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:	Горохова Александра Сергеевна
Группа:	РК6-56Б
Тип задания:	лабораторная работа
Тема:	Модель Лотки–Вольтерры

Студент ________ подпись, дата Горохова А.С. Фамилия, И.О.

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

Москва, 2022

Содержание

Модель Лотки–Вольтерры

Задание

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

$$\frac{d}{dt}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{bmatrix}, \tag{1}$$

где x - количество “жертв”, y - количество “хищников”, $\alpha = 3$ - коэффициент рождаемости “жертв”, $\beta = 0.002$ - 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 $x_i^{(0)} = 200i,\ y_i^{(0)} = 200j$, 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):

1. Найти аналитически все стационарные позиции заданной системы ОДУ.

2. 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.

3. Написать функцию $newton(x\_0, f, J)$, которая, используя метод Ньютона, возвращает корень векторной функции f с матрицей Якоби J и количество проведенных 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: $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$.

4. Написать функцию $gradient\_descent(x\_0, f, J)$, которая, используя метод градиентного спуска, возвращает корень векторной функции f с матрицей Якоби J и количество проведенных итераций. Используйте тот же критерий остановки, что и в предыдущем пункте.

5. Используя каждую из функций $newton()$ и $gradient\_descent()$, провести следующий анализ:

   (a) Найти стационарные позиции как нули заданной векторной функции $\boldsymbol{f(x)}$ для ряда начальных условий $x_i^{(0)} = 15i,\ y_i^{(0)} = 15j$, где $i,\ j = 0, \ldots, 200$.

   (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 $x_i^{(0)},\ y_i^{(0)}$.

   (d) Описать наблюдения, исходя из подобной визуализации результатов.

   (e) Найти математическое ожидание и среднеквадратическое отклонение количества итераций.

   (f) Выбрать некоторую репрезентативную начальную точку из $x_i^{(0)},\ y_i^{(0)}$ и продемонстрировать степень сходимости метода с помощью соответствующего loglog - графика.

6. Проанализировав полученные результаты, сравнить свойства сходимости метода Ньютона и метода градиентного спуска.

Цель выполнения лабораторной работы

Цель выполнения лабораторной работы: исследовать модель Лотки-Вольтерры.

1 Базовая часть

1.1 Разработка функции, возвращающей дискретную траекторию системы

Метод Рунге-Кутты 4-го порядка для систем ОДУ [1] приведен в формуле 2.

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

Ниже в листинге 1 приведен код функции rk4(x_0, t_n, f, h), разработанной с помощью метода Рунге-Кутты 4-го порядка. Она возвращает дискретную траекторию системы ОДУ с правой частью, заданной функцией f, начальным условием x_0, шагом по времени h и конечным временем t_n.

Листинг 1. Функция для метода Рунге - Кутты 4-го порядка

```
def rk4(x_0, t_n, f, h):
    t = np.arange(0, t_n, h)
    n = t.size
    kolvo = x_0.size
    x = np.zeros((kolvo, n))
    x[:, 0] = x_0
    for i in range(n - 1):
        k1 = h * f(t[i], x[:, i])
        k2 = h * f(t[i] + h / 2, x[:, i] + k1 / 2)
        k3 = h * f(t[i] + h / 2, x[:, i] + k2 / 2)
        k4 = h * f(t[i] + h, x[:, i] + k3)
        koef = (k1 + 2 * k2 + 2 * k3 + k4) / 6
        x[:, i + 1] = x[:, i] + koef
    return x
```

1.2 Анализ полученной траектории

С помощь функции rk4(x_0, t_n, f, h), представленной в листинге 1, были получены траектории заданной системы 1 для ряда начальных условий $x_i^{(0)} = 200i,\ y_i^{(0)} = 200j$, где $i,\ j = 1, \ldots, 10$. Все полученные траектории представлены на одном графике на рисунке 1. Для вычисления траекторий были взяты значения $h = 0.01$ и $t\_n = 20$.

