Лабораторная работа №2

Задача о погоне

Ильин Андрей Владимирович

Содержание

# Цель работы

Решить задачу о погоне. Смоделировать кривую погони средствами Julia и OpenModelica.

# Задачи

1. Провести аналогичные рассуждения и вывод дифференциальных уравнений, если скорость катера больше скорости лодки в n раз.
2. Построить траекторию движения катера и лодки для двух случаев.
3. Определить по графику точку пересечения катера и лодки

# Термины

* Задача о погоне — классических задача из области дифференциальных уравнений. [1]
* Кривая погони — кривая, представляющая собой решение задачи о «погоне». [2]
* Julia – это открытый свободный высокопроизводительный динамический язык высокого уровня, созданный специально для технических (математических) вычислений. Его синтаксис близок к синтаксису других сред технических вычислений, таких как Matlab и Octave. [3]
* OpenModelica — свободное открытое программное обеспечение для моделирования, симуляции, оптимизации и анализа сложных динамических систем. Основано на языке Modelica. [4]

# Теоретическая справка

Для построения кривой погони, нам необходимо знать начальные условия и уравнения кривой. Ориентируясь на рассуждения из пособия к лабораторной работы[1] можно вывести общие формулы. Благодаря общим формулам можно будет написать программу, которая будет строить разные кривые погони в зависимости от исходных данных (расстояния и разницы в скорости).

Пусть

* n - разница в скорости, то есть скорость катера в n раз больше лодки;
* a - расстояние между катером и лодкой в момент рассеивания тумана.

Тогда общие начальные условия для первого случая выглядят следующим образом:

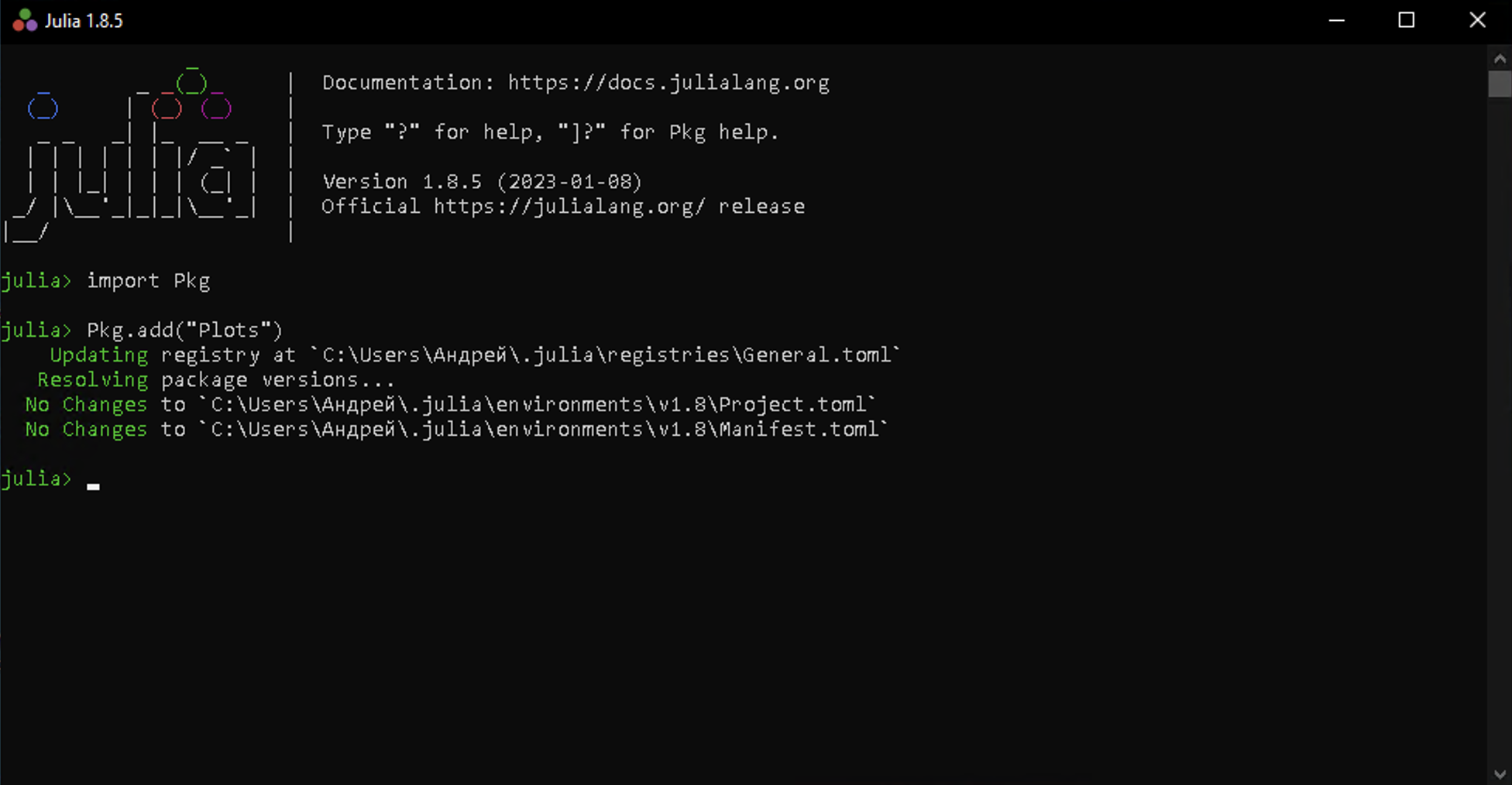
Для второго случая:

Уравнение кривой в общем случае выглядит следующим образом:

# Выполнение лабораторной работы.

1. Установим пакет в Julia необходимый для построения графика. (Рис. 1)

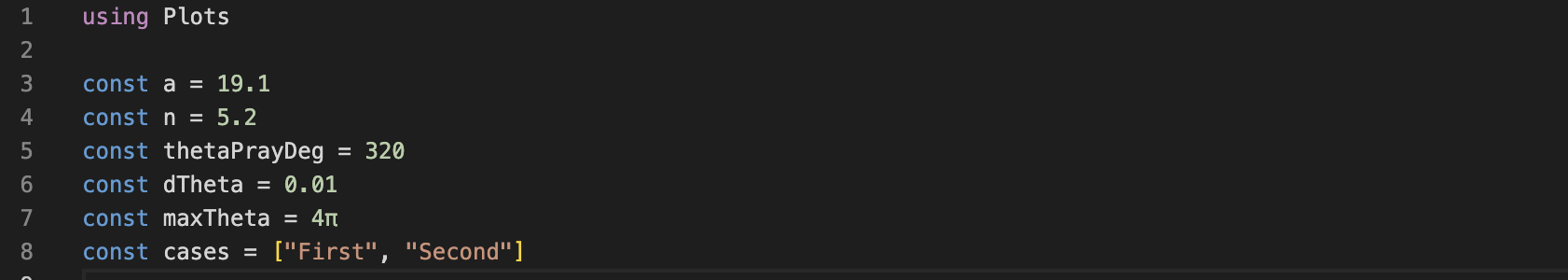
import Pkg  
Pkg.add("Plots")



Установка пакета ‘Plots’

1. Приступим к написанию скрипта. Первым делом подкючим пакет “Plots” и объявим необходимые константы. Константа a и n - являются входными данными, thetaPrayDeg - сторона в которую поплыла лодка, dTheta - шаг для равномерного разбиения периода, maxTheta - длина периода построения, cases - содержит два кейса, по которым мы будем итерироваться. (Рис. 2)

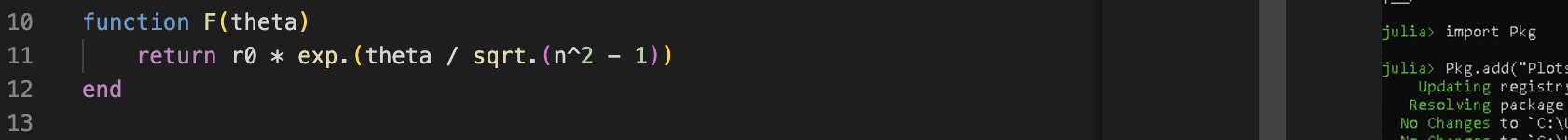
using ... # используется для поключения пакета  
const ... # используется для объявления константы



