Отчёт по лабораторной работе №2  
Математическое моделирование

Задача о погоне. Вариант №39

Выполнила: Прасолов Валерий Сергеевич,  
НФИбд-01-20, 1032202459

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:**

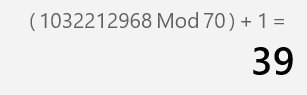
* Тангенциальная скорость - составляющая вектора скорости, перпендикулярная линии, соединяющей источник и наблюдателя. Измеряется собственному движению - угловому перемещению источника.
* Радиальная скорость — проекция скорости точки на прямую, соединяющую её с выбранным началом координат.
* Полярная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

# Задание

Задания лабораторной работы разделены по вариантам. Мой вариант 39

(исходя из формулы ).

Этот же вариант будет использоваться для всех последующих лабораторных работ.



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

Задача о погоне. Вариант 39:

**На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии 21 км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в 5,5 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 .
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 , а катер (или , в зависимости от начального положения катера относительно полюса). Время, за которое они пройдут это расстояние, вычисляется как $x\over v$ или ${12,2 - x}\over{4,1v}$ (${12,2 + x}\over{4,1v}$). 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):

$$ \left[ \begin{array}{cl}
{{x}\over{v}} = {{21 - x}\over{5,5v}}\\
{{x}\over{v}} = {{21 + x}\over{5,5v}}
\end{array} \right. $$

Из данных уравнений можно найти расстояние, после которого катер начнёт раскручиваться по спирали. Для данных уравнений решения будут следующими: $x\_1 = {{122}\over{51}}$, $x\_2 = {{122}\over{31}}$. 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: $v\_r = {dr\over dt} = v$ - радиальная скорость и $v\_\tau = r{d\theta\over dt}$ - тангенциальная скорость.

$$ v\_\tau = {{\sqrt{1581}v}\over{10}} $$

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

$$ \left\{ \begin{array}{cl}
{dr\over dt} = v \\
r{d\theta\over dt} = {{\sqrt{1581}v}\over{10}}
\end{array} \right. $$

с начальными условиями

$$ \left\{ \begin{array}{cl}
\theta\_0 = 0 \\
r\_0 = x\_1 = {{122}\over{51}}
\end{array} \right. $$

или

$$ \left\{ \begin{array}{cl}
\theta\_0 = -\pi \\
r\_0 = x\_2 = {{122}\over{31}}
\end{array} \right. $$

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

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

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

## Решение с помощью программ

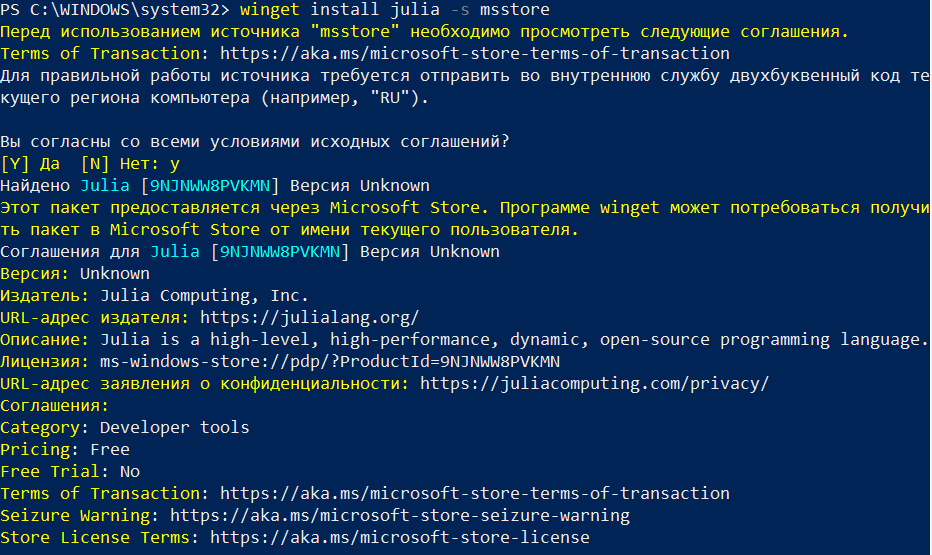
### OpenModelica

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

### Julia

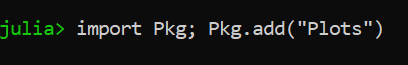
Решить дифференциальное уравнение, расписанное в постановке задачи лабораторной работы, поможет библиотека DifferentialEquations. Итоговые изображения в полярных координатах будут строиться через библиотеку Plots. [1]

Установим Julia:



“Установщик Julia.exe”

