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:	Кильдишев Петр Степанович
Группа:	РК6-56Б
Тип задания:	лабораторная работа
Тема:	Модель Лотки–Вольтерры

Студент ____________ Кильдишев П.С.
подпись, дата Фамилия, И.О.

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

Москва, 2022

Содержание

Модель Лотки–Вольтерры

Задание

Дана модель Лотки–Вольтерры в виде системы ОДУ $\frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}(\boldsymbol{x})$, 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”, $\delta = 0.0006$– коэффициентрождаемости “хищников”, $\gamma = 0.5$ – коэффициент убыли “хищников”.
Требуется (базовая часть):

1. Написать функцию rk4(x_0, t_n, f, h), возвращающуя 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):

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 $\boldsymbol{f}(\boldsymbol{x})$ для ряда начальных условий $x_i^{(0)} = 15i,\ y_i^{(0)} = 15j$, где $i,\ j = 0, \dots, 200$.
   (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 $x_i^{(0)},\ y_i^{(0)}$.
   (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

Все приведенные ниже примеры программного кода были реализованы на языке Python версии 3.9.

Разработка функции Рунге-Кутты

Правая часть ДУ (1) записана как:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{bmatrix}. \tag{2}$$

Метод Рунге-Кутты 4-го порядка для нахождения траектории динамической системы (решения задачи Коши по заданным начальным условиям) (1) с учетом того, что $\boldsymbol{f}(\boldsymbol{x})$ явно не зависит от координаты t:

$$\boldsymbol{x}^{(0)} = \begin{bmatrix} x^{(0)} \\ y^{(0)} \end{bmatrix},$$

$$\boldsymbol{k}_1 = h\boldsymbol{f}(\boldsymbol{x}^{(i)}), \boldsymbol{k}_2 = h\boldsymbol{f}(\boldsymbol{x}^{(i)} + \frac{1}{2}\boldsymbol{k}_1),$$

$$\boldsymbol{k}_3 = h\boldsymbol{f}(\boldsymbol{x}^{(i)} + \frac{1}{2}\boldsymbol{k}_2), \boldsymbol{k}_4 = h\boldsymbol{f}(\boldsymbol{x}_{(i)} + \boldsymbol{k_3}),$$

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{x}^{(i)} + \frac{1}{6}(\boldsymbol{k}_1 + 2\boldsymbol{k}_2 + 2\boldsymbol{k}_3 + \boldsymbol{k}_4),$$

где h - шаг по времени, $i = 0, 1, \dots, n$; $n = \frac{t\_n}{h}$, t_n - конечное время.

Несмотря на то, что t не используется в методе Рунге-Кутты явно, значения $t^{(i)}$, где $i = 0, \dots, n + 1$ важно знать:

$$t^{(0)} = 0,$$

$$t^{(i+1)} = t^{(i)} + h, i = 1, \dots, n.$$

Программная реализация метода Рунге-Кутты представлена в Листинге 1:

Листинг 1. Программная реализация метода Рунге-Кутты

```
def r_k4(x_0, t_n, f, h):
    t = np.array([i * h for i in range(0, int(t_n // h) + 2)])
    w = [x_0]
    for i in range(len(t) - 1):
        k1 = h * f(w[i])
        k2 = h * f(w[i] + 1 / 2 * k1)
        k3 = h * f(w[i] + 1 / 2 * k2)
        k4 = h * f(w[i] + k3)
        w.append(w[i] + 1 / 6 * (k1 + 2 * k2 + 2 * k3 + k4))
    return t, w
```

Нахождение траектории по начальным условиям

Нахождение траектории для ряда начальных условий $x_i^{(0)} = 200i, y_j^{(0)} = 200j$. где $i, j = 1, \dots, 10$ приведено в Листинге 2:

Листинг 2. Lf[Нахождение траектории для ряда начальных условий

```
    for i in range(1, 11):
        for j in range(1, 11):
            t, w = r_k4(np.array([200 * i, 200 * j]), t_n, f, h)
```

Фазовый портрет модели Лотки-Вольтерры

Полученные траектории для заданной системы для ряда начальных условий из задания 2 в виде фазового портрета (Рис. 1):

