**Содержание**

[**Введение** 2](#_Toc90638394)

[**1. Методы обнаружения спонтанных изменений (разладки) наблюдаемых процессов** 3](#_Toc90638395)

[**2. Программа имитационного моделирования МА-алгоритма для гауссовского процесса** 9](#_Toc90638396)

[**3. Программа генерации случайных чисел** 15](#_Toc90638397)

[**4. Первоначальная апробация программы имитационного моделирования** 16](#_Toc90638398)

[**Разработка программы для определения заданного порога** 17](#_Toc90638399)

[**Выводы** 20](#_Toc90638400)

[**Список литературы** 21](#_Toc90638401)

[**Листинг программы** 22](#_Toc90638402)

### Программа имитационного моделирования МА-алгоритма для гауссовского процесса

Наблюдаемый временной ряд**:** статистически независимые дискретные значения *xi* с заданной функцией распределения вероятностей (с функцией плотности  или интегральным законом распределения вероятностей ) и фиксированными параметрами (например, математическим ожиданием или дисперсией ). Разладка состоит в скачкообразном изменении значения контролируемого параметра от начального значения («норма») до некоторого уровня, существенно отличающегося от «нормы»- состояние «разладки». Далее рассматривается наиболее часто встречающийся вариант разладки по математическому ожиданию , кода состоянию «норма» соответствует значение , а состоянию «разладка» - .

Контролирующий алгоритм: решающая функция МА-алгоритма задается следующим соотношением:

 (5)

Фактически здесь значения решающей функции являются результатом усреднения значений контролируемого временного ряда в скользящем окне шириной *N* (в стеке объемом *N*).

Решение о наличии разладки принимается, если значение решающей функции достигает или превышает величину решающего порога *Н*, т.е. при . Величина порога *Н* выбирается, исходя из требования обеспечить заданное значение среднего интервала между ложными тревогами . Когда речь идет о разладке по математическому ожиданию и  , для определения *Н* обычно используется теоретическое соотношение:

. (6)

При известной функции  определение порога *Н* в принципе не представляет никаких трудностей. Однако с практической точки зрения целесообразно перейти к нормированной форме записи решающей функции. Это связано с тем, что дисперсия решающей функции, определяемой формулой (5), зависит от значения *N*:



Такая зависимость приводит к тому, что для обеспечения одного и того же значения  величина решающего порога *Н* будет различной в зависимости от *N*. Для нормировки значений  их следует поделить на , в результате чего получим ту искомую форму записи решающей функции, которая и будет использоваться в дальнейшем:

 , (6)

где  - нормированные значения контролируемого временного ряда,

т.е. случайные числа с нормированным нормальным распределением.

Определение решающего порога *h* производится с помощью соотношения, аналогичного (6) по заданной величине :

. (7)

А) Задание исходных данных: среднего *теоретического* интервала между ложными тревогами , параметра *N*, определение решающей границы *h* по таблице:

Таблица 1.

Зависимость h от выбора 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 50 | 100 | 250 | 500 | 1000 |
| *h* | 2,054 | 2,326 | 2,652 | 2,878 | 3,090 |

Б) Задание параметра разладки (δ) – математическое ожидания *mX ≥*  0.

Суть работы алгоритма: *L –* кратный запуск процесса имитации работы МА-алгоритма. При каждом *j* – ом запуске (*j* = 1, 2, …, *L*) фиксируется номер такта *i*, на котором среднее значение данных, содержащихся в стеке из *N* значений,  достигло или превзошло решающую границу *h*: *g(i)* ≥ *h* . На этом *j* - ый запуск заканчивается. Зафиксированное значения *i* запоминается как *Tj*.

Последовательность операций на каждом *j* –ом запуске:

*Предварительная часть:*

1п) Генерируется *N* значений *x*1p, *x*2p,…,*xN*pc *mX =*  0 дисперсией = 1.

2п) Получение перенормированных значений контролируемого процесса

 и заполнение стека значениями , ,…, .

