



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:	Новичкова Мария Алексеевна
Группа:	РК6-56Б
Тип задания:	лабораторная работа
Тема:	Модель Лотки-Вольтерры

Студент

подпись, дата

Новичкова М.А.
Фамилия, И.О.

Преподаватель

подпись, дата

Фамилия, И.О.

Москва, 2022

Содержание

Модель Лотки-Вольтерры		3
Задание		3
Цель выполнения лабораторной работы		5
1	Базовая часть	5
	Разработка функции, возвращающей дискретную траекторию системы	5
	Анализ полученной траектории системы для ряда начальных условий	6
2	Продвинутая часть	7
	Поиск стационарных позиций системы	7
	Разработка функции вычисления корня системы с помощью метода Ньютона	9
	Разработка функции вычисления корня системы с помощью метода градиентного спуска	10
	Анализ методов нахождения корней системы	11
	Итоги анализа методов	15
Заключение		16

Модель Лотки-Вольтерры

Задание

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

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

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)`, возвращающую дискретную траекторию системы ОДУ с правой частью, заданной 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 \mathbf{x} и возвращающими соответственно вектор и матрицу. В качестве критерия остановки следует использовать ограничение на относительное улучшение: $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_\infty < \epsilon$, 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 $\mathbf{f}(\mathbf{x})$ 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: Исследование методов нахождения корней нелинейных систем и их сходимости.

1 Базовая часть

Разработка функции, возвращающей дискретную траекторию системы

Функция `rk4(x_0, t_n, f, h)` была разработана на основе метода Рунге-Кутты 4-го порядка для систем ОДУ [1]:

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

Ниже приведен листинг 1, содержащий код функции `rk4(x_0, t_n, f, h)`, которая 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 возвращающая дискретную траекторию системы ОДУ с помощью метода Рунге-Кутты 4-го порядка

```
1 def rk4(x0, tn, f, h):
2     t = np.arange(0, tn, h)
3     time = t.size
4     kolvo = x0.size
5     x = np.zeros((time, kolvo))
6     x[0] = x0
7     for k in range(time - 1):
8         k1 = h * f(t[k], x[k])
9         k2 = h * f(t[k] + h / 2, x[k] + k1 / 2)
10        k3 = h * f(t[k] + h / 2, x[k] + k2 / 2)
11        k4 = h * f(t[k] + h, x[k] + k3)
12        dx = (k1 + 2 * k2 + 2 * k3 + k4) / 6
```

```

13     x[k + 1] = x[k] + dx
14     return x, t

```

Анализ полученной траектории системы для ряда начальных условий

Для заданной системы для ряда начальных условий $x_i^{(0)} = 200i$, $y_i^{(0)} = 200j$, где $i, j = 1, \dots, 10$ были найдены траектории для системы (1). Все полученные траектории представлены на одном графике в виде фазового портрета на рисунке 1.

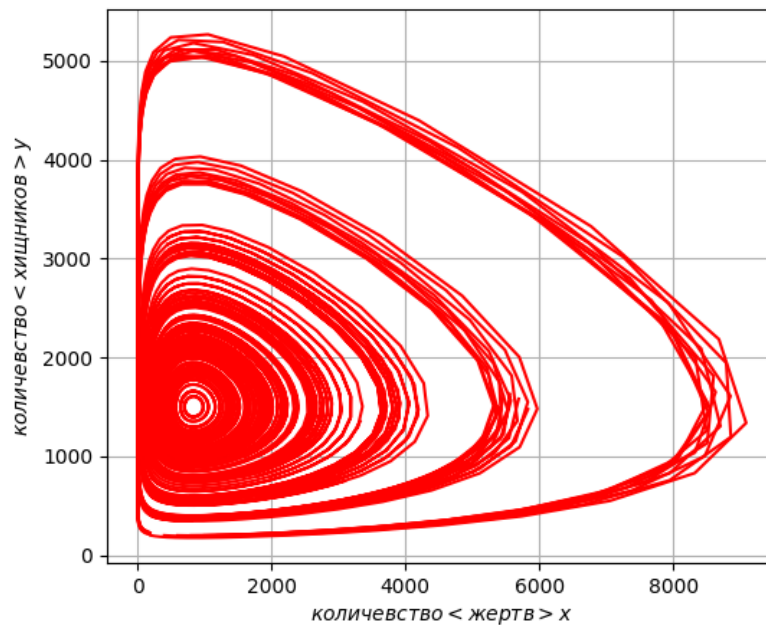


Рис. 1. Фазовый портрет траекторий для системы ОДУ (1) для ряда начальных условий

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 $x^{(0)} = 800$, $y^{(0)} = 600$. Рисунок 2 подтверждает сделанные выводы.