Начало написания скрипта

1. Напишем функцию, которая будет являтся уравнением нашей кривой. (Рис. 3)

# синтаксис функции  
function funcName(args)  
 #script  
end



Уравнение кривой

1. Дальше откроем цикл, внутри которого будем итерироваться по двум кейсам (в данных кейсах разные начальные условия). Для написания цикла используем следующую конструкцию (Рис. 4):

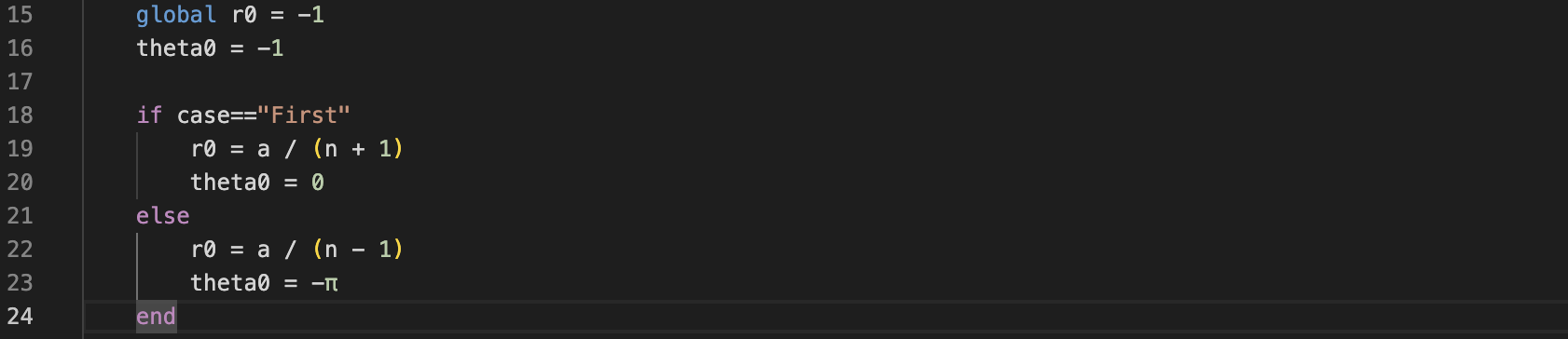
# синтаксис цикла  
for item in list  
 #script  
end

Открытие цикла

Открытие цикла

1. Внутри цикла сначла определим начальные условия в зависимости от кейса. Для этого воспользуемся ветвлением (Рис. 5):

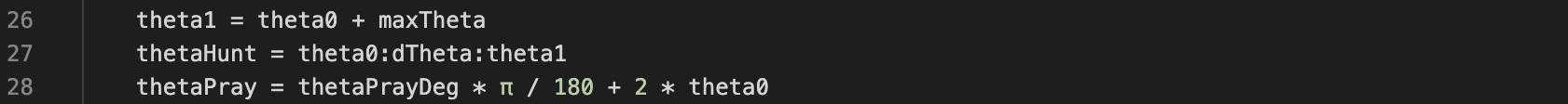
# синтаксис ветвления  
if condition  
 ...  
else  
 ...  
end



