**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Нижегородский государственный технический университет**

**им. Р.Е. Алексеева» (НГТУ)**

**Кафедра: «Цифровая экономика»**

**Дисциплина: «Численные методы»**

**Курсовая работа**

**Тема:** «**Численное моделирование конкурентной борьбы агрегаторов такси на основе модели Лотки - Вольтерры с использованием реальных данных**»

Выполнил:

студент 3-го курса группы 21-САИ

Кожеватов Алексей Дмитриевич

Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

д.ф.м.н., проф. Катаева Лилия Юрьевна

27.12.2023  
Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород, 2023

# Содержание

[Содержание 2](#_Toc154514468)

[Введение 3](#_Toc154514469)

[Формулировка задачи. 3](#_Toc154514470)

[Исходные данные 4](#_Toc154514471)

[Методы получения параметров модели, алгоритмы 4](#_Toc154514472)

[Интегральный метод 6](#_Toc154514473)

[Логарифмический интегральный метод 8](#_Toc154514474)

[Описание методов и инструментов, используемых для решения системы. 9](#_Toc154514476)

[Таблица идентификаторов переменных 11](#_Toc154514477)

[Реализация задачи в Mathcad 15 13](#_Toc154514478)

[Написание программы для реализации метода 13](#_Toc154514479)

[Ручной счёт 13](#_Toc154514480)

[Нахождение коэффициентов 13](#_Toc154514481)

[Реализация в С++ 15](#_Toc154514482)

[Реализация в Golang 16](#_Toc154514483)

[Реализация задачи в Microsoft Office Excel 2013 17](#_Toc154514484)

[Результаты и их анализ 21](#_Toc154514485)

[Динамическая система. 24](#_Toc154514486)

[Статическая система. 28](#_Toc154514487)

[Анализ результатов работы программы, реализующей метод Рунге-Кутта. 29](#_Toc154514489)

[Вывод 32](#_Toc154514490)

[Список используемой литературы 33](#_Toc154514491)

[Приложения 34](#_Toc154514492)

[Приложение 1. Описание и код программ на языке C++, для нахождения параметров модели. 34](#_Toc154514493)

[Приложение 2. Описание и код программ на языке Go, для нахождения параметров модели интегральным и лог. интегральным методами. 38](#_Toc154514494)

[Приложение 3. Описание и код программ на языке Go, для решения системы методом Рунге-Кутта. 44](#_Toc154514496)

[Приложение 4. Участие в конференции «VIII Молодежная конференция по управлению проектами. 50](#_Toc154514498)

# Введение

В данной работе рассматривается модель Лотки-Вольтерра [1], описывающая изменение долей рынка взаимодействующих предприятий. Эта модель применяется для анализа и прогнозирования изменений долей с учетом различных типов взаимодействий таких как конкуренция за ресурсы, хищничество или симбиоз. Для обеспечения экономической безопасности страны [2] и обеспечения конкурентной борьбы предприятий необходимо сформировать рынок услуг. Прогнозирование и анализ на основе модели Лотки-Вольтерра позволяет определить динамику изменения каждой доли и влияние различных факторов на нее, что может быть полезно при планировании коммерческих мероприятий, изучении рынка или разработке стратегий управления экономическими ресурсами.

На основе прогнозов в области экономики можно принимать обоснованные решения о вложениях, инвестициях, распределении ресурсов и других аспектах. Это способствует снижению рисков и увеличению эффективности деятельности экономических участников.

Основная цель данного исследования заключается в анализе взаимодействия и динамики долей предприятий, занимающихся производством молочной продукции в Нижегородской области. Отдельное внимание уделяется прогнозированию на основе разрабатываемой модели.

В ходе исследования будут разработаны несколько вариантов модели Лотки-Вольтерры с использованием различных методов и программ. После этого будут проведены прогнозы, проведен анализ каждой модели и оценка их погрешностей.

Для построения модели Лотки-Вольтерры необходимо обладать данными о компаниях. Важно определить параметры, по которым будет создаваться модель. Выручка является подходящим показателем, так как она отражает эффективность работы предприятия, его конкурентоспособность и позицию на рынке. Чем выше выручка, тем успешнее предприятие.

# Формулировка задачи.

Параметры модели Лотки-Вольтерры, описывающей конкуренцию трех предприятий на рынке, отражают различные характеристики этой конкуренции. Например, параметры, описывающие коэффициенты рождаемости и смертности для каждой популяции (то есть коэффициенты, определяющие скорость изменения численности предприятий), могут отражать такие факторы, как эффективность производства, затраты на производство, качество продукции и другие факторы, влияющие на конкурентоспособность предприятий.

Модель представляет собой систему дифференциальных уравнений [2], где −доли лидирующих компаний, − это параметры, которые определяют влияние внешних факторов, − параметр, описывающий влияние доли рынка на долю этой же компании, − параметр, описывающий влияние доли рынка на долю , − параметр, описывающий влияние доли рынка на долю и т.д.

# Исходные данные

В первую очередь был найден список предприятий, осуществляющих производство интересующих услуг. Список был получен со специализированного сайта [5-6]. После этого для каждого предприятия были собраны данные о выручке за период 2015-2022 гг.

Модель составлена для трёх лидирующих компаний-агрегаторов: «ЯНДЕКС.ТАКСИ», «СИТИ-МОБИЛ» и «Gett». Долей рынка для предприятий послужила приближенная мера − доля выручки, полученная путем вычисления доли выручки компании от суммарной выручки за год (рис 1).

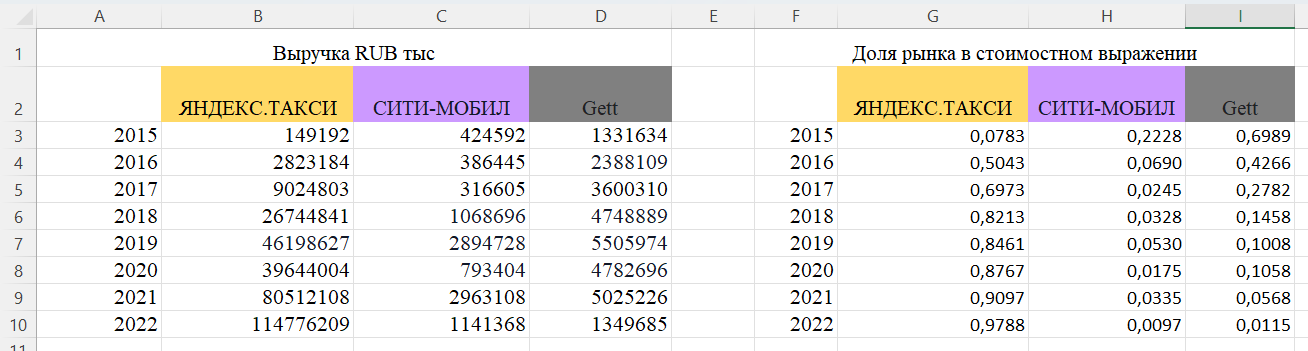


Рисунок 1 – выручка всех предприятий и доли участников, по данным которых будет составлена модель

# Методы расчёта коэффициентов взаимодействия

Для нахождения коэффициентов взаимодействия a, b, c системы дифференциальных уравнений (1) применяются интегральный и логарифмический методы поиска коэффициентов.

Интегральный метод используется для нахождения коэффициентов модели путём интегрирования системы дифференциальных уравнений, которые описывают взаимодействие между компаниями.

Далее для нахождения коэффициентов взаимодействия создаётся система:

или в виде уравнения , где , , матрица Х считается следующим образом:

Отсюда выражаются коэффициенты взаимодействия такие, что (3)

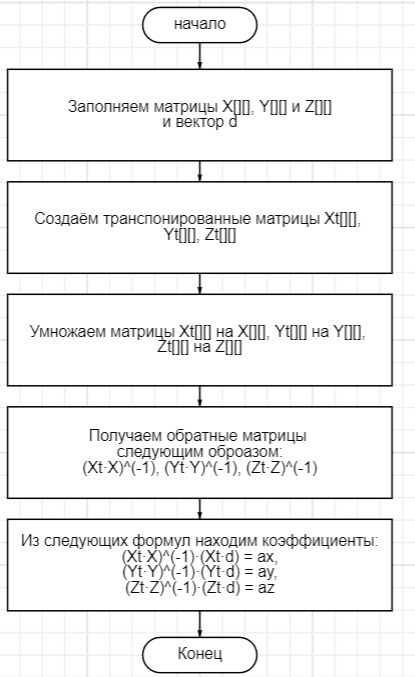


Рисунок 2– Блок-схема для интегрального метода нахождения коэффициентов взаимодействия

Логарифмический метод основан на использовании логарифмических преобразований и интегрирования уравнений модели.

Обе части уравнения делим на x и получаем: и проводим интегрирование на интервале [t0, tn].

Обе части уравнения делим на y и получаем: и проводим интегрирование на интервале [t0, tn].

Обе части уравнения делим на z и получаем: и проводим интегрирование на интервале [t0, tn].

Далее для нахождения коэффициентов взаимодействия создаётся система:

или в виде уравнения , где , , матрица Х считается следующим образом:

Отсюда выражаются коэффициенты взаимодействия такие, что . (5)



Рисунок 3– Блок-схема для логарифмического метода нахождения коэффициентов взаимодействия

# Описание методов и инструментов, используемых для решения системы.

Рисунок 4 – блок-схема метода Рунге-Кутта 4 порядка

Метод Рунге-Кутта [9].

Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

Вычисление нового значения проходит в четыре стадии:

Где h – величина шага сетки по x.

Используя формулы (11-15) метод Рунге-Кутта для системы дифференциальных уравнений примет вид:

# Таблица идентификаторов переменных

В таблице 1 представлены идентификаторы переменных, используемых для нахождения параметров модели

Таблица 1

Таблица идентификаторов переменных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Математические  обозначения | Mathcad 15 | C++ | Golang | Комментарий |
|  | findCIM | findCIM | findCIM | Функция для нахождения параметров модели интегральным методом |
|  | findLIM | findLIM | findLIM | Функция для нахождения параметров модели лог. интегральным методом |
| d | dx | dx | dx | Столбец свободных членов |
|  | toReturn | toReturn | toReturn | Вспомогательная переменная |
|  | temp | temp | temp | Вспомогательная переменная |
| X | X | X | X | Матрица (двумерный массив). Ее необходимо составить в ходе методов. |
| a | axIntegral | ax | ax | Массив содержащий параметры модели для 1-й компании. |
| a | ayIntegral | ay | ay | Массив содержащий параметры модели для 2-й компании. |
| a | azIntegral | az | az | Массив, содержащий параметры модели для 3-й компании. |
| x, y, z | x, y, z | x, y, z | x, y, z | Исходные доли рынка |
| i | i | i | i | Номер итерации |
| n | n | n | n | Количество данных, по которым составляется модель. Количество итераций |

# Реализация задачи в Mathcad 15

### Написание программы для реализации метода

# Ручной счёт

## Нахождение коэффициентов

y = z =

Находим матрицу X по формулам интегрального метода (1-5).

По аналогии заполняются все строки матрицы X:

….

Представляем матрицу Х в транспонированную матрицу Xt:

По формуле (3) находим коэффициенты.

