МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра теории вероятностей и математической статистики

ИВАНКОВИЧ
Илья Валерьевич

ПОСЛЕДОВАТЕЛЬНАЯ СТАТИСТИЧЕСКАЯ ПРОВЕРКА ГИПОТЕЗ В СЛУЧАЕ МНОГИХ АЛЬТЕРНАТИВ

Магистерская диссертация
специальность 1-31 80 09 прикладная математика и информатика

Научный руководитель:
заведующий кафедрой теории вероятностей и математической статистики,
доктор физ.-мат. наук,
доцент Харин А.Ю.

Допущен к защите

«____»_____2021 г.

Заведующий кафедрой теории вероятностей и математической статистики, доктор физ.-мат. наук, доцент А.Ю. Харин

Минск, 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В прикладной математике всё большую важность приобретает направление, связанное с компьютерным анализом данных. Среди задач анализа данных значительную часть составляют задачи статистической проверки гипотез о параметрах наблюдаемых явлений или процессов [2]. Эффективным подходом для решения таких задач является вероятностно-статистический подход, называемый последовательным анализом Вальда [6] по имени американского математика, в 1947 предложившего такую схему решения.

Широко известным примером эффективного последовательного подхода является последовательный критерий отношения вероятностей (критерий Вальда) в проверке гипотез. Последовательный критерий отношений вероятностей зачастую требует примерно на 50% меньше наблюдений, чем наиболее эффективный критерий, основанный на фиксированном количестве наблюдений [9].

Такая эффективность обусловлена, тем что число наблюдений не фиксируется заранее, а зависит от самих наблюдений. Критерием завершения процесса наблюдения и основанием для вынесения решения в пользу одной из гипотез является достижение заданной точности (малых значений вероятностей ошибок). Это создаёт в свою очередь сложности анализа [3,5]. Так, характеристики эффективности последовательного теста даже проверки двух простых гипотез в явном виде получены лишь для простейших моделей наблюдений [7,8].

Магистерская диссертация посвящена анализу эффективности последовательных тестов, а именно исследованию зависимостей характеристик эффективности (вероятностей ошибок и среднего числа наблюдений) от параметров в случае, когда альтернатив больше, чем две. Кроме того, проводится применение полученных результатов анализа абстрактных последовательных тестов на реальных статистических данных по заболеваемости коронавирусом в Беларуси и Украине в 2020 году.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

***Перечень ключевых слов*:** СТАТИСТИЧЕСКИЙ ПОСЛЕДОВАТЕЛЬНЫЙ АНАЛИЗ, ВЕРОЯТНОСТИ ОШИБОК, КРИТЕРИЙ ОТНОШЕНИЯ ВЕРОЯТНОСТЕЙ, ИСКАЖЕНИЯ, ГИПОТЕЗЫ, КОМПЬЮТЕРНЫЕ ЭКСПЕРИМЕНТЫ, ЭФФЕКТИВНОСТЬ, ЗАБОЛЕВАЕМОСТЬ КОРОНАВИРУСОМ.

Цель работы – исследование показателей эффективности для последовательных статистических тестов более двух простых гипотез.

Задачи исследования:

1) исследовать существующие последовательные тесты (критерии) статистической проверки многих (более 2) альтернатив в рамках простейшей модели наблюдений;
2) провести анализ эффективности и робастности рассмотренных последовательных тестов;
3) исследовать эффективность матричного последовательного теста;
4) применить разработанную теорию для компьютерного анализа данных, связанных с заболеваемостью коронавирусом, с целью выявления типовых траекторий эпидемиологического процесса.

Объект исследования – многоальтернативные последовательные статистические тесты.

Предмет исследования – эффективность последовательных статистических тестов и влияние на нее искажений в наблюдениях.

Результаты работы и их новизна – в работе получены следующие основные результаты:

1) построен однопороговый последовательный тест проверки более двух простых гипотез, основанный на апостериорных вероятностях;
2) построен матричный последовательный тест проверки более двух простых гипотез;
3) исследованы зависимости характеристик эффективности последовательного теста;

4) исследована возможность использования последовательного теста для определения характера поведения распространения инфекции COVID-19.

Они являются развитием известных результатов и впервые исследуют робастность последовательного теста, основанного на апостериорных вероятностях, а также детально анализируют эффективность в зависимости от параметров; для модели с трендом построена модификация последовательного теста.

Область применения – результаты применены в работе для анализа эпидемиологических процессов по заболеваемости коронавирусом.

Структура магистерской диссертации – диссертацая изложена на 62 страницах. Состоит из разделов «Введение», «Общая характеристика работы», 4 глав, разделов «Заключение» и «Список использованных источников». В первой главе приводятся основные формулы и понятия. Вторая глава посвящена особенностям компьютерной реализации. В третьей главе анализируются результаты экспериментов и делаются краткие выводы. В четвертой главе рассматривается возможность применения матричного последовательного теста в анализе эпидемиологического процесса по заболеваемости COVID-19. В работе имеется приложение 53 с. – 62 с., 33 рисунка, 7 таблиц и 9 источников.

АГУЛЬНАЯ ХАРАКТАРЫСТЫКА ПРАЦЫ

Пералік ключавых слоў : СТАТЫСТЫЧНЫ ПАСЛЯДОУНЫ АНАЛІЗ, ВЕРАГОДНАСЦЬ ПАМЫЛАК, КРЫТЭРЫЙ АДНОСІН ВЕРАГОДНАСЦІ, СКАЖЭННІ, ГІПОТЭЗЫ, КАМПУТАРНЫЯ ЭКСПЕРЫМЕНТЫ, ЭФЕКТЫУНАСЦЬ, ЗАХВОРВАННЕ КАРОНАВІРУСАМ.

Мэта работы – даследаванне паказчыкаў эфектыўнасці для паслядоўных статыстычных тэстаў больш за дзве простыя гіпотэзы.

Задачы даследавання:

1) даследаваць існуючыя паслядоўныя тэсты (крытэрыі) статыстычнай праверкі шматлікіх (больш за 2) альтэрнатыў у рамках найпростай мадэлі назіранняў;
2) правесці аналіз эфектыўнасці і рабаснасці разгледжаных паслядоўных тэстаў;
3) даследаваць эфектыўнасць матрычнага паслядоўнага тэсту;
4) прымяніць распрацаваную тэорыю для камп'ютэрнага аналізу дадзеных, звязаных з захворваннем каронавірусам, з мэтай выяўлення тыпавых траекторый эпідэміялагічнага працэсу.

Аб'ект даследавання – шматальтэрнатыўныя паслядоўныя статыстычныя тэсты.

Прадмет даследавання – эфектыўнасць паслядоўных статыстычных тэстаў і ўплыў на яе скажэнняў у назіраннях.

Вынікі працы і іх навізна – у працы атрыманы наступныя асноўныя вынікі:

1) пабудаваны аднопарогавы паслядоўны тэст праверкі дзвух простых гіпотэз, заснаваны на апастэрыёрных верагоднасцях;
2) пабудаваны матрычны паслядоўны тэст праверкі больш за дзве простыя гіпотэзы;
3) даследаваны залежнасці характарыстык паслядоўнага тэсту;
4) даследавана магчымасць выкарыстання паслядоўнага тэсту для вызначэння характару паводзін распаўсюджвання інфекцыі COVID-19.

Яны з'яўляюцца развіццём вядомых вынікаў і ўпершыню даследуюць рабаснасць паслядоўнага тэсту, заснаванага на апастэрыёрных верагоднасцях,

а таксама дэталёва аналізуюць эфектыўнасць у залежнасці ад параметраў; для мадэлі з трэндам пабудавана мадыфікацыя паслядоўнага тэсту.

Вобласць прымянення **–** вынікі ўжыты ў працы для аналізу эпідэміялагічных працэсаў па захворванні коронавирусом.

Структура магістарскай дысертацыі **–** дысертацыя выкладзена на 61 старонцы. Складаецца з раздзелаў «Увядзенне», «Агульная характарыстыка работы", 4 глаў, раздзелаў «Заключэнне» і «Спіс выкарыстаных крыніц». У першай главе прыводзяцца асноўныя формулы і паняцці. Другая глава прысвечана асаблівасцям камп′ютарнай рэалізацыі. У трэціяй главе аналізуюцца вынікі эксперыментаў і робяцца кароткія высновы. У чацвёртай главе разглядаецца магчымасць прымянення матрычнага паслядоўнага тэсту ў аналізе эпідэміялагічнага працэсу па захворванні COVID-19. У працы маецца дадатак 53 с. - 62 с ., 33 малюнка, 7 табліц і 9 крыніц.

SUMMARY

Keywords: CONSISTENT STATISTICAL ANALYSIS, PROBABILITY OF ERRORS, CRITERIA FOR PROBABILITY RELATIONS, DISTORTIONS, HYPOTHESES, COMPUTER EXPERIMENTS, EFFICIENCY, INCIDENCE WITH CORONAVIRUS.

Objective **–** performance study for group sequential statistical tests of more than two simple hypotheses.

Research objective:

1) explore the existing sequential tests (criteria) for statistical verification of many (more than 2) alternatives within the framework of the simplest observation model;
2) analyze the efficiency and robustness of the considered sequential tests;
3) investigate the effectiveness of the matrix sequential test;
4) apply the developed theory for computer experiments related to the incidence of coronavirus in order to identify typical trajectories of the epidemiological process.

Object of study **–** multialternative consistent sequential statistical tests.

Subject of study **–** effectiveness of sequential statistical tests and the influence of distortion in observations.

Work results and their novelty **–** the following main results were obtained in the work:

1) built a one-threshold sequential test for more than two simple hypotheses based on posterior probabilities;
2) built matrix sequential tests for more than two simple hypotheses
3) investigated dependences of the characteristic's effectiveness of the sequential test;
4) investigated possibility of using a sequential test for determining the behavior of the spread for COVID-19;

They are an extension of the known results and for the first time investigated the robustness of a sequential test based on posterior probabilities, also analyze in detail the efficiency depending on the parameters; for the model with a trend, a modification of the sequential test is built.

Scope **–** the results were applied in the work to analyze the epidemiological processes in terms of the incidence of coronavirus.

Structure of the master's dissertation – the dissertation is presented on 62 pages. Consists of sections "Introduction", "General characteristics of the work", 4 chapters, sections "Conclusion" and "List of sources used". The first chapter provides basic formulas and concepts. The second chapter is devoted to the peculiarities of computer implementation. The third chapter analyzes the results of the experiments and gives brief conclusions. The fourth chapter examines the possibility of using a matrix sequential test in the analysis of the epidemiological process for the incidence of COVID-19. The work contains an appendix 53 p. - 62 p., 33 figures, 7 tables and 9 sources.

ГЛАВА 1
ПОСЛЕДОВАТЕЛЬНЫЙ ПОДХОД В СТАТИСТИЧЕСКОЙ ПРОВЕРКЕ ГИПОТЕЗ

В данной главе раскрываются основные понятия по теме магистерской работы, даются ключевые определения, включая статистические гипотезы, области принятия решений, групповые тесты. Более подробно рассматривается последовательный критерий отношения вероятностей, а также последовательный тест Байеса.

1.1 Основные понятия

Последовательный анализ [6] – метод статистического исследования, в котором количество наблюдений, необходимых в процессе испытания, заранее не определено. Решение об окончании эксперимента зависит на каждой данной стадии эксперимента от результатов предыдущих наблюдений. Достоинство данного метода, применительно к проверке статистических гипотез, заключается в том, что он позволяет сконструировать такую методику проверки, которая требует, в среднем, существенно меньшего числа наблюдений, чем равная ей по надежности проверка, основанная на заранее определенном количестве наблюдений.

Наблюдения производятся по одному (или, более общим образом, группами) и анализируются в ходе самого эксперимента с тем, чтобы на каждом этапе решить, требуются ли ещё наблюдения (решение о продолжении эксперимента) или наблюдений уже достаточно (решение об остановке эксперимента). Когда эксперимент остановлен, заключительное статистическое решение принимается на основе всех наблюденных в эксперименте данных.

На практике данные часто подвержены искажениям: зависимости, выбросы и др. Искажения могут влиять на точность статистических тестов.

Приведем необходимые понятия, определения и свойства для того, чтобы построить и исследовать последовательные статистические тесты при наличии выбросов.

Пусть x — случайная величина, $f(t)$ – функция плотности вероятности, тогда $x_1, x_2, \dots x_n$ — последовательность независимых наблюдений величины x. Вероятность получения выборки, совпадающей с наблюденной, определяется произведением $f(x_1)f(x_2)\dots f(x_n)$, в силу независимости наблюдений. Это произведение называется также совместным распределением вероятностей выборки $x_1, x_2, \dots x_n$.

Статистическая гипотеза — некоторое предположение о виде распределения и свойствах случайной величины, формулируемое на основе применения статистических методов к данным выборки.

Проверка гипотезы – это вычисление некоторой случайной величины, называемой критерием, у которого известно точное или приближенное распределение. Обозначим эту величину через z, ее значение является функцией от элементов выборки $z = z(x_1, x_2, \dots, x_n)$.

Процедура проверки гипотезы заключается в следующем: каждому значению критерия предписывается одно из двух решений – отвергнуть или принять гипотезу. Соответственно все выборочное пространство критерия и тем самым множество значений делятся на два непересекающихся подмножества S_0 и S_1. Если значение критерия z попадает в область S_1, то гипотеза отклоняется, а множество называется – *областью отклонения гипотезы или критической областью*, если в область S_0 – гипотеза принимается и соответственно множество называется *областью принятия гипотезы или областью допустимых значений*. Выбор одной из двух областей однозначно определяет и другую.

1.2 Последовательный критерий отношений вероятностей

Предположим, есть M гипотез, среди которых только одна верна. Будем стремиться к быстрому и последовательному увеличению объёма информации о правильной гипотезе с учётом затрат на процесс сбора данных. Однако, в отличие от классической проблемы последовательного тестирования гипотез, при принятии решения мы можем выбрать одно из K доступных действий и, следовательно, осуществлять некоторый контроль над «информационным содержанием» собранной выборки. Далее будем называть это обобщение *активной* проблемой последовательного тестирования гипотез.

Наиболее известным примером данной проблемы является случай проверки двоичной гипотезы (M = 2) с пассивным выбором (K = 1), впервые изученный Вальдом [1]. В этом случае оптимальный результат может быть достигнут при использовании последовательного теста отношения вероятностей.

Пусть *f(x,θ)* означает распределение рассматриваемой случайной величины и H_0 является гипотезой о том, что $\theta = \theta_0$, H_1 — гипотеза о том, что $\theta = \theta_1$; следовательно, распределение x задается выражением $f(x, \theta_0)$, когда справедлива H_0 и выражением $f(x, \theta_1)$, когда справедлива H_1.

Предположим x - последовательные наблюдения $x_1, x_2, \dots x_n$. Для любого положительного целого числа m вероятность получения выборки $x_1, x_2, \dots, x_m$ определяется выражением:

$$p_{1m} = f(x_1, \theta_1) \dots f(x_m, \theta_1) \tag{1.1}$$

когда справедлива гипотеза H_1, выражением:

$$p_{0m} = f(x_1, \theta_0) \dots f(x_m, \theta_0) \tag{1.2}$$

когда справедлива гипотеза H_0.

Последовательный критерий отношения вероятностей для проверки гипотезы H_0 относительно H_1 определяется следующим образом. Выбираются две положительные величины A и B $(A > B)$. На каждой стадии эксперимента (в m-м для любого целого значения m) вычисляется отношение вероятностей $\frac{p_{1m}}{p_{0m}}$. Если

$$B < \frac{p_{1m}}{p_{0m}} < A \tag{1.3}$$

то эксперимент продолжается и производится дополнительное наблюдение. Если

$$\frac{p_{1m}}{p_{0m}} \geq A \tag{1.4}$$

то процесс оканчивается отклонением гипотезы H_0 (принятием H_1). Если

$$\frac{p_{1m}}{p_{0m}} \leq B \qquad (1.5)$$

то процесс оканчивается принятием гипотезы H_0.

Если для некоторой выборки $p_{1m} = p_{0m} = 0$, то будем приравнивать величину $\frac{p_{1m}}{p_{0m}}$ единице. Если для некоторой выборки $(x_1, \dots, x_m)$ получим $p_{1m} > 0$, но p_{0m}= 0, то неравенство (1.4) считается выполненным, и H_0 отвергается, и наоборот.

1.3 Последовательный тест, основанный на апостериорных вероятностях

Как упоминалось ранее, в случае двух простых гипотез оптимальный результат может быть достигнут при использовании последовательного теста отношения вероятностей. Проверим, будет ли оптимальным данный тест при M > 2, то есть в случае трех и более простых гипотезах.

С теоретической точки зрения, наиболее надежным тестом для принятия решения среди M гипотез является последовательная процедура Байеса. Этот тест является оптимальным в том смысле, что он сводит к минимуму риск Байеса для заданного набора функций стоимости и предыдущих вероятностей. К сожалению, с точки зрения реализации, сложность этого теста является серьезной проблемой. На каждом этапе последовательного процесса необходимо найти ожидаемый риск принятия решения, а также риск продолжения теста.

При наблюдениях, проводимых по одному за раз, каждый этап теста является решением задачи, включающей либо выбор дополнительного наблюдения, либо завершение последовательного теста. Ожидаемый риск, связанный с решением, достаточно легко определяется, когда тест завершается. Однако его трудно рассчитать при дополнительном наблюдении.

Будем считать, что $p_{0m}, p_{1m}, \dots, p_{lm}$ известны и пусть имеются априорные вероятности $p_i,\ i = 1,2, \dots, l$ этих гипотез:

$$\sum_{i=1}^{l} p_i = 1 \qquad (1.6)$$

Гипотезы можно интерпретировать как классы, тогда эти вероятности - шансы встретить представителя соответствующего класса.

После n-го наблюдения будем вычислять апостериорную вероятность $P(\theta = i|x_1, \dots, x_n)$ используя формулу Байеса:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \qquad (1.7)$$

где $P(B) = \sum_{i=1}^{N} P(A_i) * P(B|A_i)$.

В этом случае формула Байеса принимает вид:

$$P\left(A_j\middle|B\right) = \frac{P\left(B\middle|A_j\right) * P\left(A_j\right)}{\sum_{i=1}^{N} P(A_i) * P(B|A_i)} \qquad (1.8)$$

Будем сравнивать после каждого наблюдения апостериорные вероятности с заданным порогом. Будем обозначать порог через α.

Останавливаем тест, когда одна из них превзойдет порог, и выносим решение в пользу соответствующей гипотезы.

1.4 Матричный последовательный тест

Рассмотрим случай временных рядов с трендом [4].

