ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Кафедра прикладной математики и искусственного интеллекта
Направление подготовки - «Прикладная математика»

КУРСОВАЯ РАБОТА

по дисциплине: «Численные методы»

Система типа хищник-жертва. Модель Холлинга-Тернера.

Студент 2 курса
группы 09-221
«____» ________ 2024 г. ______________ М.А. Саитов
Научный руководитель
ассистент б.с.
«____» ________ 2024 г. ______________ О.В. Глазырина

Казань, 2024 год

Содержание

1 Цель работы

Целью работы является исследование модели взаимодейстия "Хищник - жертва", выявление зависимостей поведения модели в зависимости от значения параметров, описывающих систему, а так же применение метода

2 Теоретические основы выполнения работы

Модель Вольтерра используется для представления взаимодейстия двух типов. При моделировании системы "Хищник — Жертва" будем учиывать следующие ограничения:

1. Жертва может найти достаточно пищи для пропитания
2. При каждой встрече с хищником последний убивает жертву
3. Норма рождаемости жертв x_b, нормы естественной смертности жертв x_d и хищников c являются постоянными, $a = x_b - x_d > 0$.
4. Число случаев, когда хищник убивает жертву, зависит от вероятности их встречи и, следовательно, пропорционально произведению xy.
   В результате встреч с жертвами число хищников увеличивается.

Обозначим количества хищников и жертв в момент времени t через $y = y(t)$ и $x = x(t)$ соответственно. Тогда с учетом сделанных допущений, модель можно записать в виде:

$$\frac{dx}{dt} = ax - bxy$$
$$\frac{dy}{dt} = -cy + dxy \tag{1}$$

где коэффициенты b, d - коэффициенты убийства жертв и воспроизводства хищников, a - описывает рождаемость жертв, c - ествестенную смерть жертв.

Произведя в системе (1) замены:

$$X = \frac{d}{a}x,\ \ Y = \frac{b}{a}y,\ \ \tau = at,\ \ \sigma = \frac{c}{a} \tag{2}$$

уравнение (1) преобразуется к виду:

$$\frac{dX}{d\tau} = X - XY$$
$$\frac{dY}{d\tau} = -\sigma Y + XY \tag{3}$$

с начальными условиями:

$$X(0) = X_0,\ \ Y(0) = Y_0 \tag{4}$$

описывающими количество особей каждого из вида.

3 Метод Рунге-Кутты

Для решения систем дифференциалных уравнений будем использовать метод Рунге – Кутты 4-го порядка точности:

$$\begin{aligned} k_1 &= f(t_n, y_n), \\ k_2 &= f(t_n + \frac{h}{3}, y_n + \frac{hk_1}{3}), \\ k_3 &= f(t_n + \frac{2h}{3}, y_n - \frac{hk_1}{3} + hk_2), \\ k_4 &= f(t_n + h, y_n + h(k_1 - k_2 + k_3)), \\ y_{n+1} &= y_n + \frac{h(k_1 + 3k_2 + 3k_3 + k_4)}{8} \end{aligned} \tag{5}$$

где $y'(t, y) = k(t, y)$. Задано начльное условие $y(0) = y_0$. Этот метод является методом с постоянным шагом h на отрезке $[a, b]$, значения решения вычисляются в $n = \frac{b-a}{h}$ точках.

Используя приведенный выше метод решить тестовую задачу:

$$y_1' = \frac{y_1}{2 + 2t} - 2ty_2, \quad y_2' = \frac{y_2}{2 + 2t} + 2ty_1, \tag{6}$$

на отрезке [0, 2].

4 Задание

В ходе выполнения работы необходимо:

1. Решить тестовую задачу (6), написать процедуру интегрирования на произвольном отрезке $[a, b]$

2. Для тестовой задачи (6) построить графики зависимости максимальной погрешности решения e и $\frac{e}{h^4}$ от шага h.

3. Для двух наборов начальных условий (4) и нескольких значений параметра σ расчитать динамику популяции. Привести графики решений в координатах (X, Y) и дать их интерпритацию.

