ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Отчет по лабораторной работе по дисциплине: Вероятностные модели инфокоммуникационных процессов

Лабораторная работа №1. «Исследование псевдослучайных последовательностей»

Студента заочного отделения
2 курса 12002153 группы
Пасивенко А.Ю.

Проверил:

Белгород 2023

Цель работы: Научиться генерировать последовательности случайных чисел и считать их вероятностные характеристики.

Теоретическая часть

В настоящее время создано большое количество разнообразных программных продуктов для моделирования случайных процессов, однако, следует знать, что задача формирования случайных величин с заданным законом распределения по-прежнему полностью не решена. Возникает вопрос: почему? Ответ состоит в том, что нельзя искусственно получить ***случайный поток*** с заданными вероятностными свойствами, так как все существующие методы получения случайных чисел используют рекуррентные формулы, реализующие детерминированные алгоритмы. То есть, на практике находят широкое применение способы получения потоков «случайных» величин, которые удовлетворяют определенным критериям на случайность, хотя на самом деле таковыми не являются.

Для формирования потоков случайных величин необходимо иметь последовательность чисел со случайным законом распределения, обычно равномерным. Формирование таких последовательностей задача довольно сложная. Строго говоря, утверждение о том, что конечная последовательность чисел, цифр или событий любого рода случайна, относится не к фактическому виду последовательности, а к способу ее получения. Случайным, например, является процесс подбрасывания монеты, отдельные результаты которого не могут быть предсказаны заранее, но соответствуют некоторому распределению вероятностей. Любую последовательность событий, сформированных таким способом, не совсем точно можно назвать *случайной.* Однако важно понимать, что когда последовательность называют случайной, то имеют в виду не характер и свойства последовательности апостериори, а априорные условия ее формирования.

Функцию получения равномерно распределенных чисел в интервале от 0 до 1 выполняют так называемые ***датчики случайных чисел,*** встраиваемые во все существующие программные средства для моделирования систем. На основе таких датчиков, как будет показано далее, можно получить последовательность случайных величин с любым распределением вероятностей.

Простейшие датчики случайных чисел генерируют значения случайной величины, имеющей равномерное распределение в интервале (0, 1). Основная проблема разработки таких программ состоит в необходимости обеспечить *случайные числа* с помощью *детерминированного процесса* исполнения. Реально такие программы генерируют псевдослучайные числа. Качество последовательностей, генерируемых датчиком случайных чисел, оценивается близостью к действительно случайным с помощью многих тестов. Как показал многолетний опыт разработки, наиболее эффективными алгоритмами работы датчиков случайных чисел являются так называемые *конгруэнтные генераторы*. Алгоритмы их работы можно описать следующими соотношениями:

алгоритм 1 $$x_{i+1} = (ax_i + c) \bmod(m) \qquad (1)$$

или алгоритм 2 $$x_{i+1} = (xx_i) \bmod(m) . \qquad (2)$$

Здесь всюду a, c, x_1, x— целые числа, а в качестве выходного псевдослучайного числа используется действительное $U_j = x_j/m$. Операции вычисления по модулю mod(m) обеспечиваются работой алгоритма в компьютерах с конечной разрядностью. Приведем сначала демонстрационный пример работы такого алгоритма при следующих параметрах: .

Так как $32=2^5$, следовательно, здесь используются пятиразрядные машинные слова для представления целых чисел. Пока эти числа не превышают 31, результаты операций очевидны. При превышении этого значения будет использоваться остаток от деления на целое, кратное 32 так, как это показано на рисунке 1.

$x_0 = 1$	00001	$U = 1/32 = 0{,}03125$
$x_1 = 3$	00011	$U = 3/32 = 0{,}09375$
$x_2 = 9$	01000	$U = 9/32 = 0{,}28125$
$x_3 = 27$	11011	$U = 27/32 = 0{,}84375$
$x_4 = 81 \bmod 32 = 17$	10001	$U = 17/32 = 0{,}53125$
$x_5 = 51 \bmod 32 = 19$	10011	$U = 19/32 = 0{,}59375$
$x_6 = 57 \bmod 32 = 25$	11001	$U = 25/32 = 0{,}78125$
$x_7 = 75 \bmod 32 = 11$	01011	$U = 11/32 = 0{,}34375$

Если продолжить вычисление, то на следующем шаге будет получена единица и цикл замкнется. Далее последовательность нельзя будет считать псевдослучайной. Ясно, что чем больше период, тем последовательность лучше отражает свойства случайного потока. Нетрудно убедиться, что период будет зависеть как от выбора *m,* так и от начального значения *x.*

Таким образом, получить равномерное распределение чисел в единичном интервале можно используя либо формулу суммирования (1), либо формулу умножения чисел (2) по заданному модулю m.

Для того чтобы убедиться в равномерности распределения полученных чисел можно воспользоваться методом на основе анализа гистограмм, построенных после проведения нескольких циклов работы датчика. Гистограмма – это разновидность диаграммы, столбцы которой

пропорциональны числу значений случайной величины в заданных интервалах. Для построения гистограммы необходимо разделить весь диапазон значений случайной величины на несколько смежных поддиапазонов (интервалов), например на 10 или 100, а затем подсчитать, по сколько случайных величин из сформированного массива попало в каждый поддиапазон.

Программу, реализующую построение гистограммы можно написать самостоятельно, или, в случае затруднений, использовать имеющуюся в системе MATLAB специальную функцию *hist(Y)*. Здесь *Y* – массив случайных величин, для которого строится гистограмма для 10 интервалов. При необходимости задать другое количество интервалов (часто рекомендуется значение 100), следует использовать функцию вида *hist(Y,n)*, где параметр *n* определяет заданное число интервалов.

При построении гистограммы подсчитывается число попаданий полученных значений в эквидистантные интервалы, на которые разбивается вся область возможных значений чисел. Чем больше число выполненных испытаний, тем точнее полученное распределение стремится к равномерному распределению.

Известно, что полной, исчерпывающей характеристикой случайной величины является ее закон распределения. Генерирование случайной величины с заданным законом распределения вероятностей, как известно, может выполнено на основе использования датчика случайной величины с равномерным распределением в интервале значений [0, 1] с последующим преобразованием полученных значений в соответствии с обратной функцией нужного закона распределения вероятностей.

Пусть, например, необходимо реализовать поток с интервалами между поступающими событиями, распределенными по показательному закону, отличному от равномерного. Для этого, как правило, используют

механизм нелинейного преобразования случайной величины. Суть такого преобразования состоит в следующем.

Если случайная величина *X* распределена по равномерному закону на интервале (0, 1), то значения неслучайной действительной функции *f(X)* также будут являться случайной величиной *Y,* функция распределения которой будет равна $F(y)=f^{t}(y)=P\{Y<=y\}$. Соответственно обращая это утверждение, получаем, что для генерации случайной величины с функцией распределения *F(y)* можно построить детерминированную функцию $f(x)=F^{-1}(x)$ и получать искомые случайные числа как значения этой функции от аргумента, определяемого числом, являющимся случайной величиной с равномерным законом распределения на интервале *(1,0)*.

Учитывая вышеизложенное, рассмотрим в качестве важного практического примера получение случайных чисел, распределенных по экспоненциальному закону. Этот механизм используется, например, для формирования пуассоновского потока, в котором при описании межсобытийного интервала времени имеет место именно экспоненциальное распределение. Пусть необходимо генерировать случайные числа с функцией плотности вероятности

$$P(x)=\frac{1}{\lambda}e^{-x/\lambda},\ x\geq 0$$

из случайных чисел с равномерной плотностью вероятности на интервале (0, 1).

