Отчёт по лабораторной работе №2

Математическое моделирование

Чекалова Лилия Руслановна

Содержание

Список таблиц

Список иллюстраций

Цель работы

- Построение математической модели для выбора правильной стратегии при решении задач поиска, в частности, задачи о погоне
- Визуализация полученной модели с помощью средств языков Julia и OpenModelica

Задание

- Провести вывод дифференциальных уравнений в соответствии с заданными условиями
- Построить траекторию движения катера и лодки для двух случаев
- Определить по графику точку пересечения катера и лодки

Теоретическое введение

Будем рассматривать задачу следующего содержания: На море в тумане катер береговой охраны преследует лодку браконьеров. Через определенный промежуток времени туман рассеивается, и лодка обнаруживается на расстоянии k км от катера. Затем лодка снова скрывается в тумане и уходит прямолинейно в неизвестном направлении. Известно, что скорость катера в n раза больше скорости браконьерской лодки. Необходимо определить по какой траектории необходимо двигаться катеру, чтобы нагнать лодку.

В варианте 65 k=18.4, n=4.6.

Полагаем $t_0 = 0$, $x_{л0} = 0$ - место нахождения лодки браконьеров в момент обнаружения, $x_{к0} = k$ - место нахождения катера береговой охраны относительно лодки браконьеров в момент обнаружения лодки.

Вводим полярные координаты. Считаем, что полюс - это точка обнаружения лодки браконьеров $x_{л0}$ (theta = $x_{л0} = 0$), а полярная ось r проходит через точку нахождения катера береговой охраны.

Траектория катера должна быть такой, чтобы и катер, и лодка все время были на одном расстоянии от полюса theta, 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.

Чтобы найти расстояние x (расстояние после которого катер начнет двигаться вокруг полюса), необходимо составить простое уравнение. Пусть через время t катер и лодка окажутся на одном расстоянии x от полюса. За это время лодка пройдет x, а катер k-x (или k+x, в зависимости от начального положения катера относительно полюса). Время, за которое они пройдут это расстояние, вычисляется как x/v или (k-x)/(nv) (во втором случае (k+x)/(nv)). Так как время одно и то же, то эти величины одинаковы. Тогда неизвестное расстояние x можно найти из следующего уравнения:

x/v = (k-x)/(nv) в первом случае или x/v = (k+x)/(nv) во втором. Отсюда находим x_1 и x_2.

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 - радиальная скорость и v_t - тангенциальная скорость. Радиальная скорость - это скорость, с которой катер удаляется от полюса, v_r = dr/dt = v, так как нам нужно, чтобы эта скорость была равна скорости лодки.

Тангенциальная скорость – это линейная скорость вращения катера относительно полюса, v_t = r(dtheta/dt).

Мы вычисляем тангенциальную скорость, получаем систему дифференциальных уравнений с начальными условиями и в результате преобразований получаем уравнение (рис. @fig:001). Решив это уравнение мы получим траекторию движения катера в полярных координатах.