3п) Вычисление начальных значений решающей функции  и сравнение  с решающим порогом *h* :

* если для какого-либо *k* (*k* = 1,2,…,*N*) окажется , то предварительная часть реализуется повторно, начиная с пункта 1п);
* если все , то предварительная часть завершается и осуществляется переход к основной части алгоритма.

\* *Примечание:* *в стеке из N значений на момент перехода к основной части будет содержаться следующий набор значений:* , ,…, 

*Основная часть:*

На каждом такте *i* основной части (*i* = 1, 2,….):

1. Генерация значения *xi* c заданным значением *mX* ≥ 0 и = 1;

Варианты задания *mX* : *mX* =0 для определения оценки  ;

*mX* = 0,5; 1,0; 1,5; 2,0; 2,5; 3,0 –для определения  при различных величинах разладки.

1. Получение очередного перенормированного значения 
2. Обновление содержимого стека: самое раннее значение из стека исключается; значение  помещается на последнее место в стеке;

\* *Примечание: например,* *в стеке из N значений в момент после появления значения x1 и обновления стека будет содержаться следующий набор значений:* ,…,, ;

1. Вычисление текущего значения решающей функции на *i*-ом такте:

*gi* = сумма *N* значений, содержащихся в стеке;

1. Сравнение значения *gi* с решающим порогом *h* :

* если *gi* <*h*, то перейти к следующему (*i* +1)-му такту данного *j*-го запуска;
* если *gi* ≥*h*, то данный *j*-ый запуск завершается, фиксируется номер такта *i* , запоминается значение *Тj* , равное зафиксированному значению *i*;
* после этого перейти к следующему (*j* +1)-му запуску, начиная с предварительной части.

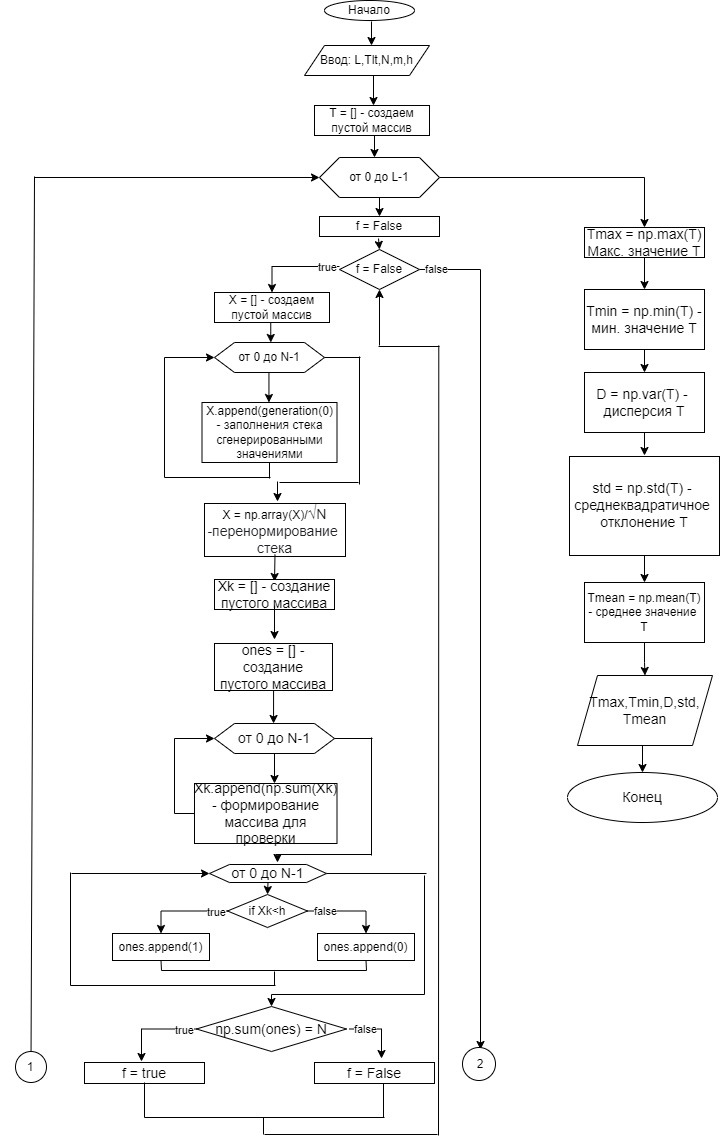
Имитационный эксперимент заканчивается при завершении *L*-го запуска. Результаты эксперимента: значения *Тj* (*j* = 1, 2, …, *L*).

