Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №3 по дисциплине «Моделирование»

Тема Генераторы псевдослучайных чисел

Студент Слепокурова М.Ф.

Группа ИУ7-76Б

Оценка (баллы) _____

Преподаватель Рудаков И.В.

Москва — 2023 г.

Постановка задачи

Изучить методы генерирования псевдослучайных чисел и критерии оценки случайности последовательности. Реализовать собственный критерий оценки и проверить его работу на одноразрядных, двухразрядных и трехразрядных последовательностях, полученных табличным и выбранным алгоритмическим методом. Реализовать возможность ручного ввода последовательности чисел и оценки ее случайности по разработанному критерию.

Теория

Алгоритмический метод

В качестве алгоритмического метода генерации псевдослучайных чисел был выбран линейно конгруэнтный алгоритм. Для реализации метода необходимо задать следующие константы:

- $m > 0$ — модуль;
- $0 \leq a \leq m$ — множитель;
- $0 \leq c \leq m$ — приращение;
- $0 \leq x_0 \leq m$ — начальное значение.

Тогда последовательность можно сгенерировать с помощью реккурентной формулы:

$$x_{n+1} = (ax_n + c) \mod m \tag{1}$$

Для того, чтобы проверить работоспособность разработанного критерия оценки случайности, в рамках выполнения работы использовались случайно сгенерированные библиотекой начальные значения m, a, c, x_0.

Табличный метод

Для табличного метода была использована таблица некоррелированных случайных чисел из книги «Million Random Digits with 100,000 Normal Deviates». Суть метода заключается в последовательном чтении уже сгенерированной таблицы значений.

Критерий оценки

Для оценки случайности последовательностей был разработан собственный критерий. В основе созданного критерия лежит оценка частот вхождения элементов последовательности и оценка приращений между соседними элементами.

Для каждого элемента последовательности предварительно вычисляется количество его вхождений в эту последовательность, после чего вычисляется максимальное количество повторяющихся количеств вхождений — k.

Затем для последовательности вычисляются приращения между ее соседними элементами, после чего вычисляется максимальное количество повторяющихся приращений — m.

Тогда для последовательности длиной l запишем предварительную формулу для вычисления коэффициента:

$$rate' = \frac{k + m + 1}{l} \tag{2}$$

Заметим, что максимальное значение коэффициента = 2, когда $m + 1$ и $k = l$. Для того, чтобы коэффициент принимал значения в диапазоне от 0 до 1, необходимо поделить полученное значение на максимальное, т.е. пополам. Получается, чем выше значение коэффициента, тем менее случайна последовательность — такую оценку сложно воспринимать, поэтому инвертируем полученное значение путем его вычитания из нового максимального значения диапазона, равного единице. Примечание: используем значение $m + 1$, а не m, чтобы получать значение коэффициента 0 для последовательностей, состоящих из одного повторяющегося элемента.

Тогда итоговую формулу для вычисления коэффициента случайности можно записать как:

$$rate = 1 - \frac{k + m + 1}{2l} \tag{3}$$

Теперь имеем коэффициент, принимающий значения из диапазона от 0 до 1, причем чем больше значение, тем более случайна рассматриваемая последовательность.

Такой способ не позволяет отследить сложные функциональные зависимости между элементами последовательности, однако хорошо справляется с периодичными последовательностями и последовательностями, содержащими множество повторяющихся элементов.

Средства реализации

Для реализации приложения был выбран язык программирования Python, в стандартную библиотеку которого входит графическая библиотека Tkinter, использовавшаяся для реализации пользовательского интерфейса.

