Министерство науки и высшего образования Российской Федерации

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет Систем Управления и Робототехники**

**Направление подготовки:**

*15.04.06 Мехатроника и робототехника*

**КУРСОВАЯ РАБОТА**

по дисциплине: Моделирование и управление движением роботов

**по теме:** *Планирование* *траектории квадрокоптера методом случайного дерева и управление с помощью контроллера, использующего метод бэкстеппинга*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнили студенты | | Веснин М.А. | | |  |
|  |  | | Миргазов Э.Р. | |  |
|  |  | | Топольницкий А.А. | |  |
| Преподаватель | | Колюбин С.А. | | |  |
|  | |  | | |  |
|  | | Подпись преподавателя | | Дата |  |
| Защита | |  | |  |  |

Санкт-Петербург

2023 г.

Оглавление

[Введение 3](#_Toc130735859)

[Глава 1. Кинематический анализ и динамическая модель квадрокоптера 4](#_Toc130735860)

[Глава 1.1. Кинематический анализ квадрокоптера 4](#_Toc130735861)

[Глава 1.2. Динамическая модель квадрокоптера 5](#_Toc130735862)

[Глава 2. Система управление квадрокоптером 8](#_Toc130735863)

[Глава 3. Планирование траектории 11](#_Toc130735864)

[Глава 4. Результаты численного моделирования 12](#_Toc130735865)

[Заключение 20](#_Toc130735866)

[Список литературы 21](#_Toc130735867)

[Приложение 22](#_Toc130735868)

# Введение

В современном мире квадрокоптеры находят всё бо́льшее применение, поскольку могут выполнять целый ряд задач: профилактика и ликвидация ЧС, обеспечение обороны и национальной безопасности объектов промышленности, сельского хозяйства и продовольствия, природных ресурсов, а также служб мониторинга, длительного авиационного патрулирования земной и водной поверхностей. Помимо этого, данный тип роботов можно применять для задач поиска пропавших людей или в задачах противодействия преступникам [1].

Для успешного выполнения описанных выше задач необходимо три элемента: сам квадрокоптер, траектория, по которой должен двигаться робот, и система управления, обеспечивающая движение по спланированной траектории с высокой точностью. Существует много различных систем управления для квадрокоптеров – как линейные системы, так и нелинейные. Классическим примером является ПД-регулятор, однако это не единственный подход, как можно реализовать управление.

Целью данной работы является планирование траектории квадрокоптера с помощью метода случайного дерева и синтез системы управления для поддержания движения по заданной траектории. Для выполнения поставленной цели необходимо решить несколько задач:

1. Выполнить кинематический анализ квадрокоптера;
2. Построить динамическую модель данного робота;
3. Реализовать планирование траектории движения;
4. Синтезировать регулятор для управления движением робота;
5. Провести численное моделирование системы;
6. Сравнить результаты применяемой системы управления с классическим ПД-регулятором.

# Глава 1. Кинематический анализ и динамическая модель квадрокоптера

## **Глава 1.1. Кинематический анализ квадрокоптера**

Поведение квадрокоптера можно описать тремя углами:

1. – угол крена – угол вокруг оси Ох в инерциальной СК;
2. – угол тангажа – угол вокруг оси Оу в инерциальной СК;
3. – угол рыскания – угол вокруг оси Оz в инерциальной СК;

В работе планируется использовать квадрокоптер из библиотеки для MATLAB под названием Robotics Toolbox за авторством Питера Корка, поэтому возьмём конфигурационную схему возьмём из книги автора [2]. Она представлена ниже. Можно заметить, что ось Oz имеет направление внизу, пропеллеры 1 и 3 вращаются против часовой стрелки, 2 и 4 по часовой.

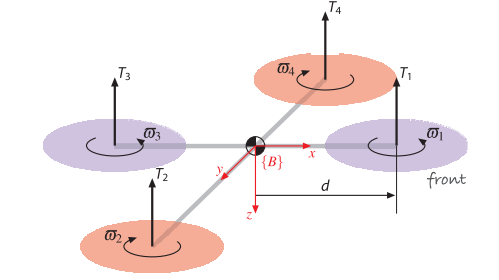


Рисунок . Конфигурация рассматриваемого квадрокоптера

Существует несколько способов получения матрицы поворота, необходимой для связи инерциальной системы координат и системы координат квадрокоптера. Чтобы получить в нашем случае подобную матрицу, необходимо сначала ввести три матрицы вращения вокруг осей:

Таким образом, результирующая матрица вращения представляет собой преобразование с использованием углов Эйлера инерциальной системы координат в систему координат тела:

Если перемножить матрицы , то получим переход в систему координат тела. Если применить операцию транспонирования к матрице, то будет получена итоговая матрица:

Угловые скорости от углов Эйлера связаны с системой координат тела следующим выражением [3]:

## **Глава 1.2. Динамическая модель квадрокоптера**

Для описания динамики квадрокоптера приняты следующие допущения [4],[5]:

1. Квадрокоптер представляет собой твёрдое тело с симметричной структурой;
2. Пропеллеры являются твёрдыми телами;
3. Центр масс совпадает с центом давления;
4. Сопротивлением воздуха пренебрегается.

С помощью уравнений Ньютона – Эйлера можно записать динамическую модель квадрокоптера в общем виде:

где – суммарный момент, описанный ниже; – матрица инерции 3 х 3, – вектор угловой скорости.

Рассмотрим моменты. Каждым пропеллером создаётся подъёмная сила, выражаемая формулой:

где – константа, зависящая от плотности воздуха, куба радиуса лезвия пропеллера, количества пропеллеров и длины хорды лезвия; – скорость вращения пропеллера. – суммарная тяга.

Далее, для момента по каждой оси можно получить следующие формулы. Знаки в формулах объясняются направлениями вращения пропеллеров, схема приведена была выше:

где – расстояние от центра масс до оси вращения пропеллера, – коэффициент, зависящий от тех же факторов, что и Можно заметить, что управление по рысканию можно осуществлять, управляя скоростями вращения пропеллеров.

Теперь вернёмся к уравнениям (7) и (8) и распишем их чуточку подробнее.

где – моменты, создаваемые пропеллерами по осям, по сути, это сумма . Для учёта результирующего гироскопического эффекта используется формула:

где , – момент инерции ротора двигателя.

Для расчёта центростремительной силы применяется формула:

Если в уравнение (8) переписать с учётом (14) и (15), то вращательная составляющая динамики будет выглядеть следующим образом:

Наконец, объединяя линейные и вращательные составляющие в одну систему, получим (20):

# Глава 2. Система управление квадрокоптером

Для начала введём обозначения, которыми удобно в дальнейшем пользоваться:

Тогда система (16) будет иметь вид:

где

**Глава 2.1. Метод бэкстеппинга для управления углами**

На примере уравнений для крена рассмотрим порядок применения ***Бэкстеппинга.*** Необходимы два уравнения:

Данный метод можно удобно разложить на два последующих шага. **Шаг первый** – определяем ошибку, относящуюся к и его производной:

Возьмём функцию Ляпунова и её производную следующего вида:

Чтобы удовлетворять условию устойчивости функции Ляпунова (вводится новое виртуальное управление в соответствии с формулой ниже:

где – положительная константа (коэффициент). Тогда производная функции Ляпунова имеет следующий вид:

Во **втором шаге** вводим новую функцию Ляпунова следующего вида:

Ниже приведён расчёт производной для этой функции Ляпунова:

Устойчивость всей системы, системы крена, обеспечивается с помощью второй положительной константы

И тогда производная функции Ляпунова имеет вид:

что говорит о том, что система асимптотически устойчива.

Используя тот же подход для управления по тангажу и рысканию, можно получить итоговую систему уравнений, обеспечивающую управление по углам Эйлера:

При этом:

где соответственно ошибка и положительные константы функций Ляпунова. Коэффициенты необходимо настраивать.

**Глава 2.2. Способ получения значений углов**

Поскольку в Robotics Toolbox существует модель с квадрокоптером, можем некоторые моменты взять оттуда. Например, управление тягой осуществляется с помощью ПД регулятора по формуле:

Контроллер, отвечающий за поведение робота, связывает ошибку в координатах *x* и *у* в системе координат тела с желаемыми углами Эйлера в инерциальной системе координат.

где

# Глава 3. Планирование траектории

Как уже упоминалось, планирование является важной задачей при работе с робототехническими системами. Существует много разных способ, рассмотрим некоторые из них [6].