По тому же принципу находим коэффициенты b матрицы Y и c матрицы Z:

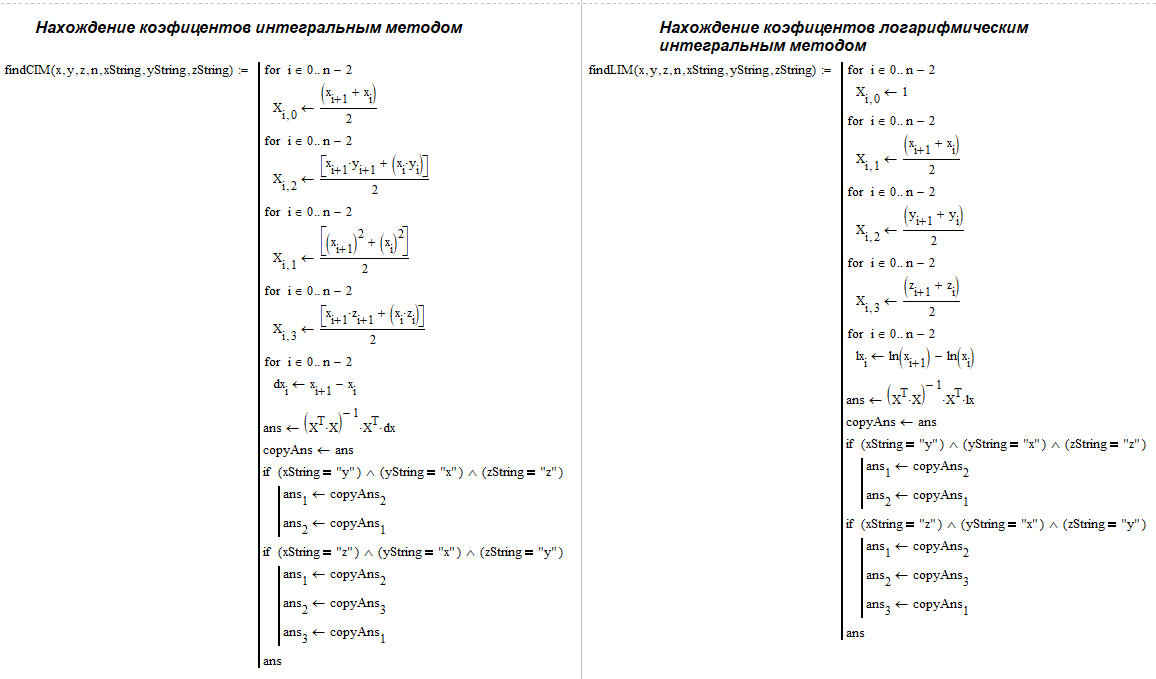
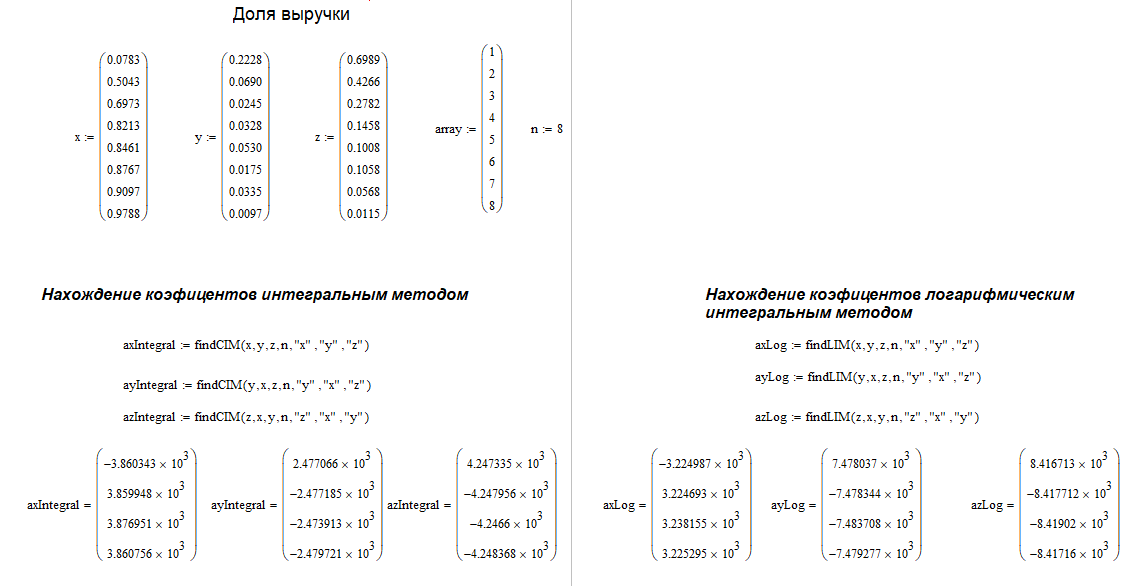


Рисунок 5 – Программы для нахождения параметров модели интегральным и лог. Интегральным методами.

Рисунок 6 – Результат программ для нахождения параметров модели интегральным и лог. Интегральным методами*.*

# Реализация в С++

Результат работы программы на языке программирования C++, отвечающий за нахождение параметров модели Лотки-Вольтерры с помощью интегрального и лог. интегрального методов. Полный код программы задания в Приложении 1.

Среда разработки: Visual Studio 2022

Результат работы программы представлен на рисунке 7

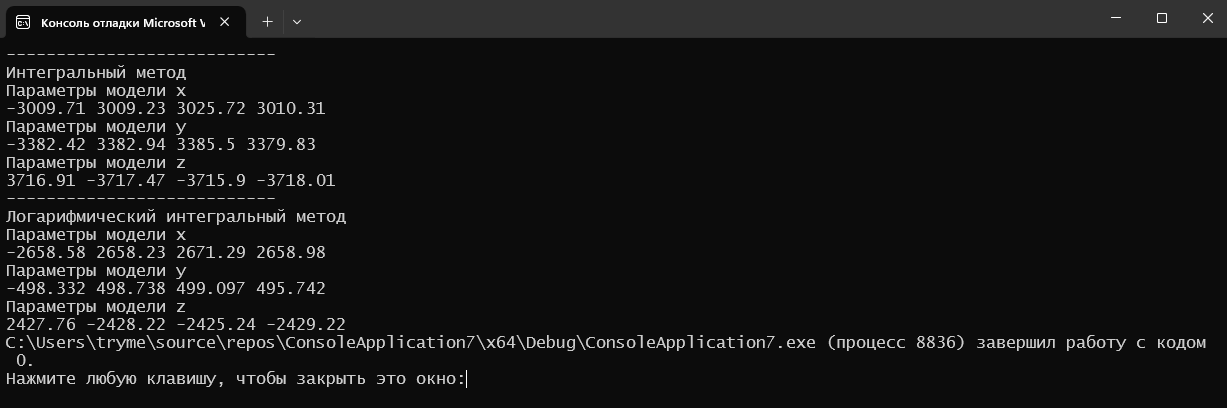


Рисунок 7 - Результат работы программы на С++, реализующей интегральный и лог. интегральный методы.

# Реализация в Golang

Результат работы программы на языке программирования Golang, отвечающий за нахождение параметров модели Лотки-Вольтерры с помощью интегрального и лог. интегрального методов. Полный код программы задания в Приложении 2.

Среда разработки: go.dev

Результат работы программы представлен на рисунке 8

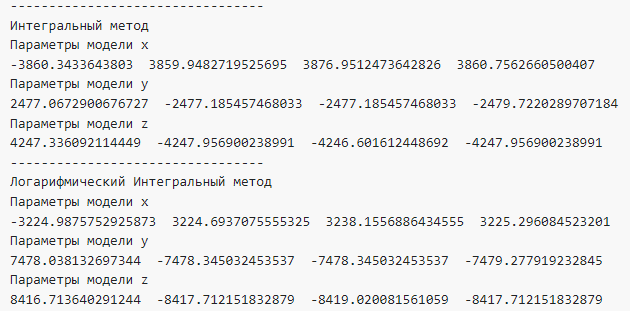


Рисунок 8- Результат работы программы на языке Golang, реализующей интегральный и лог. Интегральный методы.

Результат работы программы на языке программирования Golang, решающей систему уравнений с помощью метода Рунге-Кутта на языке программирования Golang.

Полный код программы задания в Приложении 4.

Среда разработки: go.dev

Результат работы программы представлен на рисунках 18-19

# Реализация задачи в Microsoft Office Excel 2013

На рисунках 7-15 представлена реализация, отвечающая за нахождение параметров модели Лотки-Вольтерры с помощью интегрального и лог. интегрального методов в Microsoft Office Excel.

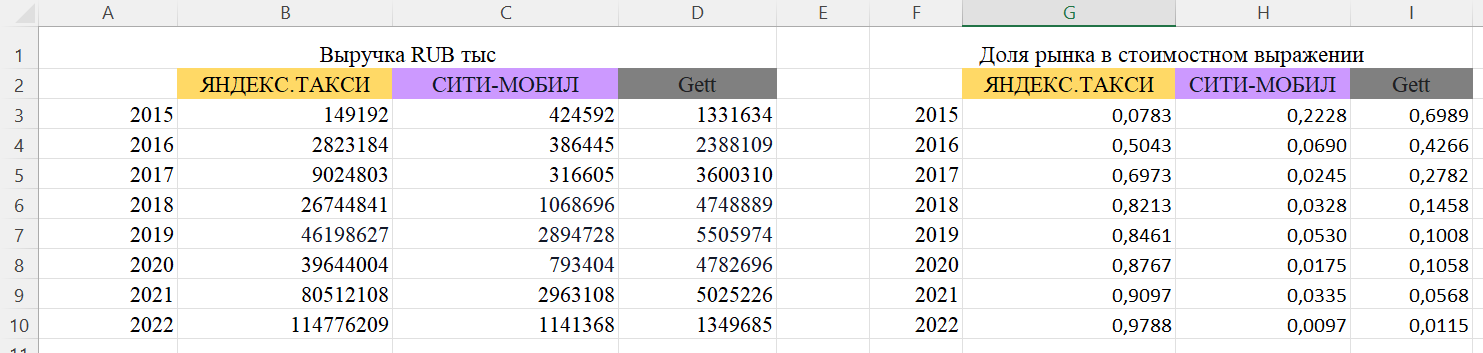


Рисунок 7 – Статистические данные, к которым будет применяться интегральный и лог. интегральный методы в Microsoft Office Excel.

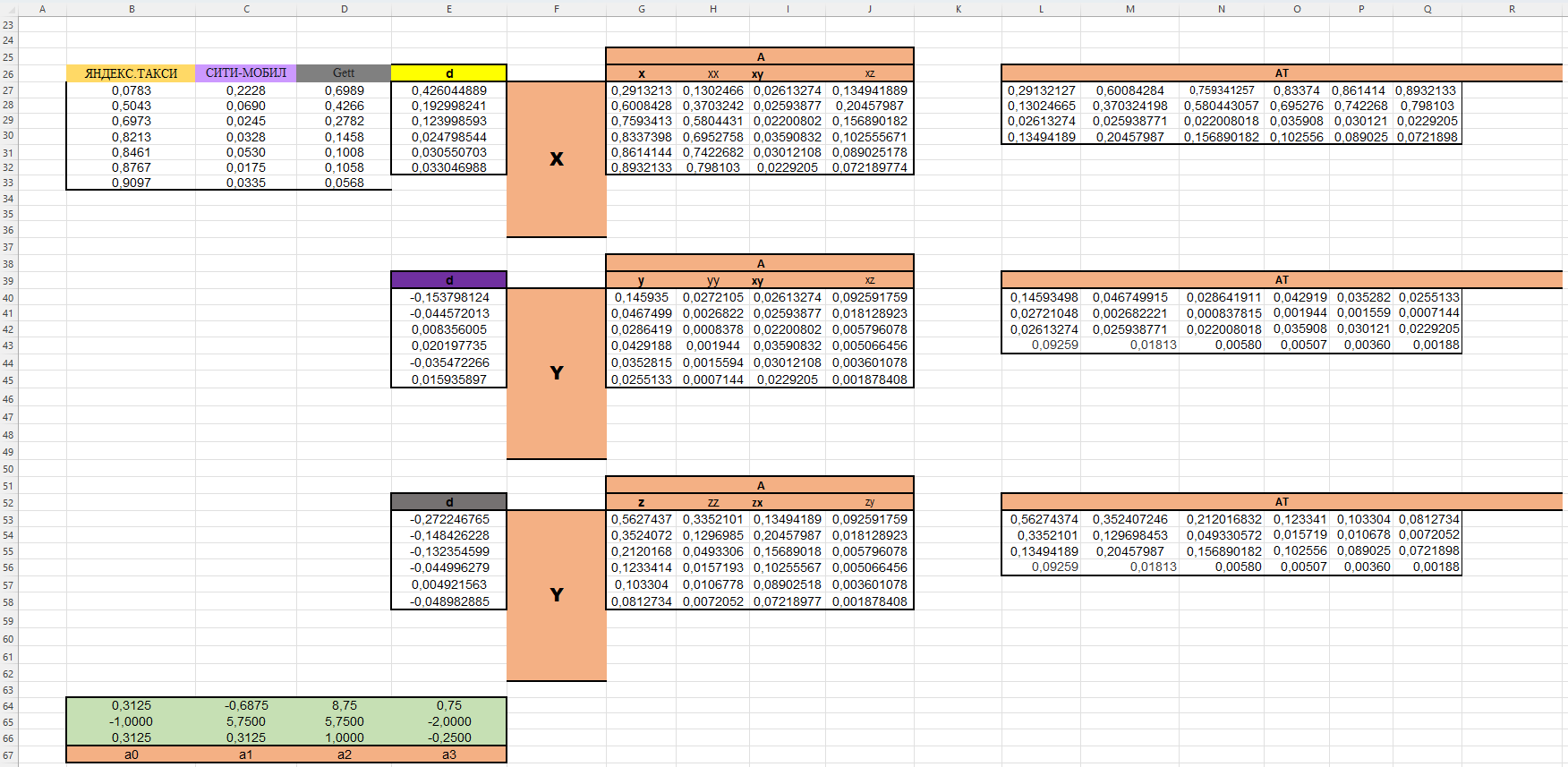


Рисунок 8– Реализация интегрального метода в Microsoft Office Excel без формул часть 1.

