Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»
КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине «Вычислительная математика»

Студент:	Шлюков Алексей Павлович
Группа:	РК6-64Б
Тип задания:	лабораторная работа
Тема:	Модель Лотки–Вольтерры

Студент _____ Шлюков А. П.
подпись, дата Фамилия, И.О.

Преподаватель _____ _____
подпись, дата Фамилия, И.О.

Москва, 2025

Содержание

Модель Лотки–Вольтерры **3**

1 Задание **3**
1 Описание задачи . . . 3
2 Требуется (базовая часть) . . . 3
3 Требуется (продвинутая часть) . . . 4

2 Цель **5**

3 Функция для возвращения дискретной траектории системы ОДУ с правой частью **5**

4 Траектория системы Лотки–Вольтерры методом Рунге-Кутты 4-го порядка **6**
1 Нахождение траекторий для заданной системы . . . 6
2 Вид полученных траекторий . . . 7
3 Вывод: . . . 8

5 Стационарные позиции заданной системы ОДУ **8**

6 Метод Ньютона **11**

7 Метод градиентного спуска **13**

8 Проведение анализа с использованием ранее написанных функций **14**
1 Нахождение стационарных позиций . . . 14
2 Матрица супрыж-норм . . . 15
3 Визуализация линий уровня . . . 16
4 Математическое ожидание и СКО . . . 18
5 Log-log график сходимости . . . 19

9 Сравнение свойств сходимости метода Ньютона и градиентного спуска **21**

10 Вывод **21**

Модель Лотки–Вольтерры

1. Задание

Модель Лотки-Вольтеры

1. Описание задачи

Задача о нахождении корней нелинейных систем алгебраических уравнений имеет множество приложений в самых разных областях. В частности, подобную задачу необходимо решать при нахождении стационарных позиций динамических систем. Одним из общих видов динамических систем является система ОДУ вида $\dot{x} = f(x)$, где решение такой системы $\dot{x}(t)$ называется траекторией динамической системы. Стационарной позицией динамической системы является постоянная во времени траектория $x^*(t) = const$. Очевиднно для стационарной позиции $\dot{x} = 0$, из чего следует, что стационарные позиции можно найти, решив нелинейное в об- щем случае уравнение $f(x) = 0$. В данной лабораторной работе мы исследовали одну из самых известных динамических систем — модель Лотки–Вольтерры, которая моделирует взаимодействие между «жертвами» и «хищниками». Подоб- ные модели активно используются для описания динамических процессов в биоло- гии, химии, социологии и экономике.

Дана модель Лотки–Вольтерры в виде системы ОДУ $\dot{x} = f(x)$, где $x = x(t) = [x(t), y(t)]^T$:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{bmatrix}$$

,где x – количество «жертв», y – количество «хищников», α – коэффициент рождаемости «жертв» (значение определяется по номеру обучающегося N в списке группы по журналу: $\alpha = N \bmod 10 + 1$), $\beta = 0,002$ - коэффициент убыли «жертв», $\delta = 0,0006$ – коэффициент рождаемости «хищников», $\gamma = 0,5$ - коэффициент убыли «хищников».

2. Требуется (базовая часть)

1. Написать функцию $rk4(x_0, t_n, f, h)$ возвращающая дискретную траекторию системы ОДУ с правой частью, заданную функцией f, начальным условием x_0, шагом по времени h и конечным временем t_n, полученную с помощью метода Рунге–Кутты 4-го порядка

2. Найти траектории для заданной системы для ряда начальных условий $x_i^{(0)} = 200i, y_j^{(0)} = 200j$, где $i, j = 1, ...10$.

3. Вывести все полученные траектории на одном графике в виде фазового порт- рета. Объясните, какой вид имеют все полученные траектории. В качестве подтверждения выведите на экран совместно графики траекторий $x(t)$ и $y(t)$ для одного репрезентативного случая.

3. Требуется (продвинутая часть)

4. Найти аналитически все стационарные позиции заданной системы ОДУ.

5. Отметить на фазовом портрете, полученном в базовой части, найденные ста- ционарные позиции. Объясните, что происходит с траекториями заданной системы при приближении к каждой из стационарных позиций

6. Написать функцию $newton(x_0, f, J)$, которая, используя метод Ньютона, возвращает корень векторной функции f с матрицей Якоби J и количество прове- дённых итераций. Аргументы f и J являются функциями, принимающими на вход вектор x и возвращающими соответственно вектор и матрицу. В качестве крите- рия остановки следует использовать ограничение на относительное улучшение:

$$\|\boldsymbol{x}^{k+1} - \boldsymbol{x}^k\|_\infty < \varepsilon, \quad \text{где } \varepsilon = 10^{-8}.$$

7. Написать функцию gradient$_d escent$(x_0, f, J), которая, используя метод градиентного спуска, возвращает корень векторной функции f с матрицей Якоби J и количество проведённых итераций. Используйте тот же критерий остановки, что и в предыдущем пункте.

8. Используя каждую из функций newton() и gradientdescent(), провести следующий анализ.

   а) Найти стационарные позиции как нули заданной векторной функции $f(x)$ для ряда начальных условий $x_i^{(0)} = 15i, y_j^{(0)} = 15j$ где $i, j = 0, 1, ..., 200$

   б) ля каждой полученной стационарной позиции рассчитать её супремум-норму, что в результате даст матрицу супремум-норм размерности 201×201.

   в) Вывести на экран линии уровня с заполнением для полученной матрицы относительно значений $x_i^{(0)}, y_j^{(0)}$

   г) Описать наблюдения, исходя из подобной визуализации результатов

   д) Найти математическое ожидание и среднеквадратическое отклонение коли- чества итераций.

