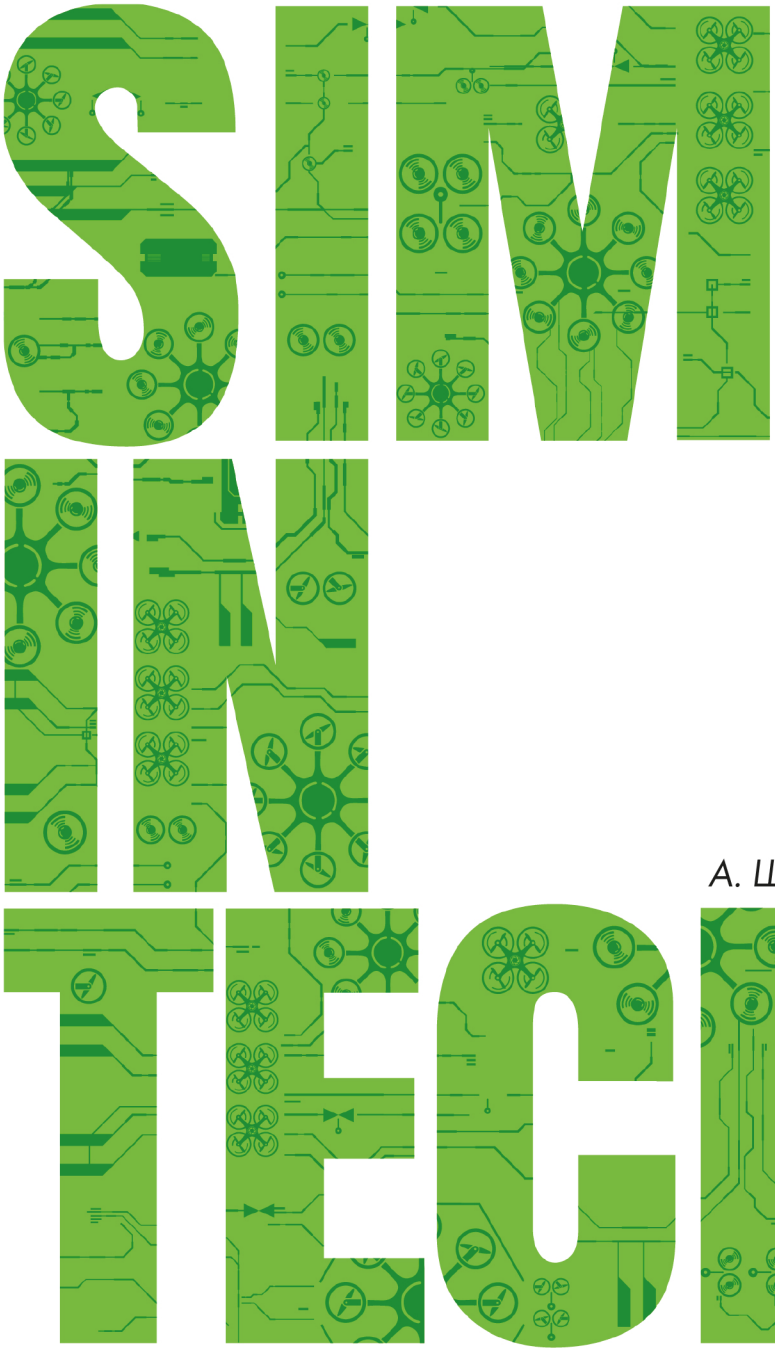


А. Щекатуров

Методика моделирования динамики октокоптера

Практикум по моделированию систем автоматического регулирования

ДМК Пресс
издательство

А. М. Щекатуров

Методика моделирования динамики октокоптера

ДМК Пресс
издательство
Москва, 2021

УДК 629.7.02
ББК 39.52
Щ37

Щекатуров А. М.
Щ37 Методика моделирования динамики октокоптера. – М.: ДМК Пресс, 2021. – 228 с.

ISBN 978-5-97060-933-0

Настоящая методика написана для пользователей, осваивающих среду динамического моделирования SimInTech, перед которыми стоит задача реализации математической модели динамики какого-либо технического объекта, записанной в форме системы нелинейных дифференциальных и алгебраических уравнений, с целью проведения расчетов и отладки алгоритмической части системы управления объектом.

На примере задачи моделирования динамики октокоптера представлено пошаговое описание процесса реализации такой модели в SimInTech, моделирования системы управления и регулирования для октокоптера. Выполнен последовательный переход от теоретической проработки модели к её реализации в виде конкретной расчетной схемы во входо-выходных соотношениях.

Методика предназначена для студентов технических специальностей, выпускников ВУЗов или инженеров, желающих самостоятельно научиться математическому моделированию динамики в среде SimInTech на примере беспилотного летательного аппарата коптерного типа. Она может быть полезна также и тем, кто хочет самостоятельно разработать свою динамическую модель беспилотного летательного аппарата.

УДК 629.7.02
ББК 39.52

ISBN 978-5-97060-933-0

Содержание

▼ 5

▼ 6

▼ 7

▼ 8

▼ 9

▼ 10

▼ 11

Введение

В настоящей методике изложены теоретические основы и подход к моделированию динамики полета октокоптера, во многом сходные с работами [1] и [2], а также реализация данного подхода с пошаговыми пояснениями в среде динамического моделирования SimInTech.

Если вам требуется реализовать модель динамики БПЛА или синтезировать регулятор высоты коптера со стабилизатором положения коптера в пространстве или даже спроектировать свой собственный полетный контроллер, а вы ничего про это не знаете, то данная методика послужит одной из хороших «точек входа» в тему.

В отечественной среде динамического моделирования SimInTech можно сделать многое в плане моделирования технических устройств, а также проектирования систем автоматического регулирования и управления этими устройствами. Практически любую модель динамики, в которой есть d/dt той или иной физической характеристики, можно легко реализовать в этой среде. Для этого первым делом необходимо сформулировать уравнения динамики моделируемого объекта (или найти их в литературе, понять, осмыслить и реализовать в среде моделирования). В сети и литературе есть довольно много описаний моделей коптеров, относительно простых, например [3], и ряд средней степени сложности; настоящая методика ориентируется на подход к моделированию, изложенный в [1] и [2], как наиболее близкий к модельно-ориентированному подходу к проектированию системы управления.

Подробное изложение теоретических аспектов модели приведено в первой части методики. С некоторыми модификациями изложенный подход пригоден и для моделирования квадрокоптера и гексакоптера. Однако в этих случаях исходные уравнения придется самостоятельно заново вывести и корректно спроецировать на оси координат.

Во второй части методики описана последовательность действий, которые необходимо совершить в среде SimInTech, а также ручкой или карандашом на листе бумаги, для того чтобы создать упрощенную, но близкую к реальности модель динамики БПЛА (а именно – октокоптера), модель его регуляторов (ориентации, высоты и положения в пространстве), макет пульта управления, небольшую техническую анимацию и набор графиков, для удобства отладки создаваемой модели.

Перечень сокращений

АСУ ТП – автоматизированная система управления;
БД – база данных (сигналов);
ВМГ – винтомоторная группа;
БПЛА – беспилотный летательный аппарат;
B – индекс системы координат, связанной с телом;
I – система координат инерциальная (связанная с Землей).

1 Постановка задачи

Модель октокоптера будем создавать в обобщенном виде, записывая параметры и характеристики объекта теми или иными переменными. Однако, чтобы модель «заработала», необходимо дать этим переменным какие-то конкретные значения. Также геометрическая конструкция тоже будет задана определенным образом, иначе невозможно получить конкретные уравнения сил и моментов.

Дано: объект массой порядка 10…25 кг, с 4, 6 или 8 парными соосными винтомоторными группами (ВМГ), расположенными по традиционной схеме квадро-, гекса- или октокоптера на жесткой раме, с 8, 12 или 16 двигателями типа (например) T-motors Antigravity 6007 KV320 или аналогичными и соответствующими им винтами (исходим из силы тяги одной ВМГ порядка 2 кг на 50 % газа). Конструкция аппарата до конца не определена.

Требуется: сконструировать и реализовать модель динамики объекта параметрическим способом, в общем виде и в объеме, достаточном для проектирования полетного контроллера и наземного пульта управления коптером.

В настоящей методике приведено описание логически завершенной работы по поставленной задаче. Этой части недостаточно для построения полноценного полетного контроллера, но достаточно для ознакомления с основными возможностями среды SimInTech и с введением в тему динамического моделирования БПЛА «коптерного» типа. Здесь опущено моделирование эффектов прецессии, принимается, что реактивный момент каждой ВМГ равен нулю, а именно каждая ВМГ имеет два двигателя и винта, вращающихся всегда с равной скоростью в противоположные стороны. Не моделируются отказы оборудования, и предполагается, что объект находится только в воздухе (в штатном режиме полета); режимы посадки и взлета, аварийные ситуации, захват груза и разгрузка в приведенной модели не реализованы, а также не рассматриваются вопросы подробного моделирования датчиков, фильтрации сигналов и шумов, изгиб рамы коптера и/или винтов, работа на непроектных нагрузках, написание драйверов к той или иной аппаратуре и т. д. и т. п. Однако при необходимости приведенная модель является легко расширяемой и может стать основой для проектирования реальной прошивки полетного контроллера.

Модель, реализованная в настоящей методике, заканчивается на этапе, когда на ней можно отлаживать регуляторы и качество переходных процессов, а также реализовывать алгоритмы управления коптером (полет по заданным координатам, автоматическое выполнение тех или иных маневров и т. д.).

Настоящая методика имеет три цели: а) познакомить читателя с возможностями SimInTech, б) рассказать об одном из подходов к моделированию беспилотного летательного аппарата коптерного типа и к проектированию его системы управления и в) на примере модели октокоптера познакомить читателя и научить основным возможностям SimInTech по реализации математической модели в схеме общего вида (схеме автоматики), когда исходные и не самые простые уравнения динамики имеются в наличии у разработчика. Таким образом, здесь приведены теоретические сведения о моделировании коптеров, подход к реализации такой модели на базе среды SimInTech и пошаговое выполнение действий, необходимых для реализации модели. Для краткости изложения будет рассмотрена только модель октокоптера (две другие при необходимости можно получить и вывести самостоятельно как частные случаи октокоптера).

2 Основные положения модели

В литературе и сети приведено довольно много моделей квадрокоптеров, есть некоторые модели гекса- и октокоптеров. Однако изложенного в системном и методичном виде со всеми подробностями практически ничего нет (по крайней мере, в русскоязычном сегменте, из тех материалов, что удалось найти). Наиболее методично теоретический подход к моделированию мультироторного БПЛА изложен в работах [1] и [2].

Опуская некоторые выкладки, изложенные в этих работах (они легко гуглятся), можно выделить следующие допущения и упрощения, принятые в модели.

1. Двигатель моделируется как инерционное (апериодическое) звено первого порядка, на вход которому подается заданное значение угловой скорости, а на выходе получаем текущее (измеренное) значение угловой скорости вращения. Интегрируя скорость вращения, можно получить текущий угол поворота вала двигателя. Силу тяги ВМГ развивает пропорционально квадрату угловой скорости (равно как и реактивный момент каждого двигателя, но в нашем случае он будет 0), а именно:

$$\begin{aligned} F_M(t) &= C_T \cdot \omega^2(t), \\ M_M(t) &= C_Q \cdot \omega^2(t), \end{aligned} \tag{2.1.1}$$

где для выбранной винтомоторной группы (согласно ее характеристике) принимаем $C_T = 2.0227 \cdot 10^{-4}$ Н · с^2, а $M_M \approx 0$ Н · м · с^2, $\omega(t)$ – текущая угловая скорость, рад/с.

2. Мультироторный аппарат моделируется как твердое тело и представляет собой жесткую (недеформируемую) раму постоянной массы, симметричную по трем главным осям, с прикрепленными к ней ВМГ в одной плоскости, в которой находится и центр масс аппарата. При этом ВМГ расположены на восьми лучах (4 из них одной длины l_1, а другие 4 могут быть другой длины l_2) и жестко закреплены относительно рамы. Таким образом, можно говорить о том, что радиус-векторы центров ВМГ и орты силы тяги каждой ВМГ суть геометрические константы в системе координат, связанной с коптером. Другими словами, величина $F_M(t)$ [Н],

вычисленная в модели ВМГ, является модулем вектора силы, приложенного всегда в определенном направлении и в определенной точке рамы коптера. В процессе полета это направление будет меняться, конечно, относительно Земли, но относительно рамы коптера (и связанной с ней системы координат) оно остается неизменным.

3. На протяжении всей методики и модели используется две системы координат: а) неподвижная инерциальная, связанная с Землей, и б) подвижная, связанная с коптером. Системы координат обозначим буквами **I** и **B** (от английских слов *inertial* – инерциальный и *body* – тело). При этом оси систем направлены: x_I – вправо, y_i – на наблюдателя, z_i – вниз, x_B – вправо вдоль луча первой ВМГ, y_B – на наблюдателя вдоль луча третьей ВМГ, z_B – сверху вниз, при нормальной горизонтальной ориентации октокоптера (см. рис. 2.1.1):

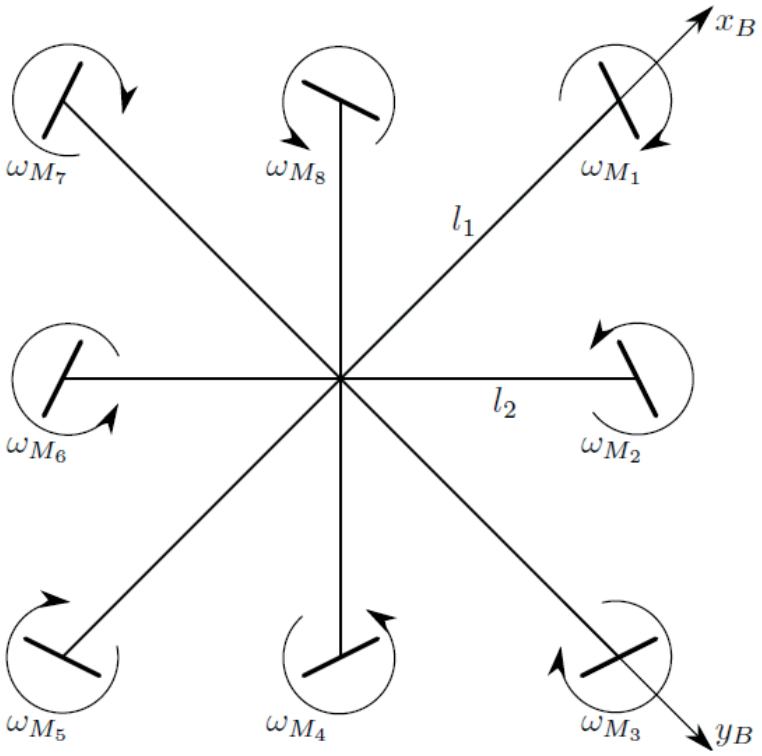


Рис. 2.1.1

Тогда в системе координат **B** (связанной с коптером) векторы центров ВМГ можно записать следующим образом (буква T обозначает транспонирование, т. е. векторы координат записываются векторами-столбцами):

$$\begin{aligned}
&\vec{r}_{M1} = \left(l_1,0,0\right)^T, \vec{r}_{M2} = \frac{1}{\sqrt{2}}\left(l_2,l_2,0\right)^T,\\
&\vec{r}_{M3} = \left(0,l_1,0\right)^T, \vec{r}_{M4} = \frac{1}{\sqrt{2}}\left(-l_2,l_2,0\right)^T,\\
&\vec{r}_{M5} = \left(-l_1,0,0\right)^T, \vec{r}_{M6} = \frac{1}{\sqrt{2}}\left(-l_2,-l_2,0\right)^T,\\
&\vec{r}_{M7} = \left(0,-l_1,0\right)^T, \vec{r}_{M8} = \frac{1}{\sqrt{2}}\left(l_2,-l_2,0\right)^T,
\end{aligned} \qquad 2.1.2$$

где l_1 – длина луча у 1, 3, 5 и 7-й ВМГ, а l_2 – длина луча у 2, 4, 6 и 8-й ВМГ.

Для того чтобы октокоптер управлялся (хотя бы немного) по курсу, в нашем случае отсутствия реактивного момента двигателей векторы сил тяги каждой ВМГ должны быть немного отклонены от вертикального направления (например, повернуты вокруг каждого луча на небольшой угол порядка 1…5°, причем в разные стороны – четные в одну, а нечетные в другую). Если обозначить этот угол как γ орты можно получить в следующем виде:

$$\begin{aligned}
&\vec{e}_{M1} = \left(0,-\sin(\gamma),-\cos(\gamma)\right)^T, \vec{e}_{M2} = \left(-\frac{\sin(\gamma)}{\sqrt{2}},\frac{\sin(\gamma)}{\sqrt{2}},-\cos(\gamma)\right)^T,\\
&\vec{e}_{M3} = \left(\sin(\gamma),0,-\cos(\gamma)\right)^T, \vec{e}_{M4} = \left(-\frac{\sin(\gamma)}{\sqrt{2}},-\frac{\sin(\gamma)}{\sqrt{2}},-\cos(\gamma)\right)^T,\\
&\vec{e}_{M5} = \left(0,\sin(\gamma),-\cos(\gamma)\right)^T, \vec{e}_{M6} = \left(\frac{\sin(\gamma)}{\sqrt{2}},-\frac{\sin(\gamma)}{\sqrt{2}},-\cos(\gamma)\right)^T,\\
&\vec{e}_{M7} = \left(-\sin(\gamma),0,-\cos(\gamma)\right)^T, \vec{e}_{M8} = \left(\frac{\sin(\gamma)}{\sqrt{2}},\frac{\sin(\gamma)}{\sqrt{2}},-\cos(\gamma)\right)^T.
\end{aligned} \qquad (2.1.3)$$

На основе этих геометрических констант (напомним, они являются константами только в системе координат **B**, связанной с коптером) строится весь дальнейший каркас модели, поэтому они важны. В общем случае винтомоторных групп может быть другое количество, направлены они могут быть в других направлениях и располагаться у коптера в других местах. О применяемых системах координат подробно изложено в [1].

4. Рассмотрим силы, которые будут учтены в модели. На коптер действуют: 4.1) сила тяжести. Направлена всегда вниз вдоль оси z_I инерциальной системы координат **I**. Сила тяжести – это постоянная по модулю величина, зависит только от массы коптера. Масса коптера принимается постоянной и не меняется (хотя в процессе моделирования ее можно будет менять, имитируя, например, дополнительный полезный груз, навешенный на коптер). По направлению сила тяжести – переменна, если рассматривать ее в связанной системе координат **B**;

4.2) силы тяги ВМГ. Их всего 8, направлены вдоль своих направлений, модуль каждой силы вычисляется в зависимости от угловой скорости вращения соответствующей ВМГ:

$$F_{Mi}(t) = C_T \cdot \omega_i^2(t), i = 1\ldots8. \quad (2.1.4)$$

Направления сил тяги определены и неизменны в относительной системе координат **B**;

4.3) сила сопротивления воздуха (возможно, с учетом ветра) – моделируется как состоящая из двух компонент. Сила сопротивления воздуха прямо пропорциональна плотности воздуха, квадрату линейной скорости объекта относительно воздуха и характерной площади сечения в выбранном направлении (коэффициент формы). Сила ветра – внешняя возмущающая сила, задается произвольным образом или при помощи дополнительной «модели ветра» (в настоящей методике подробно не рассматривается);

4.4) внешняя сила или возмущение – произвольное внешнее воздействие. В модели такая возможность заложена как простой способ в дальнейшем проверять на устойчивость регуляторы по каждому из направлений.

5. Рассмотрим моменты сил, учитываемые в модели:

5.1) реактивный момент двигателей ВМГ. В нашей модели он равен нулю из-за парности двигателей и винтов в каждой ВМГ, в общем случае его следует учитывать. В некоторых аппаратах этот момент используется для управления по курсу;

5.2) явление прецессии. В рассматриваемой модели он нулевой, так как винтомоторные группы, вращающиеся в разных направлениях на одной оси, будут давать противоположные моменты прецессии, равные по абсолютной величине. В общем случае его тоже нужно считать;

5.3) момент сопротивления воздуха – аналогично силе сопротивления воздуха прямо пропорционален плотности воздуха, квадрату угловой скорости коптера и коэффициенту формы;

5.4) опрокидывающий момент от ветровой нагрузки (подробно не рассматривается);

5.5) внешний возмущающий момент произвольного направления и величины – используется для отладки регуляторов;

5.6) моменты от сил тяги ВМГ. Поскольку винты расположены не в центре масс коптера, каждый из них будет создавать свой поворотный момент. Это, пожалуй, основной фактор, который используется для управления ориентацией коптера в пространстве.

Как принято в теории управления, архитектуру модели объекта можно условно разделить на две части: а) модель динамики объекта, т. е. самого октокоптера, – модель его движения в пространстве под действием сил и моментов безотносительно управления; б) модель системы управления – определенная логика и алгоритмы управления, регуляторы, поддерживающие в заданном значении измеряемые величины.

В создаваемой модели октокоптера можно выделить как эти две составные части – основу модели, так и еще некоторые, а именно: в) базу сигналов, в ко-

торой будут храниться постоянные и переменные именованные величины, которые могут быть использованы (считано текущее значение) в любой части модели и определены в каком-то одном (и только одном) месте модели; г) макет пульта управления со своими сущностями – кнопками, отображающими элементами и т. п.; д) техническую анимацию, отображающую в динамике основные элементы и текущие параметры модели октокоптера.

Итого в дальнейшем будем последовательно рассматривать и создавать пять элементов: модель объекта, модель системы управления, базу сигналов, макет пульта, техническую анимацию. База сигналов будет пополняться сигналами по мере необходимости, поэтому ее создание разнесено по всей второй части методики. Макет пульта и анимация сделаны в рамках одной главы, так как этот вид работы больше связан с анимацией и графическими примитивами, чем с набором расчетных схем.

Нелинейные уравнения динамики октокоптера

С учетом принятых допущений и того факта, что коптер моделируется как единое твердое тело, основа модели динамики коптера очень проста – это второй закон Ньютона, который в векторной форме выглядит следующим образом, всего два простых уравнения (3.1.1) и (3.1.2):

$$\frac{d\vec{\boldsymbol{p}}(t)}{dt} = \vec{\boldsymbol{F}}(t), \tag{3.1.1}$$

$$\frac{d\vec{\boldsymbol{L}}(t)}{dt} = \vec{\boldsymbol{M}}(t), \tag{3.1.2}$$

где $\vec{\boldsymbol{p}}(t) = m \cdot \vec{\boldsymbol{v}}(t)$ – импульс коптера, $\vec{\boldsymbol{L}}(t) = I(t) \cdot \vec{\boldsymbol{\omega}}(t)$ – момент импульса коптера, m – его масса, $I(t)$ – тензор инерции (линейный оператор момента инерции). $\vec{\boldsymbol{F}}(t)$ и $\vec{\boldsymbol{M}}(t)$ суть суммы всех сил и всех моментов, действующих на коптер.

Легко видеть, что все было бы очень просто и очевидно, если бы правые части были небольшими, а тензор инерции не зависел бы от ориентации коптера (как если бы коптер был шаром). Но из-за вращения коптера, а также из-за обилия сил и моментов конечные уравнения, записанные в инерциальной системе координат, получатся очень громоздкими даже без учета прецессии и реактивных моментов ВМГ. Поэтому используется следующий математический прием – уравнения, записанные в инерциальной системе координат, переводятся в систему координат **B**, связанную с коптером, в которой слагаемые правой части (их запись) получается гораздо более лаконичной и удобной. Подробнее этот прием описан в [1], приведем здесь лишь основные соотношения.

Подставив в приведенные уравнения второго закона Ньютона (3.1.1) и (3.1.2) значения для импульса и момента импульса коптера, для инерциальной системы координат получим более подробную их запись (3.1.3) и (3.1.4):

$$\frac{d\vec{\boldsymbol{v}}_I(t)}{dt} = \frac{1}{m}\vec{\boldsymbol{F}_I}(t), \tag{3.1.3}$$

$$\frac{d\overrightarrow{\omega_I}(t)}{dt} = I_I^{-1}\left(\overrightarrow{M_I}(t) - \frac{dI_I}{dt}\cdot\overrightarrow{\omega_I}(t)\right). \quad (3.1.4)$$

В этих уравнениях математически очень сложны правые части, а аналогичные уравнения динамики в связанной системе координат **B** (где правые части более просты) не могут быть получены так легко, поскольку система координат, связанная с коптером, вращается.

В математике есть два основных подхода к преобразованию векторов из одной системы координат в другую и обратно – матрицы поворота и кватернионы. Последние более универсальны, первые – проще. В настоящей методике и модели используются матрицы поворота. Если ориентацию коптера представить тремя углами: крена φ (roll), тангажа θ (pitch) и рыскания/курса ψ (yaw), а матрицы преобразования из системы **I** в **B** и обратно обозначить как $\boldsymbol{R_{IB}}$ и $\boldsymbol{R_{BI}} = \boldsymbol{R_{IB}^T}$, то любой вектор, записанный для системы координат **I**, можно перевести в систему координат **B**, и наоборот, используя умножение соответствующей матрицы на вектор, например для вектора суммы сил: $\overrightarrow{\boldsymbol{F_I}}(t) = \boldsymbol{R_{BI}}\overrightarrow{\boldsymbol{F_B}}(t)$, или $\overrightarrow{\boldsymbol{F_B}}(t) = \boldsymbol{R_{IB}}\overrightarrow{\boldsymbol{F_I}}(t)$.

Чтобы лучше понимать написанное далее в методике, прокомментируем еще раз, как понимать матрицы поворота и вектора в пространстве: система координат **I** неподвижна, относительно нее летает и вращается коптер, а вместе с ним и связанная система координат **B**. Просуммировав все силы, которые действуют на коптер, можно получить вектор $\vec{\boldsymbol{F}}(t)$ (аналогично и с моментом сил), и в каждый момент времени он является вектором с вполне определенной длиной и направлением в пространстве. Но раскладывая его на проекции по осям – в разных системах координат мы получим разные величины проекций. Все, что делает матрица поворота, – переводит одни проекции вектора в другие, при этом сам вектор никуда не поворачивается и не изменяет своей длины в выбранный момент времени. Если проекции сравнить с тенями вектора, то матрицы поворота преобразуют одни тени в другие, все. Больше они ничего не делают и сложностей, кроме вычислительных, не представляют.

Нам они нужны только из-за того, что вычислять и суммировать силы и моменты сил, действующие на коптер, гораздо проще в связанной системе координат **B**. В ней же проще провести численное интегрирование уравнений, чтобы получить величины угловой и линейной скоростей коптера $\overrightarrow{\boldsymbol{\omega_B}}(t)$ и $\overrightarrow{\boldsymbol{v_B}}(t)$ (в связанной системе координат), а потом обратной матрицей поворота вычислить (алгебраически – матрицы поворота это простые уравнения) эти же скорости для инерциальной системы координат **I** и там уже, интегрируя дальше, вычислить линейные и угловые координаты коптера в инерциальном пространстве.

Приведем используемое выражение для матрицы поворота из системы **I** в **B**, записав для краткости функции косинуса и синуса как *cos() = c()* и *sin() = s()*:

$$\boldsymbol{R}_{IB} = \begin{pmatrix} c(\theta)\cdot c(\psi) & c(\theta)\cdot s(\psi) & -s(\theta) \\ s(\varphi)\cdot s(\theta)\cdot c(\psi) - c(\varphi)\cdot s(\psi) & s(\varphi)\cdot s(\theta)\cdot s(\psi) + c(\varphi)\cdot c(\psi) & s(\varphi)\cdot c(\theta) \\ c(\varphi)\cdot s(\theta)\cdot c(\psi) + s(\varphi)\cdot s(\psi) & s(\varphi)\cdot s(\theta)\cdot s(\psi) - s(\varphi)\cdot c(\psi) & c(\varphi)\cdot c(\theta) \end{pmatrix} \quad (3.1.5)$$

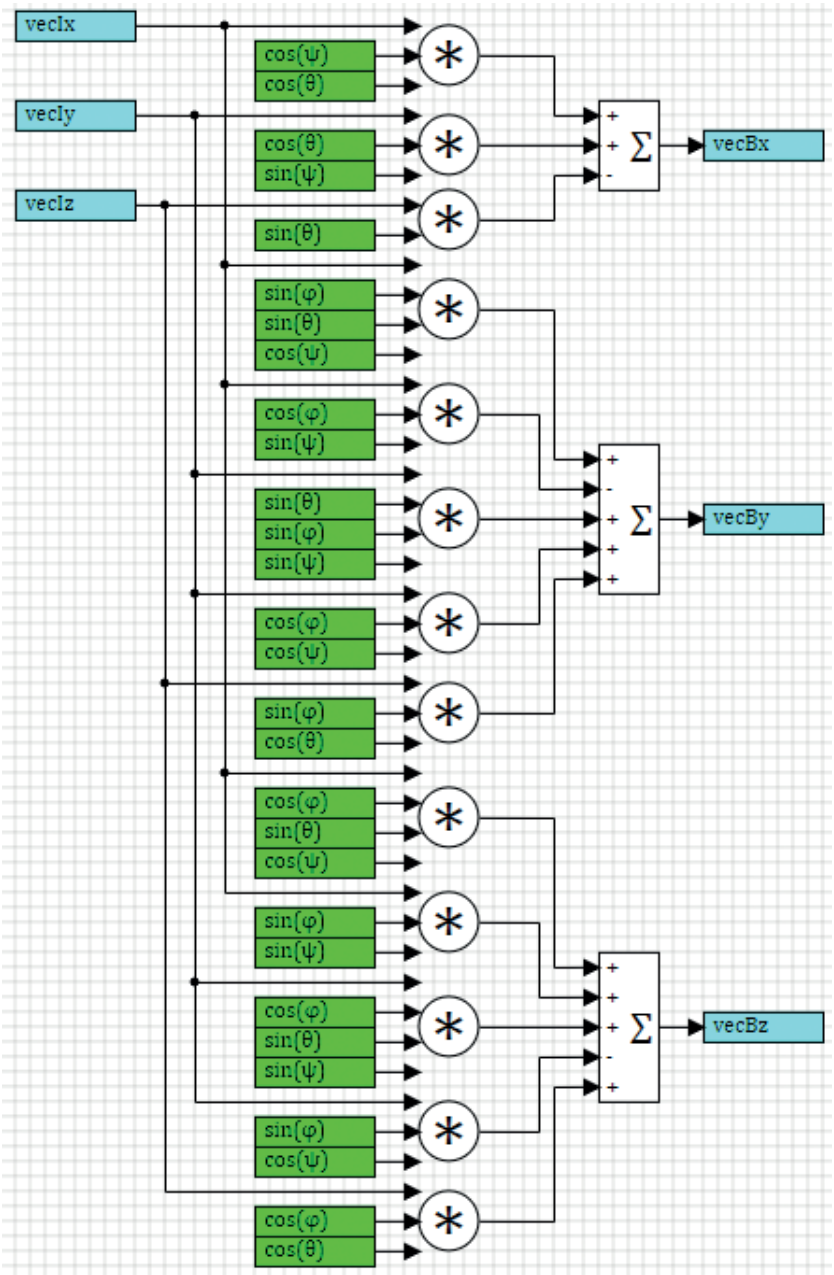


Рис. 3.1.1. Матрица поворота из системы **I** в систему координат **B**

Отметим еще раз, что в каждый момент времени эта матрица, очевидно, будет получаться разной, так как меняются углы ориентации коптера. Но это просто алгебраические вычисления, без дифференциальных уравнений. Сама матрица не константа, у нее есть производная по времени.

Реализованная методом структурного моделирования, матрица в среде SimInTech выглядит, как показано на рис. 3.1.1.

Тогда для вектора линейной скорости можно записать: $\overrightarrow{\boldsymbol{v}_B}(t) = \boldsymbol{R}_{IB}\overrightarrow{\boldsymbol{v}_I}(t)$, а адля момента импульса $\overrightarrow{\boldsymbol{L}_B}(t) = \boldsymbol{R}_{IB}\overrightarrow{\boldsymbol{L}_I}(t)$. Опуская промежуточные выкладки (в том числе производную матрицы поворота, которая получается равной в итоге векторному произведению угловой скорости объекта на саму матрицу поворота, взятому с обратным знаком), для первого из рассматриваемых уравнений динамики (3.1.3) получим следующее выражение (3.1.6) в связанной с коптером системе координат **B**:

$$\frac{d\vec{v}_B(t)}{dt} = -\overrightarrow{\boldsymbol{\omega}_B}(t)\times\overrightarrow{\boldsymbol{v}_B}(t) + \frac{1}{m}\overrightarrow{\boldsymbol{F}_B}(t), \tag{3.1.6}$$

а для второго уравнения (3.1.4), учитывая, что $\overrightarrow{\boldsymbol{L}_B}(t) = I_B \cdot \overrightarrow{\boldsymbol{\omega}_B}(t)$, и так как во вращающейся связанной системе координат тензор инерции константа и его производная по времени равна нулю, а $\frac{d\vec{\boldsymbol{L}}_B(t)}{dt} = I_B \cdot \frac{d\overrightarrow{\boldsymbol{\omega}_B}(t)}{dt}$, получим:

$$\frac{d\overrightarrow{\boldsymbol{\omega}_B}(t)}{dt} = I_B^{-1}\left(\overrightarrow{\boldsymbol{M}_B}(t) - \overrightarrow{\boldsymbol{\omega}_B}(t)\times(I_B\cdot\overrightarrow{\boldsymbol{\omega}_B}(t))\right). \tag{3.1.7}$$

Итого запись II закона Ньютона для вращающейся системы **B**, по сравнению с исходными уравнениями (3.1.3 и 3.1.4), дополнилась двумя векторными произведениями.

Переменными состояния коптера в такой записи являются две векторных величины (или 6 скалярных), а именно: вектор линейной скорости и вектор угловой скорости. Алгебраически это будет 6 переменных – три проекции линейной скорости и три проекции угловой скорости. И мы имеем записанную в форме Коши систему из 6 нелинейных уравнений первого порядка, которую легко можно реализовать в среде SimInTech и успешно решить ее тем или иным методом интегрирования.

Получив значения скоростей коптера (сначала в системе **B**), их можно матрицей обратного поворота преобразовать к системе **I**, еще раз проинтегрировать и получить уже значения координат и, следовательно, положение объекта в пространстве, в инерциальной системе координат **I**.

Единственное, что мы еще не сделали по уравнениям, – не записали выражения для сил и моментов, действующих на коптер. Сделаем это ниже, в системе координат **B**. Согласно допущениям предыдущего раздела, учитываем и обозначим индексами: $\boldsymbol{M}$ – работу двигателей, только в части создаваемой

силы тяги и моментов от нее, $\boldsymbol{D}$ – силу сопротивления воздуха (вместе с ветром), $\boldsymbol{O}$ – внешнее возмущение, вообще говоря, нулевое и задаваемое произвольно оператором модели, силу тяжести (последняя поворотного момента сил не создает):

$$\overrightarrow{\boldsymbol{F_B}}(t) = \overrightarrow{\boldsymbol{F_M}}(t) + \overrightarrow{\boldsymbol{F_D}}(t) + \overrightarrow{\boldsymbol{F_O}}(t) + mg\boldsymbol{R_{IB}}\overrightarrow{\boldsymbol{e_{Iz}}}, \tag{3.1.8}$$

$$\overrightarrow{\boldsymbol{M_B}}(t) = \overrightarrow{\boldsymbol{M_M}}(t) + \overrightarrow{\boldsymbol{M_D}}(t) + \overrightarrow{\boldsymbol{M_O}}(t). \tag{3.1.9}$$

Сила тяжести в связанной системе координат будет «поворачиваться» в зависимости от ориентации коптера.

Распишем подробнее, чему равны слагаемые.

Сила тяги:

$$\begin{aligned}\overrightarrow{\boldsymbol{F_M}}(t) = {} & C_T \cdot \omega_{M1}^2(t) \cdot \vec{\boldsymbol{e}}_{M1} + C_T \cdot \omega_{M2}^2(t) \cdot \vec{\boldsymbol{e}}_{M2} + C_T \cdot \omega_{M3}^2(t) \cdot \vec{\boldsymbol{e}}_{M3} + \\ & C_T \cdot \omega_{M4}^2(t) \cdot \vec{\boldsymbol{e}}_{M4} + C_T \cdot \omega_{M5}^2(t) \cdot \vec{\boldsymbol{e}}_{M5} + C_T \cdot \omega_{M6}^2(t) \cdot \vec{\boldsymbol{e}}_{M6} + \\ & C_T \cdot \omega_{M7}^2(t) \cdot \vec{\boldsymbol{e}}_{M7} + C_T \cdot \omega_{M8}^2(t) \cdot \vec{\boldsymbol{e}}_{M8}.\end{aligned} \tag{3.1.10}$$

Здесь уже фигурируют угловые скорости вращения ВМГ, а не коптера. Напомним, что орты сил тяги у нас записаны для системы координат **B** и там являются константами – очевидно, что их можно вычислить 1 раз, общий коэффициент вынести здесь за скобку и сильно упростить вычисления. Если бы мы записывали уравнения динамики в системе **I**, то данное выражение обросло бы еще 8 умножениями на матрицу поворота каждого орта, и это пришлось бы считать на каждом шаге расчета.

Сила сопротивления воздуха (при отсутствии ветра):

$$\overrightarrow{\boldsymbol{F_D}}(t) = -0.5\rho \cdot C_D \cdot \begin{pmatrix} A_{yz} \cdot v_x \cdot |v_x| \\ A_{xz} \cdot v_y \cdot |v_y| \\ A_{xy} \cdot v_z \cdot |v_z| \end{pmatrix}. \tag{3.1.11}$$

Внешнее возмущение: нулевое. По желанию пользователя модели он может сам установить то или иное значение позже, до расчета, или в процессе моделирования.

Момент сил тяги двигателей запишем как:

$$\begin{aligned}\overrightarrow{\boldsymbol{M_M}}(t) = {} & \overrightarrow{r_{M1}} \times \overrightarrow{e_{M1}} \cdot C_T \cdot \omega_{M1}^2(t) + \overrightarrow{r_{M2}} \times \overrightarrow{e_{M2}} \cdot C_T \cdot \omega_{M2}^2(t) + \\ & \overrightarrow{r_{M3}} \times \overrightarrow{e_{M3}} \cdot C_T \cdot \omega_{M3}^2(t) + \overrightarrow{r_{M4}} \times \overrightarrow{e_{M4}} \cdot C_T \cdot \omega_{M4}^2(t) + \\ & \overrightarrow{r_{M5}} \times \overrightarrow{e_{M5}} \cdot C_T \cdot \omega_{M5}^2(t) + \overrightarrow{r_{M6}} \times \overrightarrow{e_{M6}} \cdot C_T \cdot \omega_{M6}^2(t) + \\ & \overrightarrow{r_{M7}} \times \overrightarrow{e_{M7}} \cdot C_T \cdot \omega_{M7}^2(t) + \overrightarrow{r_{M8}} \times \overrightarrow{e_{M8}} \cdot C_T \cdot \omega_{M8}^2(t).\end{aligned} \tag{3.1.12}$$

Момент сопротивления воздуха:

$$\overrightarrow{\boldsymbol{M}_{\boldsymbol{D}}}(t) = -0.5\rho \cdot C_D \cdot \begin{pmatrix} A_{xy} \cdot \omega_x \cdot |\omega_x| \cdot l_x \\ A_{xy} \cdot \omega_y \cdot |\omega_y| \cdot l_y \\ 8A_{yz} \cdot \omega_z \cdot |\omega_z| \cdot l_z \end{pmatrix}. \quad (3.1.13)$$

Еще раз отметим, что расчет прецессии и реактивных моментов ВМГ в данной методике для краткости изложения опущен.

Чтобы не ошибиться при переходе от векторных уравнений к скалярным, записанным по осям, проще воспользоваться пакетом типа MathCAD или Maple, в котором большинство преобразований можно выполнить автоматизированно, в символьном виде и получить требуемые 6 уравнений динамики, записанные по осям подвижной системы координат **B**.

В наиболее компактной форме полученные и решаемые уравнения динамики выглядят так:

$$\frac{d\vec{\boldsymbol{v}}_B(t)}{dt} = \frac{1}{m}\left(\overrightarrow{\boldsymbol{F}_{\boldsymbol{M}}}(t) + \overrightarrow{\boldsymbol{F}_{\boldsymbol{D}}}(t) + \overrightarrow{\boldsymbol{F}_{\boldsymbol{O}}}(t)\right) + g\boldsymbol{R}_{\boldsymbol{IB}}\overrightarrow{\boldsymbol{e}_{\boldsymbol{Iz}}} - \overrightarrow{\boldsymbol{\omega}_{\boldsymbol{B}}}(t) \times \overrightarrow{\boldsymbol{v}_{\boldsymbol{B}}}(t), \quad (3.1.14)$$

$$\frac{d\overrightarrow{\boldsymbol{\omega}_{\boldsymbol{B}}}(t)}{dt} = I_B^{-1}\left(\overrightarrow{\boldsymbol{M}_{\boldsymbol{M}}}(t) + \overrightarrow{\boldsymbol{M}_{\boldsymbol{D}}}(t) - \overrightarrow{\boldsymbol{\omega}_{\boldsymbol{B}}}(t) \times (I_B \cdot \overrightarrow{\boldsymbol{\omega}_{\boldsymbol{B}}}(t))\right). \quad (3.1.15)$$

Интегрируя их и получив значения скоростей в системе **B**, можно посчитать скорость и углы ориентации в инерциальной системе координат **I**:

$$\overrightarrow{\boldsymbol{v}_{\boldsymbol{I}}}(t) = \boldsymbol{R}_{\boldsymbol{BI}}\overrightarrow{\boldsymbol{v}_{\boldsymbol{B}}}(t), \quad (3.1.16)$$

$$\begin{pmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \boldsymbol{W}_{\boldsymbol{BI}}\overrightarrow{\boldsymbol{\omega}_{\boldsymbol{B}}}(t). \quad (3.1.17)$$

$\boldsymbol{W}_{\boldsymbol{BI}}$ – матрица преобразования из угловой скорости коптера в связанной системе координат, в скорости поворота по углам Эйлера (подробнее см. в [1]).

В следующем разделе показан один из вариантов реализации этой системы уравнений в среде SimInTech как готовое решение. Пошаговое выполнение будет приведено позже, в последних разделах методики.

Архитектура решения уравнений динамики в SimInTech

Переходим к самому интересному – к реализации полученных уравнений динамики средствами среды динамического моделирования SimInTech. Как известно, любую задачу можно всегда решить несколькими способами, и чем она сложнее и многомернее, тем больше количество способов. В настоящей методике предложен один из вариантов решения, он не единственный, и в некоторых местах не самый оптимальный. Выбор решения определялся простотой реализации, с одной стороны, и удобством для методичного рассказа – с другой.

Итак, что мы получили с точки зрения математики – в системе координат **B** мы имеем 2 векторных дифференциальных уравнения (3.1.14) и (3.1.15), которые при переходе к проекциям (и скалярным уравнениям) дают 6 нелинейных дифференциальных уравнений первого порядка относительно 6 переменных: трех скоростей и трех угловых скоростей. Это так называемая 6DOF задача, т. е. задача с шестью степенями свободы. Сначала можно подумать, что раз у коптера имеется 6 степеней свободы, то должно быть и 6 переменных состояния (дифференциальных переменных). Но это так только на первый взгляд. Кроме скоростей, нам придется получить еще и координаты (три линейных и три угла положения в пространстве) – для чего еще раз проинтегрировать скорости. Таким образом, всего у коптера есть 12 степеней свободы. А если учесть еще то, что правые части дифференциальных уравнений есть не что иное, как ускорения коптера по осям, то получим как бы 18 степеней свободы. Это важно понимать, так как в дальнейшем для построения регулятора нам потребуются все 12 фазовых координат объекта плюс еще 6 измеренных (в нашем случае – вычисленных) ускорений коптера.

4.1. Решение дифференциальных уравнений методом структурного моделирования

Напомним еще раз, что индексами M, D, O в уравнениях (3.1.14) и (3.1.15) обозначены силы и моменты: M – от двигателей и винтов ВМГ, D – от сопротивления воздуха, O – внешние возмущающие силы и моменты. Для реализации уравнений в SimInTech необходимо указанные векторные уравнения спроецировать на оси x_B, y_B, z_B связанной системы координат **B**. Легко видеть, что при этом получится следующая система дифференциальных уравнений (4.1.1) для скалярных величин линейных и угловых скоростей по осям.

Как правило, для решения дифференциального уравнения методом структурного моделирования используются либо типовые звенья первого и второго порядков – если дифференциальное уравнение линейно и подходит под один из типов, – либо используется блок типа «Интегратор», на вход которому подается правая часть уравнения, а на выходе будет искомая интегральная величина. Приведем пример – заготовку для решения нашей системы из 6 дифференциальных уравнений, см. рис. 4.1.1.

$$\begin{cases} \dfrac{dv_{Bx}(t)}{dt} = \dfrac{1}{m}\left(F_{Mx}(t) + F_{Dx}(t) + F_{Ox}(t)\right) - g \cdot sin(\theta) + \\ v_{By}(t) \cdot \omega_{Bz}(t) - v_{Bz}(t) \cdot \omega_{By}(t), \\ \dfrac{dv_{By}(t)}{dt} = \dfrac{1}{m}\left(F_{My}(t) + F_{Dy}(t) + F_{Oy}(t)\right) + g \cdot cos(\theta) sin(\varphi) \\ -v_{Bx}(t) \cdot \omega_{Bz}(t) + v_{Bz}(t) \cdot \omega_{Bx}(t), \\ \dfrac{dv_{Bz}(t)}{dt} = \dfrac{1}{m}\left(F_{Mz}(t) + F_{Dz}(t) + F_{Oz}(t)\right) + g \cdot cos(\theta) cos(\varphi) \\ +v_{Bx}(t) \cdot \omega_{By}(t) - v_{By}(t) \cdot \omega_{Bx}(t), \\ \dfrac{d\omega_{Bx}(t)}{dt} = \dfrac{1}{I_{xx}}\left(M_{Mx}(t) + M_{Dx}(t) + M_{Ox}(t) + \left(I_{yy} - I_{zz}\right) \cdot \omega_{By}(t) \cdot \omega_{Bz}(t)\right), \\ \dfrac{d\omega_{By}(t)}{dt} = \dfrac{1}{I_{yy}}\left(M_{My}(t) + M_{Dy}(t) + M_{Ox}(t) + \left(I_{zz} - I_{xx}\right) \cdot \omega_{Bx}(t) \cdot \omega_{Bz}(t)\right), \\ \dfrac{d\omega_{Bz}(t)}{dt} = \dfrac{1}{I_{zz}}\left(M_{Mz}(t) + M_{Dz}(t) + M_{Ox}(t) + \left(I_{xx} - I_{yy}\right) \cdot \omega_{Bx}(t) \cdot \omega_{By}(t)\right). \end{cases} \quad (4.1.1)$$

На рис. 4.1.1 представлена «основа» динамической части модели октокоптера, которую формируют 6+3+3 блоков типа «Интегратор». Первые шесть блоков, получая на вход правые части дифференциальных уравнений (4.1.1) – ускорения коптера по осям $a_{Bx}, a_{By}, a_{Bz}, \omega_{Bx}, \omega_{By}, \omega_{Bz}$, – занимаются интегрированием и вычислением скоростей коптера по этим же осям (тоже в связанной с коптером системе B).

Следующие три интегратора принимают линейные скорости в системе координат **I** (полученные алгебраически из скоростей в системе **B** путем применения матрицы поворота) и, интегрируя их, вычисляют координаты центра масс коптера в инерциальной системе координат **I**.

И еще три блока типа «Интегратор» занимаются вычислением углов ориентации коптера, интегрируя их производные (угловые скорости) в системе **I**, полученные из угловых скоростей коптера в системе **B** применением матрицы $\boldsymbol{W_{BI}}$. Здесь же будет реализовано и вычисление нужных тригонометрических функций от углов поворота.

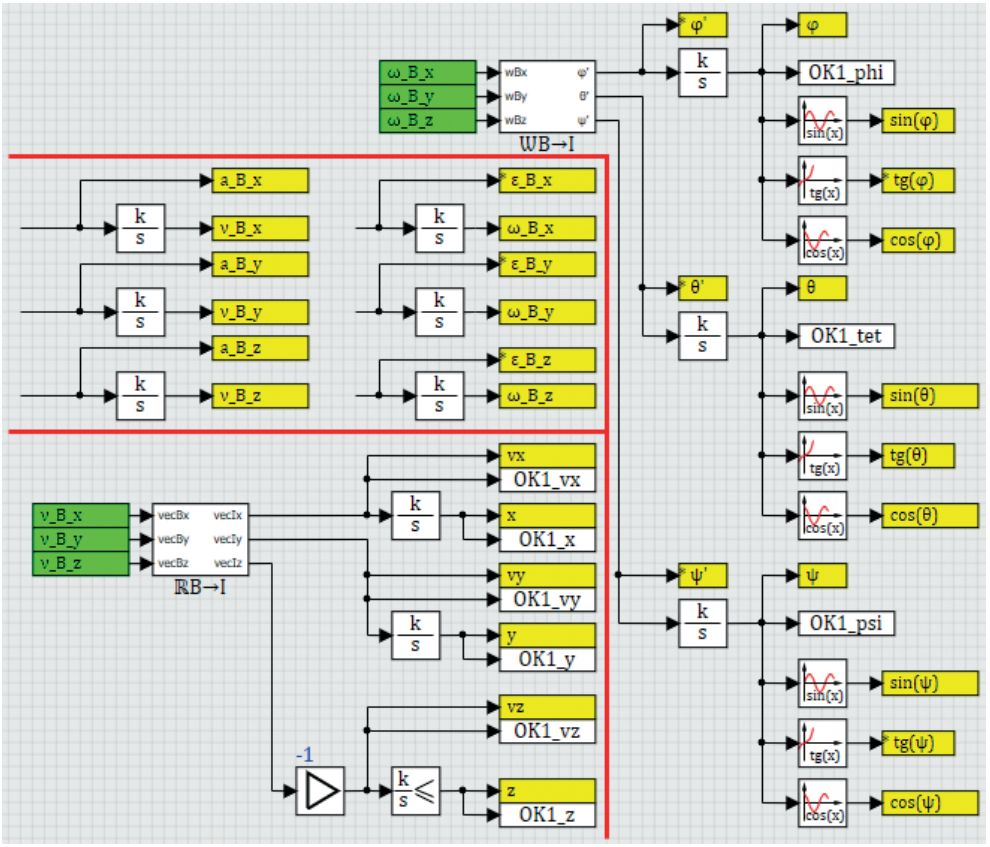


Рис. 4.1.1. Архитектура модели октокоптера

Обилие блоков «В память» и «Из памяти» обусловлено многократным использованием в других частях модели полученных величин. Например, по рис. 3.1.1 видно, что тригонометрические функции от углов ориентации используются многократно, и оптимально вычислить их в одном месте схемы (как показано на рис. 4.1.1), а потом уже использовать по мере необходимости.

Если бы у нас была более простая ситуация – одно дифференциальное уравнение второго порядка (классический второй закон Ньютона a = F/m), например в проекции на ось x, то его решение таким же способом выглядело бы как на рис. 4.1.2:

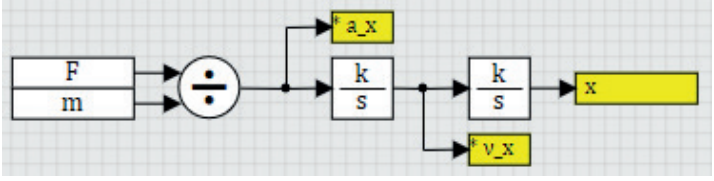


Рис. 4.1.2. Прием «двойного» интегрирования для решения уравнения второго порядка

В случае коптера используется такой же прием, т. е. если на выходе из интегратора имеется какая-то величина, то слева от интегратора – обязательно производная этой же величины. Таким образом записываются и решаются дифференциальные уравнения методом структурного моделирования, если не прибегать к встроенному языку программирования SimInTech (где зачастую удается сделать то же самое еще более просто, но менее очевидно).

В субмодели W(B-I) реализована матрица преобразования из угловой скорости в системе **B** в производные углов Эйлера, см. рис. 4.1.3 и [1] для подробностей.

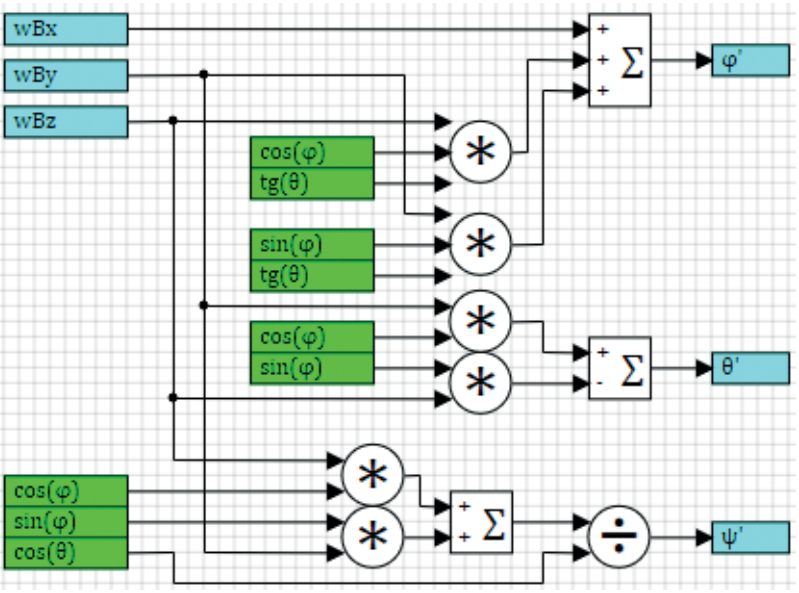


Рис. 4.1.3. Матрица $\boldsymbol{W_{BI}}$

4.2. Правые части уравнений

Для замыкания модели необходимо посчитать и реализовать все правые части у шести основных уравнений системы (4.1.1), т. е. векторные уравнения, спроецированные на оси x_B, y_B, z_B, надо развернуть и «нарисовать» в схеме SimInTech. Опуская выкладки (их читатель может выполнить самостоятельно на листке бумаги или при помощи какого-то символьного математического ПО), приведем окончательный вид уравнений. Из-за громоздкости приводить будем по слагаемым в правых частях.

Сила тяги винтомоторных групп $\overrightarrow{\boldsymbol{F_M}}(t)$:

$$F_{Mx}(t) = \frac{1}{\sqrt{2}}\left(-F_{M2}(t) - F_{M4}(t) + F_{M6}(t) + F_{M8}(t)\right)\cdot\sin(\gamma) + \left(F_{M3}(t) - F_{M7}(t)\right)\cdot\sin(\gamma), \tag{4.2.1}$$

$$F_{My}(t)=\frac{1}{\sqrt{2}}\left(F_{M2}(t)-F_{M4}(t)-F_{M6}(t)+F_{M8}(t)\right)\cdot\sin(\gamma)+\left(F_{M5}(t)-F_{M1}(t)\right)\cdot\sin(\gamma), \tag{4.2.2}$$

$$F_{Mz}(t)=-\begin{pmatrix} F_{M1}(t)+F_{M2}(t)+F_{M3}(t)+F_{M4}(t) \\ +F_{M5}(t)+F_{M6}(t)+F_{M7}(t)+F_{M8}(t)\end{pmatrix}\cdot cos(\gamma); \tag{4.2.3}$$

где $F_{Mi}(t)$ – сила тяги i-ой ВМГ в текущий момент времени.