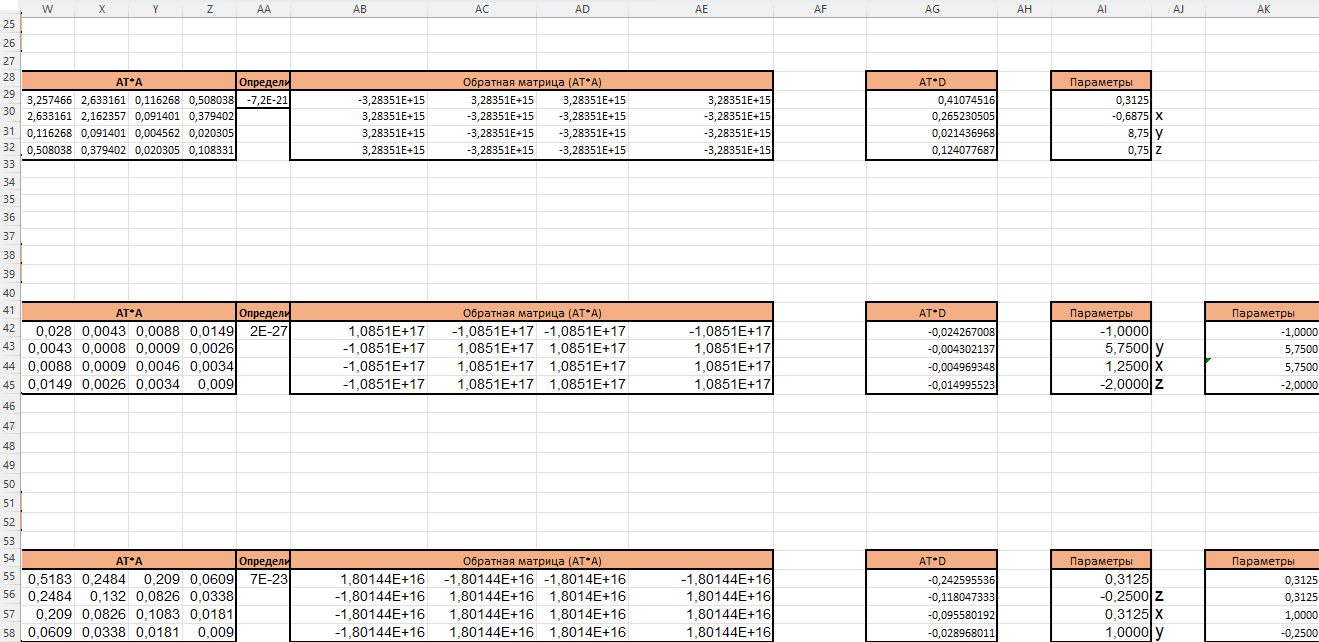


Рисунок 9 – Реализация интегрального метода в Microsoft Office Excel без формул часть 2.

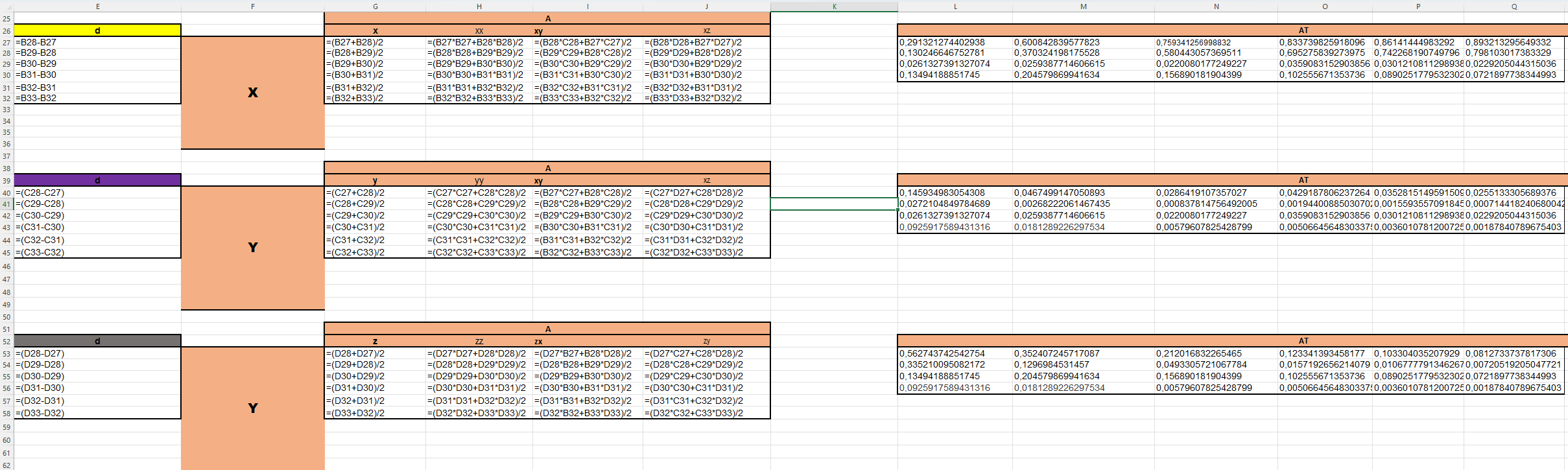


Рисунок 10– Реализация интегрального метода в Microsoft Office Excel с формулами. Часть 1.

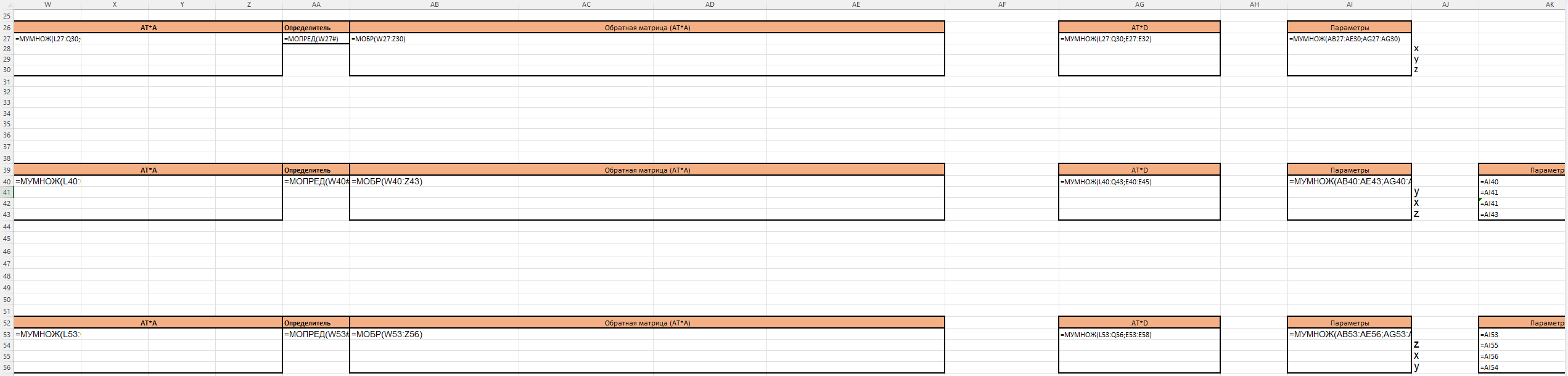


Рисунок 11– Реализация интегрального метода в Microsoft Office Excel с формулами. Часть 2.

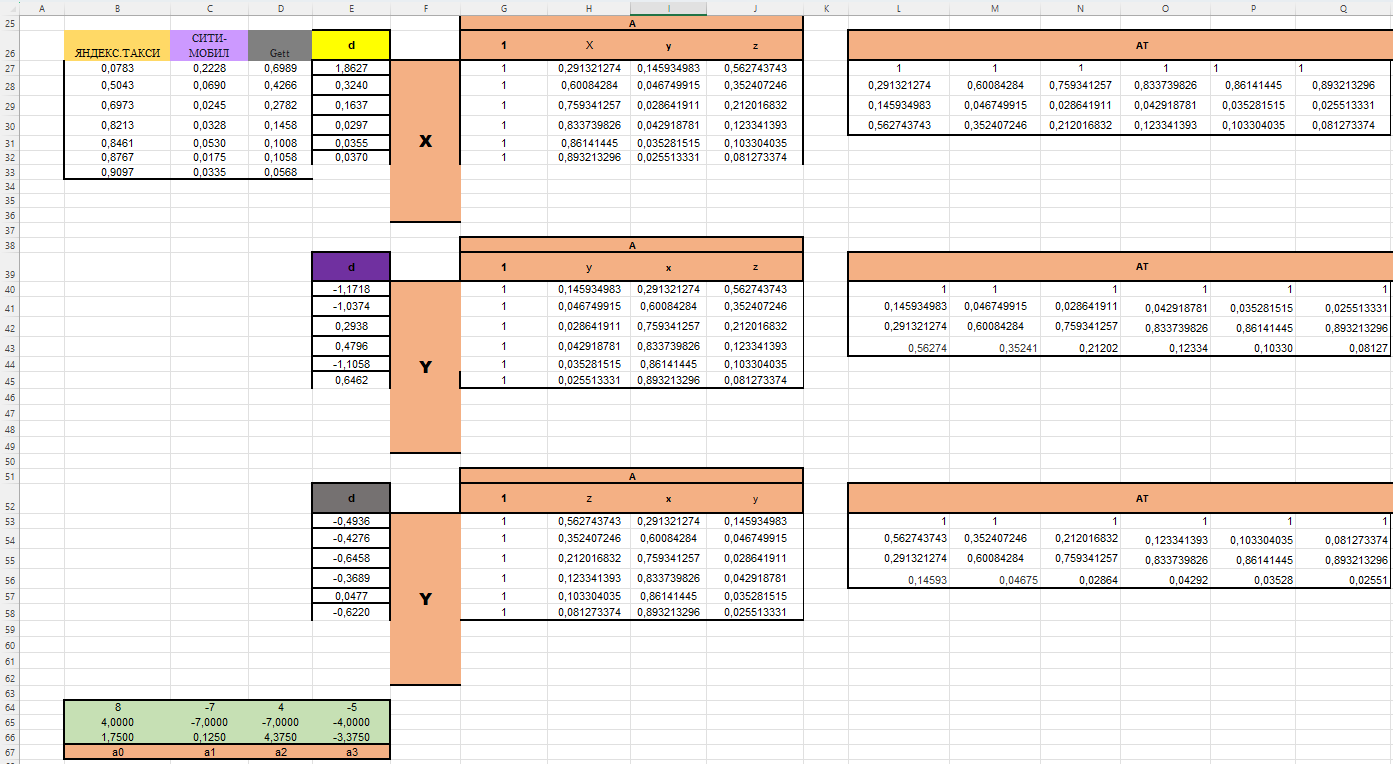


Рисунок 12– Реализация лог. интегрального метода в Microsoft Office Excel без формул. Часть 1.

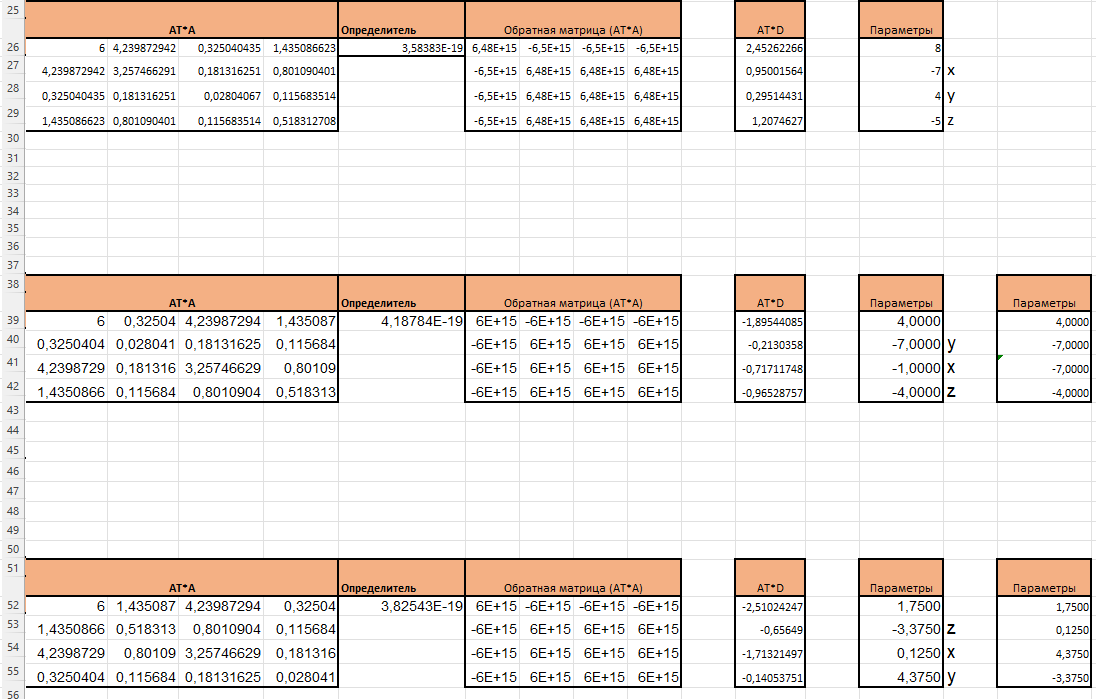


Рисунок 13– Реализация лог. интегрального метода в Microsoft Office Excel без формул. Часть 2.

