**Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Специальное машиностроение» КАФЕДРА «Подводные аппараты и роботы»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

***К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:***

Разработка системы управления траекторным движением ПА в плоскости горизонта

Студент группы СМ11-81Б Бочкарев М.А.

(Подпись, дата)

Руководитель курсового проекта Ромашко А.С.

(Подпись, дата)

*Москва, 2025 г.*

**Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой СМ11

Вельтищев В.В.

«\_\_\_\_» \_\_\_\_\_\_\_\_20\_\_г.

**З А Д А Н И Е**

**на выполнение курсового проекта**

по дисциплине «Проектирование подводных робототехнических систем» Тема курсового проекта:

Разработка системы управления траекторным движением ПА в плоскости горизонта.

Студент группы СМ11-81Б: Бочкарев М.А.

***Задание***

Необходимо синтезировать контура управления ПА и систему управления движением по траектории, соединив ее с программой построения траекторий.

***Оформление курсового проекта***

Расчетно-пояснительная записка на 44-х листах формата А4.

Дата выдачи задания « \_ » 20 г.

**Руководитель курсового проекта**  Ромашка А.С.

(Подпись, дата)

**Студент**  Бочкарев М.А.

(Подпись, дата)

Оглавление

[ВВЕДЕНИЕ 5](#_Toc210932331)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 6](#_Toc210932332)

[ОСНОВНАЯ ЧАСТЬ 7](#_Toc210932333)

[1. Синтез контуров курса и марша 9](#_Toc210932334)

[1.1 Синтез контура курса 9](#_Toc210932335)

**[1.1.1 Синтез динамической модели движения по углу курса ПА](#_Toc210932336)** [9](#_Toc210932336)

**[1.1.2 Движительно-рулевой комплекс подводного аппарата в контуре курса](#_Toc210932337)** [13](#_Toc210932337)

**[1.1.3 Построение уравнения и реализация модуля управления контура курса](#_Toc210932338)** [16](#_Toc210932338)

[1.2 Синтез контура управления маршевым движением 21](#_Toc210932339)

**[1.2.1 Синтез динамической модели маршевого движения ПА](#_Toc210932340)** [21](#_Toc210932340)

**[1.2.2 Движительно-рулевой комплекс подводного аппарата в контуре марша](#_Toc210932341)** [24](#_Toc210932341)

**[1.2.3 Построение уравнения и реализация модуля управления контура марша](#_Toc210932342)** [26](#_Toc210932342)

[2. Разработка метода движения ПА по траектории 29](#_Toc210932343)

[2.1 Выбор метода вывода аппарата в точку траектории 29](#_Toc210932344)

[2.2 Описание выбранного метода вывода ПА в точку 30](#_Toc210932345)

[С учетом формулы (2.1) мы получаем программную реализации модуля вывода ПА в точку. Модель данного модуля представлена на рисунке 2.1. 32](#_Toc210932346)

[2.3 Объединение контуров марша и курса с методом вывода ПА в точку для реализации движения по траектории 33](#_Toc210932347)

[2.3.1 Работа контура курса и модуля вывода ПА в точку для управления углом курса во время движения. 33](#_Toc210932348)

[2.3.2 Работа контура курса и модуля вывода ПА в точку для управления маршевым движением ПА 34](#_Toc210932349)

[3 Разработка интерфейса программного комплекса 36](#_Toc210932350)

[СПИСОК ИСТОЧНИКОВ 41](#_Toc210932351)

[ПРИЛОЖЕНИЕ А 42](#_Toc210932352)

# **ВВЕДЕНИЕ**

В данном курсовом проекте мы рассмотрим АНПА, выполняющий задачу съемку поверхности дна для выявления особенностей рельефа дна и формирования информации, необходимой для будущих технических работ, связанных с донной поверхностью акваторий. Особое внимание уделим параметрам контуров управления курсом и маршем для создания благоприятных динамических условий, необходимых для проведения подводной съемки с использованием высокочастотных гидролокаторов бокового обзора, а также построению траектории, позволяющей покрыть всю заданную для исследования акваторию.

Цель работы – разработать систему управления траекторным движением ПА в плоскости горизонта для прохождения заданной траектории с проведением съемки дна с использованием гидролокатора бокового обзора.

Для достижения поставленной цели будут выполнены несколько основных задач:

1. Синтез система управления движения по углу курса, состоящая из нескольких основных модулей.
2. Синтез система управления маршевым движением ПА.
3. Выбор метода вывода ПА в точку и разработан модуль вывода ПА в точку.
4. Объединение компонентов в единую систему управления траекторным движением ПА
5. Создание интерфейса для управления параметрами траектории и графического отображения результатов работы системы.

# **ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Разработка системы управления движением подводного аппарата в плоскости горизонта по траектории вида галсы, включающая в себя синтез контуров управления углом курса и маршевым движением, а также методов выхода в точку для следования заданной траектории.

Заданы следующие характеристики подводного аппарата:

1. Масса m – 90 кг
2. Присоединенная масса
3. Первый гидродинамический коэффициент
4. Второй гидродинамический коэффициент
5. Постоянная времени движителей c
6. Коэффициент усиления движителей
7. Максимальное управляющее напряжение
8. Момент инерции
9. Присоединенный момент инерции
10. Первый гидродинамический коэффициент
11. Второй гидродинамический коэффициент
12. Плечо пары движителей b – 0.3 м
13. Скорость аппарата на участках съемки V = 0.7

# ОСНОВНАЯ ЧАСТЬ

В соответствие с целью курсовой работы будут разработаны система управления траекторного движения ПА и модуль отображения траектории, входящая в состав системы управления аппарата для подводной съемки, представленной на рисунке 1.

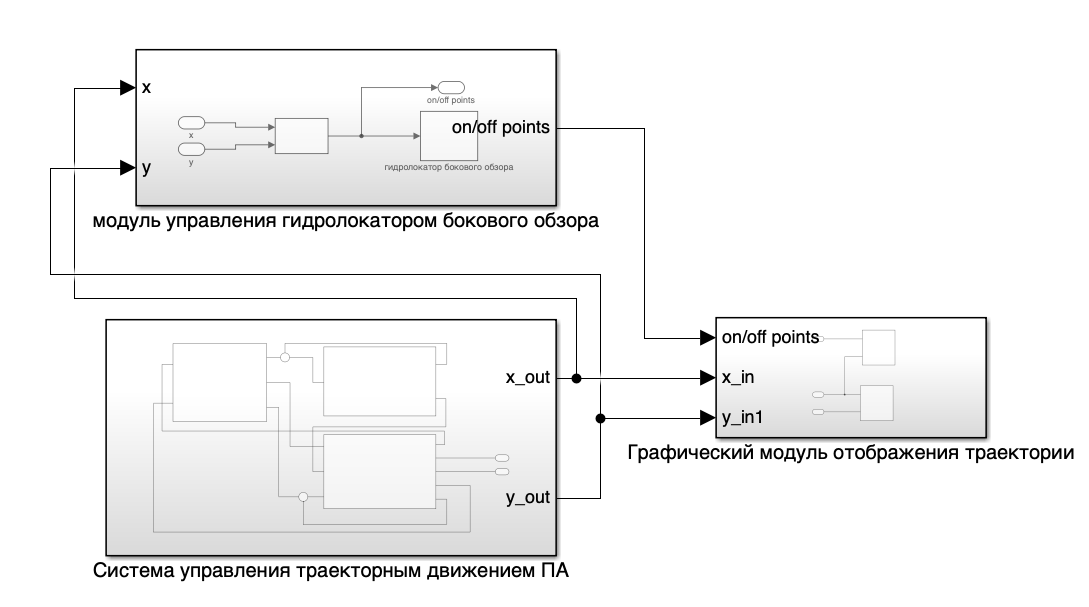


Рисунок 1 – система управления аппарата для подводной съемки.

В состав системы управления траекторным движением ПА, представленной на рисунке 2, входят контуры марша и курса, а также модуль вывода ПА в точку.

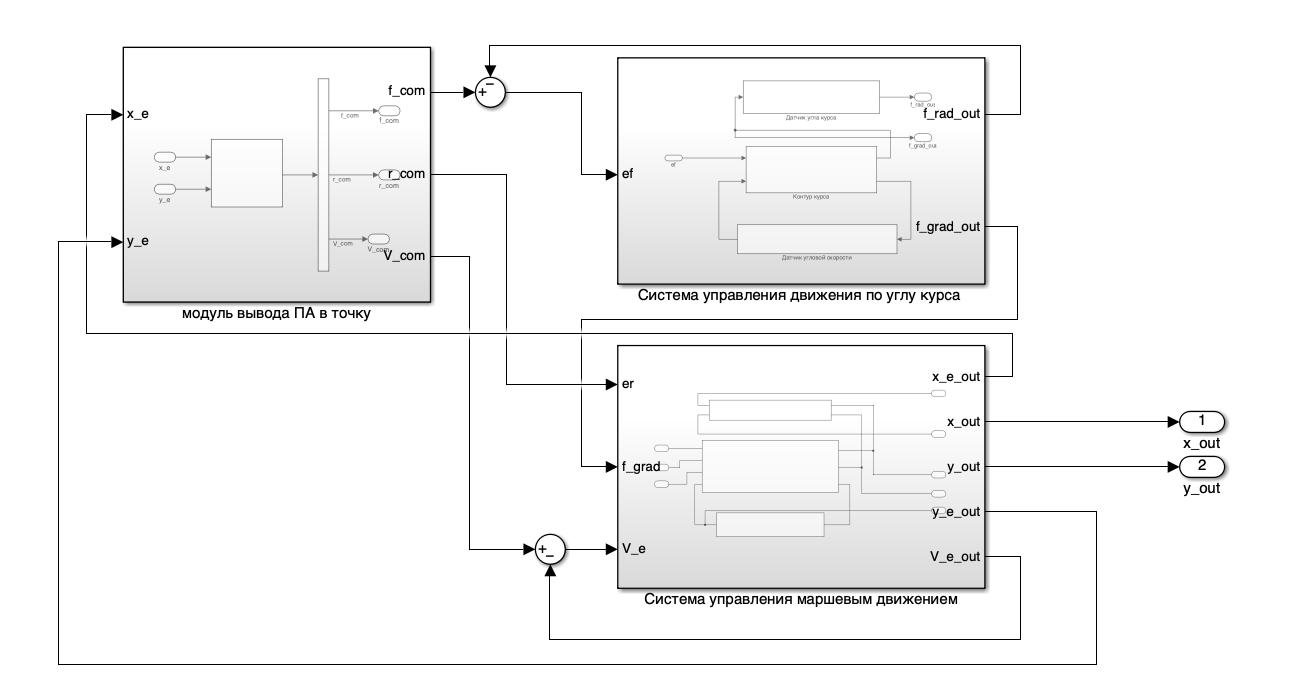


Рисунок 2 – система управления траекторным движением ПА.

Графический модуль содержит в себе интерфейс с возможностью задания параметров сектора съемки и течения в секторе и окно для отображения полученной траектории движения ПА. Графический модуль представлен на рисунке 3.

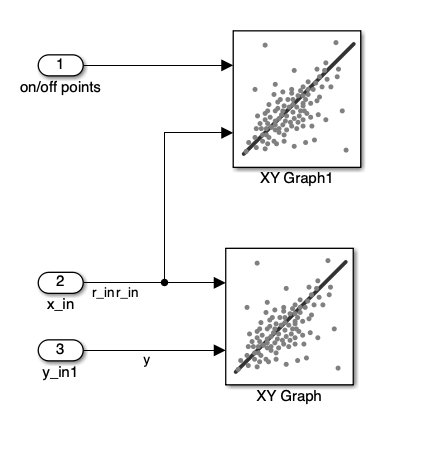


Рисунок 3 – графический модуль отображения данных.

1. **Синтез контуров курса и марша**

## **1.1 Синтез контура курса**

Синтез контура управления курсом можно разделить на три ключевых этапа – синтез динамической модели аппарата, синтез модели движительно – рулевого комплекса и синтез модели модуля управления. Схема контура курса представлена на рисунке 1.1.

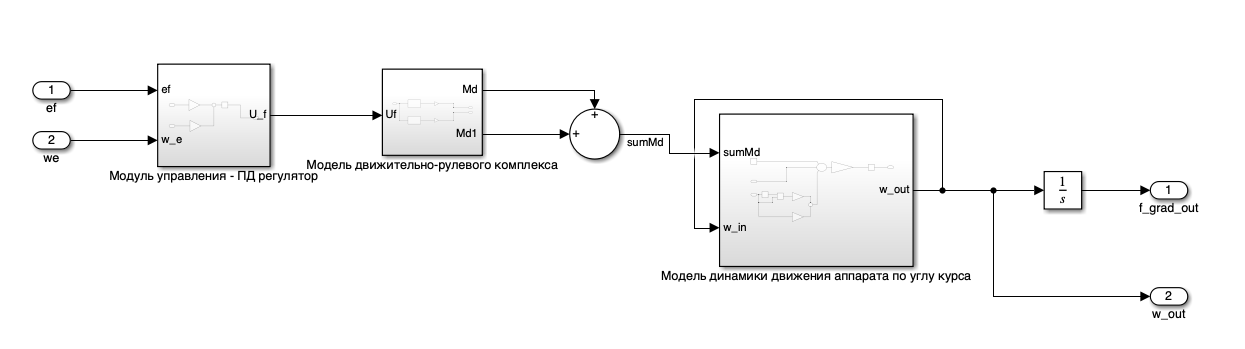


Рисунок 1.1 – контур курса

### **1.1.1 Синтез динамической модели движения по углу курса ПА**

**В основу динамической модели движения по углу курса ПА ложится формула движения аппарата по углу курса – поворот аппарата вокруг оси Y, проходящей через центр масс аппарата:**

Где:

– момент инерции аппарата относительно оси у, кг \* м^2

– присоединенный момент инерции относительно оси у, **кг \* м^2**

– угловая скорость аппарата относительно оси у, **рад/с**

– Суммарный момент сил, действующих на ПА относительно оси у, **Н \* м**

– момент перекрестных связей относительно оси у, **Н \* м**

**Согласно поставленному техническому заданию курсового проекта, аппарат движется в плоскости горизонта, вследствие чего влиянием углов крена и дифферента примем равным 0. Вследствие этого мы для проведения синтеза пренебрегаем составляющей перекрестных связей в формуле (1.1)**

**Таким образом, формула (1.1) вращательного движения ПА вокруг оси Y, проходящей через центр масс, принимает вид:**

**Формула суммарного момента, сходящего в состав формулы (1.2)\_имеет вид:**

Где:

– **момент от силы тяги движителей**, **Н \* м**

– **момент гидродинамических сил сопротивления**, **Н \* м**

– **момент от силы Архимеда**, **Н \* м**

– **момент от возмущающего воздействия**, **Н \* м**

Момент от силы Архимеда, входящий в состав формулы (1.3), численно равен произведению силы Архимеда на метацентрическую высоту и направляющие косинусы, вследствие чего, так как мы принимаем допущение, что центр масс и центр объемов нашего подводного аппарата совпадают, моментом от силы Архимеда мы пренебрегаем в будущих расчетах.