Выбор начального условия

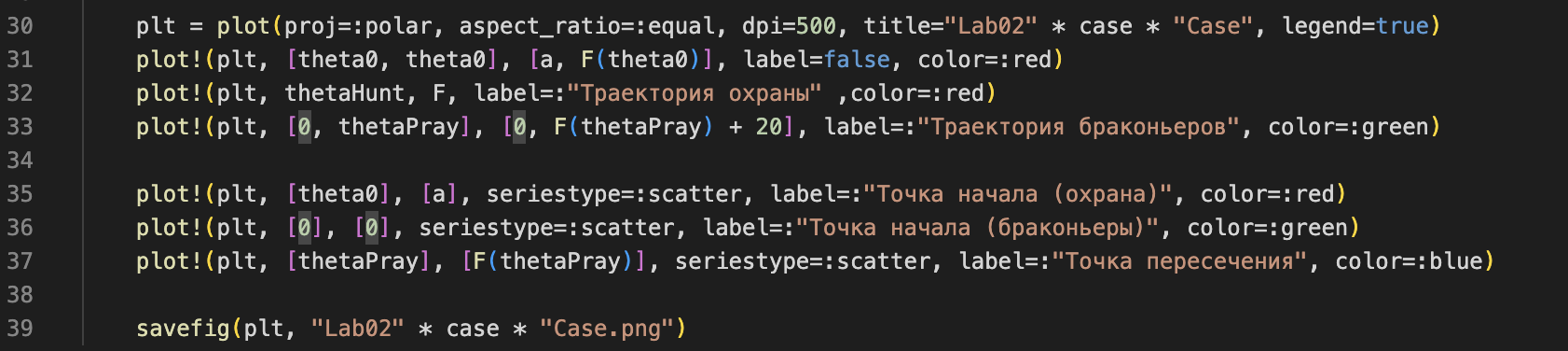
1. После этого нам необходимо равномерно разбить наш период на точки (чем ментше расстояние между точками, тем кривая погони). Также необходимо перевести градус побега лодки в радианы с учетом начального условия. (Рис. 6)