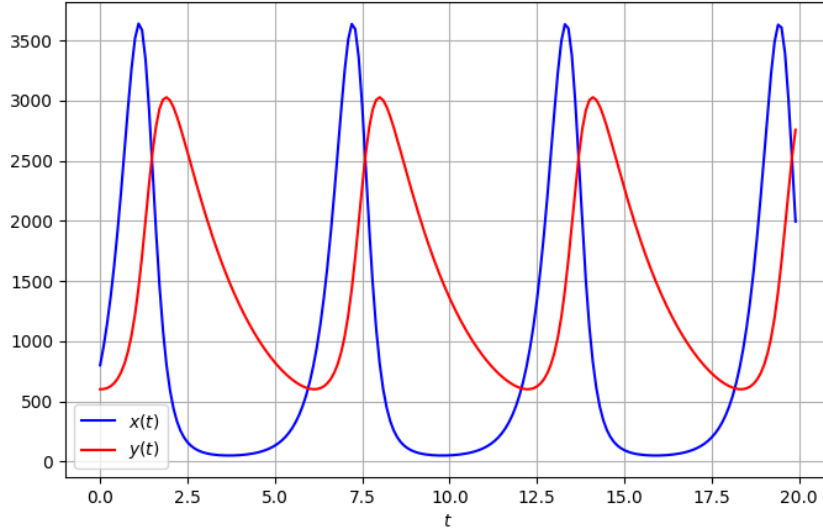


Рис. 2. Графики полученных траекторий $y(t)$ и $x(t)$ для начальных условий $x^{(0)} = 800, y^{(0)} = 600$

2 Продвинутая часть

Поиск стационарных позиций системы

Стационарной позицией динамической системы является позиция $\mathbf{x}(t)$, в которой траектория $\mathbf{x}^*(t)$ постоянна во времени $\mathbf{x}^*(t) = const$. Для стационарной позиции $\frac{d\mathbf{x}}{dt} = 0$, из чего следует, что стационарные позиции можно найти, решив нелинейное в общем случае уравнение $\mathbf{f}(\mathbf{x}) = 0$.

Далее приведено решение этого уравнения для системы (1).

$$\begin{cases} \alpha x - \beta xy = 0; \\ \delta xy - \gamma y = 0; \end{cases} \quad \begin{cases} x(\alpha - \beta y) = 0; \\ y(\delta x - \gamma) = 0; \end{cases} \quad \begin{cases} x = 0; \\ y = 0; \\ x = \frac{\alpha}{\beta}; \\ y = \frac{\gamma}{\delta}. \end{cases}$$

Таким образом, были получены две стационарные точки системы: $(0, 0)$ и $(1500, 800\frac{1}{3})$.

Так как для моделирования взаимодействия между видами их популяция не должна быть равна нулю, то точка $(0, 0)$ не будет рассматриваться.

Далее приведен рисунок 3, на котором изображен фазовый портрет траекторий системы, полученный в базовой части лабораторной работы, и отмечена стационарная точка системы.

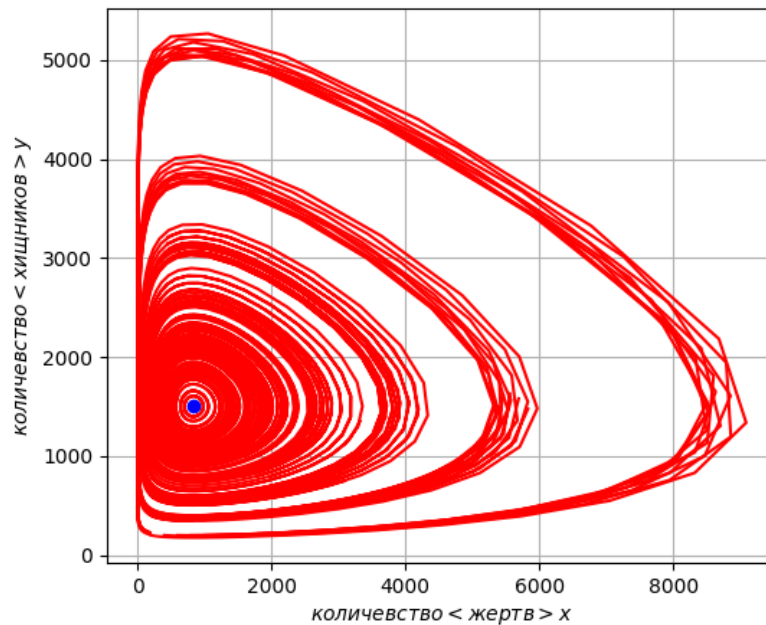


Рис. 3. Фазовый портрет траекторий для системы ОДУ (1) для ряда начальных условий с отмеченной стационарной точкой

