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

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

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине «Вычислительная математика»

Студент:	Долженко Анастасия Тимофеевна
Группа:	РК6-62Б
Тип задания:	лабораторная работа
Тема:	Модель Лотки–Вольтерры

Студент ______________ (подпись, дата) Долженко А.Т. (Фамилия, И.О.)

Преподаватель ______________ (подпись, дата) ______________ (Фамилия, И.О.)

Москва, 2025

Содержание

Модель Лотки–Вольтерры

1 Задание

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

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \alpha x - \beta x y \\ \delta x y - \gamma y \end{bmatrix}, \tag{1}$$

где x – количество «жертв», y – количество «хищников», α – коэффициент рождаемости «жертв» (значение определяется по номеру обучающегося N в списке группы по журналу: $\alpha = N \mod 10 + 1$), $\beta = 0.002$ – коэффициент убыли «жертв», $\delta = 0.0006$ – коэффициент рождаемости «хищников», $\gamma = 0.5$ – коэффициент убыли «хищников».

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 $i, j = 1, \dots, 10$.

3. Вывести все полученные траектории на одном графике в виде фазового портрета. 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.

Требуется (продвинутая часть)

4. Найти аналитически все стационарные позиции заданной системы ОДУ.

5. Отметить на фазовом портрете, полученном в базовой части, найденные стационарные 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 проведённых итераций. Аргументы f и J являются функциями, принимающими на вход вектор x и возвращающими соответственно вектор и матрицу. В качестве критерия остановки следует использовать ограничение на относительное улучшение:

$$\|x^{k+1} - x^k\|_\infty < \epsilon,$$

где $\epsilon = 10^{-8}$.

7. Написать функцию gradient_descent(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 gradient_descent(), провести следующий анализ:

   a) Найти стационарные позиции как нули заданной векторной функции $f(x)$ для ряда начальных условий $x_i^{(0)} = 15i$, $y_j^{(0)} = 15j$, где $i, j = 0, 1, \dots, 200$.
   b) Для каждой полученной стационарной позиции рассчитать её супремум-норму, что в результате даст матрицу супремум-норм размерности 201×201.
   c) Вывести на экран линии уровня с заполнением для полученной матрицы относительно значений $x_i^{(0)}, y_j^{(0)}$.
   d) Описать наблюдения, исходя из подобной визуализации результатов.
   e) Найти математическое ожидание и среднеквадратическое отклонение количества итераций.
   f) 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 метода Ньютона и метода градиентного спуска.

2 Цель выполнения лабораторной работы

Целью лабораторной работы является исследование модели Лотки–Вольтерры с помощью численных методов и анализа стационарных точек и сравнение эффективности методов Ньютона и градиентного спуска для нахождения корней системы.

3 Выполненные задачи

1. На языке программирования Python была написана функцииrk4(x_0, t_n, f, h) возвращающая траекторию системы ОДУ с заданными правой частью, начальными условиями, шагом по времени и временным интервалом, полученную с помощью метода Рунге–Кутты 4-го порядка.

2. Для заданной модели Лотки-Вольтерры были найдены траектории для ряда начальных условий и построен фазовый портрет.

3. Для заданной модели Лотки-Вольтерры были найдены и продемонстрированы на фазовом портрете стационарные точки.

4. Разработана функция newton(x0, f, J), реализующая метод Ньютона, с критерием остановки $\|x^{k+1} - x^k\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$.

5. Разработана функция gradient_descent(x0, f, J), реализующая метод градиентного спуска, с критерием остановки $\|x^{k+1} - x^k\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$.

6. Был проведен сравнительный анализ методов Ньютона и градиентного спуска.

4 Разработка функции rk4(x_0, t_n, f, h)

Согласно пункту 1 задания базовой части необходимо разработать функцию, 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.