Сила сопротивления воздуха $\overrightarrow{\boldsymbol{F_D}}(t)$ при отсутствии ветра – формулы приведены выше, проекции будут равны:

$$F_{Dx}(t)=-0.5\rho\cdot C_D\cdot A_{yz}\cdot v_x\cdot|v_x|, \tag{4.2.4}$$

$$F_{Dy}(t)=-0.5\rho\cdot C_D\cdot A_{xz}\cdot v_y\cdot|v_y|, \tag{4.2.5}$$

$$F_{Dz}(t)=-0.5\rho\cdot C_D\cdot A_{xy}\cdot v_z\cdot|v_z|. \tag{4.2.6}$$

Сила тяжести в проекции на оси подвижной системы координат, слагаемое $g\boldsymbol{R}_{\boldsymbol{IB}}\overrightarrow{\boldsymbol{e}_{\boldsymbol{Iz}}}$, очевидно, будет равно:

$$g\boldsymbol{R}_{\boldsymbol{IB}}\overrightarrow{\boldsymbol{e}_{\boldsymbol{Iz}}}=g\cdot\begin{pmatrix}-\sin(\theta)\\ \sin(\varphi)\cdot\cos(\theta)\\ \cos(\varphi)\cdot\cos(\theta)\end{pmatrix}. \tag{4.2.7}$$

И последнее слагаемое в уравнении линейной скорости коптера – векторное произведение скоростей:

$$-\overrightarrow{\boldsymbol{\omega_B}}(t)\times\overrightarrow{\boldsymbol{v_B}}(t)=\begin{pmatrix} v_{Bz}\cdot\omega_{By}-v_{By}\cdot\omega_{Bz}\\ v_{Bx}\cdot\omega_{Bz}-v_{Bz}\cdot\omega_{Bx}\\ v_{By}\cdot\omega_{Bx}-v_{Bx}\cdot\omega_{By}\end{pmatrix}. \tag{4.2.8}$$

Если аккуратно подставить полученные проекции в уравнения для производной линейной скорости коптера и дальше реализовать все это в SimInTech по осям, получим следующие структурные схемы, представленные на рис. 4.2.1, 4.2.2 и 4.2.3 (показаны проекции только на ось x_B, на другие оси результат аналогичен с точностью до слагаемых).

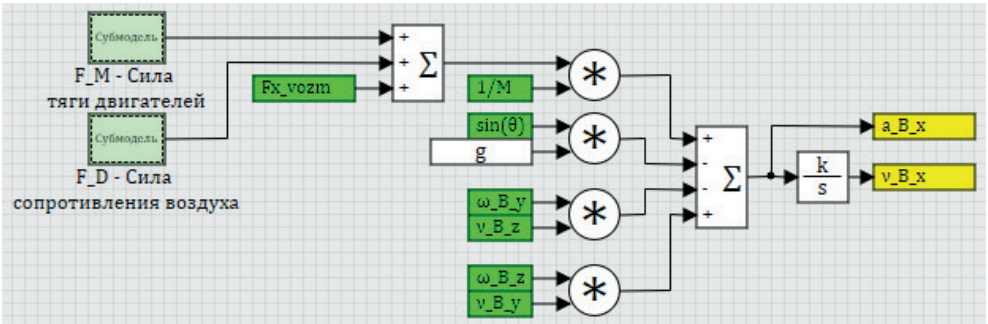


Рис. 4.2.1. Линейное ускорение коптера по оси x_B

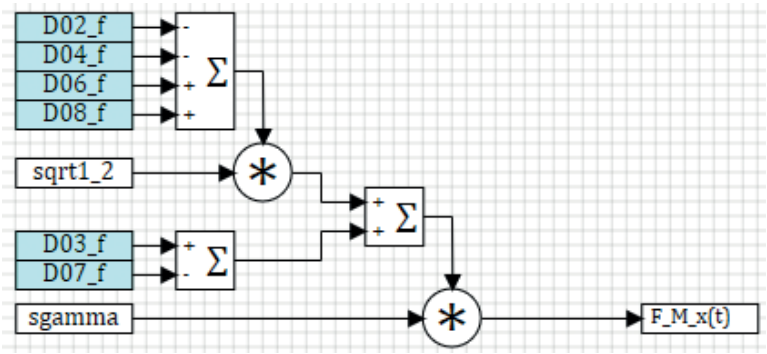


Рис. 4.2.2. Сила тяги ВМГ по оси x_B

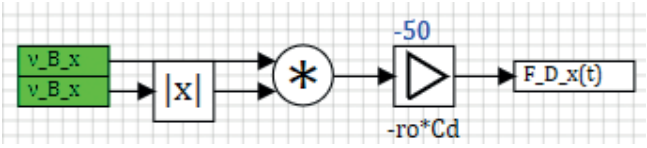


Рис. 4.2.3. Сила сопротивления воздуха по оси x_B

Итого на данном этапе мы до конца реализовали одно из шести основных уравнений (согласно рис. 4.1.1 – вычислили вход только в первый интегратор). Остальные оборванные там входы для сил вычисляются аналогично.

Рассмотрим уравнение моментов (3.1.7) и (3.1.9) для вычисления угловой скорости, а точнее – его слагаемые в правой части:

$$\overrightarrow{\boldsymbol{M_M}}(t)+\overrightarrow{\boldsymbol{M_D}}(t)-\overrightarrow{\boldsymbol{\omega_B}}(t)\times(I_B\cdot\overrightarrow{\boldsymbol{\omega_B}}(t)).$$

Сумма моментов сил тяги всех ВМГ:

$$M_{Mx}(t)=\begin{bmatrix}\frac{l_2}{\sqrt{2}}\left(-F_{M2}(t)-F_{M4}(t)+F_{M6}(t)+F_{M8}(t)\right)\\ +l_1\left(F_{M7}(t)-F_{M3}(t)\right)\end{bmatrix}\cdot\cos(\gamma), \qquad (4.2.9)$$

$$M_{My}(t)=\begin{bmatrix}\frac{l_2}{\sqrt{2}}\left(F_{M2}(t)-F_{M4}(t)-F_{M6}(t)+F_{M8}(t)\right)\\ +l_1\left(F_{M1}(t)-F_{M5}(t)\right)\end{bmatrix}\cdot\cos(\gamma),\qquad(4.2.10)$$

$$M_{Mz}(t)=\begin{bmatrix}l_2\left(F_{M2}(t)+F_{M4}(t)+F_{M6}(t)+F_{M8}(t)\right)\\ -l_1\left(F_{M1}(t)+F_{M3}(t)+F_{M5}(t)+F_{M7}(t)\right)\end{bmatrix}\cdot\sin(\gamma),\qquad(4.2.11)$$

где $F_{Mi}(t)$ – сила тяги i-й ВМГ в текущий момент времени, l_1 И l_2 – плечи сил (длины лучей рамы коптера для нечетных и четных ВМГ)

Момент сопротивления воздуха (при отсутствии ветра):

$$M_{Dx}(t)=-0.5\rho\cdot C_D\cdot A_{xy}\cdot\omega_{Bx}\cdot|\omega_{Bx}|\cdot l_x,\qquad(4.2.12)$$

$$M_{Dx}(t)=-0.5\rho\cdot C_D\cdot A_{xy}\cdot\omega_{Bx}\cdot|\omega_{Bx}|\cdot l_x,\qquad(4.2.13)$$

$$M_{Dz}(t)=-0.5\rho\cdot C_D\cdot 8\cdot A_{xy}\cdot|\omega_{Bz}|\cdot l_z.\qquad(4.2.14)$$

Векторное произведение угловой скорости на произведение тензора инерции и угловой скорости:

$$-\overrightarrow{\boldsymbol{\omega_B}}(t)\times(I_B\cdot\overrightarrow{\boldsymbol{\omega_B}}(t))=\begin{pmatrix}(I_{yy}-I_{zz})\cdot\omega_{By}\cdot\omega_{Bz}\\ (I_{zz}-I_{xx})\cdot\omega_{Bx}\cdot\omega_{Bz}\\ (I_{xx}-I_{yy})\cdot\omega_{Bx}\cdot\omega_{By}\end{pmatrix}.\qquad(4.2.15)$$

Итого после подстановки этих слагаемых в дифференциальное уравнение для угловой скорости и реализации полученного в среде SimInTech можно получить структурные схемы, как показано на рис. 4.2.4:

$$F_{Dx}(t)=-0.5\rho\cdot C_D\cdot A_{yz}\cdot v_x\cdot|v_x|,$$

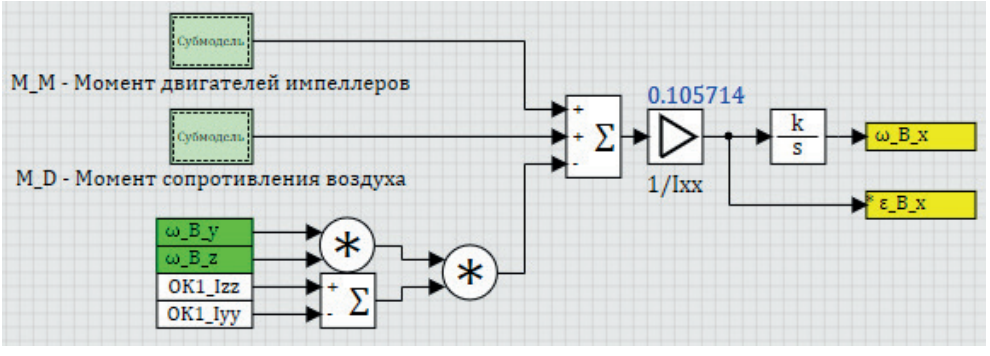


Рис. 4.2.4. Угловое ускорение коптера вокруг оси x_B

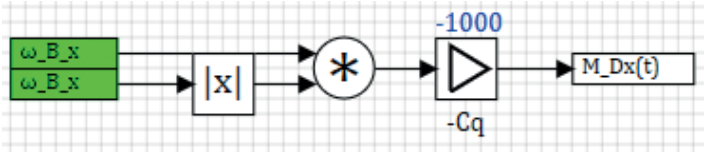


Рис. 4.2.5. Сила сопротивления воздуха при вращении вокруг x_B

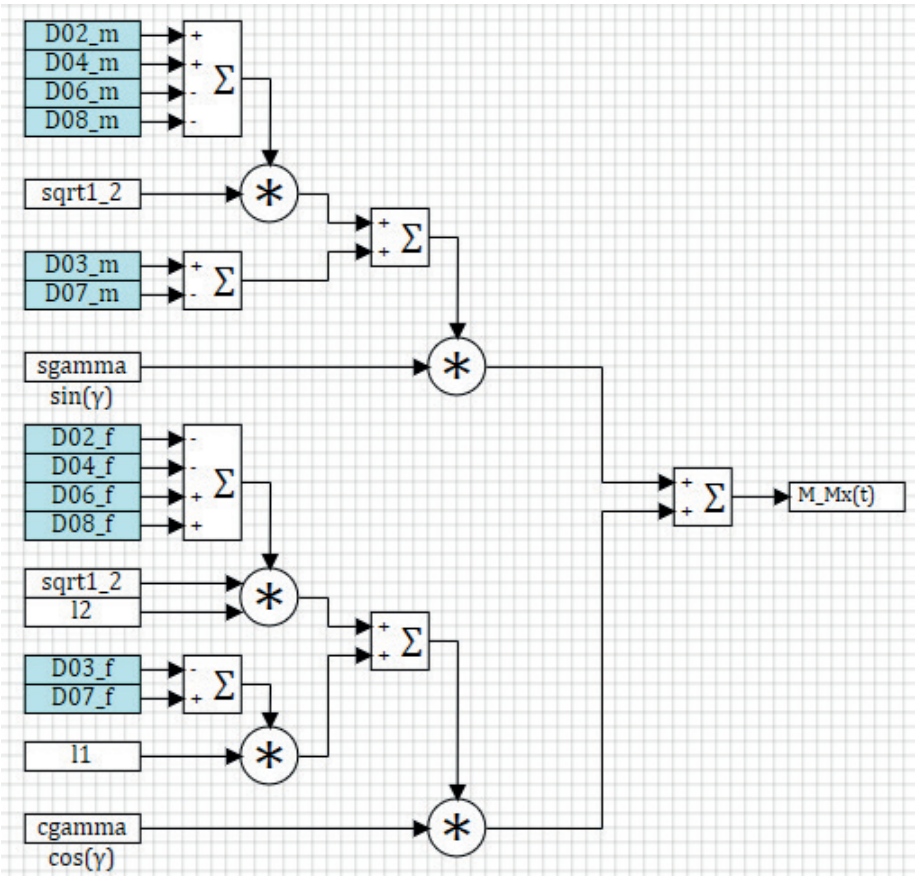
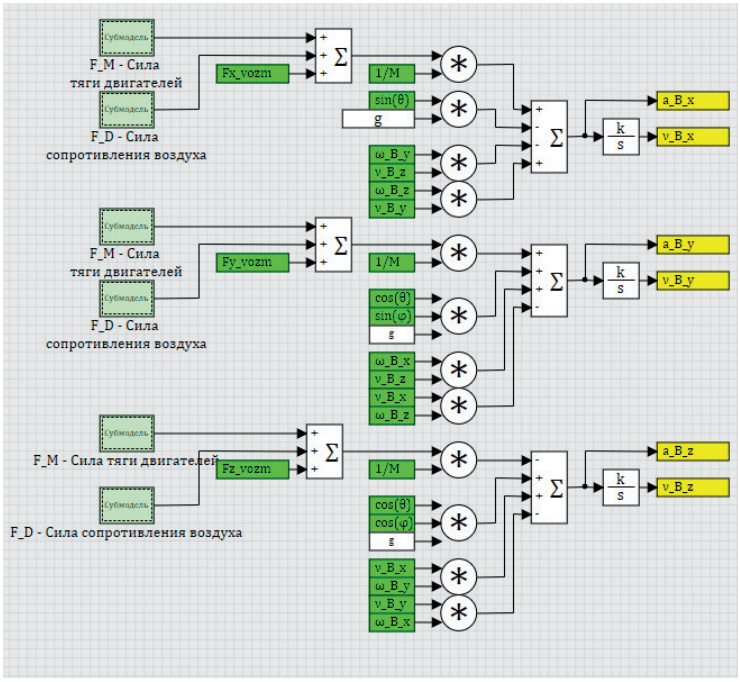


Рис. 4.2.6. Момент сил тяг ВМГ вокруг оси x_B

4.3. Система уравнений динамики октокоптера в структурном виде

Если представить все 6 реализованных уравнений рядом на схеме, получим следующую картину (рис. 4.3.1). Как видно, она довольно громоздка даже без включения сюда других дифференциальных уравнений для вычисления координат коптера и углов поворота. А поскольку в дальнейшем на каждое уравнение будут навешиваться еще дополнительные сервисные вычисления (вычисления ускорений, различных составляющих сил и моментов для отладки модели и регуляторов), целесообразнее эти уравнения структурно разделить на 6 или более субмоделей, в каждой из которых решается свое уравнение, см. рис. 4.3.2.



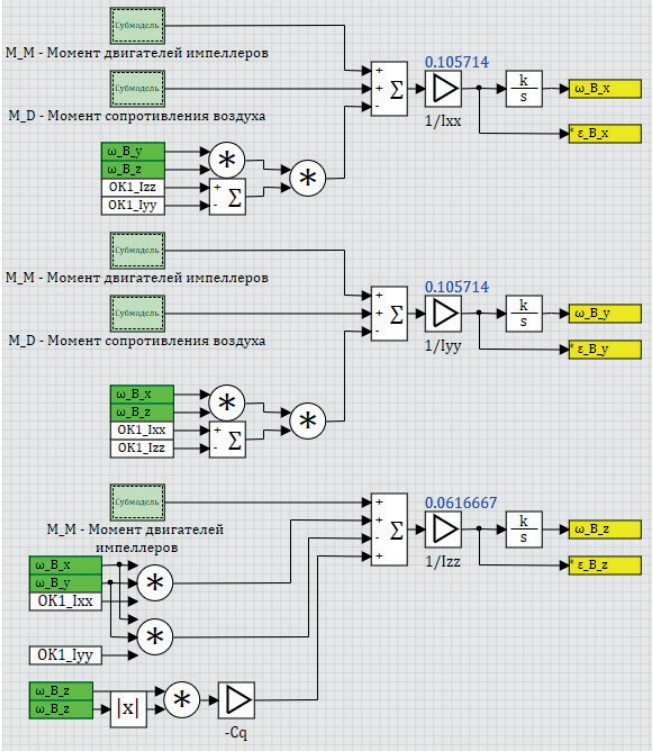


Рис. 4.3.1. Уравнения динамики коптера в связанной системе координат **B**

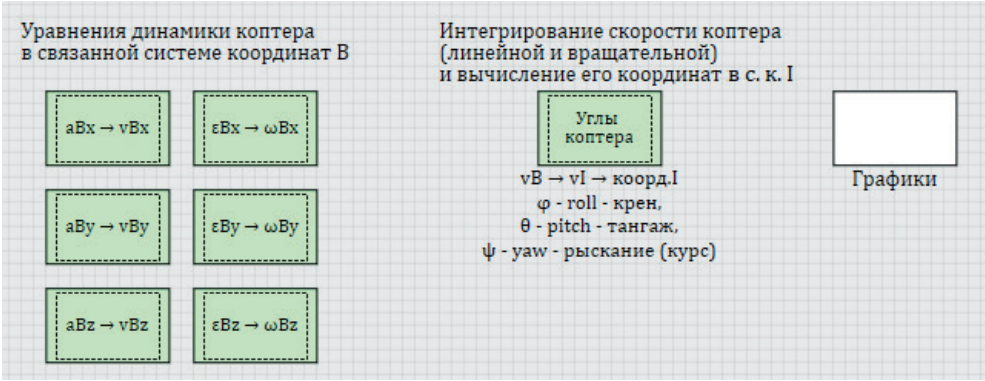


Рис. 4.3.2. Модель динамики коптера, верхний уровень

В принципе, все приведенные уравнения и структуры можно было реализовать и средствами языка программирования SimInTech, но тогда они были бы не так наглядны.

Входными величинами здесь являются силы тяги двигателей (они считаются вне этих уравнений в зависимости от текущей скорости вращения каждой ВМГ и ее силовой характеристики), внешняя возмущающая сила и возмущающий момент сил (задаются пользователем произвольно), сила тяжести. Также возможно еще влияние ветра/прецессия реактивного момента или чего-то еще – в настоящей методике это не рассматривается, для краткости мы ограничились силами тяги двигателей и притяжением Земли.

Выходными величинами являются ускорение, скорость и положение (координаты) коптера – ускорения и скорости в системах координат **B** и **I**, положение – в системе **I** (положение коптера в системе **B** практической ценности почти не имеет, так как сама система движется вместе с коптером и совпадает по положению с ним – т. е. там положение коптера всегда будет нулевым).

В случае модели квадрокоптера или гексакоптера архитектура уравнений динамики будет аналогична, и структурный вид в схемном окне SimInTech будет примерно таким же, как представлено на рис. 4.3.2. Только количество слагаемых в правых частях будет другое, и в зависимости от направления векторов сил тяги ВМГ сами слагаемые тоже будут записаны немного по-другому в соответствии с геометрией объекта.

Управление моделью коптера

В данном разделе представлено описание архитектуры системы управления (в основном регулирования ориентации в пространстве) октокоптера, выполненной в среде SimInTech. Пошаговое выполнение и набор регуляторов будет описан в следующих разделах.

5.1. База сигналов

Все, что было описано до сих пор, – это, по сути, формулировка «внутреннего» закона полета коптера – отвечающего на вопрос, как именно объект будет менять свои ускорения, скорости и координаты в зависимости от приложенных сил и моментов. То есть мы записали уравнение $\boldsymbol{F(t) = ma(t)}$ (и аналогичное для суммарного момента и углового ускорения) для октокоптера заданной геометрической конструкции. В эти уравнения входят некоторые константы – масса коптера, его моменты инерции по главным осям (составляющие тензора инерции), угол отклонения силы тяги ВМГ от вертикали, длины лучей рамы коптера и некоторые другие массогабаритные характеристики. Как правило, это константы, за редким исключением. Но так как задача поставлена в общем виде, мы не подставили еще конкретных чисел, а все эти величины заменили на символы (символьные константы и/или переменные), и теперь требуется их задать числами для работоспособности модели. Для двигателей это будет коэффициент пропорциональности между квадратом угловой скорости и силой тяги, для массы коптера – масса рамы плюс массы всех ВМГ и т. д.

В SimInTech используется плоский неструктурированный список сигналов и/или структурированная (объектно-ориентированная) база сигналов для задания констант и переменных модели, если в ней есть типовые (повторяющиеся) элементы. В данной модели были и тем и другим способом заведены следующие сигналы (переменные и константы), см. рис. 5.1.1 и 5.1.2:

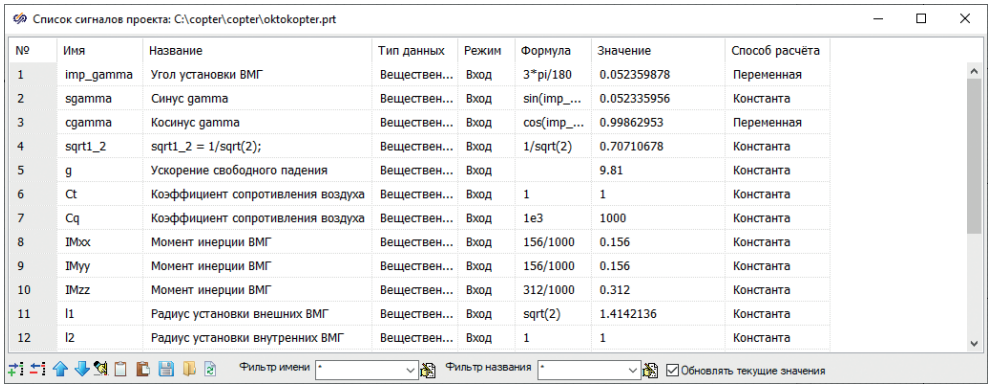


Рис. 5.1.1. Сигналы проекта

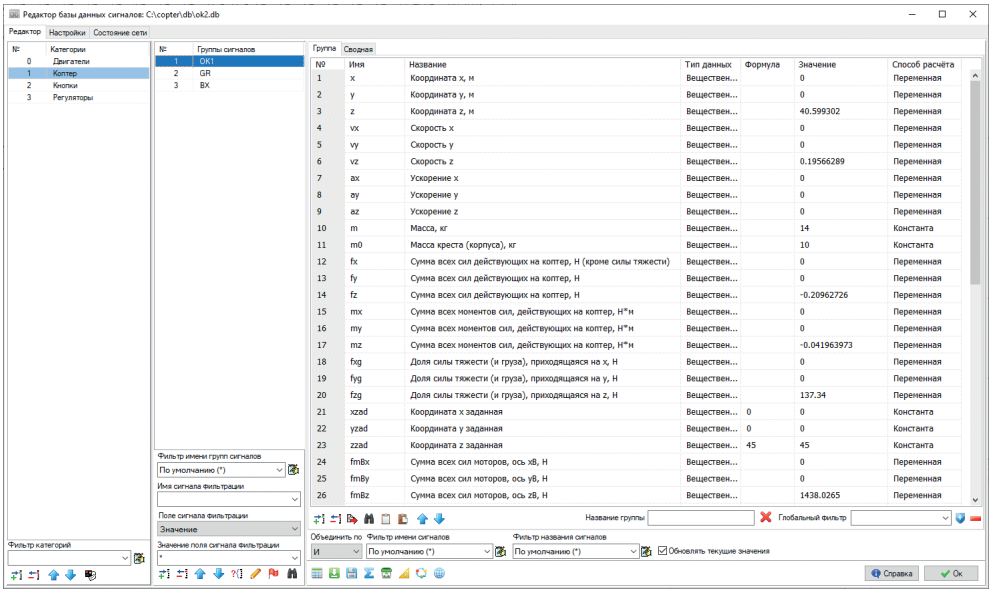


Рис. 5.1.2. База сигналов проекта

Среди повторяющихся элементов в коптере явно можно выделить ВМГ – они все однотипны и параметризуются одинаковыми (по смыслу) переменными. Также для регуляторов, которых будет всего 6 (по числу каналов регулирования), тоже можно выделить однотипные элементы. Если это будут ПИД-регуляторы, то коэффициенты пропорциональности **P**, **I**, **D** будут в наличии у каждого из регуляторов со своим конкретным настроенным значением.

Таким образом, в одном месте в структурированном виде собраны все константы, параметризующие модель, и переменные, используемые для вычислений. В дальнейшем их легко можно забирать отсюда, из базы сигналов, и использовать в других частях модели. При необходимости вносить в модель корректировки констант – это тоже удобно делать, когда они все сведены в одном месте модели, а не разбросаны в разных местах расчетной схемы.

Рассмотрим немного подробнее параметризацию двигателей (см. рис. 5.1.3). Всего в базе 8 групп сигналов, по 15 сигналов в каждой – т. е. 120 переменных и констант, описывающих состояние всех двигателей. Важными и задаваемыми пользователем (разработчиком модели) являются **wnom**, **wmin** – номинальная и минимальная частоты вращения. В дальнейшем эти значения используются в модели двигателя, ограничивая «снизу» текущую частоту вращения и для расчета текущих оборотов ВМГ в единицах измерения. Остальные переменные рассчитываются в модели и зависят от времени. В эту же категорию можно в дальнейшем добавить и коэффициенты пропорциональности между квадратом угловой скорости и силой тяги ВМГ.

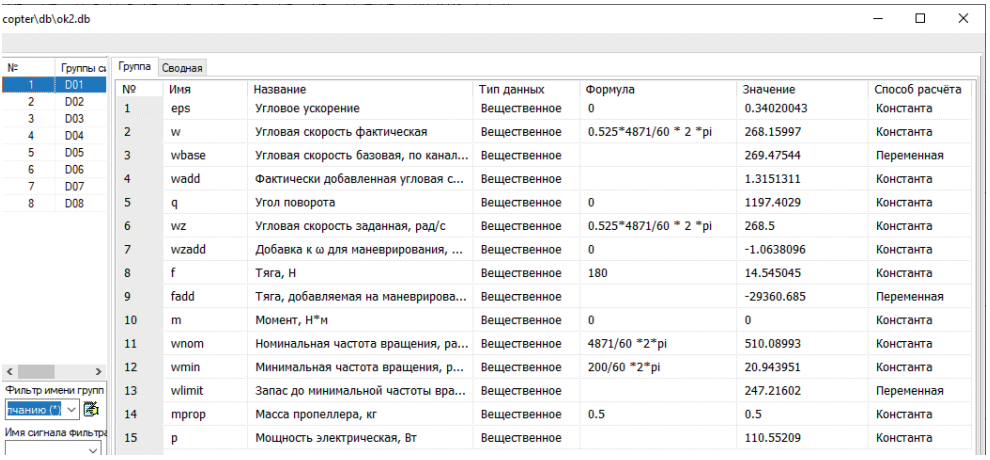


Рис. 5.1.3. Категория «Двигатели»

5.2. Оптимальное управление

С точки зрения управления октокоптер представляет собой не самую простую конструкцию – мы имеем 8 (16) двигателей, которыми можно управлять индивидуально, но практически каждый из них влияет на каждую из 12 переменных состояния (фазовых координат) коптера. То есть, если мы будем «рулить» одним из двигателей – менять его обороты в большую или меньшую сторону, это будет оказывать воздействие на каждую из координат x, y, z и на каждую из трех угловых скоростей; кроме случаев, когда вектор тяги параллелен какой-либо из координатных плоскостей системы **B,** – тогда на 4 из 12 переменных состояния данная сила тяги воздействовать не будет. И если бы мы делали все методами классической теории управления, то можно было бы записать 8 × 12 = 96 передаточных функций между 8 входными воздействиями и 12 выходными (переменными состояния) коптера.

Также изменение оборотов (частоты вращения) двигателей приводит к нелинейному изменению силы тяги, при котором сила тяги пропорциональна квадрату частоты вращения. В проектировании регуляторов коптера это предполагается как аксиома, и она довольно хорошо согласуется с экспериментальными данными.

Задача оптимального управления заключается в том, чтобы перевести коптер из точки А фазового пространства в точку Б с минимальным перерегули-

рованием по оптимальной (как правило, кратчайшей) траектории и за минимальное время. При этом налагаются ограничения на максимальную скорость, ускорения и углы наклона коптера (например, отклонение от горизонтальности). Проблема заключается в том, что в «прямую» сторону можно довольно несложно посчитать, как воздействует каждый из двигателей (8 переменных – 8 частот вращения) на каждую из 12 переменных состояния. А если учитывать еще и ускорения, то на каждую из 18 переменных. Но в задаче управления требуется наоборот – при заданных начальных и конечных координатах требуется вычислить, как именно надо управлять двигателями, чтобы коптер перешел из одного состояния в другое, причем по возможности оптимальным образом.

Если не сильно вдаваться в теоретические дебри теории оптимального управления и нелинейного программирования, такую задачу можно свести и решить методом множителей Лагранжа, а точнее – условиями и методом Каруша-Куна-Такера (в котором ограничения, накладываемые на переменные, представляют собой неравенства).

Чтобы упростить изложение, приведем краткий ход решения задачи. Для начала запишем матрицу **Г** (5.2.1) размерностью 6×8 – по количеству каналов управления (6) и количеству ВМГ (8), которая будет отображать, как именно каждый из двигателей влияет на каждый из каналов управления:

$$\boldsymbol{\Gamma} = \begin{pmatrix} f1x & f2x & f3x & f4x & f5x & f6x & f7x & f8x \\ f1y & f2y & f3y & f4y & f5y & f6y & f7y & f8y \\ f1z & f2z & f3z & f4z & f5z & f6z & f7z & f8z \\ m1x & m2x & m3x & m4x & m5x & m6x & m7x & m8x \\ m1y & m2y & m3y & m4y & m5y & m6y & m7y & m8y \\ m1z & m2z & m3z & m4z & m5z & m6z & m7z & m8z \end{pmatrix}, \qquad (5.2.1)$$

где $fi \mid x,y,z$ – коэффициент перед $\omega_{Mi}^2(t)$ для вычисления силы, которую создает i-я ВМГ в соответствующем направлении, а $mi \mid x,y,z$ – коэффициент перед $\omega_{Mi}^2(t)$ для вычисления момента сил, который создает i-я ВМГ в том же направлении:

$$\overrightarrow{f_i(t)} = C_T \cdot \overrightarrow{e_{Mi}}, \qquad (5.2.2)$$

$$\overrightarrow{m_i(t)} = C_T \cdot \overrightarrow{r_{Mi}} \times \overrightarrow{e_{Mi}}. \qquad (5.2.3)$$

Примечание: если у ВМГ будет еще реактивный момент, то у mi будет еще второе слагаемое.

Легко видеть, что матрица (5.2.1) показывает зависимость между квадратом угловой скорости i-й ВМГ и управляющими (силовыми) воздействиями по каждому из каналов управления: $u(t) = \Gamma \cdot \omega_M^2(t)$. Нам же нужно получить обратное решение – зависимость угловой скорости i-го двигателя от поданного управляющего воздействия по какому-либо из каналов управления. В общем виде эта задача имеет бесконечное множество решений, однако среди этого бесконечного множества можно выделить по тому или иному критерию (или способу) оптимальное решение.

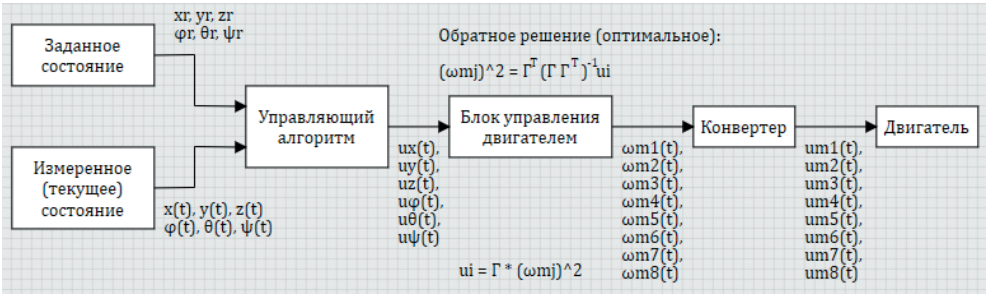


Рис. 5.2.1. Структура регулятора

Методами теории оптимального управления (подробнее см. [1, раздел 4.1.1]) задача решается поиском так называемой обратной псевдоинверсной матрицы A^+, которая вычисляется как: $A^+ = \Gamma^T(\Gamma \cdot \Gamma^T)^{-1}$. Она же и является оптимальным решением поставленной задачи с наложенными ограничениями на решение.

На практике для заданной геометрии коптера и полученных 48 чисел в матрице Γ получаем другие 48 чисел, которые определяют правило управления (микширования) двигателей при поступлении той или иной команды по какому-либо каналу управления. На рис. 5.2.1 представлена общая схема построения регулятора, который приводит коптер к заданному положению в пространстве по заданным координатам. Задатчик положения вырабатывает нужные координаты, в которые требуется привести коптер. Они сравниваются по какому-то алгоритму с измеренными координатами, и управляющий алгоритм вырабатывает 6 управляющих воздействий, а именно по одному воздействию на каждый из каналов управления.

На основе посчитанной псевдообратной матрицы A^+ блок управления двигателями вычисляет текущие заданные значения угловых скоростей для каждого из 8 двигателей, как бы суммируя пришедшие 6 команд по каналам управления и определенным образом микшируя двигатели при этом. Сформированные 8 угловых скоростей отправляются на задатчик оборотов двигателей, и осуществляется регулирование и управление коптером (см. рис. 5.2.1).

Управляющие команды по каждому из каналов управления формируются как рассогласование между заданной координатой (углом) и текущей измеренной координатой. Это в самом простом варианте. В более сложном управляющий алгоритм должен иметь в своем составе алгоритм приоритетности стабилизации положения коптера над алгоритмом перемещения в пространстве. Дело в том, что по каждому из каналов есть «запас управления» (располагаемая у ВМГ возможность). И если (например) мы сделаем приоритетным перемещение по оси x, то при большом рассогласовании между текущей координатой x и заданной регулятор будет стремиться наклонить все больше и больше коптер вокруг оси y, а при определенном наклоне уже не хватит возможностей двигателей обеспечивать стабилизацию коптера и регулирование высоты полета. Для коптера с конкретными параметрами двигателей, винтов, массы и размеров это все можно вычислить и наложить нужные ограничения на управляющие воздействия ui(t), а также их приоритет. Но это выходит за

рамки данной методики, где мы делаем модель в общем виде и с нуля. Просто отметим, что стабилизация положения для коптера – самая важная задача, перемещение – уже вторично, поскольку без стабильного положения переместиться куда-либо целенаправленно будет невозможно.

Выпишем аналитические выражения для элементов первых двух столбцов матрицы Г в случае рассматриваемого октокоптера (без реактивного момента ВМГ и без прецессии!):

$$\begin{aligned}
&f1x = 0,\ f2x = -\frac{1}{\sqrt{2}} C_T \cdot \sin(\gamma),\\
&f1y = -C_T \cdot \sin(\gamma),\ f2y = +\frac{1}{\sqrt{2}} C_T \cdot \sin(\gamma),\\
&f1z = -C_T \cdot \cos(\gamma),\ f2z = -C_T \cdot \cos(\gamma),\\
&m1x = 0,\ m2x = -\frac{1}{\sqrt{2}} C_T \cdot \cos(\gamma) \cdot l_2,\\
&m1y = -C_T \cdot \cos(\gamma) \cdot l_1,\ m2y = +\frac{1}{\sqrt{2}} C_T \cdot \cos(\gamma) \cdot l_2,\\
&m1z = -C_T \cdot \sin(\gamma) \cdot l_1,\ m2z = C_T \cdot \sin(\gamma) \cdot l_2, \ldots
\end{aligned} \tag{5.2.4}$$

Видно, что первая ВМГ (ось силы тяги которой параллельна плоскости yz) не создает никакой силы вдоль оси x и никакого момента вокруг оси x (так как пересекает ее). Поэтому при управлении по оси x или для создания вращательного момента вокруг оси x первый двигатель «бесполезен». Аналогично будет с 5-м двигателем, а для 3-го и 7-го будет такая же картина для оси y (все справедливо в системе координат **B**).

Аналитические выражения для матрицы A^+ будут гораздо более громоздкими и большого смысла не имеют – проще вычислить численно матрицу Г, а потом посчитать и искомую A^+. В нашем случае для одного из вариантов октокоптера она получилась равной (5.2.5):

$$\boldsymbol{A}^+ = \begin{pmatrix}
0 & -191.9 & -5.029 & 0 & 7.112 & -67.86\\
-191.9 & 191.9 & -7.112 & -5.029 & 5.029 & 67.86\\
191.9 & 0 & -5.029 & -7.112 & 0 & -67.86\\
-191.9 & -191.9 & -7.112 & -5.029 & -5.029 & 67.86\\
0 & 191.9 & -5.029 & 0 & -7.112 & -67.86\\
-191.9 & -191.9 & -7.112 & 5.029 & -5.029 & 67.86\\
191.9 & 0 & -5.029 & 7.112 & 0 & -67.86\\
-191.9 & 191.9 & -7.112 & 5.029 & 5.029 & 67.86
\end{pmatrix}. \tag{5.2.5}$$

Смысл матрицы примерно в следующем: она показывает, на какое количество (рад/с)^2 надо изменить квадрат частоты вращения каждой i-й ВМГ (из 8), чтобы осуществить единичное управляющее воздействие на объект по тому или иному каналу управления (каждому каналу управления соответствует свой столбец псевдообратной матрицы). Например, если мы хотим к коптеру

приложить силу величиной 1 Н по каналу управления x (первая колонка), то угловую скорость 1-го двигателя и 5-го менять не надо, а к текущим угловым скоростям других двигателей надо добавить или отнять примерно 13,85 рад/с (13,85 × 13,85 = 191,8). Это довольно приличное изменение угловой скорости, и вообще, управляемость по каналам x, у появилась в конструкции октокоптера как побочное следствие того, что каждая из ВМГ повернута вокруг своего луча еще на 3°. Поворот мы делали для управляемости по курсу, это 6-я колонка матрицы (5.2.5).

Если бы все ВМГ были расположены вертикально, то первые две колонки матрицы A^+ получились бы бесконечными – т. е. коптер с вертикально расположенными ВМГ по горизонтальным осям не управляется вообще! Управление по этим осям (в инерциальном пространстве) достигается поворотом коптера вокруг осей и перенаправлением суммарного вектора тяги в какую-либо сторону.

Третья колонка – управление по оси z – имеет отрицательные числа, так как ось z направлена вниз, а векторы силы тяги ВМГ – вверх. Обратим внимание, что абсолютные значения чисел третьей колонки гораздо меньше, чем первой и второй, и все имеют один и тот же знак. Это означает, что коптер имеет гораздо лучшую управляемость по вертикальной оси, что очевидно, так как все ВМГ как раз и работают практически в эту сторону (а не влево-вправо-вперед-назад). Четвертая колонка и пятая – это управляемость коптера по крену и тангажу, последняя, шестая, – по курсу. Видно, что по курсу коптер гораздо слабее управляем, чем по крену и тангажу. Но для наших целей (и для целей настоящей обучающей методики) этого было достаточно. Подбором направлений сил тяги ВМГ (и перевычислением матриц для новой геометрии) можно этот баланс менять.

Обратим также внимание на то, что теоретически при вычисленном и приведенном здесь количественно микшировании двигателей достигается только управление по выбранному каналу, без влияния на другие каналы управления. То есть при выбранном расположении ВМГ у октокоптера в некоторых малых пределах можно изменять направление вектора тяги без создания поворотных моментов, т. е. не наклоняя сам коптер, и он будет горизонтально лететь и управляться без наклонов. Но только в очень узком диапазоне скоростей и внешних возмущений.

Общая структура регулятора, реализованного в SimInTech, приведена на рис. 5.2.2 и 5.2.3.

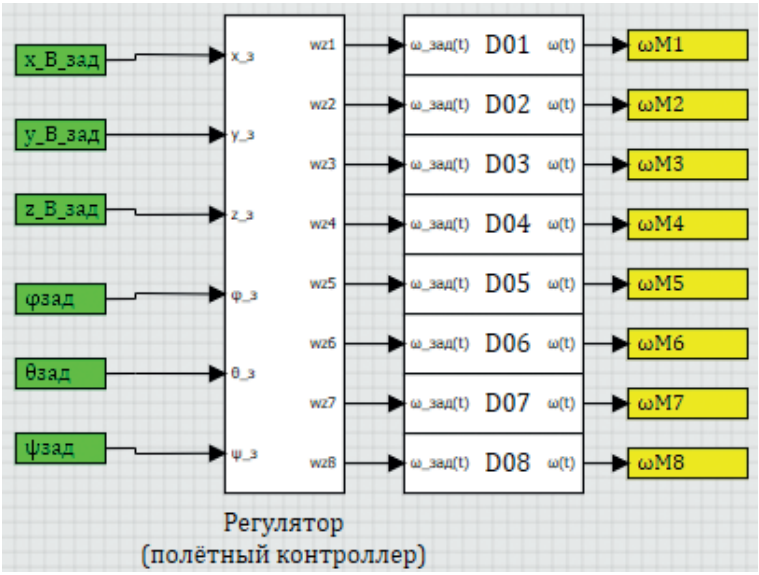


Рис. 5.2.2. Общая схема регулятора

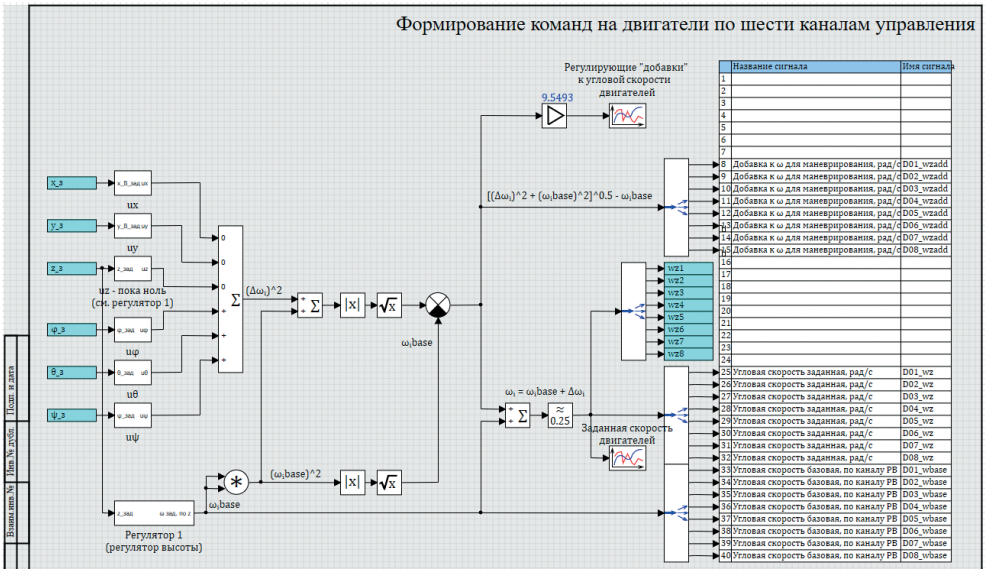


Рис. 5.2.3. Схема каналов регулятора

В самом простейшем случае на 6 входов регулятора поступают заданные фазовые координаты, они сравниваются с текущими (измеренными), и в общем случае в соответствии с матрицей A^+ и ПИД-регулированием по каждому из каналов вырабатывается 6 управляющих векторов, по 8 переменных в каждом, которые суммируются, и получаем итоговый вектор управления с заданными угловыми скоростями для каждого из двигателей. На практике не все так просто…

5.3. Регулятор высоты

Регулятор высоты можно выделить отдельно от других, поскольку он задействует примерно одинаково все двигатели, и даже без матриц Г и A^+ понятно: для того чтобы коптер летел вверх, надо увеличивать газ, а чтобы вниз – уменьшать обороты ВМГ. В качестве простейшего регулятора высоты можно взять обычный ПИД-регулятор, который справится с управлением при верно подобранных коэффициентах. На рис. 5.3.1 показан один из вариантов регулятора с доработкой ограничения вертикальной скорости, если текущая позиция отличается от заданной более чем на 5 м.

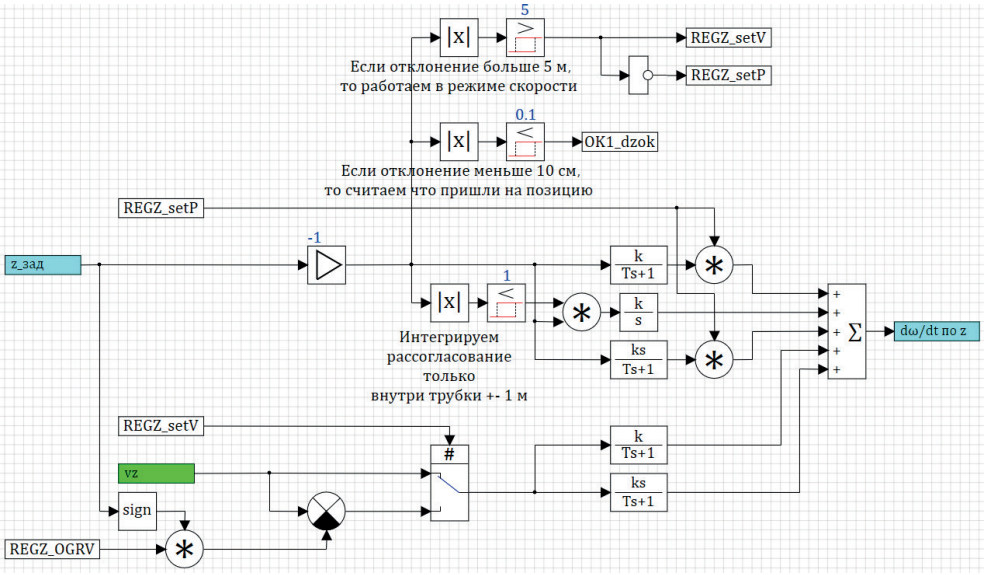


Рис. 5.3.1 . Регулятор высоты, один из вариантов

В чем основная идея регулятора: на вход подается текущая величина рассогласования между заданной высотой и измеренной высотой, подается текущая вертикальная скорость и ограничение на вертикальную скорость. Если рассогласование больше 5 м, то регулятор работает в режиме ограничения скорости и стремится к тому, чтобы вертикальная скорость vz стала равной +OGRV м/с или -OGRV м/с в зависимости от направления полета. Если рассогласование менее 5 м, то регулятор стремится к нулевой скорости полета и сведению в ноль рассогласования по высоте.

Выходной сигнал регулятора подается с коэффициентами матрицы A^+ (согласно третьей колонке) на каждый из 8 двигателей.

5.4. Регулятор ориентации

Углы φ крена и θ тангажа являются одними из самых важных для коптера, так как отвечают за стабилизацию его положения в пространстве, и должны иметь самый высший приоритет среди других каналов управления. Поскольку коп-

тер симметричен, то регуляторы с точностью до осей симметрии похожи друг на друга.

В стабильном состоянии по каждой из осей x_B и y_B угол наклона должен быть нулевым, угловая скорость должна быть нулевой, и приложенный момент сил тоже должен быть нулевым. Регулятор ориентации должен быть достаточно быстродействующим, чтобы успевать парировать внешние возмущающие моменты и не давать коптеру сильно отклоняться от нулевой позиции, так как любой наклон коптера приводит к уменьшению вертикальной составляющей силы тяги и к уводу коптера в соответствующую сторону вбок.

Для сведения к нулю трех составляющих – угла, скорости и момента – они должны быть входными сигналами в регулятор. Один из вариантов представлен на рис. 5.4.1. Заданное значение равно нулю, и в стабильном состоянии выходной сигнал сумматора будет равен нулю. Если появляется какое-то ненулевое состояние, то выход регулятора будет также отличен от нуля. Затем он умножается на вектор, равный 4-й колонке псевдообратной матрицы A^+, представленной выше, и формируется вектор усилий на 8 ВМГ, который должен компенсировать возникший дисбаланс поворота вокруг оси x_B.

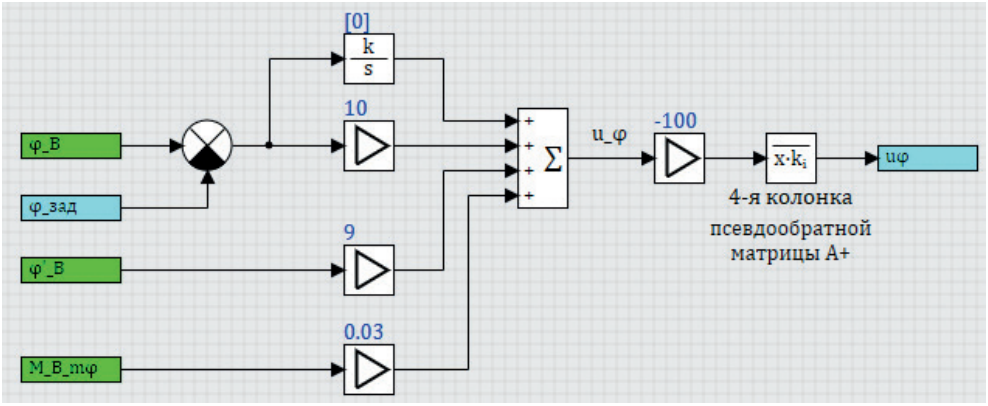


Рис. 5.4.1 . Регулятор крена

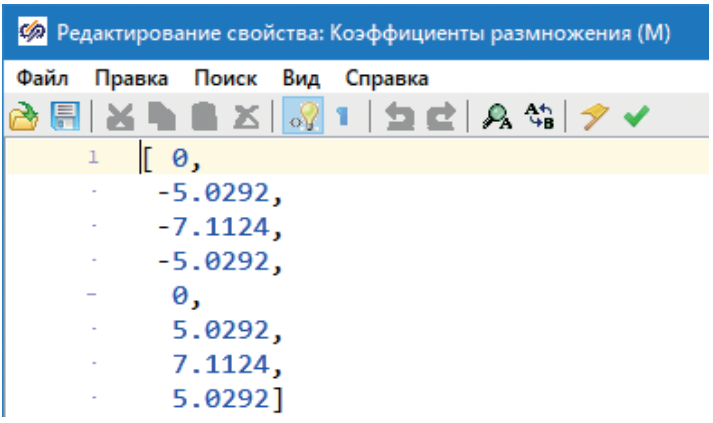


Рис. 5.4.2. Коэффициенты блока типа «Размножитель» в регуляторе крена

Аналогично выполнен и регулятор по каналу тангажа (представлен на рис. 5.4.3).

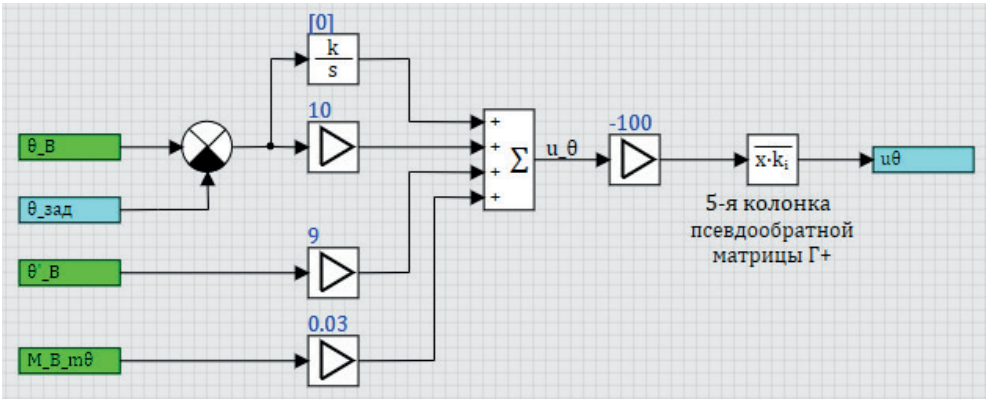


Рис. 5.4.3. Регулятор тангажа

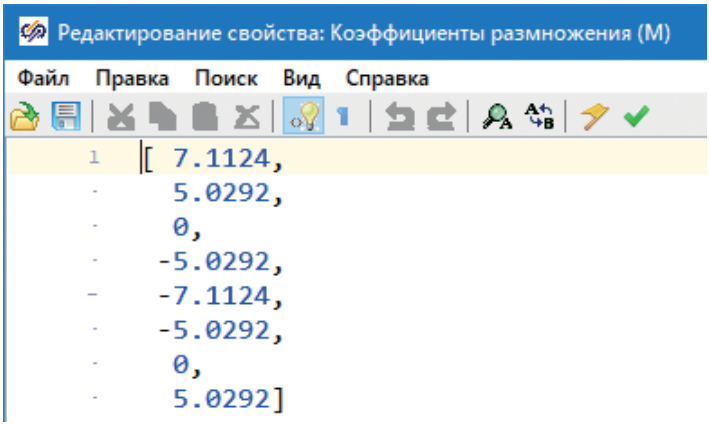


Рис. 5.4.4. Коэффициенты блока типа «Размножитель» в регуляторе тангажа

Конкретные значения на линиях регулятора, пропорциональных углу, угловой скорости и угловому ускорению (моменту сил), подбираются численным экспериментом для данного коптера (с конкретными массой, моментами инерции по осям, двигателями, размером рамы и т. д.).

5.5. Регулятор положения в пространстве

Итого регулятор ориентации коптера и регулятор высоты, работая совместно, обеспечивают «висение» коптера в определенной точке пространства в состоянии динамического равновесия. При этом у коптера еще остается некоторый запас управляемости, который позволяет ему двигаться целенаправленно вдоль координатных осей. Но данные регуляторы несколько сложнее, чем регулятор ориентации.

Во-первых, они работают в системе координат **B**, а в общем случае коптер вращается в пространстве (по курсу), и направления осей x и y системы **B** не

совпадают с такими же осями в системе **I**. Поэтому рассогласование его позиции в системе **I** надо переводить в систему **B** и дополнительно подготавливать задание на регулятор положения по осям *x* и *y*.

Во-вторых, рассогласование в этом регуляторе может быть как небольшим, так и значительным, а обычный ПИД-регулятор, как правило, не может одинаково эффективно работать с малыми и с большими отклонениями, требуется доработка регулятора – например, переключение регулятора положения в режим поддержания постоянной скорости при каких-то условиях.

В-третьих, у классического коптера с винтами, расположенными в одной плоскости и параллельными силами тяги, направленными вверх, практически нет возможности создавать боковую силу тяги – т. е., по существу, коптер является неуправляемым по осям *x* и *y*. В нашем варианте, когда винты довернуты еще на 3° вокруг своих лучей, у них появляется небольшая сила тяги, направленная в стороны, и микшированием двигателей можно создавать боковую силу тяги. Однако она очень несущественна, почти нулевая – об этом свидетельствуют большие числа в 1 и 2 колонках матрицы A+ –. Для того чтобы создать силу тяги величиной 1Н в сторону, нужно изменить частоту вращения двигателей почти на 15 рад/с (квадрат частоты вращения – на 191 $(1/с)^2$). С точки зрения управления это слишком большая величина. Поэтому коптеры управляются по направлениям x и y при помощи других каналов управления – поворачиваясь вокруг осей x и y (об этом напишем дальше).

Но структурно – если делать регулятор по каналам x и y типовым образом, регулятор может быть выполнен аналогично регуляторам ориентации – на выходе формируется управляющее воздействие по каналу x (y), и оно умножается на вектор – 1-ю (или 2-ю) колонку матрицы A^+, а полученные 8 сигналов подаются на ВМГ. Один из вариантов регулятора по каналу x и y представлен на рис. 5.5.1 и 5.5.2.