\* *Примечание:* *рекомендуемое значение L* = 10000;

*рекомендуемые значения N*: 1, 2, 4, 8, 16, 32.

В результате работы программы получаем следующие показатели: минимального и максимального значений *Тj*, вычисление среднего значения, дисперсий и среднеквадратического отклонения.

Структура программы имитационного моделирования выглядит следующим образом:



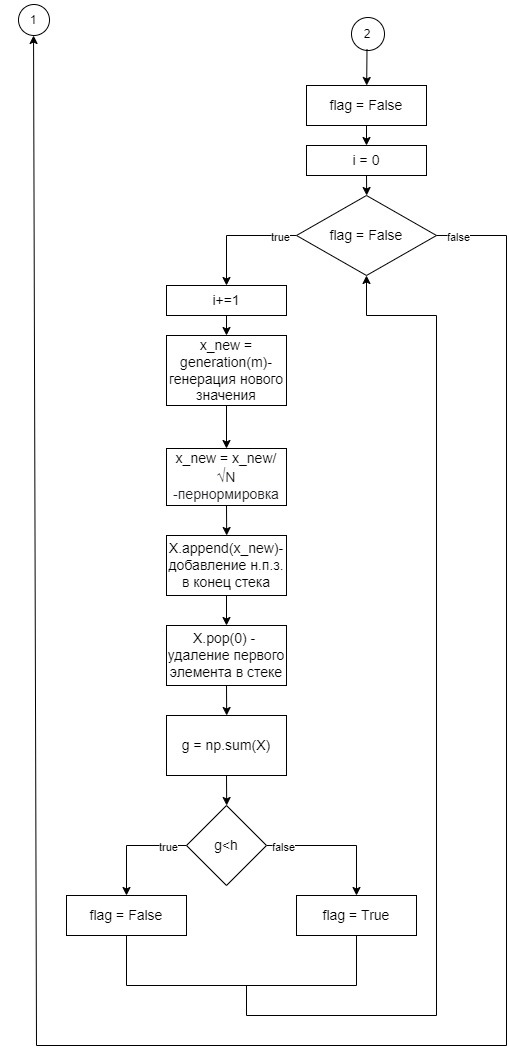


Рис.2 Блок-схема программы имитационного моделирования МА-алгоритма

Отметим, что важной частью программы имитационного моделирования является генерация случайный чисел методом Бокса-Маллера. Преимущество данного метода заключается в том, что он позволяет использовать широкий диапазон изменения значений.

### Программа генерации случайных чисел

Программа генерации случайных чисел *xi* с нормированным нормальным распределением (Метлд Бокса-Маллера)

1. Генерация на каждом такте *i* двух случайных чисел с равномерным распределением на интервале [0,1]: u1*i*;u2*i*
2. Формирование значений *xi*



1. При наличии разладки к *xi* добавляется значение *mX* из указанного ряда значений (*mX* = 0,5; 1,0; 1,5; 2,0; 2,5; 3,0).

Структура программы генерация случайный чисел методом Бокса-Маллера выглядит следующим образом:

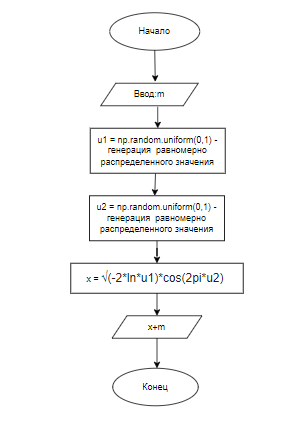


Рис.3 Блок-схема функции генерации случайных чисел.

### Первоначальная апробация программы имитационного моделирования

При первоначальной апробации программы имитационного моделирования получили следующие результаты для частного случая (Тлт=100), показаны в таблице 2:

Таблица 2.