5 Решение тестовой задачи

Тестовый вариант задачи будет решен методом Рунге - Кутты 4-го порядка точности. Дана система уравнений (6) с известным точным решением:

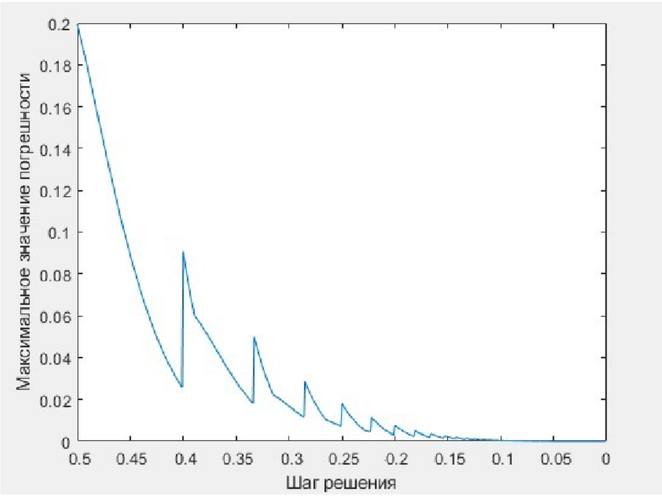
$$y_1 = \cos(t^2)\sqrt{1+t}, \quad y_2 = \sin(t^2)\sqrt{1+t} \tag{7}$$

Необходимо решить систему с начальными условиями $y_1(0) = 0, y_2(0) = 1$ на отрезке [0; 2]. Перед вычислением коэффициентов стоит заметить, что $y_1' = y_1'(y_1, y_2, t)$ и $y_2' = y_2'(y_1, y_2, t)$, тогда:

$$\begin{aligned}
k_{11} &= y_1'(y_1(i), y_2(i), t(i)) \\
k_{12} &= y_2'(y_1(i), y_2(i), t(i)) \\
k_{21} &= y_1'(y_1(i) + \frac{h}{2}k_{11}, y_2(i) + \frac{h}{2}k_{12}, t(i) + \frac{h}{2}) \\
k_{22} &= y_2'(y_1(i) + \frac{h}{2}k_{11}, y_2(i) + \frac{h}{2}k_{12}, t(i) + \frac{h}{2}) \\
k_{31} &= y_1'(y_1(i) + \frac{h}{2}k_{21}, y_2(i) + \frac{h}{2}k_{22}, t(i) + \frac{h}{2}) \\
k_{32} &= y_2'(y_1(i) + \frac{h}{2}k_{21}, y_2(i) + \frac{h}{2}k_{22}, t(i) + \frac{h}{2}) \\
k_{41} &= y_1'(y_1(i) + hk_{31}, y_2(i) + hk_{32}, t(i) + h) \\
k_{42} &= y_1'(y_1(i) + hk_{31}, y_2(i) + hk_{32}, t(i) + h)
\end{aligned} \tag{8}$$

где i = 1, . . . , $\frac{b-a}{h} - 1$

Решив таким образом систему, составим графики зависимости максимальной ошибки решения от шага h и h^4:



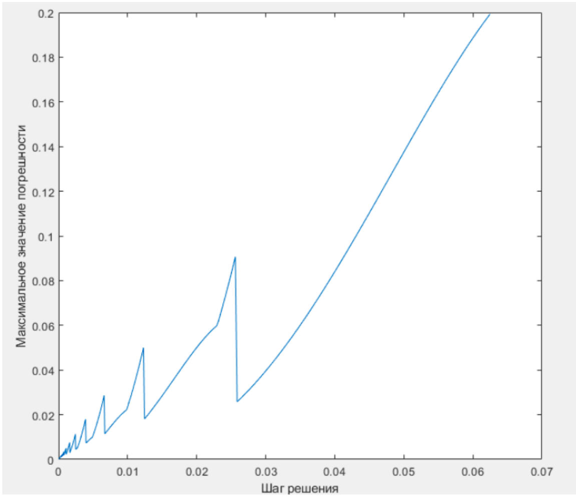


Рис.1 — Зависимость точности решения от изменения шага h.

Исходя из вида графиков, описывающих зависимость точности решения, можно сделать вывод о зависимости точности решения от шага разбиения.

6 Модель Вольтерра

Теперь мы можем, используя правило (5) решить систему (3). Выберем начальные условия $X_0 = 3, Y_0 = 0.5$, начальное число жертв и хищников, соответственно. Исходя из вида системы, отметим, что динамика популяции зависит от параметра $\sigma = \frac{c}{a}$ - отношения смертности хищников к приросту жертв. Выбрав $\sigma = 1$, т.е. установив равное отношение прироста жертв к смертности хищников получим график описывающий размеры популяций жертв и хищников с течением времени (Рис.2). Размер популяции жертв в начальный момент превышает размер популяции хищников, из-за чего веротность встречи представителей двух видов возрастает, хищник убивает жертву. С уменьшением количества жертв, количество встреч с хищниками убывает. Когда хищники не охотятся, их популяция начинает вымирать. Уменьшение популяции хищников уменьшает вероятность встреч с жертвами, что для последних, при наличии бесконечного числа пищи является причиной резкого увеличения числа представителей вида. Резкое увеличение числа жертв предполагает возрастание числа хищников. Система приходит в циклиеское, устойчивое состояние.

