ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

FUZZY МОДЕЛИ МАЛЫХ БЕСПИЛОТНЫХ АППАРАТОВ

Магистерская диссертация
обучающегося по направлению подготовки
09.04.02 Информационные системы и технологии
очной формы обучения
группы 07001635
Чуйкова Анна Геннадьевна

Научный руководитель
доцент к.т.н., доцент кафедры информационных и робототехнических систем НИУ «БелГУ» А.А. Шамраев

Рецензент
д.т.н., доцент, профессор кафедры прикладной информатики и информационных технологий НИУ «БелГУ» А.А. Черноморец

БЕЛГОРОД 2018

РЕФЕРАТ

С целью предоставления надёжного исполнения полётных задач при присутствии, как внешних возмущений, так и внутренних параметрических возмущений необходимо разработать системы, имеющие высокую степень робастности по отношению к эксплуатационным факторам при сохранении определённых ограничений на состав навигационных измерителей и сложность структуры системы управления. Кроме того при этом необходимо отметить, что при присутствии полётов могут появляться разнообразные условия, неучтённые предварительно при процедуре синтеза, “чёткой” робастной системы управления.

Помимо проблем робастности и нечёткой логики, немаловажную значимость в этой диссертационной работе занимает изучение систем управления полётом беспилотных летательных аппаратов (БПЛА).

Некоторые задачи, представляющие как теоретический, так и практический интерес в рамках обсуждаемой научно-технической задачи, продолжают оставаться недостаточно исследованными. Отмеченные условия дают основание считать тему диссертационного исследования актуальной.

Объект исследования: процесс управления малыми БПЛА лёгкого класса.

Предмет исследования: методы и модели гибридного управления движением БПЛА.

Цель исследования: развитие методов гибридного управления техническими объектами за счет использования аппарата нечёткой логики на примере БПЛА.

Задачи исследования:

– проанализировать состояние и возможности улучшения БПЛА с точки зрения возможностей увеличения производительности целевого функционирования их использования;

– разработать математические модели движения БПЛА, а кроме того реализовать их адаптацию применительно к решению задач оптимальной, либо как минимум, рациональной статистической обработки информации;

– осуществить идентификацию аэродинамических данных квадрокоптера;

– синтезировать работоспособные модифицированные правила нечёткого управления.

Научная новизна: совершенствование методов управления плохо формализованными техническими объектами на примере БПЛА на основе аппарата нечёткой логики, что обеспечивает инновации эффективного управления полётом.

Положения, выносимые на защиту: метод управления плохо формализованными техническими объектами на примере БПЛА на основе аппарата нечёткой логики.

Практическая и научная значимость: основные положения проведенного исследования могут лечь в основу дальнейших разработок систем управления с нечёткой логикой. Результаты диссертации могут быть использованы при формировании других новых методов управления беспилотными аппаратами. Кроме того, данная работа крайне актуальна в условиях скорости роста робастных систем.

Структура работы: магистерская диссертационная работа состоит из введения, 3 разделов, заключения, списка использованных источников и приложений.

В первом разделе проводится анализ существующих БПЛА и систем управления по теме исследования, дается характеристика исследуемого объекта в рамках данного проекта.

Выполненный в первом разделе обзор современного состояния разработок БПЛА позволил выявить основные тенденции их развития и предложить условную классификацию, отвечающую существующим и

перспективным разработкам. Приводится описание и обоснование необходимости применения нечётких автопилотов.

Во втором разделе разработаны и проанализированы уравнения движения, которые определяют четырёхроторную систему. Определены системы координат, в которых перемещается четырёхротор. В данном разделе составлена модель, которая проста и в то же время максимально реалистична.

В третьем разделе оцениваются параметры модели, основанные на физической системе. Это не только улучшит точность моделируемой системы, но также поможет определить точные параметры для разрабатываемой системы управления. Моделируется и оценивается система управления БПЛА. В систему вводится ПИД-регулятор на основе нечёткой логики и производится сравнительный анализ системы управления с ПИД-регулятором и ПИД-регулятором на основе нечёткой логики.

В заключении описаны результаты исследования, подведены итоги и сделаны выводы по использованию смоделированной системы управления.

В приложениях представлены программный код смоделированной системы управления с помощью языка программирования MatLab, техническая документация моделируемого летательного аппарата Crazyflie 2.0.

Работа написана на 120 страницах и содержит 51 рисунок, 2 таблицы, 66 использованных источника, 4 приложения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Использование беспилотных летательных аппаратов (БПЛА) приобретает большую популярность, что объясняется наличием существенных достоинств у данного вида авиационной техники [1, 2]: снижается стоимость проведения и подготовки полётов, регламентного обслуживания. Это особенно верно в случае использования электрической двигательной схемы, позволяющей отказаться от необходимости применения высококвалифицированной технической помощи для обслуживания летательного аппарата [1].

Область использования малых БПЛА регулярно расширяется и включает всё больше практических задач, как в гражданской, так и в военной сферах. Многофункциональность БПЛА даёт возможность использовать их с целью решения самых разнообразных задач [1]: геодезические исследования, формирование кадастровых планов, наблюдение транспортной инфраструктуры, объектов энергетики и трубопроводов, определение объемов горных выработок и отвалов, подсчёт перемещения сыпучих грузов, создание карт и планов местности, спасательные, шпионские и военные операции, выявление лесных пожаров, аэрофотосъемки и д.р.

Одним из направлений развития теории практики робастных систем управления полётом считается применение нечётких регуляторов [16,17,18]. В работах Шрама и Фербрюггена [17,18] на примерах простых ПД- и ПИД-регуляторов выявлено то, что в наибольшей степени перспективным путём использования нечётких регуляторов считается комбинированное применение элементов нечёткой логики и традиционных чётких регуляторов в замкнутой динамической системе. Данная концепция отыскала последующее развитие в данной диссертационной работе, которая создана на базе как теории нечётких, так и робастных систем управления.

Помимо проблем робастности и нечёткой логики, немаловажную значимость в этой диссертационной работе занимает изучение систем

управления полётом БПЛА. Разработками в данной сфере занимались такие известные ученые как А.А. Красовский [10], В.А. Боднер [6], И.Е. Казаков [9], И.А. Михалёв, Б.Н. Окоемов, М.С. Чикулаев [12], В.В. Павлов [13], А.Г. Шевелёв, Ю.П. Доброленский [11], D. McLean [14], B.L. Stevens, F.L. Lewis [15].

С целью предоставления надёжного исполнения полётных задач при присутствии, как внешних возмущений, так и внутренних параметрических возмущений, неопределённости параметров, ухудшения характеристик датчиков необходимо разработать системы, имеющие высокую степень робастности по отношению к ранее упомянутым эксплуатационным факторам при сохранении определённых ограничений на состав навигационных измерителей и сложность структуры системы управления. Кроме того при этом необходимо отметить, что при присутствии полётов могут появляться разнообразные условия, неучтённые предварительно при процедуре синтеза, “чёткой” робастной системы управления.

Всё же, некоторые задачи, представляющие как теоретический, так и практический интерес в рамках обсуждаемой научно-технической задачи, продолжают оставаться недостаточно исследованными. Отмеченные условия дают основание считать тему диссертационного исследования актуальной.

Объектом исследования в данной работе является процесс управления малыми БПЛА легкого класса.

Предметом исследования являются методы и модели гибридного управления движением БПЛА.

Целью работы является развитие методов гибридного управления техническими объектами за счёт использования аппарата нечёткой логики на примере БПЛА.

Научная новизна заключается в совершенствовании методов управления плохо формализованными техническими объектами на примере БПЛА на основе аппарата нечёткой логики, что обеспечивает инновации эффективного управления полётом.

Увеличение производительности целевого функционирования БПЛА за счет поиска приемлемого компромисса в отношении допустимого уровня ограниченной правильности оценивания характеристик перемещения и аэродинамических характеристик БПЛА.

Решаемые задачи.

Для достижения поставленной цели необходимо найти решение последующих научно-технические задач:

1) проанализировать состояние и возможности улучшения БПЛА с точки зрения возможностей увеличения производительности целевого функционирования их использования;

2) разработать математические модели движения БПЛА, а кроме того реализовать их адаптацию применительно к решению задач оптимальной, либо как минимум, рациональной статистической обработки информации;

3) осуществить идентификацию аэродинамических данных квадрокоптера;

4) синтезировать работоспособные модифицированные законы нечёткого управления, удовлетворяющие установленным требованиям.

Методы исследования.

Для решения сформулированных задач использовались методы динамики полёта БПЛА, современной теории автоматического управления, стохастической динамики полёта, теории нечётких множеств, главные утверждения теории нечёткого управления, методы математического моделирования, прикладные программы и универсальный программный пакет MatLab. Достоверность и обоснованность научных положений и полученных результатов гарантируются корректным использованием математических методов, моделей и алгоритмов, чёткой формулировкой допущений и условий, в рамках которых проводились расчёты, а также достаточным объемом численного моделирования исследуемых процессов с получением непротиворечивых результатов.

1 Анализ БПЛА и подходы к решению задач управления

1.1 Анализ особенностей и характеристик БПЛА

В понятии многих людей, не имеющих отношения к авиации, непилотируемые летательные устройства представляют собой ряд осложнённых версий радиоуправляемых модификаций самолётов. В некотором смысле это так и есть. Но функции данных аппаратов в последнее время сделались до такой степени различными, что обходиться только лишь таким взглядом на них больше невозможно.

Общепризнанной классификации в обществе ещё не разработали. Но существует несколько выделенных групп.

Все без исключения БПЛА согласно собственному типу и сфере исполняемых задач разделяются на 3 ключевых вида: беспилотные самолёты, беспилотные вертолеты и беспилотные аэростаты[25].

Беспилотные самолеты применяются, в первую очередь, с целью мониторинга площадных и линейных зон территории. Способны преодолевать большие дистанции, выполняя наитруднейшую аэросъёмку онлайн в любое время суток и при различных метеоусловиях. Максимальные качество работы и эффективность выполняемых задач возможны на удалении не более 70 км от наземной станции управления. Скорость – до 400 км/час. Время нахождения в полёте: от 30 минут до 8 часов.

Беспилотные вертолёты применяются с целью своевременного мониторинга локальных участков территории. Они малогабаритны и просты в управлении. Для них не требуется специализированная взлётно-посадочная полоса. Равно как и самолетные летательные аппараты, беспилотные вертолеты могут работать в любое время дня и ночи и при различных атмосферных обстоятельствах. Время полёта: от 30 минут до 3 часов.

Беспилотные аэростаты – инновационные высокоэффективные устройства, назначенные с целью поиска и исследования территории на

высоте до 400 м. Легкие, прочные, мобильные машины, умеющие длительный период работать в режиме реального времени.

Наиболее детально виды непилотируемых летательных аппаратов, можно описать в таблице 1.1.

Таблица 1.1 – Типы беспилотных летательных аппаратов

	Аэростатические	Аэродинамические			Реактивные
		Гибкое крыло	Фиксированное крыло	Вращающееся крыло	
Безмоторные	Аэростаты	Воздушные змеи и аналоги безмоторных аппаратов сверхлегкой авиации (парапланы, дельтапланы и др.)	Планеры		
Моторные	Дирижабли	Аналоги моторных аппаратов сверхлегкой авиации (парапланы, дельтапланы и др.)	БПЛА самолетного типа	БПЛА вертолетного типа	Космические реактивные аппараты

С целью более чёткого определения тех БПЛА, которые будут рассматриваться далее, необходимо подробнее остановиться на такой значительной характеристике как способ управления БПЛА.

Имеются следующие способы:

1) ручное управление оператором (или дистанционное пилотирование) с дистанционного пульта управления в пределах оптической видимости или по видовой информации, поступающей с видеокамеры переднего обзора. При этом управлении диспетчер, в первую очередь, разрешает проблему пилотирования: поддержание нужного курса, высоты и т.д.;

2) автоматическое управление гарантирует возможность полностью независимого полёта летательного аппарата согласно установленной траектории движения, на заданной высоте установленной скоростью и со стабилизацией углов ориентации. Автоматическое управление выполняется с поддержкой бортовых программных приборов;

3) полуавтоматическое управление (или дистанционное управление) – полёт осуществляется автоматически в отсутствии вмешательства человека с помощью автопилота согласно первоначально установленным характеристикам, однако при этом диспетчер способен вносить изменения в маршрут в диалоговом порядке. Следовательно, диспетчер имеет возможность воздействовать на результат функционирования, не отвлекаясь на проблемы пилотирования.

Ручное управление может являться одним из режимов для БПЛА, а может быть единственным методом управления. БПЛА, не имеющие каких-либо средств автоматического управления полётом – радиоуправляемые авиамодели – не могут рассматриваться в качестве платформы с целью исполнения серьёзных целевых задач.

Последние два метода в данный период считаются наиболее востребованными со стороны эксплуатантов непилотируемых систем, т.к. предъявляют минимальные условия к подготовке персонала и предоставляют безопасную и эффективную эксплуатацию систем беспилотных летательных агрегатов. Полностью автоматическое управление может являться оптимальным решением для задач аэрофотосъемки установленной местности, если необходимо снимать на большом удалении с участка базирования за пределами контакта с наземной станцией. Вместе с тем, так как за полёт отвечает лицо, осуществляющее запуск, то возможность воздействовать на полёт с наземной станции может помочь избежать внештатных ситуаций.

На сегодняшний день отечественные БПЛА используются по пяти гражданским направлениям помимо военно-промышленного комплекса. А именно: чрезвычайные ситуации (ЧС) (поиск людей, предупреждение ЧС, спасательные операции и т. д.); безопасность (охрана объектов и людей, а также их обнаружение); мониторинг (атомные электростанции, линии электропередач, земельные, лесные, нефтегазовые, водные ресурсы, сельское

хозяйство и пр.); аэрофотосъемка (геодезия, картография, авиаучет); наука (исследования Арктики, исследование оборудования).

Беспилотные летательные аппараты в мире и в нашей стране применяются в разных областях. Это и мониторинг, и спасательные операции, любительская либо экшен-видеосъемка, в строительстве, геодезии и картографии. Для каждой задачи подходят различные разновидности коптеров.

Разновидности беспилотников согласно предназначению:

– любительское пилотирование или фото-/видеосъёмка;

– для научных исследований;

– для силовых структур или военных (разведывательные, ударные, комбинированные);

– мониторинг, дистанционный контроль и наблюдение;

– наземные (в атмосфере) дроны и для космоса;

– многоцелевые беспилотные комплексы.

Виды беспилотников в зависимости от размера и массы:

В данной классификации беспилотные аппараты колеблются в размерах от микрокоптеров менее 10 кг, которые находятся вблизи от оператора, до тяжёлых, способных подниматься на 20 км и быть в воздухе 24 часа.

– микро- и мини-БПЛА с ближним радиусом – вес до 5 кг; удаленность от оператора – 25-40 км.;

– лёгкие дроны с малым радиусом отлетают на 10-120 км.;

– лёгкие среднего радиуса коптеры имеют вес до 100 кг.;

– средние беспилотники отлетают на 150-1000 км.;

– среднетяжелые дроны весят от 300 до 500 кг.;

– тяжелые коптеры весят от 500 кг, могут долго находиться в воздухе и способы отлетать от оператора на 300 км.

Все без исключения беспилотники разделяются согласно области использования:

– боевые – для ударов по наземным и надводным целям;

– профессиональные – оборудованы сенсорами, компасами, креплениями и GPS-системами;

– любительские, которые предназначены для развлечений.

Так же беспилотники делят на категории согласно степени защиты от окружающей среды, попадания воды, соударения с конструкциями, способности к групповым действиям и возможности транспортирования, а кроме того требующие наличия лицензии и законодательного регулирования или нет.

1.2 Особенности и обзор существующих методов решения задачи управления БПЛА

Проблема установления и систематизации непилотируемых летательных агрегатов, равно как вариации летательных агрегатов, менялась одновременно с процессами развития техники и технологии. Имеются различные представления согласно определения БПЛА, однако, ключевые из них два: классификация по функциональности и системности.

Согласно первому определению БПЛА это летательный агрегат в отсутствии экипажа на борту, оборудованный двигателем и поднимающийся в воздух в результате воздействия аэродинамических сил, управляемый автономно или дистанционно (полуавтономное и ручное управление), умеющий нести нагрузку летательного или не летательного воздействия [3].

Второе определение говорит о том, что БПЛА считаются беспилотными авиационными комплексами (БАС), под которыми подразумевается совокупность комплекса с БПЛА, куда входят наземный пункт дистанционного управления с диспетчерами, управляющими им и обеспечивающими его работу, и каналами управления и связи с

потребителями результатов функционирования БАС [4]. Так же БПЛА разделяются на четыре ключевые категории: БАС с дистанционно пилотируемым летательным аппаратом; с беспилотным автоматическим летательным аппаратом; с дистанционно управляемым летательным аппаратом и с летательным аппаратом, дистанционно управляемым авиационной системой.

Несмотря на различные определения, с точки зрения мехатроники и робототехники, БПЛА это вид специализированных роботов, который относится к классу летательных аппаратов, никак не пилотируемых летчиком, в том числе к тем, чей полёт предварительно запрограммирован на земле и не может быть скорректирован оператором в ходе его исполнения.

Система управления летательного аппарата – совокупность технических устройств, которые обеспечивают управление летательным аппаратом. Под управлением будем понимать процесс изменения характеристик перемещения аппарата в нужном направлении и целенаправленное влияние на работу отдельных систем, на характеристики летательного аппарата для достижения установленной миссии.

1.2.1 Программные методы управления БПЛА

Анализ разработок и исследований в сфере управления и роботизации миниатюрных летательных аппаратов, в частности винтокрылого типа, демонстрирует, то, что в последние десятилетия, количество изучений неоднократно повышалось согласно абсолютно всем тенденциям развития техники и технологии построения БПЛА.

Следует выделить, то, что общий объем исследований и число отечественных разработок был ограничен, при этом наблюдалась тенденция к изучению в большей степени измерительных бортовых систем беспилотных летательных аппаратов.

В мире имеется огромное число диссертаций, посвященных исследованиям миниатюрных летательных аппаратов, в особенности на английском, французском и португальском языках.

Наиболее популярной в сфере моделирования и управления квадрокоптером является работа Буабдалли [7]. Эта работа была приурочена к проблемам квазиавтономности полёта. Так же подробно были рассмотрены конструктивная часть и механическая модель квадрокоптера. В результате, вышел физический прототип миниатюрного летательного аппарата. С помощью полученной модели была изучена аэродинамика четырёхроторных летательных аппаратов и получены уравнения движения.

Работа Буабдалли стала основой абсолютно всех последующих изучений в этой сфере, в ней тщательно рассмотрены проблемы моделирования и управления. Новый закон управления полётом на основе линейно квадратического регулирования был получен на базе линеаризации функций входа. Линейная модель является идеальной, однако никак не предусматривает комплексное взаимодействие состояний полёта. В соответствии с линейным подходом, управление вращательными движениями по тангажу и крену приводит к автоматическому регулированию поступательных движений по тем же осям. Однако отсутствие учёта контуров поступательного движения является его минусом.

Помимо подходов, представленных в труде Буабдалли, имеется несколько способов и алгоритмов планирования траектории БПЛА и квадрокоптера. К числу данных способов относятся: прямой эксплицитный, глобальный эвристический прямой, поисковый комбинированный и псевдо спектральный, а так же метод прямого расположения и на базе генетического алгоритма.

1.2.2 Аппаратные средства реализации систем управления

Определение геометрической траектории, это только лишь одна часть планирования пути, её осуществление находится в зависимости от аппаратной части. Как было отмечено ранее, ключевыми аппаратными средствами с целью планирования траектории являются:

- энкодеры;
- инерционные датчики;
- системы технического зрения;
- генерирование траектории на основе данных навигационных и локационных систем глобальной навигационной спутниковой системы или системы глобального позиционирования (Global Positioning System –GPS).

Широко распространённым считается установление местоположения мобильного колёсного робота на основе энкодера. Они привлекают своей простотой, однако обладают рядом недостатков, связанных с точностью и частотой замеров, и поэтому никак не смогут применяться для воздушных и морских роботов. Таким образом, ограничимся рассмотрением других средств, их достоинств и недостатков.

С развитием микро- и нанотехнологий способы планирования траектории на базе инерционных датчиков сделались крайне распространёнными. Это относится и к малоразмерным беспилотным летательным аппаратам наподобие квадрокоптера.

Инерционные датчики как правило содержат в себе микроэлектромеханические гироскопы, акселерометры и в некоторых случаях магнитометры. На теоретическом уровне, с поддержкой данных датчиков можно получить всю необходимую информацию о положении. Однако МЭМС (микроэлектромеханические системы) датчики вращения в первую очередь работают при возникновении сил Кориолиса и показывают не угол поворота, а угловую скорость. При данном явлении появляется

потребность интегрирования в случае аналогового сигнала и суммирования – в случае дискретного сигнала. В итоге, косвенное измерение вращения будет приближенным, и зависеть от частоты дискретизации сигнала, таким образом, в результате выходной сигнал необходимо оцифровать.

Иным источником погрешности сигналов вращения является проявление дрейфа нулевой отметки в гироскопе, явления, когда даже при статическом положении возникает перемена угла на выходе гироскопа.

Для оценки пройденной линейной дистанции применяется акселерометр. Данный прибор позволяет определять величины линейных ускорений. Однако акселерометры подвергаются высокочастотным и высокоамплитудным помехам, форсирование которых осуществляется с помощью добавочных фильтров (например, фильтр Кальмана). В следствии фильтрации сигнал так же интегрируется с целью получения значения пройденной дистанции, что вызывает погрешность.

Оптическая одометрия – процесс получения информации о положении с помощью фотоаппаратов и видеокамер. Данный способ принадлежит к алгоритмам систем технического зрения. В следствии оптической одометрии получают сведения о пройденной дистанции и направлении движения. Алгоритм оптической одометрии складывается из последовательности шагов, таких как получение изображения и его коррекции, детектирование основных целевых точек в зависимости от подобранного алгоритма распознавания, контроль векторов оптических потоков и распознавание движения носителя фотоаппарата (БПЛА). Минусами метода считаются неопределенность в однотипных изображениях и необходимость в немалой вычислительной мощности.

Способ с применением навигационных систем ориентируется на спутниковую технологию, которая дает возможность осуществлять определение расстояния и устанавливать местоположение (в случае отслеживания). Сигнал спутниковой системы доступен практически повсюду на поверхности Земли. К несчастью, фиксируемые аварии демонстрируют,

то, что в 15% случаев факторами аварий беспилотных летательных аппаратов являются утраты взаимосвязи и достоверность спутниковой навигационной системы, качество которой непосредственно находится в зависимости от количества доступных спутников. Так же качество измерений находится в зависимости от месторасположения и крена орбиты спутника относительно Земного шара. Новейшие спутники устанавливают месторасположение с точностью от 60 см вплоть до одного метра.

1.3 Обоснование необходимости применения нечётких автопилотов

Одна из основных направленностей современной науки сопряжена со сложностью исследуемых предметов. В случае если ранее исследователи имели возможность анализировать только чётко описываемые явления и процессы с небольшим количеством переменных, то с развитием теоретического знания и информационных технологий возникла возможность изучения нового класса концепций, получивших единое наименование – сложные технические системы.

С целью построения систем автоматического управления сложными нелинейными, плохо формализуемыми объектами зачастую используют приборы и алгоритмы управления, произведенные на базе методов нечёткой логики (fuzzy-логики). Данные методы принципиально различаются от обычных традиционных методов автоматики «человеческим» подходом и «человеческими» способами решения задач управления.

Теория нечётких множеств, ключевые идеи которой были предложены американским математиком Лотфи Заде (Lotfi Zadeh) более 40 лет назад, дает возможность характеризовать высококачественные, неточные понятия и наши познания об окружающем мире, а кроме того оперировать данными знаниями для извлечения новой информации. Базирующиеся на данной теории методы построения информационных моделей значительно расширяют классические сферы использования компьютеров и формируют

независимое направление научно-прикладных исследований, которое приобрело специальное наименование – нечёткое моделирование.

Единой предпосылкой с целью использования нечётких систем управления считается, с одной стороны, присутствие неопределённости, связанной как с отсутствием информации, так и сложностью системы и неосуществимостью или нецелесообразностью её отображения классическими методами, а с другой – наличие объекта, необходимых управляющих воздействий, возмущений и т.п., а кроме того наличие информации качественного характера.

Среда функционирования БПЛА характеризуется присутствием заранее неопределенной информации о её параметрах. Данное, в свою очередь, приводит к неопределённости информации о динамике БПЛА как объекта управления (ОУ). Как следствие, разрабатывание алгоритмического обеспечения управления перемещением рассматриваемого вида объектов требует использования при описании процессов нечётких категорий, их представления и знания, а кроме того оперирования ими с целью формулирования определенных выводов и заключений. Одним из наиболее результативных программно-аппаратных средств, гарантирующих в подобных обстоятельствах высокое качество управления, считается, равно как общеизвестно, теория нечётких множеств с базирующимся на ней аппарате нечёткой логики. Нечёткие системы управления (НСУ) принадлежат к классу интеллектуальных систем. Отличительная черта «нечёткого» представления информации, а кроме того присутствие существенного количества входных и выходных переменных и числа формулируемых правил поведения системы дает возможность, с одной стороны, применять имеющиеся интеллектуальные технологии с целью решения обширного спектра задач управления техническими системами самого общего класса [8], с другой стороны – образовывать практически любой закон управления нечёткого регулятора, который будет представлять

собой в данном случае частотнозависимый нелинейный преобразователь для конкретного ОУ.

При этом могут использоваться разнообразные подходы (варианты) решения задачи. В отличие от НСУ, владеющих признаком «интеллектуальности в большом» и обладающих многоуровневой системой (уровни обучения, самоорганизации, прогноза событий, работы с базами знаний, формирования решений, адаптации, исполнительный уровень), нечёткие регуляторы (НР) принадлежат системам, обладающим свойством «интеллектуальности в малом». Применяемые при их функционировании знания как способ преодоления неопределённости входной информации, в частности модели ОУ или его поведения, включают только уровень работы с базами знаний и исполнительный уровень (НР на основе алгоритмов Мамдани-Заде и Цукамото). Помимо этого, они имеют способность к обучению (алгоритмы Такаги-Сугено или (T-S) – структуры). С учетом описанных ранее способностей функционирования ДПЛА, направленность на создание для них обладающих «интеллектом в большом» НСУ вряд ли имеет возможность расцениваться результативной. Использование НР как средства преодоления неопределённости, при этом неопределённости ограниченного типа, имеют возможность гарантировать необходимую степень грубости и стабильную сходимость процессов, что дает основание рассматривать такого рода аспект наиболее подходящим. Осуществление указанного подхода базируется на концепции многоцелевого аппроксиматора. Решение рационально искать на базе построения агрегированного НР с интенсивным признаком грубости, в котором, помимо аппроксиматоров чётких регуляторов, к примеру, робастного типа, или оптимальных, применяются нечёткие аппроксиматоры действующих ограниченных возмущений, приведенных к входу ОУ, используемых в качестве компенсаторов данных возмущений. Выделим при этом, то, что процедура приведения косвенно определяемых возмущений по входу ОУ считается далеко не формализованным.

Создание любого НР в данном случае обязано базироваться на фаззификации ошибки системы или выхода робастного регулятора (РР). Число входных множеств фаззификации НР избирается для каждой входной переменной соответствующего НР. Подобные множества описываются установленными значениями по уровню (обычно О; Н; П – соответственно отрицательное, нулевое, положительное значения по уровню). Их формирование в ходе реального полёта БПЛА в виде термов определенных лингвистических переменных, представляющиеся, функциями переменных состояния, исполняется оператором в зависимости от степени стабильности наблюдения изображения первоначально сопровождаемого объекта, идентифицируемого как цель, перед вторичным заходом БПЛА на него. Характеристики «заключений» нечётких правил НР могут быть определены при моделировании с помощью редактора ANFIS раздела Fuzzy программного пакета Matlab Toolbox Fuzzy, Simulink с обеспечением грубости и устойчивости любой нечёткой модели.

1.4 Применение методов нечёткой логики для проектирования СУ

Одним из наиболее перспективных направлений современной теории управления является использование регуляторов с нечёткой логикой [19,16,20,17,18]. Это особенно важно для задач управления полётом, где использование нечёткой логики даёт возможность повысить работоспособность системы при возникновении различных непредвиденных ситуаций. К таким ситуациям относятся: отказы датчиков или ухудшение их характеристик, изменение параметров объекта управления, изменение характеристик внешних возмущений и т. п. В данном значении малые БПЛА становятся более чувствительными из-за их небольшой энерговооружённости, ограниченного числа датчиков параметров полёта, их низкой точности и т.п.

Системы управления полётом (СУП) малых беспилотных летательных аппаратов должны удовлетворять ряду противоречивых требований: простоты, надёжности, дешевизны, небольшого веса и энергопотребления с одной стороны, а кроме того правильности управления полётом при наличии внешних стохастических возмущений, помех измерений и неопределённости характеристик модификации БПЛА с другой стороны. Компромиссное решение среди данных условий достигается на сегодняшний период времени с помощью использования современных способов концепции робастных систем управления.

По этой причине цель синтеза замкнутых динамических систем, использующих нечёткую логику, с целью управления полётом небольших БПЛА считается весьма значительной и актуальной как с теоретической, так и с практической точек зрения.

Выполненный в разделе обзор современного состояния разработок БПЛА позволил выявить основные тенденции их развития и предложить условную классификацию, отвечающую существующим и перспективным разработкам.

2 Разработка моделей для управления БПЛА

Чтобы разработать точные системы управления для четырёхроторной платформы, необходимо разработать и проанализировать уравнения движения, которые определяют четырёхроторную систему. Четырехротор определяется набором нелинейных уравнений, которые затрудняют точное моделирование и управление. Как только все компоненты уравнений будут составлены, можно упростить уравнения движения, сделав несколько предположений о методе работы четырехротора. Цель состоит в том, чтобы сделать модель, которая проста и в то же время максимально реалистична.

2.1 Определение системы координат и переменных

Четырёхроторный летательный аппарат работает на понятиях переменных крутящих моментов и тяг. Каждый ротор состоит из двигателя постоянного тока и ротора с фиксированным шагом [31]. Двигатели расположены парами по горизонтальной и вертикальной осям, причем передняя пара вращается по часовой стрелке, а горизонтальная пара вращается против часовой стрелки. Эта конструкция приводит к тому, что реактивные моменты от пары двигателей точно противоположны друг другу, если они все вращаются с одинаковой скоростью. Устранение крутящего момента позволяет транспортному средству поддерживать постоянный курс при полёте. Рыскание регулируется изменением скоростей пар двигателей для создания ненулевого счетчика крутящего момента. Высота регулируется путем изменения тяги от каждого двигателя равными количествами, чтобы обеспечить чистый вектор тяги и без крутящего момента. Для перемещения в боковых направлениях относительные скорости каждого двигателя в боковой паре изменяются для создания желаемого смещения боковой тяги. Благодаря простой конструкции в небольших аппаратах, которые работают от батареи, они способны выполнять стабильное парение, и безопасны для

использования в помещениях. Четырехротор классифицируется как система с пониженным напряжением. Несмотря на то, что четырехротор может двигаться в 6 степенях свободы (3 поступательных и 3 вращательных) есть только 4 входных сигнала, которые можно контролировать (скорости 4 двигателей) [31]. Как будет показано ниже, вращательная и поступательная динамика связаны, что представляет интересную задачу управления.

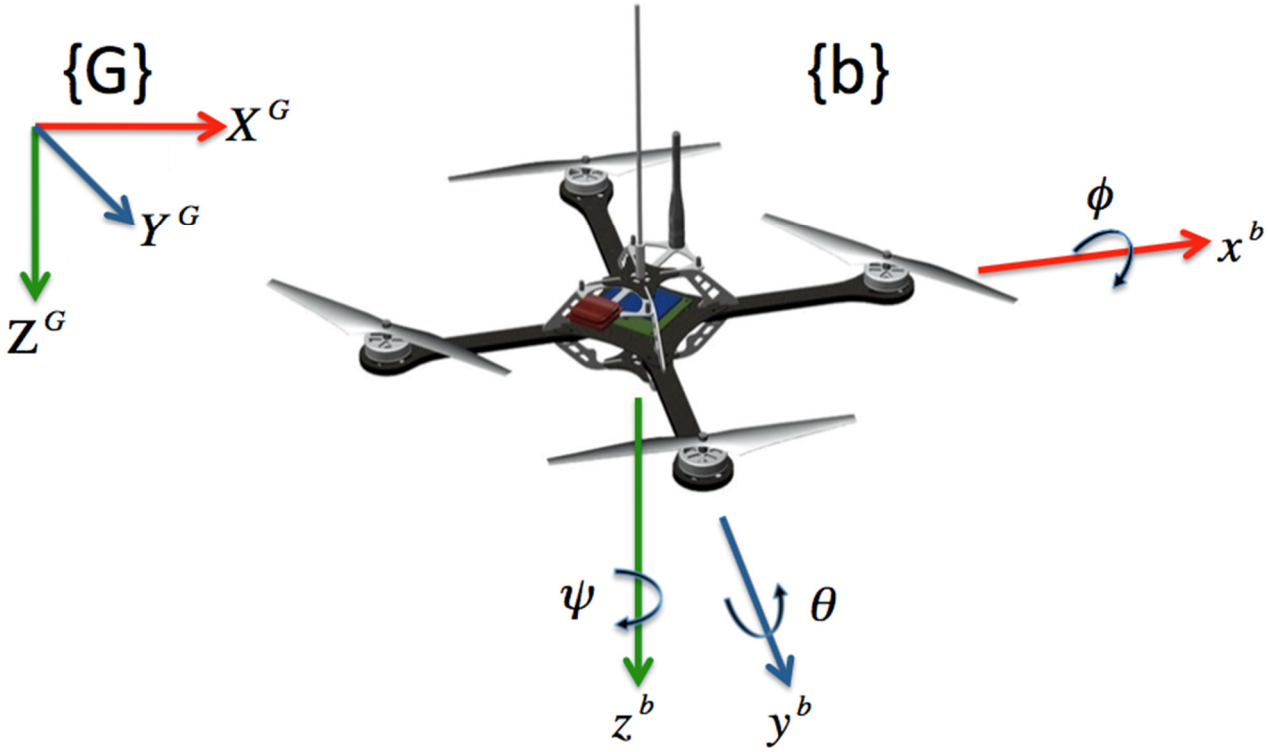

Рисунок 1 – Основная структура четырехротора

Основная структура четырехротора, используемая для разработки модели, показана на рисунке 1, изображающем углы Эйлера крена, тангажа и рыскания, систему координат тела {b} и глобальную систему координат {G}. Эта модель опирается на несколько предположений:

– конструкция четырехротора жесткая и симметричная с центром масс, выровненным с центром рамы летательного аппарата;

– тяга и сопротивление каждого двигателя пропорциональны квадрату скорости двигателя,

– пропеллеры считаются жесткими, и поэтому раскачивания лопастей незначительны (деформация лопастей пропеллера из-за высоких скоростей и гибкого материала);

– земля плоская и не вращающаяся (разница в гравитации по высоте или вращение Земли незначительна);

– окружающая среда (ветер) незначительна;

– влияние Земли незначительно.