По графику на рисунке 3 можно сделать вывод: при приближении к стационарной точке амплитуда колебаний уменьшается, и в стационарной точке амплитуда равна нулю. Это можно продемонстрировать, построив графики траекторий $y(t)$ и $x(t)$ для начальных условий $y^{(0)} = 1500$, $x^{(0)} = 800\frac{1}{3}$ (рисунок 4)

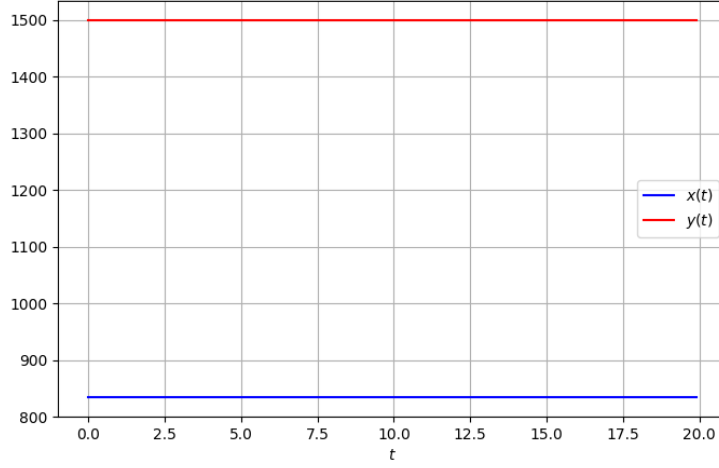


Рис. 4. графики траекторий $y(t)$ и $x(t)$ для начальных условий $x^{(0)} = 1500$, $y^{(0)} = 800\frac{1}{3}$

Разработка функции вычисления корня системы с помощью метода Ньютона

Функция `newton(x_0, f, J)` была написана на основе метода Ньютона для систем нелинейных алгебраических уравнений [1]. Согласно этому методу каждая следующая итерация находится по формуле:

$$\mathbf{x}^{(k)} = \mathbf{g}(\mathbf{x}^{(k)}) = \mathbf{x}^{(k-1)} - \mathbf{J}^{-1}(\mathbf{x}^{(k-1)})\mathbf{f}(\mathbf{x}^{(k-1)}), \quad (3)$$

где \mathbf{J} - матрица Якоби данной системы функций, составленная из частных производных этих функций по всем переменным.

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix} \quad (4)$$

Вместо вычисления обратной матрицы Якоби была использована двух-шаговая процедура:

$$\mathbf{J}(\mathbf{x}^{(k-1)})\mathbf{y}^{(k-1)} = \mathbf{f}(\mathbf{x}^{(k-1)}), \quad (5)$$

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \mathbf{y}^{(k-1)}, \quad (6)$$

где $\mathbf{y}^{(k-1)}$ находится через решение первого уравнения.

В качестве критерия остановки алгоритма было использовано ограничение на относительное улучшение: $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$.

Для программной реализации формул были использованы функции `lu()` и `solve()`, разработанные в ходе лабораторной работы №3.

Далее представлен листинг 2, содержащий код функции `newton(x_0, f, J)`, возвращающей корень векторной функции f с матрицей Якоби J и количество проведенных итераций, используя метод Ньютона.

Листинг 2. Функция вычисления корня системы с помощью метода Ньютона

```

1 def newton(x_0, f, J):
2     eps = 10**-8
3     x_1 = x_0
4     L, U, P = lu(J(x_1), True)
5     y_1 = solve(L, U, P, f(x_1))
6     x_k = x_1 - y_1
7     iter = 1
8     while np.linalg.norm(x_k - x_1, ord=np.inf) > eps:
9         iter += 1
10        x_1 = x_k
11        L, U, P = lu(J(x_1), True)
12        y_1 = solve(L, U, P, f(x_1))
13        x_k = x_1 - y_1
14    return x_k, iter

```

Разработка функции вычисления корня системы с помощью метода градиентного спуска

Функция `gradient_descent(x_0, f, J)` была написана на основе метода градиентного спуска [1]. Далее приведена формулировка этого метода.

