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 4](#_Toc212570297)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5](#_Toc212570298)

[ОСНОВНАЯ ЧАСТЬ 6](#_Toc212570299)

[1. Синтез модели ПА 7](#_Toc212570300)

[1.1 Синтез динамической модели аппарата по контуру курса 7](#_Toc212570301)

[1.2 Синтез динамической модели аппарата по контуру маршевого движения 11](#_Toc212570302)

[1.3 Движительно-рулевой комплекс подводного аппарата 14](#_Toc212570303)

[2 Синтез БФС ДРК 15](#_Toc212570304)

[3 Синтез СУ нижнего уровня 16](#_Toc212570305)

[3.1 СУ нижнего уровня в контуре курса 16](#_Toc212570306)

[3.2 СУ нижнего уровня в контуре марша 21](#_Toc212570307)

[4 Контроль полученных контуров управления 22](#_Toc212570308)

[4.1 Проверка переходного процесса контура управления углом курса 22](#_Toc212570309)

[4.2 Проверка переходного процесса контура управления маршевым движением 24](#_Toc212570310)

[5 Модуль вывода ПА в точку 26](#_Toc212570311)

[5.1 Выбор метода вывода аппарата в точку 26](#_Toc212570312)

[5.2 Описание выбранного метода 27](#_Toc212570313)

[5.3 Контроль скорости аппарата в зависимости от кривизны траектории. 28](#_Toc212570314)

[6 Разработка интерфейса программного комплекса 30](#_Toc212570315)

[СПИСОК ИСТОЧНИКОВ 35](#_Toc212570316)

[ПРИЛОЖЕНИЕ А 36](#_Toc212570317)

# 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.

В состав системы управления траекторным движением ПА, представленной на рисунке 1, входят контуры марша и курса, а также модуль вывода ПА в точку.

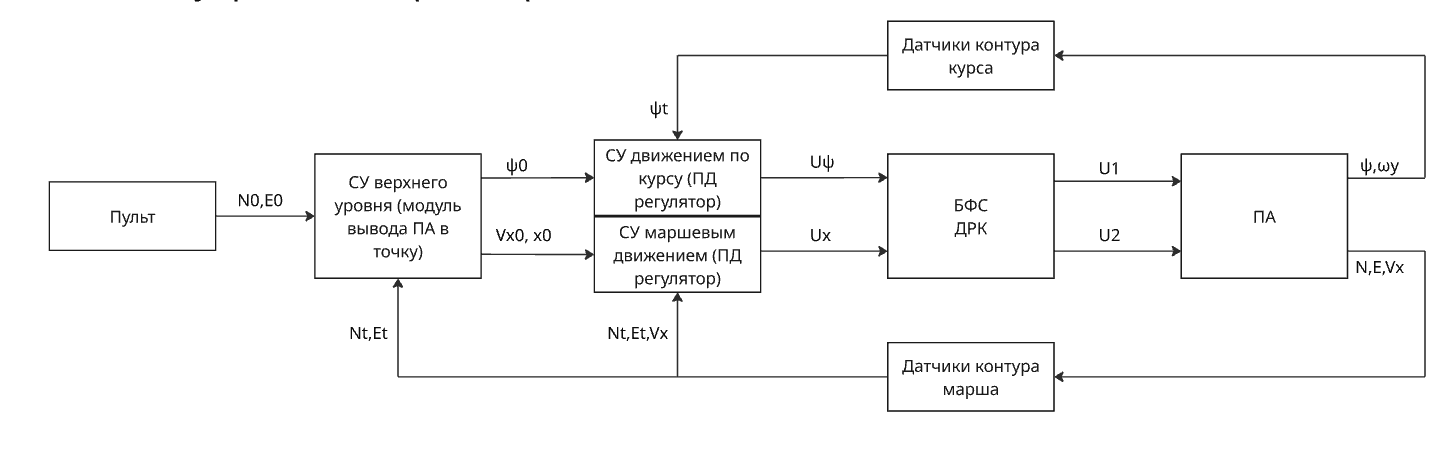


Рисунок 1 – система управления траекторным движением ПА.

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

1. Синтез модели ПА

## 1.1 Синтез динамической модели аппарата по контуру курса

Синтез динамической модели ПА является первым этапом синтеза общей математической модели подводного аппарата по сепаратным контурам курса и марша.

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), принимаем равным 0, так как согласно техническому заданию углы крена и дифферента мы принимаем равными 0.

Описание момента гидродинамических 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 1.1, в плоскости горизонта, как модели динамики вращательного движения аппарата.

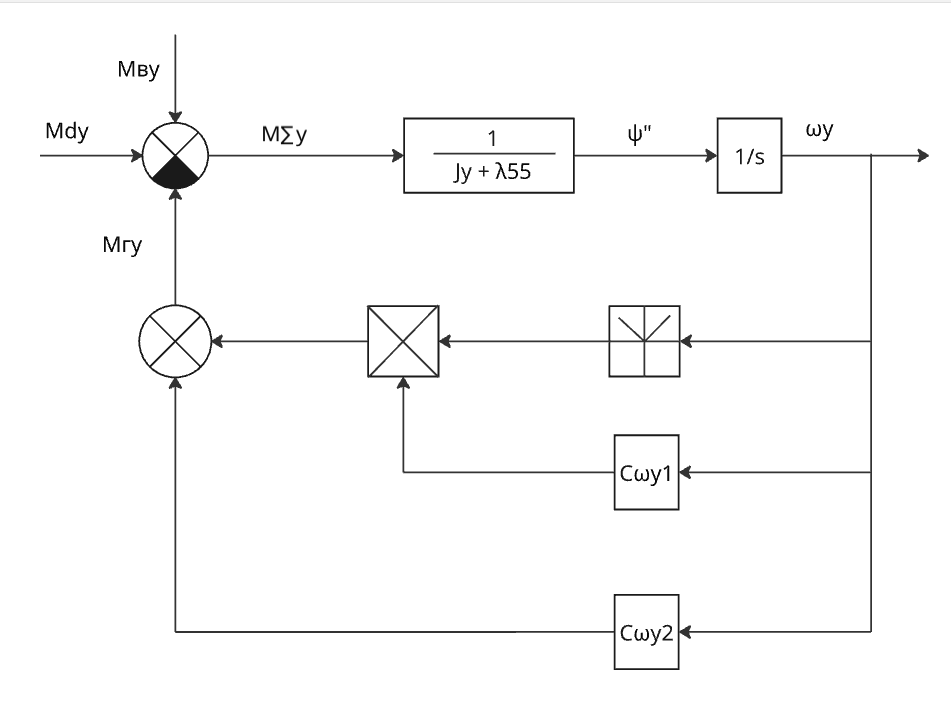


Рисунок 1.1 – динамическая модель движения аппарата по углу курса.

## 1.2 Синтез динамической модели аппарата по контуру маршевого движения

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 (1.8)– функцию перекрестных связей мы отбросим

Таким образом, формула (1.8) поступательного движения ПА вдоль оси X, проходящей через центр масс, принимает вид:

Рассмотрим величину суммарной силы по оси Х в формуле (1.9).Уравнение данной силы имеет вид:

Где:

, Н

, Н

Величиной силы Архимеда в формуле (1.9) мы можем пренебречь, так как движемся строго в плоскости горизонта с нулевыми углами крена и дифферента.

Проекцию силы гравитации в формуле (1.9) 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 (1.9) 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 (1.9) и (1.10), формула динамической модели ПА при движении по маршу принимает вид:

Уравнение (1.11) будет реализовано в нашей системе управления траекторным движением ПА, представленной на рисунке 1.2, 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.

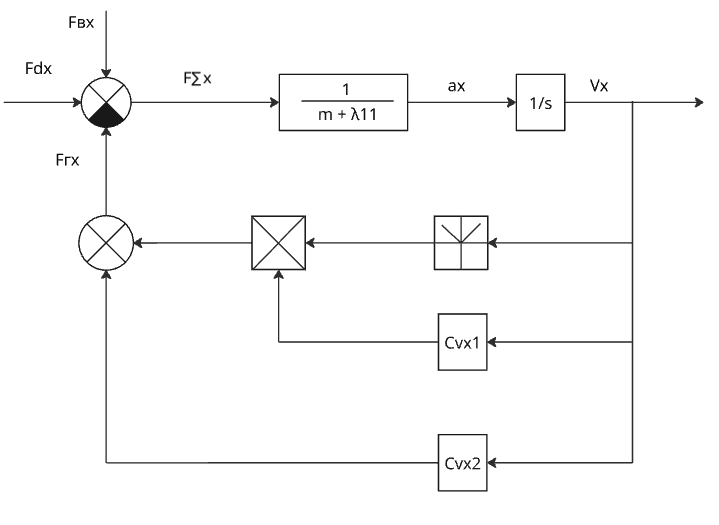


Рисунок 1.2 – динамическая модель маршевого движения ПА

## 1.3 Движительно-рулевой комплекс подводного аппарата

В качестве передаточной функции движителя принято использовать апериодическое звено первого порядка данного вида:

На аппарате имеются два винта на расстоянии  друг от друга (по ширине корпуса), так что плечо каждого до ЦМ — . Уравнение для сил тяги движителей 1 и 2 с применением функции (1.12) имеет вид:

Чистый момент вокруг :

Таким образом, опираясь на уравнения (1.13), (1.14), (1.15), мы получаем модель движительно-рулевого комплекса, представленного на рисунке 1.3, в которой в качестве входного управляющего сигнала выступает напряжения , которые подаются на комплекс с БФС ДРК.

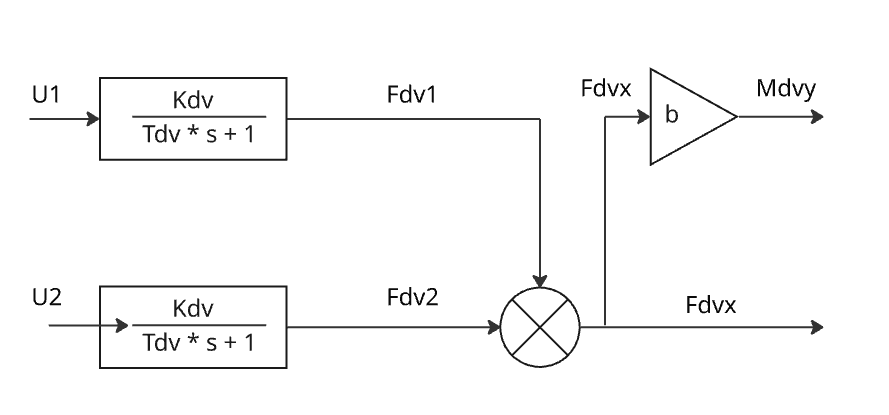


Рисунок 1.3 – модель движительно-рулевого комплекса

# 2 Синтез БФС ДРК

Так как наши движители работают одновременно в контурах курса и марша, они получают сигналы управления с СУ нижнего уровня обоих контуров единовременно.

