



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:	Новокшанов Евгений Андреевич
Группа:	РК6-56Б
Тип задания:	лабораторная работа
Тема:	Решение нелинейных систем алгебраических уравнений

Студент

подпись, дата

Новокшанов Е. А.
Фамилия, И.О.

Преподаватель

подпись, дата

Фамилия, И.О.

Москва, 2022

Решение нелинейных систем алгебраических уравнений		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
	Аналитический поиск стационарных позиций заданной системы ОДУ	7
	Анализ зависимости траекторий заданной системы от стационарных позиций	8
	Программная реализация метода метод Ньютона для решения систем нелинейных алгебраических уравнений	8
	Разработка функции, использующей метод градиентного спуска	10
	Анализ данных с использованием разработанных функций	11
	Анализ полученных результатов и сравнение свойств сходимости методов	17
Заключение		17

Решение нелинейных систем алгебраических уравнений

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 $rk4(x_0, t_n, f, h)$, 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: $\|x^{(k+1)} - x^{(k)}\|_\infty < \epsilon$, 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 $f(x)$ 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 $\log\log$ - графика.
6. Проанализировав полученные результаты, сравнить свойства сходимости метода Ньютона и метода градиентного спуска.

Цель выполнения лабораторной работы

Анализ методов нахождения корней нелинейной системы алгебраических уравнений на примере модели Лотки–Вольтерры.

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]:

$$\begin{aligned}
 \omega_0 &= \alpha, \\
 k_1 &= h f(t_i, \omega_i), \\
 k_2 &= h f\left(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_1\right), \\
 k_3 &= h f\left(t_i + \frac{h}{2}, \omega_i + \frac{1}{2}k_2\right), \\
 k_4 &= h f(t_i + h, \omega_i + k_3), \\
 \omega_{i+1} &= \omega_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad i = 0, 1, \dots, m-1.
 \end{aligned} \tag{2}$$