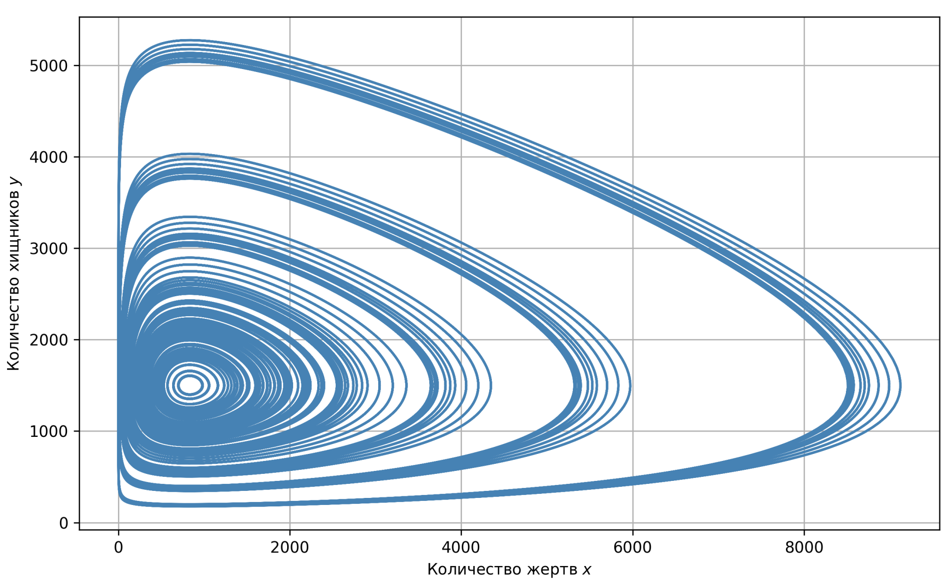

Рис. 1. Фазовый портрет траекторий для $x_i^{(0)} = 200i,\ y_i^{(0)} = 200j$, где $i,\ j = 1, \ldots, 10$

Полученные траектории имеют вид концентрических кривых.

Рисунок 1 демонстрирует, что популяции хищников y и жертв x колеблются вокруг некоторой точки. При этом колебания популяции хищников происходят с запаздыванием относительно колебаний популяции жертв.

Далее на рисунке 2 приведены графики траекторий $y(t)$ и $x(t)$ для начальных условий $x^{(0)} = 1000,\ y^{(0)} = 500$.

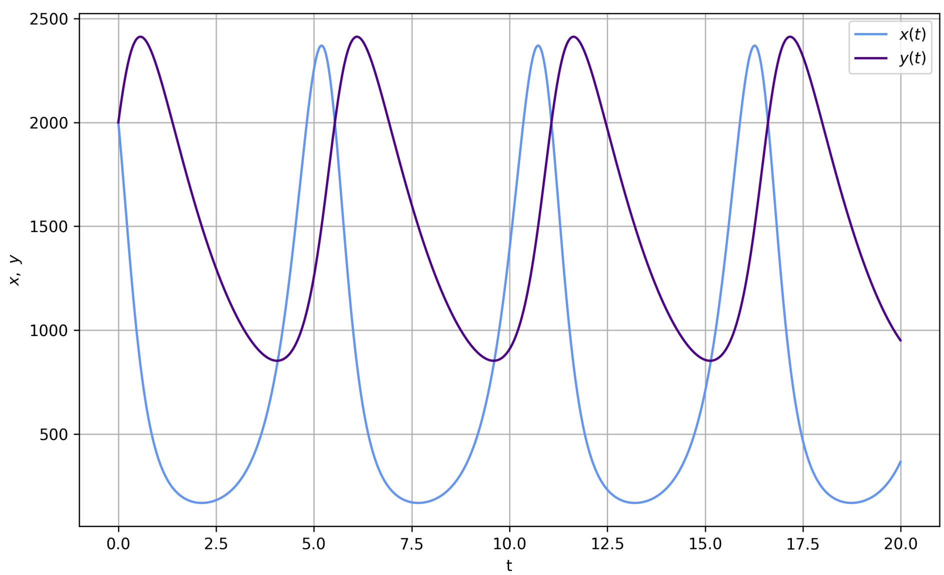

Рис. 2. Графики полученных траекторий $y(t)$ и $x(t)$ для начальных условий $x^{(0)} = 1000,\ y^{(0)} = 500$

