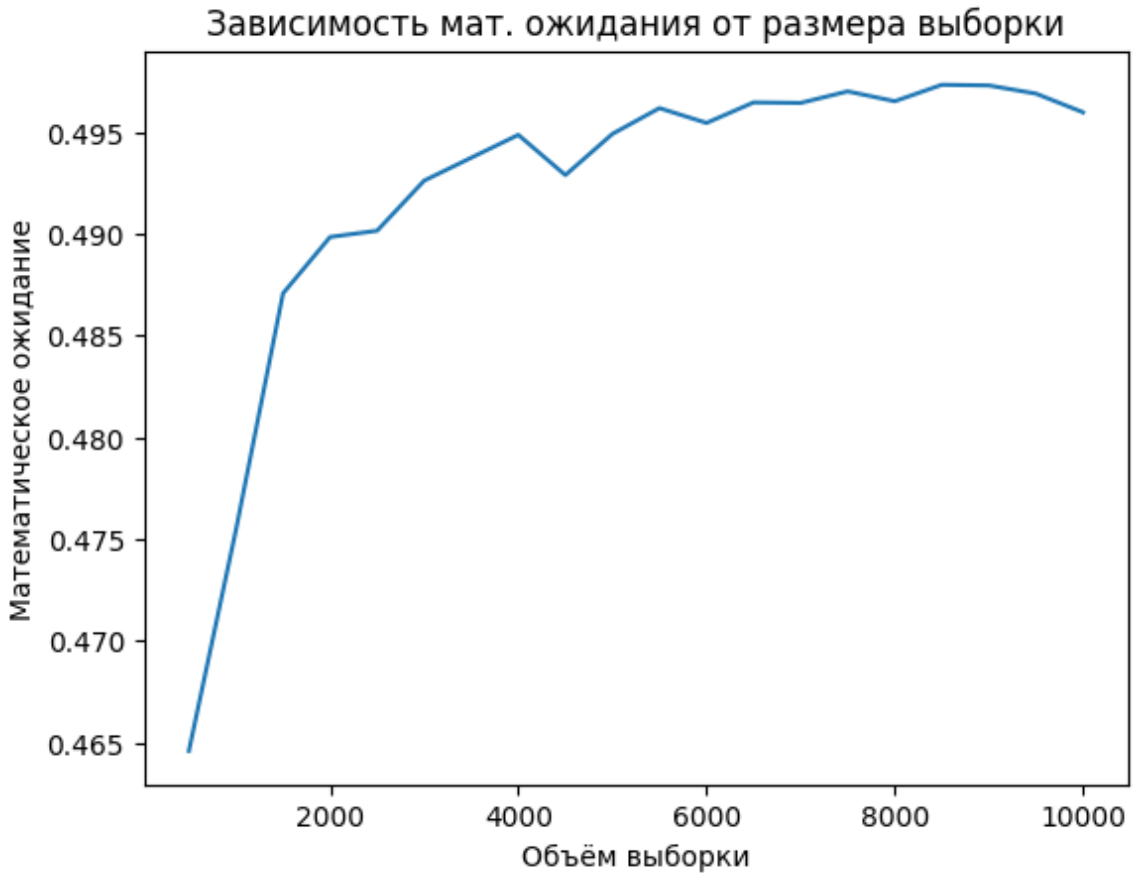
— Математическое ожидание = 0.4980023437500002
— Среднеквадратичное отклонение = 0.28966520307281585
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.642%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.048%