Пускай $x_1, x_2, \dots x_n$ теперь будут наблюдениями временного ряда с трендом:

$$\boldsymbol{x_t = \theta^T \psi(t) + \xi_t, t \geq 1}, \qquad (1.9)$$

где $\psi(t) = \left(\psi_1(t), \dots, \psi_l(t)\right), t \geq 1$ — векторы базовой функции тренда, $\theta = (\theta_1, \dots, \theta_l)^T \in \mathrm{R}^l$ — неизвестный вектор коэффициентов и $\{\xi_t, t \geq 1\}$ —

последовательность независимых одинаково распределенных случайных величин при $N_1(0,\sigma^2)$. Гипотезы будут рассматриваться относительно вектора θ.

Обозначим

$$\boldsymbol{\Lambda_n(i,j) = ln\ (\prod_{t=1}^{n} \frac{n_1(x_t;(\theta_i)^T\psi(t),\sigma^2)}{n_1(x_t;(\theta_j)^T\psi(t),\sigma^2)})}; \qquad (1.10)$$

$$\tau_i = \inf\left\{n \in N:\ \Lambda_n(i,j) >\ b_{ij}, j \in \{1,\dots,M\}\backslash\{i\}\right\}, i \in 1,\dots,M,$$

где $B = (b_{ij}), i,j \in \{1,\dots,M\}$, предопределенная матрица тестовых порогов (с её помощью контролируется вероятность ошибки).

Имеется несколько вариантов задания этой матрицы, и каждый обеспечивает выполнение определенных критериев.

Момент остановки теста N_b и окончательное решение в пользу одной из гипотез d_b определяется следующим образом:

$$N_b = \min\{\tau_i : i \in \{1,\dots,M\}\},\quad d_b = \arg\min_{i\in\{1,\dots,M\}} \tau_i$$

ГЛАВА 2
ОСОБЕННОСТИ В КОМПЬЮЕТРНОЙ РЕАЛИЗАЦИИ

В данной главе исследуется процесс принятия решений для случая трех гипотез для случайных наблюдений с нормальным распределением. Также в разделе описаны основные моменты реализации. Рассмотрен отдельно случай для модели с выбросами.

2.1 Процесс принятия решений

Сформулируем три гипотезы. После заданного количества наблюдений начнем вычислять апостериорные вероятности.

Для принятия решения “какая гипотеза выбрана” необходимо, чтобы соответствующая апостериорная вероятность пересекла заданную границу.

Для оценки вероятностей ошибок проведем наш тест 10000 раз. Заведем 6 счетчиков, которые будут увеличиваться в зависимости от того, какая гипотеза была принята. Если справедлива гипотеза H_0 и P($H_1|H_0$) превысила порог, увеличиваем счетчик A1 (ошибки первого рода), если же P($H_2|H_0$) превысила порог – то увеличиваем B1(ошибки второго рода) . В случае, когда справедлива гипотеза H_1 и P($H_0|H_1$) превышает порог, увеличиваем счётчик A2, если же P($H_2|H_1$) превысила порог – то увеличиваем B2. В случае, когда справедлива гипотеза H_2 и P($H_0|H_2$) превышает порог, увеличиваем счётчик A3, если же P($H_1|H_2$) превысила порог – то увеличиваем B3.

Для нормального распределения вероятностей имеет место:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

где σ — среднеквадратичное отклонение (σ^2 — дисперсия), μ — математическое ожидание. Будем изучать наше распределение относительно математического ожидания, то есть мы знаем параметр $\sigma = 1$.

Для нормального распределения воспользуемся встроенным в язык Java классом, отвечающим за генерацию случайных чисел. Процедура new Random().nextGaussian() - возвращает следующее значение случайной величины, имеющей стандартное нормальное (гауссовское) распределение вероятностей

со средним значением 0 и стандартным отклонением 1 от последовательности этого генератора случайных чисел.

2.2 Ключевые моменты реализации последовательного теста, основанного на апостериорных вероятностях

С помощью процедур `getCurrentpba` и `getCurrentApostreors` вычисляем апостериорную вероятность, используя формулу Байеса:

```
protected void getCurrentpba(){
        double x=getRandom();
        for(int i=0;i<hypCount;i++) {
            pba[i] *= densityFunction(x, hypotesis[i]);
        }
    }
    protected void getCurrentApostreors(){
        double sum=0;
        for(int i=0;i<hypCount;i++){
            sum+=aprior[i]*pba[i];
        }
        for(int i=0;i<hypCount;i++){
            apostrior[i]=aprior[i]*pba[i]/sum;
        }
    }
```

Используя процедуры `makeStep` и `makeNSteps`, достигаем заданного количества наблюдений, после которых начинаем вычислять апостериорные вероятности:

```
    protected void makeStep(){
        currentstep++;
        getCurrentpba();
        getCurrentApostreors();
    }
    protected void makeNSteps(){
        for(int i=0;i<stepsBeforeBayes;i++){
            makeStep();
        }
    }
```

Процедура `checkPs` отвечает за остановку теста. Если апостериорная вероятность превысила заданный порог `breakpoint` – возвращаем номер наблюдения.

```
    protected int checkPs(){
        for(int i=0;i<hypCount;i++) {
            if(apostrior[i]>breakpoint)
                return (i+1);
```

```
        }
        return 0;
    }
```

Используя процедуру `generateTest`, реализуем последовательные тесты.

```
    public int[] generateTest(){
        makeNSteps();
        int x=checkPs();
        while(x==0){
            makeStep();
            x=checkPs();
        }
        int[] returner=new int[2];
        returner[0]=--x;
        returner[1]=currentstep;
        return returner;
    }
}
```

2.3 Моделирование искажений

Предположим, модель подвержена “выбросам”. Пусть ε – это вероятность появления “выбросов” (засорения) и пусть наблюдения x генерируются с вероятностью 1- ε из гипотетического распредления , а с вероятностью ε – дисперсия будет увеличиваться в κ раз (засоряющие распределение).

Процедура `getRandom` отвечает за реализацию этого случая.

```
  @Override
    public double getRandom(){
        if(epsilon>Math.random()){
            return (rg.nextGaussian()*variance*varianceCoeffi-
cient)+mean;
        }else{
            return getRandom();
        }
    }

}
```

2.4 Ключевые моменты реализации матричного последовательного теста

Используя процедуры `countMatrixCellMultiplication` и `countMatrixCell` вычислим элементы матрицы Λ_n, которая далее будет сравниваться с матрицей порогов.

```
private double countMatrixCellMultiplication(int i, int j, int n){
    double upper;
    double lower;
    double multiplication=1;
    for(int t=1; t<=n; t++){
        upper = n1(observations.get(t-1), hypothesisList[i],
countTrend(t));
        lower = n1(observations.get(t-1), hypothesisList[j],
countTrend(t));
        multiplication*=upper/lower;
    }
    return multiplication;
}

private double countMatrixCell(int i, int j, int n){
    return Math.log(countMatrixCellMultiplication(i, j, n));
}
```

Процедура `isExceedingMatrixB` поэлементно сравнивает матрицу Λ_n с матрицей порогов B:

```
protected boolean isExceedingMatrixB(double[][] matrix){
    for(int i = 0; i< numberOfHypothesis; i++){
        for(int j = 0; j< numberOfHypothesis; j++){
            if(matrix[i][j] > matrixB[i][j]){
                if(i!=j){
                    return true;
                }
            }
        }
    }
    return false;
}
```

За остановку теста и вынесение окончательного решения в пользу одной из гипотез, то есть за определение N_b и d_b отвечают процедуры `getTiList` и `chooseHypothesis`

```
private double[] getTiList(double[][] matrix){
    double[] tiList = new double[numberOfHypothesis];
    for(int i = 0; i< numberOfHypothesis; i++){
```

```
            double ti = 9999999;
            for(int j = 0; j< numberOfHypothesis; j++){
                if(matrix[i][j] > matrixB[i][j]){
                    if(i!=j){
                        if(matrix[i][j]<ti){
                            ti=matrix[i][j];
                        }
                    }
                }
            }
            tiList[i]=ti;
        }
        return tiList;
    }


    private int chooseHypothesis(double[] tiList){
        double nb = 9999999;
        for(int i = 0; i < numberOfHypothesis; i++){
            if(tiList[i]<nb){
                nb=tiList[i];
            }
        }
        for(int i = 0; i< numberOfHypothesis; i++){
            if(tiList[i]==nb){
                return i;
            }
        }
        return -1;
    }
```

ГЛАВА 3
КОМПЬЮТЕРНЫЕ ЭКСПЕРИМЕНТЫ И АНАЛИЗ РЕЗУЛЬТАТОВ

В настоящей главе приводятся результаты компьютерных экспериментов, осуществленных на языке *Java*, согласно теории описанной ранее. Анализируются вероятности ошибок и среднее число наблюдений в зависимости от величины порога и априорных вероятностей. Реализация 3D графиков осуществлена с помощь системы *Matlab*.

3.1 Влияние изменений величины порога и априорных вероятностей на вероятности ошибок и среднее число наблюдений для последовательного теста, основанного на апостериорных вероятностях

Рассмотрим случай, когда значение параметра (мат. ожидание) гипотез H_0, H_1, H_2 равны 0, 0.2 и 0.4 соответственно, вероятности p_0, p_1, p_2 равны 0.33, 0.33,0.34 соответственно и порог для апостериорной вероятности равен 0.9.

Таблица 2.1 – Результаты экспериментов при $\boldsymbol{\theta_0 = 0, \theta_1 = 0.2, \theta_2 = 0.4}$, $\boldsymbol{p_0 = 0.33, p_1 = 0.33, p_2 = 0.34}$ и $\boldsymbol{\alpha = 0.9}$

	Первая гипотеза		Вторая Гипотеза		Третья гипотеза	
Вероятность ошибок *x*–го рода	2	3	1	3	1	2
	0.0388	6.0E-4	0.0925	0.097	0.001	0.0424
Среднее количество наблюдений	105		176		104	

Примечание. Источник : собственная разработка по данным приложения А.

Заметим, что для первой гипотезы вероятность ошибок второго рода больше вероятности ошибок третьего рода. Это обосновывается тем, что шансы встретить представителя соседнего класса (в данном случае – вторая гипотеза) для первой гипотезы значительно выше, чем шансы встретить представителя не соседнего класса (в данном случае – третья гипотеза). Для третьей гипотезы ситуация аналогична. В случае, когда справедлива вторая гипотеза, ошибки первого и третьего рода почти равны. Это следует из того,

что первая и третья гипотеза являются соседними классами для второй гипотезы.

Среднее количество шагов (наблюдений) для “центральной” гипотезы больше, чем для “крайних” гипотез. Это обосновывается тем, что у второй гипотезы имеются два соседних класса и, следовательно, значительная часть решений выносится не в пользу второй гипотезы. За счёт этого снижается точность теста и для достижения заданной точности, которая определяется с помощью порога, необходимо совершать больше наблюдений.

Посмотрим, как влияют изменения значения порога на результат.

Таблица 2.2 – Результаты экспериментов при $\boldsymbol{\theta_0 = 0, \theta_1 = 0.2, \theta_2 = 0.4}$, $\boldsymbol{p_0 = 0.33, p_1 = 0.33, p_2 = 0.34}$ и $\boldsymbol{\alpha = 0.95}$ и $\boldsymbol{\alpha = 0.8}$

Порог равен 0.95	Порог равен 0.8
Первая гипотеза A1:0.0236 B1:1.0E-4 Среднее количество наблюдений:146 ---------- Вторая гипотеза A2:0.0497 B2:0.044 Среднее количество наблюдений:238 ---------- Третья гипотеза A3:0.001 B3:0.0204 Среднее количество наблюдений:146	Первая гипотеза A1:0.0719 B1:0.01 Среднее количество наблюдений:63 ---------- Вторая гипотеза A2:0.1944 B2:0.2014 Среднее количество наблюдений:102 ---------- Третья гипотеза A3:0.0104 B3:0.0684 Среднее количество наблюдений:63

Примечание. Источник : собственная разработка по данным приложения А.

Нетрудно заметить, что при увеличении порога среднее количество шагов (наблюдений) возрастает, а вероятности ошибок уменьшаются.

Рассмотрим случай, когда значения параметров для гипотез H_0, H_1, H_2 равны 0, 0.1 и 0.2 соответственно, вероятности p_0, p_1, p_2 равны 0.33, 0.33,0.34 соответственно и порог равен 0.9 и посмотрим, на что влияет уменьшение “расстояния” между гипотезами.

Таблица 2.3 – Результаты экспериментов при $\boldsymbol{\theta_0 = 0, \theta_1 = 0.1, \theta_2 = 0.2}$, $\boldsymbol{p_0 = 0.33, p_1 = 0.33, p_2 = 0.34}$ и $\boldsymbol{\alpha = 0.9}$

	Первая гипотеза		Вторая Гипотеза		Третья гипотеза	
Вероятность ошибок х–го рода	2	3	1	3	1	2
	0.0451	3.0E-4	0.0725	0.054	0.0003	0.0321
Среднее количество шагов	613		776		635	

Примечание. Источник : собственная разработка по данным приложения А.

Из результатов, указанных в таблице 2.3, можно увидеть, что при уменьшении параметра Δ (Δ– параметр, который характеризует удаленность гипотез) количество наблюдений значительно возрастает. Это объясняется тем, что при достаточной “близости” гипотез достижение заданной точности становится более трудоемкой задачей. Так как зачастую решения могут выноситься не в пользу правильной гипотезы, то за счёт большего количества наблюдений мы можем добиться заданной точности.

Посмотрим, влияет ли изменение порога на результат.

Таблица 2.4 – Результаты экспериментов при $\boldsymbol{\theta_0 = 0, \theta_1 = 0.1, \theta_2 = 0.2}$, $\boldsymbol{p_0 = 0.33, p_1 = 0.33, p_2 = 0.34}$ и $\boldsymbol{\alpha = 0.95}$ и $\boldsymbol{\alpha = 0.8}$

Порог равен 0.95	Порог равен 0.8
Первая гипотеза A1:0.0336 B1:3.0E-4 Среднее количество наблюдений:806 ---------- Вторая гипотеза A2:0.0447 B2:0.024 Среднее количество наблюдений:938 ---------- Третья гипотеза A3:0.009 B3:0.0144 Среднее количество наблюдений:846	Первая гипотеза A1:0.0619 B1:0.005 Среднее количество наблюдений:683 ---------- Вторая гипотеза A2:0.1344 B2:0.1004 Среднее количество наблюдений:997 ---------- Третья гипотеза A3:0.0133 B3:0.0552 Среднее количество наблюдений:763

Примечание. Источник : собственная разработка по данным приложения А.

Как и ожидалось, вне зависимости от Δ, увеличение порога ведет к увеличению наблюдений и уменьшению вероятности ошибок.

Теперь рассмотрим случай, когда значения параметров для гипотез H_0, H_1, H_2 равны 0, 0.2 и 0.4 соответственно и порог равен 0.9, но изменим вероятности p_0, p_1, p_2. Пускай априорные вероятности равняются 0.6, 0.2 и 0.2 соотвественно в первом случае; 0.2, 0.6, 0.2 во втором случае и 0.2, 0.2 и 0.6 в третьем случае.

Таблица 2.5 – Результаты экспериментов при $\boldsymbol{\theta_0 = 0, \theta_1 = 0.2, \theta_2 = 0.4}$ и $\boldsymbol{\alpha = 0.9}$

		Первая гипотеза		Вторая Гипотеза		Третья гипотеза	
$\boldsymbol{p_0 = 0.6}$ $\boldsymbol{p_1 = 0.2}$ $\boldsymbol{p_2 = 0.2}$	Вероятность ошибок *x*– го рода	2	3	1	3	1	2
		0.0062	0.001	0.267	0.094	0.024	0.0424
	Среднее количество наблюдений	60		169		106	
$\boldsymbol{p_0 = 0.2}$ $\boldsymbol{p_1 = 0.6}$ $\boldsymbol{p_2 = 0.2}$	Вероятность ошибок *x*– го рода	2	3	1	3	1	2
		0.1547	6.0E-4	0.026	0.025	4.1E-5	0.1349
	Среднее количество наблюдений	133		131		133	
$\boldsymbol{p_0 = 0.6}$ $\boldsymbol{p_1 = 0.2}$ $\boldsymbol{p_2 = 0.2}$	Вероятность ошибок *x*– го рода	2	3	1	3	1	2
		0.0351	0.0189	0.0865	0.2641	0.001	0.0189
	Среднее количество наблюдений	108		167		63	

Проанализируем результаты в случае, когда априорная вероятность $\boldsymbol{p_0}$ превосходит в несколько раз априорные вероятности $\boldsymbol{p_1}$ и $\boldsymbol{p_2}$. Отметим,что в случае, когда первая гипотеза является верной среднее число количества

наблюдений и вероятности ошибок второго и третьего рода значительно снижаются. Однако для случаев, когда верными являются вторая и третья гипотеза вероятность ошибок первого и третьего, и первого и второго рода соотвественно возрастает. Аналогичное поведение мы можем отметить и в случае, когда априорная вероятность $\boldsymbol{p_2}$ превосходит в несколько раз априорные вероятности $\boldsymbol{p_0}$ и $\boldsymbol{p_1}$ – единственным отличием является то, что результаты улучшаются для случая, когда верной является третья гипотеза и ухудшаются для случаев, когда верными является первая и вторая гипотеза. Для случая когда априорная вероятность $\boldsymbol{p_1}$ превосходит в несколько раз априорные вероятности $\boldsymbol{p_0}$ и $\boldsymbol{p_2}$ количество необходимых наблюдений снижается для "центральной" гипотезы и возрастает для "крайних" гипотез. Вероятности ошибок первого и третьего рода в случае, когда "центральная" гипотеза является верной уменьшаются, но для случаев, когда верными являются первая и третья гипотеза вероятности ошибок второго и третьего, и первого и второго рода соотвественно возрастают.

Из вышесказанного можем сделать вывод, что скос априорных вероятностей в одну из сторон улучшает результаты для той гипотезы, в чью сторону был произведен скос, однако ухудшает результаты для остальных гипотез.

3.2 Влияние изменений величины порога на вероятности ошибок и среднее число наблюдений для матричного последовательного теста

Проанализируем насколько устойчив матричный тест к изменению величины порога. Пускай значения параметра (мат. ожидание) гипотез H_0, H_1, H_2 равны 0, 0.2 и 0.4 соответственно и порог для апостериорной вероятности равен 30.

Таблица 2.6 – Результаты экспериментов при $\boldsymbol{\theta_0 = 0, \theta_1 = 0.2, \theta_2 = 0.4}$ и $\boldsymbol{b = 30}$

	Первая гипотеза		Вторая Гипотеза		Третья гипотеза	
Вероятность ошибок *x*–го рода	2	3	1	3	1	2
	0.0135	0.0005	0.0417	0.0557	0.0004	0.0241
Среднее количество наблюдений	57		88		52	

Примечание. Источник : собственная разработка по данным приложения А.

Можем отметить, что как и в предыдущей подглаве количество шагов для второй (центральной) гипотезы больше, чем для “крайних” гипотез, так как из-за наличия двух “соседних” классов часть решений выносится в их пользу. Однако по сравнению с тестом, основанным на апостериорных вероятностях матричному тесту требуется значительно меньшее количество наблюдений. Это можно объяснить тем, что он рассчитывается не на «наихудший» случай, а рассматривает все варианты порогов для отношения вероятностей.