Результаты программы при параметрах: L = 10000, h = 2,326, Тлт =100

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N |  | | | | | | |
| *δ* = 0 | *δ* = 0,5 | *δ* = 1 | *δ* = 1,5 | *δ* = 2 | *δ* = 2,5 | *δ* = 3 |
| 1 | 101.548 | 29.795 | 10.842 | 4.936 | 2.652 | 1.756 | 1.351 |
| 2 | 111.618 | 23.782 | 7.958 | 3.729 | 2.400 | 1.862 | 1.517 |
| 4 | 148.310 | 21.584 | 7.052 | 3.825 | 2.750 | 2.376 | 2.067 |
| 8 | 222.587 | 20.824 | 7.468 | 4.882 | 3.787 | 3.161 | 2.711 |
| 16 | 367.04 | 22.457 | 9.631 | 6.668 | 5.143 | 4.253 | 3.636 |
| 32 | 633.932 | 26.608 | 13.418 | 9.279 | 7.141 | 5.831 | 4.955 |

Полученные результаты показывают, что вместо ожидаемых значений при отсутствии разладки имеет место нарастаний, т.к. при *δ* = 0 мы должны получить оценку Тлт, примерно равную 100. По всей видимости, увеличение Тлт с ростом *N* связано с тем, что значения решающей функции коррелированы. Таким образом появляется задача: найти зависимость порога *h* от количества значений в стеке *N* для различных значений параметра Тлт*.*

### Разработка программы для определения заданного порога

Для решения задачи, выявленной при первоначальной апробации программы имитационного моделирования предлагается использовать алгоритм шагового поиска для определения порога *h* в зависимости от количества значений в стеке (N).

Описание алгоритма:

Вводим параметры *L*, Tлт, *m, a, k*, *h0*, где *a* – шаг изменения порога *h*, *k* – максимальная допустимая разница между фактическим Тлт и введенным, *h0* – начальное значение порога *h*, взятое из таблицы 1.

Вычисляем фактическое значение Tлт для текущего *h* (на первой итерации *h* = *h0*). Сравниваем его с входным Tлт. Если различия превосходят *k* по модулю, изменяем *h* на величину шага, затем вычисляем фактическое значение Tлт до тех пор, пока не будет достигнута заданная точность.

Структура данной программы выглядит следующим образом:

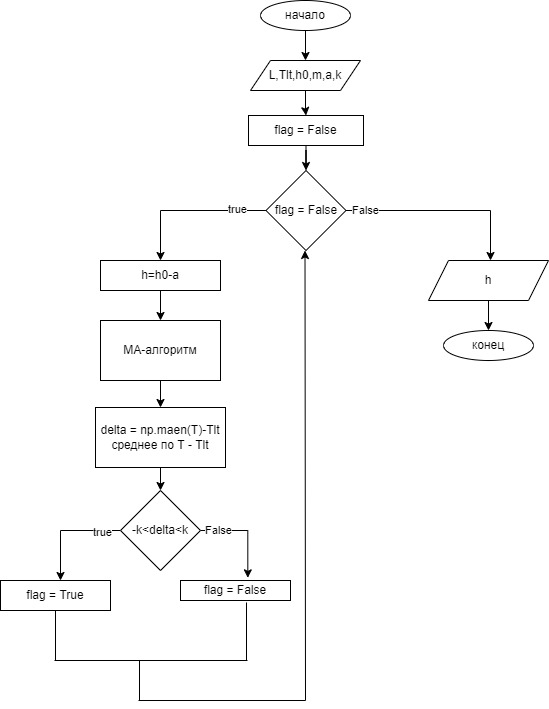


Рис.4 Блок-схема программы для определения зависимости *h*(*N*)

С помощью данной программы собрали справочную информацию о зависимости *h*(*N*) для разных значений Тлт:

Таблица 3.