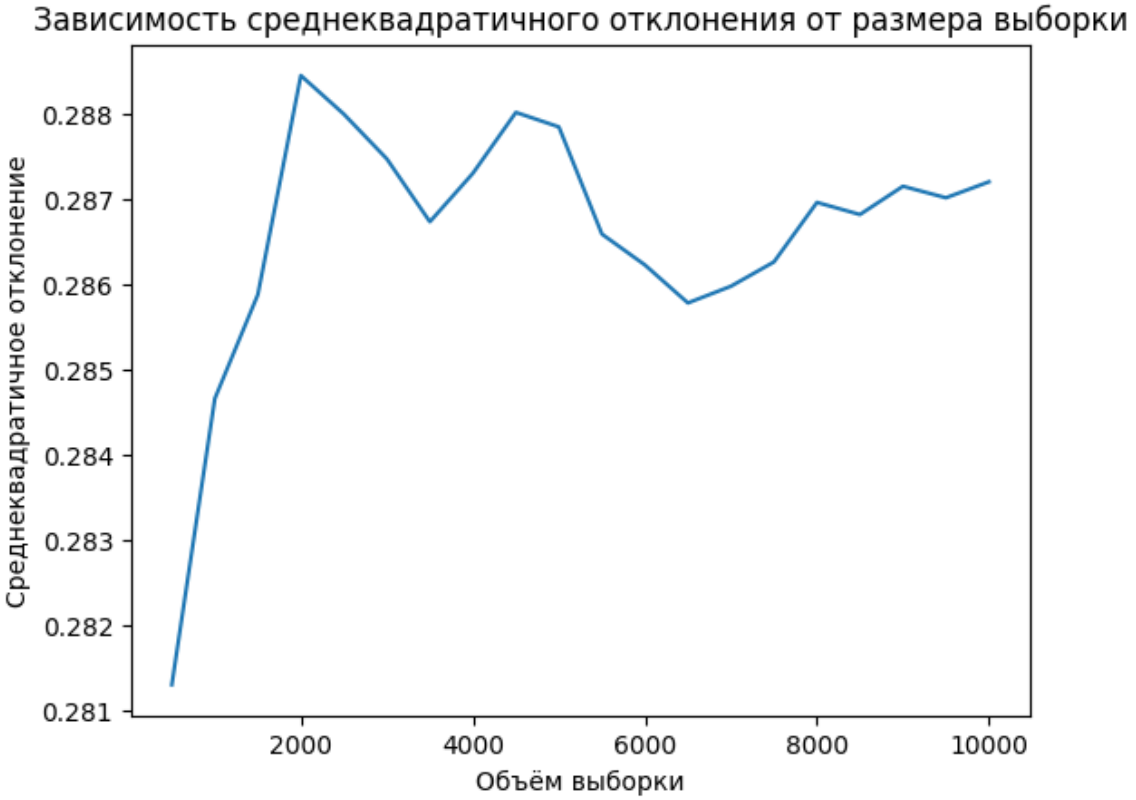




2.1.2 Аддитивный метод

— Математическое ожидание = 0.4959591796874994
— Среднеквадратичное отклонение = 0.2872104606725695
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.215%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.224%

