Отчет по лабораторной работе №2

по дисциплине: Математическое моделирование

Ким Михаил Алексеевич

Содержание

# 1 Цель работы

Ознакомиться с базовым функционалом языка программирования Julia, а также интерактивной командной строкой REPL. Ознакомиться с базовым функционалом языка моделирования Modelica и программным обеспечением OpenModelica. Используя эти средства, пострить математическую модель, представляющую собой частный случай задачи о погоне.

# 2 Задание

1. Записать уравнение, описывающее движение катера, с начальными условиями для двух случаев (в зависимости от расположения катера относительно лодки в начальный момент времени).
2. Построить траекторию движения катера и лодки для двух случаев.
3. Найти точку пересечения траектории катера и лодки

# 3 Теоретическое введение

## 3.1 Julia

Julia является высокоуровневым языком программирования, разработанным с целью создания высокоэффективного и легкого для использования языка программирования. Он объединяет лучшие аспекты других языков программирования, таких как Python, R и Matlab [1].

Julia может использоваться в любой области применения, включая инженерные вычисления, анализ данных, машинное обучение и научные вычисления. Она также поддерживает многопоточность и масштабируемость, позволяя программистам писать более эффективные программы, способные обрабатывать большие данные [2].

В целом, Julia является мощным и гибким языком программирования, который может использоваться для решения различных задач. Он предоставляет программистам широкий диапазон возможностей для создания более мощных и высокопроизводительных приложений [3].

## 3.2 OpenModelica

OpenModelica – это свободно распространяемый открытый исходный инструментальный комплекс для моделирования, анализа и разработки систем и проектирования. Он был основан в 1999 году на базе проекта Modelica [4].

OpenModelica является бесплатным программным обеспечением, поддерживает интеграцию с различными языками программирования и позволяет пользовательским приложениям и инструментам работать в единой среде. Области применения OpenModelica включают системное моделирование и проектирование, моделирование сложных автоматизированных систем и многое другое. Он также используется в целях обучения [5].

# 4 Выполнение лабораторной работы

## 4.1 Подготовка системы для работы

### 4.1.1 Установка Julia

1. Устанавливаем язык программирования Julia вместе с интерактивной командной строкой REPL при помощи команд (рис. [1](#fig:01), [2](#fig:02), [3](#fig:03), [4](#fig:04)).

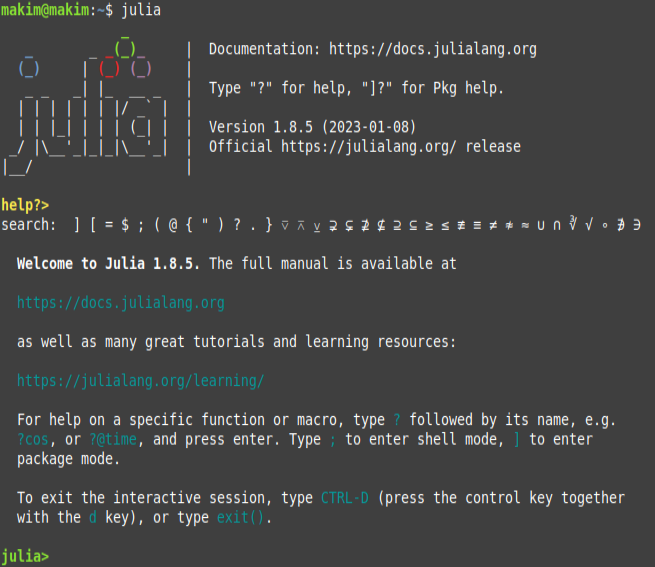
* wget https://julialang-s3.julialang.org/bin/linux/x64/1.8/julia-1.8.5-linux-x86\_64.tar.gz  
  tar zxvf julia-1.8.5-linux-x86\_64.tar.gz  
  sudo cp -r julia-1.8.5 /opt/  
  sudo ln -s /opt/julia-1.8.5/bin/julia /usr/local/bin/julia

* Figure 1: Загрузка архива с официального сайта
* Figure 1: Загрузка архива с официального сайта

* Figure 2: Распаковка и установка архива
* Figure 2: Распаковка и установка архива

* Figure 3: Копирование архива в директорию /opt
* Figure 3: Копирование архива в директорию /opt

* Figure 4: Создание символической ссылки
* Figure 4: Создание символической ссылки

* 
* Figure 5: Установка завершена

1. Устанавливаем дополнительные библиотеки (рис. [6](#fig:06), [7](#fig:07), [8](#fig:08), [9](#fig:09), [10](#fig:10)).