Так же можем наблюдать, что вероятности ошибок второго рода и третьего рода для первой гипотезы, первого и третьего рода для второй гипотезы и первого и второго рода для третьей гипотезы, посчитанные с помощью теста, основанного на апостериорных вероятностях значительно меньше, чем соотвествующие вероятности посчитанные матричным последовательным тестом. Это связанно с тем фактом, что используется первым большее число наблюдений (что на практике часто неприемлемо) и он расчитан на «наихудший» случай.

Посмотрим, влияет ли изменение порога на результат.

Таблица 2.7 – Результаты экспериментов при $\boldsymbol{\theta_0 = 0}$, $\boldsymbol{\theta_1 = 0.2}$, $\boldsymbol{\theta_2 = 0.4}$, $\boldsymbol{b =}$ $\boldsymbol{40}$ и $\boldsymbol{b = 15}$

Порог равен 40	Порог равен 15
Первая гипотеза A1:0.0042 B2:3.E-6 Среднее количество наблюдений:71 ---------- Вторая гипотеза A2:0.0327 B2:0.0296 Среднее количество наблюдений:114 ---------- Третья гипотеза A3: 6.5.E-5 B3:0.0079 Среднее количество наблюдений:72	Первая гипотеза A1:0.0324 B1:0.0009 Среднее количество наблюдений:28 ---------- Вторая гипотеза A2:0.0746 B2:0.0801 Среднее количество наблюдений:43 ---------- Третья гипотеза A3:0.0007 B3:0.0381 Среднее количество наблюдений:29

Примечание. Источник : собственная разработка по данным приложения А.

Из результатов представленных в таблице 2.7 следует, что между величиной порога и средним количеством наблюдений существует зависимость, а именно при росте порога количество наблюдений возрастает, при уменьшении порога – падает. Вероятности ошибок же увеличиваются при уменьшении порога.

Если сравнить результаты представленные в таблице 2.2 с результатами представленными в таблице 2.7, то мы снова можем отметить тот факт, что матричный последовательный тест добился более точных результатов, используя в среднем меньшее количество наблюдений. Кроме того из результатов следует, что матричный тест является более устойчивым к изменению величины порога.

3.3 Влияние искажений на вероятности ошибок для теста, основанного на апостериорных вероятностях

Проанализируем влияние выбросов на модель. Рассмотрим случай с четырьмя гипотезами и построим 3D-зависимости вероятности (доли) решений в пользу каждой из гипотез от ε и Δ при некоторых фиксированных k. Δ– параметр, который характеризует удаленность гипотез.

Пусть верным решением является принятие гипотезы H_0. Поверхность синего цвета будет описывать поведение вероятности принятия правильного решения, оранжевого – вероятности ошибок 2-го рода, зеленого – вероятности ошибок 3-го рода, красного – вероятности ошибок 4-го рода.

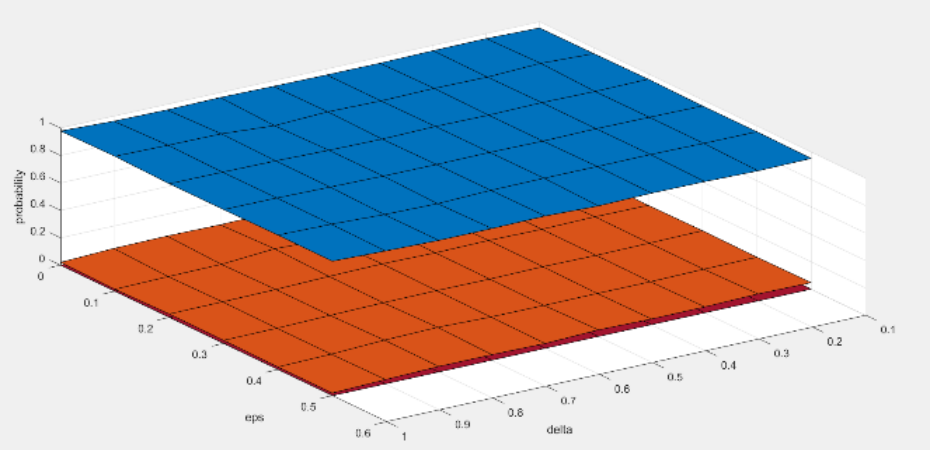

Рисунок 3.1 –Вероятности ошибок второго, третьего и четвертого рода при k=1 в случае, когда верным решением является принятие гипотезы H_0

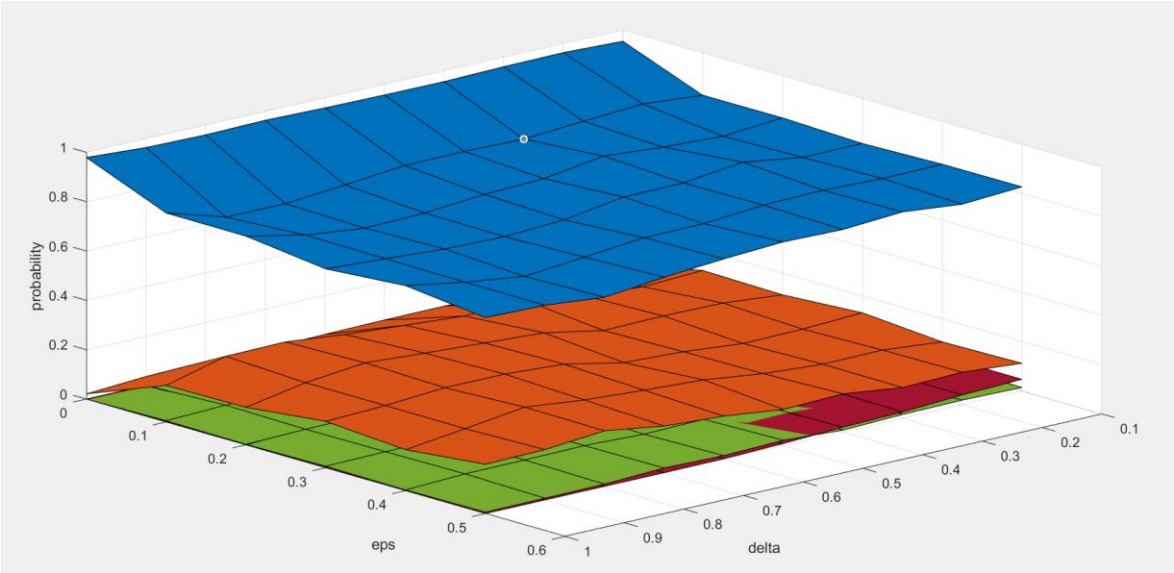

Рисунок 3.2 –Вероятности ошибок второго, третьего и четвертого рода при k=2 в случае, когда верным решением является принятие гипотезы H_0

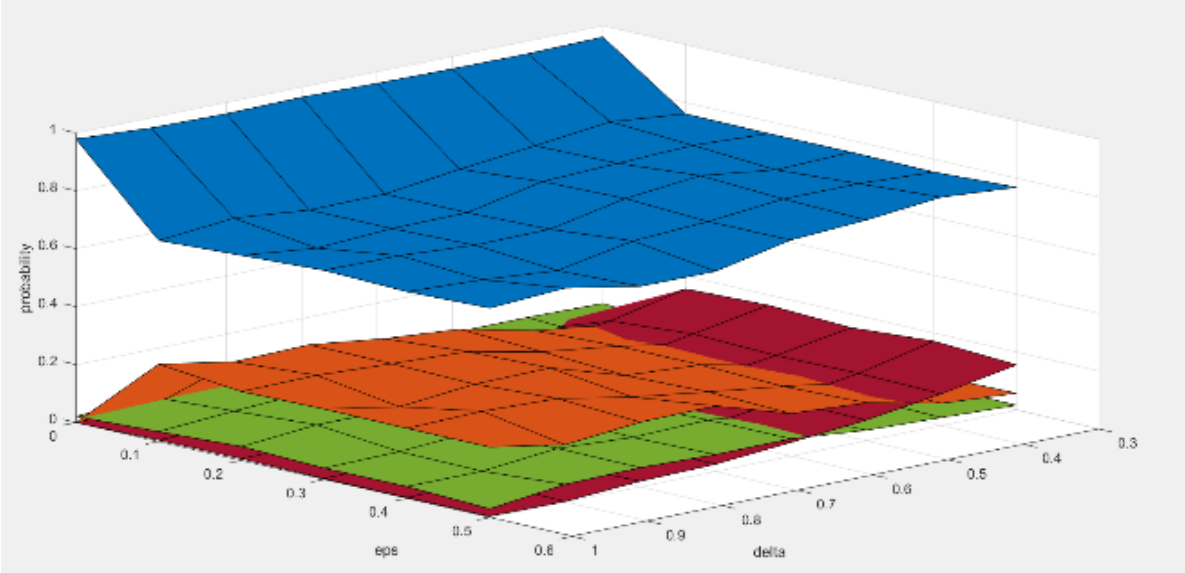

Рисунок 3.3 – Вероятности ошибок второго, третьего и четвертого рода при k=3 в случае, когда верным решением является принятие гипотезы H_0

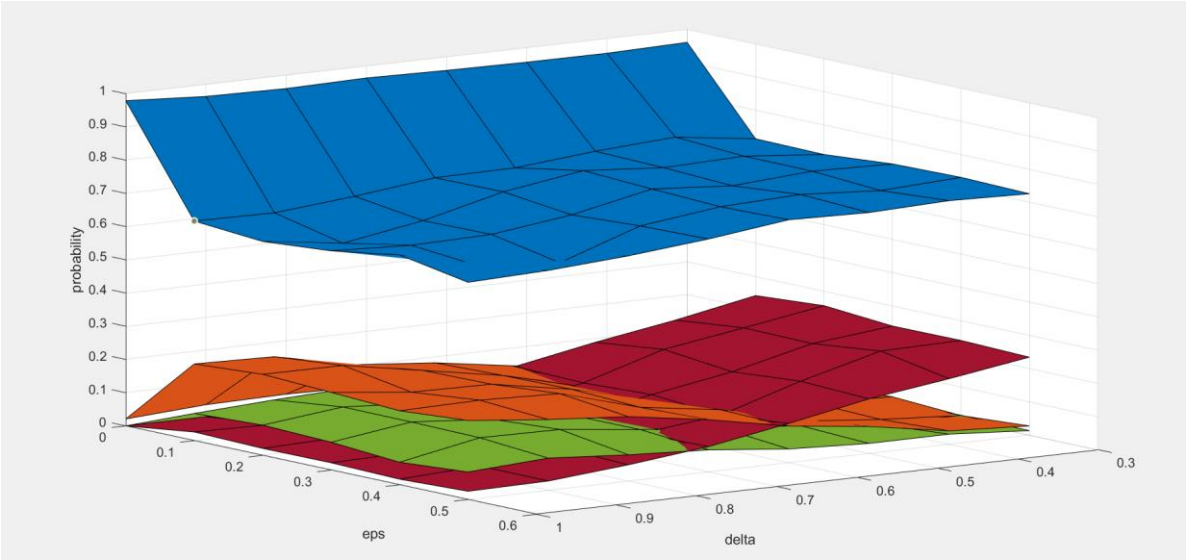

Рисунок 3.4 – Вероятности ошибок второго, третьего и четвертого рода при k=4 в случае, когда верным решением является принятие гипотезы H_0

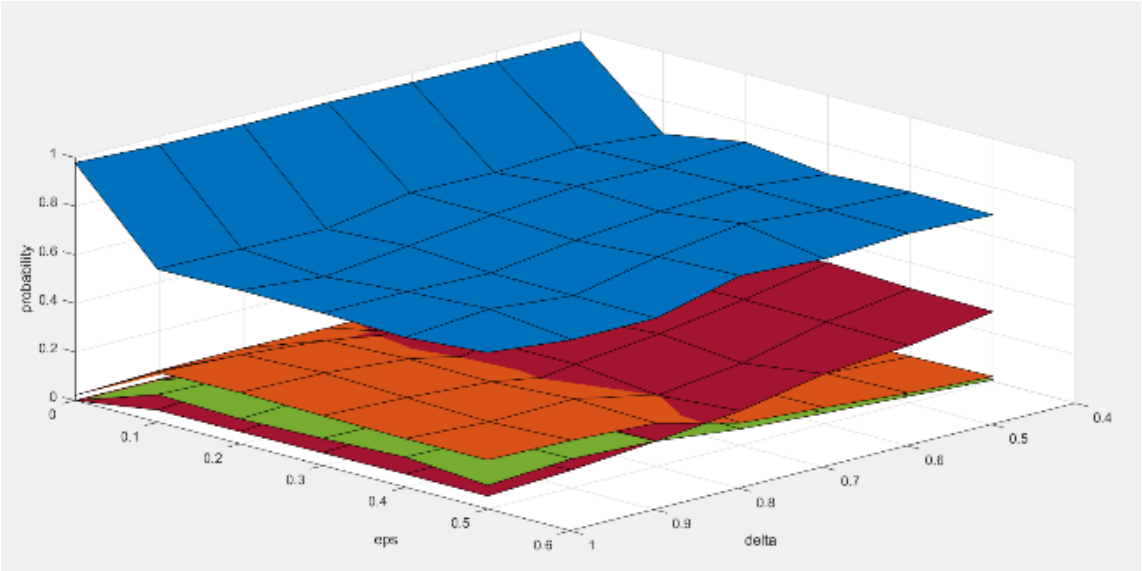

Рисунок 3.5 – Вероятности ошибок второго, третьего и четвертого рода при k=5 в случае, когда верным решением является принятие гипотезы H_0

Стоит также отметить, что искажения сильнее влияют на “не крайние” гипотезы. Это хорошо видно, если проанализировать результаты экспериментов над второй гипотезой.

Пусть верным решением является принятие гипотезы H_1. Здесь же поверхность оранжевого цвета описывает поведение вероятности принятия

правильного решения, синего – вероятности ошибок первого рода, зеленого – вероятности ошибок третьего рода, красного – вероятности ошибок четвертого рода.

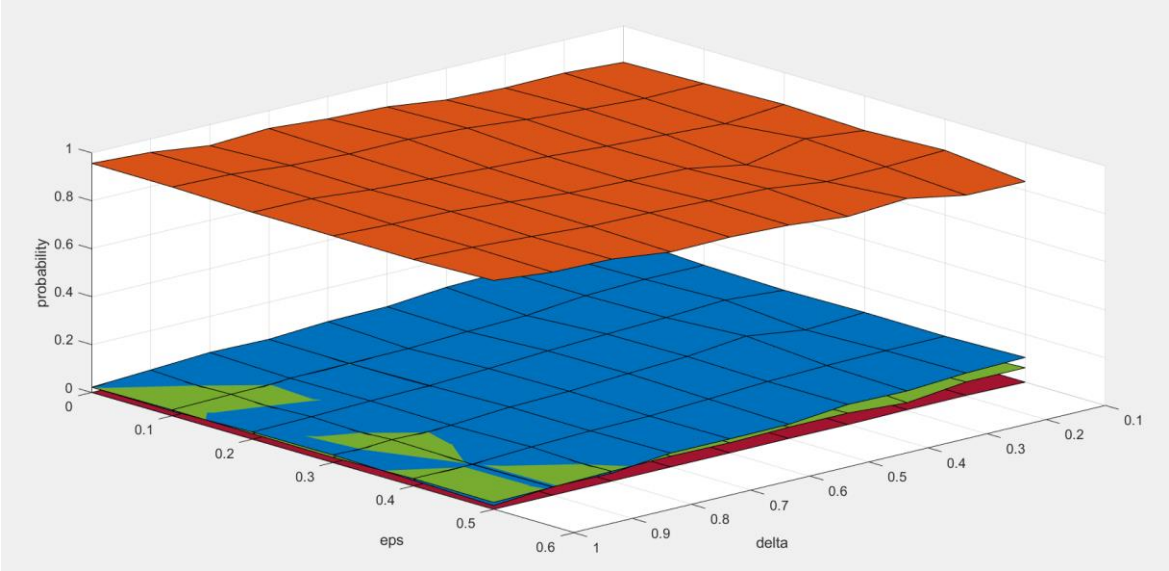

Рисунок 3.6 – Вероятности ошибок первого, третьего и четвертого рода при k=1 в случае, когда верным решением является принятие гипотезы H_1

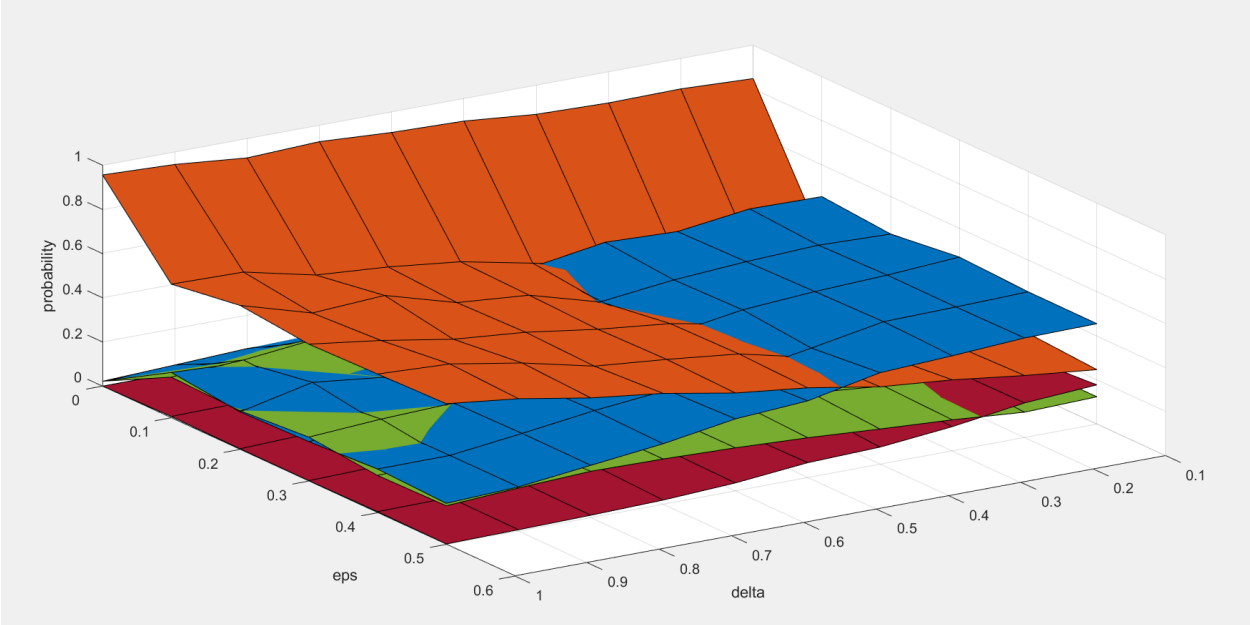

Рисунок 3.7 – Вероятности ошибок первого, третьего и четвертого рода при k=2 в случае, когда верным решением является принятие гипотезы H_1