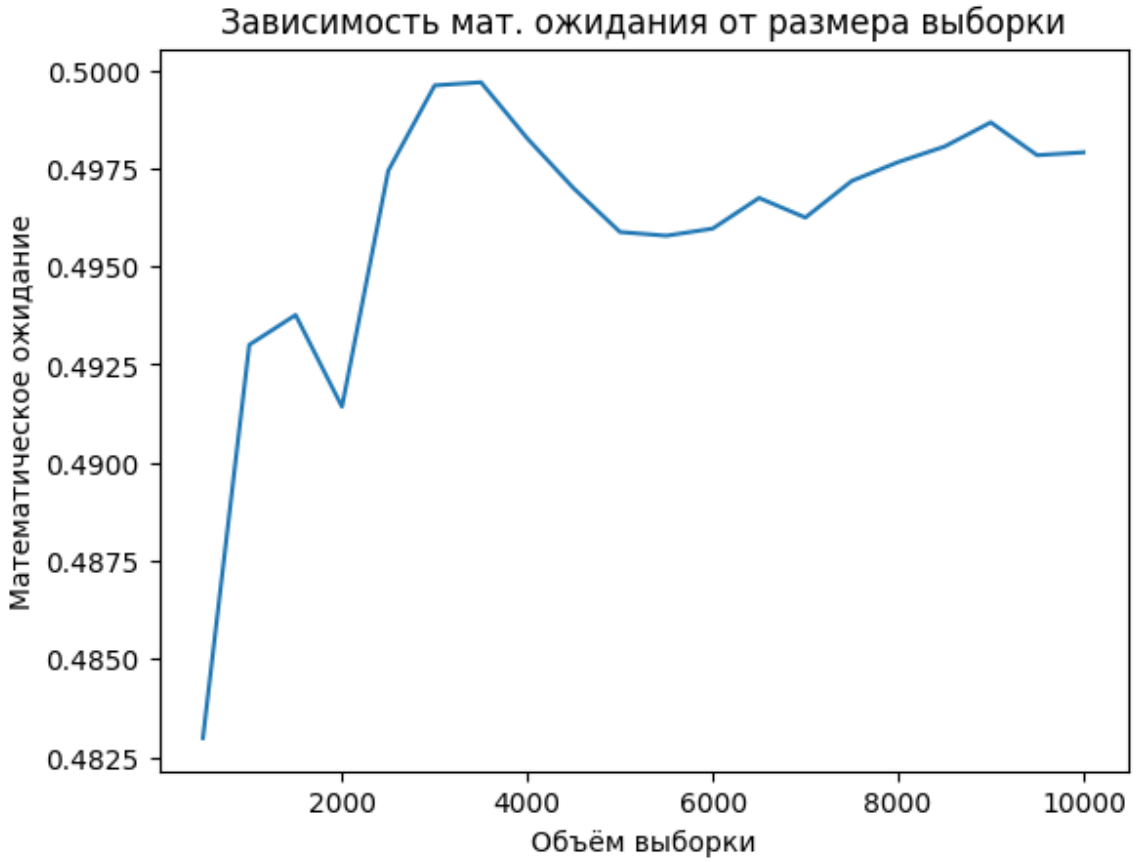


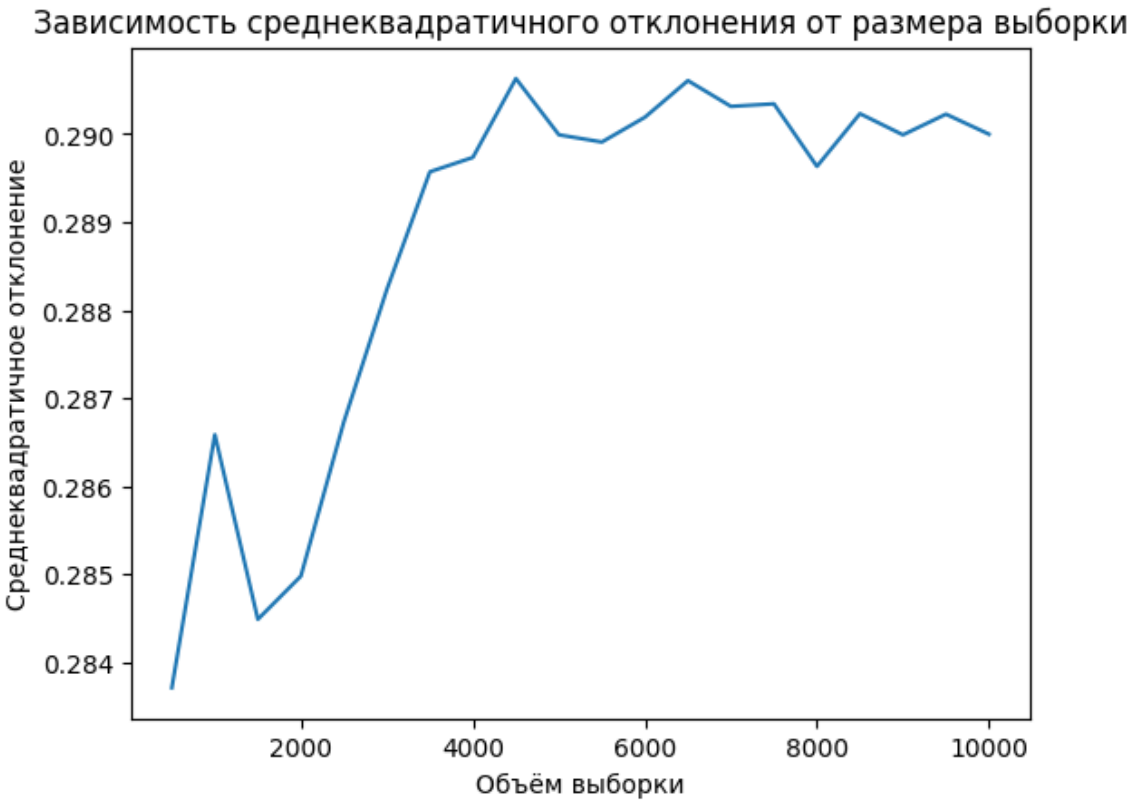


2.1.3 Пятипараметрический метод

— Математическое ожидание = 0.49790820312500017

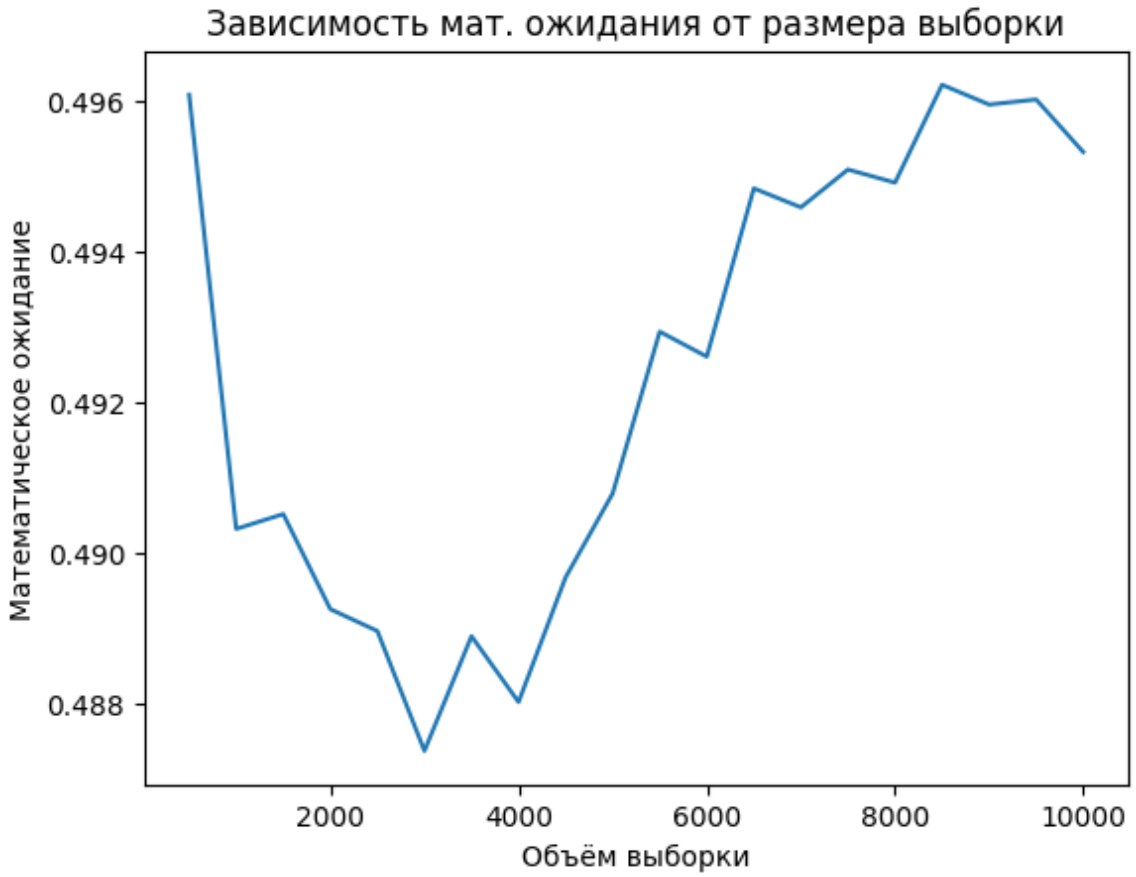
— Среднеквадратичное отклонение = 0.2899989510155476
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.409%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.003%