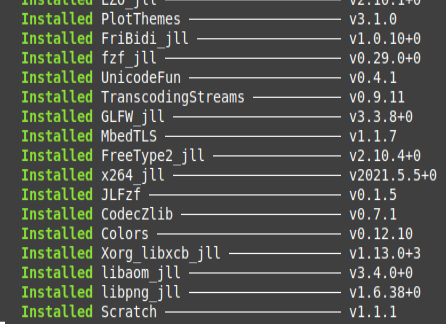
* add DifferentialEquations  
  add Plots

* Figure 6: Ввод символа «]» инициализирует менеджер пакетов
* Figure 6: Ввод символа «]» инициализирует менеджер пакетов

* Figure 7: Добавление библиотеки для работы с диф. уравнениями
* Figure 7: Добавление библиотеки для работы с диф. уравнениями

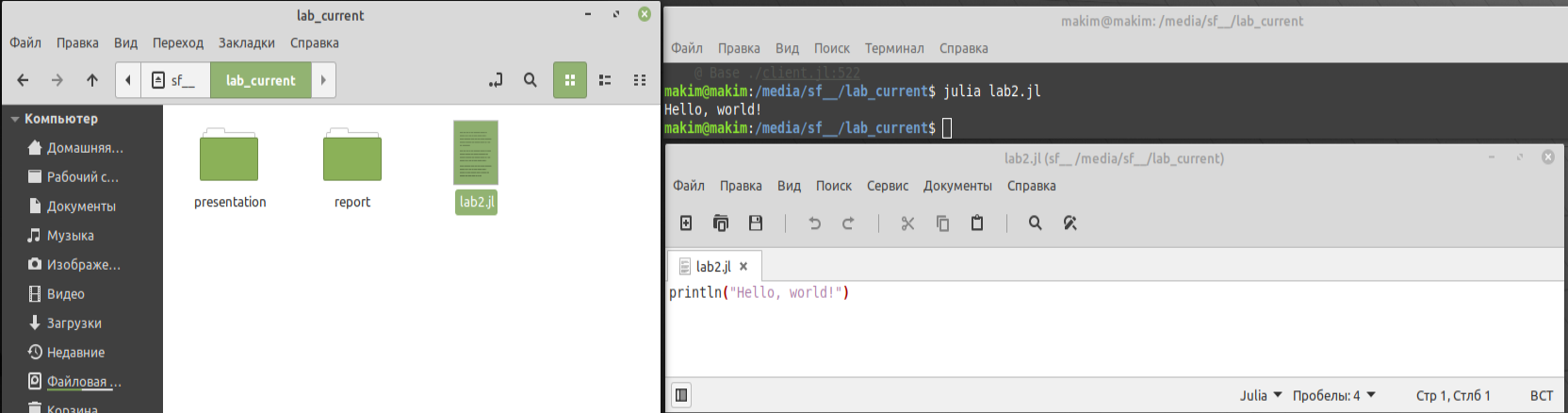
* 
* Figure 8: Установка библиотеки

* Figure 9: Добавление библиотеки для отрисоввки графиков
* Figure 9: Добавление библиотеки для отрисоввки графиков

* 
* Figure 10: Установка библиотеки

1. Проверим правильность установки языка (рис. [11](#fig:11), [12](#fig:12)).

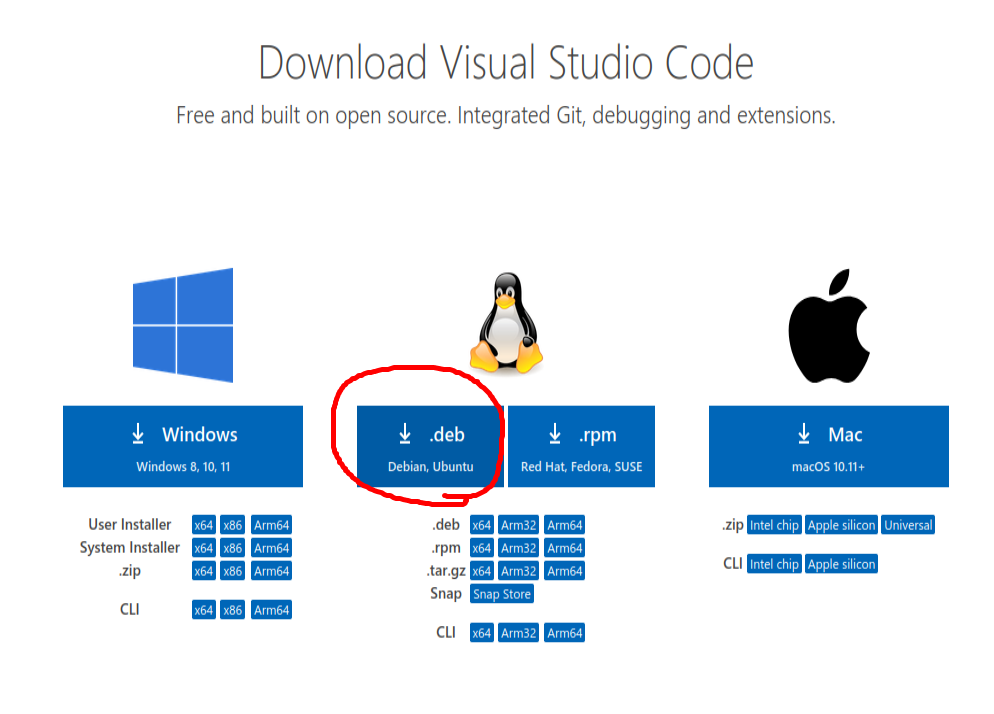
* Figure 11: Нахождение номера варианта задания к лабораторной работе
* Figure 11: Нахождение номера варианта задания к лабораторной работе