Описание момента гидродинамичсеких сил сопротивления, входящего в состав формулы (1.3), мы произведем на основе циркуляционно-отрывной теории обтекания[1]. Согласно данной теории, мы выделим две основные составляющие:

**Циркуляционная (подъёмная) часть.** Появляется из-за циркуляции вокруг корпуса/оперения (по Кутта–Жуковскому): локальные боковые силы  создают момент . В АУВ это даёт полезный момент на рулях/килях (у нас он попадает в , т.е. «управляющий»), а также паразитный момент при дрейфе/угле сноса.

**Отрывная (вихревая) часть = сопротивление поперечному обдуву.** При повороте корпуса с угловой скоростью  каждая «полоса» корпуса на расстоянии  видит поперечную скорость . Отрыв порождает боковую силу , а момент:

Где:

N – **момент гидродинамического сопротивления, Н\*м**

– **коэффициент нелинейного демпфирования по рысканию**, **Н·м·с²**

– **угловая скорость, рад/с**

– **плечо силы, м**

– **плотность жидкости**, **кг/м³**

– **коэффициент поперечного сопротивления**

– **эффективная «толщина пластины»/поправка на форму**

– **полуширина корпуса (от диаметральной плоскости до борта) на сечении**

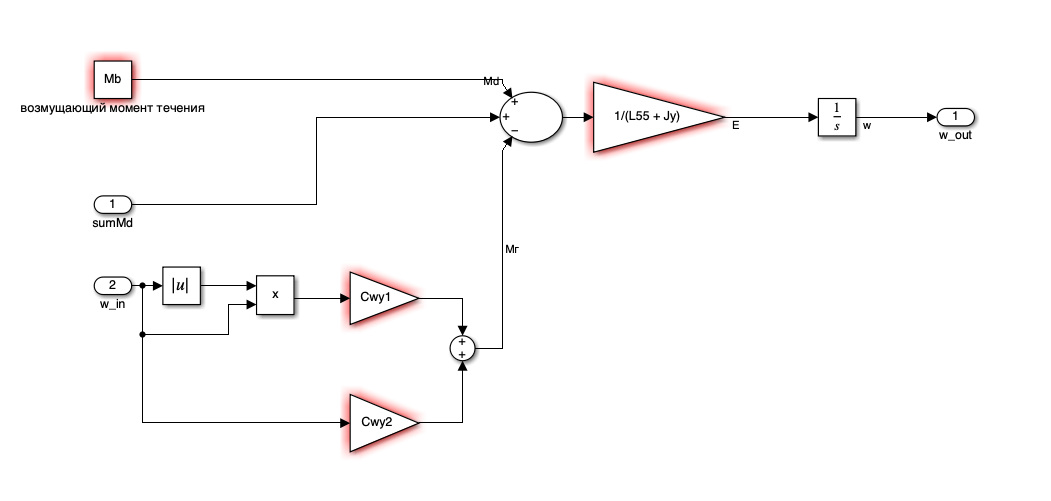
Момент , входящий в состав формулы (1.4), выступает в роли квадратичного демпфирующего момента, который будет использован далее при расчете момента гидродинамических сил. При малых  и доминировании «вязкостной» части получается ещё и **линейный** демпфер , который нужно прибавить к квадратичному демпферу в формуле (1.4) для получения полного момента гидродинамического сопротивления. В конечном итоге мы получаем формулу момента сопротивления гидродинамических сил:

Обозначив полученные в формуле (1.5) коэффициенты как Cωy1 и Cωy2 – приведенные гидродинамические коэффициенты, мы получим конечную формулу для момента гидродинамических сил сопротивления **:**

При сверх малых скоростях коэффициенты Cωy1 и Cωy2 из формулы (1.6) представляют собою функции от угловой скорости, однако в нашей задаче угловые скорости не являются сверхмалыми, поэтому приведенные гидродинамические коэффициенты мы будем считать постоянными.

С учетом формулы (1.2), формулы (1.3) и формулы (1.6), мы получаем уравнение динамики вращательного движения аппарата вокруг оси Y, проходящей через центр масс ПА:

**Уравнение (1.7) будет реализована в нашей системе управления траекторным движением ПА, представленной на рисунке 1.2, в плоскости горизонта, как модели динамики вращательного движения аппарата.**

****

**Рисунок 1.2 – динамическая модель движения аппарата по углу курса.**

### **1.1.2 Движительно-рулевой комплекс подводного аппарата в контуре курса**

**Для получения центральной формулы движительно-рулевого комплекса нам необходимо провести линеаризацию нашей динамической модели вокруг некоторой . Для данного процесса найдем частную производную формулы (1.7) динамики движения:**

Используя формулу (1.8), построим линеаризованную динамическую модель, посчитаем ее передаточную функцию:

***Где:***

На аппарате имеются два винта на расстоянии  друг от друга (по ширине корпуса), так что плечо каждого до ЦМ — . Если тяга винта линейно зависит от управляющего сигнала (напряжения/ШИМ), то

Где:

Чистый момент вокруг :

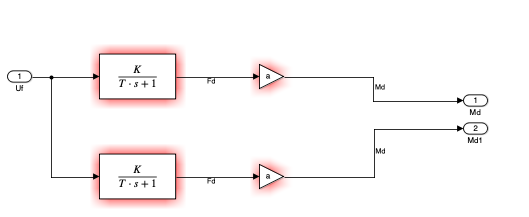
Для чистого момента по рысканью задаём **дифференциальное управление**: . Тогда с использованием формул (1.10) и (1.11) мы получаем формулу для момента:

Где:

. – коэффициент усиления движительного комплекса

Используя формулы (1.9) и (1.12) мы получаем уравнение модели движительно-рулевого комплекса аппарата

Таким образом мы получаем уравнение модели движительно-рулевого комплекса (1.13), представленную на рисунке 1.3, в которой в качестве входного управляющего сигнала выступает напряжение , которое подается на комплекс с модуля управления.

****

**Рисунок 1.3 – модель движительно-рулевого комплекса**

### **1.1.3 Построение уравнения и реализация модуля управления контура курса**

**В качестве модуля управления было принято решение выбрать PD регулятор, представленный на рисунке 1.4, так как это минимально-достаточный стабилизатор, который** даёт нужную «жёсткость» по углу и демпфирование по скорости, сводя полную (трёхпорядковую) систему к желаемой системе второго порядка с быстрым полюсом привода.

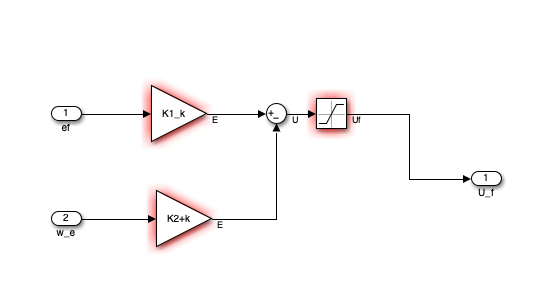


Рисунок 1.4 – PD регулятор

С учетом выбора PD регулятор мы получаем уравнение для модуля управления:

*Где:*

***-***

***ω - величина угловой скорости аппарата, рад/с***

**Коэффициенты К1 и К2 в уравнении (1.14)– коэффициенты PD регулятора. Для их расчета запишем передаточную функцию контура скорости Wc(p) с применением формул (1.13) и (1.7):**

***Где:***

***После приведения передаточной функции (1.15) к каноническому виду передаточная функция замкнутого контура угловой скорости ПА примет вид колебательного звена:***

***Где:***

***После преобразования выражения для коэффициента демпфирования мы получаем следующие формулы для***  *и* ***:***

*С точки зрения оптимальных характеристик колебательного звена мы примем*  ***и***

***Такие параметры обеспечат перерегулирования меньше 5% и минимальное время переходного процесса***

***После подстановки данных значений вместе с***  *(наихудший с точки зрения устойчивости случай) и найденными прежде параметрами мы получаем из формул (1.17) и (1.18):*

*Далее запишем передаточную функцию разомкнутого контура углового положения:*

***Где:***

***Для приближенных расчетов примем равной 1, тогда передаточная функция (1.19) примет вид:***

***Согласно исходному перерегулированию равному 5-7 процентов, опираясь на работу Бесекерсокго, мы примем коэффициент М = 1.03[2]. Исходя из данного выбора, найдем добротность нашей системы при заданных параметрах по функции (1.21).***

***Подставляем полученное значение в формулу (1.20) и получаем:***

***Коэффициенты для модуля управления, представляющего собой PD регулятор, были найдены.***

Таким образом, мы получаем систему уравнений, состоящую из формул (1.7), (1.13) и (1.14), на основании которой будет смоделирован контур курса для управления траекторным движением ПА:

1.1.4 – Проверка переходного процесса контура управления углом курса

Чтобы убедиться в правильности полученных нами уравнений (1.7), (1.13) и (1.14), ставших основой модели движения по углу курса, мы построим переходной процесс контура – реакцию на единичное воздействие нашей системы, и считаем два ключевых параметра для нас – время установившегося процесса и перерегулирование.

Для построения будем использовать написанный на языке Python комплекс программ для нашей системы.

После запуска моделирования получаем переходной процесс, представленный на рисунке 1.5

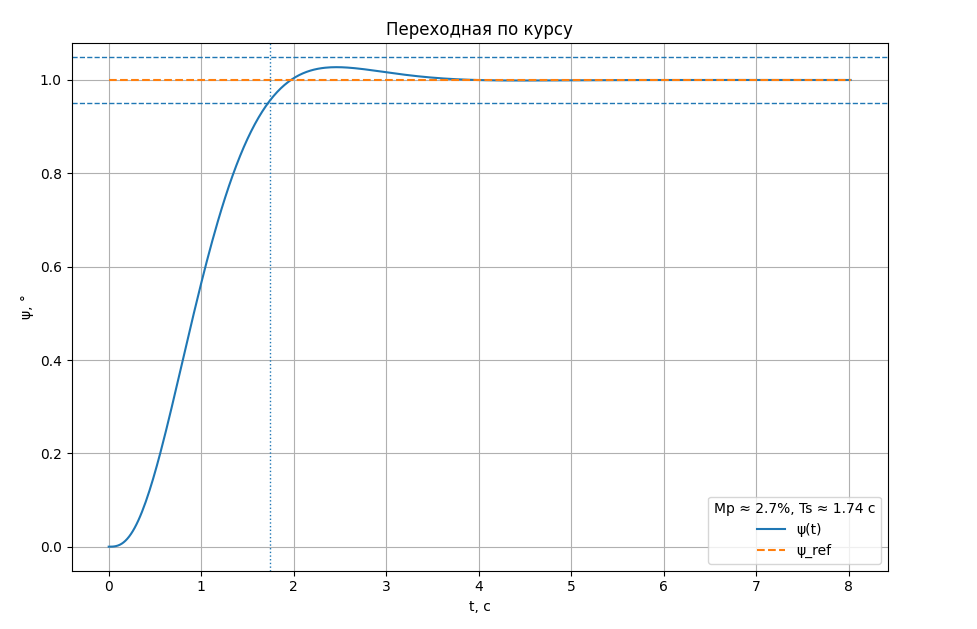


Рисунок 1.5 – переходной процесс контура курса

Полученные время установившегося процесса и перерегулирование соответствуют требованиям нашей системы.

## **1.2 Синтез контура управления маршевым движением**

синтез контура движения по маршу, представленную на рисунке 1.6, мы также разделим на три этапа – синтез динамической модели, синтез модели движительно-рулевого комплекса и синтез модуля управления контуром.

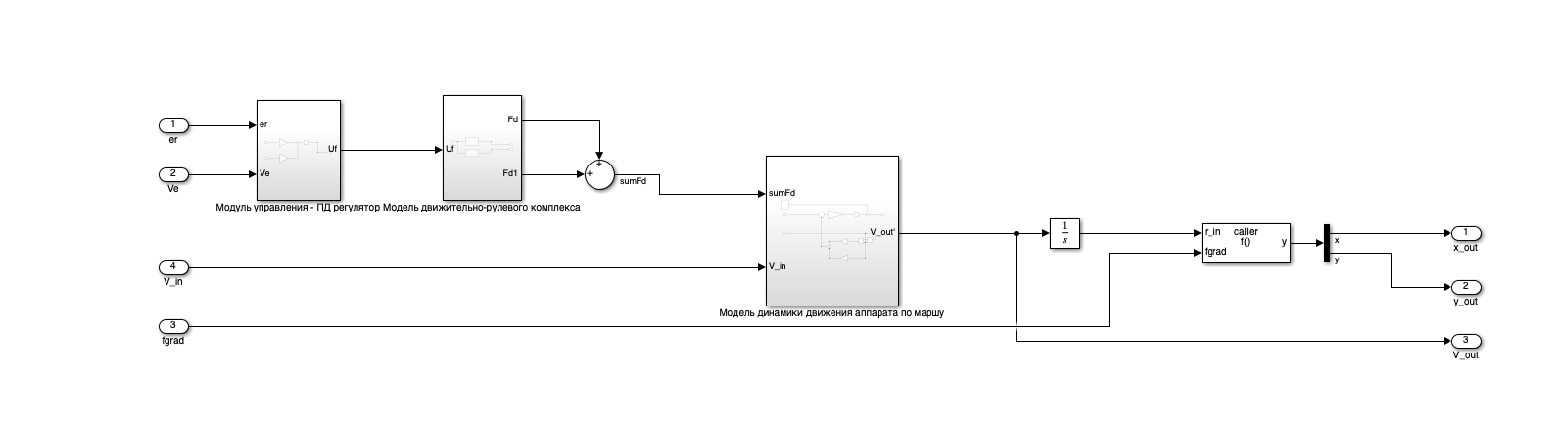


Рисунок 1.6 – контур управления маршевым движением

### **1.2.1 Синтез динамической модели маршевого движения ПА**

**В основу динамической модели маршевого движения ПА ложится формула поступательного движения аппарата вдоль оси X, проходящей через центр масс аппарата:**