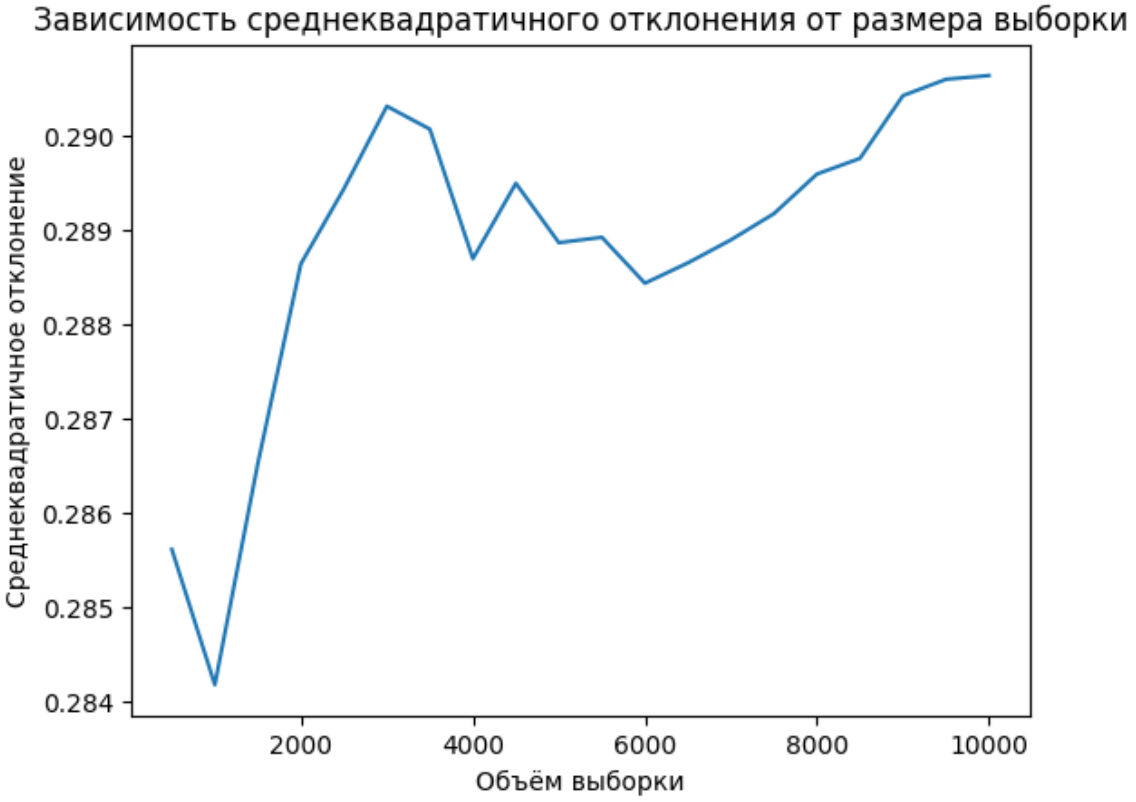




2.1.4 Регистр сдвига с обратной связью (РСЛОС)

— Математическое ожидание = 0.49532001953125054
— Среднеквадратичное отклонение = 0.29063739013628703
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.923%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.613%





