Отчёт по лабораторной работе

Лабораторная работа №5 (вариант 10)

Сергеев Тимофей Сергеевич

Содержание

# 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

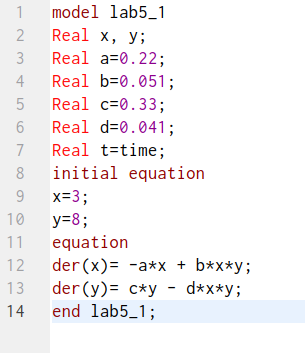
**Julia** – высокоуровневый высокопроизводительный свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков (например, MATLAB и Octave), 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]

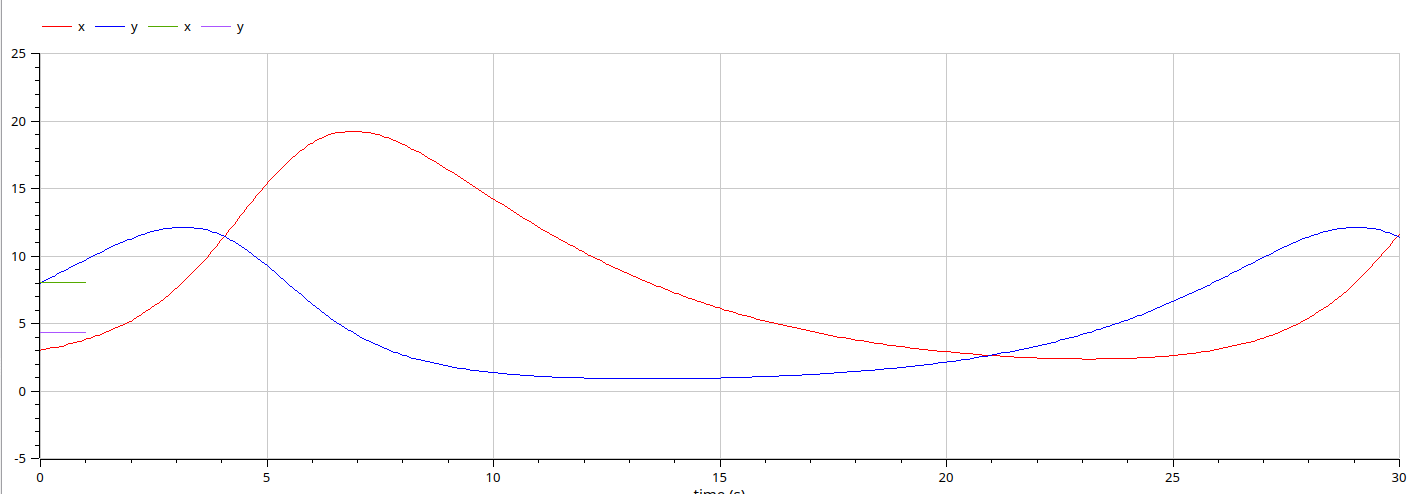
# 4 Выполнение лабораторной работы

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 х и у, подставив данные из условия. После этого пропишем решение наших дифференциальных уравнений (рис. ??).



