

Отчёт по лабораторной работе №2

Математическое моделирование

Задача о погоне. Вариант №30

Выполнила: Малащенко Марина Владимировна,
НФИбд-01-20, 1032202459

Содержание

| | |
|---|-----------|
| Цель работы | 4 |
| Теоретическое введение | 5 |
| Задание | 7 |
| Задачи: | 8 |
| Выполнение лабораторной работы | 9 |
| Математическая модель | 9 |
| Решение с помощью программ | 11 |
| OpenModelica | 11 |
| Julia | 11 |
| Результаты работы кода на Julia | 15 |
| Анализ полученных результатов | 18 |
| Вывод | 19 |
| Список литературы. Библиография | 20 |

Список иллюстраций

| | | |
|---|--|----|
| 1 | (рис. 1. Формула вычисления варианта и её вывод) | 7 |
| 1 | “Установщик Julia.exe” | 12 |
| 2 | “Проверка установки библиотек” | 13 |
| 3 | “Компляция программы lab02.jl” | 15 |
| 4 | “Полученный график. Первый случай” | 16 |
| 5 | “Полученный график. Второй случай” | 17 |

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 вообще, 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

Задания лабораторной работы разделены по вариантам. Мой вариант 30

(исходя из формулы $N_{student} \bmod K_{ofvariants} + 1$).

Этот же вариант будет использоваться для всех последующих лабораторных работ.

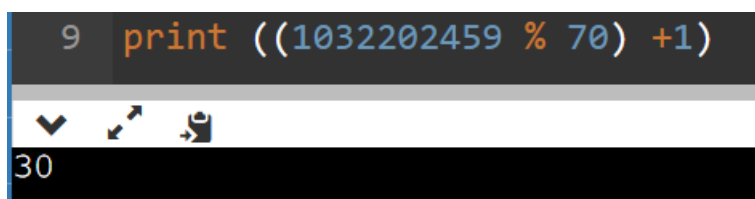
A screenshot of a code editor with a dark background. The first line of code is `9 print ((1032202459 % 70) + 1)`. Below the code, there are three small icons: a checkmark, a double-headed arrow, and a cursor. The output of the code, the number 30, is displayed in a separate box at the bottom.

Рис. 1: (рис. 1. Формула вычисления варианта и её вывод)

Задача о погоне. Вариант 30:

На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 12,2 км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в 4,1 раза больше скорости браконьерской лодки.

Задачи:

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

Выполнение лабораторной работы

Математическая модель

1. 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 $(12,2; 0)$. Обозначим скорость лодки v .
2. 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.
3. Чтобы найти расстояние x (расстояние после которого катер начнет двигаться вокруг полюса), необходимо составить следующие уравнение. Пусть через время t катер и лодка окажутся на одном расстоянии x от полюса. За это время лодка пройдет x , а катер $12,2 + x$ (или $12,2 - x$, в зависимости от начального положения катера относительно полюса). Время, за которое они пройдут это расстояние, вычисляется как $\frac{x}{v}$ или $\frac{12,2-x}{4,1v}$ ($\frac{12,2+x}{4,1v}$). Так как время должно быть одинаковым, эти величины тоже будут друг другу равны. Из этого получаем объединение из двух уравнений (двух из-за двух разных изначальных позиций катера относительно полюса):

$$\begin{cases} \frac{x}{v} = \frac{12,2-x}{4,1v} \\ \frac{x}{v} = \frac{12,2+x}{4,1v} \end{cases}$$

Из данных уравнений можно найти расстояние, после которого катер начнёт раскручиваться по спирали. Для данных уравнений решения будут следующими: $x_1 = \frac{122}{51}$, $x_2 = \frac{122}{31}$. Задачу будем решать для двух случаев. После того, как катер береговой охраны окажется на одном расстоянии от полюса, что и лодка, он должен сменить прямолинейную траекторию и начать двигаться вокруг полюса удаляясь от него со скоростью лодки v . Для этого скорость катера раскладываем на две составляющие: $v_r = \frac{dr}{dt} = v$ - радиальная скорость и $v_\tau = r \frac{d\theta}{dt}$ - тангенциальная скорость.

$$v_\tau = \frac{\sqrt{1581}v}{10}$$

4. Решение исходной задачи сводится к решению системы из двух дифференциальных уравнений:

$$\begin{cases} \frac{dr}{dt} = v \\ r \frac{d\theta}{dt} = \frac{\sqrt{1581}v}{10} \end{cases}$$