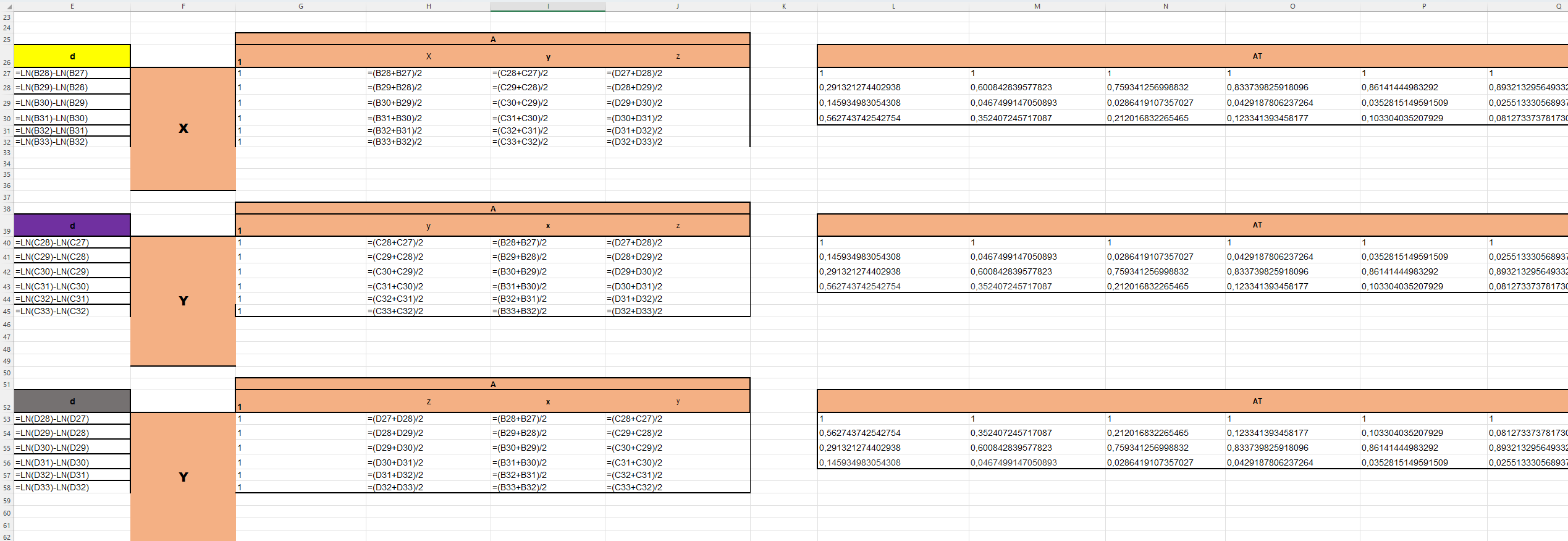


Рисунок 14– Реализация лог. интегрального метода в Microsoft Office Excel с формулами. Часть 1.

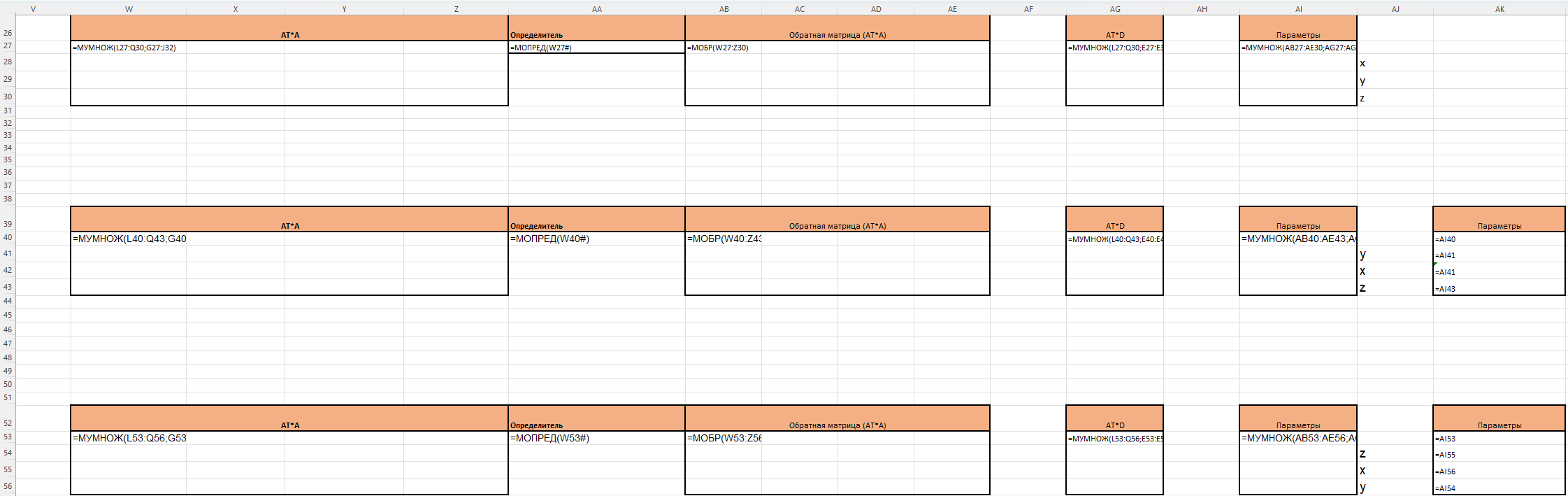


Рисунок 15– Реализация лог. интегрального метода в Microsoft Office Excel с формулами. Часть 2.

# Результаты и их анализ

Параметры каждой модели отличаются. Отличие происходит только из-за крайне малого определителя матрицы, встречаемого в ходе решения методом отражения, что мешает вычислению обратной матрицы и сталкивается с техническими особенностями программ при вычислении малых чисел с плавающей точкой. В остальных местах ход решения полностью совпадает у всех методов. Лишним подтверждением особенностей вычисления является совпадения некоторых параметров модели.

Системы, получившиеся в ходе работы перечислены ниже.

Где, каждая система, полученная с помощью определённого метода и программы или языка программирования, 31 – интегральный метод Excel, 32 – лог. интегральный метод Excel, 33 – интегральный метод Golang, 34 - лог. интегральный метод Golang, 35 - интегральный метод Mathcad, 36 - лог. интегральный метод Mathcad, 37 - интегральный метод C++, 38 - лог. интегральный метод C++.

Полученные системы демонстрируют ожидаемые изменения долей рынка в зависимости от текущих долей. Проанализируем корректность систем.

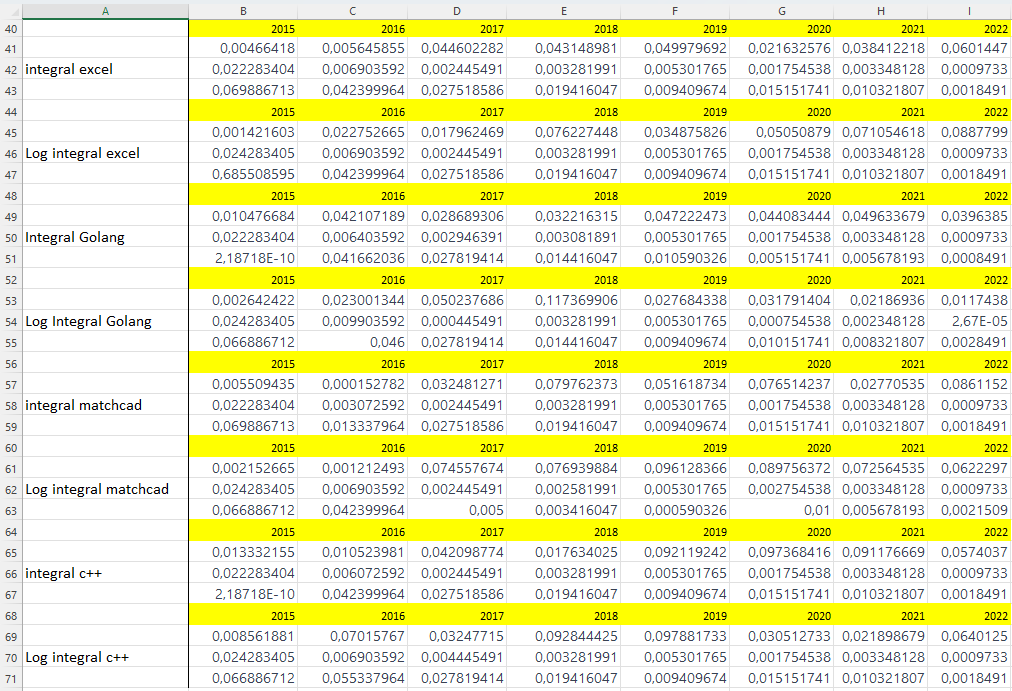


Рисунок 16– Абсолютные ошибки по каждому году, по каждой компании, по каждому методу, показывающие отклонения реальных данных от модельных.

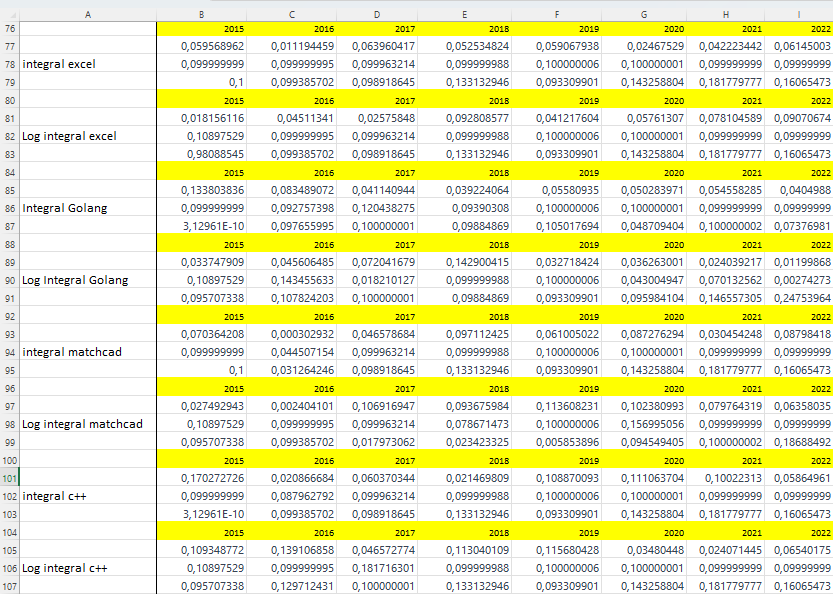


Рисунок 17– Относительные ошибки в процентах по каждому году, по каждой компании, по каждому методу, показывающие отклонения реальных данных от модельных.

## Динамическая система.

Изначально системы (31-39) показывают ожидаемое отклонение от показателей текущего кода. С помощью системы такого типа можно создать динамическую систему, которая на основе предыдущих показателей, покажет ожидаемые:

где − доли на рынке двух самых прибыльных предприятий, − доля остальных предприятий по Нижегородской области, − это параметры, которые определяют влияние внешних факторов, − параметр, описывающий влияние доли рынка на долю этой же компании, − параметр, описывающий влияние доли рынка на долю , − параметр, описывающий влияние доли рынка на долю , − изменения ожидаемые в следующем году для компании x и т.д. Динамическая система примет вид:

Составляем динамическую систему (40) для каждой системы (31-38) и восстанавливаем данные с 2015 по 2022гг.

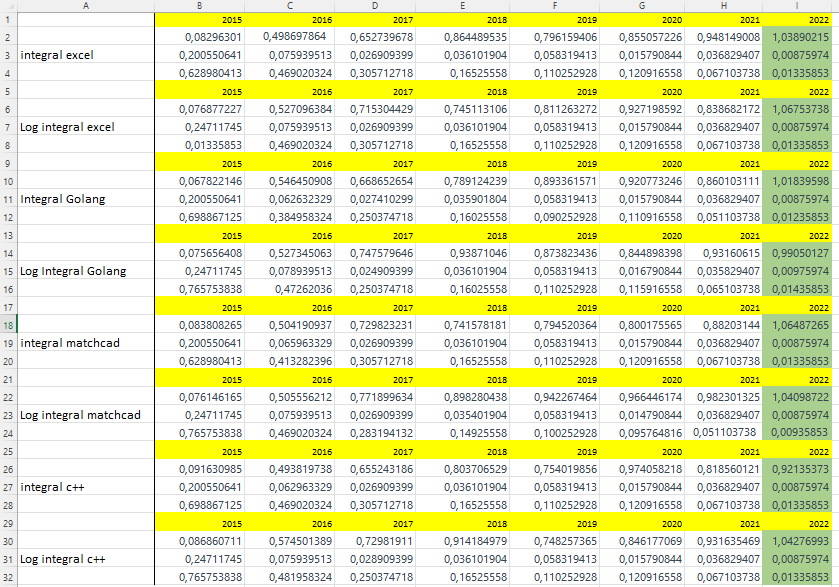


Рисунок 18– Данные, полученные с помощью динамической модели.