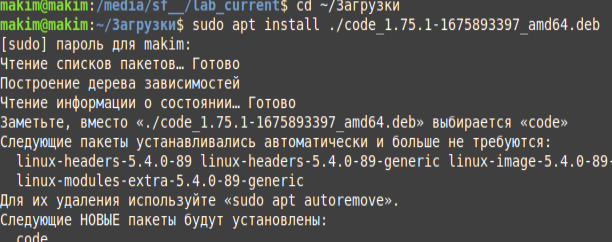
* 
* Figure 12: Вывод «Hello, world!» в консоль

### 4.1.2 Установка Visual Studio Code

1. Скачиваем файл .deb с официального сайта (рис. [13](#fig:13), [14](#fig:14), [15](#fig:15)).

* sudo apt install ./code\_1.75.1-1675893397\_amd64.deb

* 
* Figure 13: Интерактивное окно скачивания

* 
* Figure 14: Установка пакетов

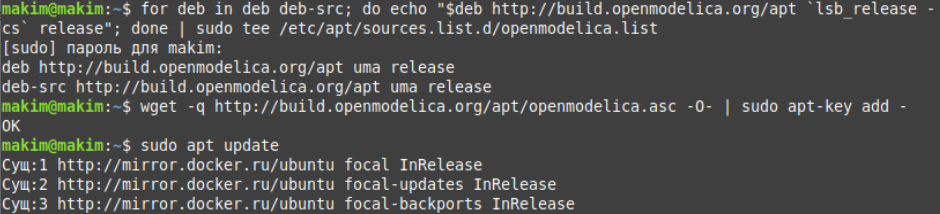
1. Использование UTF-символов в коде Julia в VSC (рис. [15](#fig:15)).

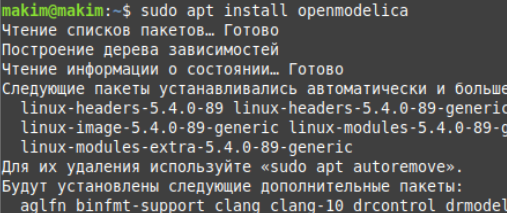
* Figure 15: Специальные символы доступны на Julia
* Figure 15: Специальные символы доступны на Julia

### 4.1.3 Установка OpenModelica

1. Устанавливаем программное обеспечение OpenModelica при помощи команд (рис. [16](#fig:16), [17](#fig:17)).

* for deb in deb deb-src; do echo "$deb http://build.openmodelica.org/apt `lsb\_release -cs` release"; done | sudo tee /etc/apt/sources.list.d/openmodelica.list  
  wget -q http://build.openmodelica.org/apt/openmodelica.asc -O- | sudo apt-key add -  
  sudo apt update  
  sudo apt install openmodelica

* 
* Figure 16: Процесс установки OpenModelica. 1

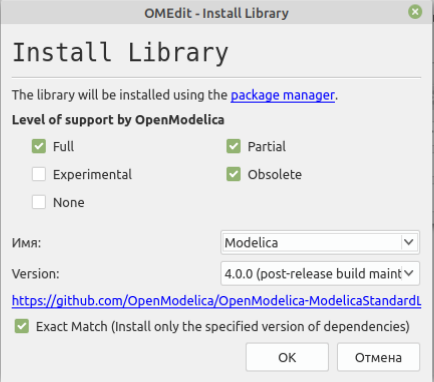
* 
* Figure 17: Процесс установки OpenModelica. 2

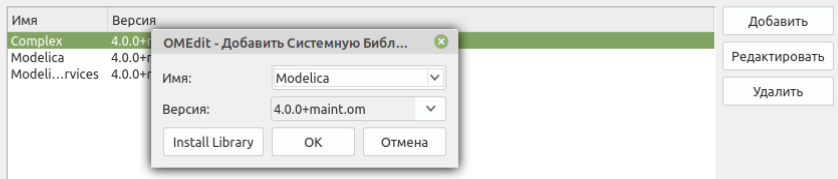
1. Проверяем правильность установки (рис. [18](#fig:18)).

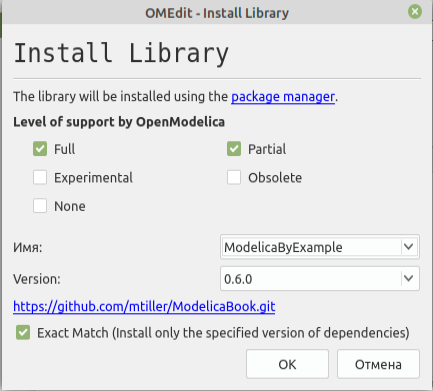
* /usr/bin/omc --version

* Figure 18: Получение версии OpenModelica
* Figure 18: Получение версии OpenModelica

1. Устанавливаем дополнительные библиотеки (рис. [19](#fig:19), [20](#fig:20), [21](#fig:21)).