Зависимости *h*(*N*) для разных значений Тлт

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *N* | *h*  Тлт *= 50* | *h*  Тлт *=* 100 | *h*  Тлт *= 250* | | *h*  Тлт *= 500* | *h*  Тлт *= 1000* |
| 1 | 2.054 | 2.326 | 2.652 | 2.878 | | 3.09 |
| 2 | 1.998 | 2.281 | 2.621 | 2.855 | | 3.071 |
| 3 | 1.917 | 2.221 | 2.575 | 2.816 | | 3.049 |
| 4 | 1.834 | 2.158 | 2.534 | 2.781 | | 3.019 |
| 5 | 1.769 | 2.103 | 2.489 | 2.745 | | 2.99 |
| 6 | 1.709 | 2.051 | 2.447 | 2.709 | | 2.957 |
| 7 | 1.641 | 2.001 | 2.411 | 2.674 | | 2.923 |
| 8 | 1.594 | 1.959 | 2.374 | 2.646 | | 2.889 |
| 9 | 1.532 | 1.911 | 2.35 | 2.615 | | 2.856 |
| 10 | 1.492 | 1.874 | 2.308 | 2.589 | | 2.831 |
| 11 | 1.441 | 1.836 | 2.278 | 2.571 | | 2.807 |
| 12 | 1.396 | 1.8 | 2.262 | 2.546 | | 2.785 |
| 13 | 1.361 | 1.758 | 2.226 | 2.526 | | 2.766 |
| 14 | 1.329 | 1.729 | 2.195 | 2.494 | | 2.756 |
| 15 | 1.271 | 1.69 | 2.169 | 2.471 | | 2.741 |
| 16 | 1.239 | 1.677 | 2.139 | 2.446 | | 2.713 |

1. **Результаты анализа статистических характеристик МА-алгоритма**

Для каждого значения среднего интервала между ложными тревогами получены результаты для различных значений разладки.

Характерный пример результатов для Тлт = 100:

Таблица 3.

Результаты имитационного моделировании при Тлт = 100

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | *δ* = 0 | *δ* = 0.5 | *δ* = 1 | *δ* = 1,5 | *δ* = 2 | *δ* = 2.5 | *δ* = 3 |
| 1 | 100.008 | 29.491 | 10.786 | 4.936 | 2.703 | 1.764 | 1.328 |
| 2 | 100.445 | 22.291 | 7.422 | 3.618 | 2.35 | 1.846 | 1.603 |
| 3 | 100.255 | 18.942 | 6.473 | 3.49 | 2.499 | 2.054 | 1.804 |
| 4 | 100.542 | 17.039 | 6.008 | 3.545 | 2.692 | 2.248 | 1.972 |
| 5 | 100.32 | 15.888 | 5.884 | 3.699 | 2.891 | 2.401 | 2.095 |
| 6 | 99.834 | 14.957 | 5.904 | 3.871 | 3.05 | 2.554 | 2.223 |
| 7 | 100.359 | 14.642 | 5.927 | 4.075 | 3.185 | 2.659 | 2.341 |
| 8 | 100.88 | 14.326 | 6.133 | 4.236 | 3.343 | 2.774 | 2.407 |
| 9 | 100.498 | 13.957 | 6.21 | 4.406 | 3.427 | 2.87 | 2.478 |
| 10 | 100.466 | 14.042 | 6.47 | 4.502 | 3.554 | 2.974 | 2.575 |
| 11 | 100.461 | 13.977 | 6.583 | 4.661 | 3.676 | 3.042 | 2.64 |
| 12 | 100.259 | 13.809 | 6.73 | 4.742 | 3.766 | 3.145 | 2.688 |
| 13 | 99.608 | 13.738 | 6.835 | 4.879 | 3.852 | 3.182 | 2.745 |
| 14 | 99.983 | 13.488 | 6.99 | 4.979 | 3.919 | 3.265 | 2.787 |
| 15 | 99.945 | 13.582 | 7.097 | 5.115 | 4 | 3.28 | 2.846 |
| 16 | 100.817 | 13.83 | 7.291 | 5.202 | 4.051 | 3.373 | 2.914 |

### Выводы

В ходе данной работы рассмотрена литература по проблематике обнаружения разладки временных рядов, составлен краткий обзор, реализована программа имитационного моделирования, основанная на МА-алгоритме.