Листинг кода

```
from random import randint

def generateTabular(count, low=0, high=100):
  seq = []
  with open('table.txt') as file:
    for line in file:
      for curr in line.split(" ")[1:]:
        if (len(seq) < count and curr != ''): seq.append(low + int(curr) % (
            high - low))
      if (len(seq) >= count): break
  return seq

def __settingsLinearCongruent():
  m = randint(100000, 1000000)
  a = randint(10000, 100000)
  c = randint(10000, 100000)
  x0 = randint(10000, 100000)
  return m, a, c, x0

def generateLinearCongruent(count, low=0, high=100):
  m, a, c, x0 = __settingsLinearCongruent()
  seq = []
  seq.append(x0)
  for i in range(1, count + 1):
    curr = (seq[i-1] * a + c) % m
    seq.append(low + curr % (high - low))
  return seq[1:]

def __occurrenceRate(seq):
  occurrence = dict()
  for num in seq:
    if (not occurrence.get(num)): occurrence[num] = 1
    else: occurrence[num] += 1
  return occurrence

def __increment(seq):
  increment = []
  for i in range(1, len(seq)):
    increment.append(seq[i] - seq[i-1])
  return increment

def randomnessFactor(seq):
  occurence = __occurrenceRate(seq)
  increment = __increment(seq)
  k = max(__occurrenceRate(occurence.values()))
  m = max(__occurrenceRate(increment).values())
  return 1 - (k + m + 1) / len(seq) / 2
```

Демонстрация работы программы

На рисунке 1 изображен пример работы программы для последовательностей из 100 элементов. Заметим, что в данном случае были выбраны неоптимальные начальные значения для алгоритмического метода для последовательностей одноразрядных и двухразрядных чисел. Полученные последовательности имеют малый период и, соответственно, большое количество повторяющихся элементов, поэтому их коэффициент случайности значительно ниже в сравнении с другими. Коэффициенты случайности для всех трех последовательностей, полученных табличным методом, велики.