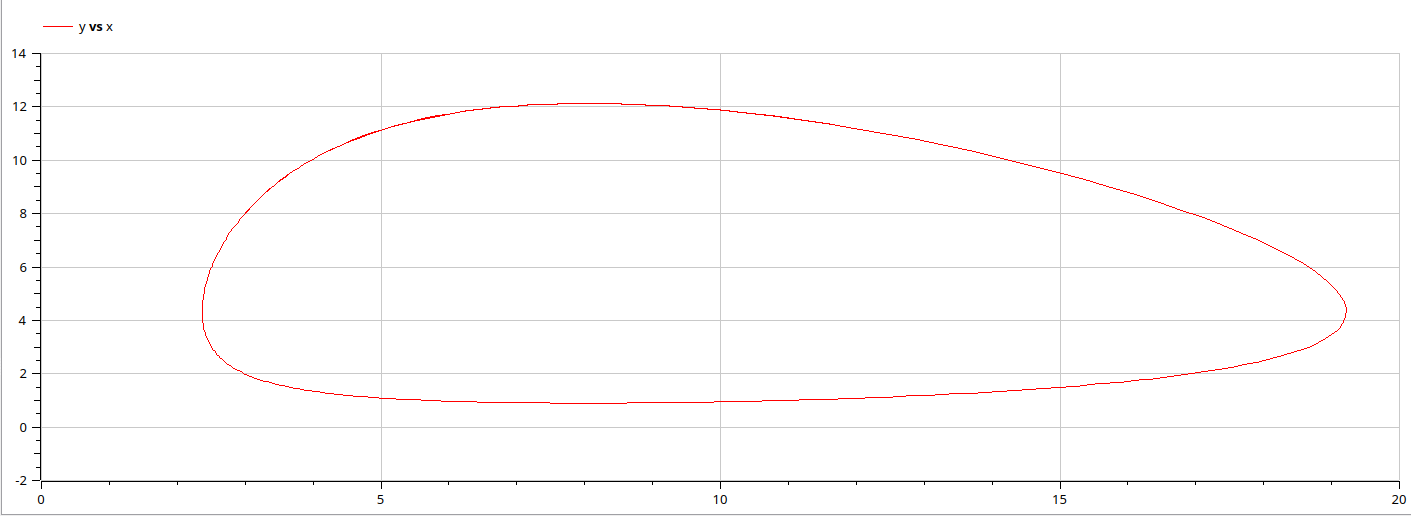
Реализация модели на языке Modelica

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



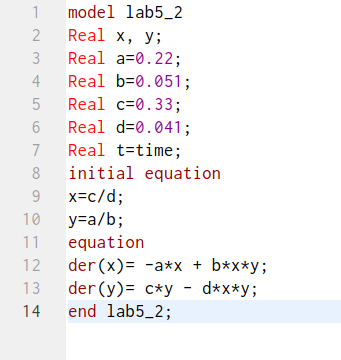
Результат моделирования на языке Modelica

1. 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 х, после чего поставить галочку у у. В результате на экране появится фазовый портрет (рис. ??).



