|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** |

|  |  |
| --- | --- |
| **Институт** | ИВТИ |
| **Кафедра** | УИТ |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(бакалаврская работа)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Направление** | | | | 27.03.04 Управление в технических системах | | | | | | | | | | |
|  | | | | | (код и наименование) | | | | | | | | | |
| **Образовательная программа** | | | | | | | | | Системы и технические средства автоматизации и управления | | | | | |
| **Форма обучения** | | | | | | | | очная | | | | | | |
|  | | | | | | | | (очная/очно-заочная/заочная) | | | | | | |
| **Тема:** | Разработка МА-метода обнаружения разладки гауссовского временнóго ряда | | | | | | | | | | | | | |
| **Студент** | | | А-02-18 | | | | | | |  | | | Локтюшов В.А. | |
|  | | группа | | | | | | | | подпись | | | фамилия и инициалы | |
| **Руководитель ВКР** | | | | | | | д.т.н. | | профессор | | | |  | Филаретов Г.Ф. |
|  | | | | | | | уч. степень | | должность | | | | подпись | фамилия и инициалы |
| **Консультант** | | | | | |  | | |  | | | |  |  |
|  | | | | | | уч. степень | | | должность | | | | подпись | фамилия и инициалы |
| **Внешний консультант** | | | | | |  | | |  | | | |  |  |
|  | | | | | | уч. степень | | | должность | | | | подпись | фамилия и инициалы |
|  | | | | | | | | | | | | | | |
| организация | | | | | | | | | | | | | | |
| **«Работа допущена к защите»** | | | | | | | | | | | | | | |
| **Заведующий кафедрой** | | | | | д.т.н. | | | | | | доцент |  | | Бобряков А.В. |
|  | | | | | уч. степень | | | | | | звание | подпись | | фамилия и инициалы |
|  | | | | | | | | | | | | **Дата** | |  |

**Москва, 2022**

|  |
| --- |
| **МИНОБРНАУКИ РОССИИ**  федеральное государственное бюджетное образовательное  учреждение высшего образования  **«Национальный исследовательский университет «МЭИ»** |

|  |  |
| --- | --- |
| **Институт** | ИВТИ |
| **Кафедра** | УИТ |

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ (бакалаврскую работу)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Направление** | | | Управление в технических системах (27.03.04) | | | |
| (код и наименование) | | | | | | |
| **Направленность (профиль)** | | | | | |  |
| Управление и информатика в технических системах | | | | | | |
|  | | | | | | |
| **Форма обучения** | | | | | **Очная** | |
|  | (очная/очно-заочная/заочная) | | | | | |
| **Тема:** | **Разработка МА**-**метода обнаружения разладки гауссовского временнóго ряда** | | | | | |
|  | | | | | | |
| **Студент** | | **А-02-18 Локтюшов В.А.** | | | | |
| группа подпись фамилия и инициалы | | | | | | |
| **Научный руководитель** | | | | **д.т.н профессор Филаретов Г.Ф.** | | |
| уч. степень должность подпись фамилия и инициалы | | | | | | |
| **Консультант** | | | |  | | |
| уч. степень должность подпись фамилия и инициалы | | | | | | |
| **Консультант** | | | |  | | |
| уч. степень должность подпись фамилия и инициалы | | | | | | |
| **Зав. кафедрой** | | | | **д.т.н доцент Бобряков А.В.** | | |
| уч. степень звание подпись фамилия и инициалы | | | | | | |
|  | | | | | | |
| **Место выполнения работы: Кафедра Управления и интеллектуальных технологий** | | | | | | |

**СОДЕРЖАНИЕ РАЗДЕЛОВ ЗАДАНИЯ И ИСХОДНЫЕ ДАННЫЕ**

1. Изучение литературы по проблематике обнаружения разладки случайных процессов, известным методам обнаружения и их классификации. Составление краткого обзора.
2. *МА (Moving Average)* – алгоритм: назначение, структура, принцип действия и основные характеристики. Программная реализация и апробация.
3. Имитационное моделирование как инструмент исследования МА-алгоритма. Разработка программных средств имитационного моделирования (ПО ИМ-МА). Планирование имитационного эксперимента для исследования варианта разладки гауссовского процесса по математическому ожиданию.
4. Реализация имитационного эксперимента и обработка его результатов.
5. Сопоставление *МА*-алгоритма с другими известными алгоритмами обнаружения разладки по показателю эффективности.

6. Анализ и обсуждение полученных результатов.

7. Написание и оформление работы.

**Исходные данные:**

1.Задача обнаружения разладки и алгоритмы ее решения - описание.

2. Экспериментальные данные для сопоставительного анализа.

**ПЕРЕЧЕНЬ ГРАФИЧЕСКОГО МАТЕРИАЛА**

|  |  |
| --- | --- |
| **Количество слайдов в презентации** | **5** |
|  |  |

1. Наименование работы. Постановка задачи исследования.
2. Алгоритмы решения задачи обнаружения разладки и их классификация. МА(Moving Average) – алгоритм и его особенности.
3. Имитационный эксперимент: назначение, структура ПО ИМ-МА, планирование имитационного эксперимента.
4. Результаты имитационного эксперимента (основные характеристики МА-алгоритма).
5. Сопоставительный анализ МА-алгоритма. Выводы.

**РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА**

|  |
| --- |
| 1. Мердок Дж. Контрольные карты /Пер с англ. М.: Финансы и статистика, 1986. – 151с |
| 1. 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. |
| 1. Программная система STATISTIKA: Электронный ресурс: [**StatSoft-statistica.ru**](https://statsoft-statistica.ru/) |

АННОТАЦИЯ

В работе рассмотрены вопросы, связанные с решением задачи обнаружения спонтанного изменения (разладки) вероятностных свойств временного ряда в реальном масштабе времени.Предметом рассмотрения является последовательный параметрический алгоритм обнаружения, основанный на методе скользящего среднего: «*Moving Average*» или «*МА*-алгоритм». Целью исследования является определение статистических характеристик МА-алгоритма для обеспечения синтеза подходящей контролирующей процедуры.

Работа состоит из введения, шести разделов, заключения, списка литературы из 6 наименований, одного приложения, содержит 40 страниц текста, 8 рисунков, 12 таблиц.

|  |
| --- |
|  |

**Содержание**

[**Введение 5**](#_Toc105242131)

[**1. Алгоритмы обнаружения спонтанных изменений (разладки) наблюдаемых процессов 6**](#_Toc105242132)

[**2. Алгоритм имитационного моделирования МА-алгоритма для гауссовского процесса 12**](#_Toc105242133)

[**3. Экспериментальное определение заданного порога обнаружения 21**](#_Toc105242134)

[**4. Разработка программы для определения порогов *Н* для различных *N* 22**](#_Toc105242135)

[**5. Результаты анализа статистических характеристик *МА*-алгоритма 25**](#_Toc105242136)

[**6. Сопоставительный анализ *МА*-алгоритма 26**](#_Toc105242137)

[**Заключение 30**](#_Toc105242138)

[**Список литературы 32**](#_Toc105242139)

[**Приложение: Листинг программы 33**](#_Toc105242140)

### Введение

Данная работа посвящена разладкевременного рядаи исследованию одного из методов ее обнаружения. Под разладкой понимается спонтанное изменение характеристик наблюдаемого случайного временного ряда, происходящее в неизвестный, заранее непредсказуемый момент времени.