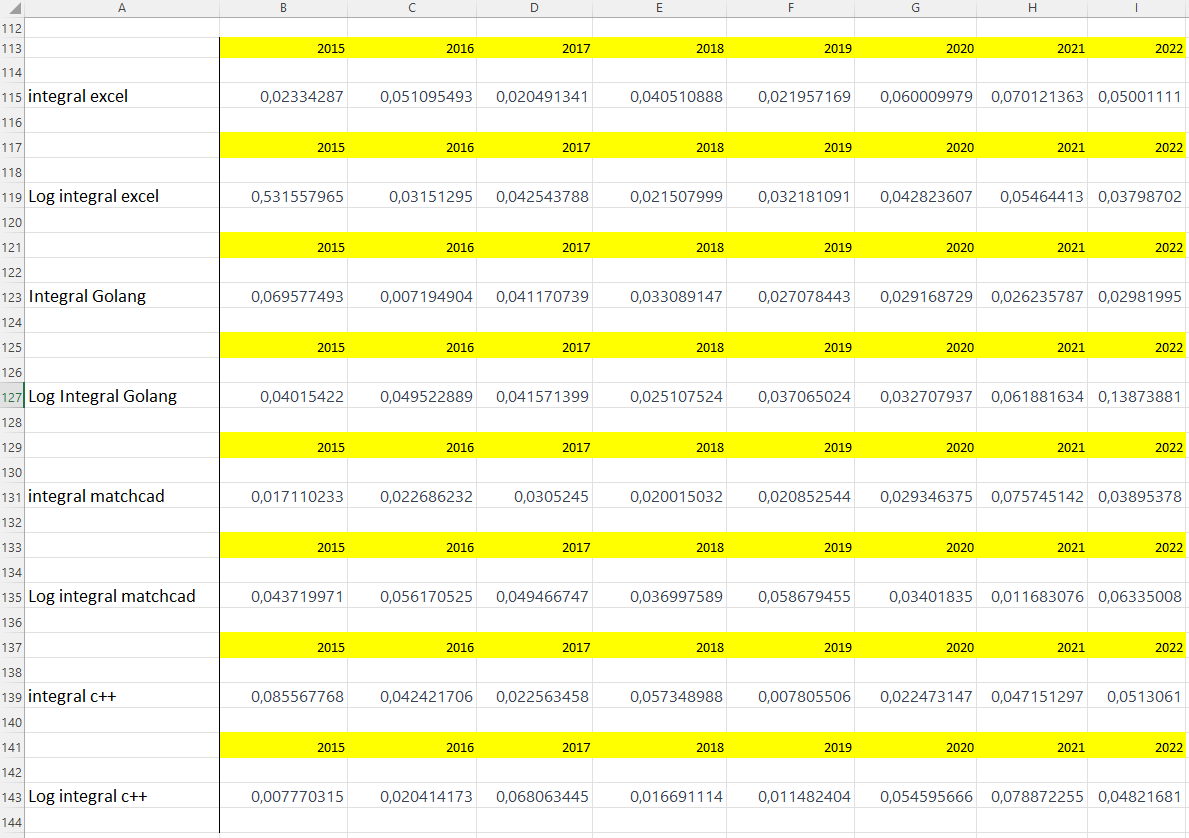


Рисунок 19– Среднеквадратичное отклонение реальных данных и данных, полученных с помощью динамических систем для каждой компании.

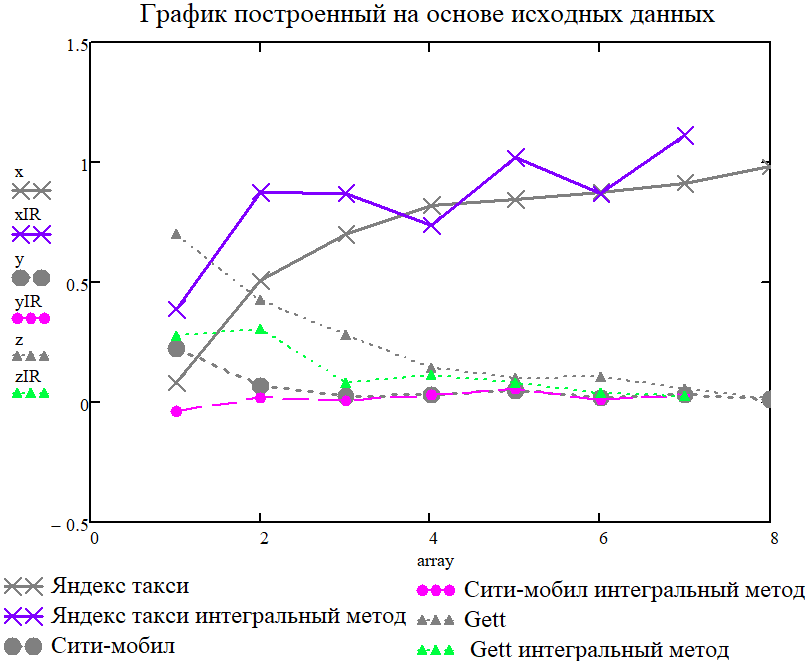


Рисунок 20– График составленных модельных данных системы (35) и реальных.

## Статическая система.

Также была составлена статическая система с помощью аппроксимации статистических данных.

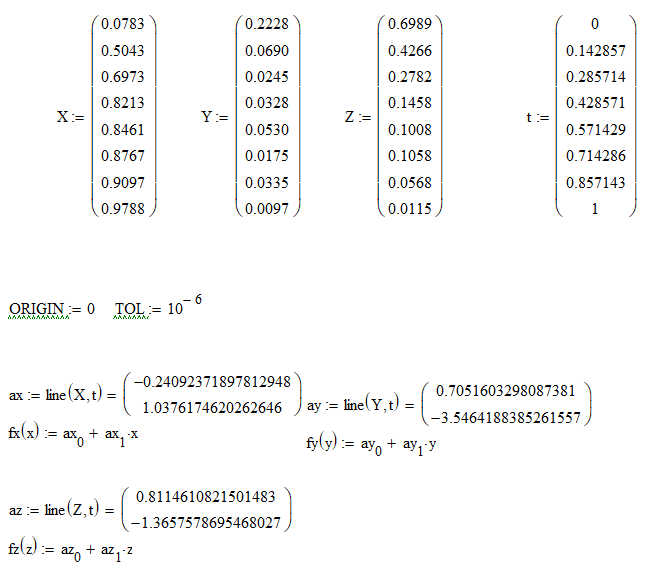


Рисунок 21– Процесс составления статической модели в Mathcad.

Полученная модель имеет вид:

Статический вид модели, при котором составляющие функции системы не зависят показателей компаний.

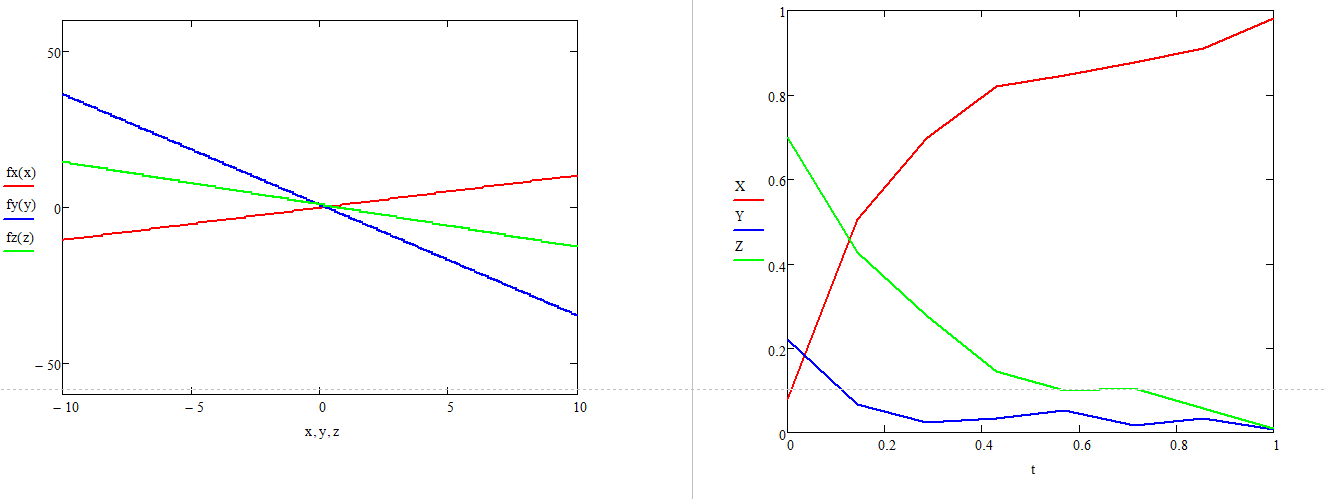


Рисунок 21– Графическое сравнение графика исходных данных и полученной модели (40).

Ручной счёт методом Рунге-Кутта.

Для интегрального метода:

Метод Рунге-Кутта:

Перепишем систему дифференциальных уравнений в виде:

= 0.1

x0 = 0.0783, y0 = 0.2228, z0 = 0.6989

Ищем k1 для функций x, y и z соответственно:

Теперь для k2:

k3:

k4:

Находим значения x, y, z:

Далее находятся точки вплоть до n.

## Анализ результатов работы программы, реализующей метод Рунге-Кутта.

Дальнейший анализ заключался в подстановке параметров модели в программу, решающую систему методом Рунге-Кутта. Результаты работы программы изображены на рисунке 22. В первой колоне отображена компания «ЯНДЕКС.ТАКСИ», второй – «СИТИ-МОБИЛ», а в третей – «Gett».

.

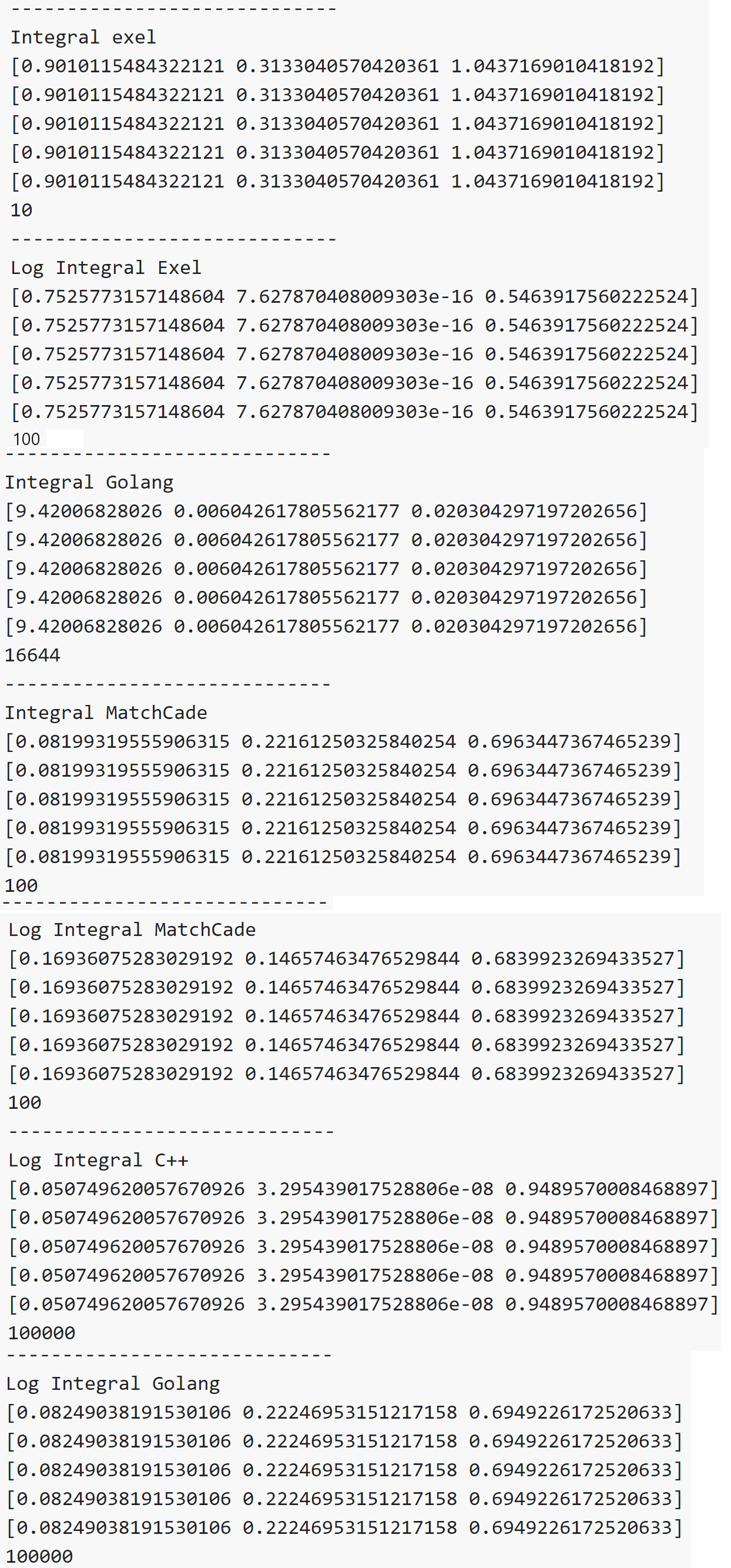


Рисунок 22– Результаты прогноза на основе реальных статистических данных.