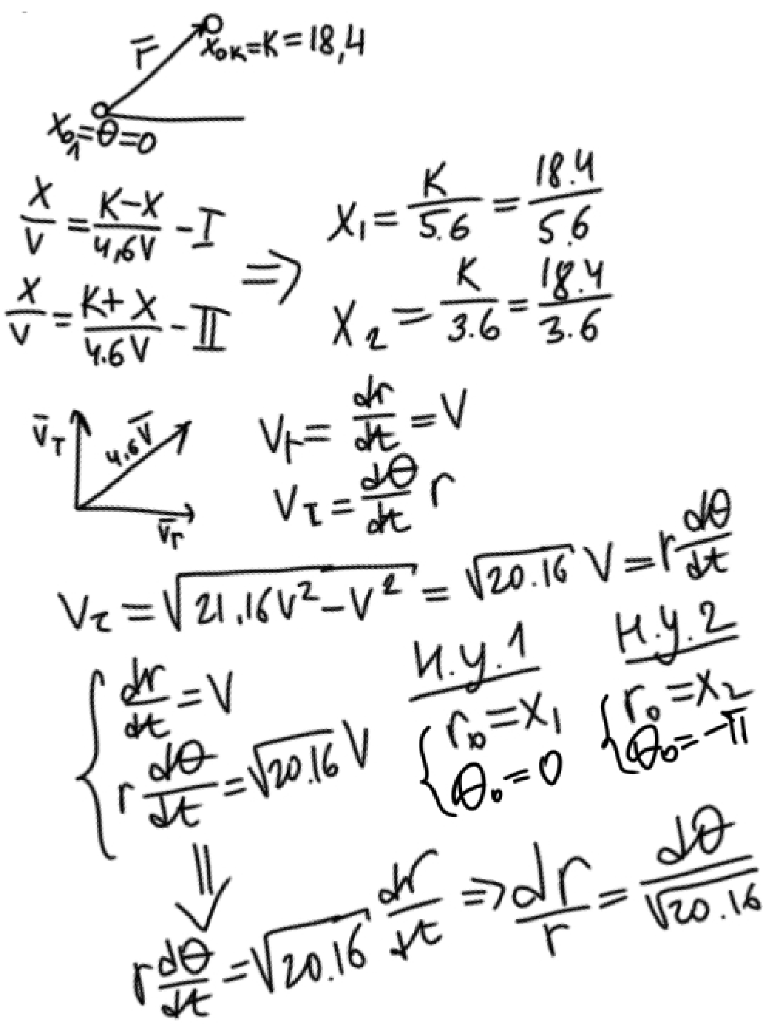

Рис. 0.1: Выведение итогового уравнения из рассуждений

Более подробно см. в [1].

Выполнение лабораторной работы

Решаем выведенное нами ранее дифференциальное уравнение с разделяющими переменными и подставляем начальные условия, чтобы вычислить константу для каждого случая (рис. @fig:002).

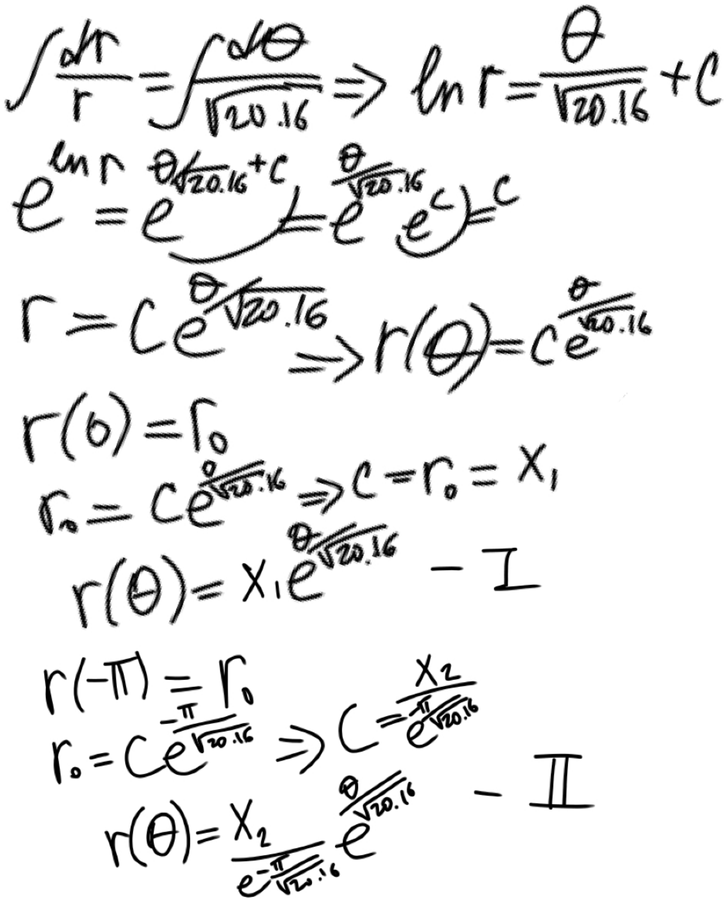

Рис. 0.1: Решение уравнения

Используя библиотеку Plots, пишем программу на Julia (рис. @fig:003).

```
using Plots

fi = 3pi/4
fi0_1 = 0
r0_1 = 18.4/5.6
fi0_2 = -pi
r0_2 = 18.4/3.6

function F1(theta, r0, fi0)
    return r0/exp(fi0/sqrt(20.16))*exp(theta/sqrt(20.16))
end

th = collect(0:0.01:2pi)
rth = F1.(th, r0_1, fi0_1)

r = F1.(fi, r0_1, fi0_1)

x = r*cos(fi)
y = r*sin(fi)

@show x
@show y

plt = plot(
    proj=:polar,
    aspect_ratio=:equal,
    dpi=300,
    legend=true,
    title="Задача о погоне"
)

plot!(plt,
th,
rth,
xlabel="theta",
ylabel="r(theta)",
color=:red,
label="Траектория катера")

plot!(plt,
[fi, fi],
[0, 20],
xlabel="fi",
ylabel="r",
color=:green,
label="Траектория лодки")
```