**Где:**

**По аналогии с контуром курса, составляющую в формуле (1.22)– функцию перекрестных связей мы отбросим**

**Таким образом, формула (1.22) поступательного движения ПА вдоль оси X, проходящей через центр масс, принимает вид:**

**Рассмотрим величину суммарной силы по оси Х в формуле (1.23).Уравнение данной силы имеет вид:**

***Где:***

***, Н***

***, Н***

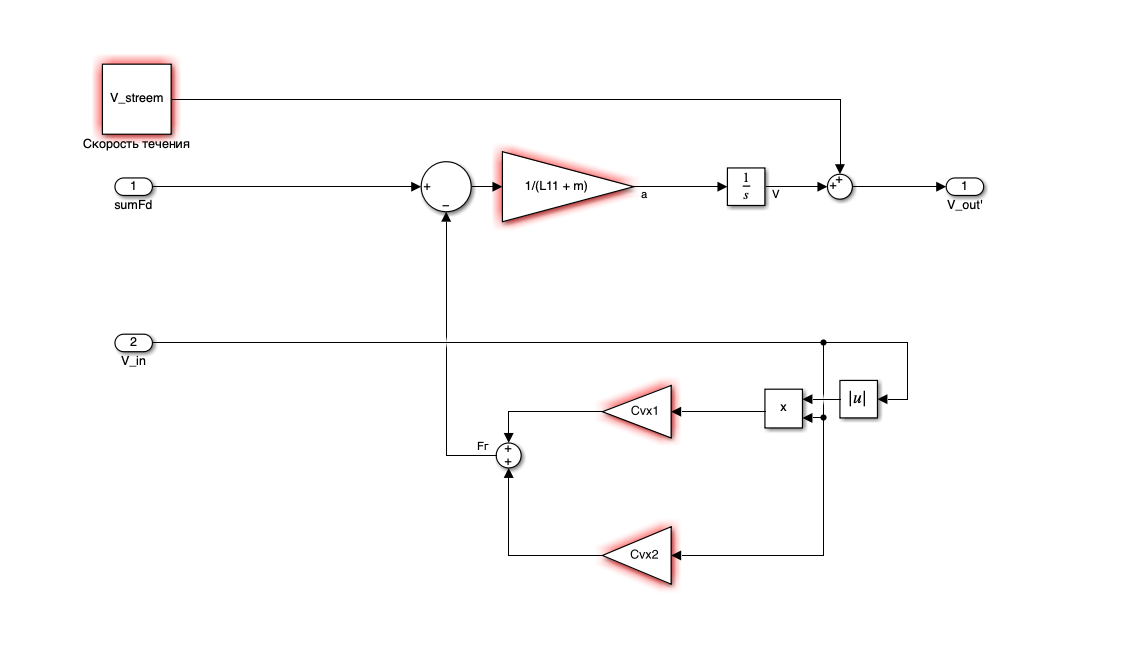
Величиной силы Архимеда в формуле (1.24) мы можем пренебречь, так как движемся строго в плоскости горизонта с нулевыми углами крена и дифферента**.**

**Проекцию силы гравитации в формуле (1.24) мы принимаем за 0, так как одним из важных допущений при построении динамической модели контура марша является равенство углов крена и дифферента нулю, что позволяет принять направление воздействия силы гравитации соосным с нормалью подводного аппарата, а из этого следует равенство проекции этой силы на ось X нулю.**

Гидродинамические силы сопротивления в формуле (1.24) вязкостной природы по аналогии с контуром курса мы распишем, опираясь на циркуляционно-отрывную теорию обтекания. Таким образом, для данных сил мы получаем следующее уравнение:

Исходя из формул (1.24) и (1.25), формула динамической модели ПА при движении по маршу принимает вид:

**Уравнение (1.26) будет реализовано в нашей системе управления траекторным движением ПА, представленной на рисунке 1.7, в плоскости горизонта, как модели динамики поступательного движения аппарата. В качестве входных сигналов в нашей модели выступают сила тяги движителей и возмущающая сила, поэтому для дальнейшего анализа контура марша мы переходим к моделированию движительно-рулевого комплекса.**

******

***Рисунок 1.7 – динамическая модель маршевого движения ПА***

### **1.2.2 Движительно-рулевой комплекс подводного аппарата в контуре марша**

**Для получения центрального уравнения движительно-рулевого комплекса нам необходимо провести линеаризацию нашей динамической модели вокруг некоторой . Для данного процесса нам необходимо найти частную производную формулы (1.26) динамики движения. По аналогии с расчетом для контура курса мы получаем следующее значение частной производной:**

Построим линеаризованную систему управления, посчитаем ее передаточную функцию:

Тяга правого и левого винтов на нашем аппарате была посчитана нами еще при анализе контура курса и составляет:

Так как при маршевом движении оба винта работают в одном направлении с равной тягой, мы получаем следующее выражении для общей тяги движителей исход из формулы (1.29):

Где:

 – коэффициент усиления движительного комплекса.

Исходя из передаточной функции (1.28) и формулы (1.30) для силы тяги мы получаем следующую зависимость:

Аналогично контуру курса, мы получили модель движительно-рулевого комплекса на основе уравнения (1.31) с управляющим напряжением , представленную на рисунке 1.8.

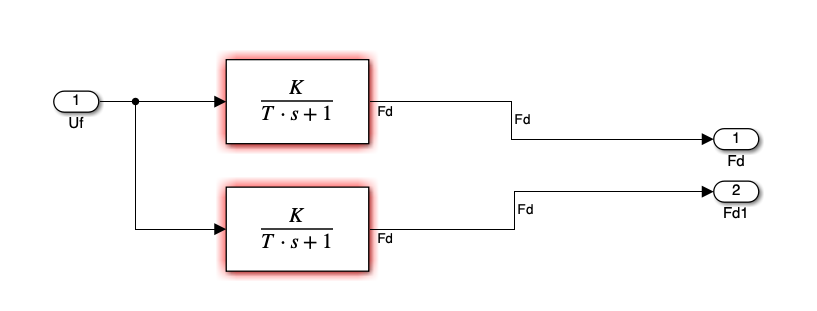
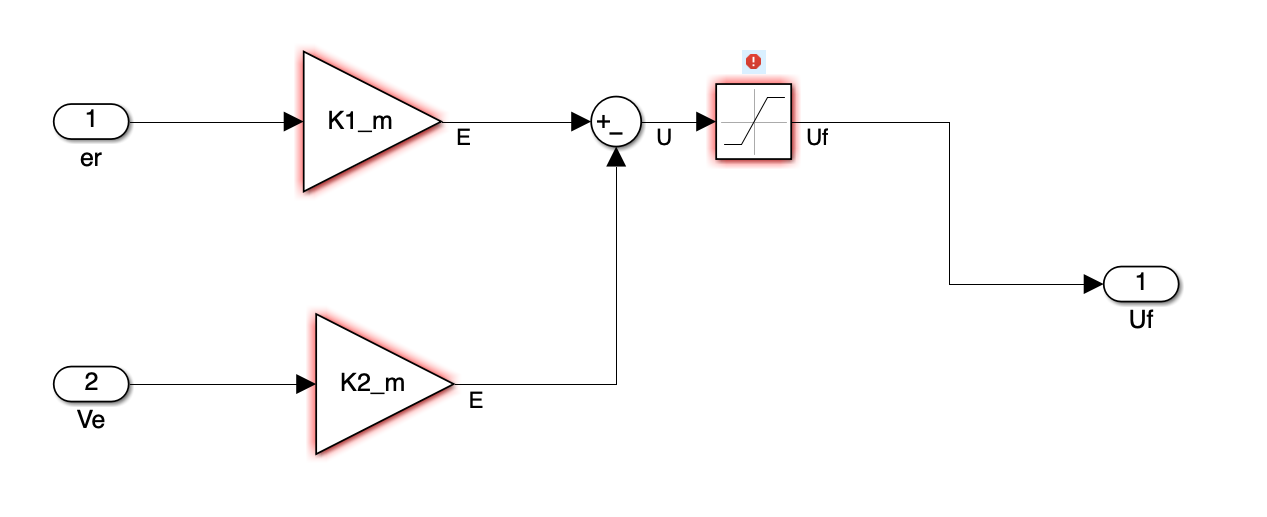


Рисунок 1.8 – модель движительно-рулевого комплекса в контуре марша.

### **1.2.3 Построение уравнения и реализация модуля управления контура марша**

**Так же, как и в контуре курса, мы выберем PD регулятор, представленный на рисунке 1.9, в качестве элемента формирования управляющего сигнала.**

****

**Рисунок 1.9 – модель PD регулятора в контуре маршевого движения.**

**Уравнение регулятора имеет вид**:

Проведя аналогичный синтез с выбором ряда параметров для переходного процесса, мы получаем коэффициенты в формуле (1.32).

Таким образом, мы получаем систему уравнений из уравнений (1.26), (1.31) и (1.32), на основании которой будет смоделирован контур марша для управления траекторным движением ПА:

1.2.4 – Проверка переходного процесса контура маршевого движения

Чтобы убедиться в правильности полученных нами уравнений (1.26), (1.31) и (1.32), по аналогии с контуром курса при помощи программного комплекса построим переходной процесс контура маршевого движения

После запуска моделирования получаем переходной процесс, представленный на рисунке 1.10

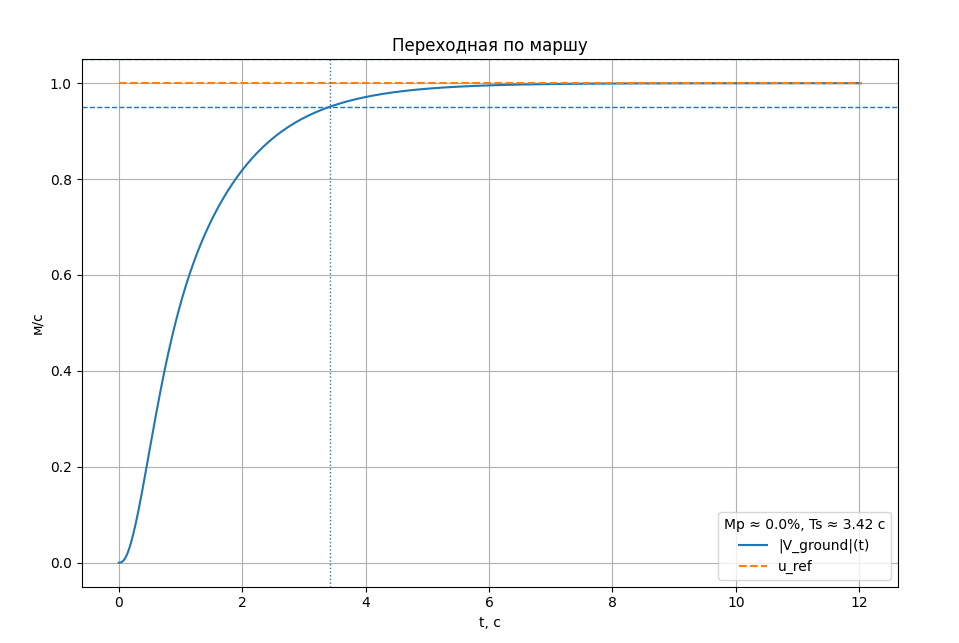


Рисунок 1.10 – переходной процесс контура маршевого движения

Полученные время установившегося процесса и перерегулирование также соответствуют требованиям нашей системы.

На основании двух проведенных проверок можно сделать заключение, что контуры управления в нашей системе соответствуют заданному техническому заданию.

1. **Разработка метода движения ПА по траектории**

Для разработки метода движения ПА по траектории необходимо прежде всего определиться с ключевыми характеристиками желаемой траектории ПА, на основании которых будет произведен будущий выбор метода движения. В качестве выбранной траектории рассматривается траектория формы ГАЛС, на некоторых участках которой мы будем производить съемку с использованием гидролокаторов бокового обзора, а значит необходимые для нас требования – плавное изменение траектории на поворотах и поддержание линейной траектории на прямых участках без рывков. Так как наша траектория задается набором точек, то первостепенная задача – выбрать метод вывода аппарата в точку траектории.

* 1. **Выбор метода вывода аппарата в точку траектории**

Приведем несколько основных методов вывода аппарата в точку траектории:

* 1. метод прямого наведения: заключается в управлении углом курса подводного аппарата на каждом такте работы системы управления таким образом, чтобы подводный аппарат все время приближался к заданной точке.
  2. метод наведения по линии визирования: заключается в управлении углом курса подводного аппарата на каждом такте работы системы управления таким образом, чтобы подводный аппарат все время двигался по прямой, соединяющей предыдущую и заданную точку траектории.
  3. метод минимизации бокового отклонения: предполагает автоматическую коррекцию линейных перемещений с учётом незапланированных смещений аппарата от заданного положения.
  4. метод наведения в «субточку»: продольная ось аппарата всегда направлена не в точку цели, а в точку на прямой, соединяющей две точки.

Методы прямого наведения и наведения по линии визирования предполагают жесткое следование по прямолинейной траектории к следующей точке траектории, что при возникновении отклонений нашего аппарата от линии траектории приведет к резким изменениям курса аппарата для корректировки движения. В целях проведения качественной съемки резкие изменения курса нам недопустимы. Метод минимизации бокового отклонения отлично подходит для прямолинейной траектории аппарата и был бы выбран нами, но наша траектория в форме ГАЛС, а значит нам нужен метод, эффективно работающий на поворотах.

Самым подходящим для поставленной задачи является метод наведения в «субточку», так как он позволяет совершить плавный разворот без резких изменений угла курса и позволяет поддерживать линейную траекторию для качественной съемки. Стоит учесть, что при воздействии внешних воздействий данный метод обеспечивает плавный возврат ПА на линию траектории, что помогает избежать потери качества съемки вследствие внешних возмущений. Более подробно о преимуществах и особенностях данного метода будет изложено в следующем пункте.

* 1. Описание выбранного метода вывода ПА в точку

Метод работает в глобальной системе координат . Положение аппарата , курс . Цель . Задача, которую реализует данный метод вывода ПА в точку - войти в цель с заданным углом захода  в глобальных осях.

Суть метода заключается в ведении аппарата не в саму цель, а в виртуальную «субточку» , расположенную за целью по направлению требуемого курса захода; по мере движения субточка плавно сближается с целевой точкой вплоть до совпадения, что обеспечивает достижение ПА заданной точки траектории. Важным параметром такого метода является базовая длина упреждения  – расстояние, на котором аппарат определяет позицию субточки. Более подробно данный принцип вывода ПА в точку траектории показан на рисунке 1.

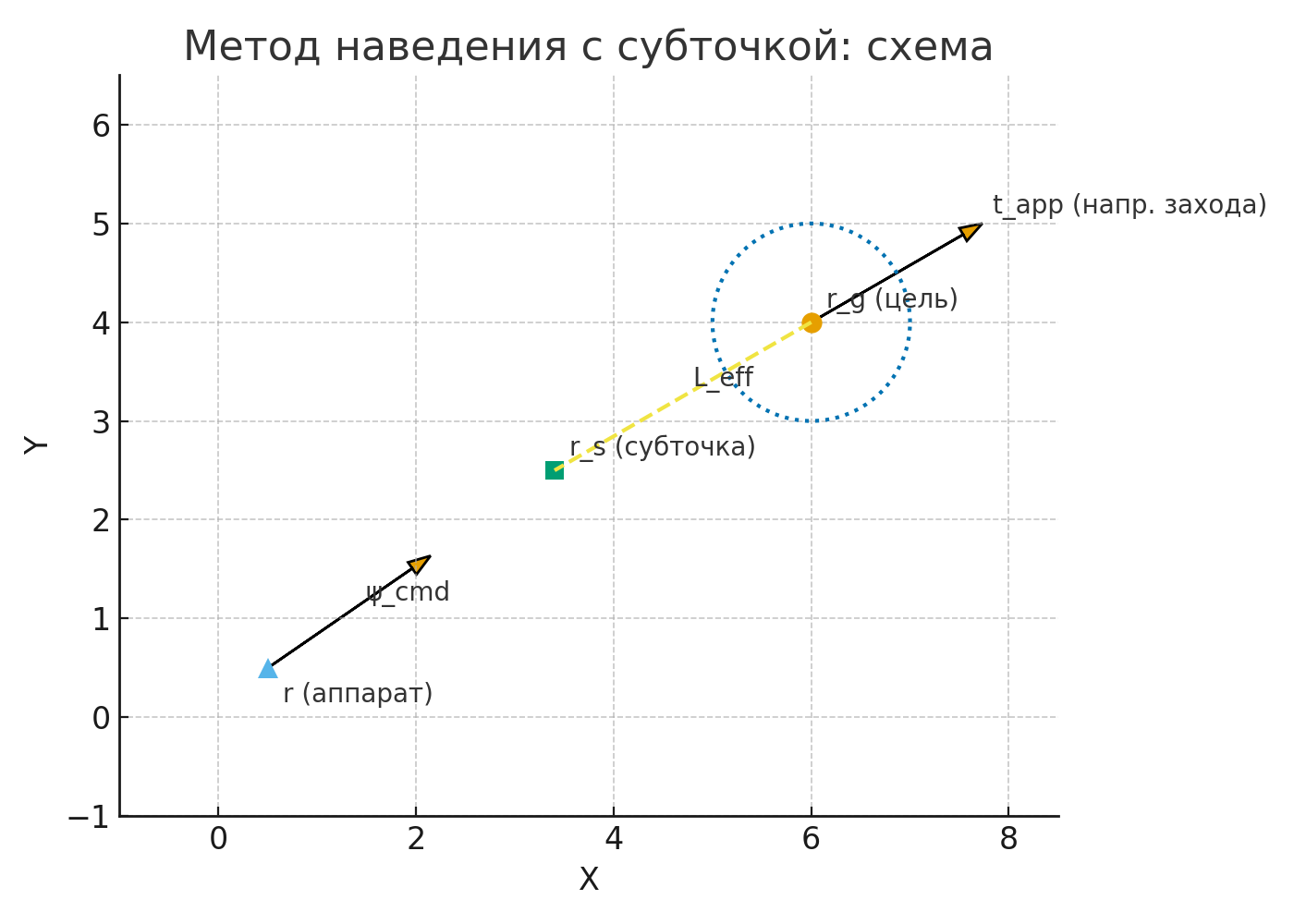


Рисунок 2.1 – схема метода наведения в субточку.