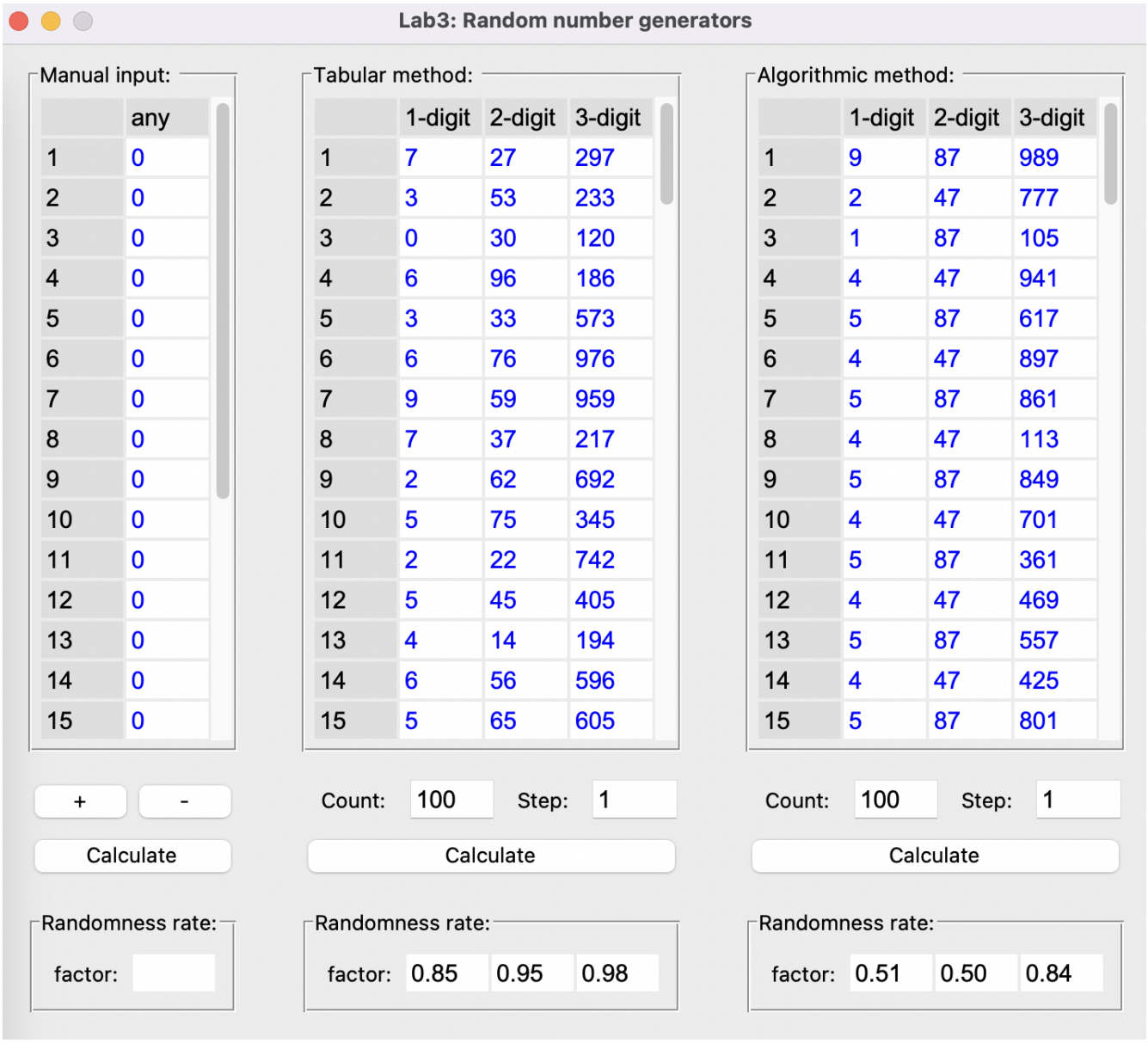


Рисунок 1 – Пример работы программы — 1

На рисунке 2 изображен пример работы программы для последовательностей из 10000 элементов. Заметим, что в данном случае были выбраны неоптимальные начальные значения для алгоритмического метода для последовательности одноразрядных чисел, полученных алгоритмическим методом. Полученная последовательность состоит из одного повторяющегося элемента, ввиду чего ее коэффициент случайности минимален. Для последовательностей двухразрядных и трехразрядных чисел, полученных алгоритмическим методом, были подобраны оптимальные значения, поэтому коэффициенты случайности для них велики. Коэффициенты случайности для всех трех последовательностей, полученных табличным методом, велики.