* 
* Figure 19: Установка библиотеки Modelica в редакторе OMEdit

* 
* Figure 20: Добавление библиотеки в перечень используемых в OMEdit

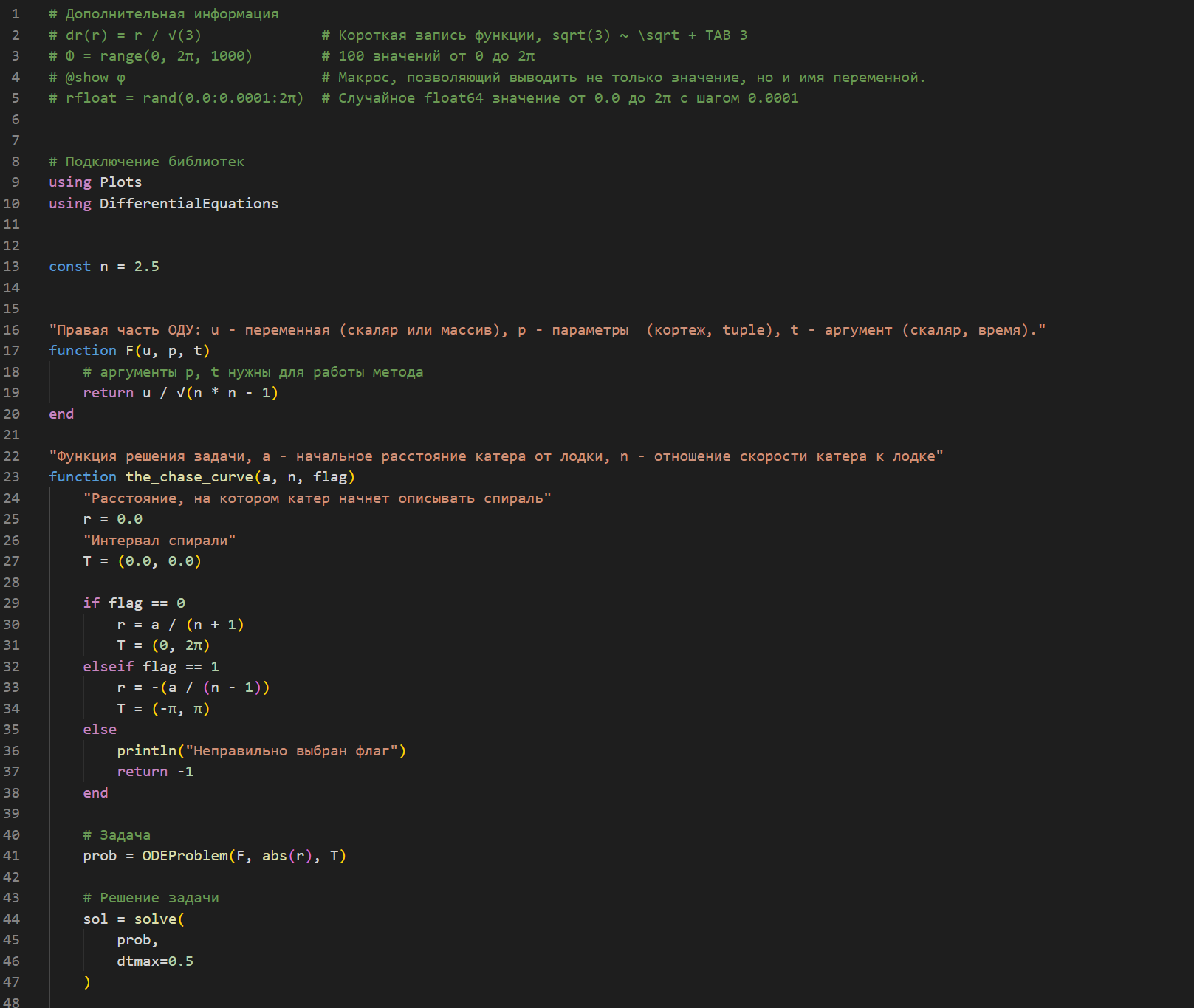
* 
* Figure 21: Установка библиотеки ModelicaByExample в редакторе OMEdit