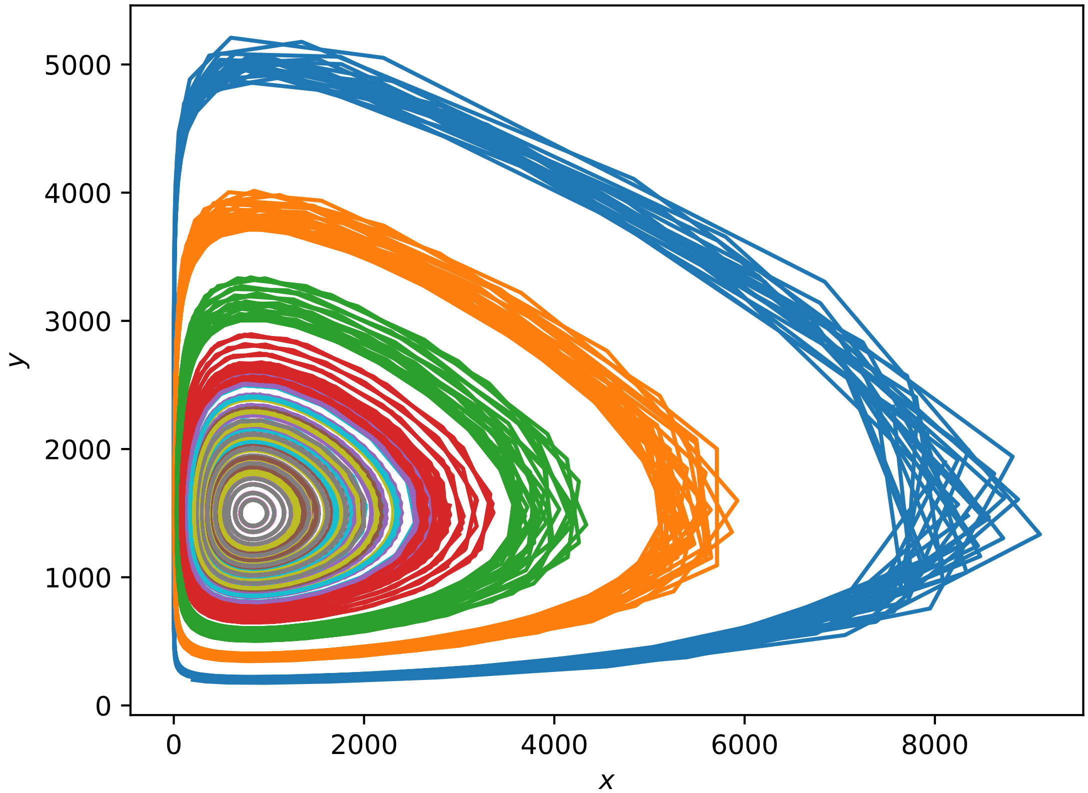

Рис. 1. Фазовый портрет модели Лотки-Вольтерры

Полученные траектории имеют вид циклических элипсоид.

По полученным траекториям видно, что при росте количества "жертв" x, начинает расти количество "хищников" y. "Хищники" начинают "поглощать" популяцию "жертв", от чего количество "хищников" растет еще больше. Затем популяция "жертв" становится мала, от чего "хищникам" становится нечем "питаться" и их количество начинает стремительно уменьшаться, пока не возвратится в исходное состояние. Таким образом цикл повторяется.

Для подтверждения вышесказанного приведены графики $x(t), y(t)$ на одной координатной плоскости (Рис. 2):