Для применения метода на практике есть ряд требований и ограничений, без соблюдения которых вывести аппарат в требуемую точку траектории будет невозможно, или же вывод будет сопровождаться резкими скачками управляющего сигнала, которые мы должны избежать. Необходимо соблюсти следующие правила:

* 1. Необходимо выбрать  соизмеримым с минимально достижимым радиусом поворота на рабочей скорости; на практике стоит брать диапазон от одного до трёх таких радиусов, чтобы аппарат успевал развернуться без петли.
  2. Для корректного входа в зону достижения точки необходимо брать , чтобы субточка не оказалась за аппаратом до входа в область достижения точки траектории.
  3. При боковом сносе следует адаптировать  по модулю скорости и поперечному отклонению , сохраняя формулу  .  Поперечная ошибка относительно оси захода:

## С учетом формулы (2.1) мы получаем программную реализации модуля вывода ПА в точку. Модель данного модуля представлена на рисунке 2.1.

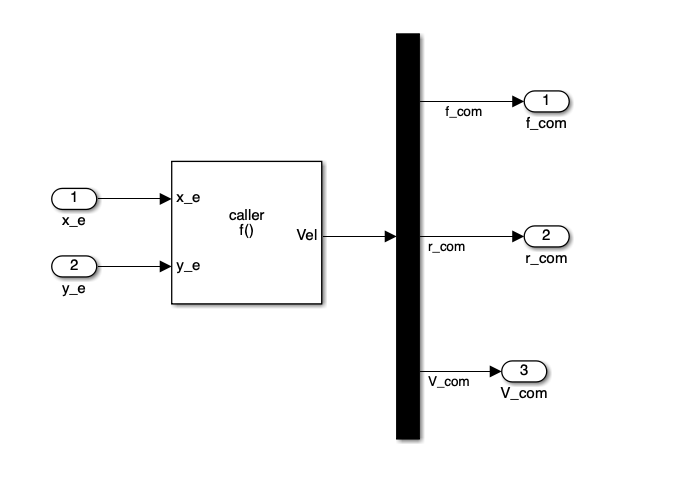


Рисунок 2.1 – модель модуля вывода ПА в точку

## 2.3 Объединение контуров марша и курса с методом вывода ПА в точку для реализации движения по траектории

Далее приступим к формированию всей системы управления траекторным движением ПА, опираясь на полученные контуры управления углом курса и маршевым движением ПА, а также на модуль вывода аппарата в точку траектории, основанной на методе движения в «субточку». При формировании сигналов управления учтен процесс съемки с использованием гидролокаторов бокового обзора, для которого на прямых участках происходит замедление аппарата до 0.7 метров в секунду. Также важным моментом является замедление аппарата на поворотах для плавного прохождения изменения курса на 90 градусов. Расчет замедления производится исходя из приближения аппарата к субточке, находящейся в точке окончания прямого отрезка ГАЛС.

Система управления работает в несколько этапов с шагом дискретизации, являющимся открытым параметром, задающимся оператором.

### 2.3.1 Работа контура курса и модуля вывода ПА в точку для управления углом курса во время движения.

Работа нашей системы по управлению углом курса будет происходить в несколько ключевых этапов:

1. Модуль вывода аппарата в точку получает на вход точку траектории подводного аппарата, на основании которой происходит выбор «субточки» и расчет желаемого угла курса .
2. Желаемый угол курса подается на контур курса вместе со значениями текущей угловой скорости и текущего угла курса подводного аппарата, полученными с датчиков угла и угловой скорости. На основании этих данных производится расчет ошибки по углу курса eψ​=wrap(ψcmd​−ψm​). Ошибка по курсу и измеренная величина угловой скорости передается модулю управления контуром курса.
3. Модуль управления контуром курса, реализованный в виде PD регулятора, рассчитывает на основе полученных данных значение управляющего сигнала **, который подается на движительно-рулевой комплекс ПА.**
4. **На основе управляющего сигнала движительно-рулевой комплекс подводного аппарата создаст необходимый момент тяги движителей, который будет передан в модель динамики вращения подводного аппарата вокруг оси Y.**
5. **За счет момента произойдет изменение угла курса ПА, которое будет рассчитано с применением динамической модели вращение аппарата по углу курса. Измененный угол курса будет измерен датчиком угла, а угловая скорость, сформированная моментом вокруг оси Y, будет измерена датчиком угловой скорости. Данные значения в качестве сигналов обратной связи будут переданы модулю управления контуром**
6. **Модуль вывода ПА в точку производит пересчет** желаемого угла курса относительно нового положения ПА и передаст новый желаемые угол курса модулю управления контуром курса, таким образом замыкая систему управления углом курса подводного аппарата

### 2.3.2 Работа контура курса и модуля вывода ПА в точку для управления маршевым движением ПА

Управление маршевым движением подводного аппарата также происходит в несколько этапов с участием модуля вывода аппарата в точку траектории и контура маршевого движения ПА. Следует выделить несколько основных процессов работы системы:

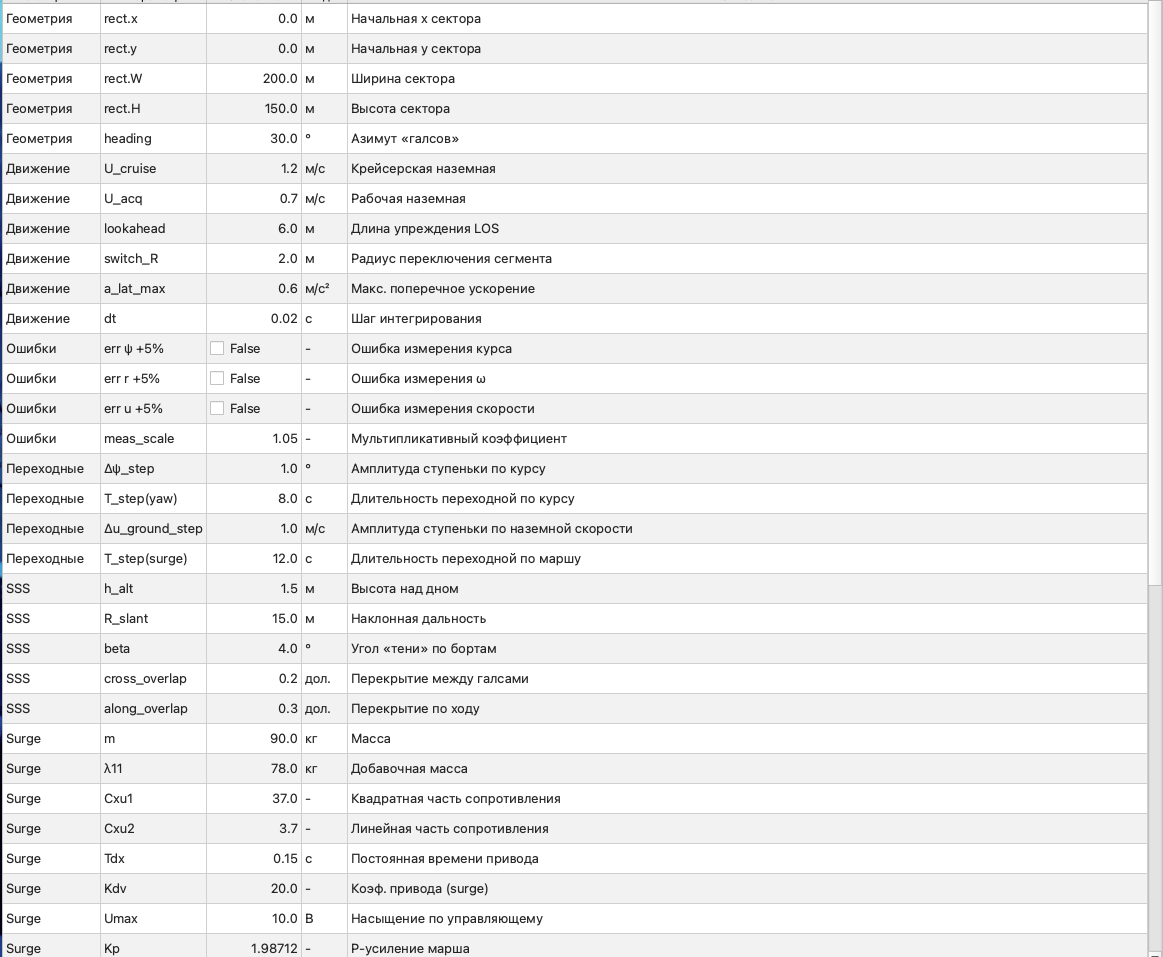
1. Модуль вывода аппарата в точку получает на вход следующую точку траектории. На основании координат этой точки высчитывается кривизна траектории при достижении следующей точки. Значение кривизны позже используется в расчете желаемой скорости подводного аппарата, чтобы обеспечить замедление аппарата при входе в поворот, что позволит избежать значительных выходов аппарата за пределы исследуемой зоны. Также, кроме расчета кривизны, модуль оценивает данную точку с точки зрения проведения съемки с использованием гидролокатором бокового обзора. Если точка является частью участка съемки, то модуль устанавливает желаемую скорость равной 0,7 метров в секунду, в противном случае скорость устанавливается в зависимости от необходимости совершать поворот (максимальная скорость на участках без поворота составляет 1,25 метров в секунду). На основе выбранной точки устанавливается желаемая координата, равная координате субточки. Желаемая координата и скорость передаются на контур марша.
2. На контур марша поступают желаемые значения с модуля вывода ПА в точку и обратная связь в виде данных с датчиков положения и линейной скорости. На основе этих данных считается ошибка по положению и скорости, которые приходят на управляющий модуль контура марша. По формуле высчитывается управляющий сигнал, который поступает на движительно-рулевой комплекс.
3. Управляющий сигнал в движительно-рулевом комплексе пересчитывается в силу тяги движителя по оси X, которая приходит на динамическую модель движения аппарата, где пересчитывается в скорость и пройденное расстояние.
4. Датчики угла и положения в качестве сигналов обратной связи отправят на модуль вывода аппарата в точку новые значения положения и скорости, тем самым замкнув систему.

Таким образом два контура вместе с модулем вывода аппарата в точку обеспечивают движение аппарата по заданной траектории.

# 3 Разработка интерфейса программного комплекса

Интерфейс нашей системы управления должен включать в себя поле для графического отображения полученной траектории и графиков параметров нашей системы (например: скорости и угла курса). Также в интерфейсе должна быть предусмотрена возможность задания параметров работы системы, чтобы сделать программный комплекс более эффективным и адаптивным к внешним факторам исследуемой зоны.

Для введения параметров было принято решение создать таблицу с редактируемыми полями, изначально заполненными стандартными значениями. При каждом запуске данные поля будут считывается, и система будет строить траекторию и проводить моделирование с учетом данных параметров. Таблица представлена на рисунке 3.1.

Рисунок 3.1 – таблица ввода параметров моделирования

Для отображения полученной траектории и графиков изменения ключевых параметров движения ПА на интерфейсе предусмотрено графическое поле с несколькими разделами, которое представлено на рисунке 3.2.

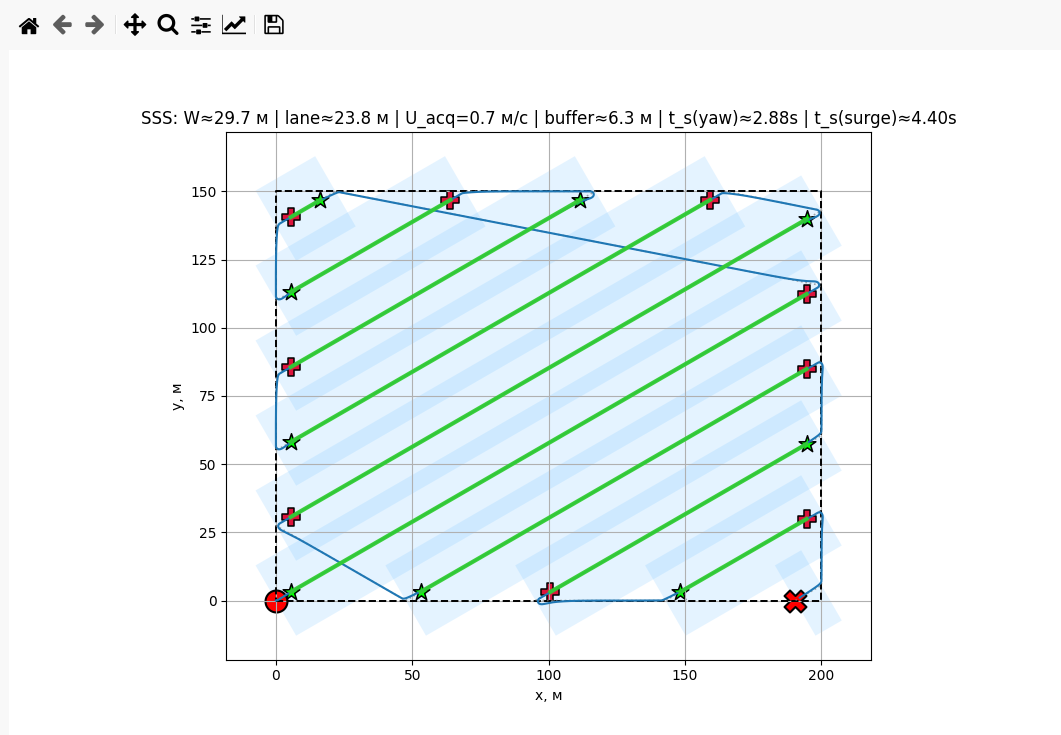


Рисунок 3.2 – графическое поле для вывода траектории

На данном графике отмечены при помощи большого кружку и крестика – начальная и конечная точки движения, при помощи звездочек и крестиков малых – точки начала и конца съемки соотвественно, при помощи зеленных линий – участки движения со съемкой, и при помощи прозрачных прямоугольников – зоны перекрытия гидролокатора бокового обзора.