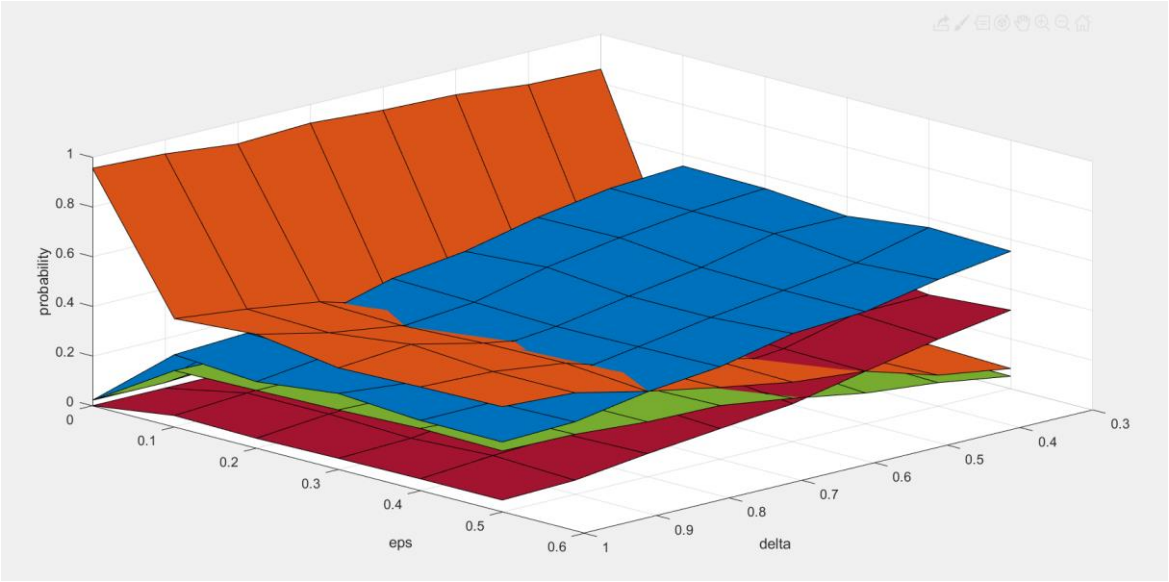

Рисунок 3.8 – Вероятности ошибок первого, третьего и четвертого рода при k=3 в случае, когда верным решением является принятие гипотезы H_1

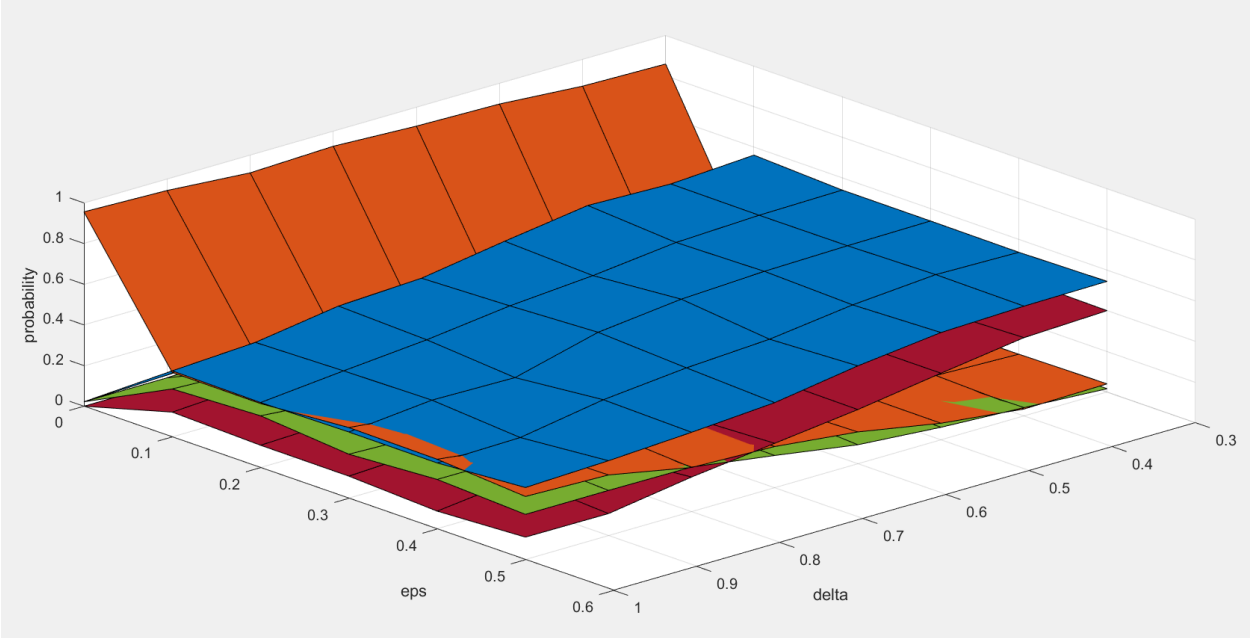

Рисунок 3.9 – Вероятности ошибок первого, третьего и четвертого рода при k=4 в случае, когда верным решением является принятие гипотезы H_1

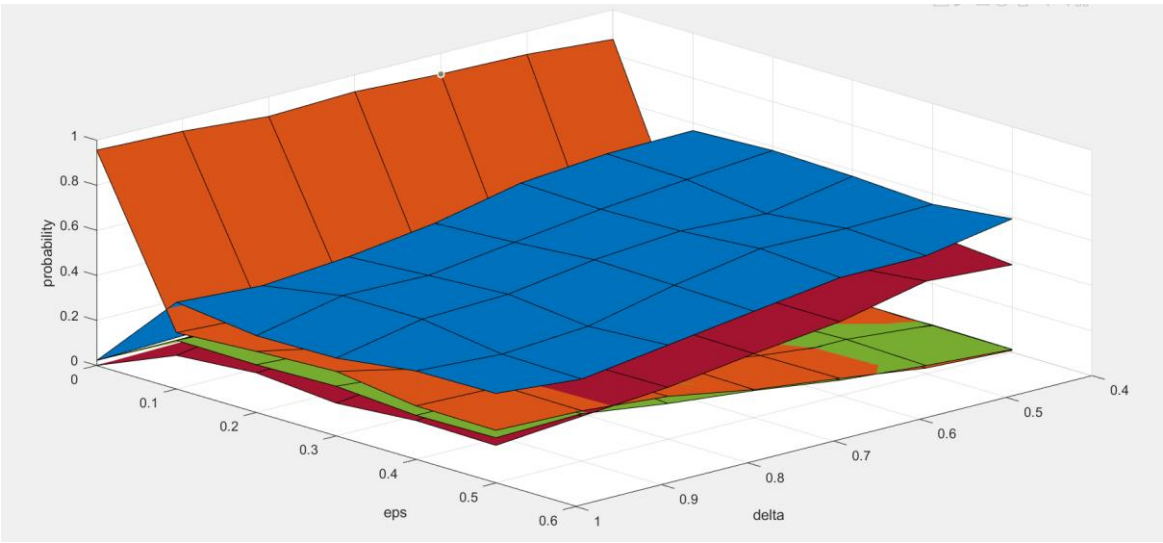

Рисунок 3.10 – Вероятности ошибок первого, третьего и четвертого рода при k=5 в случае, когда верным решением является принятие гипотезы H_1

Проанализировав Рисунки 3.1 – 3.10 можно увидеть, что при увеличении параметра k вероятность принятия правильного решения уменьшается, а вероятности ошибок возрастают.

Увеличение Δ влечет за собой увеличение вероятности принятия правильных решений. Это объясняется тем фактом, что при удалении друг от друга уменьшаются шансы ошибочно принять за первую гипотезу вторую (т.е. соседнюю для первой) или же за вторую-первую или третью (т.е. соседние для второй) и так далее.

Как упоминалось в главе 2.3, наша модель подвержена выбросам, то есть с вероятностью ε дисперсия будет увеличиваться в κ раз. Следовательно, на Рисунках 3.1 - 3.5 можно наблюдать, что при росте ε вероятности ошибок второго, третьего и четвертого рода, а на Рисунках 3.6 - 3.10 вероятности ошибок первого, третьего и четвертого рода растут.

Теперь же построим 3D-зависимости вероятности (доли) решений в пользу каждой из гипотез от ε и k при некоторых фиксированных Δ для теста, основанного на апостериорных вероятностях.

Как и ранее, рассмотрим два случая : случай, когда верна гипотеза H_0 (Рисунки 3.11 – 3.15) и случай, когда верна гипотеза H_1 (Рисунки 3.16 – 3.20).

В первом случае, поверхность синего цвета будет описывать поведение вероятности принятия правильного решения, оранжевого – вероятности ошибок 2-го рода, зеленого – вероятности ошибок 3-го рода, красного – вероятности ошибок 4-го рода.

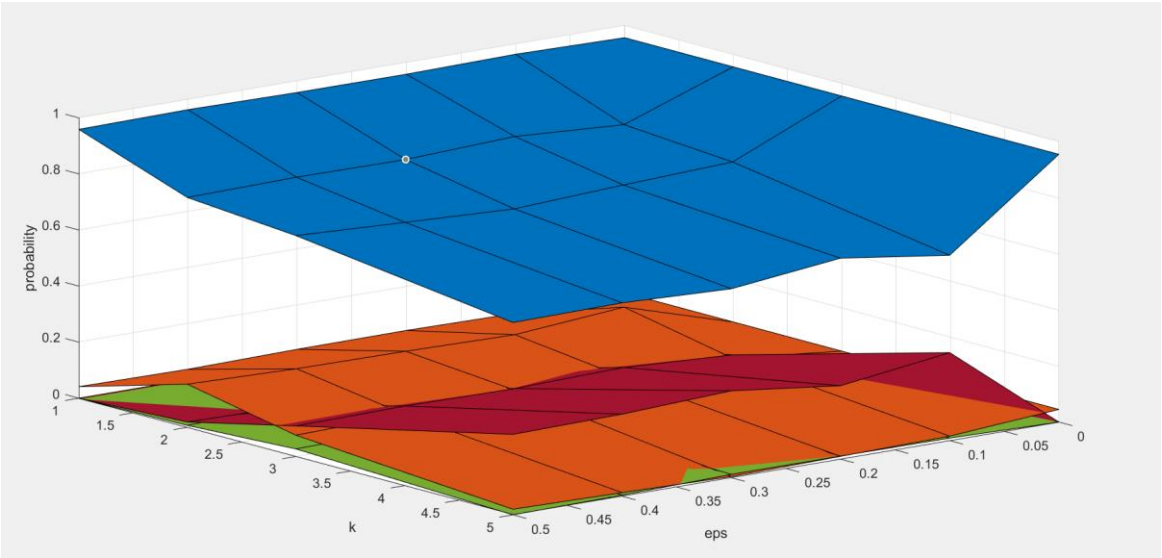

Рисунок 3.11 – Вероятности ошибок первого, третьего и четвертого рода при Δ=0.2 в случае, когда верным решением является принятие гипотезы H_0

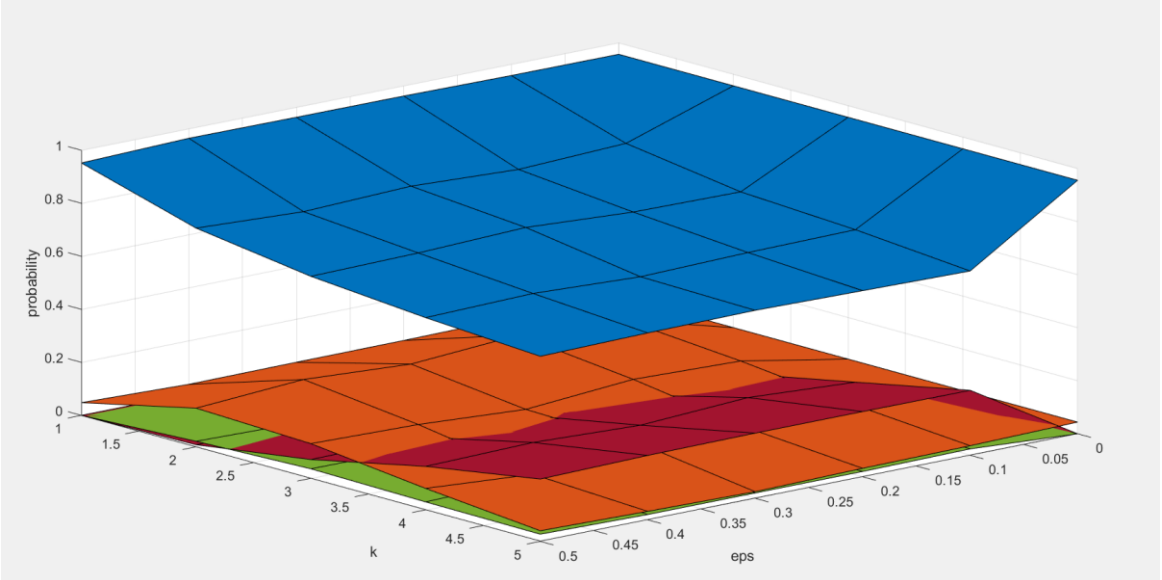

Рисунок 3.12 – Вероятности ошибок первого, третьего и четвертого рода при Δ=0.4 в случае, когда верным решением является принятие гипотезы H_0

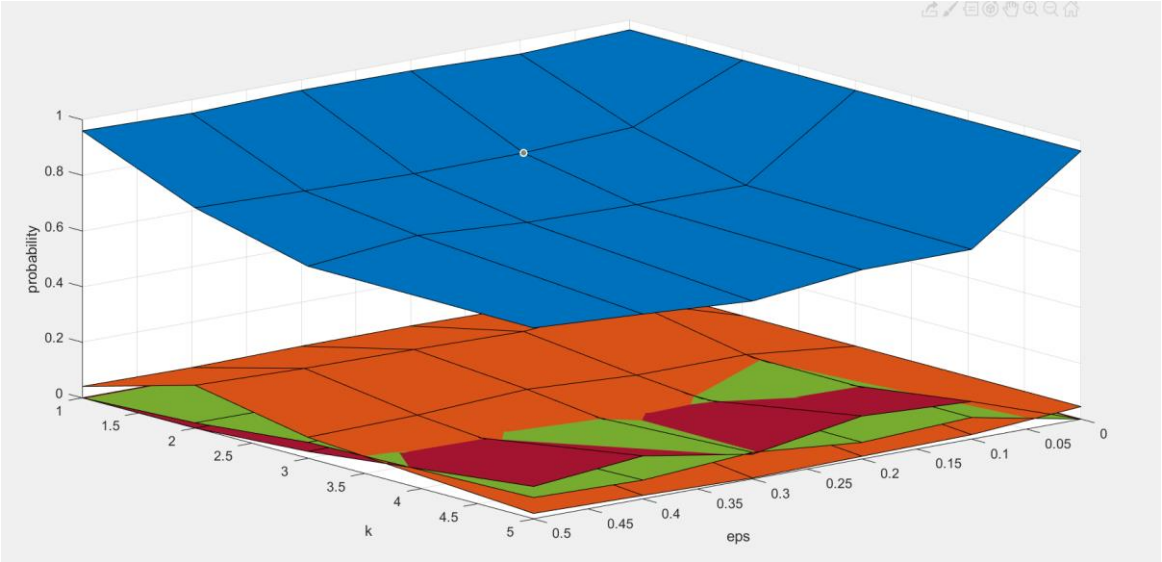

Рисунок 3.13 – Вероятности ошибок первого, третьего и четвертого рода при Δ=0.6 в случае, когда верным решением является принятие гипотезы H_0

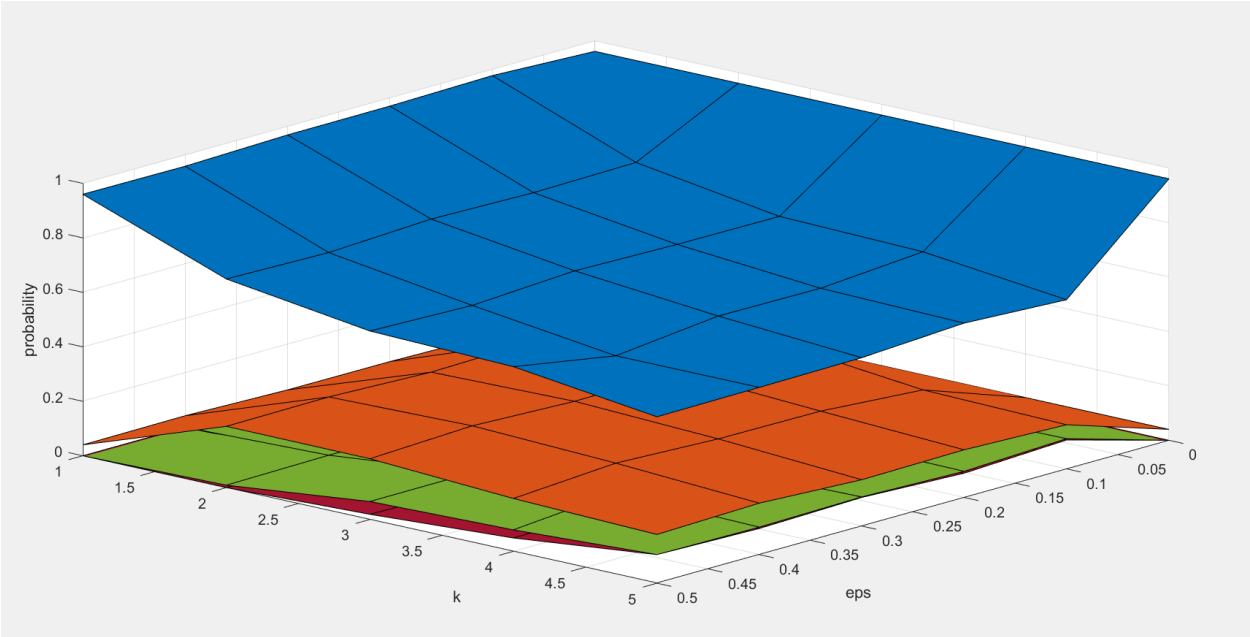

Рисунок 3.14 – Вероятности ошибок первого, третьего и четвертого рода при Δ=0.8 в случае, когда верным решением является принятие гипотезы H_0

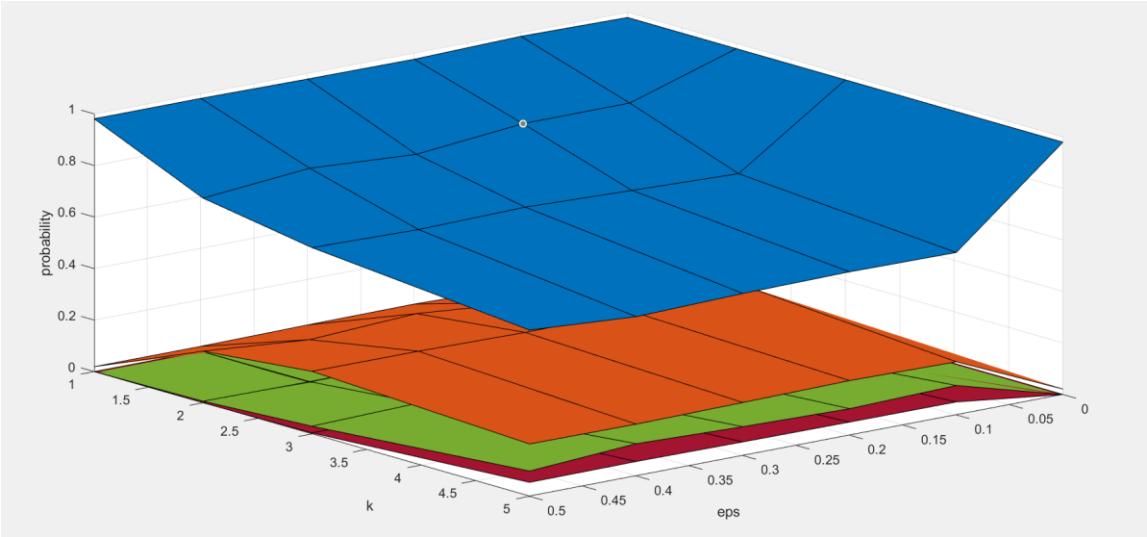

Рисунок 3.15 – Вероятности ошибок первого, третьего и четвертого рода при Δ=1 в случае, когда верным решением является принятие гипотезы H_0

Во втором случае, поверхность оранжевого цвета будет описывать поведение вероятности принятия правильного решения, синего – вероятности ошибок первого рода, зеленого – вероятности ошибок третьего рода, красного – вероятности ошибок четвертого рода.