Положение четырехротора задается в глобальной системе отсчета, а скорость и угловая скорость определяются в каркасе корпуса четырехротора. Важно отметить, что разные общие датчики производят измерения в обеих системах координат. GPS, например, измеряет положение четырехротора и скорость земли в глобальной системе отсчета, в то время как акселерометры и гироскопы скорости производят измерения в корпусе кузова. Для более улучшенной навигации, включая траекторию движения, точки маршрута указаны в глобальной системе координат.

2.2 Кинематика поступательного движения

Переменные состояния для скорости находятся в системе координат тела, а переменные состояния для положения находятся в глобальной системе координат. Поэтому необходимо определить матрицу вращения для преобразования переменных между системами координат. Преобразование между глобальной системой координат и системой координат тела описано ниже. Для упрощения представления cos и sin были сокращены до c и s соответственно.

$$x^b = R_G^b X^G = R(\phi)R(\theta)R(\psi)X^G \tag{1}$$

$$\mathrm{R}_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}(\phi) & s(\phi) \\ 0 & -s(\phi) & c(\phi) \end{bmatrix}; \mathrm{R}_\theta = \begin{bmatrix} c(\theta) & 0 & -s(\theta) \\ 0 & 1 & 0 \\ c(\theta) & 0 & -c(\theta) \end{bmatrix} \text{ и } \mathrm{R}_\psi = \begin{bmatrix} c(\psi) & s(\psi) & 0 \\ -s(\psi) & \mathrm{c}(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$R_G^b = \begin{bmatrix} c(\psi)c(\theta) & s(\psi)c(\theta) & -s(\theta) \\ c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & s(\phi)s(\psi)s(\theta) + c(\phi)c(\psi) & c(\theta)s(\phi) \\ c(\phi)c(\psi)s(\theta) + s(\phi)s(\psi) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) & c(\phi)c(\theta) \end{bmatrix} \quad (3)$$

$$X^G = R_b^G x^b = R(\phi)^T R(\theta)^T R(\psi)^T x^b = (R_G^b)^T x^b \quad (4)$$

$$R_b^G = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & c(\phi)c(\psi)s(\theta) + s(\phi)s(\psi) \\ s(\psi)c(\theta) & s(\psi)s(\psi)s(\theta) + c(\phi)c(\psi) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\phi)c(\theta) \end{bmatrix} \quad (5)$$

Где $X^G -$ глобальное положение четырехротора (Север); $Y^G -$ глобальное положение четырехротора (Восток); $Z^G -$ глобальное положение четырехротора (Низ); $x^b -$ локальное положение четырехротора в системе координат тела (Север); $y^b -$ локальное положение четырехротора в системе координат тела (Восток); $z^b -$ локальное положение четырехротора в системе координат тела (Низ); $\phi -$ угол крена; $\theta -$ угол тангажа; $\psi -$ угол рыскания; $p -$ угловая скорость по крену; $q -$ угловая скорость по тангажу; $r -$ угловая скорость по рысканию.

2.3 Кинематика вращательного движения

Так как угловые скорости определены в системе координат тела, а углы Эйлера определены в промежуточных системах координат, мы можем использовать полученную выше матрицу вращения, чтобы определить зависимость между угловыми скоростями и временными производными углов Эйлера, как показано ниже. Угловые скорости – это векторы, направленные вдоль каждой оси вращения и не равные временной производной углов Эйлера. Вывод ниже предполагает, что производная по времени каждой скорости Эйлера мала. Модель кинематики вращательного движения четырехротора (6-9).

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = R(\phi)R(\theta)\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} + R(\phi)\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} \quad (6)$$

$$\omega = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta) \\ 0 & \mathrm{c}(\phi) & s(\phi)c(\theta) \\ 0 & -s(\phi) & c(\phi)c(\theta) \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = S\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (7)$$

$$S^{-1} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & \mathrm{c}(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \quad (8)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & \mathrm{c}(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix}\begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (9)$$

Если предположить, что углы Эйлера малы (около 0), то матрица S становится единичной матрицей, а угловые скорости примерно равны производной по времени углов Эйлера.

2.4 Уравнения движения мотора

Четырехроторы используют различные типы моторов для генерации тяги и крутящих моментов, необходимых для управления платформой. Каждый двигатель питается от электрических батарей, которые держат бортовую платформу четырехротора. Электронные контроллеры скорости (ESC) получают требуемую скорость вращения двигателя контроллером полёта и отправляют эти команды каждому отдельному двигателю. На рисунке 2 показана базовая модель тела четырехротора, на которой изображены тяги и силы каждого двигателя на четырёхроторе [33]. Стоит обратить внимание, что ось z определена отрицательно в направлении силы тяжести и в противоположном направлении векторов силы двигателя.

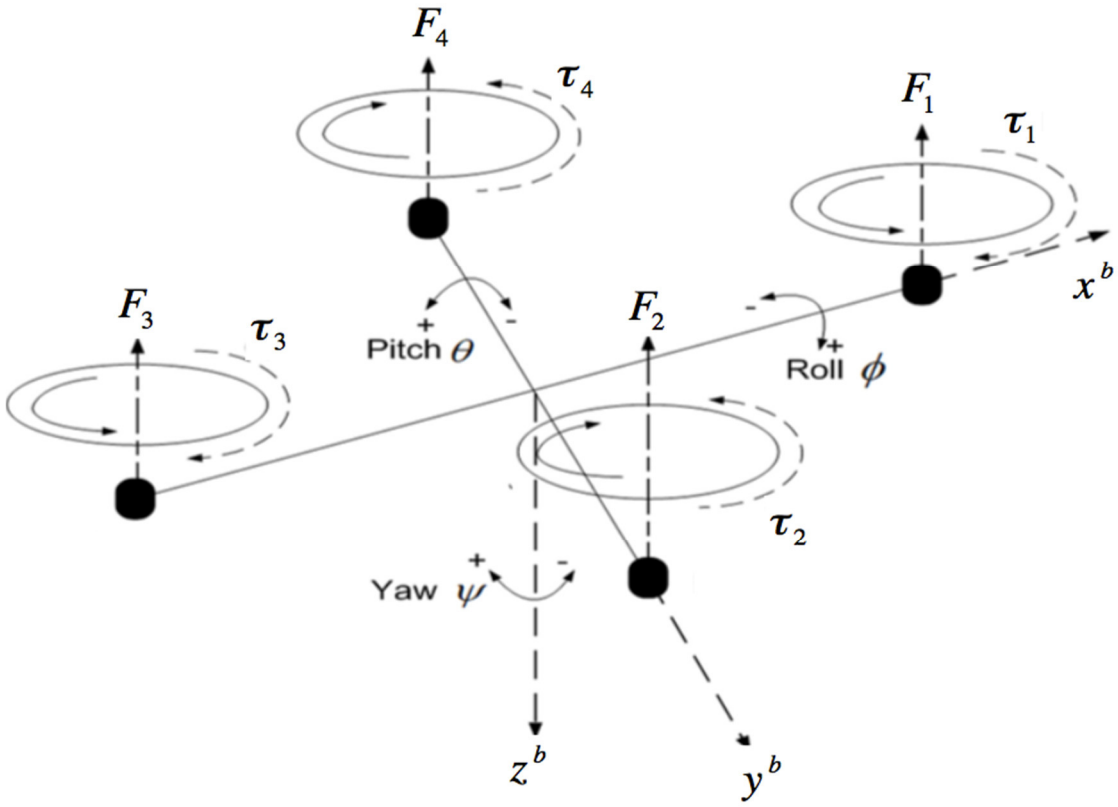

Рисунок 2 – Базовая модель тела четырехротора

В первой серии уравнений ниже приводятся уравнения крутящего момента и мощности типичного мотора без сердечника. Предполагаем, что сопротивление двигателя и ток без нагрузки приблизительно равны 0. Так как на четырехроторе все двигатели одинаковы, для каждого двигателя применяется одна модель. Получаем уравнение для величины мощности, которую производит двигатель в одном упрощенном уравнении.

В формулах 10-12 $\tau -$ крутящий момент мотора; $K_t -$ константа пропорциональности крутящего момента; $I -$ входной ток; $I_0 -$ ток без нагрузки; $V -$ падение напряжения на моторе; $R_m -$ сопротивление мотора; $K_v -$ обратный коэффициент электродвижущей силы (ЭДС); $\omega -$ угловая скорость мотора; $P_m -$ мощность мотора.

$$\tau = K_t(I - I_0) \tag{10}$$

$$V = IR_m + K_v\omega \tag{11}$$

$$P_m = IV = \frac{(\tau + K_t I_0)(R_t I_0 R_m + \tau R_m + K_t K_v \omega)}{K_t^2} \quad (12)$$

Предположим, что $R_m \approx I_0 \approx 0$ и $P_m = \frac{K_v \tau \omega}{K_t}$.

Следующие уравнения используют приведенные выше уравнения мощности и крутящего момента для составления обобщенного уравнения тяги. Потребляемая мощность при нависании – идеальная мощность, необходимая для создания тяги, перемещая колонну воздуха. Предположим, что скорость летательного аппарата низкая, поэтому скорость при нависании – это скорость воздуха при нависании (формула 14). Из этого уравнения видно, что более крупные пропеллеры приводят к меньшему потреблению энергии. В то время как каждый ротор оставляет за собой след после его движения, поэтому предполагаем, что скорость свободного потока воздуха равна нулю. Это приводит к тому, что тяга каждого двигателя пропорциональна произведению квадрата угловой скорости вала ротора и коэффициента подъемной силы (формула 16). Это уравнение описывает взаимосвязь между скоростью двигателя и результирующей силой тяги, что является основой для стабильного управления летательным средством.

В формулах 13-20 P_h – мощность зависания; P_m – мощность двигателя; T – тяга зависания; v_h – индуцируемая скорость при зависании; ω – угловая скорость двигателя; ρ – плотность воздуха; A – площадь, охваченная лопастями ротора; K_v – обратный коэффициент ЭДС; K_τ – константа пропорциональности крутящего момента; K_T – коэффициент тяги; τ – вращающий момент двигателя.

$$P_h = T v_h \quad (13)$$

$$T = 2\rho A v_h^2 \quad (14)$$

$$\tau = K_\tau T \quad (15)$$

$$P_m = \frac{K_v \tau \omega}{K_t} = \frac{K_v K_\tau K_\omega}{K_t} \quad (16)$$

$$P_h = \frac{T^{\frac{2}{3}}}{\sqrt{2\rho A}} \quad (17)$$

$$\frac{K_v K_\tau K_\omega}{K_t} = \frac{T^{\frac{2}{3}}}{\sqrt{2\rho A}} \tag{18}$$

$$T = \left[\left(\frac{K_v K_\tau \sqrt{2\rho A}}{K_t}\right)\omega\right]^2 \tag{19}$$

$$T = K_T \omega^2 \tag{20}$$

2.5 Динамика поступательного движения

Линейные уравнения движения определены в глобальной системе отсчета. Ускорение четырехротора в глобальной системе координат равно сумме силы тяжести, осевой силы двигателей и линейной силы трения, приводящей к сопротивлению. Вектор тяги в системе координат тела преобразуется в глобальную систему координат с использованием матрицы вращения, определенной ранее. Результирующая сила, создаваемая четырьмя двигателями в глобальной системе координат, определена в формуле 25. Кроме того, результирующая глобальная сила сопротивления вдоль каждой поступательной оси определяется в формуле 26 как произведение коэффициентов сопротивления и линейных скоростей в каждом направлении в глобальной системе координат. Это простая модель силы трения окружающей среды. Более сложное моделирование будет включать в себя отдельные константы трения для каждой оси, а также моделирование трения в корпусе тела вместо инерциальной системы отсчета.

В формулах 21-26 m – масса квадрокоптера; g – ускорение силы тяжести; $\ddot{X}^G$ – ускорение в глобальной системе координат; $\dot{X}^G$ – скорость в глобальной системе координат; F_g – сила тяжести в глобальной системе координат; F_d – сила сопротивления в глобальной системе координат; F_T^G – общая сила тяги в глобальной системе координат; F_T^b – общая сила тяги в системе координат тела; R_b^G – матрица вращения тела к глобальной системе координат; F_i – тяга каждого мотора; ω – угловая скорость мотора; K_T – коэффициент тяги; $K_{dx} = K_{dy} = K_{dz}$ – коэффициент сопротивления.

$$m\ddot{X}^G = F_g - F_T^G - F_d \tag{21}$$

$$\ddot{X}^G = \begin{bmatrix} \ddot{X}^G \\ \ddot{Y}^G \\ \ddot{Z}^G \end{bmatrix} \text{ и } F_g = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{22}$$

$$F_d = \begin{bmatrix} K_{dx} & 0 & 0 \\ 0 & K_{dy} & 0 \\ 0 & 0 & K_{dz} \end{bmatrix} \begin{bmatrix} \dot{X}^G \\ \dot{Y}^G \\ \dot{Z}^G \end{bmatrix} \tag{23}$$

$$F_T^G = R_b^G F_T^b \tag{24}$$

$$F_T^b = \sum_{i=1}^{4} F_i \tag{25}$$

$$F_T^G = R_b^G \sum_{i=1}^{4} F_i = \begin{bmatrix} 0 \\ 0 \\ K_T \sum_{i=1}^{4} \omega_i^2 \end{bmatrix} \tag{26}$$

2.6 Динамика вращательного движения

Вращательные уравнения движения определяются в системе координат тела так, что вращение может быть вычислено относительно центра четырёхротора, а не центра глобальной системы координат.

В формулах 27-41 ω – угловая скорость; $\dot{\omega}$ – скорость углового ускорения; J_b – момент инерции тела квадрокоптера; J_m – момент инерции двигателя; J_r – момент инерции ротора; J_x – момент инерции четырехротора в x_b; J_y – момент инерции четырехротора в y_b; J_z – момент инерции четырехротора в z_b; τ_m – крутящий момент двигателя; τ_g – крутящий момент гироскопического эффекта; $\tau_{\phi,\theta,\psi}$ – крутящий момент по осям тела; τ_D – крутящий момент; K_d – константа пропорциональности вращающего момента; K_T – коэффициент силы тяги; R – радиус пропеллера; A – сечение пропеллера; C_D – безразмерный коэффициент; ρ – плотность воздуха; ℓ – длина плеча.

$$J_b\dot{\omega} = \tau_m - \tau_g - (\omega * J_b\omega) \tag{27}$$

$$J_b = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \dot{\omega} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} (\omega * J_b\omega) = \begin{bmatrix} \dot{\theta}\dot{\psi}(J_z - J_y) \\ \dot{\psi}\dot{\phi}(J_x - J_z) \\ \dot{\theta}\dot{\phi}(J_y - J_z) \end{bmatrix} \tag{28}$$

$$J_b\dot{\omega} = \tau_m - \tau_\psi \tag{29}$$

При зависании $\dot{\omega} = 0$ так $\tau_\psi = \tau_D$.

$$\tau_D = \frac{1}{2} R_\rho C_D A(\omega R)^2 = K_d\omega^2 \tag{30}$$

В уравнении 30 показано, что крутящий момент из-за сопротивления является пропорциональным произведению константы сопротивления и квадрата угловой скорости ротора.

$$\tau_\psi = \tau_D = (-1)^{i+1} K_d\omega_i^2 = K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \tag{31}$$

$$\tau_{\phi,\theta} = \sum r * T \tag{32}$$

$$\tau_\phi = \ell K_T(\omega_4^2 - \omega_2^2) \tag{33}$$

$$\tau_\theta = \ell K_T(\omega_1^2 - \omega_3^2) \tag{34}$$

$$\tau_m = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} \ell K_T(\omega_4^2 - \omega_2^2) \\ \ell K_T(\omega_1^2 - \omega_3^2) \\ K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \tag{35}$$

$$\tau_g = \omega * G_Z \sum_{i=1}^{4} J_r\omega_i \tag{36}$$

$$\tau_g = \begin{bmatrix} \dot{\theta} \\ -\dot{\phi} \\ 0 \end{bmatrix} J_r \sum_{i=1}^{4} (-1)^{i+1}\omega_i = \begin{bmatrix} J_r\dot{\theta}(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ -J_r\dot{\phi}(\omega_1 - \omega_2 + \omega_3 - \omega_4) \\ 0 \end{bmatrix} \tag{37}$$

Поскольку предполагалось, что четырехротор является симметричным, то матрица момента инерции четырехротора симметрична. Эти уравнения состоят из трех терминов, включая аэродинамические эффекты и крутящие моменты двигателя. Во-первых, гироскопический эффект, возникающий в

результате вращения твердого тела. Во-вторых, двигатели производят крутящий момент крена, тангажа и рыскания. Это уравнение тяги (формула 36) предполагает, что четырехротор работает в стабильном полёте и что пропеллеры поддерживают постоянную тягу и не ускоряются. Это допущение приводит к тому, что крутящий момент вокруг глобальной оси z равен крутящему моменту из-за сопротивления. В-третьих, перекрестное произведение, которое описывает гироскопический эффект, возникающий в результате вращения винта, связанного с вращением тела. Это связано с тем, что ось вращения ротора сама движется с угловой скоростью рамы.

2.7 Уравнения движения

Решение для глобального поступательного ускорения и угловых ускорений в системе отчета тела приводит к следующим уравнениям, показанными в формулах 38-41.

$$\begin{bmatrix} \dot{X}^G \\ \dot{Y}^G \\ \dot{Z}^G \end{bmatrix} = \begin{bmatrix} c(\psi)c(\theta) & c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & c(\phi)c(\psi)s(\theta) + s(\phi)s(\psi) \\ s(\psi)c(\theta) & s(\psi)s(\psi)s(\theta) + c(\phi)c(\psi) & c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\phi)c(\theta) \end{bmatrix} \begin{bmatrix} \dot{x}^b \\ \dot{y}^b \\ \dot{z}^b \end{bmatrix} \quad (38)$$

$$\begin{bmatrix} \ddot{X}^G \\ \ddot{Y}^G \\ \ddot{Z}^G \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-[c(\phi)c(\psi)s(\theta) + s(\phi)s(\psi)]F_T^b - K_{dx}\dot{X}^G) \\ \frac{1}{m}(-[c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi)]F_T^b - K_{dy}\dot{Y}^G) \\ \frac{1}{m}\left(-[c(\phi)c(\theta)]F_T^b - K_{dz}\dot{Z}^G\right) + g \end{bmatrix} \quad (39)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & s(\phi)t(\theta) & c(\phi)t(\theta) \\ 0 & c(\phi) & -s(\phi) \\ 0 & \frac{s(\phi)}{s(\theta)} & \frac{c(\phi)}{c(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (40)$$

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_z}[(J_y - J_z)qr - J_r q(\omega_1 - \omega_2 + \omega_3 - \omega_4) + \ell K_T(\omega_4^2 - \omega_2^2)] \\ \frac{1}{J_z}[(J_z - J_x)qr - J_r q(\omega_1 - \omega_2 + \omega_3 - \omega_4) + \ell K_T(\omega_1^2 - \omega_3^2)] \\ \frac{1}{J_z}[(J_x - J_y)pq + K_d(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix} \quad (41)$$

Используется глобальная система отсчёта для поступательного положения, так как это та же система координат, что использует GPS-датчик. Аналогично, система отсчета тела выбирается для ориентации, так как инерционное измерительное устройство – inertial measurement unit (IMU), включающее в себя акселерометр, магнитометр и гироскоп также выполняют измерения в корпусе тела. Эти уравнения являются как нелинейными, так и высокосвязанными. Их можно упростить, используя допустимые предположения для разработки и внедрения стабильных систем управления.

В этом разделе была составлена и проанализирована математическая модель движения четырёхротора. Как только были составлены все компоненты уравнений, эти уравнения движения были упрощены, сделано несколько предположений о методе работы четырёхротора.

3 Разработка физической модели

Этот раздел основан на построенной математической модели, описанной ранее в разделе 2. В этом разделе оцениваются параметры модели, основанные на физической системе. Это не только улучшит точность моделируемой системы, но также поможет определить точные параметры для разрабатываемой системы управления. Чем точнее исходные параметры управления, тем меньше времени требуется на настройку значений.

3.1 Параметры физической модели

3.1.1 Момент инерции корпуса четырёхротора

Параметры физической модели основаны на наборе Crazyflie 2.0. Моменты инерции можно вычислить несколькими способами. В приведенных ниже уравнениях в центре тела моделируется полая сфера. Центр тела – это центр четырёхротора, на котором закрепляется все датчики, батарея, распределительный щит питания, плата. Эти уравнения, а также соответствующие значения вычисляются в формулах 42-46, где J_x – момент инерции четырехротора в x_b; J_y – момент инерции четырехротора в y_b; J_z – момент инерции четырехротора в z_b; m_s – масса собранного корпуса и электроники; r_s – радиус собранного корпуса; m_m – масса мотора; r – радиус мотора; h – высота мотора; ℓ – длина плеча.

$$J_x = J_y = \frac{1}{12} m_m (3r^2 + h^2) \tag{42}$$

$$J_z = \frac{1}{2} m_m r^2 \tag{43}$$

$$J_x = J_y = J_z = \frac{2}{3} m_s r_s^2 \tag{44}$$

$$J_x = J_y = \frac{2}{3} m_s r_s^2 + 2\left(\frac{1}{12} m_m (3r^2 + h) + m_m \ell^2\right)$$

$$J_x = J_y = \frac{2}{3}(1.1 + .08^2) + 2(\frac{1}{12}.072(3.028^2 + .036^2) + .072 * .56^2 \quad (45)$$

$$J_x = J_y = .04989$$

$$J_z = \frac{2}{3} m_s r_s^2 + 4(\frac{1}{2} m_m r^2 + m_m \ell^2)$$

$$J_z = \frac{2}{3}(1.1 * .08^2) + 4(\frac{1}{2}(.072 * .028^2) + .072 * .56^2) \quad (46)$$

$$J_z = .24057$$

3.1.2 Момент инерции пропеллера

Чтобы вычислить момент инерции пропеллеров, можно воспользоваться прямым подходом. Из технической документации берем вес пропеллера (6 гр.) и его приблизительную плотность. После чего половина пропеллера делиться на три части. Затем измеряется значение этих частей. Зная объем этих частей и плотность пропеллера (однородное тело), вычисляется вес этих частей. Зная массу, размеры и расстояние от оси вращения этих частей (таблица 3.1), можно рассчитать их моменты инерции, которые вносят вклад в общий момент инерции всего пропеллера.

Таблица 3.1 – Данные половины пропеллера

Часть	Масса (гр.)	Ширина (см)	Высота (см)	r (см) (расстояние от оси вращения)
I	0.17	0.3	0.25	0.9
II	0.26	0.44	0.28	0.5
III	0.12	0.5	0.12	0

Момент инерции тонкой прямоугольной пластины высотой h и ширины w и массы m рассчитывается с помощью уравнения (47).

$$I = m\left(\frac{h^2}{12} + \frac{w^2}{12}\right) \quad (47)$$

Используя теорему о параллельной оси, можно вычислить момент инерции тела с массой m с уже известным моментом инерции I_0 вокруг произвольной оси. Новая ось вращения находится на расстоянии r от исходной оси и параллельна ей (рисунок 3).

$$I = I_0 + mr^2 \tag{48}$$

Рисунок 3 – Схема параллельной оси

Тогда ось вращения перемещается в конец пластины (рисунок 4).

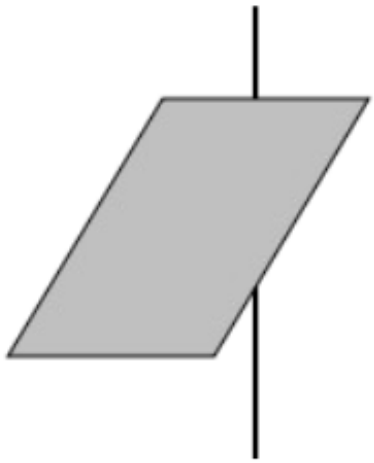

Рисунок 4 – Тонкая пластина, вращающаяся вокруг боковой оси

Затем

$$\mathrm{I} = \mathrm{I_0} + \mathrm{mr^2} = \mathrm{m}\left(\frac{\mathrm{h^2}}{12} + \frac{\mathrm{w^2}}{12}\right) + \mathrm{m}\frac{\mathrm{h^2}}{4} = \mathrm{m}\left(\frac{\mathrm{h^2}}{3} + \frac{\mathrm{w^2}}{12}\right) \tag{49}$$

Используя значения из таблицы и уравнение (49) для вычисления момента инерции каждой части, момент инерции всего пропеллера равен:

$$I_p = 2(I_I + I_{II} + I_{III}) = 0.0402 * 10^{-5}\text{кг/м}^2 \tag{50}$$

3.2 Моделирование скорости гироскопа

Используемые датчики были стандартными с платой Crazyflie 2.0 (приложение Г). Чтобы улучшить симуляцию и, следовательно, проектирование контроллера, нужно точно имитировать производительность датчиков. Эти параметры будут использоваться для добавления ошибок в моделируемые измерения датчиков. Затем можно использовать фильтр, чтобы уменьшить влияние шума датчика, одновременно объединяя различные измерения датчика, чтобы рассчитать более точную калибровку и информацию о местоположении.

Гироскоп скорости МЭМС содержит небольшой вибрационный рычаг. Чтобы измерить вращение, изменение частоты рычага из-за эффекта Кориолиса (сила Кориолиса – одна из сил инерции, использующаяся при рассмотрении движения материальной точки относительно вращающейся системы отсчёта, добавление силы Кориолиса к действующим на материальную точку физическим силам позволяет учесть влияние вращения системы отсчёта на такое движение.) измеряется при повороте рычага. Если гироскопы совпадают с осью x, y и z четырехротора, гироскопы измеряют угловые скорости тела p, r и q.

Гироскоп можно моделировать как комбинацию истинной угловой скорости, смещения нуля и плотности шума. Истинная угловая скорость измеряется в радианах в секунду. Шум характеризуется как нулевой средний белый Гауссовский шум. Смещение зависит от температуры и может быть оценено для каждого полёта. Эти элементы создают следующие уравнения для моделирования гироскопа. Эта же модель используется для каждой оси.

В формулах 51-57 $\tilde{p}$ – измеренная угловая скорость вдоль оси x_b; $\tilde{q}$ – измеренная угловая скорость вдоль оси y_b; $\tilde{r}$ – измеренная угловая скорость вдоль оси z_b; p – истинная угловая скорость вдоль оси x_b; q – истинная

угловая скорость вдоль оси y_b; $r -$ истинная угловая скорость вдоль оси z_b; $\beta -$ отклонение измерения; $X -$ случайная переменная «белый шум»; $\sigma -$ стандартное отклонение; $\mu -$ среднее отклонение; $t -$ время.

$$X \sim N(\mu, \sigma^2) \tag{51}$$

$$X_p \sim N(0,\ \sigma_p^2) \tag{52}$$

$$X_q \sim N(0,\ \sigma_q^2) \tag{53}$$

$$X_r \sim N(0,\ \sigma_r^2) \tag{54}$$

$$\tilde{p}(t) = p(t) + \beta_p + X_p(t) \tag{55}$$

$$\tilde{q}(t) = q(t) + \beta_q + X_q(t) \tag{56}$$

$$\tilde{r}(t) = r(t) + \beta_r + X_r(t) \tag{57}$$

Чтобы оценить смещение и стандартное отклонение белого шума, IMU выполняли на частоте 100 Гц в течение 10 минут. Результаты были записаны в CSV-файле, используя файл Arduino .pde (приложение B) для записи данных датчика. Затем был запущен скрипт MatLab (приложение A) для расчета характеристик шума и построения записанных измерений гироскопа и смоделированных измерений гироскопа. Оба этих выхода показаны на рисунке 5.

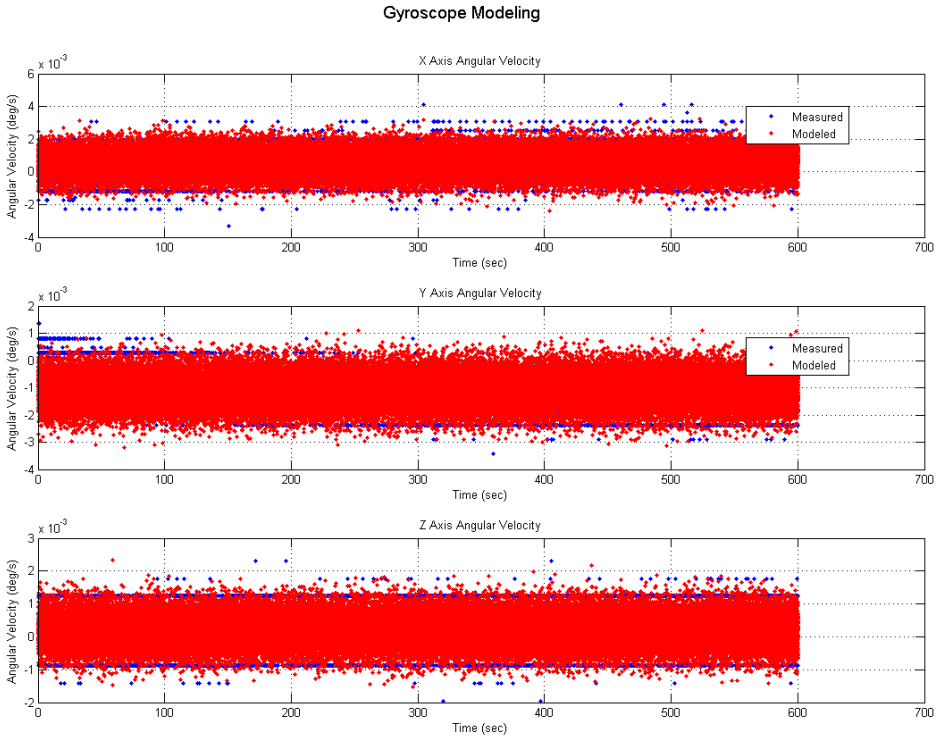

Рисунок 5 – Характеристики шума и построения записанных измерений гироскопа и смоделированных измерений гироскопа

Из графиков на рисунке 5 видно, что записанные измерения совпадают с моделированными значениями, а значит, модель измерений гироскопа можно считать верной.

3.3 Моделирование акселерометра

Акселерометр MEMS содержит торсионные рычаги, прикрепленные к небольшой пластине. Пластина вращается под ускорением, что изменяет емкость между пластиной и окружающими стенками. Если акселерометр совмещён с осью x, y и z четырехротора, то выход акселерометра представляет собой линейное ускорение в корпусе.

Акселерометр может быть смоделирован как комбинация истинной угловой скорости, смещения нуля и плотности шума. Истинное линейное ускорение измеряется в метрах в секунду в квадрате. Шум характеризуется

как нулевой средний белый Гауссовский шум. Смещение зависит от температуры и может быть оценено для каждого полёта. Эти элементы создают следующие уравнения для моделирования гироскопа. Эта же модель используется для каждой оси.

В формулах 58-64 $\tilde{\ddot{x}}^b -$ измеренное ускорение вдоль оси x_b; $\tilde{\ddot{y}}^b -$ измеренное ускорение вдоль оси y_b; $\tilde{\ddot{z}}^b -$ измеренное ускорение вдоль оси z_b; $\ddot{x}^b -$ истинное ускорение вдоль оси x_b; $\ddot{y}^b -$ истинное ускорение вдоль оси y_b; $\ddot{z}^b -$ истинное ускорение вдоль оси z_b; $\beta -$ отклонение измерения; $X -$ случайная переменная «белый шум»; $\sigma -$ стандартное отклонение; $\mu -$ среднее отклонение; $t -$ время.

$$X \sim N(\mu, \sigma^2) \tag{58}$$

$$X_x \sim N(0,\ \sigma_x^2) \tag{59}$$

$$X_y \sim N(0,\ \sigma_y^2) \tag{60}$$

$$X_z \sim N(0,\ \sigma_z^2) \tag{61}$$

$$\tilde{\ddot{x}}^b(t) = \ddot{x}^b(t) + \beta_x + X_x(t) \tag{62}$$

$$\tilde{\ddot{y}}^b(t) = \ddot{y}^b(t) + \beta_y + X_y(t) \tag{63}$$

$$\tilde{\ddot{z}}^b(t) = \ddot{z}^b(t) + \beta_z + X_z(t) \tag{64}$$

Характеристики шума акселерометра оценивались так же, как и гироскоп. Измерения регистрировали при 100 Гц с использованием Arduino и после обработки в MatLab. Ниже приведены характеристики шума и сравнение измеренных и смоделированных значений ускорения (рисунок 6). $\beta_x = 0.17,\ \beta_y = 0.32,\ \beta_z = -8.68,\ \sigma_x = 9.40 * 10^{-3},\ \sigma_y = 1.10 * 10^{-2}, \sigma_z = 1.60 * 10^{-2}$.