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - t^{(k)} \frac{\mathbf{z}^{(k)}}{\|\mathbf{z}^{(k)}\|_2}, \quad (7)$$

$$\text{где } \mathbf{z}^{(k)} = \mathbf{J}^T(\mathbf{x}^{(k-1)}) \mathbf{f}(\mathbf{x}^{(k-1)}). \quad (8)$$

Оптимальное значение $t^{(k)}$ было найдено двумя методами. В качестве первого метода был использован алгоритм, описанный в лекциях [1]. В качестве второго метода был использован метод наискорейшего спуска:

$$t^{(k)} = \operatorname{argmin}_t h(t),$$

$$\text{где } h(t) = g(\mathbf{x}^{(k-1)} - t \frac{\mathbf{z}^{(k)}}{\|\mathbf{z}^{(k)}\|_2}), \text{ а } g(\mathbf{x}) = \langle \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}) \rangle. \quad (9)$$

В качестве критерия остановки было использовано ограничение на относительное улучшение: $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|_\infty < \epsilon$, где $\epsilon = 10^{-8}$.

Код разработанной функции `gradient_descent(x_0, f, J)` представлен далее в листинге 3. Функция возвращает корень векторной функции f с матрицей Якоби J и количество проведенных итераций, используя метод градиентного спуска.

Листинг 3. Функция вычисления корня системы с помощью метода градиентного спуска

```

1 def gradient_descent(x_0, f, J):
2     eps = 10 ** -8
3     x_1 = x_0
4     J_t = np.transpose(J(x_1))
5     z_k = J_t.dot(f(x_1))
6     t_k = count_t(x_1, z_k)
7     x_k = x_1 - t_k*z_k/(np.linalg.norm(z_k, ord=2))
8     iter = 1
9     while np.linalg.norm(x_k-x_1, ord=np.inf)> eps:
10         iter += 1
11         x_1 = x_k
12         J_t = np.transpose(J(x_1))
13         z_k = J_t.dot(f(x_1))
14         t_k = count_t(x_1, z_k)
15         x_k = x_1 - t_k * z_k / (np.linalg.norm(z_k, ord=2))
16     return x_k, iter

```

Анализ методов нахождения корней системы

Для анализа методов с помощью каждой из функций `gradient_descent` (листинг 3) и `newton` (листинг 2) были найдены стационарные позиции как нули заданной векторной функции $\mathbf{f}(\mathbf{x})$ для ряда начальных условий $x_i^{(0)} = 15i$, $y_i^{(0)} = 15j$, где $i, j = 0, \dots, 200$.

Для каждой полученной стационарной позиции была рассчитана её супремум - норма, в результате чего были получены матрицы супремум-норм для стационарных позиций, найденных с помощью функций `newton()` и `gradient_descent`, размерности 201×201 .

Далее для каждой полученной матрицы были сформированы графики линий уровня с заполнением, относительно значений $x_i^{(0)}$, $y_i^{(0)}$.

Далее на рисунках 5, 6 и 7 представлены соответственно линии уровня для метода Ньютона, для метода градиентного спуска с поиском оптимального

$t^{(k)}$ по алгоритму, представленному в лекциях и для метода градиентного спуска с поиском оптимального $t^{(k)}$ по алгоритму скорейшего спуска. По горизонтальной оси расположены начальные параметры $x^{(0)}$, по вертикальной оси - начальные параметры $y^{(0)}$. В синей зоне норма примерно равна 1500, в темно-синей зоне - примерно 0.