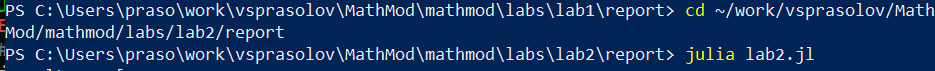
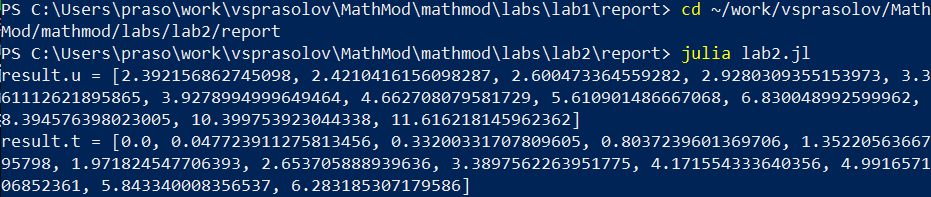
Установим нужные библиотеки, проверим их установку:

 “Проверка установки библиотек”

Код программы:

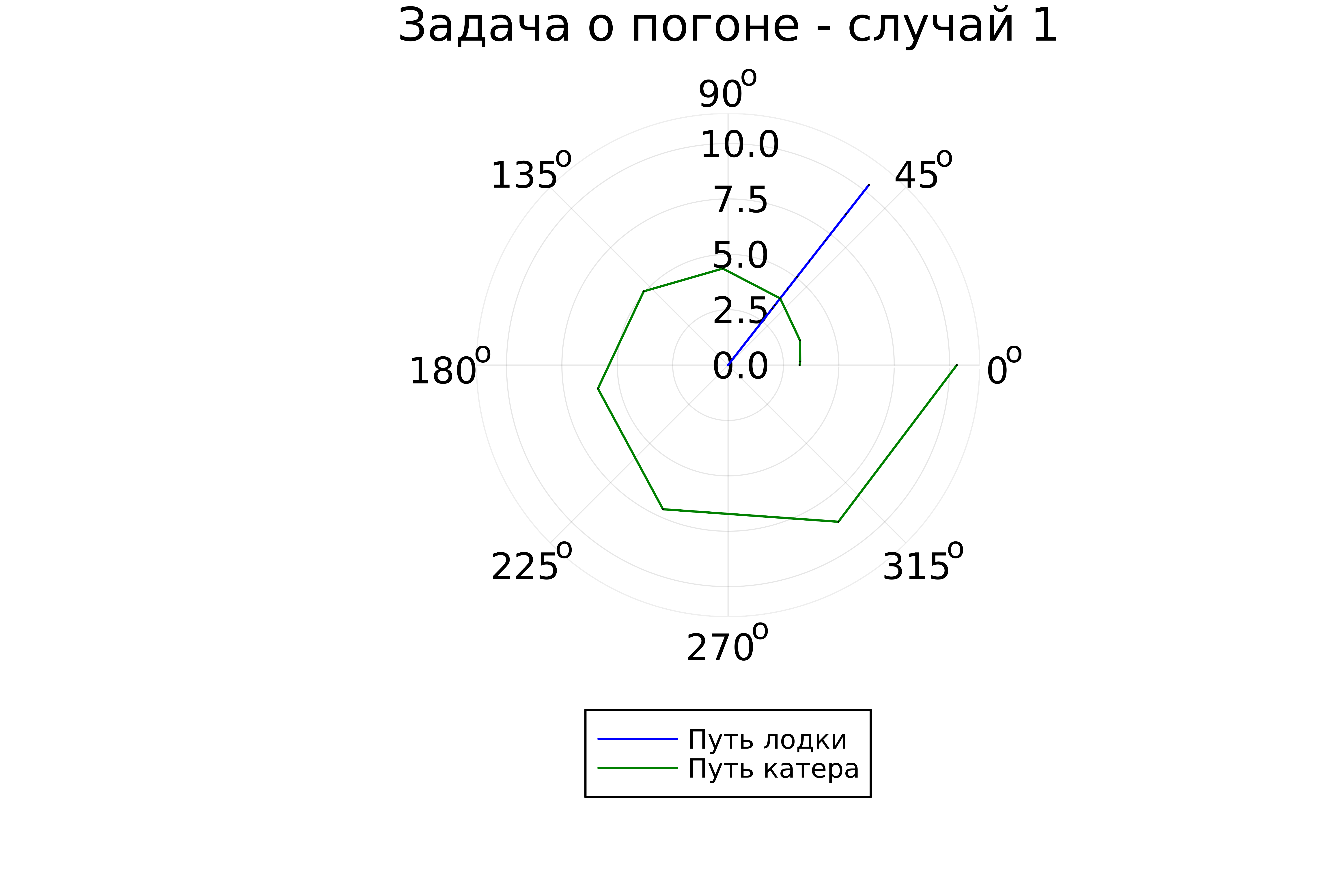
using Plots  
using DifferentialEquations  
  