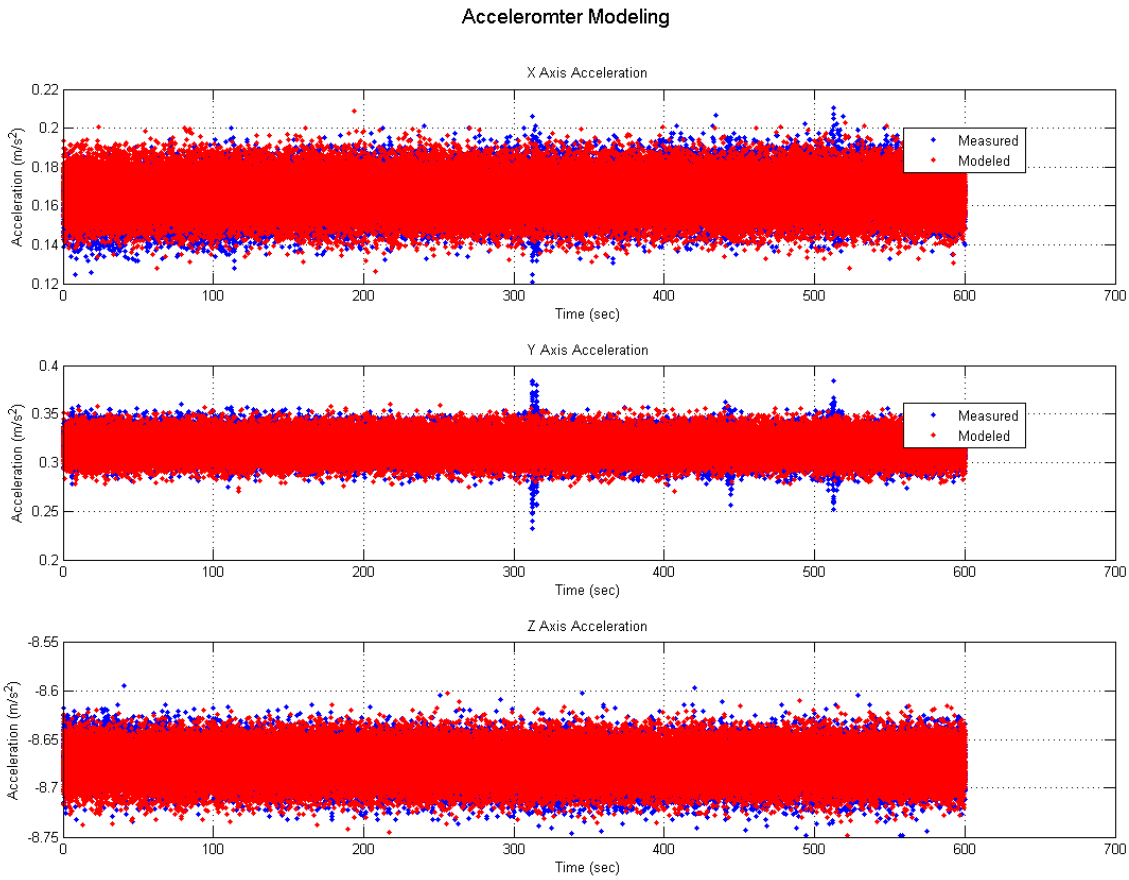

Рисунок 6 – Характеристики шума и сравнение измеренных и смоделированных значений ускорения

Для гироскопа и акселерометра были сделаны упрощения для разработки модели. Во-первых, каждый датчик был неподвижным по мере сбора данных. В действительности ожидается, что эти датчики будут в постоянном движении. Кроме того, датчики измерялись при постоянной температуре. В действительности изменения высоты будут приводить к изменениям температуры, которые будут влиять на отклонения датчика. Оба этих фактора не учитываются в этой простой модели.

3.4 Параметры моторной системы

Двигатели постоянного тока, такие как четыре мотора 12000 Кв, используемые на этом четырёхроторе, популярны у четырёхроторов, потому что они небольшие и с высоким крутящим моментом. Это высокое отношение крутящего момента к весу приводит к высокоэффективному коэффициенту, который требует меньшего веса батареи и большего количества времени полёта между зарядами. Высокие крутящие моменты позволяют двигателям быстрее менять скорость, что, в конечном счёте, контролирует движение четырехротора. Высокий крутящий момент также устраняет необходимость в коробке передач, которая ещё больше снижает вес.

Основные физические принципы, используемые для описания обычного двигателя постоянного тока, могут быть применены к бесколлекторному двигателю постоянного тока. Упрощенное электрическое уравнение (закон Кирхгофа) и динамическое уравнение для двигателя, связанного с нагрузкой (второй закон Ньютона), определены ниже во временной области. Кроме того, обратная электродвижущая сила (ЭДС) пропорциональна угловой скорости двигателя. Закон Фарадея может быть использован для описания напряжения, создаваемого изменяющимся магнитным полем в катушке. Наконец, закон Лоренца, который описывает силу на катушке в магнитном поле, приводит к тому, что созданный крутящий момент пропорционален току двигателя. Эти уравнения показаны ниже во временной области.

В формулах 65-80 PWM_{app} – используемый ШИМ сигнал; u_{emf} – обратная ЭДС; i – ток двигателя; R – сопротивление катушек и обмоток; L – индуктивность катушек; J – инерционная нагрузка; K_e – постоянная ЭДС; K_τ – постоянная якоря; K_f – постоянная трения двигателя; ω – скорость мотора.

$$PWM_{app}(t) = Ri(t) + L\frac{di(t)}{dt} + u_{emf}(t) \quad (65)$$

$$J\frac{d\omega}{dt} = \sum \tau_i \quad (66)$$

$$u_{emf}(t) = K_e\omega(t) \quad (67)$$

$$\tau(t) = K_\tau i(t) \quad (68)$$

$$PWM_{app}(t) = Ri(t) + L\frac{di(t)}{dt} + K_e\omega(t) \quad (69)$$

$$J\frac{d\omega}{dt} = K_\tau i(t) - K_f\omega(t) \quad (70)$$

3.5 Вычисление динамики двигателя

Чтобы смоделировать динамику системы двигателя, необходимо рассчитать передаточную функцию, описывающую динамику двигателя. Выходной передаточной функцией является скорость пропеллера, а входной - напряжение двигателя. Передаточная функция позволит нам вычислить постоянную времени двигателя и коэффициент усиления постоянного тока. В приведенных ниже уравнениях уравнения преобразуются из временной области в частотную область с использованием преобразования Лапласа.

$$PWM_{app}(s) = (sL + R)I(s) + K_e\omega(s) \quad (71)$$

$$K_\tau I(s) = \left(Js + K_f\right)\omega(s) \quad (72)$$

$$G(s) = \frac{\omega(s)}{PWM_{app}(s)} = \frac{K_\tau}{(sL+R)\left(Js+K_f\right)+K_eK_\tau} \quad (73)$$

$$G(s) = \frac{K_\tau}{JLs^2+\left(RJ+K_fL\right)s+RK_f+K_eK_\tau} \quad (74)$$

$$RJ \gg K_fL \quad (75)$$

$$K_eK_\tau \gg RK_f \quad (76)$$

$$G(s) = \frac{K_\tau}{JLs^2+RJs+K_eK_\tau} \quad (77)$$

Где $L \approx 0$.

$$G(s) = \frac{K_\tau}{RJs + K_e K_\tau} = \frac{\frac{1}{K_e}}{\frac{RJ}{K_e K_\tau}s + 1} \quad (78)$$

Усиление мотора:

$$K_{DC} = \frac{1}{K_e} \quad (79)$$

Постоянная времени:

$$\tau = \frac{RJ}{K_e K_\tau} \quad (80)$$

Сделано несколько предположений о снижении передаточной функции для приближения системы первого порядка. Сила, обусловленная трением в двигателе без сердечника постоянного тока, в основном обусловлена сопротивлением подшипника и, следовательно, очень мала. Кроме того, поскольку двигатель мал, можно предположить, что индуктивность очень мала и пренебречь этим элементом. Это приводит к передаточной функции описанной выше. Кроме того, вычисляются уравнения для постоянной времени и коэффициента усиления постоянного тока.

Вместо того чтобы пытаться измерить отдельные переменные, был разработан эксперимент для измерения реакции двигателя и оценки параметров двигателя. Входной передаточной функцией является напряжение двигателя, а выход – скорость вала двигателя. Мы можем контролировать вход двигателя с помощью ESC (Electronic Stability Control – Система Курсовой Устойчивости) и измерять скорость пропеллера с помощью фотоинтерпретатора (фотодатчик типа передачи, который обычно состоит из светоизлучающих элементов и светопринимающих элементов, выровненных друг против друга в одной упаковке, которая работает путем обнаружения световой блокировки, когда целевой объект находится между обоими элементами, действуя как оптический переключатель).

Arduino Uno использовался для управления входным сигналом двигателя и записи скорости пропеллера. Это популярный микроконтроллер на базе ATmega328Pchipset. Он включает в себя несколько цифровых и

аналоговых входов и выходов, а также USB-соединение. Идея состоит в том, чтобы посылать переменные входы ESC с помощью штыря ШИМ (Широтно-импульсная модуляция), который затем преобразуется в напряжение к двигателю. Фотоинтерпретатор использует светодиод и оптический приемник для создания простой схемы. Когда объект проходит между светодиодом и приемником, электрическая цепь разрушается. Можно использовать этот датчик для обнаружения пропеллера каждый раз, когда он разрушает контур, записывает метку времени и вычисляет скорость пропеллера. Эти данные затем могут быть проанализированы с использованием сценария MatLab для оценки постоянного коэффициента усиления и постоянной времени и проверки модели двигателя относительно записанных измерений.

Следует отметить размер выталкивающего резистора. Поскольку будет считываться вход фотоинтерпретатора на цифровом входном штыре, этот резистор помогает вывести напряжение цепи вниз на землю, когда цепь разомкнута, создавая более чёткое различие между открытым и замкнутым контуром, записанным как 0 или 1. На рисунке 7 показана настройка электроники.

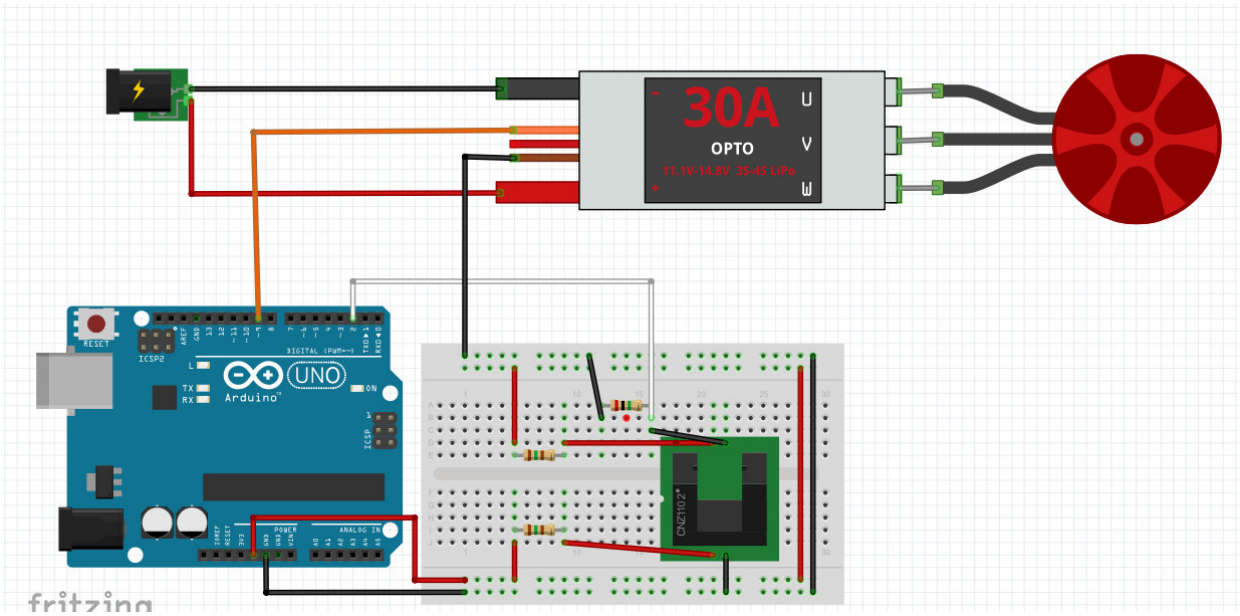

Рисунок 7 – Схема подключения

Хотя использовались некоторые предположения для упрощения уравнений двигателя, фактическая система нелинейна из-за аэродинамических сил сопротивления на пропеллере. Таким образом, вышеприведенная модель известна как линейная инвариантная система времени. Поскольку уравнения линеаризуются, модель будет точной только вокруг заданной точки. Настраиваемая точка выбрана для точки линеаризации. Поэтому, когда мы отправляем моторные команды в нашем эксперименте, мы будем хранить значения вблизи входного заданного значения.

Из-за сделанной ранее калибровки двигателей имеем, что минимальное значение дроссельной заслонки составляет около 1000 микросекунд, а максимальное значение дроссельной заслонки составляет около 2000 микросекунд. При полёте четырехротора в ручном режиме с нависанием, ввод заданного значения составляет около 1425 микросекунд. Затем сценарий Arduino отправляет команды двигателя около 1425 в течение каждых 5 секунд. Прерывание используется для измерения времени, когда пропеллер проходит через фотоинтерпретатор и разрывает цепь. Результатом этого эксперимента является CSV-файл со столбцами, содержащими временную метку, входной сигнал ШИМ и скорость пропеллера.

Затем используется скрипт MatLab (приложение А) для ввода CSV-файла и анализа данных. Передаточная функция, выведенная в формуле, показанной выше, с постоянным коэффициентом усиления и постоянной времени в качестве переменных. Вход в передаточную функцию – это набор команд ШИМ, отправленных фактическому двигателю в эксперименте. Выходной передаточной функцией является соответствующие имитированные обороты пропеллера в минуту. Эти значения строятся относительно экспериментальных значений для определения неизвестных параметров системы. Используя постоянную времени 0.15 секунды и усиление постоянного тока 6.5, моделируемые значения оборотов в минуту

близко соответствуют экспериментальным значениям, выполним проверку модели (рисунок 8).

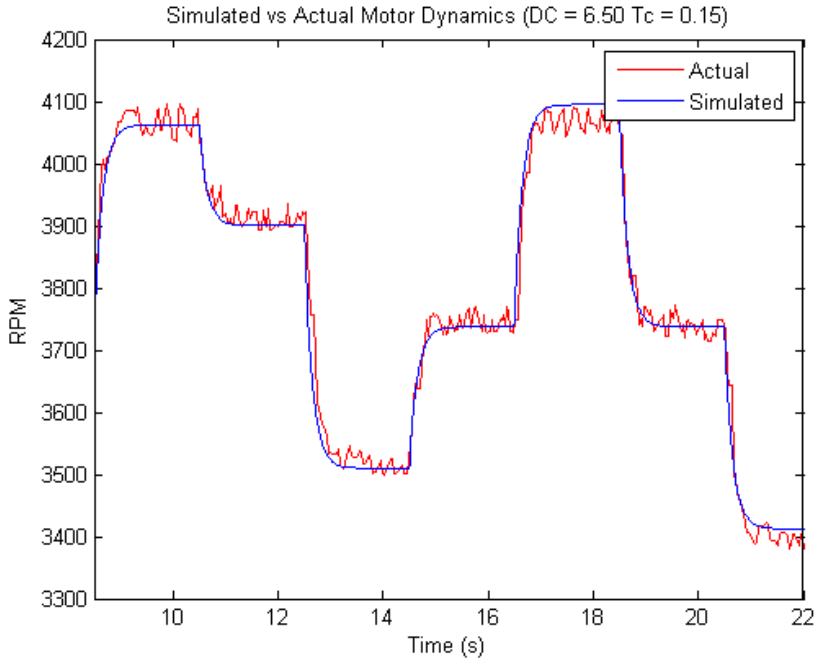

Рисунок 8 – Проверка модели

Чтобы реализовать модель двигателя с использованием фактических входных данных ШИМ, передаточная функция должна была быть преобразована из непрерывного времени в дискретную модель времени. Это было сделано с помощью команды MatLab c2d. Использовали аппроксимацию порядка нулевого порядка, а также время выборки ШИМ 30 Гц, рассчитанное по записанным данным полёта. Расчеты для преобразования модели из частотной области во временную область и вычисления окончательного разностного уравнения показаны ниже. Преобразование z используется для вычисления разностного уравнения. Преобразование z используется в цифровых системах управления, аналогичных тому, как преобразование Лапласа использовалось ранее в системах непрерывного контроля. Разностное уравнение используется для

оценки скорости двигателя в реальном времени с учетом текущих значений ШИМ.

В формулах 81-84 PWM_{app} – используемый ШИМ сигнал; ω – скорость мотора; τ – постоянная времени; K_{DC} – усиление постоянного тока.

$$G(s) = \frac{\omega(s)}{PWM_{app}(s)} = \frac{K_{DC}}{\tau s+1} = \frac{6.5}{.15s+1} \quad (81)$$

$$G(z) = \frac{\omega(z)}{PWM_{app}(z)} = \frac{1.24}{z-.813} \quad (82)$$

$$PWM_{app}(z)(1.214) = \omega(z)(z - .813) \quad (83)$$

$$\omega(k) = .813 * \omega(k-1) + 1.214 * PWM_{app}(k-1) \quad (84)$$

3.6 Коэффициент тяги

Коэффициент тяги можно оценить с использованием параметров двигателя, определенных выше. Напомним, что коэффициент тяги является пропорциональной константой, которая связывает тягу каждого двигателя с квадратом угловой скорости пропеллера. Это проще всего рассчитать в точке нависания, поскольку известно, что тяга каждого двигателя равна четверти массы четырехротора, и можно оценить скорость пропеллера из ранее собранных данных. Ниже приведен расчет коэффициента тяги.

В формулах 85-87 T_h – сила тяги зависания; ω_h – угловая скорость двигателя при зависании; K_T – коэффициент силы тяги; m – масса квадрокоптера; g – ускорение силы тяжести.

$$T_h = K_T \omega_h^2 \quad (85)$$

$$\frac{mg}{4} = K_T \omega_h^2 \quad (86)$$

$$K_T = \frac{mg}{4\omega_h^2} = \frac{1.5*9.8}{4*(392)^2} \quad (87)$$

$$K_T = 2.39 * 10^{-5}$$

3.7 Максимальная скорость двигателя

Чтобы повысить точность моделирования, нужно ввести ограничения на скорость двигателя. Это не позволит системе управления вычислять управляющие сигналы, которые не реалистичны для реальной четырёхроторной системы. Установка пределов на скорость вращения двигателя также помогает в настройке контрольных коэффициентов усиления и предотвращает чрезмерное использование контрольных коэффициентов. Команды ШИМ создают рабочий цикл, который может быть соотнесен с количеством напряжения, посылаемого двигателю. Из предыдущих экспериментов известно, что четырехротор колеблется вблизи команды ШИМ 1425, которая соответствует напряжению около 5,35 вольт. Для данного сигнала ШИМ можно оценить, что пропеллер вращается со скоростью около 5745 оборотов в минуту. Из технической документации двигателя известно, что постоянная скорость без нагрузки (Kv) составляет 14000 об./мин. Загруженное Kv – 10000. Из технической документации мотора можно узнать, что каждый мотор производит 42 г тяги 1A и 4.2 Вт при полной мощности, что составляет 4.2 вольт. Поэтому можно ожидать, что пропеллер будет вращаться не быстрее, чем 9350 об./мин.

3.8 Оценка параметров с помощью моделирования

Первая часть идентификатора системы попыталась определить общую задержку системы четырехротора от входа RC в положение системы. Чтобы вычислить, сколько времени требуется, чтобы четырехротор получил команду и переместился в нужную позицию. Для этого сравниваем RC-входы (желаемый крен, тангаж, скорость рыскания и тягу) с креном, тангажем, скоростью рыскания и вертикальным ускорением, которые ожидаются. Затем можно изменить временные метки входа RC, чтобы определить, на каком смещении входы RC лучше всего выравниваются с ожидаемыми значениями.

Это смещение является расчетной задержкой в системе четырехротора. Ниже, на рисунках 9-12, приведены графики этого сравнения.

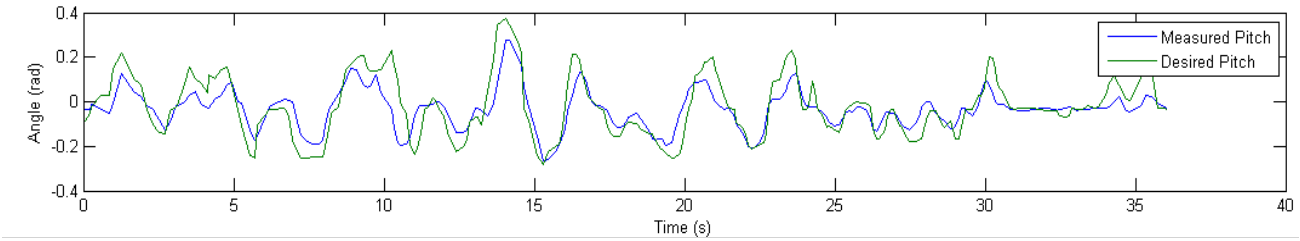

Рисунок 9 – Желаемое и ожидаемое значение тангажа

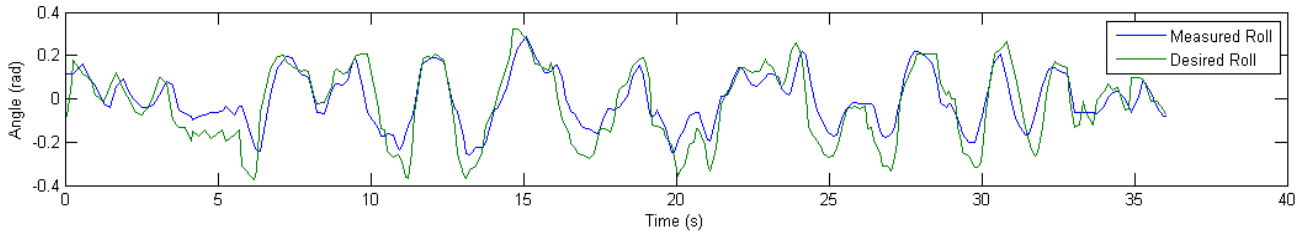

Рисунок 10 – Желаемое и ожидаемое значение крена

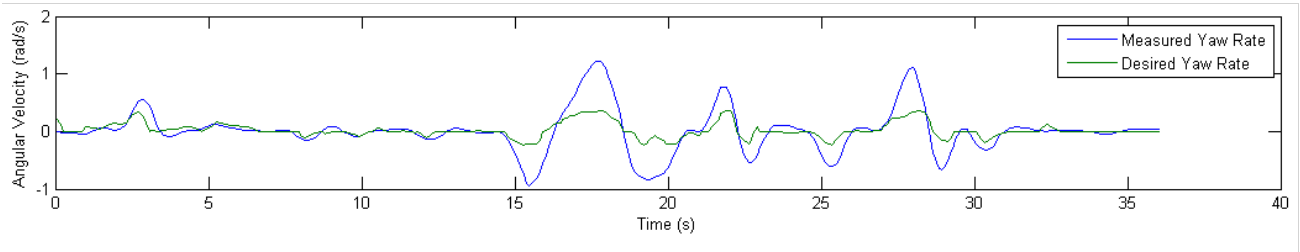

Рисунок 11 – Желаемое и ожидаемое значение рыскания

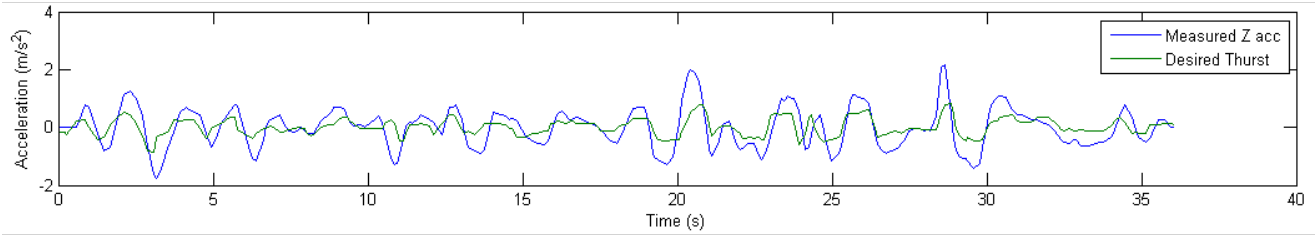

Рисунок 12 – Желаемое и ожидаемое значение тяги

Затем исследовали уравнения движения ускорения перемещения. Использовались уравнения движения, разработанные в разделе 2. Простая

регрессия выполнялась на выходе модели для расчета наилучших коэффициентов соответствия. Углы IMU и расчетные скорости вращения двигателя были входом в модель, а также оценочными коэффициентами. Из приведенных ниже графиков (рисунки 13-15) видно, что модель достаточно хорошо согласуется с ожидаемыми ускорениями.

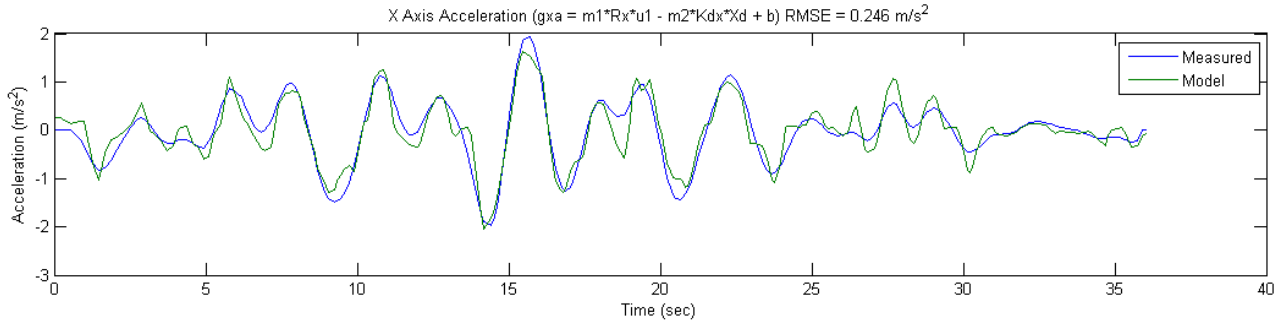

Рисунок 13 – Смоделированные и ожидаемые ускорения по оси X

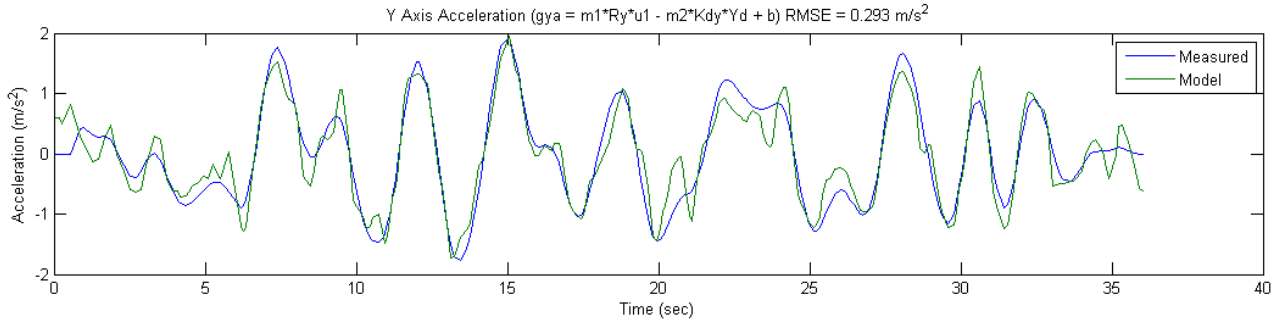

Рисунок 14 – Смоделированные и ожидаемые ускорения по оси Y

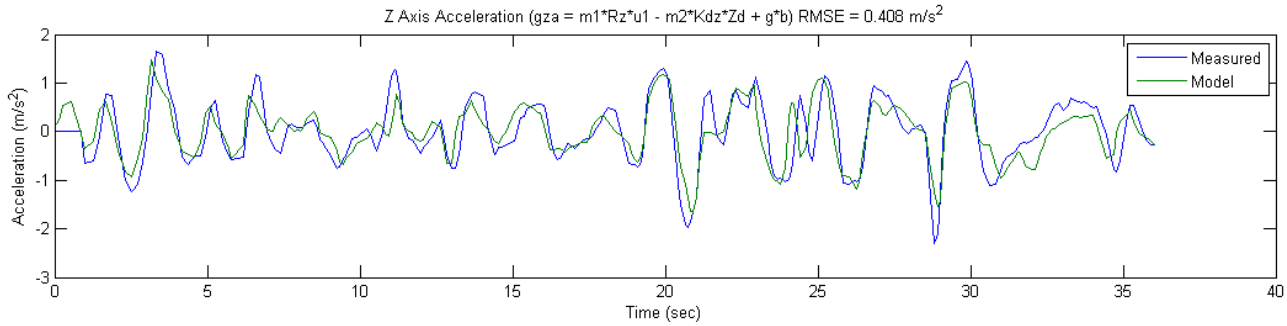

Рисунок 15 – Смоделированные и ожидаемые ускорения по оси Z

Наконец, провели аналогичные вычисления на уравнениях вращательного ускорения движения. Опять же, измерения IMU и вычисленные скорости пропеллера были введены в уравнения, разработанные в разделе 2. На выходе модели была проведена регрессия, сравнивающая значения с ожидаемыми для определения коэффициента. К сожалению, поскольку большинство коэффициентов в вращательной динамике связаны вместе, невозможно оценить отдельные неизвестные параметры. Однако результаты являются разумными, и общие коэффициенты могут использоваться из модели для проектирования контроллера. Ниже изображены графики смоделированных и ожидаемых угловых ускорений (рисунки 16-18).

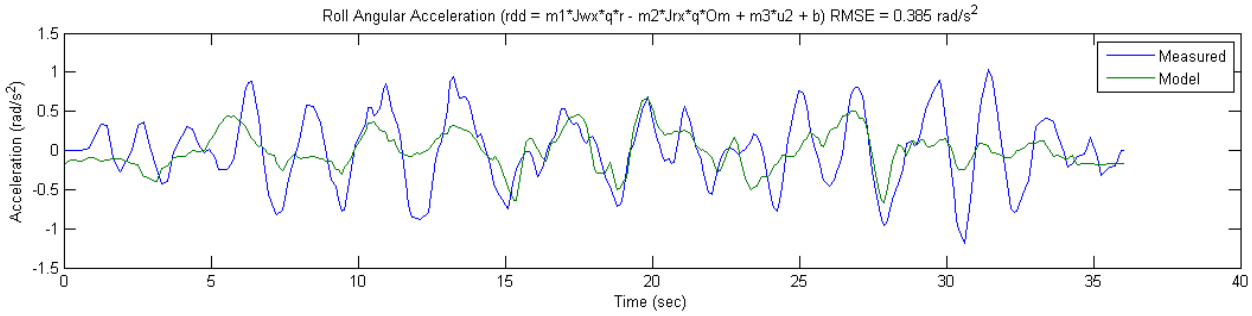

Рисунок 16 – График смоделированного и ожидаемого углового ускорения по крену

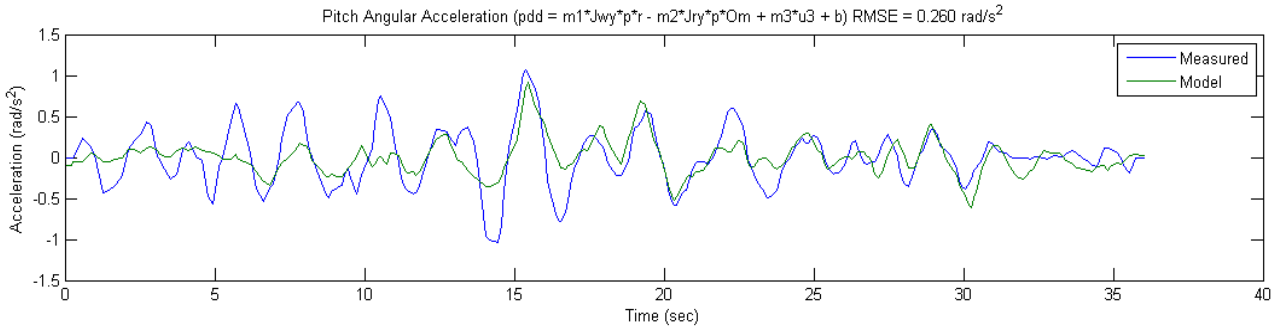

Рисунок 17 – График смоделированного и ожидаемого углового ускорения по тангажу

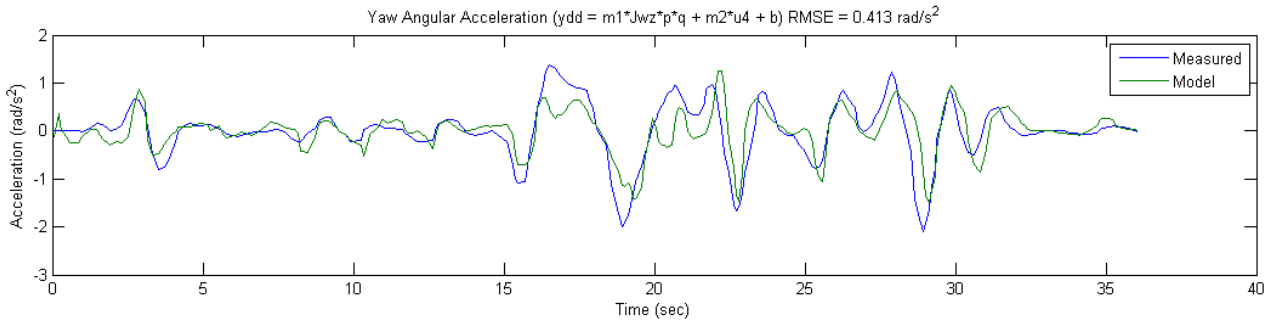

Рисунок 18 – График смоделированного и ожидаемого углового ускорения по рысканию

Эта модель фиксирует нелинейную динамику четырёхроторной системы. Чтобы использовать эту модель для моделирования контроллера, необходимо записать уточненные параметры системы:

$K_T = 1.33 * 10^{-5}$ (коэффициент тяги);

$K_d = 1.39 * 10^{-6}$ (постоянная пропорциональности крутящего момента);

$K_{dx} = 0.164$ (постоянная пропорциональности крутящего момента);

$K_{dy} = 0.139$ (постоянная пропорциональности крутящего момента);

$K_{dz} \approx 0$ (постоянная пропорциональности крутящего момента);

$J_x = 0.05$ (момент инерции четырехротора в x_b);

$J_y = 0.05$ (момент инерции четырехротора в y_b);

$J_z = 0.24$ (момент инерции четырехротора в z_b);

$J_r = 0.013$ (момент инерции ротора).