Предметом обсуждения является алгоритм, основанный на методе скользящего среднего: «*Moving Average*» или «*МА*-алгоритм». Целью исследования является определение статистических характеристик и возможного синтеза контролирующего алгоритма. Следовательно, требуется собрать информацию для практического использования данного алгоритма.

Реализация метода является программа имитационного моделирования временного ряда, с помощью которого определяются основные характеристики алгоритма.

### Алгоритмы обнаружения спонтанных изменений (разладки) наблюдаемых процессов

Разладка стохастических процессов – произвольное изменение параметров случайного процесса, происходящее в неизвестный момент времени.

Задача обнаружения «разладки» заключается в том, чтобы как можно быстрее и точнее определить время, в котором происходит произвольное изменение параметров стохастического процесса.

В настоящее время методы решения данной задачи активно изучаются. Это связано с потребностью в их использовании в контролирующих системах и системах мониторинга. Приведем несколько примеров использования данной задачи на практике: контроль или мониторинг нужных параметров на станках или другом оборудование, отслеживание определенных сигналов, после которых нужно вводить какое-либо воздействие. Данная задача имеет место в сейсмологии, когда необходимо быстро обнаружить сам факт землетрясения с тем, чтобы предотвратить определенные нежелательные последствия, которые могут повлечь за собой материальные потери.

Исходя из вышеуказанного, уточним постановку задачи: наблюдается последовательность случайных значений, вероятностные свойства которой изменяются в некоторый неизвестный момент времени. Требуется как можно быстрее, т.е. с минимальным запаздыванием обнаружить такое изменение. Следовательно, алгоритм решения данной задачи должен удовлетворять определенным требованиям.

Существуют два подхода к решению задачи: апостериорный и последовательный. Если говорить про первый, то при таком варианте записывается временной ряд, далее данный ряд подвергается обработке, при которой как раз и оценивается момент появления разладки. Апостериорный подход не рассматривается в данной работе, так как перед нами стоит задача нахождения разладки стохастического временного ряда в режиме реального времени. Именно последовательный подход решает данную задачу. Стоит отметить, что алгоритм, реализующий последовательный подход должен соответствовать определенным требованиям: иметь минимальное среднее значение запаздывания в обнаружение разладки , c другой стороны требуется максимизировать среднее значение интервала между ложными тревогами .

Перечисленные требования противоречивы. Для решения данной проблемы нужно зафиксировать средний интервал между ложными тревогами и добиться минимизации среднего значения запаздывания в обнаружение разладки. Таким образом, будут учтены обе характеристики.

Рассмотрим классификацию последовательных алгоритмов, предназначенных для решения разладки стохастического временного ряда. Выделяют четыре основные группы: байесовские, эвристические, алгоритмы, основанные на подходе Неймана-Пирсона, алгоритмы, базирующиеся на идеях последовательного анализа.

В байесовских алгоритмах минимизируется средний риск, складывающийся из потерь от наличия необнаруженной разладки или ложных обнаружений.

Эвристические алгоритмы – это такие алгоритмы, в которых используются интуитивные представления о наилучшей организации обнаружения разладки и удобные при практической реализации; с математической точки зрения не полностью обоснованы. В практическом применении с их помощью иногда получают в достаточной мере хорошие результаты, хотя в некоторых случаях эвристические алгоритмы могут давать неверные результаты.

Алгоритмы, основанные на подходе Неймана-Пирсона – это, фактически, алгоритмы, использующие классические критерии проверки гипотез. В них осуществляется многократное повторение процедуры проверки гипотез относительно вида функции распределения вероятностей или какого-либо конкретного значения параметра - например: математического ожидания, дисперсии и т.д. Наиболее известный вариант такого алгоритма – это, так называемые, контрольные карты Шухарта, предназначенные для отслеживания изменчивости некоторых параметров и для решения задачи стабильности технологического процесса.

Алгоритмы, базирующиеся на идеях последовательного анализа, основаны на использовании статистики отношения правдоподобия, но в отличии от классической процедуры в ней реализуется операция отражения от нижней решающей границы.

Если говорить о практическом применении, то чаще всего используют алгоритмы, основанные на подходе Неймана-Пирсона или базирующиеся на идеях последовательного анализа. В их числе

- классический алгоритм Шухарта (карта Шухарта);

- алгоритм экспоненциального сглаживания или *EWMA*-алгоритм;

- алгоритм кумулятивных сумм (АКС или *CUSUM*-алгоритм);

- алгоритм скользящего среднего – «*Moving Average*» или «МА-алгоритм».

Все эти алгоритмы основаны на вычислении значений решающей функции  по значениям временного ряда *xi* (*i = 1, 2,...,n,…*) в ритме с процессом их поступления. Вычисленное значение  затем сопоставляется с определённым пороговым уровнем *Н*, разделяющим область возможных значений решающей функции (статистики) на две подобласти: и , где:

- подобласть продолжения наблюдений,

- критическая подобласть.

Если – имеет место ситуации отсутствия разладки. В противном случае, подаётся сигнал о наличии разладки.

Рассмотрим перечисленные алгоритмы более подробно.

Как известно, в начале 20 века развивалась сфера телефонизации. Компания «*American Telephone and Telegraph*» столкнулась с проблемой неопределенности количества ремонтных бригад для того, чтобы быстро исправлять неполадки в сети, тем самым увеличивая качество предоставляемой услуги. Изначально, неполадки в сети были вызваны выходом из строя ламповых усилителей сигнала. Данным вопросом заинтересовался Уолтер Шухарт. Решив данную задачу, Шухарт создал концепцию для современных систем регулирования качеством и стал своего рода «прародителем» методов обнаружения «разладки» случайных процессов. Карта Шухарта формируется следующим образом: получаем выборочно информацию о процессе через равные интервалы времени; эти интервалы задаются по времени или по конкретному показателю контролируемой продукции (например, каждая партия изделия). Для каждой сформированной группы одинакового размера, вычисляется характеристика, например: среднее арифметическое группы, размах характеристик группы или стандартное отклонение.

Карта Шухарта – это график значений наблюдаемой характеристики в зависимости от момента поступления соответствующей информации. В данном графике присутствует центральная линия (*CL*), которая соответствует эталонному значению контролируемой характеристики (в большинстве случаев это линия является средним арифметическим значений на некотором временном интервале. Кроме того, на этом графике присутствуют две статистически определяемые границы: верхняя контрольная граница и нижняя, *UCL* и *LCL* соответственно, как изображено на рис. 1.