Сначала находится функцию распределения по плотности вероятности

$$F(x)=\int_{-\infty}^{x} f(y)dy=\begin{cases}1-e^{-x/\lambda}, & x\geq 0\\ 0, & x<0.\end{cases}$$

Затем определяется обратная функция, решая такое уравнение:

$$y = 1 - e^{-x/\lambda}$$
$$e^{-x/\lambda} = 1 - y$$
$$-x/\lambda = \ln(1 - y)$$
$$x = -\lambda \ln(1 - y)$$

Далее можно записать, что обратная функция, определенная конечно только на интервале (0, 1), будет иметь вид:

$$F^{-1}(y) = -\lambda \ln(1 - y)$$

Имея последовательность равномерно распределенных случайных чисел на интервале (0,1), можно вычислять значения случайных чисел с экспоненциальным распределением и средним значением X с помощью следующего алгоритма:

$$X_i = -\lambda \ln(1 - U_i). \qquad (3)$$

Поступая аналогично, можно в принципе, генерировать случайные межсобытийные интервалы с любым заданным законом распределения.

Так, например, имея последовательность равномерно распределенных случайных чисел на интервале (0,1), можно вычислять значения случайных чисел на основе показательного закона распределения с помощью следующего алгоритма:

$$Xi = -\ln(1 - Ui) / \lambda \qquad (4)$$

Здесь U_i- случайная величина с равномерным распределением на интервале (0,1).

Математическое ожидание в этом случае в соответствии с теорией должно быть равно среднеквадратическому отклонению случайной величины, а численно $-1/\lambda$.

В системе MATLAB можно генерировать массив из *n* случайных чисел, равномерно распределенных на интервале (0, 1) с помощью функции X = rand(n).

Для полученной псевдослучайной последовательности можно рассчитать ее вероятностные характеристики:

Математическим ожиданием случайной величины U_i называется сумма произведений случайной величины на ее вероятность, т.е.

Но, так как вероятности случайной величины U_i неизвестны, то оценка математического ожидания для случайной последовательности производится по формуле:

$$M_U = \frac{1}{N}\sum_{i=1}^{N} U_i \qquad (5)$$

Дисперсией называется математическое ожидание квадрата отклонения случайной величины от ее математического ожидания, т.е.

$$D_U = \frac{1}{N-1}\sum_{i=1}^{N} (U_i - M_U)^2 \qquad (6)$$

Среднеквадратическим отклонением называется корень квадратный из дисперсии, т.е.

$$\sigma_U = \sqrt{D_U} \qquad (7)$$

Аавтокорреляционная функция (нормированная) представляет собой последовательность коэффициентов корреляции, зависящих от величины сдвига, как от аргумента.

$K(r)=1/D \cdot M[(xi - m)(xi + r-m)]$ (8)

Ее оценка вычисляется:

K*(r)=1/D*(N-r-1)n-2∑i=1[(xi-m*)(xi+r-m*)]= =1/ D*(1/N-r-1)n-2∑i=1xixi+r-(N-r)/(N-r-1)m*

(9)

Листинг датчика на основе алгоритма сложения:

```
a = 565;
c = 323;
m = 56238423983;
x = ones;
n = 1000;
u = 1/m;
for i=1:n
    x(i+1) = mod((a*x(i)+c),m);
    u(i+1) = x(i+1)/m;
end
figure
histogram(u,n);

M = sum(u)/n;
disp("M = "); disp(M);
Dr = zeros;
for i=1:n
    Dr(i) = (u(i)-M)^2;
end
D = sum(Dr)/n;
disp("D = "); disp(D);
```

Результат работы датчика сложения

Для построения гистограммы использовался массив N = 1000 при a = 565, c =323, m = 56238423983.

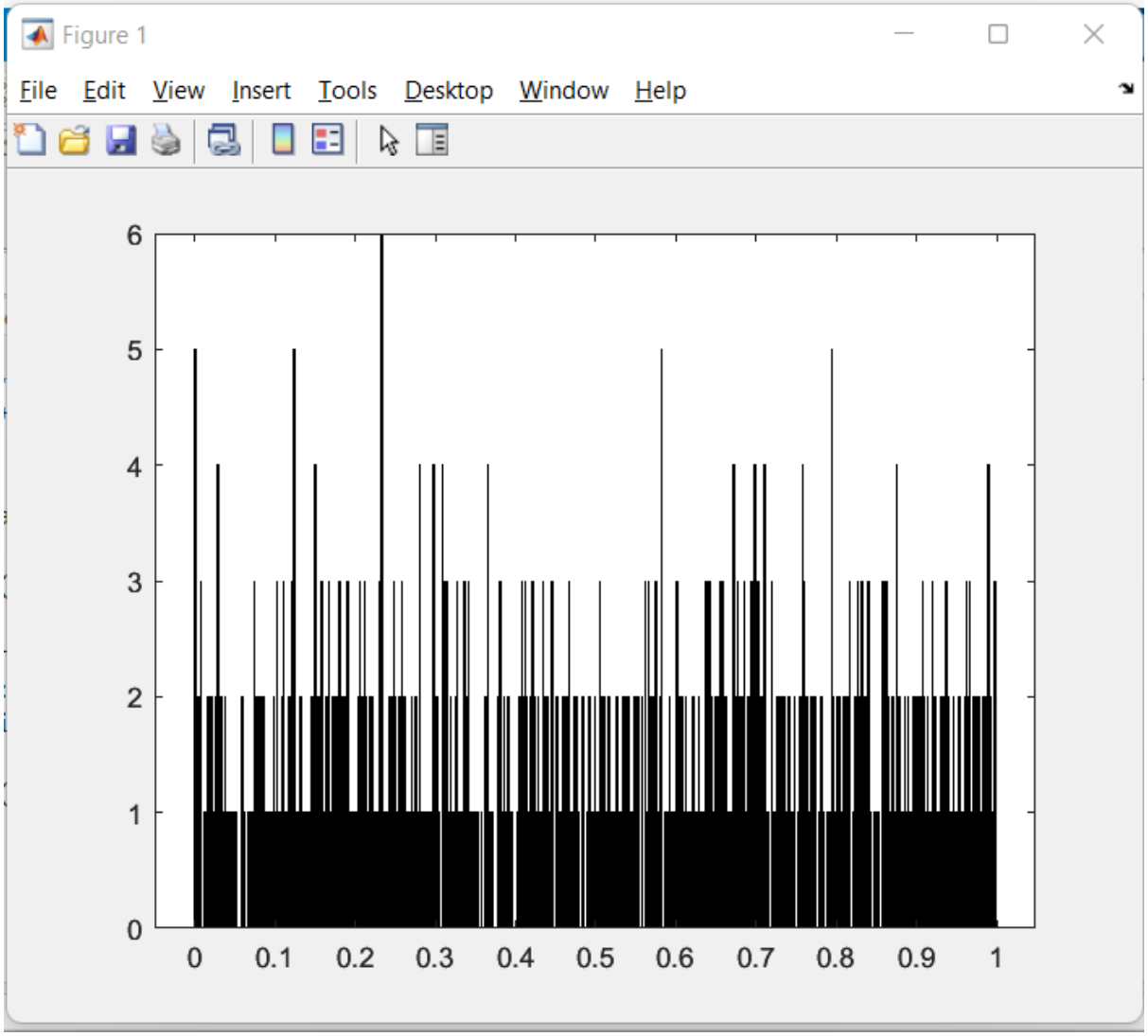

Рисунок 1 – Гистограмма, где размерность массива N равна 1000

```
M =
    0.5086

D =
    0.0852
```

Рисунок 2 – Полученные математическое ожидание и дисперсия

Рассматривая полученные данные (рисунки 1-2) затруднительно сделать вывод о том смог ли зациклиться датчик сложения. Результаты дисперсии и математического ожидания приблизительно равны

теоретическим данным. Для более точного анализа работы датчика увеличим размерность последовательности до 10000.