3.9 Общая стратегия управления

Четырёхротор – это классическая малоэффективная система. Четырёхротор способен перемещаться в 6 степенях свободы (3 поступательных и 3 вращательных), но имеет только 4 управляющих входа (скорости каждого двигателя). Хотя четырехротор может перемещаться непосредственно по вертикальной оси z без изменения какого-либо другого состояния, он должен изменить свое отношение к перемещению по

горизонтальным осям x и y. Поскольку невозможно управлять 6 степенями свободы только с четырьмя управляющими входами, то вместо этого проектируем контроллеры для стабилизации вокруг желаемых положений x, y, z и желаемого направления. Четырехротор должен иметь возможность безопасно перемещаться в нужном положении, сохраняя при этом устойчивые углы крена и тангажа. Современные четырёхроторы имеют электронные регуляторы скорости (ESC), которые управляют угловыми скоростями каждого ротора [42]. Это приводит к следующим входам прямого управления:

1) u(1) – результирующая тяга четырех роторов;

2) u(2) – разность тяги между двигателями на оси x, что приводит к изменению угла наклона крена и последующему движению в боковом направлении x;

3) u(3) – разность тяги между двигателями на оси y, что приводит к изменению угла тангажа и последующему движению в боковом направлении y;

4) u(4) – разность крутящего момента между роторами по часовой стрелке и против часовой стрелки, что приводит к моменту, который вращает четырехротор вокруг вертикальной оси z.

Мы можем использовать приравнивание для уравнений силы тяги и момента, чтобы определить управляющие входы в четырёхроторную систему. Управляющие входы определены ниже [42]. Нужно обратить внимание, что инвертирование этой матрицы позволяет найти для желаемых скоростей двигателя заданные значения для четырех управляющих входов. Поэтому, если можно вычислить управляющие входы, то можно сопоставить эти значения с требуемыми скоростями для каждого двигателя. Затем эти скорости могут быть отправлены на ESC.

В формулах 88-100 F_T^b – общая сила тяги в системе координат тела; ω – угловая скорость мотора; $\tau_{\phi,\theta,\psi}$ – крутящий момент по осям тела; K_d – константа пропорциональности крутящего момента; K_T – коэффициент силы

тяги; $\ell -$ длина плеча; $u_1 -$ общая тяга всех моторов; $u_2 -$ входной сигнал крена; $u_3 -$ входной сигнал тангажа; $u_4 -$ входной сигнал рыскания.

$$\begin{bmatrix} F_T^b \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T & K_T \\ 0 & -\ell K_T & 0 & \ell K_T \\ -\ell K_T & 0 & -\ell K_T & 0 \\ K_d & -K_d & K_d & -K_d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \quad (88)$$

Общая стратегия управления будет опираться на 3 подсистемы управления. Датчики на борту четырехротора измеряют угловую скорость и поступательное ускорение. Эти значения сравниваются с желаемым отношением и позициями для расчета сигнала ошибки, который подается в систему управления. В первой подсистеме ошибкой позиции является входной сигнал, и требуемые углы крена и тангажа являются выходами. Затем в подсистему ориентации вводятся ошибки крена, тангажа и рыскания. Эта система выводит желаемые угловые скорости для каждой оси. Они подаются в конечную систему управления. Выходы этой системы – это управляющие входы, которые затем могут быть отображены на желаемые скорости вращения двигателя. Эта общая стратегия может быть использована для разработки систем управления с разной степенью сложности.

3.10 Упрощенные уравнения движения

Сложная модель движения четырехротора затрудняет точное понимание поведения четырехротора и устранение неисправностей в системах управления в симуляции. Для этого исследования рассматриваются только основные элементы, описывающие поведение четырехротора при зависании, желаемое рабочее состояние. Элементы, которые существенно влияют на поведение четырехротора на высоких скоростях, игнорируются. Можно сделать несколько предположений, которые уменьшают сложность описанных выше нелинейных уравнений для разработки основных стратегий управления. Затем можно будет повысить сложность системной модели для

разработки более точных систем управления в будущем. Модель, полученная в разделе 2, воспроизводится ниже с введенными управляющими входами.

$$\begin{bmatrix}\ddot{X}^G\\ \ddot{Y}^G\\ \ddot{Z}^G\end{bmatrix}=\begin{bmatrix}\frac{1}{m}(-[c(\phi)c(\psi)s(\theta)+s(\phi)s(\psi)]u_1-K_{dx}\dot{X}^G)\\ \frac{1}{m}(-[c(\phi)s(\psi)s(\theta)-c(\psi)s(\phi)]u_1-K_{dy}\dot{Y}^G)\\ \frac{1}{m}\left(-[c(\phi)c(\theta)]u_1-K_{dz}\dot{Z}^G\right)+g\end{bmatrix} \tag{89}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\begin{bmatrix}1 & s(\phi)t(\theta) & c(\phi)t(\theta)\\ 0 & c(\phi) & -s(\phi)\\ 0 & \frac{s(\phi)}{c(\theta)} & \frac{c(\phi)}{c(\theta)}\end{bmatrix}\begin{bmatrix}p\\ q\\ r\end{bmatrix} \tag{90}$$

$$\begin{bmatrix}\ddot{\phi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix}=\begin{bmatrix}\frac{1}{J_x}\left(J_y-J_z\right)qr-J_rq(\omega_1-\omega_2+\omega_3-\omega_4)+u_2\\ \frac{1}{J_y}(J_z-J_x)pr-J_rp(\omega_1-\omega_2+\omega_3-\omega_4)+u_3\\ \frac{1}{J_z}[\left(J_x-J_y\right)pq+u_4]\end{bmatrix} \tag{91}$$

Основное предположение, сделанное для упрощения модели, заключается в том, что четырехротор будет работать вокруг стабильного нависания с малыми углами ориентации и минимальными вращательными и поступательными скоростями и ускорениями. Поскольку нет аэродинамических подъемных поверхностей, будем считать, что аэродинамические силы и моменты незначительны. Эффекты, которые считаются пренебрежимо малыми, рассматриваются, как нарушения в системе управления и могут быть компенсированы соответствующей системой управления. Эти предположения описываются математически, а также полученными упрощенными уравнениями движения. Уравнения, приведенные ниже, продолжают уменьшаться в сложности, поскольку делается больше предположений.

$$\dot{X}^G=\dot{Y}^G=\dot{Z}^G=0 \tag{92}$$

$$\dot{\psi}=\dot{\theta}=\dot{\phi}=0 \tag{93}$$

$$\phi=\theta=0 \tag{94}$$

$$c(\phi)\approx c(\theta)\approx 1 \tag{95}$$

$$s(\phi) \approx s(\theta) \approx 0 \tag{96}$$

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{97}$$

$$\begin{bmatrix} \ddot{X}^G \\ \ddot{Y}^G \\ \ddot{Z}^G \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-[c(\phi)c(\psi)s(\theta) + s(\phi)s(\psi)]u_1) \\ \frac{1}{m}(-[c(\phi)s(\psi)s(\theta) - c(\psi)s(\phi)]u_1) \\ \frac{1}{m}(-[c(\phi)c(\theta)]u_1) + g \end{bmatrix} \tag{98}$$

Где $\psi = 0, \mathrm{c}(\psi) = 1, s(\psi) = 0$.

$$\begin{bmatrix} \ddot{X}^G \\ \ddot{Y}^G \\ \ddot{Z}^G \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(-[\theta c(\psi) + \phi s(\psi)]u_1) \\ \frac{1}{m}(-[\theta s(\psi) - \phi c(\psi)]u_1) \\ -\frac{1}{m}(u_1) + g \end{bmatrix} = \begin{bmatrix} -\frac{1}{m}\theta u_1 \\ \frac{1}{m}\phi u_1 \\ -\frac{1}{m}(u_1) + g \end{bmatrix} \tag{99}$$

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{J_x}(u_2) \\ \frac{1}{J_y}(u_3) \\ \frac{1}{J_z}(u_4) \end{bmatrix} \tag{100}$$

Хотя эти уравнения упрощают процесс проектирования и анализа управления, все же можно использовать более точные нелинейные уравнения движения в моделируемой среде для оценки надежности разрабатываемой системы управления. Стоит обратить внимание, что в окончательных уравнениях поступательное движение было отделено от положения, предполагая, что направление остается около 0 градусов. Движение вдоль глобальной оси x и y можно контролировать, независимо изменяя значения крена и тангажа. Кроме того, для того, чтобы четырехротор поддерживал устойчивую высоту, результирующий вектор тяги, деленный на массу четырехротора, должен быть равен ускорению силы тяжести. На приведенных ниже графиках (рисунки 19-24) сравнивается упрощенная модель, полученная выше, с измерениями сложной модели с основополагающими измерениями, описанными в разделах 3.1-3.7.

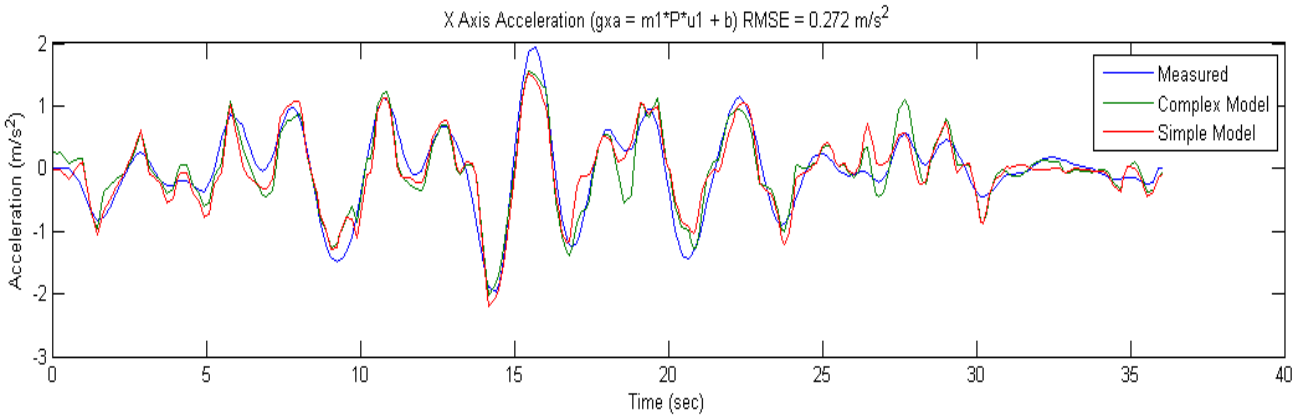

Рисунок 19 – График ускорения по оси X

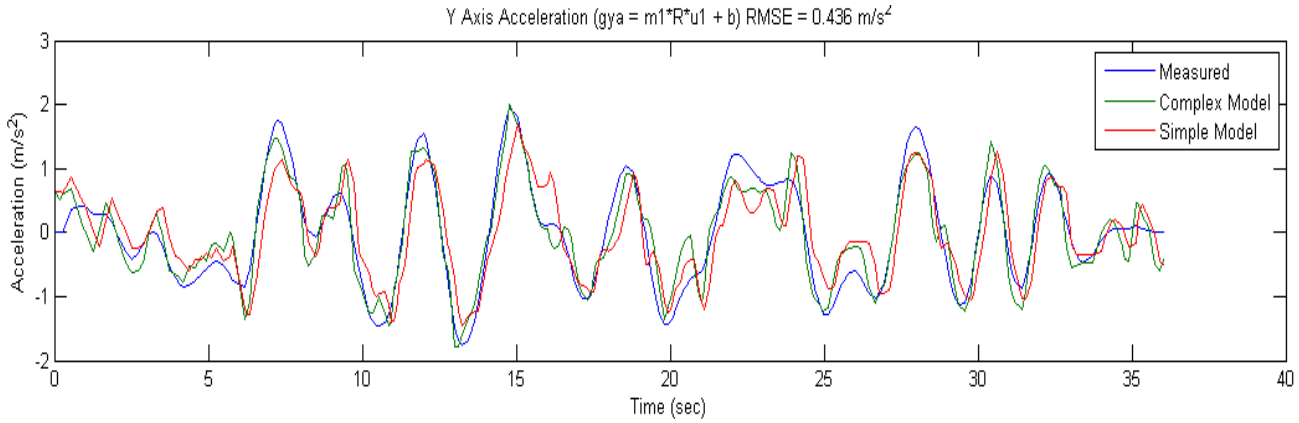

Рисунок 20 – График ускорения по оси Y

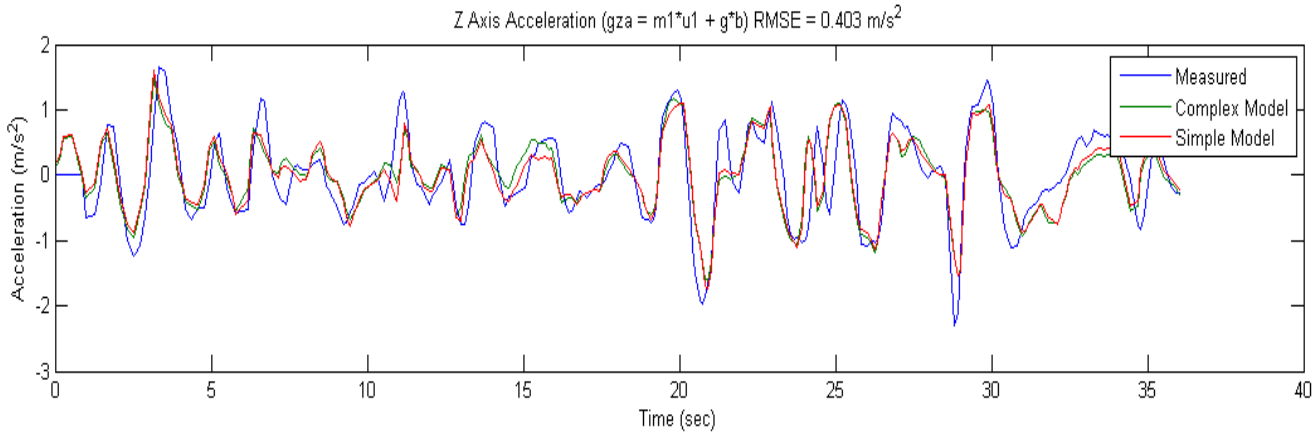

Рисунок 21 – График ускорения по оси Z

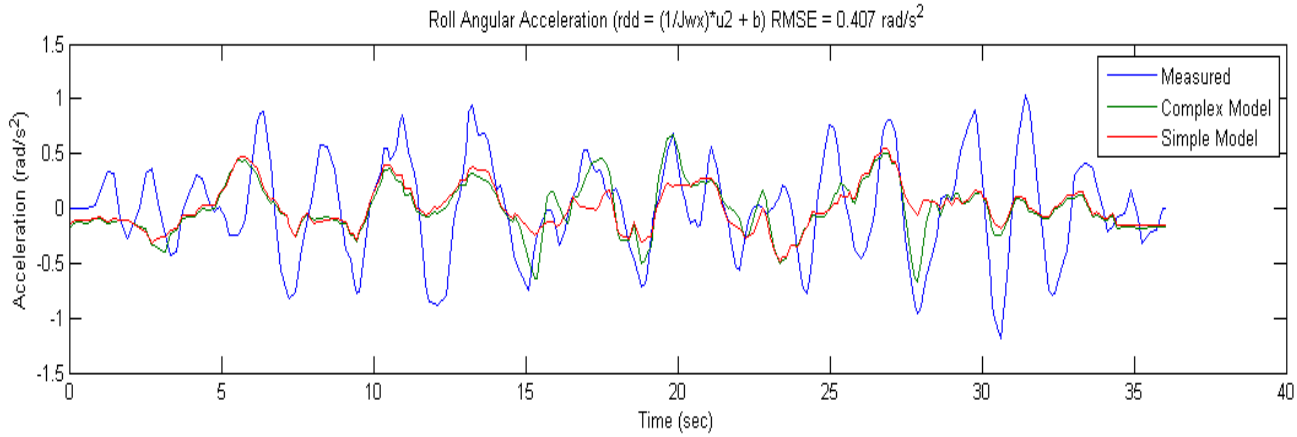

Рисунок 22 – График углового ускорения по крену

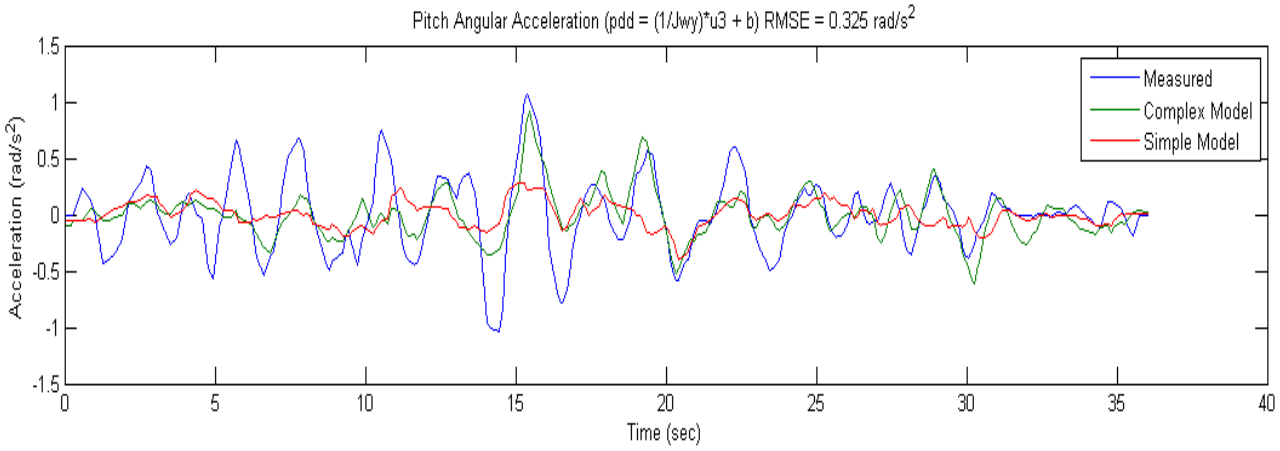

Рисунок 23 – График углового ускорения по тангажу

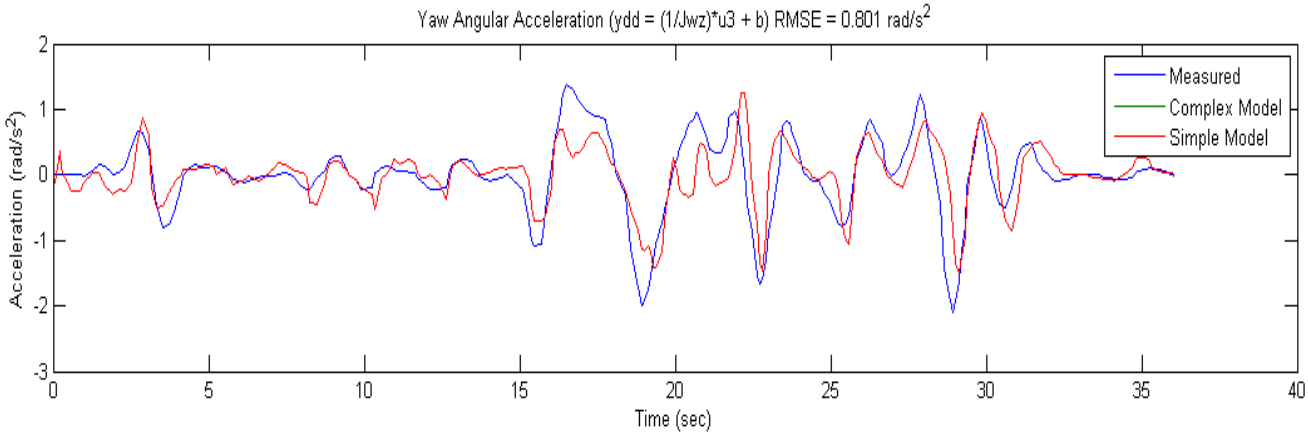

Рисунок 24 – График углового ускорения по рысканию

Несмотря на то, что линеаризованная модель очень точна, как нелинейная модель, она хорошо сравнивается с основополагающими измерениями. Предположения, необходимые для линеаризации модели, проверяются, и модель может использоваться в качестве основы для проектирования системы управления.

3.11 Вложенные циклы управления

Управление четырёхроторами часто реализуется с использованием вложенных циклов управления. Самый внутренний цикл управляет угловыми скоростями каждой оси четырехротора. Этот цикл должен работать на высокой частоте из-за быстрой динамики четырехротора. Следующий самый внешний цикл управляет ориентацией и высотой четырехротора. Небольшие изменения положения аппарата в воздухе напрямую связаны с поступательным ускорением. Это означает, что небольшие ошибки положения могут привести к большим и нежелательным поступательным смещениям очень быстро. Одно из преимуществ запуска цикла регулирования скорости на высокой частоте заключается в том, что встроенные датчики, такие как акселерометр и гироскоп, также работают на высокой частоте. Это означает, что можно использовать самые последние и точные измерения при расчете контрольных значений. Этот регулятор высоты или положения может получать желаемую высоту и положение аппарата в воздухе от пульта дистанционного управления или от внешнего цикла управления поступательным ускорением.

Ниже, на рисунке 25, приведена блок-схема, демонстрирующая эти вложенные циклы управления и систему четырёхроторных установок.

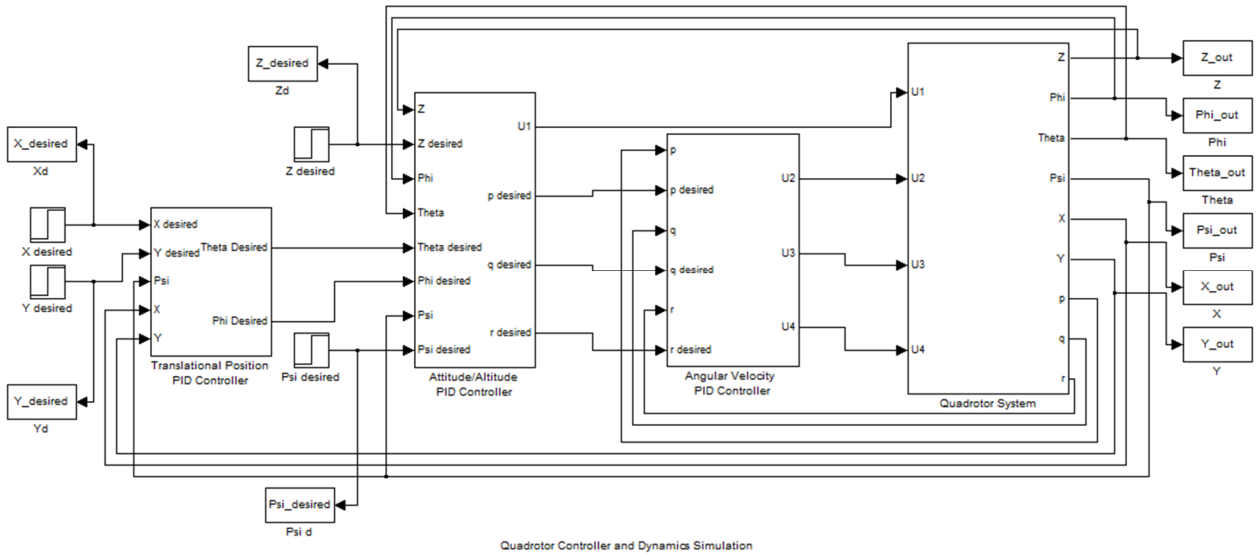

Рисунок 25 – Схема управления четырёхротором

Внешний цикл управления контролирует поступательное движение вдоль оси x и y. Этот цикл управления выводит желаемые углы крена и тангажа для внутреннего цикла управления на основе ошибки в оси x и y. Для этого внешнего цикла управления есть несколько входов. Это могут быть координаты маршрута, созданные на наземной станции управления или, например, алгоритм планирования маршрута.

3.12 Управление положением

В то время как контроллер положения или высоты может поддерживать стабильный полёт для четырехротора и отслеживать желаемые положение или высоту от RC или GCS, он не может контролировать поступательное движение четырехротора. Независимо от того, насколько точным может быть контроллер положения, малые углы поворота приведут к поступательному движению или дрейфу четырехротора. Чтобы предотвратить это поступательное дрейфующее поведение, реализован контроллер положения. Для того чтобы этот контроллер функционировал, необходимо выполнить некоторую относительную или глобальную локализацию. Обычно это делается с камерой для положения телесного кадра

(визуальный сервомотор) или GPS для глобального позиционирования в системе координат. Этот внешний контроллер может работать с меньшей скоростью по сравнению с контроллером внутреннего цикла. Входы в контроллер положения представляют собой измерения положения, а выходы – требуемые углы крена и тангажа для маневра четырехротора в желаемое положение x и y. Этот контроллер также обеспечивает траекторию следования, где последовательность последовательных координат может быть подана на контроллер положения, чтобы заставить четырехротор следовать конкретному трехмерному пути.

В формулах 101-102 $k_p -$ пропорциональное управляющее усиление; $k_I -$ интегральное управляющее усиление; $k_D -$ дифференциальное управляющее усиление; $X_d^G -$ желаемое глобальное положение X; $Y_d^G -$ желаемое глобальное положение Y; $\phi_d -$ желаемый угол крена; $\theta_d -$ желаемый угол тангажа.

$$\theta_d(t) = k_p + \left(X_d^G - X^G\right) + k_I \int_0^t \left(X_d^G - X^G\right)dt + k_D(\dot{X}_d^G - \dot{X}^G) \qquad (101)$$

$$\phi_d(t) = k_p + \left(Y_d^G - Y^G\right) + k_I \int_0^t \left(Y_d^G - Y^G\right)dt + k_D(\dot{Y}_d^G - \dot{Y}^G) \qquad (102)$$

3.13 Управление положением и высотой

В этом разделе описывается система управления положением и высотой. Первоначально необходимо реализовать этот контроллер, чтобы стабилизировать четырехротор для стабильного зависания. Это означает, что все требуемые углы равны 0, и желаемая высота устанавливается на некоторое низкое значение для настройки. Позже эти требуемые значения положения и высоты могут варьироваться в зависимости от RC или внешних входов контроллера контура [11]. Поскольку разрабатываемая упрощенная модель устранила любую перекрестную связь в уравнениях движения, управление положением может быть реализовано с помощью независимого управляющего входа для каждого угла Эйлера.

Управление высотой аналогично управлению полной тяги четырехротора. Это важно, так как этот вход управления также влияет на движение вдоль оси x и y [11]. Это можно увидеть в поступательных уравнениях движения, где u1 – вход в направлениях x и y. Поэтому более высокие значения регулятора приводят к более резким движениям в направлениях x и y с учетом постоянных углов крена и тангажа. В создаваемой упрощенной модели вертикальное ускорение отделено от угла поворота. Однако это не происходит, когда четырехротор поворачивается, поскольку компоненты вертикального ускорения также находятся в плоскостях x и y. Поэтому управление высотой компенсирует как смещение силы тяжести, так и отклонение обобщенного вектора тяги из-за наклона. Полностью ПИД-регуляторы подробно описаны ниже.

В формулах 103-106 $u_1 -$ общая тяга всех моторов; $p_d -$ желаемая угловая скорость вдоль x^b; $q_d -$ желаемая угловая скорость вдоль y^b; $r_d -$ желаемая угловая скорость вдоль z^b; $k_p -$ пропорциональное управляющее усиление; $k_I -$ интегральное управляющее усиление; $k_D -$ дифференциальное управляющее усиление; $\phi_d -$ желаемый угол крена; $\theta_d -$ желаемый угол тангажа; $\psi_d -$ желаемый угол рыскания; $Z_d^G -$ желаемая высота Z; $m -$ масса квадрокоптера; $g -$ ускорение силы тяжести; $e(t) -$ значение ошибки во времени t.

$$u_1(t) = \frac{1}{c(\phi)c(\theta)}\left(k_p e_z(t) + k_I \int_0^t e_z(t)dt + k_D \frac{e_z(t)-e_z(t-1)}{dt}\right) + mg \quad (103)$$

$$p_d(t) = \left(k_p e_\phi(t) + k_I \int_0^t e_\phi(t)dt + k_D \frac{e_\phi(t)-e_\phi(t-1)}{dt}\right. \quad (104)$$

$$q_d(t) = \left(k_p e_\theta(t) + k_I \int_0^t e_\theta(t)dt + k_D \frac{e_\theta(t)-e_\theta(t-1)}{dt}\right. \quad (105)$$

$$r_d(t) = \left(k_p e_\psi(t) + k_I \int_0^t e_\psi(t)dt + k_D \frac{e_\psi(t)-e_\psi(t-1)}{dt}\right. \quad (106)$$

Следует обратить внимание, что управление рысканием или курсом часто выполняется только с помощью управления ПД-регулятора, поэтому коэффициент усиления интегратора равен 0. Код для реализации этих алгоритмов управления в дискретном времени в дополнение к некоторым изменениям для повышения производительности основного ПИД-регулятора.

3.14 Управление угловой скоростью

Регулятор скорости – это контроллер нижнего уровня, разработанный для системы четырёхроторов. Этот контроллер получает требуемые угловые скорости от контроллера положения. Затем он вычисляет ошибку между желаемыми скоростями и скоростью, измеренной гироскопом [10]. Эта ошибка затем используется для вычисления трех управляющих сигналов. Они объединены с входом управления тягой, вычисленным контроллером высоты, который затем преобразуется в требуемые скорости двигателя и отправляется двигателям через ESC.

$$u_2(t) = (k_p e_p(t) + k_I \int_0^t e_p(t)dt + k_D \frac{e_p(t) - e_p(t-1)}{dt} \quad (107)$$

$$u_3(t) = (k_p e_q(t) + k_I \int_0^t e_q(t)dt + k_D \frac{e_q(t) - e_q(t-1)}{dt} \quad (108)$$

$$u_4(t) = (k_p e_r(t) + k_I \int_0^t e_r(t)dt + k_D \frac{e_r(t) - e_r(t-1)}{dt} \quad (109)$$

В формулах 107-109 $u_2 -$ входной сигнал крена; $u_3 -$ входной сигнал тангажа; $u_4 -$ входной сигнал рыскания; $k_p -$ пропорциональное управляющее усиление; $k_I -$ интегральное управляющее усиление; $k_D -$ дифференциальное управляющее усиление; $p -$ желаемая угловая скорость вдоль оси x^b; $q -$ желаемая угловая скорость вдоль оси y^b; $r -$ желаемая угловая скорость вдоль оси z^b; $e(t) -$ значение ошибки во времени t.

3.15 Блок управления двигателем

С учетом управляющих входов можно вычислить отдельные скорости, необходимые для каждого двигателя. Эти скорости будут отправлены в ESC, который затем отправит команды подключенному двигателю, чтобы заставить его вращаться с требуемой скоростью, чтобы получить необходимые моменты тяги и управления. Желаемые скорости двигателя можно вычислить, взяв обратную матрицу управления, полученную в разделе выше (3.9), как показано ниже (110). Обратите внимание, что все двигатели содержат несколько входных элементов управления тягой и рысканием, а также соответствующие входы для каждой оси. Двигатели, управляющие моментом тангажа, включают в себя компоненты входного сигнала управления тангажа, а двигатели, управляющие моментом крена, включают в себя входные компоненты управления креном. Эти уравнения предназначены для ориентации четырехроторов «+» и должны быть изменены, если четырехротор должен летать в конфигурации «X».