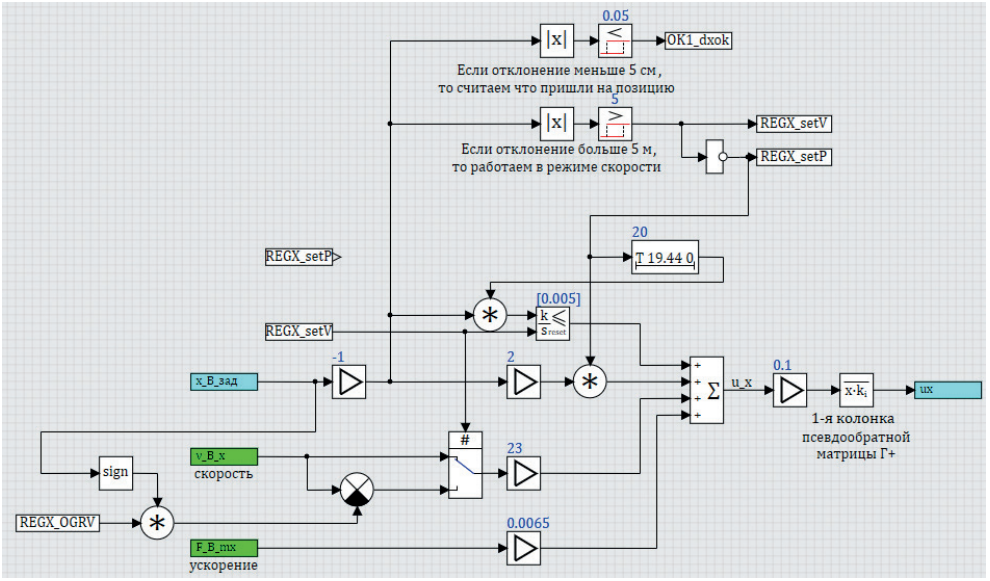


Рис. 5.5.1. Регулятор по каналу *x*

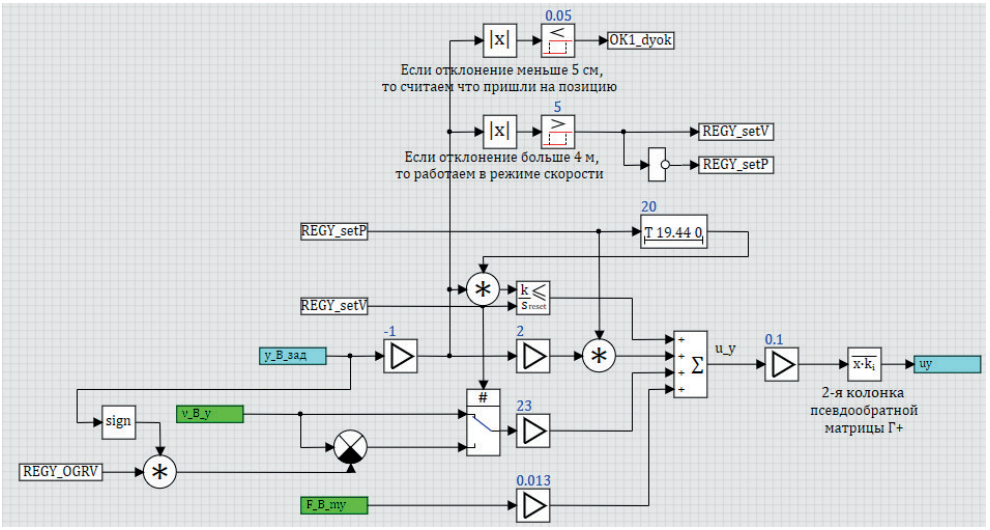


Рис. 5.5.2. Регулятор по каналу *y*

Регуляторы двухрежимные, при отклонении от заданной позиции более чем на 5 м переключаются в режим работы «V» и поддерживают скорость по направлению на уровне REGX_OGRV (или REGY_OGRV) м/с. При этом интегрирующая ветка регулятора отключается. При переходе в режим позиции интегрирующая ветка включается в работу с некоторой задержкой – чтобы коптер успел подлететь к заданной точке и не набралась существенная величина на интеграторе за время «подлета».

Но отметим еще раз – такой подход будет справедлив и оправдан при существенной управляемости коптера по горизонтальным осям, что может быть достигнуто относительно большим наклоном винтов от вертикальной оси.

Приведенные здесь регуляторы, хотя и кажутся сложными на первый взгляд, являются только лишь базовыми версиями, которые позволяют управлять коптерами. Дальнейшая разработка модели может (и должна) привести к более сложным регуляторам и к повышению качества переходных процессов.

В сумме на выходе всех 6 каналов управления (по курсу регулятор во многом аналогичен регуляторам ориентации, и для сокращения материала не приводим его здесь) мы имеем по каналу регулятора высоты некоторую «базовую» желаемую частоту вращения для каждой из 8 ВМГ и некоторую «добавку», сформированную остальными 5 каналами управления. Единственный нюанс – микшированные добавки, так как это не прямая добавка к частоте вращения, а добавка к квадрату частоты вращения, и, для того чтобы вычислить добавку именно к частоте вращения, надо еще дополнительно проделать несложные математические вычисления, см. рис. 5.5.3:

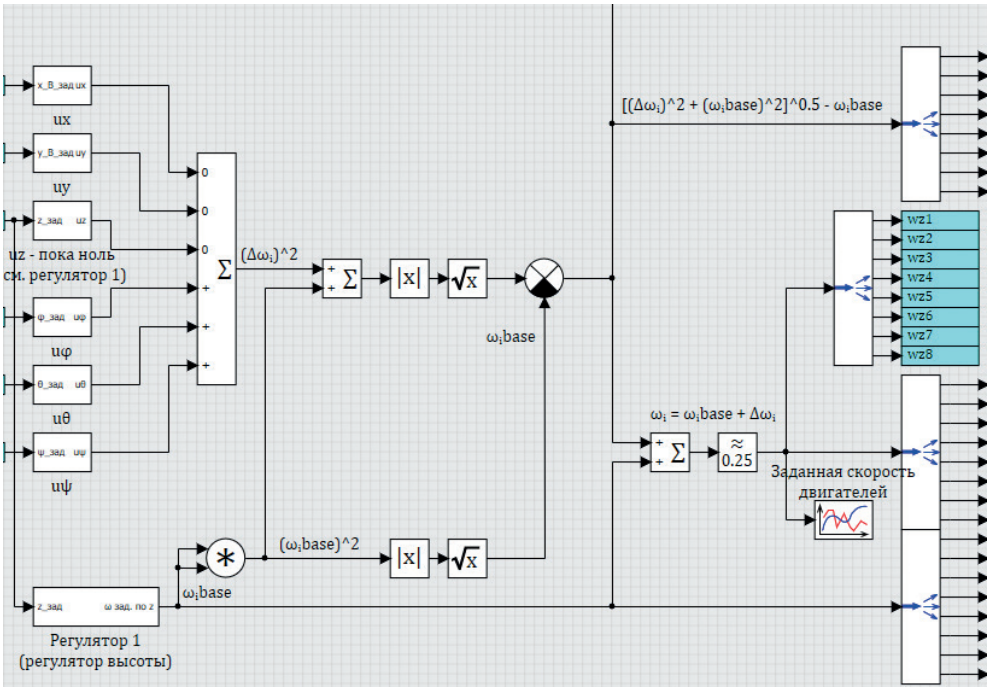


Рис. 5.5.3. Суммирование каналов управления

Сложность вычисления в том, что добавка частоты вращения зависит и от нужного квадрата добавки, и от самой базовой частоты вращения – и чтобы вычислить итоговую частоту вращения как сумму базовой и добавки надо предварительно сделать еще ряд вычислений:

$$\omega_i = \omega_{ibase} + \Delta\omega_i, \tag{5.5.1}$$

$$\Delta\omega_i = \sqrt{\left(\sum \omega_i\right)^2 + \left(\omega_{ibase}\right)^2} - \sqrt{\left(\omega_{ibase}\right)^2}, \tag{5.5.2}$$

где квадраты скоростей – это сигналы, выходы каналов управления.

Таким образом, в зависимости от текущего уровня «базовой» частоты вращения (которая определяется в основном массой коптера и, возможно, груза и получается на выходе из регулятора высоты) определяется нужная «добавка» угловой скорости для каждой из ВМГ, суммируется с базовой частотой вращения и отправляется как задатчик на регулятор двигателя каждой из ВМГ.

5.6. Регулятор положения в пространстве по каналам крена и тангажа

Опишем кратко подход, который используется для формирования задания по крену и тангажу для того, чтобы коптер летел в заданном направлении. Для простоты предположим, что курс коптера нулевой (угол ψ равен нулю всегда), тогда для того, чтобы коптер двигался вдоль оси X, требуется изменить его

угол тангажа θ; а для того, чтобы он двигался вдоль оси Y, требуется изменить угол крена.

Путем несложных выкладок можно получить следующие формулы для вычисления угла наклона:

$$\varphi_{зад} = \arcsin\left(y_{задI}\right), \tag{5.6.1}$$

$$\theta_{зад} = \arcsin\left(\frac{x_{задI}}{\cos\left(\varphi_{зад}\right)}\right), \tag{5.6.2}$$

где $y_{задI}$ и $x_{задI}$ – рассогласования заданных значений координат и измеренных в инерциальной системе отсчета, а выражения под арксинусом дополнительно ограничиваются некоторой величиной (полученной исходя из запасов управляемости коптера по крену и тангажу), например диапазоном $\left[-\frac{\pi}{16};+\frac{\pi}{16}\right]$.

Если такие величины подать вместо нулевого значения на вход регуляторам крена и тангажа, то регулятор будет дополнительно доворачивать коптер на небольшой угол для полета вдоль своих осей x и y. Если учитывать еще и возможность изменения курса, то формулы получатся посложнее, но смысл управления будет тем же самым. Реализация такого управления показана на рис. 5.6.1.

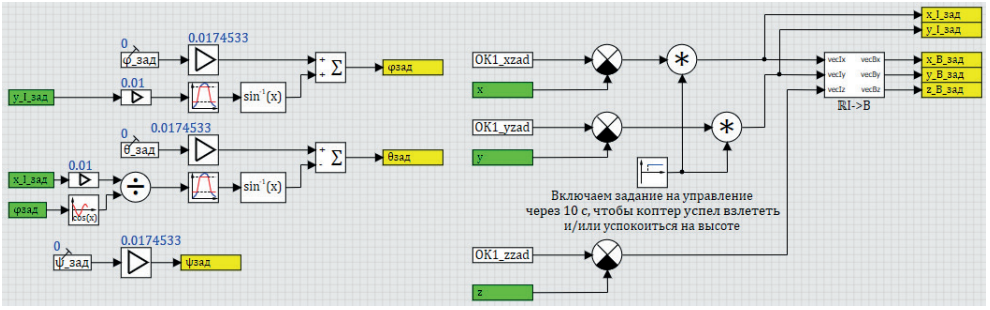


Рис. 5.6.1. Формирование задания на крен и тангаж

Пульт управления

6

С целью оперативного контроля и управления моделью коптера в SimInTech может быть разработан макет пульта управления, на который выведены основные параметры полета и органы управления, позволяющие изменять текущее задание по координатам, или еще какие-либо настроечные параметры модели. Также, в дополнение к расчетной схеме при помощи технической анимации среды SimInTech, может быть сделана плоская анимация (в плоскостях X–Z и X–Y) модели коптера, на которой можно в режиме реального времени наблюдать за полетом модели. При помощи расширенных средств SimInTech, может быть выполнена и анимация в трехмерном пространстве с привязкой координат и углов поворота объектов в пространстве к расчетным переменным.

6.1. Пульт управления, его подключение через базу сигналов

На пульт управления (рис. 6.1.1) при помощи типовых средств выведены текущие координаты полета и заданные значения координат (в инерциальной системе отсчета), а кнопками можно задавать новые заданные значения (менять уставки для регуляторов). Также можно в произвольный момент времени подавать возмущающие силы по направлениям коптера.

На пульте управления были выполнены некоторые прототипы приборов, например авиагоризонт, см. рис. 6.1.2.

Рис. 6.1.1. Макет пульта управления

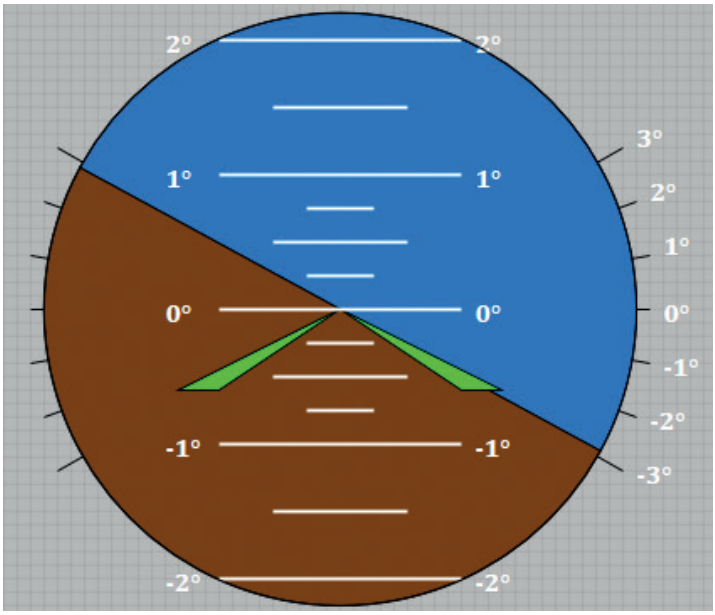


Рис. 6.1.2. Прототип авиагоризонта с утрированно увеличенными углами

6.2. Анимация

При помощи технической анимации SimInTech можно создавать различные анимационные иллюстрации к модели. Например, для октокоптера с целью визуальной отладки модели было сделано два вида на модель коптера в пространстве – вид сбоку (плоскость x–z, рис. 6.2.1) и вид сверху (плоскость x–y, рис. 6.2.2).

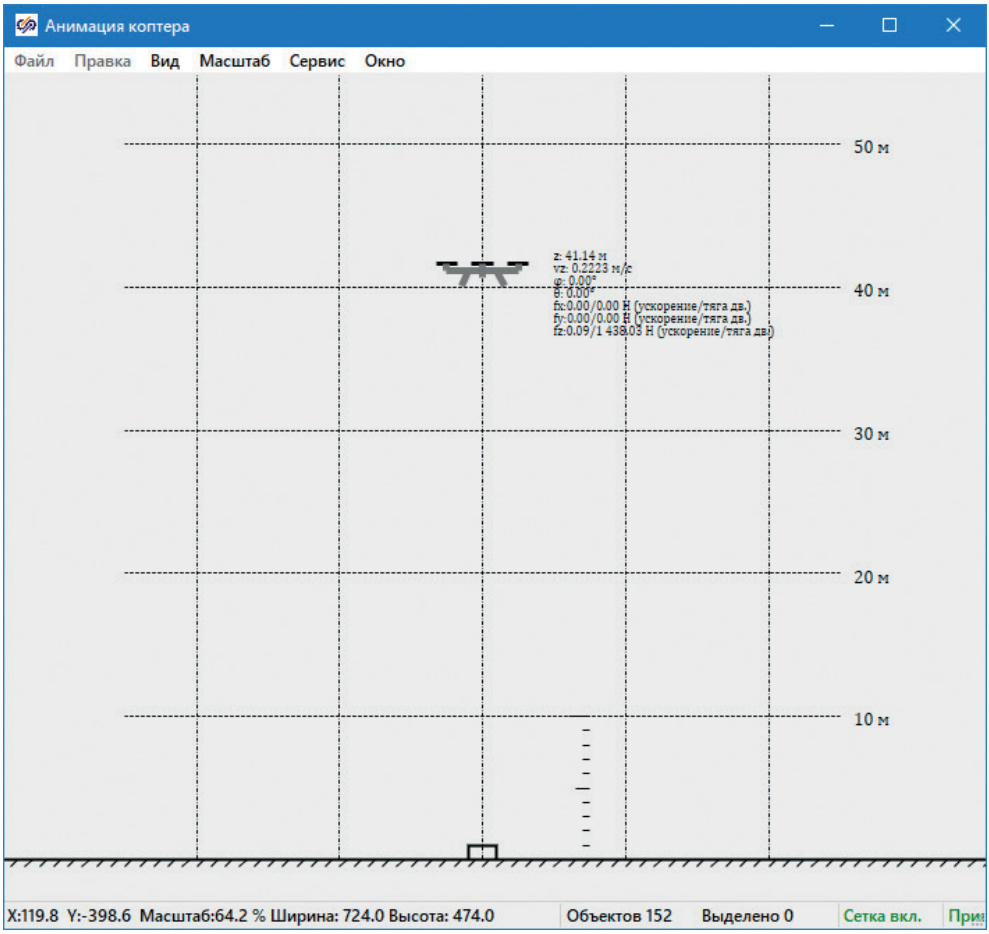
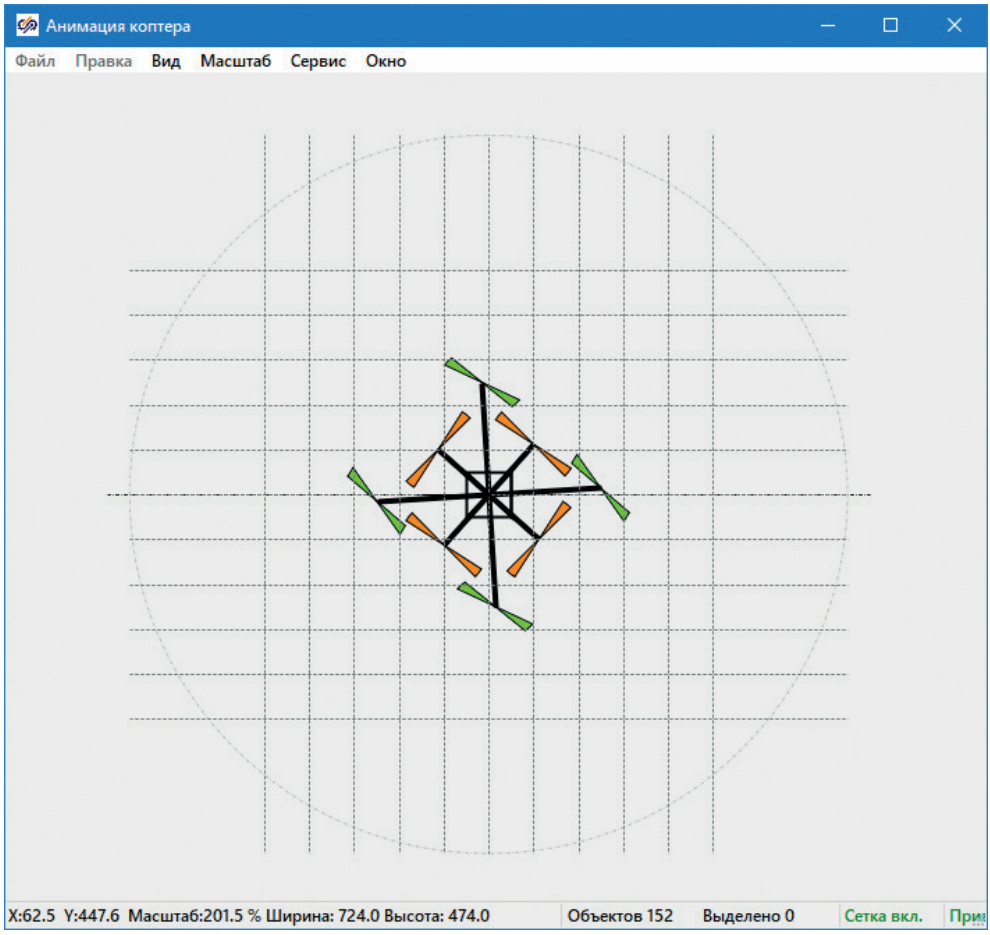


Рис. 6.2.1. Анимация, вид сбоку



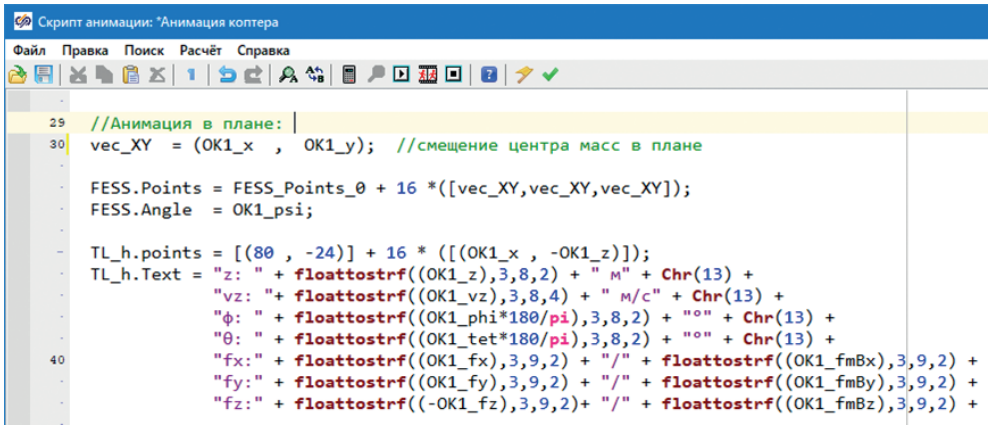


Рис. 6.2.2. Анимация, вид сверху и часть кода анимации

Модуль трехмерной анимации SimInTech позволяет реализовать пространственную анимацию модели и ее окружения для анализа поведения модели в пространстве. На практике даже плоская анимация позволяет на ранних стадиях и оперативно выявить огрехи исполнения регуляторов или модели, становятся видны нефизичные проявления в случае ошибки в уравнениях, что позволяет исправить их и/или скорректировать нужным образом в правильную сторону.

В процессе создания модели, как правило, выполняется большое количество тестовых расчетов и моделирований, некоторая часть из них приведена ниже в этом разделе.

6.3. Тестовый полет 1

На рисунках ниже представлены графики полета по следующей программе: исходное состояние коптера – координаты x = 0 м, y = 0 м и 40 м высоты, затем вертикальный взлет до 60 м, полет вправо на 20 м (по оси x), движение вниз на отметку +40 м и задание еще одного смещения влево на 10 м, не дожидаясь завершения маневра вниз. Фазовый портрет полета (координата центра масс коптера в процессе перемещений) представлен на рис. 6.3.1.

Отметим, что при горизонтальном полете коптер немного «просел» вниз – это произошло из-за снижения вертикальной составляющей силы тяги при наклоне коптера. Очевидно, что регуляторы еще требуют доработки.

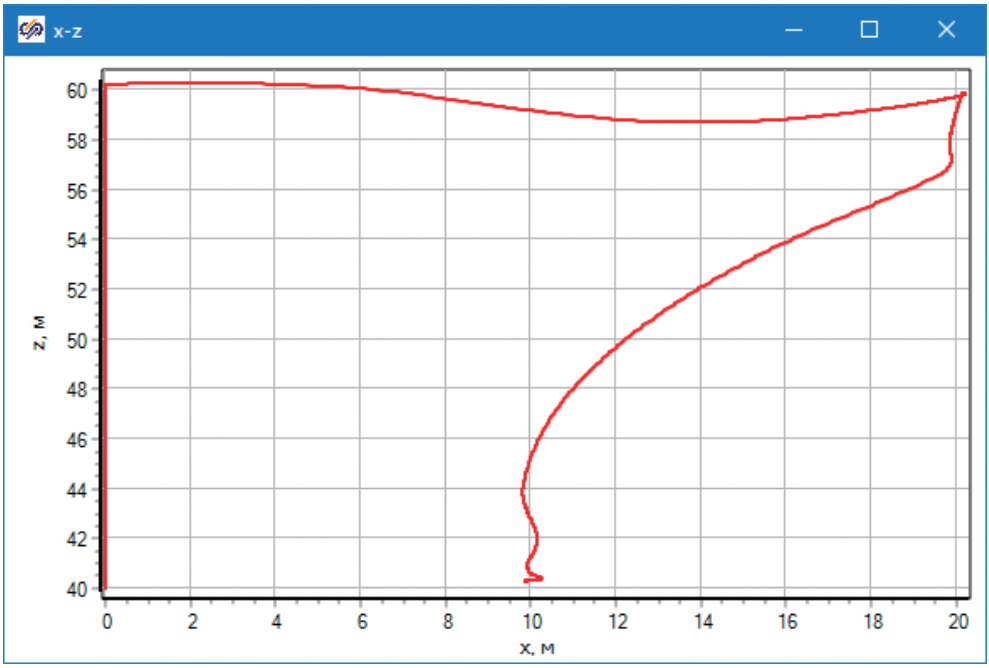


Рис. 6.3.1. Тестовый полет 1, фазовый портрет в координатах x–z, м

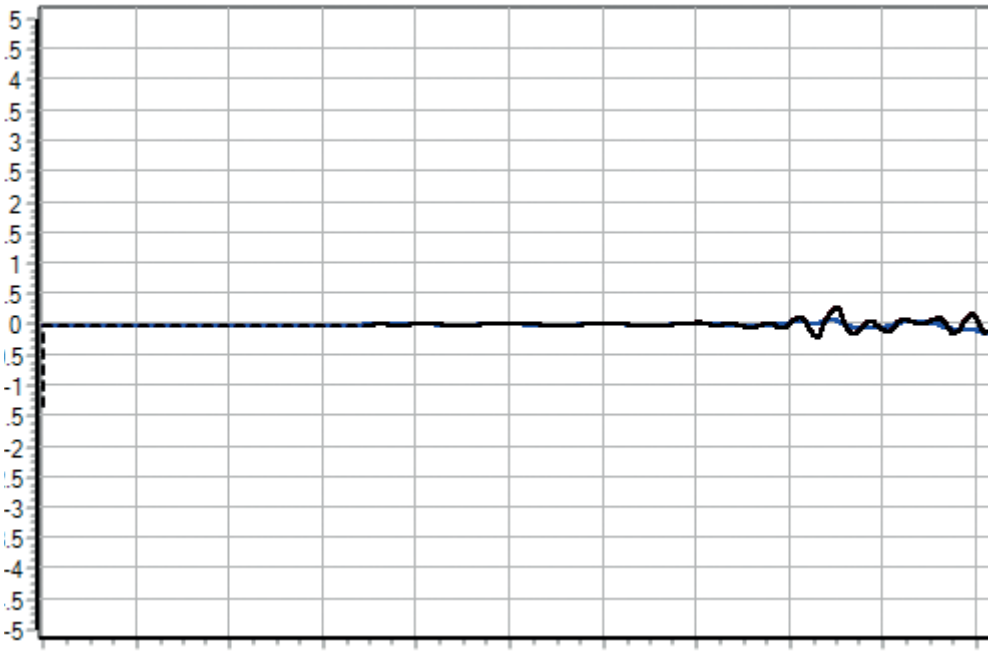


Рис. 6.3.2. Тестовый полет 1, крен (заданный и измеренный)

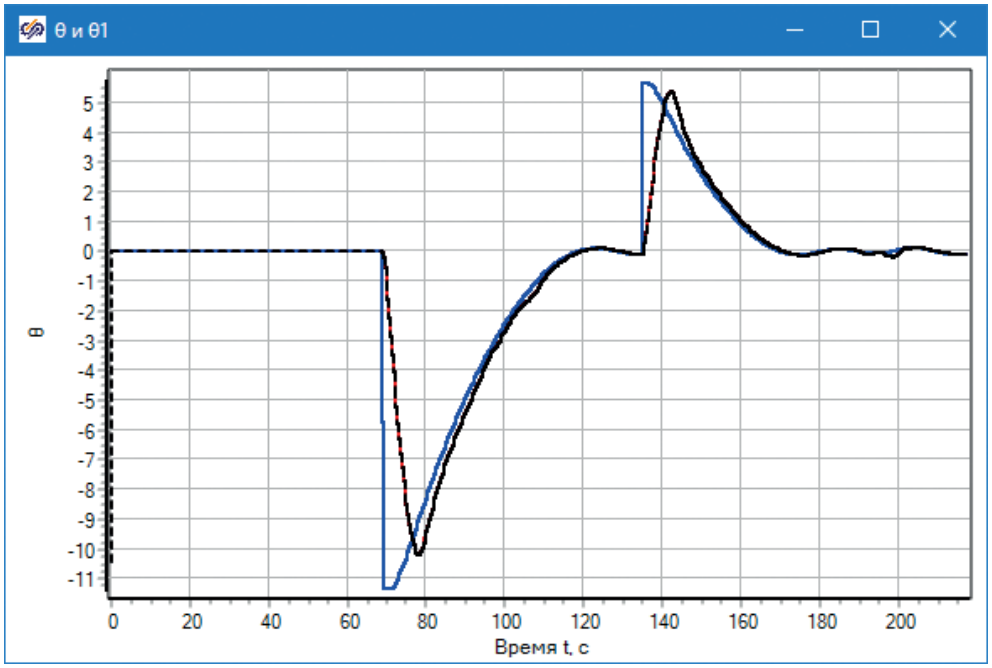


Рис. 6.3.3. Тестовый полет 1, тангаж (заданный и измеренный)

На рис. 6.3.2 и 6.3.3 представлены заданные (синяя линия) и измеренные углы (черная линия) крена и тангажа. Так как полет проходил вдоль оси x (в основном), то угол крена находится в околонулевых значениях. Заданный угол тангажа меняется скачком в моменты поступления с пульта команд на изменение заданной координаты x.

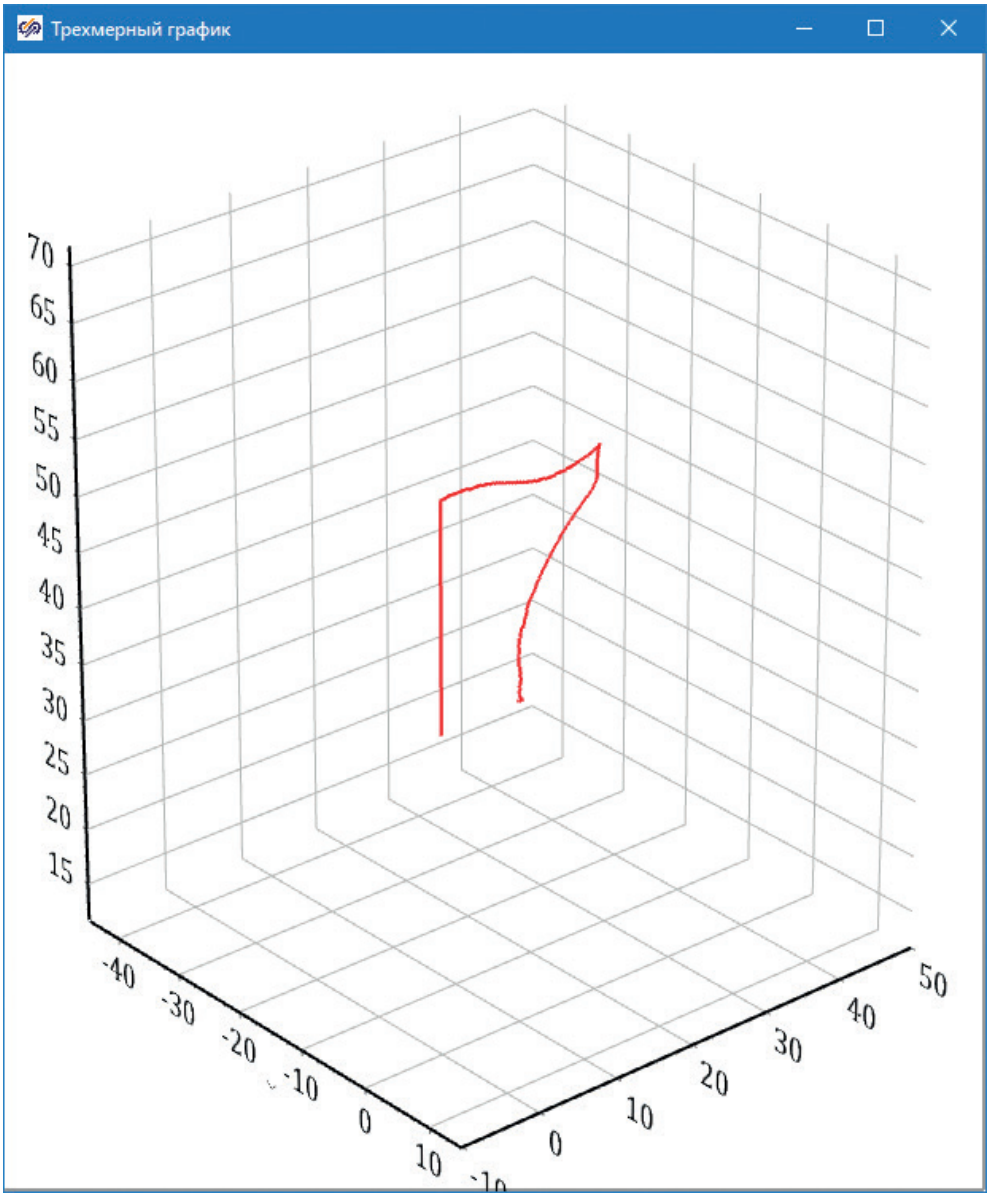


Рис. 6.3.4. Тестовый полет 1, траектория полета в пространстве

Рис. 6.3.5. Угловая скорость двигателей

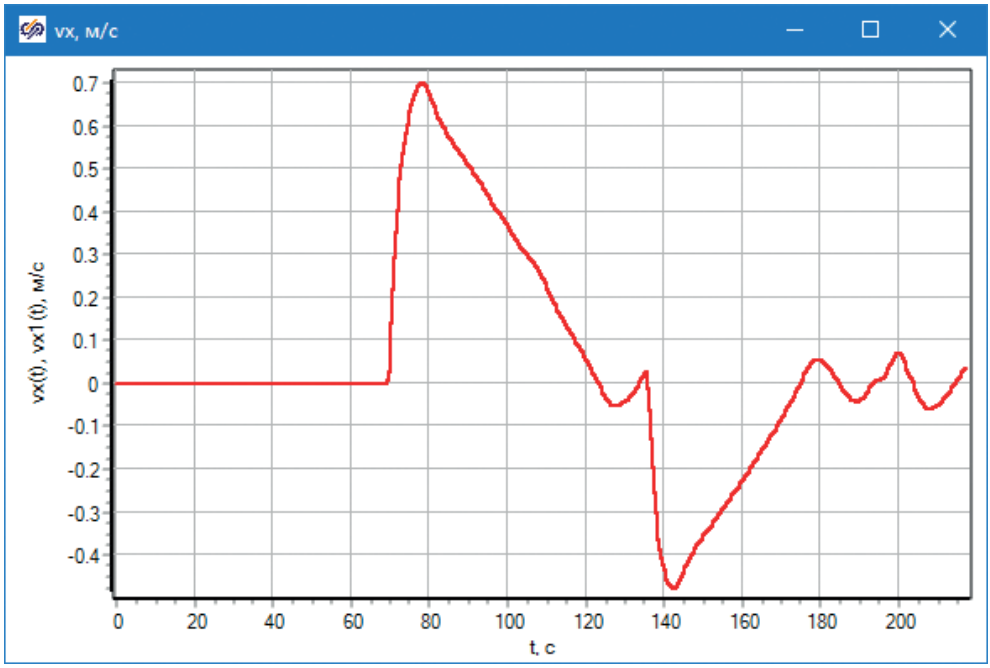


Рис. 6.3.6. Скорость коптера вдоль оси x_l

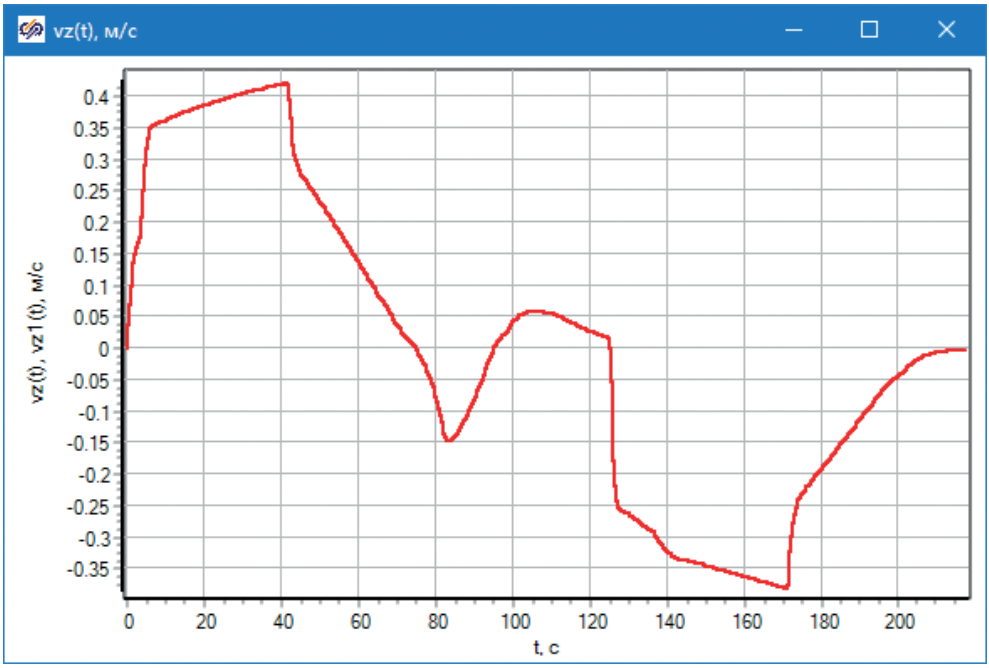


Рис. 6.3.7. Вертикальная скорость коптера (по оси z)

На рис. 6.3.6 и 6.3.7 представлены скорости коптера вдоль оси x и z. Путем совместного анализа этих графиков и графика траектории полета (а также графиков внутренних параметров регуляторов) можно проводить анализ действия регулятора и дальнейшую их модификацию, корректировку и улучшение.

6.4. Тестовый полет 2

Исходное состояние то же самое: координаты 0, 0, 40 м. Подается команда на +10 м по оси x, и в момент, когда коптер пролетел 1 м, подается такая же команда на +10 м по оси y, коптер летит под двум направлениям одновременно. Когда он достигает 10 м по обеим осям, подается команда вернуться по оси x в ноль, и, когда он возвращается, подается команда вернуться по оси y в ноль. Заданная высота полета сохраняется все время +45 м.

На рис. 6.4.1 приведена фазовая траектория полета по осям x–y, на рис. 6.4.2 – по осям x–z (видно, что коптер взлетел до 45 м и потом слабо отклоняется от этой высоты).

На следующих рисунках (6.4.3…6.4.7) приведены остальные параметры модели и коптера в этом тестовом полете.

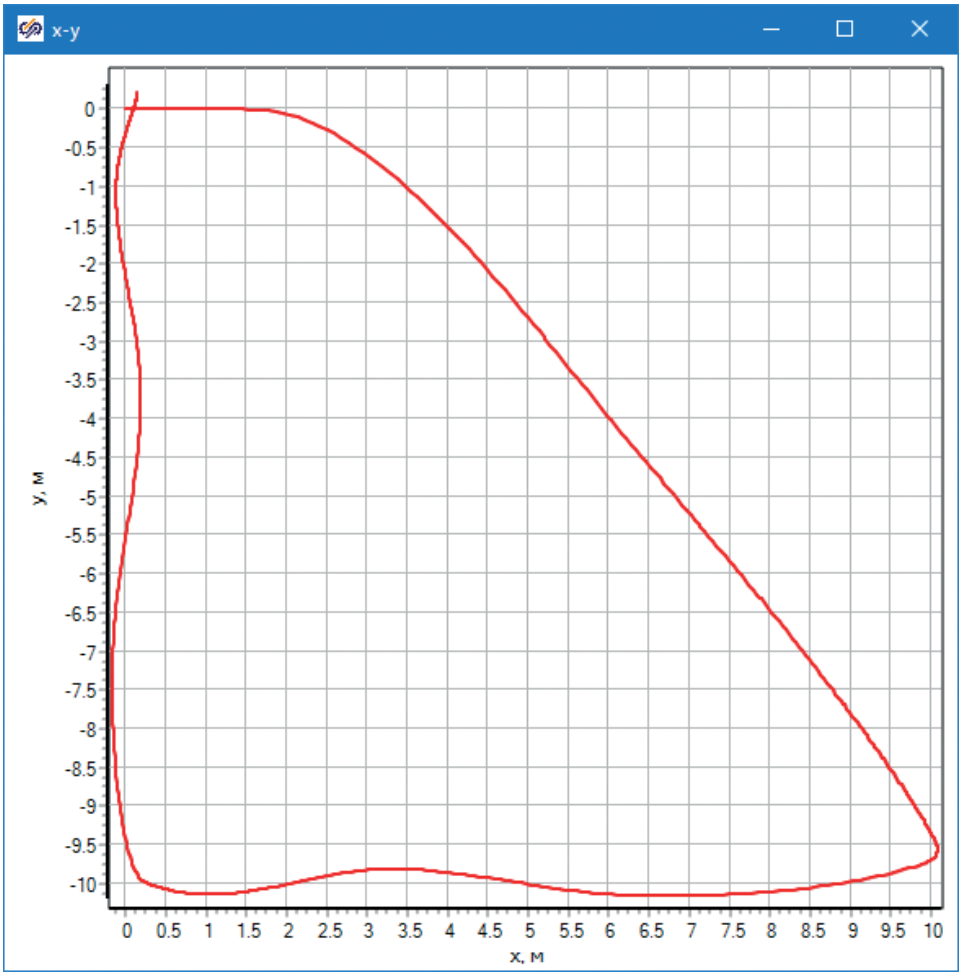


Рис. 6.4.1. Тестовый полет 2, траектория в осях x–y, м

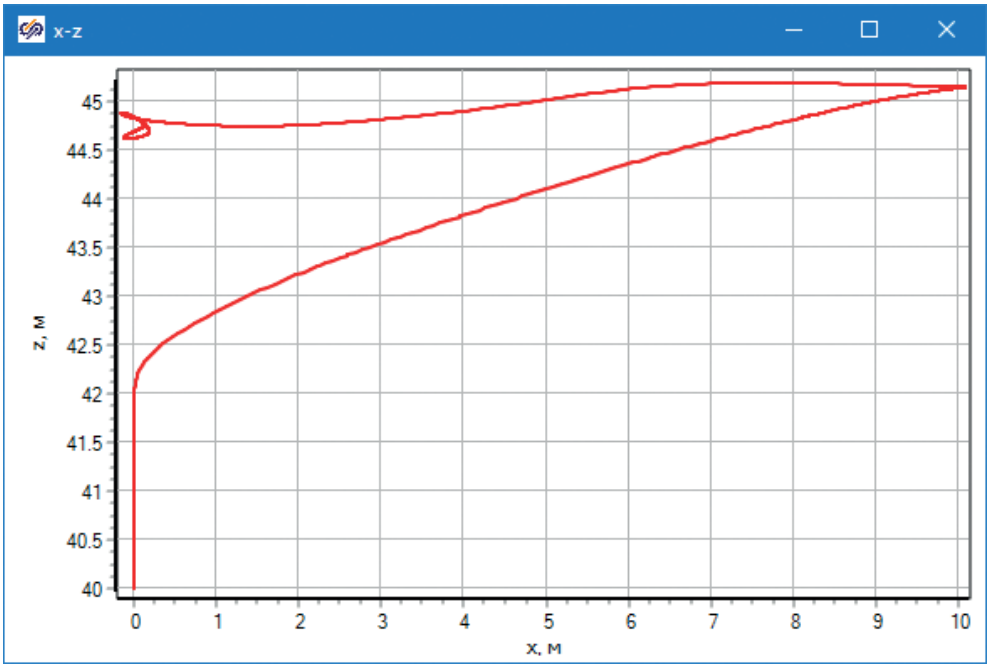


Рис. 6.4.2. Тестовый полет 2, траектория в осях x–z, м

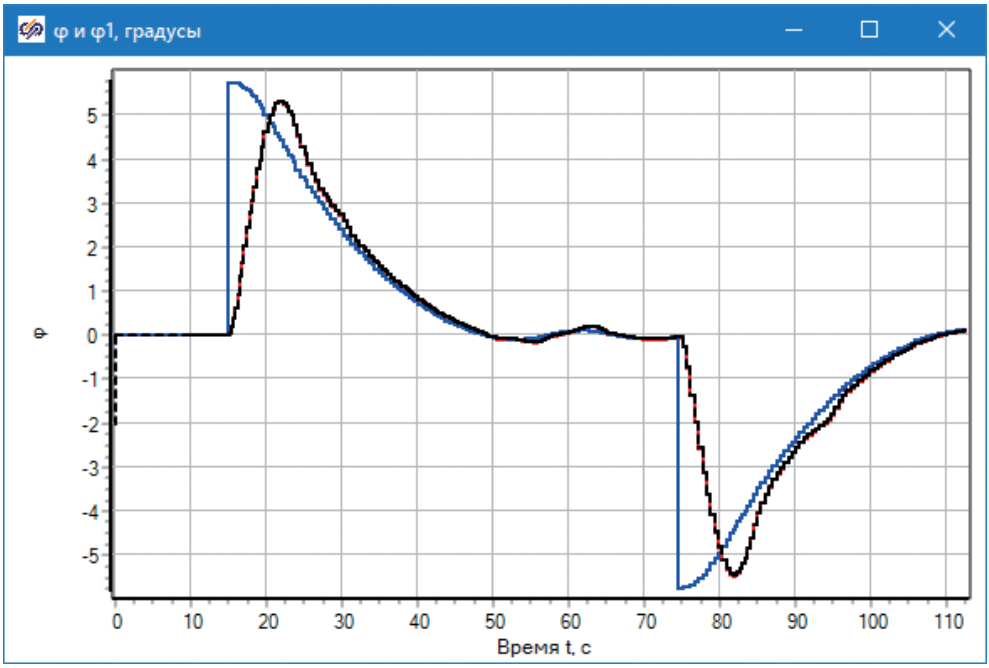


Рис. 6.4.3. Тестовый полет 2, крен (заданный и измеренный)

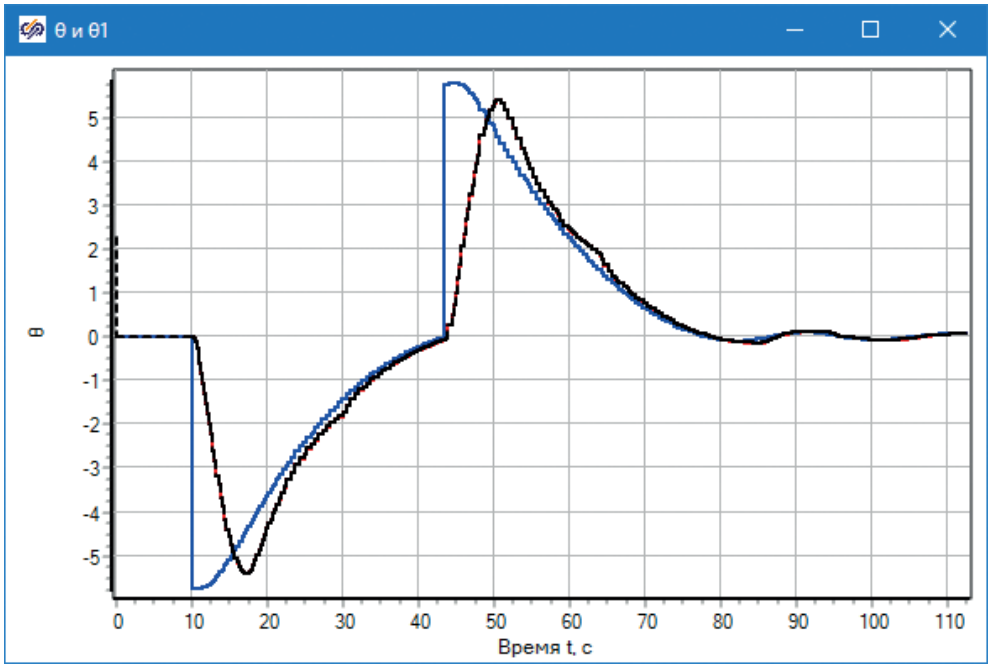


Рис. 6.4.4. Тестовый полет 2, тангаж (заданный и измеренный)

Рис. 6.4.5. Тестовый полет 2, угловая скорость двигателей

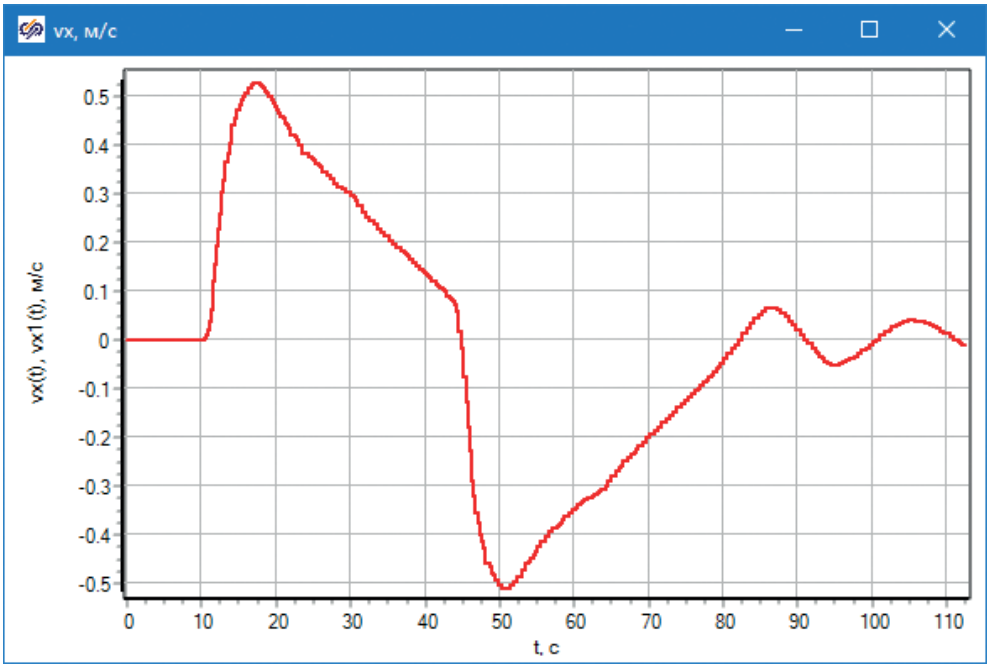


Рис. 6.4.6. Тестовый полет 2, скорость коптера вдоль оси x_l

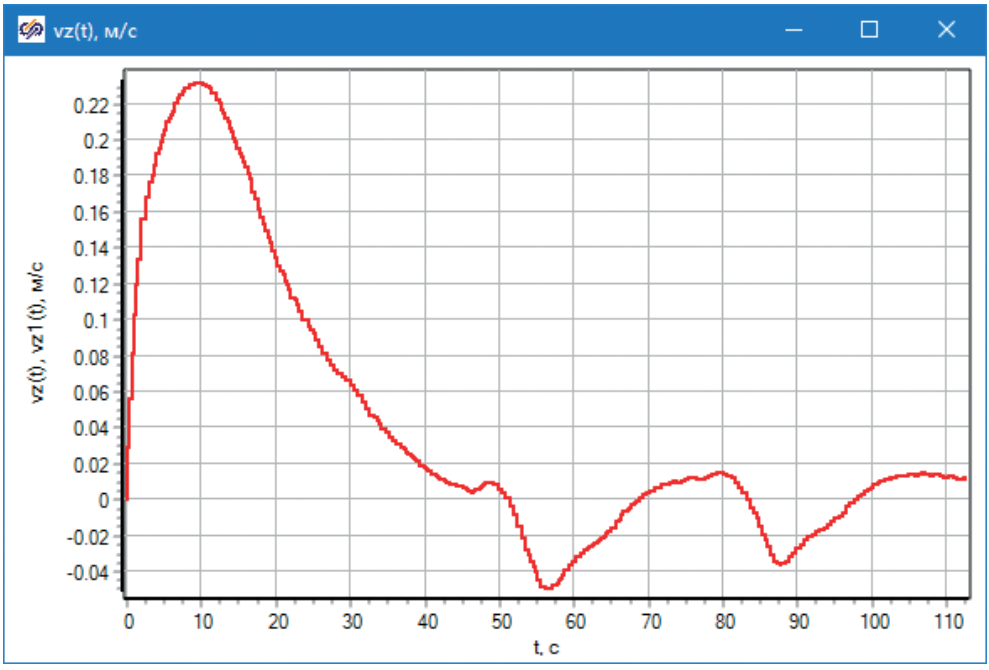


Рис. 6.4.7. Тестовый полет 2, вертикальная скорость коптера (по оси z)

6.5. Тестовый полет 3

Горизонтальный полет на относительно большое расстояние. Исходное состояние аналогично тестовым полетам 1 и 2: координаты 0, 0, 40 м. Задатчик координаты x выставляется на +100 м, и происходит полет в одном направлении.

В этом тесте был смоделирован полет на 100 м по горизонтали в направлении оси x. Так как ограничения по наклону коптера в модели выставлены на уровне около 11°, регулятор не может развить скорость, большую, чем позволяет данный угол наклона и совместная работа сил тяги двигателей и силы сопротивления воздуха. С другой стороны, в регуляторе высоты не предусмотрена поправка на наклон коптера – регулятор высоты спроектирован из предположения горизонтальности коптера, что (предположительно) должно дать сильную просадку по высоте в процессе полета. На рисунках ниже (6.5.1…6.5.5) представлены некоторые графики этого тестового полета.