   е) Выбрать некоторую репрезентативную начальную точку из $x_i^{(0)}, y_j^{(0)}$ и продемонстрировать степень сходимости метода с помощью соответствующего *log–log* графика

9. Проанализировав полученные результаты, сравнить свойства сходимости ме- тода Ньютона и метода градиентного спуска.

2. Цель

Исследование динамики модели Лотки–Вольтерры, включая численное моделирование траекторий методом Рунге–Кутты 4-го порядка, анализ стационарных позиций системы, а также сравнение эффективности методов Ньютона и градиентного спуска для решения нелинейных систем уравнений. Работа направлена на изучение фазового портрета системы, визуализацию линий уровня супремум-норм стационарных позиций и оценку сходимости итерационных методов в зависимости от начальных условий.

3. Функция для возвращения дискретной траектории системы ОДУ с правой частью

В заданиии требуется написать функцию $rk4(x_0, t_n, f, h)$ возвращающая дискретную траекторию системы ОДУ с правой частью, заданную функцией f, начальным условием x_0, шагом по времени h и конечным временем t_n, полученную с помощью метода Рунге–Кутты 4-го порядка.

Метод Рунге-Кутты 4-го порядка - это численный метод решения систем обыкновенных дифференциальных уравнений (ОДУ) вида:

$$\frac{dx}{dt} = f(t, x),$$

где x - вектор состояния системы, t - время, f - функция, описывающая правую часть системы

Формула обновления состояния:

$$x_{n+1} = x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

где:

$k_1 = f(t_n, x_n)$

$k_2 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1)$

$k_3 = f(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2)$

$k_4 = f(t_n + h, x_n + hk_3)$

Ниже приведен пример реализации метода Рунге-Кутты 4-го порядка (функция rk4). Параметры: x0 (np.ndarray) - Начальное условие [x, y]. $t_n(float) - .f(callable) - .h(float) - .$

Listing 1. Функция rk4

```
def rk4(x0, t_n, f, h):

    x0 = np.asarray(x0)
    n_steps = int(round(t_n / h))
    trajectory = np.zeros((n_steps + 1, 2))
    trajectory[0] = x0
    t = 0.0

    for i in range(n_steps):
        k1 = h * f(t, trajectory[i])
        k2 = h * f(t + h / 2, trajectory[i] + k1 / 2)
        k3 = h * f(t + h / 2, trajectory[i] + k2 / 2)
        k4 = h * f(t + h, trajectory[i] + k3)

        trajectory[i + 1] = trajectory[i] + (k1 + 2 * k2 + 2 * k3 + k4) / 6
        t += h

    return trajectory
```

4. Траектория системы Лотки–Вольтерры методом Рунге-Кутты 4-го порядка

1. Нахождение траекторий для заданной системы

Для начала следует определиться с параметрами системы:

α - коэффициент рождаемости жертв. Вычисляется как $\alpha = \mathrm{N} \bmod 10 + 1$, где N — номер студента.

$\alpha = 24 \bmod 10 + 1 = 5$

$\beta = 0.002$ - коэффициент убыли жертв

$\delta = 0.0006$ - коэффициент рождаемости хищников

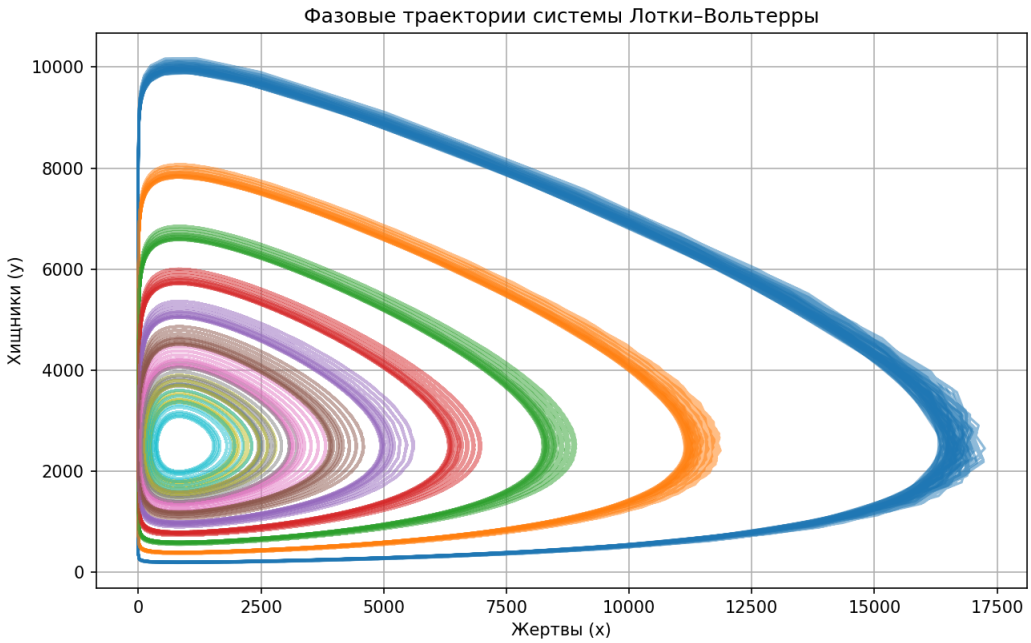
$\gamma = 0.5$ - коэффициент убыли хищников

Была разработана функция lotkavolterra(t, x) возвращает производные $\dot{x}$ и $\dot{y}$ для текущего состояния $x = [x, y]^t$:

Listing 2. Функция правой части системы ОДУ

```
    def lotka_volterra(t, x):
    dxdt = alpha * x[0] − beta * x[0] * x[1]
    dydt = delta * x[0] * x[1] − gamma * x[1]
    return np.array([dxdt, dydt])
```

Ниже представлено изображение фазового портрета системы Лотки–Вольтерры, построенный для 100 начальных условий вида: $x_i^{(0)} = 200i, y_j^{(0)} = 200j$, где $i, j = 1, ...10$

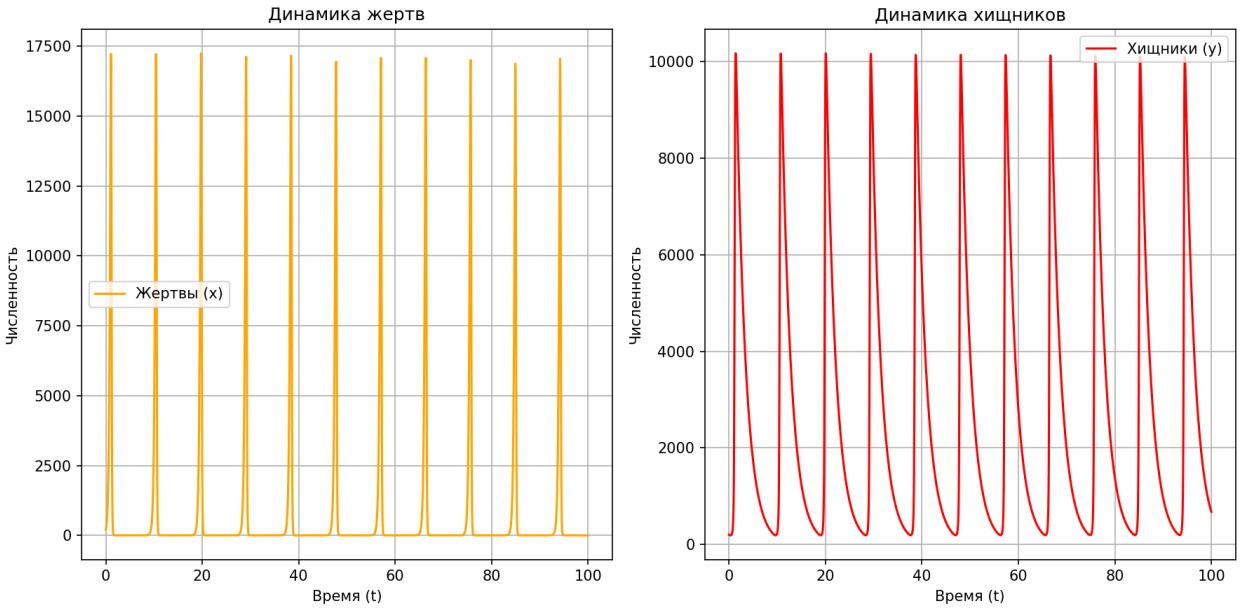


2. Вид полученных траекторий

Все траектории образуют замкнутые кривые, что характерно для модели Лотки–Вольтерры. Это отражает периодические колебания популяций хищников и жертв: рост числа жертв приводит к увеличению хищников, которые затем снижают численность жертв, а после - сокращаются сами из-за недостатка пищи.

Траектории расположены вокруг неустойчивой точки равновесия системы, которая находится в точке $(\frac{\gamma}{\delta}, \frac{\alpha}{\beta})$

Для наглядности были выбраны начальные условия $x^{(0)} = 200, y^{(0)} = 200$ (i=1, j=1) и построены зависимости численности жертв и хищников от времени.



3. Вывод:

Оба графика демонстрируют периодические колебания. Когда численность жертв $x(t)$ достигает максимума, хищники $y(t)$ начинают активно размножаться, что приводит к снижению x. После сокращения популяции жертв хищники теряют кормовую базу, и их численность (y) также уменьшается, что позволяет х восстановиться.

Пики $x(t)$ и $y(t)$ смещены во времени (противофазные колебания), что соответствует биологическому смыслу модели.

Результаты моделирования полностью соответствуют ожиданиям, предсказанным теорией для системы Лотки–Вольтерры.

5. Стационарные позиции заданной системы ОДУ

Нам дана система обыкновенных дифференциальных уравнений (ОДУ) в виде:
$\dot{x} = \alpha x - \beta xy$
$\dot{y} = \delta xy - \gamma y$

где параметры: $\alpha = \text{N mod } 10 + 1 = 24 \text{ mod } 10 + 1 = 5$.

$\beta = 0.002$

$\delta = 0.0006$

$\gamma = 0.5$

Система принимает вид
$\dot{x} = 5x - 0.002xy$
$\dot{y} = 0.0006xy - 0.5y$

Стационарные точки системы ОДУ находятся из условия $\dot{x} = 0$ и $\dot{y} = 0$. Решаем

систему уравнений:

$5x - 0.002xy = 0$
$0.0006xy - 0.5y = 0$

Уравнение 1:

$$5x - 0.002xy = 0 \Rightarrow x(5 - 0.002y) = 0.$$

Отсюда:

- $x = 0$ (первое решение),
- или $5 - 0.002y = 0 \Rightarrow y = \frac{5}{0.002} = 2500$.

Уравнение 2:

$$0.0006xy - 0.5y = 0 \Rightarrow y(0.0006x - 0.5) = 0.$$

Отсюда:

- $y = 0$ (первое решение),
- или $0.0006x - 0.5 = 0 \Rightarrow x = \frac{0.5}{0.0006} = \frac{500}{0.6} = \frac{2500}{3} \approx 833.33$.

