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

«ПРИМЕНЕНИЕ МОДЕЛИ ЛОТКИ-ВОЛЬТЕРРА ДЛЯ ПРОГНОЗИРОВАНИЯ КОНКУРЕНЦИИ ПРОИЗВОДИТЕЛЕЙ НАПИТКОВ НИЖЕГОРОДСКОЙ ОБЛАСТИ»

Выполнил:
студент 3-го курса группы 21-САИ
Краличев Игорь Евгеньевич

Краличев И.Е.
Студент

Проверил:
д.ф.м.н., проф. Катаева Лилия Юрьевна
14.12.2023
Оценка:_____

Подпись преподавателя:_____

Нижний Новгород, 2023

Содержание

Введение **2**
Цель работы **2**
Формулировка задачи **3**
Описание переменных **3**
Данные исследования **4**
Блок-схемы алгоритмов **5**
Методы расчёта коэффициентов взаимодействия **5**
Интегральный метод **5**
Логарифмический метод **6**
Описание системы дифференциальных уравнений **7**
Решение системы дифференциальных уравнений **8**
Анализ методов **13**
Заключение **14**
Приложение 1 ..... **15**
Приложение 2 ..... **16**
Приложение 3 ..... **17**
Приложение 4 ..... **26**
Приложение 5 ..... **29**

Введение

Производство напитков играет ключевую роль в современном обществе и имеет огромное значение для потребителей, экономики и социальной сферы. Оно обеспечивает людей разнообразными напитками, удовлетворяет их потребности в питье и способствует поддержанию здоровья. Производство напитков также является важным фактором экономического развития, создавая рабочие места и стимулируя инновации в отрасли [1, 2].

Актуальность выбранной сферы исследования, связанной с применением модели Лотки-Вольтерра для прогнозирования конкуренции производителей напитков в Нижегородской области, неоспорима. В современном мире, где рынки насыщены разнообразными продуктами и услугами, конкуренция становится ключевым фактором успеха для любого бизнеса. Особенно это касается производителей напитков, которые сталкиваются с огромным количеством конкурентов и постоянно меняющимися предпочтениями потребителей.

Нижегородская область, с ее разнообразной экономикой и активно развивающимся рынком напитков, представляет особый интерес для изучения конкуренции в данной сфере. Анализ и прогнозирование конкуренции между производителями напитков в этом регионе имеет большое значение для разработки эффективных стратегий бизнеса, повышения конкурентоспособности предприятий и удовлетворения потребностей потребителей.

В условиях динамического рынка производства напитков, где конкуренция становится все более ожесточенной, прогнозирование конкуренции становится ключевым элементом успешной стратегии компаний. Одним из подходов, используемых для анализа и прогнозирования конкуренции, является модель Лотки-Вольтерра.

Модель Лотки-Вольтерра, основанная на концепции взаимодействия между видами в экосистеме, предоставляет уникальный подход к анализу и прогнозированию конкуренции. Применение данной модели позволяет учесть различные факторы, такие как рыночные условия, потребительские предпочтения, маркетинговые стратегии и многое другое, что влияет на соперничество между производителями напитков.

Применение модельных подходов обеспечит возможность оценить, как разные элементы влияют на стабильность отрасли, и предсказать её будущие тенденции. Данные, полученные в результате моделирования, окажутся важными для определения стратегий управления.

Цель работы

Сделать прогноз распределения долей компаний на 2022 год, основываясь на предоставленных компаниями данных за 2011-2021 года и сравнить полученные результаты с официальными данными за 2022 год.

Формулировка задачи

В данной работе используется модель Лотки-Вольтера, которая рассматривается с точки зрения экономики, где компании рассматриваются как отдельные виды, а система описывает их взаимодействие между собой:

$$\begin{cases} \frac{dx}{dt} = x(a_0 + a_1 \cdot x + a_2 \cdot y + a_3 \cdot z) \\ \frac{dy}{dt} = y(b_0 + b_1 \cdot x + b_2 \cdot x + b_3 \cdot z) \\ \frac{dz}{dt} = z(c_0 + c_1 \cdot x + c_2 \cdot x + c_3 \cdot z) \end{cases}, \quad (1)$$

где x, y, z – значения долей компаний на рынке; a_i, b_i, c_i – коэффициенты взаимодействия компаний ($i = 0 \ldots 3$).

Для нахождения коэффициентов взаимодействия используются два метода: интегральный и логарифмический.

Решение системы дифференциальных уравнений будет сделано с использованием численного метода Рунге-Кутта четвёртого порядка и метода Адамса. С их помощью можно проверить точность полученных коэффициентов взаимодействия компаний.

Решив систему дифференциальных уравнений, можно провести анализ эффективности решения, сравнив полученные данные с фактическими значениями, представленными компаниями в официальных источниках.

Описание переменных

Переменные, используемые в данной работе представлены в таблице 1.

Таблица 1

Описание переменных

Название переменной или функции	Описание
a_i	Коэффициент взаимодействия компаний
b_i	Коэффициент взаимодействия компаний
c_i	Коэффициент взаимодействия компаний
x	Первая компания
y	Вторая компания
z	Третья компания
n	Количество лет для исследования
m	Количество компаний для исследования
d[]	Вектор изменения долей за промежуток времени, в нашем случае за один год
A[][]	Матрица с соотношением компаний на рынке по долям
res[][]	Матрица результата для вычисления численных методов решения системы дифференциальных уравнений
X[][],Xt[][], XtmultXinverse[][]	Вспомогательные матрицы для расчётов коэффициентов системы

k1x, k1y, k1z, k2x, k2y, k2z, k3x, k3y, k3z, k4x, k4y, k4z	Промежуточные вычисления в методе Рунге-Кутта

Данные исследования

Исследование конкурентной борьбы основано на трёх компаниях-производителей напитков Нижегородской области. В данной работе были выбраны следующие компании: x - ООО "Акваника" [3], y - ООО "РОСМ" [4] и z – ООО "Городецкие источники" [5] Нижегородской области. Сведения о выручках компаний с 2011 по 2022 год были взяты с официальной отчётности компаний, которая находится в открытом доступе и представлены в таблице 1.

Таблица 2

Список производителей напитков по выручке

	x (руб.)	y (руб.)	z (руб.)
2011	100 000 000	65 100 000	24 800 000
2012	302 000 000	110 000 000	80 000 000
2013	625 000 000	136 200 000	122 100 000
2014	694 000 000	114 100 000	98 000 000
2015	600 000 000	71 400 000	42 800 000
2016	700 000 000	63 280 000	35 900 000
2017	700 000 000	93 800 000	40 800 000
2018	1 000 000 000	89 500 000	44 700 000
2019	1 000 000 000	68 400 000	39 700 000
2020	1 100 000 000	82 000 000	35 900 000
2021	1 700 000 000	120 000 000	44 800 000
2022	2 300 000 000	160 100 000	42 500 000

Для того, чтобы понять, какую часть рынка занимают данные компании, необходимо вычислить доли этих в отрасли производства напитков.

Доля компании на рынке рассчитывается по формуле: $Д_р = \frac{TR_n}{TR_{общ}}$,

где $Д_р$ - доля компании на рынке, TR_n-выручка анализируемой компании за год, $TR_{общ}$-общая сумма выручки трёх компаний за год. Доли для рассматриваемых компаний на рынке представлены в таблице 2.