# синтаксис разбиения  
linspace = start:step:end



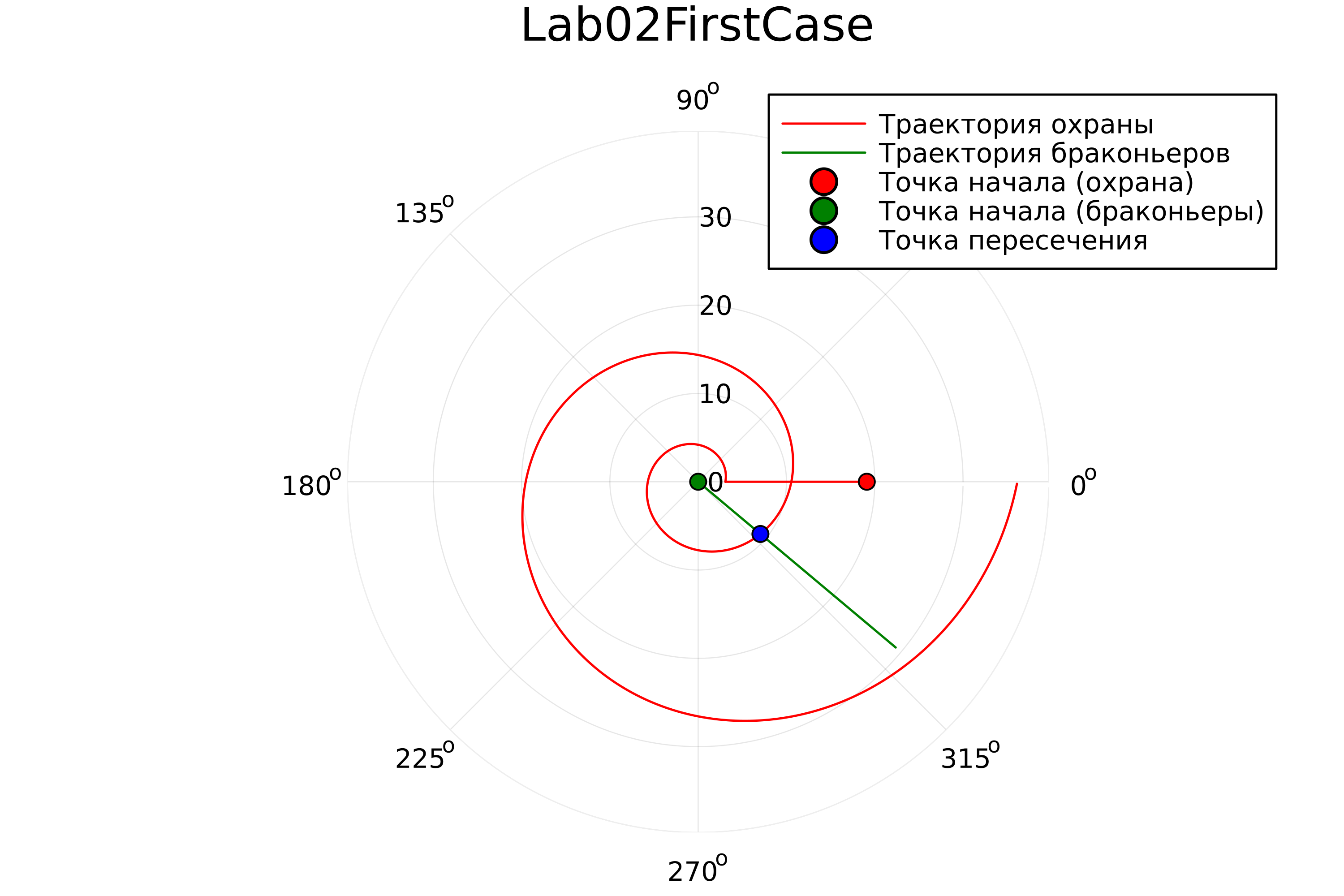
Выбор начального условия

1. Построим необходимые графики и сохраним “полотно”. (Рис. 7)