2 Продвинутая часть

2.1 Нахождение стационарных позиций для системы 1

Стационарной позицией динамической системы является постоянная во времени траектория $\boldsymbol{x}^*(t) = const$. Для стационарной позиции $\frac{dx}{dt} = 0$, следовательно стационарные позиции можно найти, решив нелинейное в общем случае уравнение $\boldsymbol{f(x) = 0}$.

$$\begin{bmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{bmatrix} = \vec{0} \tag{3}$$

$$\begin{cases} x(3 - 0.002y) = 0, \\ y(0.0006x - 0.5) = 0. \end{cases} \tag{4}$$

Из системы уравнений 4 было получено 2 стационарные точки:

$$[833.33;\ 1500]^T.$$

$$[0;\ 0]^T.$$

Точка $(0, 0)$ не будет рассматриваться, поскольку для моделирования взаимодействия между видами их популяция не должна быть равна нулю.

На фазовом портрете, полученном в базовой части, была отмечена найденная стационарная позиция.

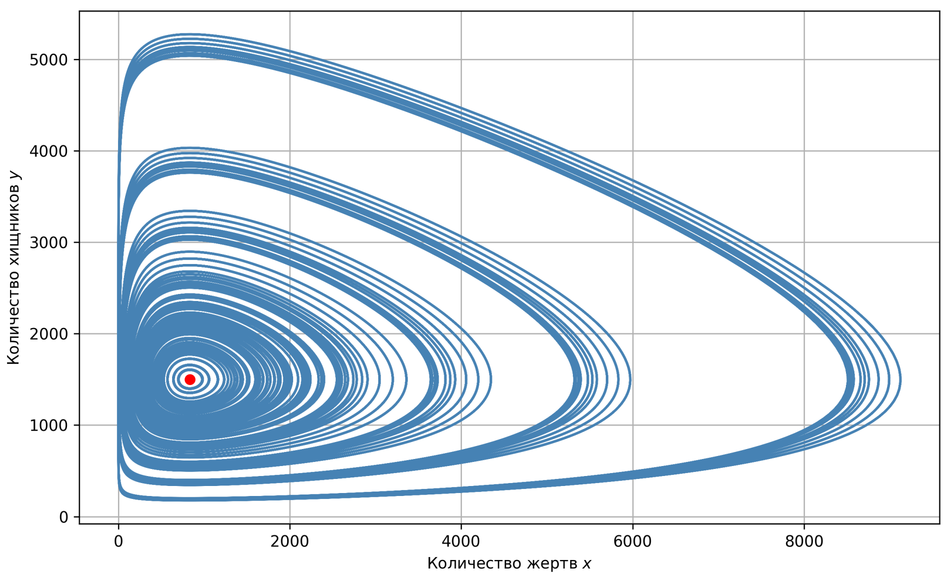

Рис. 3. Фазовый портрет траекторий, полученный в базовой части, с отмеченной стационарной позиций

Рисунок 3 демонстрирует, что популяции хищников y и жертв x колеблются вокруг найденной стационарной точки.

2.2 Разработка функции для метода Ньютона

Для написания функции $newton(x\_0, f, J)$ был использован метод Ньютона для систем нелинейных алгебраических уравнений [1], формулировка которого приведена ниже в выражении 5.

$$\boldsymbol{x}^{(k)} = \boldsymbol{g}(\boldsymbol{x}^{(k)}) = \boldsymbol{x}^{(k-1)} - \boldsymbol{J}^{-1}(\boldsymbol{x}^{(k-1)})\boldsymbol{f}(\boldsymbol{x}^{(k-1)}). \tag{5}$$

При этом $\boldsymbol{J}$ - матрица Якоби данной системы функций, составленная из частных производных этих функций по всем переменным, и имеющая следующую форму:

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_n(\boldsymbol{x})}{\partial x_n} \end{bmatrix} \tag{6}$$

Вместо вычисления обратной матрицы Якоби были выполнены следующие преобразования:

$$\boldsymbol{J}(\boldsymbol{x}^{(k-1)})\boldsymbol{y}^{(k-1)} = \boldsymbol{f}(\boldsymbol{x}^{(k-1)}), \tag{7}$$

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} - \boldsymbol{y}^{(k-1)}, \tag{8}$$

где $\boldsymbol{y}^{(k-1)}$ находится через решение первого уравнения.
Для программной реализации выражений 7 и 8 были использованы функции $lu()$ и $solve()$ из лабораторной работы №3.