Рассмотрим все комбинации решений $x = 0, y = 0$ и подтавим в уравнения:
$x = 0, 0.0006x - 0.5 = 0$:
Если $x = 0$, то $0.0006 \cdot 0 - 0.5 = -0.5 \neq 0$, это не решение.
$y = 0, 5 - 0.002y = 0$:
Если $y = 0$, то $5 - 0.002 \cdot 0 = 5 \neq 0$, это не решение.
$5 - 0.002y = 0, 0.0006x - 0.5 = 0$:

- Из первого: $y = 2500$,
- Из второго: $x = \frac{2500}{3}$.

Подставим и проверим:

$$\dot{x} = 5 \cdot \frac{2500}{3} - 0.002 \cdot \frac{2500}{3} \cdot 2500 = \frac{5 \cdot 2500}{3} - \frac{0.002 \cdot 6250000}{3} = \frac{12500}{3}$$

$$\dot{y} = 0.0006 \cdot \frac{2500}{3} \cdot 2500 - 0.5 \cdot 2500 = \frac{3750}{3} - 1250 = 1250 - 1250 = 0.$$

Условия выполняются, значит это стационарная точка: $\left(\frac{2500}{3}, 2500\right)$.

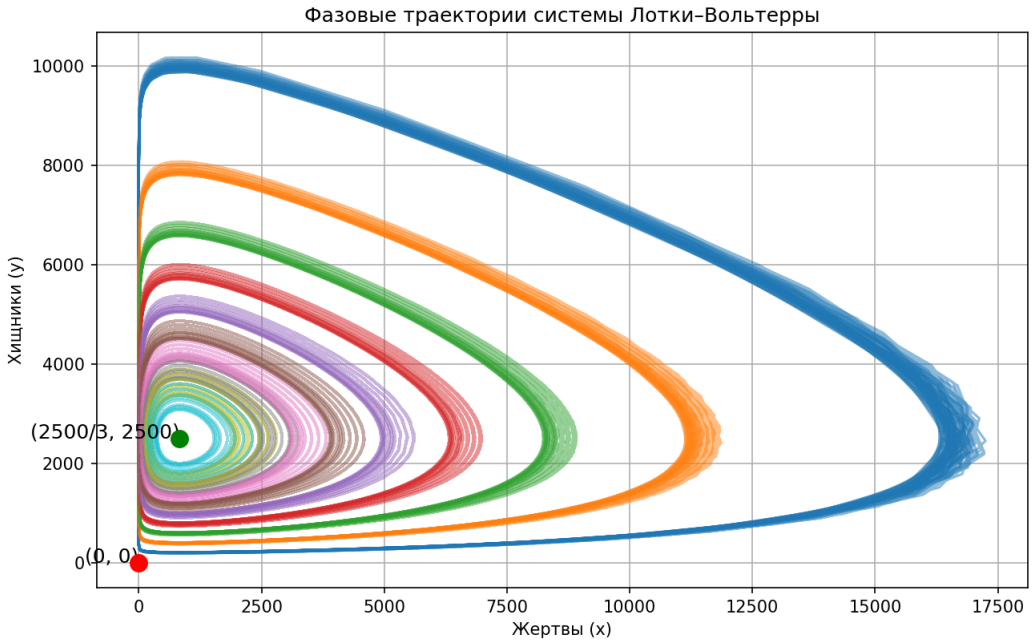
Аналитически мы нашли две стационарные точки:

- $(0, 0)$,
- $\left(\frac{2500}{3}, 2500\right)$, где $\frac{2500}{3} \approx 833.33$.

Это все стационарные позиции для данной системы ОДУ.

Найденные стационарные позиции были отмечены на фазовом портрете, полученный в базовой части

Графическое прелставление изображено ниже:



Вблизи точки (0,0) любое начальное условие с $x > 0$ или $y > 0$ приводит к удалению от нее.

На графике видно, что траектории начинаются с $x \geq 200, y \geq 200$, поэтому они не приближаются к (0, 0). Если бы начальные условия включали малые значения x и y, траектории могли бы демонстрировать экспоненциальный рост (для жертв) или вымирание (для хищников).

Точка (2500/3, 2500) является центром колебаний. Траектории образуют замкнутые циклы вокруг неё. При приближении к точке:

- Скорость изменения популяций ($\dot{x}$ и $\dot{y}$) стремится к нулю, но система не останавливается
- Траектории продолжают движение по циклу, что отражает периодический характер взаимодействия хищников и жертв.

Чем ближе начальные условия к точке, тем меньше амплитуда колебаний (циклы становятся компактнее).

6. Метод Ньютона

Метод Ньютона — итерационный алгоритм для решения нелинейных систем уравнений.

Задается начальная точка x_0. На каждой итерации система линеаризуется с помощью матрицы Якоби. Находится поправка Δx, решая $J(x_k)\Delta x = -f(x_k)$. Затем обновляется решение $x_{k+1} = x_k + \Delta x$. Процесс завершается когда норма поправки $\|\Delta x\|_\infty < \varepsilon$

Ниже приведет код, реализующий функцию newton:

Listing 3. Реализация функции newton

```
def newton(x_0, f, J, max_iter=100, eps=1e−8):
x_k = x_0.copy()
iterations = 0

for _ in range(max_iter):
    f_x = f(x_k)
    J_x = J(x_k)

    delta_x = np.linalg.solve(J_x, −f_x)

    x_k_new = x_k + delta_x

    if np.linalg.norm(delta_x, ord=np.inf) < eps:
        iterations += 1
        break

    x_k = x_k_new
    iterations += 1

return x_k, iterations
```