Почти во всех моделях при решении методом Рунге-Кутта происходит один сценарий. «ЯНДЕКС.ТАКСИ» вытесняет конкурентов. Если изучить график реальных статистических данных (рис 23), то, действительно, можно прийти к выводу, что «ЯНДЕКС.ТАКСИ» стремительно наращивает долю рынка, несмотря на то что он уже к 2021 году имел основную его часть 90% и в настоящий момент предоставляет реальную угрозу для существования остальных компаний.



Рисунок 23– График статистических данных за 2021-2022 гг.

# Вывод

В результате работы составлены модели Лотки-Вольтерры с использованием различных методов, программ и языков программирования. Было выяснено, что использование метода наименьших квадратов для нахождения параметров модели может привести к вычислительным ошибкам или построению неправильной модели.

Также представляют интерес заметные различия между наборами значений параметров в соответствующих моделях. Это явление иллюстрирует общепринятый факт, что для чрезмерно детерминированной (переопределенной) математической системы существует множество решений.

С помощью метода Рунге-Кута и прогноза на основе реальных данных (рис. 19) было подтверждено, что логарифмический интегральный метод является улучшенной версией интегрального. Точность и корректность такой модели выше.

С помощью Рунге-Кутта были построены обоснованные теории о дальнейшем развитии такой системы. Однако, необходимо учитывать, что прогнозирование результатов на основе моделей имеет свои ограничения и не может полностью учесть все переменные и неожиданные события, которые могут повлиять на рыночные условия. Поэтому важно принимать во внимание и другие факторы и делать анализ рыночных условий на регулярной основе.

Подводя итог проведённой работы, можно сделать вывод, что использование модели Лотки-Вольтерра способно достаточно точно спрогнозировать развитие рынка на ближайший год.

В целом, прогнозирование и анализ рыночных условий позволяют компаниям принимать более обоснованные и информированные решения, что может способствовать их успешной деятельности. Хорошо продуманный и точный прогноз точно поможет компании достичь успеха.

# Список используемой литературы

1. Трубецков Д.И Феномен математической модели Лотки-Вольтерры и сходных с ней / Трубецков Д.И // Изв.вузов "ПНД". — 2011. — № 2. — С. 69.
2. Устойчивое развитие и угрозы экономической безопасности / С. Н. Митяков, Д. Н. Лапаев, Л. Ю. Катаева, С. А. Рамазанов // Экономика и предпринимательство. – 2019. – №
3. Д.А. Масленников, С. Н. Митяков, Л. Ю. Катаева, Т. А. Федосеева, Выявление особенностей стратегического развития регионов на основе статистического анализа индикаторов. Нижегородский государственный технический университет им. Р. Е. Алексеева, г. Нижний Новгород. Экономика региона т.15, вып. 3, 2019. С.2.
4. P. H. Kloppers, Johanna C. Greeff Lotka–Volterra model parameter estimation using experiential data // Applied Mathematics and Computation – 2013 – Vol. 224. – P.818.
5. База нормативных документов и законодательных актов [Электронный ресурс]. URL:: <https://e-ecolog.ru/> (дата обращения 16.11.2023)
6. Бухгалтерский учет. Налоги. Аудит. [Электронный ресурс] URL:: <https://www.audit-it.ru/> (дата обращения 16.11.2023)
7. Ресурс ФБО [Электронный ресурс]. URL: https://bo.nalog.ru/ (дата обращения: 19.11.2023).
8. P. H. Kloppers, Johanna C. Greeff Lotka–Volterra model parameter estimation using experiential data // Applied Mathematics and Computation – 2013 – Vol. 224. – P. – 818–820
9. Катаева Л.Ю., Масленников Д.А., Лощилова Н.А., Белоцерковская И.Е., Галина Н.В., Федосеева Т.А., Ильичева М.Н., Численные методы решения прикладных задач, 2014. С. 225 – 240.

# Приложения

## Приложение 1. Описание и код программ на языке C++, для нахождения параметров модели.

Описание программы:

Программа состоит из нескольких функций, на которые возложены базовые матричные операции или другие простые действия. Главная идея заключена в двух функциях, которые реализуют методы нахождения параметров.

Среда разработки: Visual Studio 2022

Полный код программы на С++:

#include <iostream>

#include <vector>

#include <locale.h>

#include <cmath>

using namespace std;

// Печатает двумерный массив

void print2DArray(vector<vector<double>> arr, bool strong) {

for (auto row : arr) {

for (auto val : row) {

if (strong) {

printf("%-8.9f", val);

}

else {

printf("%-8.2f", val);

}

}

std::cout << std::endl;

}

}

// Печатает одномерный массив в консоль.

void print1DArray(vector<double> arr) {

for (auto val : arr) {

std::cout << val << " ";

}

}

// Транспонирует двумерную матрицу.

vector<vector<double>> transpose2DArray(vector<vector<double>> arr) {

int n = arr.size();

int m = arr[0].size();

vector<vector<double>> transposed(m, vector<double>(n));

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

transposed[j][i] = arr[i][j];

}

}

return transposed;

}

// Умножение двух двумерных массивов.

vector<vector<double>> multiply2DArrays(vector<vector<double>> a, vector<vector<double>> b) {

int n = a.size();

int m = a[0].size();

int p = b[0].size();

std::vector<vector<double>> result(n, vector<double>(p));

for (int i = 0; i < n; i++) {

for (int j = 0; j < p; j++) {

for (int k = 0; k < m; k++) {

result[i][j] += a[i][k] \* b[k][j];

}

}

}

return result;

}

// Умножение двумерного массива на одномерный .

vector<double> matrixMultiplication(vector<vector<double>> matrix, vector<double> vectoR) {

int rows = matrix.size();

int cols = matrix[0].size();

vector<double> result(rows);

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

result[i] += matrix[i][j] \* vectoR[j];

}

}

return result;

}

// Находит обратную матрицу методом Гаусса.

std::vector<std::vector<double>> inverseMatrix(std::vector<std::vector<double>> matrix) {

int n = matrix.size();

// Создаем расширенную матрицу [matrix | I]

std::vector<std::vector<double>> augmentedMatrix;

augmentedMatrix.resize(n, std::vector<double>(2 \* n));

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

augmentedMatrix[i][j] = matrix[i][j];

}

augmentedMatrix[i][i + n] = 1.0f;

}

// Приводим расширенную матрицу к диагональному виду методом Гаусса-Жордана

for (int i = 0; i < n; i++) {

// Делим i-ю строку на matrix[i][i] для приведения элемента matrix[i][i] к 1

double divisor = augmentedMatrix[i][i];

for (int j = 0; j < 2 \* n; j++) {

augmentedMatrix[i][j] /= divisor;

}

// Обнуляем все элементы столбца, кроме элемента matrix[i][i]

for (int k = 0; k < n; k++) {

if (k == i) {

continue;

}

double factor = augmentedMatrix[k][i]; // Фактор, на который нужно умножить i-ю строку перед вычитанием

for (int j = 0; j < 2 \* n; j++) {

augmentedMatrix[k][j] -= factor \* augmentedMatrix[i][j];

}

}

}

// Извлекаем обратную матрицу из правой части расширенной матрицы

std::vector<std::vector<double>> inverse;

inverse.resize(n, std::vector<double>(n));

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

inverse[i][j] = augmentedMatrix[i][j + n];

}

}

return inverse;

}

// Находит параметры модели интегральным методом.

vector<double> findCIM(vector<double> x, vector<double> y, vector<double> z, std::string xString, std::string yString, std::string zString) {

int n = x.size();

vector<vector<double>> X(n - 2, vector<double>(4));

vector<double> d;

for (int i = 0; i < n - 2; i++) {

X[i][0] = (x[i + 1] + x[i]) / 2;

X[i][1] = (x[i + 1] \* x[i + 1] + x[i] \* x[i]) / 2;

X[i][2] = (x[i + 1] \* y[i + 1] + x[i] \* y[i]) / 2;

X[i][3] = (x[i + 1] \* z[i + 1] + x[i] \* z[i]) / 2;

d.push\_back(x[i + 1] - x[i]);

}

std::vector<std::vector<double>> XT = transpose2DArray(X);

std::vector<std::vector<double>> temp = multiply2DArrays(XT, X);

temp = inverseMatrix(temp);

d = matrixMultiplication(XT, d);

d = matrixMultiplication(temp, d);

vector<double> copyD;

copyD = d;

if (xString == "y" && yString == "x" && zString == "z") {

d[1] = copyD[2];

d[2] = copyD[1];

}

if (xString == "z" && yString == "x" && zString == "y") {

d[1] = copyD[2];

d[2] = copyD[3];

d[3] = copyD[1];

}

return d;

}

// Находит параметры модели логарифмическим интегральным методом.

vector<double> findCLIM(vector<double> x, vector<double> y, vector<double> z, std::string xString, std::string yString, std::string zString) {

int n = x.size();

vector<vector<double>> X(n - 2, vector<double>(4));

vector<double> d;

for (int i = 0; i < n - 2; i++) {

X[i][0] = 1;

X[i][1] = (x[i + 1] + x[i]) / 2;

X[i][2] = (y[i + 1] + y[i]) / 2;

X[i][3] = (z[i + 1] + z[i]) / 2;

d.push\_back(log(x[i + 1]) - log(x[i]));

}

std::vector<std::vector<double>> XT = transpose2DArray(X);

std::vector<std::vector<double>> temp = multiply2DArrays(XT, X);

temp = inverseMatrix(temp);

d = matrixMultiplication(XT, d);

d = matrixMultiplication(temp, d);

vector<double> copyD;

copyD = d;

if (xString == "y" && yString == "x" && zString == "z") {

d[1] = copyD[2];

d[2] = copyD[1];

}

if (xString == "z" && yString == "x" && zString == "y") {

d[1] = copyD[2];

d[2] = copyD[3];

d[3] = copyD[1];

}

return d;

}

int main() {

setlocale(LC\_ALL, "Russian");

vector<double> x = { 0.0783, 0.5043, 0.6973, 0.8213, 0.8461, 0.8767, 0.9097, 0.9788 };

vector<double> y = { 0.2228, 0.0690, 0.0245, 0.0328, 0.0530, 0.0175, 0.0335, 0.0097 };

vector<double> z = { 0.6989, 0.4266, 0.2782, 0.1458, 0.1008, 0.1058, 0.0568, 0.0115 };

vector<double> ax = findCIM(x, y, z, "x", "y", "z");

vector<double> ay = findCIM(y, x, z, "y", "x", "z");

vector<double> az = findCIM(z, x, y, "z", "x", "y");

cout << "---------------------------\n";

cout << "Интегральный метод" << endl;

cout << "Параметры модели x" << endl;

print1DArray(ax);

cout << "\n";

cout << "Параметры модели y" << endl;

print1DArray(ay);

cout << "\n";

cout << "Параметры модели z" << endl;

print1DArray(az);

cout << "\n";

ax = findCLIM(x, y, z, "x", "y", "z");

ay = findCLIM(y, x, z, "y", "x", "z");

az = findCLIM(z, x, y, "z", "x", "y");

cout << "---------------------------\n";

cout << "Логарифмический интегральный метод" << endl;

cout << "Параметры модели x" << endl;

print1DArray(ax);

cout << "\n";

cout << "Параметры модели y" << endl;

print1DArray(ay);

cout << "\n";

cout << "Параметры модели z" << endl;

print1DArray(az);

return 0;

}

## Приложение 2. Описание и код программ на языке Go, для нахождения параметров модели интегральным и лог. интегральным методами.

Программа состоит из нескольких унитарный функций, на которые возложены базовые матричные операции или другие простые действия. Главная идея заключена в двух функциях, которые реализуют методы нахождения параметров. Функция findCIM реализует формулы (1-5) для нахождений системы с помощью интегрального метода, а функция findCLIM реализует формулы (6-9) для нахождения системы с помощью лог. интегрального метода.