Для объединения данных сигналов в единый управляющий сигнал на движитель, а также установки ограничения по максимально возможному управляющему сигналу, в данной системе реализован блок формирования сигнала движительно-рулевого комплекса.

В состав блока входят два ограничителя сигнала – на входе при получении сигнала с СУ нижнего уровня контуров курса и марша для предотвращения подачи исходно слишком высокого уровня сигнала на сам блок, и на выходе для ограничения уже сформированных после сложения сигналов . Так как при подаче положительного сигнала на второй движитель он создает вращение вокруг оси Y по часовой стрелке, при сложении сигнала управления на данный движитель с СУ контура курса он будет умножен на -1.

Таким образом, мы получаем БФС ДРК системы, представленный на рисунке 2.1

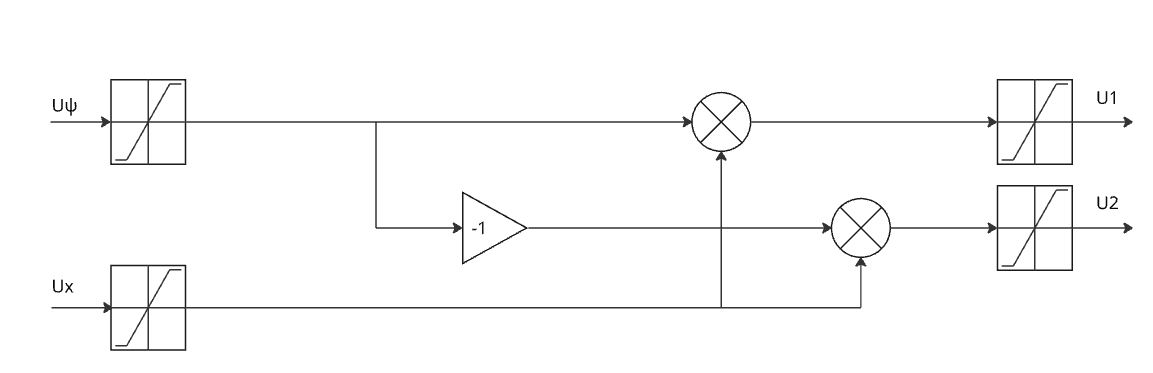


Рисунок 2.1 – БФС ДРК

# 3 Синтез СУ нижнего уровня

## 3.1 СУ нижнего уровня в контуре курса

В качестве СУ нижнего уровня в контуре курса было принято решение выбрать PD регулятор, 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. Конечный вид контура курса представлен на рисунке 3.1

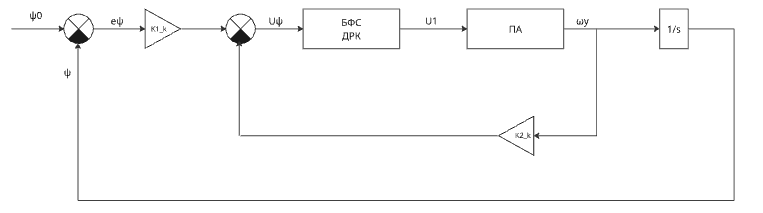


Рисунок 3.1 – контур управелния углом курса

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

Где:

- величина угловой скорости аппарата, рад/с

Коэффициенты и в уравнении (3.1)– коэффициенты PD регулятора. Для их расчета необходимо провести линеаризацию динамичсеких моделей ПА в контурах курса.

Для линеаризации конутра курса найдем частную производную уравнения (1.7) вокруг точки ωy'. Полученная линеаризованная модель представлена на рисунке 3.2

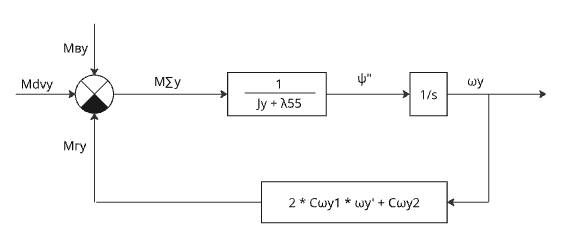


Рисунок 3.2 – линеаризованная динамическая модель ПА в контуре курса

Запишем передаточную функцию контура скорости Wc(p) с применением формул (1.15) и (1.7):

Где:

После приведения передаточной функции (3.2) 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 (3.4) и (3.5):

Далее запишем передаточную функцию разомкнутого контура углового положения:

Где:

Для приближенных расчетов примем равной 1, тогда передаточная функция (3.3) 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 (3.6).

Подставляем полученное значение в формулу (3.7) и получаем:

Коэффициенты для СУ нижнего уровня контура курса, представляющего собой PD регулятор, были найдены.

## 3.2 СУ нижнего уровня в контуре марша

По аналогии с СУ контура курса для СУ нижнего уровня контура маршевого движения был выбран PD регулятор. Тогда контур управления маршем принимает вид, представленный на рисунке 3.3.

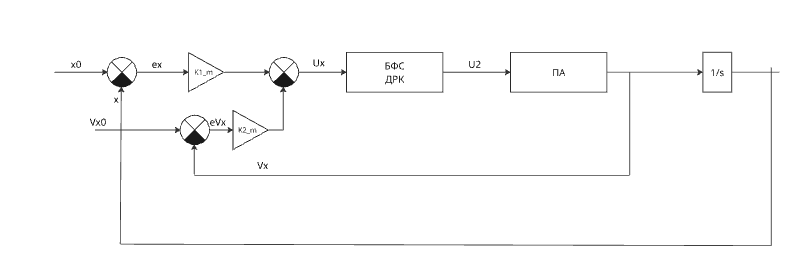


Рисунок 3.3 – контур управления движением по маршу

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

Где:

- величина ошибка по линейной скорости аппарата, м/с

Коэффициенты и в уравнении (3.9)– коэффициенты PD регулятора. Для их расчета необходимо провести линеаризацию динамичсеких моделей ПА в контурах марша.

Для линеаризации конутра курса найдем частную производную уравнения (1.11) вокруг точки Vx'. Полученная линеаризованная модель представлена на рисунке 3.4

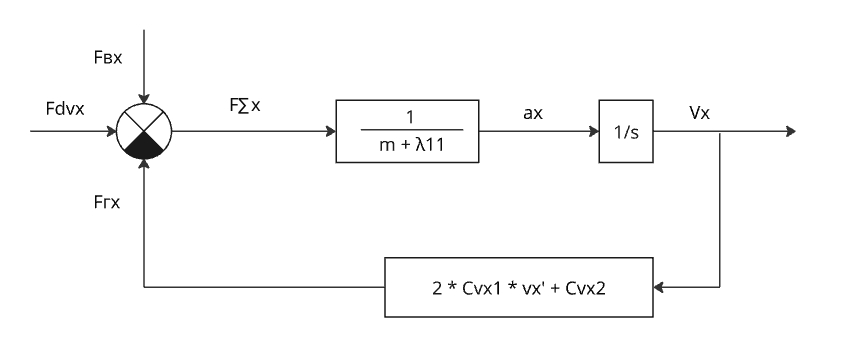


Рисунок 3.4 – линеаризованная динамическая модель ПА в контуре марша

По аналогии с контуром курса находим коэффициенты для контура маршевого движения.

Коэффициенты для СУ нижнего уровня контура марша, представляющего собой PD регулятор, были найдены.

# 4 Контроль полученных контуров управления

## 4.1 Проверка переходного процесса контура управления углом курса

Чтобы убедиться в правильности модели контура управления углом курса, мы построим переходной процесс контура – реакцию на единичное воздействие нашей системы, и считаем два ключевых параметра для нас – время установившегося процесса и перерегулирование.

Так как в системе реализован БФС ДРК, который не был учтен при синтезе контуров, мы повысим значения коэффициентов PD регулятора для достижения нужных значений времени переходного процесса и перерегулирования. После пересчета было принято решение принять

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

После запуска моделирования получаем переходной процесс контура курса, представленный на рисунке 4.1

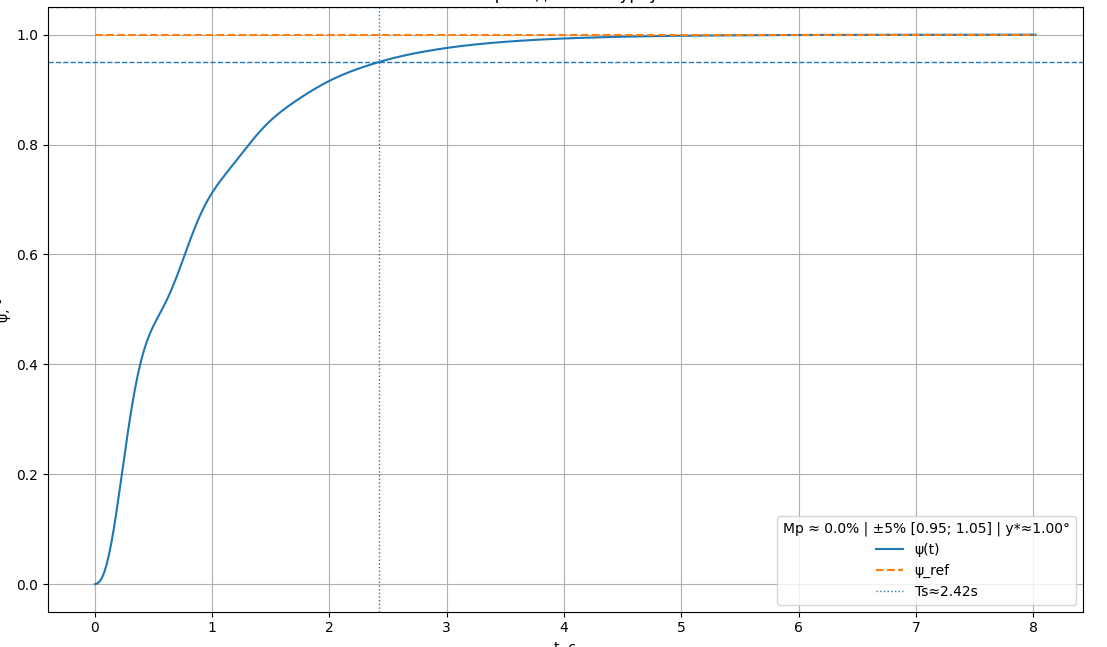


Рисунок 4.1 – переходной процесс в контуре курса

Полученные время установившегося процесса и перерегулирование соответствуют требованиям нашей системы.