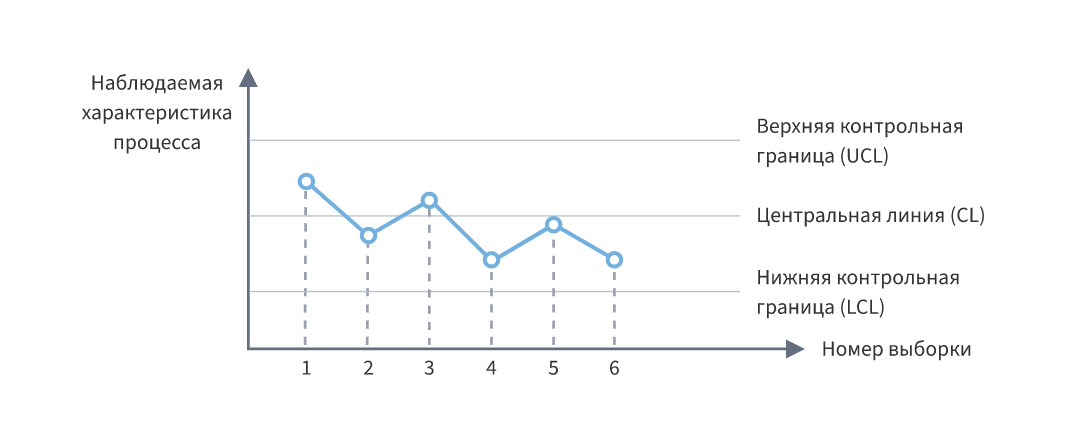


Рисунок 1 - Контрольная карта Шухарта.

Решающая функция алгоритма Шухарта, в большинстве случаев, формируется на некотором такте *i* следующим образом:

  (1)

или иначе: **, где 

В данном алгоритме осуществляется группирование данных с последующим вычислением средних значений в каждой группе. На практике обновление значений *gi* производится на каждом такте, кратном *N*, когда самое раннее значение суммы удаляется и заменяется на новое. Такую процедуру трудно назвать последовательной. Как средство обнаружения разладки карты Шухарта не нашли своего применения; их основное предназначение - выявление степени стабильности наблюдаемого процесса.

Алгоритм экспоненциально взвешенного скользящего среднего (*EWMA* - алгоритм) – в данном алгоритме реализуется непрерывное вычисление среднего значения для временного ряда по мере поступления новых наблюдений. *EWMA* широко используется в сфере финансовой деятельности, его основными приложениями являются технический анализ и моделирование волатильности. Название «экспоненциальное взвешивание» фигурирует не просто так. В более ранних наблюдениях присваиваются меньшие веса (веса уменьшаются экспоненциально по мере старения данных). Новые наблюдения имеют большой вес и, тем самым, отражая последние тенденции изменения наблюдаемых значений контролируемого процесса.

Единственный параметр в данном алгоритме – параметр экспоненциального сглаживания *α.*

*EWMA*-алгоритм основан на стандартной формуле экспоненциального сглаживания, когда очередное значение решающей функции определяется по формуле:

 (2)

где *α* – параметр сглаживания.

На практике при этом следует учесть, что при первоначальном запуске алгоритма будет иметь место переходной процесс длительностью порядка *3/α*.

Общий вид *EWMA* контрольной карты приведен на рис. 2.

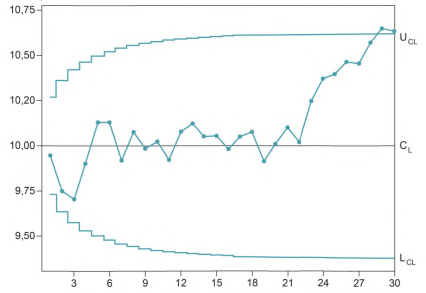


Рисунок 2 0 Общий вид Общий вид *EWMA* контрольной карты

*CUSUM* - алгоритм основан на идеи последовательный анализ Вальда.

В алгоритме кумулятивных сумм в решающая функция выглядит следующим образом:

  (3)

где – приращение решающей функции, подобное имеющему место в стандартном последовательном критерии отношения вероятностей Вальда [15]:

] (4)

где *-* функция плотности распределения вероятности при отсутствии разладки; - функция плотности распределения вероятности при наличии разладки.

Решающая функция не может иметь отрицательных значений, что в принципе должно обеспечивать более высокое быстродействие алгоритма в сравнении с другими алгоритмами. Решение о наличии «разладки» принимается по достижении решающей функцией верхнего порога, выбираемого с учетом требований на средний интервал между ложными тревогами.

На рис. 3 представлен пример реализации случайного процесса с разладкой по математическому ожиданию, а на рис. 4 поведение решающей функции CUSUM-алгоритма.

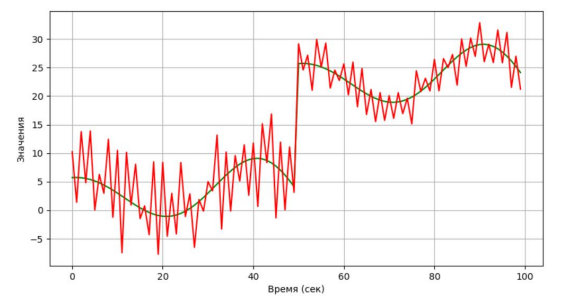


Рисунок 3 – Реализация контролируемого случайного процесса

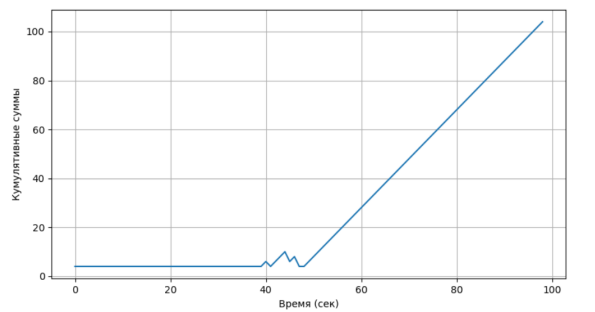


Рисунок 4 – График изменения накопленных сумм

Доказано, что CUSUM-алгоритм является оптимальным в смысле минимизации среднего времени обнаружения разладки по крайней мере в асимптотике при  . Кроме того, CUSUM-алгоритм является наиболее изученным алгоритмом, когда речь идет о разладке временных рядов по математическому ожиданию. Именно поэтому зачасту он используется в качестве образца, с которым сопоставляются свойства других алгоритмов обнаружения разладки.

Алгоритм скользящего среднего – «*Moving Average*» или «*МА*-алгоритм».

Решающая функция данного *МА*-алгоритма задается следующим соотношением:

 (5)

Фактически, здесь значениями решающей функции являются результат усреднения значений контролируемого временного ряда в скользящем окне шириной *N*.