Ниже в листинге 2 представлен код функции $newton(x\_0, f, J)$, которая, используя метод Ньютона, возвращает корень векторной функции f с матрицей Якоби J и количество проведенных итераций. Она принимает на вход начальное условие x_0 и аргументы f и J. При этом f и J - функции, принимающие на вход вектор x и возвращающие соответственно вектор и матрицу. В качестве критерия остановки было использовано ограничение на относительное улучшение: $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$.

Листинг 2. Функция newton, разработанная с помощью метода Ньютона

```python
def newton(x_0, f, J):
    eps = 10 ** (-8)
    x_1 = x_0
    L, U, P = lu(J(x_1), True)
    s = solve(L, U, P, f(x_1))
    x_k = x_1 - s
    i = 1
    while np.linalg.norm(x_k - x_1, ord=np.inf) > eps:
        i += 1
        x_1 = x_k
        L, U, P = lu(J(x_1), True)
        s = solve(L, U, P, f(x_1))
        x_k = x_1 - s
    return x_k, i
```

2.3 Разработка функции для метода градиентного спуска

Для написания функции $gradient\_descent(x\_0, f, J)$ был использован метод градиентного спуска [1], формульный вид которого приведен ниже.

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

$$\text{где } \boldsymbol{z}^{(k)} = \boldsymbol{J}^T(\boldsymbol{x}^{(k-1)})\boldsymbol{f}(\boldsymbol{x}^{(k-1)}). \tag{10}$$

Для нахождения оптимального значения для $\boldsymbol{t}^{(k)}$ используется метод “дробления шага”, приведенный в лекциях [1].

В качестве критерия остановки было использовано ограничение на относительное улучшение: $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$.
В листинге 3 приведена программная реализация функции $gradient\_descent(x\_0, f, J)$, которая, используя метод градиентного спуска, возвращает корень векторной функции f с матрицей Якоби J и количество проведенных итераций. Она принимает на вход начальное условие x_0, а также аргументы f и J. При этом f и J - функции, принимающие на вход вектор x и возвращающие соответственно вектор и матрицу.

Листинг 3. Функция для метода градиентного спуска

```
def gradient_descent(x_0, f, J):
    eps = 10 ** (-8)
    x_1 = x_0
    J_t = np.transpose(J(x_1))
    z_k = J_t.dot(f(x_1))
    t_k = count_t(x_1, z_k)
    x_k = count_res(x_1, t_k, z_k)
    iter = 1
    while np.linalg.norm(x_k - x_1, ord=np.inf) > eps:
        iter += 1
        x_1 = x_k
        J_t = np.transpose(J(x_1))
        z_k = J_t.dot(f(x_1))
        t_k = count_t(x_1, z_k)
        x_k = count_res(x_1, t_k, z_k)
    return x_k, iter
```

Помимо метода "дробления шага", для вычисления оптимального шага t может быть использован метод наискорейшего спуска.
Метод градиентного спуска с дроблением шага заключается в аппроксимировании параболой зависимости целевой функции от шага при фиксированном направлении поиска, а также в дальнейшем аналитическом вычислении экстремума полученной параболы.
Метод наискорейшего спуска заключается в нахождении такого шага интегрирования из набора данных, при котором значение $g(x)$ уменьшается больше всего.

2.4 Анализ данных

2.4.1 Поиск стационарных позиций

Для проведения анализа были найдены стационарные позиции как нули заданной векторной функции $\boldsymbol{f(x)}$ для ряда начальных условий $x_i^{(0)} = 15i$, $y_i^{(0)} = 15j$, где $i,\ j = 0, \ldots, 200$.
Для этого с помощью функций $newton()$ и $gradient\_descent$, представленных в листингах 2 и 3, был рассчитан корень векторной функции f с матрицей Якоби J.