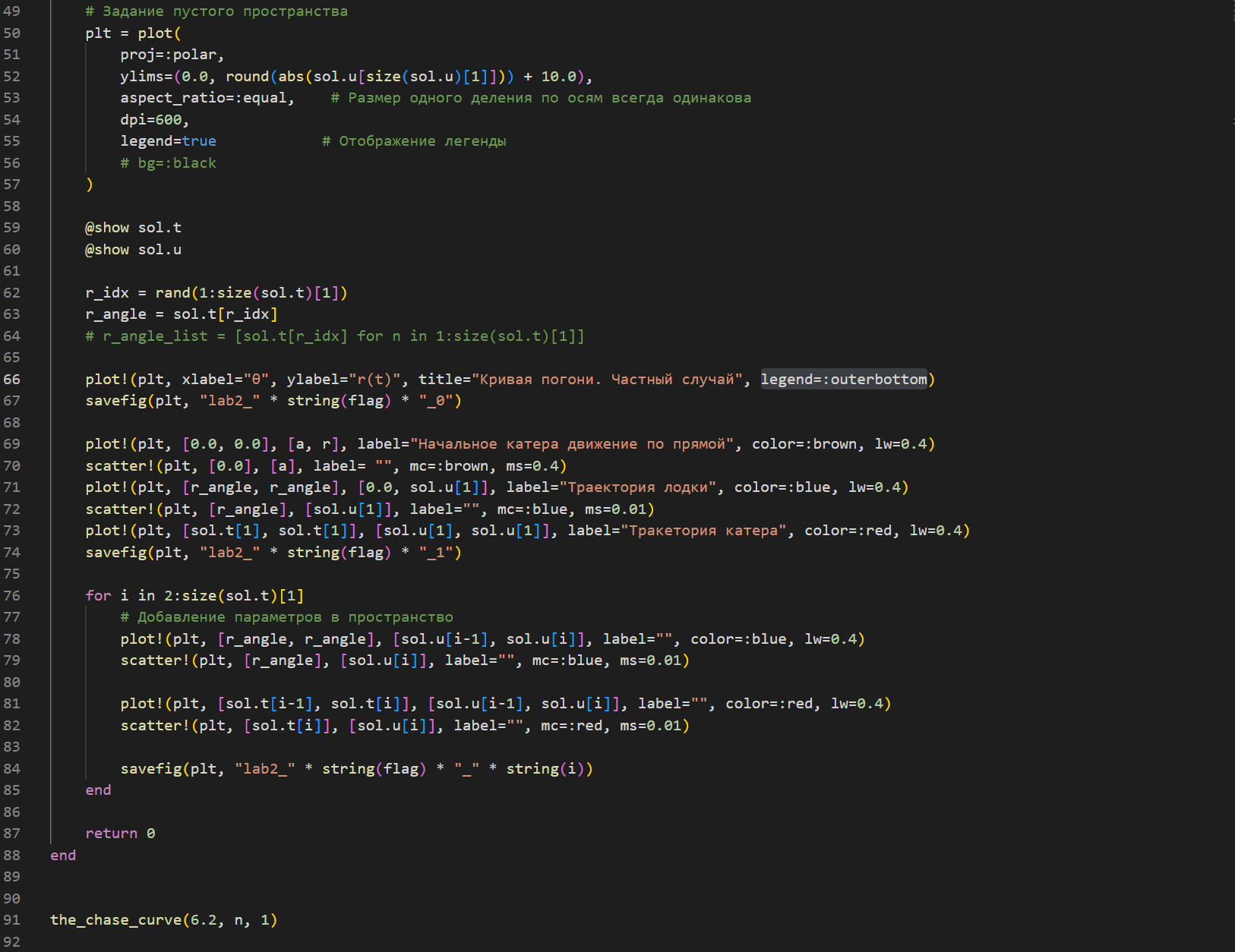
## 4.2 Выполнение лабораторной работы

### 4.2.1 Julia

1. Пишем программу, воспроизводящую модель на языке программирования Julia (рис. [22](#fig:22), [23](#fig:23)).