Формула метода Рунге-Кутты 4-го порядка для систем ОДУ имеет вид:

$$\begin{aligned}
\boldsymbol{w}_0 &= \boldsymbol{\alpha}, \\
\boldsymbol{k}_1 &= h\boldsymbol{f}(t_i, \boldsymbol{w}_i), \\
\boldsymbol{k}_2 &= h\boldsymbol{f}\left(t_i + \frac{h}{2}, \boldsymbol{w}_i + \frac{1}{2}\boldsymbol{k}_1\right), \\
\boldsymbol{k}_3 &= h\boldsymbol{f}\left(t_i + \frac{h}{2}, \boldsymbol{w}_i + \frac{1}{2}\boldsymbol{k}_2\right), \\
\boldsymbol{k}_4 &= h\boldsymbol{f}\left(t_i + h, \boldsymbol{w}_i + \boldsymbol{k}_3\right), \\
\boldsymbol{w}_{i+1} &= \boldsymbol{w}_i + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4), \quad i = 0, 1, \ldots, m-1.
\end{aligned} \tag{2}$$

где $\boldsymbol{w}_i$ – вектор состояния системы на i-м шаге, $\boldsymbol{f}$ – вектор-функция правых частей системы ОДУ, h – шаг интегрирования, $\boldsymbol{\alpha}$ – вектор начальных условий, $\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3, \boldsymbol{k}_4$ – вспомогательные векторы.

Таким образом, в соответствии с формулой 2 на языке программирования Python была разработана функция rk4(x_0, t_n, f, h). На вход функции подаются: вектор-функция правых частей систем ОДУ f, вектор начальных условий x_0, шаг по времени h и конечное время t_n. Функция возвращает дискретную траекторию системы ОДУ, полученную с помощью метода Рунге-Кутты 4-го порядка.

Программная реализация формулы 2 представлена на листинге **??**:

Листинг 1. Программная реализация функции rk4(x_0, t_n, f, h)

```python
def rk4(x_0, t_n, f, h):

    t = np.arange(0, t_n + h, h)
    trajectory = np.zeros((len(t), len(x_0)))
    trajectory[0] = x_0

    for i in range(len(t) - 1):
```

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

    return t, trajectory
```

5 Поиск траекторий модели Лотки-Вольтерры для заданных начальных условий

Согласно пункту 2 задания базовой части необходимо найти траектории для заданной системы ОДУ модели Лотки-Вольтерры для ряда начальных условий $x_i^{(0)} = 200i$, $y_j^{(0)} = 200j$, где $i, j = 1, \ldots, 10$.

Согласно заданию, модель Лотки-Вольтерры задаётся системой ОДУ 1:

В соответствии с заданием было определено значение константы α: $\alpha = N mod 10 + 1 = 10 mod 10 + 1 = 1$, где $N = 10$ - номер студента в списке группы.

На языке программирования Python была написана функция lotka_volterra(t, trajectory), которая реализует правую часть системы 1. На вход функция принимает: t - текущее время, $trajectory$ - вектор состояния системы $[x, y]^T$. Функция возвращает вектор производных $[\dot{x}, \dot{y}]^T$, вычисленных согласно 1.

Программная реализация функции lotka_volterra(t, trajectory) представлена на листинге 2:

Листинг 2. Программная реализация функции lotka_volterra(t, trajectory)

```
def lotka_volterra(t, trajectory):

    x_val = trajectory[0]
    y_val = trajectory[1]

    dx_dt = alpha * x_val - beta * x_val * y_val
    dy_dt = delta * x_val * y_val - gamma * y_val

    return np.array([dx_dt, dy_dt])
```

Далее был написан код на языке программирования Python для вычисления траекторий заданной системы ОДУ модели Лотки-Вольтерры для ряда начальных условий $x_i^{(0)} = 200i$, $y_j^{(0)} = 200j$, где $i, j = 1, \ldots, 10$. Программная реализация вычисления траекторий представлена на листинге 3:

Листинг 3. Вычисление траектории модели Лотки-Вольтерры для заданных начальных условий