Далее для каждой полученной стационарной позиции с помощью функции $np.linalg.norm()$ была найдена её супремум - норма. В итоге были получены матрицы супремум-норм для стационарных позиций, найденных с помощью функций $newton()$ и $gradient\_descent$, размерности 201×201. Расчет матриц супремум-норм для каждого метода реализован в виде двойного цикла по начальным данным.

2.4.2 Вывод полученных данных в виде линий уровня

Ниже на рисунках 4, 5 и 6 представлены линии уровня с заполнением для полученных матриц относительно значений $x_i^{(0)}$, $y_i^{(0)}$. В белой зоне норма примерно равна 1500, в темно-синей зоне - примерно 0. По оси x расположены начальные параметры $x^{(0)}$, по оси y - начальные параметры $y^{(0)}$.

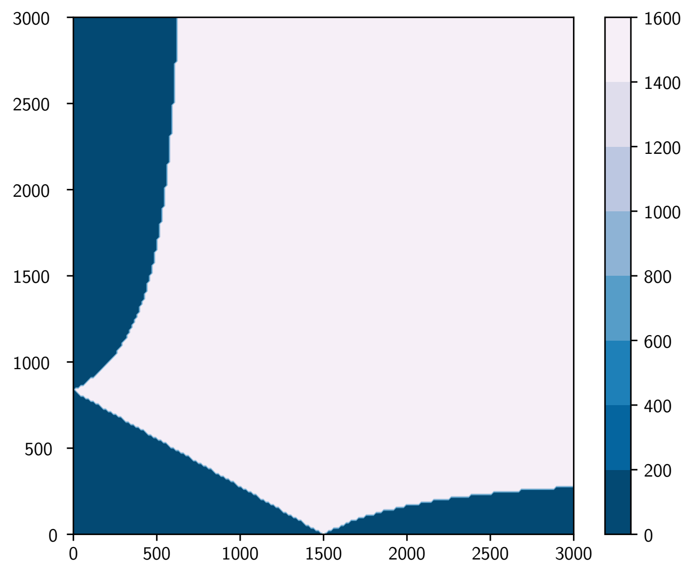

Рис. 4. Линии уровня с заполнением для полученной матрицы относительно значений $x_i^{(0)}$, $y_i^{(0)}$ для метода Ньютона

Поскольку для поиска оптимального шага t в методе градиентного спуска были реализованы два алгоритма, то в итоге получились разные результаты. Для метода градиентного спуска с нахождением t методом "дробления шага" получены линии уровня, представленные на рисунке 5. Для метода градиентного спуска с нахождением t методом наискорейшего спуска, где значения шага ищутся в интервале от 2^{-30} до 2^{30}, получены линии уровня, представленные на рисунке 6.

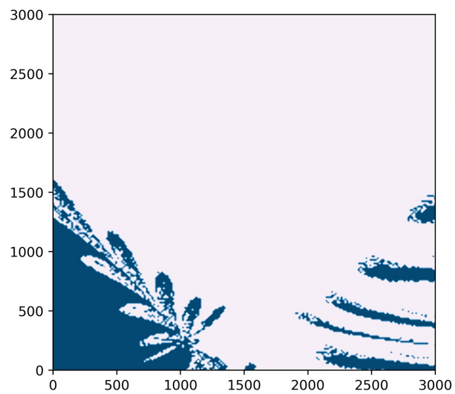

Рис. 5. Линии уровня для метода градиентного спуска с нахождением t методом "дробления шага"

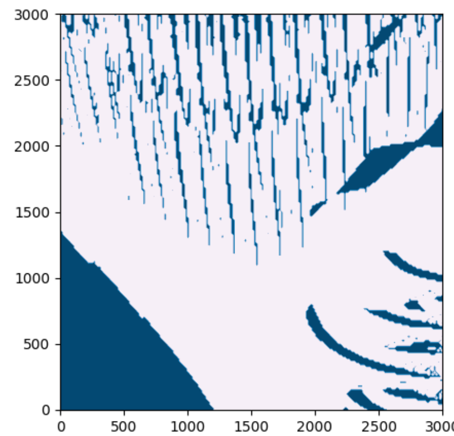

Рис. 6. Линии уровня для метода градиентного спуска с нахождением t методом наискорейшего спуска

