Отчет по лабораторной работе № 2

Задача о погоне

Хусаинова Динара Айратовна

Содержание

# Цель работы

Изучить основы языков программирования Julia и OpenModelica. Освоить библиотеки этих языков, которые используются для построения графиков и решения дифференциальных уравнений. Решить задачу о погоне.

# Теоретическое введение

**Справка о языках программирования:**

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.

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. По своим возможностям приближается к таким вычислительным средам как Matlab Simulink, Scilab xCos, имея при этом значительно более удобное представление системы уравнений исследуемого блока.

**Математическая справка:**

Дифференциальное уравнение — уравнение, которое помимо функции содержит её производные. Порядок входящих в уравнение производных может быть различен (формально он ничем не ограничен). Производные, функции, независимые переменные и параметры могут входить в уравнение в различных комбинациях или отсутствовать вовсе, кроме хотя бы одной производной. Не любое уравнение, содержащее производные неизвестной функции, является дифференциальным.

В отличие от алгебраических уравнений, в результате решения которых ищется число (несколько чисел), при решении дифференциальных уравнений ищется функция (семейство функций).

Дифференциальное уравнение порядка выше первого можно преобразовать в систему уравнений первого порядка, в которой число уравнений равно порядку исходного дифференциального уравнения.

**Физические термины:**

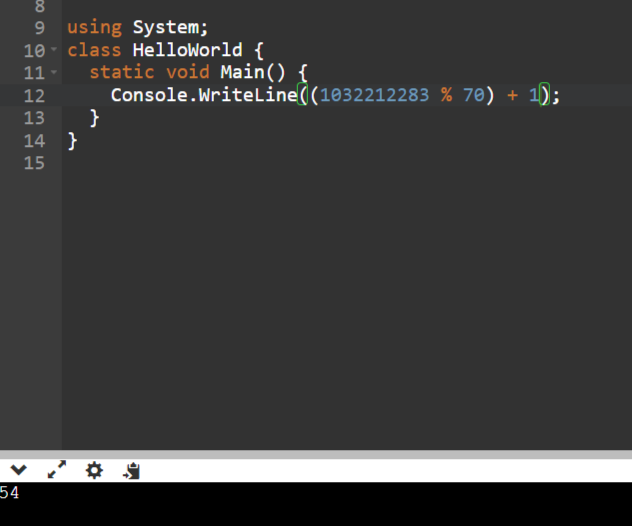
* Тангенциальная скорость - составляющая вектора скорости, перпендикулярная линии, соединяющей источник и наблюдателя. Измеряется собственному движению - угловому перемещению источника.
* Радиальная скорость — проекция скорости точки на прямую, соединяющую её с выбранным началом координат.
* Полярная система координат — двумерная система координат, в которой каждая точка на плоскости определяется двумя числами — полярным углом и полярным радиусом.

# Задание

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

# Выполнение лабораторной работы

Расчитываю свой вариант (рис. 1).



Номер варианта

1. На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 17,7 км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в 3,8 раза больше скорости браконьерской лодки.
2. Примем за момент отсчета времени момент первого рассеивания тумана. Введем полярные координаты с центром в точке нахождения браконьеров и осью, проходящей через катер береговой охраны. Тогда начальные координаты катера (17,7; 0). Обозначим скорость лодки v.
3. Траектория катера должна быть такой, чтобы и катер, и лодка все время были на одном расстоянии от полюса. Только в этом случае траектория катера пересечется с траекторией лодки. Поэтому для начала катер береговой охраны должен двигаться некоторое время прямолинейно, пока не окажется на том же расстоянии от полюса, что и лодка браконьеров. После этого катер береговой охраны должен двигаться вокруг полюса удаляясь от него с той же скоростью, что и лодка браконьеров.
4. Пусть время t - время, через которое катер и лодка окажутся на одном расстоянии от начальной точки.