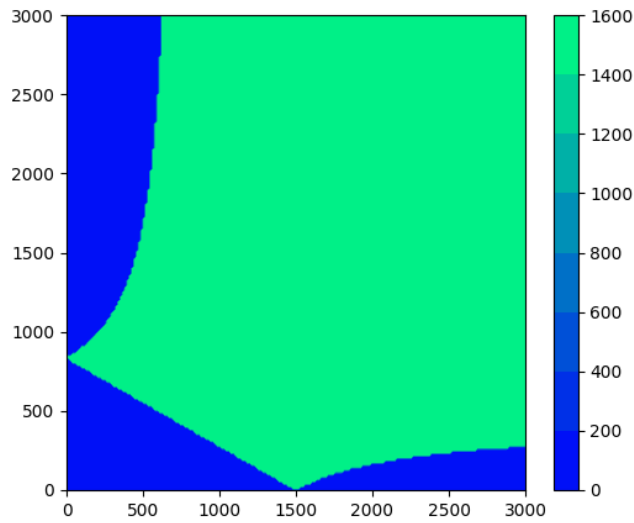


Рис. 5. Линии уровня для метода Ньютона

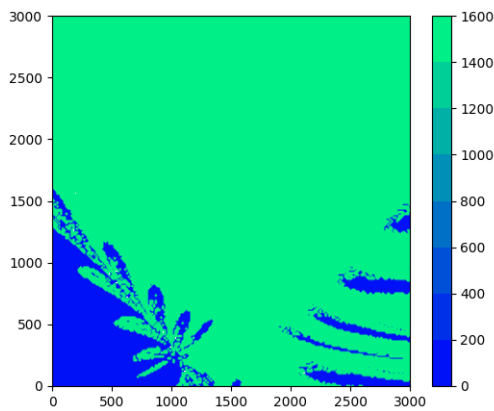


Рис. 6. Линии уровня для метода градиентного спуска с поиском оптимального $t^{(k)}$ по алгоритму, представленному в лекциях

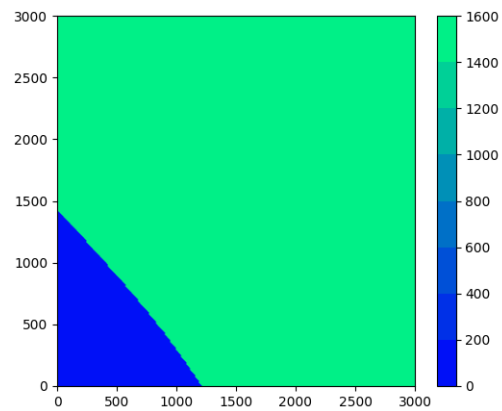


Рис. 7. Линии уровня для метода градиентного спуска с поиском оптимального $t^{(k)}$ по алгоритму скорейшего спуска

По рисунку 5 можно сделать вывод: метод Ньютона дает достаточно точное приближение к ответу в независимости от начального приближения.

По рисунку 6 можно сделать вывод о том, что метод градиентного спуска с поиском оптимального $t^{(k)}$ по алгоритму поиска шага с помощью аппроксимации функции шага параболлой, который представлен в лекциях, имеет погрешности, зависящие от начальных условий. Однако при поиске шага по алгоритму наискорейшего спуска алгоритм градиентного спуска становится более точным (рисунок 7).

Для каждого метода поиска корней системы были рассчитаны математическое ожидание и среднеквадратическое отклонение количества итераций.

Математическое ожидание рассчитывается как сумма всех количеств итераций деленная на количество экспериментов: $M_{iter} = \frac{\sum_{i=1}^n x_i}{n}$.

Среднеквадратическое отклонение рассчитывается по формуле :

$S_{iter}(x) = \sqrt{\frac{1}{n(n-1)} \cdot \sum_{i=1}^n (x_i - M_{iter})^2}$, где x_i - количество итераций в i -ом эксперименте, n - количество экспериментов.

Полученные результаты представлены в таблице 1.

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

	$M_{iter}(x)$	$S_{iter}(x)$
Метод Ньютона	6.6742	0.0086
Метод градиентного спуска с поиском шага первым способом	65.67441	7.32464
Метод градиентного спуска с поиском шага наискорейшим спуском	58.36304	0.26257

График для демонстрации сходимости методов строится путем вычисления λ для репрезентативного случая и соответствующего корня.

$$\lambda = \frac{|x^{k+1} - x^*|}{|x^k - x^*|^\alpha}, \quad (10)$$

где $\alpha = 1$ при линейной сходимости и 2 при квадратичной, x^* - точка, в которую сходится последовательность $x^{(k)}$.

Далее представлены графики сходимости для методов Ньютона и градиентного спуска с поиском шага наискорейшим спуском (рисунки 8, 9).