Данный алгоритм учитывает ряд чисел, зафиксированный в подмножестве (стеке) длиной *N*. Подмножество изменяется путем сдвига элементов вперед, то есть из стека исключается первый (самый ранний) элемент, при этом добавляется новое значение в конец стека.

Данный алгоритм как средство обнаружения разладки в реальном масабе времени изучен в настоящее время очень слабо и, поэтому, не пользуется популярностью на практике. Возможно, это связано с тем, что его считают некоторым частным случаем алгоритма Шухарта. Однако, это неверно, что, в частности, проявляется при выборе решающей границы *Н*.

Исходя из сказанного, будем исследовать данный алгоритм с целью полного определения всех его свойств. При этом в конечном итоге необходимо получить всю информацию, необходимую пользователю для синтеза контролирующего алгоритма с заданными свойствами. Для такого синтеза необходимо прежде всего найти зависимость решающего порога *Н* от выбранного пользователем среднего интервала между ложными тревогами *Тлт*. Далее для первоначальной оценки быстродействия алгоритма следует оценить среднее значение запаздывания в обнаружение разладки для различных уровней контролируемого параметра разладки δ. И, наконец, в заключение, необходимо определить показатели эффективности данного алгоритма обнаружения *Е* = *Тлт / ,* что позволитсопоставить его с другими алгоритмами обнаружения разладки аналогичного назначения.

В качестве инструмента исследования МА-алгоритм выбран метод имитационного моделирования.

### Алгоритм 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» - . В качестве параметра разладки δ будем использовать безразмерную величину  .

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

 (6)

Фактически, здесь значениями 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:

. (7)

При известной функции  определение порога *Н* не представляет 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*:

(8)

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:

, (9)

где - нормированные значения контролируемого временного ряда,

т.е. случайные числа с нормированным нормальным распределением.

Определение решающего порога *Н* производится с помощью соотношения, аналогичного (7) по заданной величине :

. (10)

1) Задание исходных данных: среднего теоретического интервала между ложными тревогами , параметра *N*, определение и задание решающей границы *H* по табл. 1.

Таблица 1

Значения *H* для различных 

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

2) Задание параметра разладки δ – математическое ожидания .

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 данном этапе, *j* -ый запуск заканчивается. Зафиксированные значения *i* запоминаются как *Tj*.

Последовательность операций на каждом *j* –ом запуске описана ниже:

*Предварительная часть:*

1п) Генерируется *N* значений *x1p, x2p,…,xNp* c *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,...):

* 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;

Получение очередного перенормированного значения :

1) 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: ,…,, ;

2) Вычисление текущего значения решающей функции на *i*-ом такте:

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

3) Сравнение значения *gi* с решающим порогом *H*:

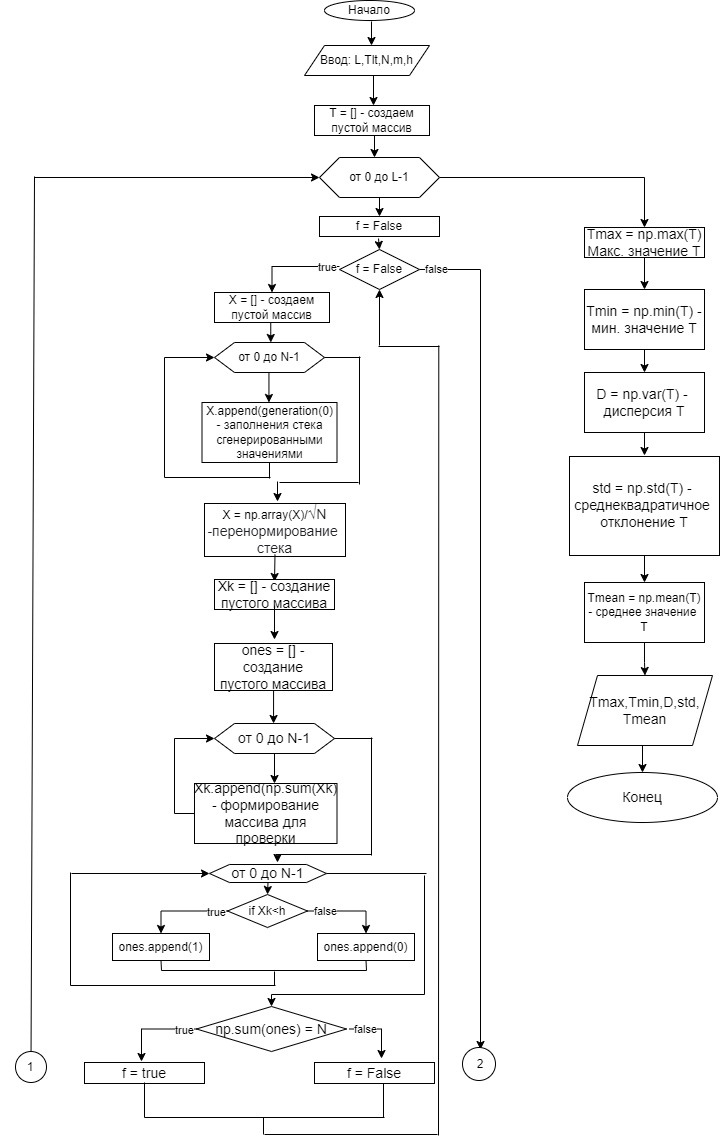
* если gi <*H*, то перейти к следующему (i +1)-му такту данного j-го запуска;
* если gi ≥*H*, то данный j-ый запуск завершается, фиксируется номер такта i, запоминается значение Тj, равное зафиксированному значению i;
* после этого перейти к следующему (j+1)-му запуску, начиная с предварительной части.

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: минимальное и максимальное значения *Тj*; вычисление среднего значения, дисперсий и среднеквадратического отклонения.

Блок-схема программы имитационного моделирования изображена на рис. 5.



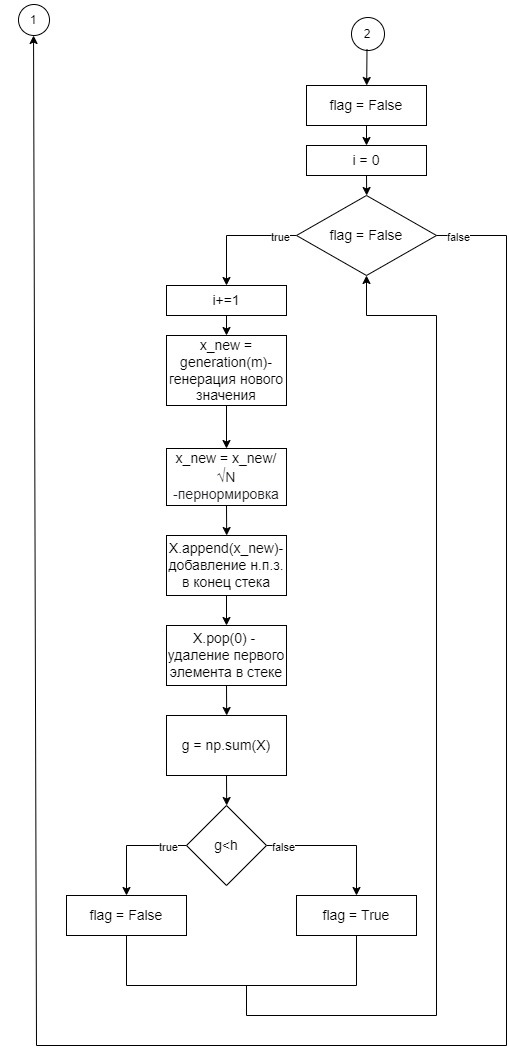


Рисунок 5 - Блок-схема программы имитационного моделирования *МА*-алгоритма

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 *xi* с нормированным нормальным распределением (Метод Бокса-Маллера)