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(x_0, t_n, f, h):
2     t = np.arange(0, t_n, h)
3     n = len(t)
4     res_x = np.zeros((n, 2))
5     res_x[0] = x_0
6     for i in range(n-1):
7         k1 = h * f(res_x[i])
8         k2 = h * f(res_x[i] + 0.5 * k1)
9         k3 = h * f(res_x[i] + 0.5 * k2)
10        k4 = h * f(res_x[i] + k3)
11        res_x[i+1] = res_x[i] + (k1 + 2 * k2 + 2 * k3 + k4) / 6.
12    return res_x

```

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.

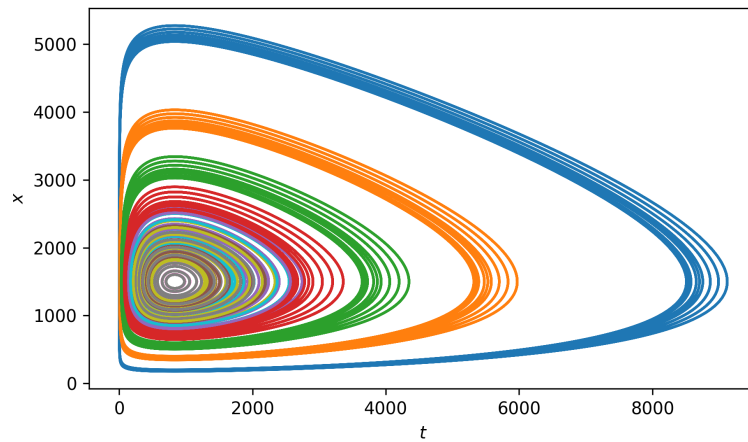


Рис. 1. Фазовый портрет траекторий для системы ОДУ (1) для ряда начальных условий

Приведенный Рисунок 1 демонстрирует колебание популяции хищников y и жертв x вокруг некоторой точки. Можно заметить, что колебание популяции хищников происходят с запаздыванием относительно колебаний популяции жертв. Так же при сильной разнице между начальными условиями будет сильно отличаться количество хищников и жертв всем времени моделирования.

Далее на рисунке 2 приведены графики траекторий популяции жертв $y(t)$ и хищников $x(t)$ для начальных условий $x^{(0)} = 400$, $y^{(0)} = 1600$. На Рисунке 2 подтверждаются сделанные выводы.

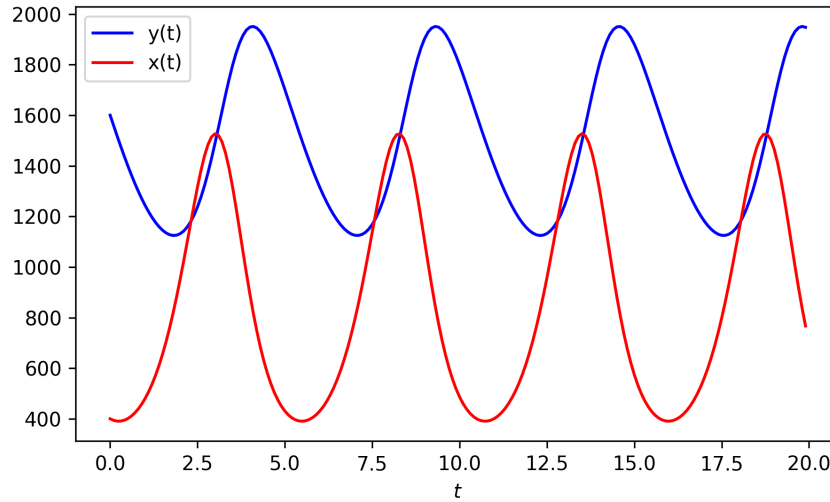


Рис. 2. Графики полученных траекторий $y(t)$ и $x(t)$ для начальных условий $x^{(0)} = 400$, $y^{(0)} = 1600$

2 Продвинутая часть

Аналитический поиск стационарных позиций заданной системы ОДУ

Стационарной позицией динамической системы является постоянная во времени траектория $x^*(t) = const$, являющаяся решением ОДУ вида $\frac{dx}{dt} = f(x)$. Для стационарной позиции $\frac{dx}{dt} = 0$, откуда следует, что стационарную позицию можно найти решив нелинейное уравнение $f(x) = 0$:

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

Следовательно, подставив $\alpha, \beta, \gamma, \delta$, получаем систему уравнений:

$$\begin{cases} 3x - 0.002xy = 0 \\ 0.0006yx - 0.5y = 0 \end{cases} \quad (4)$$

$$\begin{cases} (3 - 0.002y)x = 0 \\ (0.0006x - 0.5)y = 0 \end{cases} \quad (5)$$

Из последней системы(5) получаем 2 решения, 2 стационарные позиции: $[0; 0]^T$ и $[833.33; 1500]^T$.

Анализ зависимости траекторий заданной системы от стационарных позиций

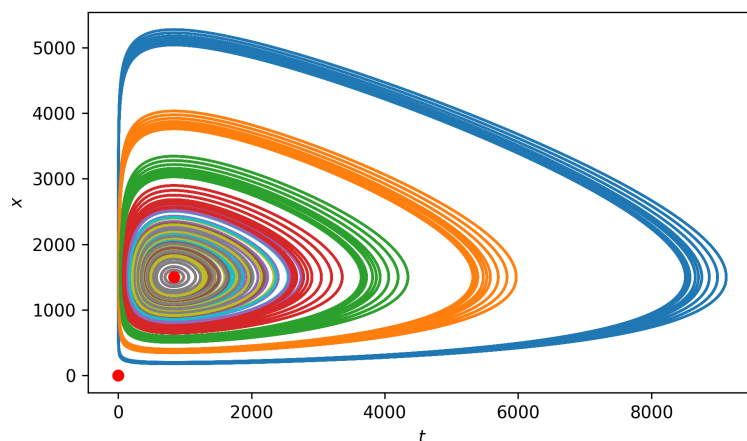


Рис. 3. Фазовый портрет траекторий для системы ОДУ (1) для ряда начальных условий с указанием найденных стационарных позиций

Стационарные позиции описывают модель, при которой изменение популяции не происходит. По рисунку 3 видно, что с течением времени траектория стремится к каждой из стационарных позиций, что говорит о том, что в итоге траектория сведется к не нулевой стационарной позиции, то есть изменение популяции не будет. Траектория не стремится к нулевой стационарной позиции, так как популяция никогда не станет равна 0, при не нулевых начальных условиях.

Программная реализация метода метод Ньютона для решения систем нелинейных алгебраических уравнений

Для написания функции $newton(x_0, f, J)$ был использован метод Ньютона для систем нелинейных алгебраических уравнений [1]. Формулировка метода приведена ниже в формуле 6.

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

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

следующим образом:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{bmatrix} \quad (7)$$

Для избежания временных затрат на вычисление обратной матрицы Якоби был применен и реализован следующий алгоритм из двух действий:

$$\mathbf{J}(\mathbf{x}^{(k-1)})\mathbf{y}^{(k-1)} = \mathbf{f}(\mathbf{x}^{(k-1)}), \quad (8)$$

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \mathbf{y}^{(k-1)}, \quad (9)$$

где $\mathbf{y}^{(k-1)}$ находится через решение первого уравнения. Для реализации алгоритма, построенного на формулах 8 и 9 были использованы функции $lu(A, permute)$ и $solve(L, U, P, b)$, которые были разработаны в лабораторной работе №3.

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

Листинг 2. Функция `newton`, разработанная с помощью метода Ньютона