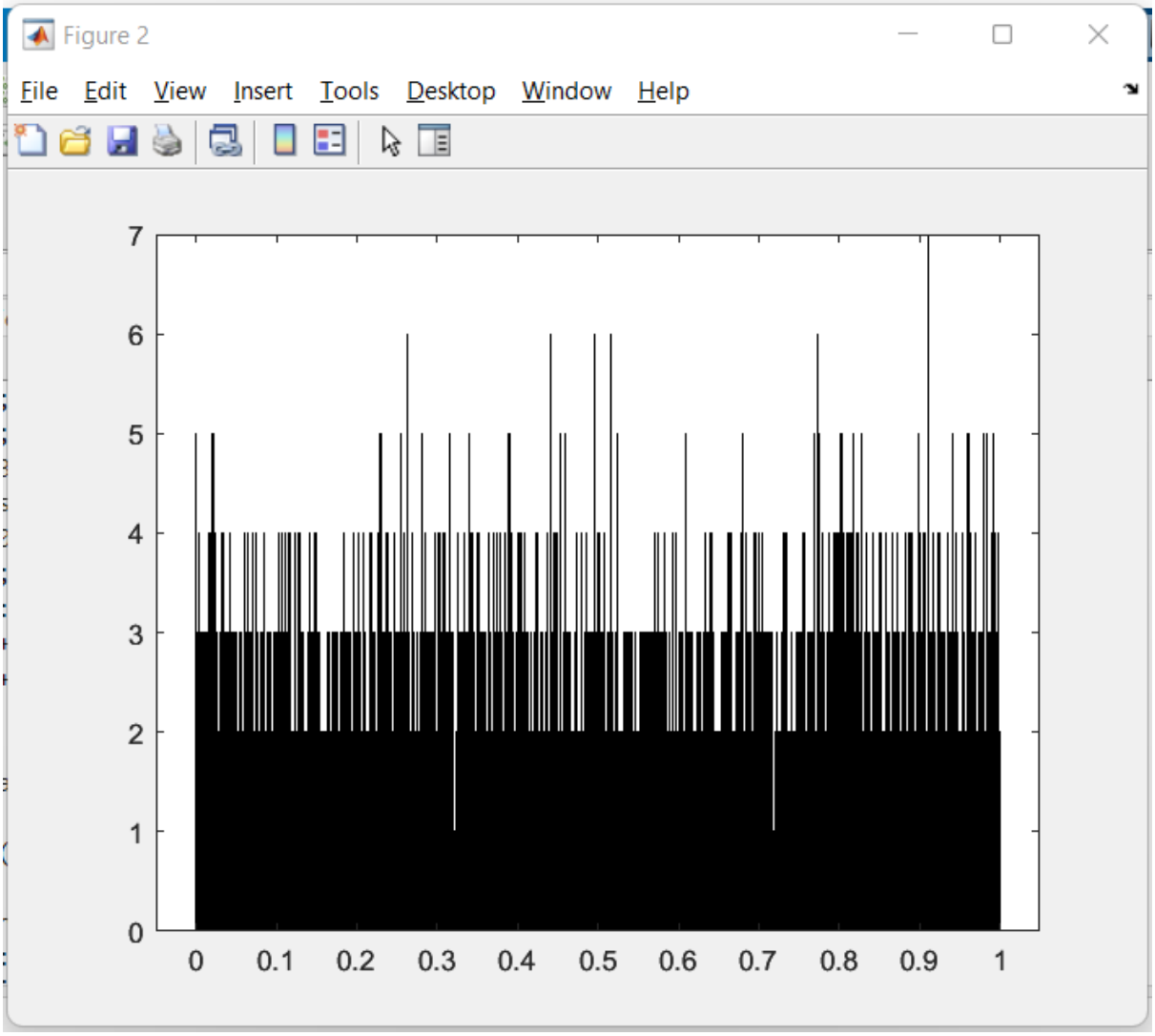

Рисунок 3 – Гистограмма, где размерность массива N равна 10000

```
M =
    0.5021

D =
    0.0835
```

Рисунок 4 – Полученные математическое ожидание и дисперсия

Рассматривая полученные данные (рисунки 3-4) затруднительно сделать вывод о том смог ли зациклиться датчик сложения. Результаты дисперсии и математического ожидания приблизительно равны теоретическим данным. Для более точного анализа работы датчика увеличим размерность последовательности до 100000.