1. Генерация на каждом такте *i* двух случайных чисел с равномерным распределением на интервале [0,1]: *u1i ;u2i*
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).*

Блок-схема программы генерации случайный чисел методом Бокса-Маллера представлена на рис. 6.

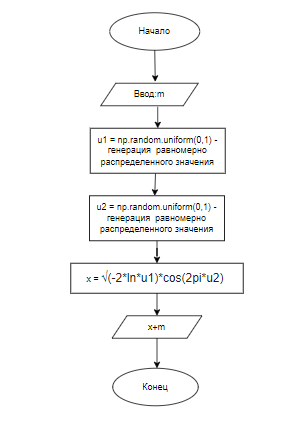


Рисунок 6 - Блок-схема функции генерации случайных чисел.

Характеристики программы имитационного моделирования *МА*-алгоритма (вместе с программой генерации случайных чисел) следующие:

Тип ЭВМ - ПК с процессорами Pentium или аналогичными им.

ОС - MS Windows XP/7/8/10.

Язык программирования – Python.

Объем программы - 2,62 КБ

На данную программу получено Свидетельство о государственной регистрации (см. рис. программы для ЭВМ», которое изображено на рис.7.



Рисунок 7 – Свидетельство о государственной регистрации программы

### Экспериментальное определение порога обнаружения

При первоначальной апробации программы имитационного моделирования были получены следующие результаты в условиях отсутствия разладки (то есть параметр δ = 0) для различных значений *Тлт* . Результаты представлены в табл. 2.

Таблица 2

Результаты моделирования для значений *Тлт* при *L =* 10000*, δ* = 0

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| N | *Тлт=50;*  *H*=2,054 | *Тлт=100;*  *H=2,326* | *Тлт=250;*  *H=2,652* | *Тлт=500;*  *H=2,878* | *Тлт=1000;*  *H=3,090* |
| 1 | 49.964 | 99.889 | 249.871 | 500.626 | 999.609 |
| 2 | 58.407 | 111.943 | 262.309 | 548.951 | 1021.504 |
| 3 | 69.131 | 130.198 | 306.756 | 547.574 | 1140.922 |
| 4 | 80.569 | 150.346 | 356.804 | 656.382 | 1298.228 |
| 5 | 92.235 | 168.814 | 377.912 | 715.454 | 1417.948 |
| 6 | 102.467 | 188.043 | 434.896 | 812.835 | 1587.006 |
| 7 | 113.701 | 207.672 | 449.387 | 887.599 | 1586.955 |
| 8 | 123.788 | 225.452 | 503.881 | 890.851 | 1827.576 |
| 9 | 134.764 | 242.135 | 561.891 | 1031.322 | 1864.311 |
| 10 | 144.133 | 261.731 | 588.901 | 1084.298 | 1985.465 |
| 11 | 156.111 | 279.601 | 633.107 | 1149.117 | 2002.569 |
| 12 | 165.488 | 297.372 | 674.671 | 1253.188 | 2471.025 |
| 13 | 177.124 | 318.108 | 689.271 | 1283.166 | 2488.675 |
| 14 | 188.438 | 333.401 | 728.491 | 1406.606 | 2352.131 |
| 15 | 199.211 | 351.876 | 778.782 | 1419.037 | 2618.369 |
| 16 | 205.891 | 366.645 | 859.919 | 1502.097 | 2727.901 |

Полученные результаты показывают, что значения *Н*, используемые в алгоритме Шухарта и приведенные в табл. 1, неприменимы для *Moving Average* для всех *N* , кроме *N* = 1. Вместо ожидаемого постоянного значения *Тлт* имеет место увеличение его значения при увеличении с ростом *N.* По всей видимости, увеличение *Тлт* с ростом *N* связано с тем, что значения решающей функции при *N*>1 коррелированы. Таким образом, появляется новая задача -найти зависимость порога *H* от показателя *Тлт* для различных значений интервала скользящего усреднения *N*.

### Разработка программы для определения порогов *Н* для различных *N*

Для решения проблемы, выявленной при первоначальной апробации программы имитационного моделирования, необходимо использовать алгоритм шагового поиска для определения порога *H* в зависимости от количества значений в стеке *N*.

Описание алгоритма представлено ниже:

Во-первых, вводятся параметры *L*, *Tлт*, *m, a, k*, *h0*, где: *a* – шаг изменения порога *h*; *k* – максимально допустимая разница между фактическим *Тлт* и введенным значением; *h0* – начальное значение порога *h*, взятое из табл. 1.

Во-вторых, вычисляется фактическое значение *Tлт* для текущего *h* (на первой итерации *h* = *h0*). Также, добавляем переменную *h1*, которая изначально совпадает со значением *h0*, эту переменную приравниваем к *h* в цикле проверки. Это введение помогает избежать случая, при котором разница между средним значением *Tлт* и ранее введенным будет меньше левой границы – *k*.

В-третьих, с помощью данного алгоритма возвращаемся к предыдущему значению *h* и получаем новое значение *h (*путем вычитания из шага изменения порога, деленного на коэффициент *r*, в данном случае *r* = 2). Если различия превосходят *k* по модулю, изменяем *h* на величину шага, затем вычисляем фактическое значение *Tлт* до тех пор, пока не будет достигнут корректный результат.

Блок-схема данной программы представлена на рис. 8.