```

1  def newton(x_0, f, J):
2      E = 1e-8
3      x = []
4      x.append(x_0)
5      L, U, P = lu(J(x_0), True)
6      Y = solve(L, U, P, f(x_0))
7      x_next = x_0 - Y
8      x.append(x_next)
9      i = 1
10     while(abs(np.linalg.norm(x[i]-x[i-1], ord=np.inf)) > E):
11         L, U, P = lu(J(x[i]), True)
12         Y = solve(L, U, P, f(x[i]))
13         x_next = x[i] - Y
14         x.append(x_next)
```

```

15     i += 1
16     return len(x)-1, x[-1], k

```

Разработка функции, использующей метод градиентного спуска

В ходе программной реализации метода градиентного спуска была разработана функция *gradient_descent*(*x_0*, *f*, *J*). Для этого был использован метод градиентного спуска [1], алгоритм которого расписан ниже (10).

$$\mathbf{z}^{(k)} = \mathbf{J}^T(\mathbf{x}^{(k-1)})\mathbf{f}(\mathbf{x}^{(k-1)}), \quad \mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} - \mathbf{t}^{(k)} \frac{\mathbf{z}^{(k)}}{\|\mathbf{z}^{(k)}\|_2}; \quad (10)$$

При этом, для нахождения значения шага $\mathbf{t}^{(k)}$ используется алгоритм "дробного шага" приведенный в лекциях [1].

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

В листинге 3 приведена программная реализация метода градиентного спуска. Функция *gradient_descent*(*x_0*, *f*, *J*) возвращает корень векторной функции *f* с матрицей Якоби *J* и количество проведенных итераций. Входными значениями являются начальное условие *x_0*, а также аргументы *f* и *J*, где *f* и *J* - функции, принимающие на вход вектор *x* и возвращающие соответственно вектор $\overrightarrow{f(x)}$ и матрицу *J*(*x*).

Листинг 3. Функция *gradient_descent*, разработанная с помощью метода градиентного спуска

```

1  def gradient_descent(x_0, f, J):
2      eps = 1E-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 = tSearch(x_1, z_k)
7      x_k = resSearch(t, x_1, z_k)
8      count = 1
9      while np.linalg.norm(x_k-x_1, ord=np.inf)> eps:
10         count += 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_res = tSearch(x_1, z_k)
15         x_k = resSearch(t_res, x_1, z_k)
16     return count, x_k

```

Так же были рассмотрены несколько методов выбора оптимального шага на каждой итерации:

- Метод градиентного спуска с дроблением шага:
Этот метод заключается в аппроксимировании параболой зависимости целевой функции от шага при фиксированном направлении поиска, и дальнейшем аналитическом вычислении экстремума полученной параболы.
- Метод наискорейшего спуска:
Этот метод заключается в нахождении такого шага интегрирования из набора данных, при котором значение $g(x)$ уменьшается больше всего.

Анализ данных с использованием разработанных функций

Поиск стационарных позиций как нулей заданной функции с рядом начальных условий

Поиск стационарной позиции как нуля заданной функции проводится с помощью программной реализации метода Ньютона и метода градиентного спуска, описанных выше в листингах 2 и 3 соответственно. Начальный набор данных, описанный в условии, генерируется в цикле, как показанов в следующем листинге.

Листинг 4. Генерация начального набора данных

```
1 start_data = np.array([15 * i for i in range(0, 201)])
```

Расчет матрицы супремум-норм

Расчет матриц супремум-норм для каждого из методов, реализован в виде двойного цикла по начальным данным, которые были сгенерированы ранее (5).

Листинг 5. Заполнение матрицы супремум-норм

```
1 for x_0 in range(n):
2     for y_0 in range(n):
3         count += 1
4         os.system('CLS')
5         ne_start = time.time()
```