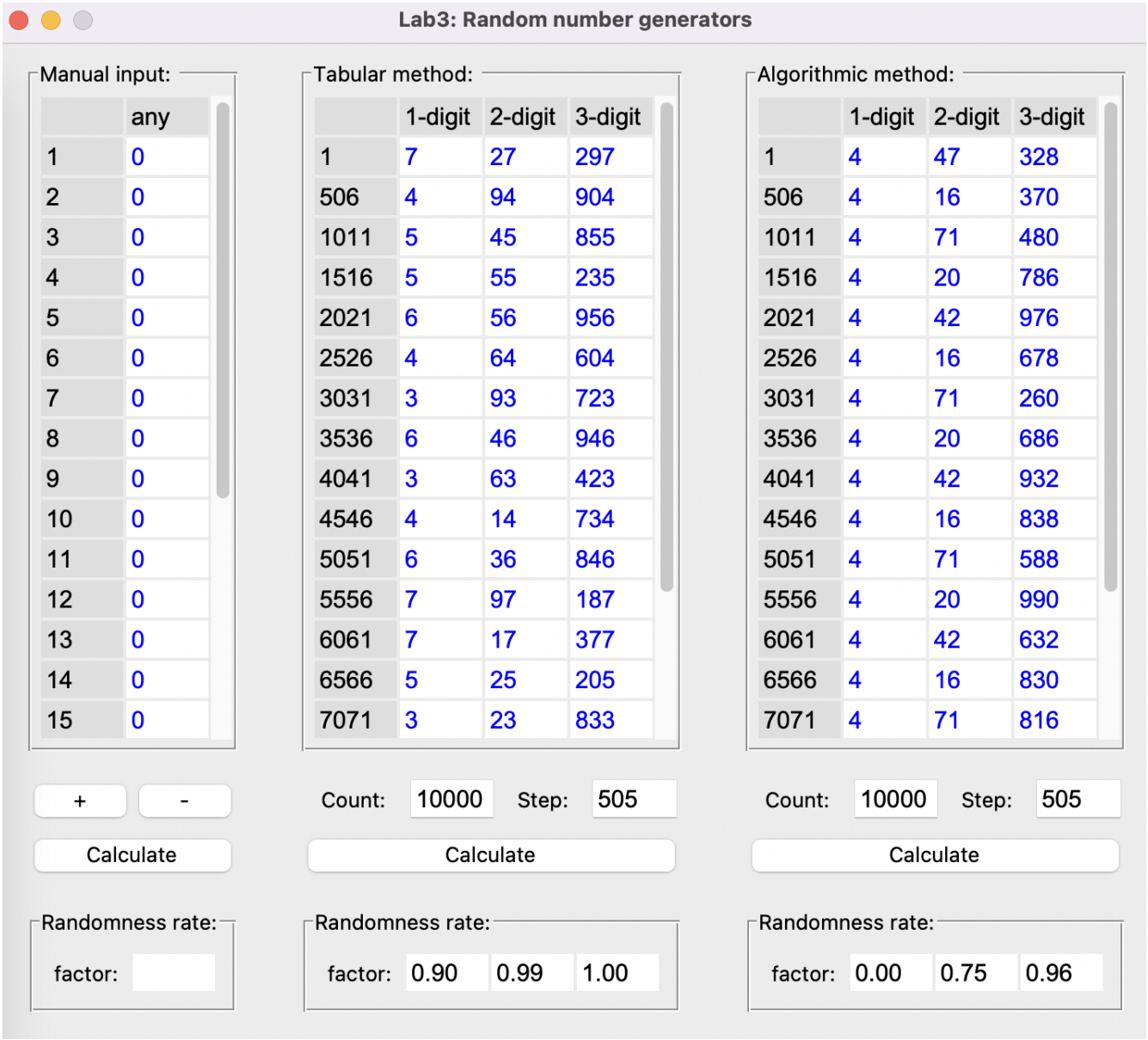
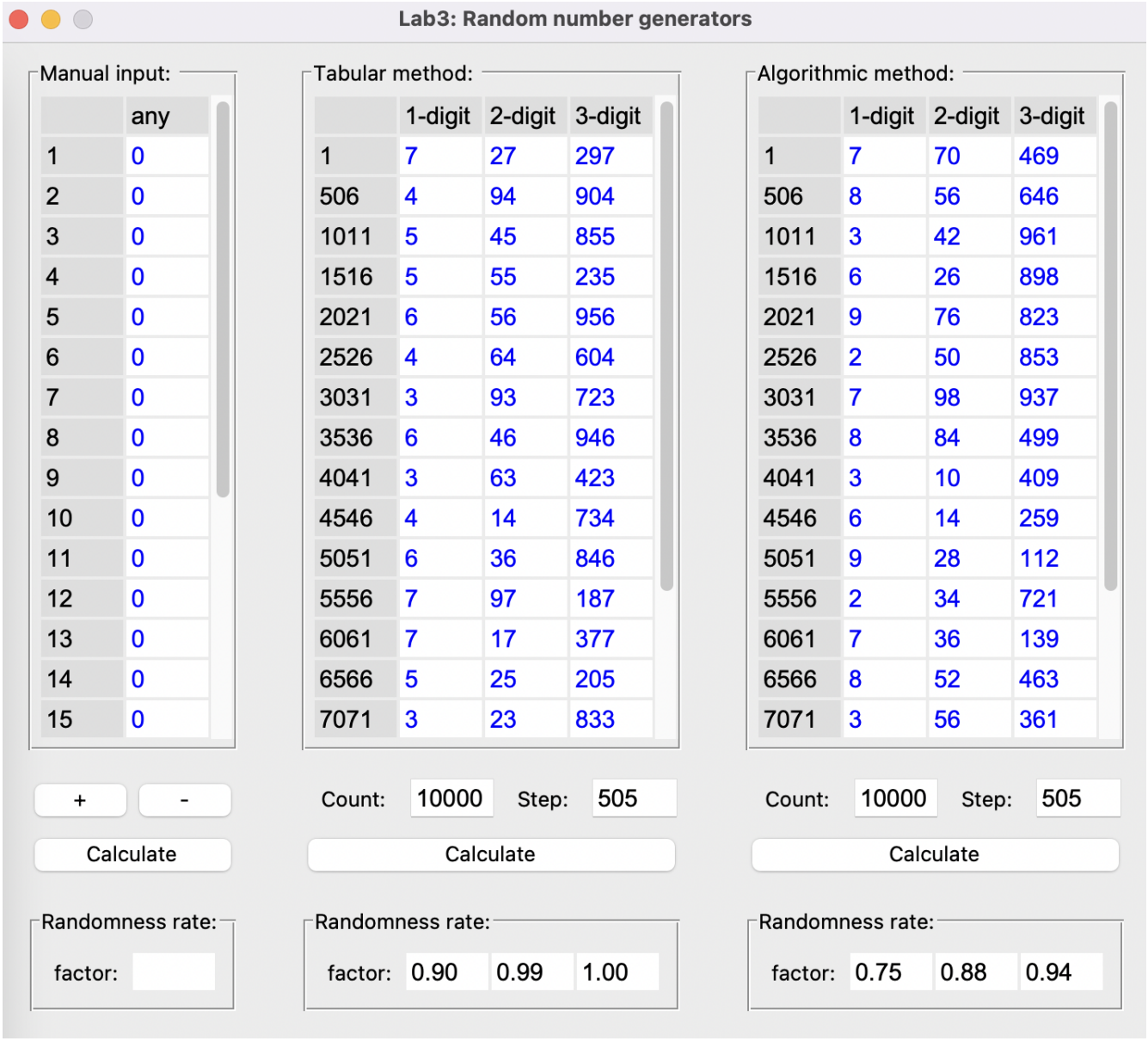