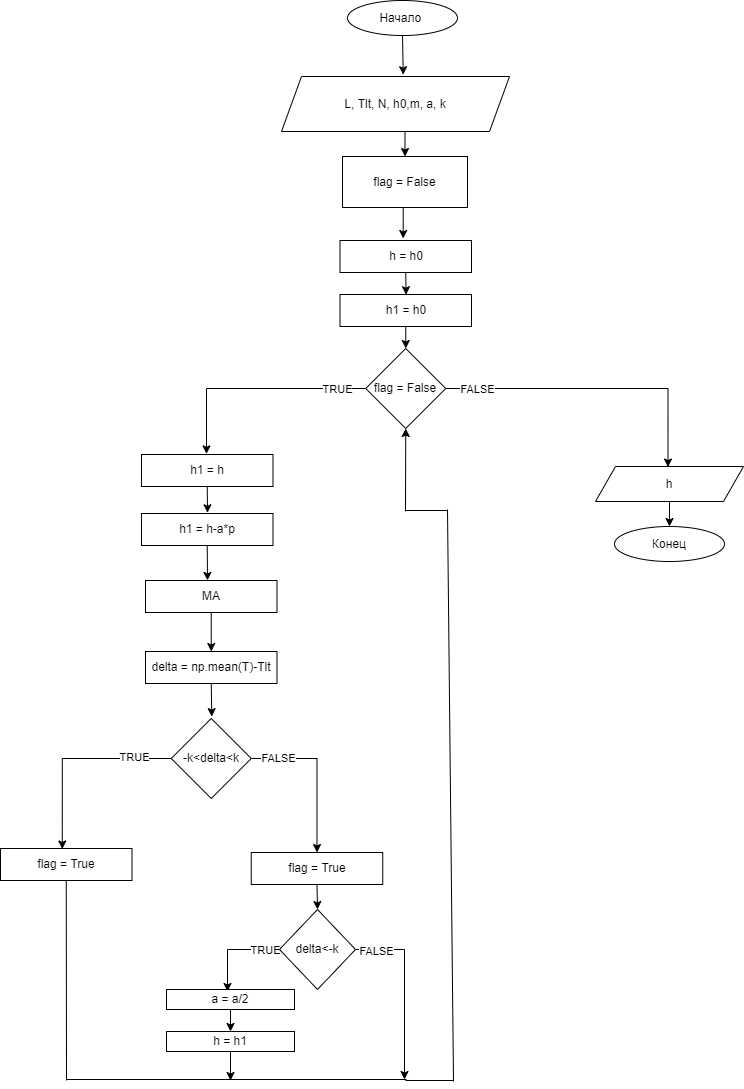


Рисунок 8 - Блок-схема программы для определения зависимости *H*(*N*)

С помощью данной программы получена справочная информация о зависимости *H*(*N*) для разных значений *Тлт* и двух значений длительности имитационного эксперимента *L* (табл. 3 и 4).

Таблица 3

Зависимости *H(N)* для разных значений *Тлт* при *L = 10000*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *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 |

Увеличив параметр *L* (число запусков процесса имитации работы), были получены более точные результаты о зависимости *H*(*N*) для разных значений *Тлт*.

Таблица 4

Зависимости *H(N)* для разных значений *Тлт* при *L = 100000*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *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 | 1.981 | 2.276 | 2.622 | 2.855 | 3.073 |
| 3 | 1.906 | 2.216 | 2.574 | 2.818 | 3.044 |
| 4 | 1.83 | 2.154 | 2.529 | 2.78 | 3.014 |
| 5 | 1.758 | 2.098 | 2.485 | 2.746 | 2.981 |
| 6 | 1.692 | 2.043 | 2.445 | 2.708 | 2.951 |
| 7 | 1.635 | 1.997 | 2.41 | 2.677 | 2.923 |
| 8 | 1.578 | 1.953 | 2.372 | 2.646 | 2.879 |
| 9 | 1.528 | 1.91 | 2.336 | 2.619 | 2.874 |
| 10 | 1.476 | 1.869 | 2.307 | 2.59 | 2.851 |
| 11 | 1.434 | 1.831 | 2.278 | 2.566 | 2.828 |
| 12 | 1.388 | 1.796 | 2.25 | 2.542 | 2.806 |
| 13 | 1.346 | 1.76 | 2.22 | 2.517 | 2.789 |
| 14 | 1.302 | 1.726 | 2.193 | 2.499 | 2.767 |
| 15 | 1.264 | 1.696 | 2.168 | 2.474 | 2.753 |
| 16 | 1.228 | 1.665 | 2.148 | 2.456 | 2.75 |

Таким образом, при увеличении количества итерации прохода алгоритма *L*, зависимость *H*(*N*) становится более качественной. Кроме того, оказывается возможны требуемую зависимость аппроксимировать с помощью определенной функцией.

### Результаты анализа статистических характеристик

### *МА*-алгоритма

Для каждого значения среднего интервала между ложными тревогами *Тлт* , равного 100, 250, 500 и 1000 получены значения среднего времени запаздывания в обнаружения разладки . Они отображены в табл. 5 – 8.

Таблица 5

Результаты имитационного моделировании при *Тлт =* 100и *L =* 100000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | *δ* = 0 | *δ* = 0.5 | *δ* = 1 | *δ* = 1,5 | *δ* = 2 | *δ* = 2.5 | *δ* = 3 |
| 1 | 100.151 | 29.709 | 10.814 | 4.885 | 2.684 | **1.755** | **1.331** |
| 2 | 99.853 | 21.711 | 7.294 | 3.605 | **2.345** | 1.841 | 1.594 |
| 3 | 100.331 | 18.729 | 6.387 | **3.452** | 2.49 | 2.058 | 1.804 |
| 4 | 99.957 | 16.806 | 6.03 | 3.531 | 2.683 | 2.249 | 1.966 |
| 5 | 99.909 | 15.716 | 5.911 | 3.684 | 2.897 | 2.412 | 2.098 |
| 6 | 99.773 | 14.98 | **5.905** | 3.868 | 3.043 | 2.551 | 2.215 |
| 7 | 99.712 | 14.542 | 6.011 | 4.051 | 3.196 | 2.671 | 2.324 |
| 8 | 99.817 | 14.234 | 6.102 | 4.219 | 3.329 | 2.774 | 2.407 |
| 9 | 100.224 | 13.969 | 6.244 | 4.366 | 3.446 | 2.881 | 2.487 |
| 10 | 100.253 | 13.822 | 6.383 | 4.521 | 3.552 | 2.968 | 2.535 |
| 11 | 100.264 | 13.742 | 6.545 | 4.633 | 3.664 | 3.041 | 2.634 |
| 12 | 100.061 | 13.652 | 6.711 | 4.774 | 3.748 | 3.122 | 2.677 |
| 13 | 100.068 | **13.636** | 6.85 | 4.97 | 3.829 | 3.181 | 2.732 |
| 14 | 99.861 | 13.643 | 6.992 | 4.974 | 3.927 | 3.253 | 2.813 |
| 15 | 100.004 | 13.735 | 7.133 | 5.089 | 4.003 | 3.311 | 2.854 |
| 16 | 99.866 | 13.738 | 7.276 | 5.177 | 4.071 | 3.371 | 2.898 |