Таблица 3

Список производителей напитков по долям на рынке

	x	y	z
2011	0.527	0.343	0.131
2012	0.614	0.224	0.162
2013	0.711	0.155	0.139
2014	0.778	0.128	0.110
2015	0.857	0.102	0.061
2016	0.885	0.080	0.045
2017	0.839	0.112	0.049
2018	0.890	0.080	0.040
2019	0.902	0.062	0.036

2020	0.905	0.067	0.029
2021	0.911	0.064	0.024
2022	0.919	0.064	0.017

Блок-схемы алгоритмов

Методы расчёта коэффициентов взаимодействия

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

Интегральный метод

Для нахождения коэффициентов взаимодействия создаётся система:

$$\begin{pmatrix} d_{1,0} \\ d_{2,1} \\ \cdots \\ d_{n,n-1} \end{pmatrix} = \begin{pmatrix} \overline{x_{1,0}} & \overline{x^2{}_{1,0}} & \overline{xy_{1,0}} & \overline{xz_{1,0}} \\ \overline{x_{2,1}} & \overline{x^2{}_{2,1}} & \overline{xy_{2,1}} & \overline{xz_{2,1}} \\ \cdots & \cdots & \cdots & \cdots \\ \overline{x_{n,n-1}} & \overline{x^2{}_{n,n-1}} & \overline{xy_{n,n-1}} & \overline{xz_{n,n-1}} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix},$$

или в виде уравнения $d = X \cdot a$, где $d_{j+1,j} = x(t_{j+1}) - x(t_j)$, $j = 0,1,\dots,(n-1)$, матрица X считается следующим образом:

$$\overline{x_{J+1,J}} = \frac{\mathrm{x_{i+1}} + \mathrm{x_i}}{2}$$

$$\overline{x^2{}_{J+1,J}} = \frac{(\mathrm{x_{i+1}})^2 + (\mathrm{x_i})^2}{2}$$

$$\overline{xy_{J+1,J}} = \frac{(\mathrm{x_{i+1}}) \cdot (\mathrm{y_{i+1}}) + (\mathrm{x_i}) \cdot (\mathrm{y_i})}{2}$$

$$\overline{xz_{J+1,J}} = \frac{(\mathrm{x_{i+1}}) \cdot (\mathrm{z_{i+1}}) + (\mathrm{x_i}) \cdot (\mathrm{z_i})}{2}$$

$$\mathrm{i} = \overline{0, \mathrm{n} - 2}$$

Отсюда выражаются коэффициенты взаимодействия такие, что $(\mathrm{X^T X})^{-1} \mathrm{X^T d} = \mathrm{a}$.

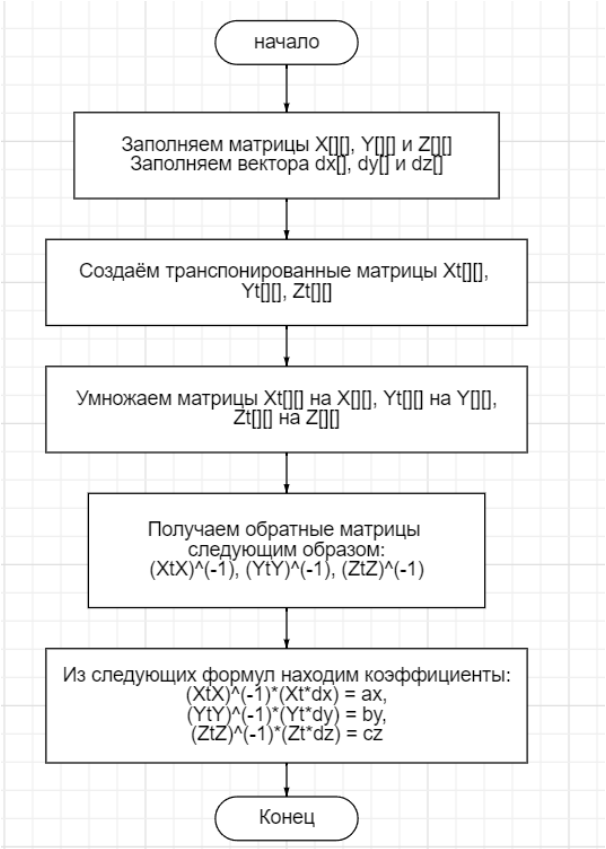

Рисунок 1.Блок-схема для интегрального метода нахождения коэффициентов взаимодействия

Логарифмический метод

Для нахождения коэффициентов взаимодействия создаётся система:

$$\begin{pmatrix} l_{1,0} \\ l_{2,1} \\ \dots \\ l_{n,n-1} \end{pmatrix} = \begin{pmatrix} 1 & \overline{x_{1,0}} & \overline{y_{1,0}} & \overline{z_{1,0}} \\ 1 & \overline{x_{2,1}} & \overline{y_{2,1}} & \overline{z_{2,1}} \\ \dots & \dots & \dots & \dots \\ 1 & \overline{x_{n,n-1}} & \overline{y_{n,n-1}} & \overline{z_{n,n-1}} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix},$$

или в виде уравнения $l = X \cdot a$, где $l_{j+1,j} = lnx(t_{j+1}) - lnx(t_j)$, $j = 0,1,\dots,(n-1)$, матрица X считается следующим образом:

$$\overline{x_{J+1,J}} = \frac{(\mathrm{x_{i+1}}) + (\mathrm{x_i})}{2}$$

$$\overline{y_{J+1,J}} = \frac{(\mathrm{y_{i+1}}) + (\mathrm{y_i})}{2}$$

$$\overline{z_{J+1,J}} = \frac{(\mathrm{z_{i+1}}) + (\mathrm{z_i})}{2}$$

$$\mathrm{i} = \overline{0, \mathrm{n} - 2}$$

Отсюда выражаются коэффициенты взаимодействия такие, что $(\mathrm{X^T X})^{-1}\mathrm{X^T} l = \mathrm{a}$.

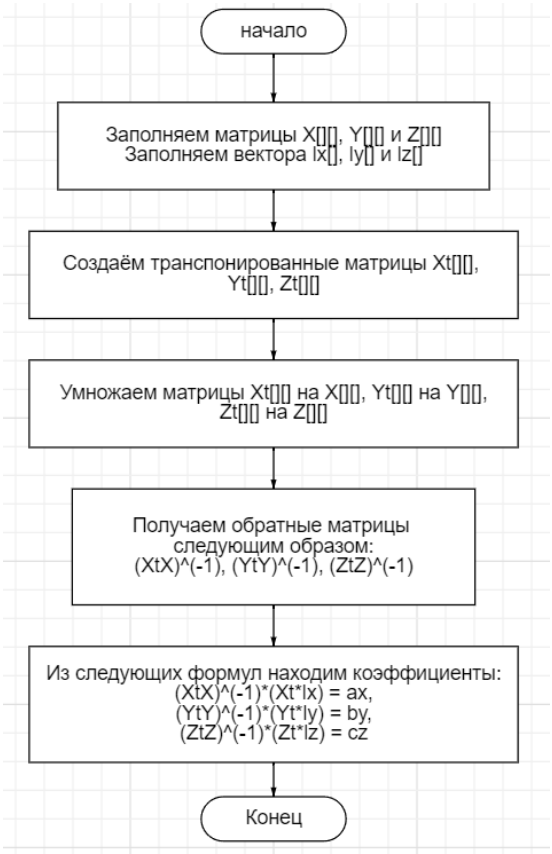

Рисунок 2.Блок-схема для логарифмического метода нахождения коэффициентов взаимодействия

Описание системы дифференциальных уравнений

Модель Лотки-Вольтерра, описывающая взаимодействие двух видов «хищник-жертва» в общем виде выглядит следующим образом:

$$\frac{dX_i}{dt} = X_i \cdot (a_{i,0} + \sum a_{i,j} X_i), \quad i,j = 1,2,\dots,n,$$

Где n-количество взаимодействующих видов, $\frac{dX_i}{dt}$ – описывает быстроту изменения численности каждого из видов, $a_{i,0}$ – описывает рост популяции или его сокращение (зависит от знака коэффициента), $a_{i,j}$- отражает взаимодействие между видами с точки зрения хищничества, если коэффициент отрицательный, взаимовыручки, если коэффициент положительной или отсутствия взаимодействия, если коэффициент равен нулю.

В данной работе рассматривается конкуренция трёх компаний за ресурсы и модель Лотки-Вольтерра в этом случае имеет вид системы дифференциальных уравнений (1). В ней x, y, z – значения долей компаний на рынке, a_i, b_i, c_i – коэффициенты конкуренции компаний ($i = 0 \dots 3$).

Если рассмотреть схему взаимодействия компаний, то можно выделить следующие ситуации: x^2 - деятельность компании x при отсутствии взаимодействий с другими компаниями, x·y- взаимодействие компании x с компанией y, x·z- взаимодействие компании x с компанией z.