## 4.2 Проверка переходного процесса контура управления маршевым движением

По аналогии с контуром курса мы построим переходные процессы контура марша – переходной процесс по скорости и координате маршевого движения.

Из-за наличия блока формирования сигнала коэффициенты PD регулятора также претерпели изменения в контуре марша. Мы принимаем и

Переходной процесс по координате представлен на рисунке 4.2.

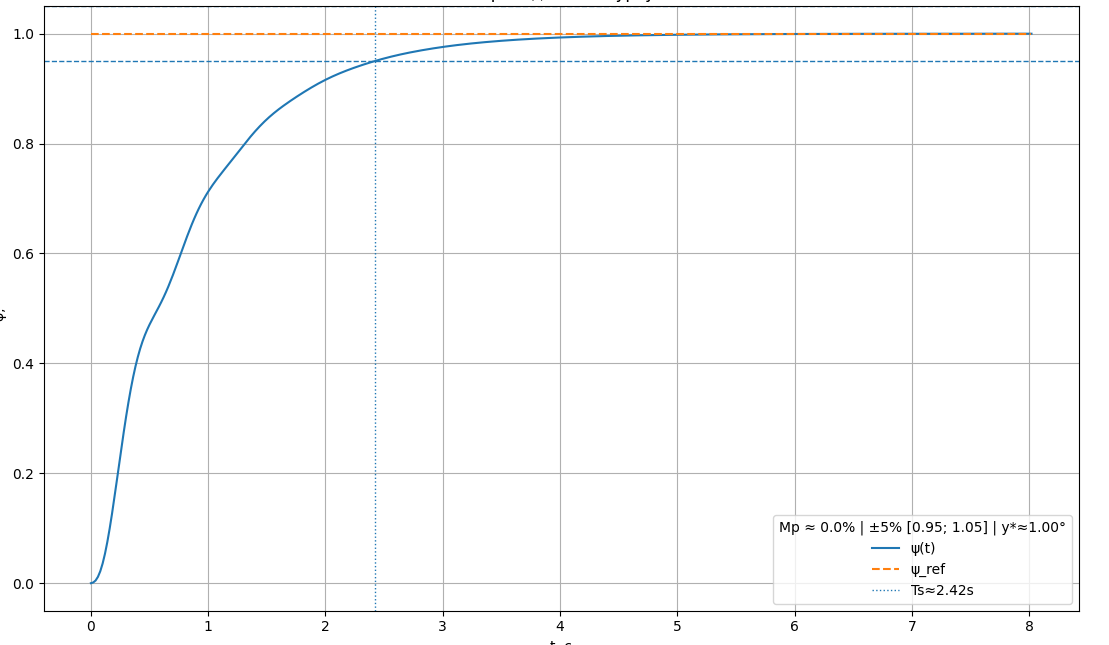


Рисунок 4.2 – переходной процесс контура марша по координате

Переходной процесс по скорости представлен на рисунке 4.3

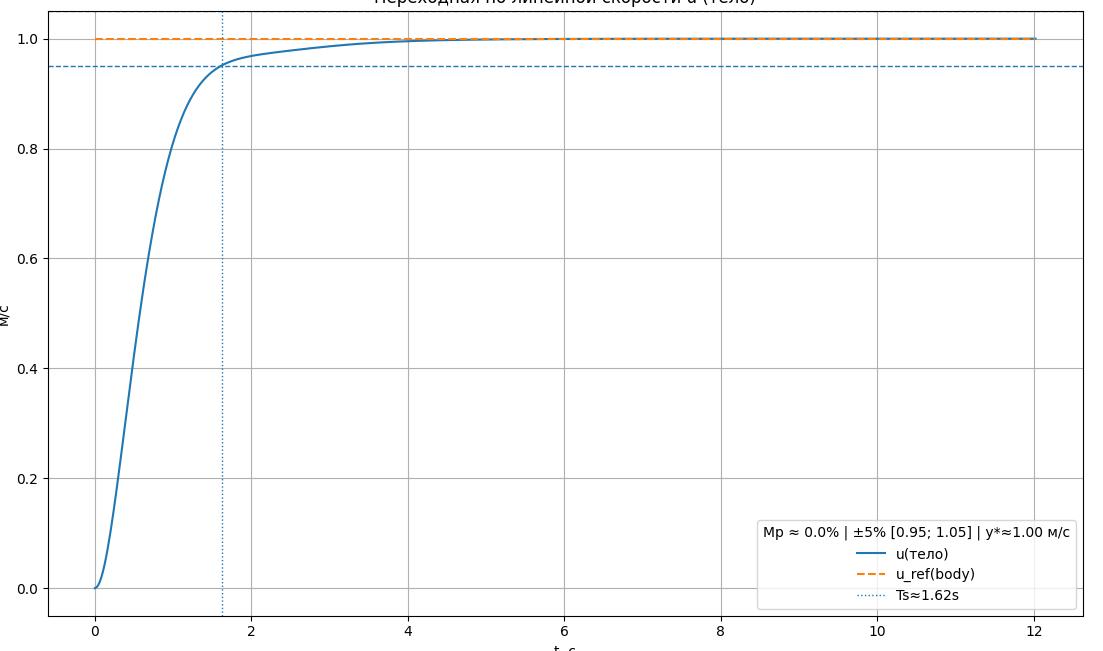


Рисунок 4.3 – переходной процесс в контуре марша по скорости

Полученные время установившегося процесса и перерегулирование соответствуют требованиям нашей системы.

Можно сделать вывод, что контуры управления углом курса и маршевым движением работают корректно и соответствуют техническому заданию.

# 5 Модуль вывода ПА в точку

5.1 Выбор метода вывода аппарата в точку

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.

5.2 Описание выбранного метода

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 5.1.

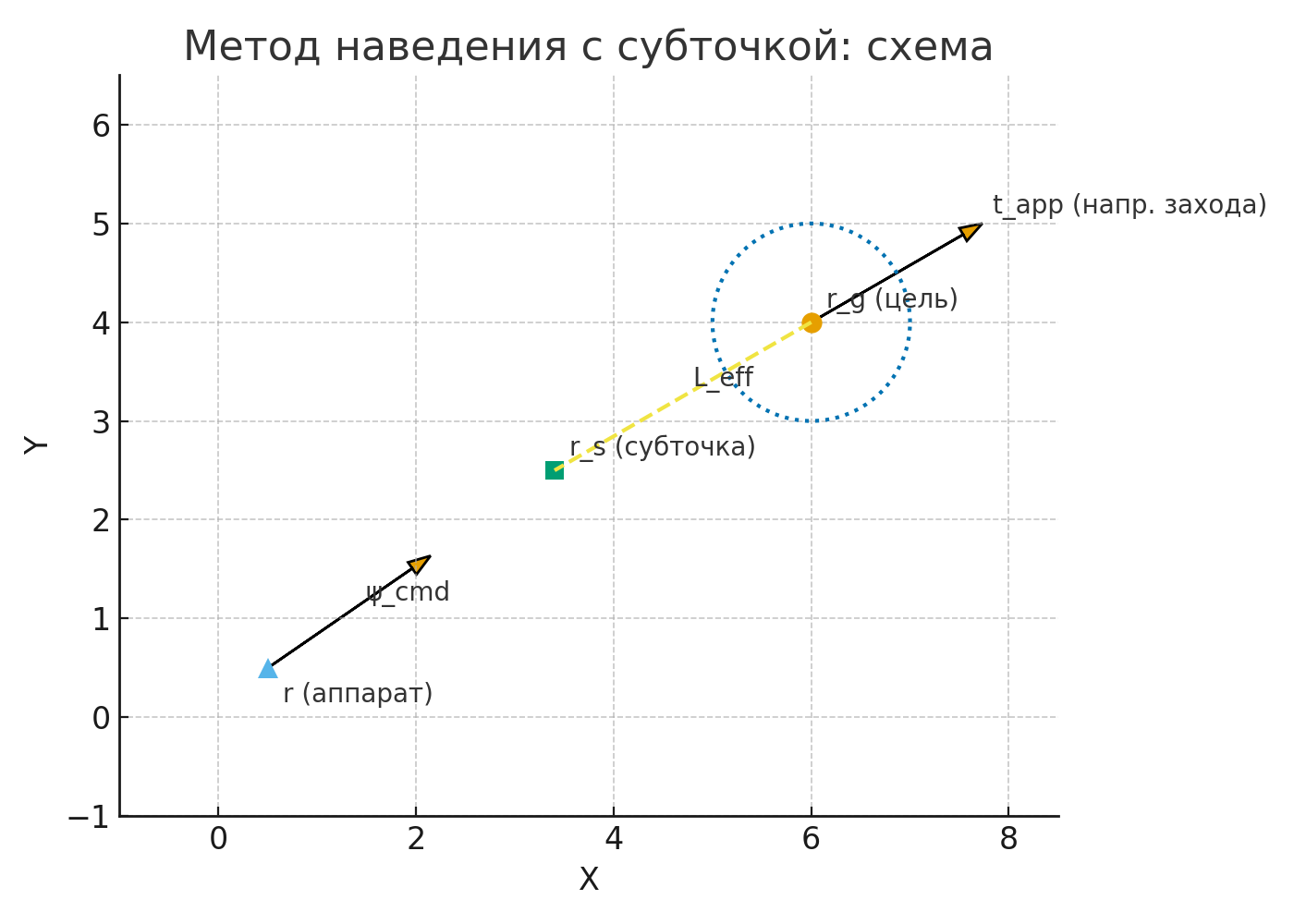


Рисунок 5.1 – схема метода наведения в субточку.

Расчет командного и x0 производится по следующим формулам:

(5.1)

(5.2)

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.

5.3 Контроль скорости аппарата в зависимости от кривизны траектории.

Ключевым понятием при использовании метода вывода ПА в точку с применением метода «субточки» при контроле скорости аппарата является поперечная ошибка. Чтобы рассчитать поперечную ошибку, нам нужно рассчитать направление t пути от текущей точки к целевой по формуле:

Тогда поперечная ошибка с применением формулы (5.3) — это скалярная величина, равная signed-расстоянию от текущей точки ПА до этой линии:

(5.4)

Знак будет показывать, справа или слева от маршрута находится аппарат, что важно для управления поворотом аппарата.