```

6     print(f"Осталось: {100 - count/(n**2)*100 :.1f}%, прошло{(ne_start -
      start)/60:.3f} минут, ")
7     c, x, k = newtonV2(np.array([start_data[x_0], start_data[y_0]]), f, J)
8     sum_newton.append(c)
9     res2[x_0][y_0] = np.linalg.norm(x, ord=np.inf)
10    c, x, k = gradient_descent_M(np.array([start_data[x_0], start_data[y_0]]), f, J)
11    sum_grad.append(c)
12    res[x_0][y_0] = np.linalg.norm(x, ord=np.inf)

```

Вывод полученных данных в виде линий уровня

Для метода Ньютона были получены следующие линии уровня:

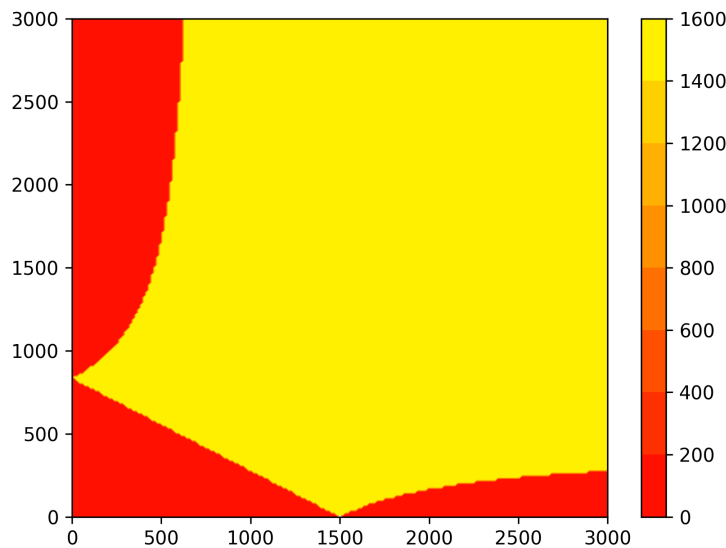


Рис. 4. Линии уровня для матрицы полученной с помощью метода Ньютона

Здесь и далее красный цвет соответствует начальным условиям, при которых корнем ОДУ является $[0; 0]$, а желтый цвет соответствует начальным условиям при которых корнем ОДУ является $[833.33; 1500]$. Так как при программной реализации метода градиентного спуска были разработаны несколько методов поиска оптимального шага, то для каждого из них получены разные результаты. Для метода наискорейшего спуска, где значения шага ищутся в интервале от 2^{-30} до 2^{30} , получена следующая визуализация полученных данных:

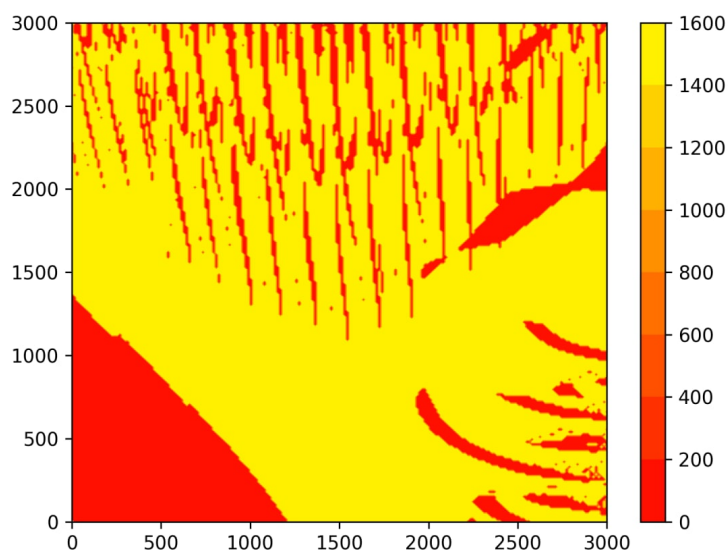


Рис. 5. Линии уровня для матрицы полученной с методом градиентного спуска(наискорейший спуск)

Для метода наискорейшего спуска, где значения шага ищутся в интервале от 2^{-27} до 2^3 , получена следующая визуализация полученных данных:

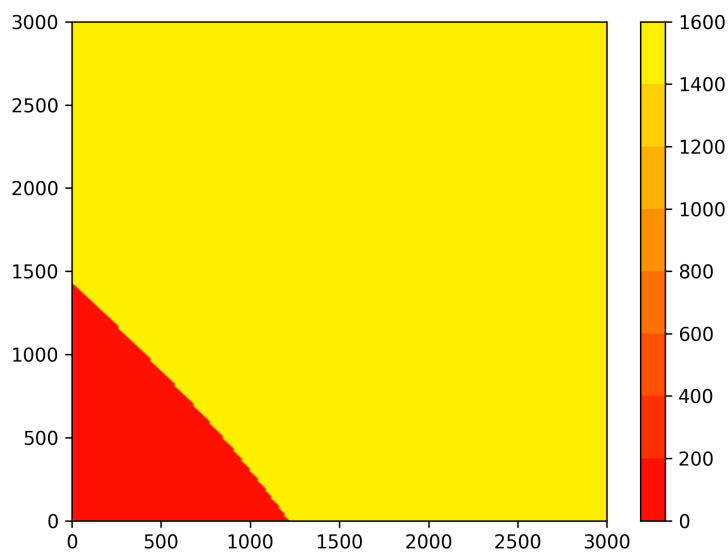


Рис. 6. Линии уровня для матрицы полученной с методом градиентного спуска(наискорейший спуск с малым шагом)

Для метода градиентного спуска с дроблением шага получена следующая визуализация полученных данных:

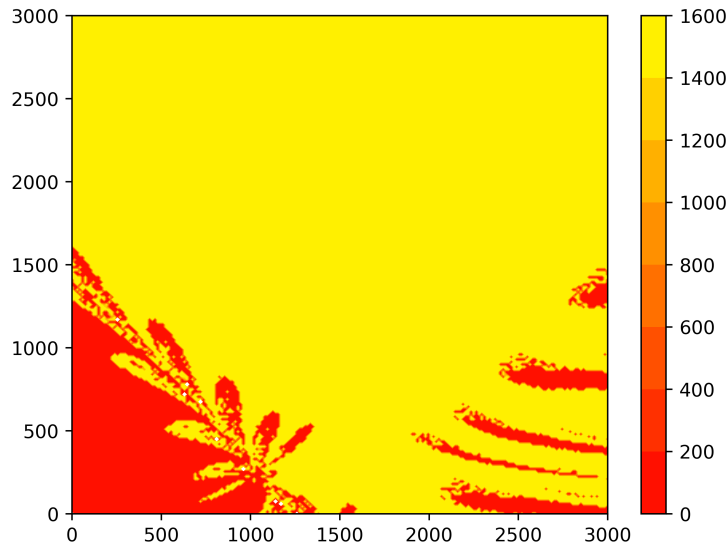


Рис. 7. Линии уровня для матрицы полученной с методом градиентного спуска с дроблением шага