Рисунок 2 – Пример работы программы — 2

На рисунке 3 изображен пример работы программы для последовательностей из 10000 элементов. В данном случае были выбраны оптимальные начальные значения для всех трех последовательностей, полученных алгоритмическим методом. Коэффициент случайности для последовательности одноразрядных чисел несколько ниже двух других, т.к. в нем явно выражена периодичность.



	any
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0

	1-digit	2-digit	3-digit
1	7	27	297
506	4	94	904
1011	5	45	855
1516	5	55	235
2021	6	56	956
2526	4	64	604
3031	3	93	723
3536	6	46	946
4041	3	63	423
4546	4	14	734
5051	6	36	846
5556	7	97	187
6061	7	17	377
6566	5	25	205
7071	3	23	833

	1-digit	2-digit	3-digit
1	7	70	469
506	8	56	646
1011	3	42	961
1516	6	26	898
2021	9	76	823
2526	2	50	853
3031	7	98	937
3536	8	84	499
4041	3	10	409
4546	6	14	259
5051	9	28	112
5556	2	34	721
6061	7	36	139
6566	8	52	463
7071	3	56	361

Рисунок 3 – Пример работы программы — 3

На рисунке 4 изображен пример работы программы для последовательности, введенной вручную. Данная последовательность представляет собой арифметическую прогрессию с шагом 1, т.е. не является случайной. Коэффициент случайности имеет значение 0.5 ввиду того, что элементы в последовательности не повторяются, однако приращение между ними одинаково, т.е. присутствует связь между элементами.