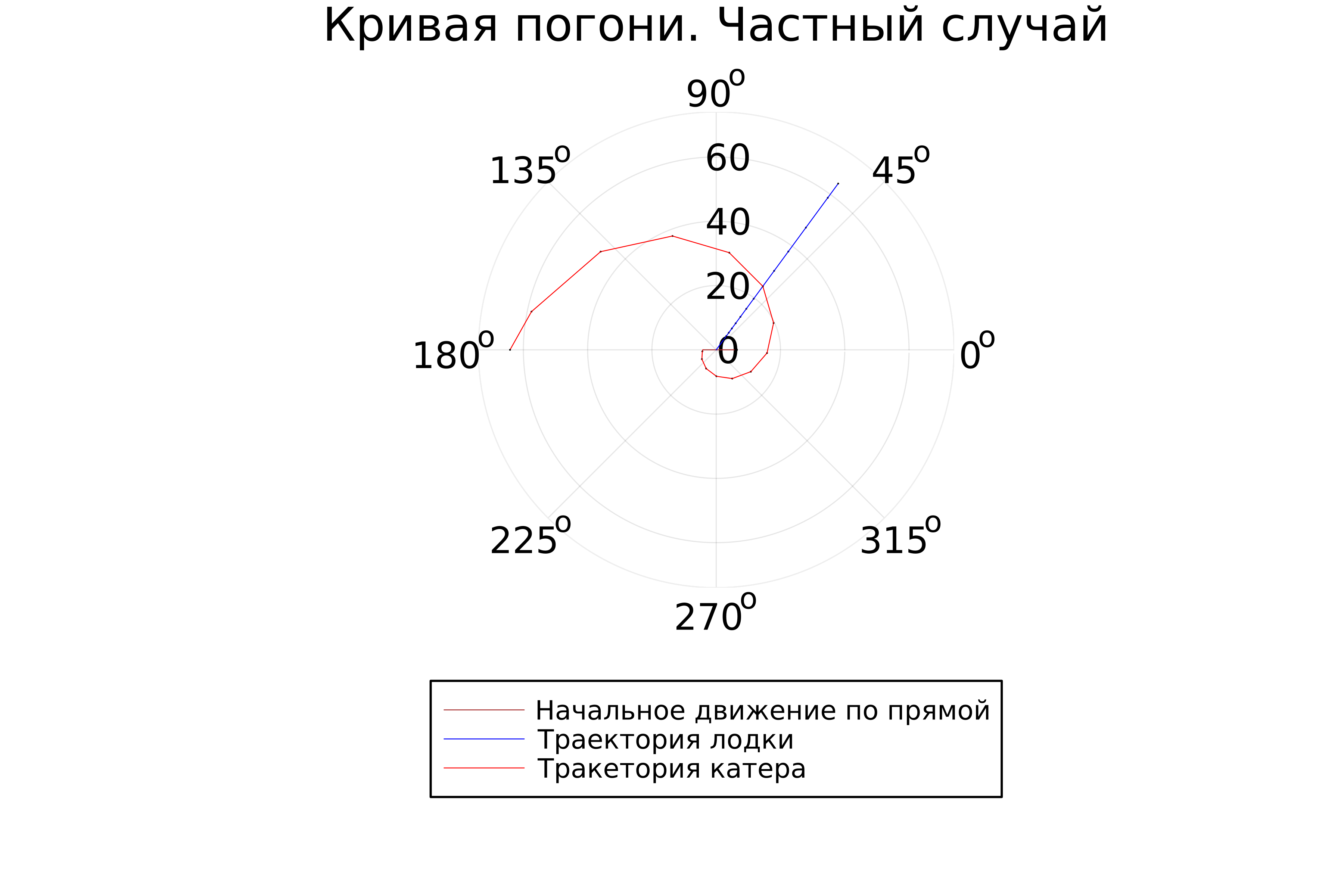
* # Дополнительная информация  
  # dr(r) = r / √(3) # Короткая запись функции, sqrt(3) ~ \sqrt + TAB 3  
  # Φ = range(0, 2π, 1000) # 100 значений от 0 до 2π  
  # @show ϕ # Макрос, позволяющий выводить не только значение, но и имя переменной.  
  # rfloat = rand(0.0:0.0001:2π) # Случайное float64 значение от 0.0 до 2π с шагом 0.0001  
    
    
  # Подключение библиотек  
  using Plots  
  using DifferentialEquations  
    
    
  const n = 2.5  
    
    
  "Правая часть ОДУ: u - переменная (скаляр или массив), p - параметры (кортеж, tuple), t - аргумент (скаляр, время)."  
  function F(u, p, t)  
   # аргументы p, t нужны для работы метода  
   return u / √(n \* n - 1)  
  end  
    
  "Функция решения задачи, a - начальное расстояние катера от лодки, n - отношение скорости катера к лодке"  
  function the\_chase\_curve(a, n, flag)  
   "Расстояние, на котором катер начнет описывать спираль"  
   r = 0.0  
   "Интервал спирали"  
   T = (0.0, 0.0)  
    
   if flag == 0  
   r = a / (n + 1)  
   T = (0, 2π)  
   elseif flag == 1  
   r = -(a / (n - 1))  
   T = (-π, π)  
   else  
   println("Неправильно выбран флаг")  
   return -1  
   end  
    
   # Задача  
   prob = ODEProblem(F, abs(r), T)  
    
   # Решение задачи  
   sol = solve(  
   prob,  
   dtmax=0.5  
   )  
    
   # Задание пустого пространства  
   plt = plot(   
   proj=:polar,  
   ylims=(0.0, round(abs(sol.u[size(sol.u)[1]])) + 10.0),  
   aspect\_ratio=:equal, # Размер одного деления по осям всегда одинакова  
   dpi=600,  
   legend=true # Отображение легенды  
   # bg=:black  
   )  
    
   @show sol.t  
   @show sol.u  
    
   r\_idx = rand(1:size(sol.t)[1])  
   r\_angle = sol.t[r\_idx]  
   # r\_angle\_list = [sol.t[r\_idx] for n in 1:size(sol.t)[1]]  
    
   plot!(plt, xlabel="θ", ylabel="r(t)", title="Кривая погони. Частный случай", legend=:outerbottom)  
   savefig(plt, "lab2\_" \* string(flag) \* "\_0")  
    