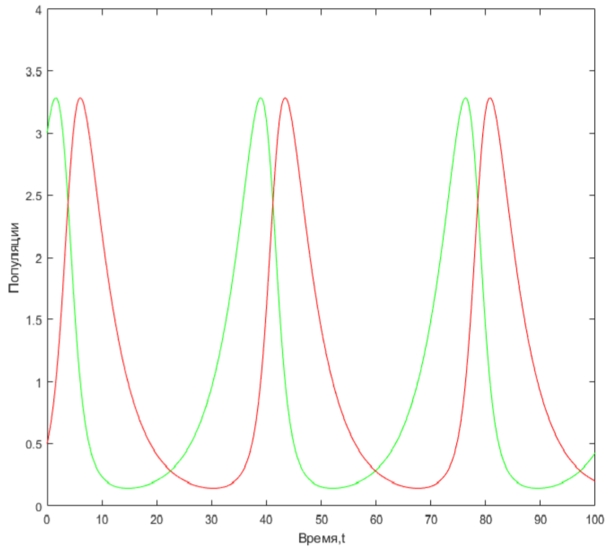


Рис.2 — Размер популяций жертв (зеленый) и хищников (красный) с течением времени.

Теперь выберем $\sigma = 3$, что определит троекратное увеличение убыли хищников к росту популяций жертв. Поведение популяций видов изменилось, численность популяций изменяется не так сильно, однако, за тот же промежуток времени происходит большее число "колебаний" числа представителей вида (Рис.3).

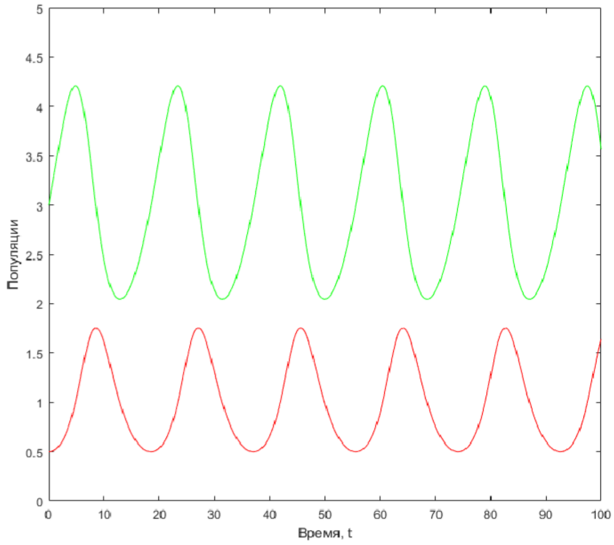


Рис.3 — Изменение популяций видов при $\sigma = 3$.

Применим те же параметры для системы с начальными условиями $X_0 = 1, Y_0 4$, что соответствует ситуации, когда в начальный момент число хищников в 4 раза превышает количество жертв. Тенденция стаблизации системы соханяется (Рис.3), однако при значении $\sigma = 3$ популяция хищников убывает сильнее из-за малого количества доступной изначально пищи и значения коэффициента.