Рис. 0.2: Программа на Julia

В начале программы указываем начальные условия, затем описываем функцией решенное нами уравнение и передаем в нее данные для построения графика. Для нахождения точки пересечения пользуемся формулами перевода из полярной системы координат в декартовую (рис. @fig:004).

$$\begin{cases} x = r\cos\varphi \\ y = r\sin\varphi \end{cases}$$

Рис. 0.3: Формулы для перевода из полярной системы координат

Рисуем графики, находим точку пересечения и сохраняем полученное изображение в файл (рис. @fig:005).

```
plot!(plt,
[fi],
[r],
seriestype=:scatter,
color=:blue,
label="Точка пересечения катера и лодки")

savefig(plt, "lab2_1.png")
```

Рис. 0.4: Нахождение точки пересечения и сохранение в файл

Исследовав график (рис. @fig:006), получаем, что для первого случая координаты точки пересечения равны (-3.9, 3.9).

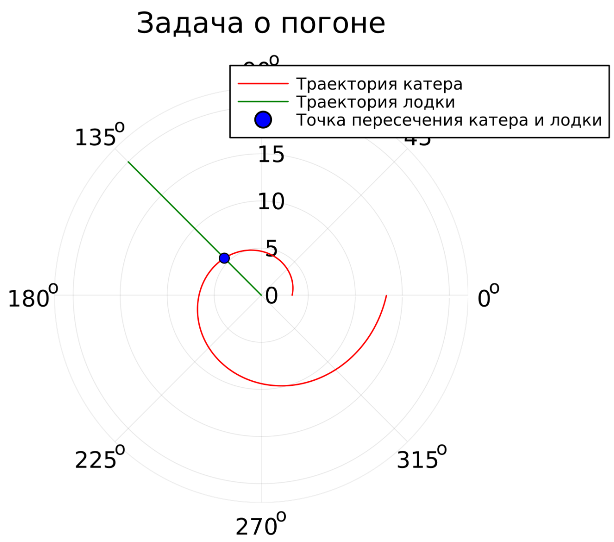

Рис. 0.5: График Julia, первый случай

Для второго случая меняем входные данные для нашей функции, описывающей уравнение (рис. @fig:007).

```
th = collect(-pi:0.01:2pi)
rth = F1.(th, r0_2, fi0_2)
r = F1.(fi, r0_2, fi0_2)
```

Рис. 0.6: Замена входных данных

Рассмотрев график для второго случая (рис. @fig:008), получаем координаты точки пересечения, равные (-12.3, 12.3).

Рис. 0.7: График Julia, второй случай

Далее создаем модель на OpenModelica (рис. @fig:009). Задаем необходимые начальные условия в виде параметров, а нужные нам уравнения указываем в разделе equation, в том числе уравнения перевода из полярной системы координат в декартовую.

```
model Pursuit
parameter Real a=sqrt(20.16);
parameter Real k1= 18.4/5.6;
parameter Real k2 = 18.4/3.6;
constant Real pi = 3.14;
Real theta(start=-pi/2);
Real r(start=k1);
// Real r(start=k2);
// Real theta(start=-3*pi/2);
Real x(start=0);
Real y(start=0);
Real r1(start=0);
Real f1(start=0);
equation
x = time;
y = -x;
der(r) = 1;
der(theta) = a / r;
r1 = r*cos(theta);
f1 = -r*sin(theta);
end Pursuit;
```

Рис. 0.8: Программа на OpenModelica

Настраиваем параметры симуляции, указываем начальное, конечное время и число интервалов (рис. @fig:010) и формат вывода (рис. @fig:011).