Также можно выделить следующие случаи взаимодействия компаний: если коэффициент взаимодействия получился со знаком «+», то можно

сказать, что компании находятся во взаимовыгодных отношениях, если коэффициент получился со знаком «-», то можно судить о том, что одна компания доминирует на рынке над другой, если коэффициент взаимодействия равен нулю, тогда считается, что конкуренция не происходит.

При использовании интегрального метода [7] были получены следующие значения коэффициентов: a_0=-0.697, a_1=0.659, a_2=0.850, a_3=1.558, b_0=5.111, b_1=-4.995, b_2=-6.136, b_3=-6.316, c_0=18.486, c_1=-18.679, c_2=-15.530, c_3=-21.849.

При использовании логарифмического метода [7] были получены следующие коэффициенты: a_0=-0.706, a_1=0.665, a_2=0.884, a_3=1.552, b_0=7.404, b_1=-7.292, b_2=-8.844, b_3=-7.916, c_0=13.439, c_1=-13.738, c_2=-10.539, c_3=-16.333.

Решение системы дифференциальных уравнений

Решение представлено в приложении 1.

Графики решения, полученные методом Рунге-Кутта в программе Mathcad 15:

$res := rkfixed(v, 0, 11, 12, Dint)$ (Рунге – Кутта)

$k := 2011..2022$

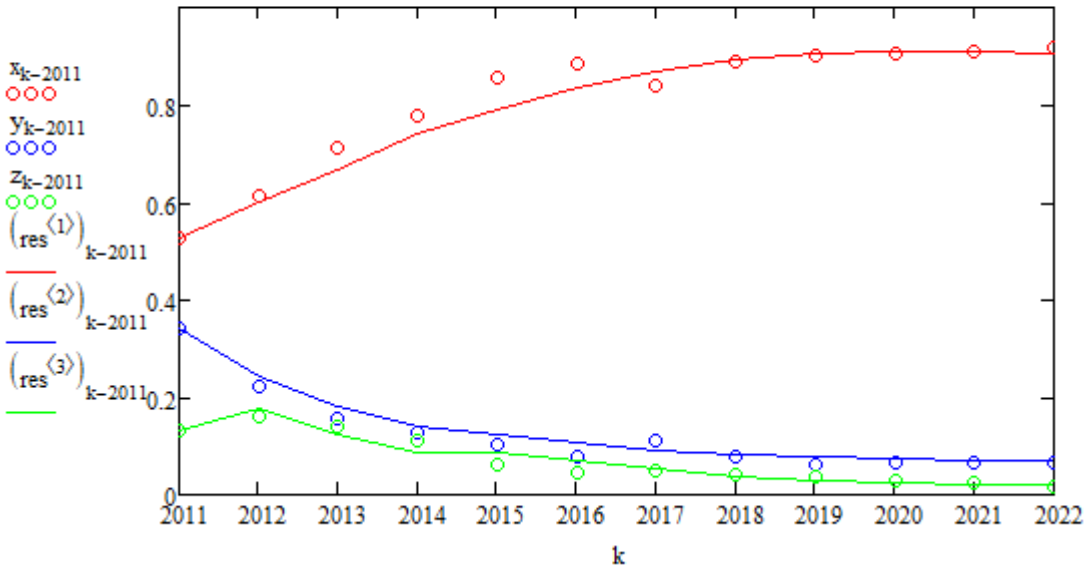

Рисунок 3. Решение системы методом Рунге-Кутта с интегральным методом нахождения коэффициентов

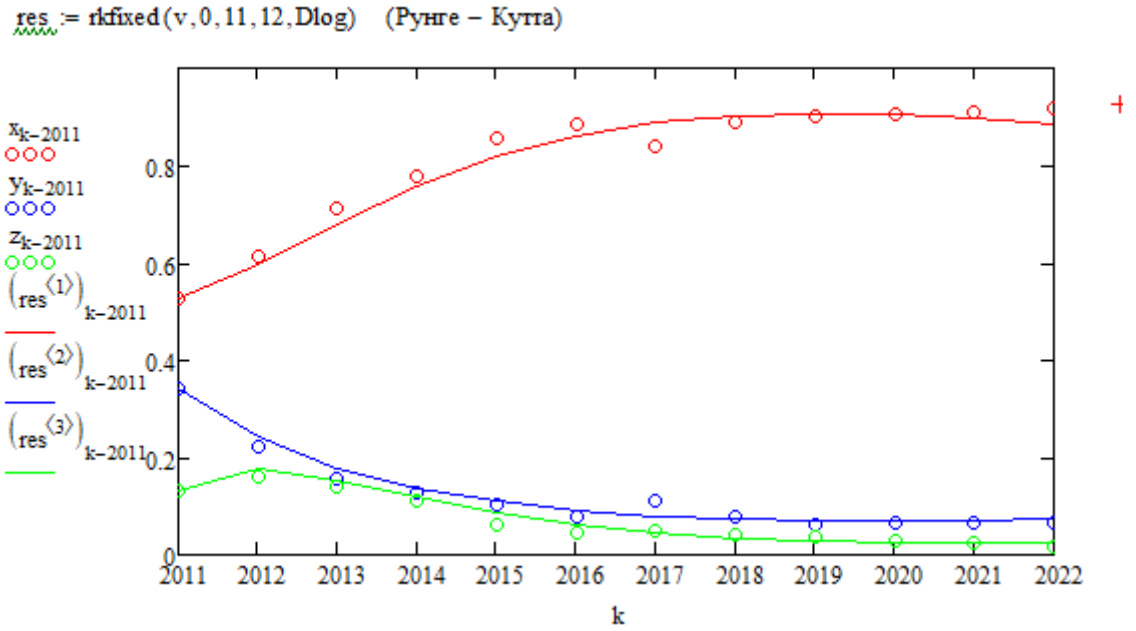

Рисунок 4. Решение системы методом Рунге-Кутта с логарифмическим методом нахождения коэффициентов

Решение представлено в приложении 2.

Графики решения, полученные методом Адамса в программе Mathcad 15:

Точками на графиках представлены данные из таблицы 2, линиями прогнозируемые значения решения системы.

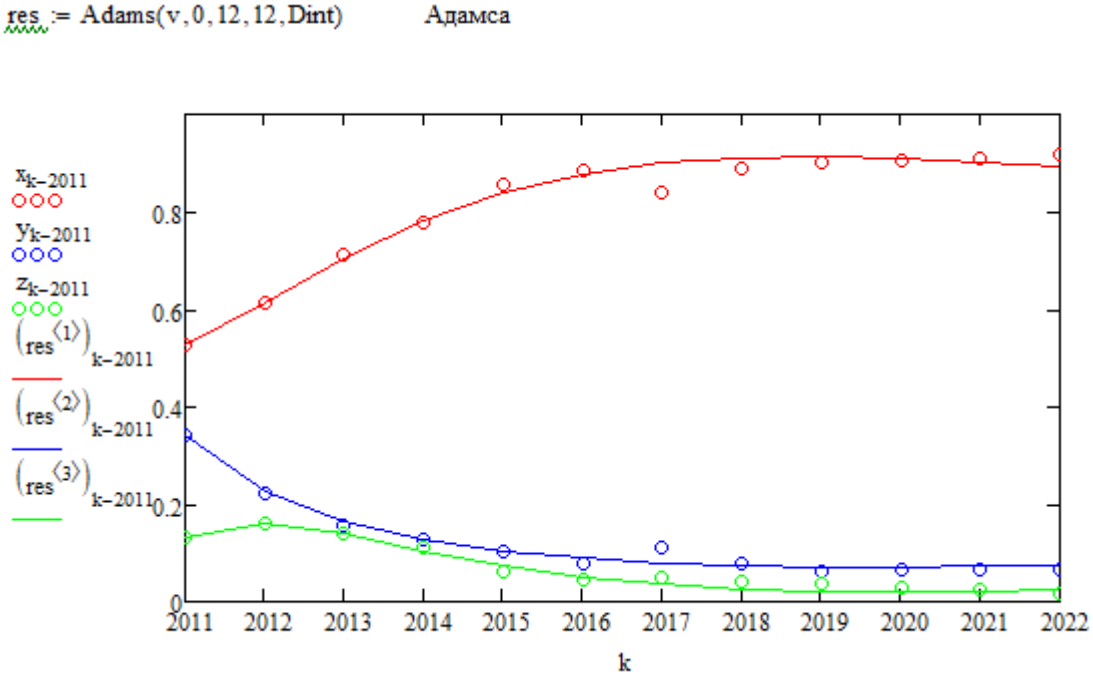

Рисунок 5. Решение системы методом Адамса с интегральными методом нахождения коэффициентов