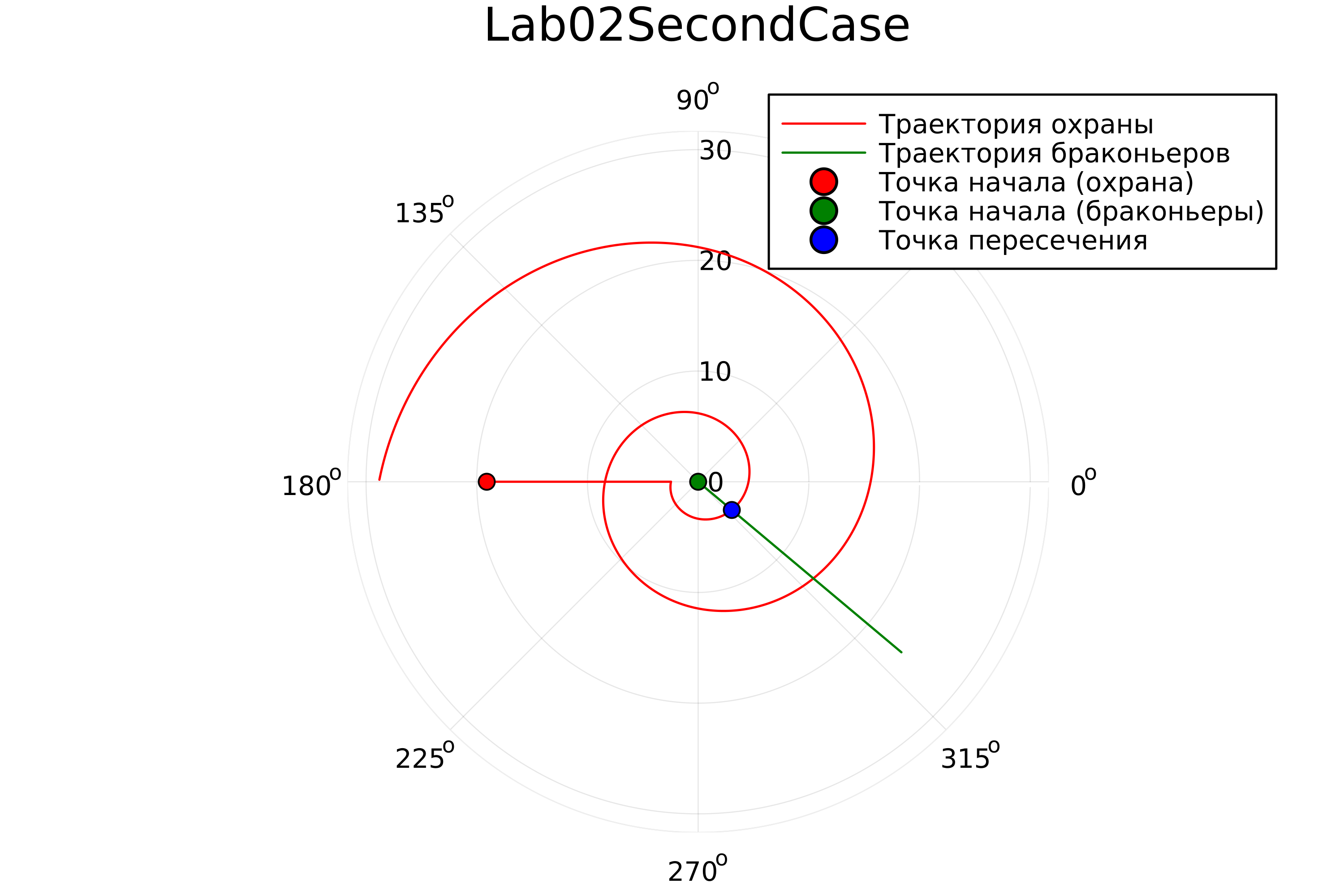


Скрипт для построения графиков

1. Просмотрим вывод программы (Рис. 8 - 9)



Кривая погони №1



Кривая погони №2

1. Построение кривой погони для данной задачи выполнять на OpenModelica стандартными инструментами невозможно. Так как в стандартные инструменты не выходит построение графиков в полярных координатах, в связи с этим данную лабораторную работу выполнять на OpenModelica нет необходимости.

# Листинг

using Plots  
  
const a = 19.1  
const n = 5.2  
const thetaPrayDeg = 320  
const dTheta = 0.01  
const maxTheta = 4pi  
const cases = ["First", "Second"]  
  
function F(theta)  
 return r0 \* exp.(theta / sqrt.(n^2 - 1))  
end  
  
for case in cases  
 global r0 = -1  
 theta0 = -1  
  
 if case=="First"  
 r0 = a / (n + 1)  
 theta0 = 0  
 else  
 r0 = a / (n - 1)  
 theta0 = -pi  
 end  
  
 theta1 = theta0 + maxTheta  
 thetaHunt = theta0:dTheta:theta1  
 thetaPray = thetaPrayDeg \* pi / 180 + 2 \* theta0  
  
 plt = plot(proj=:polar, aspect\_ratio=:equal, dpi=500, title="Lab02" \* case \* "Case", legend=true)  
 plot!(plt, [theta0, theta0], [a, F(theta0)], label=false, color=:red)  
 plot!(plt, thetaHunt, F, label=:"Траектория охраны" ,color=:red)  
 plot!(plt, [0, thetaPray], [0, F(thetaPray) + 20], label=:"Траектория браконьеров", color=:green)  
  
 plot!(plt, [theta0], [a], seriestype=:scatter, label=:"Точка начала (охрана)", color=:red)  
 plot!(plt, [0], [0], seriestype=:scatter, label=:"Точка начала (браконьеры)", color=:green)  
 plot!(plt, [thetaPray], [F(thetaPray)], seriestype=:scatter, label=:"Точка пересечения", color=:blue)  
  
 savefig(plt, "Lab02" \* case \* "Case.png")  
end

# Анализ результатов

Работа выполненна без непредвиденных проблем в соответствии с руководством. Ошибок и сбоев не произошло.

# Выводы

Мы улучшили практичесские навыки в области дифференциальных уравнений, а также приобрели нвыки моделирования на Julia.

# Библиография

* [1] Задача о погоне (https://esystem.rudn.ru/mod/resource/view.php?id=967233)
* [2] Кривая погони (https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%B2%D0%B0%D1%8F\_%D0%BF%D0%BE%D0%B3%D0%BE%D0%BD%D0%B8)
* [3] Julia (http://www.unn.ru/books/met\_files/JULIA\_tutorial.pdf)
* [4] OpenModelica (https://ru.wikipedia.org/wiki/OpenModelica)