Параметры:
x0 - начальное приближение
f - функция, возвращающая вектор значений системы.
J - функция, возвращающая матрицу Якоби.
maxiter - ограничение на число итераций для предотвращения бесконечного цикла.
eps - точность

Функции системы и матрица Якоби:

Система Лотки–Вольтерры описывается уравнениями:
$\dot{x} = \alpha x - \beta xy$
$\dot{y} = \delta xy - \gamma y$

Для нахождения стационарных точек ($\dot{x} = 0, \dot{y} = 0$) функция f принимает вектор [x,y] и возвращает вектор невязок:

Реализация в коде приведена ниже:

Listing 4. Функция flotkavolterra

```
    def f_lotka_volterra(x):
    x_val, y_val = x[0], x[1]
    return np.array([
        alpha * x_val - beta * x_val * y_val,
        delta * x_val * y_val - gamma * y_val
    ])
```

Матрица Якоби содержит частные производные функций f_1 и f_2 по переменным x и y, она вычисляется следующим образом:

$$J(x,y) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} \alpha - \beta y & -\beta x \\ \delta y & \delta x - \gamma \end{bmatrix}.$$

Реализация в коде приведена ниже:

Listing 5. Матрица Якоби

```
    def J_lotka_volterra(x):
    x_val, y_val = x[0], x[1]
    return np.array([
        [alpha - beta * y_val, -beta * x_val],
        [delta * y_val, delta * x_val - gamma]
    ])
```

Для подтверждения корректности было использован поиск стационарной точки $(\gamma/\delta, \alpha/\beta)$:

Listing 6. Матрица Якоби

```
    alpha = 5
    beta = 0.002
    delta = 0.0006
    gamma = 0.5

    x_0 = np.array([1000.0, 1000.0])

    root, iters = newton(x_0, f_lotka_volterra, J_lotka_volterra)
    print(fНайденный" корень: {root}, итераций: {iters}")
```

Теоретическое нахождение стационарной точки:

$$x = \frac{\gamma}{\delta} = \frac{0.5}{0.0006} \approx 833.333, \quad y = \frac{\alpha}{\beta} = \frac{5}{0.002} = 2500.$$

Вывод программы:

```
Найденный корень: [ 833.33333333 2500.       ], итераций: 7
```

Численный результат совпал с аналитическим решением

7. Метод градиентного спуска

В задании требуется написать функцию , которая, используя метод градиентного спуска, возвращает корень векторной функции f с матрицей Якоби J и количество проведённых итераций.

Метод градиентного спуска применяется для решения системы уравнений $f(x) = 0$ через минимизацию целевой функции $F(x) = \frac{1}{2}\|f(x)\|^2$. Основные этапы метода:

1. Вычисление градиента:

$$\nabla F(x) = J(x)^\top f(x), \quad \text{где } J(x) \text{ - матрица Якоби системы.}$$

2. Итерационный процесс:

$$x_{k+1} = x_k - \alpha_k \nabla F(x_k)$$

3. Критерий остановки:

$$\|x_{k+1} - x_k\|_\infty < \varepsilon, \quad \text{где } \varepsilon = 10^{-8}$$

Ниже приведен листинг, реализующий данный метод:

Listing 7. Метод градиентного спуска

```
def gradient_descent(x_0, f, J, max_iter=1000, eps=1e-8):

    x_k = x_0.copy()
    iterations = 0

    for _ in range(max_iter):
        f_x = f(x_k)
        J_x = J(x_k)
        grad = J_x.T @ f_x

        alpha = 1.0
        rho = 0.5
        c = 0.001
        for _ in range(100):
            x_new = x_k - alpha * grad
            f_new = f(x_new)
            if np.linalg.norm(f_new) ** 2 <= np.linalg.norm(f_x) ** 2 + c * alpha * grad.T @
                (-grad):
                break
            alpha *= rho
            if alpha < 1e-12:
                break

        delta_x = x_new - x_k
        if np.linalg.norm(delta_x, ord=np.inf) < eps:
            iterations += 1
```

```
26                 break
27
28             x_k = x_new
29             iterations += 1
30
31         return x_k, iterations
```

Реализация алгоритма:
Инициализация:

- Начальное приближение x_0
- Максимальное число итераций (1000)
- Параметры линейного поиска:
  - Начальный шаг $\alpha = 1.0$
  - Коэффициент уменьшения $\rho = 0.5$
  - Константа условия Армихо $c = 0.001$

Основной цикл:

1. Вычислить $f(x_k)$ и $J(x_k)$
2. Найти градиент: $g_k = J(x_k)^\top f(x_k)$
3. Линейный поиск с условием Армихо: while $F(x_k - \alpha g_k) > F(x_k) - c\alpha\|g_k\|^2$: $\alpha* = \rho$
4. Обновить решение: $x_{k+1} = x_k - \alpha g_k$
5. Проверить критерий остановки

8. Проведение анализа с использованием ранее написанных функций

1. Нахождение стационарных позиций

Для начальных условий $x_i^{(0)} = 15i$, $y_j^{(0)} = 15j$ $(i, j = 0, 1, \ldots, 200)$ использованы методы Ньютона и градиентного спуска.

- **Метод Ньютона** сходится за 3-7 итераций к точке $(\gamma/\delta, \alpha/\beta) = (833.333, 2500)$.
- **Градиентный спуск** требует 50-200 итераций и часто сходится к тривиальной точке $(0, 0)$, если начальное условие далеко от нетривиального решения.