В данном графическом поле мы также можем посмотреть график изменения скорости, представленный на рисунке 3.3 и график изменения угла курса, предсатвленный на рисунке 3.4. На графическом поле также представлены графики переходных процесоов, которые были показаын ранее при нанализе контуров курса и маршевого движения.

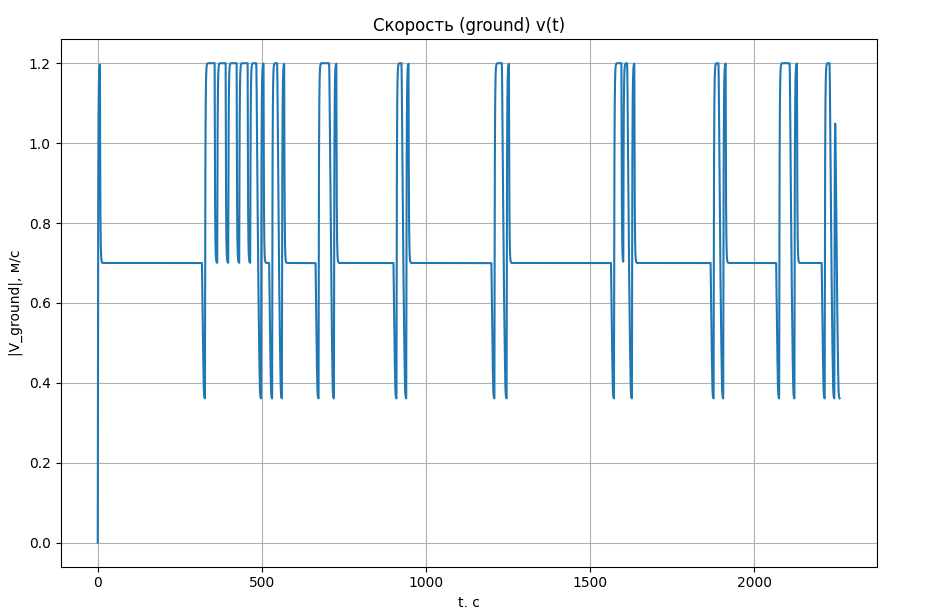


Рисунок 3.3 – график изменения скорости аппарата

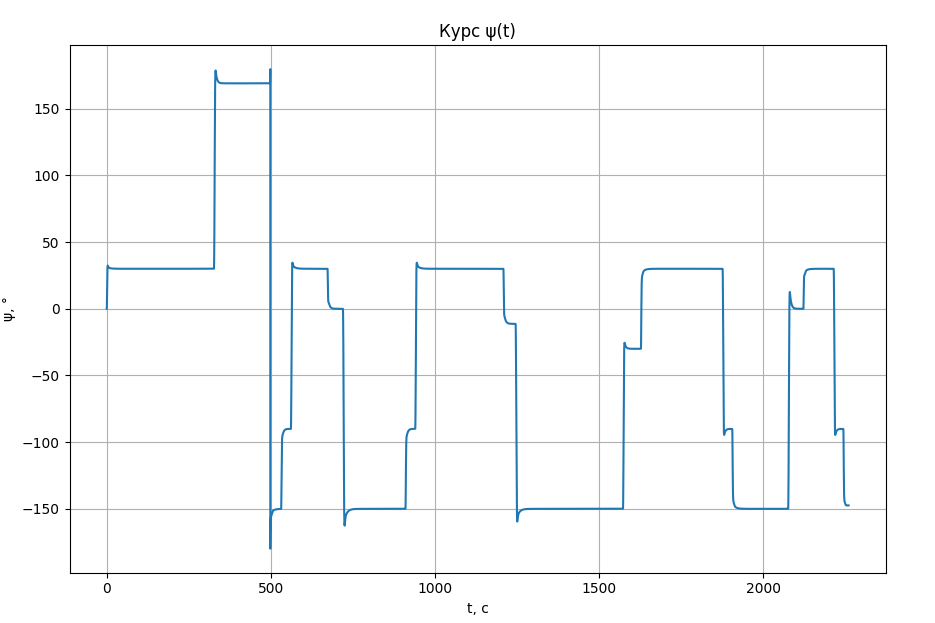


Рисунок 3.4 – график изменения угла курса аппарата

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта были выполнены все поставленные задачи. Удалось успешно достичь цели и разраотать систему управления траекторным движением ПА, которая учитывает характеристики аппарата и особенности окружающей местности (течение, размер зоны) при выполнении поставленной задачи сканирования дна выделенной зоны. Для системы был разработан удобный интерфейс управлениря параметрами и вывода результатов моделирования работы системы.

# СПИСОК ИСТОЧНИКОВ

1. Теория подводных лодок. А.П. Барсуков/ А.И. Ермолаев/ А.Н. Шмырев., Москва: Военное издательство Министерства обороны СССР, 1968., 656 с.
2. **Теория систем автоматического регулирования**. Издание третье, исправленное. Бесекерский В. А./ Попов Е. П., издательство «Наука», Главная редакция физико-математической литературы, М., 1975, 768 стр.