В формулах 110-114 $\omega -$ угловая скорость мотора; $K_d -$ константа пропорциональности вращающего момента; $K_T -$ коэффициент силы тяги; $\ell -$ длина плеча; $u_1 -$ общая тяга всех моторов; $u_2 -$ входной сигнал крена; $u_3 -$ входной сигнал тангажа; $u_4 -$ входной сигнал рыскания.

$$\begin{bmatrix}\omega_1^2\\ \omega_2^2\\ \omega_3^2\\ \omega_4^2\end{bmatrix} = \begin{bmatrix}K_T & K_T & K_T & K_T\\ 0 & -\ell K_T & 0 & \ell K_T\\ \ell K_T & 0 & -\ell K_T & 0\\ K_d & -K_d & K_d & -K_d\end{bmatrix}^{-1} = \begin{bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{bmatrix} \tag{110}$$

$$\omega_1^2 = \frac{u_1}{4K_T} + \frac{u_3}{2\ell K_T} + \frac{u_4}{4K_d} \tag{111}$$

$$\omega_2^2 = \frac{u_1}{4K_T} - \frac{u_2}{2\ell K_T} - \frac{u_4}{4K_d} \tag{112}$$

$$\omega_3^2 = \frac{u_1}{4K_T} - \frac{u_3}{2\ell K_T} + \frac{u_4}{4K_d} \tag{113}$$

$$\omega_4^2 = \frac{u_1}{4K_T} + \frac{u_2}{2\ell K_T} - \frac{u_4}{4K_d} \tag{114}$$

Также важно установить ограничения на управляющие входы, которые могут быть рассчитаны системой управления. Эти управляющие входы преобразуются в скорости двигателя, которые затем отправляются на ESC и в конечном итоге на двигатели [42]. Поэтому нельзя использовать контрольные коэффициенты, которые приводят к желаемым скоростям двигателя, которые невозможны в реальной системе. Вычисления контрольных пределов изображены ниже на основе физической системы.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} K_T & K_T & K_T & K_T \\ 0 & -\ell K_T & 0 & \ell K_T \\ \ell K_T & 0 & -\ell K_T & 0 \\ K_d & -K_d & K_d & -K_d \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{115}$$

$$u_1^{max} = 4K_T\omega_{max}^2 = 43.5 \tag{116}$$

$$u_1^{min} = 4K_T\omega_{min}^2 = 0 \tag{117}$$

$$u_2^{max} = u_3^{max} = \ell K_T\omega_{max}^2 = 6.25 \tag{118}$$

$$u_2^{min} = u_3^{min} = \ell K_T\omega_{min}^2 = -6.25 \tag{119}$$

$$u_4^{max} = 2K_d\omega_{max}^2 = 2.25 \tag{120}$$

$$u_4^{min} = -2K_d\omega_{min}^2 = -2.25 \tag{121}$$

В формулах 115-121 ω – угловая скорость мотора (рад/с); ω_{max} – максимальная скорость мотора (920 рад/с); ω_{min} – минимальная скорость мотора (0 рад/с); K_d – постоянная пропорциональности крутящего момента ($1.39 * 10^{-6}$); K_T – коэффициент силы тяги ($1{,}33 * 10^{-5}$); ℓ – длина плеча (0,59 см); u_1 – общая тяга всех моторов; u_2 – входной сигнал крена; u_3 – входной сигнал тангажа; u_4 – входной сигнал рыскания.

3.16 Упрощенное управление ПИД-регулятором

Наиболее распространенным типом системы управления является система управления пропорционально-интегрально-дифференциальным регулятором (ПИД). ПИД-регулятор известен как система обратной связи с замкнутым контуром. Контроллер вычисляет разницу между фактическим и желаемым состоянием и создает значение ошибки. Измерения от датчиков четырехротора возвращаются для вычисления этого сигнала ошибки, замыкания петли с выхода на вход. Этот контроллер популярен, потому что он несколько интуитивно понятен для настройки, легко реализуется в коде с небольшой вычислительной мощностью и использует измерения, доступные на борту большинства систем. Выходной сигнал любой ПИД-системы управления – это управляющее значение u, которое приведет систему ближе к желаемому состоянию. Состав ПИД-регулятора показан ниже как во временной области, так и в частотной области.

В формулах 122-129 u – управляющий входной сигнал; P – пропорциональный член; I – интегральный член; D – дифференциальный член; k_p – пропорциональное управляющее усиление; k_I – интегральное управляющее усиление; k_D – дифференциальное управляющее усиление; t – время; $x_d(t)$ – желаемое значение положения x в момент времени t; $x(t)$ – измеренное значение положения x в момент времени t; $e(t)$ – значение ошибки во времени t; $\dot{e}(t)$ – производная ошибки во времени t.

$$u = P + I + D \tag{122}$$

$$e(t) = x_d(t) - x(t) \tag{123}$$

$$P = k_p e(t) \tag{124}$$

$$I = k_I \int_0^t e(t-1)\, dt \tag{125}$$

$$P = k_p e(t) \tag{126}$$

$$D = k_D \dot{e}(t) \tag{127}$$

$$u(t) = k_p e(t) + k_I \int_0^t e(t-1)\,dt + k_D \dot{e}(t) \quad (128)$$

$$U(s) = k_p + \frac{k_I}{s} + k_D s \quad (129)$$

Пропорциональный член – любое число, большее 0, и приводит к значению, которое несколько кратно сигналу ошибки. Интегральный член кратен сумме накопленной с течением времени ошибки. Этот термин реагирует на нарастание любой оставшейся установившейся ошибки. Эта ошибка часто представляет собой нарушения (ветер, полезная нагрузка вне центра, ошибки моделирования и т.д.), которые не могут быть учтены пропорциональными или производными условиями. Поэтому интегральный член используется для небольших окончательных корректировок для уменьшения конечной установившейся ошибки. Производный член кратен скорости изменения ошибки. Он ослабляет реакцию системы управления и используется для уменьшения перерегулирования системы за пределами желаемого состояния. Ниже, на рисунке 26, представлено структурное представление базового ПИД-регулятора.

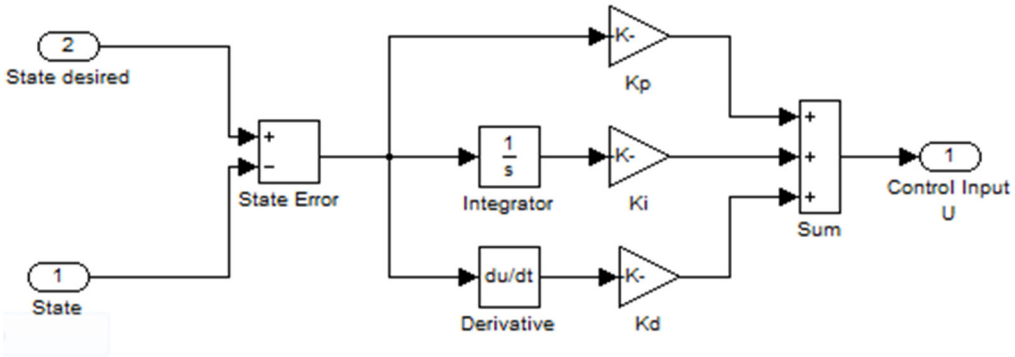

Рисунок 26 – Схема базового ПИД-регулятора

Существуют различные методы, которые можно использовать для ручной настройки ПИД-регулятора и определения наилучших значений управляющих коэффициентов при заданных параметрах реакции системы. Стоит обратить внимание, что для ручного дистанционно управляемого

полёта четырехротора необходим только регулятор положения. Контроллер положения необходим только при использовании внешнего датчика локализации, такого как GPS и навигация по маршрутным точкам. Стоит отметить, что есть некоторые характеристики физической системы, которые базовый ПИД-регулятор может не точно учитывать. К ним относятся задержки в измерениях датчиков и реакции двигателя, а также инерции транспортного средства.

3.17 Моделирование системы

В этом разделе описаны детали разработки надежной и точной среды моделирования. Точное моделирование объединяет уравнения движения, полученные в разделе 2, значения конкретных параметров, оцененные в разделе 3, и контроллер, разработанный в разделе 3 подразделе 3.8. Все эти части необходимы для создания точной имитационной среды. Точная среда моделирования позволяет проектировать и тестировать конструкции управления и алгоритмы планирования маршрута, прежде чем они будут реализованы в физической системе в реальном мире.

Альтернативой использованию сценария моделирования четырехротора является использование блок-схемы Simulink. Блок-схема хранится в файле QuadrotorSimulink.mdl. Этот файл состоит из нескольких слоев, которые будут описаны подробно. Существует несколько преимуществ этого метода. Поскольку код хранится в блоках, легко переключаться между различными контроллерами или динамикой четырехротора. Simulink также поддерживает блоки, такие как входные сигналы и диапазон применения, которые могут использоваться для анализа и настройки параметров контроллера (рисунок 27). Хотя переменные и параметры могут быть загружены из файла сценария, также можно использовать блок-диаграмму для определения и оценки неизвестных параметров системы с использованием данных полёта в реальном времени.

Значения и диапазон применения могут использоваться для управления начальными условиями четырехротора и анализа имитируемого поведения системы. Моделирование можно запустить из файла QuadSimulink.m, который также отображает некоторые данные.

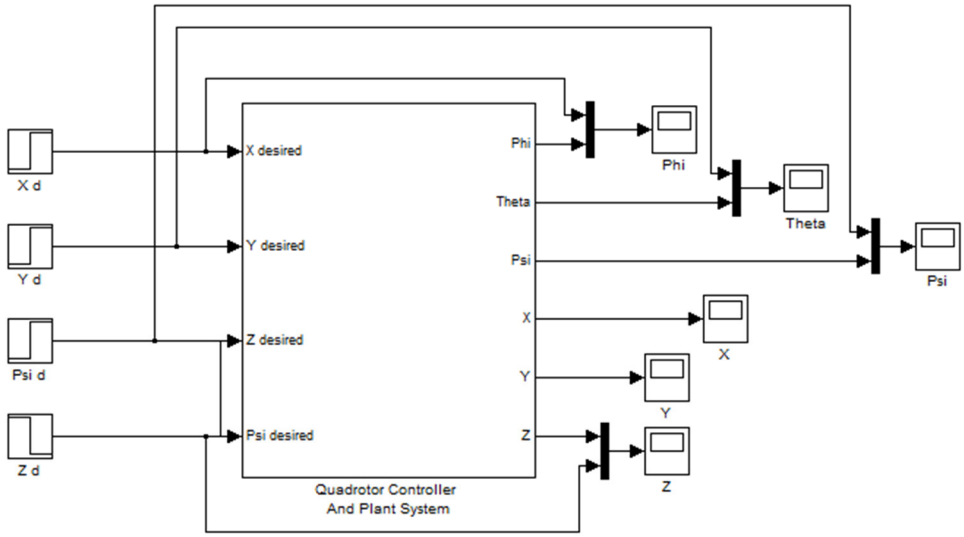

Рисунок 27 – Блок-схема системы контроллера четырехротора

На верхнем уровне блок-схема разбита на 3 первичных блока. К ним относятся внешний цикл контроллера положения, контроллер положения / высоты внутреннего цикла и динамика четырехротора. Эта серия блоков, а также их входы и выходы, показанные на рисунке 28.

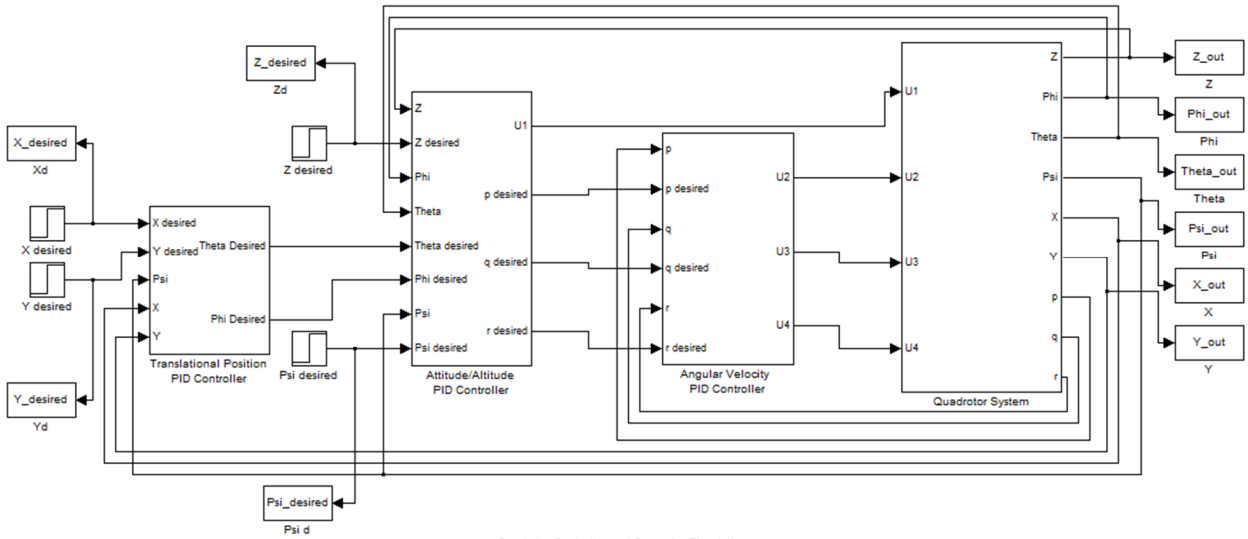

Рисунок 28 – Верхний уровень блок-схемы

Входя в поступательное положение контроллер, показывает следующую блок-схему (рисунок 29). Стоит обратить внимание, что эти уравнения не отображают уравнения для вычисления желаемого рыскания. Как правило, пользователь отправляет команды рыскания автопилота, которые переводят в желаемую скорость рыскания. Таким образом, пилот может манипулировать рысканием и иметь четырехротор, поддерживающий постоянный курс, когда рыскание центровано. Если команды рыскания перевели на определенный угол, четырехротор вернется в исходное положение, когда ось рыскания будет центрирована. Эти уравнения вычисляют желаемую скорость рыскания на основе желаемого входного угла поворота. Однако желаемая скорость рыскания обычно берется непосредственно из входов RC-команд.

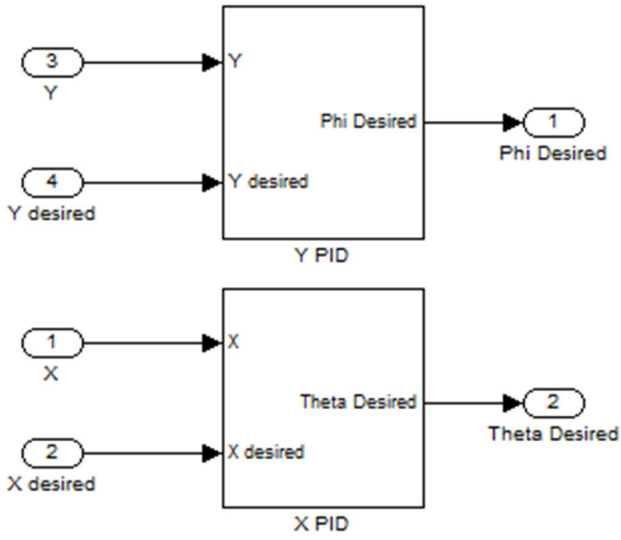

Рисунок 29 – Схема ПИД-регуляторов поступательного положения

Требуемые углы крена и тангажа, рассчитанные контроллером положения, объединены с требуемой высотой и курсом в контроллере положения-высоты, показанном ниже (рисунок 30).

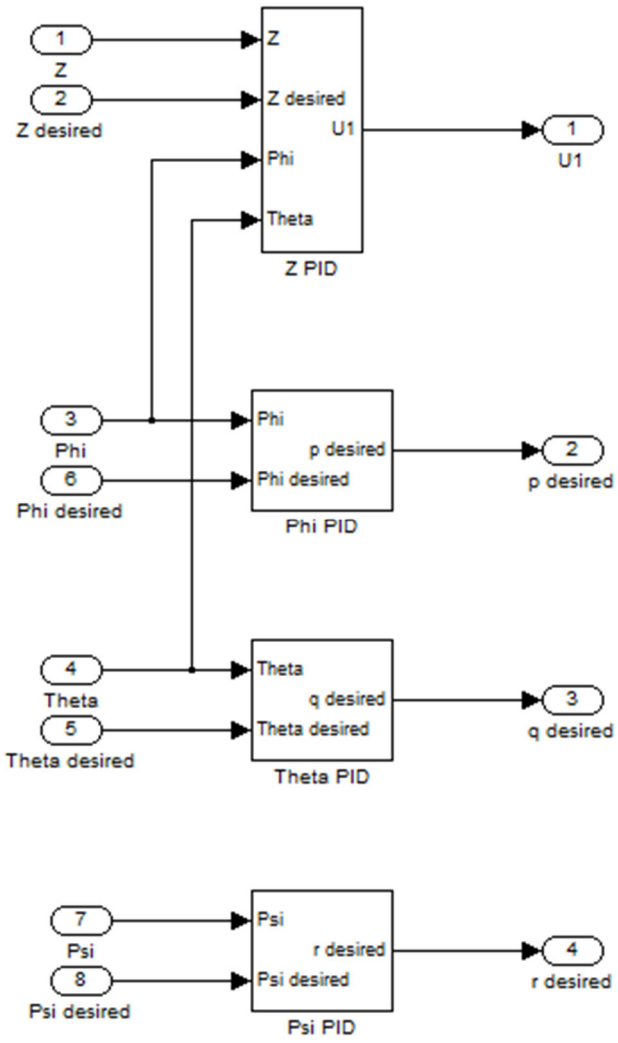

Рисунок 30 – Блок-схема ПИД-регуляторов положения-высоты

Контроллер высоты вычисляет желаемый вход управления тягой от требуемой высоты. Контроллеры ориентации принимают требуемые углы и вычисляют желаемые угловые скорости для конечных регуляторов скорости.

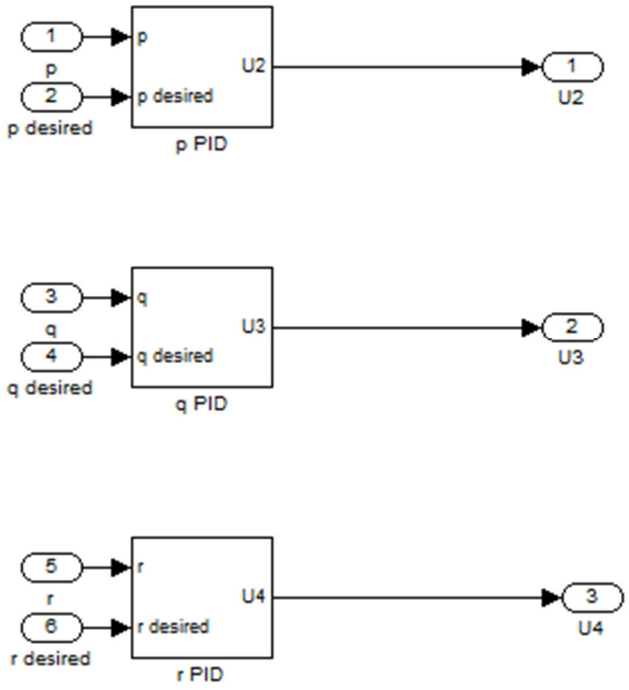

Рисунок 31 – Схема ПИД-регуляторов угловой скорости

Контроллер угловой скорости (рисунок 31) вычисляет последние три управляющих входа. Каждый блок PID содержит отдельные пропорциональные, интегральные и производные коэффициенты.

Конечные входы затем подаются в блок динамики четырехротора. Как показано ниже (рисунок 32), эта система состоит из нескольких подсистем.

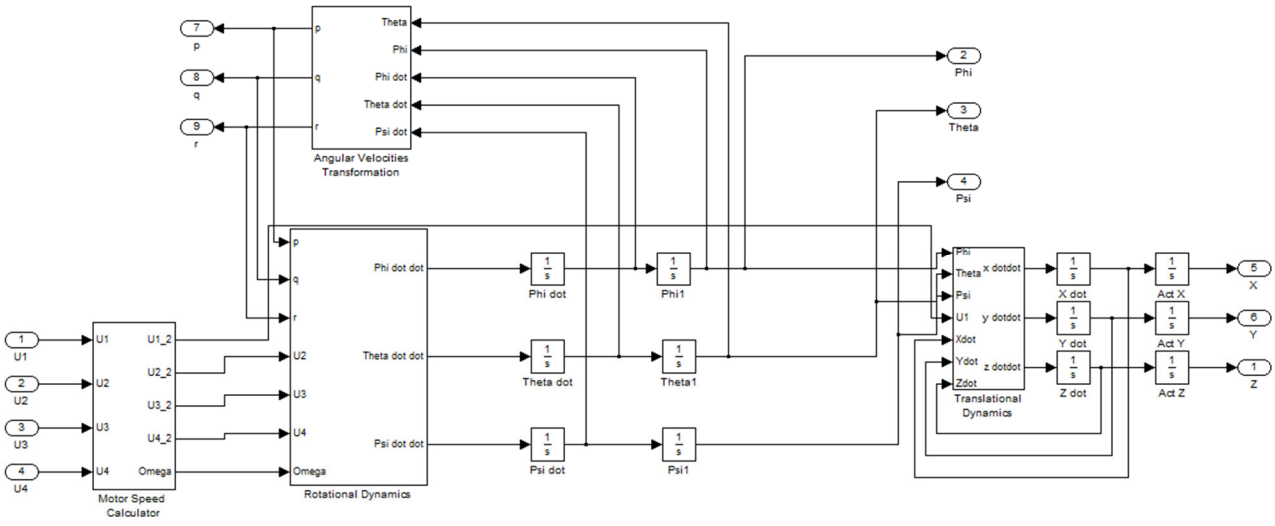

Рисунок 32 – Блок-схема динамики четырехротора

Вычислитель скорости двигателя (рисунок 33) ограничивает управляющие входы физическими параметрами двигателя.

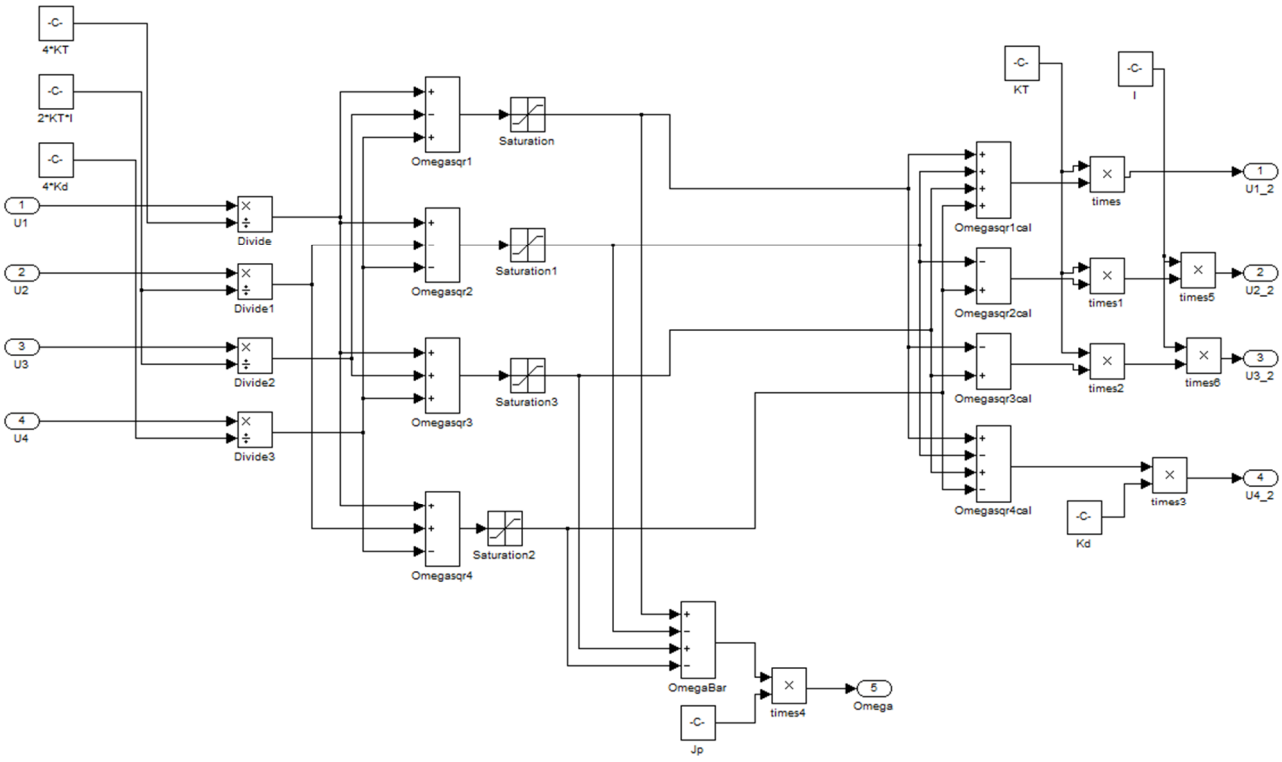

Рисунок 33 – Блок-схема вычислителя скорости двигателя

Динамика вращения (рисунок 34) вычисляет угловые ускорения.

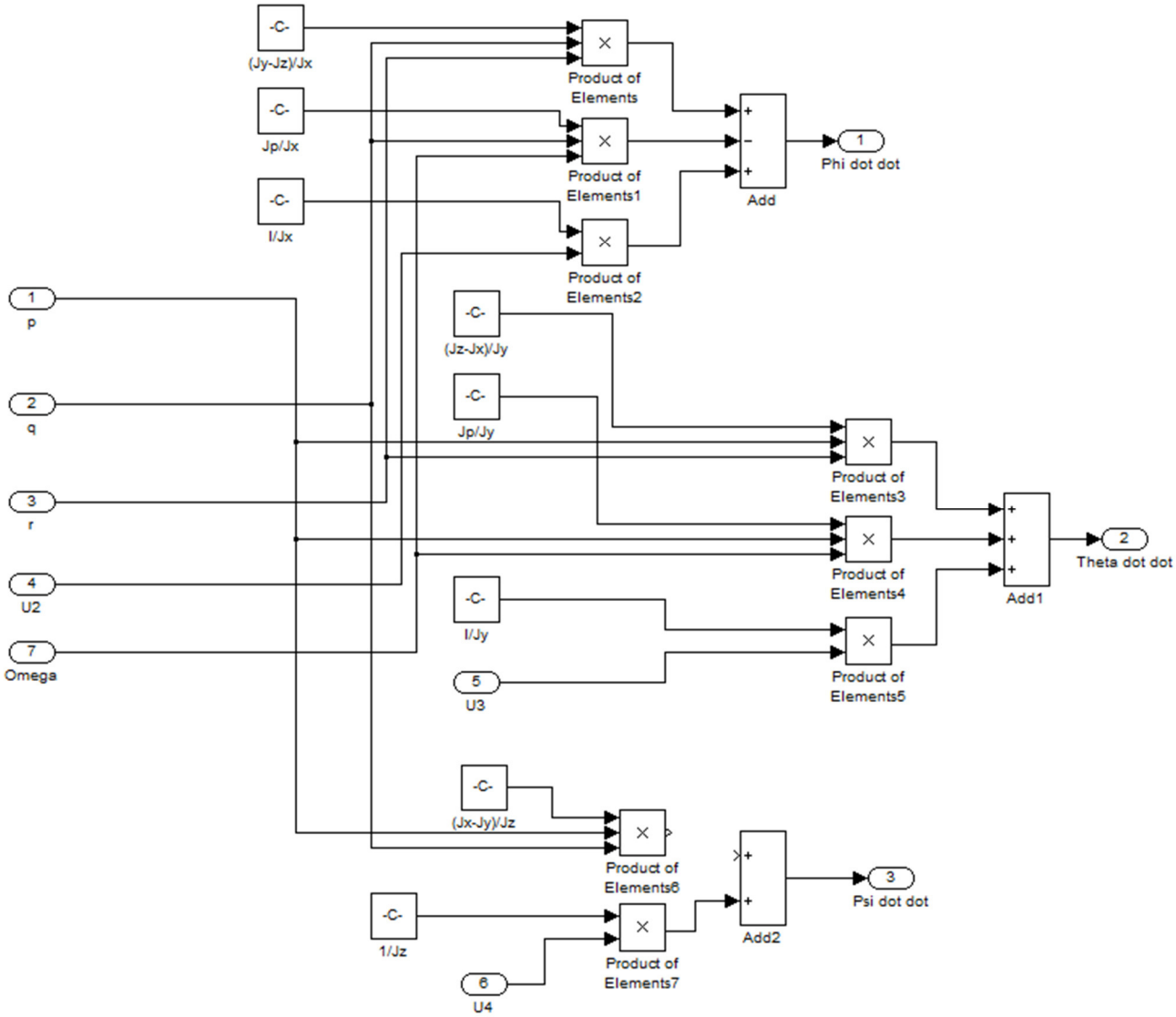

Рисунок 34 – Блок-схема для вычисления угловых ускорений

Эти значения подаются в блок преобразования угловых скоростей, который вычисляет угловые скорости (рисунок 35).

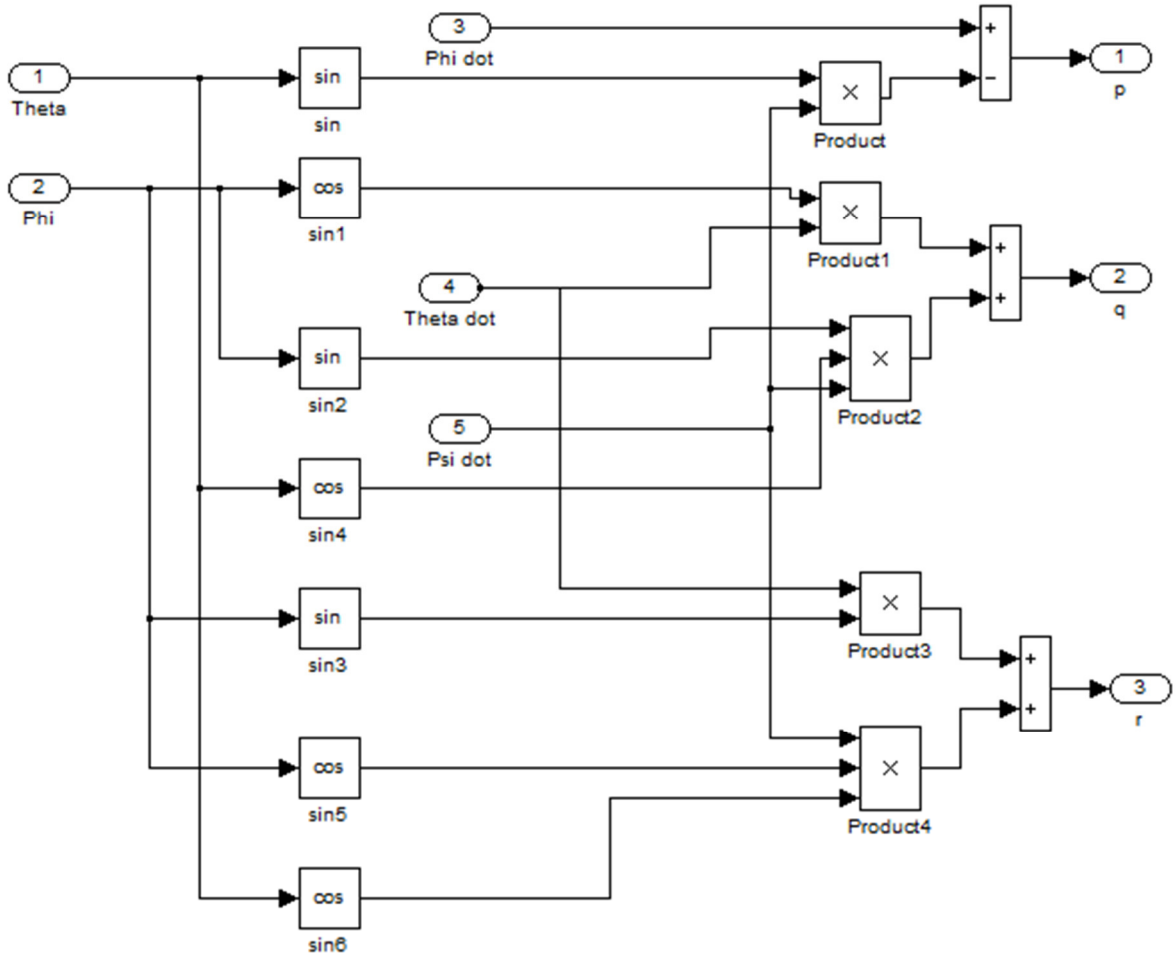

Рисунок 35 – Блок-схема для вычисления угловых скоростей

Углы также подаются в блок трансляционной динамики (рисунок 36), который вычисляет поступательные ускорения.