Среда разработки: go.dev

Полный код программы на Go:

## package main

## import (

## "fmt"

## "math"

## )

## // Печатает двумерный массив в красивой форме.

## // strong - позволяет указать много или мало знаков после запятой нужно печатать

## func print2DArray(arr [][]float64, strong bool) {

## for \_, row := range arr {

## for \_, val := range row {

## if strong {

## fmt.Printf("%-8.9f", val)

## } else {

## fmt.Printf("%-8.2f", val)

## }

## }

## fmt.Println()

## }

## }

## // Печатает одномерный массив в консоль.

## func print1DArray(arr []float64) {

## for \_, row := range arr {

## fmt.Print(row, " ")

## }

## }

## // Транспонирует двумерную матрицу.

## func transpose2DArray(arr [][]float64) [][]float64 {

## n, m := len(arr), len(arr[0])

## transposed := make([][]float64, m)

## for i := range transposed {

## transposed[i] = make([]float64, n)

## }

## for i := 0; i < n; i++ {

## for j := 0; j < m; j++ {

## transposed[j][i] = arr[i][j]

## }

## }

## return transposed

## }

## // Умножение двух двумерных массивов.

## func multiply2DArrays(a [][]float64, b [][]float64) [][]float64 {

## n, m, p := len(a), len(a[0]), len(b[0])

## result := make([][]float64, n)

## for i := range result {

## result[i] = make([]float64, p)

## }

## for i := 0; i < n; i++ {

## for j := 0; j < p; j++ {

## for k := 0; k < m; k++ {

## result[i][j] += a[i][k] \* b[k][j]

## }

## }

## }

## return result

## }

## // Умножение двумерного массива на одномерный .

## func matrixMultiplication(matrix [][]float64, vector []float64) []float64 {

## rows := len(matrix)

## cols := len(matrix[0])

## result := make([]float64, rows)

## for i := 0; i < rows; i++ {

## for j := 0; j < cols; j++ {

## result[i] += matrix[i][j] \* vector[j]

## }

## }

## return result

## }

## // Находит обратную матрицу методом Гаусса.

## func inverseMatrix(matrix [][]float64) [][]float64 {

## // Проверяем, что матрица является квадратной

## n := len(matrix)

## if n == 0 || len(matrix[0]) != n {

## return nil

## }

## // Создаем расширенную матрицу [A, I], где I - единичная матрица

## extendedMatrix := make([][]float64, n)

## for i := range extendedMatrix {

## extendedMatrix[i] = make([]float64, 2\*n)

## for j := range extendedMatrix[i] {

## if j < n {

## extendedMatrix[i][j] = matrix[i][j]

## } else if j-n == i {

## extendedMatrix[i][j] = 1

## }

## }

## }

## // Приводим extendedMatrix к диагональному виду, получаем на месте [I, A^(-1)]

## for i := 0; i < n; i++ {

## // Делим текущую строку на диагональный элемент

## diagonalElement := extendedMatrix[i][i]

## for j := i; j < 2\*n; j++ {

## extendedMatrix[i][j] /= diagonalElement

## }

## // Вычитаем текущую строку из остальных строк

## for k := 0; k < n; k++ {

## if k != i {

## multiplier := extendedMatrix[k][i]

## for j := i; j < 2\*n; j++ {

## extendedMatrix[k][j] -= multiplier \* extendedMatrix[i][j]

## }

## }

## }

## }

## // Извлекаем обратную матрицу из полученного extendedMatrix

## inverse := make([][]float64, n)

## for i := range inverse {

## inverse[i] = make([]float64, n)

## for j := range inverse[i] {

## inverse[i][j] = extendedMatrix[i][j+n]

## }

## }

## return inverse

## }

## // Находит параметры модели интегральным методом.

## func findCIM(x, y, z []float64, xString, yString, zString string) []float64 {

## n := len(x)

## X := make([][]float64, n-1)

## for i := range X {

## X[i] = make([]float64, 4)

## }

## var d []float64

## for i := 0; i < n-1; i++ {

## X[i][0] = (x[i+1] + x[i]) / 2

## X[i][1] = (x[i+1]\*x[i+1] + x[i]\*x[i]) / 2

## X[i][2] = (x[i+1]\*y[i+1] + x[i]\*y[i]) / 2

## X[i][3] = (x[i+1]\*z[i+1] + x[i]\*z[i]) / 2

## d = append(d, x[i+1]-x[i])

## }

## XT := transpose2DArray(X)

## temp := multiply2DArrays(XT, X)

## temp = inverseMatrix(temp)

## d = matrixMultiplication(XT, d)

## d = matrixMultiplication(temp, d)

## //Всопомогательная матрица, которая хранит значения, чтобы поменять местами значения в оригинальном массиве

## copyD := d

## if (xString == "y") && (yString == "x") && (zString == "z") {

## d[1] = copyD[2]

## d[2] = copyD[1]

## }

## if (xString == "z") && (yString == "x") && (zString == "y") {

## d[1] = copyD[2]

## d[2] = copyD[3]

## d[3] = copyD[1]

## }

## return d

## }

## // Находит параметры модели логарифмическим интегральным методом.

## func findCLIM(x, y, z []float64, xString, yString, zString string) []float64 {

## n := len(x)

## X := make([][]float64, n-1)

## for i := range X {

## X[i] = make([]float64, 4)

## }

## var d []float64

## for i := 0; i < n-1; i++ {

## X[i][0] = 1

## X[i][1] = (x[i+1] + x[i]) / 2

## X[i][2] = (y[i+1] + y[i]) / 2

## X[i][3] = (z[i+1] + z[i]) / 2

## d = append(d, math.Log(x[i+1])-math.Log(x[i]))

## }

## XT := transpose2DArray(X)

## temp := multiply2DArrays(XT, X)

## temp = inverseMatrix(temp)

## d = matrixMultiplication(XT, d)

## d = matrixMultiplication(temp, d)

## copyD := d

## if (xString == "y") && (yString == "x") && (zString == "z") {

## d[1] = copyD[2]

## d[2] = copyD[1]

## }

## if (xString == "z") && (yString == "x") && (zString == "y") {

## d[1] = copyD[2]

## d[2] = copyD[3]

## d[3] = copyD[1]

## }

## return d

## }

## func main() {

## x := []float64{0.0783, 0.5043, 0.6973, 0.8213, 0.8461, 0.8767, 0.9097, 0.9788}

## y := []float64{0.2228, 0.0690, 0.0245, 0.0328, 0.0530, 0.0175, 0.0335, 0.0097}

## z := []float64{0.6989, 0.4266, 0.2782, 0.1458, 0.1008, 0.1058, 0.0568, 0.0115}

## fmt.Println("---------------------------------")

## fmt.Println("Интегральный метод")

## ax := findCIM(x, y, z, "x", "y", "z")

## ay := findCIM(y, x, z, "y", "x", "z")

## az := findCIM(z, x, y, "z", "x", "y")

## fmt.Println("Параметры модели x")

## print1DArray(ax)

## fmt.Println("\nПараметры модели y")

## print1DArray(ay)

## fmt.Println("\nПараметры модели z")

## print1DArray(az)

## fmt.Println()

## fmt.Println("---------------------------------")

## fmt.Println("Логарифмический Интегральный метод")

## ax = findCLIM(x, y, z, "x", "y", "z")

## ay = findCLIM(y, x, z, "y", "x", "z")

## az = findCLIM(z, x, y, "z", "x", "y")

## fmt.Println("Параметры модели x")

## print1DArray(ax)

## fmt.Println("\nПараметры модели y")

## print1DArray(ay)

## fmt.Println("\nПараметры модели z")

## print1DArray(az)

## fmt.Println()

## }

## Приложение 3. Описание и код программ на языке Go, для решения системы методом Рунге-Кутта.

Программа состоит из структуры, которая хранит в себе параметры модели и функции (f(x)), которые используют эти параметры. Главная задача возложена на функцию Runge\_Kutta, которая создает экземпляр структуры с переданными ей параметры и совершает итерационный обход с указанными количеством раз и шагом.

Полный код программы, решающей систему уравнений с помощью метода Рунге-Кутта на языке программирования Golang

Среда разработки: go.dev

Полный код программы на Go:

package main

import (

"fmt"

"math"

)

type f struct {

parameter [][]float64

}

func (ftype f) f1(x, y, z float64) float64 {

return x \* (ftype.parameter[0][0] + ftype.parameter[0][1]\*x + ftype.parameter[0][2]\*y + ftype.parameter[0][3]\*z)

}

func (ftype f) f2(x, y, z float64) float64 {

return y \* (ftype.parameter[1][0] + ftype.parameter[1][1]\*x + ftype.parameter[1][2]\*y + ftype.parameter[1][3]\*z)

}

func (ftype f) f3(x, y, z float64) float64 {

return z \* (ftype.parameter[2][0] + ftype.parameter[2][1]\*x + ftype.parameter[2][2]\*y + ftype.parameter[2][3]\*z)

}

func NewF(parameters [][]float64) \*f {

p := new(f)

p.parameter = parameters

return p

}

func Runge\_Kutta(parameter [][]float64, x, y, z, h float64, n int) (toReturn [][]float64, j int) {

j = 0

temp := make([]float64, 3)

functions := NewF(parameter)

// Задаем начальные значения

// Выполняем итерации метода Рунге-Кутта

for i := 0; i < n; i++ {

j++

k1x := h \* functions.f1(x, y, z)

k1y := h \* functions.f2(x, y, z)

k1z := h \* functions.f3(x, y, z)

k2x := h \* functions.f1(x+0.5\*k1x, y+0.5\*k1y, z+0.5\*k1z)

k2y := h \* functions.f2(x+0.5\*k1x, y+0.5\*k1y, z+0.5\*k1z)

k2z := h \* functions.f3(x+0.5\*k1x, y+0.5\*k1y, z+0.5\*k1z)

k3x := h \* functions.f1(x+0.5\*k2x, y+0.5\*k2y, z+0.5\*k2z)

k3y := h \* functions.f2(x+0.5\*k2x, y+0.5\*k2y, z+0.5\*k2z)

k3z := h \* functions.f3(x+0.5\*k2x, y+0.5\*k2y, z+0.5\*k2z)

k4x := h \* functions.f1(x+k3x, y+k3y, z+k3z)

k4y := h \* functions.f2(x+k3x, y+k3y, z+k3z)

k4z := h \* functions.f3(x+k3x, y+k3y, z+k3z)

x += (k1x + 2\*k2x + 2\*k3x + k4x) / 6

y += (k1y + 2\*k2y + 2\*k3y + k4y) / 6

z += (k1z + 2\*k2z + 2\*k3z + k4z) / 6

if x == math.Inf(1) || y == math.Inf(1) || z == math.Inf(1) || x == math.Inf(-1) || y == math.Inf(-1) || z == math.Inf(-1) {

break

}

if math.IsNaN(x) || math.IsNaN(y) || math.IsNaN(z) {

break

}

if x > 1000 || y > 1000 || z > 1000 || x < (-1000) || y < (-1000) || z < (-1000) {

break

}

temp[0] = x

temp[1] = y

temp[2] = z

toReturn = append(toReturn, temp)

// Выводим результаты

//fmt.Printf("x = %.3f, y = %.3f, z = %.3f\n", x, y, z)

}

return toReturn, j

}

func print2D(arr [][]float64) {

if len(arr) > 5 {

for i := 0; i < 5; i++ {

fmt.Println(arr[len(arr)-1-i])

}

} else {

for i := len(arr) - 1; i > 0; i-- {

fmt.Println(i)

}

}

}

func main() {

originalX := []float64{0.0783, 0.5043, 0.6973, 0.8213, 0.8461, 0.8767, 0.9097, 0.9788}

originalY := []float64{0.2228, 0.0690, 0.0245, 0.0328, 0.0530, 0.0175, 0.0335, 0.0097}

originalZ := []float64{0.6989, 0.4266, 0.2782, 0.1458, 0.1008, 0.1058, 0.0568, 0.0115}

IntegralExel := [][]float64{{0.3125, -0.6875, 8.75, 0.75}, {-1, 5.75, 5.75, -2}, {0.3125, 0.3125, 1, -0.25}} //изменил

LogIntegralExel := [][]float64{{8, -7, 4, -5}, {4, -7, -7, -4}, {1.75, 0.125, 4.375, -3.375}}

IntegralGolang := [][]float64{{-3860.3433, 3859.9482, 3876.9512, 3860.7562}, {2477.0672, -2477.1854, -2477.1854, -2479.7220}, {4247.3360, -4247.9569, -4247.9569, -4247.9569}}

LogIntegralGolang := [][]float64{{-3224.9875, 3224.6937, 3238.1556, 3225.2960}, {7478.0381, -7478.3450, -7478.3450, -7479.2779}, {8416.7136, -8417.7121, -8419.0200, -8417.7121}}