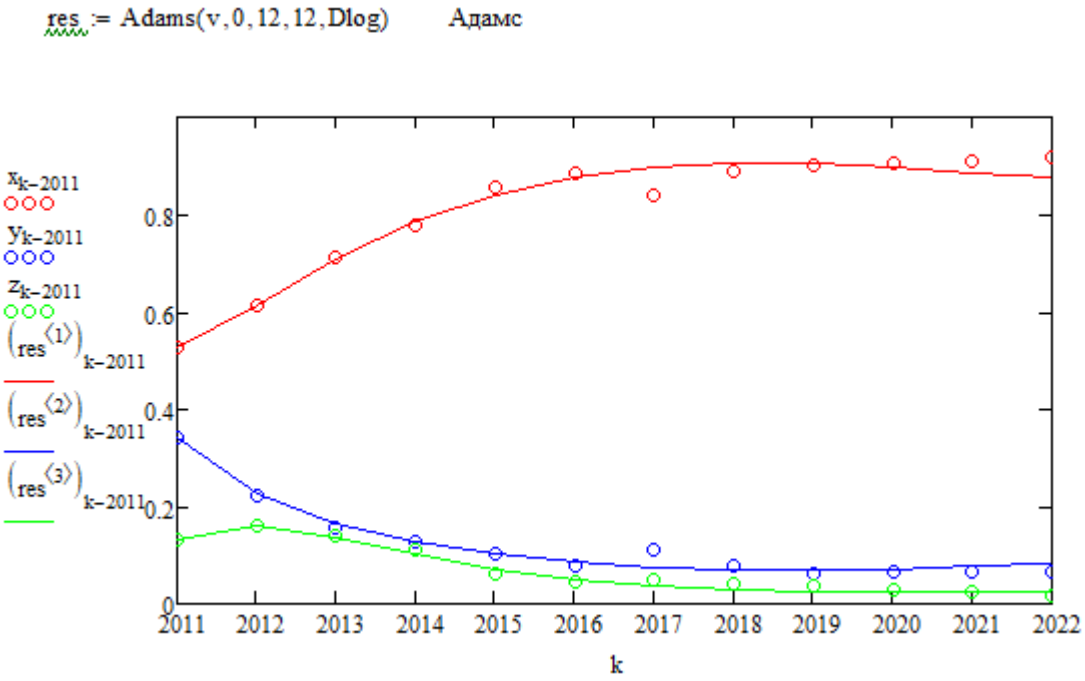

Рисунок 6. Решение системы методом Адамса с логарифмическим методом нахождения коэффициентов

Решения, полученные методом Рунге-Кутта на языке программирования Java в программе Apache NetBeans IDE 12.4, версия Java16.0.1:

Точками на графиках представлены данные из таблицы 2, линиями прогнозируемые значения решения системы.

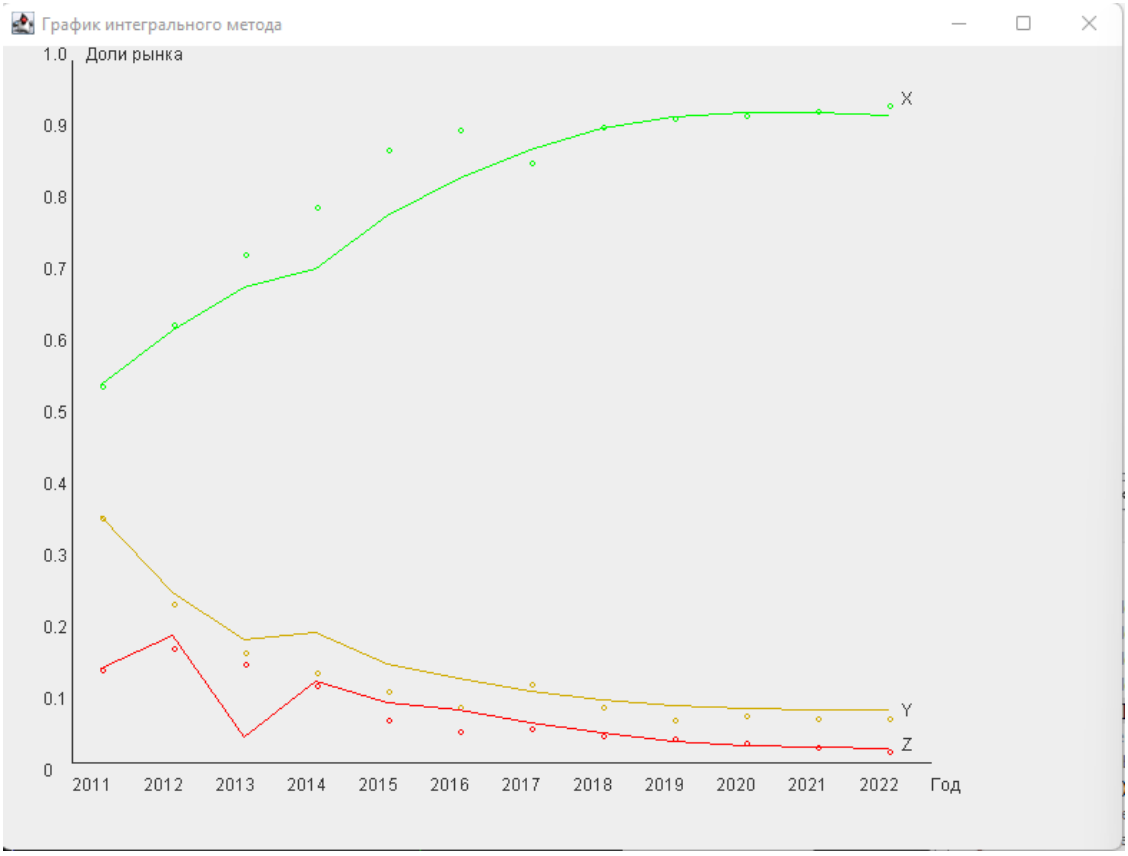

Рисунок 7. Решение системы методом Рунге-Кутта с интегральным методом нахождения коэффициентов