Из рисунка 4 видно, что метод Ньютона дает достаточно точное приближение в независимости от начального приближения, в то время как точность метода градиентного спуска с нахождением t методом "дробления шага" (рис. 5) сильно зависит от начального условия, о чем свидетельствуют "лепестки". Наиболее плохая картина получилась для метода градиентного спуска с нахождением t методом наискорейшего спуска (рис. 6) - многие начальные условия x_0, несмотря на свою удаленность от стационарной позиции $[0;0]^T$ и близость к $[833.33;1500]^T$, сходятся к стационарной точке $[0;0]^T$. Данное явление наблюдается из-за слишком большого шага.

2.4.3 Поиск математического ожидания и среднеквадратичного отклонения

По формулам ниже были найдены математическое ожидание 11 и среднеквадратичное отклонение 12 количества итераций для каждого метода.

$$\boldsymbol{M}_{iter}(x) = \frac{\sum\limits_{i=1}^{n} x_i}{n}, \tag{11}$$

где x - количество итераций в i-ом эксперименте, n - количество экспериментов.

$$\boldsymbol{S}_{iter}(x) = \sqrt{\frac{1}{n(n-1)} \cdot \sum_{i=1}^{n} (x_i - M_{iter}(x))^2}, \tag{12}$$

где x - количество итераций в i-ом эксперименте, n - количество экспериментов. Полученные результаты представлены в таблице 1.

Таблица 1. Математическое ожидание и среднеквадратичного отклонение количества итераций для каждого метода

	$\boldsymbol{M}_{iter}(x)$	$\boldsymbol{S}_{iter}(x)$
Метод Ньютона	6.6742	0.0086
Метод градиентного спуска с поиском шага дроблением шага	65.6745	7.3246
Метод градиентного спуска с поиском шага наискорейшим спуском	42.7829	0.1034

2.4.4 Демонстрирация степени сходимости рассмотренных методов

По формуле 13 были найдены сходимости методов Ньютона и градиентного спуска.

$$\frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|^{\alpha}} = \lambda, \tag{13}$$

где x^* - стационарная точка, $x^{(k)}$ - репрезентативная точка, α - степень сходимости. При этом $\alpha = 1$ для линейной сходимости и $\alpha = 2$ - для квадратичной.

Ниже на рисунках 7 и 8 приведены loglog - графики степени сходимости методов Ньютона и градиентного спуска для репрезентативной начальной точки $x^{(0)} = [2000; 300]^T$.

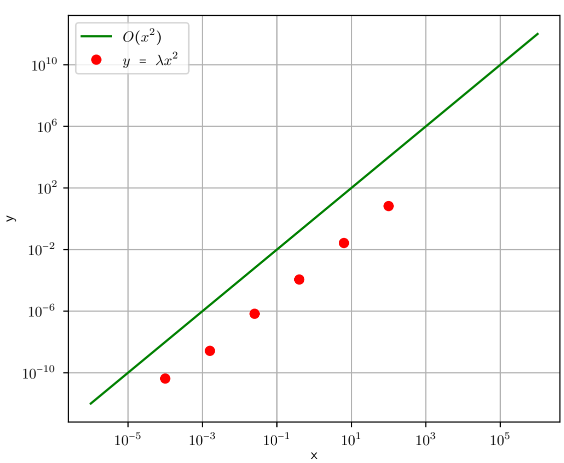

Рис. 7. Сходимость метода Ньютона

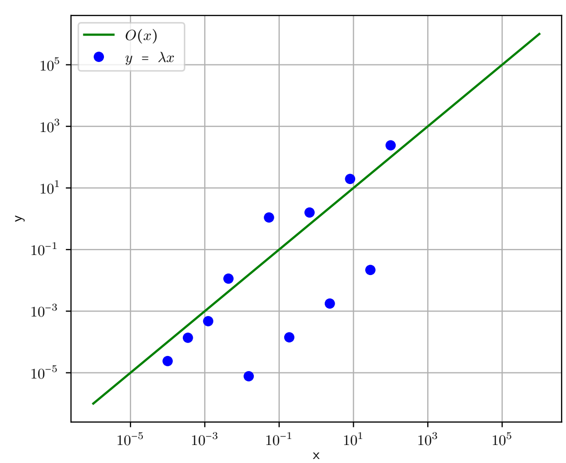

Рис. 8. Сходимость метода градиентного спуска

По рисункам 7 и 8 можно сделать вывод, что метод Ньютона имеет квадратичную сходимость, в то время как метод градиентного спуска - линейную.