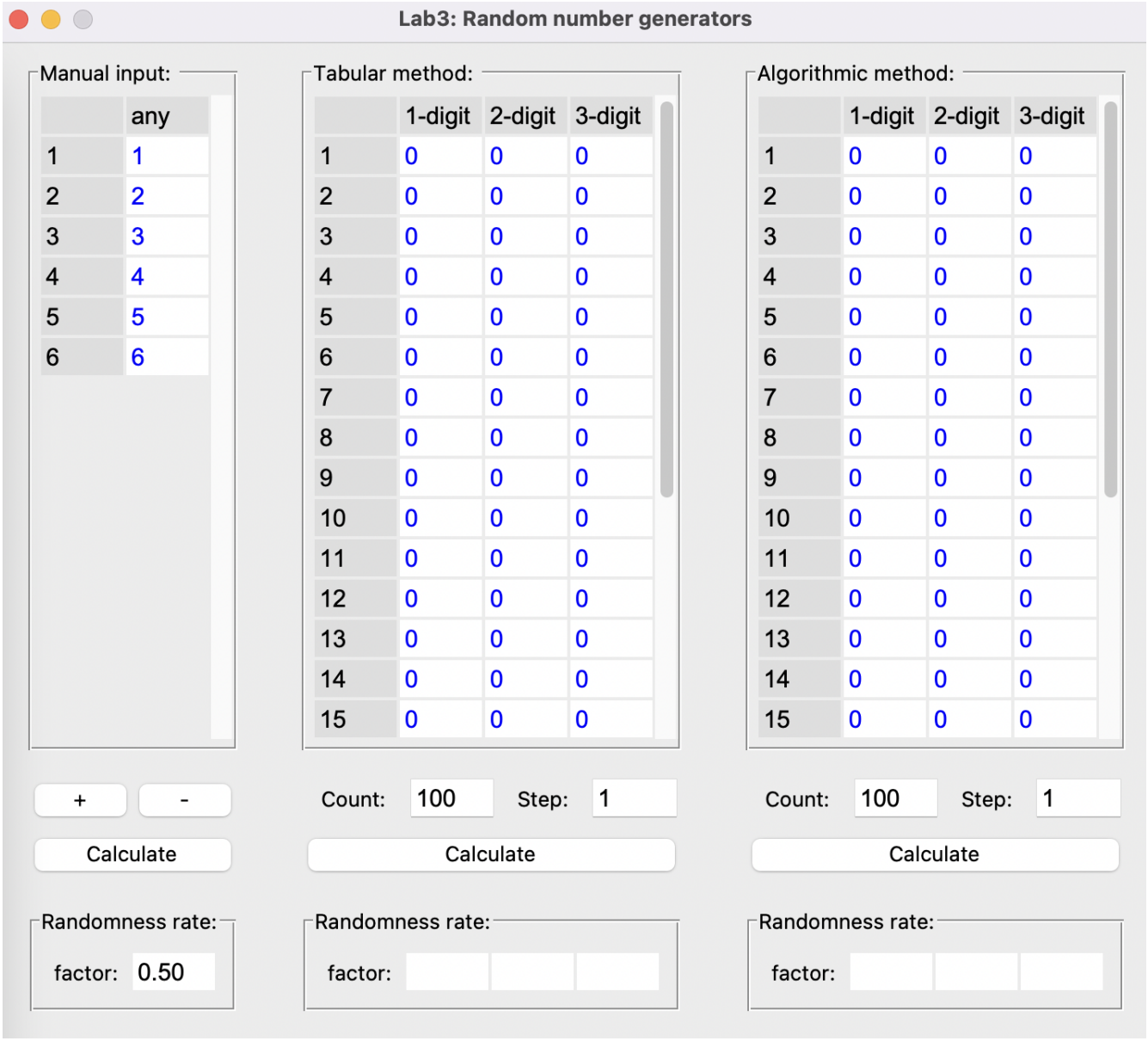


Рисунок 4 – Пример работы программы — 4

На рисунке 5 изображен пример работы программы для последовательности, введенной вручную. Данная последовательность состоит из одного повторяющегося элемента, поэтому коэффициент случайности для нее получился минимальным.