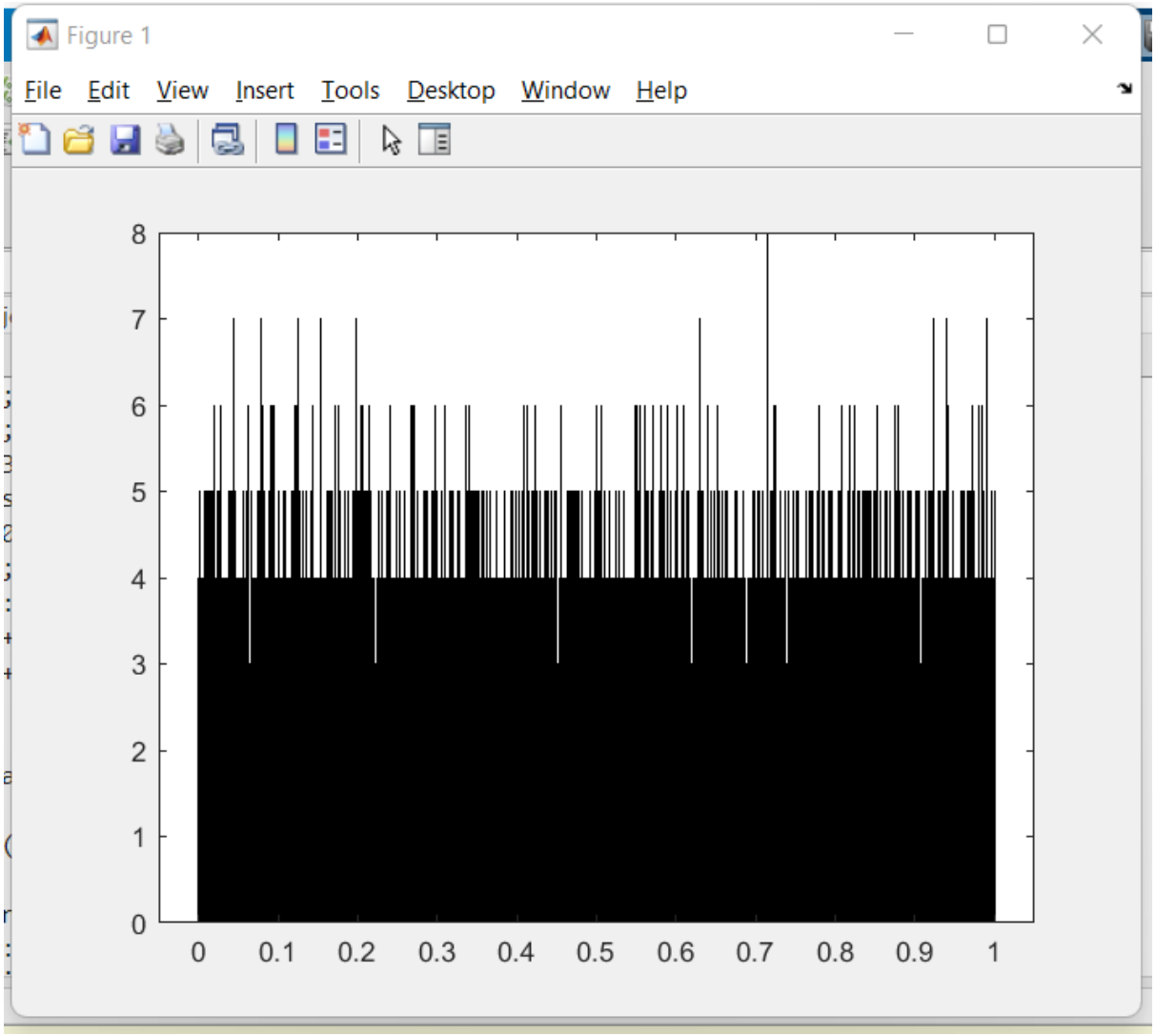

Рисунок 5 – Гистограмма, где размерность массива N равна 100000

```
M =
    0.5002

D =
    0.0838
```

Рисунок 6 – Полученное математическое ожидание и дисперсия

Рассматривая полученные данные (рисунки 5-6) затруднительно сделать вывод о том смог ли зациклиться датчик сложения. Результаты дисперсии и математического ожидания приблизительно равны теоретическим данным. Для более точного анализа работы датчика увеличим размерность последовательности до 1000000.

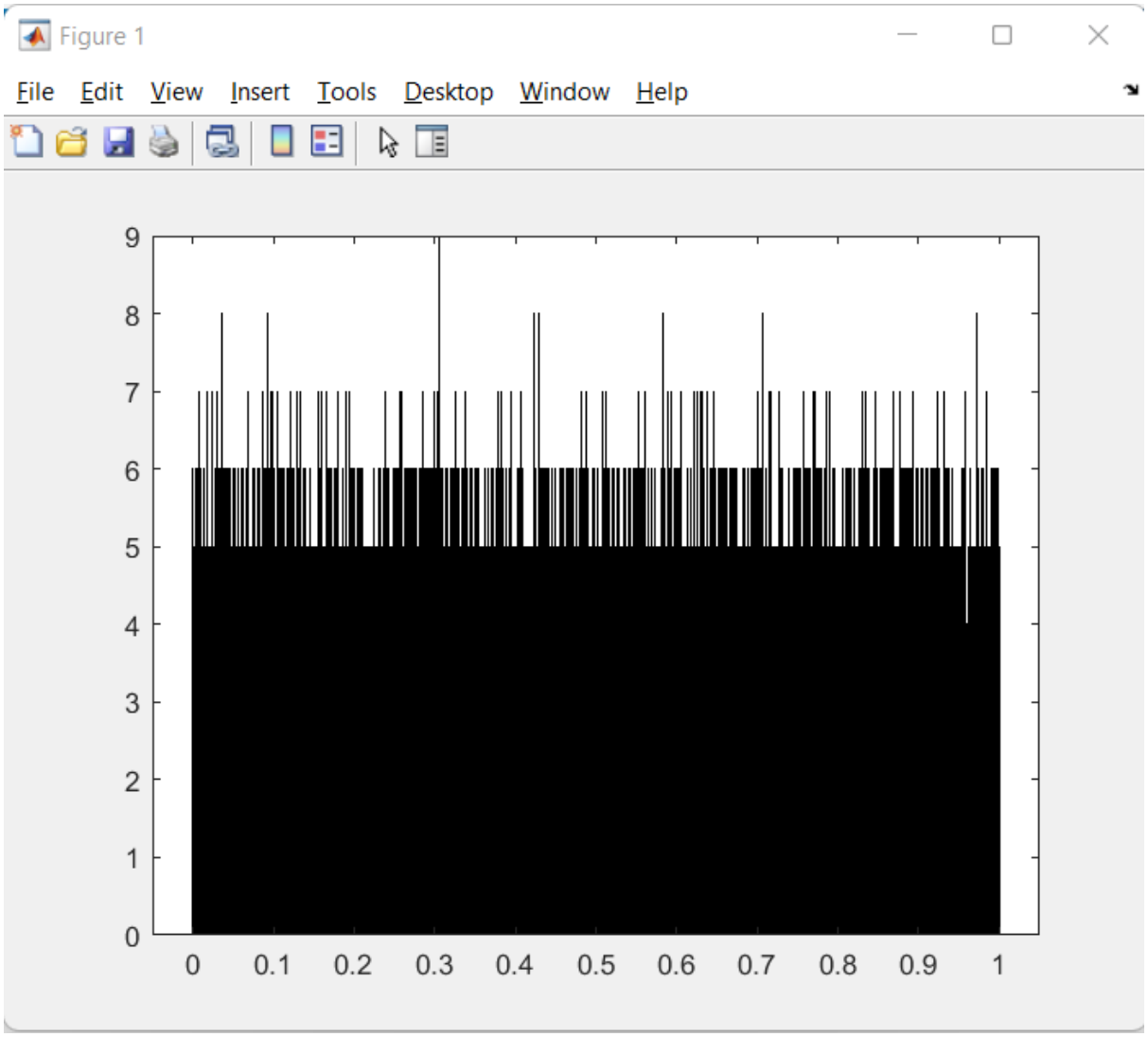

Рисунок 7 – Гистограмма, где размерность массива N равна 1000000

```
M =
    0.4999

D =
    0.0833
```

Рисунок 8 – Полученное математическое ожидание и дисперсия

Рассматривая полученные данные (рисунки 7-8) можно сделать вывод о том, что датчик сложения не зациклился. Математическое ожидание и дисперсия примерно равны теоретическим и меняются незначительно. В каждый интервал попадает одинаковое количество точек.

Листинг датчика на основе алгоритма умножения:

```
x = 565;
m = 56238423983;
arr = ones;
n = 1000;
u = 1/m;
for i=1:n
    arr(i+1) = mod(x*arr(i), m);
    u(i+1) = arr(i+1)/m;
end
figure
histogram(u,n);

M = sum(u)/n;
disp("M = "); disp(M);
Dr = zeros;
for i=1:n
    Dr(i) = (u(i)-M)^2;
end
D = sum(Dr)/n;
disp("D = "); disp(D);
```

Результат работы датчика умножения

Для построения гистограммы использовался массив N = 1000 при x = 565, m =56238423983.