# ПРИЛОЖЕНИЕ А

|  |
| --- |
| import json  import math  from dataclasses import dataclass  from typing import List, Dict, Any, Tuple  import matplotlib  matplotlib.use("QtAgg")  from matplotlib.backends.backend\_qtagg import FigureCanvasQTAgg, NavigationToolbar2QT  from matplotlib.figure import Figure  import matplotlib.patches as mpatches  try:  import serial # type: ignore  except Exception:  serial = None  RAD2DEG = 180.0 / math.pi  DEG2RAD = math.pi / 180.0  def wrap\_rad(a: float) -> float:  while a <= -math.pi:  a += 2 \* math.pi  while a > math.pi:  a -= 2 \* math.pi  return a  class Kurs:  def \_\_init\_\_(  self,  Jy=90.0, L55=87.0,  Cwy1=75.0, Cwy2=7.4,  Tdv=0.15, Kdv=20.0, b=0.3,  Umax=10.0,  K1=0.6772091636, # В/град  K2=0.6157978918, # В/(град/с)  Mdist=0.0,  Fcur\_yaw\_mag=0.0,  Fcur\_yaw\_dir\_deg=0.0,  l\_yaw\_arm=1.0,  use\_5pct\_error\_psi=False,  use\_5pct\_error\_r=False,  meas\_scale=1.05,  dt=0.02,  ):  self.Jy, self.L55 = Jy, L55  self.Cwy1, self.Cwy2 = Cwy1, Cwy2  self.Tdv, self.Kdv, self.b = Tdv, Kdv, b  self.Km = 2.0 \* b \* Kdv  self.Umax = abs(Umax)  self.K1, self.K2 = K1, K2  self.Mdist = float(Mdist)  self.Fcur\_yaw = float(Fcur\_yaw\_mag)  self.th\_cur\_yaw = float(Fcur\_yaw\_dir\_deg) \* DEG2RAD  self.l\_yaw\_arm = float(l\_yaw\_arm)  self.dt = dt  self.psi = 0.0  self.wy = 0.0  self.Mdy = 0.0  self.use\_5pct\_error\_psi = use\_5pct\_error\_psi  self.use\_5pct\_error\_r = use\_5pct\_error\_r  self.meas\_scale = float(meas\_scale)  def \_meas\_psi(self, psi\_true):  return self.meas\_scale \* psi\_true if self.use\_5pct\_error\_psi else psi\_true  def \_meas\_r(self, r\_true):  return self.meas\_scale \* r\_true if self.use\_5pct\_error\_r else r\_true  def control\_voltage(self, psi\_rad, wy\_rad\_s, psi\_dist\_deg):  psi\_meas = self.\_meas\_psi(psi\_rad)  r\_meas = self.\_meas\_r(wy\_rad\_s)  e\_deg = psi\_dist\_deg - psi\_meas \* RAD2DEG  while e\_deg <= -180.0:  e\_deg += 360.0  while e\_deg > 180.0:  e\_deg -= 360.0  wy\_deg\_s = r\_meas \* RAD2DEG  u = self.K1 \* e\_deg - self.K2 \* wy\_deg\_s  if u > self.Umax:  u = self.Umax  if u < -self.Umax:  u = -self.Umax  return u  def f(self, psi, wy, Mdy, psi\_dist\_deg):  Upsi = self.control\_voltage(psi, wy, psi\_dist\_deg)  dMdy = (self.Km \* Upsi - Mdy) / self.Tdv  Jtot = self.Jy + self.L55  Mcur = self.l\_yaw\_arm \* self.Fcur\_yaw \* math.sin(self.th\_cur\_yaw - psi)  Mext = self.Mdist + Mcur  dwy = (Mdy + Mext - self.Cwy1 \* abs(wy) \* wy - self.Cwy2 \* wy) / Jtot  dpsi = wy  return dpsi, dwy, dMdy  def rk4\_step(self, psi\_dist\_deg):  y1 = (self.psi, self.wy, self.Mdy)  k1 = self.f(\*y1, psi\_dist\_deg)  y2 = tuple(y1[i] + 0.5 \* self.dt \* k1[i] for i in range(3))  k2 = self.f(\*y2, psi\_dist\_deg)  y3 = tuple(y1[i] + 0.5 \* self.dt \* k2[i] for i in range(3))  k3 = self.f(\*y3, psi\_dist\_deg)  y4 = tuple(y1[i] + self.dt \* k3[i] for i in range(3))  k4 = self.f(\*y4, psi\_dist\_deg)  self.psi = y1[0] + (self.dt / 6.0) \* (k1[0] + 2 \* k2[0] + 2 \* k3[0] + k4[0])  self.wy = y1[1] + (self.dt / 6.0) \* (k1[1] + 2 \* k2[1] + 2 \* k3[1] + k4[1])  self.Mdy = y1[2] + (self.dt / 6.0) \* (k1[2] + 2 \* k2[2] + 2 \* k3[2] + k4[2])  self.psi = wrap\_rad(self.psi)  def step(self, psi\_dist\_deg):  self.rk4\_step(psi\_dist\_deg)  return self.psi \* RAD2DEG  class MarshevPD:  def \_\_init\_\_(  self,  m=90.0, lam11=78.0,  Cxu1=37.0, Cxu2=3.7,  Tdx=0.15, Kdv=20.0,  Umax=10.0,  Kp=1.98712, Kd=1.20176,  at=12.0,  wn\_ref=1.2, zeta\_ref=0.7,  Ucur\_mag=0.0, Ucur\_dir\_deg=0.0,  use\_5pct\_error\_u=False,  meas\_scale=1.05,  dt=0.02,  ):  self.Meff = m + lam11  self.C1, self.C2 = Cxu1, Cxu2  self.Tdx = Tdx  self.Kdx = 2.0 \* Kdv  self.Umax = abs(Umax)  self.Kp = max(0.0, Kp)  self.Kd = max(0.0, Kd)  self.at = at  self.wn = max(1e-6, wn\_ref)  self.zr = max(0.2, zeta\_ref)  self.Ucur\_mag = float(Ucur\_mag)  self.Ucur\_dir = float(Ucur\_dir\_deg) \* DEG2RAD  self.use\_5pct\_error\_u = use\_5pct\_error\_u  self.meas\_scale = float(meas\_scale)  self.dt = dt  self.u = 0.0  self.x = 0.0  self.T = 0.0  self.ur = 0.0  self.urd = 0.0  def \_update\_ref\_filter(self, u\_ref):  ydd = self.wn \* self.wn \* (u\_ref - self.ur) - 2.0 \* self.zr \* self.wn \* self.urd  self.urd += self.dt \* ydd  self.ur += self.dt \* self.urd  return self.ur, self.urd  def \_drag(self, u):  return self.C1 \* abs(u) \* u + self.C2 \* u  def \_u\_dot(self, u, T):  return (T - self.\_drag(u)) / self.Meff  def \_ucmd\_from\_Tcmd(self, Tcmd, T):  u\_unsat = (self.Tdx / self.Kdx) \* (self.at \* (Tcmd - T)) + T / self.Kdx  if u\_unsat > self.Umax:  u\_unsat = self.Umax  if u\_unsat < -self.Umax:  u\_unsat = -self.Umax  return u\_unsat  def f(self, u, T, x, u\_ref\_ground, psi):  Uc\_par = self.Ucur\_mag \* math.cos(self.Ucur\_dir - psi)  u\_ref\_body = max(0.0, u\_ref\_ground - Uc\_par)  uref, uref\_dot = self.\_update\_ref\_filter(u\_ref\_body)  udot = self.\_u\_dot(u, T)  u\_meas = self.meas\_scale \* u if self.use\_5pct\_error\_u else u  e = uref - u\_meas  edot = uref\_dot - udot  Tff = self.C2 \* u\_meas + self.C1 \* abs(u\_meas) \* u\_meas  Tcmd = self.Meff \* (self.Kp \* e + self.Kd \* edot) + Tff  U = self.\_ucmd\_from\_Tcmd(Tcmd, T)  dT = (self.Kdx \* U - T) / self.Tdx  du = udot  dx = u  return du, dT, dx  def rk4\_step(self, u\_ref\_ground, psi):  u1, T1, x1 = self.u, self.T, self.x  k1 = self.f(u1, T1, x1, u\_ref\_ground, psi)  u2 = u1 + 0.5 \* self.dt \* k1[0]  T2 = T1 + 0.5 \* self.dt \* k1[1]  x2 = x1 + 0.5 \* self.dt \* k1[2]  k2 = self.f(u2, T2, x2, u\_ref\_ground, psi)  u3 = u1 + 0.5 \* self.dt \* k2[0]  T3 = T1 + 0.5 \* self.dt \* k2[1]  x3 = x1 + 0.5 \* self.dt \* k2[2]  k3 = self.f(u3, T3, x3, u\_ref\_ground, psi)  u4 = u1 + self.dt \* k3[0]  T4 = T1 + self.dt \* k3[1]  x4 = x1 + self.dt \* k3[2]  k4 = self.f(u4, T4, x4, u\_ref\_ground, psi)  self.u = u1 + (self.dt / 6.0) \* (k1[0] + 2 \* k2[0] + 2 \* k3[0] + k4[0])  self.T = T1 + (self.dt / 6.0) \* (k1[1] + 2 \* k2[1] + 2 \* k3[1] + k4[1])  self.x = x1 + (self.dt / 6.0) \* (k1[2] + 2 \* k2[2] + 2 \* k3[2] + k4[2])  def step(self, u\_ref\_ground, psi=0.0):  self.rk4\_step(u\_ref\_ground, psi)  return self.u  def sss\_swath\_and\_spacing(R, h, beta\_deg=4.0, cross\_overlap=0.15) -> Tuple[float, float]:  if R <= h:  raise ValueError("R\_slant должен быть > h\_alt.")  beta = math.radians(beta\_deg)  W = 2.0 \* math.sqrt(max(0.0, R \* R - h \* h)) - 2.0 \* h \* math.tan(beta / 2.0)  W = max(0.0, W)  S\_lane = (1.0 - cross\_overlap) \* W  return W, S\_lane  def build\_lawnmower\_in\_rect\_from\_corner(rect, lane\_spacing, heading\_deg, margin=0.0, eps=1e-6):  xmin, ymin, W, H = rect  xmin += margin; ymin += margin  xmax = xmin + (W - 2 \* margin)  ymax = ymin + (H - 2 \* margin)  if xmax <= xmin or ymax <= ymin:  raise ValueError("Слишком большой margin.")  p0 = (xmin, ymin)  th = math.radians(heading\_deg)  vx, vy = math.cos(th), math.sin(th)  nx, ny = -vy, vx  if nx < 0.0: nx, ny = -nx, -ny  corners = [(xmin, ymin), (xmax, ymin), (xmax, ymax), (xmin, ymax)]  proj = [(cx - p0[0]) \* nx + (cy - p0[1]) \* ny for (cx, cy) in corners]  d\_min, d\_max = min(proj), max(proj)  delta = max(1e-9, float(lane\_spacing))  def sort\_left\_first(A, B, eps=1e-9):  if A[0] < B[0] - eps: return (A, B)  if A[0] > B[0] + eps: return (B, A)  return (A, B) if A[1] <= B[1] else (B, A)  def clip\_line(d):  Px = p0[0] + d \* nx; Py = p0[1] + d \* ny  if abs(vx) < eps:  if Px < xmin - eps or Px > xmax + eps: return None  tx\_min, tx\_max = -math.inf, math.inf  else:  t1 = (xmin - Px) / vx; t2 = (xmax - Px) / vx  tx\_min, tx\_max = (min(t1, t2), max(t1, t2))  if abs(vy) < eps:  if Py < ymin - eps or Py > ymax + eps: return None  ty\_min, ty\_max = -math.inf, math.inf  else:  t3 = (ymin - Py) / vy; t4 = (ymax - Py) / vy  ty\_min, ty\_max = (min(t3, t4), max(t3, t4))  t\_enter = max(tx\_min, ty\_min); t\_exit = min(tx\_max, ty\_max)  if not (t\_enter < t\_exit): return None  A = (Px + t\_enter \* vx, Py + t\_enter \* vy)  B = (Px + t\_exit \* vx, Py + t\_exit \* vy)  if (A[0]-B[0])\*\*2 + (A[1]-B[1])\*\*2 < (10\*eps)\*\*2: return None  A, B = sort\_left\_first(A, B); return (A, B)  def canonical(seg):  A, B = seg  a = (round(A[0], 9), round(A[1], 9))  b = (round(B[0], 9), round(B[1], 9))  return a, b  segments = {}; k = 0  while True:  d\_list = [0.0] if k == 0 else [k\*delta, -k\*delta]  any\_added = False  for d in d\_list:  if d < d\_min - eps or d > d\_max + eps: continue  seg = clip\_line(d)  if seg is None: continue  key = canonical(seg)  if key in segments: continue  segments[key] = seg; any\_added = True  if not any\_added and (k\*delta > max(abs(d\_min), abs(d\_max)) + 5\*delta): break  k += 1  if k > 10000: break  if not segments: return [p0]  def seg\_d(seg):  A, B = seg  cx, cy = (0.5\*(A[0]+B[0]), 0.5\*(A[1]+B[1]))  return (cx - p0[0])\*nx + (cy - p0[1])\*ny  segs = list(segments.values()); segs.sort(key=seg\_d)  def dist2(P, Q): return (P[0]-Q[0])\*\*2 + (P[1]-Q[1])\*\*2  waypoints = [p0]  first\_idx = min(range(len(segs)), key=lambda i: dist2(p0, segs[i][0]))  A0, B0 = segs[first\_idx]; waypoints.extend([A0, B0]); cur = B0  for idx, seg in enumerate(segs):  if idx == first\_idx: continue  A, B = seg  if dist2(cur, A) <= dist2(cur, B):  waypoints.extend([A, B]); cur = B  else:  waypoints.extend([B, A]); cur = A  return waypoints  def los\_pure\_pursuit(pos, wp\_i, wp\_j, Ld):  x, y = pos  x1, y1 = wp\_i; x2, y2 = wp\_j  vx, vy = (x2 - x1), (y2 - y1)  seg\_len2 = vx\*vx + vy\*vy  if seg\_len2 < 1e-9:  return 0.0, (x2, y2), 0.0  t = ((x - x1)\*vx + (y - y1)\*vy) / seg\_len2  t\_clamp = max(0.0, min(1.0, t))  xs = x1 + t\_clamp\*vx; ys = y1 + t\_clamp\*vy  seg\_len = math.sqrt(seg\_len2)  s = min(t\_clamp\*seg\_len + Ld, seg\_len)  xt = x1 + (s/seg\_len)\*vx; yt = y1 + (s/seg\_len)\*vy  psi\_ref = math.atan2(yt - y, xt - x)  ex, ey = x - xs, y - ys  nx, ny = -vy/seg\_len, vx/seg\_len  e\_ct = ex\*nx + ey\*ny  return psi\_ref, (xt, yt), e\_ct  def limit\_speed\_by\_turn(epsi, d\_target, yaw\_sys: Kurs, a\_lat\_max=0.6, v\_cap=None, eps=1e-6):  epsi\_abs = abs(epsi)  d = max(eps, d\_target)  kappa\_req = 2.0 \* math.sin(epsi\_abs) / d  omega\_max = yaw\_sys.b \* yaw\_sys.Umax  v\_omega = omega\_max / max(eps, kappa\_req)  v\_a = math.sqrt(max(0.0, a\_lat\_max / max(eps, kappa\_req)))  v\_lim = min(v\_omega, v\_a)  if v\_cap is not None:  v\_lim = min(v\_lim, v\_cap)  return v\_lim  def send\_camera\_cmd(event, port="/dev/ttyUSB0", baudrate=9600, timeout=1.0):  cmd = bytes([0xFF])  if serial is None:  print(f"[CAMERA] {event}: send {cmd.hex().upper()} to {port} (emu)")  return True  try:  with serial.Serial(port=port, baudrate=baudrate, timeout=timeout) as ser:  ser.write(cmd); ser.flush()  print(f"[CAMERA] {event}: sent {cmd.hex().upper()} to {port}")  return True  except Exception as e:  print(f"[CAMERA][ERROR] {event}: {e}")  return False  def draw\_cov\_rectangle(ax, S, E, width, color="#6EC1FF", alpha=0.15, label=None):  import math  import matplotlib.patches as mpatches  dx, dy = E[0] - S[0], E[1] - S[1]  L = math.hypot(dx, dy)  if L < 1e-6:  return  ux, uy = dx / L, dy / L  nx, ny = -uy, ux  w2 = width / 2.0  p1 = (S[0] + nx \* w2, S[1] + ny \* w2)  p2 = (E[0] + nx \* w2, E[1] + ny \* w2)  p3 = (E[0] - nx \* w2, E[1] - ny \* w2)  p4 = (S[0] - nx \* w2, S[1] - ny \* w2)  ax.add\_patch(mpatches.Polygon(  [p1, p2, p3, p4],  closed=True,  facecolor=color,  edgecolor="none",  alpha=alpha,  label=label,  ))    @dataclass  class SimResult:  xs: List[float]  ys: List[float]  wps: List[Tuple[float, float]]  rect: Tuple[float, float, float, float]  acq\_spans: List[Tuple[Tuple[float, float], Tuple[float, float]]]  camera\_events: List[Tuple[str, Tuple[float, float], float]]  t\_hist: List[float]  psi\_deg: List[float]  u\_ground: List[float]  e\_ct\_hist: List[float]  meta: Dict[str, Any]  def run\_simulation(p: Dict[str, Any]) -> SimResult:  rect = (p["rect\_x"], p["rect\_y"], p["rect\_w"], p["rect\_h"])  heading\_deg = p["heading\_deg"]  h\_alt = p["h\_alt"]; R\_slant = p["R\_slant"]  beta\_deg = p["beta\_deg"]; cross\_overlap = p["cross\_overlap"]; along\_overlap = p["along\_overlap"]  U\_cruise = p["U\_cruise"]; U\_acq = p["U\_acq"]  lookahead = p["lookahead"]; switch\_R = p["switch\_R"]  a\_lat\_max = p["a\_lat\_max"]; dt = p["dt"]  use\_5pct\_error\_psi = p["use\_5pct\_error\_psi"]  use\_5pct\_error\_r = p["use\_5pct\_error\_r"]  use\_5pct\_error\_u = p["use\_5pct\_error\_u"]  meas\_scale = p["meas\_scale"]  yaw\_kwargs = dict(  Jy=p["Jy"], L55=p["L55"], Cwy1=p["Cwy1"], Cwy2=p["Cwy2"],  Tdv=p["Tdv"], Kdv=p["Kdv\_yaw"], b=p["b"], Umax=p["Umax\_yaw"],  K1=p["K1"], K2=p["K2"],  Mdist=p["Mdist"],  Fcur\_yaw\_mag=p["Fcur\_yaw\_mag"], Fcur\_yaw\_dir\_deg=p["Fcur\_yaw\_dir\_deg"], l\_yaw\_arm=p["l\_yaw\_arm"],  use\_5pct\_error\_psi=use\_5pct\_error\_psi, use\_5pct\_error\_r=use\_5pct\_error\_r,  meas\_scale=meas\_scale, dt=dt  )  surge\_kwargs = dict(  m=p["m"], lam11=p["lam11"], Cxu1=p["Cxu1"], Cxu2=p["Cxu2"],  Tdx=p["Tdx"], Kdv=p["Kdv\_surge"], Umax=p["Umax\_surge"],  