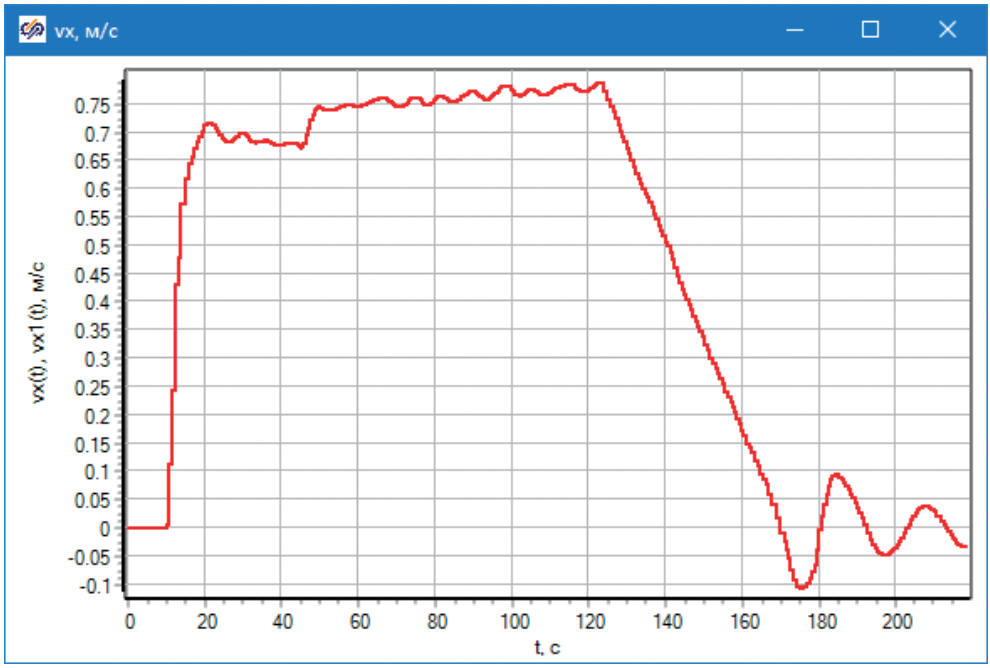


Рис. 6.5.1. Тестовый полет 3, горизонтальная скорость

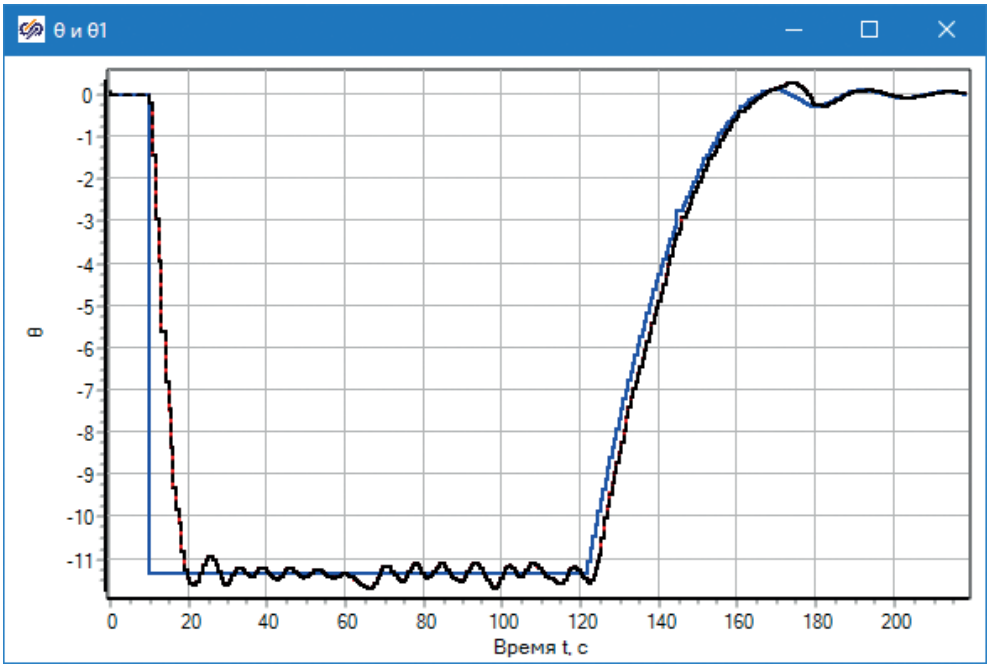


Рис. 6.5.2. Тестовый полет 3, тангаж

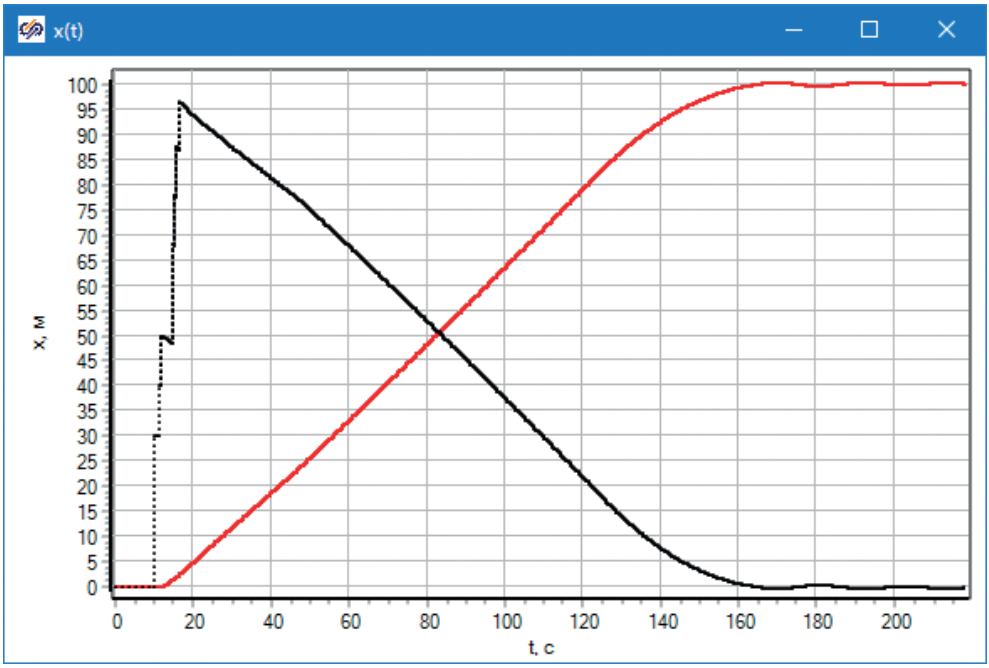


Рис. 6.5.3. Тестовый полет 3, координата x и рассогласование между заданным положением и текущим

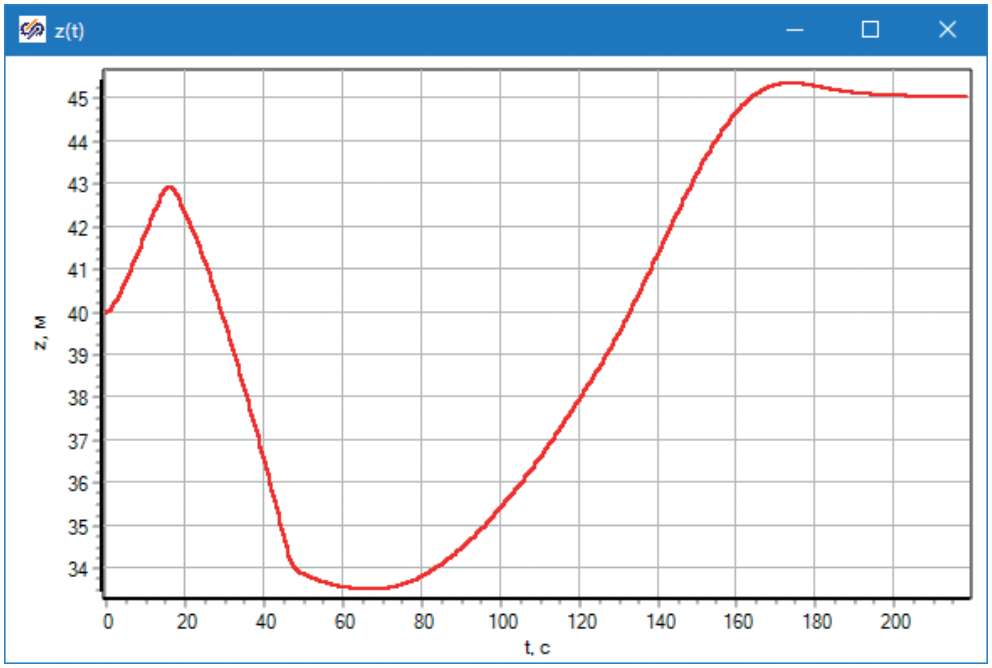


Рис. 6.5.4. Тестовый полет 3, координата z

Рис. 6.5.5. Тестовый полет 3, работа двигателей

Из графиков видно, что полет происходит примерно с постоянной скоростью, а когда просадка по высоте превысила 10 м, двигатели перешли на чуть больший базовый режим работы (хотя регулятор высоты в этом варианте модели не следит за наклоном коптера), что привело к его постепенному подъему и возврату на высоту 45 м, а также к небольшому увеличению горизонтальной установившейся скорости полета с 0.7 до 0.75 м/с.

В целом приведенные тестовые моделирования показывают необходимость дальнейшей доработки регуляторов и донастройки их на параметры конкретного коптера, так как текущее качество переходных процессов можно назвать удовлетворительным, но не хорошим.

6.6. Выводы по разделу

Среда SimInTech позволяет просто, в едином интерфейсе и достаточно эффективно создавать как модели динамики технических объектов (например, октокоптера), так и виртуальный аналог его системы управления, включая алгоритмическую часть, модель исполнительных механизмов и датчиков, панель управления и элементы анимации. Имея перед собой реализованную модель и модель системы управления, можно удобно анализировать совместное поведение регуляторов и объекта, уточнять модель и проектировать регуляторы, достигая требуемого качества управления и регулирования.

Регуляторы, представленные в настоящей методике, специально не сделаны «хорошими» (хотя они и обеспечивают приемлемое качество регулирования) чтобы не раскрывать некоторые ноу-хау. Отметим, что работа, часть которой представлена в методике, была выполнена с трудозатратами всего около 2 чел./мес., практически с нуля без сформированного ранее задела, хотя и с помощью заимствований из литературы и других статей. Во многом такие результаты достигнуты за счет применения SimInTech.

Достигнутый результат, а именно структурированная модель объекта и модель его системы управления позволяют заниматься дальнейшей разработкой и уточнением как модели объекта, так и системы управления, верифицировать модель на экспериментальных данных и проводить в дальнейшем подбор коэффициентов регуляторов для достижения оптимального качества регулирования. Модель системы управления и генератор кода SimInTech могут служить основой для разработки прошивки полетного контроллера, работающего без операционной системы или с операционной системой реального времени типа КПДА (QNX).

На базе полученной модели динамики при добавлении в нее более подробной модели электродвигателя и аккумуляторной системы можно получить расчетную модель для вычисления максимального времени полета и энергозатрат при варьировании сценария полета. В следующих разделах будет описано пошаговое создание подобной модели с нуля при помощи стандартной библиотеки блоков и графических примитивов.

Практика: набор модели в виде схемы SimInTech

Начиная с данного раздела, мы будем пошагово реализовывать модель, изложенную теоретически и в уже «готовом», законченном виде выше.

7.1. Уравнения динамики для реализации в схеме

Полученная система уравнений в виде (4.1.1), в принципе, подходит для ее реализации на схеме.

Если эту систему уравнений решить (численно проинтегрировать), то в результате решения мы получим зависимости линейной и угловой скоростей от времени **в системе координат B**. Для получения траектории движения объекта (его фазовой траектории в пространстве 6 степеней свободы) необходимо еще раз проинтегрировать полученные скорости и получить координаты октокоптера в пространстве – три линейные и три угловые координаты. Поскольку в конечном счете координаты нас интересуют в инерциальной системе отсчета **I**, связанной с Землей, необходимо над решением этой системы, над полученными векторами линейной и угловой скорости в системе B, выполнить еще дополнительно ряд алгебраических преобразований, чтобы получить эти же векторы в инерциальной системе отсчета **I** и уже в ней интегрировать скорости второй раз.

В полученной системе уравнений (4.1.1), кроме факта ее нелинейности, видно, что в правых частях стоят проекции сил и моментов, действующих на коптер. В упрощенной модели октокоптера из всех сил и моментов мы оставили только три вида – силы тяги ВМГ и опрокидывающий момент от них же; силу и момент сопротивления воздуха; возмущающую силу и момент, которые для отладки модели можно будет задавать произвольно в процессе моделирования.

Примечание: учет дополнительных сил и моментов (от явления прецессии, от ветровой нагрузки, ground-эффект, реактивный момент ВМГ, зависимость силы тяги ВМГ от изгиба лопастей или от набегающего ветра и расположения вращающегося винта относительно ветра, нагрузка коптера полезным грузом, несимметричность конструкции и т.д. и т. п.) приведет к количественному росту системы уравнений, к увеличению числа слагаемых в правых частях уравнений, но качественно система уравнений останется той же. В целях сокращения излагаемого материала мы ограничились минимальным набором сил и моментов, достаточным для целей обучения SimInTech и качественного понимания моделирования октокоптера. Двигатели будут создавать тягу и поворотные моменты, воздух будет сопротивляться скорости полета и «демпфи-

ровать» движение, внешние возмущения в моменты их подачи будут выводить октокоптер из равновесия.

Опуская выкладки (их можно осуществить либо вручную, либо в каком-то математическом ПО символьных вычислений), проекции сил и моментов на оси подвижной системы координат **B** для геометрии октокоптера с симметрично расположенными винтами будут равны (4.2.1…4.2.15). Выпишем уравнения для сил и моментов от двигателей (индекс M), воздуха (индекс D) и внешнего возмущения (индекс O) еще раз, с некоторым упрощениями. Например, $\sin(\gamma)$ и $\cos(\gamma)$ можно вынести за скобки и т. п.:

$$\begin{cases} F_{Mx}(t) = \sin(\gamma)\cdot\left(\dfrac{F_{M8}+F_{M6}-F_{M4}-F_{M2}}{\sqrt{2}}+F_{M3}-F_{M7}\right), \\ F_{My}(t) = \sin(\gamma)\cdot\left(\dfrac{F_{M8}-F_{M6}-F_{M4}+F_{M2}}{\sqrt{2}}+F_{M5}-F_{M1}\right), \\ F_{Mz}(t) = -\cos(\gamma)\cdot\left(F_{M1}+F_{M2}+F_{M3}+F_{M4}+F_{M5}+F_{M6}+F_{M7}+F_{M8}\right), \\ F_{Dx}(t) = -0.5\rho\cdot C_D\cdot A_{yz}\cdot v_{Bx}\cdot\left|v_{Bx}\right|, \\ F_{Dy}(t) = -0.5\rho\cdot C_D\cdot A_{xz}\cdot v_{By}\cdot\left|v_{By}\right|, \\ F_{Dz}(t) = -0.5\rho\cdot C_D\cdot A_{xy}\cdot v_{Bz}\cdot\left|v_{Bz}\right|, \\ F_{Ox}(t) = var, \\ F_{Oy}(t) = var, \\ F_{Oz}(t) = var. \end{cases} \tag{7.1.1}$$

$$\begin{cases} M_{Mx}(t) = \cos(\gamma)\cdot\left(l_2\cdot\dfrac{F_{M8}+F_{M6}-F_{M4}-F_{M2}}{\sqrt{2}}+l_1\cdot\left(F_{M7}-F_{M3}\right)\right), \\ M_{My}(t) = \cos(\gamma)\cdot\left(l_2\cdot\dfrac{F_{M8}-F_{M6}-F_{M4}+F_{M2}}{\sqrt{2}}+l_1\cdot\left(F_{M1}-F_{M5}\right)\right), \\ M_{Mz}(t) = \sin(\gamma)\cdot\begin{pmatrix}\left(F_{M8}+F_{M6}+F_{M4}+F_{M2}\right)\cdot l_2 \\ -\left(F_{M7}+F_{M5}+F_{M3}+F_{M1}\right)\cdot l_1\end{pmatrix}, \\ M_{Dx}(t) = -0.5\rho\cdot C_D\cdot A_{yz}\cdot\omega_{Bx}\cdot\left|\omega_{Bx}\right|\cdot l_x, \\ M_{Dy}(t) = -0.5\rho\cdot C_D\cdot A_{xy}\cdot\omega_{By}\cdot\left|\omega_{By}\right|\cdot l_y, \\ M_{Dz}(t) = -0.5\rho\cdot C_D\cdot 8A_{xy}\cdot\omega_{Bz}\cdot\left|\omega_{Bz}\right|\cdot l_z, \\ F_{Ox}(t) = var, \\ F_{Oy}(t) = var, \\ F_{Oz}(t) = var, \end{cases} \tag{7.1.2}$$

где F_{Mi} – силы тяги [Н] каждой ВМГ, i = 1…8;
l_1 – длина луча [м] для ВМГ №1, 3, 5, 7;
l_2 – длина луча [м] для ВМГ №2, 4, 6, 8;

γ –малый угол [~3°] поворота каждой ВМГ вокруг своего луча, для создания управляемости октокоптера по курсу;
ρ – плотность воздуха [кг/м^3], можно принять равной 1.2 кг/м^3;
C_D – коэффициент формы для вычисления сопротивления воздуха (для формы куба он равен 1) [безразмерная величина];
A_{xy} (и другие площади) – характерные «опорные» площади [м^2] в соответствующих плоскостях, для вычисления сил и моментов сопротивления воздуха. Коэффициент 8 в вычислении момента $M_{Dz}(t)$ стоит для учета 8 лучей, каждый из которых испытывает сопротивление воздуха при вращении коптера вокруг вертикальной оси z_B. В нулевом приближении можно считать октокоптер кубом, тогда $l_x = l_y = l_z$, а площади $A_{xy} = A_{xz} = A_{zy} = l_x^2$.

Напомним, что ВМГ немного повернуты относительно лучей на малый угол γ ~3°, причем поочередно – четные ВМГ в одну сторону, нечетные в другую, а $l_1 > l_2$ в принятой конструкции.

Возмущающие силы и моменты – вообще говоря, нулевые и произвольно могут меняться в процессе расчета (заранее неизвестны и не зависят от состояния коптера и его фазовых переменных).

Примечание: выписанные правые части уравнений – еще не полные, так как углы ориентации коптера, входящие в эту систему, в общем случае зависят от угловых скоростей коптера и являются тоже переменными состояния. Но пока что для начала набора схемы хватит и этих 6 уравнений.

7.2. Набор уравнений в схеме общего вида

Как только мы получили систему дифференциальных уравнений в форме (4.1.1) и определили правые части до последнего слагаемого, можно приступать к набору этих уравнений в SimInTech. Без уравнений, выписанных в четком и понятном виде на листе бумаги (или на экране компьютера), создавать расчетную схему «из головы» не рекомендуется. По мере набора схемы нам будут требоваться те или иные константы и переменные – мы будем добавлять их по мере необходимости.

Но сначала организуем рабочую папку проекта. Она может быть в любом месте компьютера, для удобства описания будем считать, что модель будем создавать на диске **C**, и она всегда будет находиться в папке **c:\copter**. Внутри этой папки создайте еще четыре папки, а именно: **control**, **copter**, **db** и **sub**, как показано на рис. 7.2.1. Имена папок непринципиальны, но дальнейшее описание будет опираться на эти имена.

В папке **control** будет содержаться макет пульта управления, а позже – модель контроллера, в папке **copter** – математическая модель объекта, в **db** – база данных сигналов, в **sub** – типовые подпрограммы (в нашем случае она будет одна – типовая подпрограмма ВМГ, которая будет 8 раз использована в основной модели).

Далее, запустите SimInTech и в главном меню выберите пункт **Файл → Новый проект → Схема модели общего вида**. Должно появиться схемное окно, в котором мы и будем набирать расчетную схему. Файл проекта можно сразу же сохранить на диск, дав ему имя, например **c:\copter\copter\octocopter.prt**.

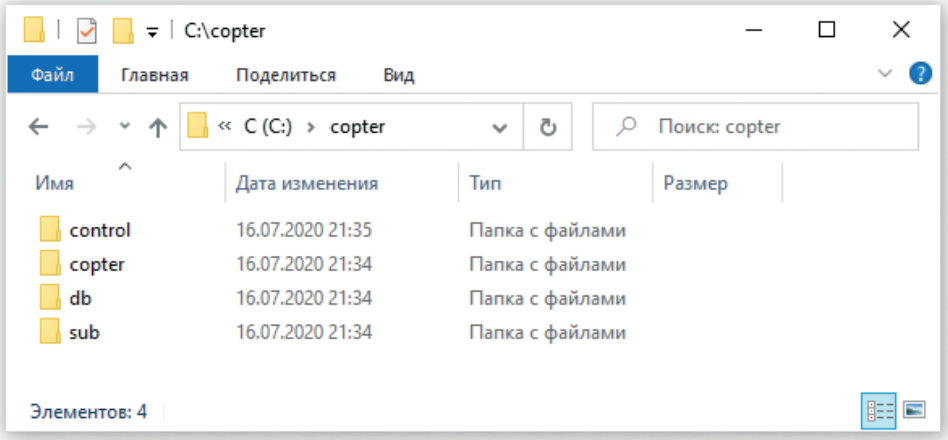


Рис. 7.2.1. Структура проекта

Для этого надо выбрать пункт меню **Файл → Сохранить проект как...** и в появившемся типовом окне Windows перейти в нужную папку и набрать там имя файла **octocopter.prt**, как показано на рис. 7.2.2.

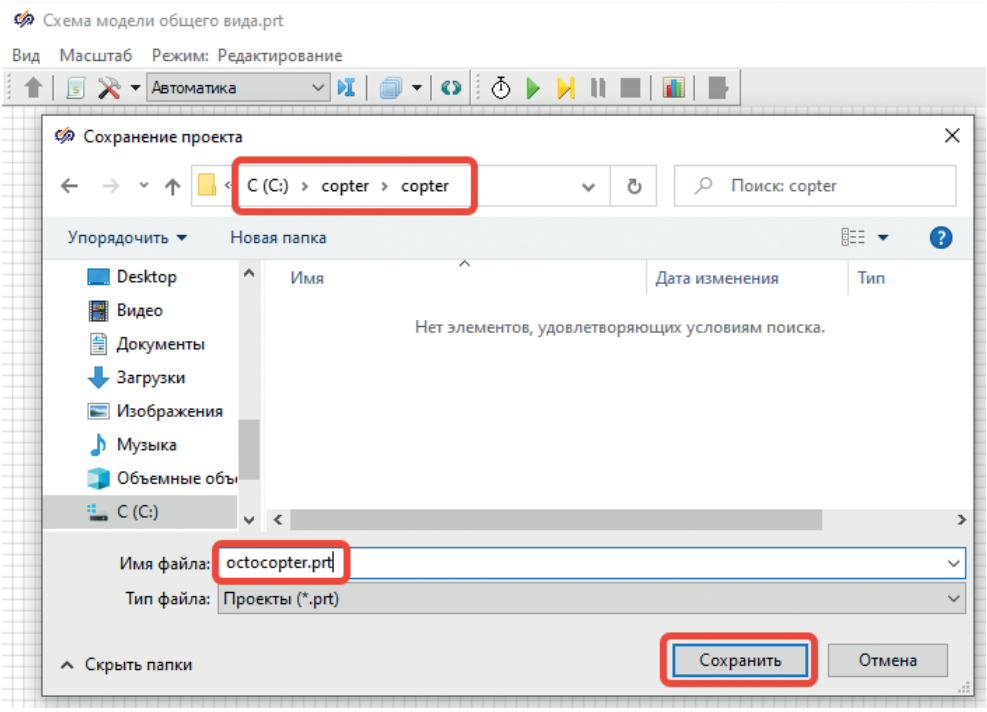


Рис. 7.2.2. Сохранение нового файла проекта

После этого в схемном окне в заголовке вы должны увидеть имя **octocopter.prt** – значит, файл сохранился корректно. Необходимость разнесения разных проектов в свои папки обусловлена только удобством. Дело в том, что SimInTech при каждом сохранении делает также резервную копию файла в той же директории, и если в одной директории одновременно создавать несколько файлов, то через время там получится чрезмерное количество файлов, которые придется периодически чистить, и в целом будет работать менее удобно. Кроме этого, для подключения базы данных мы будем использовать относительные пути, и для этого тоже желательно, чтобы каждый проект был в своей директории. Кроме того, около каждого файла проекта могут еще сохраняться служебные файлы для рестартов (исходных состояний), файлы менеджера данных, файлы индексатора проекта – все это удобнее делать, когда на 1 папку приходится 1 файл проекта.

Итак, мы создали файл модели типа «схема общего вида», в котором будем набирать расчетную модель динамики октокоптера. Проанализировав полученную выше систему дифференциальных уравнений, легко видеть, что мы имеем дело с 6 переменными состояния, которые суть скорости коптера (три линейные и три угловые) в связанной системе координат **B**, а именно: $v_{Bx}(t), v_{By}(t), v_{Bz}(t), \omega_{Bx}(t), \omega_{By}(t), \omega_{Bz}(t)$. Поскольку язык программирования и система именования сигналов и переменных в SimInTech не имеет нижних индексов (для совместимости с языком Си в кодогенераторе), но зато поддерживается юникод для сигналов типа «В память» и «Из памяти», на расчетной схеме будем обозначать эти скорости как v_B_x, v_B_y, v_B_z, ω_B_x, ω_B_y, ω_B_z. И **именно эти переменные будут являться основой** модели динамики.

7.3. Интегратор как способ записи дифференциального уравнения 1 порядка

Каждое из 6 уравнений системы по структуре приблизительно одинаково – в правой части набор слагаемых, часть из которых явно содержит переменные состояния. Чтобы на расчетной схеме не тянуть линии связи и не создавать лишнюю «паутину», для каждой из переменных состояния заведем «бирку», при помощи блока «В память» и при помощи ответного блока «Из памяти» будем использовать значение линии связи по мере необходимости в других частях расчетной схемы. Основной блок, при помощи которого будет решаться каждое дифференциальное уравнение, будет блок типа «Интегратор», который расположен на вкладке **Динамические**, см. рис. 7.3.1:

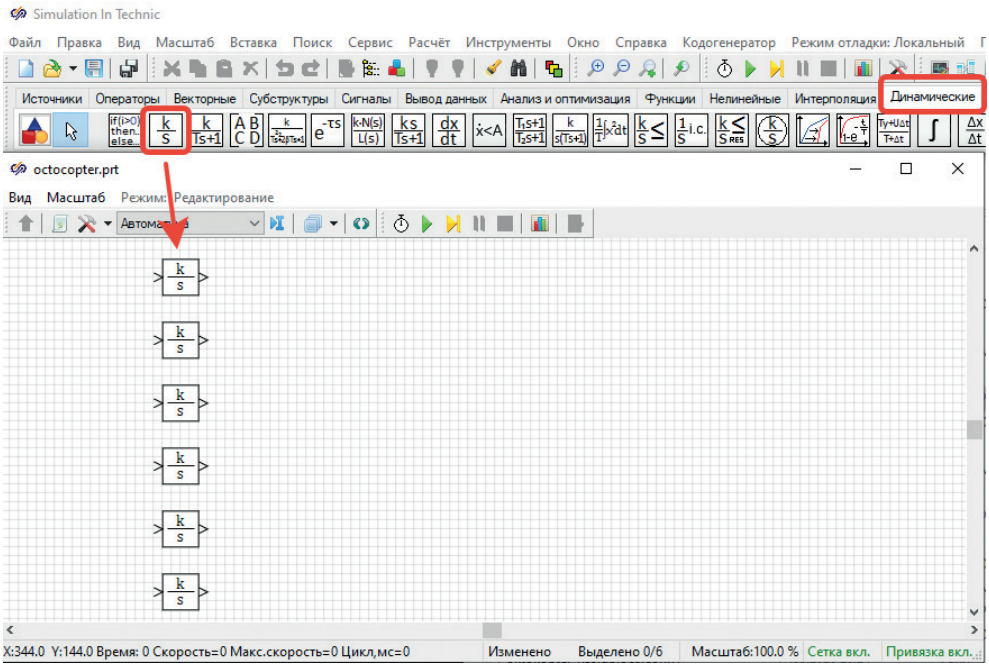


Рис. 7.3.1. Шесть блоков типа «Интегратор»

Разместите на схеме 6 таких блоков (для этого надо выбрать в палитре блоков вкладку **Динамические**, там выбрать второй блок и, перенеся курсор мыши на схемное окно, расположить на нем блок. Действие повторить еще 5 раз).

Этот блок работает следующим образом (при коэффициенте усиления k = 1): на вход подается какая-либо величина, на выходе формируется интеграл этой величины. Соответственно, если на вход подавать производную чего-либо, на выходе будет сама величина (производную которой подали на вход). Мы и будем так использовать блок – перед входом каждого из блоков будем формировать сумму слагаемых и подавать на вход интегратора правую часть каждого из уравнений системы, а на выходе из блока получать дифференциальную переменную (переменную состояния октокоптера). Другими словами, на входы блоков будем подавать ускорения октокоптера, на выходах получать скорости – все просто!

7.4. Блоки «В память»

Разместите справа от каждого интегратора блок типа «В память» из вкладки **Субструктуры** (рис. 7.4.1) и задайте каждому из блоков имя v_B_x, v_B_y, v_B_z, ω_B_x, ω_B_y, ω_B_z – как показано ниже на рис. 7.4.2 и 7.4.3.

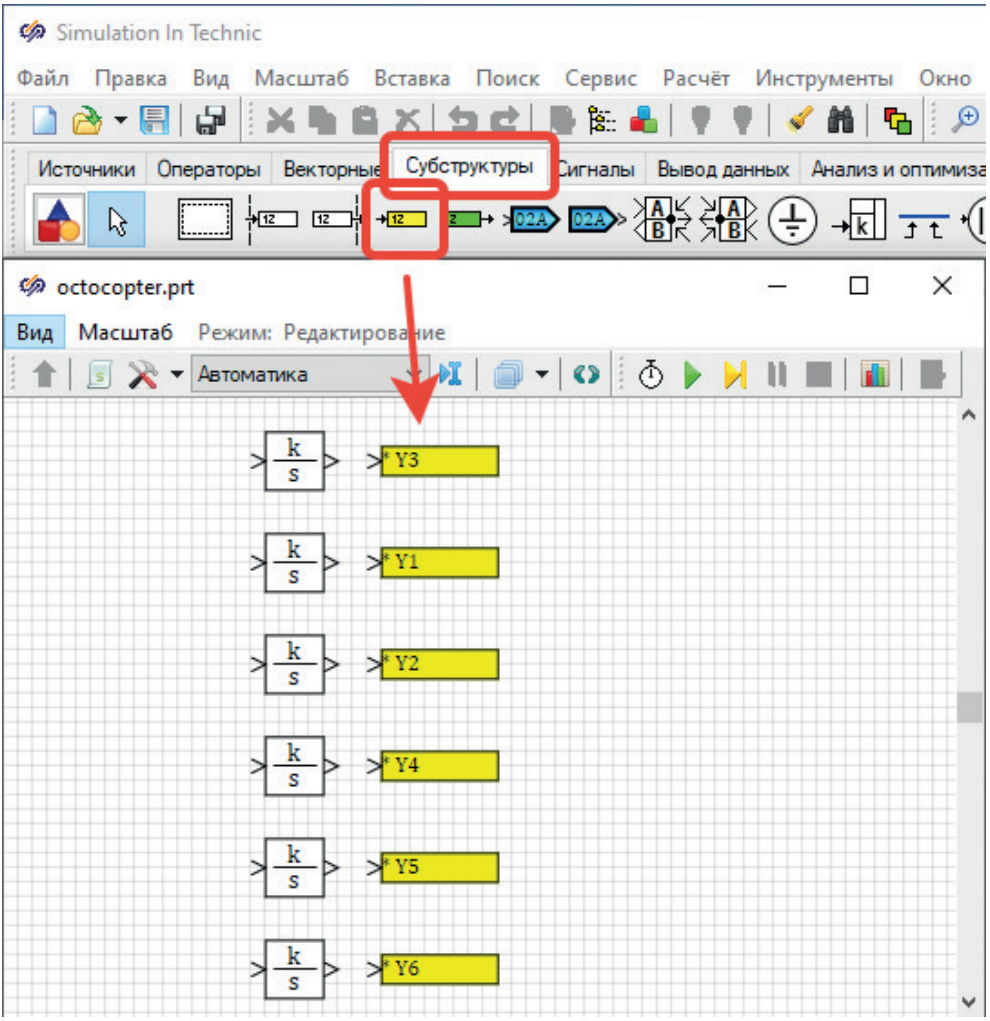


Рис. 7.4.1. Первичное размещение блоков типа «В память»

При первом размещении у блоков будут имена типа Y1, Y2 и т. д. Для задания своего имени надо щелкнуть двойным щелчком по каждому блоку, там ввести требуемое имя переменной, затем нажать на синюю стрелку вправо для добавления нового имени переменной в список переменных блока и убрать имя по умолчанию – выделив его и нажав синюю стрелку влево, затем нажать кнопку **Ок** (рис. 7.4.2). После чего имя переменной на схеме должно поменяться на нужное.

На каждом из блоков будет проставлена звездочка перед именем переменной, означающая, что эта переменная (линия связи) еще нигде не используется в проекте – это нормально, по мере набора схемы звездочки уйдут.

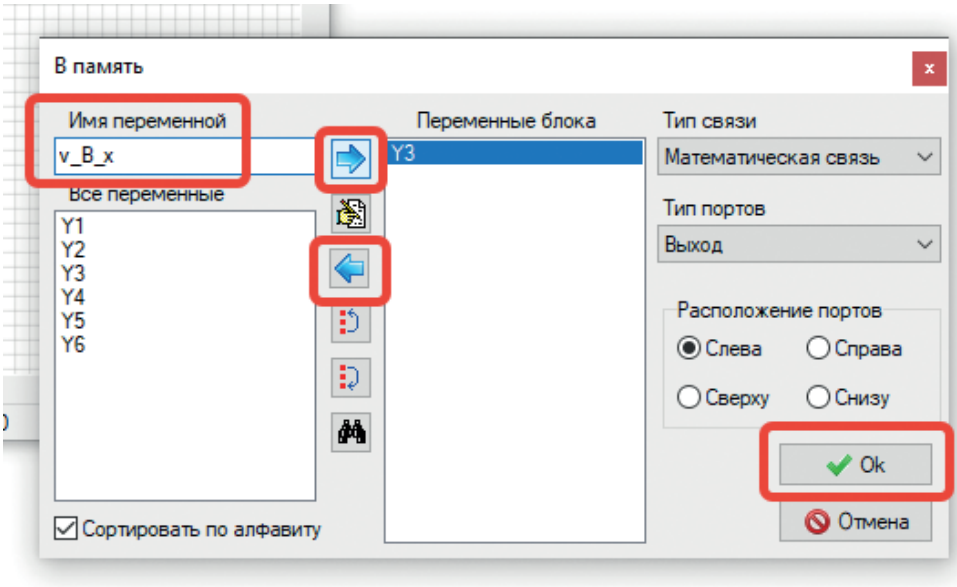


Рис. 7.4.2. Задание осмысленных имен в блоках типа «В память»

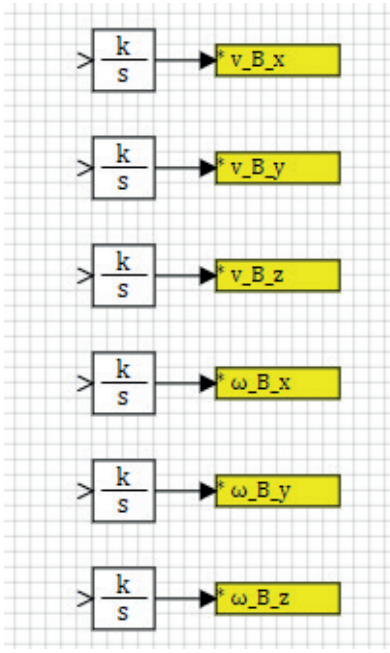


Рис. 7.4.3. Все имена заданы

Соедините каждый из интеграторов с блоком типа «В память» линией связи (рис. 7.4.3).

Примечание: блоки «В память» не формируют каких-либо новых сигналов или сигналов базы данных в проекте – это «внутренние» метки расчетной схемы, по сути, это имена или бирки для линий связи, которые могут быть использованы только блоками «Из памяти». Блоки могут работать в векторном режиме, в нашем случае каждый блок соответствует одной линии связи со скалярной величиной.

7.5. Блоки «Из памяти»

Если посмотреть на правые части уравнений системы, можно увидеть там нелинейные их части в виде произведений скоростей (например, для первого уравнения это будут два слагаемых $v_{By}(t)\cdot\omega_{Bz}(t)-v_{Bz}(t)\cdot\omega_{By}(t)$). Эти слагаемые уже сейчас можно набрать на расчетной схеме при помощи блоков типа «Из памяти» и типовых блоков «Перемножитель» и «Сумматор», см. рис. 7.5.1:

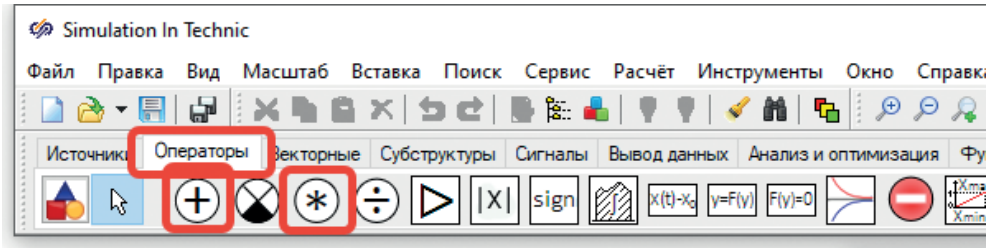


Рис. 7.5.1. Сумматор и перемножитель

Блок «Из памяти» находится рядом с блоком «В память» и имеет зеленый цвет. Наберите при помощи этих блоков схему для первого уравнения, похожую на рис. 7.5.4.

Последовательность действий:

- разместить на схеме два блока типа «Из памяти»;
- задать каждому из них по два имени переменных (**v_B_y** и ω_**B_z** для первого блока, **v_B_z** и ω_**B_y** для второго), как показано на рис. 7.5.2;
- разместить два «Перемножителя» напротив каждого из блоков «Из памяти»;
- разместить один «Сумматор» и задать ему вектор коэффициентов **[1, 1, 1, -1]** (показано на рис. 7.5.3), при этом у блока появится 4 входных порта;
- соединить линиями связи размещенные блоки.

Результат сравнить с рис. 7.5.4.

Что можно увидеть здесь нового – на 4 блоках типа «В память» звездочка исчезла. Это произошло из-за того, что мы задействовали эти 4 сигнала в расчетной схеме. Количество использований сигнала – неограниченное, а определен он, конечно, должен быть только в одном месте расчетной схемы. Идентичный результат мы бы получили и в варианте, когда протянули бы 4 линии связи от выхода соответствующего интегратора на нужный вход перемножителей.

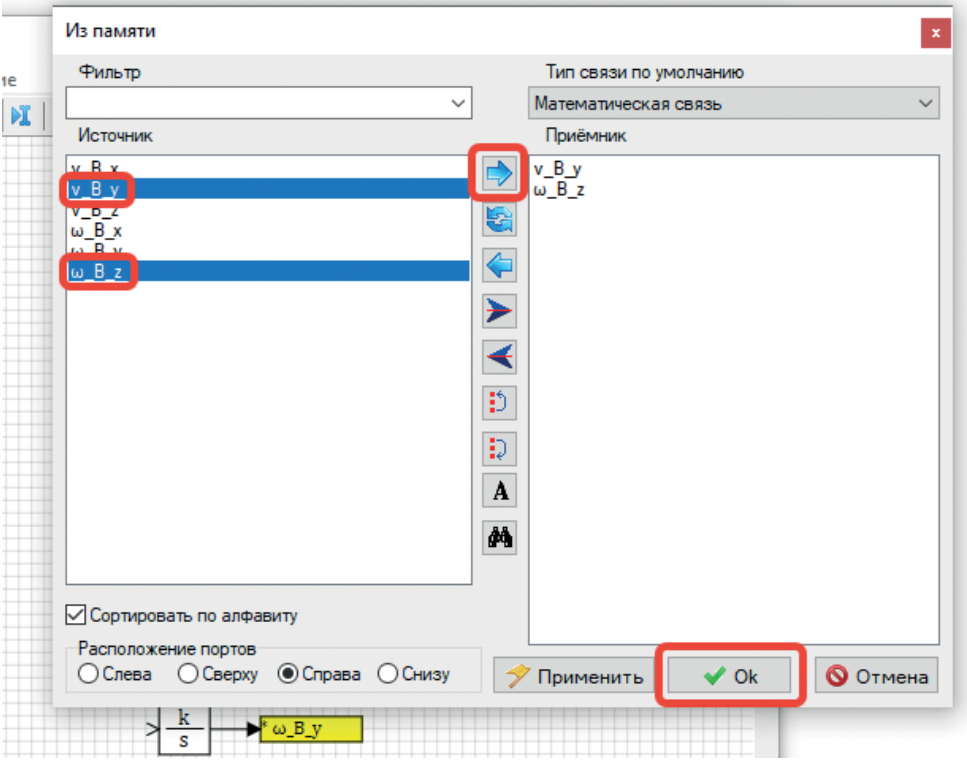


Рис. 7.5.2. Назначение переменных у блока «Из памяти»

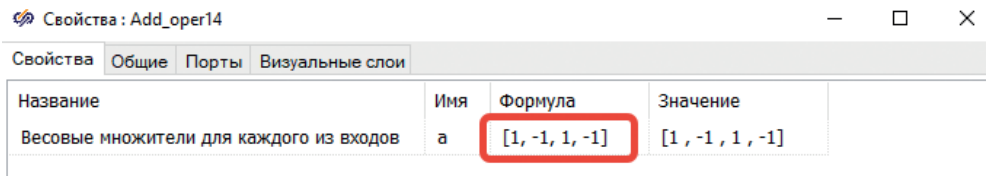


Рис. 7.5.3. Коэффициенты сумматора

В некоторых случаях это оправдано, но в нашем случае – схема была бы путаной, поэтому для наглядности использованы блоки типа «В память» и «Из памяти», а в дальнейшем, когда модель будет наращиваться и мы разнесем ее на разные листы (субструктуры), без этих блоков будет уже не обойтись. При этом схема остается связной математически, хотя и будет расположена на разных листах.

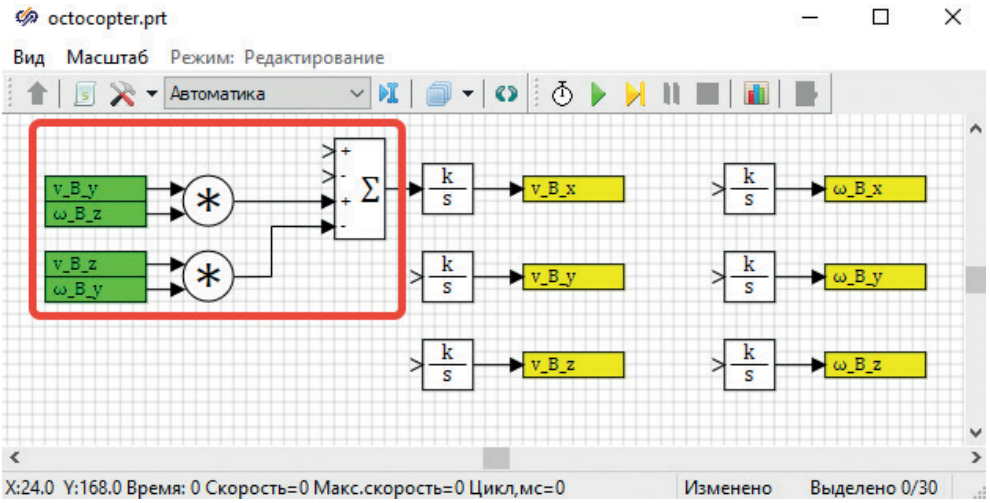


Рис. 7.5.4. Нелинейная часть первого уравнения системы

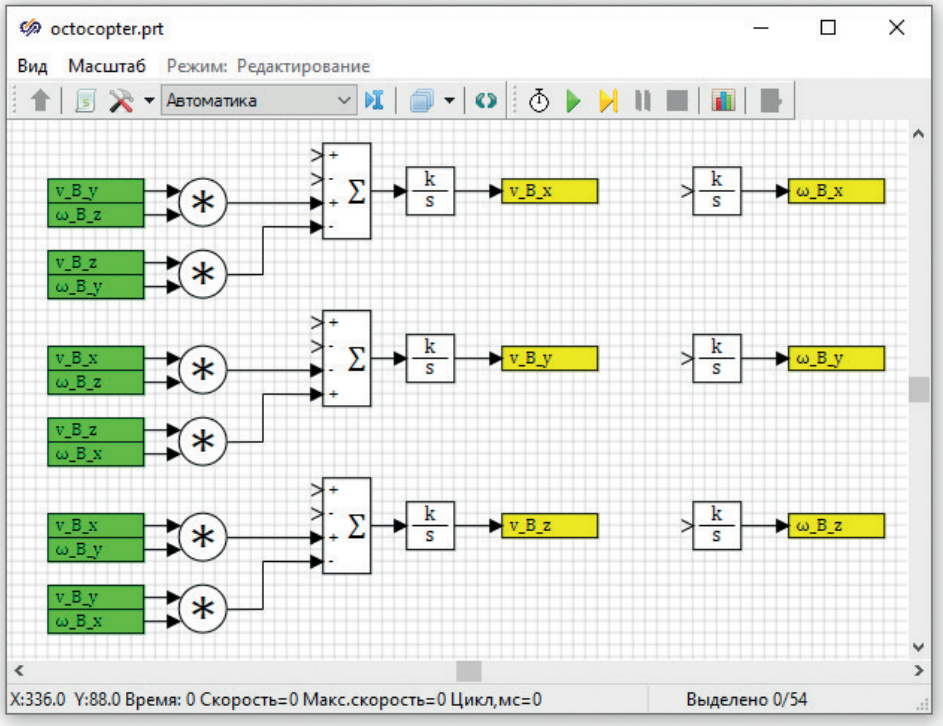


Рис. 7.5.5. Нелинейные части первых трех уравнений системы

Сумматору мы задали такие коэффициенты из-за того, что первое слагаемое и третье идет со знаком плюс (в первом уравнении), а второе и четвертое – со знаком минус (см. выше исходную систему дифференциальных уравнений).

Первое и второе слагаемые сделаем чуть позже, пока что по аналогии сделайте самостоятельно второе и третье уравнения, а результат сравните с рис. 7.5.5.

Дальше есть некоторая «сложность» – для набора следующих слагаемых нам потребуются константы и переменные проекта, а именно – масса коптера, ускорение свободного падения **g**, составляющие тензора инерции коптера **Ixx**, **Iyy**, **Izz** и некоторые другие. Для корректного и удобного задания этих и других величин следует использовать сигналы проекта и/или сигналы базы данных проекта, а лучше – сочетание этих инструментов.

7.6. Сигналы проекта

Некоторые из сигналов проекта имеет смысл задать при помощи простого списка – локальных сигналов данного prt-файла (те переменные, или константы, которые будут использованы только в текущей схеме). В нашем случае это могут быть такие константы, как угол поворота ВМГ $\gamma = 3°$, ускорение свободного падения и некоторые другие. Те же сигналы, которые могут быть типизированы для того или иного объекта модели, или предполагается их передавать в/из системы управления или с/на пульт управления, лучше задавать сразу через базу сигналов. В базе сигналов можно делать «классы» объектов и потом задавать несколько экземпляров данного класса – так мы поступим с сигналами для ВМГ, так как они одинаковым образом должны быть математически описаны (с точностью до имен сигналов).

Задайте 7 констант (сигналов) через пункт главного меню **Сервис → Сигналы**, нажав там 7 раз на кнопку с зеленым плюсиком и аккуратно после этого заполнив все поля, как представлено на рис. 7.6.1:

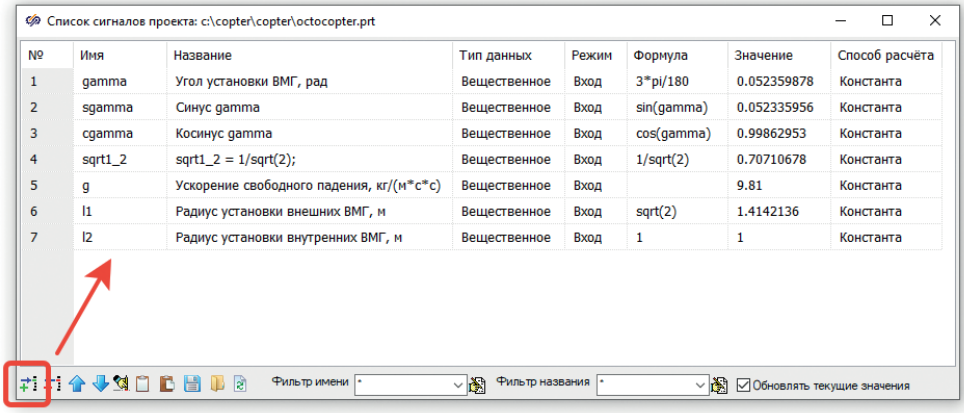


Рис.7.6.1. Сигналы проекта

По рисунку должно быть все понятно, поясним только некоторые моменты.

В колонке **Имя** задается уникальное имя переменной, должно быть на английском языке и уникально в пределах всего проекта. Этот перечень не пересекается с именами блоков «В память» и «Из памяти», но лучше во избежание путаницы в дальнейшем не создавать одинаковых сигналов вообще. Мы создаем 7 сигналов с именами: **gamma**, **sgamma**, **cgamma**, **sqrt1_2**, **g**, **l1**, **l2**.

В колонке **Название** допустимо вписывать любой текст – это описание сигнала для пользователя модели, в расчете не используется.

В колонке **Тип данных** задается тип переменной (или константы). В нашем случае все сигналы – вещественные числа.

Колонка **Режим** в настоящей версии SimInTech не используется (устаревшее).

Колонка **Формула** может быть заполнена интерпретируемым выражением, согласно синтаксису встроенного в SimInTech языка программирования, и тогда это выражение будет интерпретироваться по правилу, заданному в колонке **Способ расчета**.

Колонка **Значение** отображает текущее значение сигнала в соответствии с типом данных. Здесь мы видим начальные значения сигналов. Так как все создаваемые сигналы – константы, это будет текущим значением сигналов на каждом шаге расчета.

Колонка **Способ расчета** определяет способ расчета формулы (или интерпретируемого выражения), если оно задано в колонке **Формула**. Способ «переменная» означает, что формула будет интерпретироваться на каждом шаге расчета, «константа» означает, что формула будет интерпретирована при инициализации проекта на расчет (1 раз). «Заблокирована» означает, что формула не будет интерпретироваться вообще (никогда). Нам подходит вариант «константа», так как все 7 сигналов должны быть посчитаны перед расчетом, а дальше они не изменяют своего значения.

Под внешними ВМГ подразумеваются те винтомоторные группы, которые расположены на нечетных лучах (номера 1, 3, 5, 7), под внутренними – те, которые расположены на четных лучах октокоптера, см. рис. второго раздела 2.1.1. Обратите внимание также на этом рисунке на направление осей связанной системы координат **В**.

7.7. База сигналов

Некоторые из требуемых переменных и констант – хотя их можно было бы создать здесь же в перечне сигналов проекта – лучше завести сразу через базу сигналов, подключаемую к проекту. База сигналов в SimInTech – это файловая БД с некоторыми специализированными опциями для моделирования и расчетов. Подробнее можно ознакомиться в справочной системе, здесь опишем только 4 цели, с которыми мы будем использовать базу сигналов.

1. Описание октокоптера как объекта сделаем в отдельной категории базы данных, так как в будущем, возможно, потребуется сделать модель с 2, 3 или N октокоптерами, моделируемыми одновременно, – тогда объектно-ориентированное описание придется весьма кстати.
2. Описание ВМГ выделим в отдельную группу сигналов.
3. Описание настроечных коэффициентов для регуляторов тоже вынесем в отдельную категорию для удобства настройки в будущем регуляторов.
4. База сигналов потребуется для пакетного режима расчета, когда модель коптера, модель регуляторов и пульт управления должны моделироваться совместно, с синхронизацией модельного времени и обмена сигналами на каждом шаге расчета между моделями пакета. Синхронизацию и обмен данными обеспечивает в том числе модуль базы сигналов.

Для того чтобы в новом проекте подключить базу сигналов, ее следует добавить в пункте меню **Параметра проекта → База данных**, как представлено на рис. 7.7.1:

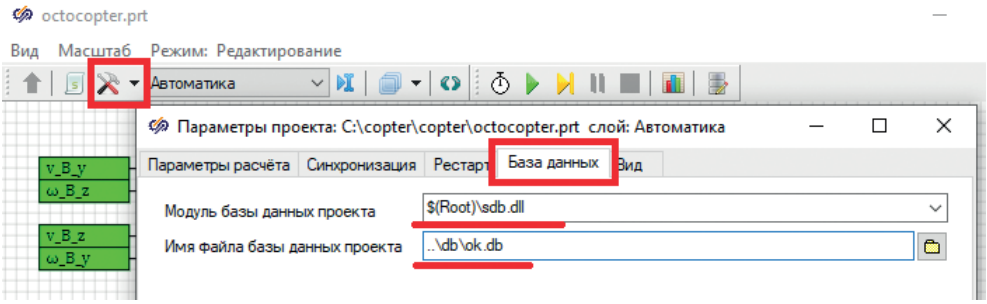


Рис. 7.7.1. Подключение базы сигналов

Следует выбрать модуль расширения **$(Root)\sdb.dll** (единственный вариант в стандартной поставке SimInTech из выпадающего списка), дальше вписать относительный путь к новому файлу базы как «**..\db\ok.db**» – директорию db мы создали ранее, а имя файла базы можно назначить произвольно (но все проекты одной комплексной модели в будущем, как правило, должны использовать это же имя). После этого в директории **db** должны появиться файлы **ok.db** и **ok.dbconf**, а в главном меню SimInTech активироваться пункт **Инструменты → База данных…**.

Примечание: в этот момент лучше всего сохранить проект, закрыть его и снова открыть, чтобы он подгрузился с уже прописанной базой сигналов, и корректно проинициализировались все пути к файлу базы данных.

После нового открытия файла проекта зайдите в пункт меню **Инструменты → База данных…** и создайте в базе новые категории с именами «ВМГ», «Коптер», «Регуляторы», «Кнопки», пользуясь кнопкой с плюсом внизу соответствующей колонки, как показано на рис. 7.7.2:

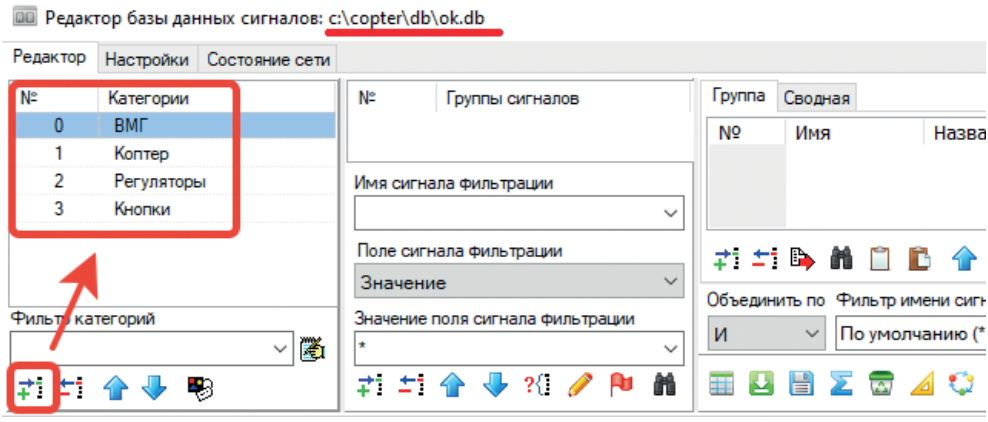


Рис. 7.7.2. Создание новых категорий в базе сигналов

Для созданий категории надо нажать на плюс, потом двойным щелчком по появившейся категории с именем «Новая катего рия» перейти в интерфейс редактирования категории, в котором задать соответствующее имя. Проделайте это 4 раза, чтобы результат совпал с рис. 7.7.2.

7.8. Категория «Коптер»

После этого зайдите в редактор категории «Коптер» и задайте там аккуратно шаблонные сигналы, как они показаны на рис. 7.8.1. Ситуация здесь (по колонкам) аналогична сигналам проекта, описанным выше. Пока что добавляем здесь 20 сигналов, позже по необходимости будем дополнять категорию. Смысл сигналов понятен из описания, и для реализации уравнений нам из этих сигналов потребуется всего 6 – это главные моменты инерции (точнее, составляющие тензора инерции) **Ixx**, **Iyy**, **Izz**. Сигналы **Ixx1**, **Iyy1**, **Izz1** являются обратными числами для главных моментов и чисто служебные – чтобы не делить на каждом шаге расчета 1 на одно и то же число, вычислим частные 1/**Ixx**, 1/**Iyy**, 1/**Izz** один раз при инициализации расчета и дальше будем пользоваться посчитанными числами.