IntegralMatchCade := [][]float64{{-3860, 3859.14, 3876, 3860}, {2477, -2477, -2473, -2479}, {4247, -4247, -4246, -4248}}

LogIntegralMatchCade := [][]float64{{-3224.987, 3224.693, 3238.155, 3225.295}, {7478.037, -7478.344, -7483.708, -7479.277}, {8416.713, -8417.712, -8419.02, -8417.16}}

LogIntegralC := [][]float64{{-3009.71, 3009.23, 3025.72, 3010.31}, {-3382.42, 3382.94, 3385.5, 3379.83}, {3716.91, -3717.47, -3715.9, -3718.01}}

IntegralExelArr, i0 := Runge\_Kutta(IntegralExel, originalX[0], originalY[0], originalZ[0], 0.1, 10)

LogIntegralExelArr, i1 := Runge\_Kutta(LogIntegralExel, originalX[0], originalY[0], originalZ[0], 0.1, 100)

IntegralGolangArr, i2 := Runge\_Kutta(IntegralGolang, originalX[0], originalY[0], originalZ[0], 0.00007, 100000)

LogIntegralGolangArr, i3 := Runge\_Kutta(LogIntegralGolang, originalX[0], originalY[0], originalZ[0], 0.00019, 100)

IntegralMatchCadeArr, i4 := Runge\_Kutta(IntegralMatchCade, originalX[0], originalY[0], originalZ[0], 0.00014, 100)

LogIntegralMatchCadeArr, i5 := Runge\_Kutta(LogIntegralMatchCade, originalX[0], originalY[0], originalZ[0], 0.0000034, 100000)

LogIntegralCArr, i6 := Runge\_Kutta(LogIntegralC, originalX[0], originalY[0], originalZ[0], 0.000053, 100000)

fmt.Println("-----------------------------")

fmt.Println("Integral exel")

print2D(IntegralExelArr)

fmt.Println(i0)

fmt.Println("-----------------------------")

fmt.Println("Log Integral Exel")

print2D(LogIntegralExelArr)

fmt.Println(i1)

fmt.Println("-----------------------------")

fmt.Println("Integral Golang")

print2D(IntegralGolangArr)

fmt.Println(i2)

fmt.Println("-----------------------------")

fmt.Println("Integral MatchCade")

print2D(IntegralMatchCadeArr)

fmt.Println(i3)

fmt.Println("-----------------------------")

fmt.Println("Log Integral MatchCade ")

print2D(LogIntegralMatchCadeArr)

fmt.Println(i4)

fmt.Println("-----------------------------")

fmt.Println("Log Integral C++")

print2D(LogIntegralCArr)

fmt.Println(i5)

fmt.Println("-----------------------------")

fmt.Println("Log Integral Golang")

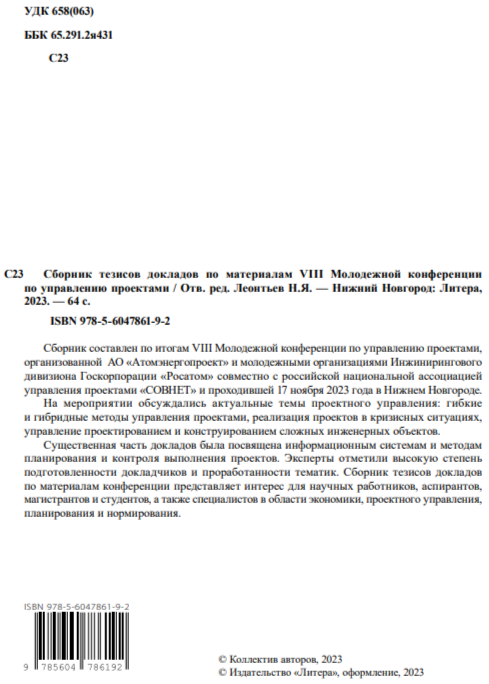
print2D(LogIntegralGolangArr)

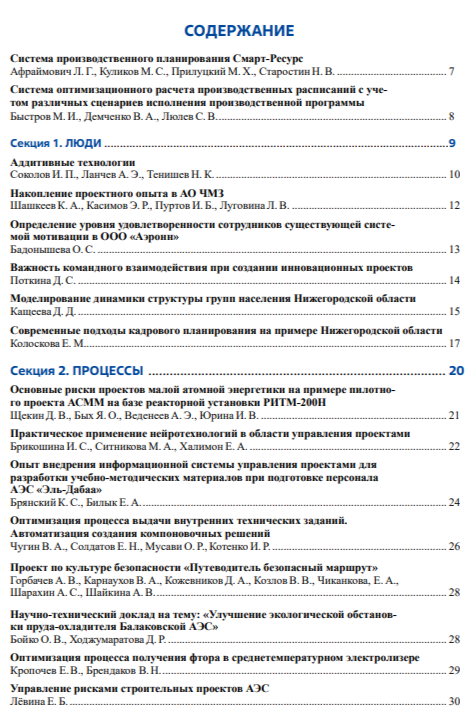
fmt.Println(i6)  
}

## Приложение 5. Участие в конференции «VIII Молодежная конференция по управлению проектами.

* 1. Дата: 17 ноября 2022 10:00 — 17 ноября 2023 18:00
  2. Место: АНО КПЦ «Академия Маяк» им. А.Д. Сахарова (Нижне-Волжская наб. 11)
  3. Организатор: «Росатом.»









УДК 330.341.2

Численное моделирование конкурентной борьбы агрегаторов такси на основе модели Лотки - Вольтерры с использованием реальных данных

Кожеватов А.Д.

*Нижегородский государственный технический университет им. Р. Е. Алексеева «НГТУ», Нижний Новгород, e-mail:* alexey.kozhevatov@mail.ru

Для обеспечения экономической безопасности страны [1, 2] и обеспечения конкурентной борьбы предприятий необходимо сформировать рынок услуг. Анализ рынка агрегаторов такси является важным для бизнеса ведь предсказание такой динамики позволяет участникам рынка принимать эффективные стратегические решения, направленные на достижение выгоды. В данной работе рассматривается конкурентная борьба агрегаторов такси таких как Яндекс такси, Сити-мобил и Gett. Анализ данных за период с 2015-2022 гг показал, что уже с 2016 году агрегатор Яндекс такси начинает занимать лидирующие позиции и стремится поглотить рынок [3, 4]. Рис 1. На рисунке виден динамичный рост доли рынка занимаемого Яндекс такси и снижение долей конкурентов почти до нуля к 2022 году.

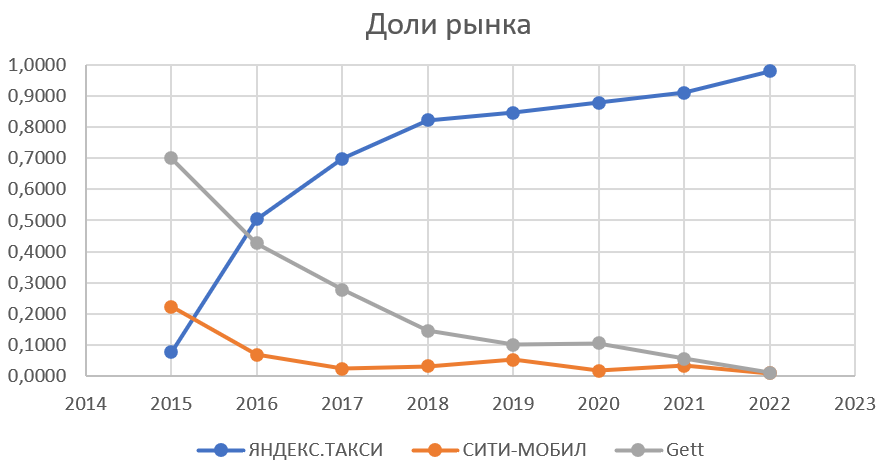


Рисунок 1 - Соответствие кривых статистическим данным о доле рынка для компаний за период 2015-2022 годов

При использовании интегрального метода, получены коэффициенты математической модели, такие коэффициенты как внутренний рост компаний и коэффициент взаимодействия компаний на рынке, описывающей конкурентную борьбу между данными агрегаторами. Введем обозначений для формализации конкурентной борьбы рассматриваемых компаний: x – доля рынка Яндекс такси, y – доля рынка Сити-мобил, z – доля рынка Gett, темпы изменения доли рынка компаний обозначены , *,* для компаний x,y и z соответственно, коэффициенты ,отвечают за внутренний рост компаний в то время как коэффициенты показывают взаимодействие и могут быть: положительными, отрицательными или нулевыми и будут отражать, взаимодействуют ли компании с точки зрения хищничества, конкуренции, мутуализма или нет вообще.

Проведен анализа полученной математической модели составлен и осуществлен прогноз долей рынка для агрегаторов такси на территории России (2023, 2024 гг.). Использование численных методов [5] дает возможность получить решение системы уравнений (2) и осуществить прогноз поведения рассматриваемых агрегаторов такси. На рис.1. показана динамика изменения доли рынка для рассматриваемых агрегаторов такси, полученная на основе решения системы (2).

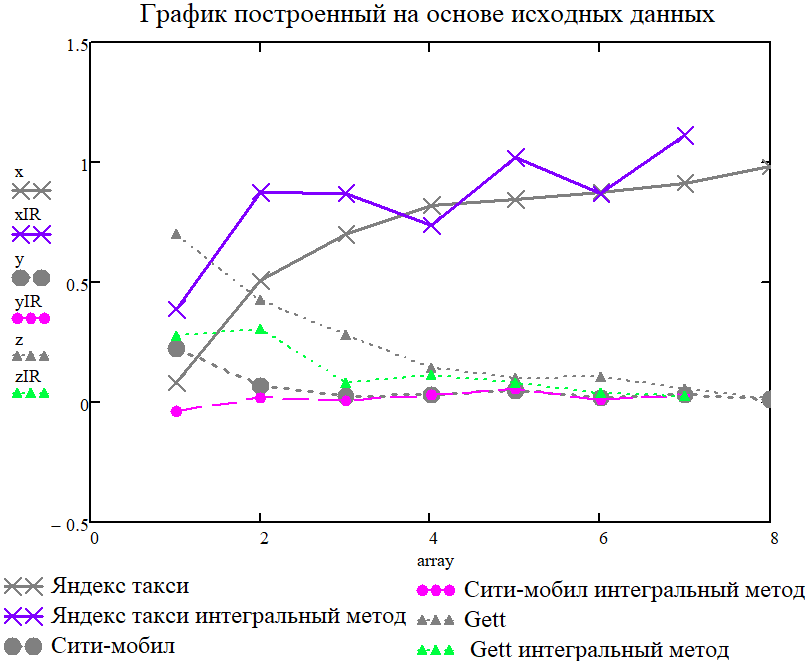


Рисунок 1 Изменение доли рынка, занимаемой соответствующим агрегатором такси

Однако, в силу сильного влияния внешних условий (неопределённость в мировой экономике, включая внешние шоки, политические события, изменение инвестиционной составляющей) на динамику развития данных компаний не дает возможности осуществить точный прогноз на длительный период времени, но прогноз на год достаточно точен и имеет отклонение меньше 5%.

Используемые источники

1. Устойчивое развитие и угрозы экономической безопасности / С. Н. Митяков, Д. Н. Лапаев, Л. Ю. Катаева, С. А. Рамазанов // Экономика и предпринимательство. – 2019. – № 10(111). – С. 111-114. – EDN LFGHVR.

2. Оперативный мониторинг экономической безопасности России / С. Н. Митяков, Л. Ю. Катаева, Е. С. Митяков, С. А. Рамазанов // Инновационное развитие экономики. – 2019. – № 5-2(53). – С. 213-223. – EDN LRUNWI.

3. База нормативных документов и законодательных актов. URL:: <https://e-ecolog.ru/> (дата обращения 16.11.2023)

4. Бухгалтерский учет. Налоги. Аудит. URL:: <https://www.audit-it.ru/> (дата обращения 16.11.2023)

5. Масленников, Д.А. Применение численных методов для решения прикладных задач экономики: учебное пособие / [Д. А. Масленников](https://elibrary.ru/author_items.asp?refid=1052084860&fam=%D0%9C%D0%B0%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D0%B8%D0%BA%D0%BE%D0%B2&init=%D0%94+%D0%90), [М. Н. Ильичева](https://elibrary.ru/author_items.asp?refid=1052084860&fam=%D0%98%D0%BB%D1%8C%D0%B8%D1%87%D0%B5%D0%B2%D0%B0&init=%D0%9C+%D0%9D), [Л. Ю. Катаева](https://elibrary.ru/author_items.asp?refid=1052084860&fam=%D0%9A%D0%B0%D1%82%D0%B0%D0%B5%D0%B2%D0%B0&init=%D0%9B+%D0%AE). - Нижний Новгород: Нижегородский государственный технический университет им. Р.Е. Алексеева, 2018. - 187 с.