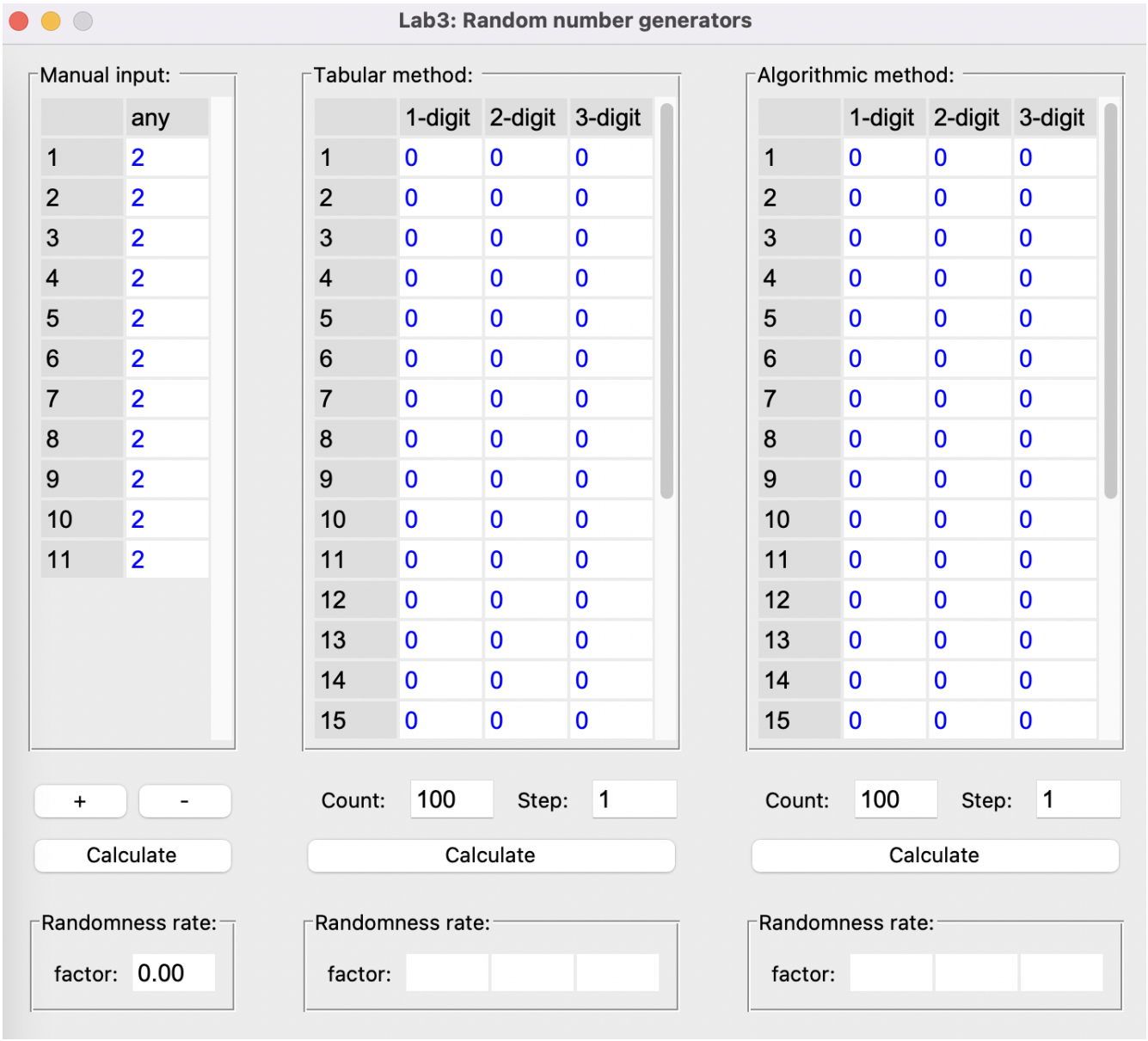


Рисунок 5 – Пример работы программы — 5

На рисунке 6 изображен пример работы программы для последовательности, введенной вручную. Данная последовательность случайна, поэтому коэффициент случайности для нее велик.