Описание наблюдений

Опираясь на полученные данные, можно сделать вывод, что метод Ньютона и наискорейшего спуска с малым шагом формируют матрицу с наименьшим количеством отклонений и ошибок. Метод градиентного спуска с дроблением шага показывает сильную зависимость от начальных условий, о чем свидетельствуют "лепестки". Метод наискорейшего спуска с большим интервалом значений показывает самый плохой результат: из-за слишком высокого шага многие начальные условия x_0 решение ОДУ, несмотря на свою удаленность от стационарной позиции $[0; 0]^T$ и близость к $[833.33; 1500]^T$, сходится к $[0; 0]^T$.

Математическое ожидание и среднеквадратическое отклонение количества итераций

Математическое ожидание:

- Метод Ньютона: 6.674 итерации
- Метод градиентного спуска:
 - Наискорейший спуск с большим интервалом: 42.782 итерации
 - Наискорейший спуск с малым шагом: 201.291 итерации
 - Дробный шаг: 82.592 итерации

Среднеквадратическое отклонение:

- Метод Ньютона: 0.008
- Метод градиентного спуска:
 - Наискорейший спуск с большим интервалом: 0.103
 - Наискорейший спуск с малым шагом: 0.423
 - Дробный шаг: 5.067

Демонстрация степени сходимости метода Ньютона и градиентного спуска

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

$$\lambda = \frac{|x^{k+1} - x^*|}{|x^k - x^*|^\alpha}, \quad (11)$$

где $\alpha = 1$ при линейной сходимости и 2 при квадратичной. Степень сходимости метода Ньютона:

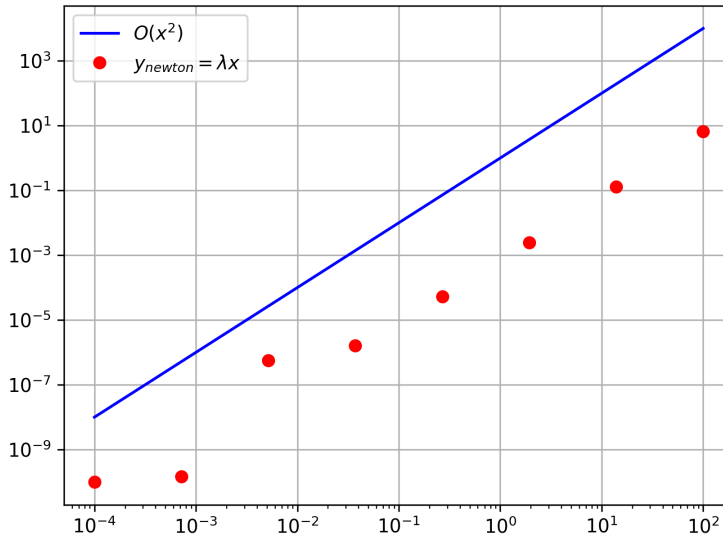


Рис. 8. График сходимости метода ньютона

Степень сходимости метода градиентного спуска:

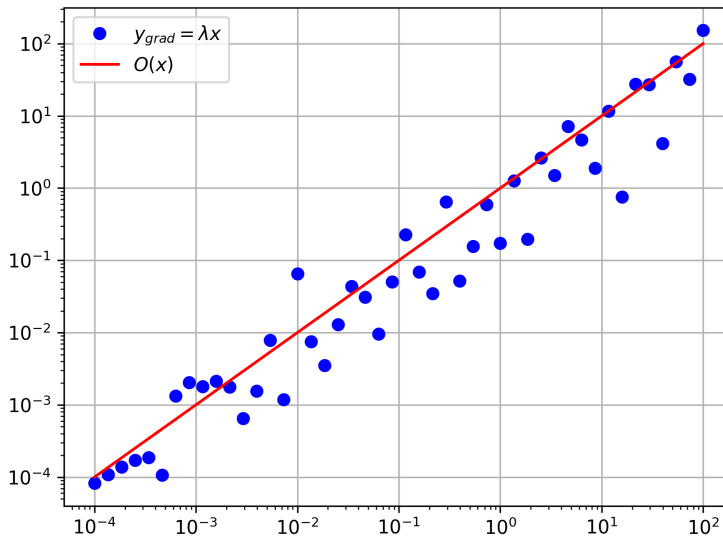


Рис. 9. График сходимости метода градиентного спуска

Данные графики 9 и 8 подтверждают, что метод Ньютона имеет квадратичную сходимость, а метод градиентного спуска - линейную.