*Метод декомпозиции на ячейки*. Бывает точным или приблизительным, рассмотрим на примере точной декомпозиции:

1. Всё свободное пространство разделяется на треугольные или трапецеидальные ячейки;
2. Составляются графы связности с вершинами в их центрах и ребрами, соответствующими общим сторонам смежных ячеек;
3. В итоге получаем два типа точек:
   1. тип 1, белые, соответствуют свободному пространству;
   2. тип 2, чёрные, соответствуют запрещённой зоне пространства;
4. Далее определяются вершины графа, которым соответствуют начальное и конечное положения, затем начинается поиск последовательности переходов по белым ячейкам.

*Метод вероятностной дорожной карты* используется для быстрой генерации пути и основан использовании случайных выборок в пространстве, состоит из четырёх шагов:

1. Осуществляется выбор нескольких узлов случайным образом;
2. Соседние узлы соединяются между собой отрезками, не пересекающими запрещённую зону, при заданной норме в этом пространстве;
3. Первые два шага повторяются, чтобы покрыть достаточно большую область, охватывающую начальное и конечное положение;
4. Выбирается последовательность отрезков, позволяющая выполнить траекторию.

Ещё одним алгоритмом планирования пути, как раз используемым в нашей работе, является алгоритм *двунаправленного быстроисследующего случайного дерева* [7]. Алгоритм работает следующим образом:

1. Задаются начальное и конечное положения робота, в нашем случае квадрокоптера. Начальное положение является первой вершиной дерева;
2. Затем в пространство добавляется случайная точка, на расстоянии, равном заданному шагу алгоритма;
3. Далее происходит проверка – не попала ли точка на наше препятствие. Если проверка пройдена, то вершина и линия между ней и предыдущей точкой добавляются к общему дереву;
4. Если новая случайная и конечная точки совпали, то построении траектории выполнено. В противном случае действия повторяются.

Для того, чтобы траектория, полученная методом случайного дерева, была более гладкой, с помощью функции *optimizePath* была проведена оптимизация пути. Данная функция работает на основе алгоритма Левенберга-Марквардта, о котором можно прочитать здесь [8], и эта функция позволяет задать желаемое расстояние, на котором объект должен держаться от разного рода препятствий.

# Глава 4. Результаты численного моделирования

На рисунке ниже приведена построенная карта с препятствиями и маршрутом. Чёрным помечены преграды, красным – спланированный маршрут, полученный с помощью метода быстроисследующего случайного дерева. Синим – ветви дерева. Фиолетовым – оптимизированный вариант траектории. Начальное положение (40,2), конечное – (90,90). Как видно по рисунку, неоптимизированный маршрут хоть и создаёт траекторию от начала до конца, но в некоторых точках проходит вплотную к преградам, что в реальности недопустимо. Оптимизированный маршрут же старается поддерживать расстояние до препятствий не меньше 1 метра.

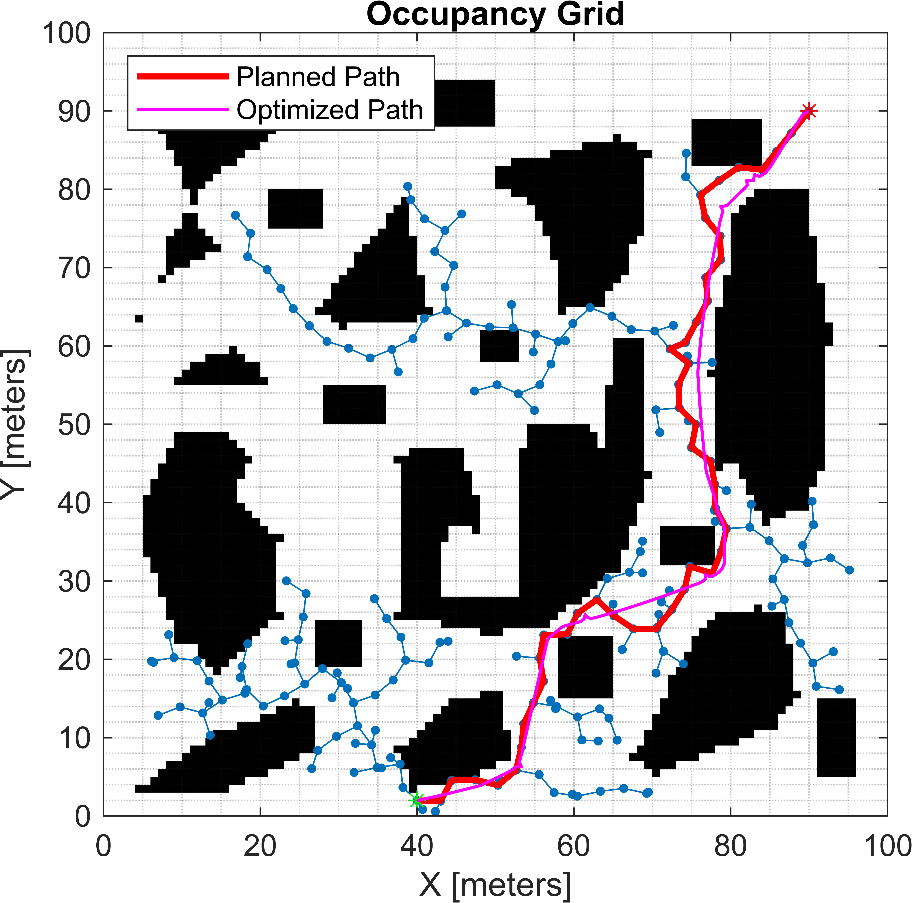


Рисунок . Спланированная траектория методом быстроисследуемого случайного дерева

Далее, на основе планирования алгоритма траектории были получены параметры этой траектории – координаты х и у. В дальнейшей работе предполагались следующие параметры:

Таблица . Параметры моделирования

|  |  |
| --- | --- |
| Высота полёта | 5 метров |
| Угол рыскания | Поддерживается 0 радиан |
| Масса квадрокоптера | 4 кг |
| Момент инерции ротора двигателя | 6,49е-5 кг \* м2 |