```
tn = 50 # конечное время
```

```
h = 0.05 # шаг по времени

# Вычисление траекторий для различных начальных условий
trajectories = []
for i in range(1, 11):
    for j in range(1, 11):
        x0 = np.array([200*i, 200*j])
        t, trajectory = rk4(x0, tn, lotka_volterra, h)
        trajectories.append((t, trajectory))
```

Таким образом, были получены траектории модели Лотки-Вольтерры для заданных начальных условий в виде массива trajectories, в котором парами хранятся временные сетки и состояние системы в каждом узле временной сетки.

6 Построение фазового портрета модели Лотки-Вольтерры

Согласно пункту 3 задания базовой части для визуализации динамики системы Лотки-Вольтерры был построен фазовый портрет, отображающий все полученные траектории и представленный на рисунке 1.

Полученные траектории на фазовом портрете 1 представляют собой семейство замкнутых гладких кривых, концентрически расположенных вокруг некоторой точки равновесия. Это свидетельствует о циклическом изменении количества особей в популяциях.

Внешние кривые соответствуют случаям, когда начальные популяции значительно отклоняются от равновесия, что приводит к большим амплитудам колебаний численности особей. Внутренние кривые отражают мягкие колебания при начальных условиях, близких к точке равновесия.

В качестве репрезентативного случая был выбран случай с начальными условиями $x_0 = 1000$, $y_0 = 1000$ ($i = 5$, $j = 5$). Полученные графики амплитуд количества особей в популяциях представлены на рисунке 2.

На рисунке 2 показаны:

- Фазовый сдвиг между пиками популяций
- Периодичность колебаний численности популяций с постоянной амплитудой
- Взаимозависимость числа особей жертв и хищников

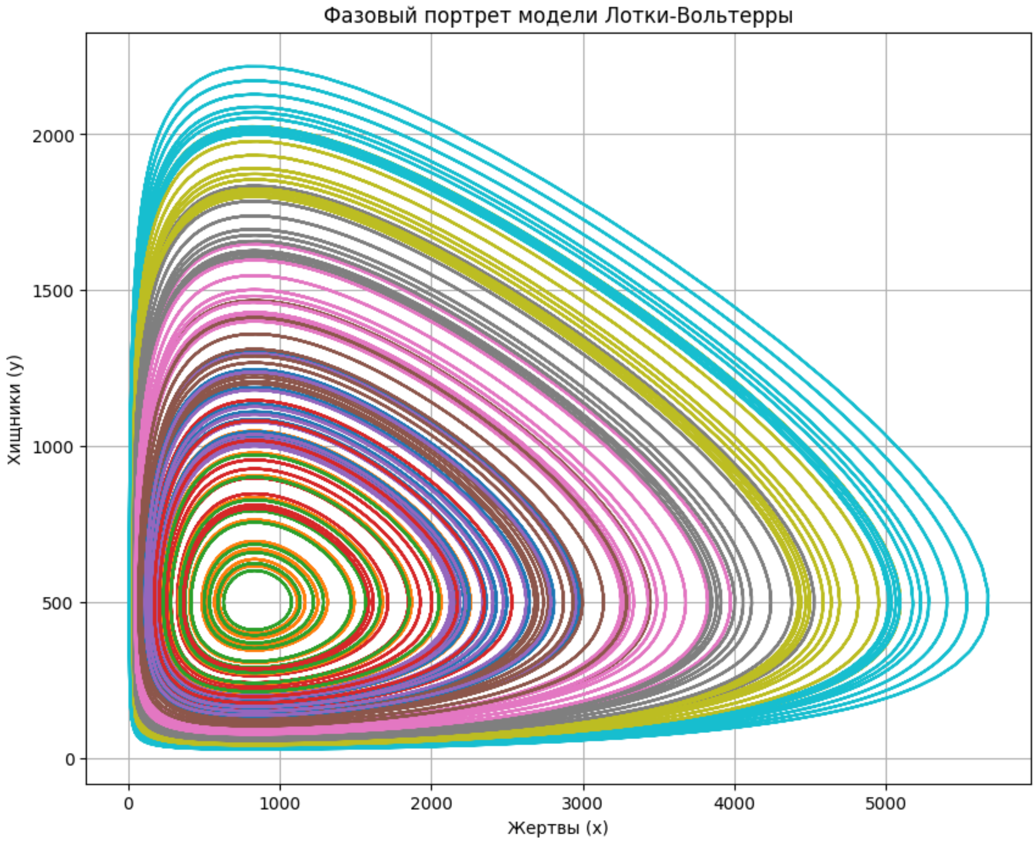

Рис. 1. Фазовый портрет модели Лотки-Вольтерры

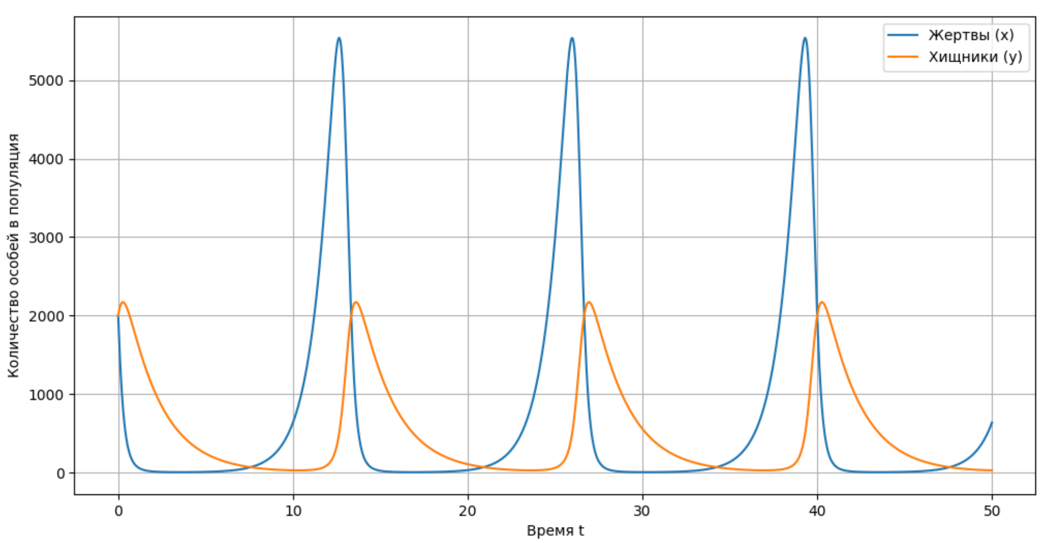

Рис. 2. Динамика популяций жертв и хищников для случая для начальных условий $x_0 = 1000$, $y_0 = 1000$ ($i = 5$, $j = 5$)