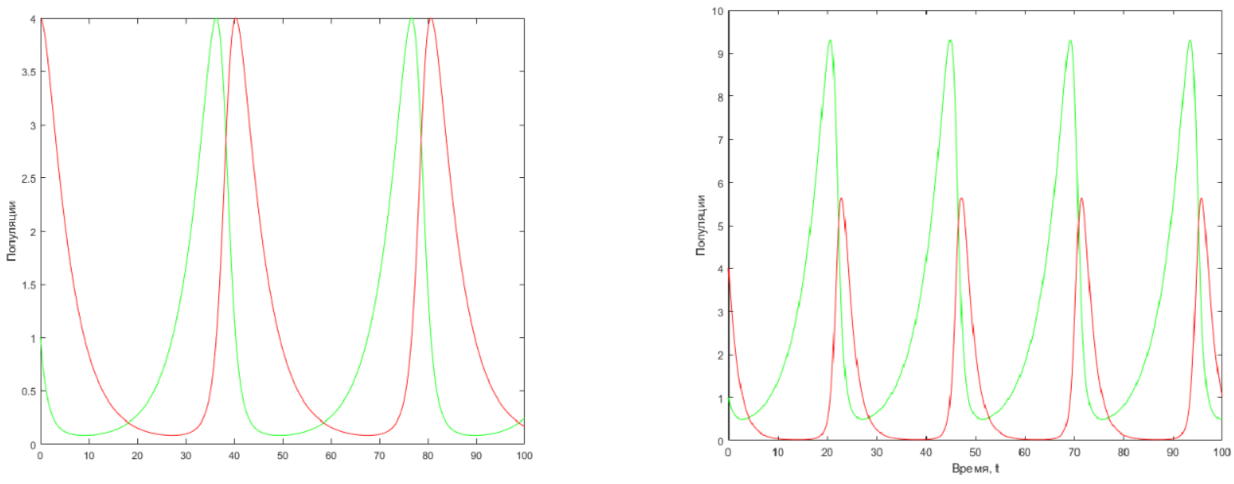


Рис.4 — Вид популяции при других начальных условиях.

Представим графики в координатах (X, Y) для двух расмотренных начальных условий (Рис. 5).

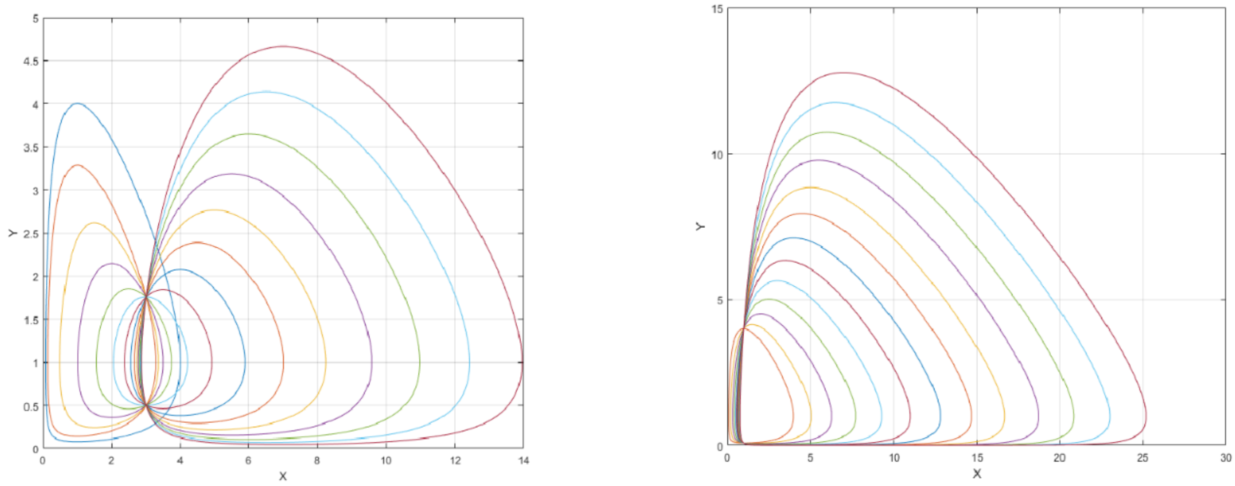


Рис.5 — Графики (X, Y) в зависимости от параметра σ. Линии проведены для значений лежащих в отрезке $[1, 7]$ через 0.5.

7 Заключение

Модель Вольтерра может быть использована для моделирования системы типа "Хищник–жертва". Наблюдая за моделью можно выявить основные закономерности поведения популяций. В ходе работы были рассмотрены модели с различными значениями начального размера популяций, коэффициентов определяющих отношение убыли одного вида к приросту другого. При выполнении задачи был использован метод Рунге–Кутты для решения систем дифференциалных уравнений. Метод позволяет решать уравнения с достаточной степенью точности, достаточной для корректного моделирования системы.

8 Список литературы

1. Глазырина Л.Л., Карчевский М.М. Численные методы: учебное пособие. — Казань: Казан. ун-т, 2012. — 122

2. Глазырина Л.Л.. Практикум по курсу «Численные методы». Решение систем линейных уравнений: учеб. пособие. — Казань: Изд-во Казан. ун-та, 2017. — 52 с.