# расстояние от лодки до катера  
const a = 21  
const n = 5.5  
  
# расстояние начала спирали  
const r0 = a/(n + 1)  
const r0\_2 = a/(n - 1)  
# интервал  
const T = (0, 2\*pi)  
const T\_2 = (-pi, pi)  
  
function F(u, p, t)  
 return u / sqrt(n\*n - 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", 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, "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]]  
  
#xoлст2  
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=: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, "lab02\_02.png")

Скомпилируем файл командной в PShell:

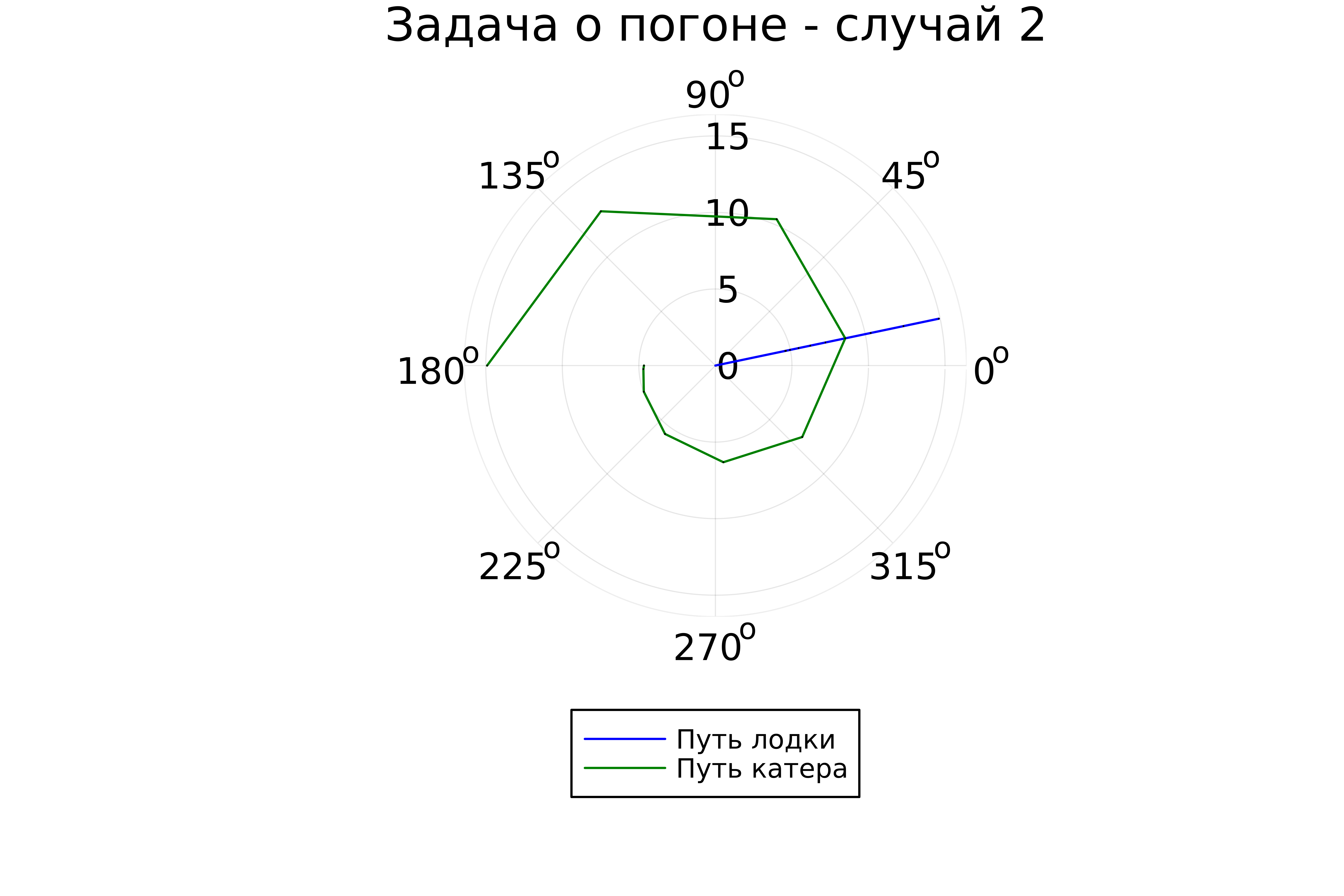
 

### Результаты работы кода на Julia

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



“Полученный график. Первый случай”



“Полученный график. Второй случай”

# Анализ полученных результатов

Мною были построены графики для обоих случаев. На них получилось отрисовать трактерию 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/