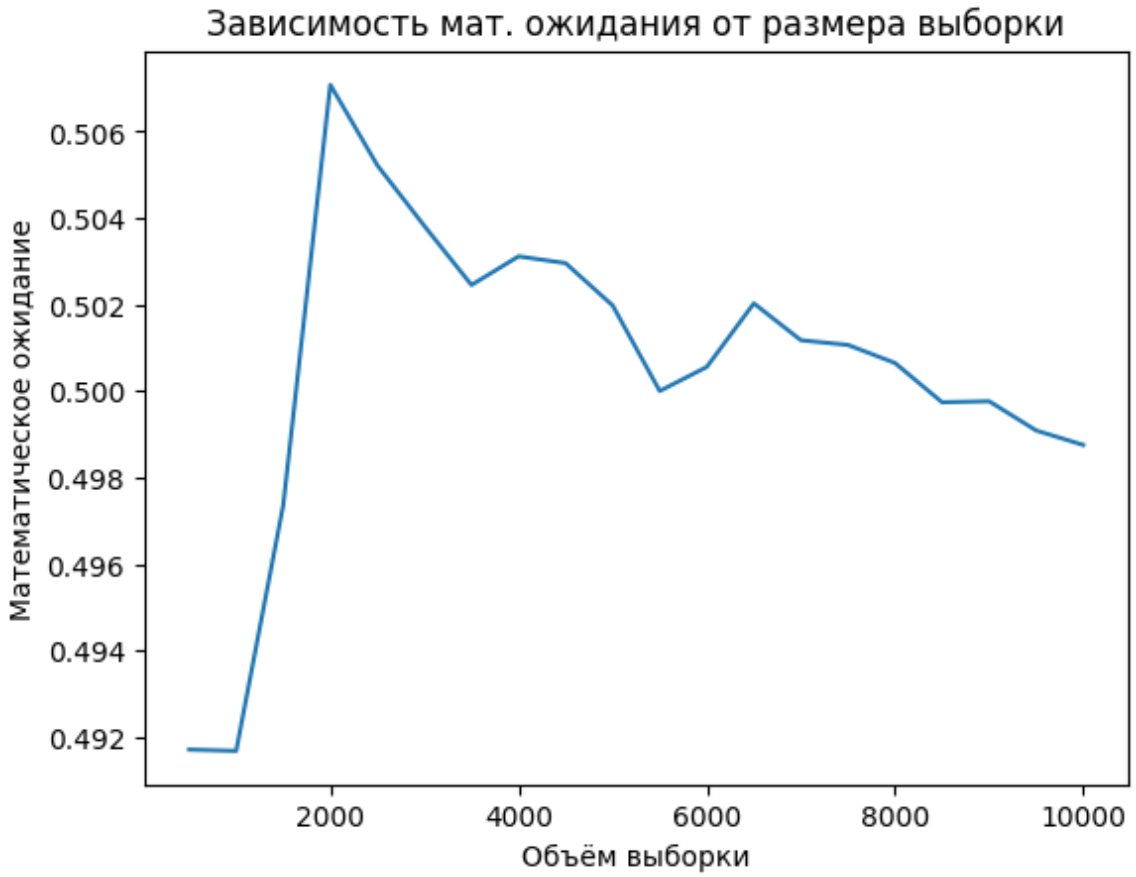
2.1.5 Нелинейная комбинация РСЛОС

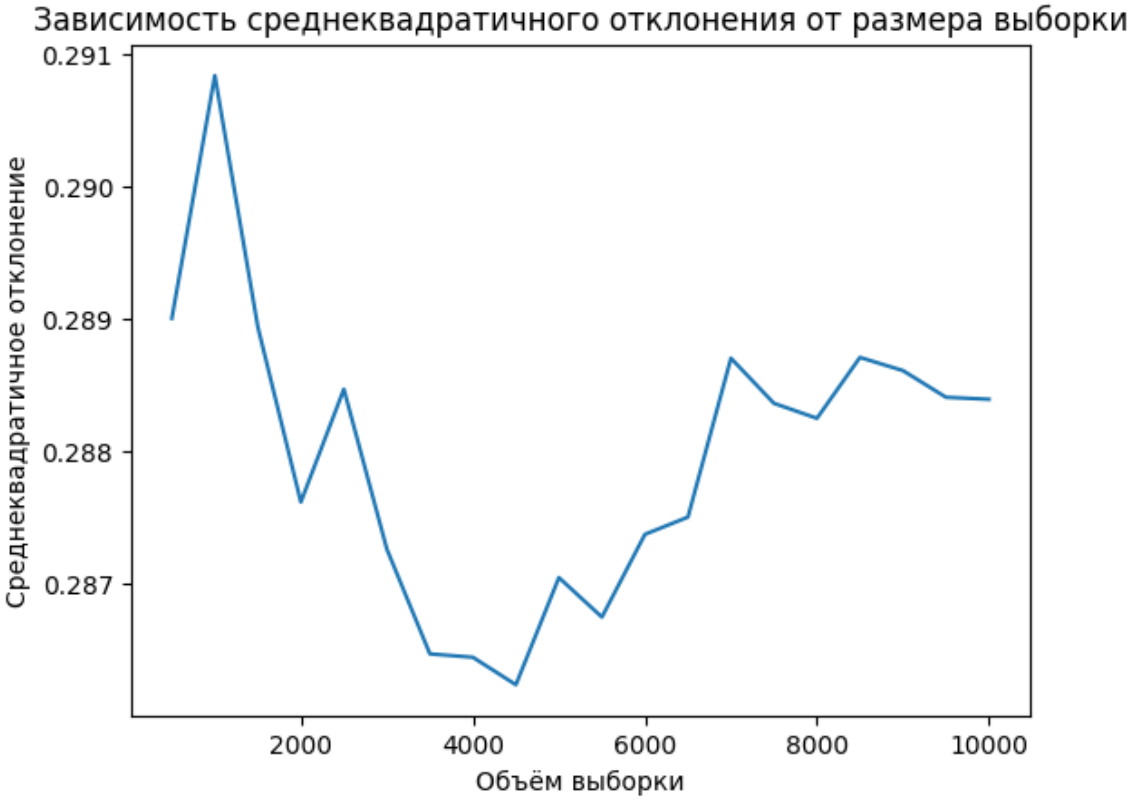
— Математическое ожидание = 0.5033218749999994

— Среднеквадратичное отклонение = 0.29075993794219074
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.134%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.497%

2.1.6 Вихрь Мерсенна

— Математическое ожидание = 0.4987525390624997
— Среднеквадратичное отклонение = 0.2883897385970719
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.643%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.469%