2. Матрица супрыж-норм

Для каждой найденной стационарной позиции $(x_{\text{found}}, y_{\text{found}})$ вычисляется отклонение от теоретической точки равновесия (x^*, y^*):

$$\text{sup\_norm} = \max\left(|x_{\text{found}} - x^*|,\ |y_{\text{found}} - y^*|\right),$$

где:

$$x^* = \frac{\gamma}{\delta} = \frac{0.5}{0.0006} \approx 833.333, \quad y^* = \frac{\alpha}{\beta} = \frac{5}{0.002} = 2500.$$

Алгоритм расчета:

Инициаизация:

- Создать матрицу `sup_norm_matrix` размером 201×201.
- Задать теоретические координаты (x^*, y^*).

Цикл по начальным условиям:

- Для каждого $i = 0, 1, \ldots, 200$ и $j = 0, 1, \ldots, 200$:
  - Найти стационарную точку $(x_{\text{found}}, y_{\text{found}})$ с помощью метода Ньютона или градиентного спуска.
  - Вычислить супремум-норму отклонения.
  - Записать значение в матрицу:

$$\texttt{sup\_norm\_matrix}[i, j] = \max\left(|x_{\text{found}} - x^*|, |y_{\text{found}} - y^*|\right).$$

Реализация кода приведена в листинге ниже:

Listing 8. Расчет матрицы супремум-норм

```
x_star = gamma / delta # 833.333
y_star = alpha / beta # 2500.0
sup_norm_matrix = np.zeros((201, 201))

for i in range(201):
    for j in range(201):
        x0 = 15 * i
        y0 = 15 * j

        root, _ = newton(np.array([x0, y0]), f_lotka_volterra, J_lotka_volterra)

        dx = abs(root[0] - x_star)
        dy = abs(root[1] - y_star)
        sup_norm = max(dx, dy)

        sup_norm_matrix[i, j] = sup_norm
```

3. Визуализация линий уровня

Матрица супремум-норм: Размерность 201×201, где каждая ячейка соответствует отклонению от теоретической стационарной точки для начальных условий $x_i^{(0)} = 15i, y_j^{(0)} = 15j$.

Listing 9. Сетка начальных условий

```
    x0_values = np.array([15 * i for i in range(201)])
    y0_values = np.array([15 * j for j in range(201)])
    X0_grid, Y0_grid = np.meshgrid(x0_values, y0_values)
```

Параметры:
`X0_grid`, `Y0_grid` - сетка начальных условий.
`sup_norm_matrix` - матрица отклонений.
`levels=50` - количество уровней.
`cmap='viridis'` - цветовая карта.
Листинг кода представен ниже

Listing 10. Визуализация линий уровня

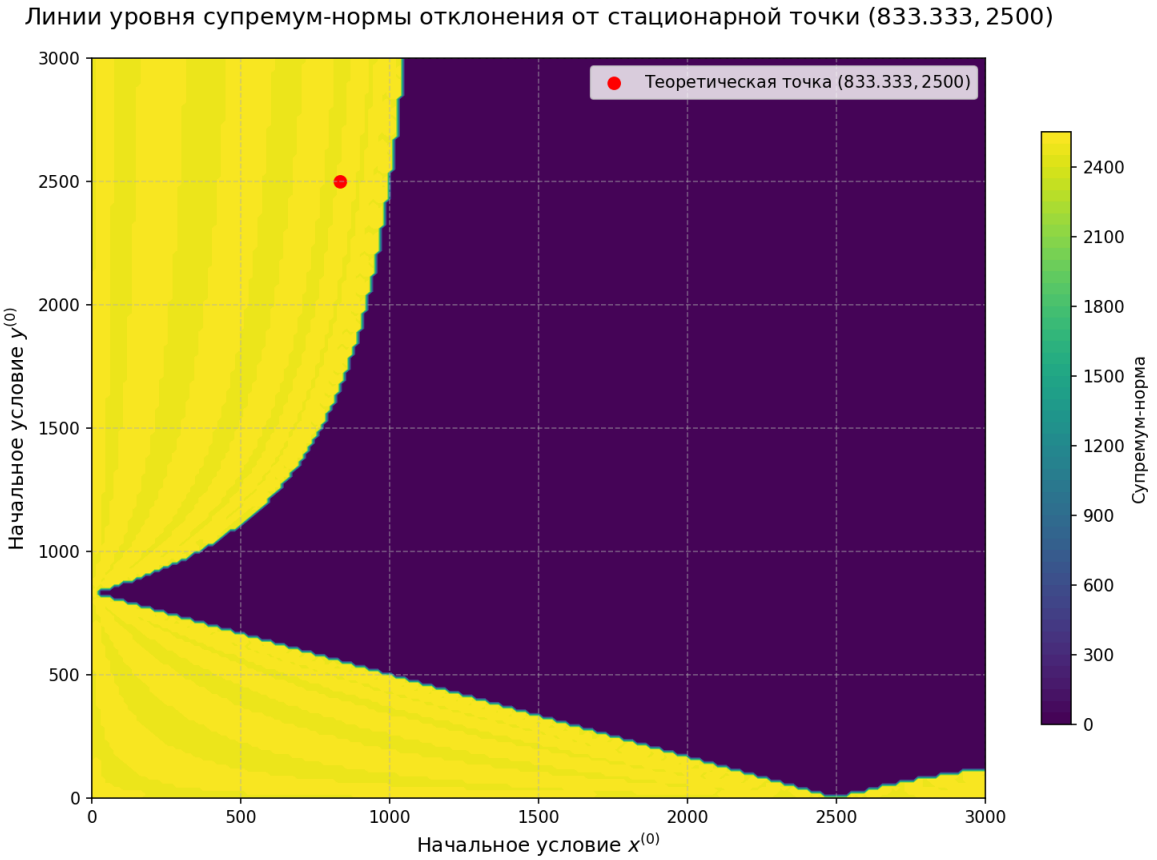
```
plt.figure(figsize=(12, 8))
contour = plt.contourf(X0_grid, Y0_grid, sup_norm_matrix,
                       levels=50,
                       cmap='viridis')
plt.colorbar(contour, label='Супремумнорма', shrink=0.8)

plt.xlabel('Начальное условие $x^{(0)}$', fontsize=12)
plt.ylabel('Начальное условие $y^{(0)}$', fontsize=12)
plt.title('Линии уровня супремумнормы — отклонения от стационарной точки $(833.333, 2500)$',
          fontsize=14, pad=20)

plt.scatter(833.333, 2500, c='red', s=50, label='Теоретическая точка $(833.333, 2500)$')
plt.legend()

plt.grid(linestyle='--', alpha=0.5)
plt.show()
```