9 Листинг программы

```cpp

#include <vector>
#include <cmath>
#include <iostream>
#include <algorithm>
#include <functional>
#include <numeric>
#include <iomanip>
#include <map>
#include <string>
#include <iterator>
#include <cassert>
#include <fstream>

int main() {
    double h = 0.0001;
    double a = -10;
    double b = -5;
    int N = static_cast<int>((b - a) / h);
    std::vector<double> x(N);
    std::iota(x.begin(), x.end(), a);
    std::transform(x.begin(), x.end(), x.begin(), [h](double xi) { return xi * h; });

    auto y1 = [](double x, double y2) { return -y2 / x; };
    auto y2 = [](double x, double y1) { return -y1 / x; };

    std::vector<double> y1Vals(N, 0);
    std::vector<double> y2Vals(N, 0);
    y1Vals[0] = -10.0 / 3;
    y2Vals[0] = 10.0 / 3;

    for (int i = 0; i < N - 1; ++i) {
        double k11 = y1(x[i], y2Vals[i]);
        double k12 = y2(x[i], y1Vals[i]);
        double k21 = y1(x[i] + h / 4, y2Vals[i] + h * k12 / 4);
        double k22 = y2(x[i] + h / 4, y1Vals[i] + h * k11 / 4);
        double k31 = y1(x[i] + h / 2, y2Vals[i] + h * k22 / 2);
        double k32 = y2(x[i] + h / 2, y1Vals[i] + h * k21 / 2);
        double k41 = y1(x[i] + h, y2Vals[i] + h * k12 - 2 * h * k22 + 2 * h * k32);
        double k42 = y2(x[i] + h, y1Vals[i] + h * k11 - 2 * h * k21 + 2 * h * k31);
        y1Vals[i + 1] = y1Vals[i] + h * (k11 + 4 * k31 + k41) / 6;
        y2Vals[i + 1] = y2Vals[i] + h * (k12 + 4 * k32 + k42) / 6;
    }

    // Plotting using a file
    std::ofstream plotFile("plot.dat");
    for (int i = 0; i < N; ++i) {
        plotFile << x[i] << " " << y1Vals[i] << " " << y2Vals[i] << "\n";
```

```
    }
    plotFile.close();

    // Error analysis part
    std::vector<double> h_values;
    for (double hi = 0.5; hi >= 0.001; hi -= 0.001) {
        h_values.push_back(hi);
    }
    int L = h_values.size();
    std::vector<double> errors(L, 0);
    int j = 0;

    auto y1_correct = [](double x) { return x / 3; };
    auto y2_correct = [](double x) { return -x / 3; };

    for (double hi : h_values) {
        int Ni = static_cast<int>((b - a) / hi);
        std::vector<double> xi(Ni);
        std::iota(xi.begin(), xi.end(), a);
        std::transform(xi.begin(), xi.end(), xi.begin(), [hi](double
           xii) { return xii * hi; });

        std::vector<double> y1Vals(Ni, 0);
        std::vector<double> y2Vals(Ni, 0);
        y1Vals[0] = -10.0 / 3;
        y2Vals[0] = 10.0 / 3;

        for (int i = 0; i < Ni - 1; ++i) {
            double k11 = y1(xi[i], y2Vals[i]);
            double k12 = y2(xi[i], y1Vals[i]);
            double k21 = y1(xi[i] + hi / 4, y2Vals[i] + hi * k12 / 4)
               ;
            double k22 = y2(xi[i] + hi / 4, y1Vals[i] + hi * k11 / 4)
               ;
            double k31 = y1(xi[i] + hi / 2, y2Vals[i] + hi * k22 / 2)
               ;
            double k32 = y2(xi[i] + hi / 2, y1Vals[i] + hi * k21 / 2)
               ;
            double k41 = y1(xi[i] + hi, y2Vals[i] + hi * k12 - 2 * hi
                * k22 + 2 * hi * k32);
            double k42 = y2(xi[i] + hi, y1Vals[i] + hi * k11 - 2 * hi
                * k21 + 2 * hi * k31);
            y1Vals[i + 1] = y1Vals[i] + hi * (k11 + 4 * k31 + k41) /
               6;
            y2Vals[i + 1] = y2Vals[i] + hi * (k12 + 4 * k32 + k42) /
               6;
        }

        std::vector<double> y1Ans(Ni), y2Ans(Ni);
        std::transform(xi.begin(), xi.end(), y1Ans.begin(),
           y1_correct);
        std::transform(xi.begin(), xi.end(), y2Ans.begin(),
           y2_correct);
```

```cpp

        double er1 = *std::max_element(y1Vals.begin(), y1Vals.end())
           - *std::max_element(y1Ans.begin(), y1Ans.end());
        double er2 = *std::max_element(y2Vals.begin(), y2Vals.end())
           - *std::max_element(y2Ans.begin(), y2Ans.end());
        errors[j] = std::max(er1, er2);
        j++;
    }

    std::vector<double> er(L);
    for (int i = 0; i < L; ++i) {
        er[i] = errors[i] / std::pow(h_values[i], 4);
    }

    // Plotting error data using a file
    std::ofstream errorPlotFile("error_plot.dat");
    for (int i = 0; i < L; ++i) {
        errorPlotFile << h_values[i] << " " << er[i] << "\n";
    }
    errorPlotFile.close();
}
```