По значению поперечной ошибки, полученному при помощи формулы (5.4), мы рассчитываем кривизну траектории по формуле:

где:

Далее мы рассчитываем необходимую скорость с учетом кривизны траектории, найденной по формуле (5.5), по формуле:

где:

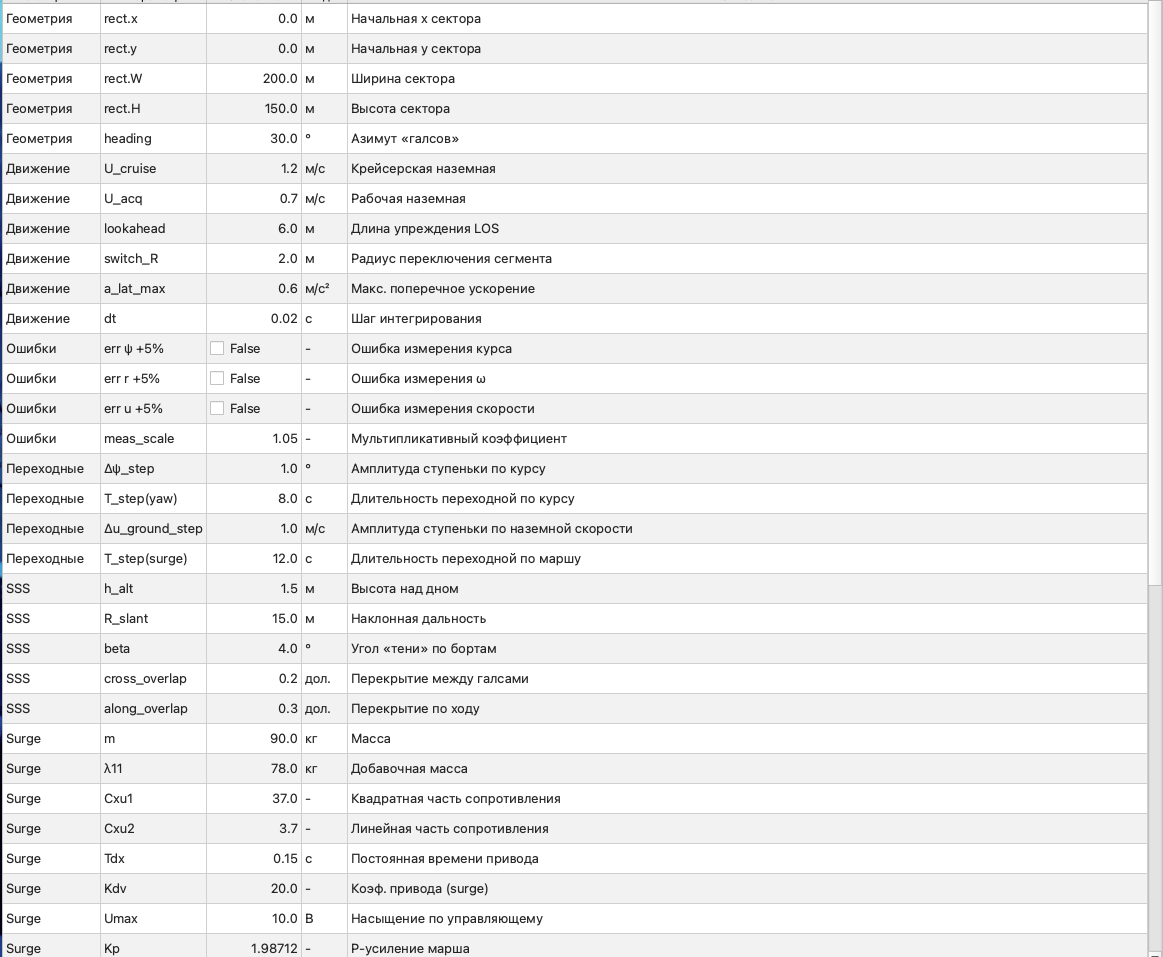
Найденную скорость по формуле (5.6) мы ограничиваем предельными минимальным и максимальным значением для определения командной Vx0:

Таким образом мы получаем все сигналы управления внешней СУ, реализованной в виде модуля вывода ПА в точку.

# 6 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 6.1.

Рисунок 6.1 – 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 6.2.

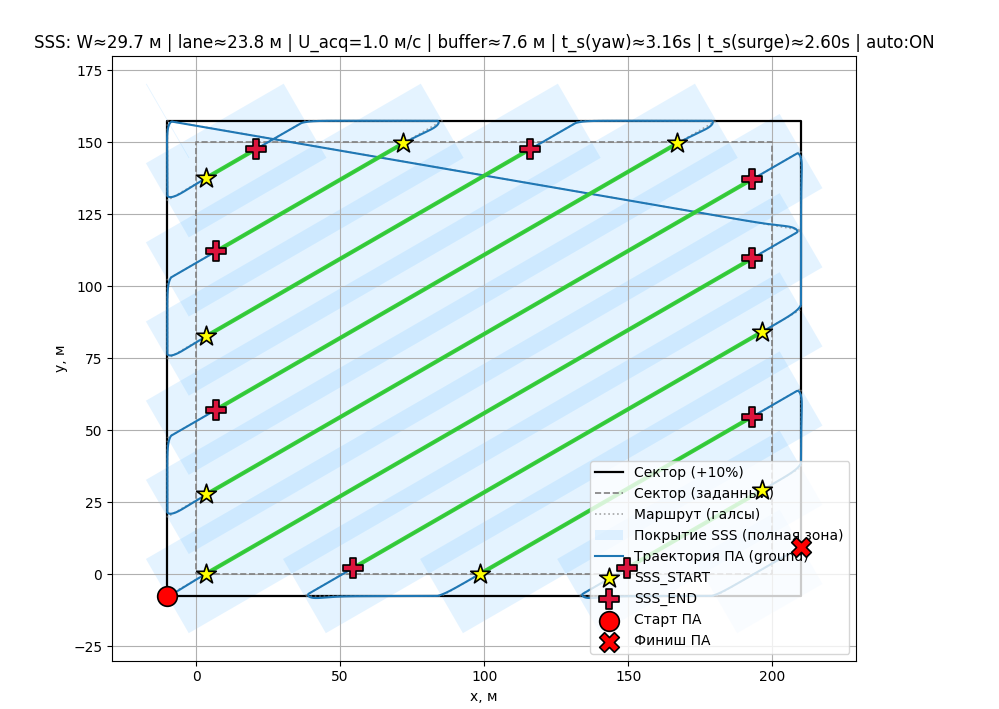


Рисунок 6.2 – графическое поле для вывода траектории

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. Для полного покрытия зоны в построении траектории был учтен запас в 10 процентов относительно ключевой области.

В данном графическом поле мы также можем посмотреть график изменения скорости, представленный на рисунке 6.3 и график изменения угла курса, предсатвленный на рисунке 6.4. На графическом поле также представлены графики переходных процесоов, которые были показаын ранее при нанализе контуров курса и маршевого движения.