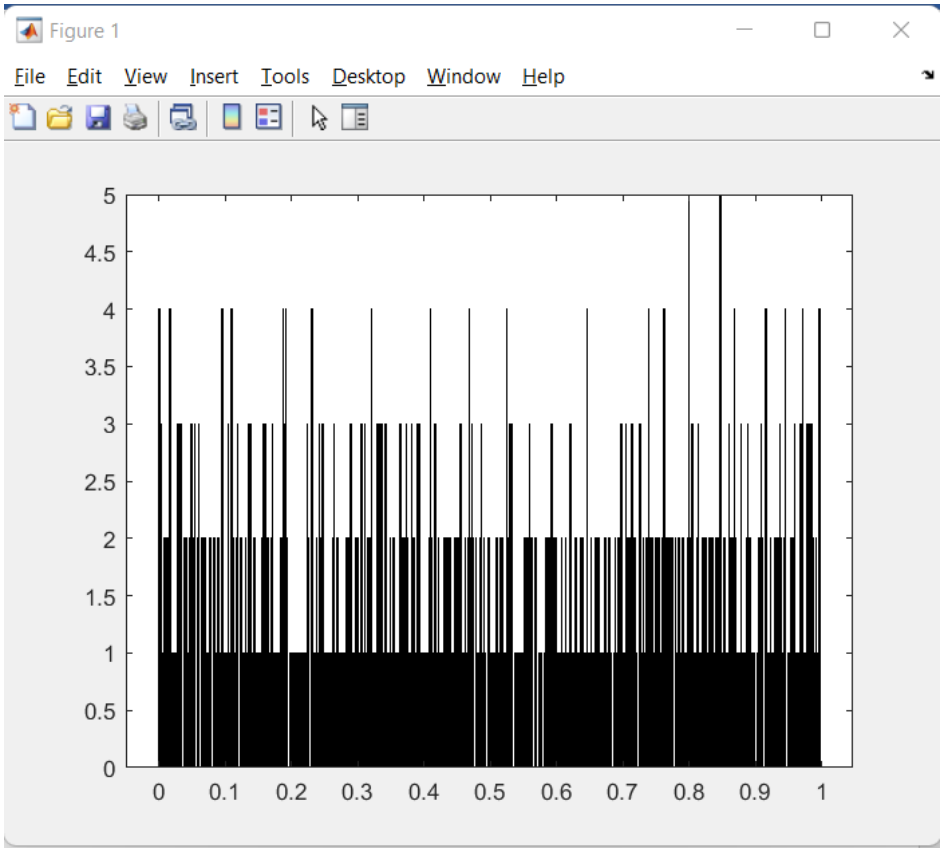

Рисунок 9 – Гистограмма, где размерность массива N равна 1000

```
M =
    0.5065

D =
    0.0861
```

Рисунок 10 – Полученные математическое ожидание и дисперсия

Анализируя, гистограмму на рисунке 9 заметим, что при данной размерности массива N невозможно прийти к точному выводу смог ли зациклиться датчик умножения. При этом рассчитанные математическое ожидание и дисперсия приблизительно равны теоретическим, поэтому для более точных результатов следует увеличить размерность N до 100000.

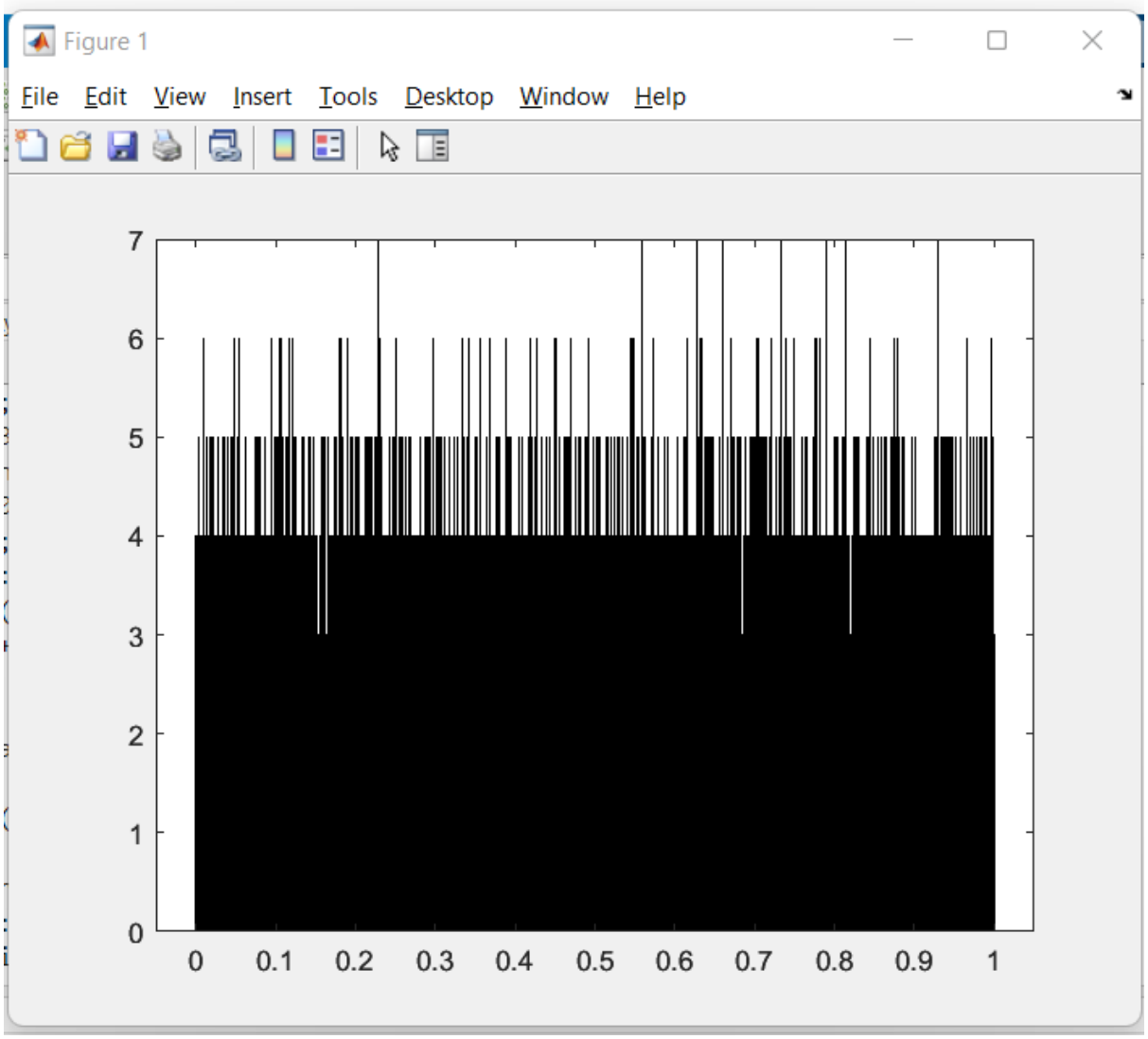

Рисунок 11 – Гистограмма, где размерность массива N равна 100000

```
M =
    0.5022

D =
    0.0834
```

Рисунок 12 – Полученные математическое ожидание и дисперсия

Изучая, гистограмму на рисунке 11 заметим, что при данной размерности массива N можно прийти к выводу, что данный датчик умножения не зациклился. Математическое ожидание и дисперсия примерно равны теоретическим и меняются незначительно.

Результат работы датчика, встроенного в Matlab

Для построения гистограммы использовался массив N = 1000.