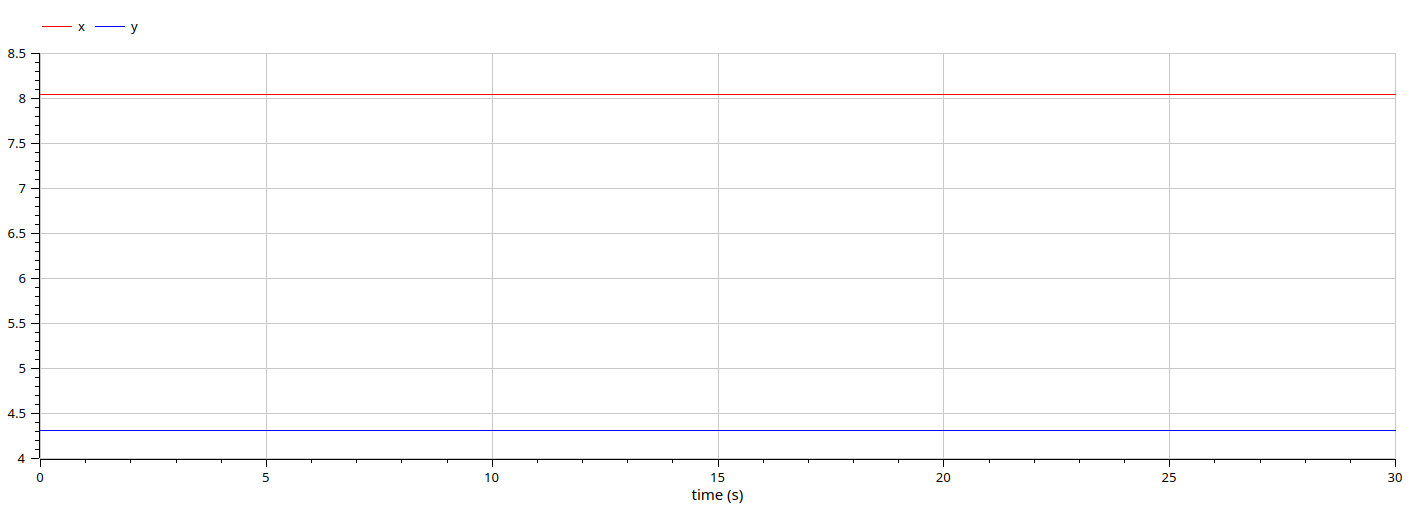
Фазовый портрет на языке Modelica

1. Найдём стационарное состояние системы, заменив начальные условия на нужные формулы (рис. ??).



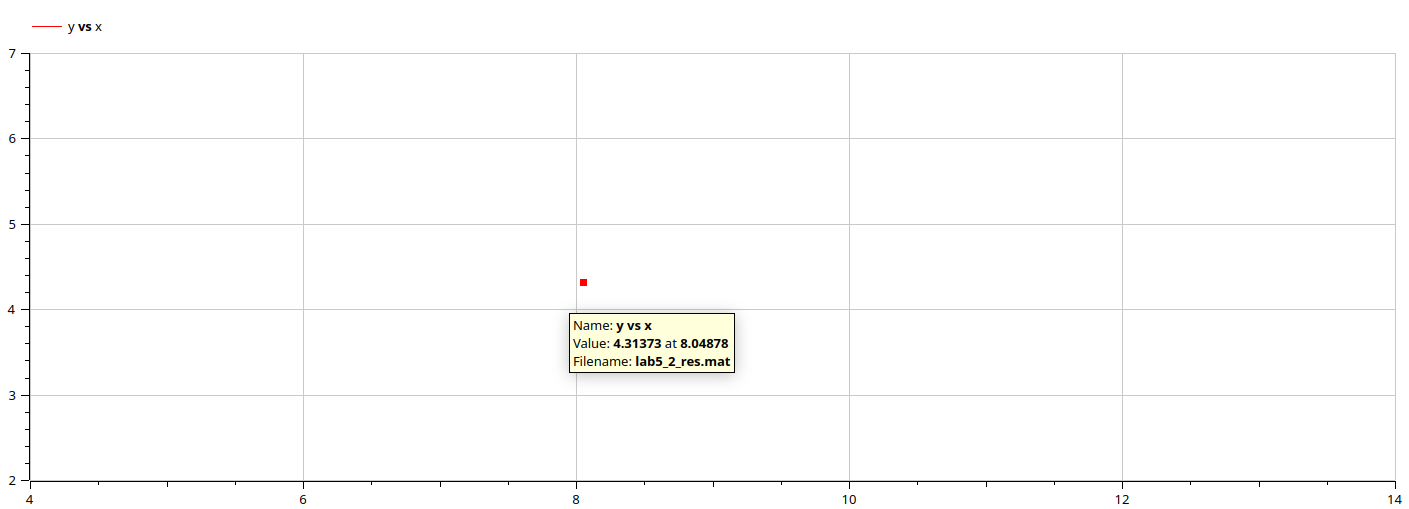
Реализация второго случая на языке Modelica

1. 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. ??).



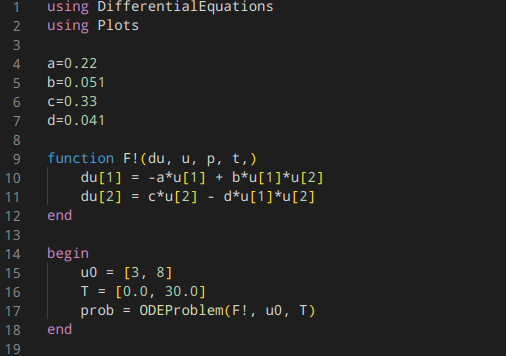
Результат второго случая на языке Modelica

1. 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 х, после чего поставить галочку у у. В результате на экране появится фазовый портрет. По нему видно, что стационарное состояние найдено верно. (рис. ??).