Kp=p["Kp"], Kd=p["Kd"], at=p["at"],  wn\_ref=p["wn\_ref"], zeta\_ref=p["zeta\_ref"],  Ucur\_mag=p["Ucur\_mag"], Ucur\_dir\_deg=p["Ucur\_dir\_deg"],  use\_5pct\_error\_u=p["use\_5pct\_error\_u"], meas\_scale=meas\_scale, dt=dt  )  def estimate\_ts\_yaw(sys: Kurs, psi\_step\_deg=1.0, tol=0.02, Tmax=8.0):  tloc = 0.0; target = psi\_step\_deg; reached = False; ts = Tmax  while tloc < Tmax:  tloc += sys.dt  y = sys.step(target)  err = abs(target - y)  if not reached and err <= tol\*abs(target): reached = True; ts = tloc  if reached and err > tol\*abs(target): reached = False; ts = Tmax  return ts  def estimate\_ts\_surge(sys: MarshevPD, u\_step=1.0, tol=0.02, Tmax=12.0):  tloc = 0.0; target = u\_step; reached = False; ts = Tmax; psi0 = 0.0  while tloc < Tmax:  tloc += sys.dt  y\_rel = sys.step(target, psi=psi0)  y\_ground = max(0.0, y\_rel + sys.Ucur\_mag \* math.cos(sys.Ucur\_dir - psi0))  err = abs(target - y\_ground)  if not reached and err <= tol\*abs(target): reached = True; ts = tloc  if reached and err > tol\*abs(target): reached = False; ts = Tmax  return ts  ts\_yaw = estimate\_ts\_yaw(Kurs(\*\*yaw\_kwargs), psi\_step\_deg=1.0, Tmax=8.0)  ts\_surge = estimate\_ts\_surge(MarshevPD(\*\*surge\_kwargs), u\_step=1.0, Tmax=12.0)  ts\_dom = max(ts\_yaw, ts\_surge)  turn\_buffer = 1.2 \* U\_cruise \* ts\_dom  W\_use, lane\_spacing = sss\_swath\_and\_spacing(R\_slant, h\_alt, beta\_deg, cross\_overlap)  s\_ping = (1.0 - along\_overlap) \* W\_use  dt\_ping = s\_ping / max(1e-6, U\_acq)  wps = build\_lawnmower\_in\_rect\_from\_corner(rect, lane\_spacing, heading\_deg, margin=0.0)  def build\_acq\_spans\_on\_straights(wps\_list, buf):  spans\_ = []  for i in range(1, len(wps\_list) - 1, 2):  A = wps\_list[i]; B = wps\_list[i + 1]  L = math.hypot(B[0]-A[0], B[1]-A[1])  if L <= 2\*buf: continue  dx = (B[0]-A[0]) / L; dy = (B[1]-A[1]) / L  S = (A[0] + dx\*buf, A[1] + dy\*buf)  E = (B[0] - dx\*buf, B[1] - dy\*buf)  spans\_.append((S, E))  return spans\_  acq\_spans = build\_acq\_spans\_on\_straights(wps, turn\_buffer)  spans\_ex = []  for S, E in acq\_spans:  dx, dy = E[0]-S[0], E[1]-S[1]  L = math.hypot(dx, dy)  if L < 1e-6: continue  ux, uy = dx/L, dy/L  spans\_ex.append({"S": S, "E": E, "u": (ux, uy), "L": L,  "start\_fired": False, "end\_fired": False})  total\_dist = sum(math.hypot(wps[i+1][0]-wps[i][0], wps[i+1][1]-wps[i][1]) for i in range(len(wps)-1))  span\_len\_total = sum(math.hypot(E[0]-S[0], E[1]-S[1]) for (S, E) in acq\_spans)  span\_len\_total = max(0.0, min(span\_len\_total, total\_dist))  T\_nom = (span\_len\_total / max(1e-6, U\_acq)) + ((total\_dist - span\_len\_total) / max(1e-6, U\_cruise))  T\_end = 1.25 \* T\_nom  yaw = Kurs(\*\*yaw\_kwargs)  surge = MarshevPD(\*\*surge\_kwargs)  x, y = wps[0]  if len(wps) >= 2:  dx0, dy0 = wps[1][0]-wps[0][0], wps[1][1]-wps[0][1]  yaw.psi = math.atan2(dy0, dx0)  xs, ys = [x], [y]  psis\_deg = [yaw.psi \* RAD2DEG]  us\_ground = []  e\_ct\_hist = [0.0]  t\_hist = [0.0]  camera\_events: List[Tuple[str, Tuple[float, float], float]] = []  event\_perp\_tol = 2.0 # м  Ucx = surge.Ucur\_mag \* math.cos(surge.Ucur\_dir)  Ucy = surge.Ucur\_mag \* math.sin(surge.Ucur\_dir)  t = 0.0; i\_seg = 0  x\_prev, y\_prev = x, y  max\_steps = int(2.0 \* T\_end / dt) + 10000  steps = 0  while t < T\_end and i\_seg < len(wps)-1 and steps < max\_steps:  steps += 1  wp\_i = wps[i\_seg]; wp\_j = wps[i\_seg+1]  psi\_ref, target\_pt, e\_ct = los\_pure\_pursuit((x, y), wp\_i, wp\_j, lookahead)  xt, yt = target\_pt  d\_target = math.hypot(xt - x, yt - y)  capturing = False  for sp in spans\_ex:  Sx, Sy = sp["S"]; ux, uy = sp["u"]; L = sp["L"]  s\_prev = (x\_prev - Sx)\*ux + (y\_prev - Sy)\*uy  s = (x - Sx)\*ux + (y - Sy)\*uy  d\_perp = abs((x - Sx)\*(-uy) + (y - Sy)\*ux)  if (not sp["start\_fired"]) and (s\_prev < 0.0) and (s >= 0.0) and (d\_perp <= event\_perp\_tol):  send\_camera\_cmd("START")  camera\_events.append(("START", (Sx, Sy), t))  sp["start\_fired"] = True  if (not sp["end\_fired"]) and (s\_prev < L) and (s >= L) and (d\_perp <= event\_perp\_tol):  send\_camera\_cmd("END")  Ex, Ey = sp["E"]  camera\_events.append(("END", (Ex, Ey), t))  sp["end\_fired"] = True  if (0.0 <= s <= L) and (d\_perp <= event\_perp\_tol):  capturing = True  dist\_to\_end = math.hypot(wp\_j[0]-x, wp\_j[1]-y)  if capturing:  u\_ref\_ground = U\_acq  else:  u\_ref\_ground = U\_cruise \* max(0.3, min(1.0, dist\_to\_end / (2.0 \* lookahead)))  epsi = wrap\_rad(psi\_ref - yaw.psi)  u\_ref\_turn\_cap = limit\_speed\_by\_turn(epsi, d\_target, yaw\_sys=yaw, a\_lat\_max=a\_lat\_max, v\_cap=u\_ref\_ground)  u\_ref\_ground = max(0.0, min(u\_ref\_ground, u\_ref\_turn\_cap))  yaw.step(psi\_ref \* RAD2DEG)  surge.step(u\_ref\_ground, psi=yaw.psi)  psi = yaw.psi  Vx = surge.u \* math.cos(psi) + Ucx  Vy = surge.u \* math.sin(psi) + Ucy  v\_ground = math.hypot(Vx, Vy)  x\_prev, y\_prev = x, y  x += Vx \* dt; y += Vy \* dt  if dist\_to\_end < p["switch\_R"]:  i\_seg += 1  t += dt  xs.append(x); ys.append(y)  psis\_deg.append(psi \* RAD2DEG)  us\_ground.append(v\_ground)  e\_ct\_hist.append(e\_ct)  t\_hist.append(t)  meta = dict(  W\_use=W\_use, lane\_spacing=lane\_spacing, s\_ping=s\_ping, dt\_ping=dt\_ping,  turn\_buffer=turn\_buffer, U\_acq=U\_acq, U\_cruise=U\_cruise,  ts\_yaw=ts\_yaw, ts\_surge=ts\_surge  )  return SimResult(  xs=xs, ys=ys, wps=wps, rect=rect, acq\_spans=acq\_spans,  camera\_events=camera\_events, t\_hist=t\_hist, psi\_deg=psis\_deg,  u\_ground=us\_ground, e\_ct\_hist=e\_ct\_hist, meta=meta  )  @dataclass  class StepResponses:  t\_yaw: List[float]  yaw\_deg: List[float]  yaw\_ref: List[float]  yaw\_band: Tuple[float, float]  yaw\_Mp\_pct: float  yaw\_Ts: float | None  t\_surge: List[float]  u\_ground: List[float]  u\_ref\_ground: List[float]  surge\_band: Tuple[float, float]  surge\_Mp\_pct: float  surge\_Ts: float | None  def \_step\_metrics(t: List[float], y: List[float], ref: List[float], tol=0.05) -> Tuple[Tuple[float, float], float, float | None]:  if not t:  return (0.0, 0.0), 0.0, None  yr = ref[-1]; eps = 1e-9  A = max(abs(yr), eps)  y\_low = yr - tol \* A  y\_high = yr + tol \* A  sgn = 1.0 if yr >= 0 else -1.0  peak = max((yi - yr) \* sgn for yi in y)  Mp\_pct = max(0.0, 100.0 \* peak / A)  Ts = None  for i in range(len(t)):  if all(abs(y[j] - yr) <= tol \* A for j in range(i, len(t))):  Ts = t[i]; break  return (y\_low, y\_high), Mp\_pct, Ts  def compute\_step\_responses(p: Dict[str, Any]) -> StepResponses:  dt = p["dt"]  yaw\_kwargs = dict(  Jy=p["Jy"], L55=p["L55"], Cwy1=p["Cwy1"], Cwy2=p["Cwy2"],  Tdv=p["Tdv"], Kdv=p["Kdv\_yaw"], b=p["b"], Umax=p["Umax\_yaw"],  K1=p["K1"], K2=p["K2"],  Mdist=p["Mdist"],  Fcur\_yaw\_mag=p["Fcur\_yaw\_mag"], Fcur\_yaw\_dir\_deg=p["Fcur\_yaw\_dir\_deg"], l\_yaw\_arm=p["l\_yaw\_arm"],  use\_5pct\_error\_psi=p["use\_5pct\_error\_psi"], use\_5pct\_error\_r=p["use\_5pct\_error\_r"],  meas\_scale=p["meas\_scale"], dt=dt  )  sys\_yaw = Kurs(\*\*yaw\_kwargs)  T\_yaw = max(dt, float(p["yaw\_step\_T"]))  amp\_yaw = float(p["yaw\_step\_amp\_deg"])  t\_yaw = [0.0]; yaw\_deg = [sys\_yaw.psi \* RAD2DEG]; yaw\_ref = [amp\_yaw]  tcur = 0.0  while tcur < T\_yaw:  tcur += dt  y = sys\_yaw.step(amp\_yaw)  t\_yaw.append(tcur); yaw\_deg.append(y); yaw\_ref.append(amp\_yaw)  yaw\_band, yaw\_Mp, yaw\_Ts = \_step\_metrics(t\_yaw, yaw\_deg, yaw\_ref, tol=0.05)  surge\_kwargs = dict(  m=p["m"], lam11=p["lam11"], Cxu1=p["Cxu1"], Cxu2=p["Cxu2"],  Tdx=p["Tdx"], Kdv=p["Kdv\_surge"], Umax=p["Umax\_surge"],  Kp=p["Kp"], Kd=p["Kd"], at=p["at"],  wn\_ref=p["wn\_ref"], zeta\_ref=p["zeta\_ref"],  Ucur\_mag=p["Ucur\_mag"], Ucur\_dir\_deg=p["Ucur\_dir\_deg"],  use\_5pct\_error\_u=p["use\_5pct\_error\_u"], meas\_scale=p["meas\_scale"], dt=dt  )  sys\_surge = MarshevPD(\*\*surge\_kwargs)  T\_surge = max(dt, float(p["surge\_step\_T"]))  amp\_u = float(p["surge\_step\_amp\_ground"])  psi\_hold = 0.0  t\_surge = [0.0]  ug0 = max(0.0, sys\_surge.u + sys\_surge.Ucur\_mag \* math.cos(sys\_surge.Ucur\_dir - psi\_hold))  u\_ground = [ug0]; u\_ref\_ground = [amp\_u]  tcur = 0.0  while tcur < T\_surge:  tcur += dt  sys\_surge.step(amp\_u, psi=psi\_hold)  ug = max(0.0, sys\_surge.u + sys\_surge.Ucur\_mag \* math.cos(sys\_surge.Ucur\_dir - psi\_hold))  t\_surge.append(tcur); u\_ground.append(ug); u\_ref\_ground.append(amp\_u)  surge\_band, surge\_Mp, surge\_Ts = \_step\_metrics(t\_surge, u\_ground, u\_ref\_ground, tol=0.05)  return StepResponses(  t\_yaw=t\_yaw, yaw\_deg=yaw\_deg, yaw\_ref=yaw\_ref, yaw\_band=yaw\_band, yaw\_Mp\_pct=yaw\_Mp, yaw\_Ts=yaw\_Ts,  t\_surge=t\_surge, u\_ground=u\_ground, u\_ref\_ground=u\_ref\_ground, surge\_band=surge\_band, surge\_Mp\_pct=surge\_Mp, surge\_Ts=surge\_Ts  )  @dataclass  class ParamRow:  category: str  key: str  label: str  value: Any  ptype: str # 'float'|'int'|'bool'  unit: str  desc: str  def default\_param\_rows() -> List[ParamRow]:  return [  ParamRow("Геометрия", "rect\_x", "rect.x", 0.0, "float", "м", "Начальная x сектора"),  ParamRow("Геометрия", "rect\_y", "rect.y", 0.0, "float", "м", "Начальная y сектора"),  ParamRow("Геометрия", "rect\_w", "rect.W", 200.0, "float", "м", "Ширина сектора"),  ParamRow("Геометрия", "rect\_h", "rect.H", 150.0, "float", "м", "Высота сектора"),  ParamRow("Геометрия", "heading\_deg", "heading", 30.0, "float", "°", "Азимут «галсов»"),  ParamRow("SSS", "h\_alt", "h\_alt", 1.5, "float", "м", "Высота над дном"),  ParamRow("SSS", "R\_slant", "R\_slant", 15.0, "float", "м", "Наклонная дальность"),  ParamRow("SSS", "beta\_deg", "beta", 4.0, "float", "°", "Угол «тени» по бортам"),  ParamRow("SSS", "cross\_overlap", "cross\_overlap", 0.20, "float", "дол.", "Перекрытие между галсами"),  ParamRow("SSS", "along\_overlap", "along\_overlap", 0.30, "float", "дол.", "Перекрытие по ходу"),  ParamRow("Движение", "U\_cruise", "U\_cruise", 1.2, "float", "м/с", "Крейсерская наземная"),  ParamRow("Движение", "U\_acq", "U\_acq", 0.7, "float", "м/с", "Рабочая наземная"),  ParamRow("Движение", "lookahead", "lookahead", 6.0, "float", "м", "Длина упреждения LOS"),  ParamRow("Движение", "switch\_R", "switch\_R", 2.0, "float", "м", "Радиус переключения сегмента"),  ParamRow("Движение", "a\_lat\_max", "a\_lat\_max", 0.6, "float", "м/с²", "Макс. поперечное ускорение"),  ParamRow("Движение", "dt", "dt", 0.02, "float", "с", "Шаг интегрирования"),  ParamRow("Yaw", "Jy", "Jy", 90.0, "float", "кг·м²", "Момент инерции по курсу"),  ParamRow("Yaw", "L55", "L55", 87.0, "float", "кг·м²", "Добавочный момент инерции"),  ParamRow("Yaw", "Cwy1", "Cwy1", 75.0, "float", "-", "Квадратная часть сопротивления"),  ParamRow("Yaw", "Cwy2", "Cwy2", 7.4, "float", "-", "Линейная часть сопротивления"),  ParamRow("Yaw", "Tdv", "Tdv", 0.15, "float", "с", "Постоянная времени привода"),  ParamRow("Yaw", "Kdv\_yaw", "Kdv", 20.0, "float", "-", "Коэф. привода (yaw)"),  ParamRow("Yaw", "b", "b", 0.3, "float", "-", "Коэф. привода в омегу"),  ParamRow("Yaw", "Umax\_yaw", "Umax", 10.0, "float", "В", "Насыщение по управляющему"),  ParamRow("Yaw", "K1", "K1", 0.6772091636, "float", "В/°", "P-усиление"),  ParamRow("Yaw", "K2", "K2", 0.6157978918, "float", "В/(°/с)", "D-усиление"),  ParamRow("Yaw", "Mdist", "Mdist", 0.0, "float", "Н·м", "Пост. возм. момент (опц.)"),  ParamRow("Yaw.Течение", "Fcur\_yaw\_mag", "F\_cur (yaw)", 0.0, "float", "Н", "Сила течения(для момента)"),  ParamRow("Yaw.Течение", "Fcur\_yaw\_dir\_deg", "θ\_cur (yaw)", 0.0, "float", "°", "Азимут течения"),  ParamRow("Yaw.Течение", "l\_yaw\_arm", "l\_arm", 1.0, "float", "м", "Плечо приложения силы"),  ParamRow("Surge", "m", "m", 90.0, "float", "кг", "Масса"),  ParamRow("Surge", "lam11", "λ11", 78.0, "float", "кг", "Добавочная масса"),  ParamRow("Surge", "Cxu1", "Cxu1", 37.0, "float", "-", "Квадратная часть сопротивления"),  ParamRow("Surge", "Cxu2", "Cxu2", 3.7, "float", "-", "Линейная часть сопротивления"),  ParamRow("Surge", "Tdx", "Tdx", 0.15, "float", "с", "Постоянная времени привода"),  ParamRow("Surge", "Kdv\_surge", "Kdv", 20.0, "float", "-", "Коэф. привода (surge)"),  ParamRow("Surge", "Umax\_surge", "Umax", 10.0, "float", "В", "Насыщение по управляющему"),  ParamRow("Surge", "Kp", "Kp", 1.98712, "float", "-", "P-усиление марша"),  ParamRow("Surge", "Kd", "Kd", 1.20176, "float", "1/с", "D-усиление марша"),  ParamRow("Surge", "at", "a\_t", 12.0, "float", "1/с", "Скорость привода"),  ParamRow("Surge", "wn\_ref", "ω\_n", 1.2, "float", "рад/с", "Собств. частота фильтра ссылки"),  ParamRow("Surge", "zeta\_ref", "ζ", 0.7, "float", "-", "Демпфирование фильтра ссылки"),  ParamRow("Surge.Течение", "Ucur\_mag", "U\_cur", 0.0, "float", "м/с", "Скорость течения"),  ParamRow("Surge.