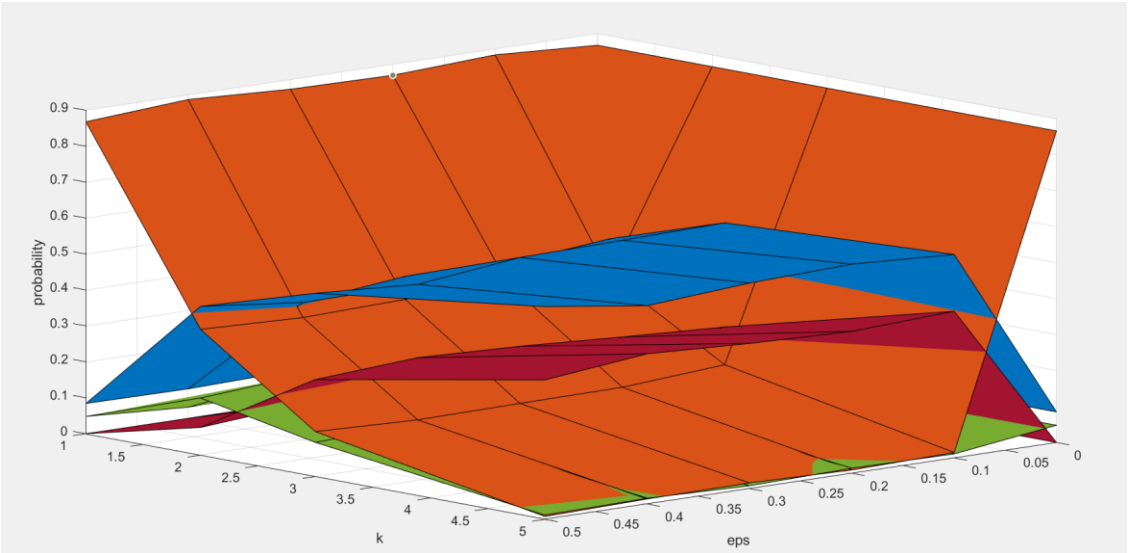

Рисунок 3.16 – Вероятности ошибок второго, третьего и четвертого рода при Δ=0.2 в случае, когда верным решением является принятие гипотезы H_1

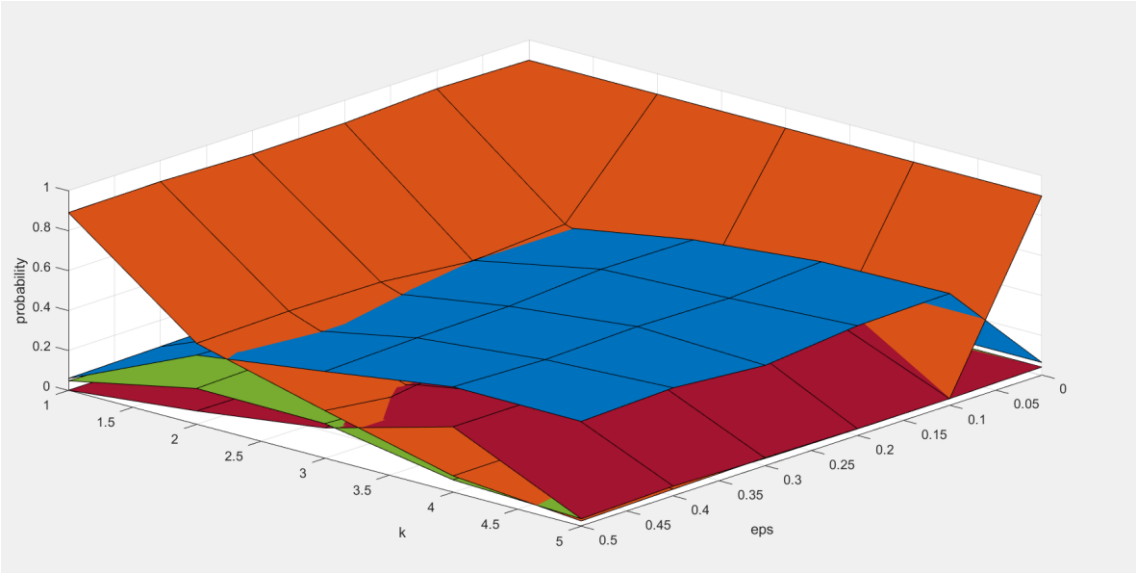

Рисунок 3.17 – Вероятности ошибок второго, третьего и четвертого рода при Δ=0.4 в случае, когда верным решением является принятие гипотезы H_1

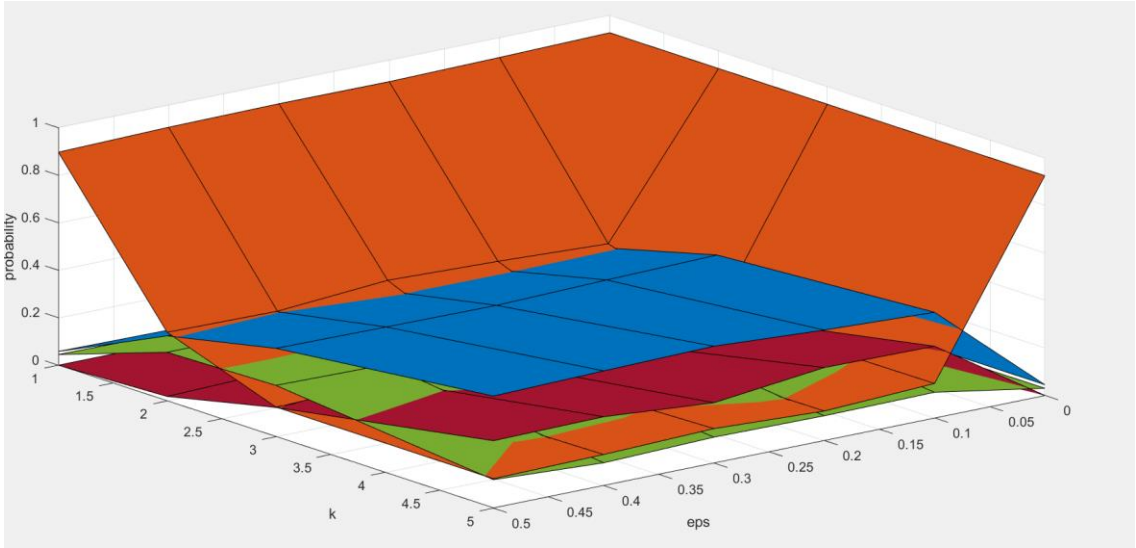

Рисунок 3.18 – Вероятности ошибок второго, третьего и четвертого рода при Δ=0.6 в случае, когда верным решением является принятие гипотезы H_1

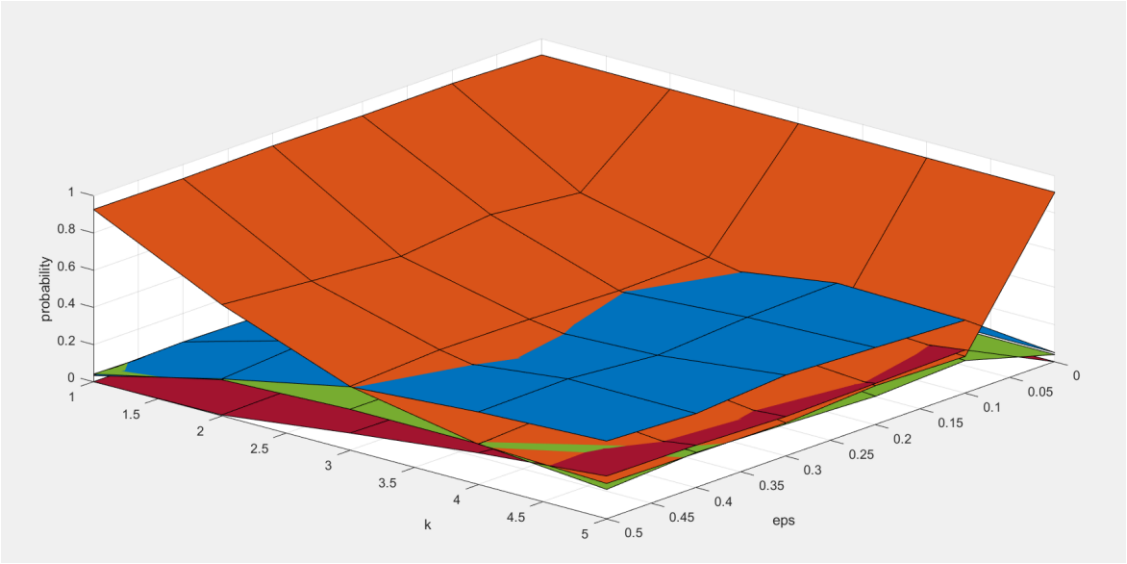

Рисунок 3.19 – Вероятности ошибок второго, третьего и четвертого рода при Δ=0.8 в случае, когда верным решением является принятие гипотезы H_1

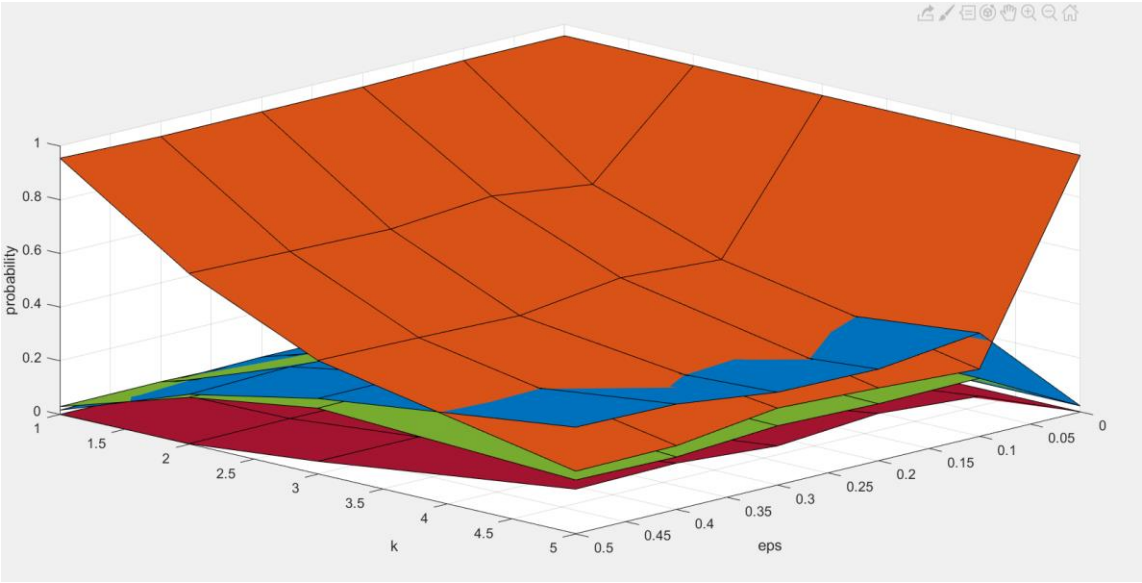

Рисунок 3.20 – Вероятности ошибок второго, третьего и четвертого рода при Δ=1 в случае, когда верным решением является принятие гипотезы H_1

Как отмечалось ранее, удаление гипотез друг от друга (увеличение параметра Δ) положительно влияет на результат – вероятность принятия правильного решения возрастает.

На рисунках 3.16 – 3.20 можно наблюдать, что с увеличением параметра k “засорение” значительнее ухудшает результаты.

Так же, можно отметить, что с ростом величины ε увеличивается количество выбросов, а следовательно сильно искажаются результаты вычислительных экспериментов.

Рисунки 3.16 – 3.20 дают понять, что, к сожалению, нельзя говорить об оптимальности данного теста в случае более двух простых гипотез. Вероятность принятия правильного решения не достигает даже 0.2 при Δ=1.

Заметим, что результаты для случая, когда верным решением является принятие гипотезы H_3, симметричны результатам случая, когда верным решением является принятие гипотезы H_0, а результаты случая, когда верным решением является принятие гипотезы H_1, симметричны результатам случая, когда верным решением является принятие гипотезы H_2.

3.4 Влияние искажений на вероятности ошибок для последовательного матричного теста

Проанализируем насколько матричный тест устойчив к искажениям. Рассмотрим случай с тремя гипотезами и построим 3D-зависимости вероятности (доли) решений в пользу каждой из гипотез от ε и k.

В первом случае, поверхности синего цвета будет соответствовать поведение вероятности принятия правильного решения, оранжевого – вероятности ошибок 2-го рода, зеленого – вероятности ошибок 3-го рода. Во втором случае, поверхности синего цвета будет соответствовать поведение вероятности ошибок 1-го рода, оранжевого – вероятности принятия правильного решения, зеленого – вероятности ошибок 3-го рода. В третьем случае, поверхности синего цвета будет соответствовать поведение вероятности ошибок 1-го рода, оранжевого – вероятности ошибок 2-го рода, зеленого – вероятности принятия правильного решения.

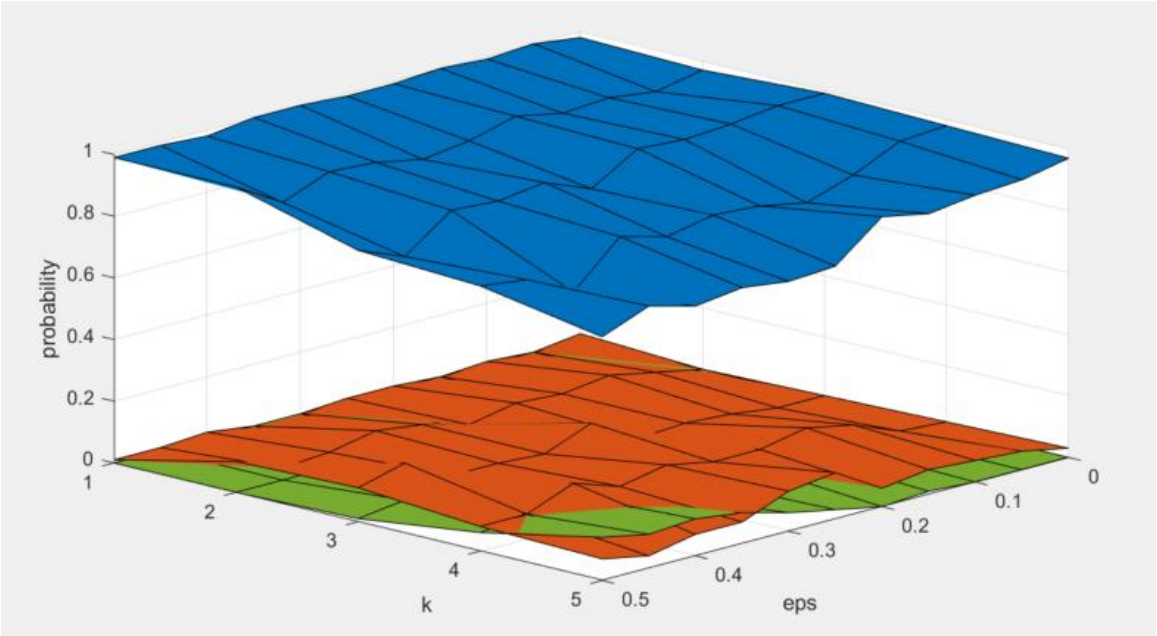

Рисунок 3.21 – Вероятности ошибок второго и третьего рода Δ=0.2 в случае, когда верным решением является принятие гипотезы H_0

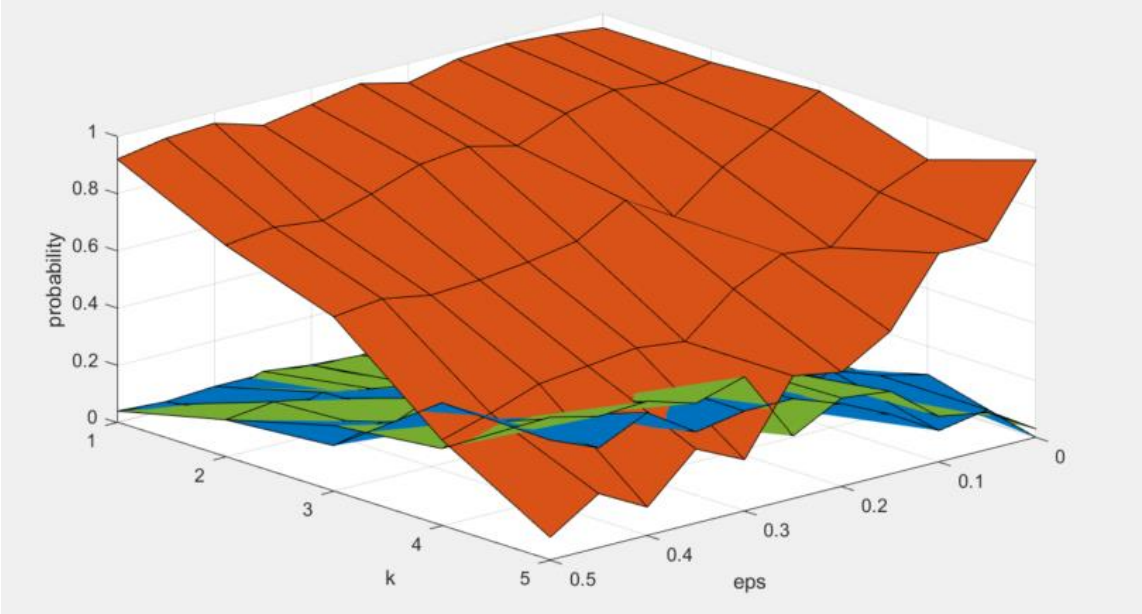

Рисунок 3.22 – Вероятности ошибок первого и третьего рода Δ=0.2 в случае, когда верным решением является принятие гипотезы H_1