7 Поиск стационарных точек модели Лотки-Вольтерры

Согласно пункту 4 задания продвинутой части необходимо аналитически найти все стационарные точки заданной модели Лотки-Вольтерры.

Согласно заданию задана система ОДУ:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \alpha x - \beta x y \\ \delta x y - \gamma y \end{bmatrix}, \tag{3}$$

где $\alpha = 1$, $\beta = 0.002$, $\delta = 0.0006$, $\gamma = 0.5$.

Стационарные точки находятся из условий $\dot{x} = 0$ и $\dot{y} = 0$:

Решение первого уравнения $\dot{x} = 0$:

$$\alpha x - \beta x y = 0$$

$$x(\alpha - \beta y) = 0$$

$$\left[\begin{array}{l} x = 0 \\ y = \dfrac{\alpha}{\beta} = \dfrac{1}{0.002} = 500 \end{array}\right.$$

Решение второго уравнения $\dot{y} = 0$:

$$\delta x y - \gamma y = 0$$

$$y(\delta x - \gamma) = 0$$

$$\left[\begin{array}{l} y = 0 \\ x = \dfrac{\gamma}{\delta} = \dfrac{0.5}{0.0006} = 833\dfrac{1}{3} \approx 833.33 \end{array}\right.$$

Таким образом, были получены две стационарные точки:

1. $(x_1, y_1) = (0, 0)$
2. $(x_2, y_2) = (833.33, 500)$

Стоит отметить, что дробные значения в стационарной точке модели Лотки–Вольтерры (833.33,500) допустимы, поскольку система ОДУ рассматривает популяции как непрерывные величины, а коэффициенты модели $(\alpha, \beta, \gamma, \delta)$ получены путём усреднения биологических параметров.

В соотвествии с пунктом 5 задания продвинутой части найденные стационарные точки были показаны на фазовом портрете на рисунке 3.

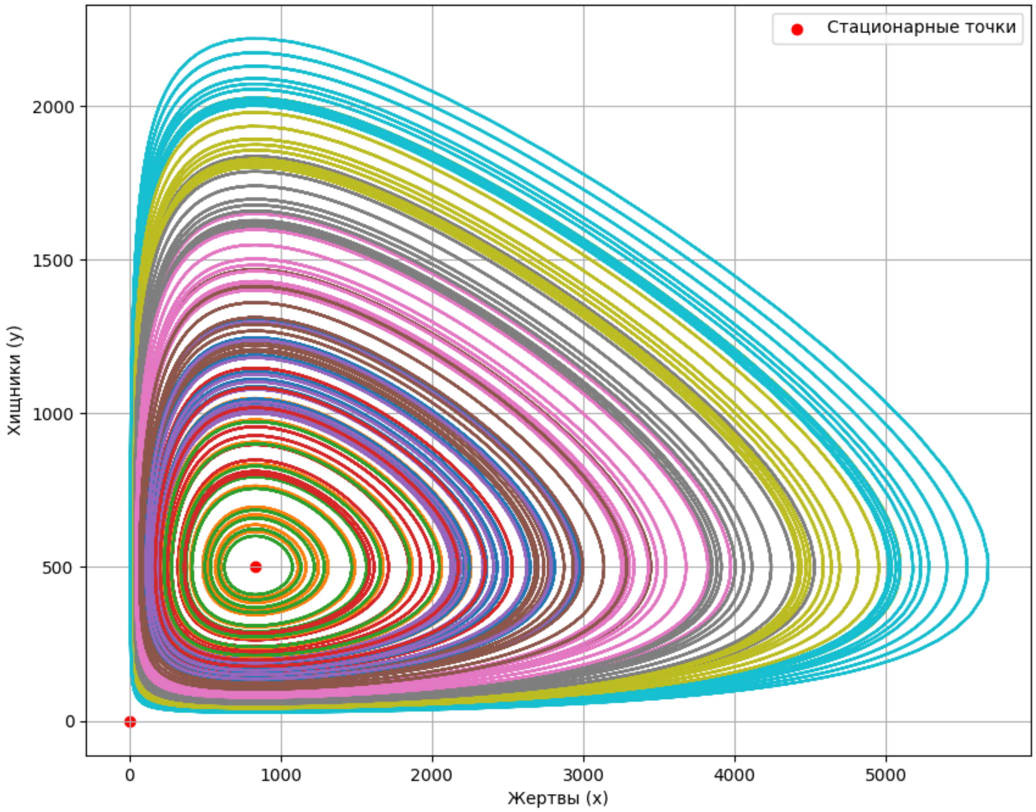

Рис. 3. Фазовый портрет модели Лотки-Вольтерры

При приближении к точке $(0, 0)$ траектории системы наблюдается резкий рост популяции жертв при малых отклонениях от нулевых значений.

В окрестности точки $(833.33, 500)$ траектории образуют замкнутые циклы. Амплитуда колебаний сохраняется при малых отклонениях, а форма траекторий приближается к эллиптической.

8 Разработка функции newton(x_0, f, J, eps)