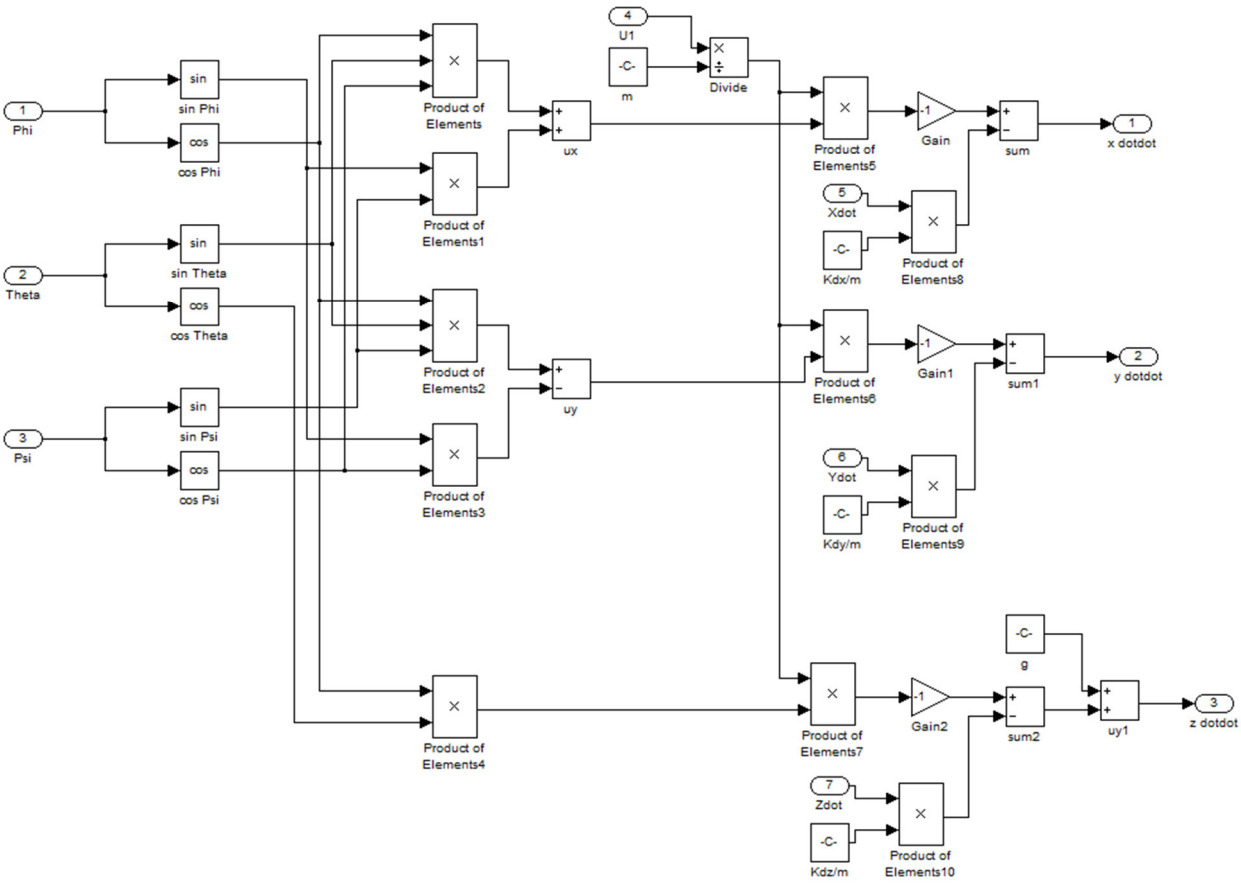

Рисунок 36 – Блок-схема трансляционной динамики

В данной части был описан процесс создания симуляционной модели в среде Simulink. В модели были учтены все входящие и ограничивающие параметры.

Далее описывается код, необходимый для реализации симулятора в MatLab.

Все функции, описанные в данном разделе содержаться в приложении Б.

Созданные сценарии MatLab позволяют быстро и легко выполнять небольшие изменения, такие как физические параметры или контрольные прибыли, а также большие изменения, такие как уравнения динамики или усиление управления. Он также позволяет быстро генерировать данные и манипулировать построением данных для анализа [21, 22]. Ниже приведено

краткое описание основных файлов и функций, необходимых для запуска и управления имитированным квадрокоптером.

Симулятор запускается с помощью файла quadrotor_sim.m. Этот файл инициализирует среду моделирования следующими командами. Создается глобальная переменная Quad, которая будет содержать все четырехроторные переменные.

Функция init_plot.m рисует трехмерную среду, в которой симулированное движение четырехротора будет визуализироваться внутри. Затем вызывается plot_quad_model.m. Этот скрипт начинается с загрузки файла Quad_plotting_model.mat, который создается путем запуска define_quad_model.m. Этот файл .mat использует физические размеры квадротора, такие как длина каркаса, толщина каркаса и радиус пропеллера, чтобы определить вершины каждого плеча и двигателя в трех измерениях для построения графика. Затем сценарий plot_quad_model.m рисует исходную четырехроторную модель в трехмерной графической среде.

В следующем разделе запускается скрипт quad_variables.m, который определяет первичные переменные для моделирования, включая физические параметры четырехротора, начальные и желаемые условия, параметры моделирования и усиление регулятора. Значения физических параметров взяты из раздела 3. Функция quad_dynamics_nonlinear.m использует начальные значения для расчета начальных скоростей и ускорений четырёхроторной модели. В этих симуляторах используются нелинейные уравнения движения, полученные в раздеел 2. Следует обратить внимание, что дискретное интегрирование используется для вычисления скорости и положения из значений ускорения.

Следующий раздел в quadrotor_sim.m представляет собой цикл, который запускает фактическое моделирование после завершения всей инициализации.

Цикл выполняется в течение времени, указанного в t_plot. Первые две области зарезервированы для реализации функций для имитации шумов

датчиков, а также для реализации фильтра, такого как расширенный фильтр Калмана (Расширенный фильтр Калмана (EKF) отличается от простого фильтра Калмана тем, что может быть использован в нелинейных процессах. Он позволяет не только уточнять оценку положения робота на карте, но и положение всех обнаруженных ориентиров.), для фильтрации измерений датчика перед отправкой их на контроллер. Сценарий моделирования шума датчика реализован в файле sensor_meas.m. Этот файл обновляет глобальное положение, линейное ускорение и переменные скорости вращения, используя модель датчика, полученную в разделе 3.

Затем вызывается функция position_PID.m, которая действует как контроллер положения. Выходы этой функции, требуемый угол крена и тангажа, являются входами в attitude _PID.m, который является контроллером положения или высоты. Оба являются реализациями ПИД-регулятора и реализуют уравнения, полученные в разделе 3.8. Стоит заметить, что желаемые позиции в глобальной системе координат должны быть преобразованы в систему координат тела.

Ниже приведен код контроллера положения или высоты. Выходы этого контроллера представляют собой три требуемые угловые скорости и желаемую тягу.

Регулятор скорости реализован в функции rate_PID.m. Этот контроллер управления принимает требуемые угловые скорости, вычисленные из контроллера положения, и вычисляет оставшиеся три управляющих входа. Затем они отображаются в командах двигателя в следующем коде:

Затем функция quad_motor_speed.m используется для преобразования управляющих входов в скорости двигателя, увеличения скоростей двигателя в зависимости от характеристик двигателя, а затем пересчета управляющих входов для уравнений движения. Соответствующий код вычисляет управляющий вход для формул преобразования скорости двигателя, полученных в разделе 3.8.

После вычисления управляющих входов с использованием физических пределов двигателя входные сигналы управления вводятся в функцию динамики четырехротора, quad_dynamics_nonlinear.m, чтобы обновить положение четырехротора в воздухе.

Последний скрипт в системе моделирования функция plot_quad.m вызывается каждые три итерации для построения текущей позиции четырехротора в трехмерной среде. Это позволяет визуализировать поведение квадрокоптера. Эта функция обновляет вершины рамы и двигателей квадрокоптера, используя положение и отношение выхода квадрокоптера из уравнений движения. Пример этой визуализации показан на рисунке 37.

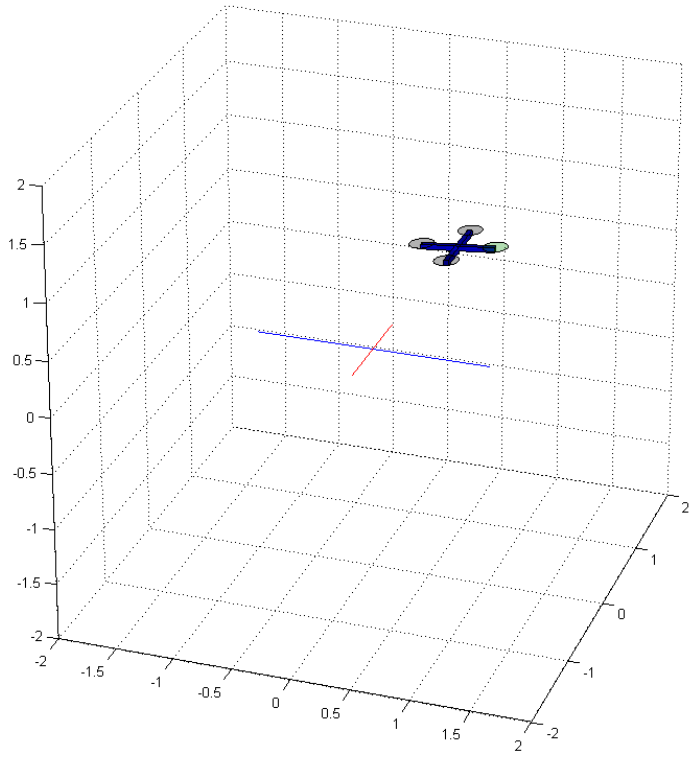

Рисунок 37 – Симуляция полёта и зависания

Как только симуляция завершена, вызывается функция plot_data.m. Эта функция используется для построения различных показателей

производительности квадрокоптера во время моделирования. Графики, изображающие реакцию системы на входные данные, показаны на рисунках 38-41, проверяя моделирование системы управления.

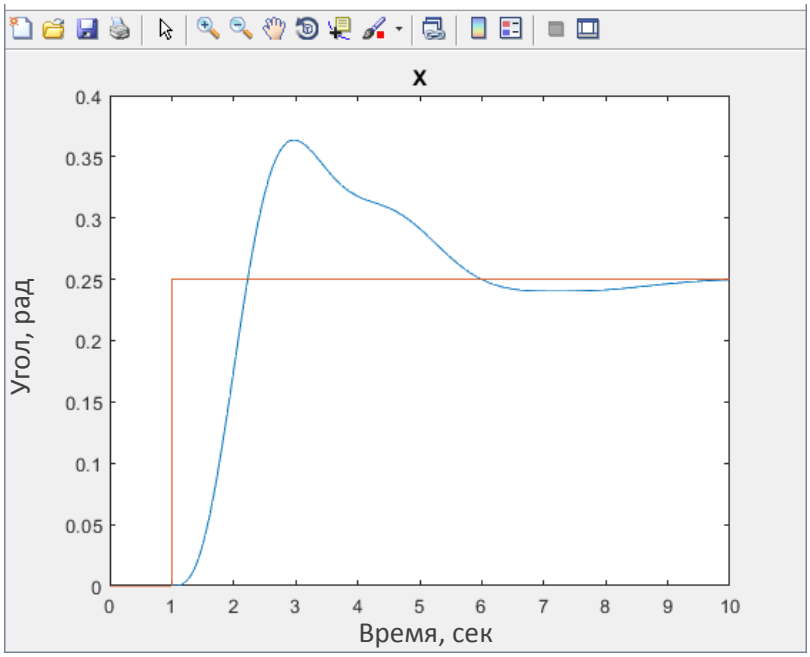

Рисунок 38 – График реакции системы на входные данные относительно оси X в течение 10 секунд

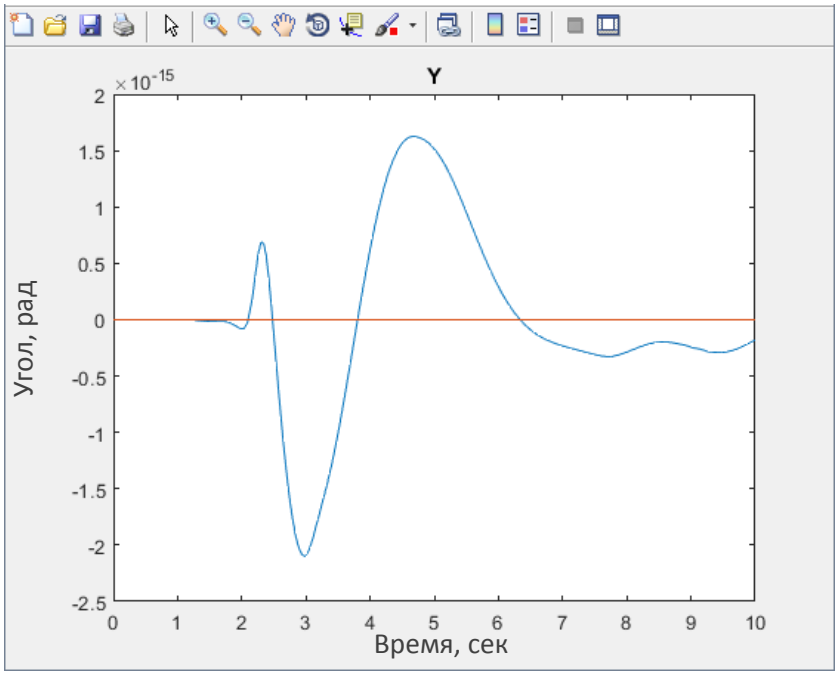

Рисунок 39 – График реакции системы на входные данные относительно оси Y в течение 10 секунд

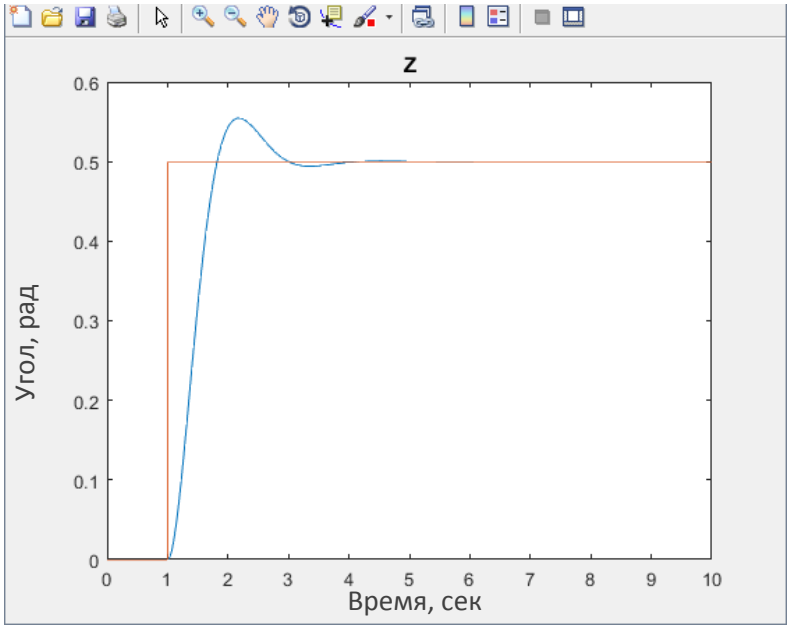

Рисунок 40 – График реакции системы на входные данные относительно оси Z в течение 10 секунд

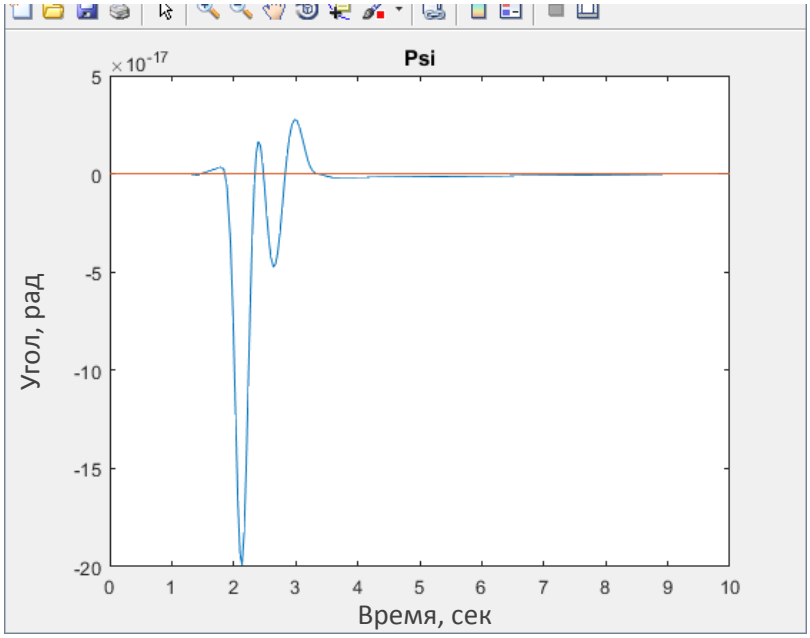

Рисунок 41 – График реакции системы на входные данные относительно оси рыскания в течение 10 секунд

Они включают фактическое и желаемое положение и углы ориентации, а также рассчитанные входы управления. Эти графики могут использоваться для настройки усиления ПИД-регулятора на определенные характеристики производительности.

3.18 Настройка ПИД-регулятора на основе нечёткой логики

В этом разделе описаны некоторые изменения, которые могут быть внесены в основной ПИД-регулятор, для повышения эффективности полёта и стабильности.

Отличие ПИД-регулятора с контроллером, основанным на нечёткой логике, от обычного заключается в том, что коэффициенты усиления в пропорциональной и интегрирующей цепях регулятора не являются статическими, т. е. зависят от состояния системы в текущий момент времени. Это позволяет качественно изменить процесс управления. Учесть параметры сигналов в системе (скорость изменения сигнала, ускорение), а также сделать процесс управления более адаптивным.

Настроим СУ с помощью нечёткого регулятора (по методу Мамдани, иначе метод максимума-минимума) [17, 18, 21,22].

Нечёткие алгоритмы – это упорядоченное множество нечётких инструкций (правил) в формулировки которых содержатся нечёткие указания (термы). Например: нечёткие алгоритмы могут включать в себя такие инструкции: а) $x \approx 5$; б) x очень мало; в) слегка увеличить x; г) если x находится в интервале (a,b), то выбрать y в интервале (c,d); д) если x малое, то y большое, иначе y не большое [8].

Методы нечёткой логики есть правила формирования результирующей функции принадлежности при выполнении одного или нескольких нечётких правил.

Алгоритм Мамдани (Mamdani) [17, 18] был предложен в 1975 г. английским математиком Е. Мамдани в качестве метода для управления паровым двигателем и получил наибольшее применение в системах нечёткого вывода. Формально алгоритм Мамдани может быть определен следующим образом [17, 18]:

1) формирование базы правил систем нечёткого вывода;

2) процедура фаззификации (введение нечёткости) входных переменных. Каждому значению отдельной входной переменной ставится в соответствие значение функции принадлежности соответствующего ей терма входной лингвистической переменной – μ1(x), μ2(x), ..., μn(x), где μ1 (x), ..., μn(x) – функции принадлежности для переменной x;

3) нечёткий вывод или нахождение степеней истинности (уровней «отсечки») для предпосылок(«Condition No1», «Condition No2», ..., «Condition NoN») каждого из правил. Процедура нахождения степеней истинности также называется агрегированием;

4) процедура активизации, то есть нахождение усечённой функции принадлежности для выходной переменной. Для этой цели можно использовать один из модифицированных методов нечёткой композиции:

- min – активация: $\mu'(y) = \min\{ci, \mu(y)\}$;
- prod – активация: $\mu'(y) = ci, \mu(y)$;
- average – активация: $\mu'(y) = 0{,}5(ci+\mu(y))$.

В приведённых методах множество $C = (c1, c2, ..., cq)$ – множество степеней истинности для каждого из правил; q– общее количество подзаключений («Conclusion No1», «Conclusion No2», ..., «Conclusion NoN») в базе правил; μ(x) – функция принадлежности терма, который является значением некоторой выходной переменной из универсума Y;

5) композиция (процедура аккумуляции) или объединение найденных усечённых функций с целью получения итогового нечёткого множества для выходной переменной и результирующей функции принадлежности;

6) процедура дефаззификации (приведение к чёткости) выходных переменных. На этапе дефаззификации может быть применён метод центра тяжести или метод биссектрисы площади.

Метод Мамдани может применяться для:

- для одного простого правила;

– для одного фокусирующего правила (для логической связи «И»);

– для нескольких правил.

Необходимо определить чёткое значение входного параметра на основе правил и чёткого значения температуры и правил.

Результатом применения любого метода НЛВ является некоторое нечёткое множество, которое описывается функцией принадлежности. В такой ситуации неопределенность выбора сохраняется. Для определения окончательного решения (конкретного значения Y) необходимо совершить переход от полученного нечёткого множества к единственному значению Y, которое признается в качестве решения поставленной задачи, такой переход называется дефаззификацией [8].

Для дефаззификации был выбран метод центра тяжести. Он заключается в том, что в качестве решения выбирается абсцисса центра тяжести площади, расположенной под результирующей функцией принадлежности.

$$y_0 = \frac{\int_y y \cdot \mu_B(y) dy}{\int_y \mu_B(y) dy} \tag{130}$$

Нечёткий регулятор представляет собой в данном случае ПИД-регулятор. Fuzzy Logic Controller – блок, с помощью которого будет осуществляться настройка системы управления, добавлен в исходную блок-схему в MATLAB/Simulink (рисунок 42) [21].

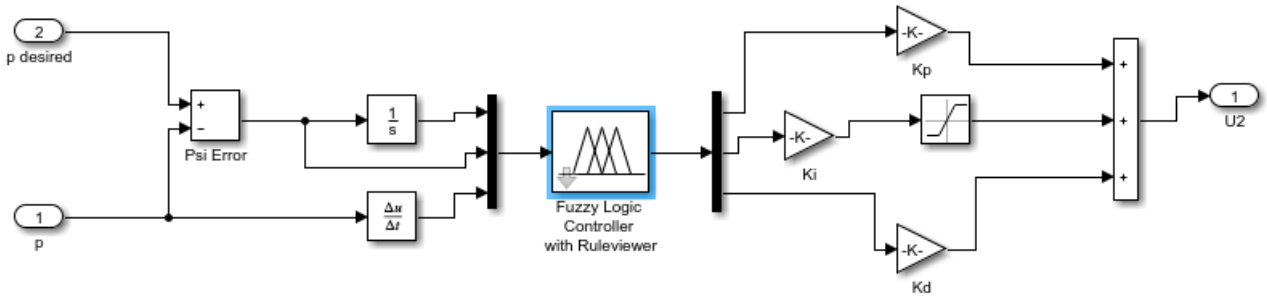

Рисунок 42 – Блок-схема с добавлением в нее Fuzzy Logic Controller в среде MATLAB/Simulink

Для применения методов нечёткой логики, прежде всего, необходимо преобразовать обычные чёткие переменные в нечёткие. Процесс такого преобразования называется фаззификацией (от английского "fuzzy"-"нечёткий") [17, 18]. Для повышения скорости работы нечёткого регулятора и достижения гладкости процесса введём две входные лингвистические переменные: ошибка по угловой скорости и скорость изменения данной ошибки (рисунок 43). Выходная переменная – напряжение.

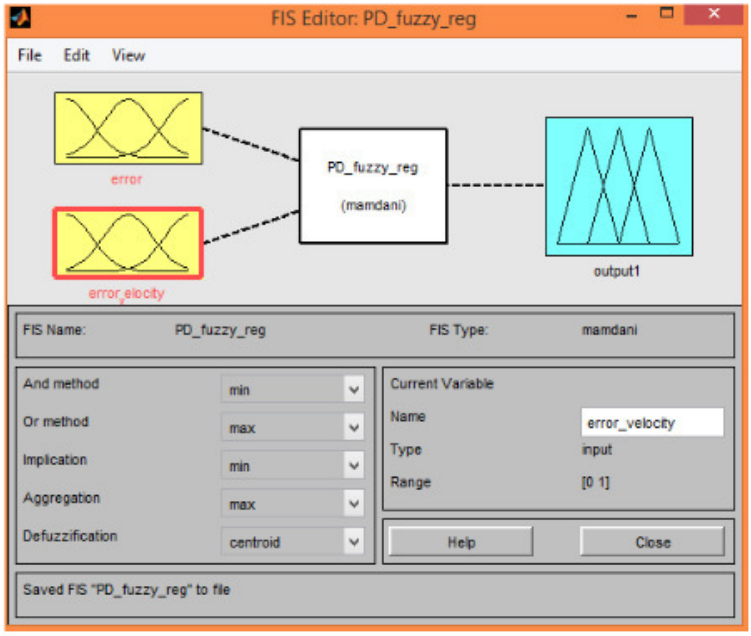

Рисунок 43 – Интерфейс блока Fuzzy Editor

Для данных лингвистических переменных зададим графики функций принадлежности треугольной формы и введём термы: N – отрицательная, Z – нулевая, P – положительная, PS – малая положительная, PM – средняя положительная, PB – большая положительная (рисунки 44-45).

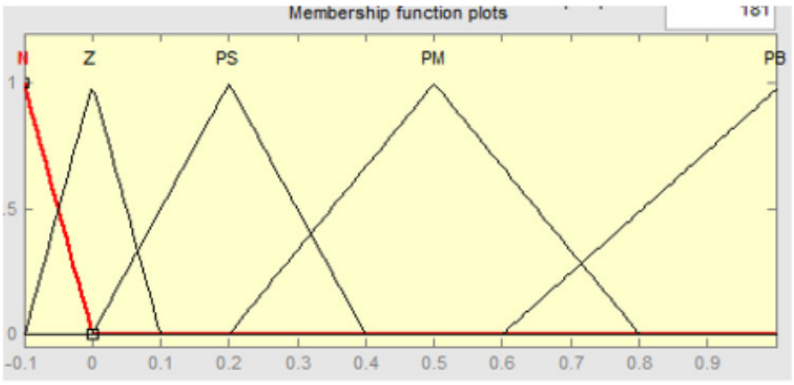

Рисунок 44 – График функции принадлежности ошибки по угловой скорости

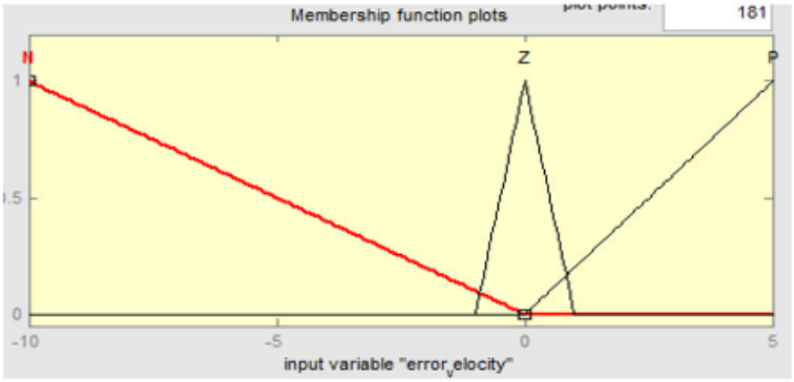

Рисунок 45 – График функций принадлежности скорости изменения ошибки по угловой скорости

Зададим график функции принадлежности треугольной формы для выходной переменной – «Напряжение» - и введём для неё термы: NM – среднее отрицательное, NS – малое отрицательное, Z – нулевое, PS – малое положительное, среднее положительное, PB – большое положительное, PL – очень большое положительное (рисунок 46).

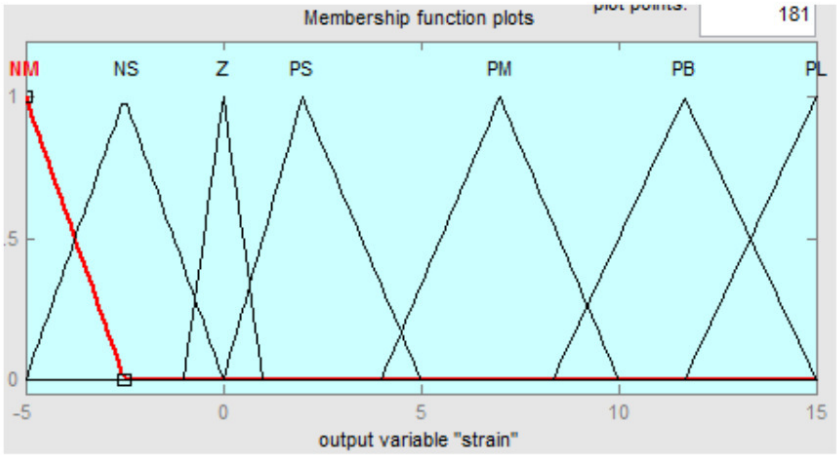

Рисунок 46 – График функций принадлежности выходной переменной

Управление подачей напряжение будет осуществляться с помощью 15 правил, которые показаны на рисунке 47.

1. If (error is N) and (error_velocity is N) then (strain is PS) (1)
2. If (error is N) and (error_velocity is Z) then (strain is PS) (1)
3. If (error is N) and (error_velocity is P) then (strain is Z) (1)
4. If (error is Z) and (error_velocity is N) then (strain is Z) (1)
5. If (error is Z) and (error_velocity is Z) then (strain is PM) (1)
6. If (error is Z) and (error_velocity is P) then (strain is PS) (1)
7. If (error is PS) and (error_velocity is N) then (strain is Z) (1)
8. If (error is PS) and (error_velocity is Z) then (strain is PS) (1)
9. If (error is PS) and (error_velocity is P) then (strain is PS) (1)
10. If (error is PM) and (error_velocity is N) then (strain is NM) (1)
11. If (error is PM) and (error_velocity is Z) then (strain is NS) (1)
12. If (error is PM) and (error_velocity is P) then (strain is PS) (1)
13. If (error is PB) and (error_velocity is N) then (strain is PS) (1)
14. If (error is PB) and (error_velocity is Z) then (strain is PB) (1)
15. If (error is PB) and (error_velocity is P) then (strain is PL) (1)

Рисунок 47 – Список правил

Благодаря этим правилам было получено две модификации ПИД-регулятора.

Первая модификация сводит к минимуму ошибку путем дифференцирования. Ошибка вычисляется как разность между заданным значением и текущим входным значением. Следовательно, производная от ошибки равна разности между производной заданного значения и

производной от измеренного входящего значения. Это означает, что любое изменение заданной величины приводит к большому скачку ее производной. Заданное значение остается относительно постоянным. При достижении желаемой траектории заданная величина может продолжать изменяться. Несмотря на это, эти мгновенные изменения заданного значения приведут к большим производным, которые будут подаваться в ПИД-регулятор. Учитывая высокую нелинейную динамику четырехротора, это может привести к нестабильности. Вместо этого предполагаем, что производная от заданного значения часто не изменяется и устанавливает свою производную на 0. Это приводит к тому, что производная от ошибки равна отрицательному значению производной измеренного входящего значения. Величину производной часто можно измерить с помощью бортового датчика. Следует отметить, что значение Kd, коэффициент усиления производной, имеет противоположный знак Kd, используемый в традиционной реализации ПИД.

Вторая модификация улучшает стабильность, устанавливая ограничения на интегральный управляющий компонент. Как описывалось ранее, интегральный компонент вычисляет сумму ошибки с течением времени. Цель этого компонента – компенсировать любые постоянные нарушения, такие как ошибки среды или моделирования. Этот компонент имеет наибольшее влияние, когда система уже достигла устойчивого состояния, но имеет постоянную погрешность. Интегральный компонент увеличит управляющий выход для медленного уменьшения ошибки установившегося состояния. Однако если есть большая ошибка из-за нарушения или внезапного изменения заданного значения, интегральный компонент может быстро стать большим значением, поскольку он продолжает увеличиваться, пока система приближается к желаемой заданной точке. Даже когда система приближается к желаемому значению, интегральный компонент все ещё очень большой и может заставить систему перерегулироваться, стать неустойчивой или колебаться. Кроме того, из-за большого интегрального нарастания, если заданное значение изменяется в

противоположном направлении, интегральный компонент задерживает реакцию контроллера, поскольку он ожидает уменьшения целочисленного значения. Чтобы исправить это, включаем только интегральный компонент, когда находимся в области вокруг устойчивого состояния, которое контролируется. Это позволяет интегральному компоненту окончательно исключить любую устойчивую ошибку. Кроме того, мы можем наполнить выход контроллера в соответствии с минимальным и максимальным значениями входного сигнала двигателя.

После внесенных модификаций в моделируемую систему произвелась симуляция полёта квадрокоптера при тех же входных данных используемых до модификации, при этом были получены графики реакции системы, изображенные на рисунках 48-51.