Анализ полученных результатов и сравнение свойств сходимости методов

Проанализировав полученные результаты, можно сделать следующий вывод:

Метод Ньютона является самым точным и быстрым методом среди всех рассмотренных методов. Это достигается за счет того, что метод обладает квадратичной сходимостью, что обеспечивает высокую производительность и точность вычислений даже при малом шаге. Метод градиентного спуска требует больше итераций на каждом шаге за счет линейной сходимости метода. Метод градиентного спуска с дроблением шага избавляет нас от проблемы выбора $t^{(k)}$ на каждом шаге, заменяя на проблему выбора ϵ, δ и $t^{(0)}$, к которым градиентный метод менее чувствителен. Из-за этого мы получали матрицы супремум-норм, вычисленные методом дробления шага, с меньшим количеством отклонений, чем матрица вычисленная методом наискорейшего спуска с большим интервалом, так как наискорейший спуск напрямую влияет на выбор $t^{(k)}$, к которому градиентный спуск очень чувствителен.

Заключение



1. Была разработана функция реализующая метод Рунге-Кутты 4-го порядка и проанализирован фазовый портрет для модели Лотки-Вольтерры;
2. Были найдены стационарные позиции заданной ОДУ и описана зависимость траектории показанной на фазовом портрете от этих позиций;
3. Были программно реализованы методы Ньютона и градиентного спуска, а так же проанализирована их точность и производительность;
4. Были проанализированы полученные результаты, а так же проведено сравнение методов на основе 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

Новокшанов Е. А. Отчет о выполнении лабораторной работы по дисциплине «Вычислительная математика». [Электронный ресурс] — Москва: 2022. — 18 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, Новокшанов Е.
вёрстка: А.

2022, осенний семестр