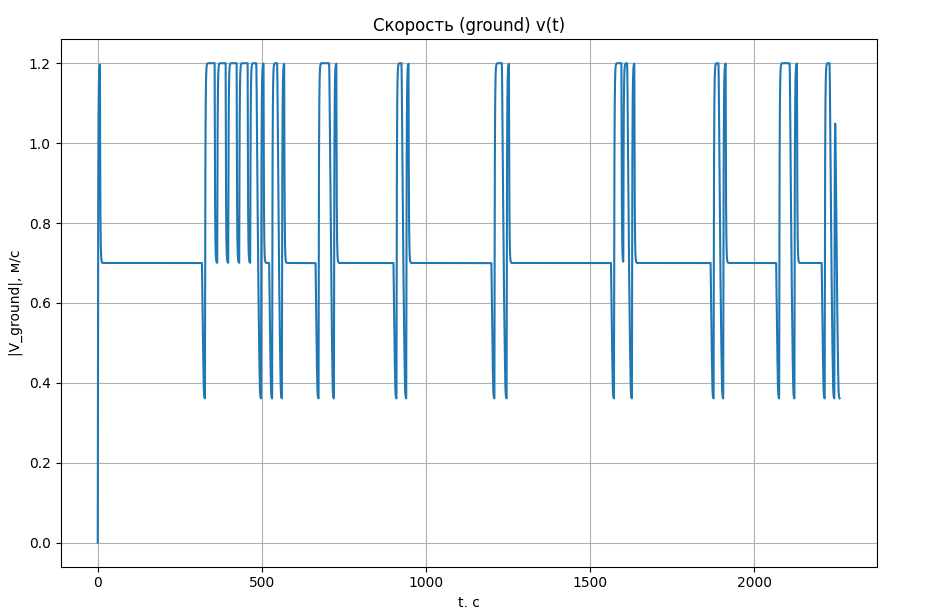


Рисунок 6.3 – график изменения скорости аппарата

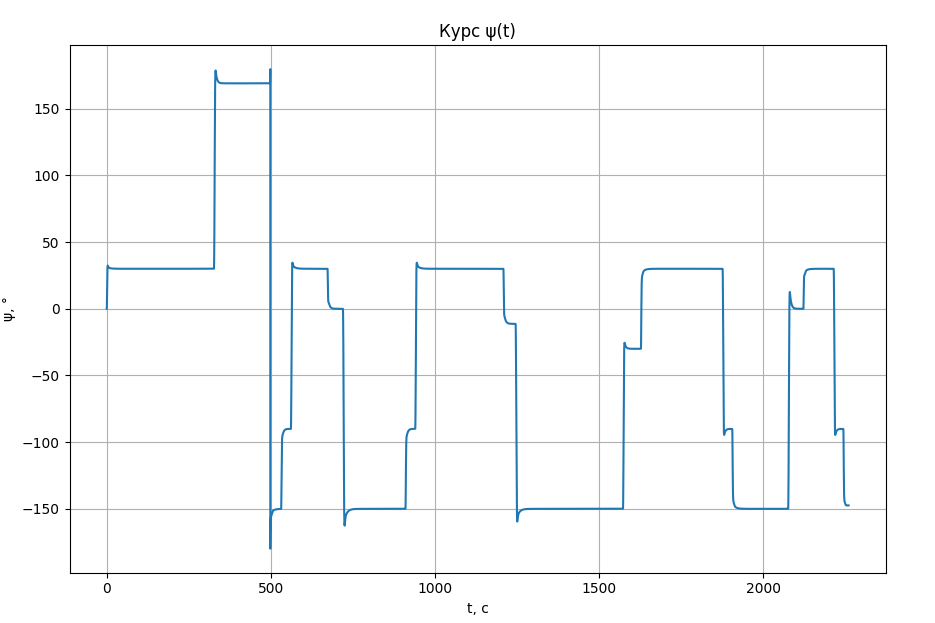


Рисунок 6.4 – график изменения угла курса аппарата

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

|  |
| --- |
| # -\*- coding: utf-8 -\*-  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  from PyQt6 import QtWidgets, QtGui, QtCore  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  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, omega\_max, 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  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\_sonar\_cmd(event: str, enable: bool, port: str, baudrate: int, timeout: float = 1.0) -> bool:  if not enable:  return True  cmd = bytes([0xFF])  if serial is None:  return True  try:  with serial.Serial(port=port, baudrate=baudrate, timeout=timeout) as ser:  ser.write(cmd)  ser.flush()  return True  except Exception:  return False  def draw\_cov\_rectangle(ax, S, E, width, color="#6EC1FF", alpha=0.15, label=None):  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))  class YawPD:  def \_\_init\_\_(self, K1\_kurs, K2\_kurs, use\_err\_psi, use\_err\_r, meas\_scale):  self.K1 = float(K1\_kurs)  self.K2 = float(K2\_kurs)  self.use\_err\_psi = bool(use\_err\_psi)  self.use\_err\_r = bool(use\_err\_r)  self.meas\_scale = float(meas\_scale)  def \_meas\_psi(self, psi):  return self.meas\_scale \* psi if self.use\_err\_psi else psi  def \_meas\_r(self, r):  return self.meas\_scale \* r if self.use\_err\_r else r  def command(self, psi\_ref\_deg, psi\_rad, r\_rad\_s) -> float:  psi\_m = self.\_meas\_psi(psi\_rad)  r\_m = self.\_meas\_r(r\_rad\_s)  e\_deg = psi\_ref\_deg - psi\_m \* RAD2DEG  while e\_deg <= -180.0:  e\_deg += 360.0  while e\_deg > 180.0:  e\_deg -= 360.0  return self.K1 \* e\_deg - self.K2 \* (r\_m \* RAD2DEG)  class SurgePDRef:  def \_\_init\_\_(self, Meff, C1, C2, K1\_marsh, K2\_marsh, wn, zeta, use\_err\_u, meas\_scale, dt):  self.Meff = float(Meff)  self.C1 = float(C1)  self.C2 = float(C2)  self.K1 = max(0.0, float(K1\_marsh))  self.K2 = max(0.0, float(K2\_marsh))  self.wn = max(1e-6, float(wn))  self.zeta = max(0.2, float(zeta))  self.use\_err\_u = bool(use\_err\_u)  self.meas\_scale = float(meas\_scale)  self.dt = float(dt)  self.ur = 0.0  self.urd = 0.0  def \_meas\_u(self, u):  return self.meas\_scale \* u if self.use\_err\_u else u  def \_update\_ref\_filter(self, u\_ref):  ydd = self.wn \* self.wn \* (u\_ref - self.ur) - 2.0 \* self.zeta \* self.wn \* self.urd  self.urd += self.dt \* ydd  self.ur += self.dt \* self.urd  return self.ur, self.urd  def command\_T(self, u\_ref\_body, u\_body, udot\_body) -> float:  uref, uref\_dot = self.\_update\_ref\_filter(u\_ref\_body)  u\_meas = self.\_meas\_u(u\_body)  e = uref - u\_meas  edot = uref\_dot - udot\_body  Tff = self.C2 \* u\_meas + self.C1 \* abs(u\_meas) \* u\_meas  return self.Meff \* (self.K1 \* e + self.K2 \* edot) + Tff  class AUV\_BFS\_DRC:  def \_\_init\_\_(self, p: Dict[str, Any]):  self.Jy = p["Jy"]  self.L55 = p["L55"]  self.Cwy1 = p["Cwy1"]  self.Cwy2 = p["Cwy2"]  self.Meff = p["m"] + p["lam11"]  self.Cx1 = p["Cxu1"]  self.Cx2 = p["Cxu2"]  self.dt = p["dt"]  self.Fcur\_yaw = p["Fcur\_yaw\_mag"]  self.th\_cur\_yaw = p["Fcur\_yaw\_dir\_deg"] \* DEG2RAD  self.l\_yaw\_arm = p["l\_yaw\_arm"]  self.Ucur\_mag = p["Ucur\_mag"]  self.Ucur\_dir = p["Ucur\_dir\_deg"] \* DEG2RAD  self.Mdist = p["Mdist"]  self.Kdx\_thr = p["Kdx\_thr"]  self.Tdx\_thr = p["Tdx\_thr"]  self.Umax\_thr = abs(p["Umax\_thr"])  self.mix\_surge = p["mix\_gain\_surge"]  self.mix\_yaw = p["mix\_gain\_yaw"]  self.yaw\_ctrl = YawPD(p["K1\_kurs"], p["K2\_kurs"], p["use\_5pct\_error\_psi"], p["use\_5pct\_error\_r"], p["meas\_scale"])  self.surge\_ctrl = SurgePDRef(self.Meff, self.Cx1, self.Cx2, p["K1\_marsh"], p["K2\_marsh"], p["wn\_ref"], p["zeta\_ref"], p["use\_5pct\_error\_u"], p["meas\_scale"], self.dt)  self.psi = 0.0  self.r = 0.0  self.u = 0.0  self.x = 0.0  self.y = 0.0  self.TL = 0.0  self.TR = 0.0  self.\_last\_M\_thr = 0.0  self.\_last\_M\_net = 0.0  def \_drag\_u(self, u):  return self.Cx1 \* abs(u) \* u + self.Cx2 \* u  def \_thruster\_dyn(self, T, U):  return T + self.dt \* ((self.Kdx\_thr \* U - T) / self.Tdx\_thr)  def \_mix\_and\_saturate(self, U\_surge, U\_yaw):  UL = self.mix\_surge \* U\_surge - self.mix\_yaw \* U\_yaw  UR = self.mix\_surge \* U\_surge + self.mix\_yaw \* U\_yaw  m = max(abs(UL), abs(UR), 1e-9)  if m > self.Umax\_thr:  scale = self.Umax\_thr / m  UL \*= scale  UR \*= scale  return UL, UR  def step(self, u\_ref\_ground, psi\_ref\_deg):  Uc\_par = self.Ucur\_mag \* math.cos(self.Ucur\_dir - self.psi)  u\_ref\_body = max(0.0, u\_ref\_ground - Uc\_par)  udot = (self.TL + self.TR - self.\_drag\_u(self.u)) / self.Meff  U\_yaw\_cmd = self.yaw\_ctrl.command(psi\_ref\_deg, self.psi, self.r)  Tcmd\_sum = self.surge\_ctrl.command\_T(u\_ref\_body, self.u, udot)  U\_surge\_cmd = Tcmd\_sum / (2.0 \* self.Kdx\_thr)  UL, UR = self.\_mix\_and\_saturate(U\_surge\_cmd, U\_yaw\_cmd)  self.TL = self.\_thruster\_dyn(self.TL, UL)  self.TR = self.\_thruster\_dyn(self.TR, UR)  M\_thr = self.l\_yaw\_arm \* (self.TR - self.TL)  M\_cur = self.l\_yaw\_arm \* self.Fcur\_yaw \* math.sin(self.th\_cur\_yaw - self.psi)  Jtot = self.Jy + self.L55  rdot = (M\_thr + self.Mdist + M\_cur - self.Cwy1 \* abs(self.r) \* self.r - self.Cwy2 \* self.r) / Jtot  self.r += self.dt \* rdot  self.psi = wrap\_rad(self.psi + self.dt \* self.r)  udot = (self.TL + self.TR - self.\_drag\_u(self.u)) / self.Meff  self.u += self.dt \* udot  Vx = self.u \* math.cos(self.psi) + self.Ucur\_mag \* math.cos(self.Ucur\_dir)  Vy = self.u \* math.sin(self.psi) + self.Ucur\_mag \* math.sin(self.Ucur\_dir)  self.x += self.dt \* Vx  self.y += self.dt \* Vy  self.\_last\_M\_thr = M\_thr  self.\_last\_M\_net = M\_thr + self.Mdist + M\_cur - self.Cwy1 \* abs(self.r) \* self.r - self.Cwy2 \* self.r  return UL, UR, M\_thr, self.\_last\_M\_net  @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]  uL\_hist: List[float]  uR\_hist: List[float]  TL\_hist: List[float]  TR\_hist: List[float]  Mthr\_hist: List[float]  Mnet\_hist: List[float]  meta: Dict[str, Any]  def \_estimate\_Ts(p: Dict[str, Any]) -> Tuple[float, float]:  dt = p["dt"]  test = AUV\_BFS\_DRC(p)  tloc = 0.0  Ts\_yaw = 8.0  reached = False  while tloc < 8.0:  tloc += dt  test.step(0.0, 1.0)  err = abs(1.0 - test.psi \* RAD2DEG)  if not reached and err <= 0.02:  reached = True  Ts\_yaw = tloc  if reached and err > 0.02:  reached = False  Ts\_yaw = 8.0  test = AUV\_BFS\_DRC(p)  tloc = 0.0  Ts\_surge = 12.0  reached = False  while tloc < 12.0:  tloc += dt  test.step(1.0, 0.0)  Vx = test.u \* math.cos(test.psi) + p["Ucur\_mag"] \* math.cos(p["Ucur\_dir\_deg"] \* DEG2RAD)  Vy = test.u \* math.sin(test.psi) + p["Ucur\_mag"] \* math.sin(p["Ucur\_dir\_deg"] \* DEG2RAD)  ug = math.hypot(Vx, Vy)  err = abs(1.0 - ug)  if not reached and err <= 0.02:  reached = True  Ts\_surge = tloc  if reached and err > 0.02:  reached = False  Ts\_surge = 12.0  return Ts\_yaw, Ts\_surge  def \_auto\_tune\_for\_2s(p: Dict[str, Any]) -> Dict[str, Any]:  if not p.get("auto\_tune\_2s", False):  return p  p = dict(p)  Ts\_yaw, Ts\_surge = \_estimate\_Ts(p)  if Ts\_yaw > 2.0:  scale\_yaw = min(4.0, max(0.5, Ts\_yaw / 2.0))  p["K1\_kurs"] \*= scale\_yaw  p["K2\_kurs"] \*= scale\_yaw  if Ts\_surge > 2.0:  scale\_surge = min(4.0, max(0.5, Ts\_surge / 2.0))  p["K1\_marsh"] \*= scale\_surge  p["K2\_marsh"] = p["K2\_marsh"] \* scale\_surge if "K2\_marsh" in p else p["K2\_marsh"]  p["K2\_marsh"] \*= scale\_surge  p["wn\_ref"] = min(4.0 \* p["wn\_ref"], p["wn\_ref"] \* scale\_surge)  return