с начальными условиями

$$\begin{cases} \theta_0 = 0 \\ r_0 = x_1 = \frac{122}{51} \end{cases}$$

или

$$\begin{cases} \theta_0 = -\pi \\ r_0 = x_2 = \frac{122}{31} \end{cases}$$

Исключая из полученной системы производную по t , можно перейти к следующему уравнению (с неизменными начальными условиями):

$$\frac{dr}{d\theta} = \frac{10r}{\sqrt{1581}}$$

Решением этого уравнения с заданными начальными условиями и будет являться траектория движения катера в полярных координатах. [3]

Решение с помощью программ

OpenModelica

К сожалению, OpenModelica не адаптирована к использованию полярных координат, поэтому адекватное отображение результатов данной задачи там невозможно. [2]

Julia

Программный код решения на Julia

Решить дифференциальное уравнение, расписанное в постановке задачи лабораторной работы, поможет библиотека DifferentialEquations. Итоговые изображения в полярных координатах будут строиться через библиотеку Plots. [1]

Установим Julia:

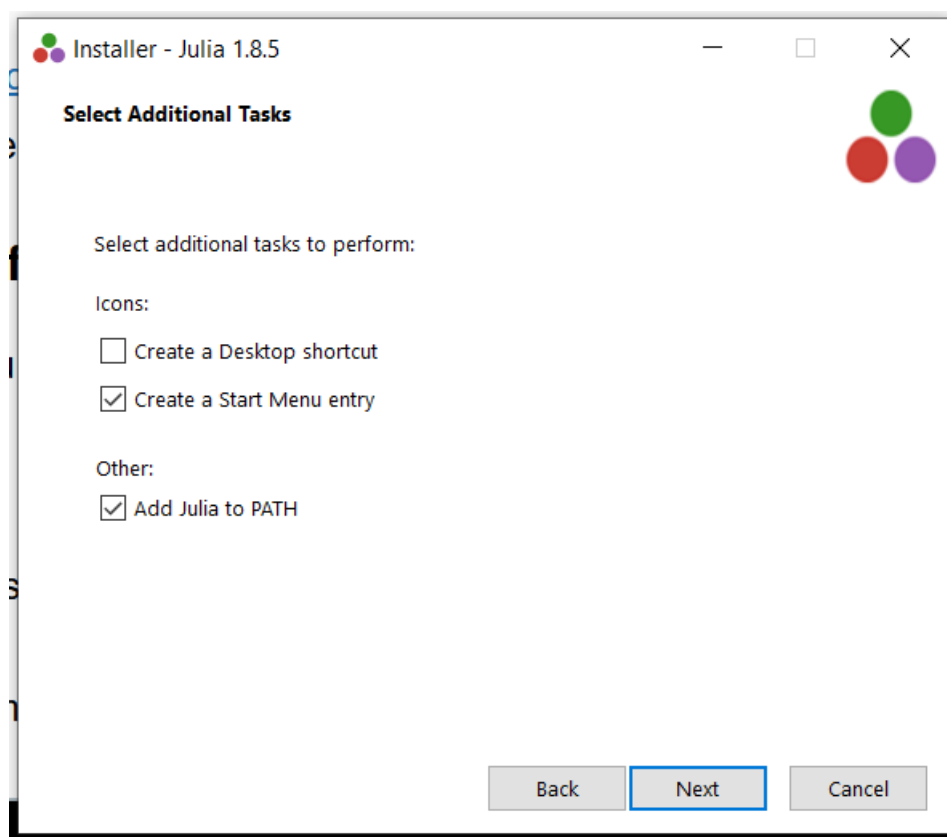


Рис. 1: “Установщик Julia.exe”

Установим нужные библиотеки, проверим их установку:

```
Julia 1.10.0-DEV

Documentation: https://docs.julialang.org
Type "?" for help, "]"? for Pkg help.
Version 1.10.0-DEV.656 (2023-02-24)
Commit f0eadd076f (0 days old master)

julia> using Plots
julia> using DifferentialEquations
julia>
```

Рис. 2: “Проверка установки библиотек”

Код программы:

```
using Plots
using DifferentialEquations

# расстояние от лодки до катера
const a = 12.2
const n = 4.1

# расстояние начала спирали
const r0 = a/(n + 1)
const r0_2 = a/(n - 1)
# интервал
const T = (0, 2*pi)
const T_2 = (-pi, pi)

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