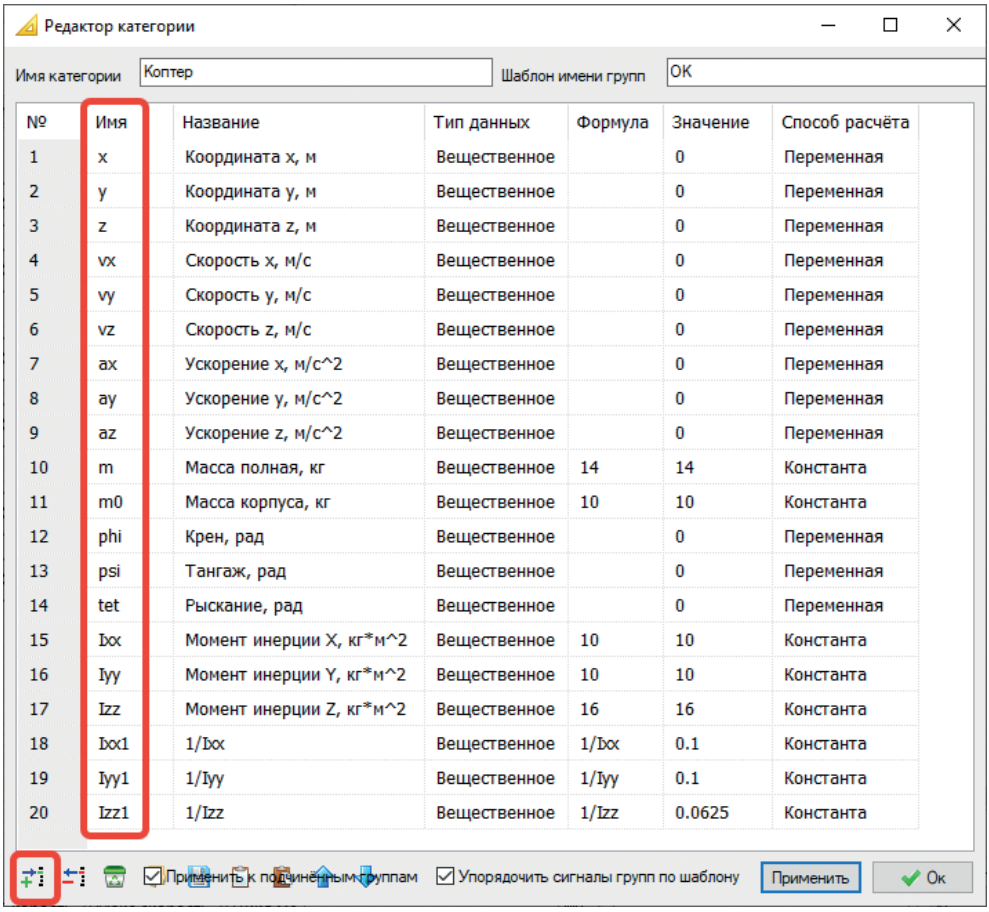
Массу рамы коптера принимаем в первом приближении 10 кг, а для восьми вращающихся масс ВМГ примем, что каждая из них обладает массой 0,5 кг, следовательно, полная масса будет 14 кг. Другие сигналы – это вычисляемые переменные, они пригодятся нам в будущем для вывода на кадр, передачи информации в регулятор и т. п.

После добавления шаблонных сигналов нажмите кнопку **Ок**, чем вы завершите редактирование категории «Коптер».

Примечание: тем, что мы создали шаблон категории и задали 20 шаблонных сигналов, – этим действием мы еще не создали сами сигналы. Пока что просто подготовили категорию, описывающую такой объект, как «коптер», или твердое осесимметричное тело в пространстве трех координат.

Далее, в интерфейсе базы сигналов выделите (еще раз) категорию «Коптер» и зеленой кнопкой в колонке **Группы сигналов** добавьте группу с именем **OC** (лучше английскими буквами, это сокращение от слов Octo Copter), как показано на рис. 7.8.2.

Этим действием создастся 1 экземпляр объекта с именем **OC**, описываемого сигналами по шаблону категории «Коптер». При этом в проекте появится 20 новых глобальных сигналов (пока что примерно аналогичных тем сигналам, которые мы завели в локальных сигналах проекта). Но к этим сигналам в будущем уже будет доступ и из соседних проектов, а не только из данного проекта, который будет содержать модель коптера.



№	Имя	Название	Тип данных	Формула	Значение	Способ расчёта
1	x	Координата x, м	Вещественное		0	Переменная
2	y	Координата y, м	Вещественное		0	Переменная
3	z	Координата z, м	Вещественное		0	Переменная
4	vx	Скорость x, м/с	Вещественное		0	Переменная
5	vy	Скорость y, м/с	Вещественное		0	Переменная
6	vz	Скорость z, м/с	Вещественное		0	Переменная
7	ax	Ускорение x, м/с^2	Вещественное		0	Переменная
8	ay	Ускорение y, м/с^2	Вещественное		0	Переменная
9	az	Ускорение z, м/с^2	Вещественное		0	Переменная
10	m	Масса полная, кг	Вещественное	14	14	Константа
11	m0	Масса корпуса, кг	Вещественное	10	10	Константа
12	phi	Крен, рад	Вещественное		0	Переменная
13	psi	Тангаж, рад	Вещественное		0	Переменная
14	tet	Рыскание, рад	Вещественное		0	Переменная
15	Ixx	Момент инерции X, кг*м^2	Вещественное	10	10	Константа
16	Iyy	Момент инерции Y, кг*м^2	Вещественное	10	10	Константа
17	Izz	Момент инерции Z, кг*м^2	Вещественное	16	16	Константа
18	Ixx1	1/Ixx	Вещественное	1/Ixx	0.1	Константа
19	Iyy1	1/Iyy	Вещественное	1/Iyy	0.1	Константа
20	Izz1	1/Izz	Вещественное	1/Izz	0.0625	Константа

Рис. 7.8.1. Шаблонные сигналы категории «Коптер».

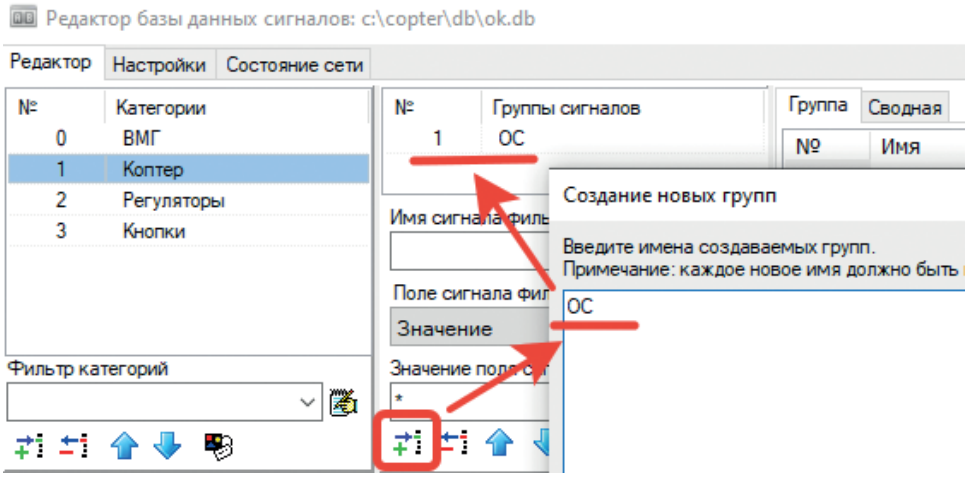


Рис. 7.8.2. Создание новой группы сигналов

Таким образом, в проекте появилось 20 новых сигналов со следующими именами:

```
OC_x, OC_y, OC_z,
OC_vx, OC_vy, OC_vz,
OC_ax, OC_ay, OC_az,
OC_m, OC_m0,
OC_Ixx, OC_Iyy, OC_Izz, OC_Ixx1, OC_Iyy1, OC_Izz1,
OC_phi, OC_psi, OC_tet.
```

В интерфейсе БД в колонке сигналов появятся эти сигналы, но без имени группы сигналов, но для проекта имена будут именно такими – состоящими из имени группы сигналов и имени сигнала (записанного в шаблоне категории).

Этими сигналами можно сразу начать пользоваться в проекте, что мы скоро и сделаем.

Сохраните проект, при этом будет задан вопрос о сохранении базы данных; так как она изменилась – ответить следует утвердительно.

7.9. Категория «ВМГ»

Аналогично тому, как мы создали шаблонные сигналы для категории «Коптер», задайте шаблон категории «ВМГ» в соответствии с рис. 7.9.1. Здесь константами будут являться номинальная частота вращения коптера в радианах (число **pi** является встроенной в SimInTech константой) и масса вращающейся части ВМГ в кг. Остальные три сигнала – переменные, понятные из описания.

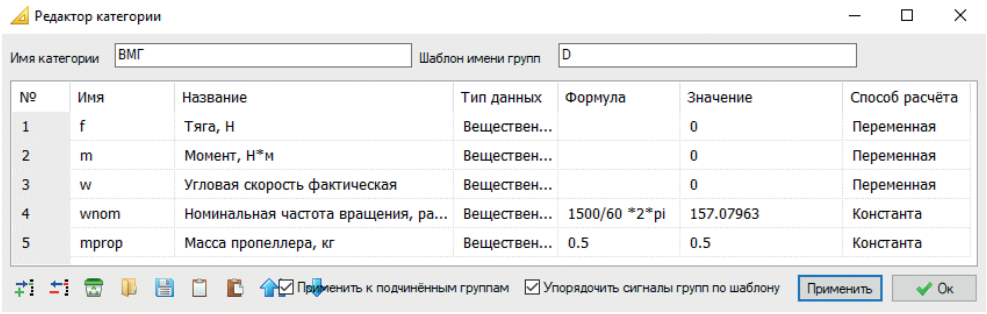


Рис. 7.9.1. Шаблон категории «ВМГ»

Далее, создайте в этой категории 8 (восемь) групп сигналов с именами D1, D2, D3, D4, D5, D6, D7, D8, как показано на рис. 7.9.2.

Тем самым мы добавили в проект еще 8 × 5 = 40 новых сигналов с именами D1_f, D1_m, D1_w, D1_wnom, D1_mprop, D2_f, D2_m, D2_w, D2_wnom, D2_mprop и т. д.

Сохраните проект, при этом будет задан вопрос о сохранении базы данных; так как она изменилась – ответить следует утвердительно.

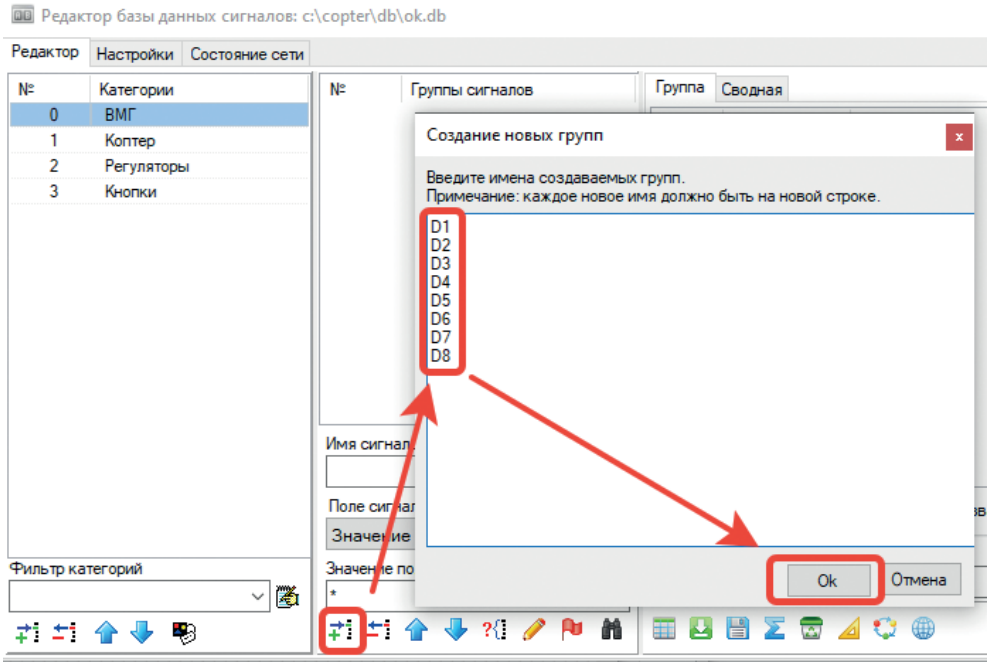


Рис. 7.9.2. Создание 8 групп сигналов для категории «ВМГ»

7.10. Масса коптера

Масса коптера вычисляется довольно просто – как сумма масс рамы и 8 масс ВМГ. Для тренировки использования сигналов базы данных давайте выполним этот расчет в свободной части расчетной схемы. Конечный результат представлен на рис. 7.10.1.

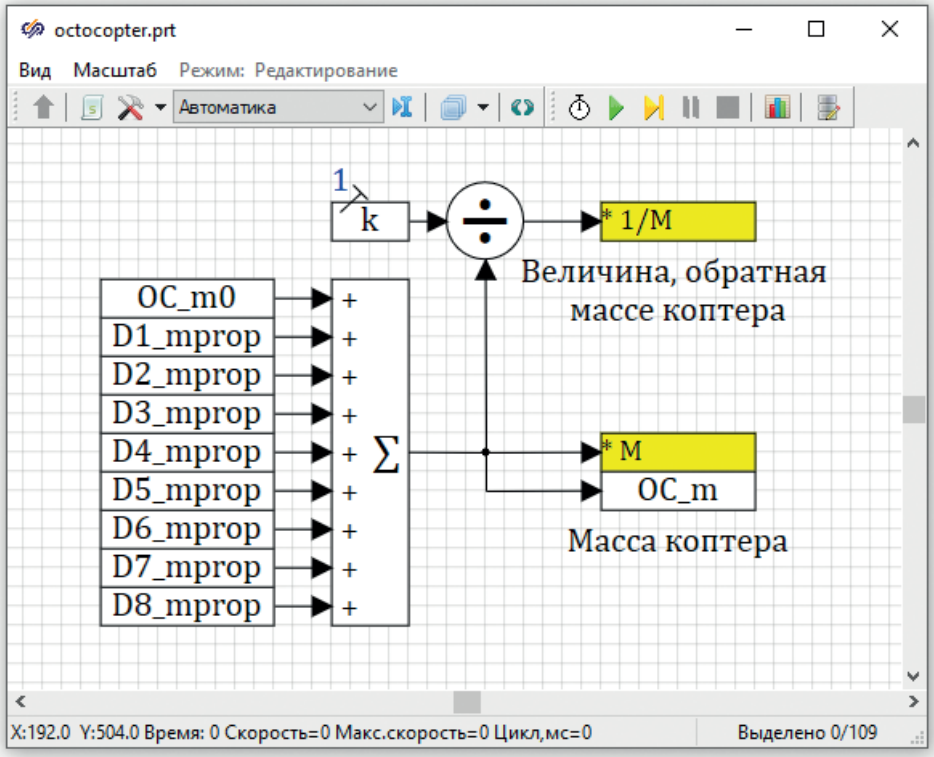


Рис. 7.10.1. Расчет массы коптера и обратной величины

Здесь, помимо знакомых блоков «В память», использованы еще блоки типа «Чтение из списка сигналов» (9 шт.) и блоки типа «Запись в список сигналов» (1 шт.) – это первые два блока из линейки блоков «Сигналы». Также есть 1 блок типа «Константа» и 1 блок типа «Делитель» (из линеек блоков «Константы» и «Операторы»). Найдите их в палитре блоков самостоятельно… Задайте вектор из 9 единиц в блоке сумматора и соедините все линиями связи. Второй входной порт у блока «Делитель» можно дополнительно разместить снизу для удобства проведения линии связи (рис. 7.10.2) – для этого надо зайти в свойства блока двойным щелчком по нему, и там перейти во вкладку **Порты**.

Смысл схемы рис. 7.10.1 очевиден, поясним только лишь, что этот же результат можно бы было получить, вычислив все вручную в базе сигналов. Однако если предположить, что масса коптера по какой-то причине может менять-

ся в процессе расчета (например, при добавлении груза...), то необходимость в такой схеме возникает. На каждом шаге расчета из базы сигналов будет считываться 9 сигналов, вычисляться их сумма и обратная сумме величина. Эти две величины мы будем использовать на расчетной схеме дальше, а полученная сумма также будет записываться в сигнал **OC_m** базы сигналов (для дальнейшего вывода на пульт управления, например).

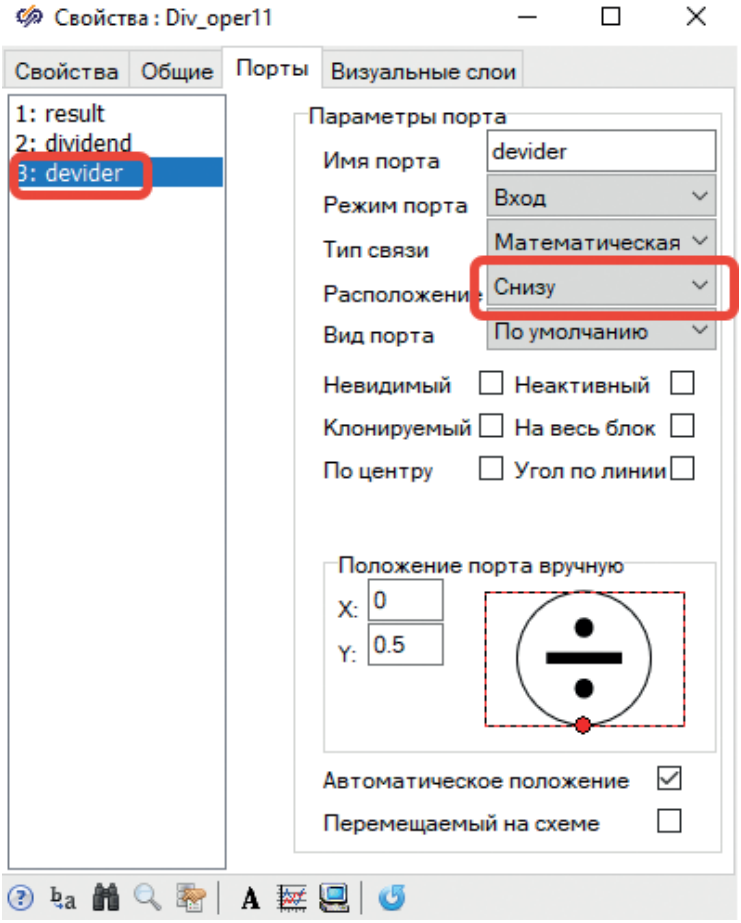


Рис. 7.10.2. Размещение порта снизу, у блока «Делитель»

7.11. Силы, действующие на коптер

Если вернуться к исходной системе уравнений (4.1.1), то можно легко увидеть, что практически все сигналы у нас заведены (кроме, быть может, плотности воздуха и характерных площадей коптера) для окончательного набора уравнений на расчетной схеме.

Приступим к набору оставшихся слагаемых.

$F_{Mx}(t)$, $F_{Dx}(t)$, $F_{Ox}(t)$ должны быть просуммированы и разделены на массу коптера, прежде чем поступать на общий сумматор перед интегратором, а так как

они сами по себе тоже являются сложными суммами, структурно их будет проще вынести в отдельные субмодели. Разместите три блока типа «Субмодель» (линейка **Субструктуры**) на схеме, еще один сумматор, задав ему весовые множители **[1, 1, 1]**, блок типа умножения и блок «Из памяти» рядом с первым сумматором на схеме, как показано на рис. 7.11.1, и соедините все линиями связи. Блок «Из памяти» должен читать линию связи, обозначенную как **1/M** (см. предыдущий подраздел).

Получилась «заготовка» для первого слагаемого первого уравнения. Далее, зайдите (двойным щелчком) по очереди внутрь в каждую из трех субмоделей и разместите там по 1 блоку типа «Порт выхода» из вкладки **Субструктуры**, задав имя каждому блоку как Fmx(t), Fdx(t) и Fox(t). Последовательность диалоговых окон очевидна и показана на рис. 7.11.2. Выйти из уровня субмодели можно или двойным щелчком на пустом месте расчетной схемы, либо нажав зеленую кнопку вверх (видна на рис. 7.11.2) в панели инструментов схемного окна, находясь на уровне вложенности субмодели. Для двух других субмоделей последовательность действий аналогична.

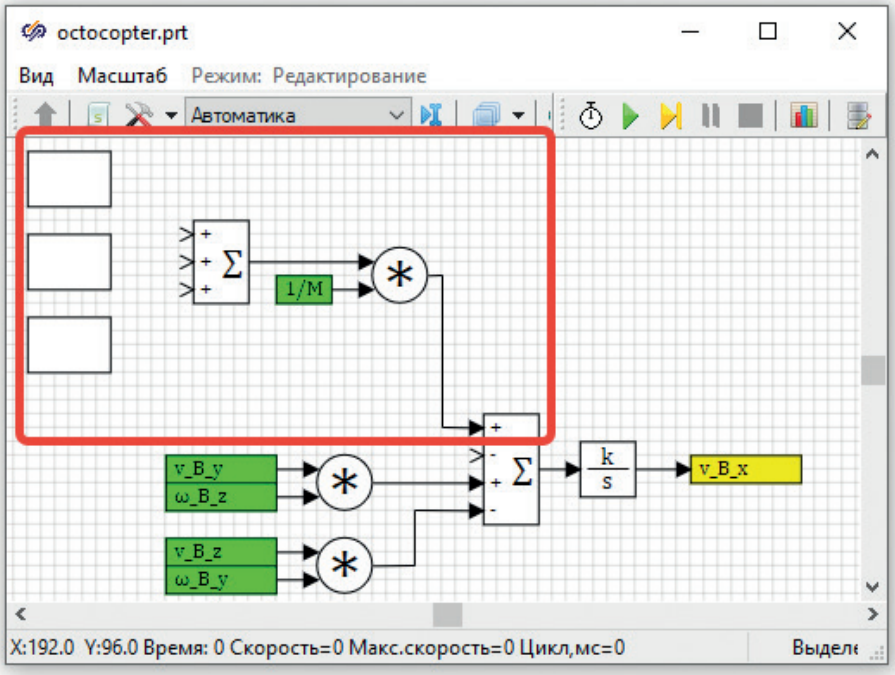


Рис. 7.11.1. Подготовка субструктур для 1 уравнения исходной системы

В итоге у каждой из субмоделей должен появиться 1 выходной порт, на каждой из субмоделей он будет подписан (по умолчанию, а вообще поведение внешнего вида и подписей портов субмодели можно изменять), внешний вид схемы должен преобразоваться к виду рис. 7.11.3, если соединить линиями связи субмодели и размещенный новый сумматор.

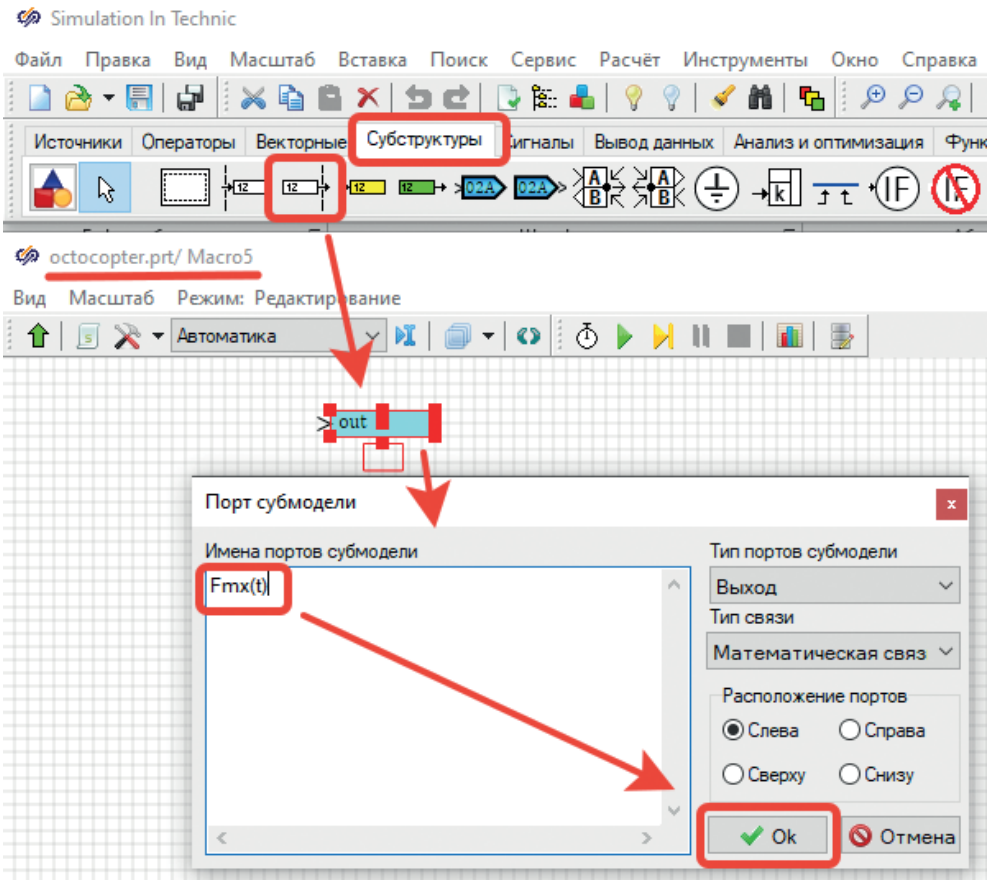


Рис. 7.11.2. Задание имени Fmx(t) для порта выхода первой субмодели

Перейдем к наполнению субмоделей. Попробуйте сделать это самостоятельно, по рис. 7.11.4, 7.11.5 и 7.11.6.

Напомним формулы для сил по оси Bx:

$$F_{Mx}(t) = \sin(\gamma)\cdot\left(\frac{F_{M8}+F_{M6}-F_{M4}-F_{M2}}{\sqrt{2}}+F_{M3}-F_{M7}\right),\ F_{Dx}(t) = -0.5\rho\cdot C_D\cdot A_{yz}\cdot v_{Bx}\cdot|v_{Bx}|$$

остальные см. (7.1.1).

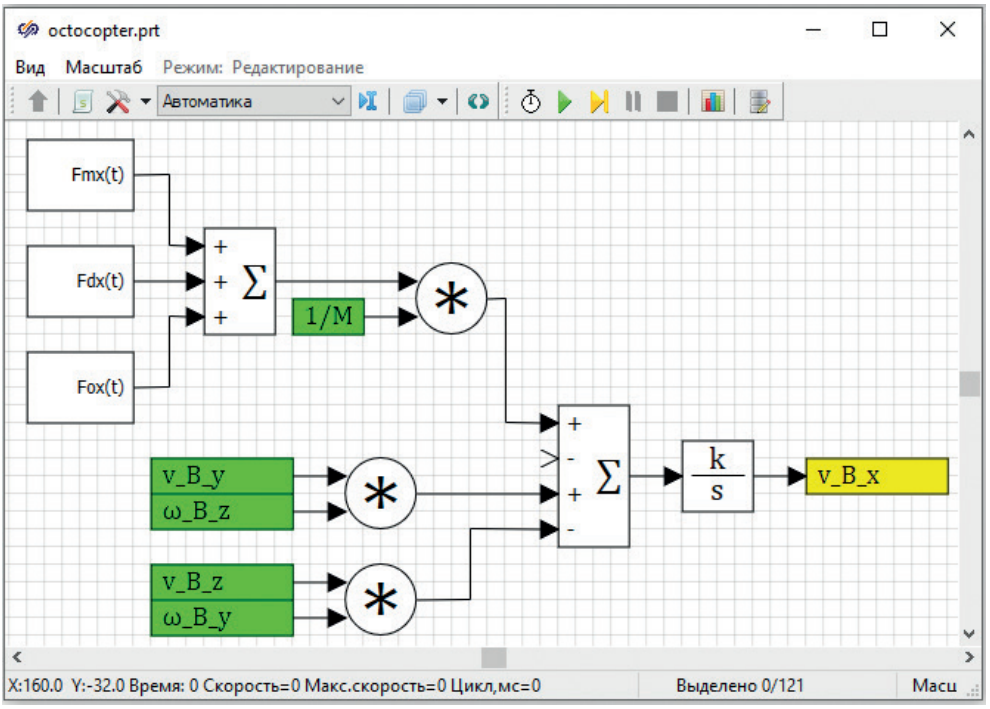


Рис. 7.11.3. Первое уравнение, набранное почти полностью

Так как на этапе создания модели коптера до конца не была определена компоновка и параметры коптера, то сила сопротивления воздуха сделана с очень примерно заданным коэффициентом порядка –(0.1…50), который в себя включает и плотность воздуха, и характерную площадь, и коэффициент формы (возможно, эта величина задана слишком большой по абсолютной величине, но для целей обучения это не принципиально). Этот коэффициент можно будет варьировать и смотреть на получающийся результат моделирования – по сути, этот коэффициент выполняет роль «демпфера» в модели коптера. Однако этот коэффициент сильно влияет на регуляторы – при изменении динамических характеристик объекта управления регуляторы надо будет перенастраивать… Возмущающая сила пока задана как 0 (ноль), в дальнейшем заменим ее на более существенную величину, задаваемую с пульта управления.

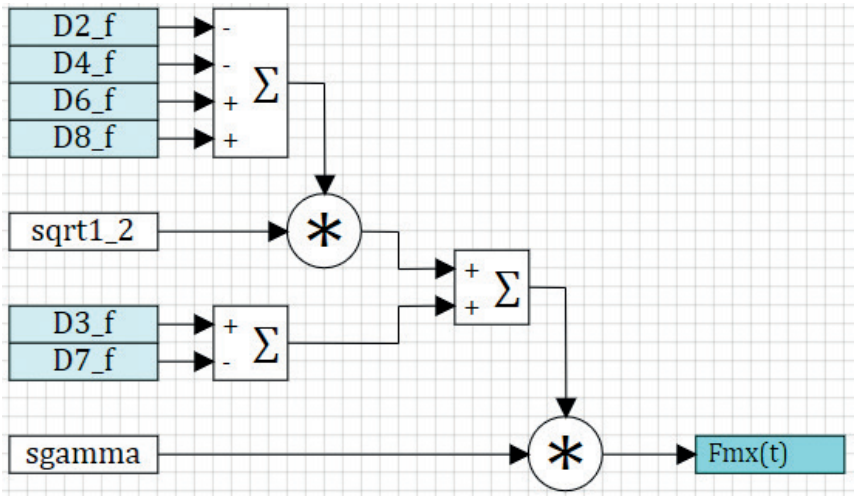


Рис. 7.11.4. Вычисление Fmx(t)

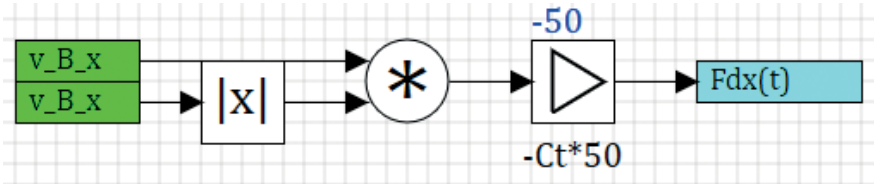


Рис. 7.11.5. Вычисление Fdx(t)

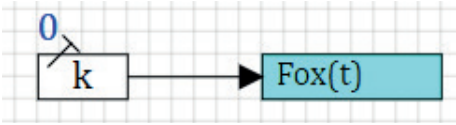


Рис. 7.11.6. Вычисление Fox(t)

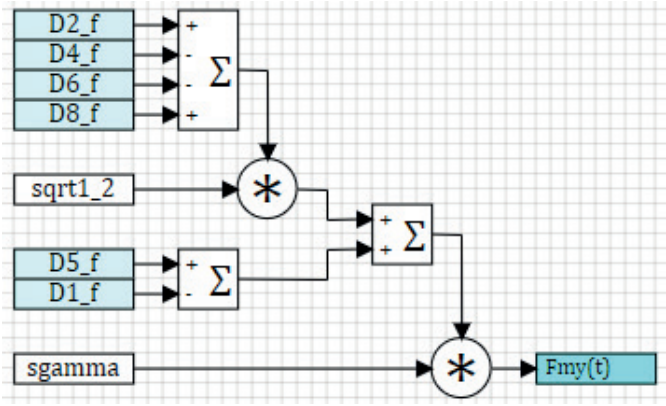


Рис. 7.11.7. Вычисление Fmy(t)

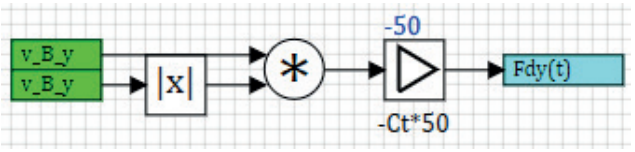


Рис. 7.11.8. Вычисление Fdy(t)

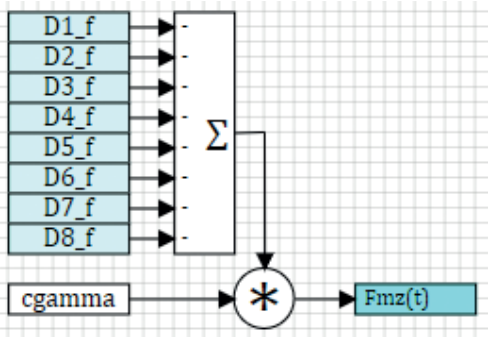


Рис. 7.11.9. Вычисление Fmz(t)

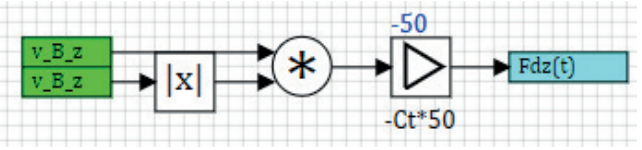


Рис. 7.11.10. Вычисление Fdz(t)

Остальные слагаемые сил, по осям Y и Z, наберите самостоятельно по аналогии. Рисунки соответствующих субмоделей представлены под номерами 7.11.7, 7.11.8, 7.11.9 и 7.11.10. Возмущающие силы Foy(t) = Foz(t) = 0, задайте их пока что нулевыми, как и Fox(t) на рис. 7.11.6. Обратите внимание на сумматор для Fmz(t) и на отрицательные коэффициенты по каждому из его входов – это важно, так как слагаемое идет с минусом (ось Z_B направлена вниз в модели, поэтому силы тяги ВМГ, действуя вверх, направлены вдоль отрицательного направления оси Z_B).

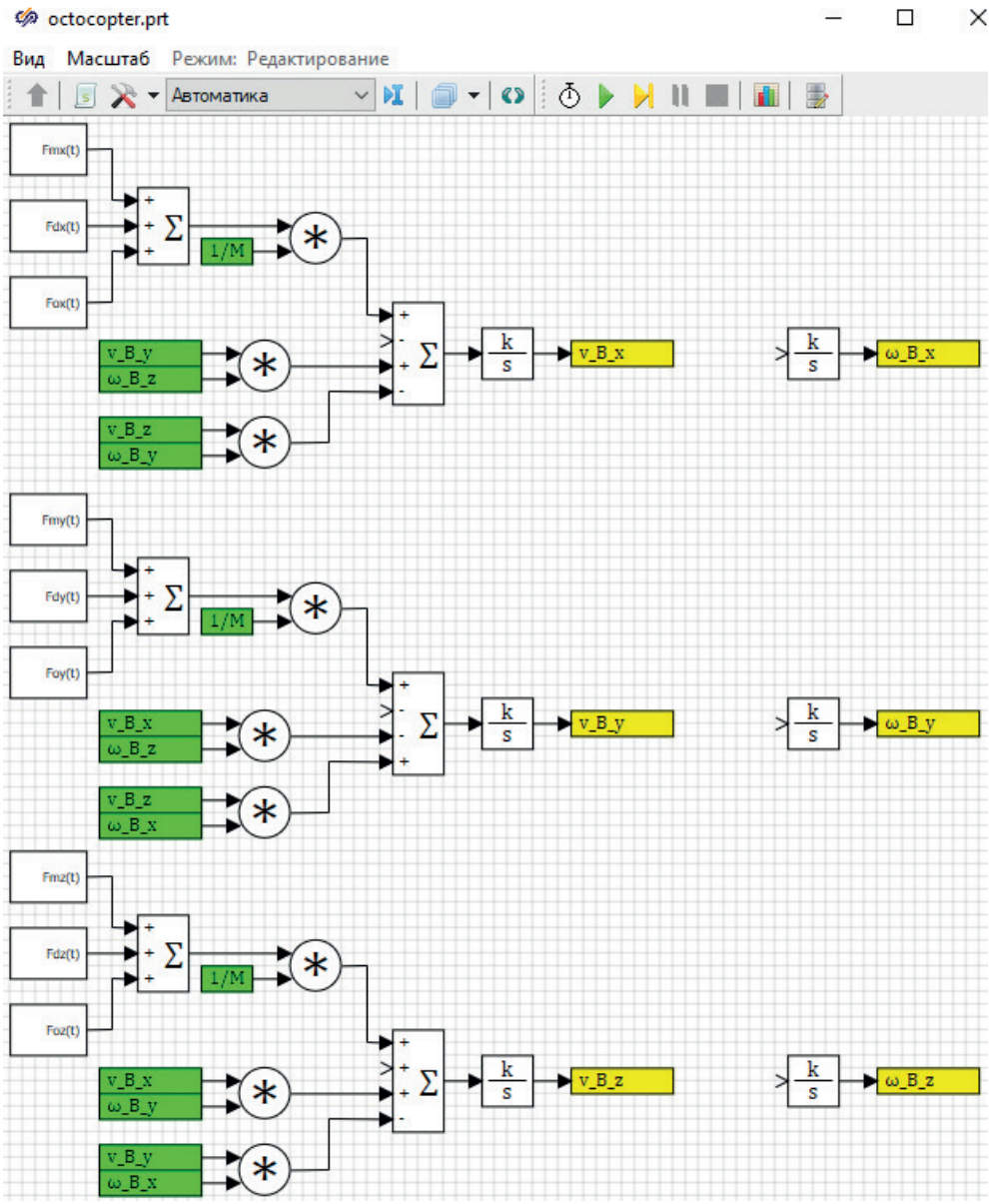


Рис. 7.11.11. Промежуточный вид набираемой системы уравнений

Промежуточный вид первых трех уравнений, который у вас должен получиться, пока без силы тяжести представлен на рис. 7.11.11. Видно, что постепенно система уравнений, представленная в структурном виде, разрастается, и в дальнейшем будет целесообразно разнести каждое из уравнений в свою субмодель. Пока что будем набирать их рядом, чтобы можно было легко увидеть ошибки и/или что-то недоделанное, сравнивая уравнения между собой.

Отметим еще раз, что подобные уравнения – в некоторых случаях – удобнее набирать непосредственно на языке программирования SimInTech. Но в данной учебной методике было принято решение все делать в структурном виде в целях обучения.

Также отладка модели, представленной в структурном виде, является более легкой процедурой, чем отладка программы, написанной на языке программирования. И для других пользователей модели структура важна – гораздо удобнее работать со схемой, набранной аккуратно и логически структурированной на отдельные смысловые части.

7.12. Сила тяжести и ориентация коптера

Если посмотреть на исходные уравнения (4.1.1), то видно, что сила тяжести с поправкой на ориентацию коптера входит как слагаемое в первые три уравнения, в первом уравнении с отрицательным знаком, во втором и третьем с положительным – это мы предусмотрели в сумматорах, размещенных ранее на схеме. Момента сила тяжести не создает. Требуемые сигналы ориентации коптера (углы крена и тангажа) пока еще не вычисляются в модели, но соответствующие сигналы в базу сигналов мы уже завели, поэтому можем ими воспользоваться на расчетной схеме. Только надо будет посчитать еще тригонометрические функции (синус и косинус) от них.

Напомним, см. формулу (3.1.17), что углы ориентации будем вычислять как интеграл от производных $\varphi'(t)$, $\theta'(t)$ и $\psi'(t)$ коптера, которые матрицей преобразования $\boldsymbol{W}_{BI}$ могут быть получены из угловых скоростей коптера по осям в связанной системе координат **B**. Координаты же x, y, z коптера в системе **I** будем вычислять как интегралы от скоростей коптера в этой же инерциальной системе, а сами скорости в системе **I** – при помощи матрицы поворота $\boldsymbol{R}_{BI}$, см. формулы (3.1.16) и (3.1.17).

Таким образом, для реализации этих дифференциальных уравнений нам потребуется еще 6 блоков типа интегратора для вычисления трех линейных и трех угловых координат коптера в инерциальной системе **I**. В общую исходную систему мы не стали этого записывать выше, чтобы не усложнять ее запись и реализацию.

Примечание: база сигналов, как «внешнее» хранилище (пусть и в оперативной памяти) сигналов по отношению к расчетной схеме, обладает одним недостатком – сигналы, записываемые в нее и потом используемые в других частях расчетной схемы, задерживаются на 1 шаг расчета, так как цикл расчета идет таким образом: чтение сигналов из базы, расчет, запись сигналов в базу данных. Далее цикл повторяется, и на следующем шаге на входах будут значения, рассчитанные на предыдущем шаге. Поэтому для передачи сигналов между разными уравнениями системы будем использовать механизм блоков «В память» и «Из памяти» – они соединяют линиями связи разные участки схемы напрямую (не через базу сигналов), и задержки на шаг интегрирования не возникает, что повышает точность вычисления и математическую устойчивость решения.

Если отдельно выделить слагаемые силы тяжести в проекциях на оси подвижной системы координат **B**, то слагаемое $g\boldsymbol{R}_{IB}\overrightarrow{\boldsymbol{e}_{Iz}}$, очевидно, будет равно, согласно (4.2.7):

$$g\boldsymbol{R}_{IB}\overrightarrow{\boldsymbol{e}_{Iz}} = g\cdot\begin{pmatrix} -\sin(\theta) \\ \sin(\varphi)\cdot\cos(\theta) \\ \cos(\varphi)\cdot\cos(\theta) \end{pmatrix} = \begin{pmatrix} -\sin(\theta)\cdot g \\ \sin(\varphi)\cdot\cos(\theta)\cdot g \\ \cos(\varphi)\cdot\cos(\theta)\cdot g \end{pmatrix}.$$

Для реализации этих проекций нам не хватает тригонометрических функций от углов ориентации коптера, ускорение свободного падения мы завели ранее как сигнал проекта. Сделаем заготовку для них рядом с набранной схемой в виде блоков «В память» с именами φ, θ и ψ (сразу для трех углов) и поставив блоки вычисления синуса и косинуса от этих углов рядом, и еще блоков «В память» с именами sin(φ), cos(φ), sin(θ), cos(θ), sin(ψ), cos(ψ), а также tg(...) от этих углов, как показано на рис. 7.12.1.

На приведенной (рис.7.12.1) схеме предполагается, что мы будем вычислять (каким-то способом слева от вновь поставленных интеграторов) производные от углов, интегрировать их и получать сами углы, которые будем записывать в сигналы базы данных **OC_phi**, **OC_tet** и **OC_psi**, а также в блоки типа «В память». От этих же сигналов вычисляются тригонометрические функции и записываются в свои сигналы «В память» для дальнейшего использования в расчетной схеме – они нам пригодятся для реализации как матриц поворота, так и для вычисления проекции силы тяжести (на самом деле слагаемые для силы тяжести – это тоже результат действия матрицы поворота $\boldsymbol{R}_{IB}$, последнего ее столбца).

Блоки вычисления тригонометрических функций – это один блок из линейки **Функции** с названием «Синусоидальная функция». Выпадающим меню в свойствах блока можно выставить расчет как синуса, так и косинуса или тангенса. Попробуйте проделать это самостоятельно, см. рис. 7.12.2.

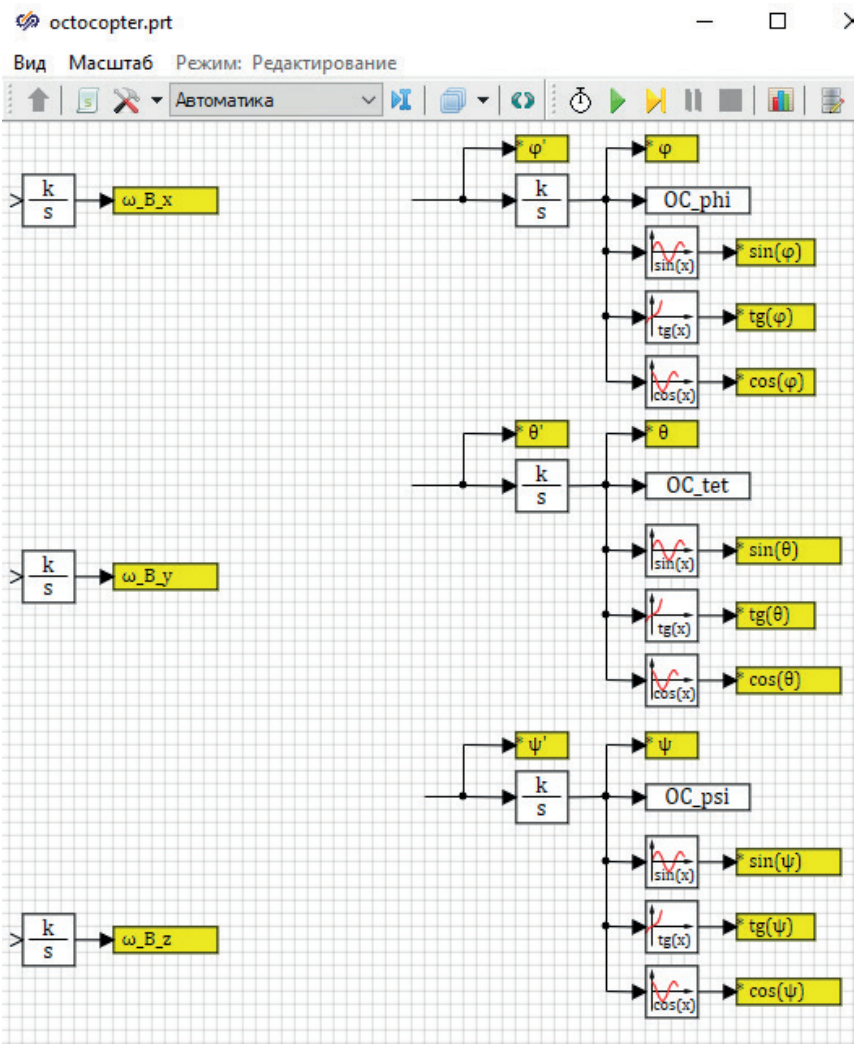


Рис. 7.12.1. Углы ориентации коптера

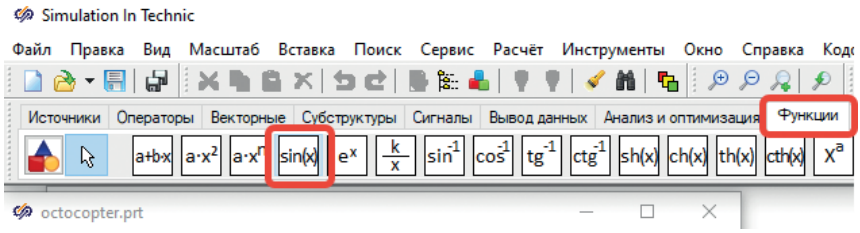


Рис. 7.12.2. Блок типа «Синусоидальная функция»

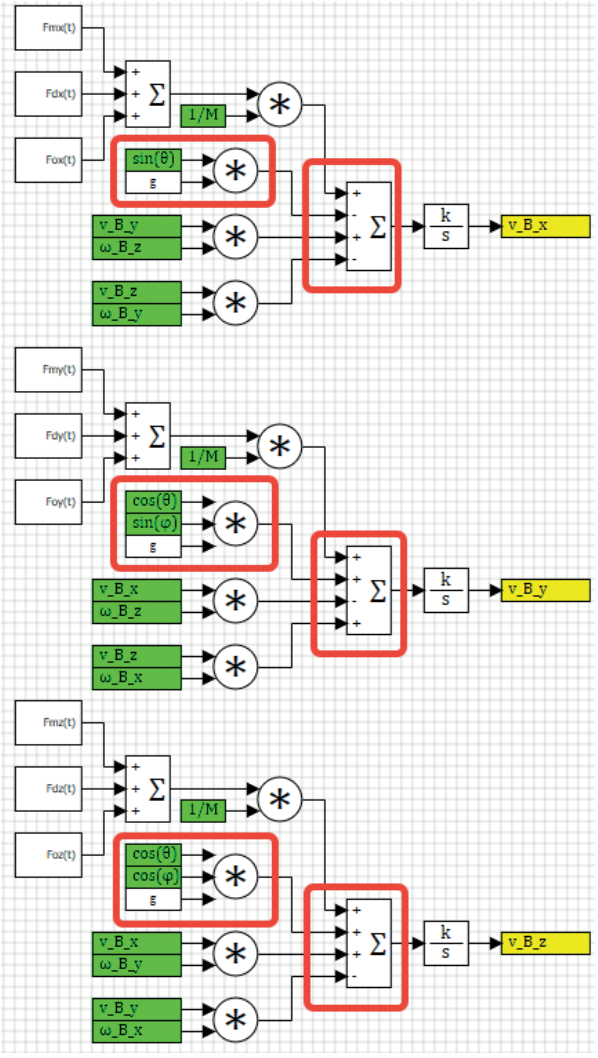


Рис. 7.12.3. Первые три уравнения системы, набранные полностью

Итак, «получив» линии связи, на которых будет вычисляться sin() и cos() от углов крена и тангажа, можем воспользоваться ими и набрать недостающие слагаемые в уравнениях системы – сделайте это аналогично рис. 7.12.3. Для этого потребуется блоки типа «Чтение из списка сигналов», где следует указать имя сигнала **g**, и блоки типа «Из памяти», которыми следует считать нужные тригонометрические функции. Обратите внимание, что перечень доступных сигналов «Из памяти» стал шире и не совпадает с базой данных – это особый перечень, только для блоков «Из памяти» и «В память». По мере использования сигналов звездочки на желтых блоках «В память» будут пропадать, означая, что сигнал хотя бы один раз в схеме используется.

Что следует еще раз проверить – знаки сумматоров, последовательно: **[1, -1, 1, -1]**, **[1, 1, -1, 1]**, **[1, 1, 1, -1]**. Либо сверьте их со знаками в исходной системе уравнений.

Таким образом, мы до конца реализовали первые три уравнения исходной системы (4.1.1) и подготовили интеграторы для вычисления углов ориентации коптера, схему для вычисления их тригонометрических функций и угловой скорости коптера в связанной системе координат **B**.

7.13. Матрицы преобразования

Прежде чем переходить к набору следующих трех уравнений системы (касательно угловых скоростей), давайте реализуем матрицу поворота $\boldsymbol{R}_{BI}$ и матрицу преобразования угловой скорости в системе **B** в производные углов ориентации ($\boldsymbol{W}_{BI}$). Основная идея в следующем – так как эти матрицы нам потребуются не один раз, проще их сделать в виде субмодели, на вход которой (на три входа) подаются проекции какого-либо вектора в системе **B**, а на выходе (на трех выходах) – проекции этого вектора в системе **I**. Все вычисления находятся внутри субмодели, см. рис. 7.13.1. После этого такую субмодель можно многократно использовать по месту надобности, копируя ее из одного места модели в другое.

Чтобы субмодель реализовать, надо разместить ее на схеме, а затем внутри нее проставить три порта входа субмодели и три порта выхода субмодели. Все блоки находятся на вкладке **Субструктуры**. После этого надо подписать каждый из портов, чтобы отличать их друг от друга. Результат должен быть похож (снаружи субмоделей) на рис. 7.13.1. Лучше еще поставить подпись под каждым блоком, чтобы был понятен смысл происходящего внутри субмодели.

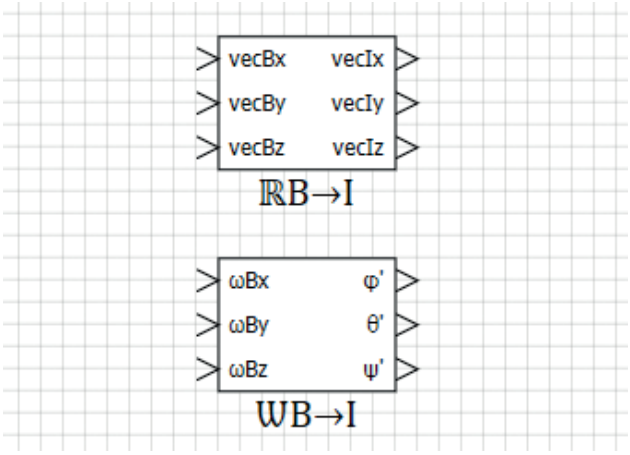


Рис. 7.13.1. Субмодели типовых подпрограмм – матриц преобразований

Далее внутри каждой из субмоделей необходимо корректно реализовать вычисления (7.10.1) (здесь буквами c() и s() обозначены тригонометрические функции cos() и sin()) b (7.10.2), см. так же уравнения (3.1.5):

$$\boldsymbol{R}_{BI} = \begin{pmatrix} c(\theta)\cdot c(\psi) & s(\varphi)\cdot s(\theta)\cdot c(\psi) - c(\varphi)\cdot s(\psi) & c(\varphi)\cdot s(\theta)\cdot c(\psi) + s(\varphi)\cdot s(\psi) \\ c(\theta)\cdot s(\psi) & s(\varphi)\cdot s(\theta)\cdot s(\psi) + c(\varphi)\cdot c(\psi) & s(\varphi)\cdot s(\theta)\cdot s(\psi) - s(\varphi)\cdot c(\psi) \\ -s(\theta) & s(\varphi)\cdot c(\theta) & c(\varphi)\cdot c(\theta) \end{pmatrix}; \quad (7.10.1)$$

$$\boldsymbol{W}_{BI} = \begin{pmatrix} 1 & \dfrac{sin(\theta)\cdot sin(\varphi)}{cos(\theta)} & cos(\varphi)\cdot tg(\theta) \\ 0 & cos(\varphi) & -sin(\varphi) \\ 0 & \dfrac{sin(\varphi)}{cos(\theta)} & \dfrac{cos(\varphi)}{cos(\theta)} \end{pmatrix}. \quad (7.10.2)$$

Подробнее о том, как получить матрицы поворота, можно посмотреть в сети или в [1]. Результат реализации представлен на рис. 7.13.2 и 7.13.3 соответственно. Проверьте аккуратно все слагаемые и элементы матриц – от их правильности зависит верность дальнейших расчетов модели.

Видно, что вычисления тригонометрических функций здесь используются многократно. То, что мы определили блоками «В память» сами значения синусов и косинусов, немного сэкономит процессорное время, так как вычисления тригонометрических функций будут происходить один раз на каждом шаге расчета, а не многократно в каждом месте, где требуется та или иная функция.

Теперь, набрав эти «типовые» подпрограммы, легко дальше дополнить расчетную схему. Например, чтобы вычислить входы в интеграторы углов ориентации, надо просто поставить субмодель с матрицей $\boldsymbol{W}_{BI}$, а на ее вход подать проекции угловых скоростей в системе **B**, которые у нас являются результатом расчета основной системы уравнений. Смотрите рис. 7.13.4. Далее можно перейти к набору трех оставшихся уравнений для этих угловых скоростей.

Матрица $\boldsymbol{R}_{BI}$ нам потребуется немного позже, но примерно с той же целью – преобразовать вектор линейной скорости из системы **B** в систему координат **I** и уже в системе **I** проинтегрировать скорость (проекции скорости), чтобы получить координаты коптера в системе координат **I**. На рис. 3.1.1 представлена матрица $\boldsymbol{R}_{IB}$, которая тоже потребуется немного позднее, и ее можете набрать самостоятельно.