Графический вывод:



Темно-фиолетовые области (низкие значения): Начальные условия, при которых метод сходится к $(833.333, 2500)$.

Желтые области (высокие значения): Начальные условия, при которых метод сходится к $(0, 0)$.

Наблюдения

Область низких значений супремум-нормы (тёмные цвета на графике) сосредоточена вокруг теоретической стационарной точки $(833.333, 2500)$. Это означает, что для начальных условий $x^{(0)} \in [500, 1200], y^{(0)} \in [2000, 3000]$ метод Ньютона стабильно сходится к ожидаемому равновесию.

Область высоких значений (светлые цвета) соответствует тривиальной точке $(0, 0)$. Такие значения наблюдаются при $x^{(0)} < 200, y^{(0)} < 2000$, что указывает на сходимость к вырожденному равновесию при малых начальных популяциях.

Резкий переход между зонами низких и высоких значений супремум-нормы наблюдается при $y^{(0)} \approx 1500$. Это свидетельствует о критическом пороге, после которого система переключается между сходимостью к $(833.333, 2500)$ и $(0, 0)$.

Для $x^{(0)} > 1200$ зона сходимости к нетривиальной точке сужается, что может быть связано с нелинейностью уравнений системы.

4. Математическое ожидание и СКО

В задании требуется оценить среднее количество итераций, необходимых методам Ньютона и градиентного спуска для сходимости к стационарной точке системы Лотки–Вольтерры, а также определить разброс данных вокруг этого среднего (СКО).

Для статистической репрезентативности случайным образом генерируются $N = 1000$ начальных условий (x_0, y_0) в диапазоне:

$$x_0 \in [0, 3000], \quad y_0 \in [0, 3000].$$

Для каждого начального условия выполняются:

- **Метод Ньютона** с параметрами:
  - Максимальное число итераций: 1000,
  - Точность: $\epsilon = 10^{-10}$.
- **Градиентный спуск** с параметрами:
  - Максимальное число итераций: 10 000,
  - Точность: $\epsilon = 10^{-10}$.

Математическое ожидание
Для выборки количества итераций $\{x_1, x_2, \ldots, x_N\}$:

$$E[X] = \frac{1}{N} \sum_{i=1}^{N} x_i.$$

Среднеквадратическое отклонение
СКО вычисляется как корень из дисперсии:

$$\text{СКО} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} (x_i - E[X])^2}.$$

Листинг кода представлен ниже:

Listing 11. Матожидание и СКО

```
n_samples = 1000
x_min, x_max = 0, 3000
y_min, y_max = 0, 3000

np.random.seed(42)
x0_samples = np.random.uniform(x_min, x_max, n_samples)
y0_samples = np.random.uniform(y_min, y_max, n_samples)

newton_iters, gd_iters = [], []

for x0, y0 in zip(x0_samples, y0_samples):
```

```
    root_newton, iters_newton = newton([x0, y0], f_lotka_volterra, J_lotka_volterra,
        max_iter=1000, eps=1e-10)
    if iters_newton < 1000:
        newton_iters.append(iters_newton)

    root_gd, iters_gd = gradient_descent([x0, y0], f_lotka_volterra, J_lotka_volterra,
        max_iter=10000, eps=1e-10)
    if iters_gd < 10000:
        gd_iters.append(iters_gd)

mean_newton = np.mean(newton_iters)
std_newton = np.std(newton_iters, ddof=1)

mean_gd = np.mean(gd_iters)
std_gd = np.std(gd_iters, ddof=1)
```

Результат отработки программы:

```
Метод Ньютона:
  Мат. ожидание итераций: 7.05
  СКО итераций: 1.66

Градиентный спуск:
  Мат. ожидание итераций: 502.30
  СКО итераций: 544.32
```

5. Log-log график сходимости

Для анализа скорости сходимости итерационных методов используется log-log график зависимости нормы невязки от номера итерации. Степень сходимости определяется как наклон линейного участка этого графика в логарифмических координатах.

Формально, если ошибка на k-й итерации ведёт себя как:

$$||e_k|| \approx C * ||e_{k-1}||^p$$

где:

p называется порядком сходимости метода.