```

end

# задача ОДУ
problem = ODEProblem(F, r0, T)

#решение
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]]

#холст1
plt = plot(proj=:polar, aspect_ratio=:equal, dpi = 1000, legend=true, bg=:white)

#параметры для холста
plot!(plt, xlabel="theta", ylabel="r(t)", title="Задача о погоне - случай 1", leg
plot!(plt, [rAngles[1], rAngles[2]], [0.0, result.u[size(result.u)[1]]], label="r
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"
scatter!(plt, result.t, result.u, label="", mc=:green, ms=0.0005)

savefig(plt, "lab02_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]]

```

```

#холст2
plt1 = plot(proj=:polar, aspect_ratio=:equal, dpi = 1000, legend=true, bg=:white)

#параметры для холста
plot!(plt1, xlabel="theta", ylabel="r(t)", title="Задача о погоне - случай 2", legend=:none)
plot!(plt1, [rAngles[1], rAngles[2]], [0.0, result.u[size(result.u)[1]]], label="r(t)")
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")
scatter!(plt1, result.t, result.u, label="", mc=:green, ms=0.0005)

savefig(plt1, "lab02_02.png")

```

Скомпилируем файл командной в PShell:

```
PS C:\Users\Марина\Documents\2022-2023\Математическое моделирование\mathmod\labs\lab02> julia lab02.jl
```

Рис. 3: “Компляция программы lab02.jl”

Результаты работы кода на Julia

На рис. @fig:005 и @fig:006 изображены итоговые графики траектории движения катера и лодки для случая обоих случаев.

Задача о погоне - случай 1

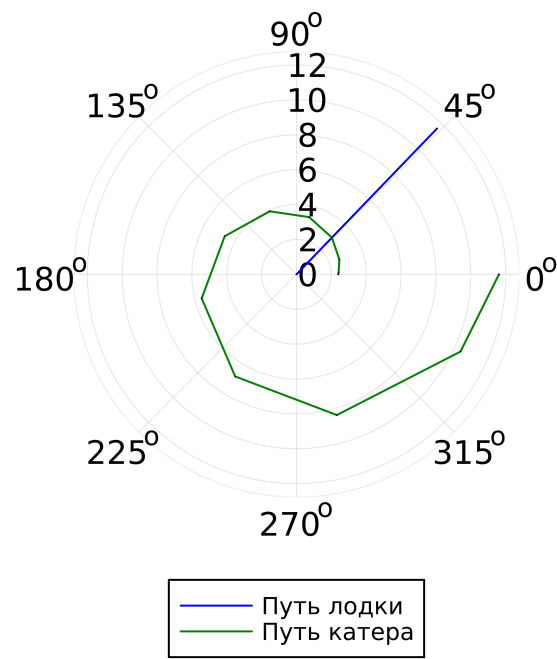


Рис. 4: “Полученный график. Первый случай”

Задача о погоне - случай 2

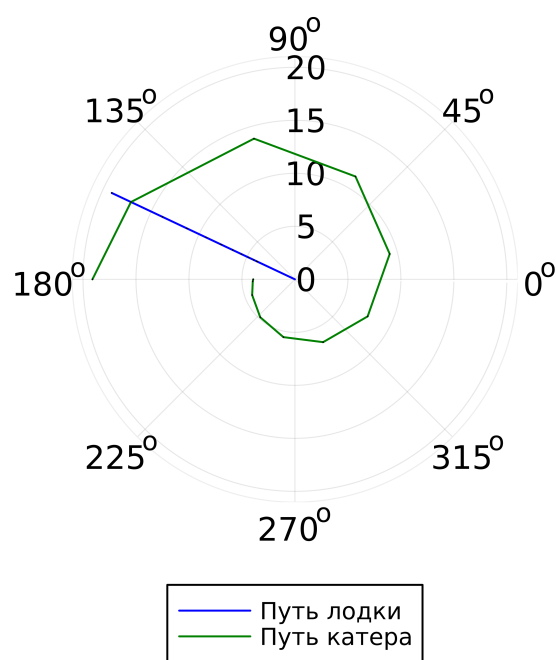


Рис. 5: “Полученный график. Второй случай”

Анализ полученных результатов

Мною были построены графики для обоих случаев. На них получилось отрисовать траекторию катера, траекторию лодки и получилось наглядно найти их точки пересечения. Мы успешно решили задачу о погоне.

Вывод

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. Библиография

- [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/>