Отчёт по лабораторной работе

Лабораторная работа №5 (вариант 10)

Сергеев Тимофей Сергеевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Для модели «хищник-жертва» постройте график зависимости численности хищников от численности жертв, а также графики изменения численности хищников и численности жертв при заданных начальных условиях. Найдите стационарное состояние системы.

2 Задание

- Написать код на языке Julia.
- Написать код на языке Modelica для случаев.
- Составить отчёт на языке Markdown и сконвертировать его в docx и pdf.
- Подготовить презентацию на языке Markdown и защитить её.

3 Теоретическое введение

Julia – высокоуровневый высокопроизводительный свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков (например, MATLAB и Octave), однако имеет некоторые существенные отличия. Julia написан на Си, C++ и Scheme. Имеет встроенную поддержку многопоточности и распределённых вычислений, реализованные в том числе в стандартных конструкциях. [1]

OpenModelica – свободное открытое программное обеспечение для моделирования, симуляции, оптимизации и анализа сложных динамических систем. Основано на языке Modelica. Активно развивается Open Source Modelica Consortium, некоммерческой неправительственной организацией. Open Source Modelica Consortium является совместным проектом RISE SICS East AB и Линчёпингского университета. [2]

Система «хищник — жертва» — сложная экосистема, для которой реализованы долговременные отношения между видами хищника и жертвы, типичный пример коэволюции. Отношения между хищниками и их жертвами развиваются циклически, являясь иллюстрацией нейтрального равновесия. [3]

4 Выполнение лабораторной работы

1. Рассмотрим код на языке Modelica. Объявим переменные и коэффициенты типа Real (потому что это тип с плавающим знаком, наиболее подходящий для решения дифференциальных уравнений). Затем инициализурем x и y, подставив данные из условия. После этого пропишем решение наших дифференциальных уравнений (рис. 4.1).

```
model lab5_1
Real x, y;
Real a=0.22;
Real b=0.051;
Real c=0.33;
Real d=0.041;
Real t=time;
initial equation
x=3;
y=8;
equation
der(x)= -a*x + b*x*y;
der(y)= c*y - d*x*y;
end lab5_1;
```

Рис. 4.1: Реализация модели на языке Modelica

2. Затем установим настройки симуляции (начальное (0) и конечное (30) время и шаг (0.1)) и запустим симуляцию. Получим следующий результат (рис. 4.2).

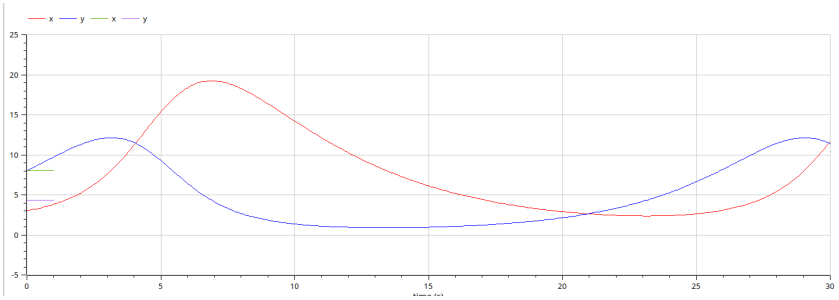


Рис. 4.2: Результат моделирования на языке Modelica

3. Затем необходимо вывести фазовый портрет. В Open Modelica для этого необходимо открыть новое окно параметрического вывода график, нажать Shift и, удерживая его, поставить галочку у x, после чего поставить галочку у y. В результате на экране появится фазовый портрет (рис. 4.3).