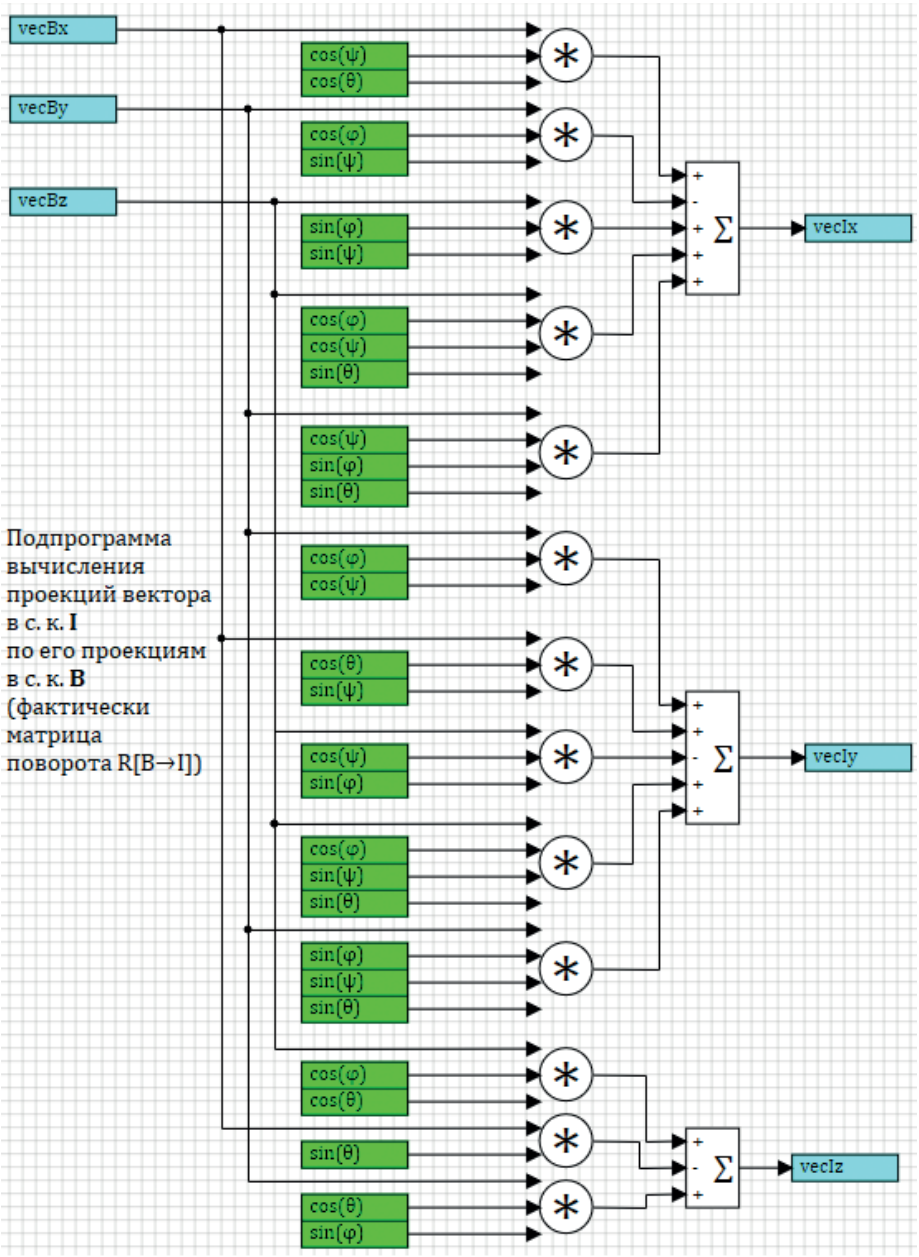


Рис. 7.13.2. Матрица $\boldsymbol{R_{BI}}$

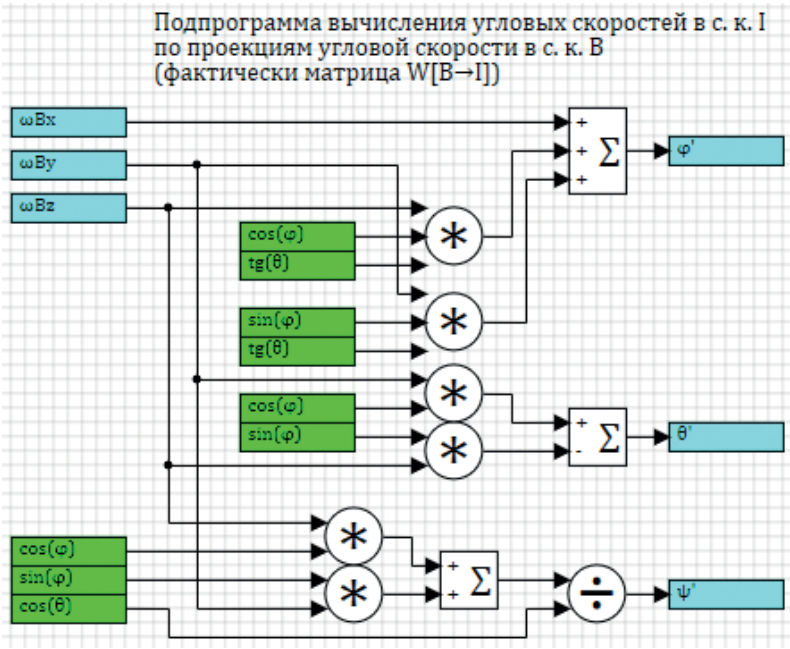


Рис. 7.13.3. Матрица W_{BI}

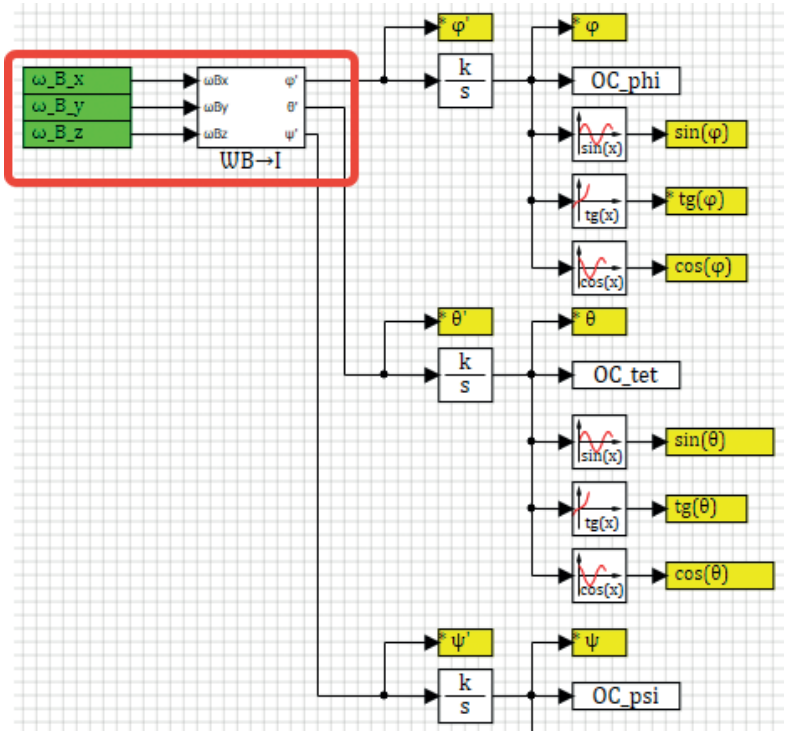


Рис. 7.13.4. Вычисление входов в интеграторы углов ориентации коптера

7.14. Моменты сил, действующих на коптер

Приступим к реализации последних трех уравнений, записанных в исходной системе, – точнее, правых частей уравнений, так как левые части мы набрали до этого (три интегратора и три блока «В память», обозначенные как ω_**B_x**, ω_**B_y**, ω_**B_z**).

Структура уравнений одинакова с точностью до слагаемых, поэтому внешне все будет выглядеть однотипно. По аналогии с тем, как мы набирали первые три уравнения, поставьте рядом с интеграторами по сумматору, в каждом будет 4 весовых множителя с +1, т. е. векторы **[1, 1, 1, 1]**. Между сумматором и интегратором поставьте блок типа «Усилитель», в котором надо будет вписать в качестве коэффициента усиления сигнал **OC_Ixx1**, **OC_Iyy1**, **OC_Izz1** соответственно – это будет аналогом деления на **Ixx**, **Iyy**, **Izz**, как показано на рис. 7.14.1. Под блоками усилителей лучше подписать, что они обозначают.

Примечание: так как в интеграторах тоже есть такое свойство, как коэффициент усиления, то оптимальнее было бы воспользоваться им и непосредственно в интеграторах поставить вместо 1 нужный сигнал – тогда усилители не потребовались бы. Но для наглядности схемы был выбран вариант, как на рисунке, – с отдельным явно выделенным коэффициентом.

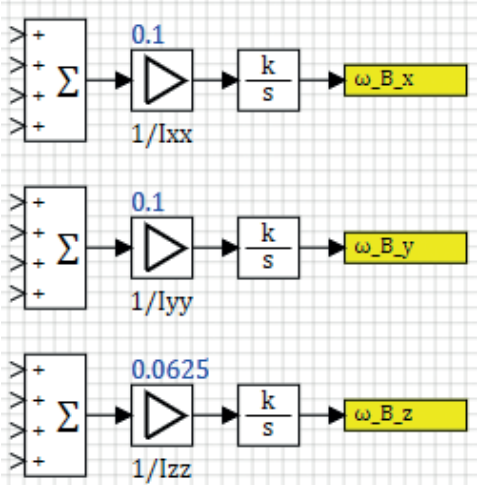


Рис. 7.14.1. Учет моментов инерции

Далее, перед сумматорами нужно поставить по три субмодели, в которых будут рассчитаны проекции моментов, действующих на коптер по осям, и отдельно вычисление четвертого слагаемого, см. рис. 7.14.2. Все делается аналогично первым трем уравнениям.

Внутри каждой из 9 новых субмоделей требуется набрать выражения, соответствующие требуемым проекциям моментов, действующих на коптер, см. формулы (7.1.2). Для произвольных Mox(t), Moy(t), Moz(t) пока что ставим нулевое значение внутри субмоделей – это нужно проставить для связности модели, а в дальнейшем там будет стоять сигнал, который будет возможно инициировать с пульта управления, имитируя внешнее воздействие на коптер.

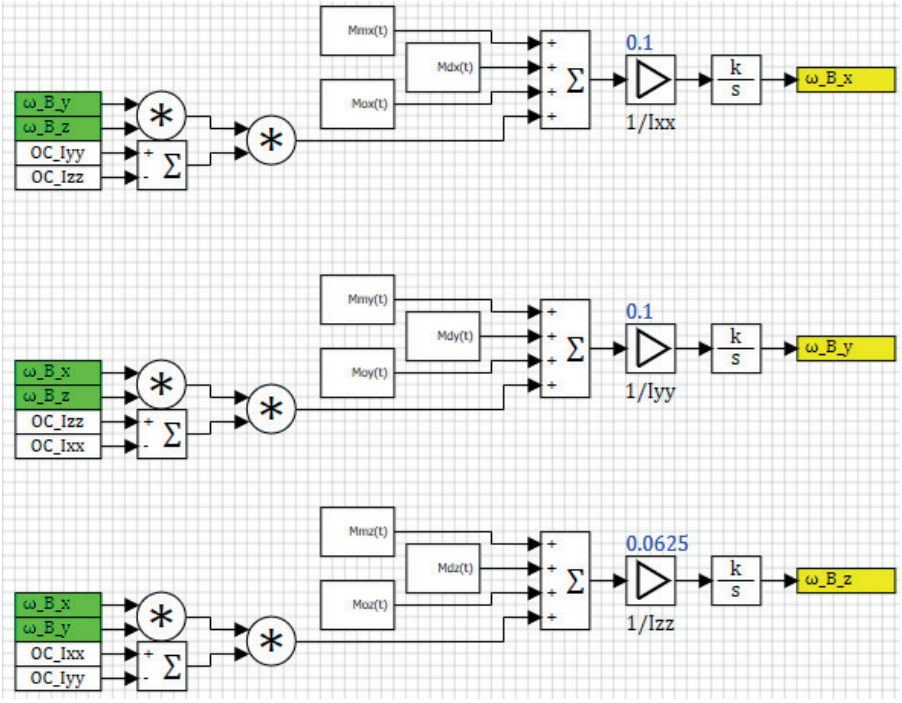


Рис. 7.14.2. Последние три уравнения основной системы

Для субмоделей Mmx(t), Mmy(t), Mmz(t) расчетная схема представлена на рис. 7.14.3, 7.14.4 и 7.14.5, а для Mdx(t), Mdy(t), Mdz(t) – на рис. 7.14.6, 7.14.7 и 7.14.8 соответственно.

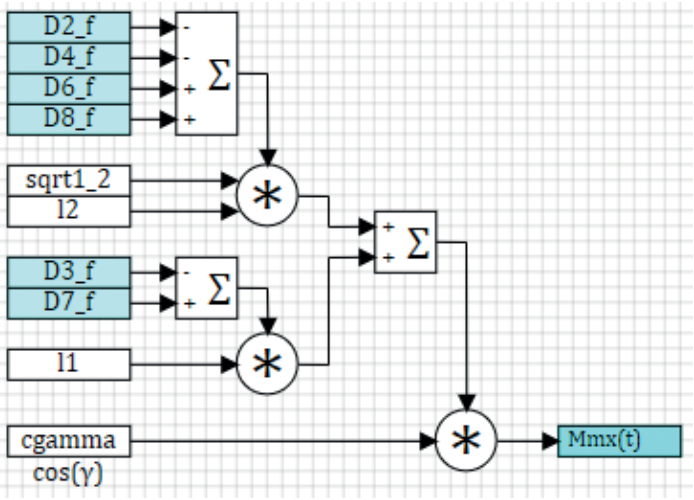


Рис. 7.14.3. Вычисление Mmx(t)

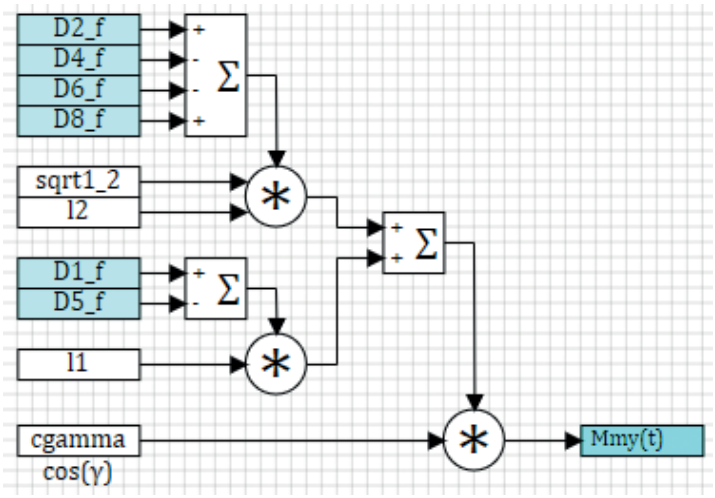


Рис. 7.14.4. Вычисление Mmy(t)

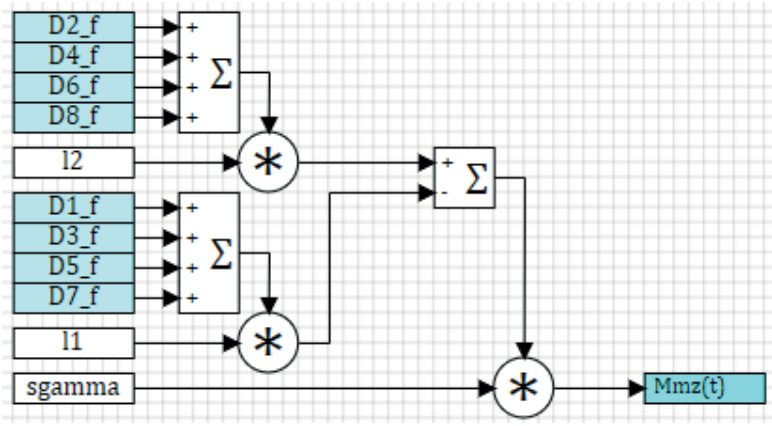


Рис. 7.14.5. Вычисление Mmz(t)

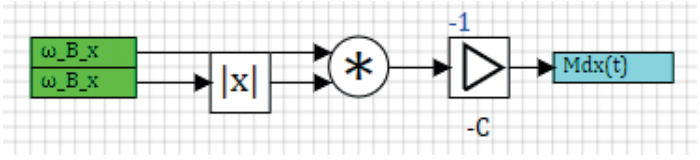


Рис. 7.14.6. Вычисление Mdx(t)

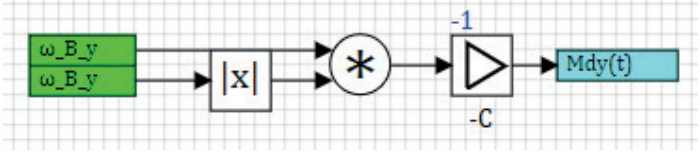


Рис. 7.14.7. Вычисление Mdy(t)

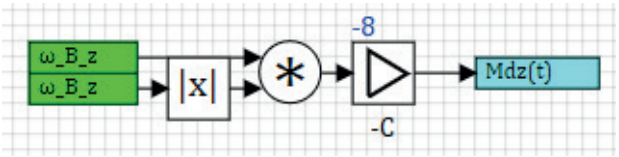


Рис. 7.14.8. Вычисление Mdz(t)

Вместо прикидочных коэффициентов –1, –1 и –8 позже (по мере поступления экспериментальных характеристик коптера) нужно поставить реальные коэффициенты. На данном этапе достаточно этих чисел. Сигналы **D1_f**, **D2_f** и им подобные здесь вставлены при помощи блоков типа «Чтение из списка сигналов» и помечены дополнительно голубым цветом, чтобы отличать то, что приходит из базы данных сигналов на схему, от чтения констант типа **l1**, **l2** или **sgamma**. А так это один и тот же блок (того же типа).

7.15. Инициализация схемы

На данном этапе набора схемы мы «замкнули» все вычисления каждого из слагаемых правых частей уравнений, тем самым получили завершенную модель динамики в нулевом приближении – и для проверки связности набранной схемы можно нажать кнопку **Инициализация** в схемном окне проекта или в главном окне SimInTech, и, если все было сделано корректно, – схема должна будет проинициализироваться. Об этом сообщит появление активной красной кнопки **Стоп** рядом с кнопкой инициализации, а также некоторые ненулевые линии связи станут фиолетово-розовыми.

Если вы набрали уже матрицу поворота $\boldsymbol{R_{BI}}$, которая пока что нам не пригодилась, система выдаст ошибку о том, что там не подключен входной порт, – в этом случае следует исключить из расчета данную субмодель (надо выделить субмодель и выбрать пункт меню **Правка → Исключить объекты**, субмодель станет серого цвета и не будет включена в следующий расчет).

Прежде чем запускать схему на расчет, давайте выставим параметры расчета – аналогично рис.7.15.1, – поставим **метод Эйлера** с постоянным шагом **1/1000** с и конечным временем расчета **10** с.

Остальные параметры расчета можно оставить заданными по умолчанию. Подробнее параметры расчета описаны в справочной подсистеме.

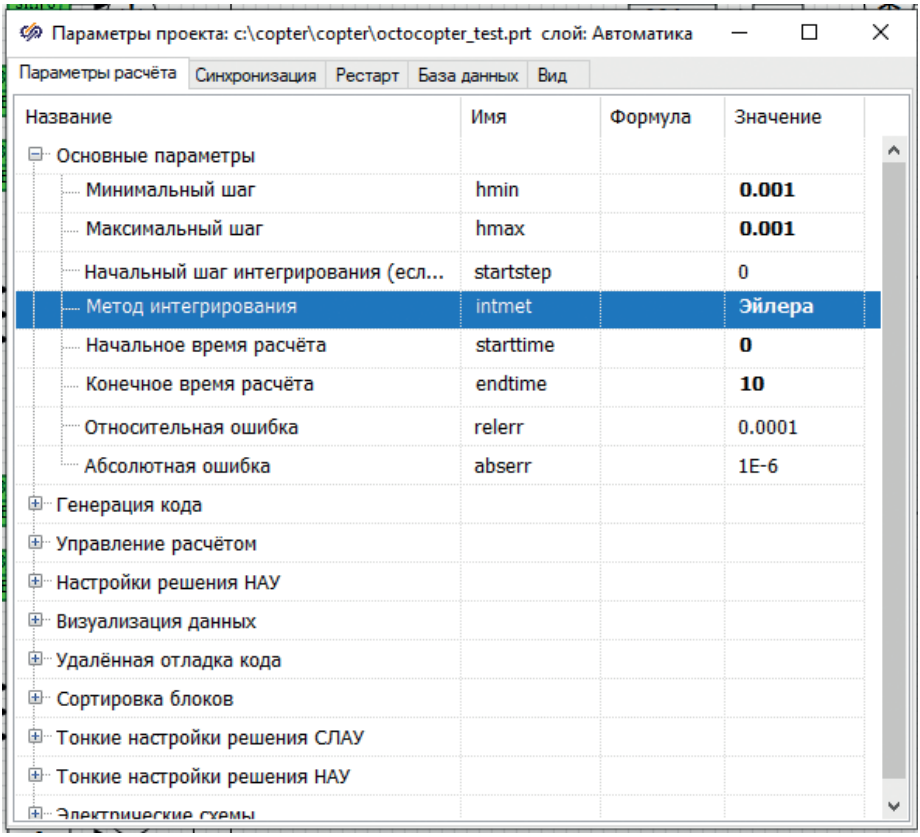


Рис. 7.15.1. Параметры расчета для тестового пуска

Если запустить схему на расчет (зеленой кнопкой **Пуск** рядом с инициализацией) – то под действием ненулевой силы тяжести коптер будет набирать вертикальную скорость (по оси z, которая направлена вниз), а так как остальные действующие силы пока что нулевые – модель ВМГ еще не сделана, и в базе сигналов стоят нулевые значения для силы тяги двигателей, – то таким моделированием мы имитируем просто свободное падение коптера. К концу расчета (10-я секунда модельного времени) скорости по осям x, y должны так и оставаться нулевыми, а скорость по оси z должна вырасти примерно до **1.657** м/с, см. рис. 7.15.2.

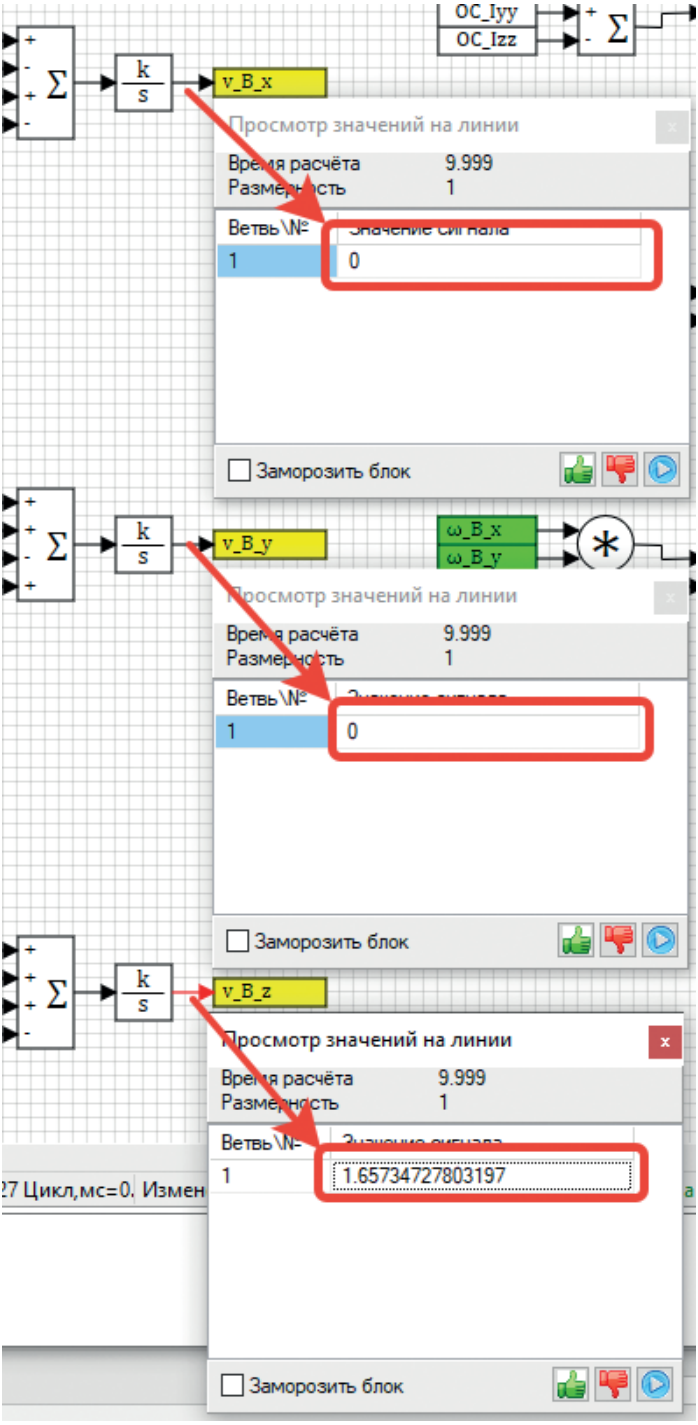


Рис. 7.15.2. Скорость к 10-й секунде

Такой результат (а не 98.1 м/с, исходя из ускорения свободного падения и 10 с полета) объясняется тем, что коптер падает не в вакууме, а в сопротивляющемся воздухе, с заданным довольно большим коэффициентом сопротивления, который сейчас стоит в модели (в нашем случае это –50 и это приблизительно в 70–100 раз больше реального коэффициента). То есть на скорости 1.657 м/с воздух в модели сопротивляется с силой, сопоставимой с силой тяжести, а на саму эту величину модель выходит уже на первых секундах моделирования, см. рис. 7.15.3. Величина коэффициента сопротивления воздуха – вещь, требующая уточнений, а пока что нам достаточно текущего результата – таким моделированием мы убеждаемся, что схема собрана топологически корректно, инициализируется на расчет верным образом, и уравнения написаны без явных ошибок (по крайней мере, в задействованной части). Если вы получили качественно другой результат – надо перепроверить набранную схему и убедиться, что при инициализации из входных воздействий есть только сила тяжести, а моментов на коптер не должно действовать никаких!

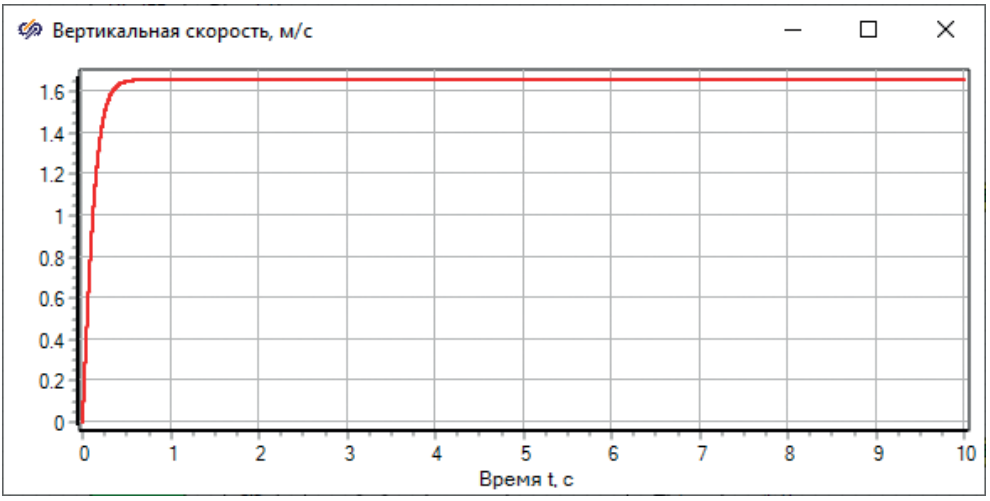


Рис. 7.15.3. Первые 10 с тестового моделирования (свободное падение вниз)

Если коэффициент поставить –1, а не –50, то скорость, на которую выходит коптер, составит 11.7 м/с (что, скорее всего, ближе к физическим реалиям) – в дальнейшем скорректируем коэффициенты сопротивления воздуха для линейного и вращательного движений, пока что пусть остаются такими, как заданы, для большей устойчивости модели и «удобства» при первичной настройке регуляторов.

7.16. Вычисление координат коптера

Проверив таким образом корректность набранной схемы (хотя бы номинальную корректность), можно перейти к завершающим шагам по модели динамики объекта – осталось набрать еще расчет координат коптера (центра его масс) в инерциальной системе отсчета **I**, а также модель электродвигателя ВМГ и самой ВМГ в плане получения силы тяги как зависимости от текущей частоты вращения ротора двигателя (или винта).

Чтобы получить текущие координаты центра масс коптера, возьмем вектор линейной скорости коптера, посчитанный для системы **B**, преобразуем его в систему **I** (соответствующей матрицей поворота) и проинтегрируем скорость в системе **I**: $\overrightarrow{\boldsymbol{v}_I}(t) = \boldsymbol{R}_{BI}\overrightarrow{\boldsymbol{v}_B}(t)$, см. рис. 7.16.1.

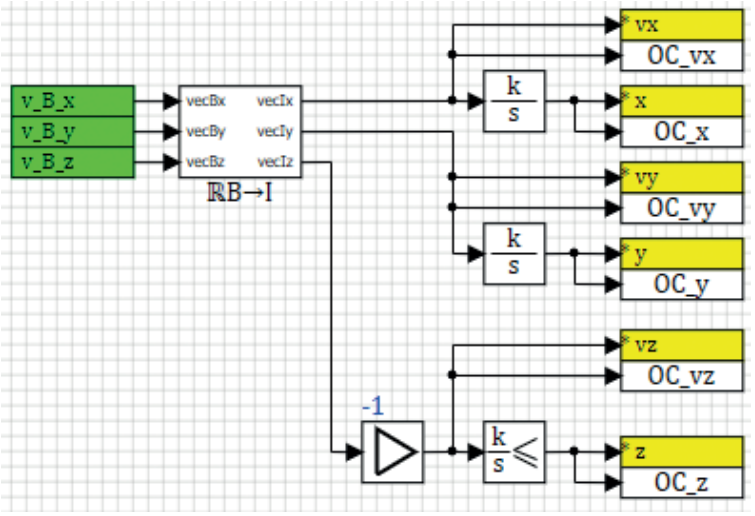


Рис. 7.16.1. Расчет координат коптера

Здесь использованы те же блоки, которыми мы пользовались ранее, за исключением блока типа «Интегратор с ограничением» – это такой же по смыслу интегратор, который интегрирует входной сигнал. Но на его выход накладывается ограничение в заданном диапазоне. Мы выставили ограничения от –1000 до 1000 м, так как полеты выше или ниже этой отметки не предполагались, а в тестовых расчетах из-за постоянно действующей силы тяжести и (порой сильной) силы тяги двигателей иногда (из-за ошибок в модели) высота оказывалась существенно выше или ниже этих величин, что заведомо неверно, – и для удобства анализа было принято решение поставить такой блок. Свойства в нем заданы, как на рис. 7.16.2, – с начальной высотой +40 м.

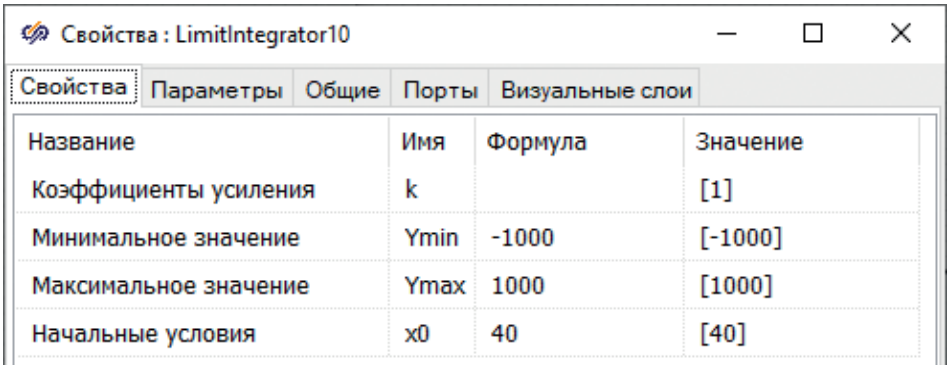


Рис. 7.16.2. Интегратор для вычисления координаты z

И еще один нюанс – для удобства анализа модели при вычислении координаты z ось была направлена вверх, т. е. положительная скорость **vz** в системе **B** (направлена вниз) дает отрицательный прирост координате z, поэтому стоит еще блок с коэффициентом –1 перед интегратором. В другой реализации модели это может быть не так. В принципе, координата z является уже величиной, которая не влияет на динамику коптера – это уже моделирование датчика (высотомера) или просто справочная информация, никакой обратной связи на уравнения динамики высота не имеет – поэтому можно ось направить и так, и так, главное, потом придерживаться выбранного направления.

Следующим шагом в моделировании будет создание модели ВМГ и задание приблизительно такой частоты вращения каждому из двигателей, чтобы коптер сохранял равновесие в пространстве (или хотя бы находился вблизи этого равновесия), для оценки адекватности модели – пока что без регуляторов, или модели полетного контроллера.

Сейчас, если запустить модель на расчет, то координата z будет уходить с +40 м в минус, начиная с плавного разгона и потом со скоростью 1.657 м/с. Остальные координаты должны оставаться нулевыми.

Давайте еще раз просмотрим, что на текущий момент получено, как это выглядит с верхнего уровня, – есть некоторая модель динамики, **выходами** которой являются три линейные координаты и три угла ориентации в неподвижной системе **I**: φ, θ, ψ, x, y, z; а переменными состояния являются шесть скоростей (линейных и угловых) в подвижной системе **B**: vBx, vBy, vBz, ωBx, ωBy, ωBz. Строго говоря, все 12 этих переменных являются переменными состояния коптера, просто линейные координаты внутри системы не используются, а углы ориентации можно трактовать одновременно и как переменные состояния, и как выходной (искомый) результат вычислений уравнений динамики.

Всего на схеме проставлено 12 интеграторов (11 обычных и 1 с ограничением), следовательно, реально записана система из 12 дифференциальных уравнений, некоторые из которых нелинейные. Поскольку внешне мы все размещали на одной плоскости, на одной расчетной схеме, за исключением отдельных слагаемых, – схема разрослась до средних размеров и уже становится неудобной для чтения. А весь «изюм» структурных схем – в наглядности реализованной математики. Предлагается систему уравнений разделить на 6 уравнений для вычисления скоростей, каждое из которых «упаковать» в свою субмодель, и еще 6 уравнений для координат – они могут разместиться и на одном листе, так как основная часть – матрица поворота – там уже скрыта в субструктуру. В итоге у вас может получиться схема, которая от рис. 7.16.3 (как есть сейчас – на одном уровне набраны все уравнения) перейдет к рис. 7.16.4 (в каждой субмодели записаны свои уравнения). С точки зрения математики это идентичные системы, а с точки зрения удобства их использования – две большие разницы. Лучше привыкать к структурированности расчетной схемы – ее так и другим людям удобнее воспринимать, и дальнейшая работа, или модификация, будет более простой и наглядной.

Входными воздействиями на модель динамики октокоптера, кроме силы тяжести, которой мы никак не можем управлять, и силы сопротивления воздуха, являются силы тяги винтомоторных групп, к моделированию которых мы дальше и приступим.

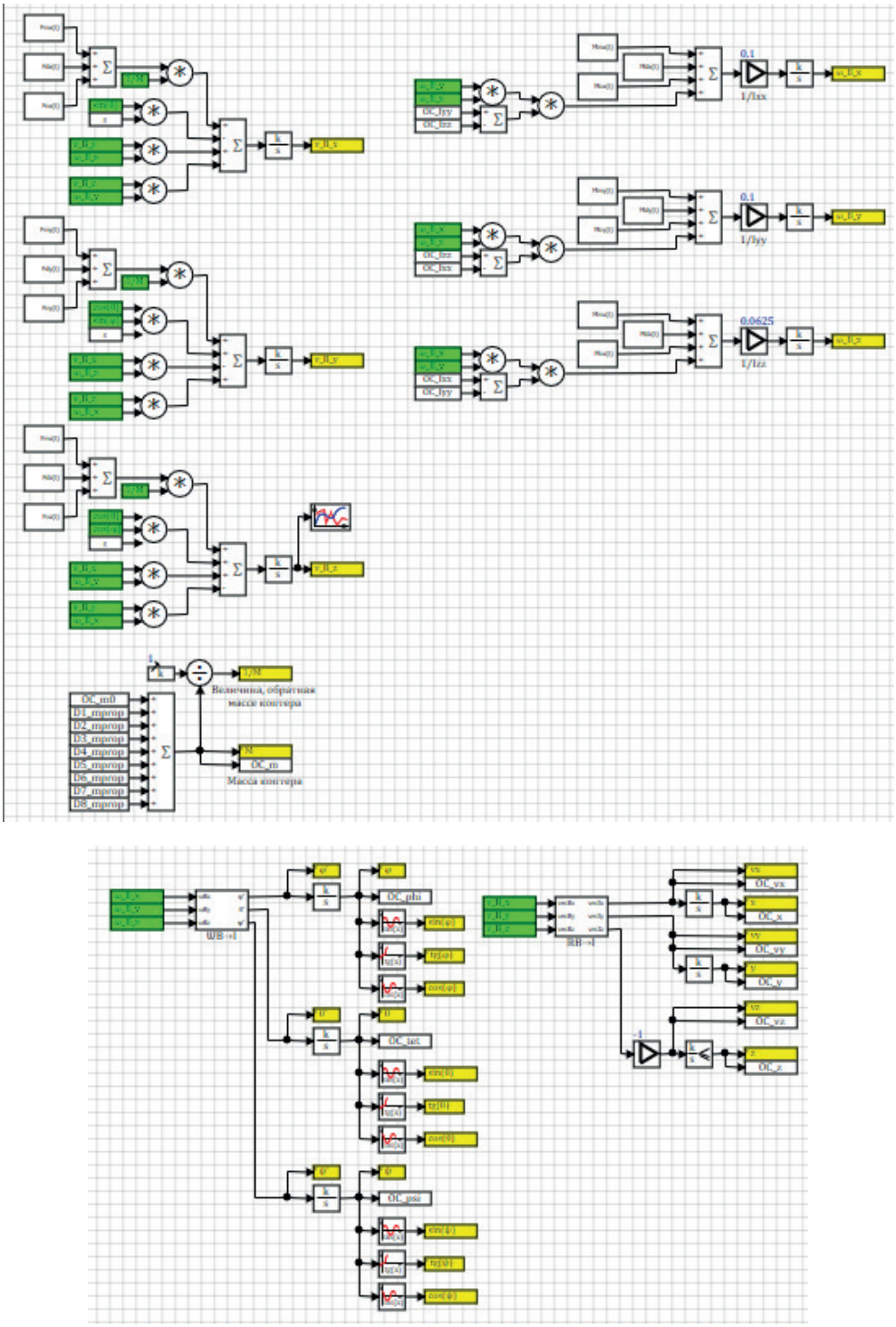


Рис. 7.16.3. Дифференциальные уравнения в бесструктурном виде

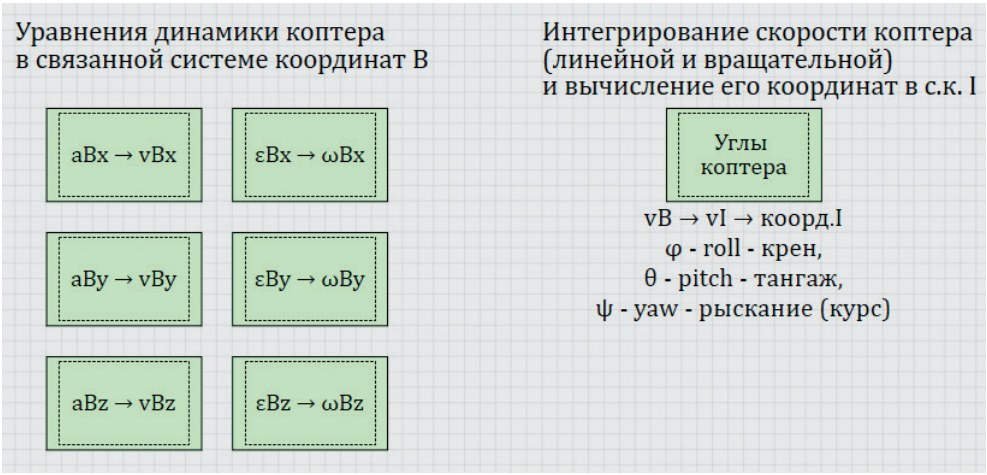


Рис. 7.16.4. Те же 12 дифференциальных уравнений, упакованные в субструктуры

Моделирование ВМГ

8

Моделирование винтомоторных групп вынесем в отдельный раздел, потому что это не относится впрямую к динамике коптера как твердого тела, движущегося в трехмерном пространстве. Кроме этого, так как в объекте 8 ВМГ, каждая из которых, в общем-то, должна вести себя одинаково, здесь имеет смысл применить так называемую типовую подпрограмму, которая будет одинаковым образом обрабатывать каждую из 8 ВМГ. Типовую подпрограмму мы сделаем 1 раз в отдельном prt-файле и используем ее 8 раз в модели коптера.

Модель ВМГ будет состоять из дифференциального уравнения первого порядка (или, что то же самое, из динамического блока) типа «Инерционное звено первого порядка», где с некоторой инерционностью выходная величина стремится к значению, подаваемому на вход. Этим моделируется «разгон» и «останов» электродвигателя.

На вход такому блоку будем подавать заданное значение $\omega_{Мзад}(t)$ угловой частоты вращения, посчитанной вручную или в регуляторе в дальнейшем, а на выходе – будет текущая (измеренная) частота вращения. Если текущую частоту вращения проинтегрировать, можно вычислить текущий угол поворота ротора электродвигателя (нам в модели это не потребуется, только если как справочная величина).

Далее, пропорционально квадрату угловой частоты вращения $\omega_{Mi}(t)$ можно вычислить тягу ВМГ и момент ВМГ (в нашем приближении момент околонулевой и не учитывается), поэтому считать будем только тягу каждой из 8 ВМГ и передавать ее – через базу сигналов – на вход в уравнения динамики коптера.

8.1. Подготовка типовой подпрограммы ВМГ

Создайте новый файл проекта типа «Схема модели общего вида» и сохраните его как **C:\copter\sub\motor.prt**. База данных для этого проекта не нужна.

Далее, непосредственно в этом файле поставьте два блока типа «Входной порт» и «Выходной порт» субмодели из вкладки **Субструктуры** без самой субмодели. Мы набираем сейчас «внутренность» субмодели, которая позже будет в основном проекте. Для порта входа задайте имя **ω_зад(t)**, а для порта выхода **ω(t)**. Между портами разместите блок типа «Инерционное звено первого порядка» из вкладки **Динамические**, соедините линиями связи и сравните результат с рис. 8.1.1 и 8.1.2.

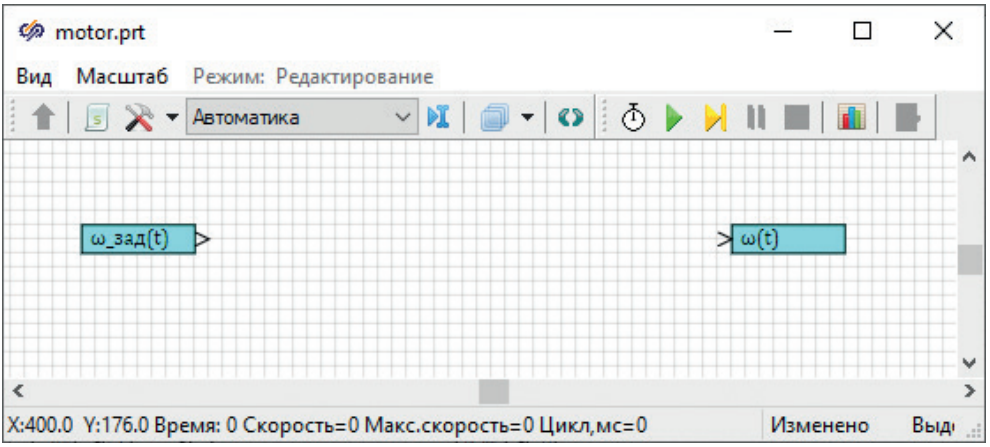


Рис. 8.1.1. Подготовка типовой подпрограммы

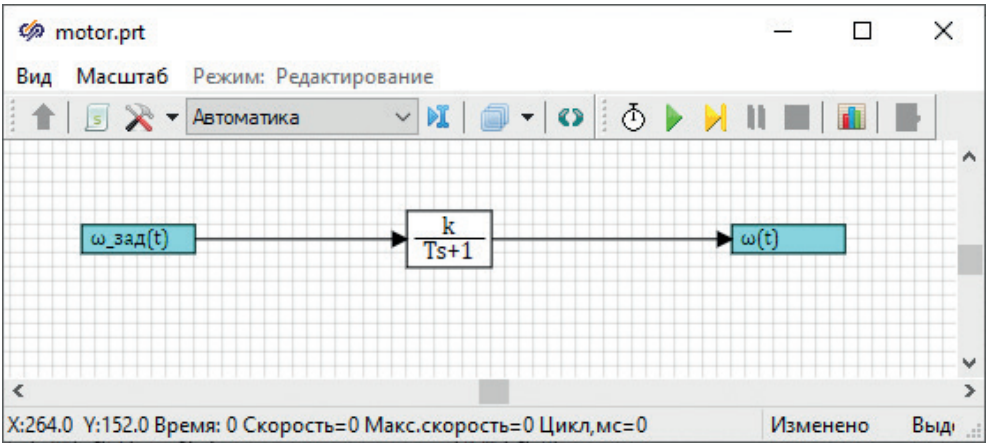


Рис. 8.1.2. Простейшая типовая подпрограмма ВМГ

В SimInTech есть возможность использовать внешний prt-файл как содержимое субмодели для другого файла проекта. Нам потребуется набранную модель ВМГ использовать 8 раз в модели, чтобы по одному и тому же принципу вычислять 8 разных угловых скоростей каждой из ВМГ. Сохраните проект **motor.prt** и закройте его.

8.2. Использование типовой подпрограммы

Вернитесь к проекту **copter.prt**, разместите на нем новую субмодель и, зайдя в ее свойства, впишите в общее свойство «Имя файла субмодели» строку «**..\sub\motor.prt**», а свойству Name дайте имя «**D1**» (без кавычек). Для примера см. рис. 8.2.1.

После этих действий сохраните файл **copter.prt**, закройте его и откройте заново, чтобы внутренне содержимое субмодели D1 загрузилось из внешнего

файла, и теперь при заходе внутрь субмодели вы должны увидеть содержимое того, что сделали ранее в файле **motor.prt**. На рис. 8.2.2 обратите внимание на заголовок окна.

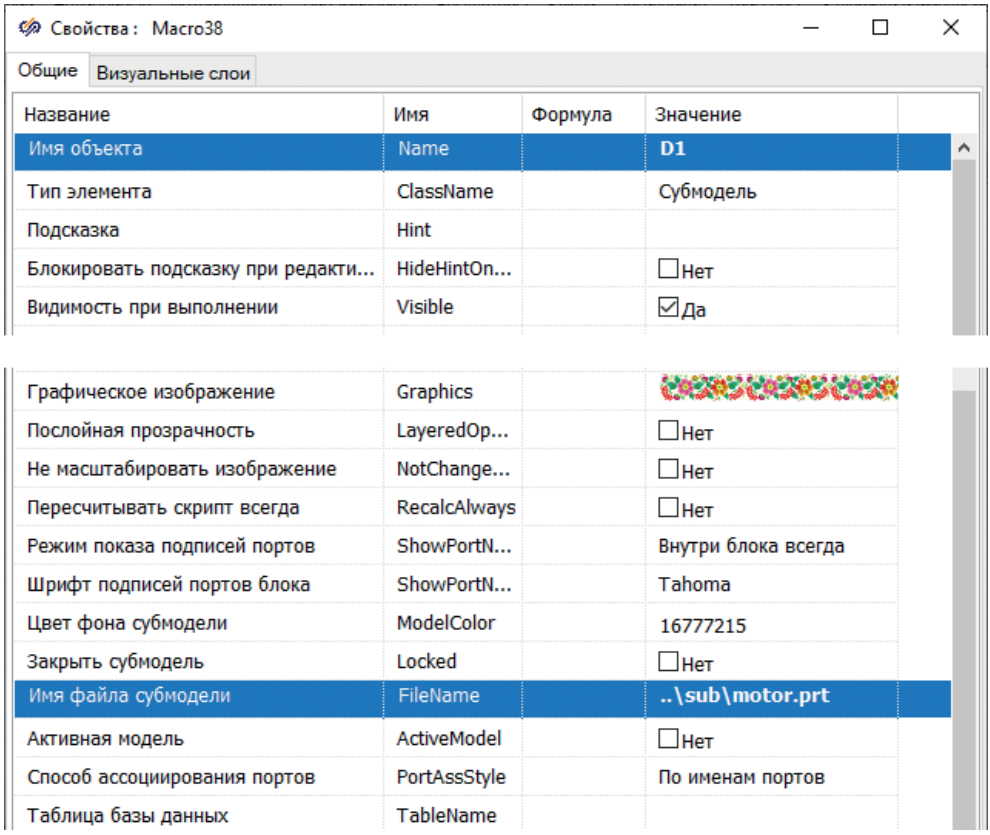


Рис. 8.2.1. Задание содержимого субмодели как внешнего файла

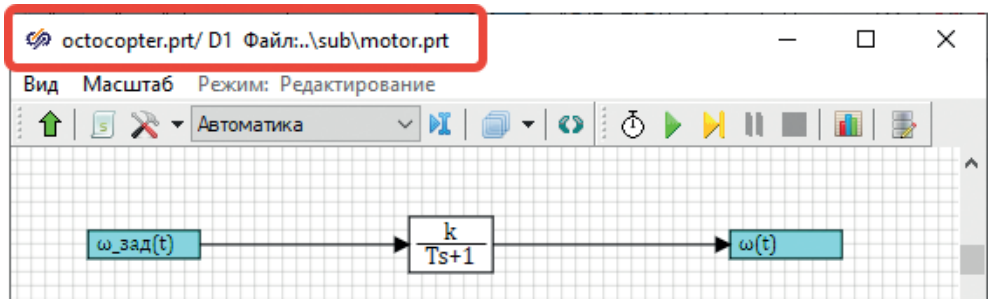


Рис. 8.2.2. Субмодель, загруженная из внешнего файла

Имя D1, которое мы дали блоку типа «Субмодель», пока не используется, но дальше мы его задействуем, а пока что именно D1 нам поможет при копировании этой субмодели еще 7 раз – настроенная таким образом субмодель,

скопированная рядом на схеме, будет принимать имена D2, D3, D4 и т. д (автоматизм SimInTech). Но, прежде чем копировать, задайте еще одно свойство у субмодели – чтобы в ее подписи отображалось имя, надо выставить свойство «Стиль подписи» в значение **Имя**, тогда внешний вид будет похож на рис. 8.2.3.

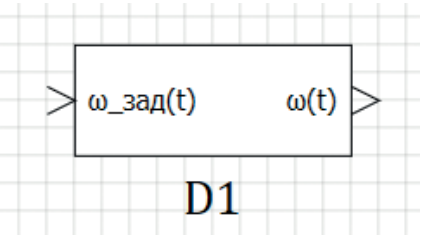


Рис. 8.2.3. Внешний вид субмодели с типовой подпрограммой двигателя ВМГ

Далее, выделив субмодель и нажав **Ctrl**+**C** или правой кнопкой по субмодели и выбрав пункт контекстного меню **Копировать** и далее **Ctrl**+**V** (Вставить), нужно разместить еще 7 таких блоков. Всего их будет 8, как на рис. 8.2.4.

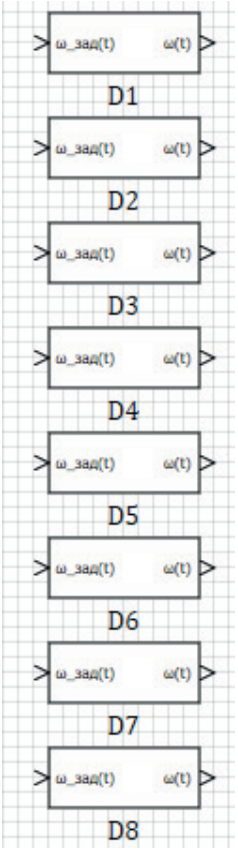


Рис. 8.2.4. Субмодели, моделирующие 8 ВМГ

Обратите внимание, что имена субмоделей «случайно» совпали с именами групп сигналов, ранее заведенных в базу данных при создании сигналов для ВМГ (см. подраздел 3.2.7, рис. 19). Дальше мы будем использовать эту «случайность», чтобы по одной и той же подпрограмме просчитать каждую из ВМГ и вычислить для каждой группы сигналов частоту вращения и силу тяги.

Примечание: часть схемы, которая находится внутри каждой из этих 8 субмоделей, реально записана в другом файле, в файле **motor.prt**. Поэтому любые изменения, которые вы далее попробуете вносить внутри любой из этих 8 субмоделей, НЕ СОХРАНЯТСЯ, если сохранить просто файл проекта **copter.prt**. Для редактирования файла подпрограммы **motor.prt** надо либо отдельно его открывать, редактировать, сохранять, потом переоткрывать **copter.prt**, чтобы подгрузились изменения, либо, отредактировав внутри какой-либо субмодели подпрограмму, выбрать пункт меню **Файл → Сохранить страницу** (при этом запишется как раз **motor.prt**), но затем надо все равно закрыть **copter.prt** и открыть заново, чтобы изменения загрузились во всех 8 экземплярах типовой подпрограммы. В настоящей методике будем использовать первый способ – открывая заново **motor.prt** и внося в него модификации, сохраняя его и потом переоткрывая **copter.prt**, загружать всю модель снова, с учетом модификации внутри подпрограммы.

8.3. Модель винта

Апериодика первого порядка, размещенная в файле **motor.prt**, имитирует двигатель в первом приближении – на контроллер двигателя подается заданная угловая частота вращения, и с какой-то инерционностью (по умолчанию в блоке стоит постоянная времени = 1 с, что нас устраивает) блок выдает текущую частоту вращения ротора.

Винт в нашей модели октокоптера развивает только силу тяги, момент мы условились считать нулевым, так как в коптере используются парные ВМГ, не создающие прецессии и реактивного момента. Это соответствует действительности при равной частоте вращения моторов и винтов в паре.

Выбрав двигатель и его характеристику, можно по ней прикинуть коэффициент пропорциональности между квадратом угловой частоты вращения и силой тяги, развиваемой винтом. В нашем примере этот коэффициент получился равным 0.000202 Н × с^2, т. е. двигатель, который мы выбрали, развивает силу тяги в Ньютонах, равную

$$F_{Mi}(t) = 0.000202 \cdot \omega_{Mi}^2(t). \tag{8.3.1}$$

Здесь угловую частоту вращения надо подставлять в рад/с, а получаемая сила тяги измеряется в Ньютонах. При таком двигателе и винте, на частоте вращения 1500 об/мин = 157.08 рад/с получаем тягу около 5 Н. Для 8 двигателей суммарная тяга будет около 40 Н, что соответствует примерно 4 кг полезной тяги. Но у нас предполагается парная ВМГ, т. е. тяга будет в 1.6–1.9 раза выше, чем у одиночного винта… В общем, для учебной модели ожидаемая «равновесная» частота вращения, при которой коптер будет висеть в воздухе на 8 таких ВМГ и поддерживать свой вес (массу 14 кг), находится в районе 300 рад/с, что составляет 2900 об/мин. Для экономии места ограничимся 8 такими двигателями, а не 16 (характеристика снята с реального двигателя одной из известных фирм-производителей).

Внутри субмодели можно использовать сигналы, которые определены на верхнем уровне по отношению к схеме, с использованием служебного слова submodel.<имя_сигнала>. К таким сигналам относятся также и свойства, и параметры у субмодели. Мы будем обращаться к свойству Name через submodel.Name – для того чтобы получать имя субмодели (D1, D2, D3 и т.д.) и формировать имя сигнала, которое хотим считать или записать в базу сигналов.

Например, нам надо, вычислив силу тяги, записать ее в базу данных в сигнал <имя группы сигналов>_f, для этого внутри субмодели надо поставить блок типа «Запись в список сигналов», а в качестве интерпретируемого выражения в колонку **Формула** задать строку вида (с кавычками) **«{submodel.name}_f»** – тогда при ее интерпретации вместо {submodel.name} препроцессор проставит строку «**D1**» (без кавычек) для первой субмодели, **D2** для второй и т. д., и субмодель будет записывать сигнал в свою ячейку базы сигналов.