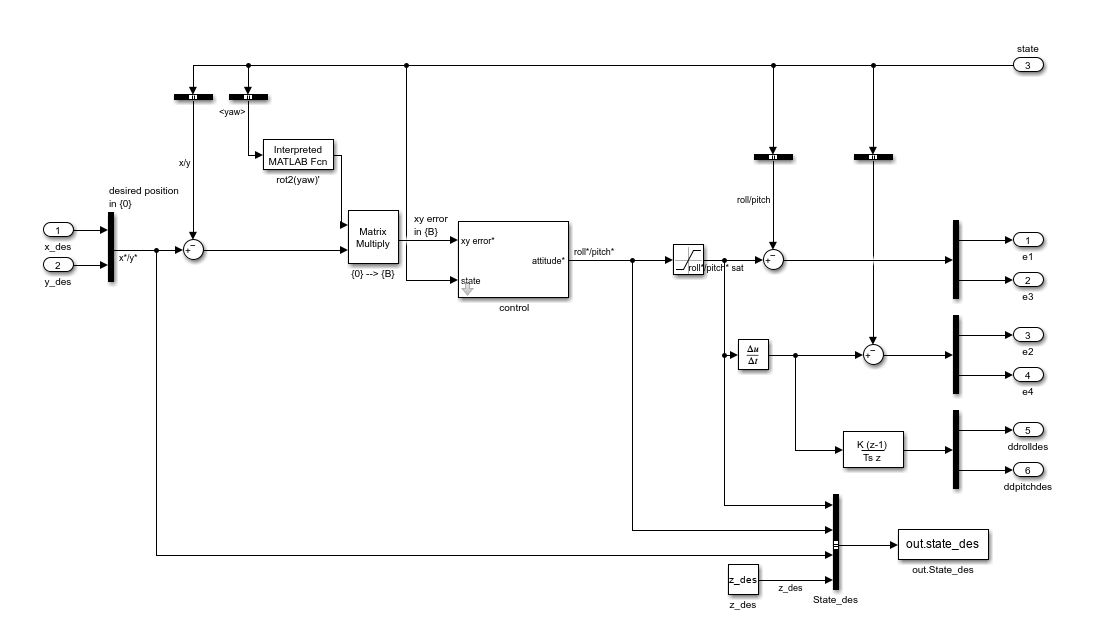


Рисунок . Часть внутренней системы модели в Simulink

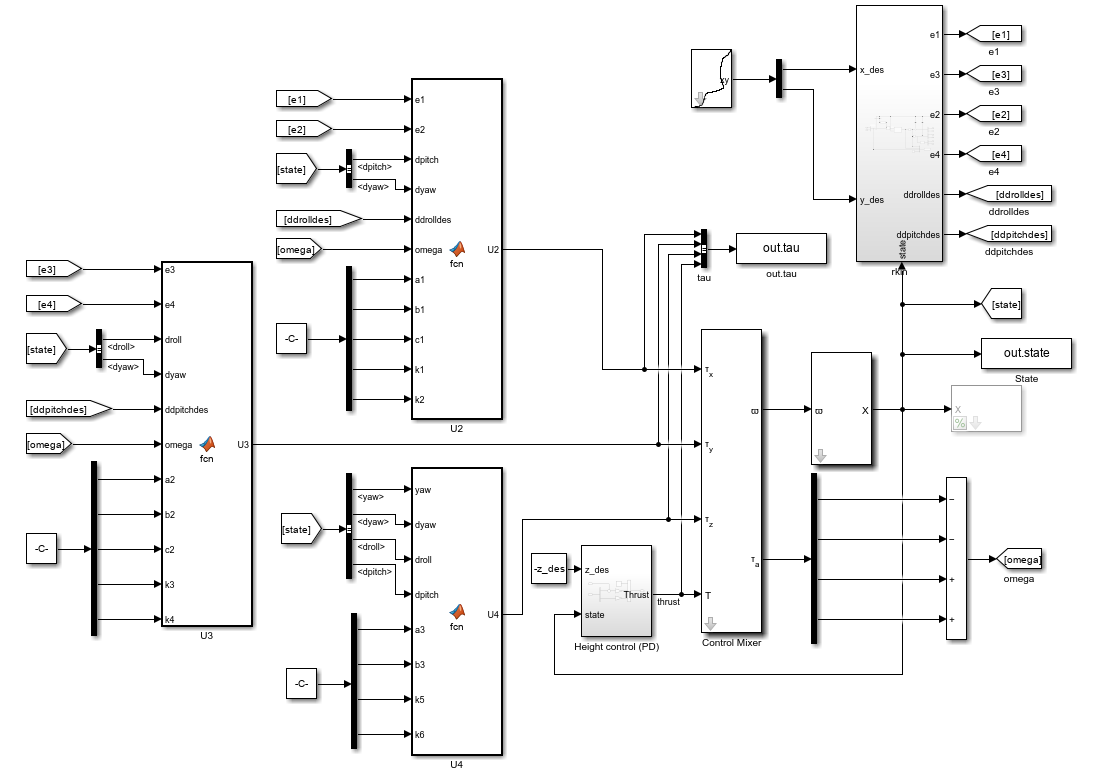


Рисунок . Внешняя часть системы управления в Simulink

Для того, чтобы оценить предлагаемый алгоритм, ниже приведены графики, отражающие реальное поведение квадрокоптера в сравнении с желаемым поведением.