   plot!(plt, [0.0, 0.0], [a, r], label="Начальное катера движение по прямой", color=:brown, lw=0.4)  
   scatter!(plt, [0.0], [a], label= "", mc=:brown, ms=0.4)  
   plot!(plt, [r\_angle, r\_angle], [0.0, sol.u[1]], label="Траектория лодки", color=:blue, lw=0.4)  
   scatter!(plt, [r\_angle], [sol.u[1]], label="", mc=:blue, ms=0.01)  
   plot!(plt, [sol.t[1], sol.t[1]], [sol.u[1], sol.u[1]], label="Тракетория катера", color=:red, lw=0.4)  
   savefig(plt, "lab2\_" \* string(flag) \* "\_1")  
    
   for i in 2:size(sol.t)[1]  
   # Добавление параметров в пространство  
   plot!(plt, [r\_angle, r\_angle], [sol.u[i-1], sol.u[i]], label="", color=:blue, lw=0.4)  
   scatter!(plt, [r\_angle], [sol.u[i]], label="", mc=:blue, ms=0.01)  
    
   plot!(plt, [sol.t[i-1], sol.t[i]], [sol.u[i-1], sol.u[i]], label="", color=:red, lw=0.4)  
   scatter!(plt, [sol.t[i]], [sol.u[i]], label="", mc=:red, ms=0.01)  
    
   savefig(plt, "lab2\_" \* string(flag) \* "\_" \* string(i))  
   end  
    
   return 0  
  end  
    
    
  the\_chase\_curve(6.2, n, 1)

* 
* Figure 22: Код лабораторной работы на Julia. 1

* 
* Figure 23: Код лабораторной работы на Julia. 2

1. Любуемся результатом (рис. [24](#fig:24), [25](#fig:25)).

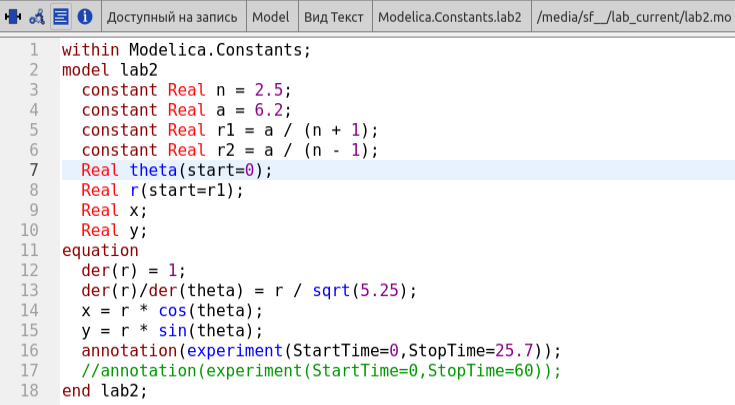
* 
* Figure 24: Результат для первого случая

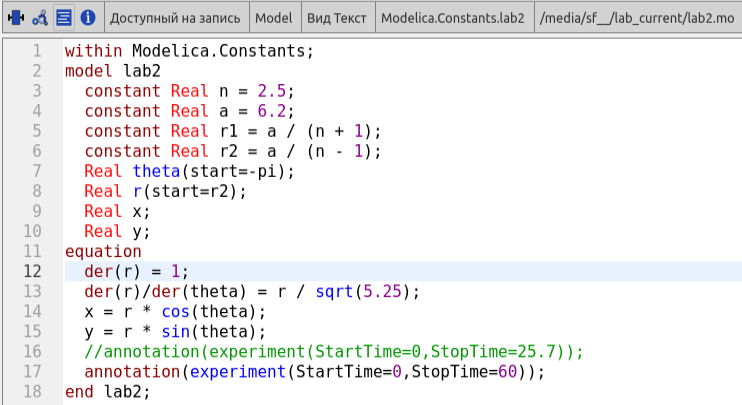
* 
* Figure 25: Результат для второго случая