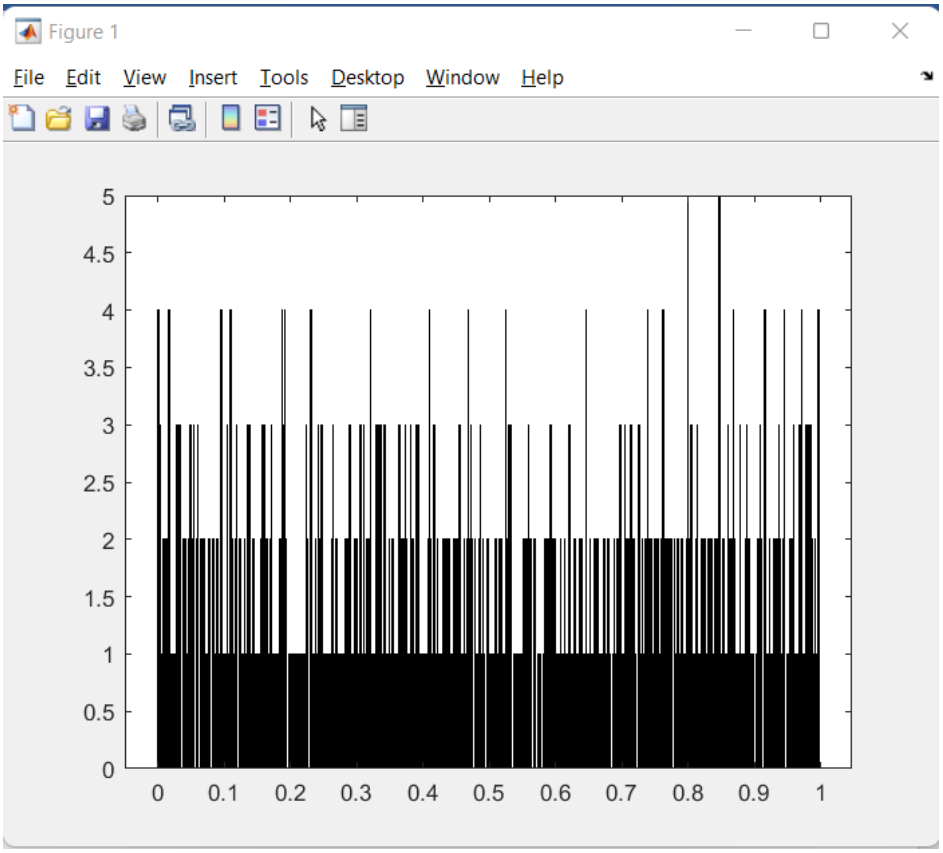

Рисунок 13 – Гистограмма размерностью 1000

```
M =
    0.5022

D =
    0.0834
```

Рисунок 14 – Математическое ожидание и дисперсия

Анализируя, гистограмму на рисунке 13 заметим, что при данной размерности массива N невозможно прийти к точному выводу смог ли зациклиться встроенный датчик rand(n). При этом рассчитанные математическое ожидание и дисперсия приблизительно равны теоретическим, поэтому для более точных результатов следует увеличить размерность N до 10000.

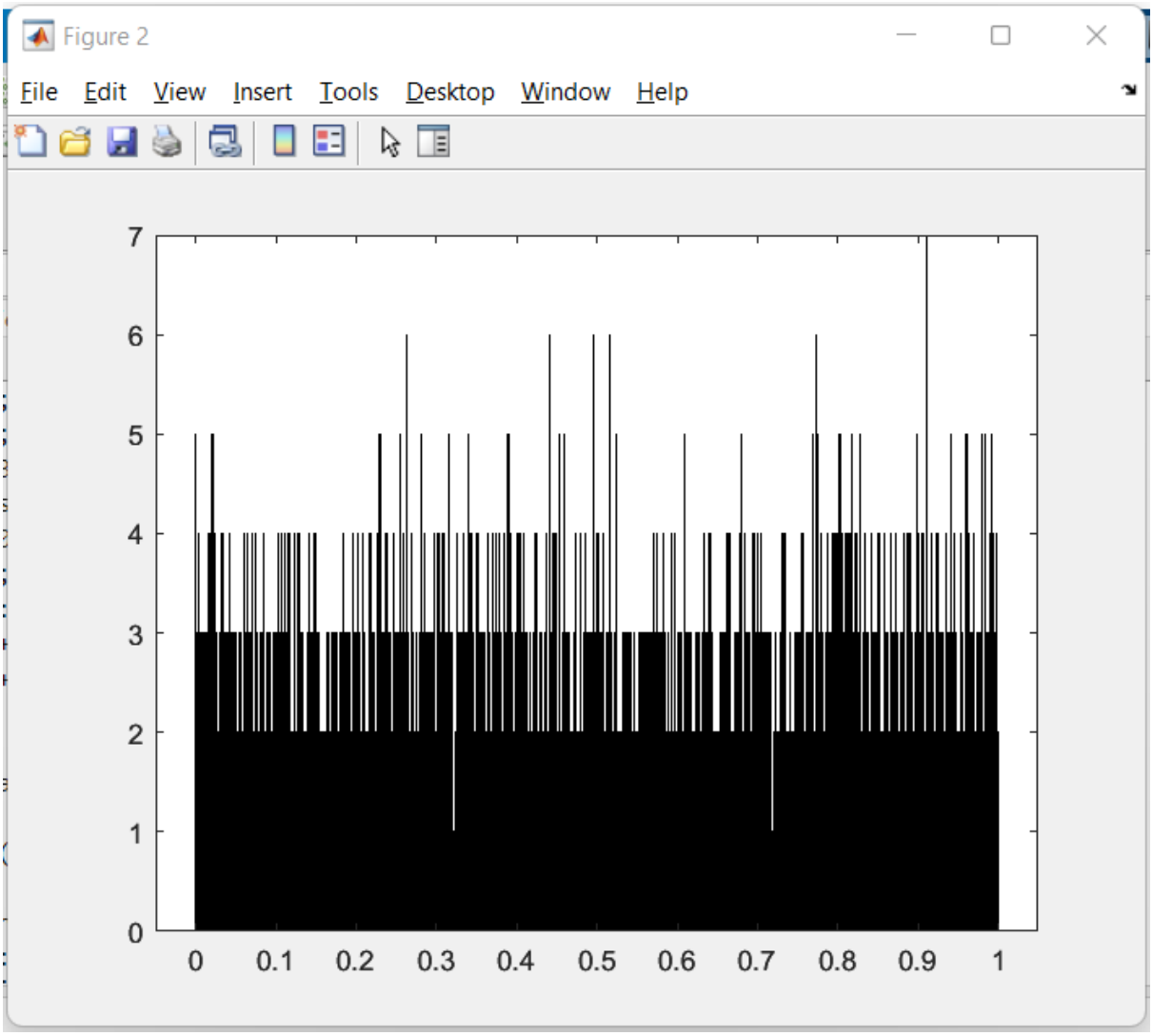

Рисунок 15 - Гистограмма размерностью 10000

```
M =
    0.5022

D =
    0.0834
```

Рисунок 16 – Математическое ожидание и дисперсия.

Анализируя, гистограмму на рисунке 15 заметим, что при данной размерности массива N невозможно прийти к точному выводу смог ли зациклиться встроенный датчик rand(n). При этом рассчитанные математическое ожидание и дисперсия приблизительно равны

теоретическим, поэтому для более точных результатов следует увеличить размерность N до 100000.