Откройте заново файл **motor.prt**, поставьте там блок типа «Параболическая функция», задав ему свойства, как на рис. 8.3.1, затем поставьте блок типа «Запись в список сигналов» и задайте там свойства, как на рис. 8.3.2, и соедините все линиями связи, как на рис. 8.3.3.

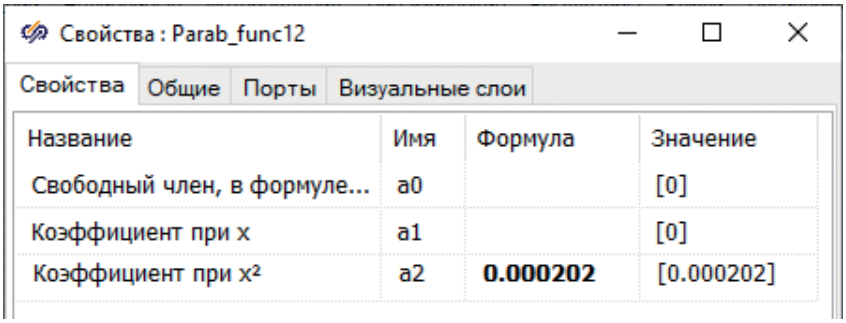


Рис. 8.3.1. Свойства параболической функции

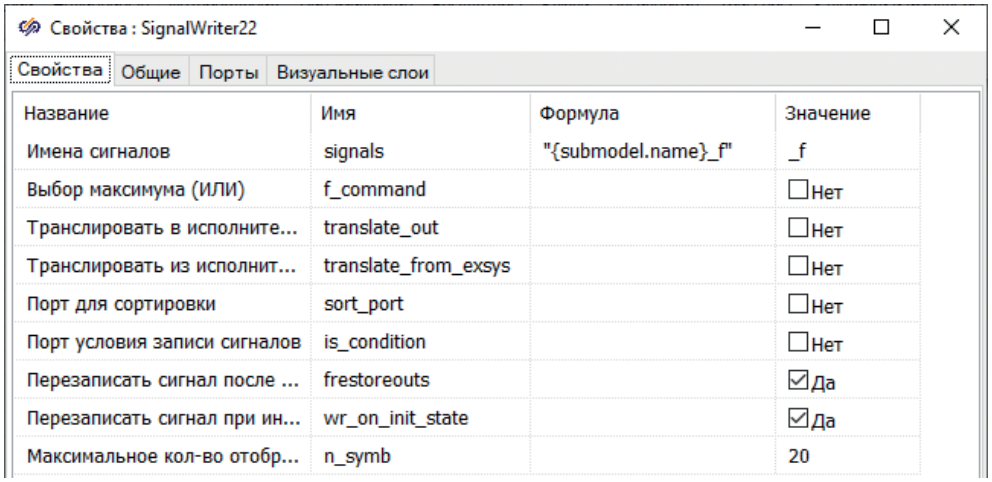


Рис. 8.3.2. Свойства записи в список сигналов

Теперь, если вы сохраните **motor.prt** и откроете заново **copter.prt**, там можно будет увидеть восемь модифицированных «внутренностей» субмоделей. Если попробовать инициализировать проект (при этом будет ошибка, так как мы еще не подключили входные порты этих субмоделей), то имена сигналов в каждой из субмоделей интерпретируются и заменятся, соответственно, на D1_f, D2_f, D3_f и т. д. до D8_f (на рис. 63 показан только хвостовик сигнала **_f** – вот на его месте будут уже имена реальных сигналов).

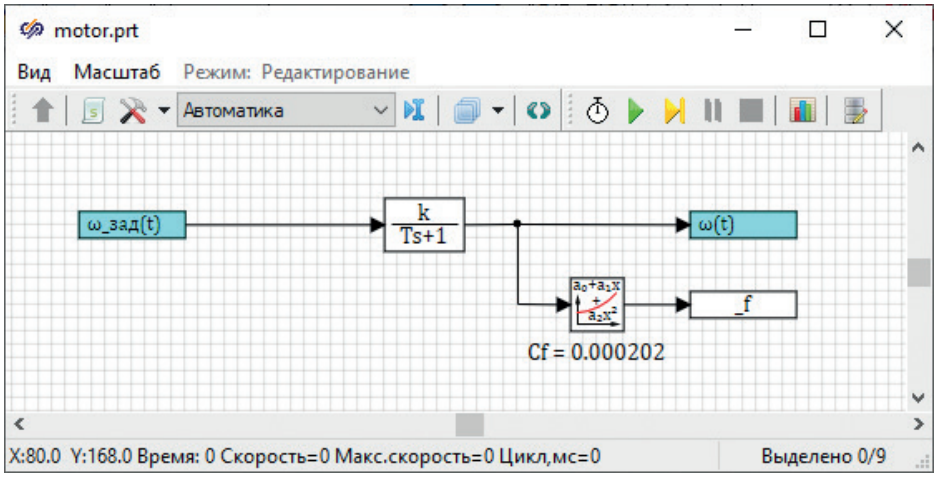


Рис. 8.3.3. Модифицированная субмодель ВМГ

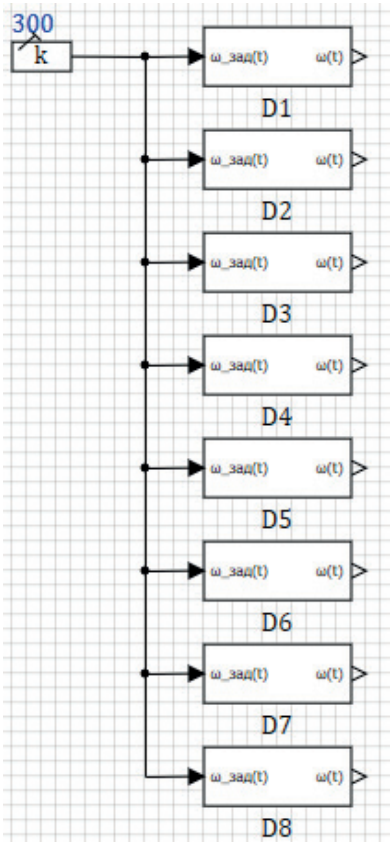


Рис. 8.3.4. Тестирование двигателей и коптера при равной частоте вращения моторов

Подайте для теста заданную частоту вращения порядка 200…300 рад/с (см. рис. 8.3.4) на каждую из субмоделей (можно сделать одним блоком типа «Константа») и посмотрите, как изменится переходной процесс на тестовых 10 с – теперь, с включенными двигателями, падение коптера уже будет не таким стремительным и прямолинейным, как было при выключенных двигателях. Коптер должен также завертеться немного по курсу, так как четные и нечетные двигатели, повернутые на 3° вокруг своих лучей, стоят на разном расстоянии от центра масс и, работая одинаково, в данном коптере будут закручивать его вокруг вертикальной оси поразному, с разной силой (точнее, моментом).

Отметим, что, хотя на схеме мы не соединили линиями связи или блоками «В память»–«Из памяти» результат расчета ВМГ и модель коптера, однако силы тяги винтов передаются в модель через базу сигналов – эти 8 сигналов, посчитанных каждый в своем экземпляре типовой подпрограммы **motor.prt**, записываются в базу сигналов, откуда считываются в уравнениях динамики октокоптера для расчета сил тяги и моментов сил, действующих на коптер. Это происходит с задержкой на 1 шаг расчета и, строго говоря, не совсем точно, но для целей моделирования задержка при передаче силы тяги в 1/1000 с не играет особой роли.

Тем более что в цепи управления коптером тоже есть некоторая инерционность, между формированием заданной частоты вращения на двигатель и получением нужной силы тяги, которая, скорее всего, сопоставима со временем порядка 0,001 с. Если бы нас это не устраивало, пришлось бы использовать механизм типа «В память» и «Из памяти» и здесь тоже.

8.4. Вывод результатов на графики, их оформление, анализ расчета

Если добавить блок типа «Временной график» (из библиотеки блоков «Вывод данных») к выходу интегратора, который вычисляет текущую высоту коптера и к линии, где вычисляется его вертикальная скорость (в системе **I** с перевернутой осью z, рис. 8.4.1), можно получить графики, показанные на рис. 8.4.2 и 8.4.3, для заданной частоты вращения соответственно 200 и 300 рад/с для двигателей ВМГ. Чтобы график у вас выглядел так же, как и на рисунках, надо сделать в его свойствах два входных порта, а также донастроить его внешний вид и выставить многошкальный режим. Подробнее об этом можно прочитать в справке по графику, а некоторые из настроек показаны на рис. 8.4.5.

Чтобы изменился заголовок окна, следует ввести описание графика, при этом после нажатия на **Ок** заголовок будет и над графиком, и в заголовке окна. Иногда удобнее оставить только в заголовке окна – для этого можно снова зайти в настройки графика, стереть там введенное ранее описание и выключить галочку **Копировать заголовок в окно** – тогда заголовок станет пустым, а в заголовке окна останется введенный ранее текст (так сделано для подготовки рис. 8.4.5).

Из графиков видно, что частоты вращения 200 рад/с недостаточно для взлета, коптер всеравно падает, а 300 рад/с – избыточно. Частота вращения 200 дает силу тяги около 8 Н, а частота 300 около 18 Н на одну ВМГ. При пересчете на весь коптер получаем силу тяги порядка 64 и 144 Н соответственно, что соответствует массам 6,5 и 14,7 кг. Поскольку масса коптера в модели составляет 14,0 кг, то частоты вращения 300 становится достаточно для подъема коптера вверх.

При этом и в том, и в другом случае за первые 10 с полета модель коптера претерпевает некоторый переходной процесс. Давайте разберем его. В начале расчета (при инициализации схемы) все винты инициализируются с нулевой частотой вращения, т. е. не создают никакой тяги. Коптер находится в начальной позиции на высоте 40 м. Начальные значения задаются в динамических блоках-интеграторах и в блоках типа «Апериодика 1-го порядка», которыми смоделированы ВМГ. Везде мы ставили 0, кроме одного интегратора, который интегрирует текущую высоту коптера – там задано 40 (м).

Из-за неработающих двигателей в первую секунду расчета коптер как бы «проваливается» вниз – вертикальная скорость достигает (в нашем случае)

величин около –1.5...–1.6 м/с. Потом, при выходе оборотов двигателей на заданную частоту вращения и создании ими тяги коптер замедляет падение, и в случае, когда силы тяги достаточно, он останавливается и начинает лететь вверх (где-то на 4...5-й секунде моделирования). Если силы тяги недостаточно – падение продолжается, но с уменьшенной скоростью, порядка –1.2 м/с. Отметим, что полученные абсолютные величины скорости – на данном этапе набора схемы – еще далеки от реальности, поскольку коэффициенты трения о воздух C_D перед квадратом линейной и угловой скоростей в уравнениях мы поставили больше, чем они будут в действительности – для большей численной устойчивости расчетной схемы. Если вы попробуете их поменять и вместо –50 там поставите –10 или –1, то получаемые вертикальные скорости будут другими, более похожими на правду. Какой именно там выставить коэффициент – это дело экспериментальной отработки коптера и корректировки его модели. Для целей обучения пока что величина не принципиальна, но следует иметь в виду условность модели в этом месте. Можно сказать, что модель коптера сейчас «летает» в сильно вязком воздухе, который хорошо «тормозит» или демпфирует любые возмущения.

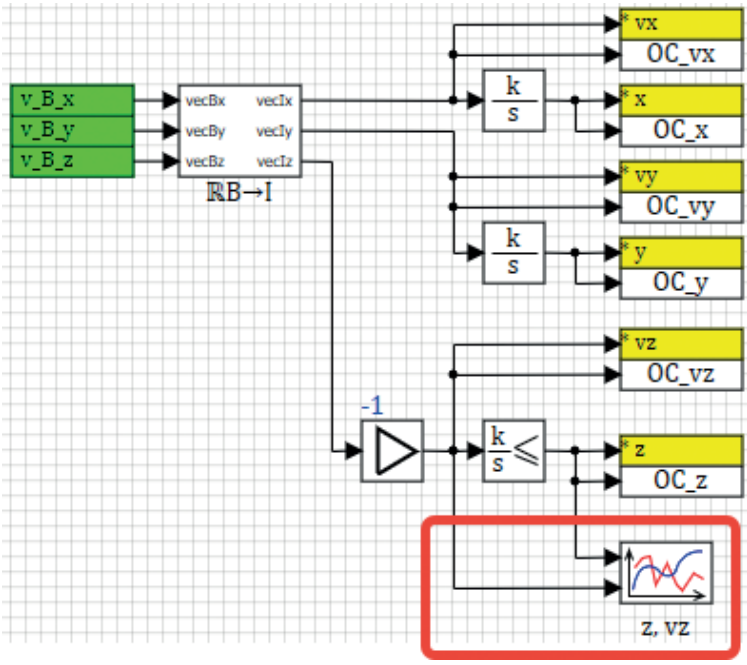


Рис. 8.4.1. Добавление графика для анализа результатов расчета

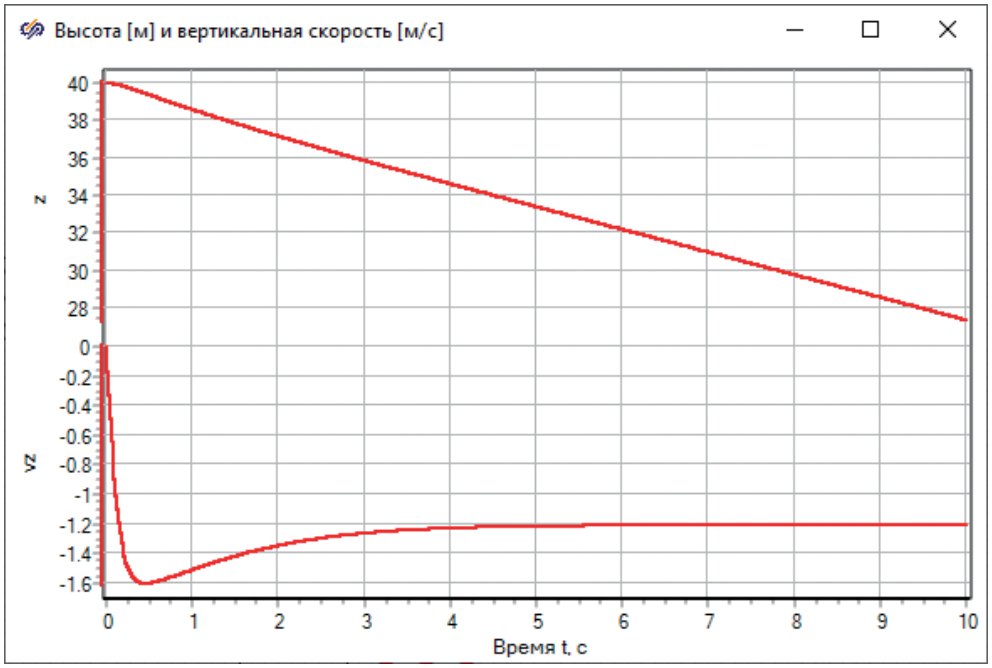


Рис. 8.4.2. Высота полета и вертикальная скорость при 200 рад/с

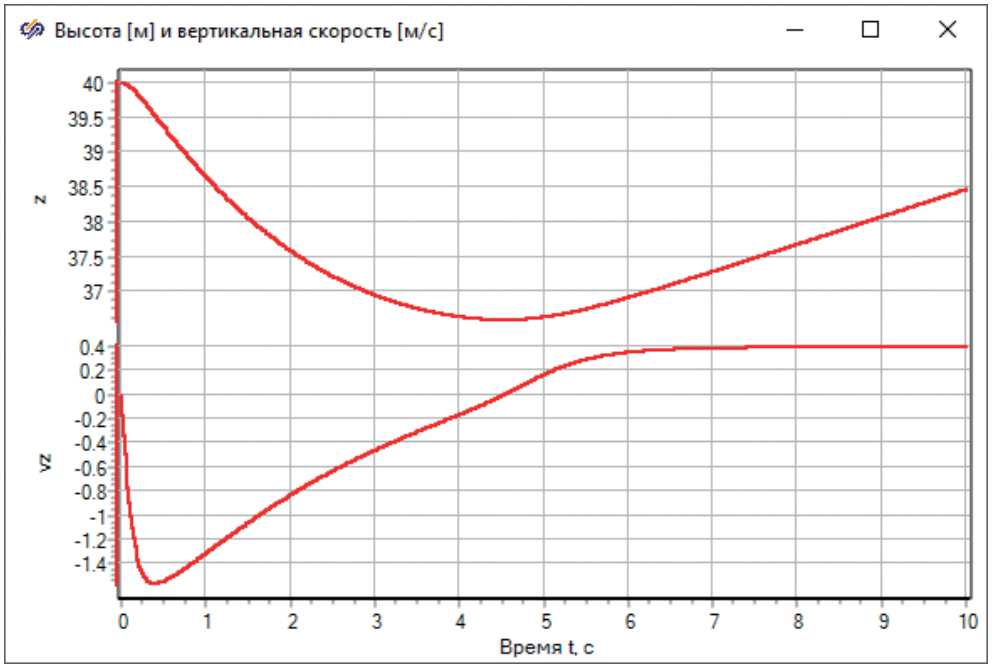


Рис. 8.4.3. Высота полета и вертикальная скорость при 300 рад/с

Для того чтобы иметь более полную картину того, что происходит в модели, рекомендуется основные расчетные параметры модели вывести на временные графики, при этом однотипные величины можно выводить на одни и те же графики, а также строить фазовые портреты – в SimInTech можно строить как двухмерные, так и трехмерные графики фазовых координат. Для примера, на рис. 8.4.4 показан набор графиков для модели, к которому мы придем к концу обучающей методики.

На рис. 8.4.4 представлены следующие графики и фазовые портреты: график фазовой траектории полета в плоскости x–y (вид сверху на коптер) и x–z (вид сбоку), трехмерный график полета с выводом как траектории полета, так и заданной точки в пространстве. Графики текущих углов ориентации коптера, текущих координат, график оборотов двигателей, линейных скоростей и ускорений. Сигналы «В память» и «Из памяти», а также сигналы базы данных позволяют сделать вывод результатов на графики удобным, собранным в одном месте, а не разбросанным по всей модели объекта.

В настоящий момент набора модели коптер может «летать» только вверх-вниз, без системы стабилизации любое отклонение от вертикального положения приведет к его нестабильному полету и переворотам, поэтому графики пока что нам не нужны в таком объеме – сначала предстоит сделать модель полетного контроллера, включающего в себя регулятор высоты и регуляторы положения коптера по направлениям в пространстве.

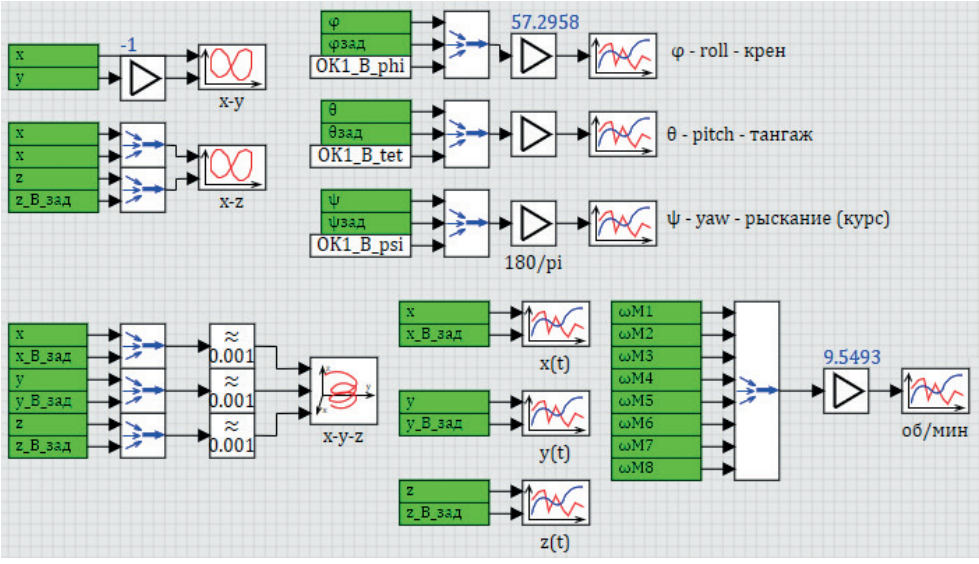


Рис. 8.4.4. Набор графиков для анализа модели октокоптера

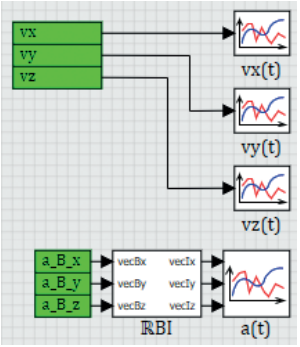


Рис. 8.4.4. *Окончание*

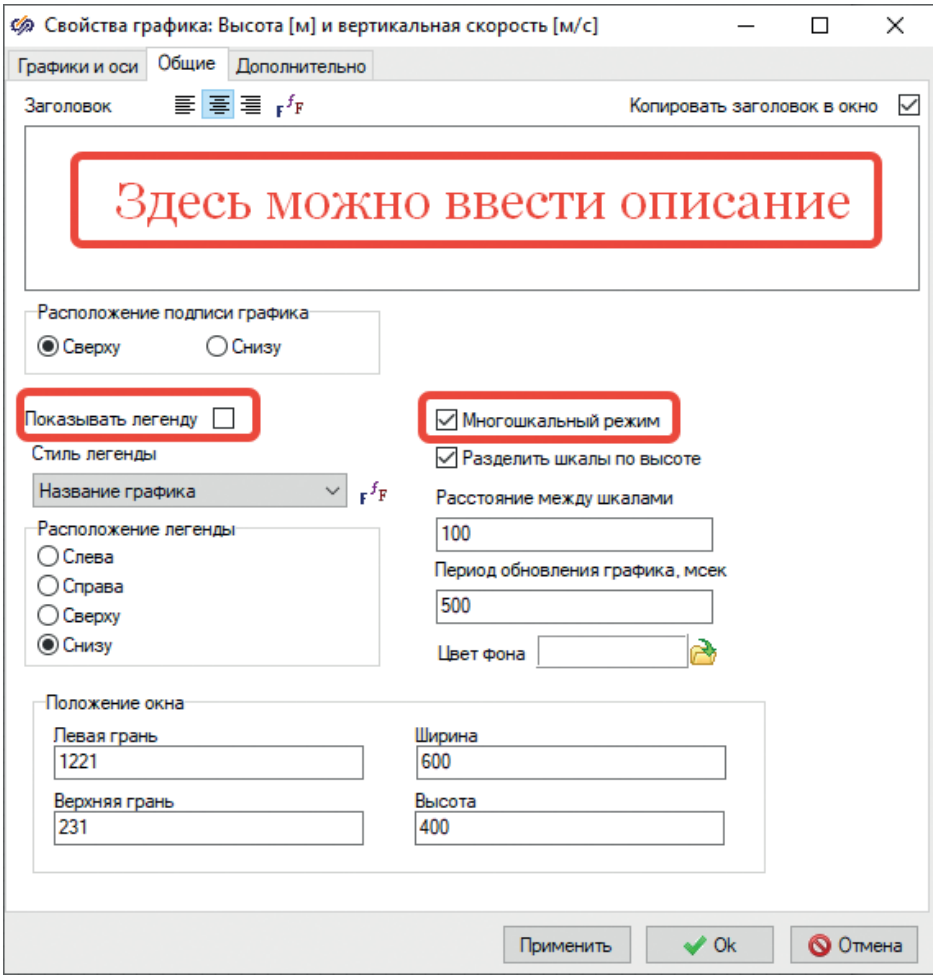


Рис. 8.4.5. Некоторые из настроек отображения графика

Модель полетного контроллера

Полетный контроллер (или регуляторы), представленный в настоящей методике, не претендует на верность исполнения для реального октокоптера (данная модель не была испытана ни на одном реальном аппарате), а служит лишь введением в тему, и это одни из наиболее простых регуляторов, которые можно придумать для стабилизации и управления коптером. Однако для целей обучения он более чем подходит и справляется с управлением моделью коптера – держит коптер в равновесии и позволяет управлять им с пульта управления, задавая новые координаты. Подробнее с описанием данного полетного контроллера и с подходами к его построению можно ознакомиться в [1].

Опишем кратко здесь основные идеи регулирования ориентацией и управления полетом, заложенные в контроллер. В конечном счете полетный контроллер должен выработать 8 сигналов – заданных частот вращения на каждый из двигателей, причем таким образом, чтобы коптер:

А) оставался в стабильном состоянии (угловые скорости и углы ориентации должны стремиться к нулю, а в равновесном состоянии должны быть равны нулю);

Б) двигался в заданном направлении, причем задание задает оператор коптера координатами (рассогласование между текущими координатами коптера и заданными должны стремиться тоже к нулю). Ручное управление и пилотирование оператором с пульта типа «джойстик» не рассматривается, но может быть доработано и в этой модели.

Задача А является наивысшим приоритетом, Б – вторична, поэтому в реальных полетных контроллерах есть разграничение приоритета между А и Б. Также, на выполнение каждой из задач тратится некоторая «возможность» коптера стабилизироваться и двигаться в пространстве, и никогда этот ресурс не должен тратиться на 100 % – даже при отсутствии возмущений, как правило, закладывают на выполнение задачи Б от 30 до 50 % от запаса «управляемости». В настоящей методике это не рассматривается и не реализуется.

Таким образом, полетный контроллер можно представить в виде суммы 6 регуляторов, каждый из которых работает по одному из 6 каналов управления – крен, тангаж, рыскание (курс), высота полета z, координаты x и y. При этом регулятор должен следить не только за самими углами и координатами, но и за скоростями по направлениям, а также за ускорениями. В модели мы не будем строить и моделировать по-настоящему датчики коптера, а будем пользоваться теми расчетными величинами, которые предоставляет модель.

По каждому из 6 каналов управления существует свой вектор верного и оптимального «микширования» управляющих команд на двигатели. Например, если мы хотим, чтобы коптер летел вверх, то обороты всех двигателей надо увеличивать, если вниз, то уменьшать (это очевидно). Если требуется вращение вокруг оси $x_в$, см. рис. 2.1.1, то вектор микширования уже несколько сложнее – двигатели 1 и 5 трогать не надо (они не создают вращающего момента вокруг этой оси), а надо в противоположные стороны управлять двигателями 3 и 7, а также парами двигателей № 2, 4 и 6, 8. Аналогично по другим осям. Теория оптимального управления дает ответ на вопрос, как именно (с какими коэффициентами) для данного геометрического расположения двигателей следует управлять двигателями для создания воздействия по данному направлению. Представим ниже ответ на данный вопрос для нашей геометрии коптера и для 6 каналов регулирования:

X: [0 , –191.9265 , 191.9265 , –191.9265 , 0 , 191.9265 , –191.9265 , 191.9265].
Y: [–191.9265 , 191.9265 , 0 , –191.9265 , 191.9265 , –191.9265 , 0 , 191.9265].
Z: [–5.0292 , –7.1124 , –5.0292 , –7.1124 , –5.0292 , –7.1124 , –5.0292 , –7.1124].
Крен φ: [0 , –5.0292 , –7.1124 , –5.0292 , 0 , 5.0292 , 7.1124 , 5.0292].
Тангаж θ: [7.1124 , 5.0292 , 0 , –5.0292 , –7.1124 , –5.0292 , 0 , 5.0292].
Курс ψ: [–67.8562 , 67.8562 , –67.8562 , 67.8562 , –67.8562 , 67.8562 , –67.8562 , 67.8562].

Этот набор чисел – коэффициентов для двигателей – говорит о том, как именно следует синхронно изменять частоту вращения каждой из 8 ВМГ (точнее, квадрат частоты вращения), для того чтобы всеми двигателями вместе создавать управляющее воздействие на коптер исключительно по заданному направлению, не затрагивая другие направления регулирования. Например, по высоте: если квадрат частоты вращения каждой из ВМГ номер 1,3,5,7 уменьшить на 5.0292, а 2,4,6,8 уменьшить на 7.1124, то создастся «единичное» положительное управляющее воздействие по оси z (она направлена вниз), и коптер начнет двигаться по этому направлению (падать вниз, если он был ранее в равновесии). При этом по другим направлениям никакого управляющего воздействия не создастся (!) – это важно для понимания процессов в полетном контроллере. Аналогично по другим каналам управления.

Из абсолютных значений приведенных чисел видно, что коптер «плохо» управляем по направлениям X и Y, а также по курсу ψ. Хотя по курсу немного лучше, чем по горизонтальным направлениям. То есть для создания единичного управляющего воздействия по этим трем каналам (x, y, ψ) требуется сильно изменять частоты вращения двигателей, в отличие от других каналов управления. Это связано с расположением векторов сил ВМГ – из-за того, что они все направлены почти вертикально вверх, а реактивный момент ВМГ в нашем коптере отсутствует, то, изменяя их по абсолютной величине, почти нет возможности поворачивать коптер вокруг вертикальной оси или двигать его влево-вправо-вперед-назад (в связанной с ним системе координат). Если бы не введенный в конструкцию малый угол гамма, то коптер был бы вообще неуправляем по курсу и по направлениям. Для курсовой устойчивости этой – относительно плохой – способности управления коптером будет достаточно, так как в полете не будет сильных возмущающих воздействий, закручивающих коптер вокруг вертикальной оси. А вот по направлениям мы воспользуемся другой идеей – для созда-

ния движения в горизонтальной плоскости будем поворачивать коптер вокруг осей x и y, чтобы перенаправлять вертикальную силу тяги в нужную сторону.

Наилучшей управляемостью, судя по приведенным числам, коптер обладает по каналу управления Z, т. е. по высоте, – и это неудивительно, так как векторы сил тяги винтов направлены практически вдоль этого направления, и все 8 ВМГ действуют заодно.

Для реализации контроллера лучше всего подходит стратегия его «изготовления» по шагам – сделать сначала один регулятор по одному из направлений, посмотреть, как он работает, выполнить базовую настройку и дальше приступить к следующему направлению. Если сделать сразу все, то очень велика вероятность, что «что-то пойдет не так», и отладить потом сложный регулятор будет гораздо труднее, чем по частям.

9.1. Регулятор высоты в нулевом приближении

Воспользуемся тем, что у нас пока отсутствуют боковые возмущения, и коптер при нулевых начальных условиях, в принципе, стабилен в пространстве – летает только вверх-вниз (при одинаковом синхронном управлении двигателями), и попробуем сделать только регулятор высоты.

В качестве основы для регулятора возьмем классический ПИД-регулятор (пропорционально-интегрально-дифференциальный), т. е. входами в этот регулятор будет как сама координата (точнее рассогласование между заданной и текущей координатой z), так и вертикальная скорость, и еще интеграл от рассогласования.

Общая схема, или идея регулятора высоты, представлена на рис. 9.1.1.

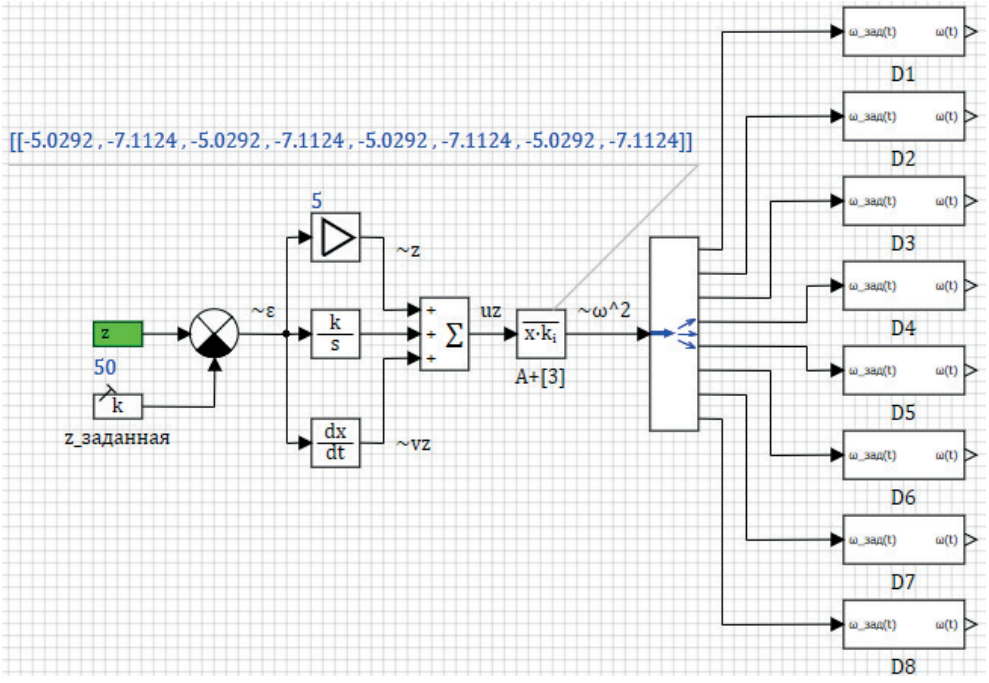


Рис. 9.1.1. Идея регулятора высоты как ПИД-регулятора

Давайте детально рассмотрим, что здесь происходит. На вход регулятору подается рассогласование между текущей (измеренной) высотой полета z и заданной высотой полета (предположим, что задана высота 50 м). Полученное рассогласование (в идеале это малое отклонение ε) регулятор должен сводить к нулю. Далее оно подается на пропорциональную ветку (в блок типа «Усилитель»), в интегрирующую ветку и в дифференцирующую ветку, результат трех вычислений суммируется, вырабатывая управляющее воздействие uz(t) по данному каналу управления (по каналу z). Управляющее воздействие подается в блок типа «Размножитель», в котором заданы коэффициенты из 3-й колонки псевдообратной матрицы A^+ (подробнее про матрицу см. [1] или [3]), которые показывают, с какой пропорциональностью надо управляющий сигнал подать на каждый из 8 двигателей. И потом при помощи блока типа «Демультиплексор» каждый из элементов полученного вектора из 8 сигналов подается на свой двигатель как заданная частота вращения. Такая схема довольно проста и **обладает рядом недостатков**, но позволяет в простом виде пояснить суть регулирования. Также в ней **есть ошибка**.

Вообще, одну и ту же задачу, как правило, можно выполнить несколькими способами, и здесь рассмотрен всего лишь один из вариантов, не претендующих на истину в последней инстанции. Недостаток схемы в следующем – псевдообратная матрица говорит не о заданной частоте вращения для двигателей, а всего лишь об отклонениях квадрата частоты вращения двигателей (от некой базовой частоты), нужных для того, чтобы скомпенсировать возникающее отклонение от равновесия (от нуля на входе в регулятор). Поэтому ошибка данной схемы в том, что мы сигнал после умножения на коэффициенты матрицы, пропорциональный квадрату частоты вращения, выдаем как саму заданную частоту вращения – что, вообще говоря, неверно, и там нужно еще поставить блок, вычисляющий квадратный корень из сигнала. Недостаток схемы заключается в том, что полетный контроллер должен задавать «базовые» частоты вращения по каждому из двигателей, исходя из полетной массы коптера и характеристики двигателей, а регулятор высоты – уже только выдавать отклонения от этих базовых величин, но для нулевого приближения данная схема тем не менее годится, потому что она тоже работает.

Еще одним из недостатков схемы является то, что заданная и измеренная высота полета сейчас берется для неподвижной системы **I**, а управляющее воздействие регулятор вырабатывает для системы **B**. То есть в наклонных положениях регулятор будет работать неверно – и чем больше наклон, тем сильнее будет ошибка… Это устраним немного позднее.

Давайте последовательно наберем схему, представленную на рис. 9.1.1, и протестируем данную идею на практике путем прямого моделирования. Нам потребуются блоки: «Из памяти» (вкладка **Субструктуры**), «Константа» (вкладка **Источники**), «Сумматор», «Сравнивающее устройство» и «Усилитель» (**Операторы**), «Интегратор» и «Производная» (**Динамические**), «Размножитель» и «Демультиплексор» (**Векторные**).

В блоке «Из памяти» прописываем сигнал **z** (ранее его создавали – это выход одного из уравнений динамики). Отметим, что здесь высота будет

положительной величиной, так как ось мы там перевернули вверх. В константе прописываем значение **50**. В усилительном звене оставьте пока коэффициент **1**, равно как и в интеграторе все оставляем по умолчанию – коэффициент **единичный**, а начальные условия **нулевые**. В сумматоре надо прописать вектор из трех единиц, это можно сделать кратко при помощи символа решетки, как **3#1** в колонке **Формула** (см. рис. 9.1.2). В размножителе аккуратно напишите 8 коэффициентов (см. рис. 9.1.3), а в демультиплексоре **8#1** (см. рис. 9.1.4).

Соедините все линиями связи.

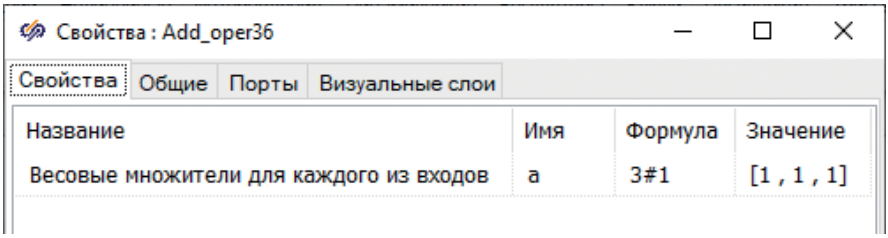


Рис. 9.1.2. Свойства сумматора, краткая запись регулярного вектора

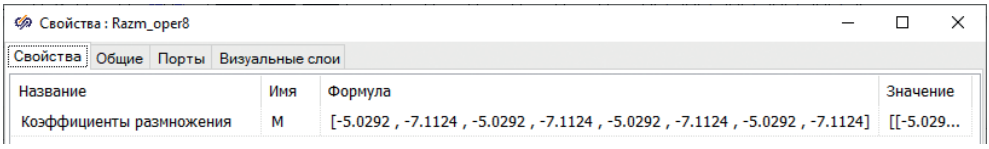


Рис. 9.1.3. Свойства размножителя, 3-я колонка матрицы A+

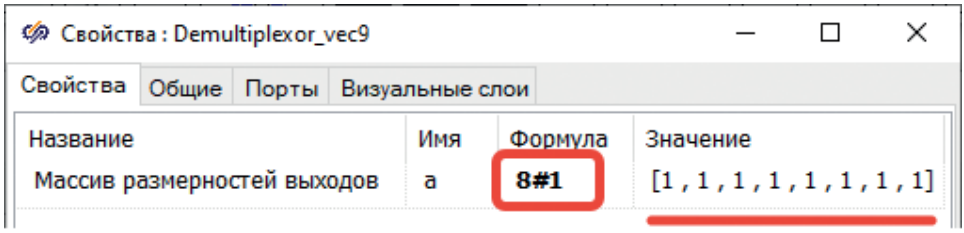


Рис. 9.1.4. Свойства демультиплексора, еще одна краткая запись регулярного вектора

После соединения всех блоков линиями связи, как на рис. 9.1.1 (в усилителе коэффициент пока единичный!), выставьте в параметрах расчета (главное меню, **Расчет → Параметры расчета**) все, как показано на рис. 9.1.5, – давайте зададим метод расчета **Адаптивный 4** как более подходящий для такого рода задач, а конечное время поставим **600** с (10 мин полета). Минимум и максимум по шагу расчета достаточно будет поставить одну десятитысячную и одну сотую секунды.

Параметры проекта: c:\copter\copter\octocopter.prt слой: Автома...

Параметры расчёта | Синхронизация | Рестарт | База данных | Вид

Название	Имя	Формула	Значение
Основные параметры			
Минимальный шаг	hmin		**0.0001**
Максимальный шаг	hmax		**0.01**
Начальный шаг интегрирования ...	startstep		0
Метод интегрирования	intmet		**Адаптивный 4**
Начальное время расчёта	starttime		0
Конечное время расчёта	endtime		**600**
Относительная ошибка	relerr		0.0001
Абсолютная ошибка	abserr		1E-6

Рис. 9.1.5. Параметры расчета для теста регулятора высоты

После этих перенастроек, запустив схему на расчет, мы увидим результат работы регулятора и модели как неустойчивой системы. График высоты и вертикальной скорости (график делали ранее) должен получиться примерно, как показано на рис. 9.1.6.

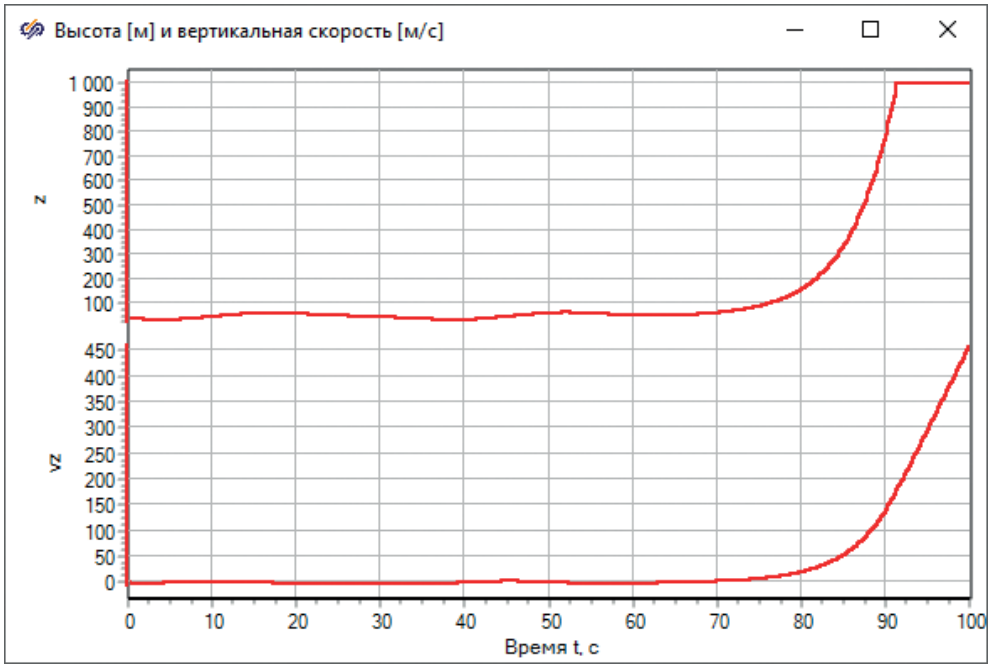


Рис. 9.1.6. Неустойчивый регулятор

В набранном регуляторе мы еще не подбирали коэффициенты ветвей, и (нетрудно видеть) проставленные единичные значения дали нам неустойчивый регулятор. Как правило, коэффициенты интегрирующей ветви должны быть на 1-2 порядка меньше, чем у пропорциональной. Потому что интеграл от рассогласования будет гораздо большей величиной, чем само рассогласование. Рассмотрим один из простых способов «отладки» регулятора. Если мы не знаем точно, что приводит к выходу из равновесия – проще всего «повесить» график на три составляющих регулятора и посмотреть, какая из них вносит самый большой вклад. Разместите рядом с регулятором временной график, задайте ему три входных порта и соедините с тремя ветвями регулятора (рис. 9.1.7), а конечное время расчета задайте в параметрах расчета (временно) = 80 с, поскольку примерно на этой секунде система идет вразнос. Результат расчета сравните с рис. 9.1.8.

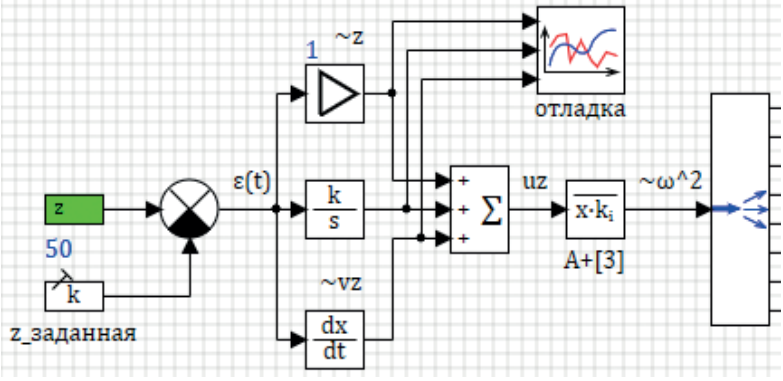


Рис. 9.1.7. Отладочный график для регулятора

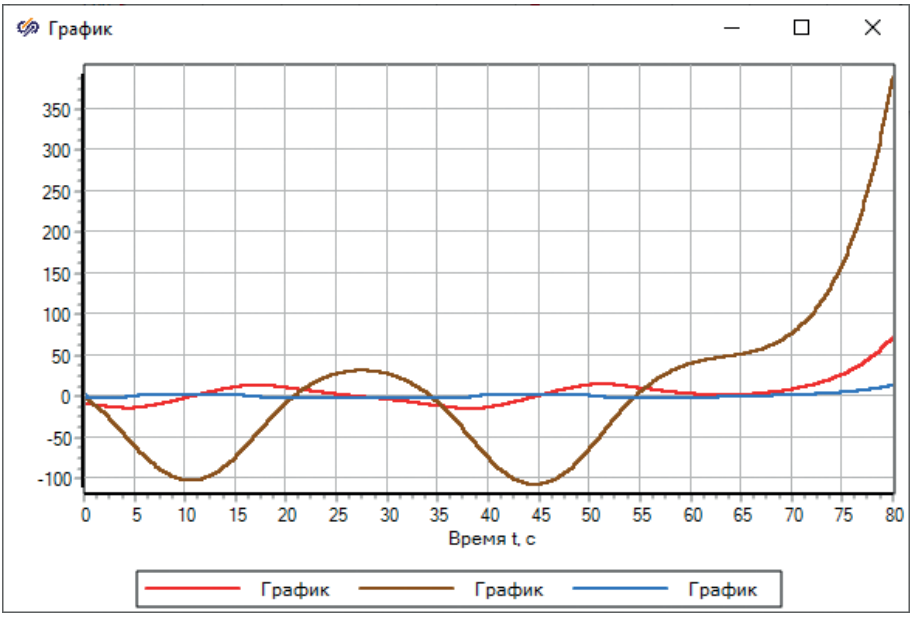


Рис. 9.1.8. Результат отладки

Из рис. 9.1.8 видно, что наибольший вклад в итоговую сумму регулятора вносит вторая ветвь, т. е. интегрирующая. А пропорциональный сигнал слишком мал, чтобы вернуть систему к равновесию. Возможный вариант для исправления – либо увеличить раз в 5 коэффициент усиления пропорциональной ветви (П-ветви), либо снизить коэффициент усиления на интеграторе (И-ветви) регулятора. Дифференцирующая ветвь пока мала и нас в «нулевом приближении» устраивает…

Если увеличить коэффициент усиления с 1 до **5** на П-ветви, переходной процесс получится, как представлено на рис. 9.1.9. Это уже гораздо лучше, но есть некоторое перерегулирование и потом незатухающие колебания довольно приличной величины (скорость меняется от плюс до минус 0.5 м/с). Давайте сделаем еще одну итерацию – вернем коэффициент усиления снова в 1, а коэффициент усиления в интеграторе сделаем в 10 раз меньше, равным **0.1**. Результат моделирования представлен на рис. 9.1.10.

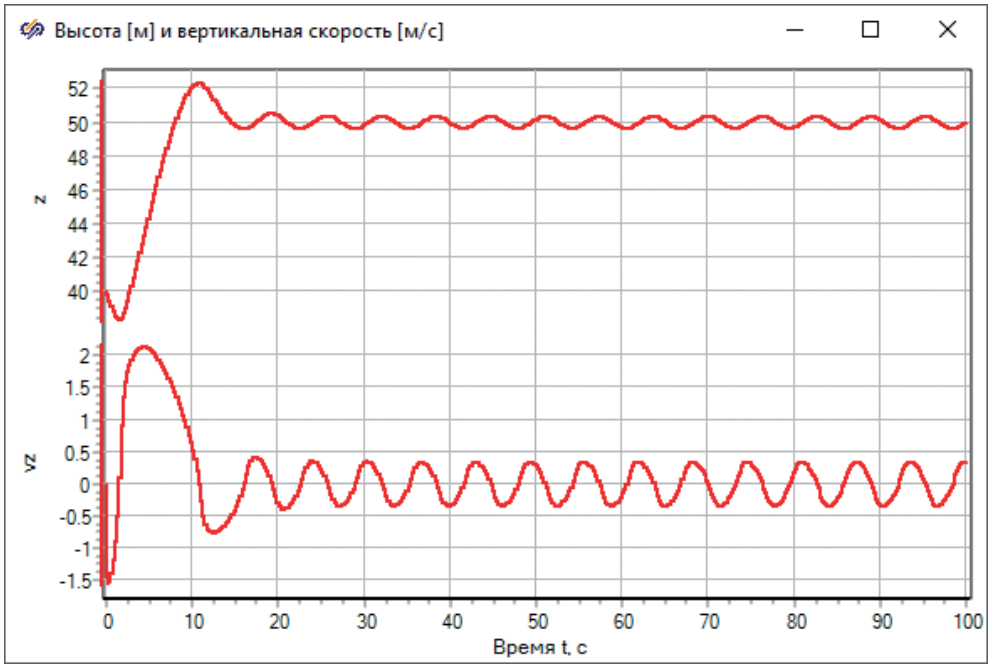


Рис. 9.1.9. Устойчивый регулятор, но «плохой» (Кп = 5, Ки = 1, Кд = 1)

Видно, что колебания по скорости стали гораздо меньше (при выходе на устойчивое висение на заданной высоте), но величина перерегулирования стала больше, а также сильнее стал первоначальный «провал» по высоте. Дело в том, что коптер у нас стартует с 40 м высоты, но с «выключенными» двигателями, и им требуется некоторое время на разгон (работает инерционное звено 1-го порядка в модели ВМГ), и регулятору также требуется некоторое время, чтобы интегратор накопил равновесное значение для частоты вращения двигателей. А мы его коэффициент снизили.

Что можно сделать для улучшения процесса? Верните снова 300 или 600 с как конечное время расчета и промоделируйте все 5–10 мин, при этом на регуляторе на-

копится «равновесное» значение на И-ветви – это будет порядка –47 (см. рис. 9.1.11). Задайте его как начальное значение в интеграторе регулятора (рис. 9.1.12).

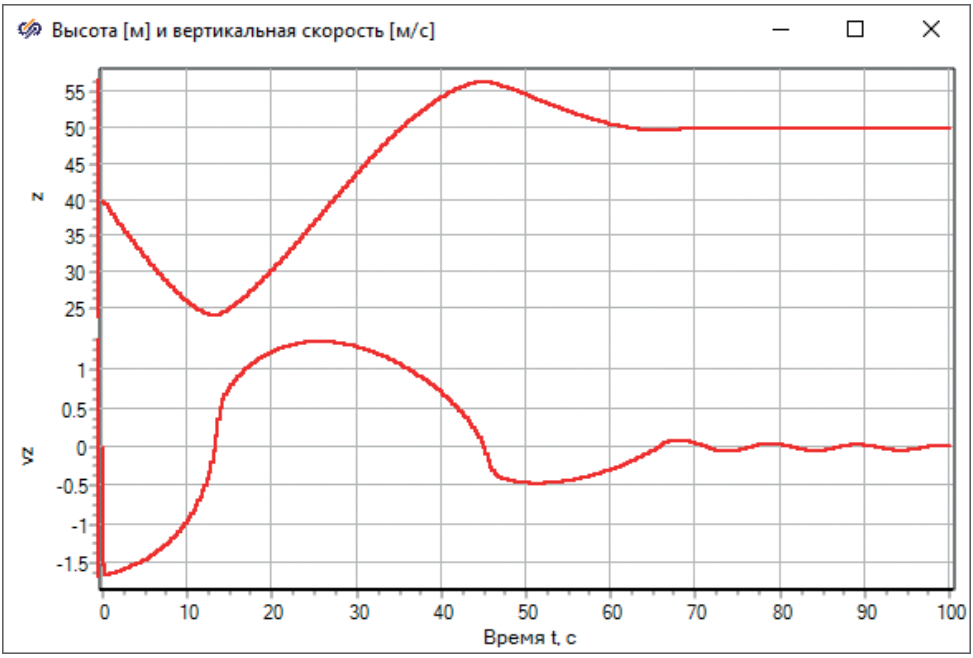


Рис. 9.1.10. Устойчивый регулятор, «средний» (Кп = 1, Ки = 0.1, Кд = 1)

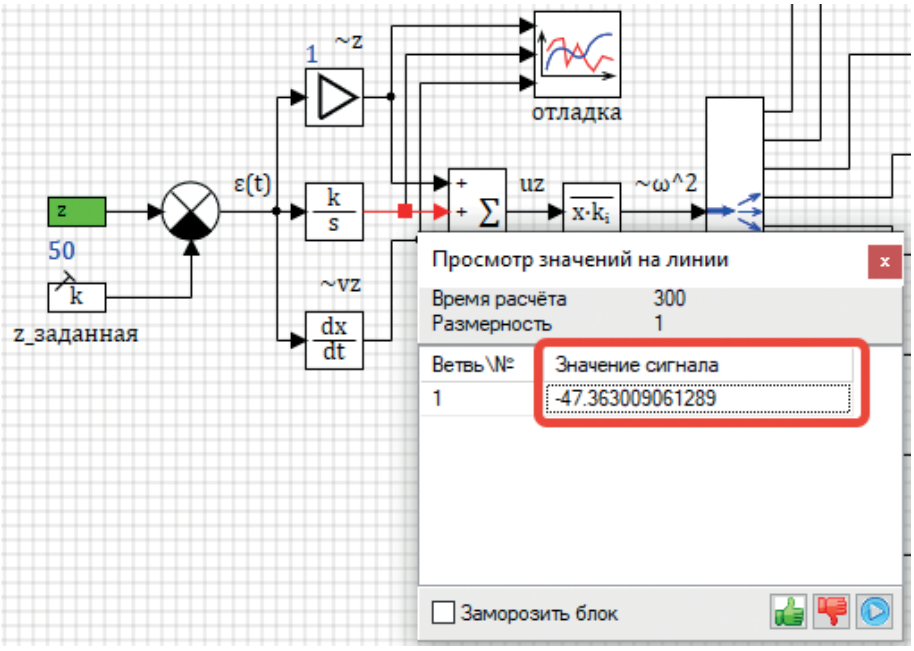


Рис. 9.1.11. Равновесное значение И-ветви в регуляторе

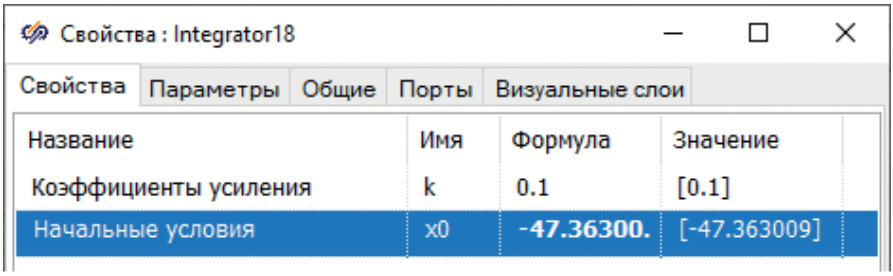


Рис. 9.1.12. Исправление регулятора высоты для нашего частного случая

После задания начального значения для интегратора, он начнет считать уже с равновесного значения и начальный провал вниз будет меньше. Хотя, так как модели двигателей все равно начинают считать с нулевой частоты вращения (текущей), – пролет вниз все же будет. Результат представлен на рис. 9.1.13.