2.5 Анализ полученных результатов

Проанализировав полученные ранее результаты, можно сделать следующие выводы.

1. Метод Ньютона является самым точным и быстрым из всех рассмотренных методов. Это достигается посредством его квадратичной сходимости, которая обеспечивает высокую производительность и точность вычислений даже при малом шаге. Однако данный метод сильно зависит от начальных приближений. Хороший способ гарантировать глобальную сходимость метода Ньютона состоит в комбинировании его с другим методом (например, градиентного спуска) для быстрого получения хорошей аппроксимации искомого оптимума. Тогда нескольких итераций метода Ньютона, с этой точкой в качестве исходной, достаточно для получения высокой точности.

2. Метод градиентного спуска имеет линейную сходимость, что значительно влияет на число итераций. Также можно сказать, что, если шаг выбирается малым (чтобы гарантировать сходимость), то метод сходится медленно. Увеличение же шага с целью ускорения сходимости может привести к расходимости метода. Метод градиентного спуска с дроблением шага избавляет от проблемы выбора $t^{(k)}$ на каждом шаге, заменяя на проблему выбора ϵ, δ и $t^{(0)}$, к которым он менее чувствителен. Вследствие этого матрицы супремум-норм, вычисленные методом градиентного спуска с дробления шага, получались с меньшим отклонением, чем матрицы, вычисленные методом градиентного спуска с наискорейшим спуском, поскольку наискорейший спуск напрямую влияет на выбор $t^{(k)}$, к которому градиентный спуск очень чувствителен. Метод наискорейшего спуска требует решения на каждом шаге задачи одномерной оптимизации, что напрямую влияет на число итераций: он требует меньшего числа операций, чем градиентный метод с дроблением шага.

Заключение

1. Была разработана функция, реализующая метод Рунге-Кутты 4-го порядка. Был проанализирован фазовый портрет для модели Лотки - Вольтерры - траектории имеют вид замкнутых концентрических прямых и колеблются вокруг некоторой точки.

2. Были найдены стационарные позиции заданной ОДУ. Описана зависимость траекторий, показанных на фазовом портрете, от этих позиций - они колеблются вокруг стационарной позиции.

3. Были реализованы методы Ньютона и градиентного спуска. Сделаны выводы об их производительности и точности. Найдены математическое ожидание и среднее квадратичное количества итераций двух методов.

4. Были проанализированы полученные результаты. Проведено сравнение методов на основе их сходимости.

Список использованных источников

1. Першин А.Ю. Лекции по курсу «Вычислительная математика». Москва, 2018-2021. С. 140. URL: https://archrk6.bmstu.ru/index.php/f/810046.

2. Соколов, А.П. Инструкция по выполнению лабораторных работ (общая). Москва: Соколов, А.П., 2018-2021. С. 9. 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. Сборник задач семинарских занятий по курсу «Вычислительная математика»: Учебное пособие. / Под редакцией Соколова А.П. [Электронный ресурс]. Москва, 2018-2021. С. 20. URL: https://archrk6.bmstu.ru. (облачный сервис кафедры РК6).

5. Першин А.Ю., Соколов А.П. Сборник постановок задач на лабораторные работы по курсу «Вычислительная математика»: Учебное пособие. [Электронный ресурс]. Москва, 2021. С. 54. URL: https://archrk6.bmstu.ru. (облачный сервис кафедры РК6).

Выходные данные

Горохова А.С.Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2022. — 14 с. URL: https://sa2systems.ru:88 (система контроля версий кафедры РК6)

2022, осенний семестр