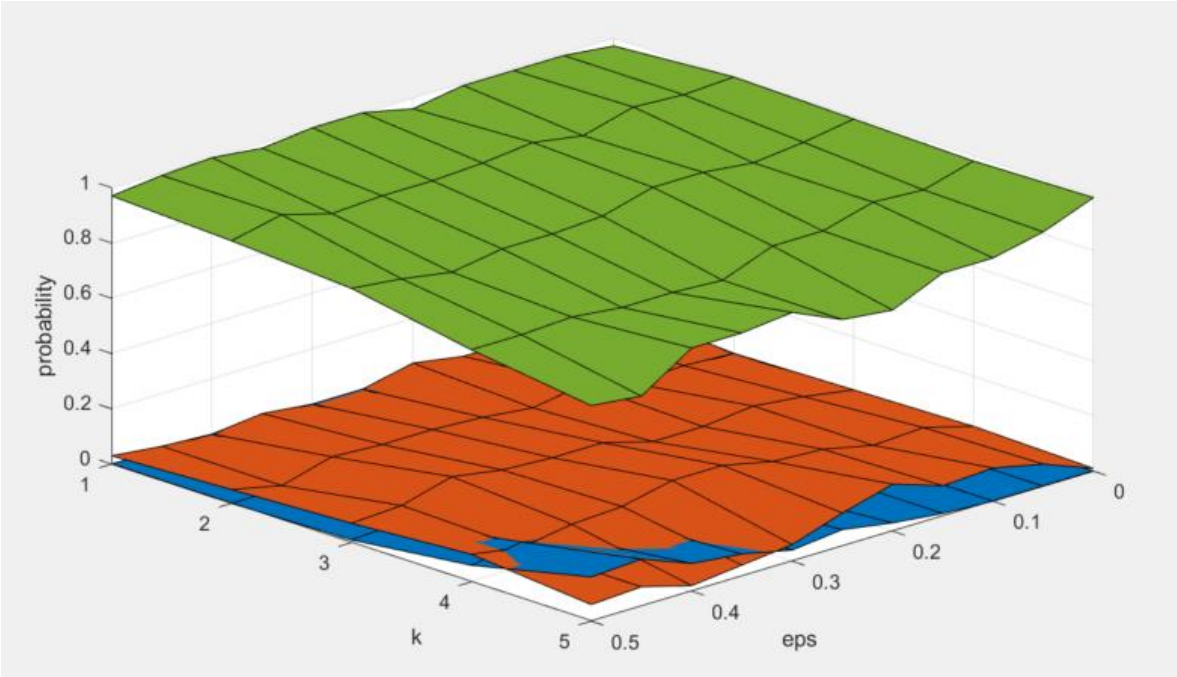

Рисунок 3.23 – Вероятности ошибок первого и третьего рода Δ=0.2 в случае, когда верным решением является принятие гипотезы H_2

Проанализировав Рисунки 3.21 – 3.23 можно сделать вывод, что матричный последовательный тест, как и тест, основанный на апостериорных вероятностях недостаточно робастен. Можно наблюдать, что при значительном росте величины ε сильно увеличивается количество выбросов, а следовательно результаты вычислительных экспериментов становятся плохо пригодными для последующего анализа.

Однако, несмотря на тот факт, что все ещё нельзя говорить об оптимальности и устойчивости данного теста в случае более двух простых гипотез, матричный последовательный тест проявляет себя лучше, чем тест использующий априорные вероятности. Сравнив соответствующие графики (Рисунок 3.11 с Рисунком 3.21 и Рисунок 3.16 с Рисунком 3.22) мы можем говорить о том,что в случае когда гипотезы находятся достаточно близко друг к другу (Δ=0.2) вероятности ошибок полученные с помощью матричного теста значительно ниже, даже при достаточно сильных выбросах. Например, в случае когда k=3 вероятность принятие ошибочного решения в худшем случае не превышает 0.4.

3.5 Влияние искажений на среднее число наблюдений

В данном пункте рассмотрим зависимости среднего числа наблюдений от ε и Δ при некоторых фиксированных k и от k и ε при некотором фиксированном Δ.

Сначала проведем эксперименты для последовательного теста, основанного на апостериорных вероятностях.

Здесь же поверхность синего цвета описывает изменение количества среднего числа наблюдений для первой гипотезы, оранжевого – для второй гипотезы, зеленого – для третьей гипотезы, красного – для четвертой гипотезы

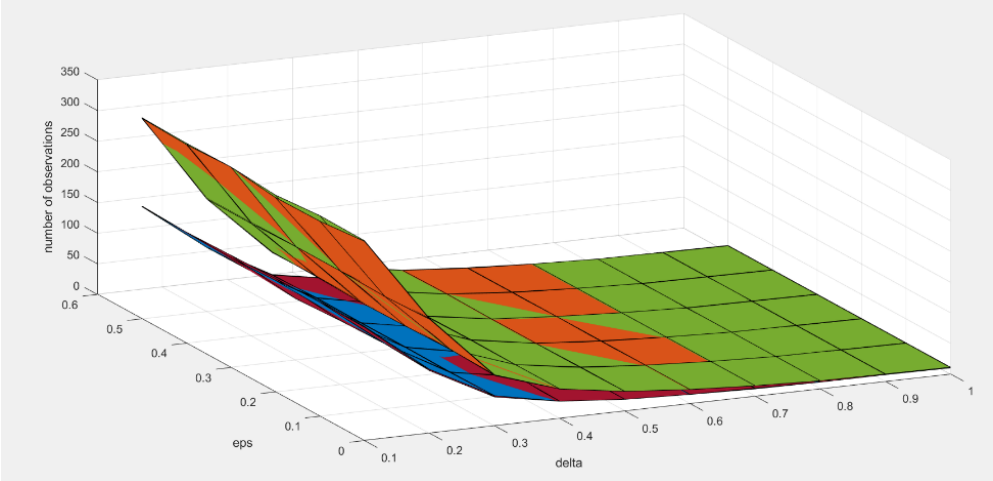

Рисунок 3.24 – Среднее число шагов (наблюдений) для первой, второй, третьей и четвертой гипотезы при $k = 1$

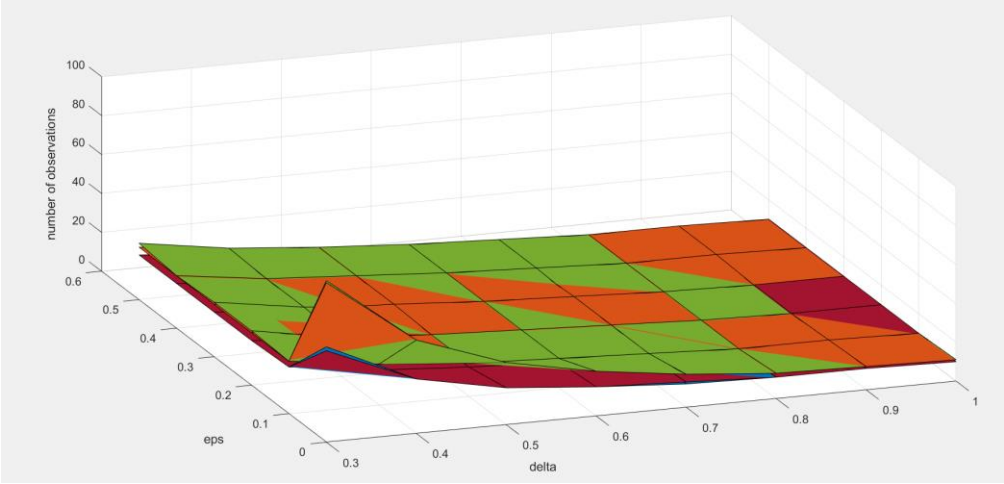

Рисунок 3.25 – Среднее число шагов (наблюдений) для первой, второй, третьей и четвертой гипотезы при $k = 3$

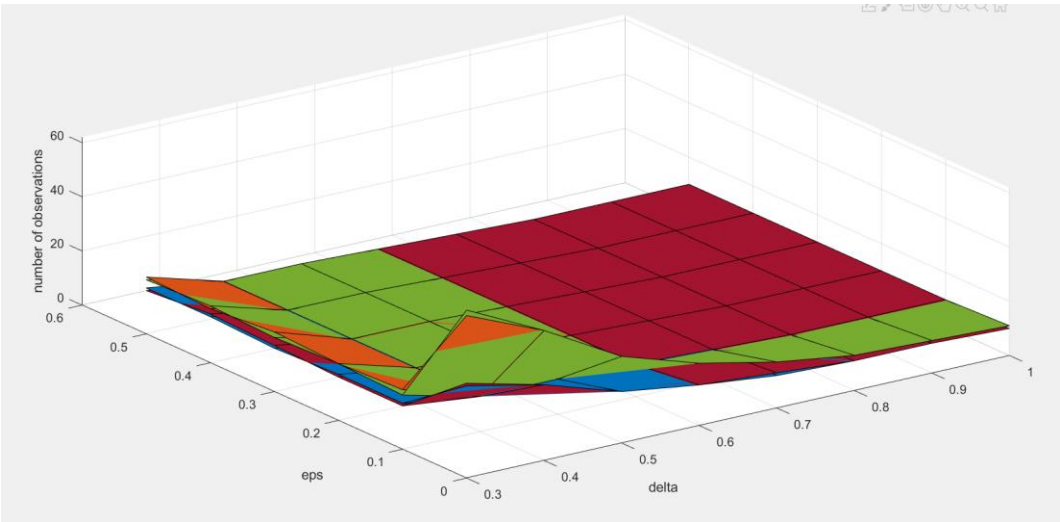

Рисунок 3.26 – Среднее число шагов (наблюдений) для первой, второй, третьей и четвертой гипотезы при $k = 5$

Можно заметить, что увеличение параметра k приводит к значительному уменьшению количества наблюдений. Тест становится более “быстрым”, но менее точным.

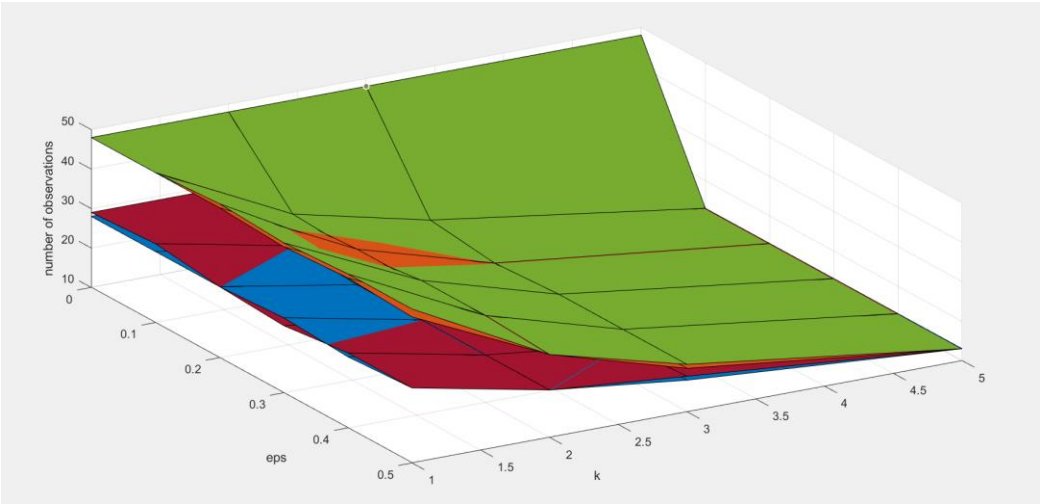

Рисунок 3.27 – Среднее число шагов (наблюдений) для первой, второй, третьей и четвертой гипотезы при $\Delta = 0.3$

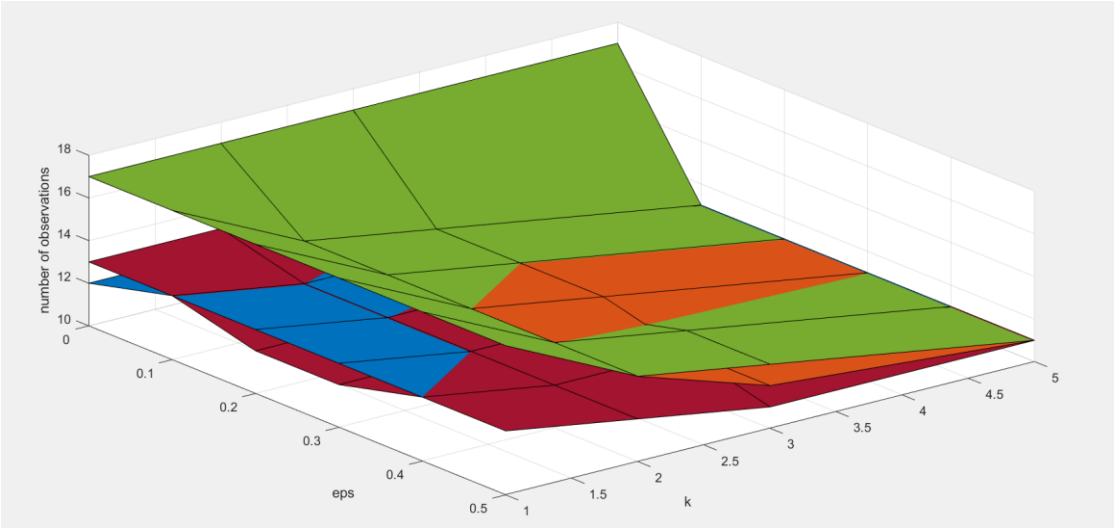

Рисунок 3.28 – Среднее число шагов (наблюдений) для первой, второй, третьей и четвертой гипотезы при $\Delta = 0.7$

Также можно отметить, что параметр ε незначительно влияет на среднее количество шагов.

Как отмечалось в главе 3.3, при уменьшении параметра Δ количество наблюдений значительно возрастает. Это обуславливается тем, что при удалении гипотез друг от друга частота принятия “неправильных” решений уменьшается и следовательно мы можем добиться заданной точности за меньшее количество шагов (наблюдений).

Теперь же проведем эксперименты для матричного последовательного теста, основанного на апостериорных вероятностях.

Пускай здесь поверхность синего цвета описывает изменение количества среднего числа наблюдений для первой гипотезы, оранжевого – для второй гипотезы, зеленого – для третьей гипотезы.

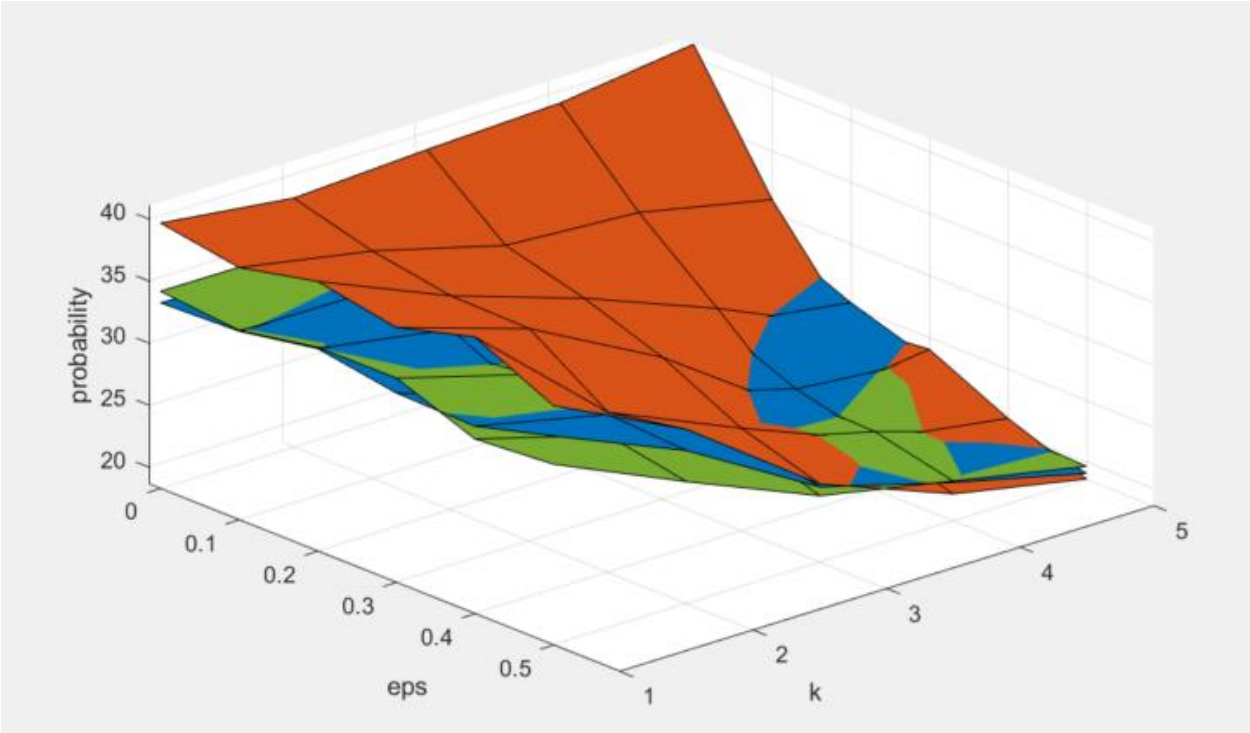

Рисунок 3.29 – Среднее число шагов (наблюдений) для первой, второй и третьей гипотезы при $\Delta = 0.3$

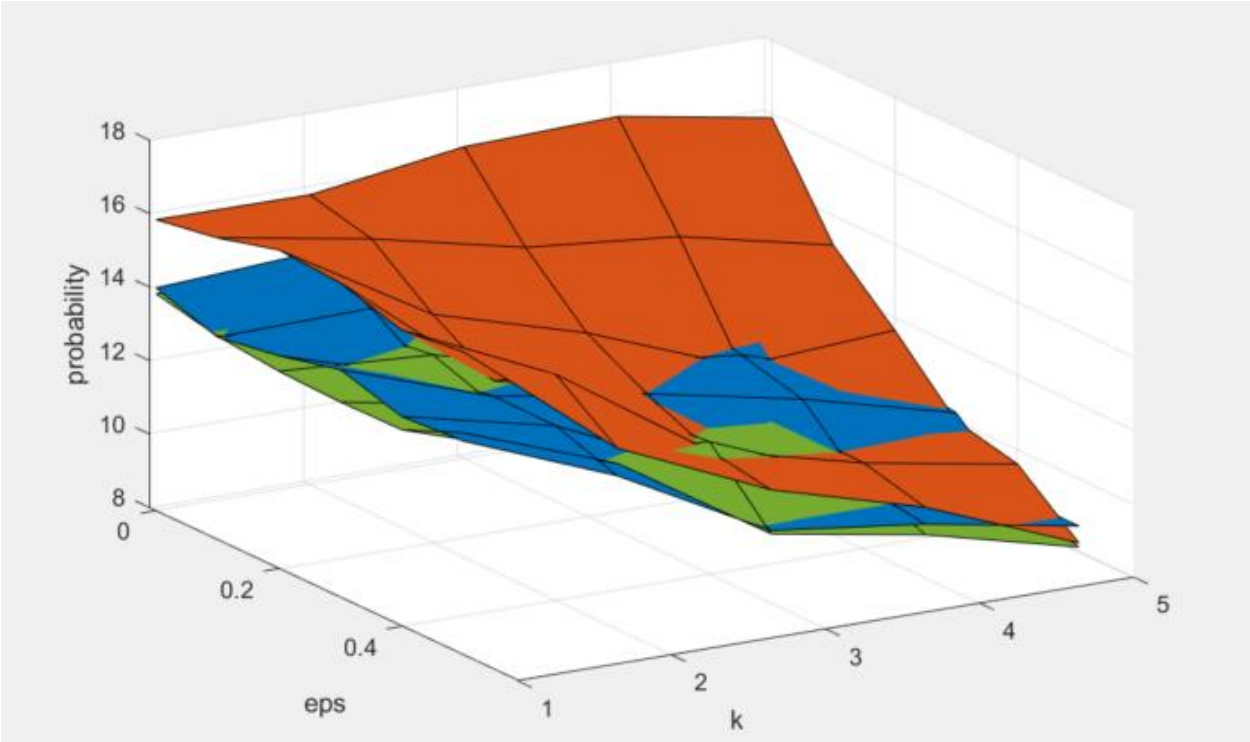

Рисунок 3.30 – Среднее число шагов (наблюдений) для первой, второй и третьей гипотезы при $\Delta = 0.7$

Исходя из результатов экспериментов можно судить о том, что для матричного теста, как и для теста, основанного на апостерирорных вероятностях увеличение параметра Δ ведет к уменьшению количества необходимых наблюдений. Однако матричный тест лучше справляется с выбросами. На рисунках 3.29 и 3.30 можно заметить, что поверхности имеют более пологую форму, то есть при росте k и ε количество необходимых наблюдений постепенно падает. Это напрямую связанно с тем, что матричный тест при выбросах имеет меньшую вероятность ошибки, чем тест, использующий апостерирорные вероятности.