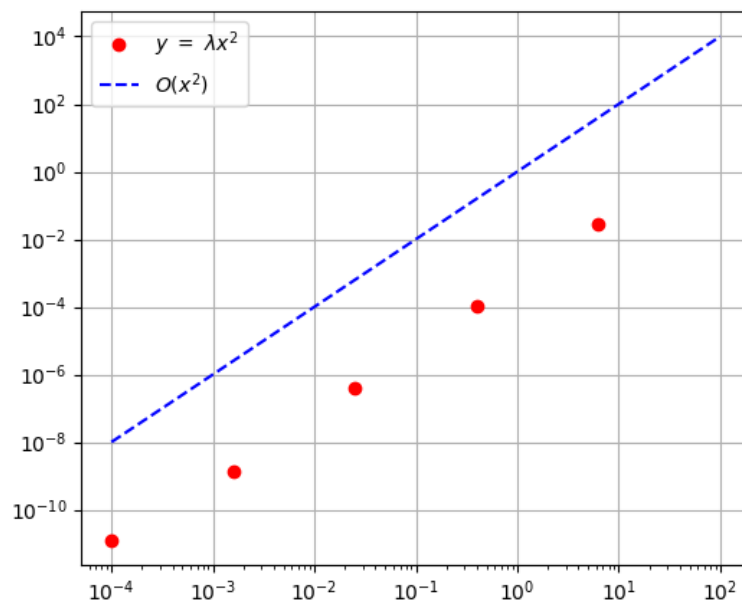


Рис. 8. График сходимости метода ньютона

Степень сходимости метода градиентного спуска:

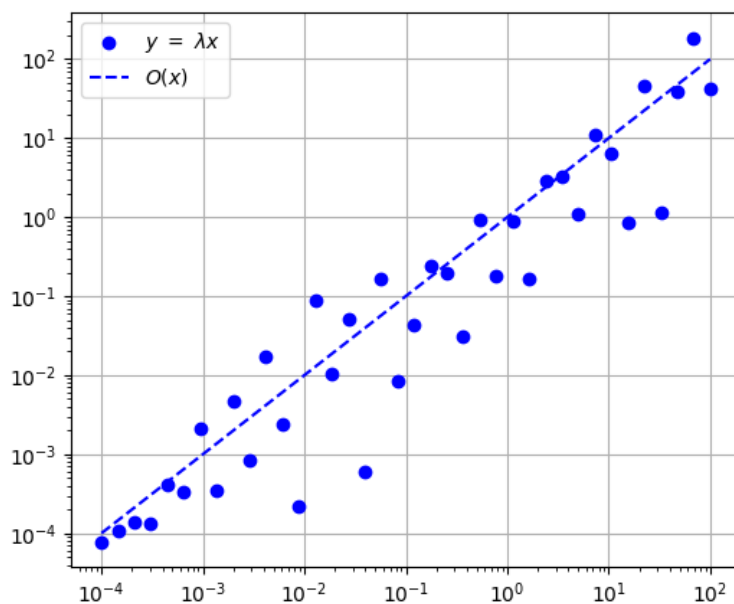


Рис. 9. График сходимости метода градиентного спуска

По графикам на рисунках 8, 9 можно сделать вывод, что метод Ньютона имеет квадратичную сходимость, в то время как метод градиентного спуска - линейную.

Итоги анализа методов

Метод Ньютона имеет квадратичную сходимость, что делает этот метод самым быстрым из рассматриваемых в работе. Математическое ожидание количества итераций метода Ньютона гораздо меньше, чем математическое ожидание для метода градиентного спуска. Но точность метода Ньютона зависит от начальных значений.

Метод градиентного спуска требует значительно большего количества итераций, что обуславливается его линейной сходимостью. Также на точность и количество итераций метода градиентного спуска влияет способ поиска значения шага.

Таким образом, для повышения точности и уменьшения количества итераций можно комбинировать два метода следующим образом. С помощью метода градиентного спуска необходимо находить приближенное значение,

и далее передавать его в качестве начального условия для метода Ньютона. Таким образом, количество итераций и точность будут оптимальными.

Заключение

1. В ходе лабораторной работы был реализован метод Рунге-Кутты 4-го порядка. С его помощью был получен фазовый портрет для модели Лотки-Вольтерры и проведен анализ этой модели. Был сделан вывод о том, что траектории популяций хищников и жертв колеблются вокруг стационарной точки, при этом колебания популяции хищников запаздывают относительно колебаний популяции жертв.
2. Были реализованы методы Ньютона и градиентного спуска для нахождения корней нелинейной алгебраической системы. На основе проведенного анализа методов были сделаны выводы о точности и сходимости методов. Также был предложен способ повысить производительность и точность методов



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

Новичкова М.А. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2022. — 17 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

Постановка:  доцент кафедры РК-6, PhD А.Ю. Першин
Решение и  студент группы РК6-56Б, Новичкова М.А.

вёрстка:

2022, осенний семестр