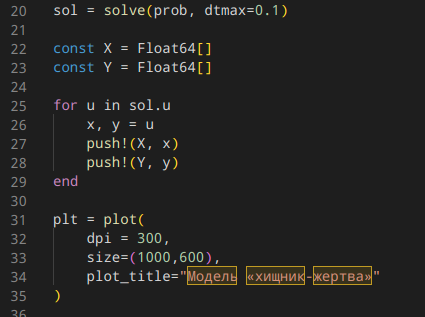


Фазовый портрет второго случая на языке Modelica

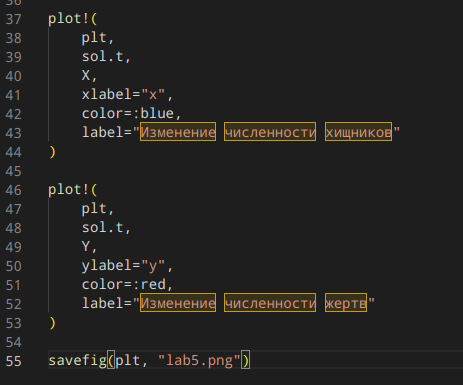
1. Теперь опишем эти случаи на языке Julia (рис. ??, ??, ??).



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

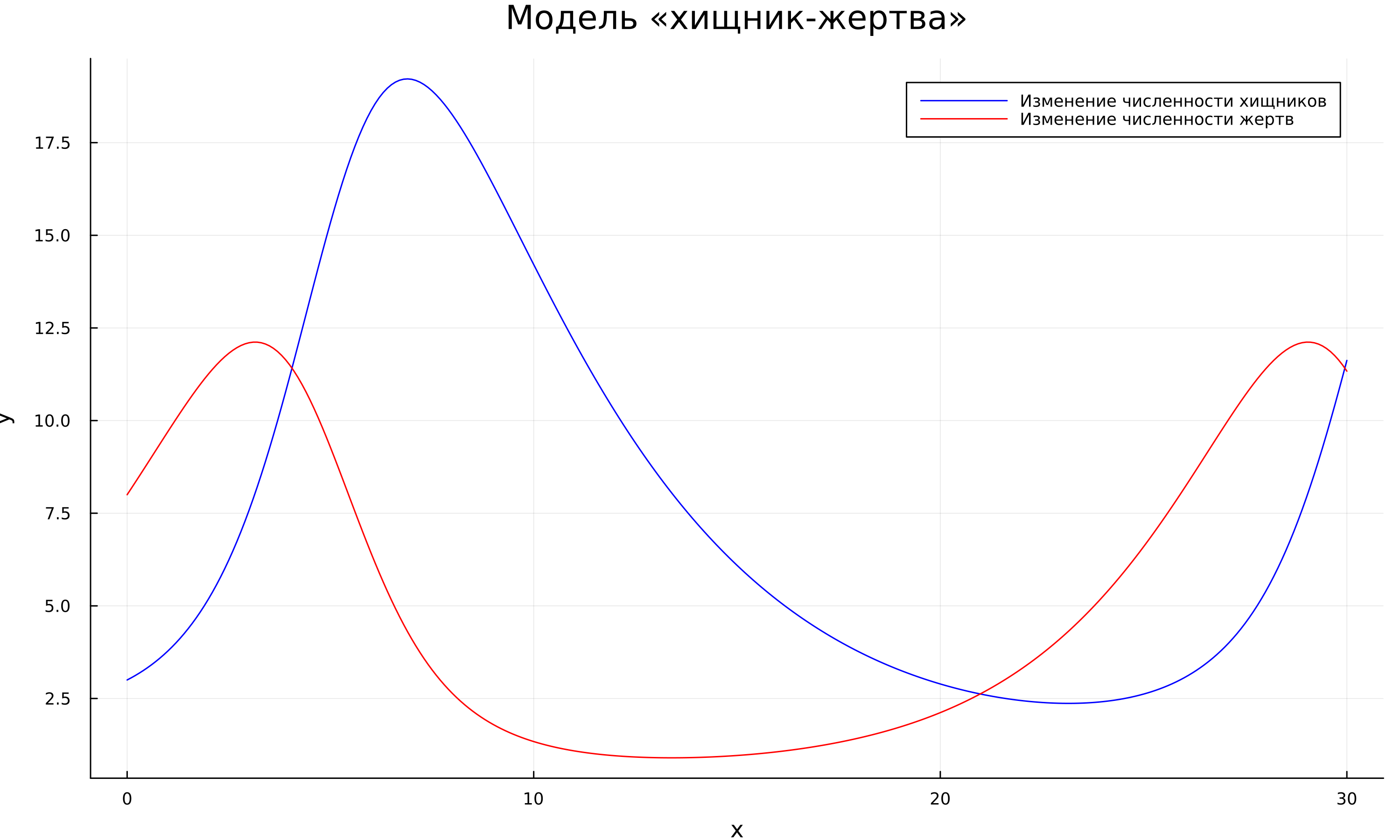


Выполним функцию с данными значениями. Создадим два пустых массива, в которые мы передадим полученные значения. Затем с помощью функционала библиотеки Plots создадим поле для вывода результата.



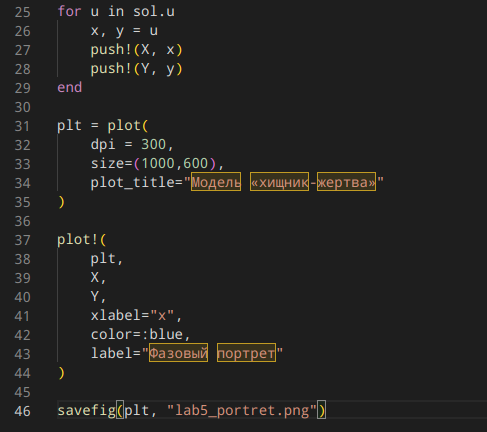
Выведем на экран полученные графы и сохраним результат в формате png

1. Получим следующий результат (рис. ??).