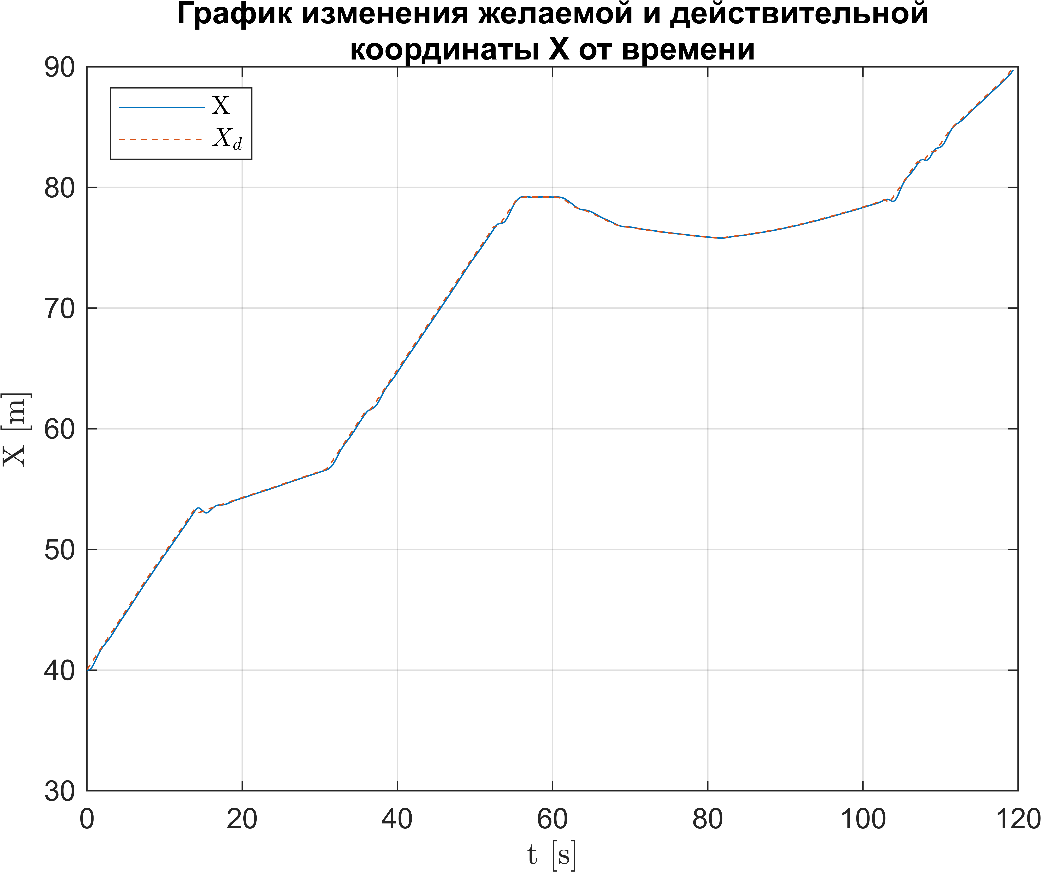


Рисунок . График изменения координаты Х

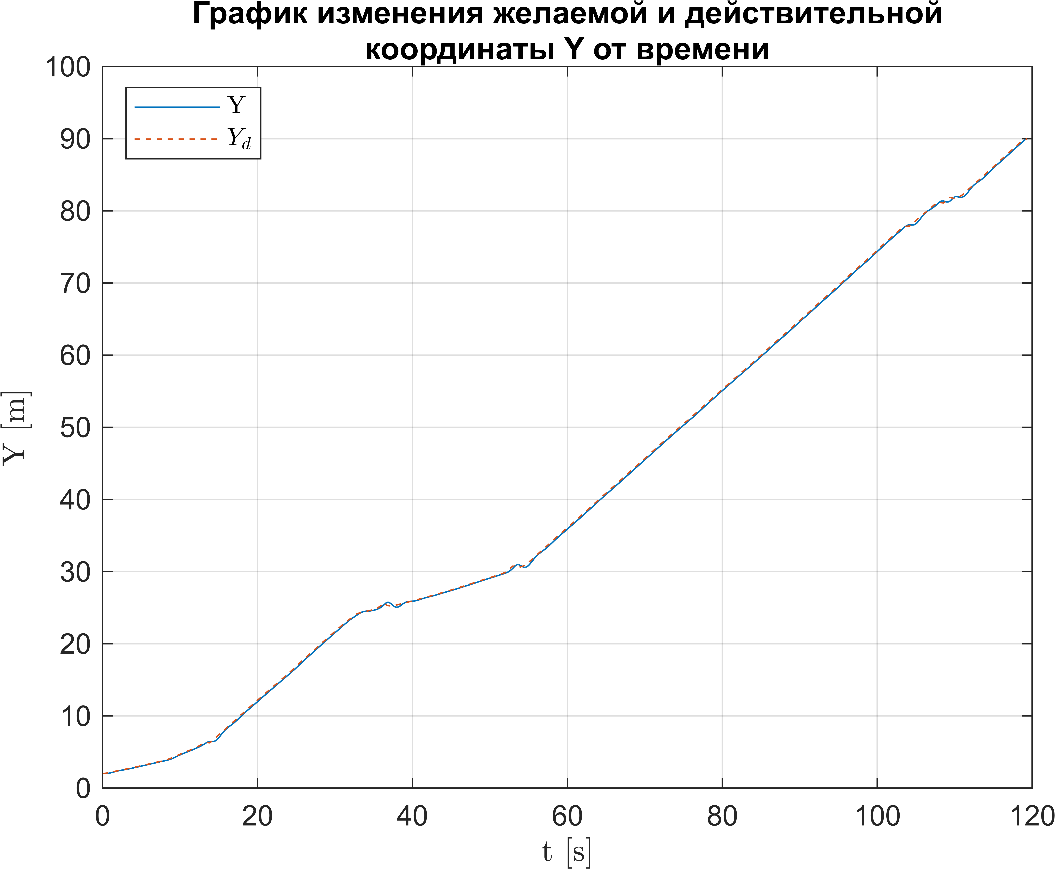


Рисунок . График изменения координаты Y

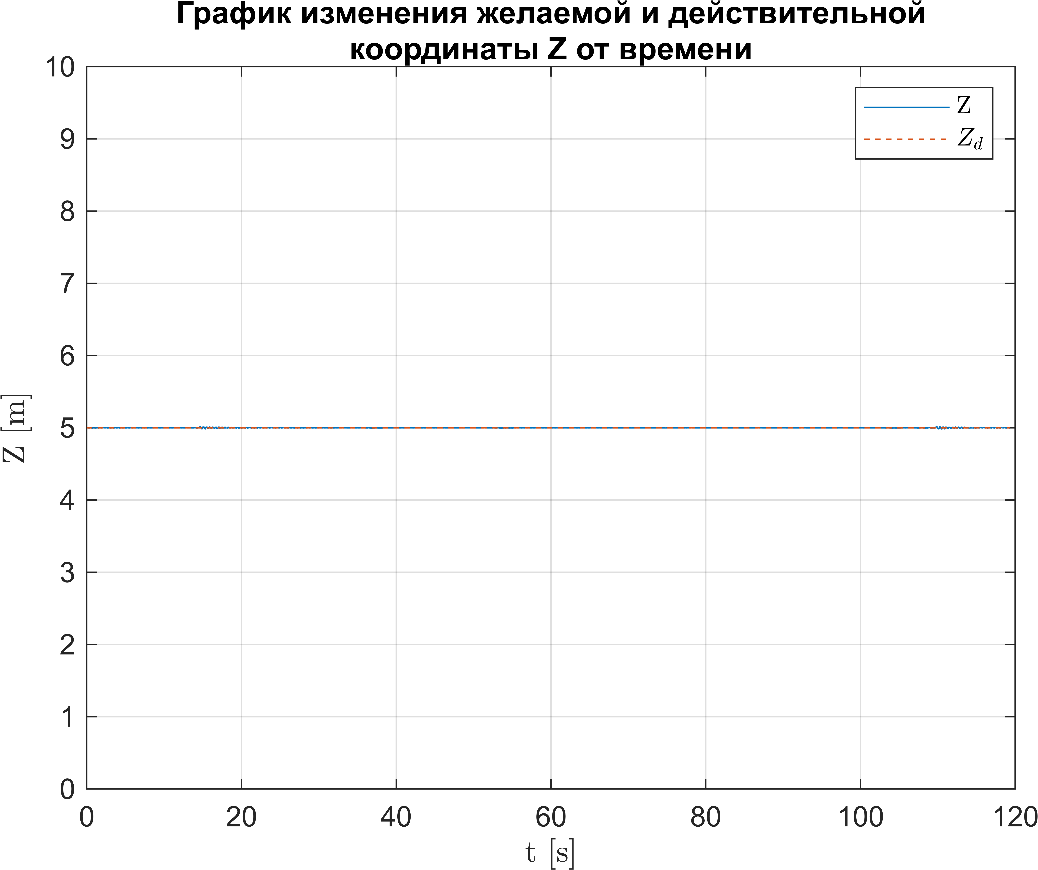


Рисунок . График изменения координаты Z

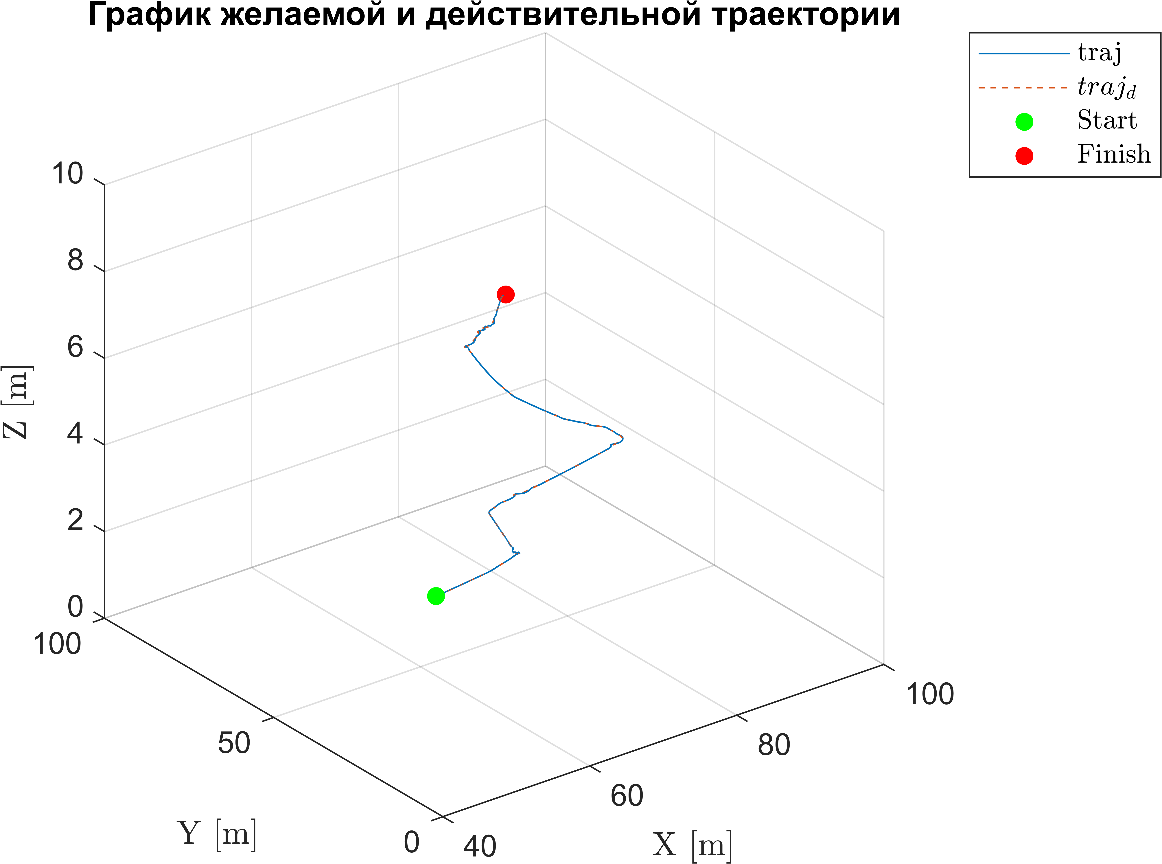


Рисунок . Трёхмерный вид траектории

На всех графиках видно, что дрон следует по заданной траектории с крайне небольшими отклонениями, что говорит о качественной настройке алгоритма бэкстеппинга.