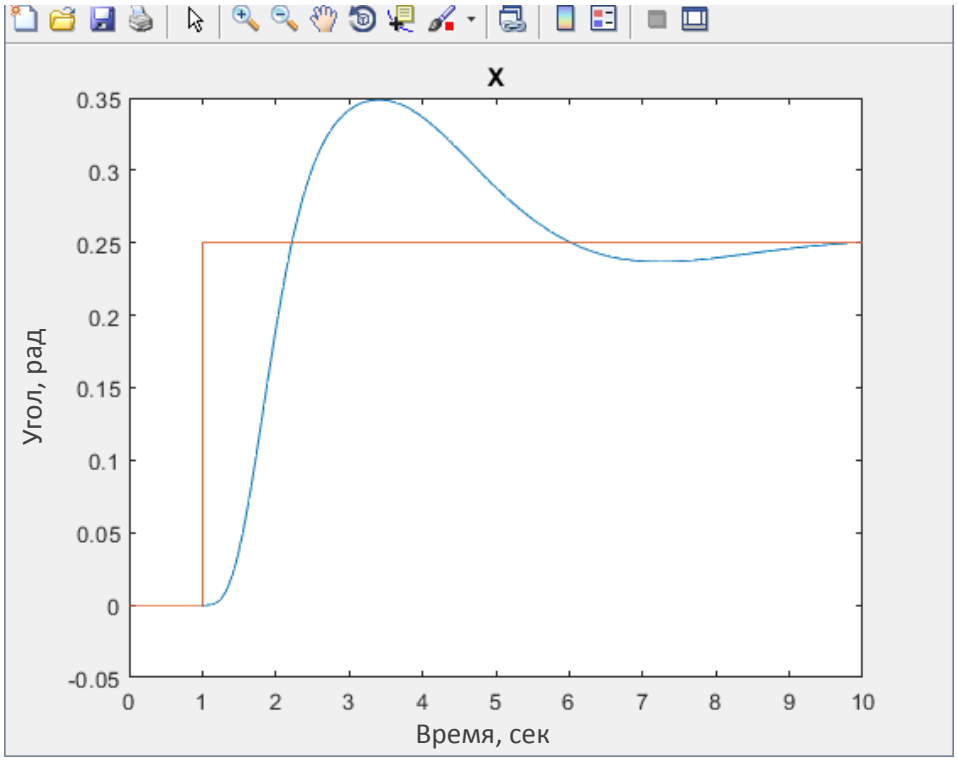

Рисунок 48 – График реакции системы с использованием нечёткой логики на входные данные относительно оси X в течение 10 секунд

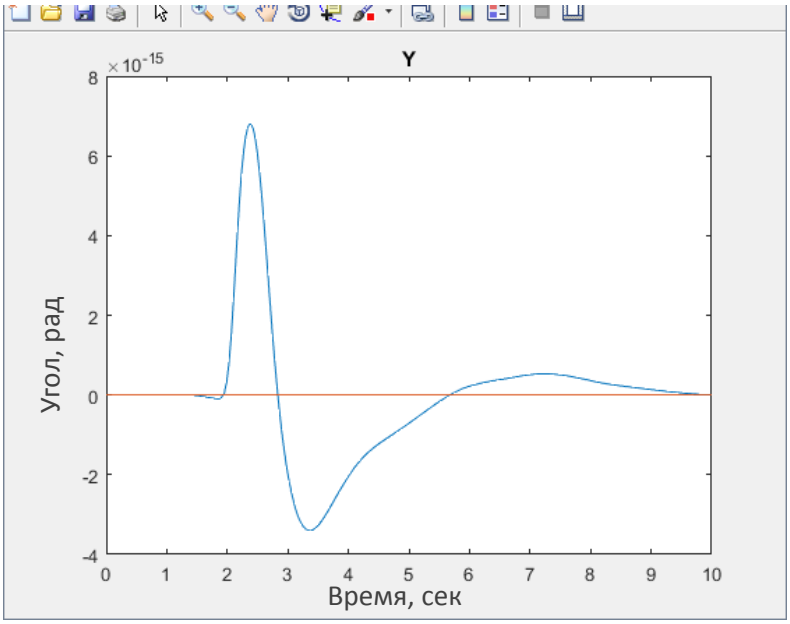

Рисунок 49 – График реакции системы с использованием нечёткой логики на входные данные относительно оси Y в течение 10 секунд

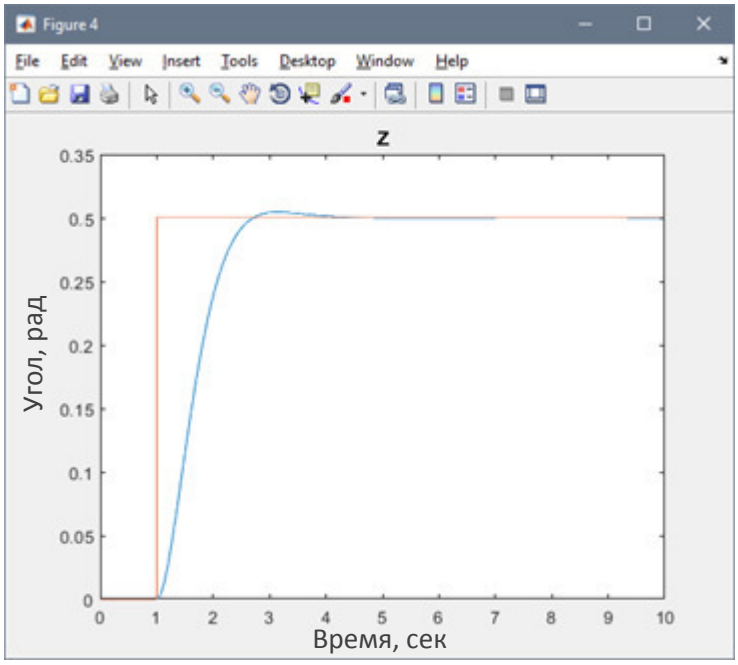

Рисунок 50 – График реакции системы с использованием нечёткой логики на входн ые данные относительно оси Z в течение 10 секунд

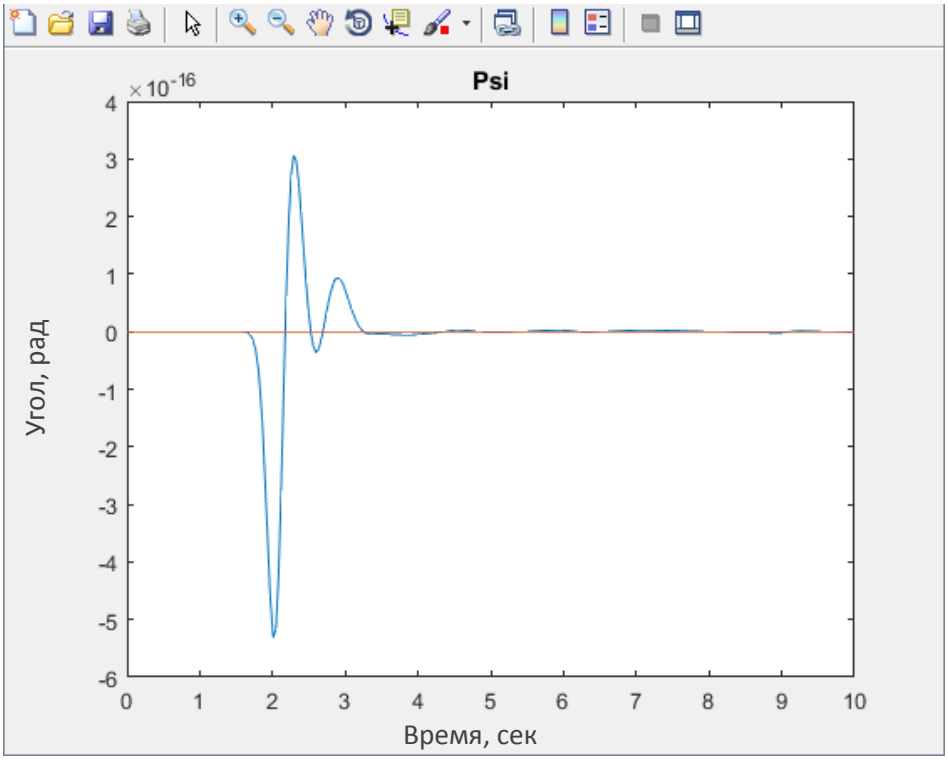

Рисунок 51 – График реакции системы с использованием нечёткой логики на входные данные относительно угла Psi в течение 10 секунд

При сравнении графиков изображенных на рисунках 38-41 и графиков изображенных на рисунках 48-51 можно увидеть, что при использовании нечёткой логики в ПИД-регуляторе моделируемое движение быстрее приходит к желаемому. Из этого следует, что процесс стабилизации квадрокоптера улучшился, а значит, желаемая цель была достигнута.

В данном разделе была разработана программная реализация, основанная на построенной математической модели, оцениваются параметры модели. Рассчитаны параметры физической модели. В среде MatLab Simulink моделируются и оцениваются: вложенные циклы управления, управление положением, управление положение и высотой, управление угловой скоростью, управление двигателями, упрощенное управление ПИД-регулятором.

Был произведён сравнительный анализ системы управления с ПИД-регулятором и ПИД-регулятором на основе нечеткой логики.

ЗАКЛЮЧЕНИЕ

В ходе выполнения магистерской диссертационной работы были выполнены задачи и достигнуты результаты такие как:

– проанализированы состояние и возможности улучшения БПЛА с точки зрения возможностей увеличения производительности целевого функционирования их использования;

– разработаны математические модели движения БПЛА, реализована их адаптация;

– осуществлена идентификация аэродинамических данных квадрокоптера;

– смоделирована модель полёта БПЛА с учетом внешних и внутренних возмущений;

– были выявлены недостатки созданной модели и путь их устранения;

– синтезированы работоспособные модифицированные правила нечёткого управления;

– на основе правил внесены изменения в исходную модель.

При сравнительном анализе исходной и итоговой модели управления было выявлено, что при использовании нечёткой логики в гибридной системе управления беспилотными аппаратами достигается лучшая устойчивость системы и ее стабилизации по оси Z на 30%, по оси Y время стабилизации улучшилось на 4 секунды, по оси Psi амплитуда изменения угла уменьшилась на 40%, по оси X движение стало менее скачкообразным.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Иноземцев, Д.П. Беспилотные летательные аппараты: Теория и практика [Текст] / Д.П. Иноземцев, –М.: Автоматизированные технологии изысканий и проектирования, 2013. №2. – С. 50-54.

2. Michael, D. S. Simulation and control of a quadrotor unmanned aerial vehicle. [Электронный ресурс] / Владелец: D. S. Michael – Режим доступа: uknowledge.uky.edu/gradschool_theses/93.

3. Моисеев, В.С., Гущина Д.С., Моисеев Г.В. Основы теории создания и применения информационных беспилотных авиационных комплексов (Серия «Современная прикладная математика и информатика») [Текст] / В.С. Моисеев, –Казань: Изд-во МОиН РТ, 2010. 196 с.

4. Павлушенко, М. И., Евстафьев Г. М., Макаренко И.К. БПЛА: история, применение, угроза распространения и перспективы развития [Текст] / М. И. Павлушенко, –М., «Права человека», 2005. 612 с.

5. Янкевич, Ю. Применение беспилотных авиационных комплексов в гражданских целях [Текст] / Ю. Янкевич, –М.: Аэрокосмический курьер, 2006, № 6. С. 55-57.

6. Боднер, В.А. Авиационные приборы [Текст] / В.А. Боднер, – М.: Машиностроение. – 1969. – 467 с.

7. S. Bouabdallah. Design and control of quadrotors with application to autonomous flying, dissertation 3727 [Текст] / 2007.

8. Алиев, Р.А., Церковный А.З., Мамедова Г.А. Управление производством при нечеткой исходной информации [Текст] / Р.А. Алиев, – М.: Энергоатомиздат, 1991. 240 с.

9. Казаков, И.Е., Доступов Б.Г. Статистическая динамика линейных автоматических систем [Текст] / И.Е. Казаков,– М.: Наука, 1962. – 332 с.

10. Красовский, А.А. Системы автоматического управления полётом и их аналитические конструирование [Текст] / А.А. Красовский, – М.: Наука, 1973. – 560 с.

11. Доброленский, Ю.П. Динамика полёта в неспокойной атмосфере

[Текст] / Ю.П. Доброленский, –М.: Машиностроение, 1969. – 256 с.

12. Михалёв, И.А., Окоемов Б.Н., Чикулаев М.С. Системы автоматической посадки [Текст] / И.А. Михалёв, –М.: Машиностроение, 1975. – 216 с.

13. Павлов, В.В. Инвариантность и автономность нелинейных систем управления [Текст] / В.В. Павлов,– К.: Наукова Думка, 1971. – 271 с.

14. Mclean D. Automatic flight control systems [Текст] / Prentice Hall Inc., Englewood Cliffs, 1990. – 593 p.

15. Stevens B.L., Lewis F.L. Aircraft control and simulation [Текст] / J. Wiley & Sons Inc., 1992. – 617 p.

16. Driankov D., Hellendoorn H., Reinfrank, M. An introduction to fuzzy control [Текст] / Springer–Verlag, Berlin, Heidelberg, New York. 1993. – 316 p.

17. Schram G., Kaymak U., Verbruggen H.B. Fuzzy logic control. Chapter 13. in a book «Robust Flight Control Design Challenge» [Текст] / Springer-Verlag. – Berlin, Heidelberg, New York. – P 135–145.

18. Schram G., Verbruggen H.B. A fuzzy control approach. Chapter 26 in a book «Robust Flight Control Design Challenge» [Текст] / Springer-Verlag. – Berlin, Heidelberg, New York. – P. 397–419.

19. Ульянов С.В. Нечеткие модели интеллектуальных систем управления: теоретические и прикладные аспекты (обзор) [Текст] / С.В. Ульянов, изве. АН СССР. Сер. Техническая кибернетика. – 1991. – № 3. –29с.

20. Lin, C.F. Advanced control systems design [Текст] / PTR Prentice Hall Inc., Englewood Cliffs, New Jersey, 1994. – 664 p.

21. Дьяконов, В.П. MATLAB 6/6.1/6.5 + SIMULINK 4/5 в математике и моделировании [Текст] / В.П. Дьяконов, – М.: СОЛОН-Пресс, 2003 – 565 с.

22. Егупова, Н.Д. Методы робастного, нейронечёткого и адаптивного управления: Учебник/ под ред. Н.Д. Егупова, 2-е изд. [Текст] / – М.: Изд.-во МГТУ им. Баумана, 2002. – 744 с.

23. Леоненков, А.В. Нечёткое моделирование в среде MATLAB и fuzzy TECH [Текст] / А.В. Леоненков,– СПб.: БХВ-Петербург, 2003. – 736 с.

24. Заде, Л. Понятие лингвистической переменной и ее применение к принятию приближенных решений [Текст] / Л. Заде, – М.: Мир, 1976. – 167 с.

25. Мартынов, А.К. Экспериментальная аэродинамика [Текст] / А.К. Мартынов, – М.: Государственное издательство оборонной промышленности, 1950. 479 с.

26. Мирошник, И.В. Теория автоматического управления. Линейные системы. [Текст] / И.В. Мирошник, –СПб.: Питер, 2005. 337 с.

27. Цепляева, Т.П., Морозова О.В. Этапы развития беспилотных летательных аппаратов [Текст] / Т.П. Цепляева, –М.: «Открытые информационные и компьютерные интегрированные технологии», 2009. № 42. 78 с.

28. Макаров, И.М., Лохин В.М., Манько С.В. и др. Интеллектуальные системы управления беспилотными летательными аппаратами на основе комплексного применения технологий нечеткой логики и ассоциативной памяти [Текст] / И.М. Макаров, –М.: Авиакосмическое приборостроение, 2002. №2 С. 29-36.

29. Куршев, Н.В., Кожевников Ю.В. Оптимальные задачи динамики полёта. 2-е изд. доп. и перераб [Текст] / Н.В. Куршев, –Казань: Изд-во КГТУ им. А.Н. Туполева, 2010. 326 с.

30. Герман-Галкин, С. Г. Модельное проектирование синхронных мехатронных систем // Matlab & Simulink. Проектирование мехатронных систем на ПК. [Текст] / С. Г. Герман-Галкин, –СПб.: КОРОНА-Век, 2008. — 368 с. — ISBN 978-5-903383-39-9

31. Белинская, Ю.С. и Четвериков В.Н. Управления четырехвинтовным вертолётом [Текст] / Ю.С. Белинская, –М.: Наука и образование, 2008.

32. Шаров, С.Н. Информационные управляющие системы беспилотных летательных аппаратов [Текст] / С.Н. Шаров, –СПб.: Балтийский гос. техн. университет. 2007. 257 с.

33. Шалыгин, А.С., Лысенко Л.Н, Толпегин О.А. Методы

моделирования ситуационного управления беспилотных летательных аппаратов. / Под ред. А. В. Ноздрачева и Л. Н. Лысенко. Том 11. Справ. библиотека разработчика – исследователя. [Текст] / –М.: Машиностроение, 2012. 584 с.

34. Хаммуд, А. Использование нейросетевых подходов в адаптивных системах управления летательными аппаратами [Текст] / А. Хаммуд, диссертация степени канд. техн. наук, –М.: МГТУ им. Н. Э. Баумана, 2004. 132 с.

35. Асаи, К., Ватада, Д., Иваи С. Прикладные нечеткие системы [Текст] / К. Асаи, Д. Ватада, С. Иваи.; Под ред. Т. Тэрано, К. Асаи, М. Сугэно; Пер. с яп. –М.: Мир, 1993. 368 с.

36. Поспелова, Д.А. Нечеткие множества в моделях управления и искусственного интеллекта [Текст] / Под ред. Д.А. Поспелова. –М.: Наука, 1986. 312 с.

37. Егупова, Н.Д. Методы робастного, нейро-нечеткого и адаптивного управления: Учебник [Текст]/ Под ред. Н.Д.Егупова; издание 2-ое, стереотипное. –М.: Изд-во МГТУ им. Н.Э.Баумана, 2002. 744 с.

38. Кузнецов А.Г. Повышение точности оценки координат малогабаритного беспилотного летательного аппарата с использованием системы технического зрения [Текст] / А.Г. Кузнецов, диссертация канд. техн. наук, –М.: МАИ, 2011. 152 с.

39. Чуйкова, А.Г. Естественнонаучные, инженерные и экономические исследования в технике, промышленности, медицине и сельском хозяйстве : материалы I молодёжной научно-практической конференции с международным участием [Текст] / под общ. ред. С. Н. Девицыной. - Белгород: ИД "Белгород" , 2017. - 693 с.

40. Чуйкова, А.Г. Обзор существующих методов решения задачи управления полётом малых БПЛА и обоснование необходимости применения нечётких автопилотов. [Текст] / IX Международная студенческая электронная научная конференция «Студенческий научный форум» - 2017.

41. Чуйкова, А.Г. Выявление устойчивости положения равновесия с использованием адаптивного регулятора с помощью второго (прямого) метода Ляпунова [Текст] / XVII Международная научно-практическая конференция «Научные тенденции: Вопросы точных и технических наук» - 2018.

42. Чуйкова, А.Г. General strategy for the management of the quadrocopter [Текст] / Общая стратегия управления квадрокоптером XLVI International scientific and practical conference: «International scientific review of the problems and prospects of modern science and education». Boston. USA. June 24-25, 2018.

43. Кулик А.С. Словарь терминов по системам управления летательных аппаратов (СУЛА) [Текст] / А.С. Кулик - Харьков «ХАИ», 2000.

44. Кулик А.С., Гордин А.Г., Нарожный В.В., Бычкова И.В., Таран А.Н. Проблематика разработки перспективных малогабаритных летающих роботов [Текст] / А.С. Кулик, Национальный аэрокосмический университет им. Н.Е. Жуковского «Харьковский авиационный институт», Украина, 2005.

45. Красильщикова, М.Н., Себрякова, Г.Г.. Управление и наведение беспилотных маневренных летательных аппаратов на основе современных информационных технологий [Текст] / Под ред. М.Н. Красильщикова, Г.Г. Себрякова. –М.: Физматлит, 2003.

46. Иванова Ю.Л. Беспилотные летательные аппараты. Состояние и тенденции развития [Текст] / Под ред. Иванова Ю.Л. –М.: Варяг, 2004.

47. Лисейцев, Н. К., Максимович, В. З.. Беспилотные самолеты вертикального взлета и посадки: Выбор схемы и определение проектных параметров [Текст] / Н. К. Лисейцев, Под ред. д-ра техн. наук, проф. Н. К. Лисейцева. –М.: МАИ-ПРИНТ, 2009.- 140с.

48. Яхъяева Г. Нечёткие множества и нейронные сети [Текст] / Г. Яхъяева, –М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2006. – 316 с.

49. Усков А.А., Круглов В.В. Интеллектуальные системы управления на основе методов нечеткой логики [Текст] / А.А. Усков, –Смоленск: Смоленская городская типография, 2003. -31 с.

50. Тэрано Т. Прикладные нечеткие системы [Текст] / Т. Тэрано, – М.: Мир, 1993г. - 368с.

51. Рыжов, А.П. Модели поиска информации в нечеткой среде [Текст] / А.П. Рыжов, –М.: Издательство ЦПИ при механико–математическом факультете МГУ, г. Москва, 2004. - 96 с.

52. Рыбин, В.В. Основы теории нечётких множеств и нечеткой логики [Текст] / В.В. Рыбин, Учебное пособие. –М.: МАИ, 2007. – 96 с.

53. Пегат, А. Нечёткое моделирование и управление [Текст] / А. Пегат, –М.: Бином. Лаборатория знаний, 2009. –798 с.: ил.

54. Новак, В., Перфильева, И.Г., Мочкорж, И. Математические принципы нечеткой логики [Текст] / В. Новак, Пер с англ.; Под ред. Аверкина А.Н. –М.: Физматлит, 2006. –352 с.

55. Круглов В.В., Усков А.А. Два подхода к самоорганизации базы правил системы нечеткого логического вывода [Текст] / В.В.Круглов, –М.: Информационные технологии. 2006. №2. 14-18 с.

56. Черных И.В. Моделирование электротехнических устройств в MATLAB, SimPowerSystems и Simulink [Текст] / И.В. Черных, –М.: 1-е издание, 2007. –288 с.

57. Цисарь, И.Ф. MATLAB Simulink. Компьютерное моделирование экономики [Текст] / И.Ф. Цисарь, –М.: Солон-Пресс, 2008. –256с.

58. Демидова Л.А., Кираковский В.В., Пылькин А.Н. Алгоритмы и системы нечеткого вывода при решении задач диагностики городских инженерных коммуникаций в среде MATLAB [Текст] / Л.А. Демидова, –М.: «Горяч.Линия-Телеком» 2005. –365 с.

59. Иглин. С.П. Математические расчеты на базе Matlab [Текст] / С.П. Иглин, – СПб.: «BHV-Санкт-Петербург» 2005. –640 с.

60. Дьяконов, В.П. Matlab 6.5 SP1/7 + Simulink 5/6. Основы применения [Текст] / В.П. Дьяконов, –М.: СОЛОН-Пресс, 2005. – 800с.

61. Дьяконов, В.П. Matlab 6.5 SP1/7 + Simulink 5/6 в математике и моделировании [Текст] / В.П. Дьяконов, –М.: СОЛОН-Пресс, 2005. – 576с.

62. Ануфриев, И., Смирнов, А., Смирнова, Е. MATLAB 7.0 в подлиннике [Текст] / И. Ануфриев, –М.: Новая техническая книга, 2005. – 629с.

63. Кривилев А. Основы компьютерной математики с использованием системы MATLAB [Текст] / А. Кривилев, –М.: Лекс-Книга, 2005. –419с.

64. Дэбни, Дж., Харман, Т. Simulink 4. Секреты мастерства [Текст] / Дж.Дэбни, Т.Харман , М: Бином. Лаборатория базовых знаний. 2003. –693с.

65. Потемкин, В. MATLAB 6: Среда проектирования инженерных приложений [Текст] / В.Потемкин, –М.: Диалог-МИФИ. 2003. –428с.

66. Данилов, А.. Компьютерный практикум по курсу «Теория управления». Simulink-моделирование в среде Matlab [Текст] / А.Данилов, –М.: МГУИЭ. 2002. –98с.

ПРИЛОЖЕНИЕ А

```
clear all;
close all;
clc;

% Add Paths
addpath logs
addpath utilities
addpath 'C:\Users\test\MATLAB\SysID\utilities\Smooth'
addpath 'C:\Users\test\MATLAB\SysID\utilities\angleStuff'
addpath 'C:\Users\test\MATLAB\SysID\utilities\rls\matlab'

%% Load Log Data
load IMUlog100Hz2;

%% формат для гарфиков

FontName = 'Arial';
FontSize = 14;
plotlinewidth=2;

%% Извлечение данных

t_start = 1;
t_end = size(IMUlog100Hz2,1);

% Время
time = IMUlog100Hz2(t_start:t_end,1)/1000;
time = time-time(1);
IMUPer = median(diff(time));

% Акселерометр (m/s^2)
x_acc = IMUlog100Hz2(t_start:t_end,2);
x_acc_bias = mean(x_acc);
x_acc_nobias = x_acc - x_acc_bias;
x_acc_mdl = x_acc_bias + std(x_acc).*randn(size(time,1),1);
DATA = struct('freq',x_acc,'rate',100);
% DATA = struct('freq',x_acc,'time',time);
[adxg,empty,empty,tauxg] = allan(DATA,'','data',0);

y_acc = IMUlog100Hz2(t_start:t_end,3);
y_acc_bias = mean(y_acc);
y_acc_nobias = y_acc - y_acc_bias;
y_acc_mdl = y_acc_bias + std(y_acc).*randn(size(time,1),1);
DATA = struct('freq',y_acc,'rate',100);
% DATA = struct('freq',x_acc,'time',time);
[adyg,empty,empty,tauyg] = allan(DATA,'','data',0);

z_acc = IMUlog100Hz2(t_start:t_end,4);
z_acc_bias = mean(z_acc);
z_acc_nobias = z_acc - z_acc_bias;
z_acc_mdl = z_acc_bias + std(z_acc).*randn(size(time,1),1);
DATA = struct('freq',z_acc,'rate',100);
% DATA = struct('freq',x_acc,'time',time);
[adzg,empty,empty,tauzg] = allan(DATA,'','data',0);

% Гироскоп (deg/s)
x_gyro = IMUlog100Hz2(t_start:t_end,6);
```

```
x_gyro_bias = mean(x_gyro);
x_gyro_nobias = x_gyro - x_gyro_bias;
x_gyro_mdl = x_gyro_bias + std(x_gyro).*randn(size(time,1),1);
DATA = struct('freq',x_gyro,'rate',100);
% DATA = struct('freq',x_acc,'time',time);
[adx,empty,empty,taux] = allan(DATA,'','data',0);

y_gyro = IMUlog100Hz2(t_start:t_end,7);
y_gyro_bias = mean(y_gyro);
y_gyro_nobias = y_gyro - y_gyro_bias;
y_gyro_mdl = y_gyro_bias + std(y_gyro).*randn(size(time,1),1);
DATA = struct('freq',y_gyro,'rate',100);
% DATA = struct('freq',x_acc,'time',time);
[ady,empty,empty,tauy] = allan(DATA,'','data',0);

z_gyro = IMUlog100Hz2(t_start:t_end,8);
z_gyro_bias = mean(z_gyro);
z_gyro_nobias = z_gyro - z_gyro_bias;
z_gyro_mdl = z_gyro_bias + std(z_gyro).*randn(size(time,1),1);
DATA = struct('freq',z_gyro,'rate',100);
% DATA = struct('freq',x_acc,'time',time);
[adz,empty,empty,tauz] = allan(DATA,'','data',0);

%тест

for(i = 1: size(x_gyro,1))
    n(i,1) = (6.468e-5/sqrt(.01))*randn(1);
    if(i>1)
        b(i,1) = b(i-1,1) + 3.818e-5*sqrt(.01)*randn(1);
    else
        b(i,1) = 1.5e-5*sqrt(.01)*randn(1);
    end
end
x_test = n + b;

figure();
plot(time,x_gyro,'.');
hold on;
plot(time,x_test,'.r');
legend('Measured','Modeled');
xlabel('Time (sec)');
ylabel('Acceleration (m/s^2)');
title('X Axis Acceleration');
grid on;

break;

% Print out model

disp(['X acc: bias = ' num2str(x_acc_bias) ' sigma = ' num2str(std(x_acc))
]);
disp(['Y acc: bias = ' num2str(y_acc_bias) ' sigma = ' num2str(std(y_acc))
]);
disp(['Z acc: bias = ' num2str(z_acc_bias) ' sigma = ' num2str(std(z_acc))
]);
disp(' ');
disp(['X gyro: bias = ' num2str(x_gyro_bias) ' sigma = ' num2str(std(x_gyro))
]);
disp(['Y gyro: bias = ' num2str(y_gyro_bias) ' sigma = ' num2str(std(y_gyro))
]);
```

```
disp(['Z gyro: bias = ' num2str(z_gyro_bias) ' sigma = ' num2str(std(z_gyro))
]);
disp(' ');

%% Ploting

% Отклонение акселерометра
figure();
loglog(tauxg,adxg,'.-b','LineWidth',2,'MarkerSize',24);
hold on;
loglog(tauyg,adyg,'.-r','LineWidth',2,'MarkerSize',24);
loglog(tauzg,adzg,'.-g','LineWidth',2,'MarkerSize',24);
legend('X acc','Y acc','Z acc');
grid on;
title(['Allan Deviation: Accelerometer
'],'FontSize',FontSize+2,'FontName',FontName);
xlabel('\tau (sec)','FontSize',FontSize,'FontName',FontName);
ylabel('\sigma_y (m/s^2)','FontSize',FontSize,'FontName',FontName);

%  Отклонение гироскопа
figure();
loglog(taux,adx,'.-b','LineWidth',2,'MarkerSize',24);
hold on;
loglog(tauy,ady,'.-r','LineWidth',2,'MarkerSize',24);
loglog(tauz,adz,'.-g','LineWidth',2,'MarkerSize',24);
legend('X gyro','Y gyro','Z gyro');
grid on;
title(['Allan Deviation: Gyroscope
'],'FontSize',FontSize+2,'FontName',FontName);
xlabel('\tau (sec)','FontSize',FontSize,'FontName',FontName);
ylabel('\sigma_y (deg/s)','FontSize',FontSize,'FontName',FontName);

% Plot Accelermoter
figure();
subplot(3,1,1);
plot(time,x_acc,'.');
hold on;
plot(time,x_acc_mdl,'.r');
legend('Measured','Modeled');
xlabel('Time (sec)');
ylabel('Acceleration (m/s^2)');
title('X Axis Acceleration');
grid on;

subplot(3,1,2);
plot(time,y_acc,'.');
hold on;
plot(time,y_acc_mdl,'.r');
legend('Measured','Modeled');
xlabel('Time (sec)');
ylabel('Acceleration (m/s^2)');
title('Y Axis Acceleration');
grid on;

subplot(3,1,3);
plot(time,z_acc,'.');
hold on;
plot(time,z_acc_mdl,'.r');
legend('Measured','Modeled');
xlabel('Time (sec)');
ylabel('Acceleration (m/s^2)');
title('Z Axis Acceleration');
```

```
grid on;
suptitle('Acceleromter Modeling');

% Plot Gyroscope
figure();
subplot(3,1,1);
plot(time,x_gyro,'.');
hold on;
plot(time,x_gyro_mdl,'.r');
legend('Measured','Modeled');
xlabel('Time (sec)');
ylabel('Angular Velocity (rad/s)');
title('X Axis Angular Velocity');
grid on;

subplot(3,1,2);
plot(time,y_gyro,'.');
hold on;
plot(time,y_gyro_mdl,'.r');
legend('Measured','Modeled');
xlabel('Time (sec)');
ylabel('Angular Velocity (rad/s)');
title('Y Axis Angular Velocity');
grid on;

subplot(3,1,3);
plot(time,z_gyro,'.');
hold on;
plot(time,z_gyro_mdl,'.r');
legend('Measured','Modeled');
xlabel('Time (sec)');
ylabel('Angular Velocity (rad/s)');
title('Z Axis Angular Velocity');
grid on;

suptitle('Gyroscope Modeling');

% Plot Amplitude Spectrum
% [f_acc_x, amp_acc_x] = myftt(x_acc_nobias, IMUPer, 0);
% [f_acc_xm, amp_acc_xm] = myftt(x_acc_mdl-mean(x_acc_mdl), IMUPer, 0);
%
% figure();
% title('Single-Sided Amplitude Spectrum');
% subplot(3,2,1);
% loglog(f_acc_x,amp_acc_x);
% hold on;
% loglog(f_acc_xm,amp_acc_xm,'r');
% xlabel('Frequency (Hz)');
% ylabel('Acceleration (m/s^2)');
% grid on;
```

ПРИЛОЖЕНИЕ Б

quadrotor_sim.m

```
clear all;
close all;
clc;
global Quad;
%% Initialize the plot
init_plot;
plot_quad_model;
%% Initialize Variables
quad_variables;
quad_dynamics_nonlinear;

%% Запуск симуляционного цикла
while Quad.t_plot(Quad.counter-1)< max(Quad.t_plot);

    % Measure Parameters (для моделирования ошибок датчика)
      sensor_meas;

    % Filter Measurements
%     Kalman_phi2;
%     Kalman_theta2;
%     Kalman_psi2;
%     Kalman_Z2;
%     Kalman_X2;
%     Kalman_Y2;

    % реализация контролеров
    position_PID;
    attitude_PID;
    rate_PID;
    % Вычисление желаемых скоростей двигателя
    quad_motor_speed;

    % Обновить положение с помощью уравнений движения
    quad_dynamics_nonlinear;

    % График положения квадрокоптера
    if(mod(Quad.counter,3)==0)
        plot_quad

%         campos([A.X+2 A.Y+2 A.Z+2])
%         camtarget([A.X A.Y A.Z])
%         camroll(0);
        Quad.counter;
        drawnow
    end

    Quad.init = 1;  % Заканчивает инициализацию после первой итерации моделирования
end

%% Plot Data
plot_data
```