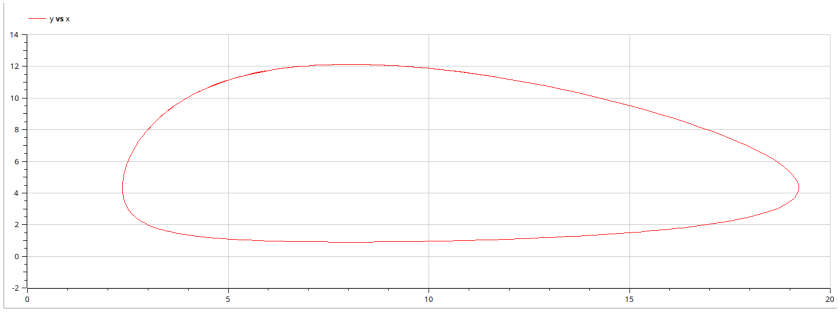


Рис. 4.3: Фазовый портрет на языке Modelica

4. Найдём стационарное состояние системы, заменив начальные условия на нужные формулы (рис. 4.4).

```
model lab5_2
Real x, y;
Real a=0.22;
Real b=0.051;
Real c=0.33;
Real d=0.041;
Real t=time;
initial equation
x=c/d;
y=a/b;
equation
der(x)= -a*x + b*x*y;
der(y)= c*y - d*x*y;
end lab5_2;
```

Рис. 4.4: Реализация второго случая на языке Modelica

5. Затем установим настройки симуляции (начальное (0) и конечное (30) время и шаг (0.1)) и запустим симуляцию. Получим следующий результат. По графику видно, что стационарное состояние найдено верно.(рис. 4.5).

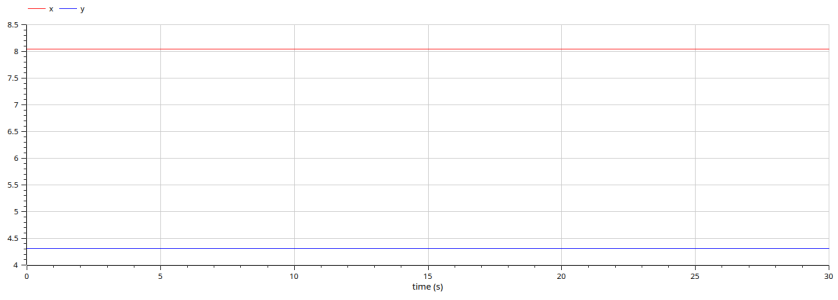


Рис. 4.5: Результат второго случая на языке Modelica

6. Затем необходимо вывести фазовый портрет. В Open Modelica для этого необходимо открыть новое окно параметрического вывода график, нажать Shift и, удерживая его, поставить галочку у x, после чего поставить галочку у y. В результате на экране появится фазовый портрет. По нему видно, что стационарное состояние найдено верно. (рис. 4.6).

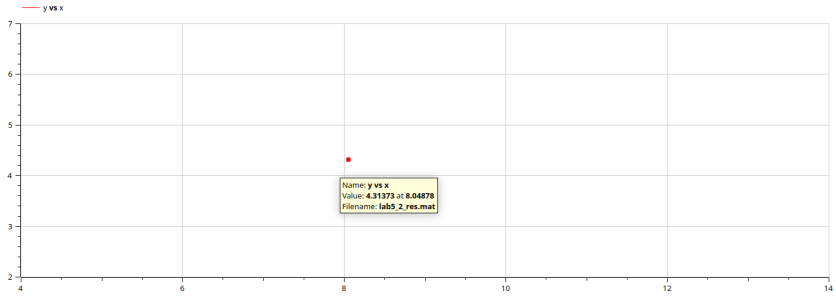


Рис. 4.6: Фазовый портрет второго случая на языке Modelica

7. Теперь опишем эти случаи на языке Julia (рис. 4.7, 4.8, 4.9).

```
using DifferentialEquations
using Plots

a=0.22
b=0.051
c=0.33
d=0.041

function F!(du, u, p, t,)
    du[1] = -a*u[1] + b*u[1]*u[2]
    du[2] = c*u[2] - d*u[1]*u[2]
end

begin
    u0 = [3, 8]
    T = [0.0, 30.0]
    prob = ODEProblem(F!, u0, T)
end

```

Рис. 4.7: Подключаем библиотеки, задаём коэффициенты и функцию, решающую дифференциальные уравнения. Затем зададим начальные условия.