### 4.2.2 OpenModelica

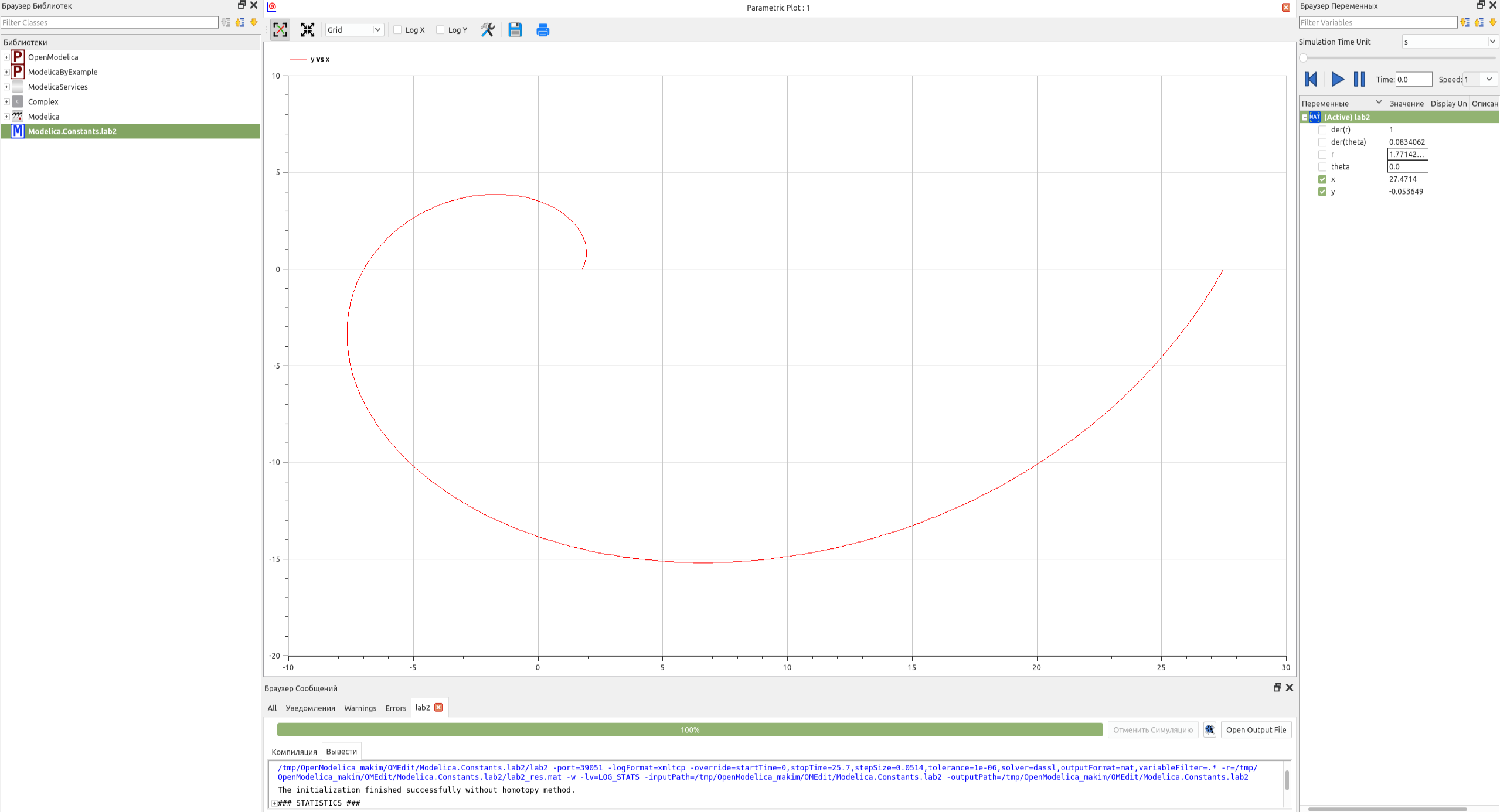
1. Пробуем написать программу, воспроизводящую модель на языке моделирования Modelica (рис. [26](#fig:26), [27](#fig:27)).

* within Modelica.Constants;  
  model lab2  
   constant Real n = 2.5;  
   constant Real a = 6.2;  
   constant Real r1 = a / (n + 1);  
   constant Real r2 = a / (n - 1);  
   Real theta(start=-pi);  
   Real r(start=r2);  
   Real x;  
   Real y;  
  equation  
   der(r) = 1;  
   der(r)/der(theta) = r / sqrt(5.25);  
   x = r \* cos(theta);  
   y = r \* sin(theta);  
   //annotation(experiment(StartTime=0,StopTime=25.7));  
   annotation(experiment(StartTime=0,StopTime=60));  
  end lab2;

* 
* Figure 26: Код лабораторной работы на Modelica для первого случая

* 
* Figure 27: Код лабораторной работы на Modelica для второго случая

1. Смотрим на результат (рис. [28](#fig:28), [29](#fig:29)).

* 
* Figure 28: Результат для первого случая

* 
* Figure 29: Результат для второго случая

### 4.2.3 Математические вычисления

1. Используемые вычисления (рис. [30](#fig:30)).

* 
* Figure 30: Вычисления

# 5 Выводы

Ознакомился с базовым функционалом языка программирования Julia, а также интерактивной командной строкой REPL. Ознакомился с базовым функционалом языка моделирования Modelica и программным обеспечением OpenModelica. Используя эти средства, пострил математическую модель, представляющую собой частный случай задачи о погоне.

**Сделал для себя вывод, что для решения данной задачи подходит язык программирования Julia и почти полностью не подходит язык моделирования Modelica.**

# Список литературы

1. Julia. Знакомство [Электронный ресурс]. Habr, 2018. URL: <https://habr.com/ru/post/423811/>.

2. Julia как язык программирования [Электронный ресурс]. MIVOCLOUD LIMITED, 2022. URL: <https://www.mivocloud.com/ru/blog/Julia-as-a-programming-language>.

3. The Julia Programming Language [Электронный ресурс]. JuliaLang.org, 2022. URL: <https://julialang.org/>.

4. OpenModelica. Introduction [Электронный ресурс]. OpenModelica. URL: <https://openmodelica.org/>.

5. OpenModelica [Электронный ресурс]. Wikimedia Foundation, Inc., 2022. URL: <https://en.wikipedia.org/wiki/OpenModelica>.