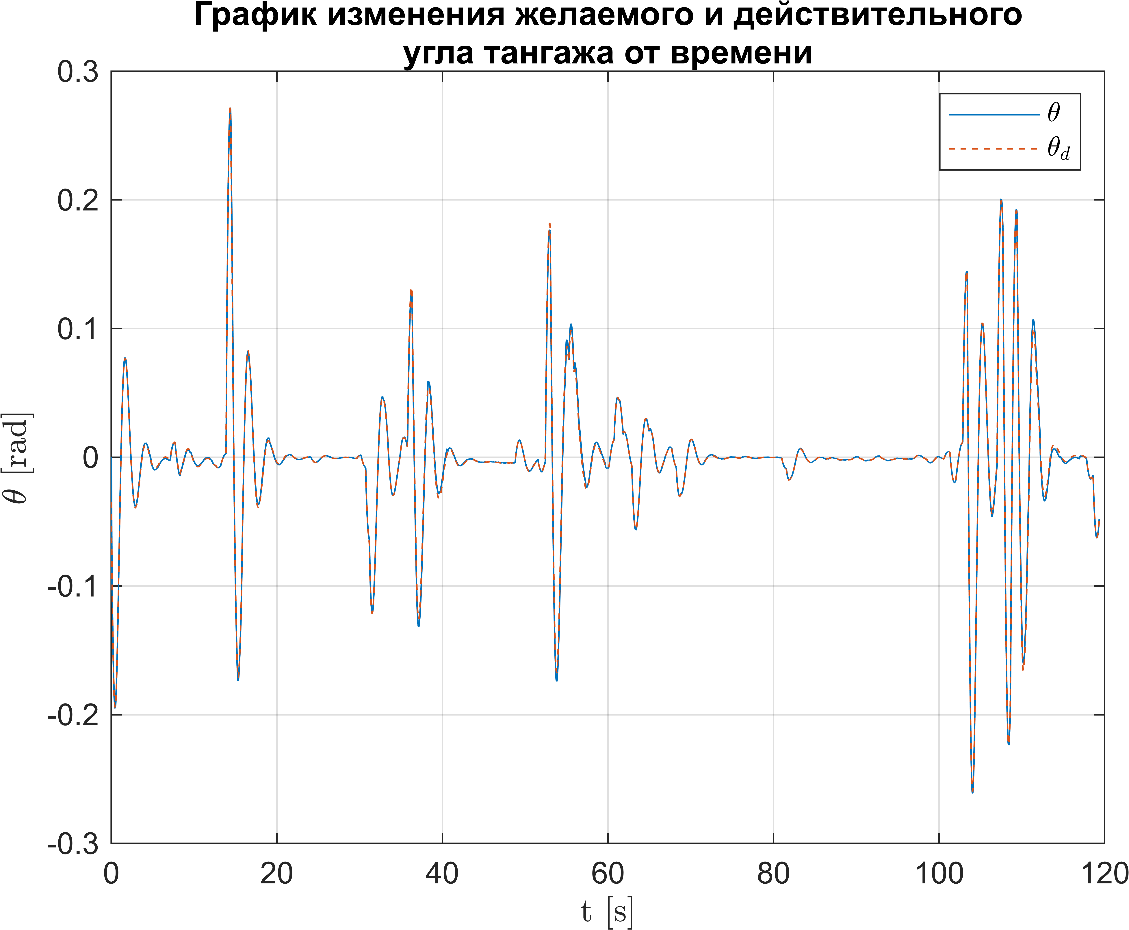


Рисунок . Изменение угла тангажа

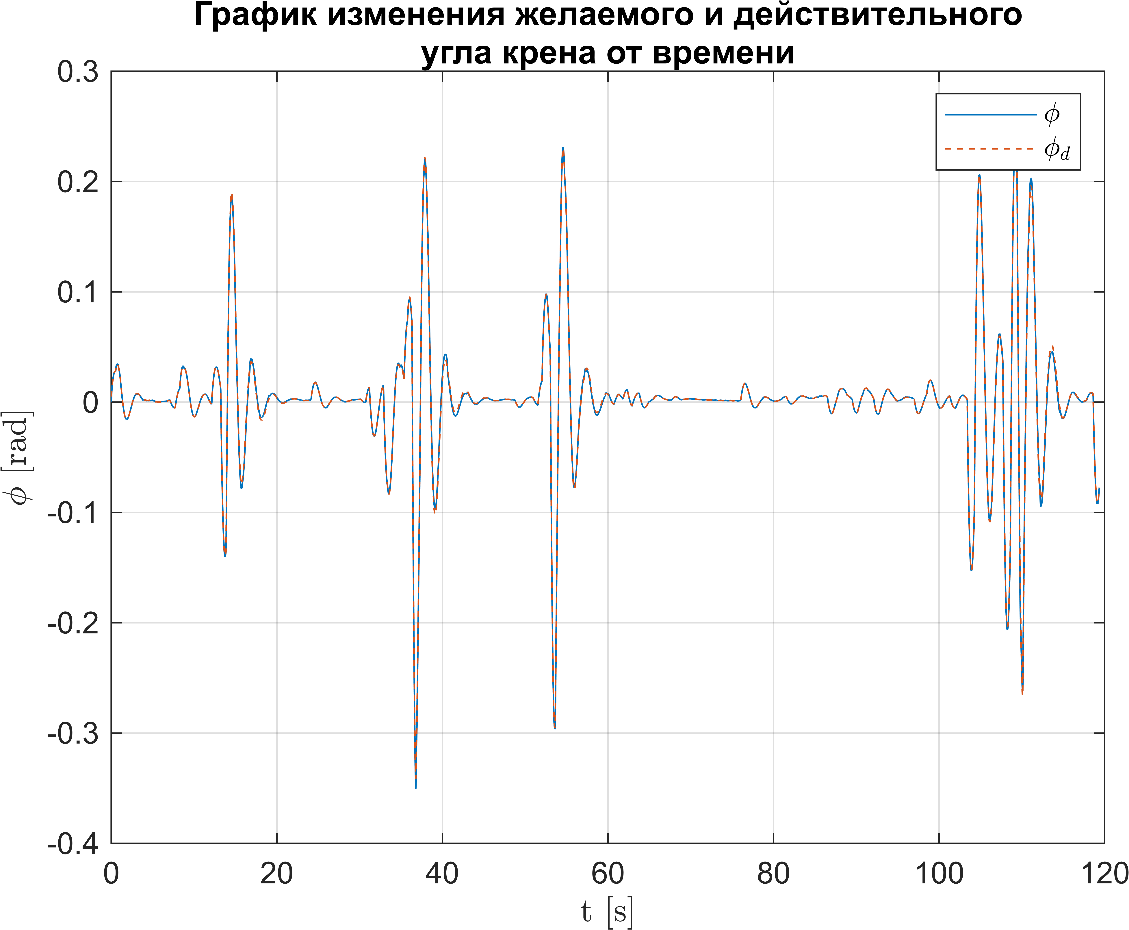


Рисунок . Изменение угла крена

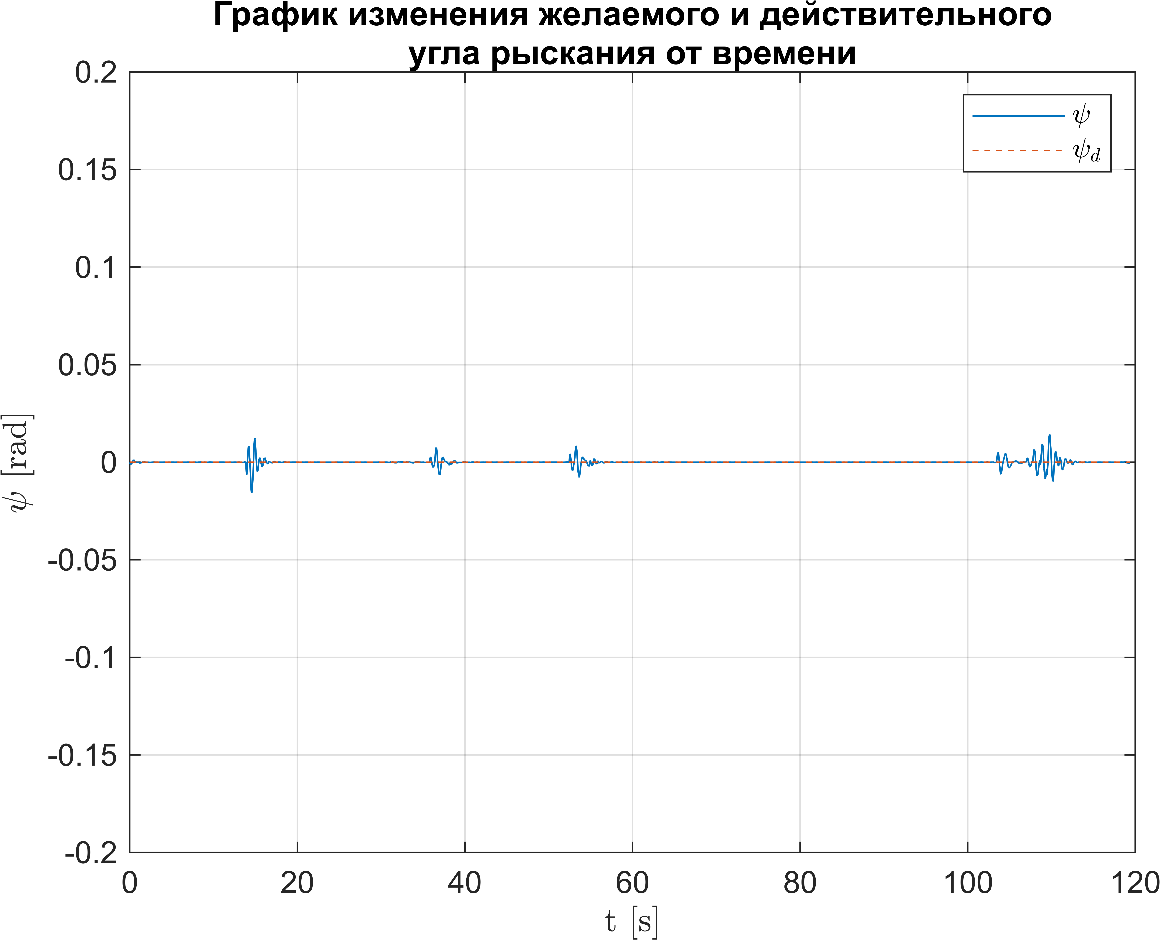


Рисунок . Изменение угла рыскания

На графиках изменения углов видно, что и здесь предлагаемый алгоритм справляется крайне хорошо. Однако, необходимо провести сравнительный анализ с другим алгоритмом. Например, возьмём ПД-регулятор. Он был также встроен в исходную модель с квадрокоптером из Robotic Toolbox. Для сравнения на одном графике были одновременно выведены суммарные ошибки по координатам и по углам для алгоритма бэкстеппинга и для ПД-регулятора.