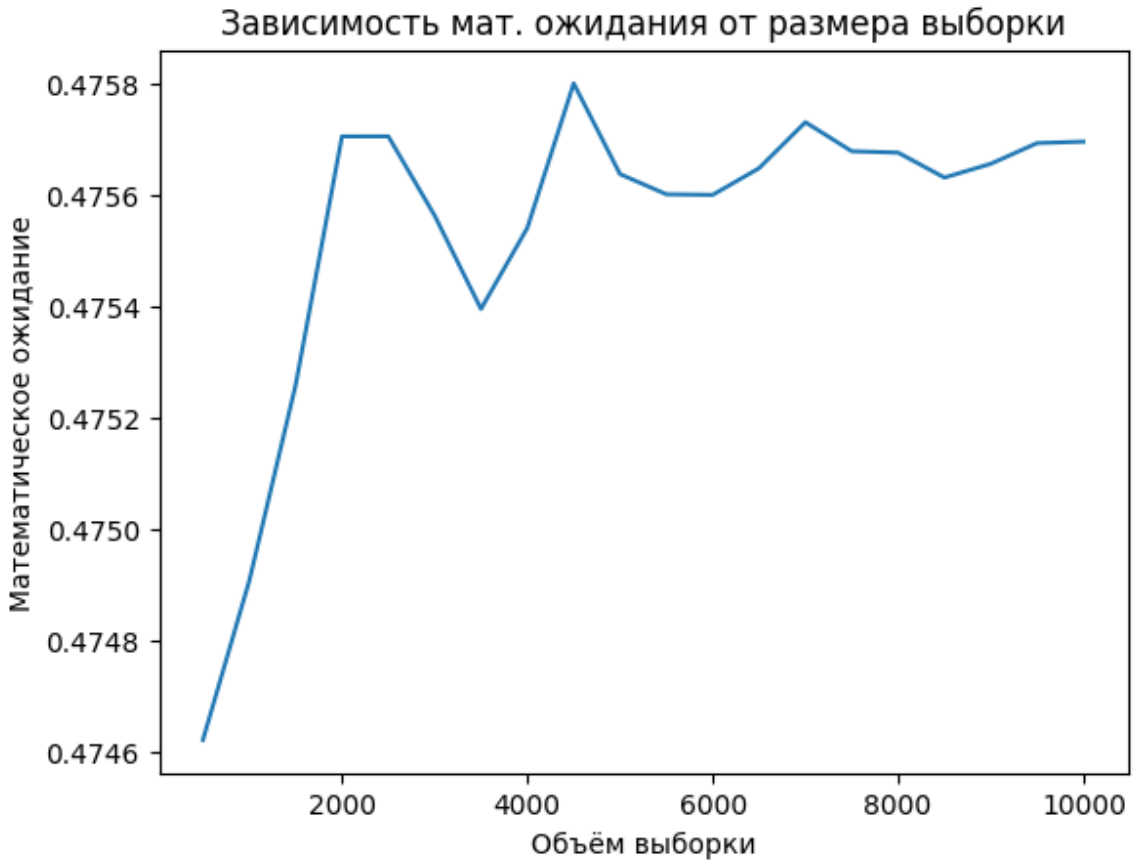
2.1.7 RC4

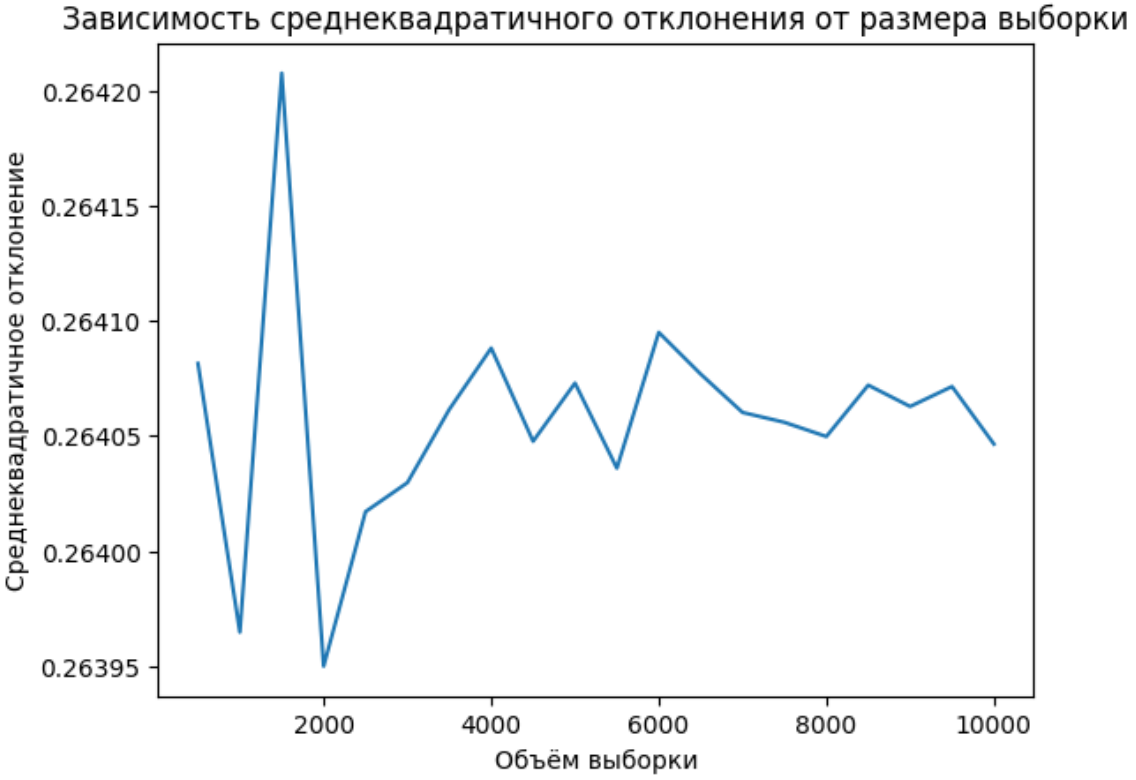
— Математическое ожидание = 0.12457070312500011

— Среднеквадратичное отклонение = 0.07160578956319438
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.29%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.003%

2.1.8 ГПСЧ на основе RSA

— Математическое ожидание = 0.47569716796875
— Среднеквадратичное отклонение = 0.26404647855613683
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.012%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.01%





2.1.9 Алгоритм Блюма-Блюма-Шуба

— Математическое ожидание = 0.005859765624999999

— Среднеквадратичное отклонение = 0.0036539883705292357
— Относительная погрешность измерения математического ожидания для выборки из 5000 элементов: 0.01%
— Относительная погрешность измерения среднеквадратичного отклонения для выборки из 5000 элементов: 0.009%

2.2 Проверка критериев

...	χ-квадрат	Серий	Интервалов	Разбиений	Перестановок	Монотонности	Конфликтов
Линейный конгруэнтный	+	-	+	+	-	+	+
Аддитивный	+	+	+	+	+	+	+
Пятипараметрический	+	+	+	+	+	+	+
РСЛОС	+	+	+	+	+	-	+
Нелинейная комб-я РСЛОС	+	-	+	-	-	-	+
Вихрь Мерсенна	+	+	+	+	+	+	+
RC4	-	-	-	-	+	+	+
RSA	-	-	+	-	-	-	-
BBS	-	-	-	-	-	-	-