$$ t = {{x }\over{v}} $$

$$ t = {{17,7-x}\over{3,8 v}} $$

$$ t = {{17,7+x}\over{3,8 v}} $$

Из этих уравнений получаем объедиение двух уравнений:

$$ \left[ \begin{array}{cl}
{{x}\over{v}} = {{17,7-x}\over{3,8 v}}\\
{{x}\over{v}} = {{17,7+x}\over{3,8 v}}
\end{array} \right. $$

Решая это, получаем два значения для x:

$$ v = {dr\over dt} $$

$$ v\_\tau = {{\sqrt{((3,8\*v)^2-v^2)}}} = {\sqrt{70}\*v \over{5}} $$

$$ \left\{ \begin{array}{cl}
{dr\over dt} = v \\
r{d\theta\over dt} = {\sqrt{70}\*v \over{5}}
\end{array} \right. $$

или

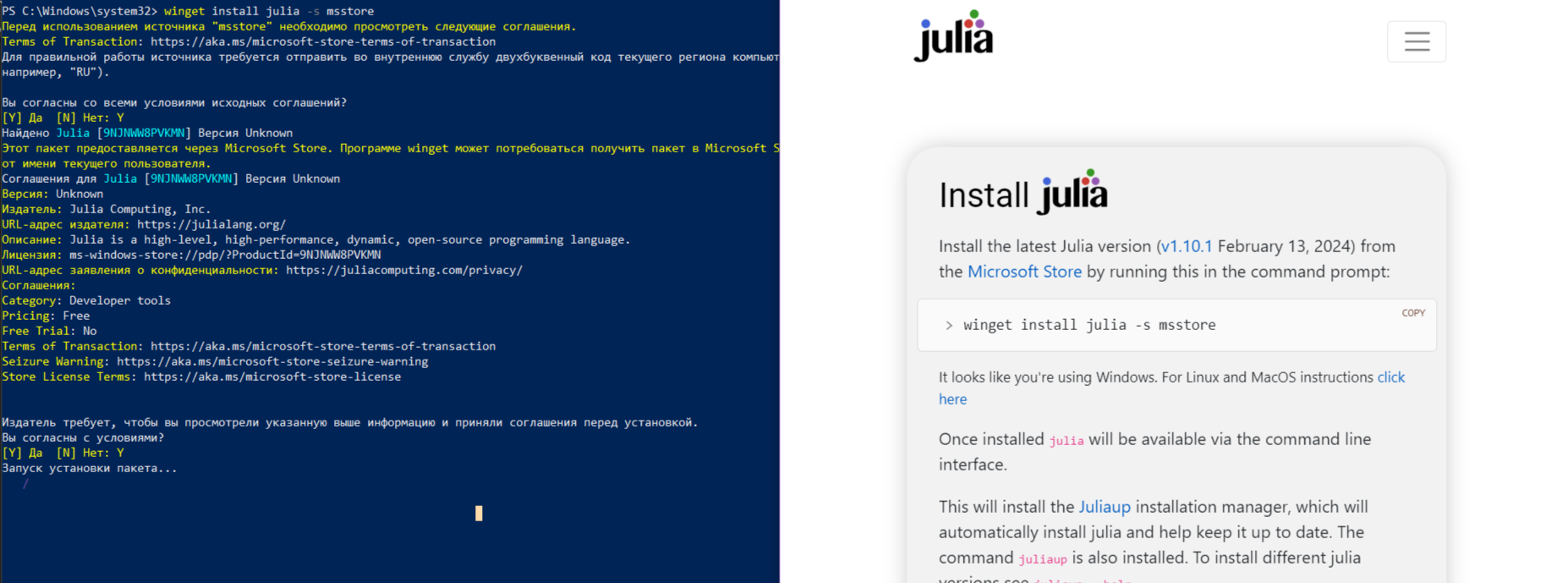
Итоговое уравнение после того, как убрали производную по t:

$$ {dr\over d\theta} = {5r\over\sqrt{70}} $$

# Моделирование с помощью Julia

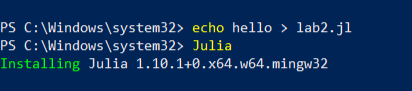
К сожалению, OpenModelica не адаптирована к использованию полярных координат, поэтому адекватное отображение результатов данный задачи там невозможно.