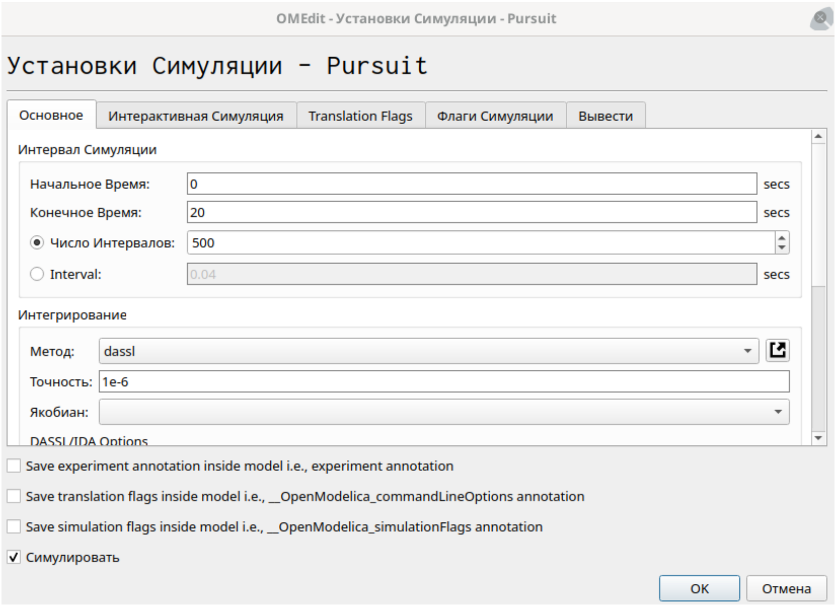

Рис. 0.9: Настройка симуляции, ч.1

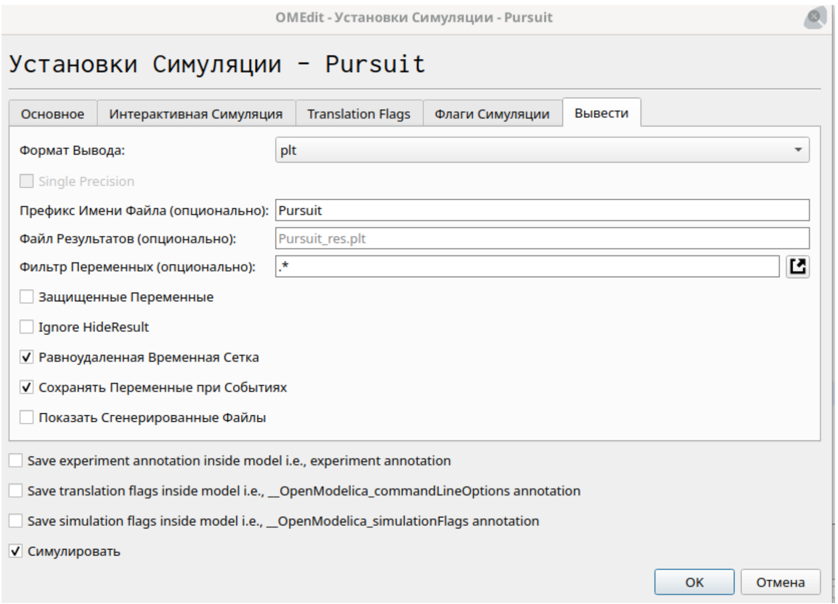

Рис. 0.10: Настройка симуляции, ч.2

Выбираем график типа Parametric Plot и указываем необходимые нам оси, линии графика, настраиваем легенду (рис. @fig:012).

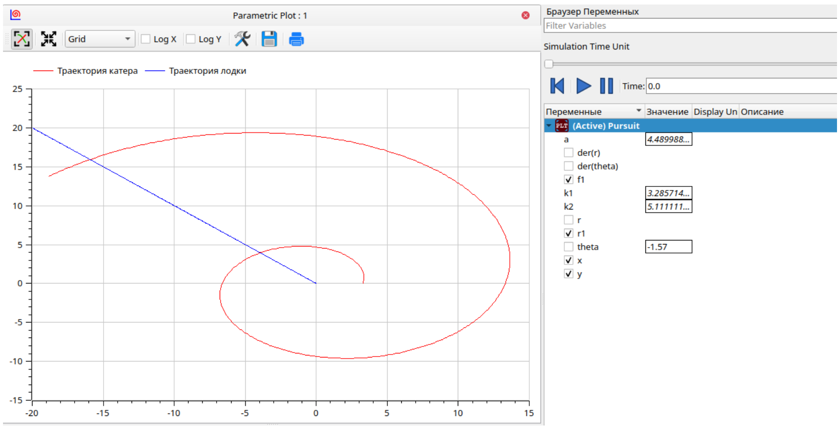

Рис. 0.11: Настройка внешнего вида графика

Рассмотрев первый график (рис. @fig:013), получаем координаты точки пересечения, равные (-3.9, 3.9).