В процессе апробации программы имитационного моделирования введено дополнительное условие: собрать информацию о зависимости *h*(*N*) при различных значениях параметра Тлт, для практического использования данного алгоритма.

Полученные результаты позволяют найти оптимальное значение параметров, для того чтобы минимизировать среднее время задержки при зафиксированном среднее значение интервала между ложными тревогами.

Полученные данные будут использоваться в дальнейшей работе исследования алгоритма. Задание на НИР выполнено в полном объеме.

### Список литературы

|  |
| --- |
| 1 Мердок Дж. Контрольные карты /Пер с англ. М.: Финансы и статистика, 1986. – 151с |
| 2 Адлер Ю.П., Максимова О.В., Шпер В.Л. Контрольные карты Шухарта в России и за рубежом: краткий обзор современного состояния (статистические аспекты). – Журнал «Стандарты и качество», июль–август, 2011. |

### Листинг программы

#Подключение библиотек  
import pandas as pd  
import numpy as np  
import random  
import math  
from tqdm import tqdm  
  
#задание исходных данных:  
print("Введите кол-во итераций (L):")  
L = int(input())  
f = False  
print("Возможные значения для среднего теоритического интервала:50,100,250,500,1000")  
Tlt = int(input('Введите значение среднего теоретического интервала между ложными тревогами: '))  
 if Tlt not in [50,100,250,500,1000]:  
 f = False  
 print("Ошибка!Введите значение из предоженного списка!")  
 print()  
 else:  
 f = True  
N = int(input("Кол-во значений в стеке: "))  
  
h = int(input("Введите значение решающей границы: "))  
   
#задание параметра разладки  
fm = False  
while(fm == False):  
 print("Возможные значения мат.ожидания: 0; 0,5; 1,0; 1,5; 2,0; 2,5; 3,0")  
 m = float(input("Введите математическое ожидание: "))  
 if m not in [0.0,0.5,1.0,1.5,2.0,2.5,3.0]:  
 fm = False  
 print("Ошибка!Введите значение из предоженного списка!")  
 print()  
 else:  
 fm = True  
  
# генерация числа  
def generation(m):  
 u1 = np.random.uniform(0,1)  
 u2 = np.random.uniform(0,1)  
 x = math.cos(2\*math.pi\*u2)\*(-2\*math.log(u1))\*\*(0.5)  
 return round(x+m,4)  
  
#программа  
T = []  
#pbar = tqdm(np.arange(0,L))  
for j in np.arange(0,L):  
 f = False  
 while(f == False):  
 X = []  
 for i in range(N):  
 X.append(generation(0))  
 X = np.array(X)/(N\*\*(1/2))  
 X = X.tolist()  
 Xk = []  
 ones = []  
 for k in range(0,N):  
 Xk.append(X[k])  
 g0 = np.sum(Xk)  
 if g0 >= h:  
 ones.append(0)  
 else:  
 ones.append(1)  
 if np.sum(ones) == N:  
 f = True  
 else:  
 f = False  
   
 # основная часть:  
 flag = False  
 i = 0  
 while(flag == False):  
 i+=1  
 x\_new = generation(m)  
 x\_new = x\_new/(N\*\*(1/2))  
 X.append(x\_new)  
 X.pop(0)  
 g = np.sum(X)  
 if g < h:  
 flag = False # переходим к i+1 такту  
 else:   
 flag = True # j - заканчивается, фикс номер i и Tj = i  
   
 T.append(i)  
# pbar.set\_description(f'Tmean = {np.mean(T):.4f}')  
  
# обработка результатов:  
Tmax = np.max(T)  
Tmin = np.min(T)  
Tmean = np.mean(T)  
D = np.var(T)  
std = np.std(T)  
# вывод результатов:  
dict\_table = {"N":N,"h":h,"Мин.знач.":Tmin,"Макс. знач.":Tmax,"Cред.знач.":Tmean,"Дисперсия":D,  
 "Cреднекв. знач":std}  
table = pd.DataFrame(dict\_table,index = [0])  
table.set\_index('N', inplace=True)  
print(table.head())