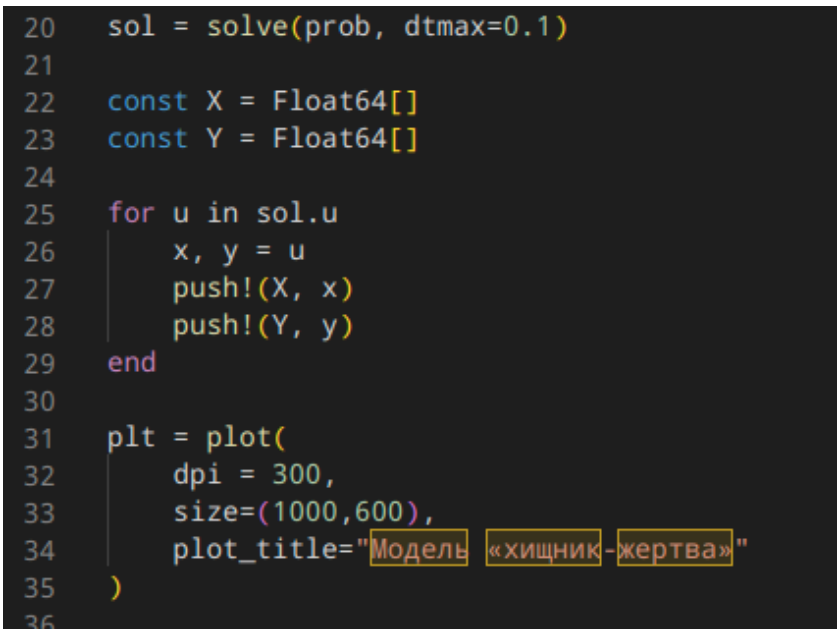


Рис. 4.8: Выполним функцию с данными значениями. Создадим два пустых массива, в которые мы передадим полученные значения. Затем с помощью функционала библиотеки Plots создадим поле для вывода результата.

```
plot!(
    plt,
    sol.t,
    X,
    xlabel="x",
    color=:blue,
    label="Изменение численности хищников"
)

plot!(
    plt,
    sol.t,
    Y,
    ylabel="y",
    color=:red,
    label="Изменение численности жертв"
)

savefig(plt, "lab5.png")
```

Рис. 4.9: Выведем на экран полученные графы и сохраним результат в формате png

8. Получим следующий результат (рис. 4.10).

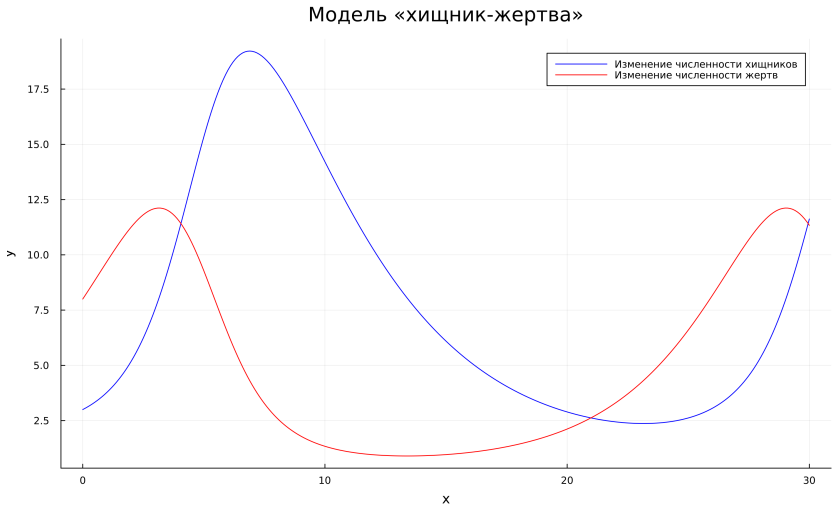


Рис. 4.10: Результат работы программы

9. Код для вывода на экран фазового портрета отличается лишь тем, что мы

хотим вывести на экран (рис. 4.11).

```
25  for u in sol.u
26      x, y = u
27      push!(X, x)
28      push!(Y, y)
29  end
30
31  plt = plot(
32      dpi = 300,
33      size=(1000,600),
34      plot_title="Модель «хищник-жертва»"
35  )
36
37  plot!(
38      plt,
39      X,
40      Y,
41      xlabel="x",
42      color=:blue,
43      label="Фазовый портрет"
44  )
45
46  savefig(plt, "lab5_portret.png")
```

Рис. 4.11: Меняем только последнюю часть кода

10. Получим следующий результат (рис. 4.12).