Согласно пункту 6 задания продвинутой части необходимо написать функцию, реализующую метод Ньютона.

Метод Ньютона является итерационным численным методом для решения систем нелинейных уравнений вида:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0},$$

где $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ - вектор-функция, а $\boldsymbol{x} \in \mathbb{R}^n$ - вектор неизвестных.

Итерационная формула метода Ньютона имеет вид:

$$\boldsymbol{x}^{(k)} = \boldsymbol{g}(\boldsymbol{x}^{(k-1)}) = \boldsymbol{x}^{(k-1)} - \boldsymbol{J}^{-1}(\boldsymbol{x}^{(k-1)})\boldsymbol{f}(\boldsymbol{x}^{(k-1)}), \tag{4}$$

где $\boldsymbol{J}(\boldsymbol{x})$ - матрица Якоби системы.

Матрица Якоби $\boldsymbol{J}(\boldsymbol{x})$ определяется как:

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}, \tag{5}$$

где f_i - компоненты вектор-функции $\boldsymbol{f}$.

На практике вместо явного вычисления обратной матрицы Якоби была использована двухшаговая процедура:

$$\boldsymbol{J}(\boldsymbol{x}^{(k-1)})\Delta\boldsymbol{x} = -\boldsymbol{f}(\boldsymbol{x}^{(k-1)}) \tag{6}$$

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} + \Delta\boldsymbol{x} \tag{7}$$

Итерационный процесс прекращается при выполнении условия:

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\|_\infty < \epsilon, \tag{8}$$

где ϵ - заданная точность (согласно заданию $\epsilon = 10^{-8}$), $\|\cdot\|_\infty$ - супремум-норма.

Таким образом, на языке программирования Python была реализована функция newton(x_0, f, J, eps). Функция принимает на вход следующие параметры: x_0 - вектор начальных приближений, f - вектор-функция системы, J - функция вычисления матрицы Якоби, eps - точность (по умолчанию равен 10^{-8}). Функция возвращает количество пройденных итераций метода Ньютона iter_count и массив решения x. Программная реализация функции представлена на листинге 4.

Листинг 4. Программная реализация функции newton(x_0, f, J, eps)