1. Скачиваем Julia ( рис. 2).



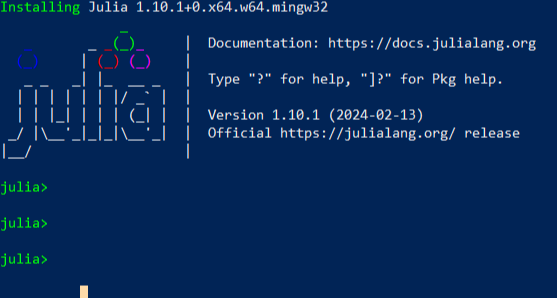
Julia

1. Создаем файл lab2.jl и запускаем Julia (рис. 3).



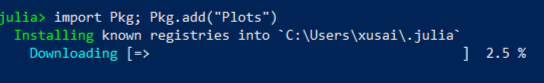
lab2.jl и запуск

1. Запуск Julia (рис. 4).

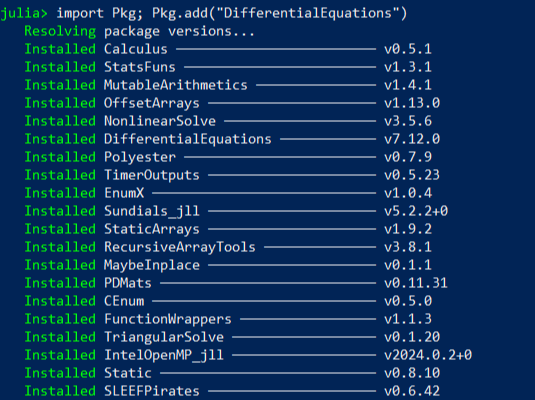


Процесс запуска

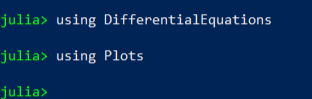
1. Скачиваем необходимые для работы пакеты b проверяем их наличие( рис. 5-7).



Plots



DifferentialEquations



Using

1. Исходный код в файле lab2.jl:

using Plots using DifferentialEquations

const a = 17.7 const n = 3.8

const r0 = a/(n + 1) const r0\_2 = a/(n - 1)

const T = (0, 2\*pi) const T\_2 = (-pi, pi)

function F(u, p, t) return u / sqrt(n\*n - 1) end

problem = ODEProblem(F, r0, T)

result = solve(problem, abstol=1e-8, reltol=1e-8) @show result.u @show result.t

dxR = rand(1:size(result.t)[1]) rAngles = [result.t[dxR] for i in 1:size(result.t)[1]]

plt = plot(proj=:polar, aspect\_ratio=:equal, dpi = 1000, legend=true, bg=:white)

plot!(plt, xlabel=“theta”, ylabel=“r(t)”, title=“Хусаинова Задача о погоне. Случай номер 1”, legend=:outerbottom) plot!(plt, [rAngles[1], rAngles[2]], [0.0, result.u[size(result.u)[1]]], label=“Путь лодки”, color=:blue, lw=1) scatter!(plt, rAngles, result.u, label=““, mc=:blue, ms=0.0005) plot!(plt, result.t, result.u, xlabel=”theta”, ylabel=“r(t)”, label=“Путь катера”, color=:green, lw=1) scatter!(plt, result.t, result.u, label=““, mc=:green, ms=0.0005)

savefig(plt, “lab2\_01.png”)