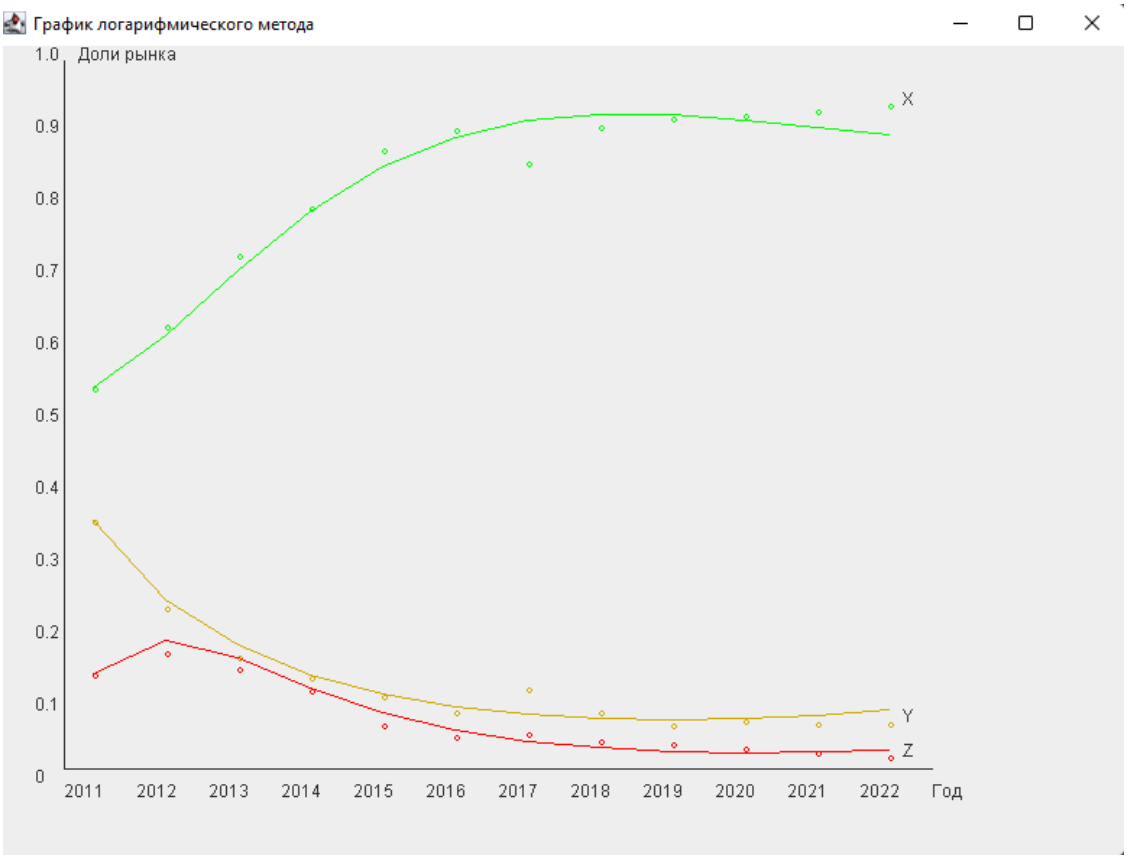

Рисунок 8. Решение системы методом Рунге-Кутта с логарифмическим методом нахождения коэффициентов

Код программы представлен в приложении 3.

Код для отображения графика с интегральным методом поиска коэффициентов представлен в приложении 4, для графика с логарифмическим методом поиска коэффициентов в приложении 5.

Результат решения интегральным методом представлен в таблице 4.

Таблица 4

Значения, полученные методом Рунге-Кутта для интегрального метода

h	x	y	z
2011	0.527	0.343	0.131
2012	0.604	0.239	0.178
2013	0.664	0.173	0.037
2014	0.691	0.182	0.114
2015	0.764	0.139	0.085
2016	0.815	0.118	0.074
2017	0.856	0.101	0.056
2018	0.885	0.088	0.041
2019	0.902	0.080	0.030
2020	0.909	0.075	0.024
2021	0.909	0.073	0.021
2022	0.904	0.073	0.020

Таблица 5

Значения, полученные методом Рунге-Кутта для логарифмического метода

h	x	y	z
2011	0.527	0.343	0.131
2012	0.601	0.235	0.178
2013	0.689	0.171	0.153
2014	0.771	0.130	0.112
2015	0.833	0.103	0.078
2016	0.874	0.086	0.054
2017	0.897	0.075	0.038
2018	0.907	0.069	0.029
2019	0.906	0.067	0.025
2020	0.899	0.070	0.023
2021	0.888	0.075	0.024
2022	0.878	0.081	0.026

Анализ методов

Чтобы сделать вывод, какой метод нахождения более точный, необходимо посчитать абсолютные погрешности результатов, сравнив их с данными из таблицы 2.

Таблица 6

Погрешность интегрального метода в сравнении с реальными значениями

Год	x	y	z
2011	0	0	0
2012	0.010	0.015	0.014
2013	0.047	0.018	0.102
2014	0.087	0.054	0.004
2015	0.093	0.037	0.024
2016	0.070	0.038	0.029
2017	0.017	0.011	0.007
2018	0.005	0.008	0.001
2019	0	0.018	0.006
2020	0.004	0.008	0.005
2021	0.002	0.009	0.003
2022	0.015	0.009	0.003

Таблица 7

Погрешность логарифмического метода в сравнении с реальными значениями

Год	x	y	z
2011	0	0	0
2012	0.013	0.011	0.016
2013	0.022	0.016	0.014
2014	0.007	0.002	0.002
2015	0.024	0.001	0.017
2016	0.011	0.006	0.009
2017	0.058	0.035	0.011
2018	0.017	0.011	0.011
2019	0.004	0.005	0.011

2020	0.006	0.003	0.006
2021	0.023	0.011	0
2022	0.041	0.017	0.009

Исходя из полученных значений за 2022 год абсолютных погрешностей можно сделать вывод, что при прогнозировании интегральный метод даёт более точный прогноз на следующий год.

Заключение

Подводя итог проведённой работы, можно сделать вывод, что использование модели Лотки-Вольтерра способно достаточно точно спрогнозировать развитие рынка на ближайший год. При подсчёте коэффициентов для вычисления прогнозов были использованы официальные данные о выручке компаний с 2011 по 2021 годы. Результаты исследования показали, что максимально приближённые значения в сравнении с фактическими данными за 2022 год выдаёт интегральный метод подсчёта коэффициентов.

Список литературы

1. Катаева, Л. Ю. Влияние индикаторов на прогнозируемость экономической безопасности региона / Л. Ю. Катаева, Д. А. Масленников, Т. А. Федосеева // Фундаментальные исследования. – 2019. – № 12-1. – С. 72-76. – DOI 10.17513/fr.42624.
2. Иконников, В. В. Экономические факторы и управленческие решения: влияние на прогнозируемость региональных индикаторов / В. В. Иконников, Л. Ю. Катаева, Д. А. Масленников // Russian Economic Bulletin. – 2023. – Т. 6, № 6. – С. 380-385.
3. Сайт "СБИС" (Система Быстрого Информирования о Ситуациях). Сведения о юридическом лице [Электронный ресурс]. URL: https://sbis.ru/contragents/5251112051/525101001(дата обращения: 11.11.2023).
4. Сайт "СБИС" (Система Быстрого Информирования о Ситуациях). Сведения о юридическом лице [Электронный ресурс]. URL: https://sbis.ru/contragents/5263028301/525701001(дата обращения: 11.11.2023).
5. Сайт "СБИС" (Система Быстрого Информирования о Ситуациях). Сведения о юридическом лице [Электронный ресурс]. URL: https://sbis.ru/contragents/5248014294/524801001(дата обращения: 11.11.2023).
6. Л. Ю. Катаева, М. Н. Ильичева, Т. А. Федосеева, Д. А. Масленников Численное решение задач экономики с использованием EXCEL, C++ и MATLAB: учебное пособие // Нижний Новгород: Нижегородский государственный технический университет им. Р.Е. Алексеева, 2020. – 229 с. – ISBN 978-5-502-01281-2. – EDN BMIJQQ.
7. P. H. Kloppers, Johanna C. Greeff Lotka–Volterra model parameter estimation using experiential data // Applied Mathematics and Computation 2013. Vol. 224. P. 818-821. DOI:10.1016/j.amc.2013.08.093.