Таблица 6

Результаты имитационного моделировании при *Тлт =* 250и *L =* 100000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | *δ* = 0 | *δ* = 0.5 | *δ* = 1 | *δ* = 1,5 | *δ* = 2 | *δ* = 2.5 | *δ* = 3 |
| 1 | 249.684 | 63.935 | 20.465 | 8.094 | 3.884 | 2.652 | **1.577** |
| 2 | 250.334 | 43.441 | 11.999 | 4.982 | **2.838** | **2.094** | 1.762 |
| 3 | 250.141 | 34.653 | 9.53 | 4.343 | 2.869 | 2.301 | 1.992 |
| 4 | 249.789 | 30.257 | 8.474 | **4.289** | 3.071 | 2.527 | 2.195 |
| 5 | 250.331 | 26.953 | 7.915 | 4.381 | 3.282 | 2.731 | 2.366 |
| 6 | 250.801 | 24.997 | 7.709 | 4.542 | 3.445 | 2.913 | 2.514 |
| 7 | 250.108 | 23.703 | **7.644** | 4.751 | 3.696 | 3.067 | 2.644 |
| 8 | 250.436 | 22.392 | 7.671 | 4.946 | 3.868 | 3.209 | 2.757 |
| 9 | 250.045 | 21.634 | 7.759 | 5.143 | 4.013 | 3.327 | 2.867 |
| 10 | 249.789 | 21.106 | 7.904 | 5.339 | 4.167 | 3.448 | 2.971 |
| 11 | 250.334 | 20.608 | 8.064 | 5.524 | 4.305 | 3.559 | 3.057 |
| 12 | 250.401 | 20.279 | 8.213 | 5.709 | 4.431 | 3.672 | 3.142 |
| 13 | 249.981 | 19.925 | 8.368 | 5.836 | 4.549 | 3.754 | 3.215 |
| 14 | 250.17 | 19.754 | 8.544 | 5.983 | 4.665 | 3.846 | 3.286 |
| 15 | 250.104 | 19.629 | 8.749 | 6.117 | 4.767 | 3.929 | 3.361 |
| 16 | 250.075 | **19.563** | 8.922 | 6.264 | 4.874 | 4.005 | 3.432 |

Таблица 7

Результаты имитационного моделировании при *Тлт =* 500и *L =* 100000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | *δ* = 0 | *δ* = 0.5 | *δ* = 1 | *δ* = 1,5 | *δ* = 2 | *δ* = 2.5 | *δ* = 3 |
| 1 | 500.554 | 114.519 | 33.212 | 11.791 | 5.261 | 2.971 | **1.819** |
| 2 | 500.317 | 73.451 | 17.504 | 6.402 | 3.359 | **2.289** | 1.878 |
| 3 | 500.553 | 56.897 | 12.947 | 5.278 | **3.204** | 2.477 | 2.817 |
| 4 | 499.951 | 47.087 | 10.988 | **4.952** | 3.355 | 2.724 | 2.357 |
| 5 | 500.344 | 41.198 | 9.979 | 4.961 | 3.574 | 2.953 | 2.549 |
| 6 | 500.401 | 37.001 | 9.434 | 5.081 | 3.816 | 3.153 | 2.717 |
| 7 | 500.421 | 34.106 | 9.153 | 5.259 | 4.026 | 3.337 | 2.871 |
| 8 | 500.385 | 31.224 | **9.063** | 5.473 | 4.231 | 3.503 | 3.009 |
| 9 | 499.698 | 30.365 | 9.066 | 5.673 | 4.424 | 3.654 | 3.133 |
| 10 | 500.402 | 29.139 | 9.121 | 5.906 | 5.592 | 3.774 | 3.245 |
| 11 | 499.677 | 28.061 | 9.237 | 6.117 | 4.763 | 3.921 | 3.351 |
| 12 | 499.845 | 27.258 | 9.368 | 6.315 | 4.895 | 4.027 | 3.449 |
| 13 | 449.613 | 27.254 | 9.613 | 6.552 | 5.091 | 4.194 | 3.572 |
| 14 | 500.338 | 26.075 | 9.761 | 6.689 | 5.179 | 4.264 | 3.647 |
| 15 | 500.098 | 25.542 | 9.928 | 6.852 | 5.311 | 4.367 | 3.715 |
| 16 | 500.451 | **25.368** | 10.133 | 7.012 | 5.437 | 4.458 | 3.808 |

Таблица 8

Результаты имитационного моделировании при *Тлт =* 1000и *L =* 100000

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N | *δ* = 0 | *δ* = 0.5 | *δ* = 1 | *δ* = 1,5 | *δ* = 2 | *δ* = 2.5 | *δ* = 3 |
| 1 | 1000.471 | 206.427 | 54.697 | 17.805 | 7.422 | 3.605 | 3.088 |
| 2 | 999.504 | 125.231 | 25.892 | 8.441 | 3.972 | **2.542** | **2.006** |
| 3 | 1000.34 | 93.022 | 17.943 | 17.912 | 6.397 | 2.666 | 2.261 |
| 4 | 1000.242 | 74.896 | 14.472 | 5.741 | **3.646** | 2.913 | 2.512 |
| 5 | 999.467 | 63.366 | 12.657 | **5.599** | 3.865 | 3.159 | 2.721 |
| 6 | 999.671 | 55.919 | 11.606 | 5.614 | 4.108 | 3.388 | 2.911 |
| 7 | 1000.432 | 50.047 | 11.042 | 5.791 | 5.763 | 3.597 | 3.077 |
| 8 | 999.613 | 44.783 | 10.552 | 5.961 | 4.551 | 3.753 | 3.219 |
| 9 | 999.564 | 42.966 | 10.636 | 6.223 | 4.791 | 3.948 | 3.276 |
| 10 | 999.609 | 40.581 | **10.498** | 6.451 | 4.988 | 4.104 | 3.512 |
| 11 | 1000.421 | 38.523 | 10.502 | 6.686 | 5.169 | 4.251 | 3.634 |
| 12 | 999.453 | 36.958 | 10.621 | 6.911 | 5.339 | 4.396 | 3.741 |
| 13 | 1000.442 | 35.641 | 10.773 | 7.115 | 5.505 | 4.531 | 3.858 |
| 14 | 1000.231 | 34.484 | 10.921 | 7.328 | 5.673 | 4.649 | 3.961 |
| 15 | 1000.382 | 33.711 | 11.136 | 7.522 | 5.821 | 4.775 | 4.064 |
| 16 | 1000.052 | **33.359** | 11.391 | 7.765 | 5.991 | 4.895 | 4.172 |

Исходя из данных, представленных в табл. 5 – 8, можно сделать вывод: для каждого значения *δ* при определенном значении  можно найти такое значение *N*, при котором получаем минимальную величину , т. е. такое значение *N* , при котором обеспечивается наибольшее быстродействие алгоритма обнаружения разладки.

Оптимальные значения *N* для различных δи  приведены в табл. 9.