p  def \_inflate\_rect\_from\_center(rect: Tuple[float, float, float, float], frac: float) -> Tuple[float, float, float, float]:  x, y, w, h = rect  cx = x + w / 2.0  cy = y + h / 2.0  w2 = w \* (1.0 + frac)  h2 = h \* (1.0 + frac)  x2 = cx - w2 / 2.0  y2 = cy - h2 / 2.0  return (x2, y2, w2, h2)  def run\_simulation(p\_in: Dict[str, Any]) -> SimResult:  p = \_auto\_tune\_for\_2s(p\_in)  rect\_orig = (p["rect\_x"], p["rect\_y"], p["rect\_w"], p["rect\_h"])  rect = \_inflate\_rect\_from\_center(rect\_orig, 0.10)  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"]  omega\_max = p["b"] \* p["Umax\_yaw"]  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)  Ts\_yaw, Ts\_surge = \_estimate\_Ts(p)  turn\_buffer = 1.2 \* U\_cruise \* max(Ts\_yaw, Ts\_surge)  guard\_before = float(p.get("guard\_before\_turn\_m", 10.0))  guard\_after = float(p.get("guard\_after\_turn\_m", 8.0))  straight\_tol = float(p.get("straight\_angle\_tol\_deg", 8.0)) \* DEG2RAD  def build\_acq\_spans\_on\_straights(wps\_list, buf\_turn, guard\_before\_m, guard\_after\_m):  spans = []  for i in range(1, len(wps\_list) - 1, 2):  A, B = wps\_list[i], wps\_list[i + 1]  L = math.hypot(B[0] - A[0], B[1] - A[1])  if L <= (buf\_turn + guard\_after\_m) + (buf\_turn + guard\_before\_m):  continue  dx, dy = (B[0] - A[0]) / L, (B[1] - A[1]) / L  S = (A[0] + dx \* (buf\_turn + guard\_after\_m), A[1] + dy \* (buf\_turn + guard\_after\_m))  E = (B[0] - dx \* (buf\_turn + guard\_before\_m), B[1] - dy \* (buf\_turn + guard\_before\_m))  spans.append((S, E))  return spans  acq\_spans = build\_acq\_spans\_on\_straights(wps, turn\_buffer, guard\_before, guard\_after)  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  psi\_span = math.atan2(uy, ux)  spans\_ex.append({"S": S, "E": E, "u": (ux, uy), "L": L, "psi\_span": psi\_span, "start\_fired": False, "end\_fired": False})  auv = AUV\_BFS\_DRC(p)  x0, y0 = wps[0]  if len(wps) >= 2:  dx0, dy0 = wps[1][0] - wps[0][0], wps[1][1] - wps[0][1]  auv.psi = math.atan2(dy0, dx0)  auv.x, auv.y = x0, y0  xs, ys = [auv.x], [auv.y]  psi\_deg = [auv.psi \* RAD2DEG]  u\_ground = []  e\_ct\_hist = [0.0]  t\_hist = [0.0]  uL\_hist, uR\_hist, TL\_hist, TR\_hist = [], [], [], []  Mthr\_hist, Mnet\_hist = [], []  camera\_events: List[Tuple[str, Tuple[float, float], float]] = []  Ucx = p["Ucur\_mag"] \* math.cos(p["Ucur\_dir\_deg"] \* DEG2RAD)  Ucy = p["Ucur\_mag"] \* math.sin(p["Ucur\_dir\_deg"] \* DEG2RAD)  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  t = 0.0  i\_seg = 0  x\_prev, y\_prev = auv.x, auv.y  EVENT\_PERP\_TOL = float(p.get("event\_perp\_tol", 5.0))  EVENT\_POINT\_TOL = float(p.get("event\_point\_tol", 3.0))  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, wp\_j = wps[i\_seg], wps[i\_seg + 1]  psi\_ref, target\_pt, e\_ct = los\_pure\_pursuit((auv.x, auv.y), wp\_i, wp\_j, p["lookahead"])  xt, yt = target\_pt  d\_target = math.hypot(xt - auv.x, yt - auv.y)  capturing = False  for sp in spans\_ex:  Sx, Sy = sp["S"]  Ex, Ey = sp["E"]  ux, uy = sp["u"]  L = sp["L"]  psi\_span = sp["psi\_span"]  s\_prev = (x\_prev - Sx) \* ux + (y\_prev - Sy) \* uy  s = (auv.x - Sx) \* ux + (auv.y - Sy) \* uy  d\_perp = abs((auv.x - Sx) \* (-uy) + (auv.y - Sy) \* ux)  is\_straight = abs(wrap\_rad(auv.psi - psi\_span)) <= straight\_tol  dS = math.hypot(auv.x - Sx, auv.y - Sy)  dE = math.hypot(auv.x - Ex, auv.y - Ey)  if (not sp["start\_fired"]) and is\_straight and (((s\_prev < 0.0) and (s >= 0.0) and (d\_perp <= EVENT\_PERP\_TOL)) or (dS <= EVENT\_POINT\_TOL)):  \_ = send\_sonar\_cmd("SSS\_START", bool(p.get("sonar\_serial\_enable", False)), str(p.get("sonar\_port", "/dev/ttyUSB0")), int(p.get("sonar\_baud", 9600)))  camera\_events.append(("SSS\_START", (Sx, Sy), t))  sp["start\_fired"] = True  if (not sp["end\_fired"]) and is\_straight and (((s\_prev < L) and (s >= L) and (d\_perp <= EVENT\_PERP\_TOL)) or (dE <= EVENT\_POINT\_TOL)):  \_ = send\_sonar\_cmd("SSS\_END", bool(p.get("sonar\_serial\_enable", False)), str(p.get("sonar\_port", "/dev/ttyUSB0")), int(p.get("sonar\_baud", 9600)))  camera\_events.append(("SSS\_END", (Ex, Ey), t))  sp["end\_fired"] = True  if is\_straight and (0.0 <= s <= L) and (d\_perp <= EVENT\_PERP\_TOL):  capturing = True  dist\_to\_end = math.hypot(wp\_j[0] - auv.x, wp\_j[1] - auv.y)  if capturing:  u\_ref\_base = p["U\_acq"]  else:  u\_ref\_base = p["U\_cruise"] \* max(0.3, min(1.0, dist\_to\_end / (2.0 \* p["lookahead"])))  epsi = wrap\_rad(psi\_ref - auv.psi)  u\_ref\_turn = limit\_speed\_by\_turn(epsi, d\_target, omega\_max, a\_lat\_max, u\_ref\_base)  u\_ref\_ground = max(0.0, min(u\_ref\_base, u\_ref\_turn))  UL, UR, M\_thr, M\_net = auv.step(u\_ref\_ground, psi\_ref \* RAD2DEG)  Vx = auv.u \* math.cos(auv.psi) + Ucx  Vy = auv.u \* math.sin(auv.psi) + Ucy  ug = math.hypot(Vx, Vy)  x\_prev, y\_prev = auv.x, auv.y  t += dt  xs.append(auv.x)  ys.append(auv.y)  psi\_deg.append(auv.psi \* RAD2DEG)  u\_ground.append(ug)  e\_ct\_hist.append(e\_ct)  t\_hist.append(t)  uL\_hist.append(UL)  uR\_hist.append(UR)  TL\_hist.append(auv.TL)  TR\_hist.append(auv.TR)  Mthr\_hist.append(M\_thr)  Mnet\_hist.append(M\_net)  if math.hypot(wp\_j[0] - auv.x, wp\_j[1] - auv.y) < p["switch\_R"]:  i\_seg += 1  meta = dict(W\_use=W\_use, lane\_spacing=lane\_spacing, s\_ping=s\_ping, dt\_ping=dt\_ping, turn\_buffer=turn\_buffer, U\_acq=p["U\_acq"], U\_cruise=p["U\_cruise"], ts\_yaw=Ts\_yaw, ts\_surge=Ts\_surge, autotune=p.get("auto\_tune\_2s", False), rect\_orig=rect\_orig)  return SimResult(xs, ys, wps, rect, acq\_spans, camera\_events, t\_hist, psi\_deg, u\_ground, e\_ct\_hist, uL\_hist, uR\_hist, TL\_hist, TR\_hist, Mthr\_hist, Mnet\_hist, 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\_u\_body: List[float]  u\_body: List[float]  u\_ref\_body: List[float]  u\_body\_band: Tuple[float, float]  u\_body\_Mp\_pct: float  u\_body\_Ts: float | None  t\_x: List[float]  x\_coord: List[float]  x\_ref: List[float]  x\_band: Tuple[float, float]  x\_Mp\_pct: float  x\_Ts: float | None  def \_step\_metrics(t: List[float], y: List[float], ref: List[float], tol=0.05):  if not t:  return (0.0, 0.0), 0.0, None  yr = ref[-1]  A = max(abs(yr), 1e-9)  y\_low, y\_high = yr - tol \* A, 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\_in: Dict[str, Any]) -> StepResponses:  p = \_auto\_tune\_for\_2s(p\_in)  dt = p["dt"]  auv = AUV\_BFS\_DRC(p)  T = max(dt, float(p["yaw\_step\_T"]))  amp = float(p["yaw\_step\_amp\_deg"])  t\_yaw = [0.0]  yaw = [auv.psi \* RAD2DEG]  yref = [amp]  tcur = 0.0  while tcur < T:  tcur += dt  auv.step(0.0, amp)  t\_yaw.append(tcur)  yaw.append(auv.psi \* RAD2DEG)  yref.append(amp)  yaw\_band, yaw\_Mp, yaw\_Ts = \_step\_metrics(t\_yaw, yaw, yref, tol=0.05)  auv = AUV\_BFS\_DRC(p)  T = max(dt, float(p["surge\_step\_T"]))  amp\_ub = float(p.get("surge\_step\_amp\_body", 1.0))  t\_ub = [0.0]  u\_b = [auv.u]  u\_ref\_b = [amp\_ub]  tcur = 0.0  while tcur < T:  tcur += dt  Uc\_par = p["Ucur\_mag"] \* math.cos((p["Ucur\_dir\_deg"] \* DEG2RAD) - auv.psi)  u\_ref\_ground = max(0.0, amp\_ub + Uc\_par)  auv.step(u\_ref\_ground, 0.0)  u\_b.append(auv.u)  u\_ref\_b.append(amp\_ub)  t\_ub.append(tcur)  u\_b\_band, u\_b\_Mp, u\_b\_Ts = \_step\_metrics(t\_ub, u\_b, u\_ref\_b, tol=0.05)  Kp\_x = float(p.get("Kp\_x", 0.9))  Kd\_x = float(p.get("Kd\_x", 0.6))  u\_max\_pos = float(p.get("u\_max\_pos", 2.0))  x\_amp = float(p.get("x\_step\_amp", 1.0))  Tpos = float(p.get("x\_step\_T", 12.0))  auv = AUV\_BFS\_DRC(p)  t\_x = [0.0]  xs = [auv.x]  xref = [x\_amp]  tcur = 0.0  while tcur < Tpos:  tcur += dt  vx\_ground = auv.u \* math.cos(auv.psi) + p["Ucur\_mag"] \* math.cos(p["Ucur\_dir\_deg"] \* DEG2RAD)  ex = x\_amp - auv.x  edot = -vx\_ground  u\_cmd = Kp\_x \* ex + Kd\_x \* edot  u\_cmd = max(0.0, min(u\_max\_pos, u\_cmd))  auv.step(u\_cmd, 0.0)  xs.append(auv.x)  xref.append(x\_amp)  t\_x.append(tcur)  x\_band, x\_Mp, x\_Ts = \_step\_metrics(t\_x, xs, xref, tol=0.05)  return StepResponses(t\_yaw, yaw, yref, yaw\_band, yaw\_Mp, yaw\_Ts, t\_ub, u\_b, u\_ref\_b, u\_b\_band, u\_b\_Mp, u\_b\_Ts, t\_x, xs, xref, x\_band, x\_Mp, x\_Ts)  @dataclass  class ParamRow:  category: str  key: str  label: str  value: Any  ptype: str  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", 2.0, "float", "м/с", "Крейсерская наземная"),  ParamRow("Движение", "U\_acq", "U\_acq", 1.0, "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", "-", "Квадратное демпфирование"),  ParamRow("Yaw", "Cwy2", "Cwy2", 7.4, "float", "-", "Линейное демпфирование"),  ParamRow("Yaw", "b", "b", 0.3, "float", "-", "эвристика: ω\_max≈b·Umax\_yaw"),  ParamRow("Yaw", "Umax\_yaw", "Umax\_yaw", 10.0, "float", "В", "эвристика капа скорости"),  ParamRow("Yaw.