ПРИЛОЖЕНИЕ А

Файл labwork.py

```
1 #!/usr/bin/env python
2 # coding: utf-8
```

```
 3
 4  # In[1]:
 5
 6
 7  import seaborn as sns
 8  import pandas as pd
 9  import numpy as np
10  import matplotlib.pyplot as plt
11  from collections import Counter
12  from scipy.stats import chi2
13  from scipy.special import binom
14  from math import sqrt, floor, factorial
15
16
17  # # Статистические свойства
18
19  # In[2]:
20
21
22  def probabilities(u):
23      return np.array([(val, cnt / len(u)) for val, cnt in Counter(u).items()])
24
25  def math_exp(probs):
26      return np.dot(probs[:, 0], probs[:, 1])
27
28  def dispersion(probs):
29      squared = np.array([(x * x, p) for x, p in probs])
30      return math_exp(squared) - math_exp(probs) ** 2
31
32  def std_dev(probs):
33      return sqrt(dispersion(probs))
34
35
36  # In[3]:
37
38
39  def graph_math_exp(data):
40      xs = np.arange(500, 10000 + 1, 500)
41      ys = np.array([math_exp(probabilities(data[:x])) for x in xs])
42
43      absolute = abs(ys[9] - ys[-1])
```

```
44      relative = absolute / ys[9] * 100
45      print(
46          f'Относительная погрешность измерения математического ожидания '
47          f'для выборки из {xs[9]} элементов: {round(relative, 3)}%'
48      )
49
50      df = pd.DataFrame({
51          'Объём выборки': xs,
52          'Математическое ожидание': ys
53      })
54
55      sns.lineplot(
56          data=df,
57          x='Объём выборки',
58          y='Математическое ожидание'
59      ).set_title(
60          'Зависимость мат. ожидания от размера выборки'
61      )
62      plt.show()
63
64  def graph_std_dev(data):
65      xs = np.arange(500, 10000 + 1, 500)
66      ys = np.array([std_dev(probabilities(data[:x])) for x in xs])
67
68      absolute = abs(ys[9] - ys[-1])
69      relative = absolute / ys[9] * 100
70      print(
71          f'Относительная погрешность измерения среднеквадратичного отклонения
            ↪  '
72          f'для выборки из {xs[9]} элементов: {round(relative, 3)}%'
73      )
74
75      df = pd.DataFrame({
76          'Объём выборки': xs,
77          'Среднеквадратичное отклонение': ys
78      })
79
80      sns.lineplot(
81          data=df,
82          x='Объём выборки',
83          y='Среднеквадратичное отклонение'
```

```
    ).set_title(
        'Зависимость среднеквадратичного отклонения от размера выборки'
    )
    plt.show()


# # Критерии

# ## Критерий $\chi$-квадрат

# In[4]:


def crit_chi_squared(observed, expected, k, alpha=0.05):
    chi_squared = np.sum(np.square(observed - expected) / expected)
    critical = chi2.ppf(1 - alpha, k - 1)
    result = chi_squared <= critical
    return result


# ## Критерий серий

# In[5]:


def crit_series(data, observed, expected, d=64):
    observed = np.zeros(d * d, dtype=int)
    for i in range(len(data) // 2):
        q, r = floor(data[2 * i] * d), floor(data[2 * i + 1] * d)
        observed[q * d + r] += 1

    expected = np.full(d * d, len(data) / (2 * d * d))
    return crit_chi_squared(observed, expected, d * d)


# ## Критерий интервалов

# In[6]:


def crit_intervals(data, q=128, d=16):
```

```
125     n = len(data)
126     m = n // q
127 
128     observed = np.zeros(m, dtype=int)
129     j, s = -1, 0
130     while s != m and j != n:
131         j, r = j + 1, 0
132         while j != n and data[j] * d < d / 2:
133             j, r = j + 1, r + 1
134         observed[min(r, m - 1)] += 1
135         s += 1
136 
137     expected = np.zeros(m)
138     p = 0.5
139     for r in range(m - 1):
140         expected[r] = m * p * (1 - p) ** r
141     expected[m - 1] = m * (1 - p) ** m
142 
143     return crit_chi_squared(observed, expected, m + 1)
144 
145 
146 # ## Критерий разбиений
147 
148 # In[7]:
149 
150 
151 def stirling2(n, k):
152     if n <= 0 or n != 0 and n == k:
153         return 1
154     elif k <= 0 or n < k:
155         return 0
156     elif n == 0 and k == 0:
157         return -1
158     else:
159         return k * stirling2(n - 1, k) + stirling2(n - 1, k - 1)
160 
161 def crit_partition(data, k, d=16):
162     n = len(data)
163     observed = np.zeros(k)
164     for i in range(n // k):
165         hand = {floor(v * d) for v in data[i * k:(i + 1) * k]}
```

```
166         observed[len(hand) - 1] += 1
167 
168     expected = np.zeros(k)
169     for r in range(1, k + 1):
170         p = 1.0
171         for i in range(r):
172             p *= d - i
173         expected[r - 1] = (n / k) * (p / d ** k) * stirling2(k, r)
174 
175     return crit_chi_squared(observed, expected, k)
176 
177 
178 # ## Критерий перестановок
179 
180 # In[8]:
181 
182 
183 def crit_permutation(data, t=4, d=1024):
184     n = len(data)
185     t_fact = factorial(t)
186     observed = np.zeros(t_fact)
187     for i in range(n // t):
188         v = [v * d for v in data[t * i:t * i + t]]
189         c = np.zeros(t, dtype=int)
190         r = t
191 
192         while r > 0:
193             s = 0
194             for j in range(r):
195                 if v[j] > v[s]:
196                     s = j