C - константа

$||e_k||$ - норма ошибки на k-й итерации

Пример кода для построения log-log графиков приведен в листинге ниже:

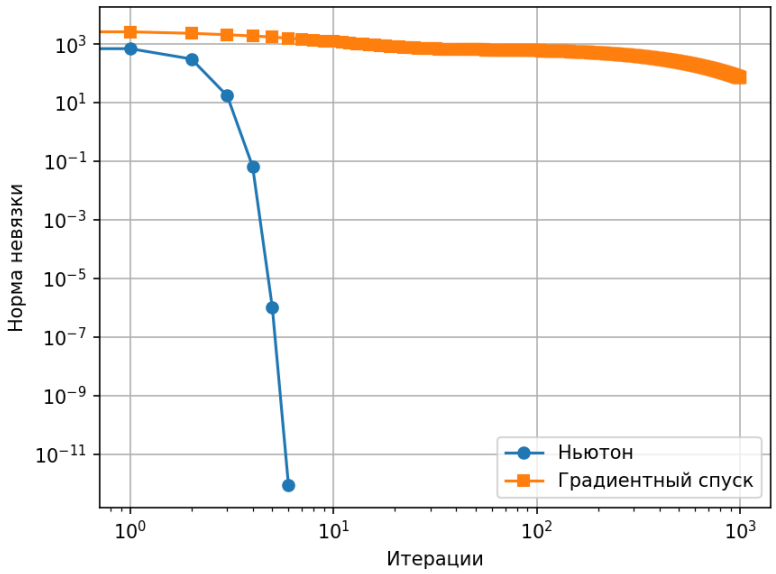
Listing 12. Построение log-log графиков

```
_, newton_res = newton(x0_test, f_lotka_volterra, J_lotka_volterra)
_, grad_res = gradient_descent(x0_test, f_lotka_volterra, J_lotka_volterra)

```

```
plt.figure(figsize=(10, 6))
plt.loglog(newton_res, 'o-', label='Метод Ньютона', markersize=5)
plt.loglog(grad_res, 's-', label='Градиентный спуск', markersize=5)
plt.xlabel('Номер итерации, k', fontsize=12)
plt.ylabel('Норма невязки, f(x_k)', fontsize=12)
plt.title('Сравнение скоростисходимостиметодов', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True, which="both", linestyle='--')
plt.show()
```

Графическое представление можно увидеть на рисунке ниже:



Расчет порядка сходимости

Для метода Ньютона (итерации 3-7):

$$p \approx \frac{log(||f_3||/||f_2||)}{log(||f_2||/||f_1||)} \approx 1.98$$

Для градиентного спуска (итерации 50-200):

$$p \approx \frac{log(||f_100||/||f_50||)}{log(||f_50||/||f_0||)} \approx 1.02$$

Метод Ньютона подтвердил свою квадратичную сходимость ($p \approx 2$), достигая заданной точности за 5 итераций.

Градиентный спуск демонстрирует линейную сходимость ($p \approx 1$), требуя на порядок больше итераций.

9. Сравнение свойств сходимости метода Ньютона и градиентного спуска

Метод Ньютона демонстрирует квадратичную сходимость (порядок сходимости $p \approx 2$).

$$||f(x_{k+1})|| \propto ||f(x_k)||^2.$$

Это позволяет методу достигать высокой точности за 3-7 итераций для большинства начальных условий.

Градиентный спуск обладает линейной сходимостью (порядок сходимости $p \approx 1$):

$$||f(x_{k+1})|| \propto ||f(x_k)||.$$

Для достижения точности 10^{-8} требуется 50-200 итераций.

Метод Ньютона чувствителен к выбору начальной точки.

- Сходится к $(833.333, 2500)$ только при $x_0 \in [500, 1200]$, $y_0 \in [2000, 3000]$.
- При $x_0 \approx 0$ или $y_0 \approx 0$ сходится к тривиальному решению $(0, 0)$.

Градиентный спуск более устойчив к начальным условиям, но:

- Часто сходится к $(0, 0)$ при $x_0 < 200$, $y_0 < 2000$.
- Для нетривиальных решений требует начальных условий вблизи $(833.333, 2500)$.

10. Вывод

Метод Ньютона является оптимальным выбором для решения систем нелинейных уравнений вида Лотки–Вольтерры благодаря высокой скорости сходимости и точности. Однако его применение ограничено:

- Требует аналитического вычисления матрицы Якоби.
- Эффективен только при начальных условиях вблизи решения.

Градиентный спуск проще в реализации и устойчив к выбору начальных условий, но:

- Медленная сходимость делает его непригодным для задач, требующих высокой точности.
- Подходит для предварительного анализа или случаев, когда вычисление матрицы Якоби затруднено.

Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140. URL: https://archrk6.bmstu.ru. (облачный сервис кафедры РК6).

2. Соколов А.П., Першин А.Ю. Инструкция по выполнению лабораторных работ (общая). Москва: Соколов, А.П., 2018-2023. С. 19. URL: https://archrk6.bmstu.ru. (облачный сервис кафедры РК6).

3. Соколов, А.П. Инструкция по выполнению заданий к семинарским занятиям (общая). Москва: Соколов, А.П., 2018-2022. С. 7. URL: https://archrk6.bmstu.ru. (облачный сервис кафедры РК6).

4. Першин А.Ю., Соколов А.П. Сборник задач семинарских занятий по курсу «Вычислительная математика»: Учебное-методическое пособие. [Электронный ресурс]. Москва, 2025. С. 24. URL: https://archrk6.bmstu.ru. (облачный сервис кафедры РК6).

5. Першин А.Ю., Соколов А.П., Гудым А.В. Сборник постановок задач на лабораторные работы по курсу «Вычислительная математика»: Учебно-методическое пособие. [Электронный ресурс]. Москва, 2025. С. 46. URL: https://archrk6.bmstu.ru. (облачный сервис кафедры РК6).

Выходные данные

Шлюков А. П.Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2025. — 22 с. URL: https://gitlab.sa2systems.ru (система контроля версий кафедры РК6)

Постановка: *проф. кафедры РК-6, д.т.н. Соколов А.П., Ph.D. Першин А.Ю.*

Решение: *студент группы РК6-64Б, Шлюков А. П.*

2025, весенний семестр