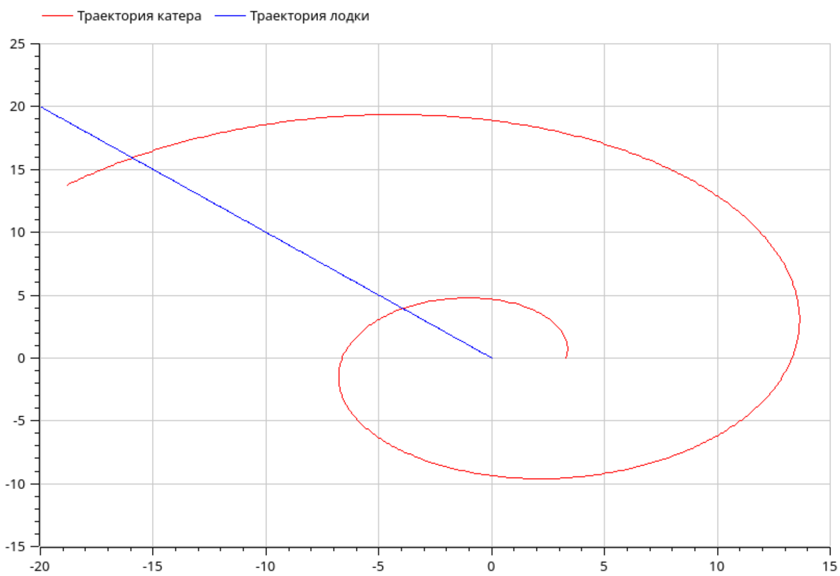

Рис. 0.12: График OpenModelica, первый случай

Проанализировав второй график (рис. @fig:014), видим, что координаты точки пересечение равны (-12.3, 12.3).

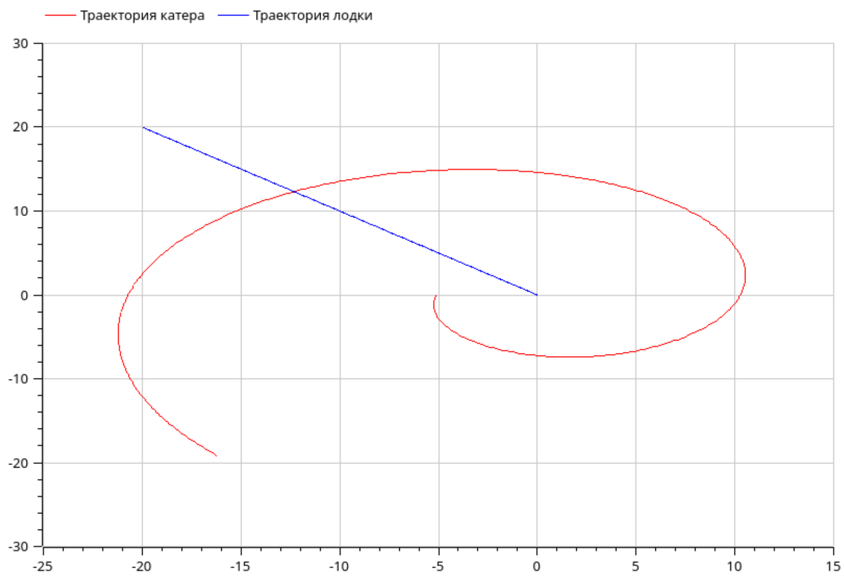

Рис. 0.13: График OpenModelica, второй случай

Сравнение языков

Julia является более интуитивно понятным языком и позволяет производить различные вычисления, не ограничиваясь рамками работы с визуализацией уравнений.

OpenModelica является более узконаправленным инструментом, ориентирующимся на создание сложных моделей с множеством уравнений в основе. Из-за этого работа с ним кажется труднее.

На результаты работы различия в подходах этих языков практически не повлияли, координаты точек пересечения в разных реализациях совпадают с точностью до одного знака после запятой.

Выводы

В ходе работы были получены навыки построения математических моделей для решения задачи о погоне и визуализации их с помощью языков Julia и OpenModelica, а также закреплены знания, связанные с решением дифференциальных уравнений. Результатом работы стали графики, наглядно демонстрирующие решение задачи о погоне.

Список литературы

1. Теоретические материалы к лабораторной работе “Задача о погоне”: https://esystem.rudn.ru/mod/reso