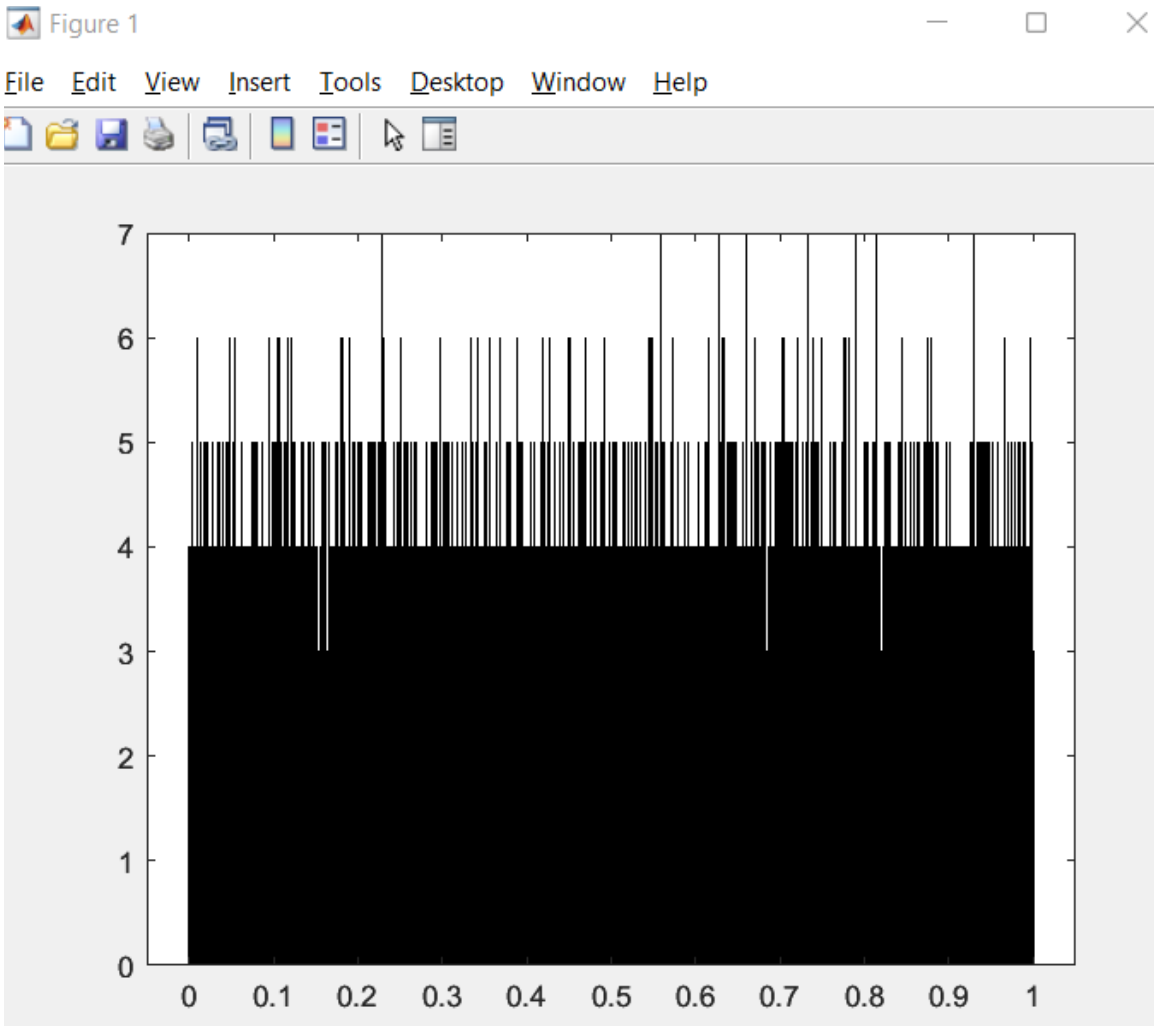

Рисунок 17 - Гистограмма размерностью 100000

```
M =
    0.5022

D =
    0.0834
```

Рисунок 18 - Математическое ожидание и дисперсия

Изучая, гистограмму на рисунке 17 заметим, что при данной размерности массива N можно прийти к выводу, что данный встроенный датчик rand(n) не зациклился. Математическое ожидание и дисперсия

примерно равны теоретическим и меняются незначительно. В каждый интервал попадает одинаковое количество точек.

Листинг датчика сложения на новом интервале значений:

```
a = 565;
c = 323;
m = 56238423983;
x = ones;
n = 1000;
u = 1/m;
Num = 24;
B = 2.5*Num;
g = zeros;

for i = 1:n
x(i+1) = mod(a*x(i) + c,m);
u(i+1) = x(i+1)/m;
g(i+1) = Num+(B-Num)*u(i+1);
end
figure
histogram(g,n);

M = sum(u)/n;
disp("M = "); disp(M);
Dr = zeros;
for i=1:n
    Dr(i) = (u(i)-M)^2;
end
D = sum(Dr)/n;
disp("D = "); disp(D);
```

Для построения гистограммы использовался массив N = 1000 при a = 565, c = 323, m = 56238423983 на интервале [24; 60].