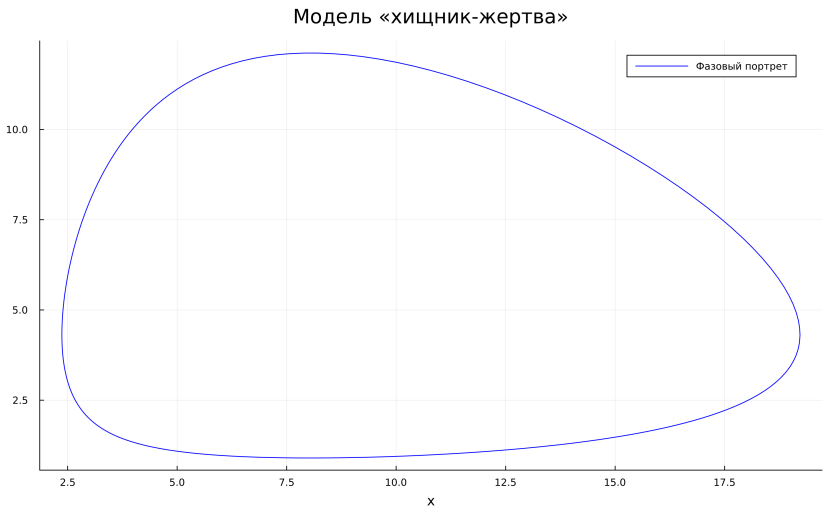
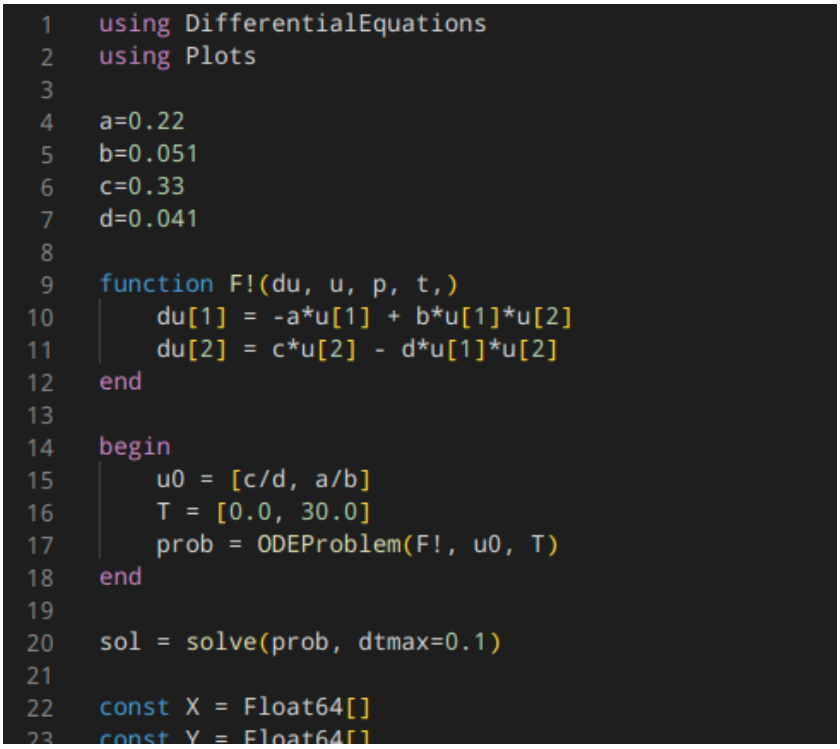


Рис. 4.12: Фазовый портрет

11. Найдём стационарное состояние системы, заменив начальные условия на нужные формулы (рис. 4.13).



```
using DifferentialEquations
using Plots

a=0.22
b=0.051
c=0.33
d=0.041

function F!(du, u, p, t,)
    du[1] = -a*u[1] + b*u[1]*u[2]
    du[2] = c*u[2] - d*u[1]*u[2]
end

begin
    u0 = [c/d, a/b]
    T = [0.0, 30.0]
    prob = ODEProblem(F!, u0, T)
end

sol = solve(prob, dtmax=0.1)

const X = Float64[]
const Y = Float64[]
```

Рис. 4.13: Изменения в коде

12. Получим следующий результат (рис. 4.14).