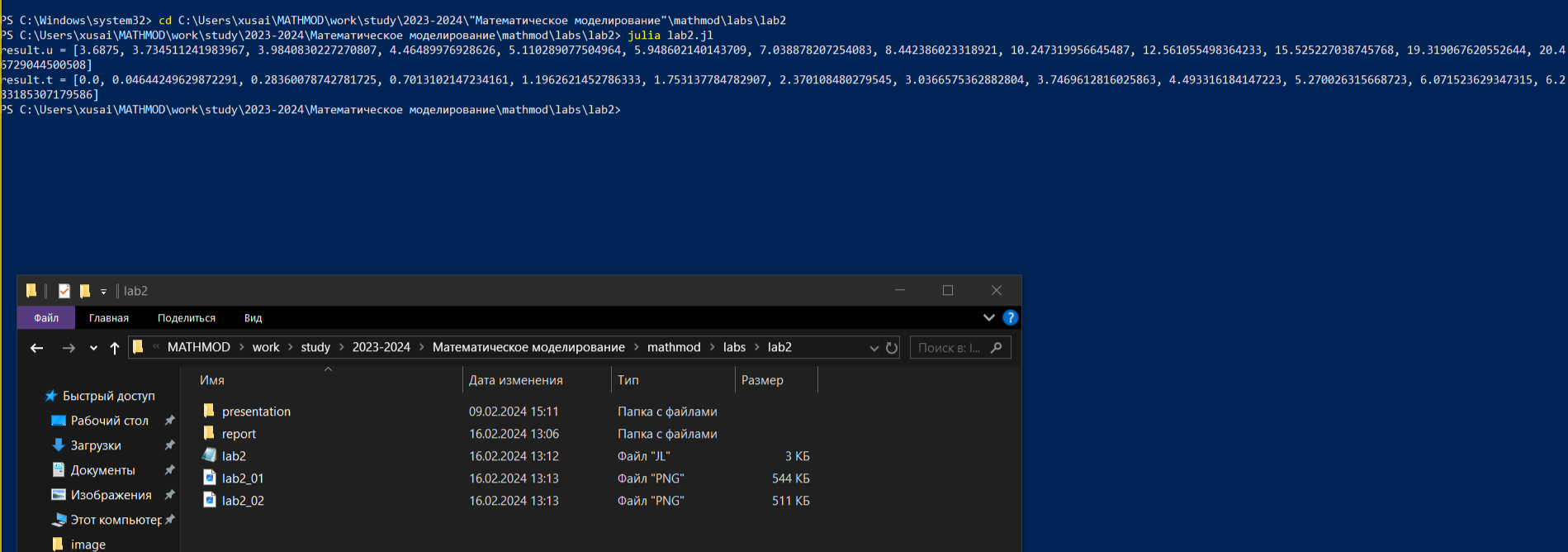
problem = ODEProblem(F, r0\_2 , T\_2) result = solve(problem, abstol=1e-8, reltol=1e-8) dxR = rand(1:size(result.t)[1]) rAngles = [result.t[dxR] for i in 1:size(result.t)[1]]

plt1 = plot(proj=:polar, aspect\_ratio=:equal, dpi = 1000, legend=true, bg=:white)

plot!(plt1, xlabel=“theta”, ylabel=“r(t)”, title=“Хусаинова Задача о погоне. Случай номер 2”, legend=:outerbottom) plot!(plt1, [rAngles[1], rAngles[2]], [0.0, result.u[size(result.u)[1]]], label=“Путь лодки”, color=:blue, lw=1) scatter!(plt1, rAngles, result.u, label=““, mc=:blue, ms=0.0005) plot!(plt1, result.t, result.u, xlabel=”theta”, ylabel=“r(t)”, label=“Путь катера”, color=:green, lw=1) scatter!(plt1, result.t, result.u, label=““, mc=:green, ms=0.0005)

savefig(plt1, “lab2\_02.png”)

1. Запуск кода ( рис. 8).



Запуск кода

1. Просмотр результата работы ( рис. 9-10).



Случай 1



Случай 2

# Выводы

Были изучены основы языков программирования Julia и OpenModelica. Освоены библиотеки этих языков, которые используются для построения графиков и решения дифференциальных уравнений. Поскольку OpenModelica не работает с полярными координатами, она пока что не была использована в данной лабораторной работе.

# Список литературы

[1] Документация по Julia: https://docs.julialang.org/en/v1/

[2] Документация по OpenModelica: https://openmodelica.org/

[3] Решение дифференциальных уравнений: https://www.wolframalpha.com/