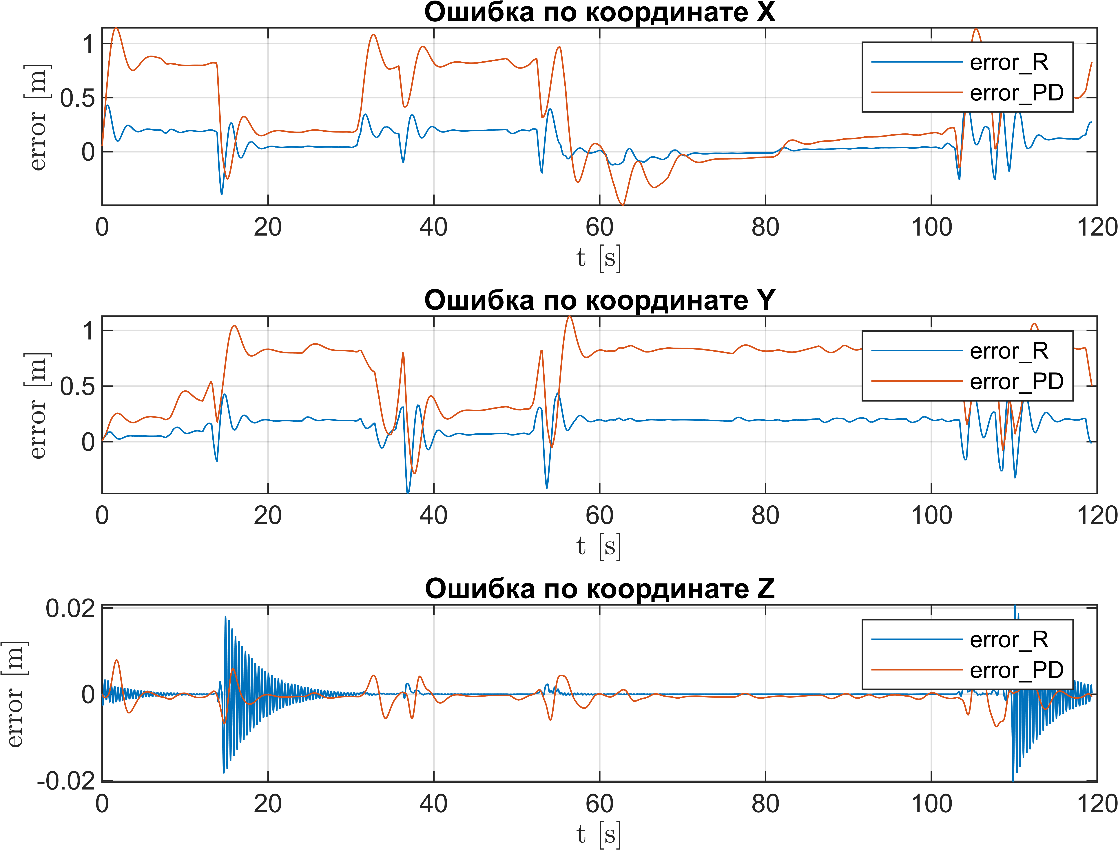


Рисунок . Сравнение ошибок по координатам для алгоритма бэкстеппинга и для изначальной системы управления с ПД-регулятором

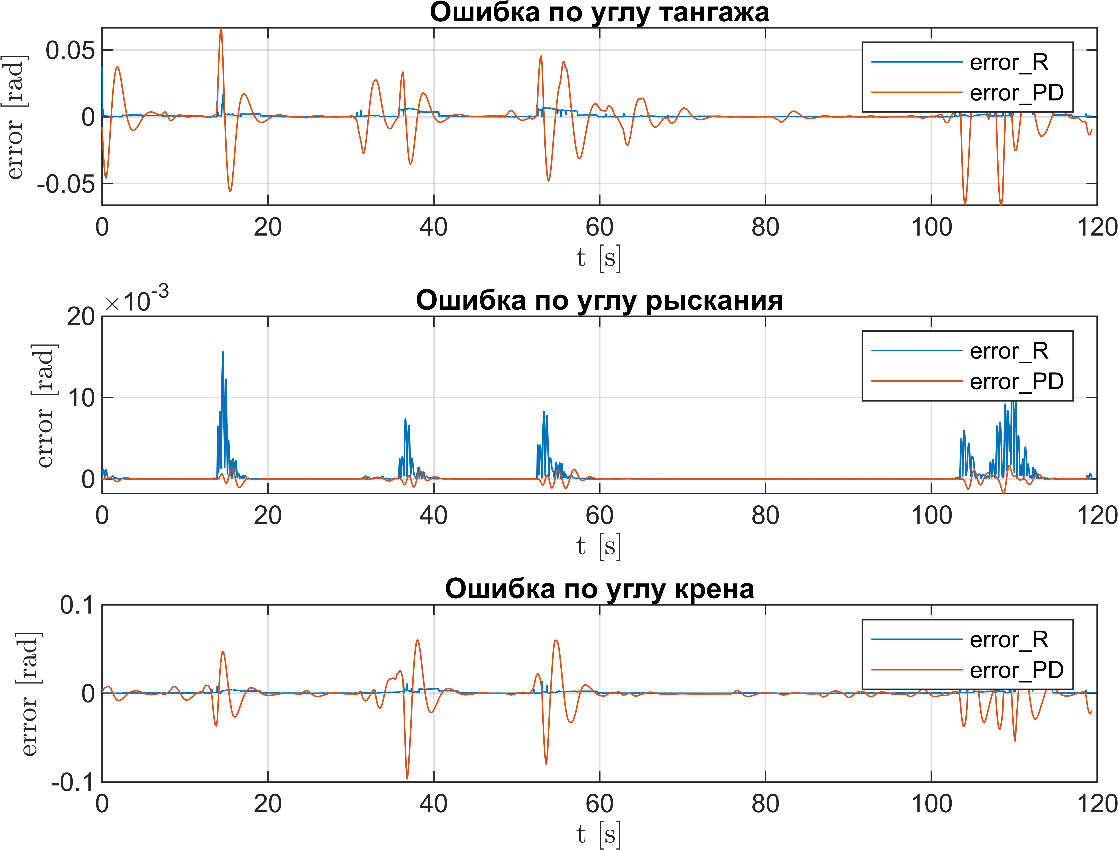


Рисунок . Сравнение ошибок по углам Эйлера для алгоритма бэкстеппинга и для изначальной системы управления с ПД-регулятором

На графиках ошибки для координат и углов Эйлера можно наблюдать, что для координат Х, Y, угла тангажа и угла крена предлагаемый алгоритм бэкстеппинга показал себя значительно лучше – ошибка всё же присутствует, но она значительно меньше, чем для регулирования ПД-регулятором. Однако, для угла рыскания ошибка у метода бэкстеппинга будет больше в некоторые моменты, чем у ПД-регулятора, а в некоторые моменты будет меньше, практически нулевой.

# Заключение

В ходе выполнения курсовой работы были решены следующие задачи:

1. Выполнен кинематический анализ робототехнической системы типа квадрокоптер и построена динамическая модель данного робота;
2. С помощью алгоритма быстроисследующего случайного дерева была спланирована траектория на карте с препятствиями, также данная траектория была оптимизирована таким образом, чтобы квадрокоптер поддерживал расстоянием минимум в 1 метр от всех препятствий;
3. Был построен алгоритм управления, базирующийся на методе бэкстеппинга;
4. Было проведено численное моделирование движения квадрокоптера по спланированной траектории и результаты моделирования сравнены с ПД-регулятором, предложенным в базовой модели из Robotic Toolbox;

В итоге было получено, что синтезированный регулятор обеспечивает управление по координатам Х, Y, углу тангажа и углу крена лучше, чем ПД-регулятор, а по углу рыскания регулятор с алгоритмом бэкстеппинга обеспечивает управление не хуже, чем ПД-регулятор.