ПД", "K1\_kurs", "K1\_kurs", 1.2, "float", "В/°", "P для курса"),  ParamRow("Yaw.ПД", "K2\_kurs", "K2\_kurs", 1.0, "float", "В/(°/с)", "D для курса"),  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("Yaw.Внешнее", "Mdist", "Mdist", 0.0, "float", "Н·м", "Пост. возмущающий момент"),  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", "-", "Квадратное сопротивление"),  ParamRow("Surge", "Cxu2", "Cxu2", 3.7, "float", "-", "Линейное сопротивление"),  ParamRow("Surge.ПД", "K1\_marsh", "K1\_marsh", 2.6, "float", "-", "P (марш)"),  ParamRow("Surge.ПД", "K2\_marsh", "K2\_marsh", 1.8, "float", "1/с", "D (марш)"),  ParamRow("Surge.ПД", "wn\_ref", "ω\_n", 1.8, "float", "рад/с", "Частота фильтра ссылки"),  ParamRow("Surge.ПД", "zeta\_ref", "ζ", 0.75, "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("ДРК", "Kdx\_thr", "Kdx\_thr", 20.0, "float", "Н/В", "Коэф. привода движителя"),  ParamRow("ДРК", "Tdx\_thr", "Tdx\_thr", 0.15, "float", "с", "Пост. времени движителя"),  ParamRow("ДРК", "Umax\_thr", "U\_thr,max", 10.0, "float", "В", "Сатурация входа движителя"),  ParamRow("ДРК", "mix\_gain\_surge", "mix\_surge", 1.0, "float", "-", "Коэф. общего канала"),  ParamRow("ДРК", "mix\_gain\_yaw", "mix\_yaw", 1.0, "float", "-", "Коэф. дифф. канала"),  ParamRow("Ошибки", "use\_5pct\_error\_psi", "err ψ +5%", False, "bool", "-", ""),  ParamRow("Ошибки", "use\_5pct\_error\_r", "err r +5%", False, "bool", "-", ""),  ParamRow("Ошибки", "use\_5pct\_error\_u", "err u +5%", False, "bool", "-", ""),  ParamRow("Ошибки", "meas\_scale", "meas\_scale", 1.05, "float", "-", ""),  ParamRow("Переходные", "yaw\_step\_amp\_deg", "Δψ\_step", 1.0, "float", "°", "Ступенька по курсу"),  ParamRow("Переходные", "yaw\_step\_T", "T\_step(yaw)", 8.0, "float", "с", "Длительность (yaw)"),  ParamRow("Переходные", "surge\_step\_amp\_body", "Δu\_body\_step", 1.0, "float", "м/с", "Ступенька по u(тело)"),  ParamRow("Переходные", "surge\_step\_T", "T\_step(surge)", 12.0, "float", "с", "Длительность (surge)"),  ParamRow("Позиция.ПД", "Kp\_x", "Kp\_x", 0.9, "float", "м⁻¹·(м/с)", "P по координате (генерирует u\_ref)"),  ParamRow("Позиция.ПД", "Kd\_x", "Kd\_x", 0.6, "float", "с⁻¹", "D по координате (через ẋ)"),  ParamRow("Позиция.ПД", "u\_max\_pos", "u\_max\_pos", 2.0, "float", "м/с", "Ограничение u\_ref в позиционном контуре"),  ParamRow("Позиция.ПД", "x\_step\_amp", "Δx\_step", 1.0, "float", "м", "Ступенька по X"),  ParamRow("Позиция.ПД", "x\_step\_T", "T\_step(x)", 12.0, "float", "с", "Длительность (x)"),  ParamRow("Автонастройка", "auto\_tune\_2s", "Автонастройка Ts≈2 c", True, "bool", "-", "Поджать Ts до ~2 c"),  ParamRow("SSS.Съёмка", "acq\_extend\_pct", "Удлинение рабочего отрезка", 0.10, "float", "дол.", "±доля длины (не используется при явных guard\*)"),  ParamRow("SSS.Съёмка", "guard\_before\_turn\_m", "Guard до поворота (END)", 12.0, "float", "м", "Завершить съёмку заранее"),  ParamRow("SSS.Съёмка", "guard\_after\_turn\_m", "Guard после поворота (START)", 8.0, "float", "м", "Начать съёмку после выхода"),  ParamRow("SSS.Съёмка", "straight\_angle\_tol\_deg", "Допуск прямолинейности, °", 8.0, "float", "°", "Макс. отклонение курса от оси галса"),  ParamRow("SSS.Съёмка", "event\_perp\_tol", "Допуск ⟂, м", 5.0, "float", "м", "Порог перпендикулярной близости"),  ParamRow("SSS.Съёмка", "event\_point\_tol", "Допуск точки, м", 3.0, "float", "м", "Порог близости к S/E"),  ParamRow("SSS.Съёмка", "sonar\_serial\_enable", "Отпр. в COM", False, "bool", "-", "Вкл/выкл отправку"),  ParamRow("SSS.Съёмка", "sonar\_port", "Порт SSS", "/dev/ttyUSB0", "str", "-", "Напр. COM3 (Windows)"),  ParamRow("SSS.Съёмка", "sonar\_baud", "Baud SSS", 9600, "int", "бод", "Скорость порта"),  ]  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.spec\_map: Dict[str, ParamRow] = {}  self.populate(rows)  self.resizeColumnsToContents()  self.horizontalHeader().setStretchLastSection(True)  def populate(self, rows: List[ParamRow]):  self.setRowCount(len(rows))  self.spec\_map = {r.key: r for r in rows}  for i, r in enumerate(rows):  cat = QtWidgets.QTableWidgetItem(r.category)  cat.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)  key = QtWidgets.QTableWidgetItem(r.label)  key.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)  if r.ptype == "bool":  val = QtWidgets.QTableWidgetItem()  val.setFlags(QtCore.Qt.ItemFlag.ItemIsUserCheckable | QtCore.Qt.ItemFlag.ItemIsEnabled | QtCore.Qt.ItemFlag.ItemIsSelectable)  val.setCheckState(QtCore.Qt.CheckState.Checked if bool(r.value) else QtCore.Qt.CheckState.Unchecked)  val.setText("True" if bool(r.value) else "False")  elif r.ptype == "int":  val = QtWidgets.QTableWidgetItem(str(r.value))  val.setFlags(QtCore.Qt.ItemFlag.ItemIsEditable | QtCore.Qt.ItemFlag.ItemIsEnabled | QtCore.Qt.ItemFlag.ItemIsSelectable)  val.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignRight | QtCore.Qt.AlignmentFlag.AlignVCenter)  elif r.ptype == "float":  val = QtWidgets.QTableWidgetItem(str(r.value))  val.setFlags(QtCore.Qt.ItemFlag.ItemIsEditable | QtCore.Qt.ItemFlag.ItemIsEnabled | QtCore.Qt.ItemFlag.ItemIsSelectable)  val.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignRight | QtCore.Qt.AlignmentFlag.AlignVCenter)  else:  val = QtWidgets.QTableWidgetItem(str(r.value))  val.setFlags(QtCore.Qt.ItemFlag.ItemIsEditable | QtCore.Qt.ItemFlag.ItemIsEnabled | QtCore.Qt.ItemFlag.ItemIsSelectable)  unit = QtWidgets.QTableWidgetItem(r.unit)  unit.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)  desc = QtWidgets.QTableWidgetItem(r.desc)  desc.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)  for c, it in enumerate((cat, key, val, unit, desc)):  self.setItem(i, c, it)  it.setData(QtCore.Qt.ItemDataRole.UserRole, 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 = self.spec\_map.get(key)  if spec is None:  continue  if spec.ptype == "bool":  params[key] = val\_item.checkState() == QtCore.Qt.CheckState.Checked  elif spec.ptype == "int":  txt = (val\_item.text() or "").strip()  try:  params[key] = int(float(txt))  except Exception:  val\_item.setBackground(QtGui.QBrush(QtGui.QColor("#ffcccc")))  raise ValueError(f"Неверное целое в параметре: {spec.label}")  elif spec.ptype == "float":  txt = (val\_item.text() or "").strip().replace(",", ".")  try:  params[key] = float(txt)  except Exception:  val\_item.setBackground(QtGui.QBrush(QtGui.QColor("#ffcccc")))  raise ValueError(f"Неверное число в параметре: {spec.label}")  else:  params[key] = (val\_item.text() or "").strip()  missing = [r.key for r in self.