```python
def newton(x_0, f, J, eps=1e-8):
    x = x_0.copy()
    iter_count = 0

    while True:
        iter_count += 1
        J_x = J(x) # Вычисление матрицы Якоби
        f_x = f(x) # Вычисление вектор-функции
        delta_x = np.linalg.solve(J_x, -f_x) # Решение СЛАУ
        x_new = x + delta_x

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

        x = x_new

    return iter_count, x
```

9 Разработка функции gradient_descent(x0, f, J, eps=1e-8)

Согласно пункту 7 задания продвинутой части необходимо разработать функцию gradient_descent(x_0, f, J), реализующую метод градиентного спуска для поиска корней системы нелинейных уравнений.

Для системы нелинейных уравнений:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0},$$

где $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n, \boldsymbol{f} = (f_1, \dots, f_n)^\top$ - вектор-функция системы, $\boldsymbol{x} = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$ - вектор неизвестных, , метод градиентного спуска минимизирует целевую функцию:

$$g(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{f}(\boldsymbol{x})\|_2^2 = \frac{1}{2}\boldsymbol{f}(\boldsymbol{x})^\top \boldsymbol{f}(\boldsymbol{x})$$

где $\|\|_2$ - евклидова норма (L2-норма), $\top$ - оператор транспонирования.

Направление спуска определяется как направление, обратное вектору градиента функции $g(x)$:

$$\boldsymbol{d}^{(k)} = -\nabla g(\boldsymbol{x}^{(k)}) = -\boldsymbol{J}^\top(\boldsymbol{x}^{(k)})\boldsymbol{f}(\boldsymbol{x}^{(k)})$$

где k - номер итерации, ∇ - оператор набла.

Итерационная формула метода градиентного спуска имеет вид:

$$\boldsymbol{z}^{(k)} = \boldsymbol{J}^\top(\boldsymbol{x}^{(k-1)})\boldsymbol{f}(\boldsymbol{x}^{(k-1)})$$

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} - t^{(k)}\frac{\boldsymbol{z}^{(k)}}{\|\boldsymbol{z}^{(k)}\|_2} \tag{9}$$

где $t^{(k)}$ определяется квадратичной интерполяцией.

На k-ом шаге метода градиентного спуска требуется найти оптимальное значение шага $t^{(k)}$, минимизирующее функцию:

$$h(t) = g\left(\boldsymbol{x}^{(k-1)} - t\frac{\boldsymbol{z}^{(k)}}{\|\boldsymbol{z}^{(k)}\|_2}\right),$$

Аппроксимация $h(t)$ производится квадратичным полиномом $P_2(t)$ по трём точкам:

- $t_1^{(k)} = 0$
- $t_3^{(k)}$ выбирается так, что $h(t_1^{(k)}) > h(t_3^{(k)})$
- $t_2^{(k)} = \frac{t_3^{(k)}}{2}$

В соответствии с интерполяцией Лагранжа выражение для $P_2(t)$ имеет вид:

$$P_2(t) = h(t_1^{(k)})\frac{(t - t_2^{(k)})(t - t_3^{(k)})}{(t_1^{(k)} - t_2^{(k)})(t_1^{(k)} - t_3^{(k)})} + h(t_2^{(k)})\frac{(t - t_1^{(k)})(t - t_3^{(k)})}{(t_2^{(k)} - t_1^{(k)})(t_2^{(k)} - t_3^{(k)})} + h(t_3^{(k)})\frac{(t - t_1^{(k)})(t - t_2^{(k)})}{(t_3^{(k)} - t_1^{(k)})(t_3^{(k)} - t_2^{(k)})}$$

Производная $P_2(t)$ имеет вид:

$$\frac{dP_2}{dt} = 2t\left(a^{(k)} + b^{(k)} + c^{(k)}\right) - \left(a^{(k)}(t_2^{(k)} + t_3^{(k)}) + b^{(k)}(t_1^{(k)} + t_3^{(k)}) + c^{(k)}(t_1^{(k)} + t_2^{(k)})\right)$$

где коэффициенты:

$$a^{(k)} = \frac{h(t_1^{(k)})}{(t_1^{(k)} - t_2^{(k)})(t_1^{(k)} - t_3^{(k)})}, \quad b^{(k)} = \frac{h(t_2^{(k)})}{(t_2^{(k)} - t_1^{(k)})(t_2^{(k)} - t_3^{(k)})}, \quad c^{(k)} = \frac{h(t_3^{(k)})}{(t_3^{(k)} - t_1^{(k)})(t_3^{(k)} - t_2^{(k)})}$$

Путём приравнивания производной к нулю находится следующее выражение для $t^{(k)}$:

$$t^{(k)} = \frac{a^{(k)}(t_2^{(k)} + t_3^{(k)}) + b^{(k)}(t_1^{(k)} + t_3^{(k)}) + c^{(k)}(t_1^{(k)} + t_2^{(k)})}{2(a^{(k)} + b^{(k)} + c^{(k)})}$$

Итерации алгоритма прекращаются при выполнении критерия:

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\|_\infty < \epsilon,$$

где $\epsilon = 10^{-8}$.

Таким образом, на языке программирования Python была разработана функция gradient_descent(x_0, f, J, eps=1e-8). Функция принимает на вход следующие параметры: x_0 - вектор начальных приближений, f - вектор-функция системы, J - функция вычисления матрицы Якоби, eps - точность (по умолчанию равен 10^{-8}). Функция возвращает количество пройденных итераций метода градиентного спуска iter_count и массив решения x.

10 Выводы

В ходе выполнения лабораторной работы были исследованы методы численного решения системы дифференциальных уравнений Лотки-Вольтерры, а также проведен анализ стационарных точек системы.

1. Был реализован метод Рунге-Кутты 4-го порядка для численного решения системы ОДУ. Были построены фазовые портреты, демонстрирующие замкнутые циклы колебания числа особей в популяциях. Также Был продемонстрирован фазовый сдвиг амплитуд числа особей в популяциях.

2. Были найдены стационарные позиции системы аналитически и отмечены на фазовом портрете, а также исследовано поведение траекторий вблизи стационарных точек.

3. Были реализованы методы Ньютона и градиентного спуска для нахождения стационарных точек.

Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140.
2. Richard L. Burden, J. Douglas Faires, Annette M. Burden. Numerical Analysis. Cengage Learning, 2015. P. 912.
3. М. Б. Лагутин. Наглядная математическая статистика. Москва. БИНОМ. Лаборатория знаний, 2009. С. 472.