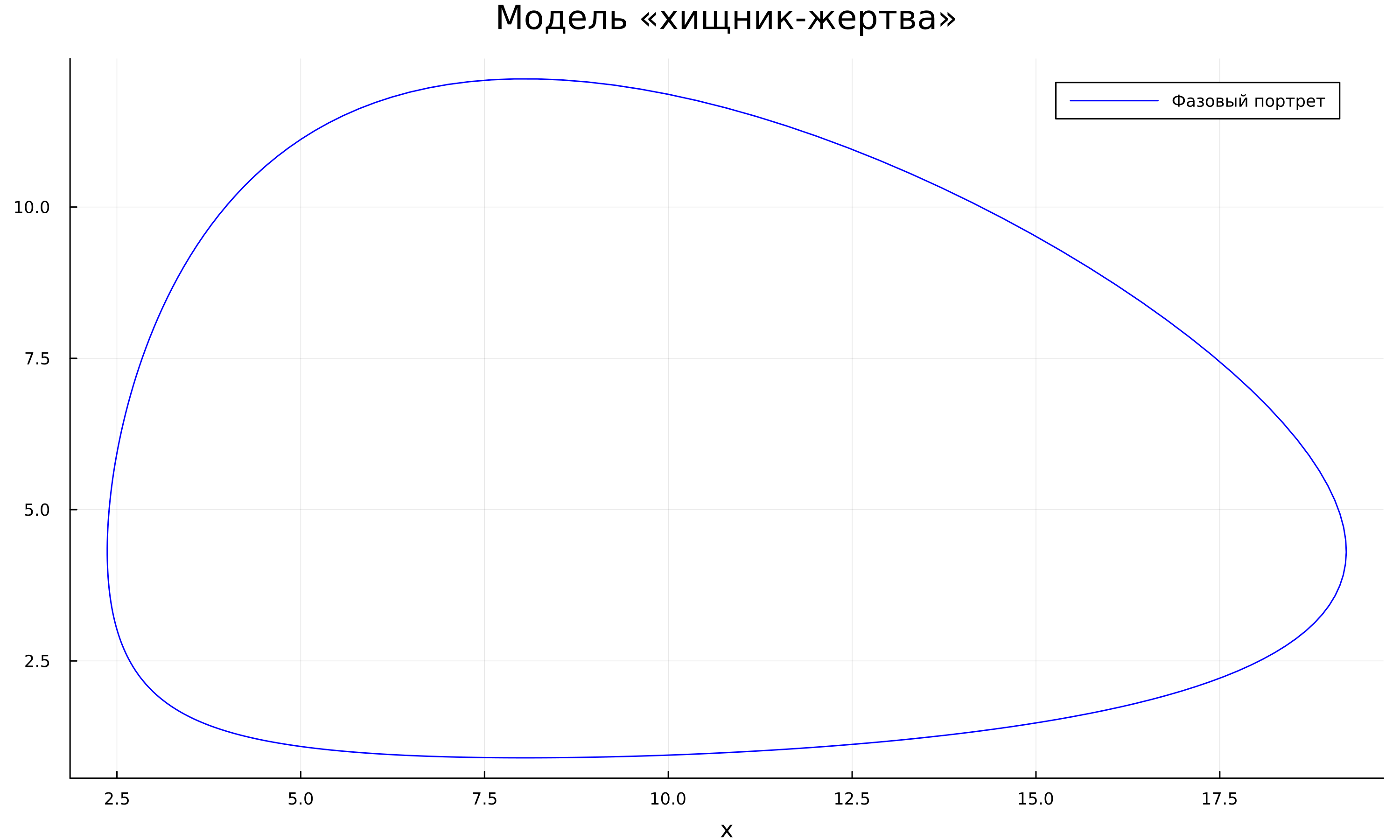
Результат работы программы

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



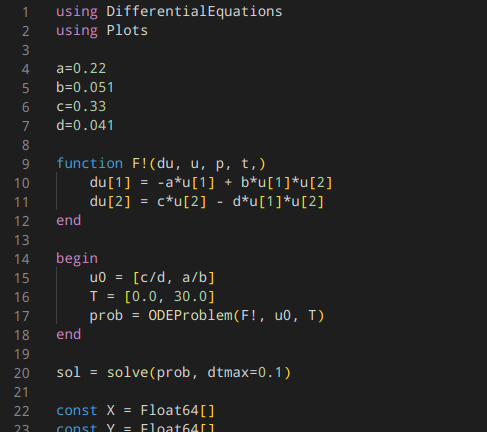
Меняем только последнюю часть кода

1. Получим следующий результат (рис. ??).