Приложение 1

$$d(x) := \left| \begin{array}{l} \text{for } i \in 0..n-2 \\ \quad d_i \leftarrow x_{i+1} - x_i \\ d \end{array} \right.$$

Интегральный метод подсчёта коэффициентов

$$intcoefX(x) := \left| \begin{array}{l} \text{for } i \in 0..n-2 \\ \quad X_{i,0} \leftarrow \frac{(x_{i+1} + x_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,1} \leftarrow \frac{\left[(x_{i+1})^2 + (x_i)^2\right]}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,2} \leftarrow \frac{\left[x_{i+1} \cdot y_{i+1} + (x_i \cdot y_i)\right]}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,3} \leftarrow \frac{\left[x_{i+1} \cdot z_{i+1} + (x_i \cdot z_i)\right]}{2} \\ X \end{array} \right.$$

$$intcoefY(y) := \left| \begin{array}{l} \text{for } i \in 0..n-2 \\ \quad Y_{i,0} \leftarrow \frac{(y_{i+1} + y_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad Y_{i,2} \leftarrow \frac{\left[(y_{i+1})^2 + (y_i)^2\right]}{2} \\ \text{for } i \in 0..n-2 \\ \quad Y_{i,1} \leftarrow \frac{\left[x_{i+1} \cdot y_{i+1} + (x_i \cdot y_i)\right]}{2} \\ \text{for } i \in 0..n-2 \\ \quad Y_{i,3} \leftarrow \frac{\left[y_{i+1} \cdot z_{i+1} + (y_i \cdot z_i)\right]}{2} \\ Y \end{array} \right.$$

$$intcoefZ(z) := \left| \begin{array}{l} \text{for } i \in 0..n-2 \\ \quad Z_{i,0} \leftarrow \frac{(z_{i+1} + z_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad Z_{i,3} \leftarrow \frac{\left[(z_{i+1})^2 + (z_i)^2\right]}{2} \\ \text{for } i \in 0..n-2 \\ \quad Z_{i,1} \leftarrow \frac{\left[x_{i+1} \cdot z_{i+1} + (x_i \cdot z_i)\right]}{2} \\ \text{for } i \in 0..n-2 \\ \quad Z_{i,2} \leftarrow \frac{\left[y_{i+1} \cdot z_{i+1} + (y_i \cdot z_i)\right]}{2} \\ Z \end{array} \right.$$

Составление матриц

$X := intcoefX(x)$ $dx := d(x)$ $Y := intcoefY(y)$ $dy := d(y)$ $Z := intcoefZ(z)$ $dz := d(z)$

Коэффициенты конкуренции

$$a := \left(X^T \cdot X\right)^{-1} \cdot X^T \cdot dx = \begin{pmatrix} -0.697 \\ 0.659 \\ 0.85 \\ 1.558 \end{pmatrix}$$

$$b := \left(Y^T \cdot Y\right)^{-1} \cdot Y^T \cdot dy = \begin{pmatrix} 5.111 \\ -4.995 \\ -6.163 \\ -6.316 \end{pmatrix}$$

$$c := \left(Z^T \cdot Z\right)^{-1} \cdot Z^T \cdot dz = \begin{pmatrix} 18.486 \\ -18.679 \\ -15.53 \\ -21.849 \end{pmatrix}$$

Приложение 2

$$lgd(x) := \left| \begin{array}{l} \text{for } i \in 0..n-2 \\ \quad lx_i \leftarrow \ln(x_{i+1}) - \ln(x_i) \\ lx \end{array} \right.$$

Логарифмический метод нахождения коэффициентов

$$lgcoefX(x) := \left| \begin{array}{l} \text{for } i \in 0..n-2 \\ \quad X_{i,0} \leftarrow 1 \\ \text{for } i \in 0..n-2 \\ \quad X_{i,1} \leftarrow \frac{(x_{i+1} + x_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,2} \leftarrow \frac{(y_{i+1} + y_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,3} \leftarrow \frac{(z_{i+1} + z_i)}{2} \\ X \end{array} \right.$$

$$lgcoefY(y) := \left| \begin{array}{l} \text{for } i \in 0..n-2 \\ \quad X_{i,0} \leftarrow 1 \\ \text{for } i \in 0..n-2 \\ \quad X_{i,1} \leftarrow \frac{(x_{i+1} + x_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,2} \leftarrow \frac{(y_{i+1} + y_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,3} \leftarrow \frac{(z_{i+1} + z_i)}{2} \\ X \end{array} \right.$$

$$lgcoefZ(z) := \left| \begin{array}{l} \text{for } i \in 0..n-2 \\ \quad X_{i,0} \leftarrow 1 \\ \text{for } i \in 0..n-2 \\ \quad X_{i,1} \leftarrow \frac{(x_{i+1} + x_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,2} \leftarrow \frac{(y_{i+1} + y_i)}{2} \\ \text{for } i \in 0..n-2 \\ \quad X_{i,3} \leftarrow \frac{(z_{i+1} + z_i)}{2} \\ X \end{array} \right.$$

Составление матриц

$X := lgcoefX(x)$ $dx := lgd(x)$ $Y := lgcoefY(y)$ $dy := lgd(y)$ $Z := lgcoefZ(z)$ $dz := lgd(z)$

Коэффициенты взаимодействия

$$a := \left(X^T \cdot X\right)^{-1} \cdot X^T \cdot dx = \begin{pmatrix} -0.706 \\ 0.665 \\ 0.884 \\ 1.552 \end{pmatrix}$$

$$b := \left(Y^T \cdot Y\right)^{-1} \cdot Y^T \cdot dy = \begin{pmatrix} 7.404 \\ -7.292 \\ -8.844 \\ -7.916 \end{pmatrix}$$

$$c := \left(Z^T \cdot Z\right)^{-1} \cdot Z^T \cdot dz = \begin{pmatrix} 13.439 \\ -13.738 \\ -10.539 \\ -16.333 \end{pmatrix}$$