ГЛАВА 4
ПОСЛЕДОВАТЕЛЬНОЕ ПРИНЯТИЕ РЕШЕНИЙ В АНАЛИЗЕ ЭПИДЕМИОЛОГИЧЕСКОГО ПРОЦЕССА ПО ЗАБОЛЕВАЕМОСТИ COVID-19

В данной главе проводится применение полученных результатов не на абстрактных тестах, а для анализа реальных статистических данных, описывающих эпидемиологические процессы заболеваемости коронавирусом COVID-19. Анализируется возможность применения матричного последовательного теста для выявления роста, падения или выхода на плато случаев заражения COVID-19.

4.1 Особенности эпидемиологических процессов COVID-19

Пандемия COVID-19 представляет собой беспрецедентный глобальный кризис здравоохранения за последние 100 лет. Его воздействие на экономику, общество и здоровье продолжает расти и, вероятно, станет одной из самых страшных глобальных катастроф со времен пандемии 1918 года и мировых войн. В нынешнем сценарии весь мир борется с COVID-19, который представляет собой высокопередающуюся и патогенную вирусную инфекцию. Вирус, вызывающий COVID-19, в основном передается через капли, образующиеся при кашле, чихании или выдохе инфицированного человека. Эти капли слишком тяжелые, чтобы висеть в воздухе, поэтому они быстро оседают на различных поверхностях. Более серьезные последствия вируса проявляются у людей, страдающих такими заболеваниями, как диабет, хронические респираторные заболевания, рак и сердечно-сосудистые заболевания, а также у пожилых людей. Из-за отсутствия надлежащей вакцины, медицинских учреждений и экспертов нынешняя ситуация становится очень опасной и неконтролируемой для многих стран.

Математики, разработчики программного обеспечения, эпидемиологи и ученые используют искусственный интеллект, машинное обучение, цифровые технологии и облачные вычисления для обнаружения инфекции, анализа скорости передачи и моделей распространения COVID-19, от которого миллионы людей во всем мире становятся опасно больными. Например,

компьютерное моделирование, основанное на статистике авиапассажиров и данных о количестве зараженных COVID-19 за пределами Китая, позволило быстро оценить масштабы вспышки в городе Ухань. Более позднее исследование, проведенное британским эпидемиологом профессором Нилом Фергюсоном из Имперского колледжа Лондона, предсказало, что в Великобритании может погибнуть более 500 000 человек, если правительство не предпримет никаких действий. Математические модели полезны для понимания поведения инфекции и изучения условий, при которых её распространение будет будет останавливаться или продолжаться. Следовательно, для того, чтобы преодолеть и уменьшить распространение инфекции, существует необходимость в подготовке математической модели для постоянного контроля ситуации и немедленного реагирования в критических случаях.

4.2 Применение последовательного подхода для принятия решений

Данные играют важную роль в информировании усилий по предотвращению и снижению рисков для здоровья населения. Каждый день публикуется множество исследований, которые помогают ответить на важные вопросы о COVID-19. Но, учитывая большое количество и различные типы опубликованных исследований, иногда может быть трудно оставаться в курсе последних исследований. Поэтому центр ресурсов по COVID-19 Джонса Хопкинса публикует ежедневные сводки в которых можно найти информацию о количестве заболевших, выздоровевших и умерших.

Данные представляют собой сводные таблицы временных рядов, которые описывают количество подтвержденных случаев, количество смертей и количество вылеченных. Все данные считываются из ежедневного отчета о болезни. Таблицы временных рядов могут обновляться, если в исторических данных выявляются неточности.

Попробуем использовать матричный последовательный тест для определения роста или снижения случаев заражения. Для эксперимента возьмем несколько разных 30-дневных промежутков, на которых мы можем наблюдать увеличение или уменьшение выявленных случаев заражения COVID-19 и посмотрим сколько наблюдений понадобится тесту для вынесения решения.

Возьмем данные за апрель 2020 года, когда присутствовал явный рост количества заболеваний и посмотрим на каком шаге тест выполнит остановку, т.е. вынесет решение в пользу гипотезы “роста”.

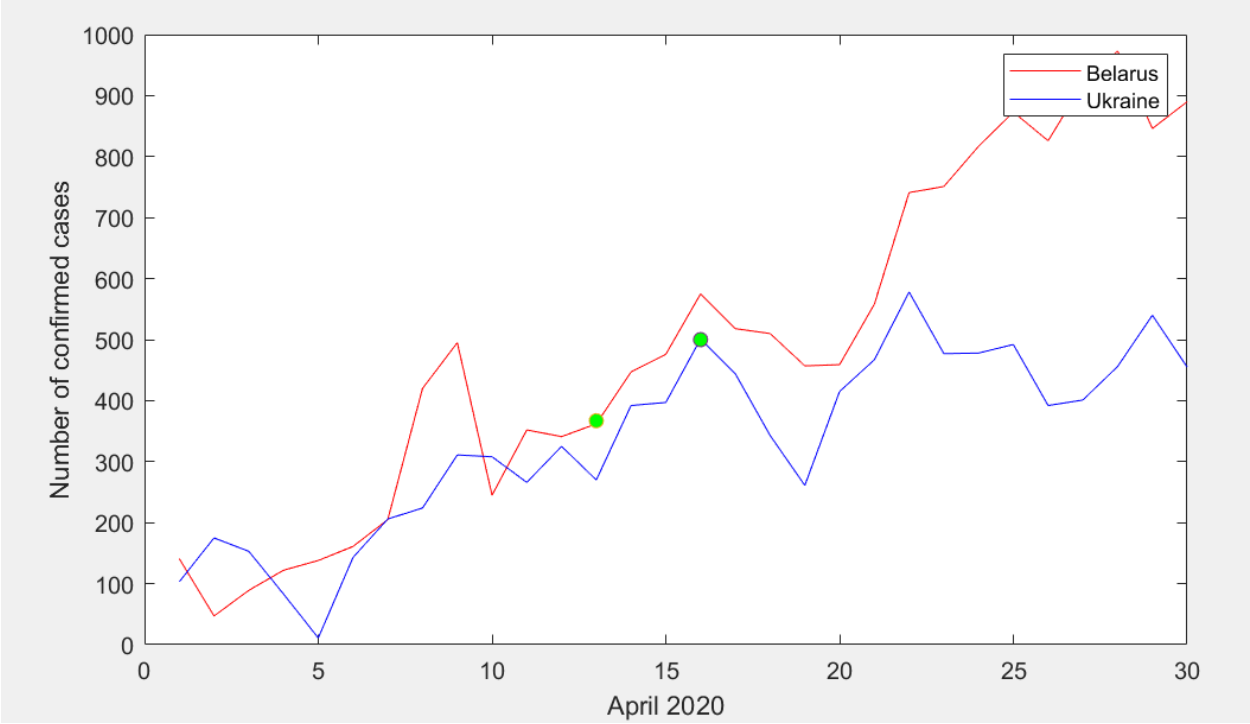

Рисунок 4.1 – Количество зафиксированных случаев COVID – 19 в Беларуси и Украине в апреле 2020 года

Исходя из рисунка 4.1 мы видим, что тест вынес решение в пользу “роста” на 13 наблюдении для Беларуси и на 16 наблюдении для Украины.

Теперь возьмем данные за июнь 2020 года, когда присутствовал рост количества заболеваний в Украине, однако в Беларуси начался резкий спад и посмотрим сколько наблюдений понадобится тесту в этом случае.

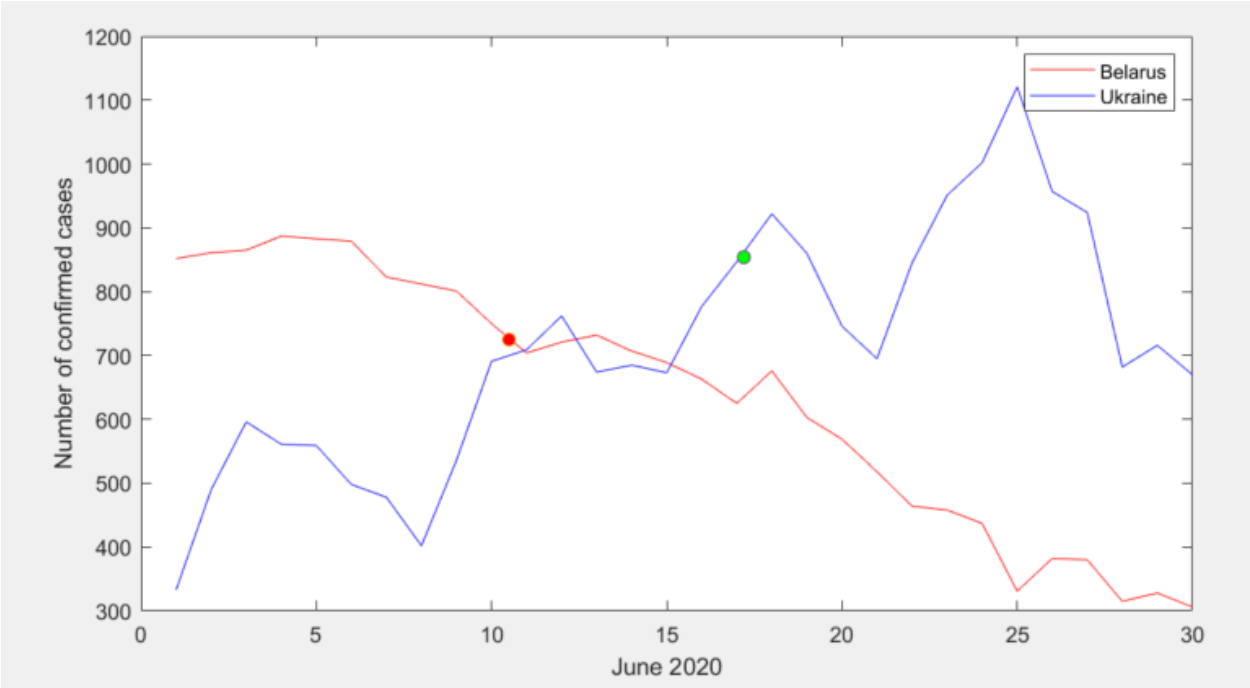

Рисунок 4.2 – Количество зафиксированных случаев COVID – 19 в Беларуси и Украине в июне 2020 года

В данном случае мы видим, что тест “решил” ,что происходит рост на 17 наблюдении для Украины. В случае Беларуси тест вынес решение в пользу гипотезы “спада” на 11 наблюдении.

Для полноты картины возьмем данные за май 2020 года, когда мы можем наблюдать плато для одной из стран (в данном случае для Беларуси).

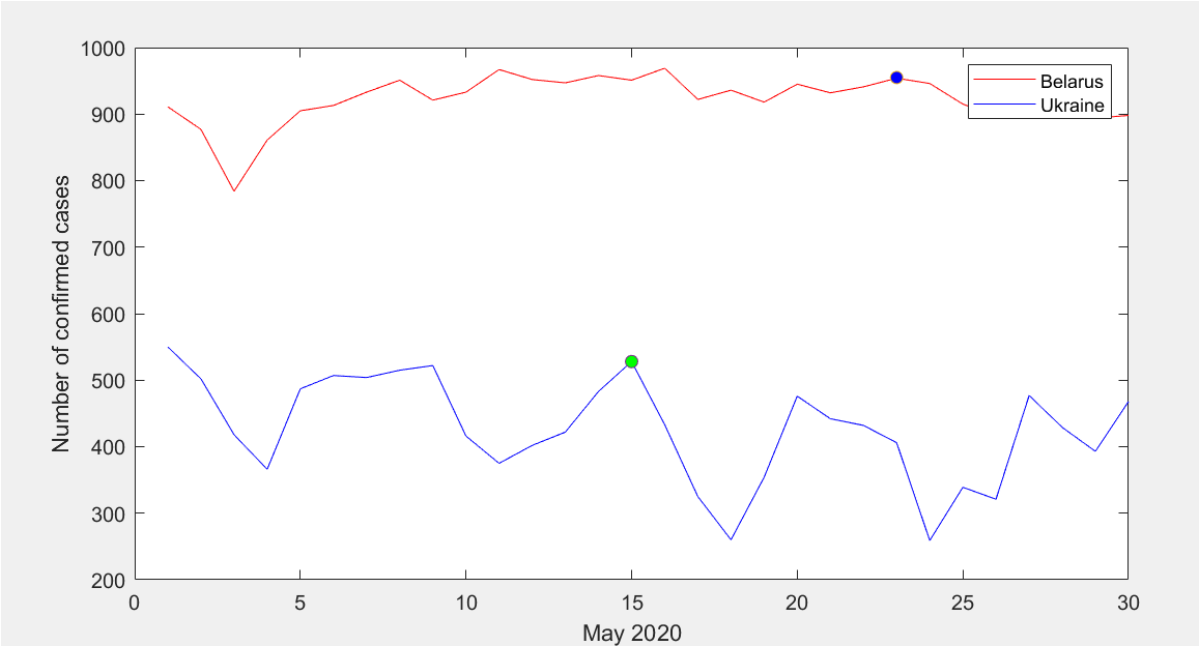

Рисунок 4.3 – Количество зафиксированных случаев COVID – 19 в Беларуси и Украине в мае 2020 года

В случае Беларуси мы видим, что решение было в пользу “выхода на плато” на 23 наблюдении. В случае Украины на 15 шаге был зафиксирован “рост”.

4.3 Краткие выводы

Как показали эксперименты на реальных данных, матричный последовательный тест может использоваться для определения характера поведения распространения короновирусной инфекции. В особенности, он может эффективно использоваться для раннего выявления с заданной точностью (малыми значениями вероятностями возможных ошибок) типовых классов траекторий эпидемиологических процессов: «рост», «спад», «плато».

ЗАКЛЮЧЕНИЕ

Теоретический анализ характеристик эффективности последовательных тестов проверки отношения гипотез является сложней задачей [6]. В магистерской диссертации для анализа использованы компьютерные эксперименты. Были получены следующие результаты:

1) Построен однопороговый последовательный тест проверки более двух простых гипотез, основанный на апостериорных вероятностях.
2) Построен матричный последовательный тест проверки более двух простых гипотез
3) Исследованы следующие виды зависимостей характеристик эффективности этого теста:

   а) вероятности ошибок от величины порога и априорных вероятностей

   б) среднее число наблюдений от величины порога и априорных вероятностей

   в) вероятности ошибок от величины ε и Δ

   г) среднее число наблюдений от ε и Δ

   д) вероятности ошибок от величины ε и k

   е) среднее число наблюдений от ε и k

4) Показано, что для обеспечения робастности (устойчивости к искажениям) последовательного теста проверки более двух простых гипотез, основанного на апостериорных вероятностях необходимо модифицировать решающую функцию, так как при значительных искажениях рассмотренных в работе тест теряет эффективность.
5) Исследована возможность использования последовательного теста для определения характера поведения распространения инфекции COVID-19.
6) Показано, что матричный последовательный тест может быть эффективно использован для задачи выявления на ранних стадиях с заданной точностью (малыми значениями вероятностями возможных ошибок) типовых классов траекторий эпидемиологических процессов: «рост», «спад» и «плато».

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *A. Wald and J. Wolfowitz* Optimal Character of the Sequential Probability Ratio Tests, Ann. Math. Statits., vol. 19, pp. 326–339, 1948.
2. *Jennison, C. and Turnbull, B. W.* Group sequential methods with applications to clinical trials. – New York: Chapman & Hall, 2000. -390 c.
3. *Kharin A.Y.* Performance and robustness evaluation in sequential hypotheses testing // Communications in Statistics – Theory and Methods. 2016. Vol. 45, No.6 P. 1663-1709
4. *Kharin A., Ton That Tu, Ivankovich I., Song Peidong* Sequential statistical testing of several simple hypotheses under distortion of the observations probability distribution. Материалы научной конференции «Актуальные проблемы стохастического анализа», посвященной 80 летию академика Ш.К.Форманова. 20-21 февраля 2021 г., Ташкент, с. 56 - 59
5. *Tu T.T., Kharin A.Y.* Sequential probability ratio test for many simple hypotheses on parameters of time series with trend, Journal of the Belarusian State University. Mathematics and Informatics, vol. 1, 2019, p. 35-45.
6. Вальд, А.Последовательный анализ / А. Вальд – М.: Физматгиз, 1960 – 328 с.
7. Ту, Т., & Харин, А. (2019). Последовательный критерий отношения вероятностей для проверки многих простых гипотез о параметрах временных рядов с трендом. *Журнал Белорусского государственного университета. Математика. Информатика*, (1), 35-45. https://doi.org/10.33581/2520-6508-2019-1-35-45
8. Харин, А.Ю. Оценивание вероятностей ошибок последовательного критерия отношения вероятностей /А.Ю. Харин, С.Ю. Чернов // Вестн. БГУ, 2011. №1.С.96-100
9. Харин А.Ю., Робастность байесовских и последовательных статистических решающих правил / Минск: БГУ,2013 207 с.