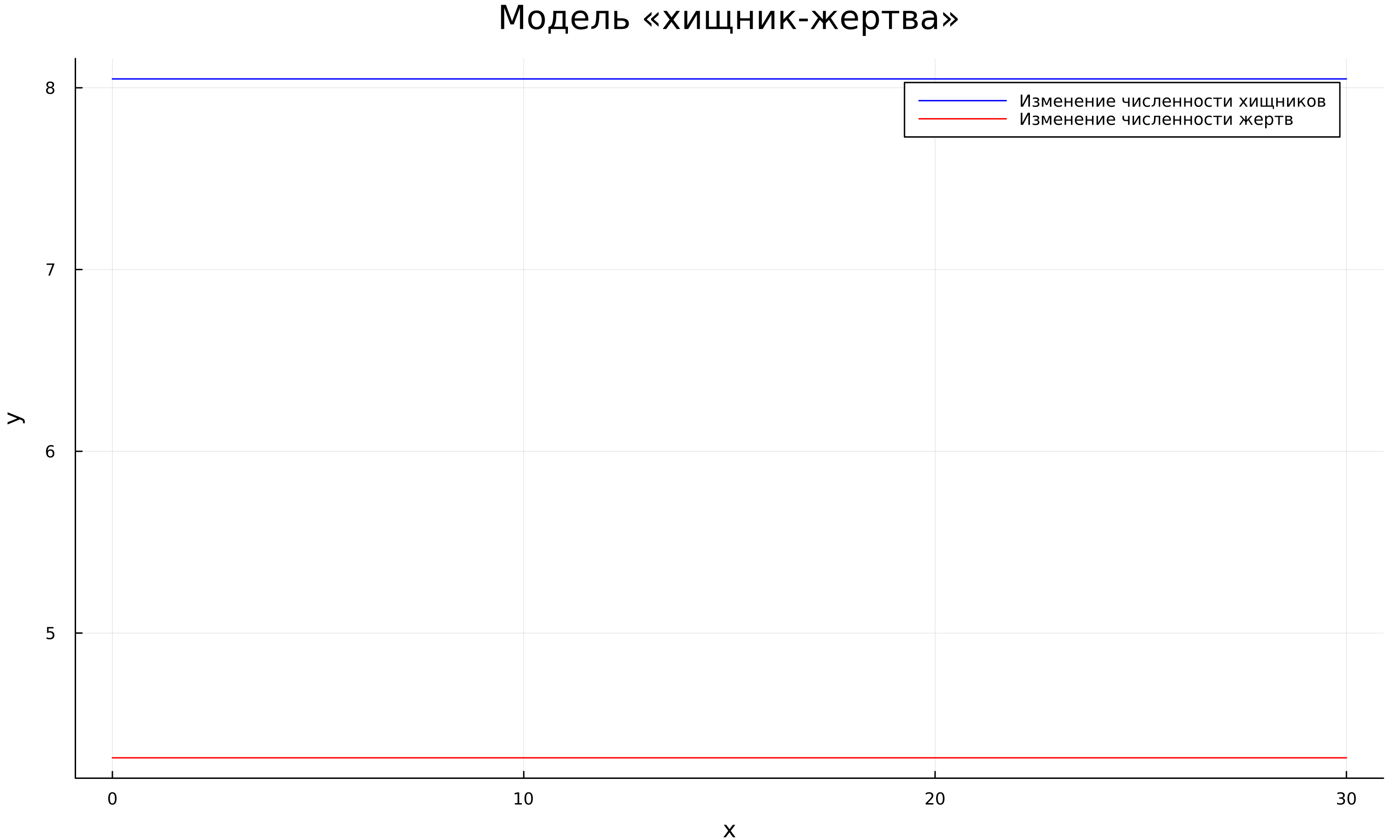
Фазовый портрет

1. Найдём стационарное состояние системы, заменив начальные условия на нужные формулы (рис. ??).



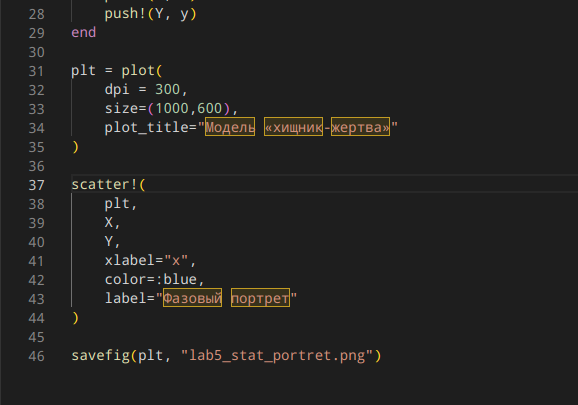
Изменения в коде

1. Получим следующий результат (рис. ??).