Функция init_plot.m

```
% This function initializes the plots

function init_plot


% figure('units','normalized','position',[.1 .1 .8 .8],'name','Quadrotor
AUS','numbertitle','off','color','w');
axes('units','normalized','position',[.2 .1 .6 .8]);
axis equal

% E1 = uicontrol('units','normalized','position',[.11 .85 .1
.07],'style','edit','fontsize',13,'string',0,'backgroundcolor','w');
```

```
% % E2 = uicontrol('units','normalized','position',[.11 .75 .1
.07],'style','edit','fontsize',13,'string',0,'backgroundcolor','w');
% % E3 = uicontrol('units','normalized','position',[.11 .65 .1
.07],'style','edit','fontsize',13,'string',0,'backgroundcolor','w');
% E4 = uicontrol('units','normalized','position',[.11 .55 .1
.07],'style','edit','fontsize',13,'string',0,'backgroundcolor','w');
% E5 = uicontrol('units','normalized','position',[.11 .45 .1
.07],'style','edit','fontsize',13,'string',0,'backgroundcolor','w');
% E6 = uicontrol('units','normalized','position',[.11 .35 .1
.07],'style','edit','fontsize',13,'string',0,'backgroundcolor','w');

% uicontrol('units','normalized','position',[.02 .83 .05
.07],'style','text','fontsize',13,'string','Altitude','backgroundcolor','w');
% % uicontrol('units','normalized','position',[.02 .73 .05
.07],'style','text','fontsize',13,'string','Roll','backgroundcolor','w');
% % uicontrol('units','normalized','position',[.02 .63 .05
.07],'style','text','fontsize',13,'string','Pitch','backgroundcolor','w');
% uicontrol('units','normalized','position',[.02 .53 .05
.07],'style','text','fontsize',13,'string','Yaw','backgroundcolor','w');
% uicontrol('units','normalized','position',[.02 .43 .05
.07],'style','text','fontsize',13,'string','X','backgroundcolor','w');
% uicontrol('units','normalized','position',[.02 .33 .05
.07],'style','text','fontsize',13,'string','Y','backgroundcolor','w');

% uicontrol('units','normalized','position',[.11 .25 .1
.07],'style','pushbutton','fontsize',13,'string','Go','callback',@Go1);

% Motors speed
% uicontrol('units','normalized','position',[.85 .83 .05
.07],'style','text','fontsize',13,'string','Front M','backgroundcolor',[.5 .7 1]);
% uicontrol('units','normalized','position',[.85 .73 .05
.07],'style','text','fontsize',13,'string','Right M','backgroundcolor',[.5 .7 1]);
% uicontrol('units','normalized','position',[.85 .63 .05
.07],'style','text','fontsize',13,'string','Rear M','backgroundcolor',[.5 .7 1]);
% uicontrol('units','normalized','position',[.85 .53 .05
.07],'style','text','fontsize',13,'string','Left M','backgroundcolor',[.5 .7 1]);

% O1 = uicontrol('units','normalized','position',[.91 .86 .08
.05],'style','text','fontsize',13,'string','0','backgroundcolor','w');
% O2 = uicontrol('units','normalized','position',[.91 .76 .08
.05],'style','text','fontsize',13,'string','0','backgroundcolor','w');
% O3 = uicontrol('units','normalized','position',[.91 .66 .08
.05],'style','text','fontsize',13,'string','0','backgroundcolor','w');
% O4 = uicontrol('units','normalized','position',[.91 .56 .08
.05],'style','text','fontsize',13,'string','0','backgroundcolor','w');

% disturbances
% uicontrol('units','normalized','position',[.13+.77 .35 .08
.07],'style','pushbutton','fontsize',13,'string','Z','callback',@d1);
% uicontrol('units','normalized','position',[.02+.77 .35 .08
.07],'style','pushbutton','fontsize',13,'string','Yaw','callback',@d2);
% uicontrol('units','normalized','position',[.13+.77 .25 .08
.07],'style','pushbutton','fontsize',13,'string','Pitch','callback',@d3);
% uicontrol('units','normalized','position',[.02+.77 .25 .08
.07],'style','pushbutton','fontsize',13,'string','Roll','callback',@d4);

% axis([-5 5 -5 5 0 5])
axis([-2 2 -2 2 -2 2]);
view(30,30)
grid on
hold on

xlabel('x')

end
```

plot_quad_model.m

```
%% Хранение данных в глобальной переменной
load Quad_plotting_model
```

```
Quad.X_arm =
patch('xdata',Quad.X_armX,'ydata',Quad.X_armY,'zdata',Quad.X_armZ,'facealpha',.9,'face
color','b');
Quad.Y_arm =
patch('xdata',Quad.Y_armX,'ydata',Quad.Y_armY,'zdata',Quad.Y_armZ,'facealpha',.9,'face
color','b');
Quad.Motor1 =
patch('xdata',Quad.Motor1X,'ydata',Quad.Motor1Y,'zdata',Quad.Motor1Z,'facealpha',.3,'f
acecolor','g');
Quad.Motor2 =
patch('xdata',Quad.Motor2X,'ydata',Quad.Motor2Y,'zdata',Quad.Motor2Z,'facealpha',.3,'f
acecolor','k');
Quad.Motor3 =
patch('xdata',Quad.Motor3X,'ydata',Quad.Motor3Y,'zdata',Quad.Motor3Z,'facealpha',.3,'f
acecolor','k');
Quad.Motor4 =
patch('xdata',Quad.Motor4X,'ydata',Quad.Motor4Y,'zdata',Quad.Motor4Z,'facealpha',.3,'f
acecolor','k');
```

define_quad_model.m

```
global Quad

%% Определить начальные координаты плеч и роторов

Quad.plot_arm = .3;      % Расстояние от центра масс до каждого двигателя (m)
Quad.plot_arm_t = .02;    % Толщина кронштейнов квадротора (m)
Quad.plot_prop = .1;    % Радиус пропеллера (m)

% Вершины значения для оси Y (m).
x1=-Quad.plot_arm;
x2=Quad.plot_arm;
y1=-Quad.plot_arm_t;
y2=Quad.plot_arm_t;
z1=-Quad.plot_arm_t;
z2=Quad.plot_arm_t;

% Расположение вершин
vertex_matrix=[x1 y1 z1;
   x2 y1 z1;
   x2 y2 z1;
   x1 y2 z1;
   x1 y1 z2;
   x2 y1 z2;
   x2 y2 z2;
   x1 y2 z2];

% матрица граней, записи относятся к вершинам в каждой грани
face_matrix=[1 2 6 5;
     2 3 7 6;
     3 4 8 7;
     4 1 5 8;
     1 2 3 4;
     5 6 7 8];


 X_arm=patch('faces',face_matrix,...
     'vertices',vertex_matrix,...
     'facecolor','b',...
     'edgecolor',[0 0 0],'facecolor','b');

% Вершины значения для оси X (m)
y1=-Quad.plot_arm;
y2=Quad.plot_arm;
x1=-Quad.plot_arm_t;
x2=Quad.plot_arm_t;
z1=-Quad.plot_arm_t;
z2=Quad.plot_arm_t;
```

```
vertex_matrix=[x1 y1 z1;
   x2 y1 z1;
   x2 y2 z1;
   x1 y2 z1;
   x1 y1 z2;
   x2 y1 z2;
   x2 y2 z2;
   x1 y2 z2];

Y_arm=patch('faces',face_matrix,...
     'vertices',vertex_matrix,...
     'facecolor','b',...
     'edgecolor',[0 0 0],'facecolor','b');

 % Рисует область ротора
 t=0:pi/10:2*pi;    %точки окружности ротора
 X=Quad.plot_prop*cos(t);
 Y=Quad.plot_prop*sin(t);
 Z=zeros(size(t))+Quad.plot_arm_t;
 C=zeros(size(t));
 Quad.C = C;

 Motor1 = patch(X+Quad.plot_arm,Y,Z,C,'facealpha',.3,'facecolor','g'); %(Motor 1)
 Motor2 = patch(X,Y-Quad.plot_arm,Z,C,'facealpha',.3,'facecolor','k'); %(Motor 2)
 Motor3 = patch(X-Quad.plot_arm,Y,Z,C,'facealpha',.3,'facecolor','k'); %(Motor 3)
 Motor4 = patch(X,Y+Quad.plot_arm,Z,C,'facealpha',.3,'facecolor','k'); %(Motor 4)

 %Координаты каждой грани в каждой оси (4 координаты x 6 граней)
 Quad.X_armX = get(X_arm,'xdata');
 Quad.X_armY = get(X_arm,'ydata');
 Quad.X_armZ = get(X_arm,'zdata');

 Quad.Y_armX = get(Y_arm,'xdata');
 Quad.Y_armY = get(Y_arm,'ydata');
 Quad.Y_armZ = get(Y_arm,'zdata');

 %Координаты каждого ротора по каждой оси (21 координата x 1 грань)
 Quad.Motor1X = get(Motor1,'xdata');
 Quad.Motor1Y = get(Motor1,'ydata');
 Quad.Motor1Z = get(Motor1,'zdata');

 Quad.Motor2X = get(Motor2,'xdata');
 Quad.Motor2Y = get(Motor2,'ydata');
 Quad.Motor2Z = get(Motor2,'zdata');

 Quad.Motor3X = get(Motor3,'xdata');
 Quad.Motor3Y = get(Motor3,'ydata');
 Quad.Motor3Z = get(Motor3,'zdata');

 Quad.Motor4X = get(Motor4,'xdata');
 Quad.Motor4Y = get(Motor4,'ydata');
 Quad.Motor4Z = get(Motor4,'zdata');
```

quad_variables.m

```
global Quad;

%% Инициализировать переменные

% Параметры моделирования
Quad.init = 0; %, используемое при инициализации
Quad.Ts = .01; % Время выборки (100 Гц)
Quad.sim_time = 10; % Время моделирования (в секундах)
Quad.counter = 1; % счетчик, который содержит значение времени

% Перечисление переменных
Quad.t_plot = [0: Quad.Ts: Quad.sim_time-Quad.Ts]; % значения времени
```

```
Quad.Xtemp = 0; % Temp-переменные, используемые для поворота и построения квадратора
Quad.Ytemp = 0; % Temp-переменные, используемые для поворота и построения квадратора
Quad.Ztemp = 0; % Temp-переменные, используемые для поворота и построения квадратора

% Экологические параметры
Quad.g = 9,81; % Гравитация (м / с ^ 2)

% Физические параметры коптера
Quad.m = 0,024; % Квадротронная масса (кг)
Quad.l = .0056; % Расстояние от центра масс до каждого двигателя (м)
Quad.t = .002; % Толщина кронштейнов квадротора для графика (м)
Quad.rot_rad = .1; % Радиус пропеллера (м)
Quad.Kd = 1,39 * 10 ^ (- 6); % Коэффициент крутящего момента (кг-м ^ 2)

Quad.Kdx = 0.16481; % Коэффициент силы поступательного сопротивления (кг / с)
Quad.Kdy = 0,31892; % Коэффициент силы поступательного сопротивления (кг / с)
Quad.Kdz = 1,1 * 10 ^ (- 6); % Коэффициент силы поступательного сопротивления (кг / с)

Quad.Jx = .05; % Момент инерции вокруг оси X (кг-м ^ 2)
Quad.Jy = .05; % Момент инерции относительно оси Y (кг-м ^ 2)
Quad.Jz = .24; % Момент инерции относительно оси Z (кг-м ^ 2)

% Параметры датчиков квадрокоптера
Quad.GPS_freq = (1 / Quad.Ts) / 1;
Quad.X_error = .01; % + / - m
Quad.Y_error = .01; % + / - m
Quad.Z_error = .02; % + / - m

Quad.x_acc_bias = 0.16594; % м / с ^ 2
Quad.x_acc_sd = 0,0093907;
Quad.y_acc_bias = 0,31691; % м / с ^ 2
Quad.y_acc_sd = 0.011045;
Quad.z_acc_bias = 8.6759; % м / с ^ 2
Quad.z_acc_sd = 0.016189;

Quad.x_gyro_bias = 0.00053417; % рад / с
Quad.x_gyro_sd = 0.00066675;
Quad.y_gyro_bias = 0,0011035; % рад / с
Quad.y_gyro_sd = 0.00053642;
Quad.z_gyro_bias = 0.00020838; % рад / с
Quad.z_gyro_sd = 0.0004403;

Quad.ground_truth = 1; % Использовать идеальные измерения датчика
Quad.sensor_unfiltered = 0; % Использовать ошибки датчика, нет фильтра
Quad.sensor_kf = 0; % Использовать ошибку датчика, фильтр Калмана

% Параметры двигателя
Quad.KT = 1.3328 * 10 ^ (- 5); % Коэффициент силы тяги (кг-м)
Quad.Jp = 0,044; % Момент взаимодействия ротора (кг-м ^ 2)
Quad.max_motor_speed = 925; % верхних пределов двигателей (рад / с)
Quad.min_motor_speed = 0; % -1 * ((400) ^ 2); % нижнего предела двигателей (не может
вращаться в обратном направлении)

Quad.Obar = 0; % суммы скоростей двигателя (O1-O2 + O3-O4, N-m)
Quad.O1 = 0; % Скорость переднего двигателя (рейданс / с)
Quad.O2 = 0; % Правая скорость двигателя (рейданс / с)
Quad.O3 = 0; % Задняя скорость двигателя (рейданс / с)
Quad.O4 = 0; % Левая скорость двигателя (рейданс / с)

% Поступательные позиции
Quad.X = 0; % Исходное положение в направлении X GF (м)
Quad.Y = 0; % Исходное положение в направлении Y GF (м)
Quad.Z = 0; % Исходное положение в направлении Z GF (м)
Quad.X_BF = 0; % Исходное положение в направлении X BF (м)
Quad.Y_BF = 0; % Исходное положение в направлении Y BF (м)
Quad.Z_BF = 0; % Исходное положение в направлении Z BF (м)

% Поступательные скорости
Quad.X_dot = 0; % Начальная скорость в направлении X GF (м / с)
Quad.Y_dot = 0; % Начальная скорость в направлении Y GF (м / с)
```

```
Quad.Z_dot = 0; % Начальная скорость в направлении Z GF (м / с)
Quad.X_dot_BF = 0; % Начальная скорость в направлении X BF (м / с)
Quad.Y_dot_BF = 0; % Начальная скорость в направлении Y BF (м / с)
Quad.Z_dot_BF = 0; % Начальная скорость в направлении Y BF (м / с)

% Угловых положений
Quad.phi = 0; % Начальное значение phi (вращение вокруг X GF, крен, радианы)
Quad.theta = 0; % Начальное значение theta (вращение вокруг Y GF, тангаж, радианы)
Quad.psi = 0; % Начальное значение psi (вращение вокруг Z GF, рыскание, радианы)

% Угловых скоростей
Quad.p = 0; % Начальное значение p (вращение угловой скорости вокруг X BF, радианы /
с)
Quad.q = 0; % Начальное значение q (вращение угловой скорости вокруг Y BF, радианы /
с)
Quad.r = 0; % Начальное значение r (вращение угловой скорости вокруг Z BF, радианы /
с)

% Желаемые переменные
Quad.X_des_GF = 1; % желаемого значения X в глобальной СК
Quad.Y_des_GF = 1; % желаемого значения Y в глобальной СК
Quad.Z_des_GF = 1; % желаемого значения Z в глобальной СК
Quad.X_des = 0; % желаемого значения X в СК тела
Quad.Y_des = 0; % желаемого значения Y в СК тела
Quad.Z_des = 0; % желаемого значения Z в СК тела

Quad.phi_des = 0; % желаемого значения phi (радианы)
Quad.theta_des = 0; % желаемого значения theta (радианы)
Quad.psi_des = pi / 6; % желаемого значения psi (радиан)

% Измеренные переменные
Quad.X_meas = 0;
Quad.Y_meas = 0;
Quad.Z_meas = 0;
Quad.phi_meas = 0;
Quad.theta_meas = 0;
Quad.psi_meas = 0;
% Переменные возмущения
Quad.Z_dis = 0; % Нарушение в направлении Z
Quad.X_dis = 0; % Нарушение в направлении X
Quad.Y_dis = 0; % Нарушение в направлении Y
Quad.phi_dis = 0; % Нарушение направления движения
Quad.theta_dis = 0; % Нарушение в направлении высоты тангажа
Quad.psi_dis = 0; % Нарушение в направлении крена

% Управляющие входы
Quad.U1 = 0; % Общая тяга (N)
Quad.U2 = 0; % Крутящий момент вокруг оси X BF (N-m)
Quad.U3 = 0; % Крутящий момент вокруг оси Y BF (N-m)
Quad.U4 = 0; % Крутящий момент вокруг оси Z BF (N-m)

% Пределы управления
Quad.U1_max = 45,5; % Quad.KT * 4 * Quad.max_motor_speed ^ 2
Quad.U1_min = 0; %
Quad.U2_max = 6.37; % Quad.KT * Quad.l * Quad.max_motor_speed ^ 2
Quad.U2_min = -6,37; % Quad.KT * Quad.l * Quad.max_motor_speed ^ 2
Quad.U3_max = 6.37; % Quad.KT * Quad.l * Quad.max_motor_speed ^ 2
Quad.U3_min = -6,37; % Quad.KT * Quad.l * Quad.max_motor_speed ^ 2
Quad.U4_max = 2,38; % Quad.Kd * 2 * Quad.max_motor_speed ^ 2
Quad.U4_min = -2.38;% Quad.Kd * 2 * Quad.max_motor_speed ^ 2

% ПИД-параметры
Quad.X_KP = .35; % KP значение в X-положении
Quad.X_KI = .25; % Значение KI при управлении положением X
Quad.X_KD = -35; % KD-значение в X-положении
Quad.X_KI_lim = .25; % Ошибка начала вычисления интегрального члена

Quad.Y_KP = .35; % KP значение в Y-положении
Quad.Y_KI = .25; % Значение KI в Y-положении
Quad.Y_KD = -35; %Значение KD в Y-положении
```

```
Quad.Y_KI_lim = .25; % Ошибка начала вычисления интегрального члена

Quad.Z_KP = 10 / 1.7; % KP в контроле высоты
Quad.Z_KI = 0 * 3; %Значение KI при управлении высотой
Quad.Z_KD = -10 / 1.980; %Значение KD при управлении высотой
Quad.Z_KI_lim = .25; % Ошибка начала вычисления интегрального члена

Quad.phi_KP = 4,5; % КП в крене 2
Quad.phi_KI = 0; %Значение KI в крене 1
Quad.phi_KD = 0; % KD в крене -5
Quad.phi_max = pi / 4; % Максимальный угол наклона крена
Quad.phi_KI_lim = 2 * (2 * pi / 360); % Ошибка при запуске вычисления интеграла

Quad.theta_KP = 4,5; % Значение КП в управлении тангажем 2
Quad.theta_KI = 0; % Значение KI в управлении тангажем 1
Quad.theta_KD = 0; % Значение KD при регулировании высоты тангажа -5
Quad.theta_max = pi / 4; % Максимальный угол тангажа
Quad.theta_KI_lim = 2 * (2 * pi / 360); % Ошибка при запуске вычисления интеграла

Quad.psi_KP = 10; % KP значение в управлении рысканием
Quad.psi_KI = 0; % KI значение в управлении рысканием .75
Quad.psi_KD = 0; % KD-значение в регулировании рыскания -5
Quad.psi_KI_lim = 8 * (2 * pi / 360); % Ошибка при запуске вычисления интеграла

Quad.p_KP = 2,7; % Значение КП в управлении тангажем 2
Quad.p_KI = 1; % Значение KI в управлении тангажем
Quad.p_KD = -.01; % Значение KD при регулировании высоты тангажем -5
Quad.p_max = 50 * (2 * pi / 360); % Максимальный угол тангажа
Quad.p_KI_lim = 10 * (2 * pi / 360); % Ошибка при запуске вычисления интеграла

Quad.q_KP = 2,7; % KP значение в управлении тангажем
Quad.q_KI = 1; % Значение KI в управлении тангажем
Quad.q_KD = -.01; % Значение KD при регулировании высоты тангажем -5
Quad.q_max = 50 * (2 * pi / 360); % Максимальный угол тангажа
Quad.q_KI_lim = 10 * (2 * pi / 360); % Ошибка при запуске вычисления интеграла

Quad.r_KP = 2,7; % KP значение в управлении тангажем
Quad.r_KI = 1; % Значение KI в управлении тангажем
Quad.r_KD = -.01; %Значение KD в управлении тангажем
Quad.r_max = 50 * (2 * pi / 360); % Максимальный угол тангажа
Quad.r_KI_lim = 10 * (2 * pi / 360); % Ошибка при запуске вычисления интеграла
```

Функция quad_dynamics_nonlinear.m

```
function quad_dynamics_nonlinear
global Quad;

%% Обновление ускорения

Quad.X_ddot = (-
[cos(Quad.phi)*sin(Quad.theta)*cos(Quad.psi)+sin(Quad.phi)*sin(Quad.psi)]*Quad.U1-
Quad.Kdx*Quad.X_dot)/Quad.m;
Quad.Y_ddot = (-[cos(Quad.phi)*sin(Quad.psi)*sin(Quad.theta)-
cos(Quad.psi)*sin(Quad.phi)]*Quad.U1-Quad.Kdy*Quad.Y_dot)/Quad.m;
Quad.Z_ddot = (-[cos(Quad.phi)*cos(Quad.theta)]*Quad.U1-
Quad.Kdz*Quad.Z_dot)/Quad.m+Quad.g;

Quad.p_dot = (Quad.q*Quad.r*(Quad.Jy - Quad.Jz) - Quad.Jp*Quad.p*Quad.Obar +
Quad.l*Quad.U2)/Quad.Jx;
Quad.q_dot = (Quad.p*Quad.r*(Quad.Jz - Quad.Jx) + Quad.Jp*Quad.q*Quad.Obar +
Quad.l*Quad.U3)/Quad.Jy;
Quad.r_dot = (Quad.p*Quad.q*(Quad.Jx - Quad.Jy) + Quad.U4)/Quad.Jz;

Quad.phi_dot   = Quad.p + sin(Quad.phi)*tan(Quad.theta)*Quad.q +
cos(Quad.phi)*tan(Quad.theta)*Quad.r;
Quad.theta_dot = cos(Quad.phi)*Quad.q - sin(Quad.phi)*Quad.r;
Quad.psi_dot   = sin(Quad.phi)/cos(Quad.theta)*Quad.q +
cos(Quad.phi)/cos(Quad.theta)*Quad.r;
```

```
%% Модель возмущений

Quad.X_ddot = Quad.X_ddot + Quad.X_dis/Quad.m;
Quad.Y_ddot = Quad.Y_ddot + Quad.Y_dis/Quad.m;
Quad.Z_ddot = Quad.Z_ddot + Quad.Z_dis/Quad.m;
Quad.phi_dot = Quad.phi_dot + Quad.phi_dis/Quad.Jx*Quad.Ts;
Quad.theta_dot = Quad.theta_dot + Quad.theta_dis/Quad.Jy*Quad.Ts;
Quad.psi_dot = Quad.psi_dot + Quad.psi_dis/Quad.Jz*Quad.Ts;

%% Обновление скорости и позиций

% Расчет скорости и положения Z
Quad.Z_dot = Quad.Z_ddot*Quad.Ts + Quad.Z_dot;
Quad.Z = Quad.Z_dot*Quad.Ts + Quad.Z;

% Расчет скорости и положения X
Quad.X_dot = Quad.X_ddot*Quad.Ts + Quad.X_dot;
Quad.X = Quad.X_dot*Quad.Ts + Quad.X;

% Расчет скорости и положения Y
Quad.Y_dot = Quad.Y_ddot*Quad.Ts + Quad.Y_dot;
Quad.Y = Quad.Y_dot*Quad.Ts + Quad.Y;

% Расчет p,q,r
Quad.p = Quad.p_dot*Quad.Ts+Quad.p;
Quad.q = Quad.q_dot*Quad.Ts+Quad.q;
Quad.r = Quad.r_dot*Quad.Ts+Quad.r;

% Расчет угловой скорости и положения
Quad.phi = Quad.phi_dot*Quad.Ts + Quad.phi;
Quad.theta = Quad.theta_dot*Quad.Ts+Quad.theta;
Quad.psi = Quad.psi_dot*Quad.Ts+Quad.psi;

%% Обновить значения переменных

% Отклонение положительной оси Z для графика
Quad.Z_plot(Quad.counter) = -Quad.Z;
Quad.Z_ref_plot(Quad.counter) = -Quad.Z_des;

Quad.X_plot(Quad.counter) = Quad.X;
Quad.X_ref_plot(Quad.counter) = Quad.X_des;

Quad.Y_plot(Quad.counter) = Quad.Y;
Quad.Y_ref_plot(Quad.counter) = Quad.Y_des;

Quad.phi_plot(Quad.counter) = Quad.phi;
Quad.phi_ref_plot(Quad.counter) = Quad.phi_des;

Quad.theta_plot(Quad.counter) = Quad.theta;
Quad.theta_ref_plot(Quad.counter) = Quad.theta_des;

Quad.psi_plot(Quad.counter) = Quad.psi;
Quad.psi_ref_plot(Quad.counter) = Quad.psi_des;

Quad.counter = Quad.counter + 1;

end
```

ПРИЛОЖЕНИЕ В

Файл для записи данных гироскопа.

```
// -*- tab-width: 4; Mode: C++; c-basic-offset: 4; indent-tabs-mode: nil -*-
//
#include <stdarg.h>
#include <AP_Common.h>
#include <AP_Progmem.h>
#include <AP_HAL.h>
#include <AP_HAL_AVR.h>
#include <AP_HAL_AVR_SITL.h>
#include <AP_HAL_Linux.h>
#include <AP_HAL_FLYMAPLE.h>
#include <AP_HAL_PX4.h>
#include <AP_HAL_Empty.h>
#include <AP_Math.h>
#include <AP_Param.h>
#include <StorageManager.h>
#include <AP_ADC.h>
#include <AP_InertialSensor.h>
#include <AP_Notify.h>
#include <AP_GPS.h>
#include <AP_Baro.h>
#include <Filter.h>
#include <DataFlash.h>
#include <GCS_MAVLink.h>
#include <AP_Mission.h>
#include <StorageManager.h>
#include <AP_Terrain.h>
#include <AP_AHRS.h>
#include <AP_Airspeed.h>
#include <AP_Vehicle.h>
#include <AP_ADC_AnalogSource.h>
#include <AP_Compass.h>
#include <AP_Declination.h>
#include <AP_NavEKF.h>
#include <AP_HAL_Linux.h>
#include <AP_Rally.h>
#include <AP_Scheduler.h>

const AP_HAL::HAL& hal = AP_HAL_BOARD_DRIVER;
AP_InertialSensor ins;
void setup(void)
{
   hal.console->println("AP_InertialSensor startup...");

#if CONFIG_HAL_BOARD == HAL_BOARD_APM2
   // we need to stop the barometer from holding the SPI bus
   hal.gpio->pinMode(40, HAL_GPIO_OUTPUT);
   hal.gpio->write(40, 1);
#endif

   ins.init(AP_InertialSensor::COLD_START,
                                    AP_InertialSensor::RATE_100HZ);
   // display initial values
   display_offsets_and_scaling();
   hal.console->println("Complete. Reading:");
}
void loop(void)
{
   int16_t user_input;

   hal.console->println();
   hal.console->println_P(PSTR(
```

```
    "Menu:\r\n"
    "    c) calibrate accelerometers\r\n"
    "    d) display offsets and scaling\r\n"
    "    l) level (capture offsets from level)\r\n"
    "    t) test\r\n"
    "    r) reboot"));

    // wait for user input
    while( !hal.console->available() ) {
        hal.scheduler->delay(20);
    }
    // read in user input
    while( hal.console->available() ) {
        user_input = hal.console->read();

        if( user_input == 'c' || user_input == 'C' ) {
            run_calibration();
            display_offsets_and_scaling();
        }

        if( user_input == 'd' || user_input == 'D' ) {
            display_offsets_and_scaling();
        }

        if( user_input == 'l' || user_input == 'L' ) {
            run_level();
            display_offsets_and_scaling();
        }

        if( user_input == 't' || user_input == 'T' ) {
            run_test();
        }

        if( user_input == 'r' || user_input == 'R' ) {
                                        hal.scheduler->reboot(false);
        }
    }
}
void run_calibration()
{
    float roll_trim, pitch_trim;
    // clear off any other characters (like line feeds,etc)
    while( hal.console->available() ) {
        hal.console->read();
    }
#if !defined( __AVR_ATmega1280__ )
    AP_InertialSensor_UserInteractStream interact(hal.console);
    ins.calibrate_accel(&interact, roll_trim, pitch_trim);
#else
            hal.console->println_P(PSTR("calibrate_accel not available on 1280"));
#endif
}
void display_offsets_and_scaling()
{
    Vector3f accel_offsets = ins.get_accel_offsets();
    Vector3f accel_scale = ins.get_accel_scale();
    Vector3f gyro_offsets = ins.get_gyro_offsets();

    // display results
    hal.console->printf_P(
        PSTR("\nAccel Offsets X:%10.8f \t Y:%10.8f \t Z:%10.8f\n"),
                accel_offsets.x,
                accel_offsets.y,
                accel_offsets.z);
    hal.console->printf_P(
```

```
        PSTR("Accel Scale X:%10.8f \t Y:%10.8f \t Z:%10.8f\n"),
             accel_scale.x,
             accel_scale.y,
             accel_scale.z);
    hal.console->printf_P(
        PSTR("Gyro Offsets X:%10.8f \t Y:%10.8f \t Z:%10.8f\n"),
             gyro_offsets.x,
             gyro_offsets.y,
             gyro_offsets.z);
}
void run_level()
{
    // clear off any input in the buffer
    while( hal.console->available() ) {
        hal.console->read();
    }
    // display message to user
    hal.console->print("Place APM on a level surface and press any key..\n");
    // wait for user input
    while( !hal.console->available() ) {
        hal.scheduler->delay(20);
    }
    while( hal.console->available() ) {
        hal.console->read();
    }
    // run accel level
    ins.init_accel();

    // display results
    display_offsets_and_scaling();
}
void run_test()
{
    Vector3f accel;
    Vector3f gyro;
    float length;
            uint8_t counter = 0;
    // flush any user input
    while( hal.console->available() ) {
        hal.console->read();
    }
    // clear out any existing samples from ins
    ins.update();
    // loop as long as user does not press a key
    while( !hal.console->available() ) {
        // wait until we have a sample
        ins.wait_for_sample();
        // read samples from ins
        ins.update();
        accel = ins.get_accel();
        gyro = ins.get_gyro();
        length = accel.length();
                        if (counter++ % 1 == 0) {
                                // display results
                                hal.console->printf_P(PSTR("Accel X:%4.2f \t Y:%4.2f \t Z:%4.2f \t len:%4.2f \t Gyro X:%4.2f \t
Y:%4.2f \t Z:%4.2f\n"),
                                                                                        accel.x, accel.y, accel.z, length, gyro.x, gyro.y, gyro.z);
                        }
    }
    // clear user input
    while( hal.console->available() ) {
        hal.console->read();
    }
}
AP_HAL_MAIN();
```

ПРИЛОЖЕНИЕ Г

Техническая документация CrazyFlie 2.0

Crazyflie 2.0 is a versatile flying development platform that only weights 27g and fits in the palm of your hand. It's advanced functionalities makes it ideal for developers and the Bluetooth LE capabilities makes it easy to fly from mobile devices. With it's small size and weight it's ideal for indoor use, but you can just as easily hover above your house as you can hover under your diningroom table. Designed as a solderless kit, the Crazyflie 2.0 is quickly assembled by attaching the motors to the circuitboard frame and it's ready to fly.

The Crazyflie 2.0 is an open project, with source code and hardware design available and documented. The platform is designed with development in mind, implementing features to make development easier and faster, such as logging and real-time parameter setting and wireless firmware update. The complete development environment for most of the projects is available in the virtual machine, so you don't need to install any toolchains to get into the development. But the virtual machine can just as well be used for flying. Aside from the firmware and software projects, there is also a number of community supported APIs written in Java, Ruby, C/C++, C# and Javascript. For anyone interested in doing more advanced development there is a development adapter kit that supports easy JTAG/SWD connection to both of the MCUs on the Crazyflie 2.0.

Supporting multiple radio protocols, the Crazyflie 2.0 can be used from a Bluetooth LE enabled mobile device or from a computer using the CrazyradioorCrazyradio PA. While flying from a mobile device works great, the real power of the platform is unlocked by connecting it to a computer using the Python client that's a available for Windows, Mac OSX and Linux. This enables you to fully use all the expansion boards, to easily trim flying parameters, graphically log data and set parameters. When connected to a computer, you also get the added benifit of being able to use any gamepad or joystick with at least 4 analog axis for flying. The device can easily be mapped inside the client.
The firmware and software is continously being updated with various improvements and new features added. The platform supports wireless firmware updates via radio and Bluetooth LE, so when a new new firmware is released it's a breeze to update it.

The Crazyflie 2.0 features a 2x10 pin expansion port, where you can connect expansion boards. Either you could use one of the expansion boards, or you could design your own using the Prototype expansion boardor Breakout expansion board.

The Crazyflie 2.0 comes as a solderless kit and has to be assembled. Please see below in the Resource section for links for the assembly instructions.

Features:
Durable design
Easy to assemble and no soldering required
Supports expansion boards with automatic detection
Supports flying from iOS and Android with Bluetooth LE support as well as from Windows/MacOSX/Linux with the Crazyradio or Crazyradio PA
Tested to above 1 km radio range LOS with Crazyradio PA
Wireless firmware update
On-board charging via standard uUSB

Dual-MCU architecture with dedicated radio/power management SoC for advanced applications
Using Crazyradio or Crazyradio PA tougether with a computer the user can log/graph/set variables in real-time via the radio and make full use of the expansion boards
Size (WxHxD): 92x92x29mm (motor-to-motor and including motor mount feet)
20 dBm radio amplifier tested to > 1 km range LOS with Crazyradio PA
Bluetooth Low Energy support with iOS and Android clients available (tested on iOS 7.1+ and Android 4.4+)
Radio backwards compatible with original Crazyflie and Crazyradio
STM32F405 main application MCU (Cortex-M4, 168MHz, 192kb SRAM, 1Mb flash)
nRF51822 radio and power management MCU (Cortex-M0, 32Mhz, 16kb SRAM, 128kb flash)
On-board LiPo charger with 100mA, 500mA and 980mA modes available
Full speed USB device interface
Partial USB OTG capability (Usb OTG present but no 5V output)
3 axis gyro (MPU-9250)
3 axis accelerometer (MPU-9250)
3 axis magnetometer (MPU-9250)
high precision pressure sensor (LPS25H)
Flight time with stock battery: 7 minutes
Charging time with stock battery: 40 minutes
Max recommended payload weight: 15 g
Expansion connector with: VCC (3.0V, max 100mA), GND, VCOM (unregulated VBAT or VUSB, max 1A), VUSB (both for input and output)
I2C (400kHz)
SPI
2 x UART
4 x GPIO/CS for SPI
1-wire bus for expansion identification
2 x GPIO connected to nRF51
8KB EEPROM

Includes:

1x Crazyflie 2.0 control board with all components mounted
5x CW propellers
5x CCW propellers
5x Motor mounts
1x LiPo battery (240mAh)
5x Coreless DC motors
2x Short expansion connector pins (1x10, 2mm spacing, 8 mm long)
2x Long expansion connector pins (1x10, 2mm spacing, 14 mm long)
1x Battery holder expansion board

Магистерская диссертация выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*