# Список литературы

1. Варламова Л.П. Применение беспилотных летательных аппаратов в обеспечении технологической безопасности // Journal of Technical and Natural Sciences 5(14), 2019. – 54 – 90.;
2. Peter Corke. Robotics, Vision and Control fundamental algorithms in MATLAB. 2nd edition, 2017. – 697.;
3. Shirsat A., Modeling and control of a Quadrotor, aircraft, UAV. Arizona State University, 2015. – p. 13 – 16.;
4. Hassani H., Mansouri A. Control system of a quadrotor UAV with an optimized backstepping controller. 2019 International Conference on ISACS, 2020. – 7.;
5. He Z, Zhao L. A simple attitude control of quadrotor helicopter based on Ziegler-Nichols rules for tuning PD parameters. ScientificWorldJournal. 2014;2014:280180. doi: 10.1155/2014/280180. Epub 2014 Dec 29. PMID: 25614879; PMCID: PMC4295143.
6. Борисов О.И., Громов В.С., Пыркин А.А., Методы управления робототехническими приложениями. Учебное пособие. — СПб.: Университет ИТМО, 2016. — 108 с;
7. Довгополик И. С., Артемов К., Борисов О. И., Забихифар С., Семочкин А. Н. АЛГОРИТМ МОДИФИЦИРОВАННОГО ИНТЕЛЛЕКТУАЛЬНОГО ДВУНАПРАВЛЕННОГО СЛУЧАЙНОГО ДЕРЕВА ДЛЯ ПЛАНИРОВАНИЯ ДВИЖЕНИЯ АНТРОПОМОРФНЫХ МАНИПУЛЯТОРОВ // Приборостроение. 2022. №3. URL: <https://cyberleninka.ru/article/n/algoritm-modifitsirovannogo-intellektualnogo-dvunapravlennogo-sluchaynogo-dereva-dlya-planirovaniya-dvizheniya-antropomorfnyh> (дата обращения 25.03.2023);
8. Как работает метод Левенберга-Марквардта, 2019. По адресу URL: <https://habr.com/ru/post/470181/> (дата обращения 25.03.2023).

# Приложение

Ниже приведён программный листинг для курсовой работы, использовалась версия MATLAB 2022b. Возможно, некоторые функции по планированию траектории не будут работать в более ранних версиях.

*Листинг программы:*

clc; clear all;

warning('off','all')

% создаем карту

% map = makemap(100)

% save map\_try.mat

%% планирование трпектории методом быстроисследующего случайного дерева

load map\_try.mat

% создает пространство состояний для планирования пути робота

ss = stateSpaceSE2;

% создает валидатор (проверяющий) пути на основе заданной карты препятствий

sv = validatorOccupancyMap(ss);

% загружаем карту препятствий

map = occupancyMap(map);

sv.Map = map;

% задаем минимальное расстояние между двумя состояниями, которые должны быть проверены на достижимость

sv.ValidationDistance = 0.1;

% задаем границы пространства состояний

ss.StateBounds = [map.XWorldLimits;map.YWorldLimits; [-pi pi]];

% создаем планировщик на основе пространства состояний и валидатора пути

planner = plannerRRT(ss,sv);

% задаем максимальную дистанцию между двумя состояниями, которые могут быть соединены в дереве RRT

planner.MaxConnectionDistance = 3;

% задаем начальную точку [x, y, yaw]

% угол рыскания на траектории мы держим постоянным yaw = 0

start = [40,2,0];

% задаем конечную точку [x, y, yaw]

goal = [90,90,0];

% задаем высоту

% высоту на тректории мы держим постоянной

z = 5;

% задаем генератор случайных чисел

rng(100,'twister');

% выполняем планирование пути робота на основе заданного планировщика, начального и конечного состояний

[pthObj,solnInfo] = plan(planner,start,goal);

figure(1)

show(map)

hold on

plot(solnInfo.TreeData(:,1),solnInfo.TreeData(:,2),'.-','MarkerSize',10);

p1 = plot(pthObj.States(:,1),pthObj.States(:,2),'r-','LineWidth',2);

plot(start(1),start(2),'\*g')

plot(goal(1),goal(2),'\*r')

% оптимизируем траекторию

options = optimizePathOptions;

options.ObstacleSafetyMargin = 1;

optPath = optimizePath(pthObj.States,map,options);

hold on

grid minor

p2 = plot(optPath(:,1),optPath(:,2),"m-",LineWidth=1);

legend([p1,p2],"Planned Path","Optimized Path",Location="northwest")

hold off

% выводим параметры траектории для simulink модели

x\_des = optPath(:,1);

y\_des = optPath(:,2);

simout\_ = sim('quadrotor\_project\_2021a');

simout\_pd = sim('quadrotor\_default\_2021a');

figure(2)

plot(simout\_.state.X.Time, simout\_.state.X.Data)

hold on

plot(simout\_.state\_des.x\_\_y\_.Time, simout\_.state\_des.x\_\_y\_.Data(:,1), ...

"LineStyle","--")

grid on

label2 = '${X\_d}$';

label1 = 'X';

legend(label1,label2,'Interpreter',"latex",Location="northwest")

xlabel('t [s]','Interpreter',"latex")

ylabel('X [m]','Interpreter',"latex")

title({'График изменения желаемой и действительной', ...

'координаты Х от времени'})

figure(3)

plot(simout\_.state.Y.Time, simout\_.state.Y.Data)

hold on

plot(simout\_.state\_des.x\_\_y\_.Time, simout\_.state\_des.x\_\_y\_.Data(:,2), ...

"LineStyle","--")

grid on

legend('Y','${Y\_d}$','Interpreter',"latex",Location="northwest")

xlabel('t [s]','Interpreter',"latex")

ylabel('Y [m]','Interpreter',"latex")

title({'График изменения желаемой и действительной', ...

'координаты Y от времени'})

figure(4)

plot(simout\_.state.Z.Time, (simout\_.state.Z.Data) \* (-1))

hold on

plot(simout\_.state.Z.Time, ones(1,length(simout\_.state.Z.Time)) \* ...

simout\_.state\_des.z\_des.Data,"LineStyle","--")

grid on

label2 = '${Z\_d}$';

label1 = 'Z';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('Z [m]','Interpreter',"latex")

ylim([0,10])

title({'График изменения желаемой и действительной', ...

'координаты Z от времени'})

figure(5)

plot3(simout\_.state.X.Data, simout\_.state.Y.Data, (simout\_.state.Z.Data) \* (-1))

hold on

plot3(simout\_.state\_des.x\_\_y\_.Data(:,1), simout\_.state\_des.x\_\_y\_.Data(:,2), ...

ones(1,length(simout\_.state.Z.Time)) \* ...

simout\_.state\_des.z\_des.Data,"LineStyle","--")

hold on

plot3(start(1),start(2),z,'g.','MarkerSize', 20)

hold on

plot3(goal(1),goal(2),z,'r.','MarkerSize', 20)

grid on

label1 = 'traj';

label2 = '${traj\_d}$';

label3 = 'Start';

label4 = 'Finish';

legend(label1,label2,label3,label4,'Interpreter',"latex")

xlabel('X [m]','Interpreter',"latex")

ylabel('Y [m]','Interpreter',"latex")

zlabel('Z [m]','Interpreter',"latex")

zlim([0,10])

title('График желаемой и действительной траектории')

figure(6)

plot(simout\_.state.pitch.Time, simout\_.state.pitch.Data)

hold on

plot(simout\_.state\_des.roll\_\_pitch\_.Time, simout\_.state\_des.roll\_\_pitch\_.Data(:,2), ...

"LineStyle","--")

grid on

label2 = '${\theta\_d}$';

label1 = '${\theta}$';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('${\theta}$ [rad]','Interpreter',"latex")

title({'График изменения желаемого и действительного', ...

'угла тангажа от времени'})

figure(7)

plot(simout\_.state.roll.Time, simout\_.state.roll.Data)

hold on

plot(simout\_.state\_des.roll\_\_pitch\_.Time, simout\_.state\_des.roll\_\_pitch\_.Data(:,1), ...

"LineStyle","--")

grid on

label2 = '${\phi\_d}$';

label1 = '${\phi}$';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('${\phi}$ [rad]','Interpreter',"latex")

title({'График изменения желаемого и действительного', ...