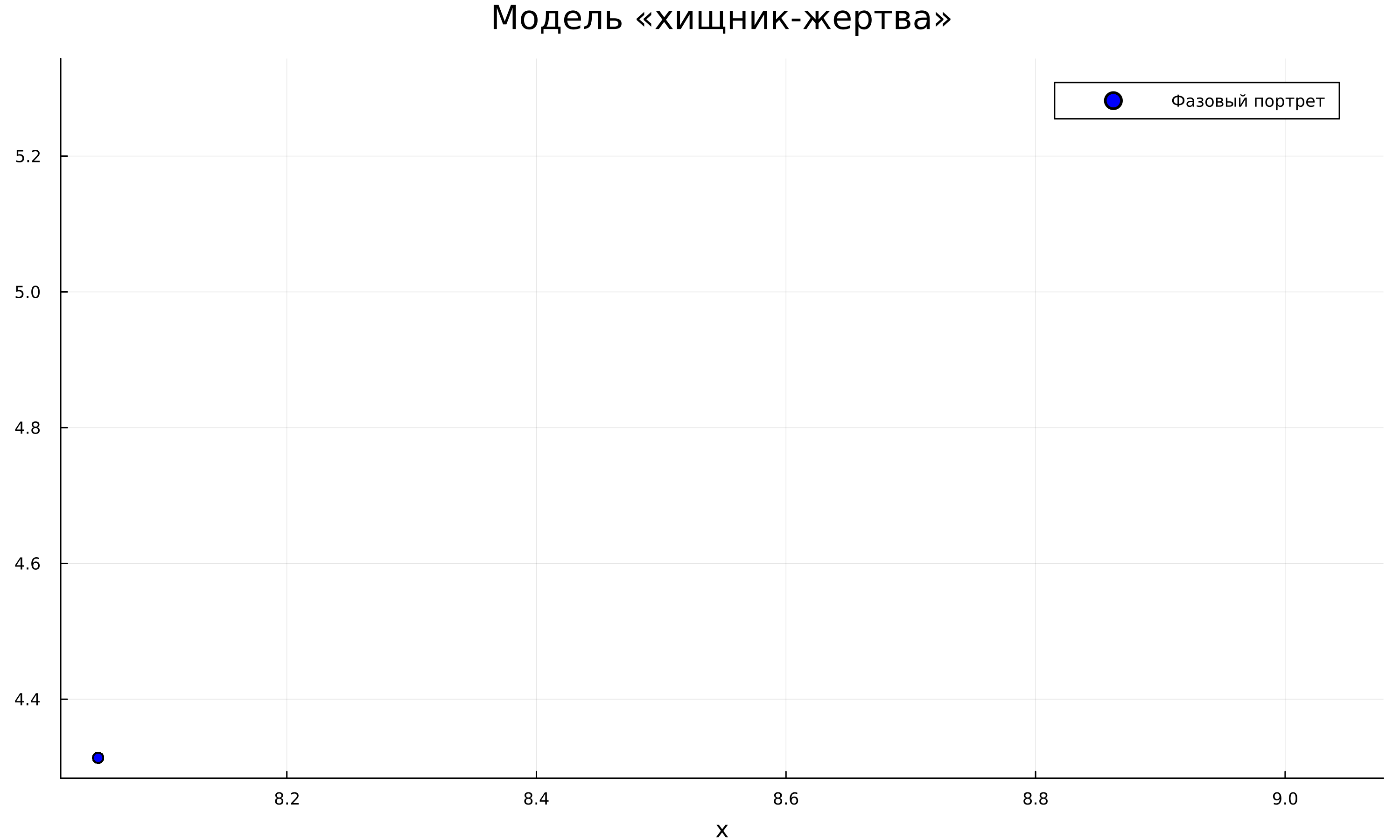
Результат работы программы для второго случая

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



Меняем только последнюю часть кода

1. Получим следующий результат (рис. ??).



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>.