Течение", "Ucur\_dir\_deg", "θ\_cur", 0.0, "float", "°", "Азимут течения"),  ParamRow("Ошибки", "use\_5pct\_error\_psi", "err ψ +5%", False, "bool", "-", "Ошибка измерения курса"),  ParamRow("Ошибки", "use\_5pct\_error\_r", "err r +5%", False, "bool", "-", "Ошибка измерения ω"),  ParamRow("Ошибки", "use\_5pct\_error\_u", "err u +5%", False, "bool", "-", "Ошибка измерения скорости"),  ParamRow("Ошибки", "meas\_scale", "meas\_scale", 1.05, "float", "-", "Мультипликативный коэффициент"),  ParamRow("Переходные", "yaw\_step\_amp\_deg", "Δψ\_step", 1.0, "float", "°", "Амплитуда ступеньки по курсу"),  ParamRow("Переходные", "yaw\_step\_T", "T\_step(yaw)", 8.0, "float", "с", "Длительность переходной по курсу"),  ParamRow("Переходные", "surge\_step\_amp\_ground", "Δu\_ground\_step", 1.0, "float", "м/с", "Амплитуда ступеньки по наземной скорости"),  ParamRow("Переходные", "surge\_step\_T", "T\_step(surge)", 12.0, "float", "с", "Длительность переходной по маршу"),  ]  class ParamTable(QtWidgets.QTableWidget):  COLS = ["Категория", "Параметр", "Значение", "Ед.", "Описание"]  def \_\_init\_\_(self, rows: List[ParamRow], parent=None):  super().\_\_init\_\_(parent)  self.setColumnCount(len(self.COLS))  self.setHorizontalHeaderLabels(self.COLS)  self.verticalHeader().setVisible(False)  self.setAlternatingRowColors(True)  self.setEditTriggers(QtWidgets.QAbstractItemView.EditTrigger.DoubleClicked | QtWidgets.QAbstractItemView.EditTrigger.SelectedClicked)  self.setSelectionBehavior(QtWidgets.QAbstractItemView.SelectionBehavior.SelectRows)  self.setShowGrid(True)  self.\_rows = rows  self.\_key\_index: Dict[int, str] = {}  self.populate(rows)  self.resizeColumnsToContents()  self.horizontalHeader().setStretchLastSection(True)  def populate(self, rows: List[ParamRow]):  self.setRowCount(len(rows))  self.\_key\_index.clear()  for i, r in enumerate(rows):  cat\_item = QtWidgets.QTableWidgetItem(r.category); cat\_item.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)  key\_item = QtWidgets.QTableWidgetItem(r.label); key\_item.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)  if r.ptype == "bool":  val\_item = QtWidgets.QTableWidgetItem()  val\_item.setFlags(  QtCore.Qt.ItemFlag.ItemIsUserCheckable  | QtCore.Qt.ItemFlag.ItemIsEnabled  | QtCore.Qt.ItemFlag.ItemIsSelectable  )  val\_item.setCheckState(QtCore.Qt.CheckState.Checked if bool(r.value) else QtCore.Qt.CheckState.Unchecked)  val\_item.setText("True" if bool(r.value) else "False")  else:  val\_item = QtWidgets.QTableWidgetItem(str(r.value))  val\_item.setFlags(  QtCore.Qt.ItemFlag.ItemIsEditable  | QtCore.Qt.ItemFlag.ItemIsEnabled  | QtCore.Qt.ItemFlag.ItemIsSelectable  )  val\_item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignRight | QtCore.Qt.AlignmentFlag.AlignVCenter)  unit\_item = QtWidgets.QTableWidgetItem(r.unit); unit\_item.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)  desc\_item = QtWidgets.QTableWidgetItem(r.desc); desc\_item.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)  self.setItem(i, 0, cat\_item)  self.setItem(i, 1, key\_item)  self.setItem(i, 2, val\_item)  self.setItem(i, 3, unit\_item)  self.setItem(i, 4, desc\_item)  for c in range(self.columnCount()):  self.item(i, c).setData(QtCore.Qt.ItemDataRole.UserRole, r.key)  self.\_key\_index[i] = r.key  self.setSortingEnabled(True)  self.sortItems(0, QtCore.Qt.SortOrder.AscendingOrder)  def to\_params(self) -> Dict[str, Any]:  params: Dict[str, Any] = {}  for r in range(self.rowCount()):  key = self.item(r, 0).data(QtCore.Qt.ItemDataRole.UserRole)  val\_item = self.item(r, 2)  spec = next((x for x in self.\_rows if x.key == key), None)  if spec is None:  continue  if spec.ptype == "bool":  val = val\_item.checkState() == QtCore.Qt.CheckState.Checked  else:  txt = (val\_item.text() or "").strip().replace(",", ".")  try:  if spec.ptype == "int":  val = int(float(txt))  else:  val = float(txt)  except Exception:  val\_item.setBackground(QtGui.QBrush(QtGui.QColor("#ffcccc")))  raise ValueError(f"Неверное число в параметре: {spec.label}")  params[key] = val  return params  def reset\_defaults(self):  self.populate(default\_param\_rows())  def load\_from\_dict(self, d: Dict[str, Any]):  for r in range(self.rowCount()):  key = self.item(r, 0).data(QtCore.Qt.ItemDataRole.UserRole)  if key not in d:  continue  spec = next((x for x in self.\_rows if x.key == key), None)  if spec is None:  continue  val\_item = self.item(r, 2)  if spec.ptype == "bool":  checked = QtCore.Qt.CheckState.Checked if bool(d[key]) else QtCore.Qt.CheckState.Unchecked  val\_item.setCheckState(checked)  val\_item.setText("True" if bool(d[key]) else "False")  else:  val\_item.setText(str(d[key]))  class PlotTab(QtWidgets.QWidget):  def \_\_init\_\_(self, title: str, parent=None):  super().\_\_init\_\_(parent)  self.fig = Figure(figsize=(6, 4), dpi=100)  self.canvas = FigureCanvasQTAgg(self.fig)  self.toolbar = NavigationToolbar2QT(self.canvas, self)  layout = QtWidgets.QVBoxLayout(self)  layout.addWidget(self.toolbar)  layout.addWidget(self.canvas)  self.ax = self.fig.add\_subplot(111)  self.ax.set\_title(title)  self.ax.grid(True)  class MainWindow(QtWidgets.QMainWindow):  def \_\_init\_\_(self):  super().\_\_init\_\_()  self.setWindowTitle("AUV Survey — табличные параметры и симуляция")  self.resize(1300, 850)  splitter = QtWidgets.QSplitter(QtCore.Qt.Orientation.Horizontal, self)  self.setCentralWidget(splitter)  self.param\_table = ParamTable(default\_param\_rows())  splitter.addWidget(self.param\_table)  self.tabs = QtWidgets.QTabWidget()  splitter.addWidget(self.tabs)  splitter.setStretchFactor(0, 3)  splitter.setStretchFactor(1, 6)  self.tab\_plan = PlotTab("План (траектория и покрытие)")  self.tab\_yaw = PlotTab("Курс ψ(t)")  self.tab\_speed = PlotTab("Скорость (ground) v(t)")  self.tab\_ect = PlotTab("Поперечная ошибка e\_ct(t)")  self.tab\_step\_yaw = PlotTab("Переходная по курсу")  self.tab\_step\_surge = PlotTab("Переходная по маршу")  self.tabs.addTab(self.tab\_plan, "План")  self.tabs.addTab(self.tab\_yaw, "Курс")  self.tabs.addTab(self.tab\_speed, "Скорость")  self.tabs.addTab(self.tab\_ect, "e\_ct")  self.tabs.addTab(self.tab\_step\_yaw, "Переходная по курсу")  self.tabs.addTab(self.tab\_step\_surge, "Переходная по маршу")    btn\_run = QtWidgets.QPushButton("Запустить")  btn\_reset = QtWidgets.QPushButton("Сброс")  btn\_save = QtWidgets.QPushButton("Сохранить пресет…")  btn\_load = QtWidgets.QPushButton("Загрузить пресет…")  btn\_run.clicked.connect(self.on\_run)  btn\_reset.clicked.connect(self.on\_reset)  btn\_save.clicked.connect(self.on\_save)  btn\_load.clicked.connect(self.on\_load)  top\_bar = QtWidgets.QWidget()  top\_layout = QtWidgets.QHBoxLayout(top\_bar)  top\_layout.addWidget(btn\_run)  top\_layout.addWidget(btn\_reset)  top\_layout.addStretch(1)  top\_layout.addWidget(btn\_save)  top\_layout.addWidget(btn\_load)  self.addToolBarBreak()  tool = QtWidgets.QToolBar("Управление")  tool.addWidget(top\_bar)  self.addToolBar(QtCore.Qt.ToolBarArea.TopToolBarArea, tool)  self.statusBar().showMessage("Готово")  def on\_reset(self):  self.param\_table.reset\_defaults()  self.statusBar().showMessage("Параметры сброшены к значениям по умолчанию.")  def on\_save(self):  try:  params = self.param\_table.to\_params()  except Exception as e:  QtWidgets.QMessageBox.critical(self, "Ошибка параметров", str(e)); return  fn, \_ = QtWidgets.QFileDialog.getSaveFileName(self, "Сохранить пресет", filter="JSON (\*.json)")  if fn:  with open(fn, "w", encoding="utf-8") as f:  json.dump(params, f, ensure\_ascii=False, indent=2)  self.statusBar().showMessage(f"Пресет сохранён: {fn}")  def on\_load(self):  fn, \_ = QtWidgets.QFileDialog.getOpenFileName(self, "Загрузить пресет", filter="JSON (\*.json)")  if fn:  with open(fn, "r", encoding="utf-8") as f:  data = json.load(f)  self.param\_table.load\_from\_dict(data)  self.statusBar().showMessage(f"Пресет загружен: {fn}")  def on\_run(self):  try:  params = self.param\_table.to\_params()  except Exception as e:  QtWidgets.QMessageBox.critical(self, "Ошибка параметров", str(e)); return  try:  res = run\_simulation(params)  except Exception as e:  QtWidgets.QMessageBox.critical(self, "Ошибка симуляции", str(e)); return  try:  steps = compute\_step\_responses(params)  except Exception as e:  QtWidgets.QMessageBox.critical(self, "Ошибка переходных", str(e)); return  for tab in (self.tab\_plan, self.tab\_yaw, self.tab\_speed, self.tab\_ect, self.tab\_step\_yaw, self.tab\_step\_surge):  tab.fig.clf(); tab.ax = tab.fig.add\_subplot(111); tab.ax.grid(True)  ax = self.tab\_plan.ax  xmin, ymin, W, H = res.rect  rx = [xmin, xmin + W, xmin + W, xmin, xmin]  ry = [ymin, ymin, ymin + H, ymin + H, ymin]  ax.plot(rx, ry, linestyle="--", color="k", linewidth=1.4, label="Сектор")  if len(res.wps) >= 2:  wx, wy = zip(\*res.wps)  ax.plot(wx, wy, ":", color="gray", alpha=0.7, linewidth=1.2, label="Маршрут (галсы)")  W\_use = res.meta["W\_use"]  drew\_cov\_label = False  for i in range(1, len(res.wps) - 1, 2):  A, B = res.wps[i], res.wps[i + 1]  label = "Покрытие свата (полная зона)" if not drew\_cov\_label else None  draw\_cov\_rectangle(ax, A, B, width=W\_use, alpha=0.18, label=label)  drew\_cov\_label = True  ax.plot(res.xs, res.ys, label="Траектория ПА (ground)")  for (S, E) in res.acq\_spans:  ax.plot([S[0], E[0]], [S[1], E[1]], linewidth=3.0, alpha=0.95, color="limegreen")  added\_start = added\_end = False  for ev, P, \_tt in res.camera\_events:  if ev == "START":  ax.scatter(P[0], P[1], s=160, color="lime", marker="\*", edgecolors="black",  linewidths=1.1, label=None if added\_start else "START")  added\_start = True  elif ev == "END":  ax.scatter(P[0], P[1], s=160, color="crimson", marker="P", edgecolors="black",  linewidths=1.1, label=None if added\_end else "END")  added\_end = True  ax.scatter(res.wps[0][0], res.wps[0][1], s=250, color="red", marker="o",  edgecolors="black", linewidths=1.5, label="Старт ПА")  ax.scatter(res.wps[-1][0], res.wps[-1][1], s=250, color="red", marker="X",  edgecolors="black", linewidths=1.5, label="Финиш ПА")  ax.set\_aspect("equal", adjustable="box")  ax.set\_xlabel("x, м"); ax.set\_ylabel("y, м")  ax.set\_title(  f"SSS: W≈{res.meta['W\_use']:.1f} м | lane≈{res.meta['lane\_spacing']:.1f} м | "  f"U\_acq={res.meta['U\_acq']:.1f} м/с | buffer≈{res.meta['turn\_buffer']:.1f} м | "  f"t\_s(yaw)≈{res.meta['ts\_yaw']:.2f}s | t\_s(surge)≈{res.meta['ts\_surge']:.2f}s"  )  self.tab\_plan.canvas.draw()  ax = self.tab\_yaw.ax  ax.plot(res.t\_hist, res.psi\_deg)  ax.set\_xlabel("t, c"); ax.set\_ylabel("ψ, °"); ax.set\_title("Курс ψ(t)")  self.tab\_yaw.canvas.draw()  ax = self.tab\_speed.ax  t\_sp = res.t\_hist[1:] if len(res.t\_hist) == len(res.u\_ground) + 1 else res.t\_hist[: len(res.u\_ground)]  ax.plot(t\_sp, res.u\_ground)  ax.set\_xlabel("t, c"); ax.set\_ylabel("|V\_ground|, м/с"); ax.set\_title("Скорость (ground) v(t)")  self.tab\_speed.canvas.draw()  ax = self.tab\_ect.ax  ax.plot(res.t\_hist, res.e\_ct\_hist)  ax.set\_xlabel("t, c"); ax.set\_ylabel("e\_ct, м"); ax.set\_title("Поперечная ошибка e\_ct(t)")  self.tab\_ect.canvas.draw()  ax = self.tab\_step\_yaw.ax  ax.plot(steps.t\_yaw, steps.yaw\_deg, label="ψ(t)")  ax.plot(steps.t\_yaw, steps.yaw\_ref, linestyle="--", label="ψ\_ref")  ax.axhline(steps.yaw\_band[0], linestyle="--", linewidth=1)  ax.axhline(steps.yaw\_band[1], linestyle="--", linewidth=1)  if steps.yaw\_Ts is not None:  ax.axvline(steps.yaw\_Ts, linestyle=":", linewidth=1)  ax.set\_title("Переходная по курсу")  ax.set\_xlabel("t, c"); ax.set\_ylabel("ψ, °")  txt = f"Mp ≈ {steps.yaw\_Mp\_pct:.1f}%"  if steps.yaw\_Ts is not None:  txt += f", Ts ≈ {steps.yaw\_Ts:.2f} c"  ax.legend(loc="best", title=txt)  self.tab\_step\_yaw.canvas.draw()  ax = self.tab\_step\_surge.ax  ax.plot(steps.t\_surge, steps.u\_ground, label="|V\_ground|(t)")  ax.plot(steps.t\_surge, steps.u\_ref\_ground, linestyle="--", label="u\_ref")  ax.axhline(steps.surge\_band[0], linestyle="--", linewidth=1)  ax.axhline(steps.surge\_band[1], linestyle="--", linewidth=1)  if steps.surge\_Ts is not None:  ax.axvline(steps.surge\_Ts, linestyle=":", linewidth=1)  ax.set\_title("Переходная по маршу")  ax.set\_xlabel("t, c"); ax.set\_ylabel("м/с")  txt = f"Mp ≈ {steps.surge\_Mp\_pct:.1f}%"  if steps.surge\_Ts is not None:  txt += f", Ts ≈ {steps.surge\_Ts:.2f} c"  ax.legend(loc="best", title=txt)  self.tab\_step\_surge.canvas.draw()  starts = sum(1 for ev, \_, \_ in res.camera\_events if ev == "START")  ends = sum(1 for ev, \_, \_ in res.camera\_events if ev == "END")  self.statusBar().showMessage(  f"L≈{(sum(math.hypot(res.xs[i+1]-res.xs[i], res.ys[i+1]-res.ys[i]) for i in range(len(res.xs)-1))):.1f} м; "  f"W≈{res.meta['W\_use']:.1f} м; lane≈{res.meta['lane\_spacing']:.1f} м; "  f"s\_ping≈{res.meta['s\_ping']:.2f} м; dt\_ping≈{res.meta['dt\_ping']:.2f} c; "  f"START={starts}, END={ends}"  )  def main():  import sys  app = QtWidgets.QApplication(sys.argv)  app.setStyle("Fusion")  pal = QtGui.QPalette()  pal.setColor(QtGui.QPalette.ColorRole.Window, QtGui.QColor(248, 248, 248))  pal.setColor(QtGui.QPalette.ColorRole.Base, QtGui.QColor(255, 255, 255))  pal.setColor(QtGui.QPalette.ColorRole.AlternateBase, QtGui.QColor(242, 242, 242))  app.setPalette(pal)  w = MainWindow()  w.show()  sys.exit(app.exec())  if \_\_name\_\_ == "\_\_main\_\_":  main() |