197             c[r - 1] = s
198             v[r - 1], v[s] = v[s], v[r - 1]
199             r -= 1
200 
201         f = 0
202         for j in range(t - 1):
203             f = (f + c[j]) * (j + 2)
204         f += c[t - 1]
205         observed[f] += 1
206 
```

```
207     expected = np.full(t_fact, n / t / t_fact)
208
209     return crit_chi_squared(observed, expected, t_fact)
210
211
212 # ## Критерий монотонности
213
214 # In[9]:
215
216
217 def crit_monotonic(data, d=1024):
218     last, cnt = data[0] * d, 0
219     c = np.zeros(6)
220     for x in data[1:]:
221         y = x * d
222         if y > last:
223             cnt += 1
224         else:
225             c[min(cnt, 5)] += 1
226             cnt = 0
227         last = y
228     c[min(cnt, 5)] += 1
229
230     a = np.array([
231         [4529.4,  9044.9,  13568.0, 18091.0,  22615.0,  27892.0],
232         [9044.9,  18097.0, 27139.0, 36187.0,  45234.0,  55789.0],
233         [13568.0, 27139.0, 40721.0, 54281.0,  67852.0,  83685.0],
234         [18091.0, 36187.0, 54281.0, 72414.0,  90470.0,  111580.0],
235         [22615.0, 45234.0, 67852.0, 90470.0,  113262.0, 139476.0],
236         [27892.0, 55789.0, 83685.0, 111580.0, 139476.0, 172860.0],
237     ])
238     b = np.array([
239         1.0 / 6.0, 5.0 / 24.0, 11.0 / 120.0,
240         19.0 / 720.0, 29.0 / 5040.0, 1.0 / 840.0,
241     ])
242
243     n = len(data)
244     m = 0.0
245     for i in range(6):
246         for j in range(6):
247             m += (c[i] - n * b[i]) * (c[j] - n * b[j]) * a[i, j]
```

```
248
249     return chi2.ppf(0.05, 6) <= m / n <= chi2.ppf(1 - 0.05, 6)
250
251
252 # ## Критерий конфликтов
253
254 # In[10]:
255
256
257 def p(n, m, k):
258     n = float(n)
259     m = float(m)
260     k = float(k)
261     return binom(n, k) * (m ** (-k)) * (1 - 1 / m) ** (n - k)
262
263 def crit_conflicts(data, q=128):
264     n = len(data)
265     m = n * q
266     unique, counts = np.unique(np.uint32(np.floor(np.array(data) * m)),
      ↪ return_counts=True)
267     empty = m - unique.shape[0]
268     observed = (sum(p(n, m, k) for k in counts) + p(n, m, 0) * empty) / m
269     expected = (n / m - 1 + p(n, m, 0)) * m
270     return observed < expected
271
272
273 # # Проверки критериев
274
275 # In[11]:
276
277
278 def prepare(xs):
279     m, *xs = xs
280     return [x / m for x in xs]
281
282 files = {
283     'Линейный конгруэнтный': 'rnd-lc.dat',
284     'Аддитивный': 'rnd-add.dat',
285     'Пятипараметрический': 'rnd-5p.dat',
286     'РСЛОС': 'rnd-lfsr.dat',
287     'Нелинейная комбинация РСЛОС': 'rnd-nfsr.dat',
```

```
288     'Вихрь Мерсенна': 'rnd-mt.dat',
289     'RC4': 'rnd-rc4.dat',
290     'RSA': 'rnd-rsa.dat',
291     'Алгоритм Блюма-Блюма-Шуба': 'rnd-bbs.dat',
292 }
293
294 data = dict()
295
296 for filename in files.values():
297     with open(filename) as f:
298         xs = list(map(float, f.read().strip().split(',')))
299     data[filename] = prepare(xs)
300
301
302 # In[12]:
303
304
305 def run_tests(data):
306     probs = probabilities(data)
307     print(f"Математическое ожидание = {math_exp(probs)}")
308     print(f"Среднеквадратичное отклонение = {std_dev(probs)}")
309     graph_math_exp(data)
310     graph_std_dev(data)
311
312     k = 5
313     observed = np.zeros(k, dtype=int)
314     for val in data:
315         for i in range(0, k):
316             if i / k <= val < (i + 1) / k:
317                 observed[i] += 1
318
319     expected = np.full(k, len(data) / k)
320
321     chi = crit_chi_squared(observed, expected, k)
322     series = crit_series(data, observed, expected)
323     intervals = crit_intervals(data)
324     partition = crit_partition(data, k=8)
325     permutation = crit_permutation(data)
326     monotonic = crit_monotonic(data)
327     conflicts = crit_conflicts(data)
328     print(f"Критерий хи-квадрат:   {chi}")
```

```
329     print(f"Критерий серий:         {series}")
330     print(f"Критерий интервалов:    {intervals}")
331     print(f"Критерий разбиений:     {partition}")
332     print(f"Критерий перестановок:  {permutation}")
333     print(f"Критерий монотонности:  {monotonic}")
334     print(f"Критерий конфликтов:    {conflicts}")
335
336
337 # ## Линейный конгруэнтный генератор
338
339 # In[13]:
340
341
342 run_tests(data["rnd-lc.dat"])
343
344
345 # ## Аддитивный генератор
346
347 # In[14]:
348
349
350 run_tests(data["rnd-add.dat"])
351
352
353 # ## Пятипараметрический генератор
354
355 # In[15]:
356
357
358 run_tests(data["rnd-5p.dat"])
359
360
361 # ## РСЛОС
362
363 # In[16]:
364
365
366 run_tests(data["rnd-lfsr.dat"])
367
368
369 # ## Нелинейная комбинация РСЛОС
```

```
370
371 # In[17]:
372
373
374 run_tests(data["rnd-nfsr.dat"])
375
376
377 # ## Вихрь Мерсенна
378
379 # In[18]:
380
381
382 run_tests(data["rnd-mt.dat"])
383
384
385 # ## RC4
386
387 # In[19]:
388
389
390 run_tests(data["rnd-rc4.dat"])
391
392
393 # ## RSA
394
395 # In[20]:
396
397
398 run_tests(data["rnd-rsa.dat"])
399
400
401 # ## Алгоритм Блюма-Блюма-Шуба
402
403 # In[21]:
404
405
406 run_tests(data["rnd-bbs.dat"])
407
```