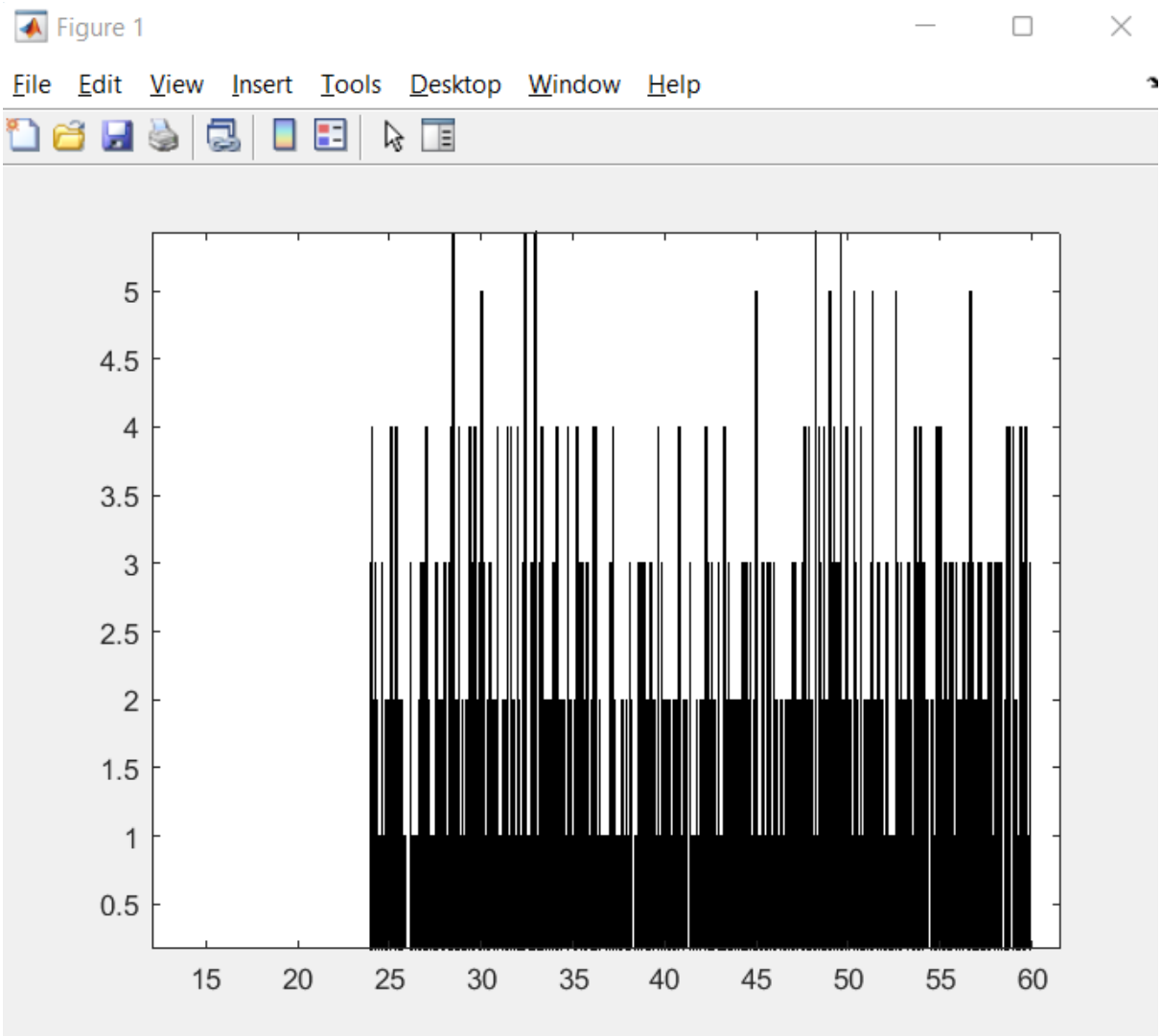

Рисунок 19 – Гистограмма, где размерность массива N равна 1000

```
M =
    0.5086

D =
    0.0852
```

Рисунок 20 – Полученные математическое ожидание и дисперсия

Анализируя, гистограмму на рисунке 19 заметим, что при данной размерности массива N невозможно прийти к точному выводу смог ли зациклиться датчик сложения на новых интервалах. Для более точных результатов следует увеличить размерность N до 10000.

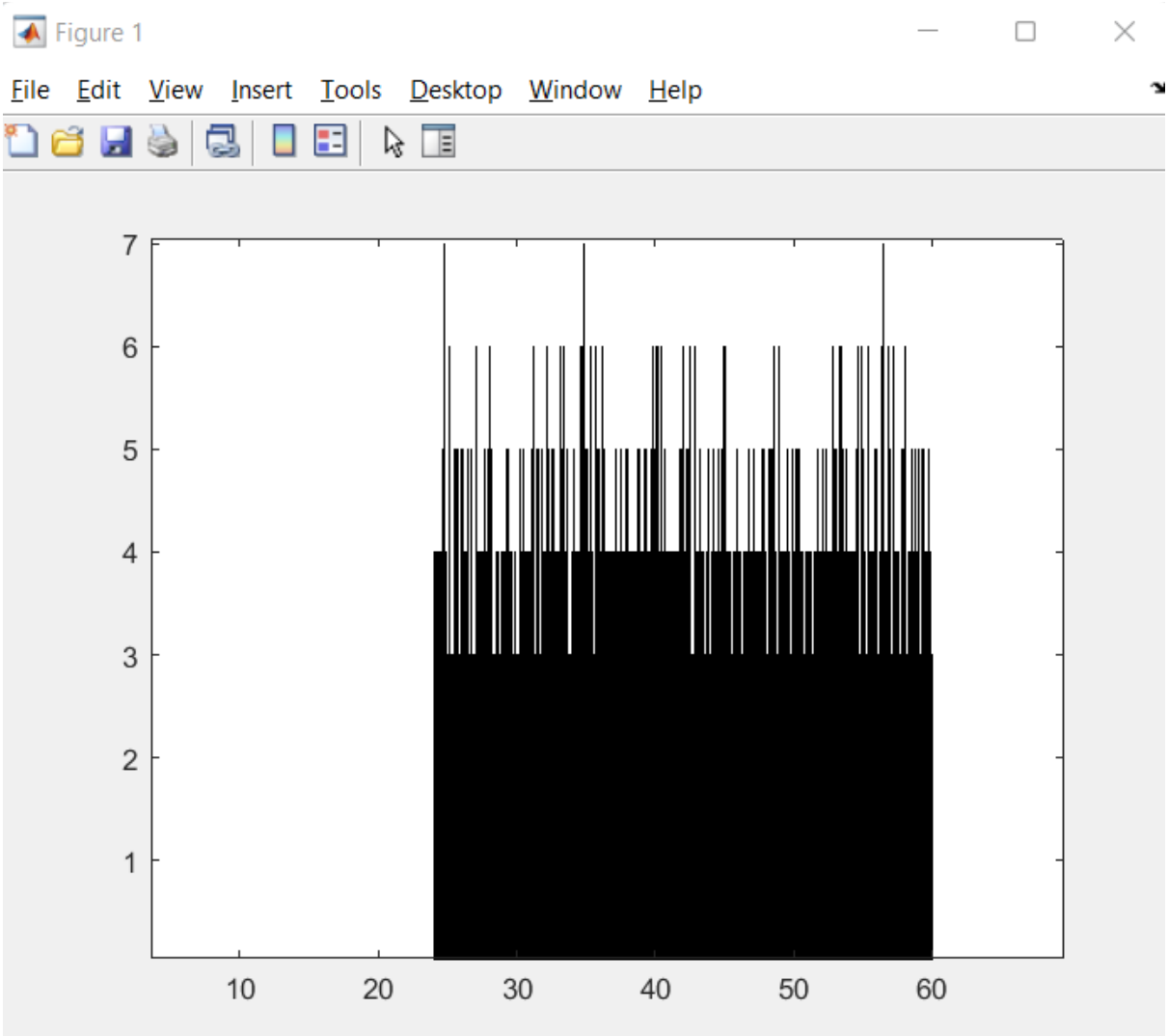

Рисунок 21 –Гистограмма, где размерность массива N равна 10000

```
M =
    0.5021

D =
    0.0835
```

Рисунок 22 – Полученные математическое ожидание и дисперсия

Анализируя, гистограмму на рисунке 21 заметим, что при данной размерности массива N невозможно прийти к точному выводу смог ли зациклиться датчик сложения на новых интервалах.

Вывод

В ходе выполнения лабораторной работы были изучены методы формирования случайных чисел при помощи различных датчиков псевдослучайных чисел на определенном интервале.

Была исследована работа двух алгоритмов формирования случайных чисел: сложения и умножения. В процессе увеличения количества формируемых случайных чисел оба датчика не зацикливаются. Чем больше производится выборка чисел, тем точнее вычисляются математическое ожидание и дисперсия. Это связано с тем, что все точки распределяются равномерно по интервалу случайных чисел. Поэтому эти методы можно назвать псевдослучайными, так как случайные числа, которые они образуют начинают повторяться.

Также в ходе выполнения был рассмотрен встроенный датчик rand(n), он также, как и собственные алгоритмы формирования при увеличении количества случайных чисел не зацикливается, что говорит об аналогичном алгоритме генерирования псевдослучайных чисел.

При увеличении интервала значений увеличивается математическое ожидание и дисперсия, при этом математическое ожидание становится больше дисперсии, так как при увеличении интервала увеличивается и разброс случайных чисел.

Контрольные вопросы к защите

1. Почему нельзя получить последовательность действительно случайных чисел программными методами?

2. Какие последовательности случайных чисел называют псевдослучайными?

3. Можно ли применять ПСП для формирования потоков случайных величин с числом значений, превышающих период ПСП?

4. На чем основан принцип построения датчика случайных чисел с равномерным распределением?

5. Каким образом достигается распределение значений случайных чисел датчика в интервале значений от 0 до 1? Можно ли изменить границы этого интервала?

6. Поясните, в чем состоит метод получения случайной величины с заданным законом распределения на основе нелинейного преобразования?

Каким образом число генерированных случайных величин влияет на степень приближения к заданному распределению вероятностей?