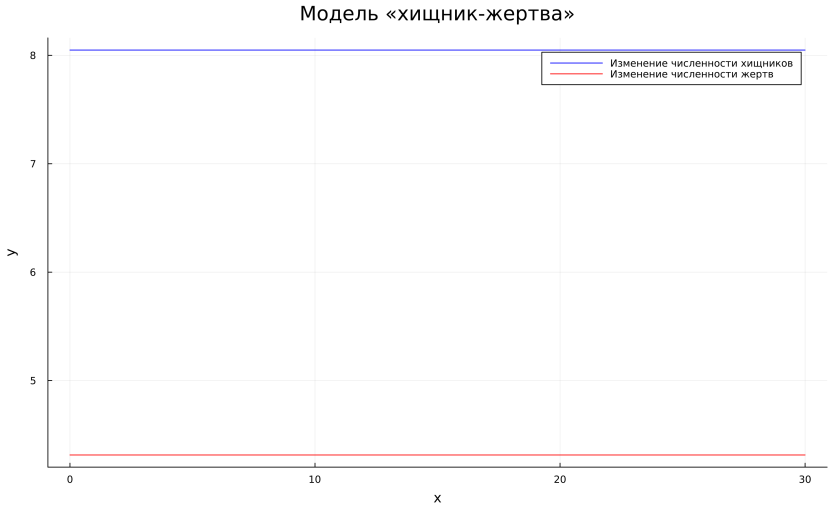


Рис. 4.14: Результат работы программы для второго случая

13. Код для вывода на экран фазового портрета отличается лишь тем, что мы хотим вывести на экран (рис. 4.15).

```
28      push!(Y, y)
29  end
30
31  plt = plot(
32      dpi = 300,
33      size=(1000,600),
34      plot_title="Модель «хищник-жертва»"
35  )
36
37  scatter!(
38      plt,
39      X,
40      Y,
41      xlabel="x",
42      color=:blue,
43      label="Фазовый портрет"
44  )
45
46  savefig(plt, "lab5_stat_portret.png")
```

Рис. 4.15: Меняем только последнюю часть кода

14. Получим следующий результат (рис. 4.16).

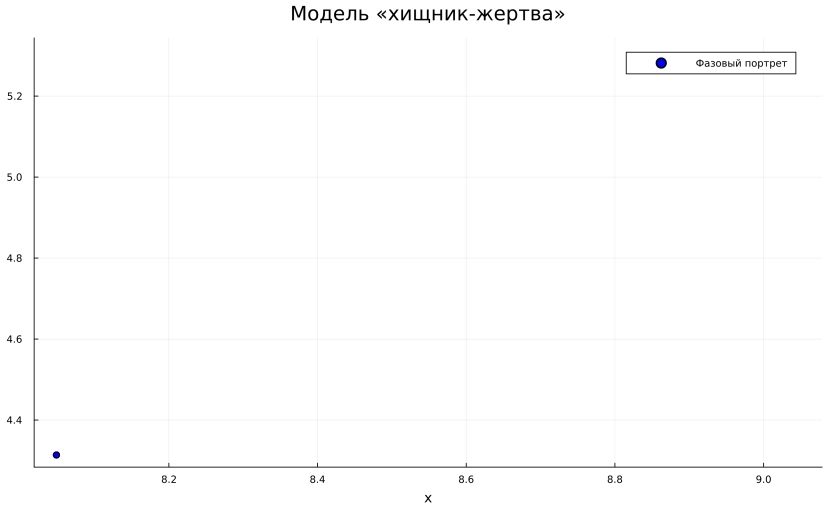


Рис. 4.16: Фазовый портрет для первого случая

5 Выводы

Выполнив данную лабораторную работу, мы продолжили знакомство с языками программирования Julia и Modelica. Сравнивая реализацию одной программы на этих двух языках, можно заметить, что реализация на языке Modelica заметно проще и более точно показывает результат, поскольку можно отследить значения переменных с максимальной точностью на любом отрезке времени.

Список литературы

1. Julia [Электронный ресурс]. Free Software Foundation, 2023. URL: https://ru.wikipedia.org/wiki/Julia_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F).
2. OpenModelica [Электронный ресурс]. Free Software Foundation, 2023. URL: https://ru.wikipedia.org/wiki/OpenModelica.
3. Система «хищник — жертва» [Электронный ресурс]. Free Software Foundation, 2023. URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%C2%AB%D1%85%D0%B8%D1%89%D0%BD%D0%B8%D0%BA_%E2%80%94_%D0%B6%D0%B5%D1%80%D1%82%D0%B2%D0%B0%C2%BB.