\_rows if r.key not in params]  if missing:  raise ValueError("В таблице отсутствуют значения: " + ", ".join(missing))  return params  def reset\_defaults(self):  self.populate(default\_param\_rows())  def load\_from\_dict(self, d: Dict[str, Any]):  legacy = {"K1": "K1\_kurs", "K2": "K2\_kurs", "Kp": "K1\_marsh", "Kd": "K2\_marsh", "Kp\_marsh": "K1\_marsh", "Kd\_marsh": "K2\_marsh"}  for old, new in legacy.items():  if old in d and new not in d:  d[new] = d[old]  for r in range(self.rowCount()):  key = self.item(r, 0).data(QtCore.Qt.ItemDataRole.UserRole)  if key not in d:  continue  spec = self.spec\_map.get(key)  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 — БФС ДРК, маршрут и переходные")  self.resize(1600, 950)  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, 7)  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\_u\_body = PlotTab("Переходная u(тело)")  self.tab\_step\_x = PlotTab("Переходная по координате X")  self.tab\_ul = PlotTab("Входы движителей U\_L/U\_R (В)")  self.tab\_tl = PlotTab("Тяга движителей T\_L/T\_R (Н)")  self.tab\_mom = PlotTab("Моменты (yaw): M\_thr и M\_net")  for tab, name in [  (self.tab\_plan, "План"),  (self.tab\_yaw, "Курс"),  (self.tab\_speed, "Скорость"),  (self.tab\_ect, "e\_ct"),  (self.tab\_step\_yaw, "Переходная (курс)"),  (self.tab\_step\_u\_body, "Переходная u(тело)"),  (self.tab\_step\_x, "Переходная по координате"),  (self.tab\_ul, "U\_L/U\_R"),  (self.tab\_tl, "T\_L/T\_R"),  (self.tab\_mom, "Моменты"),  ]:  self.tabs.addTab(tab, name)  btn\_run = QtWidgets.QPushButton("Запустить")  btn\_reset = QtWidgets.QPushButton("Сброс")  btn\_save = QtWidgets.QPushButton("Сохранить пресет…")  btn\_load = QtWidgets.QPushButton("Загрузить пресет…")  self.chk\_legend = QtWidgets.QCheckBox("Легенда")  self.chk\_legend.setChecked(True)  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(self.chk\_legend)  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.")  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\_u\_body, self.tab\_step\_x, self.tab\_ul, self.tab\_tl, self.tab\_mom):  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  ax.plot([xmin, xmin + W, xmin + W, xmin, xmin], [ymin, ymin, ymin + H, ymin + H, ymin], linestyle="-", color="k", linewidth=1.6, label=("Сектор (+10%)" if self.chk\_legend.isChecked() else None))  if "rect\_orig" in res.meta:  x0, y0, W0, H0 = res.meta["rect\_orig"]  ax.plot([x0, x0 + W0, x0 + W0, x0, x0], [y0, y0, y0 + H0, y0 + H0, y0], linestyle="--", color="gray", linewidth=1.2, label=("Сектор (заданный)" if self.chk\_legend.isChecked() else None))  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 SSS (полная зона)" if (self.chk\_legend.isChecked() and 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 = False  added\_end = False  for ev, P, \_tt in res.camera\_events:  if ev == "SSS\_START":  self.tab\_plan.ax.scatter(P[0], P[1], s=220, marker="\*", zorder=12, facecolors="yellow", edgecolors="black", linewidths=1.2, label=("SSS\_START" if self.chk\_legend.isChecked() and not added\_start else None))  added\_start = True  elif ev == "SSS\_END":  self.tab\_plan.ax.scatter(P[0], P[1], s=200, marker="P", zorder=12, facecolors="crimson", edgecolors="black", linewidths=1.2, label=("SSS\_END" if self.chk\_legend.isChecked() and not added\_end else None))  added\_end = True  ax.scatter(res.wps[0][0], res.wps[0][1], s=200, color="red", marker="o", edgecolors="black", linewidths=1.2, zorder=11, label=("Старт ПА" if self.chk\_legend.isChecked() else None))  ax.scatter(res.wps[-1][0], res.wps[-1][1], s=200, color="red", marker="X", edgecolors="black", linewidths=1.2, zorder=11, label=("Финиш ПА" if self.chk\_legend.isChecked() else None))  ax.set\_aspect("equal", adjustable="box")  ax.set\_xlabel("x, м")  ax.set\_ylabel("y, м")  autotune\_txt = " | auto:ON" if res.meta.get("autotune", False) else ""  ax.set\_title(f"SSS: W≈{res.meta['W\_use']:.1f} м | lane≈{res.meta['lane\_spacing']:.1f} м | U\_acq={res.meta['U\_acq']:.1f} м/с | buffer≈{res.meta['turn\_buffer']:.1f} м | t\_s(yaw)≈{res.meta['ts\_yaw']:.2f}s | t\_s(surge)≈{res.meta['ts\_surge']:.2f}s{autotune\_txt}")  if self.chk\_legend.isChecked():  ax.legend(loc="best")  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, label=f"Ts≈{steps.yaw\_Ts:.2f}s")  ax.set\_title("Переходная по курсу")  ax.set\_xlabel("t, c")  ax.set\_ylabel("ψ, °")  if self.chk\_legend.isChecked():  yr = steps.yaw\_ref[-1]  txt = f"Mp ≈ {steps.yaw\_Mp\_pct:.1f}% | ±5% [{steps.yaw\_band[0]:.2f}; {steps.yaw\_band[1]:.2f}] | y\*≈{yr:.2f}°"  ax.legend(loc="best", title=txt)  self.tab\_step\_yaw.canvas.draw()  ax = self.tab\_step\_u\_body.ax  ax.plot(steps.t\_u\_body, steps.u\_body, label="u(тело)")  ax.plot(steps.t\_u\_body, steps.u\_ref\_body, linestyle="--", label="u\_ref(body)")  ax.axhline(steps.u\_body\_band[0], linestyle="--", linewidth=1)  ax.axhline(steps.u\_body\_band[1], linestyle="--", linewidth=1)  if steps.u\_body\_Ts is not None:  ax.axvline(steps.u\_body\_Ts, linestyle=":", linewidth=1, label=f"Ts≈{steps.u\_body\_Ts:.2f}s")  ax.set\_title("Переходная по линейной скорости u (тело)")  ax.set\_xlabel("t, c")  ax.set\_ylabel("м/с")  if self.chk\_legend.isChecked():  yr = steps.u\_ref\_body[-1]  txt = f"Mp ≈ {steps.u\_body\_Mp\_pct:.1f}% | ±5% [{steps.u\_body\_band[0]:.2f}; {steps.u\_body\_band[1]:.2f}] | y\*≈{yr:.2f} м/с"  ax.legend(loc="best", title=txt)  self.tab\_step\_u\_body.canvas.draw()  ax = self.tab\_step\_x.ax  ax.plot(steps.t\_x, steps.x\_coord, label="x(t)")  ax.plot(steps.t\_x, steps.x\_ref, linestyle="--", label="x\_ref")  ax.axhline(steps.x\_band[0], linestyle="--", linewidth=1)  ax.axhline(steps.x\_band[1], linestyle="--", linewidth=1)  if steps.x\_Ts is not None:  ax.axvline(steps.x\_Ts, linestyle=":", linewidth=1, label=f"Ts≈{steps.x\_Ts:.2f}s")  ax.set\_title("Переходная по координате X (наружный ПД→u\_ref)")  ax.set\_xlabel("t, c")  ax.set\_ylabel("м")  if self.chk\_legend.isChecked():  yr = steps.x\_ref[-1]  txt = f"Mp ≈ {steps.x\_Mp\_pct:.1f}% | ±5% [{steps.x\_band[0]:.2f}; {steps.x\_band[1]:.2f}] | y\*≈{yr:.2f} м"  ax.legend(loc="best", title=txt)  self.tab\_step\_x.canvas.draw()  ax = self.tab\_ul.ax  ax.plot(res.t\_hist[1 : len(res.uL\_hist) + 1], res.uL\_hist, label="U\_L")  ax.plot(res.t\_hist[1 : len(res.uR\_hist) + 1], res.uR\_hist, label="U\_R")  ax.axhline(params["Umax\_thr"], linestyle=":", linewidth=1)  ax.axhline(-params["Umax\_thr"], linestyle=":", linewidth=1)  ax.set\_xlabel("t, c")  ax.set\_ylabel("В")  ax.set\_title("U\_L/U\_R (входы движителей)")  if self.chk\_legend.isChecked():  ax.legend(loc="best")  self.tab\_ul.canvas.draw()  ax = self.tab\_tl.ax  ax.plot(res.t\_hist[1 : len(res.TL\_hist) + 1], res.TL\_hist, label="T\_L")  ax.plot(res.t\_hist[1 : len(res.TR\_hist) + 1], res.TR\_hist, label="T\_R")  ax.set\_xlabel("t, c")  ax.set\_ylabel("Н")  ax.set\_title("T\_L/T\_R (тяга)")  if self.chk\_legend.isChecked():  ax.legend(loc="best")  self.tab\_tl.canvas.draw()  ax = self.tab\_mom.ax  t\_m = res.t\_hist[1 : len(res.Mthr\_hist) + 1]  ax.plot(t\_m, res.Mthr\_hist, label="M\_thr (движители)")  ax.plot(t\_m, res.Mnet\_hist, linestyle="--", label="M\_net (суммарный)")  ax.set\_xlabel("t, c")  ax.set\_ylabel("Н·м")  ax.set\_title("Моменты по курсу")  if self.chk\_legend.isChecked():  ax.legend(loc="best")  self.tab\_mom.canvas.draw()  starts = sum(1 for ev, \_, \_ in res.camera\_events if ev == "SSS\_START")  ends = sum(1 for ev, \_, \_ in res.camera\_events if ev == "SSS\_END")  total\_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))  self.statusBar().showMessage(f"L≈{total\_L:.1f} м; W≈{res.meta['W\_use']:.1f} м; lane≈{res.meta['lane\_spacing']:.1f} м; s\_ping≈{res.meta['s\_ping']:.2f} м; dt\_ping≈{res.meta['dt\_ping']:.2f} c; SSS\_START={starts}, SSS\_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() |