Приложение 3

Код программы на языке программирования Java

Главный класс с решением системы дифференциальных уравнений методом Рунге-Кутта

```
package JavaApplication1;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import javaapplication1.Graph1;
import javaapplication1.Graph2;

public class NewClass {

   public static void matMultMat(double[][] matrix1, double[][] matrix2, double[][] result, int cols1, int rows1, int cols2) {
      for (int i = 0; i < rows1; ++i) {
         for (int j = 0; j < cols2; ++j) {
            result[i][j] = 0.0;
            for (int k = 0; k < cols1; ++k) {
               result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
         }
      }
   }

   public static void matMultVec(double[][] matrix1, double[] vector1, double[] result, int cols1, int rows1) {
      for (int i = 0; i < rows1; ++i) {
         result[i] = 0.0;
         for (int k = 0; k < cols1; ++k) {
            result[i] += matrix1[i][k] * vector1[k];
         }
      }
   }

   public static double determinant(double[][] matrix, int size) {
      if (size == 1) {
         return matrix[0][0];
      } else if (size == 2) {
         return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
      } else {
         double det = 0.0;
```

```
        for (int j = 0; j < size; ++j) {
            double[][] temp = new double[size - 1][size - 1];
            for (int k = 1; k < size; ++k) {
                int tempcol = 0;
                for (int l = 0; l < size; ++l) {
                    if (l != j) {
                        temp[k - 1][tempcol] = matrix[k][l];
                        ++tempcol;
                    }
                }
            }
            double sign = (j % 2 == 0) ? 1.0 : -1.0;
            double tempd = determinant(temp, size - 1);
            det += sign * matrix[0][j] * tempd;
        }
        return det;
    }
}

public static void inverseMatrix(double[][] matrix, double[][] result, int size) {
    double det = determinant(matrix, size);
    double[][] minor = new double[size][size];

    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            double[][] tempMinor = new double[size - 1][size - 1];
            int row = 0;
            for (int k = 0; k < size; ++k) {
                if (k != i) {
                    int col = 0;
                    for (int l = 0; l < size; ++l) {
                        if (l != j) {
                            tempMinor[row][col] = matrix[k][l];
                            ++col;
                        }
                    }
                    ++row;
                }
            }
            double sign = ((i + j) % 2 == 0) ? 1.0 : -1.0;
            double minorDet = determinant(tempMinor, size - 1);
            double cofactor = sign * minorDet;
            result[j][i] = cofactor / det;
        }
```

```
        }
    }



 public static double f1(double x, double y, double z,double[][] a) {
        return x*(a[0][0] + a[0][1]*x+a[0][2]*y+a[0][3]*z);
    }

    public static double f2(double x, double y, double z,double[][] a) {
        return y*(a[1][0] + a[1][1]*x+a[1][2]*y+a[1][3]*z);
    }

    public static double f3(double x, double y, double z,double[][] a) {
        return z*(a[2][0] + a[2][1]*x+a[2][2]*y+a[2][3]*z);
    }

public static void runge(double[][] res, double[][] a, int m, int maincol) {
        double h = 1; // шаг интегрирования
        double x0 = 0.527; // начальное значение x
        double y0 = 0.343; // начальное значение y
        double z0 = 0.131; // начальное значение z

        double x=x0;
        double y=y0;
        double z=z0;
        int n = 11; // количество итераций

    for (int i = 0; i < n; i++) {
            double k1x = h * f1(x, y, z,a);
            double k1y = h * f2(x, y, z,a);
            double k1z = h * f3(x, y, z,a);

            double k2x = h * f1(x + k1x / 2, y + k1y / 2, z + k1z / 2,a);
            double k2y = h * f2(x + k1x / 2, y + k1y / 2, z + k1z / 2,a);
            double k2z = h * f3(x + k1x / 2, y + k1y / 2, z + k1z / 2,a);

            double k3x = h * f1(x + k2x / 2, y + k2y / 2, z + k2z / 2,a);
            double k3y = h * f2(x + k2x / 2, y + k2y / 2, z + k2z / 2,a);
            double k3z = h * f3(x + k2x / 2, y + k2y / 2, z + k2z / 2,a);

            double k4x = h * f1(x + k3x, y + k3y, z + k3z,a);
            double k4y = h * f2(x + k3x, y + k3y, z + k3z,a);
            double k4z = h * f3(x + k3x, y + k3y, z + k3z,a);
```

```
            x +=(k1x + 2 * k2x + 2 * k3x + k4x) / 6;
            y +=(k1y + 2 * k2y + 2 * k3y + k4y) / 6;
            z +=(k1z + 2 * k2z + 2 * k3z + k4z) / 6;

            res[0][i + 1] = x;
            res[1][i + 1] = y;
            res[2][i + 1] = z;
        }
        // Вывод матрицы res
        System.out.println("Результат по Рунге:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n+1; j++) {
                System.out.print(res[i][j] + " ");
            }
            System.out.println();
        }
}


    public static void findCoefs(double[][] matrix1, double[][] d, double[][] result, int
maincol, int n, int m, int metnum) {
        n -= 1;
        int l = m + 1;
        double[][] X = new double[n][l];
        double[][] Xt = new double[l][n];
        if (metnum == 1) {
            for (int i = 0; i < n; i++) {
                X[i][0] = (matrix1[i + 1][maincol] + matrix1[i][maincol]) / 2;
                for (int j = 1; j < l; j++) {
                    X[i][j] = (matrix1[i + 1][maincol] * matrix1[i + 1][j - 1] +
matrix1[i][maincol] * matrix1[i][j - 1]) / 2;
                }
            }
        } else {
            for (int i = 0; i < n; i++) {
                X[i][0] = 1;
                for (int j = 1; j < l; j++) {
                    X[i][j] = (matrix1[i + 1][j - 1] + matrix1[i][j - 1]) / 2;
                }
            }
        }

        for (int i = 0; i < l; i++) {
```

```
            for (int j = 0; j < n; j++) {
                Xt[i][j] = X[j][i];
            }
        }

        double[] dx = new double[n];
        if (metnum == 1) {
            for (int i = 0; i < n; i++) {
                dx[i] = d[i][maincol];
            }
        } else {
            for (int i = 0; i < n; i++) {
                dx[i] = Math.log(matrix1[i + 1][maincol]) -
Math.log(matrix1[i][maincol]);
            }
        }

        double[][] XtmultX = new double[l][l];
        double[][] XtmultXinverse = new double[l][l];

        matMultMat(Xt, X, XtmultX, n, l, l);
        inverseMatrix(XtmultX, XtmultXinverse, l);

        double[] Xtd = new double[l];
        matMultVec(Xt, dx, Xtd, n, l);

        double[] ax = new double[l];
        matMultVec(XtmultXinverse, Xtd, ax, l, l);

        for (int i = 0; i < l; i++) {
            result[maincol][i] = ax[i];
        }
        for (int i = 0; i < l; i++) {
            System.out.println("a"+i+":"+ax[i]);
        }
        System.out.println("");
    }

    public static void fprojection(double[][] matrix1, double[][] a, double[] result, int
mainrow, int n, int m) {
        for (int i = 0; i < m; i++) {
            result[i] = matrix1[mainrow][i] + matrix1[mainrow][i] * (a[i][0] + a[i][1] *
matrix1[mainrow][0] + a[i][2] * matrix1[mainrow][1] + a[i][3] *
matrix1[mainrow][2]);
```

```
    }
}

public static void main(String[] args) {
    int n = 12, m = 3;
    int l = m + 1;

    // Инициализация матрицы A
    double[][] A = new double[n][m];

//Доли компаний
// Акваника            РОСМ          Городецкие источники
A[0][0] = 0.527;    A[0][1] = 0.343;    A[0][2] = 0.131;
A[1][0] = 0.614;    A[1][1] = 0.224;    A[1][2] = 0.162;
A[2][0] = 0.711;    A[2][1] = 0.155;    A[2][2] = 0.139;
A[3][0] = 0.778;    A[3][1] = 0.128;    A[3][2] = 0.110;
A[4][0] = 0.857;    A[4][1] = 0.102;    A[4][2] = 0.061;
A[5][0] = 0.885;    A[5][1] = 0.080;    A[5][2] = 0.045;
A[6][0] = 0.839;    A[6][1] = 0.112;    A[6][2] = 0.049;
A[7][0] = 0.890;    A[7][1] = 0.080;    A[7][2] = 0.040;
A[8][0] = 0.902;    A[8][1] = 0.062;    A[8][2] = 0.036;
A[9][0] = 0.905;    A[9][1] = 0.067;    A[9][2] = 0.029;
A[10][0] = 0.911;   A[10][1] = 0.064;   A[10][2] = 0.024;
A[11][0] = 0.919;   A[11][1] = 0.064;   A[11][2] = 0.017;

    double[][] d = new double[n - 1][m];
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < m; j++) {
            d[i][j] = A[i + 1][j] - A[i][j];
        }
    }

    double[][] a = new double[m][l];
    for (int i = 0; i < m; i++) {
        findCoefs(A, d, a, i, n, m, 1);
    }
    //Подсчёт интегральным методом
    double[] intmet = new double[m];
    //fprojection(A, a, intmet, n - 1, n, m);

    double[][] res = new double[m][n];
    for (int i = 0; i < m; i++) {
        res[i][0] = A[0][i];
    }
```

```
runge(res, a, m, 0);

//Рунге для 1 графика
int[] xvg1 = new int[12];
for (int i=0;i<12;i++){
double doubleValue = res[0][i] *1000;
int value = (int) doubleValue;
value = (1000-value)/2;
System.out.println(value);
xvg1 [i] = value;
}
Graph1.vx1= xvg1;

int[] yvg1 = new int[12];
for (int i=0;i<12;i++){
double doubleValue = res[1][i] *1000;
int value = (int) doubleValue;
value = (1000-value)/2;
yvg1 [i] = value;
}
Graph1.vy1= yvg1;

int[] zvg1 = new int[12];
for (int i=0;i<12;i++){
double doubleValue = res[2][i] *1000;
int value = (int) doubleValue;
value = (1000-value)/2;
zvg1 [i] = value;
}
Graph1.vz1= zvg1;

//Доли для 1 графика
int[] xfg1 = new int[12];
for (int i=0;i<12;i++){
double doubleValue = A[i][0] *1000;
int value = (int) doubleValue;
value = (1000-value)/2;
xfg1 [i] = value;
}
Graph1.x= xfg1;
int[] yfg1 = new int[12];
for (int i=0;i<12;i++){
double doubleValue = A[i][1] *1000;
```

```
int value = (int) doubleValue;
value = (1000-value)/2;
yfg1 [i] = value;
}
Graph1.y= yfg1;
int[] zfg1 = new int[12];
for (int i=0;i<12;i++){
double doubleValue = A[i][2] *1000;
int value = (int) doubleValue;
value = (1000-value)/2;
zfg1 [i] = value;
}
Graph1.z= zfg1;

//Отрисовка 1 графика
Graph1.main(args);

for (int i = 0; i < m; i++) {
   findCoefs(A, d, a, i, n, m, 2);
}
//Подсчёт логарифмическим методом
double[] logmet = new double[m];
runge(res, a, m, 0);

//Рунге для 2 графика
int[] xvg2 = new int[12];
for (int i=0;i<12;i++){
double doubleValue = res[0][i] *1000;
int value = (int) doubleValue;
value = (1000-value)/2;
System.out.println(value);
xvg2 [i] = value;
}
Graph2.vx2= xvg2;
int[] yvg2 = new int[12];
for (int i=0;i<12;i++){
double doubleValue = res[1][i] *1000;
int value = (int) doubleValue;
value = (1000-value)/2;
yvg2 [i] = value;
}
Graph2.vy2= yvg2;
int[] zvg2 = new int[12];
for (int i=0;i<12;i++){
```

```
        double doubleValue = res[2][i] *1000;
        int value = (int) doubleValue;
        value = (1000-value)/2;
        zvg2 [i] = value;
        }
        Graph2.vz2= zvg2;
        //Доли для второго
        int[] xfg2 = new int[12];
        for (int i=0;i<12;i++){
        double doubleValue = A[i][0] *1000;
        int value = (int) doubleValue;
        value = (1000-value)/2;
        xfg2 [i] = value;
        }
        Graph2.x= xfg2;
        int[] yfg2 = new int[12];
        for (int i=0;i<12;i++){
        double doubleValue = A[i][1] *1000;
        int value = (int) doubleValue;
        value = (1000-value)/2;
        yfg2 [i] = value;
        }
        Graph2.y= yfg2;
        int[] zfg2 = new int[12];
        for (int i=0;i<12;i++){
        double doubleValue = A[i][2] *1000;
        int value = (int) doubleValue;
        value = (1000-value)/2;
        zfg2 [i] = value;
        }
        Graph2.z= zfg2;


        //Отрисовка 2 графика
        Graph2.main(args);
    }
}
```