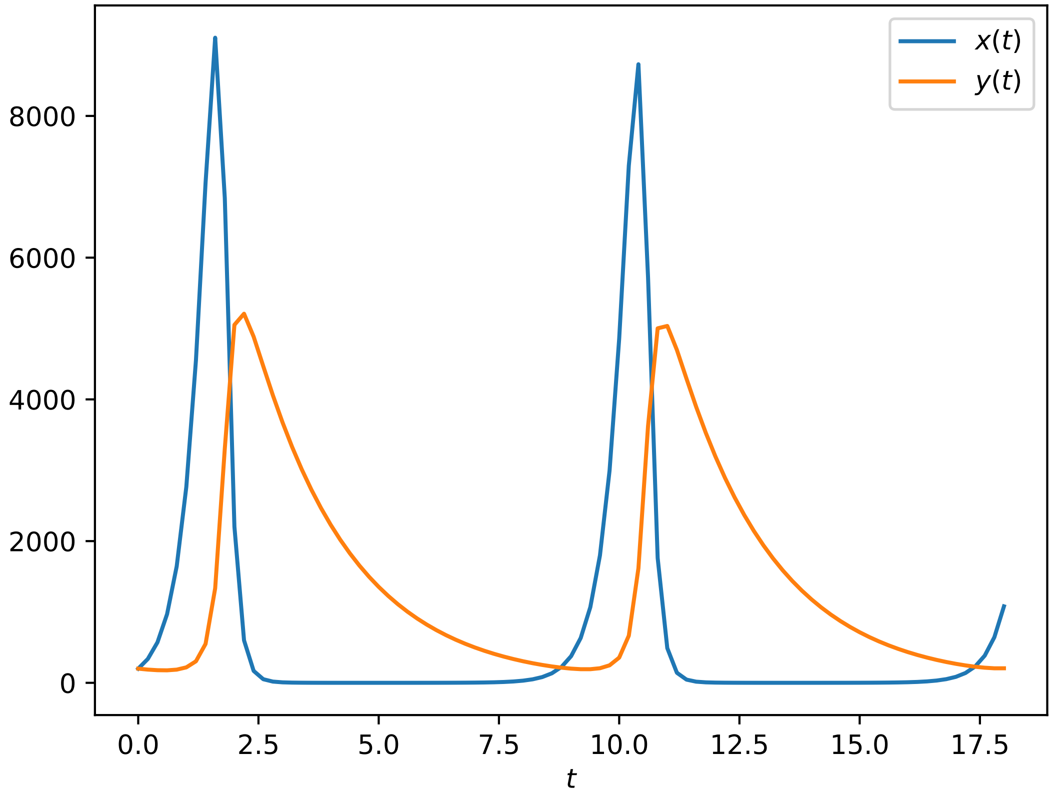

Рис. 2. Графики количества "жертв" x и "хищников" y от t

Полученные зависимости подходят под описание вышесказанного.

Вывод по базовой части

При отсутствии "хищников", количество "жертв" растет, что приводит к росту количества "хищников", в следствие чего количество "жертв" падает, из-за чего падает и количество "хищников". Затем цикл повторяется.
Такова траектория модели Лотки-Вольтерры.

2 Продвинутая часть

Аналитическон нахождение стационарных точек системы

Стационарная позиция динамической системы - постоянная во времени точка $\boldsymbol{x}^*(t) = const$. Для ее нахождения значение производной функции (2) приравнено к $\mathbf{0}$:

$$\begin{bmatrix} \alpha x - \beta xy \\ \delta xy - \gamma y \end{bmatrix} = \mathbf{0}.$$

Результат - система из 2-х уравнений:

$$\begin{cases} \alpha x - \beta xy = 0 \\ \delta yx - \gamma y = 0 \end{cases}$$

$$\begin{cases} xy = \frac{\alpha}{\beta} x \\ xy = \frac{\gamma}{\delta} y \end{cases} \tag{3}$$

Из (3):

$$y = \frac{\alpha\delta}{\beta\gamma} x. \tag{4}$$

При подстановке (4) в (3) получается:

$$x_1 = \frac{\delta}{\gamma} = \frac{2500}{3}; \tag{5}$$

$$x_2 = 0. \tag{6}$$

При подстановке (5) и (6) в (4) получается:

$$y_1 = \frac{\alpha}{\beta} = 1500, \tag{7}$$

$$y_2 = 0. \tag{8}$$

Фазовый портрет траекторий с указанием стационарных позиций

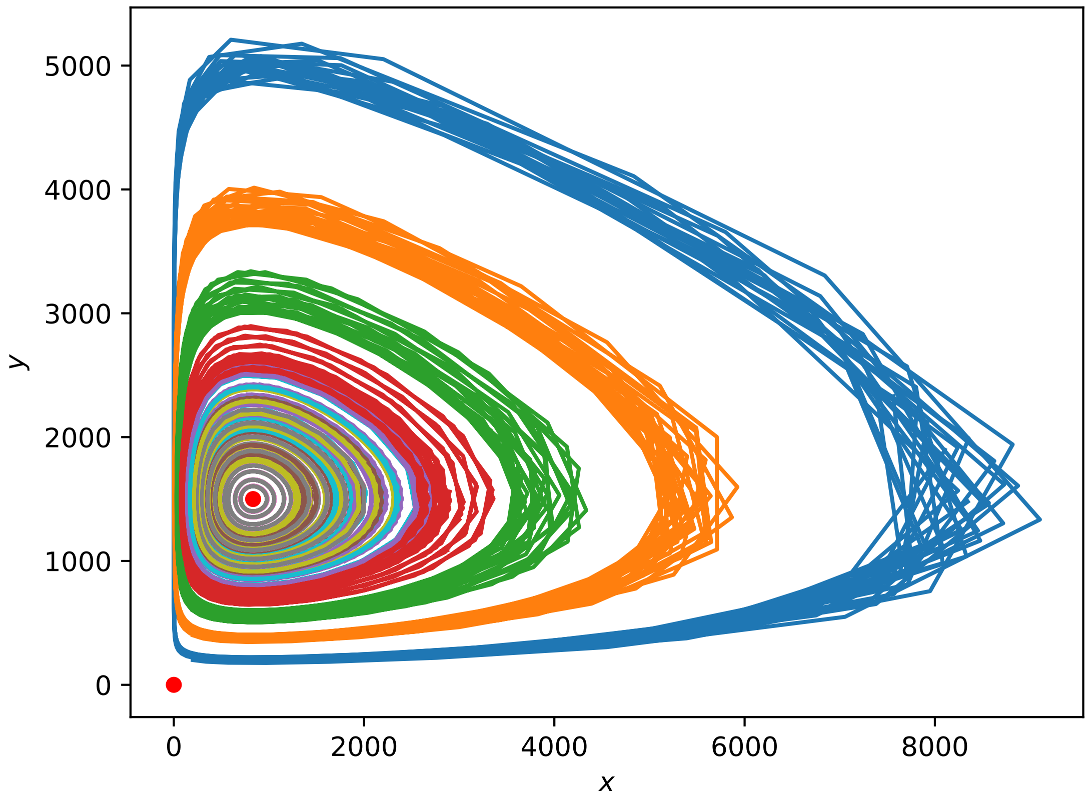

Рис. 3. Фазовый портрет траекторий для системы ОДУ (1) для ряда начальных условий с указанием найденный стационарных позиций

Стационарные точки описывают позиции, в которых система перестает изменяться. На рисунке 3 видно, динамическая система стремится к значениям стационарной точки $(\frac{2500}{3}; 1500)$ и совершает вокруг нее циклические изменения. К точке $(0; 0)$ система не стремится, так как при не нулевых значениях, x и y не будут равны 0.

Реализация метода Ньютона

Двухшаговый метод Ньютона имеет вид:

$$\boldsymbol{J}(\boldsymbol{x}^{(k-1)})\boldsymbol{y}^{(k-1)} = \boldsymbol{f}(\boldsymbol{x}^{(k-1)}), \tag{9}$$

$$\boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} - \boldsymbol{y}^{(k-1)}, \tag{10}$$

где $\boldsymbol{y}^{(k-1)}$ находится через решение первого уравнения, $\boldsymbol{J}(\boldsymbol{x}^{(k-1)})$ - матрица Якоби для точки $\boldsymbol{x}^{(k-1)}$. Для решения СЛАУ из уравнений (9) и (10) используются функция разбиения матрицы на верхне-треугольную и нижне-треугольную - lu(A, permute) и функция решения СЛАУ методом LU-разложения - solve(L, U, P, b), которые были реализованы в лабораторной работе №3.
Програмная реализация метода Ньютона представлена в Листинге 3, где в качестве критерия остановки было использовано ограничение на относительное улучшение: $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$:

Листинг 3. Программная реализация метода Ньютона

```python
def newton(x_0, f, J):
    count = 1
    J_ = J(x_0)
    P, L, U = lu_perm(J_, 1)
    b = np.array([[f(x_0)[0]], [f(x_0)[1]]], dtype=np.float64)
    y = solve(L, U, P, b)
    x_1 = x_0 - y
    while abs(np.linalg.norm(x_0, ord=np.inf) - np.linalg.norm(x_1, ord=np.inf)) >
        0.00000001:
        x_0 = x_1
        J_ = J(x_0)
        P, L, U = lu_perm(J_, 1)
        y = solve(L, U, P, f(x_0))
        x_1 = x_0 - y
        count += 1
    return x_1, count
```