'угла крена от времени'})

figure(8)

plot(simout\_.state.yaw.Time, simout\_.state.yaw.Data)

hold on

plot(simout\_.state.yaw.Time, zeros(1,length(simout\_.state.yaw.Time)), ...

"LineStyle","--")

grid on

label2 = '${\psi\_d}$';

label1 = '${\psi}$';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('${\psi}$ [rad]','Interpreter',"latex")

ylim([-0.2,0.2])

title({'График изменения желаемого и действительного', ...

'угла рыскания от времени'})

% графики для ПД

figure(9)

plot(simout\_pd.state.X.Time, simout\_pd.state.X.Data)

hold on

plot(simout\_pd.state\_des.x\_\_y\_.Time, simout\_pd.state\_des.x\_\_y\_.Data(:,1), ...

"LineStyle","--")

grid on

label1 = 'X';

label2 = '${X\_d}$';

legend(label1,label2,'Interpreter',"latex",Location="northwest")

xlabel('t [s]','Interpreter',"latex")

ylabel('X [m]','Interpreter',"latex")

title({'График изменения желаемой и действительной', ...

'координаты Х от времени для ПД'})

figure(10)

plot(simout\_pd.state.Y.Time, simout\_pd.state.Y.Data)

hold on

plot(simout\_pd.state\_des.x\_\_y\_.Time, simout\_pd.state\_des.x\_\_y\_.Data(:,2), ...

"LineStyle","--")

grid on

label1 = 'Y';

label2 = '${Y\_d}$';

legend(label1,label2,'Interpreter',"latex",Location="northwest")

xlabel('t [s]','Interpreter',"latex")

ylabel('Y [m]','Interpreter',"latex")

title({'График изменения желаемой и действительной', ...

'координаты Y от времени для ПД'})

figure(11)

plot(simout\_pd.state.Z.Time, (simout\_pd.state.Z.Data) \* (-1))

hold on

plot(simout\_pd.state\_des.z\_des.Time, (simout\_pd.state\_des.z\_des.Data) \* (-1),"LineStyle","--")

grid on

label2 = '${Z\_d}$';

label1 = 'Z';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('Z [m]','Interpreter',"latex")

ylim([0,10])

title({'График изменения желаемой и действительной', ...

'координаты Z от времени для ПД'})

figure(12)

plot3(simout\_pd.state.X.Data, simout\_pd.state.Y.Data, (simout\_pd.state.Z.Data) \* (-1))

hold on

plot3(simout\_pd.state\_des.x\_\_y\_.Data(:,1), simout\_pd.state\_des.x\_\_y\_.Data(:,2), ...

(simout\_pd.state\_des.z\_des.Data) \* (-1),"LineStyle","--")

hold on

plot3(start(1),start(2),z,'g.','MarkerSize', 20)

hold on

plot3(goal(1),goal(2),z,'r.','MarkerSize', 20)

grid on

label1 = 'traj';

label2 = '${traj\_d}$';

label3 = 'Start';

label4 = 'Finish';

legend(label1,label2,label3,label4,'Interpreter',"latex")

xlabel('X [m]','Interpreter',"latex")

ylabel('Y [m]','Interpreter',"latex")

zlabel('Z [m]','Interpreter',"latex")

zlim([0,10])

title('График желаемой и действительной траектории для ПД')

figure(13)

plot(simout\_pd.state.pitch.Time, simout\_pd.state.pitch.Data)

hold on

plot(simout\_pd.state\_des.roll\_\_pitch\_.Time, simout\_pd.state\_des.roll\_\_pitch\_.Data(:,2), ...

"LineStyle","--")

grid on

label2 = '${\theta\_d}$';

label1 = '${\theta}$';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('${\theta}$ [rad]','Interpreter',"latex")

title({'График изменения желаемого и действительного', ...

'угла тангажа от времени для ПД'})

figure(14)

plot(simout\_pd.state.roll.Time, simout\_pd.state.roll.Data)

hold on

plot(simout\_pd.state\_des.roll\_\_pitch\_.Time, simout\_pd.state\_des.roll\_\_pitch\_.Data(:,1), ...

"LineStyle","--")

grid on

label2 = '${\phi\_d}$';

label1 = '${\phi}$';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('${\phi}$ [rad]','Interpreter',"latex")

title({'График изменения желаемого и действительного', ...

'угла крена от времени для ПД'})

figure(15)

plot(simout\_pd.state.yaw.Time, simout\_pd.state.yaw.Data)

hold on

plot(simout\_pd.state.yaw.Time, zeros(1,length(simout\_pd.state.yaw.Time)), ...

"LineStyle","--")

grid on

label2 = '${\psi\_d}$';

label1 = '${\psi}$';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('${\psi}$ [rad]','Interpreter',"latex")

ylim([-0.2,0.2])

title({'График изменения желаемого и действительного', ...

'угла рыскания от времени для ПД'})

%%%%%%%%%%%%%%%%%%

% считаем ошибку по координатам для навороченного

error\_x\_R = (simout\_.state\_des.x\_\_y\_.Data(:,1) - simout\_.state.X.Data);

error\_y\_R = (simout\_.state\_des.x\_\_y\_.Data(:,2) - simout\_.state.Y.Data);

error\_z\_R = (ones(length(simout\_.state.Z.Time),1) \* ...

simout\_.state\_des.z\_des.Data - (simout\_.state.Z.Data) \* (-1));

% считаем ошибку по координатам для ПД

error\_x\_PD = (simout\_pd.state\_des.x\_\_y\_.Data(:,1) - simout\_pd.state.X.Data);

error\_y\_PD = (simout\_pd.state\_des.x\_\_y\_.Data(:,2) - simout\_pd.state.Y.Data);

error\_z\_PD = ((simout\_pd.state\_des.z\_des.Data) \* (-1) - (simout\_pd.state.Z.Data) \* (-1));

% считаем ошибку по углам для навороченного

error\_pitch\_R = abs(simout\_.state\_des.roll\_\_pitch\_.Data(:,2) - simout\_.state.pitch.Data);

error\_yaw\_R = abs(zeros(length(simout\_.state.yaw.Time), 1) - simout\_.state.yaw.Data);

error\_roll\_R = abs(simout\_.state\_des.roll\_\_pitch\_.Data(:,1) - simout\_.state.roll.Data);

% считаем ошибку по углам для ПД

error\_pitch\_PD = (simout\_pd.state\_des.roll\_\_pitch\_.Data(:,2) - simout\_pd.state.pitch.Data);

error\_yaw\_PD = (zeros(length(simout\_pd.state.yaw.Time),1) - simout\_pd.state.yaw.Data);

error\_roll\_PD = (simout\_pd.state\_des.roll\_\_pitch\_.Data(:,1) - simout\_pd.state.roll.Data);

error\_Rc = [error\_x\_R error\_y\_R error\_z\_R];

error\_Ra = [error\_pitch\_R error\_yaw\_R error\_roll\_R];

error\_PDc = [error\_x\_PD error\_y\_PD error\_z\_PD];

error\_PDa = [error\_pitch\_PD error\_yaw\_PD error\_roll\_PD];

ttlec = {'Ошибка по координате X', 'Ошибка по координате Y', 'Ошибка по координате Z'};

ttlea = {'Ошибка по углу тангажа', 'Ошибка по углу рыскания', 'Ошибка по углу крена'};

figure(16)

for i = 1 : size(error\_Rc,2)

subplot(3,1,i)

plot(simout\_.state.X.Time, error\_Rc(:,i))

hold on

plot(simout\_pd.state.X.Time, error\_PDc(:,i))

grid on

label1 = 'error\_R';

label2 = 'error\_PD';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('error [m]','Interpreter',"latex")

title(ttlec{i})

end

figure(17)

for i = 1 : size(error\_Ra,2)

subplot(3,1,i)

plot(simout\_.state.X.Time, error\_Ra(:,i))

hold on

plot(simout\_pd.state.X.Time, error\_PDa(:,i))

grid on

label1 = 'error\_R';

label2 = 'error\_PD';

legend(label1,label2,'Interpreter',"latex")

xlabel('t [s]','Interpreter',"latex")

ylabel('error [rad]','Interpreter',"latex")

title(ttlea{i})

end

mkdir images

str = [pwd,"images"];

pathdir = join(str,"\");

cd(pathdir)

for i = 1 : 17

exportgraphics(figure(i),[num2str(i),'.png'],'Resolution',1200);

end