Приложение А

generator.java

```
import java.util.Random;

public class Generator {
    protected double[] hypotesis;
    protected double[] aprior;
    protected double[] apostrior;
    protected double mean,variance;
    protected double breakpoint;
    protected double[] pba;
    protected int currentstep;
    protected Random rg;
    protected int hypCount;
    protected int stepsBeforeBayes;
    protected double epsilon;
    protected int varianceCoefficient;
    Generator (double hyp[],double apr[],double brp,double inmean,double invar,int
inn,int hypCount){
        hypotesis=hyp;
        aprior=apr;
        stepsBeforeBayes=inn;
        breakpoint=brp;
        mean=inmean;
        variance=invar;
        currentstep=0;
        this.hypCount=hypCount;
        pba=new double[hypCount];
        for(int i=0;i<hypCount;i++){
            pba[i]=1;
        }
        apostrior=new double[hypCount];
        rg=new Random();
    }

    protected double getRandom(){
        return (rg.nextGaussian()*variance)+mean;
    }

    protected double densityFunction(double x,double innermean) {
        return (1 / variance * Math.sqrt(2 * Math.PI)) * Math.exp((-(x - innermean) *
(x - innermean)) / (2 * variance * variance));
    }
    protected void getCurrentpba(){
        double x=getRandom();
        for(int i=0;i<hypCount;i++) {
            pba[i] *= densityFunction(x, hypotesis[i]);
        }
    }
    protected void getCurrentApostreors(){
        double sum=0;
        for(int i=0;i<hypCount;i++){
            sum+=aprior[i]*pba[i];
        }
        for(int i=0;i<hypCount;i++){
            apostrior[i]=aprior[i]*pba[i]/sum;
        }
    }
    protected void makeStep(){
```

```
        currentstep++;
        getCurrentpba();
        getCurrentApostreors();
    }
    protected void makeNSteps(){
        for(int i=0;i<stepsBeforeBayes;i++){
            makeStep();
        }
    }
    protected int checkPs(){
        for(int i=0;i<hypCount;i++) {
            if(apostrior[i]>breakpoint)
                return (i+1);
        }
        return 0;
    }
    public int[] generateTest(){
        makeNSteps();
        int x=checkPs();
        while(x==0){
            makeStep();
            x=checkPs();
        }
        int[] returner=new int[2];
        returner[0]=--x;
        returner[1]=currentstep;
        return returner;
    }
}
```

generator_modified.java

```
public class GeneratorModified extends Generator {
    private double epsilon;
    private int varianceCoefficient;

    GeneratorModified(double hyp[],double apr[],double brp,double inmean,double in-
var,int inn,int hypCount, double epsilon, int varianceCoefficient){
        super(hyp,apr,brp,inmean,invar,inn,hypCount);
        this.epsilon=epsilon;
        this.varianceCoefficient=varianceCoefficient;
    }

    @Override
    public double getRandom(){
        if(epsilon>Math.random()){
            return (rg.nextGaussian()*variance*varianceCoefficient)+mean;
        }else{
            return getRandom();
        }
    }

}
```

tester.java

```
public class Tester {
    protected double[] hyp;
    protected double[] apr;
    protected int hypNumber;
```

```
    protected double variance;
    protected double[][]errorMatrix;
    protected int[] counter;
    protected int stepsBeforeBayes;
    protected double breakpoint;
    protected int timestotest;
    Tester(double[] inhyp,double[] inapr,double invariance,double brp, int stepsBe-
foreBayes,int intimestotest,int hypCount){
        timestotest=intimestotest;
        hyp=inhyp;
        apr=inapr;
        variance=invariance;
        breakpoint=brp;
        this.stepsBeforeBayes=stepsBeforeBayes;
        hypNumber=hypCount;
        errorMatrix=new double[hypCount][hypCount];
        counter=new int[hypNumber];
    }
    void test(){
        int[][] temp=new int[hypNumber][2];
        for(int i=0;i<timestotest;i++){
            Generator[] gen=new Generator[hypNumber];
            for(int j=0;j<hypNumber;j++) {
                gen[j]=new Generator(hyp,apr,breakpoint,hyp[j],variance,stepsBefore-
Bayes,hypNumber);
                temp[j]=gen[j].generateTest();
            }
            for(int j=0;j<hypNumber;j++){
                errorMatrix[j][temp[j][0]]++;
                counter[j]+=temp[j][1];
            }
        }
        for(int i=0;i<hypNumber;i++){
            counter[i]/=timestotest;
            for(int j=0;j<hypNumber;j++){
                errorMatrix[i][j]/=timestotest;
            }
        }
    }
    void printArray(double[] array,int indexNotToShow){
        for(int i=0;i<hypNumber;i++){
            System.out.println(array[i]);
        }
    }
    void showresults(){
        for(int i=0;i<hypNumber;i++){
            printArray(errorMatrix[i],i);
            System.out.println(counter[i]);
        }
    }
}
```

tester_modified.java

```
package com.company;

public class TesterModified extends Tester{
    protected double epsilon;
    protected int varianceCoefficient;
    TesterModified(double[] inhyp,double[] inapr,double invariance,double brp, int
```

```
stepsBeforeBayes,int intimestotest,int hypCount,double epsilon,int varianceCoeffi-
cient){
        super(inhyp,inapr,invariance,brp,stepsBeforeBayes,intimestotest,hypCount);
        this.epsilon=epsilon;
        this.varianceCoefficient=varianceCoefficient;
    }
    @Override
    void test(){
        int[][] temp=new int[hypNumber][2];
        for(int i=0;i<timestotest;i++){
            Generator[] gen=new Generator[hypNumber];
            for(int j=0;j<hypNumber;j++) {
                gen[j]=new GeneratorModified(hyp,apr,breakpoint,hyp[j],vari-
ance,stepsBeforeBayes,hypNumber,epsilon,varianceCoefficient);
                temp[j]=gen[j].generateTest();
            }
            for(int j=0;j<hypNumber;j++){
                errorMatrix[j][temp[j][0]]++;
                counter[j]+=temp[j][1];
            }
        }
        for(int i=0;i<hypNumber;i++){
            counter[i]/=timestotest;
            for(int j=0;j<hypNumber;j++){
                errorMatrix[i][j]/=timestotest;
            }
        }
    }
    @Override
    void printArray(double[] array,int indexNotToShow){
        for(int i=0;i<hypNumber;i++){
            System.out.println(array[i]);
        }
    }
    @Override
    void showresults(){
        for(int i=0;i<hypNumber;i++){;
            printArray(errorMatrix[i],i);
            System.out.println(counter[i]);;
        }
    }

}
```

TesterMatrix.java

```
package com.company;

import javafx.util.Pair;

public class TesterMatrix {

    private int timesToTest;
    private double[] alpha;

    TesterMatrix(int timesToTest, double[] alpha){
        this.timesToTest = timesToTest;
        this.alpha = alpha.clone();
    }
```

```
    public static double[][] countMatrixB(int size, double[] alpha){
        double[][] matrixB = new double[size][size];
        for(int i=0; i<size; i++){
            for(int j=0; j<size; j++){
                matrixB[i][j]= Math.log(2/(alpha[i]));
            }
        }
        return matrixB;
    }

    public double[] createCounterArray(int size){
        double[] counterArray = new double[size];
        for(int i=0; i<size; i++){
            counterArray[i]=0;
        }
        return counterArray;
    }

    public void test(double[][] hypothesisList, double variance, double epsilon, dou-
ble k){
        int size = hypothesisList[0].length;
        GeneratorMatrix generatorMatrix = new GeneratorMatrix(size, hypothesisList,
countMatrixB(hypothesisList.length, alpha), variance);
        for(int i=0; i<hypothesisList.length; i++){
            Pair<Integer, Integer> result;
            double[] counterArray = createCounterArray(hypothesisList.length);
            int stepsForEveryTest = 0;
            for(int j=0; j<timesToTest; j++){
                result = generatorMatrix.doTest(hypothesisList[i], epsilon, k);
                counterArray[result.getKey()]++;
                stepsForEveryTest += result.getValue();
            }
            printResults(counterArray, stepsForEveryTest);
        }
    }

    private void printResults(double[] counterArray, int stepsForEveryTest){
        for(int i=0; i<counterArray.length; i++){
            System.out.println("Hypothesis " + i + " result: " + (counterAr-
ray[i]/timesToTest));
        }
        System.out.println("Average number of iterations: " + (double)stepsForEvery-
Test/timesToTest);
        System.out.println();
    }

}
```

GeneratorMatrix.java

```
package com.company;

import javafx.util.Pair;

import java.io.BufferedReader;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Random;
```

```
public class GeneratorMatrix {

    protected int size;
    protected int numberOfHypothesis;
    protected Random randomGenerator;
    protected ArrayList<Double> observations;
    protected double[][] hypothesisList;
    protected double[][] matrixB;
    protected double variance;

    GeneratorMatrix (int size, double[][] hypothesisList, double[][] matrixB, double
variance){
        randomGenerator = new Random();
        this.hypothesisList = hypothesisList.clone();
        this.size = size;
        this.numberOfHypothesis = hypothesisList.length;
        this.matrixB = matrixB.clone();
        observations = new ArrayList<>();
        this.variance = variance;
    }

    private double generateGaussian(){
        return (randomGenerator.nextGaussian()*variance);
    }

    private double generateGaussian(double epsilon, double k){
        if(epsilon>Math.random()){
            return generateGaussian()*k;
        } else {
            return generateGaussian();
        }
    }

    private double generateXt(double[] trueValue, double [] trend, double epsilon,
double k){
        double multiplication = multiplyVectors(trueValue, trend);
        double gaussian = generateGaussian(epsilon, k);
        return multiplication + gaussian;
    }


    private double[] countTrend(int t){
        double[] trend = new double[size];
        for(int i = 0; i<size; i++){
            trend[i] = t;
        }
        return trend;
    }

    private double multiplyVectors(double[] vector1, double[] vector2){
        double sum=0;
        for(int i= 0; i<vector1.length; i++){
            sum+=vector1[i]*vector2[i];
        }
        return sum;
    }

    private double n1(double xt, double[] hypothesis, double[] trend){
        double mean = multiplyVectors(hypothesis, trend);
        return (1 / variance * Math.sqrt(2 * Math.PI)) * Math.exp((-(xt - mean) * (xt
- mean)) / (2 * variance * variance));
```

```
}

private double countMatrixCellMultiplication(int i, int j, int n){
    double upper;
    double lower;
    double multiplication=1;
    for(int t=1; t<=n; t++){
        upper = n1(observations.get(t-1), hypothesisList[i], countTrend(t));
        lower = n1(observations.get(t-1), hypothesisList[j], countTrend(t));
        multiplication*=upper/lower;
    }
    return multiplication;
}

private double countMatrixCell(int i, int j, int n){
    return Math.log(countMatrixCellMultiplication(i, j, n));
}

protected double[][] countMatrix(int n){
    double[][] matrix = new double[numberOfHypothesis][numberOfHypothesis];
    for(int i = 0; i< numberOfHypothesis; i++){
        for(int j = 0; j< numberOfHypothesis; j++){
            matrix[i][j]=countMatrixCell(i, j, n);
        }
    }
    return matrix;
}

protected boolean isExceedingMatrixB(double[][] matrix){
    for(int i = 0; i< numberOfHypothesis; i++){
        for(int j = 0; j< numberOfHypothesis; j++){
            if(matrix[i][j] > matrixB[i][j]){
                if(i!=j){
                    return true;
                }
            }
        }
    }
    return false;
}

protected double[][] countMatrix(double[] trueValue, double epsilon, double k){
    int t=1;
    double [][] currentMatrix;
    while(true){
        observations.add(generateXt(trueValue, countTrend(t), epsilon, k));
        currentMatrix = countMatrix(t);
        if(isExceedingMatrixB(currentMatrix)){
            break;
        }
        t++;
    }
    return currentMatrix;
}

private double[] getTiList(double[][] matrix){
    double[] tiList = new double[numberOfHypothesis];
    for(int i = 0; i< numberOfHypothesis; i++){
        double ti = 9999999;
        for(int j = 0; j< numberOfHypothesis; j++){
            if(matrix[i][j] > matrixB[i][j]){
```

```
                    if(i!=j){
                        if(matrix[i][j]<ti){
                            ti=matrix[i][j];
                        }
                    }
                }
            }
            tiList[i]=ti;
        }
        return tiList;
    }

    private int chooseHypothesis(double[] tiList){
        double nb = 9999999;
        for(int i = 0; i < numberOfHypothesis; i++){
            if(tiList[i]<nb){
                nb=tiList[i];
            }
        }
        for(int i = 0; i< numberOfHypothesis; i++){
            if(tiList[i]==nb){
                return i;
            }
        }
        return -1;
    }

    public Pair<Integer, Integer> doTest(double[] trueValue, double epsilon, double
k){
        observations = new ArrayList<>();
        int resultHypotesis = chooseHypothesis(getTiList(countMatrix(trueValue, epsi-
lon, k)));
        int resultSteps = observations.size();
        Pair<Integer, Integer> resultPair = new Pair<>(resultHypotesis, resultSteps);
        return resultPair;
    }

}
```

GeneratorMatrix.java

```
package com.company;

import java.io.BufferedReader;
import java.io.FileReader;

public class GeneratorMatrixCovid extends GeneratorMatrix{
    GeneratorMatrixCovid(int size, double[][] hypothesisList, double[][] matrixB,
double variance) {
        super(size, hypothesisList, matrixB, variance);
    }

    private String[] readDataFromFile(String path) throws  Exception{
        String row;
        String[] data = new String[0];
        BufferedReader csvReader = new BufferedReader(new FileReader(path));
        while ((row = csvReader.readLine()) != null) {
            data = row.split(",");
        }
        return data;
    }
```

```
    private int[] parseValues(int startFromDay, int length){
        int[] resultArray = new int[length+1];
        try{
            resultArray = new int[length+1];
            String[] dataFromFile = readDataFromFile("C:\\Users\\Ilya_Ivanko-
vich\\Desktop\\belarusdata.csv");
            for(int i = 0; i< length; i++){
                resultArray[i] = Integer.parseInt(dataFromFile[i+startFromDay]);
            }
        } catch (Exception e){
            e.printStackTrace();
        }
        return resultArray;
    }

    @Override
    protected double[][] countMatrix(double[] trueValue, double epsilon, double k){
        int[] realValues = parseValues(60, 100);
        int t=1;
        double [][] currentMatrix;
        while(true){
            observations.add((double)realValues[t]); //здесь вместо генерации
использовать реальные данные
            currentMatrix = countMatrix(t);
            if(isExceedingMatrixB(currentMatrix)){
                break;
            }
            t++;
        }
        return currentMatrix;
    }
}
```

main.java

```
package com.company;


public class Main {

    public static void main(String[] args) {
       double[] hyp=new double[5];
       double[] apr=new double[5];
       double epsilon=0.5;
       int k=5;
       hyp[0]=0;
       hyp[1]=0.6;
       hyp[2]=1.2;
       hyp[3]=1.8;
       apr[0]=0.25;
       apr[1]=0.25;
       apr[2]=0.25;
       apr[3]=0.25;
       double breakingpoint=0.9;
       int stepsBeforeBayes=10;
       int hypCount=4;
       TesterModified testerModified=new TesterModified(hyp,apr,1,breaking-
point,stepsBeforeBayes,1000,hypCount,epsilon,k);
```

```
        testerModified.test();
        testerModified.showresults();

          double epsilon = 0;
          double k = 2;
          double[] alpha = {0.001, 0.001, 0.001};
          double[][] hypothesisList = {{0}, {0.5}, {1}};
          double variance = 100;
          TesterMatrix tester = new TesterMatrix(100, alpha);
          tester.test(hypothesisList, variance, epsilon, k);

          int size = hypothesisList[0].length;
          GeneratorMatrixCovid generatorMatrixCovid = new GeneratorMatrixCovid(size,
          hypothesisList, TesterMatrix.countMatrixB(hypothesisList.length, alpha),
          variance);
          Pair<Integer, Integer> resultPair = generatorMatrixCovid.doTest(null, 0, 0);
          System.out.println("Chosen hypotesis: " + resultPair.getKey());
          System.out.println("Steps performed: " + resultPair.getValue());

    }
}
```

3D surfaces

```
X=zeros(6,10);
Y=zeros(6,10);
fori=1:10
  X(:,i) = 0.1*i;
end

fori=1:6
Y(i,:) = 0.1*i-0.1;
end

P1= [0.98    1     0.99   1     0.99   0.99   1     1     0.98   0.99   0.99
0.97   0.94   0.96   0.95   0.98   0.94   0.92   0.95   0.96   0.9    0.99
0.99   0.99   0.97   0.96   0.97   0.88   0.94   0.92   0.93   0.82   0.88
0.98   0.97   0.9    0.85   0.88   0.89   0.86   0.9    0.76   0.84   0.86
0.97   0.94   0.93   0.91   0.94   0.82   0.81   0.83   0.81   0.85   0.79];

P2 =[ 0.02   0     0.01   0     0.01   0.01   0     0     0.02   0.01   0.01
0      0.03   0.06   0.04   0.05   0.02   0.06   0.08   0.05   0.04   0.1
0.01   0.01   0.03   0.04   0.03   0.12   0.06   0.07   0.07   0.15   0.11
0.02   0.03   0.09   0.12   0.09   0.09   0.12   0.09   0.14   0.08   0.08
0.03   0.06   0.07   0.08   0.06   0.15   0.13   0.07   0.07   0.04   0.07];

P3 = [0      0     0     0     0     0     0     0     0     0     0
0      0     0     0     0     0     0     0     0     0
0      0     0     0     0     0     0     0.01  0     0.03  0.01
0      0     0.01  0.03  0.03  0.02  0.02  0.01  0.1   0.08  0.06
0      0     0     0.01  0     0.03  0.06  0.1   0.12  0.11  0.14];

surf(X,Y,P1,'FaceColor',[0 0.4470 0.7410]);
hold on
surf(X,Y,P2,'FaceColor',[0.8500 0.3250 0.0980]);
hold on
surf(X,Y,P3,'FaceColor',[0.4660 0.6740 0.1880]  );
hold off
xlabel('k')
ylabel('eps')
zlabel('probability')
```