Приложение 4

Класс отображения графика интегральным методом:

```
package javaapplication1;

import javax.swing.*;
import java.awt.*;
import JavaApplication1.NewClass;

class DrawingComponent1 extends JPanel {


    @Override
    protected void paintComponent(Graphics gh) {
    int yearsg[] =  Graph1.years;
    int xg[] =  Graph1.x;
    int yg[] =  Graph1.y;
    int zg[] =  Graph1.z;
    int xvg[] =  Graph1.vx1;
    int yvg[] =  Graph1.vy1;
    int zvg[] =  Graph1.vz1;

    int ng = Graph1.n;
     int x = 50;
     Graphics2D drp = (Graphics2D)gh;
     Color Yellow = new Color (209,171,0);
     Color Red = new Color(255, 0, 0);
     Color Green = new Color(0, 255, 0);
     drp.drawLine(x, 500, 600 + x, 500);
     drp.drawLine(x, 500, x, 10);
     drp.drawString("Год", 600 + x, 520);
     drp.drawString("X", 580 + x, 41);
     drp.drawString("Y", 580 + x, 468);
     drp.drawString("Z", 580 + x, 492);
     drp.drawString("Доли рынка", 10+x, 10);
     int year = 2011;
     drp.drawString("2011", x, 520);
     for (int i = 0; i<11;i++){
     year++;
     x=x+50;
     String yearG =Integer.toString(year);
     drp.drawString(yearG,x, 520);
    }
     int y=10;
```

```
    x=50;
    double share = 1.0;
    drp.drawString("1.0", 80-x , y);
   for (int i = 0; i<9;i++){
    share = Math.round((share - 0.1)*100.0)/100.0;
    y=y+50;
    String shareG = String.valueOf(share);
    drp.drawString(shareG, 80-x , y);
   }
   drp.drawString("0", 80-x , y+50);
    drp.setColor(Green);
    drp.drawPolyline(yearsg, xvg, ng);

    for (int i = 0; i < 12; i++){
      int yearg = yearsg[i];
      int xgg = xg[i];
      drp.drawOval(yearg,xgg,3,3);
    }
    drp.setColor(Yellow);
    drp.drawPolyline(yearsg, yvg , ng);

    for (int i = 0; i < 12; i++){
      int yearg = yearsg[i];
      int ygg = yg[i];
      drp.drawOval(yearg,ygg,3,3);
    }
    drp.setColor(Red);
    drp.drawPolyline(yearsg, zvg , ng);

    for (int i = 0; i < 12; i++){
      int yearg = yearsg[i];
      int zgg = zg[i];
      drp.drawOval(yearg,zgg,3,3);
    }
   }
}

public  class Graph1 extends JFrame{
    public static int years[] = {70,120,170,220,270,320,370,420,470,520,570,620};
    public  static int x[];
    public  static int y[];
    public  static int z[];
    public  static int vx1[];
    public  static int vy1[];
```

```
    public  static int vz1[];

    public static int n = 12;

 public Graph1 () {
        super("График интегрального метода");
        JPanel jcp = new JPanel(new BorderLayout());
        setContentPane(jcp);
        jcp.add(new DrawingComponent1 (), BorderLayout.CENTER);
        jcp.setBackground(Color.gray);
        setSize(800, 600);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    }

          public static void main(String[] args)   {
        new Graph1().setVisible(true);
}
}
```

Приложение 5

Класс отображения графика интегральным методом:

```
package javaapplication1;

import javax.swing.*;
import java.awt.*;
import JavaApplication1.NewClass;

class DrawingComponent2 extends JPanel {
int yearsg[] =  Graph2.years;
int xg[] =  Graph2.x;
int yg[] =  Graph2.y;
int zg[] =  Graph2.z;
int xvg2[] =  Graph2.vx2;
int yvg2[] =  Graph2.vy2;
int zvg2[] =  Graph2.vz2;
int ng = Graph2.n;

    @Override
    protected void paintComponent(Graphics gh) {
     int x = 50;
     Graphics2D drp = (Graphics2D)gh;
     Color Yellow = new Color (209,171,0);
     Color Red = new Color(255, 0, 0);
     Color Green = new Color(0, 255, 0);
     drp.drawLine(x, 500, 600 + x, 500);
     drp.drawLine(x, 500, x, 10);
     drp.drawString("Год", 600 + x, 520);
     drp.drawString("X", 580 + x, 41);
     drp.drawString("Y", 580 + x, 468);
     drp.drawString("Z", 580 + x, 492);
     drp.drawString("Доли рынка", 10+x, 10);
     int year = 2011;
     drp.drawString("2011", x, 520);
     for (int i = 0; i<11;i++){
     year++;
     x=x+50;
     String yearG =Integer.toString(year);
     drp.drawString(yearG,x, 520);
    }
     int y=10;
     x=50;
     double share = 1.0;
```

```
    drp.drawString("1.0", 80-x , y);
   for (int i = 0; i<9;i++){
    share = Math.round((share - 0.1)*100.0)/100.0;
    y=y+50;
    String shareG = String.valueOf(share);
    drp.drawString(shareG, 80-x , y);
   }
   drp.drawString("0", 80-x , y+50);
    drp.setColor(Green);
    drp.drawPolyline(yearsg, xvg2, ng);
    for (int i = 0; i < 12; i++){
      int yearg = yearsg[i];
      int xgg = xg[i];
      drp.drawOval(yearg,xgg,3,3);
    }
    drp.setColor(Yellow);
    drp.drawPolyline(yearsg, yvg2 , ng);
    for (int i = 0; i < 12; i++){
      int yearg = yearsg[i];
      int ygg = yg[i];
      drp.drawOval(yearg,ygg,3,3);
    }
    drp.setColor(Red);
    drp.drawPolyline(yearsg, zvg2 , ng);
    for (int i = 0; i < 12; i++){
      int yearg = yearsg[i];
      int zgg = zg[i];
      drp.drawOval(yearg,zgg,3,3);
    }
   }
}

public class Graph2 extends JFrame{
    public static int years[] = {70,120,170,220,270,320,370,420,470,520,570,620};
    public  static int x[];
    public  static int y[];
    public  static int z[];
    public  static int vx2[];
    public  static int vy2[];
    public  static int vz2[];
    public static int n = 12;


 public Graph2 () {
```

```
        super("График логарифмического метода");
        JPanel jcp = new JPanel(new BorderLayout());
        setContentPane(jcp);
        jcp.add(new DrawingComponent2 (), BorderLayout.CENTER);
        jcp.setBackground(Color.gray);
        setSize(800, 600);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    }

        public static void main(String[] args)   {
        new Graph2().setVisible(true);
}
}
```