Таблица 9

Оптимальные значения *N* для различных δ и 

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | δ = 0.5 | δ = 1,0 | δ = 1,5 | δ = 2,0 | δ = 2.5 | δ = 3,0 |
| 100 | 13 | 6 | 3 | 2 | 1 | 1 |
| 250 | 16 | 7 | 4 | 2 | 2 | 1 |
| 500 | 16 | 8 | 4 | 3 | 2 | 1 |
| 1000 | 16 | 9 | 5 | 4 | 2 | 2 |

Ниже представлены показатели произведенного расчёта эффективности *МА*-алгоритма, рассчитываемого по формуле *E* = /, что отражено в табл. 10.

Таблица 10

Показатели эффективности *E* для *МА*-алгоритма при различных *δ* и 

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *δ* = 0.5 | *δ* = 1,0 | *δ* = 1,5 | *δ* = 2,0 | *δ* = 2.5 | *δ* = 3,0 |
| 100 | 7,33 | 16,93 | 28,97 | 42,64 | 56.98 | 75,13 |
| 250 | 12,78 | 32,71 | 58,29 | 88,09 | 119,4 | 158,53 |
| 500 | 19,71 | 55,17 | 100.97 | 156,05 | 218,44 | 274,87 |
| 1000 | 29,98 | 95.26 | 178,6 | 274,27 | 393,39 | 498,5 |

### 6. Сопоставительный анализ *МА*-алгоритма

Сопоставим полученные данные об эффективности *МА*–алгоритма с эффективностью алгоритма *CUSUM*.

Показатели эффективности для CUSUM-алгоритма при различных δ и , полученные с помощью программной системы STATCONT приведены в табл. 11.

Таблица 11

Показатели эффективности *Ecusum* при различных *δ* и



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *δ* = 0.5 | *δ* = 1,0 | *δ* = 1,5 | *δ* = 2,0 | *δ* = 2.5 | *δ* = 3,0 |
| 100 | 6,71 | 16,53 | 29,24 | 43,86 | 59,88 | 76,92 |
| 250 | 11,92 | 32,05 | 59,10 | 91,58 | 128,2 | 168,9 |
| 500 | 19,28 | 54,64 | 103,3 | 162,9 | 230,4 | 306,7 |
| 1000 | 32,07 | 95,15 | 183,5 | 292,4 | 418,4 | 550,7 |

Используя данные табл. 10 и 11, рассчитаем показатели относительной эффективности *МА*-алгоритма по сравнению с *CUSUM*-алгоритмом, по следующей формуле: *Є* =  (см. табл. 12).

Таблица 12

Показатели относительной эффективности *Є* при различных *δ* и



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | *δ* = 0.5 | *δ* = 1,0 | *δ* = 1,5 | *δ* = 2,0 | *δ* = 2.5 | *δ* = 3,0 |
| 100 | 1,092 | 1,024 | 0,991 | 0,972 | 0,952 | 0,977 |
| 250 | 1,072 | 1,021 | 0,986 | 0,962 | 0,931 | 0,938 |
| 500 | 1,022 | 1,010 | 0,977 | 0,958 | 0,948 | 0,896 |
| 1000 | 0,935 | 1.001 | 0,973 | 0,938 | 0,940 | 0,905 |

Анализируя табл. 12, можно сделать вывод, что *МА*–алгоритм практически не уступает по эффективности *CUSUM*-алгоритму и при малых значениях *δ* и  даже дает более точный результат.

### 

### Заключение

В ходе данной работы рассмотрена литература по проблематике обнаружения «разладки» временных рядов; составлен краткий обзор известных методов обнаружения разладки.

Кроме того, поставлена задача исследования *МА*-алгоритма, предназначенного для выявления разладки гауссовского случайного процесса по математическому ожиданию (с помощью метода имитационного моделирования).

Наконец, разработана программа имитационного моделирования *МА*-алгоритма, с помощью которой получены его основные характеристики, необходимые для синтеза соответствующей контролирующей процедуры.

По итогам данной работы, полученные результаты позволяют найти оптимальное значение параметров для того, чтобы минимизировать среднее время задержки при фиксированном среднем значении интервала между ложными тревогами.

Необходимо подчеркнуть, что проведено сопоставление характеристик оптимизированного *МА*-алгоритма и аналогичного по своему назначению *CUSUM*-алгоритма, показывающее, что *МА*-алгоритм по своей эффективности лишь незначительно уступает *CUSUM*-алгоритму, а в некоторых вариантах при малых значениях *δ* и  даже его превосходит.

Таким образом, задание ВКР 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. |

1. **Никифоров И.В.** Последовательное обнаружение изменения свойств временных рядов. – М.: Наука, 1983.
2. **Семенов Н.А., Телков А.Ю.** Применение статистических методов обнаружения DoS атак в локальной сети. Вестник ВГУ, серия: системный анализ и информационные технологии, 2012, №1,с. 82 – 87.
3. **Кацер Ю.Д., Козицин В.О., Максимов И.В**. Методы обнаружения неисправностей оборудования АЭС, Известия вузов. Ядерная энергетика, 2019, №4, с. 5 – 20.
4. **Филаретов Г.Ф.** Диалоговая программная система «STATCONT». – Приборы и системы управления, 1998, № 5, с. 16 – 18.

### *Приложение*

### 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())

**#**Листинг программы нахождения зависимости h от N

import pandas as pd

import numpy as np

import random

import math

from tqdm import tqdm

import matplotlib.pyplot as plt

# генерация числа

def generation(m):

u1 = random.uniform(0,1.0)

u2 = random.uniform(0,1.0)

x = math.cos(2\*math.pi\*u2)\*(-2\*math.log(u1,math.e))\*\*(1/2)

return x+m

L = int(input('Введите кол-во итераций (L) : '))

Tlt = int(input('Введите значение среднего теоретического интервала между ложными тревогами (Tlt) : '))

N = int(input("Кол-во значений в стеке (N) : "))

h0 = float(input("Введите нальноее значение решающей границы (h0) : "))

a = float(input("Введите шаг изменения h : "))

k = float(input("Введите значение k : "))

m = 0

h = h0

flaggg = False

while (flaggg == False):

T = []

h1 = h

print('h= ',h,'h1= ',h1,'step = ',a)

h = h - a

# print(h)

pbar = tqdm(np.arange(0,L))

for j in pbar:

# предварительная часть:

f = False

while(f == False):

X = []

for i in range(N):

X.append(generation(0)/(N\*\*(0.5)))

Xk = []

ones = []

for i in range(0,N):

Xk.append(X[i])

g0 = np.sum(Xk)

if g0 >= h:

ones.append(0)# предварительная чать реализуется повторно

else:

ones.append(1)

if np.sum(ones) == N:

f = True

else:

f = False

# основная часть:

t = []

flag = False

i = 0

while(flag == False):

i+=1

x\_new = generation(m)/(N\*\*(0.5))

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}')

delta = np.mean(T)-Tlt

if -k<delta<k:

flaggg = True

else:

flaggg = False

if delta<-k:

a = a/2

h = h1

print('-'\*80)

print(f'h = {h} \nN = {N}')