Реализация метода градиентного спуска

Метод градиентного спуска имеет вид:

$$\boldsymbol{z}^{(k)} = \boldsymbol{J}^T(\boldsymbol{x}^{(k-1)})\boldsymbol{f}(\boldsymbol{x}^{(k-1)}),\ \boldsymbol{x}^{(k)} = \boldsymbol{x}^{(k-1)} - t^{(k)}\frac{\boldsymbol{z}^{(k)}}{\|\boldsymbol{z}^{(k)}\|_2}, \tag{11}$$

где $t^{(k)}$ - шаг поиска на k-том шаге.
Для нахождения квазиоптимального шага $t^{(k)}$ используется алгоритм дробного шага:

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

где

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

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

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

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

$$g(\boldsymbol{x}) = \boldsymbol{f}^T(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x}),$$

где $t_1 = 0$, $t_2 = \frac{t_3}{2}$, а t_3 подбирается путем деления или умножения начального значения, равного 1.

Программная реализация метода градиентного спуска представлена в Листинге 4, где в качестве критерия остановки было использовано ограничение на относительное улучшение: $\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$:

Листинг 4. Программная реализация метода градиентного спуска

```
def newton(x_0, f, J):
    count = 1
    J_ = J(x_0)
    P, L, U = lu_perm(J_, 1)
    b = np.array([[f(x_0)[0]], [f(x_0)[1]]], dtype=np.float64)
    y = solve(L, U, P, b)
    x_1 = x_0 - y
    while abs(np.linalg.norm(x_0, ord=np.inf) - np.linalg.norm(x_1, ord=np.inf)) > 0.00000001:
        x_0 = x_1
        J_ = J(x_0)
        P, L, U = lu_perm(J_, 1)
        y = solve(L, U, P, f(x_0))
        x_1 = x_0 - y
        count += 1
    return x_1, count
```

Анализ нахождения стационарных точек численными методами

Нахождение стационарных позиций, как нулей заданной векторной функции (2) для ряда начальных значений $x_i^{(0)} = 15i, y_j^{(0)} = 15j$, где $i, j = 1, \dots, 200$. приведено в Листинге 5:

Листинг 5. Нахождение стационарных позиций

```
    for i in range(0, 201):
        for j in range(0, 201):
            x1, c1 = newton(np.array([15 * i, 15 * j]), f, J)
            x2, c2 = gradient_descent(np.array([15 * i, 15 * j]), f, J)
```

Супремум-норма для каждой стационарной позиции $\|\boldsymbol{x}_{\boldsymbol{ij}}^*\|_\infty$ находится на каждой итерации i и j. Из них формируется матрица.

Линии уровня для матриц супремум-норм, полученных разными методами относительно $\frac{x_i^{(0)}}{15}$ и $\frac{x_i^{(0)}}{15}$ представлены на рисунках 4 и 5:

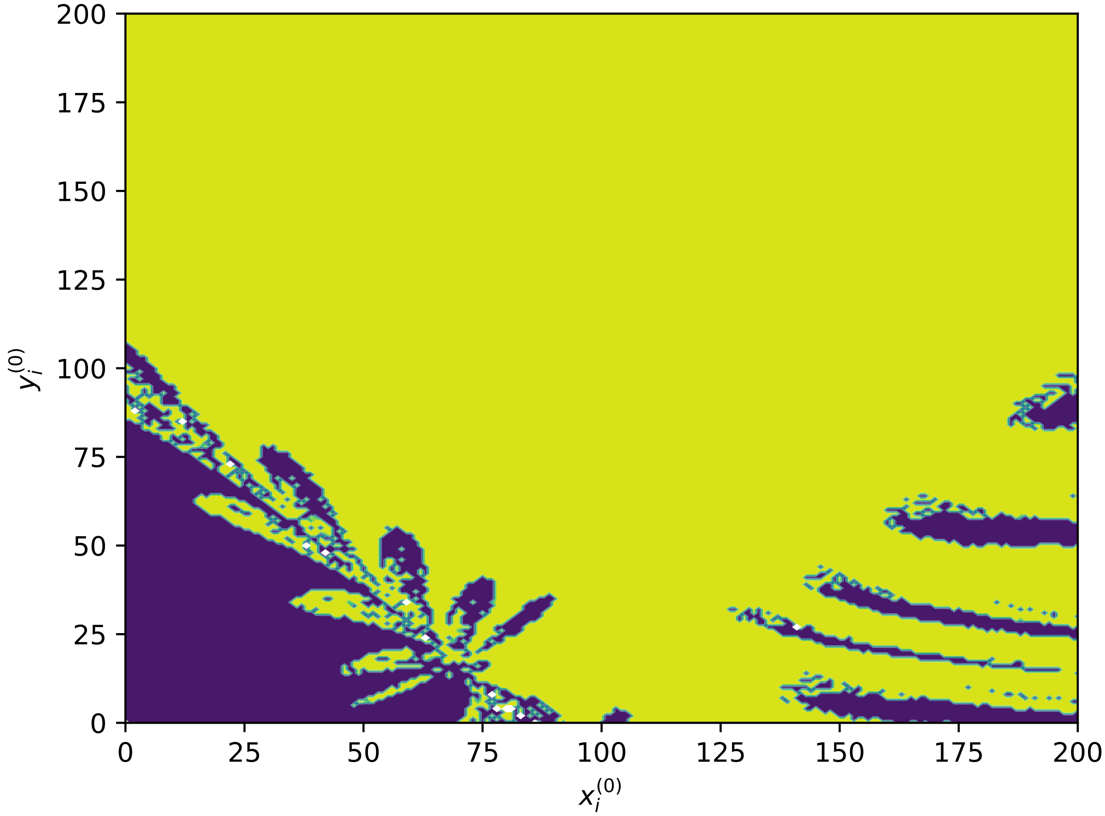

Рис. 4. Линии уровня для матрицы, полученной методом градиентного спуска относительно $\frac{x_i^{(0)}}{15}$ и $\frac{x_i^{(0)}}{15}$