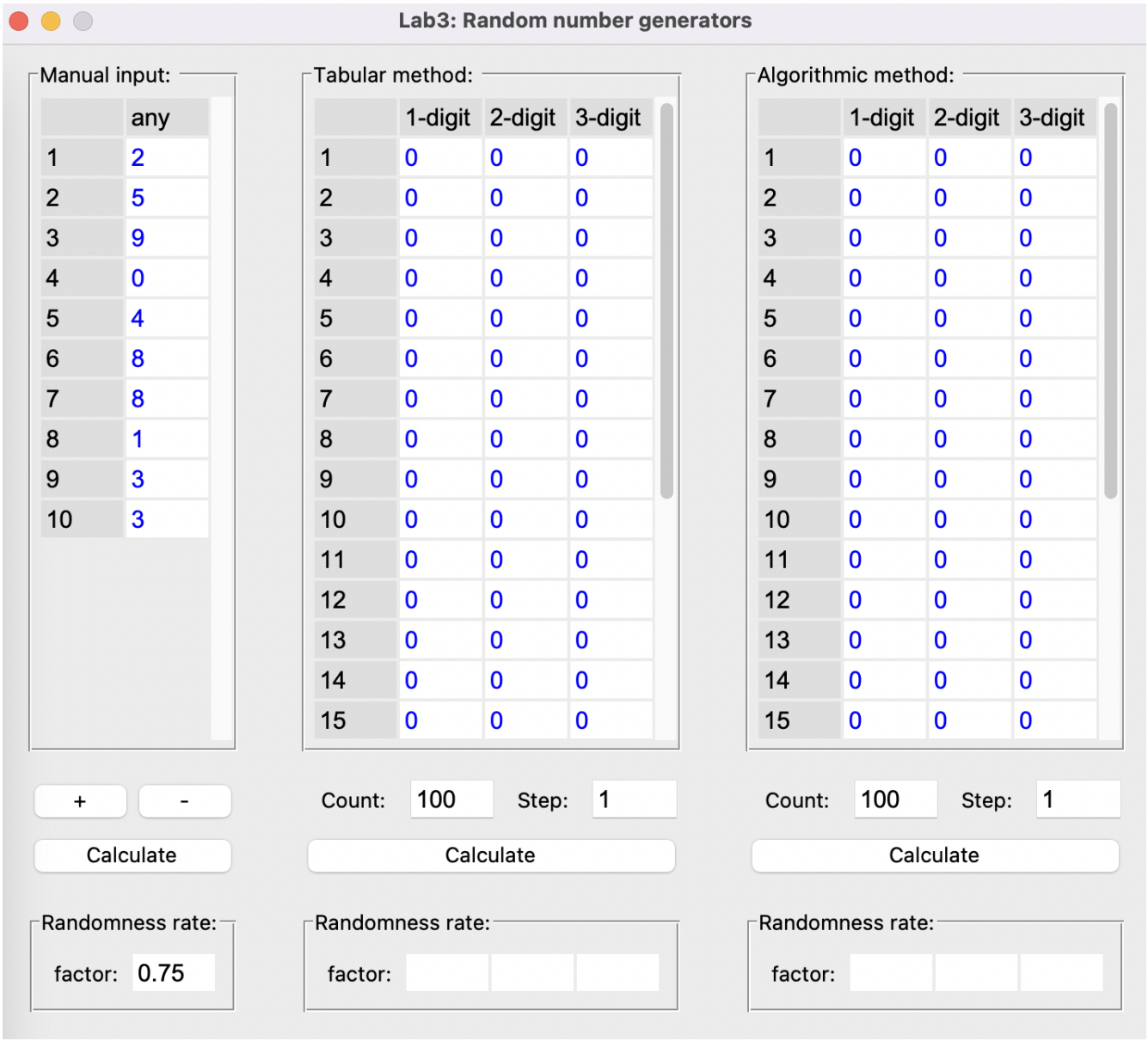


Рисунок 6 – Пример работы программы — 6