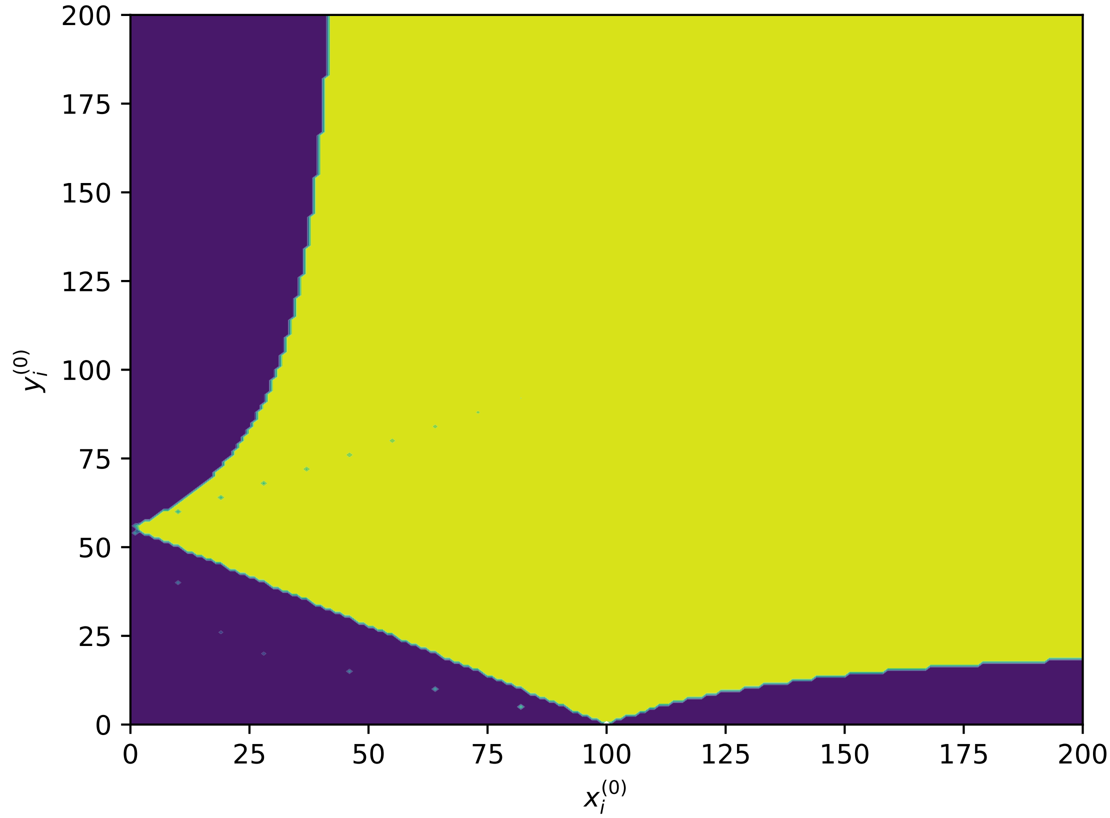

Рис. 5. Линии уровня для матрицы, полученной методом Ньютона относительно $\frac{x_i^{(0)}}{15}$ и $\frac{x_i^{(0)}}{15}$

По данным линиям уровня видно, что матрица, полученная методом Ньютона является более точной, что может быть связано с очень малыми значениями шага и вследствие накоплением машинной погрешности в методе градиентного спуска.

Математическое ожидание количества итераций вычисляется по формуле:

$$M = \frac{1}{n^2} \sum_{i=0} n \sum_{j=0} n c_{ij},$$

где $n, m = 200$ - количество итераций изменения начальных значений $x^{(0)}$ и $y^{(0)}$, c_{ij} - количество итераций метода на шаге i, j.
Значение математического ожидания количества итераций для метода Ньютона $\approx$ 6.669; для метода градиентного спуска ≈ 68.406.

Среднеквалратичное отклонение количества итераций вычисляется по формуле:

$$S = \frac{1}{n^2 - 1} \sqrt{\sum_{i=0} n \sum_{j=0} n (c_{ij} - M)^2},$$

где $n, m = 200$ - количество итераций изменения начальных значений $x^{(0)}$ и $y^{(0)}$, c_{ij} - количество итераций метода на шаге i, j; M - математическое ожидание количества итераций.
Значение среднеквалратичнго отклонения количества итераций для метода Ньютона ≈ 1.740; для метода градиентного спуска ≈ 1684.652.
Для демонстрации сходимости методов, построены графики $y = \lambda x$, где λ отражает степень сходимости функции и вычисляется по формуле:

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

где $\alpha = 1$ при линейной сходимости и 2 при квадратичной.

Для точки $x^{(0)} = 1800$, $y^{(0)} = 400$ построены графики $y = \lambda x$:

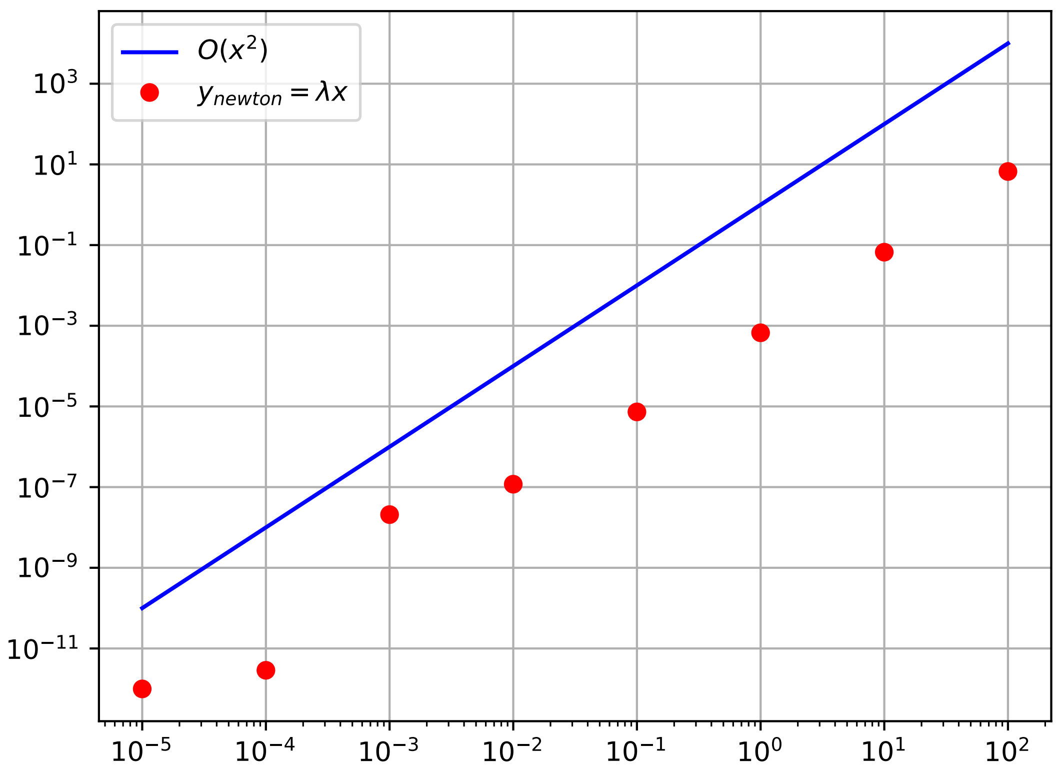

Рис. 6. График сходимости метода Ньютона

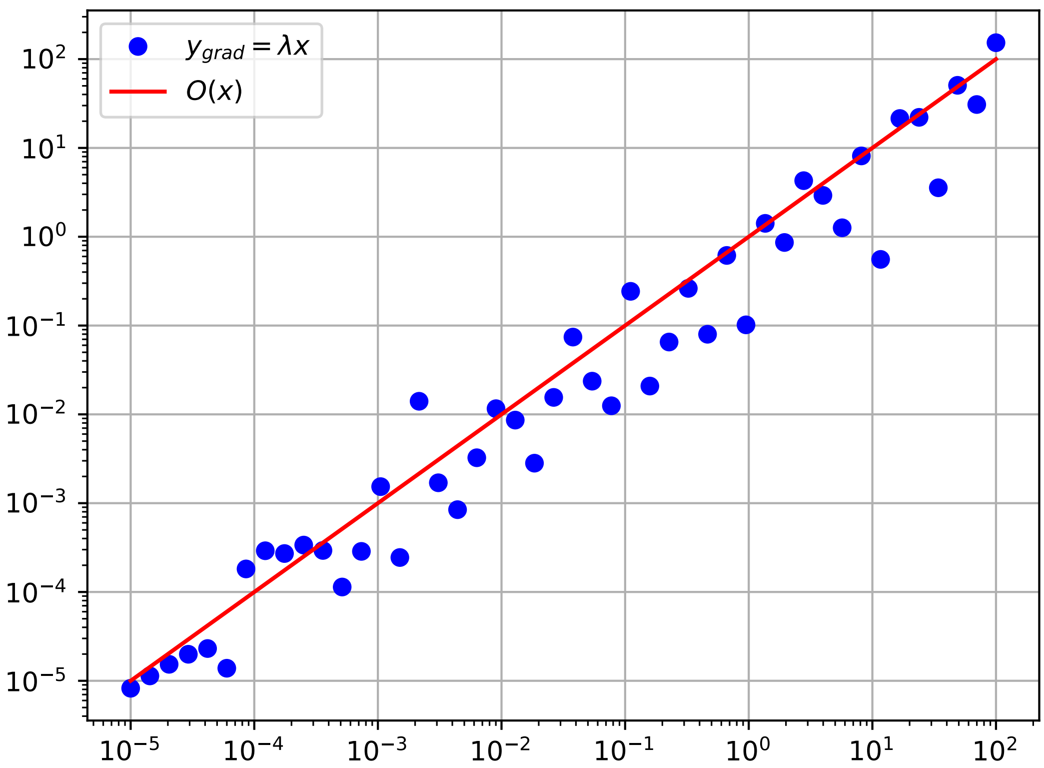

Рис. 7. График сходимости метода градиентного спуска

Данные графики подтверждают квадратичную сходимость метода Ньютона и линейную сходимость метода градиентного спуска.

Вывод по продвинутой части

По полученным данным можно сделать следующие выводы: Нахождении стационарных точек методом градиентного спуска в сравнении с нахождении методом Ньютона является более медленным процессом, так как требует большего количества итераций из-за линейной сходимости, в отличие от квадратичной у Ньютона. Также в методе градиентного спуска наблюдается меньшая точность и больший разброс значений вследствие накопления погрешности из-за большого количества итераций и маленьких значений, учавствующих в вычислениях.

3 Заключение

1. В базовой части была проанализирована траектория модели Лотки-Вольтерры;

2. В продвинутой части проанализирована точность и производительность метода 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. 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

Кильдишев П.С.Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2022. — 16 с. URL: https://sa2systems.ru:88 (система контроля версий кафедры РК6)

2022, осенний семестр