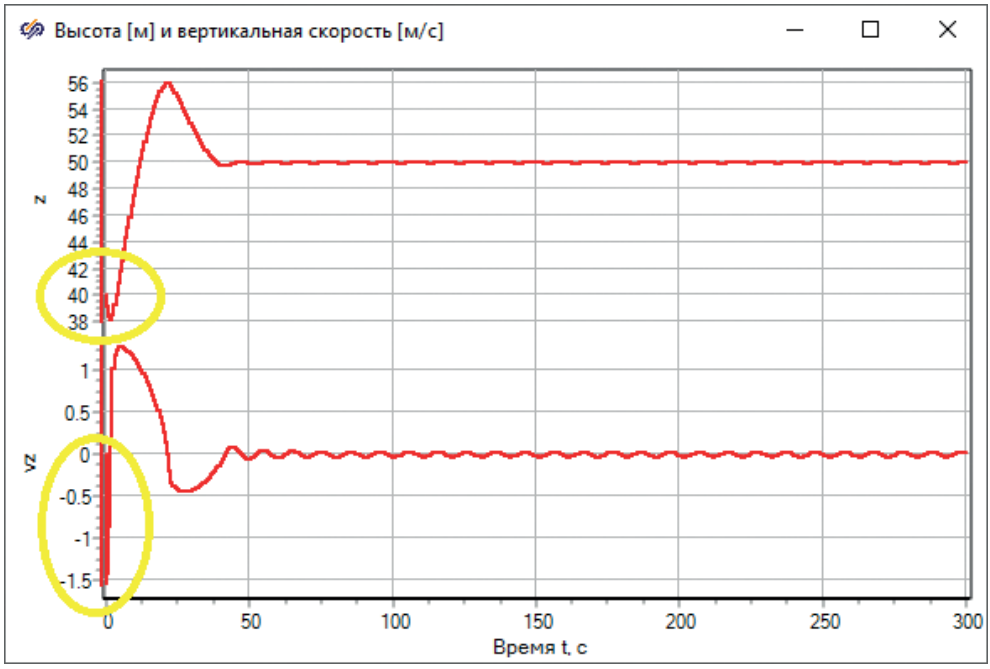


Рис. 9.1.13. Переходной процесс с ненулевыми начальными условиями в И-ветви (Кп = 1, Ки = 0.1, Кд = 1, н. у. = –47 в интеграторе)

Для полного устранения начального «неверного» переходного процесса и некоторого полета вниз на первых секундах следует еще изменить и частоту вращения в стандартной подпрограмме мотора на 300 рад/с вместо 0. Величина 300 – это значение примерно посередине, между равновесными частотами, которые вырабатывает регулятор. Конечный вид переходного процесса (уже достаточно «хорошего») представлен на рис 9.1.14. Видно, что коптер начи-

нает сразу с движения вверх, так как мы задали частоту вращения, немного большую, чем нужно для равновесного висения. Дальнейшие улучшения регулятора (в плане подбора коэффициентов) пока что не имеют смысла, так как в модели коптера мы задали слишком «вязкий» воздух – и вначале надо до конца исправить модель объекта, а потом уже подбирать под него параметры регулятора, иначе мы будем дважды выполнять одну и ту же работу. Пока что остановимся на достигнутом результате, а вообще, здесь требуется, конечно, еще улучшать качество переходного процесса – можно попробовать снизить еще усиление в И-ветви или «поиграть» общим коэффициентом усиления, а также проанализировать, что дает Д-ветвь, – скорее всего, при помощи нее удастся снизить максимумы по скорости полета.

Но для начального приближения мы получили устойчивый хороший результат, который позволит нам двигаться дальше, – теперь коптер «умеет» держать нужную высоту и не улетает ни вверх, ни вниз. Если, конечно, другие его фазовые координаты стабильны.

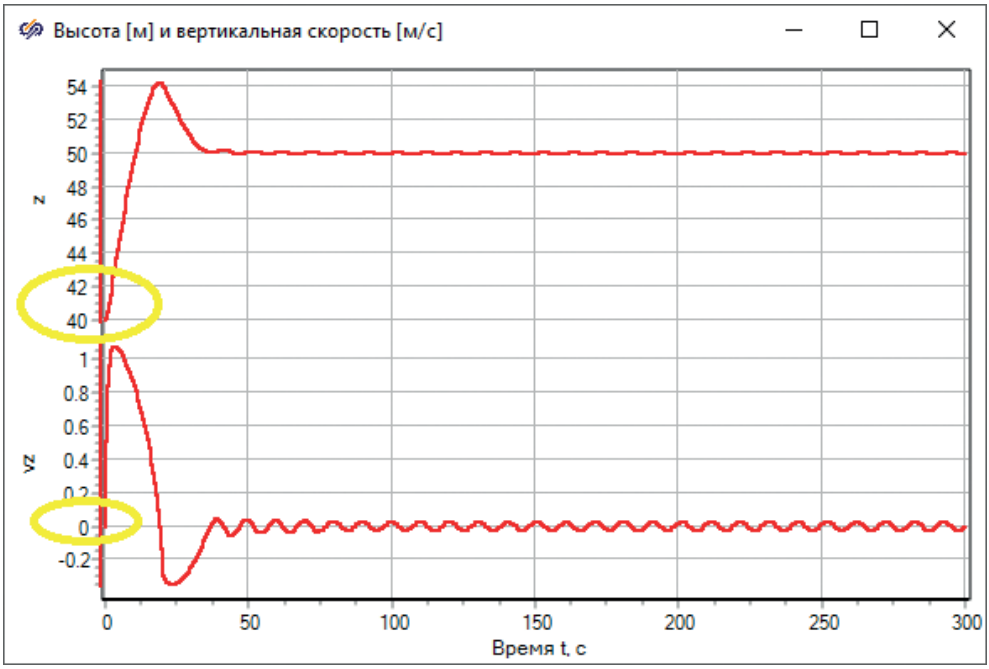


Рис. 9.1.14. Переходной процесс с ненулевыми начальными условиями в И-ветви и в модели ВМГ (Кп = 1, Ки = 0.1, Кд = 1, н. у. = –47 в интеграторе, н. у. = 300 в апериодике 1-го порядка, моделирующей ВМГ)

Примечание: для задания начальных условий в подпрограмме ВМГ следует закрыть проект **copter.prt**, открыть отдельно файл **motor.prt** из папки **sub**, задать там величину **300** вместо 0 в блоке апериодики 1-го порядка, сохранить файл и закрыть его, потом заново открыть модель коптера и запустить на расчет. Результат должен совпасть с рис. 9.1.14.

9.2. Анализ влияния регулятора высоты на курс

Если внимательно посмотреть на 3-ю колонку псевдообратной матрицы, а также вспомнить конструкцию моделируемого октокоптера (8 винтов, нечетные находятся на расстоянии 1.414 м от центра масс, четные на расстоянии 1 м, и все довернуты на 3° вокруг своих лучей – четные в одну сторону, нечетные в другую), можно увидеть важную деталь. Полученные коэффициенты матрицы учитывают эту особенность конструкции, и все нечетные коэффициенты меньше четных по абсолютной величине, причем тоже в 1.414 раза (точнее, в $\sqrt{2}$): [-5.0292 , -7.1124 , -5.0292 , -7.1124 , -5.0292 , -7.1124 , -5.0292 , -7.1124]. Это сделано для того (вернее, математически так получилось при вычислении псевдообратной матрицы), чтобы поворотный суммарный момент по другим каналам управления оставался нулевым при любой работе регулятора высоты. В нашем же случае… так как мы а) во-первых, допустили ошибку в регуляторе, б) во-вторых, запускаем коптер с двигателями, которые работают на одной и той же частоте вращения 300 в начале – они все же создают поворотный момент вокруг вертикальной оси, и коптер у нас, кроме того, что он уравновешивается по высоте, начинает еще и вращаться вокруг своей вертикальной оси… Увидим это, построив график угловой скорости ω_B_z(t) (рис. 9.2.1). Примечание: отметим (а вы проверьте) что другие переменные состояния коптера должны быть нулевыми, так как по другим каналам управления пока что никаких влияний на коптер не должно быть!! Скорости **v_B_x**, **v_B_y** и угловые скорости ω**_B_x**, ω**_B_y должны быть нулевыми** на протяжении всего переходного процесса. Если это не так, значит, где-то вы допустили ошибку, и ее надо найти и устранить.

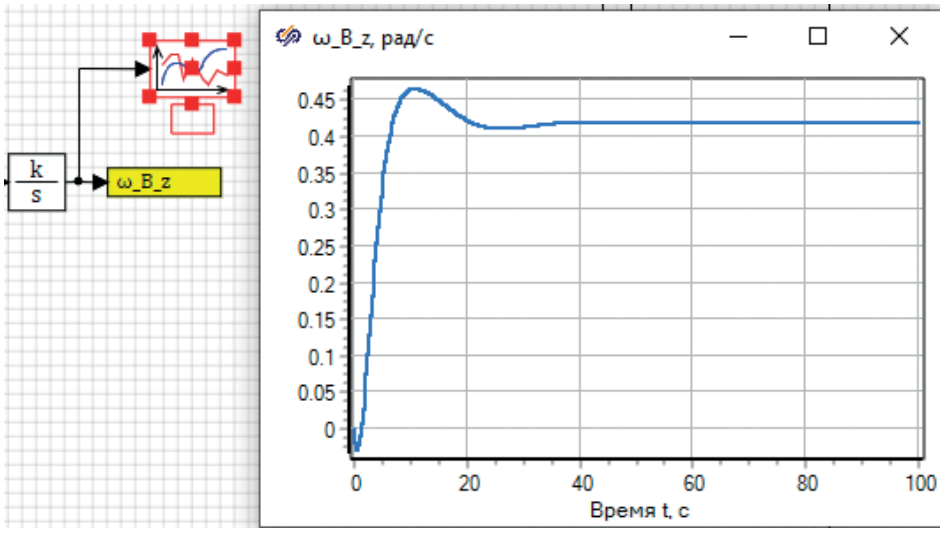


Рис. 9.2.1. «Паразитное» влияние регулятора высоты на курс коптера

То есть коптер набрал угловую скорость вращения вокруг вертикальной оси и так с ней и летит, продолжая вращаться вокруг своей оси. Потому что регулятор высоты продолжает вырабатывать частоты вращения для четных и нечетных двигателей, отличающиеся в $\sqrt{2}$ раза, а должен выдавать равновесные частоты, отличающиеся в $\sqrt{\sqrt{2}}$ раза. Очевидно, что плечи векторов сил (равные 1 м и 1.414 м) отличаются в $\sqrt{2}$ раза, и для того чтобы вращающие вокруг оси Z моменты от четных и от нечетных ВМГ стали равны друг другу, то и силы тяги ВМГ должны отличаться в $\sqrt{2}$ раза, а так как силы тяги пропорциональны **квадрату** частоты вращения, то частоты вращения должны отличаться между собой в $\sqrt[4]{2}$ раза.

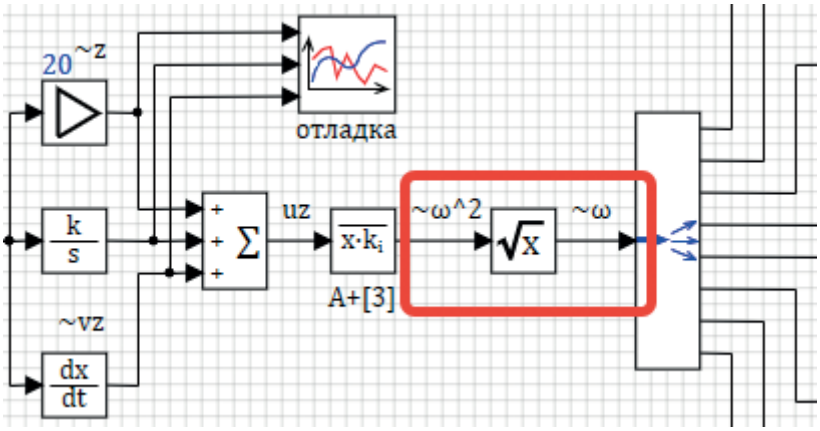


Рис. 9.2.2. Исправление одной из ошибок регулятора

9.2.1. Исправление регулятора высоты

Исправим эту ошибку, поставив блок типа «Корень квадратный» (вкладка **Функции**) между размножителем и демультиплексором (как представлено на рис. 9.2.2).

При этом регулятор перестанет работать, так как подобранные ранее коэффициенты и начальное условие регулятора были подобраны к варианту, когда он вырабатывает частоту вращения, а не квадрат частоты. Попробуйте сами подобрать новые коэффициенты с подсказкой, что новое «равновесное» значение начальных условий для регулятора равно примерно –14 000.

В варианте, когда Кп = 20, а Ки = 0.15 (коэффициенты усиления в П-ветви и в И-ветви регулятора высоты), переходной процесс будет вновь устойчивым и достаточно хорошим для данного этапа, а угловая скорость вращения будет около 0 рад/с. Результат представлен на рис. 9.2.3 и 9.2.4.

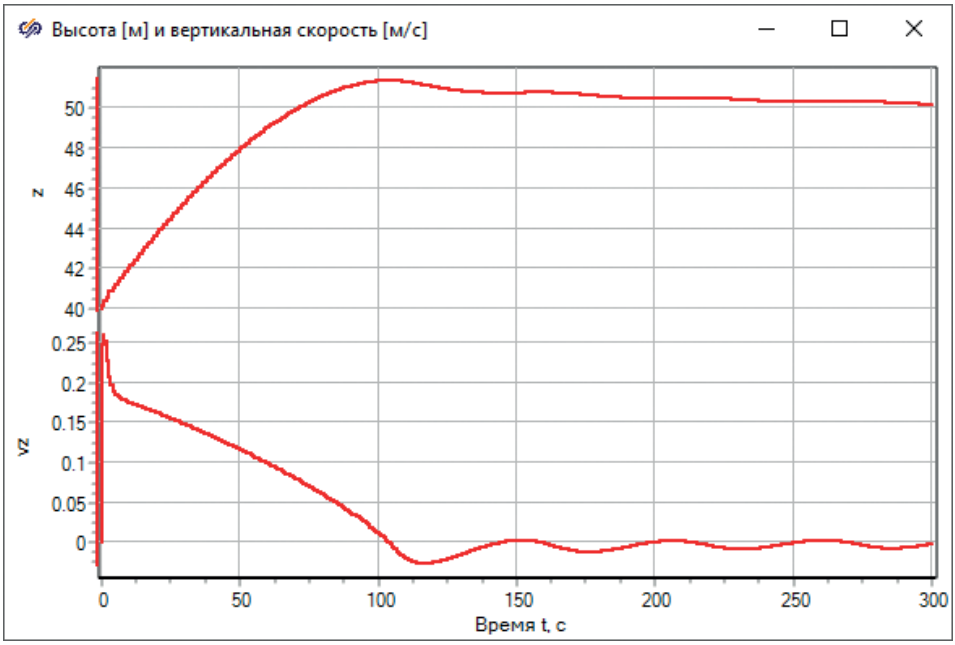


Рис. 9.2.3. Полет с «новым» регулятором высоты, выполненным почти без ошибок

Тот факт, что угловая скорость все же отлична от нуля и коптер претерпевает некоторую закрутку, объясняется тем, что мы задали начальную частоту вращения, равную 300 одинаковой для всех ВМГ, а это приводит к первоначальному импульсу вокруг оси Z, который потом сходит на нет за счет трения о (довольно вязкий пока еще) воздух.

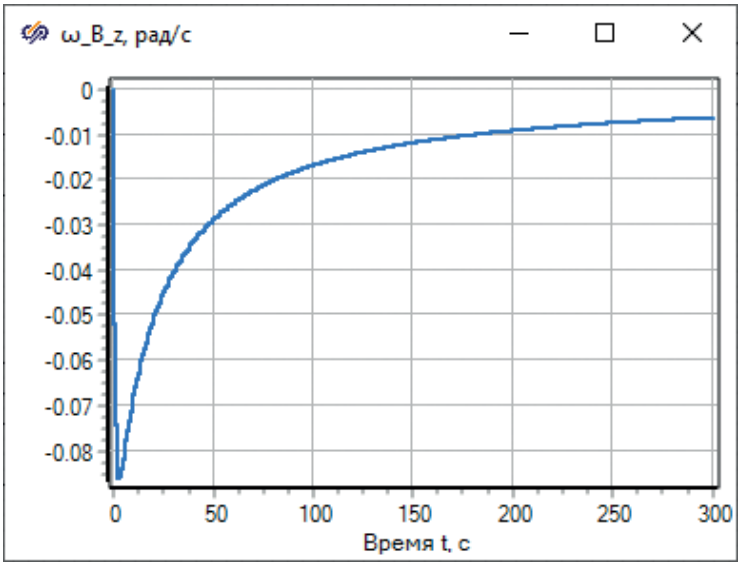
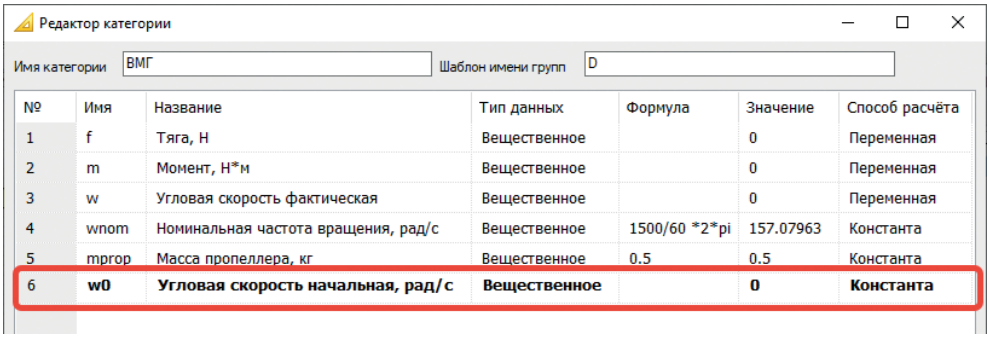


Рис. 9.2.4. Вращение коптера вокруг вертикальной оси почти отсутствует

9.2.2. Исправление начального состояния ВМГ

Давайте уберем и эту начальную закрутку – для чего нам потребуется задать разное начальное состояние по ВМГ, и это несколько сложнее сделать, чем просто задать всем 300 рад/с. Во-первых, это начальное состояние нужно где-то хранить (подпрограмма-то у нас одна!). Лучше всего для этого подходит база данных сигналов. Зайдите в нее (**Инструменты – База сигналов...**), и для категории «ВМГ» (в шаблон категории, для этого два раза щелкните по ней для этого) добавьте сигнал **w0**, как показано на рис. 9.2.5.



№	Имя	Название	Тип данных	Формула	Значение	Способ расчёта
1	f	Тяга, Н	Вещественное		0	Переменная
2	m	Момент, Н*м	Вещественное		0	Переменная
3	w	Угловая скорость фактическая	Вещественное		0	Переменная
4	wnom	Номинальная частота вращения, рад/с	Вещественное	1500/60 *2*pi	157.07963	Константа
5	mprop	Масса пропеллера, кг	Вещественное	0.5	0.5	Константа
6	**w0**	**Угловая скорость начальная, рад/с**	**Вещественное**		**0**	**Константа**

Рис. 9.2.5. Модификация категории «ВМГ»

Далее, запустите на расчет модель и где-то на 500–600-й секунде расчета, когда коптер придет в равновесие, запомните две угловые скорости для нечетного (например, первого) и четного двигателя. У нас они получились равными 265.513 и 315.751 рад/с (рис. 9.2.6). Отметим, что $(315.751/265.513)^4$ **равно ровно 2**. Это геометрически-механический факт для данной конструкции октокоптера, и у вас тоже должно так же получиться. Если масса коптера будет другой или характеристика ВМГ другая, то числа могут быть другими, но соотношение должно быть таким же.

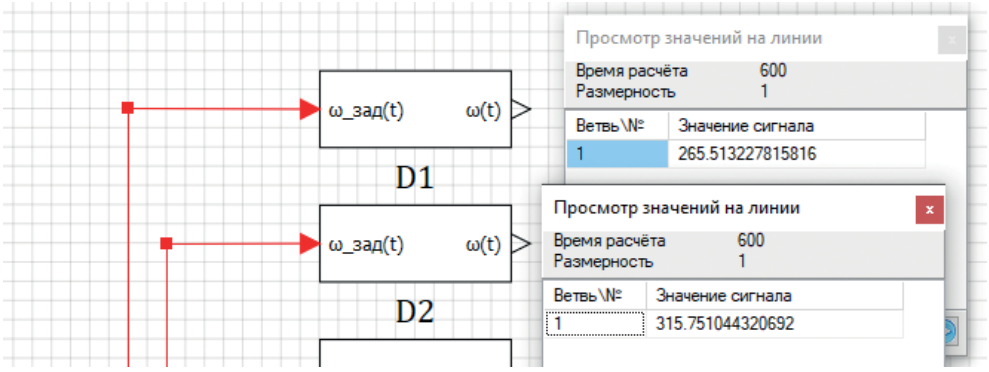


Рис. 9.2.6. Равновесные угловые скорости вращения ВМГ

Вернитесь в базу сигналов и всем нечетным двигателям проставьте значение сигнала **w0**, равное 265.513, а всем четным – 315.751. Это надо делать уже не в редакторе категории, а проходя по каждой группе сигналов, задавать зна-

чение сигнала в каждой из 8 групп. Таким образом, мы задали сигналы D1_w0 = D3_w0 = D5_w0 = D7_w0 = 265.513 и D2_w0 = D4_w0 = D6_w0 = D8_w0 = 315.751. Можно выделить несколько групп сигналов (зажимая **Ctrl** и делая клик мышкой и массово применив одно и то же значение для w0, см. рис. 9.2.7).

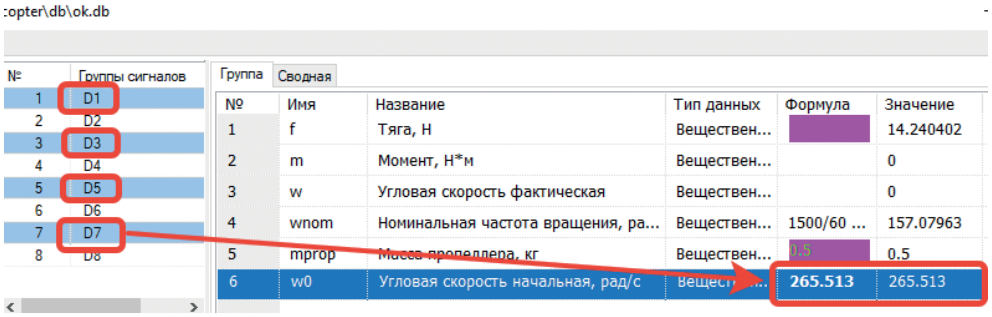


Рис. 9.2.7. Модификация базы сигналов

Сохраните проект, сохранив при этом и базу сигналов.

Откройте файл **sub/motor.prt.**

Зайдите в свойства апериодики 1-го порядка и в качестве формулы для начальных условий впишите там строку {submodel.name}_w0 (без кавычек, но с фигурными скобками, как представлено на рис. 9.2.8). Тогда при инициализации схемы вместо {submodel.name} будет по очереди подставлено D1, D2 и т. д., сигналы интерпретируются, и их значения заберутся из базы сигналов.

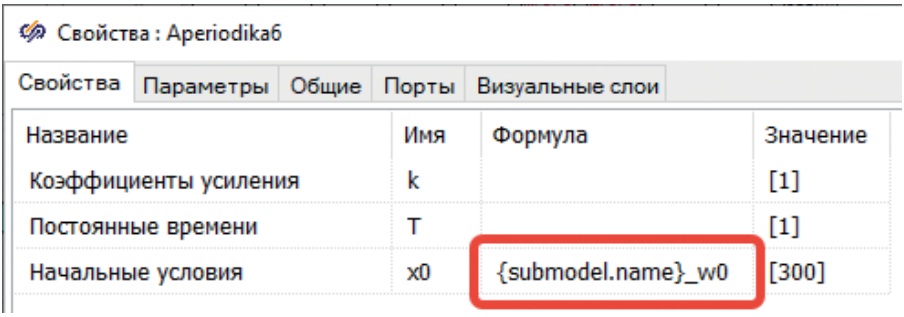


Рис. 9.2.8. Начальные условия для ВМГ в подпрограмме

При редактировании подпрограммы при этом будет ошибка, так как submodel.name сейчас равно пустой строке, а сигнала _w0 не существует. Сохраняйте все равно файл **motor.prt** и открывайте заново **copter.prt** – ошибок не должно быть, а при запуске на расчет коптер должен перестать закручиваться вокруг вертикальной оси (разве что ненамного, так как начальные частоты вращения мы выставили с точностью до 5-6 знака). График угловой скорости должен получиться аналогичным рис. 9.2.9 – это практически 0 рад/с.

Чтобы редактирование подпрограммы не выдавало ошибки, это можно либо делать в составе основного проекта – но тогда надо очень внимательно следить за сохранением изменений и всегда сохранять страницу, а не только основной файл. Либо можно в подпрограмме завести локальный сигнал проекта с име-

нем _w0, и тогда тоже ошибка выдаваться не будет при автономном редактировании подпрограммы.

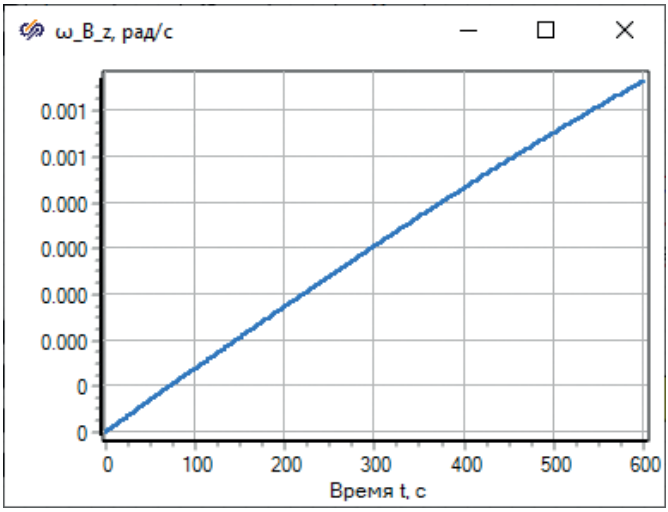


Рис. 9.2.9. «Паразитное» влияние регулятора высоты на курс сведено почти к нулю

Подведем промежуточный итог – для модели коптера, который летает в относительно вязком воздухе, мы нашли начальные равновесные частоты вращения для двигателей и настроили регулятор высоты таким образом, чтобы он более-менее хорошо работал на поддержание заданной высоты полета. При этом коптер, хотя и летает в трехмерном пространстве, пока что представлен как будто в одномерном приближении – по сути, меняется только вертикальная скорость и вертикальная координата. По остальным направлениям возмущения и силовые воздействия равны нулю…

9.2.3. Регулятор высоты в первом приближении

Теперь можно довести регулятор высоты до более верного состояния.

Что еще есть «нехорошего» в реализованном регуляторе высоты – то, что равновесные значения частот вращения мы «загнали» в регулятор, который, вообще говоря, не должен бы их содержать, а должен выдавать околонулевой сигнал в случае, когда коптер находится в равновесии. Давайте еще это исправим и потом уже перейдем к другим направлениям – а именно к стабилизации коптера по крену и тангажу.

Исправление логично было бы сделать таким образом – если у нас известно по каждому из двигателей равновесное значение $\omega(t) = w0$, а регулятор вырабатывает лишь отклонения $\pm\Delta\omega^2(t)$, то надо отделить базовые величины в блок типа «Константа», а в регуляторе оставить только отклонения от базовой величины. Один из вариантов – при помощи блоков типа «Абсолютное значение» и «Знак» (вкладка **Операторы**) – представлен на рис. 9.2.10. Блок типа «Константа» и сумматор ранее уже были использованы – вы знаете, где их найти. В константе можно вписать или числами вектор, или вписать вектор вида **[D1_w0, D2_w0, D3_w0, D4_w0, D5_w0, D6_w0, D7_w0, D8_w0]**. Попробуйте

самостоятельно реализовать эту доработку и результат расчетов сравните с рис. 9.2.11. Коэффициент пропорциональной ветки пришлось снизить при этом до Кп = 0.5, а интегральной до Ки = 5/10 000. Начальные условия в интеграторе следует вернуть к нулевым.

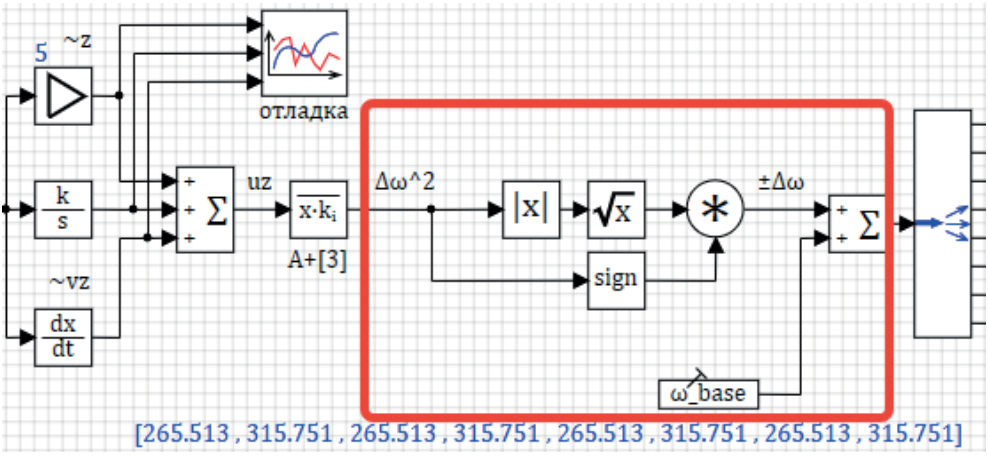


Рис. 9.2.10. Модификация регулятора высоты, базовая частота вращения вынесена отдельно

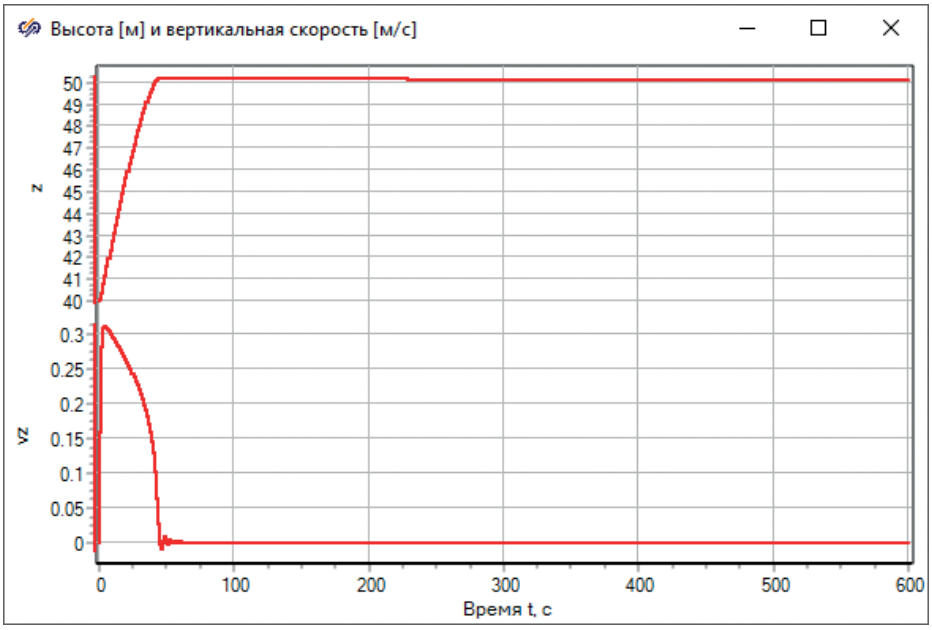


Рис. 9.2.11. Кп = 0.5, Ки = 5/10000, н. у. = 0 в регуляторе

Видно, что переходной процесс стал «лучше», чем был до этого. Таким образом, регулятор высоты в первом приближении сделан. Его еще придется перенастраивать, когда мы снизим «вязкость» воздуха в модели коптера до более реалистичных значений.

Проанализируем еще, что происходит с частотами вращения ВМГ при данном переходном процессе. Для этого можно незадействованные выходы со всех 8 субмоделей вывести на 1 график (рис. 9.2.12) и, если построить первые 100 с переходного процесса, получить график частот вращения двигателей, аналогичный рис. 9.2.13. Из графика видно, что частоты вращения четырех двигателей начинаются с одной равновесной частоты, четырех других – из другой равновесной частоты вращения, потом насколько увеличиваются (обеспечивая тем самым подъем коптера), затем снижаются до величины, меньшей равновесной (коптер при этом тормозит), и в колебательном процессе приходят вновь к равновесным величинам. Таким образом, процесс выглядит правдоподобным и не противоречит физике. В дальнейшем при анализе более сложных переходных процессов будет полезно обращать внимание на этот график для понимания тонкостей «микширования» двигателей регулятором, которое происходит в пространственных переходных процессах. Сейчас линии для двух групп двигателей (по 4 шт.) накладываются друг на друга, но, вообще говоря, в динамическом процессе они будут «разъезжаться» по оборотам, оставаясь около небольшой окрестности равновесной частоты вращения. И только иногда – в сильных неравновесных процессах, на грани потери устойчивости, они будут сильно отличаться друг от друга.

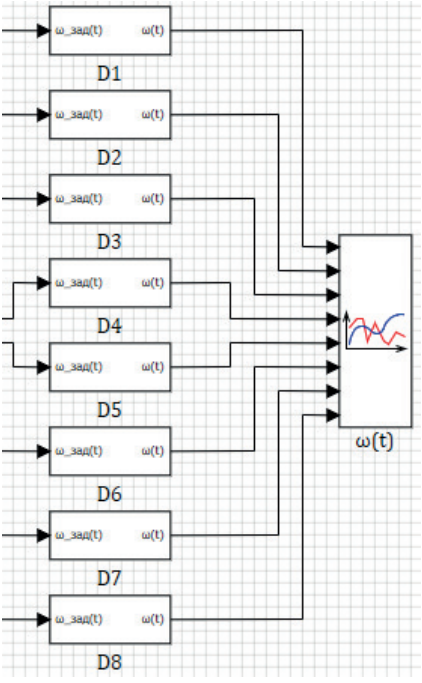


Рис. 9.2.12. График частот вращения всех двигателей ВМГ

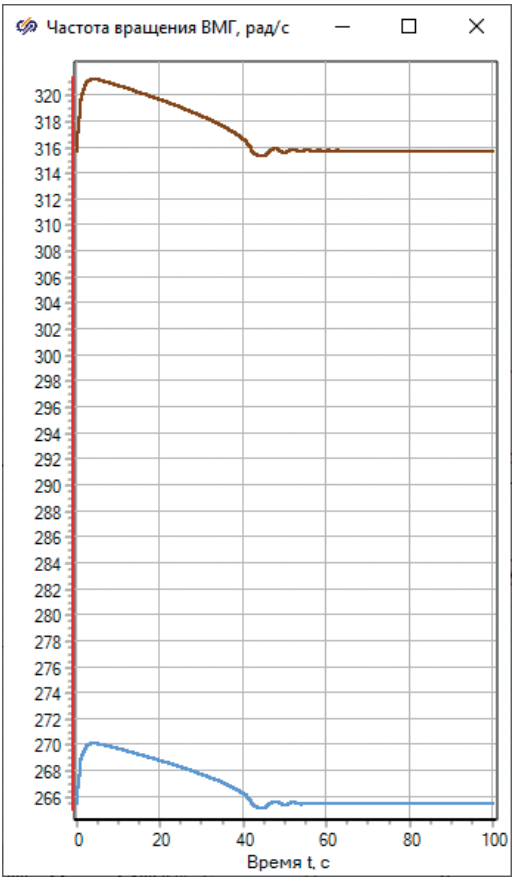


Рис. 9.2.13. Работа регулятора высоты

И еще одна доработка в регуляторе высоты – блок вычисления производной (Д-ветвь) в регуляторе сейчас сделана блоком, который «берет» производную от текущей высоты (или от рассогласования, что одно и то же). Как правило, эта операция сопровождается добавлением некоторого количества «численного» шума, так как шаг интегрирования может меняться или еще какие-то резкие изменения в вычисленной текущей высоте могут приводить к сильным скачкам производной от этого сигнала. И в этом блоке нет поправочного коэффициента, которым в будущем придется тоже манипулировать, донастраивая регулятор. Если же посмотреть на модель динамики коптера, то эта производная в нем уже считается – это вертикальная скорость **vz**. Давайте скорректируем регулятор, заведя в него (блоком «Из памяти») сигнал vz и, пропустив его через усилитель, заменим им блок вычисления производной, см. рис. 9.2.14.

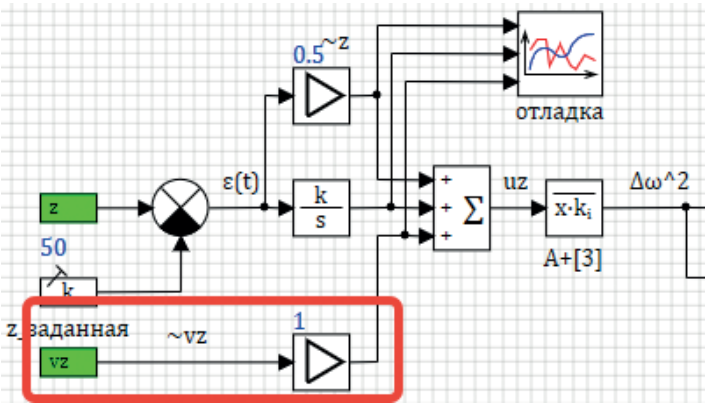


Рис. 9.2.14. Модификация Д-ветви регулятора высоты

Работа регулятора после этой процедуры не должна практически никак измениться, но схема будет (численно) более устойчивой. На этом пока что регулятор высоты завершен. Дальнейшее тестирование выявит еще ряд недостатков в нем, которые мы исправим и усовершенствуем его (например, можно ограничить вертикальную скорость какой-либо величиной – когда коптер разгонится до максимальной скорости, отключать действие регулятора в ту же сторону или отключить ненужную И-ветвь в те моменты, когда рассогласование велико, скажем, когда оно больше 5 или 10 м, так как И-ветка нужна, по сути, только для точной «доводки» коптера к заданной позиции, а на далеком расстоянии она только мешает и т. п.).

9.3. Регулятор крена и тангажа

Теперь, когда коптер «оснащен» регулятором, который держит его высоту на заданном уровне, можно приступить к реализации регуляторов, стабилизирующих коптер по углам крена и тангажа, – это наклоны коптера к горизонту вокруг осей X и Y. Заданный уровень по высоте, кстати, вы можете произвольно менять в процессе расчета, а также вместо константы задавать какую-нибудь «программу» изменения высоты, воспользовавшись либо блоком типового сигнала (например, «Меандр»), либо «Кусочно-линейной» (зависимостью) из

вкладки **Источники**, и пронаблюдать, как изменение задания отрабатывает регулятор высоты и модель коптера.

9.3.1. Идея регуляторов φ и θ

Регулятор крена и тангажа будет представлять собой тоже ПИД-регулятор, только уточненный, – кроме Д-ветви будет еще ветвь со второй производной (назовем ее Д2-ветвь) от сигнала рассогласования текущего угла от заданного. Это как бы сигнал от датчика углового ускорения, так как следить здесь надо не столько за самим наклоном, сколько за «попытками» коптера начать наклоняться – т. е. за угловой скоростью, и надо тщательно отслеживать моменты, когда она по той или иной причине резко меняется; регулятор должен работать «на опережение» самого факта наклона и факта разгона коптера по той или иной оси.

Другими словами, можно сказать, что здесь будет реализован ПИД-регулятор для угловой скорости, а не для угла наклона. Тем более что в тестовых моделированиях пока что пришлось отказаться вообще от И-ветви (сделав там коэффициент нулевым). Важный момент: поскольку есть разница между углами Эйлера и угловыми скоростями и углами наклона в связанной с коптером системе координат B, а двигатели и винты ВМГ, а также способ их микширования у нас привязан к коптеру, то регулятор крена и тангажа надо строить от угловой скорости и углов наклона, которые получаются для системы B, а не в инерциальной системе отсчета – потому что силовое воздействие всегда будет идти вокруг осей x_B и y_B независимо от ориентации коптера.

Заданный угол наклона пока что будет равен 0, так как коптер должен всегда быть горизонтальным. Далее мы доделаем заданный угол таким образом, чтобы регулятор крена и тангажа осуществлял и горизонтальный полет коптера, т. е. движение по осям X/Y тоже будет реализовано через эти два регулятора.

9.3.2. Создание регуляторов φ и θ

Внешний вид регулятора крена (φ) представлен на рис. 9.3.1, внешний вид регулятора тангажа – на рис. 9.3.2. Как видно из рисунков, ничего принципиально сложного здесь нет, все очень похоже на регулятор высоты. Входные сигналы в эти регуляторы у нас не все были созданы ранее, а только сигналы угловой скорости по осям x и y в системе B.

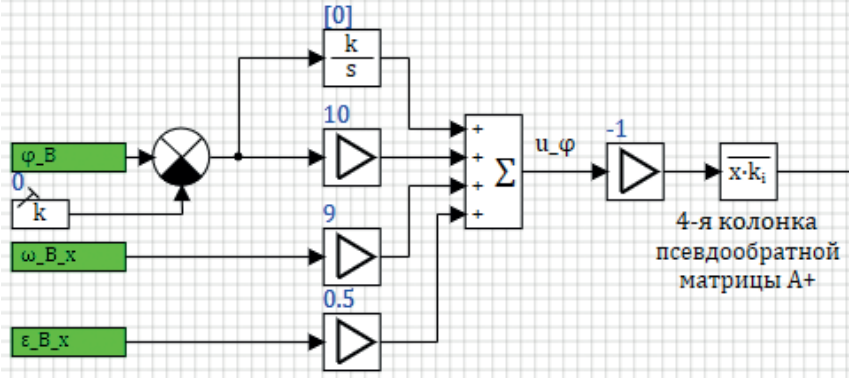


Рис. 9.3.1. Регулятор крена

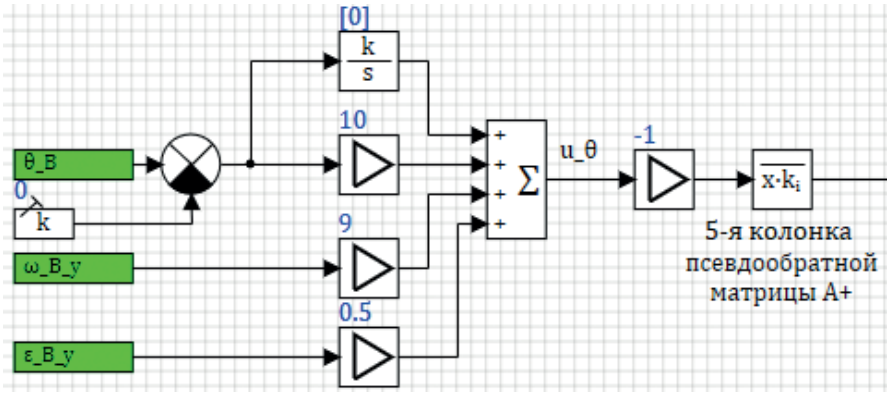


Рис. 9.3.2. Регулятор тангажа

Давайте для начала подготовим нужные входные сигналы. Угловое ускорение в модели коптера у нас, в принципе, посчитано – это входы в интеграторы, которые насчитывают угловую скорость, так что потребуется добавить три блока типа «B память» около этих регуляторов, чтобы создать три новых сигнала – угловых ускорения коптера вокруг осей в связанной системе координат **B** (рис. 9.3.3): ε_B_x, ε_B_y, ε_B_z.

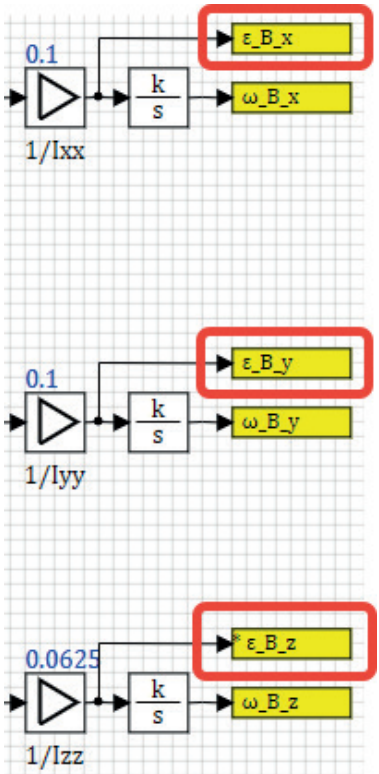


Рис. 9.3.3. Угловые ускорения коптера в системе B

С углами наклона для системы координат B ситуация несколько сложнее – в принципе, их можно получить обратным преобразованием из уже посчитанных углов Эйлера, но, чтобы не усложняться математикой, проще это сделать дополнительными интеграторами, проинтегрировав угловые скорости непосредственно в системе **B**, как показано на рис. 9.3.4. Сделайте это, добавив тем самым еще три сигнала φ_B, θ_B и ψ_B. Фактически эти углы суть углы наклона самой системы координат **B**.

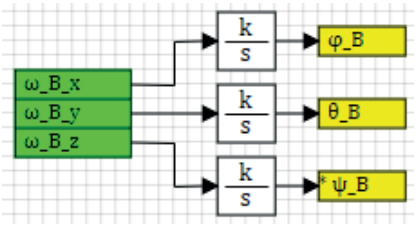


Рис. 9.3.4. Вычисление углов ориентации в системе B

Теперь, когда все входные сигналы для регуляторов созданы, вы можете самостоятельно набрать схемы рис. 9.3.1 и 9.3.2, так как все составляющие этих схем, все блоки мы ранее использовали. Подобранные коэффициенты усиления представлены на схемах, общий коэффициент усиления равен –1, поскольку при выводе псевдообратной матрицы A+ она показывает «прямое» направление воздействия, а регулятор должен работать в обратную сторону от угла наклона. В регуляторе высоты этого не требовалось, так как там мы уже брали рассогласование и вертикальную скорость, взятые с обратным знаком, – при вычислении z и vz ось была «перевернута».

Для размножителей следует проставить следующие векторы коэффициентов:
Для регулятора φ: [0, –5.0292, –7.1124, –5.0292, 0, 5.0292, 7.1124, 5.0292].
Для регулятора θ: [7.1124, 5.0292, 0, –5.0292, –7.1124, –5.0292, 0, 5.0292].

Как мы видим, при этом работают «тройки» двигателей, создавая поворотный момент вокруг соответствующей оси. Двигатели должны работать при этом так, чтобы не создавать никаких возмущений по другим каналам управления.

Как протестировать набранные регуляторы? Во-первых, выходы этих регуляторов следует сложить с результатом работы регулятора высоты и базовыми частотами вращения для ВМГ – это можно сделать, набрав блоками взятие корня квадратного из результата работы регуляторов (см. рис. 9.3.5), – но то плохой способ, так как одну и ту же операцию здесь приходится делать снова и снова. Лучше результат работы этих двух новых регуляторов сложить с результатом работы регулятора высоты, потом взять корень (1 раз) и сложить с базовыми частотами. Этот – лучший – вариант представлен на рис. 9.3.6.

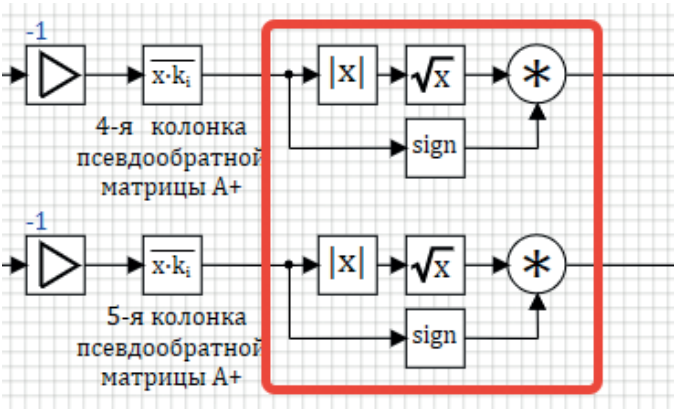


Рис. 9.3.5. «Плохой» вариант

На рис. 9.3.6 показан целиком внешний вид регуляторов на текущий момент, и можно видеть, что схема постепенно разрастается (а в сумме с моделью объекта она будет уже довольно громоздкой). Когда расчетная схема только набирается, это допустимо. Но вообще, гораздо лучше приучаться к структуризации и корректному оформлению схемы сразу по мере ее создания и, когда становятся видны составные узлы расчетной схемы, структурировать ее по субмоделям таким образом, чтобы сходные элементы модели находились бы на одном уровне. Этим займемся сразу после тестирования регулятора ориентации, прежде чем двигаться далее. Но следующую часть работы уже выполним в своей собственной субмодели, а не на самом верхнем уровне расчетной схемы.

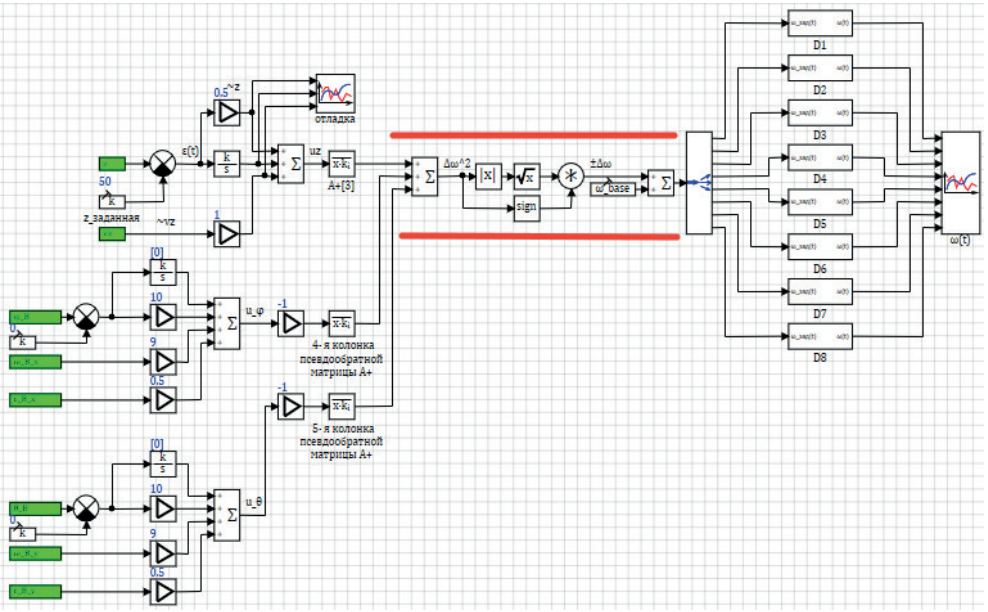


Рис. 9.3.6. Лучший вариант модели регулятора

9.3.3. Тестирование, анализ результатов

Для тестирования совместной работы двух новых регуляторов нам уже потребуется наклонять коптер по двум осям, и, чтобы оценивать результаты работы регуляторов, потребуются новые графики. Предлагаем все графики свести в одно место расчетной схемы – их будет так проще искать и вызывать по мере надобности.

Создадим для этого новую субмодель, которую дополнительно подпишем. Поставьте ее на расчетную схему (блок «Субмодель» из вкладки **Субструктуры**) и задайте ей подпись «Графики», как представлено на рис. 9.3.7. Дополнительно поставьте рядом еще две субмодели и подпишите их «Регулятор» и «Модель октокоптера» – они пока будут пустыми, а позже перенесем в них набранные составные части модели. Затем перейдите в субмодель графиков (на второй уровень вложенности) и создайте для начала временной график, на который выведите скорости vx и vy, как показано на рис. 9.3.8.

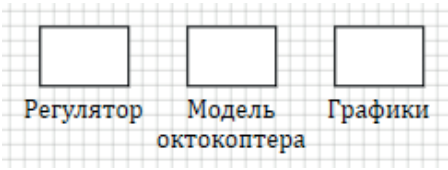


Рис. 9.3.7. Новые субмодели для структуризации расчетной схемы

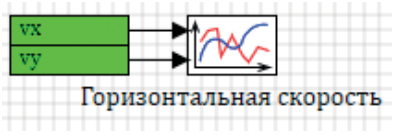


Рис. 9.3.8. Временной график для анализа горизонтальной скорости коптера

Далее, следует перенести в эту субмодель график высоты и вертикальной скорости, который мы создали ранее, а также график угловой скорости вокруг оси Z. Для того чтобы не потерялось оформление этих графиков, блоки не следует удалять с расчетной схемы, а лучше их перенести через буфер обмена, вырезав их из тех мест, где они были созданы, и «вставив» их внутри субмодели «Графики». Входные сигналы набрать заново, см. рис. 9.3.9. Обратите внимание: для того чтобы сформировать график углов наклона в градусах, мы умножаем расчетные величины крена и тангажа на коэффициент **180/pi** – его можно прямо в таком виде вписать в колонку **Формула** у соответствующих блоков.

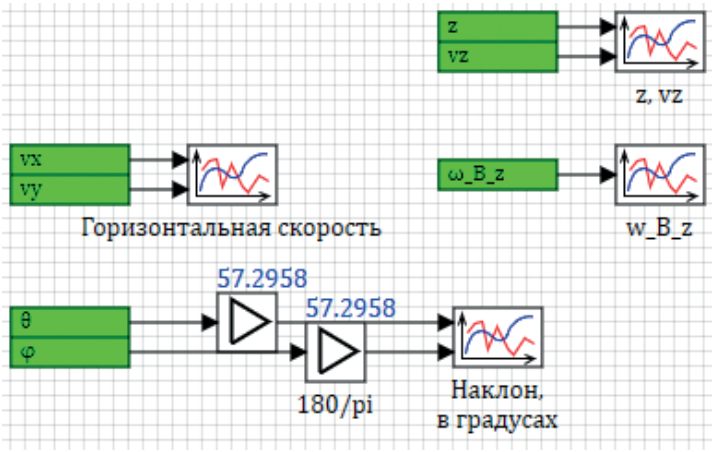


Рис. 9.3.9. Старые и новые графики для анализа регуляторов крена и тангажа

Теперь, если вы все сделали правильно, то, если запустить модель на расчет с включенными в цепь регулирования двумя новыми регуляторами, внешний вид переходного процесса либо не поменяется вообще, либо будет очень слабо отличаться – могут насчитаться угловые скорости по осям x и y на уровне величин 1e-15…1e-14 (см. рис. 9.3.10 и 9.3.11).

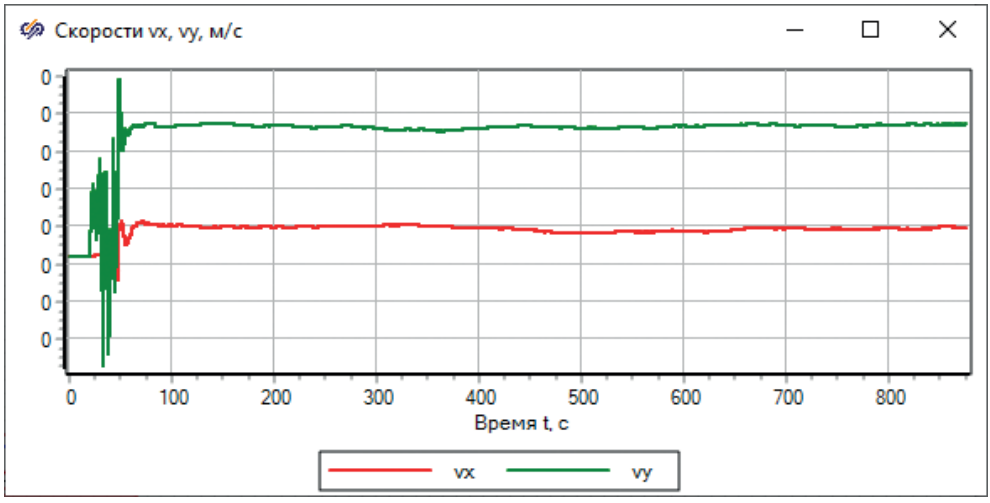


Рис. 9.3.10. Горизонтальные скорости (коптер стартует с нулевыми начальными условиями, кроме высоты = 40 м)

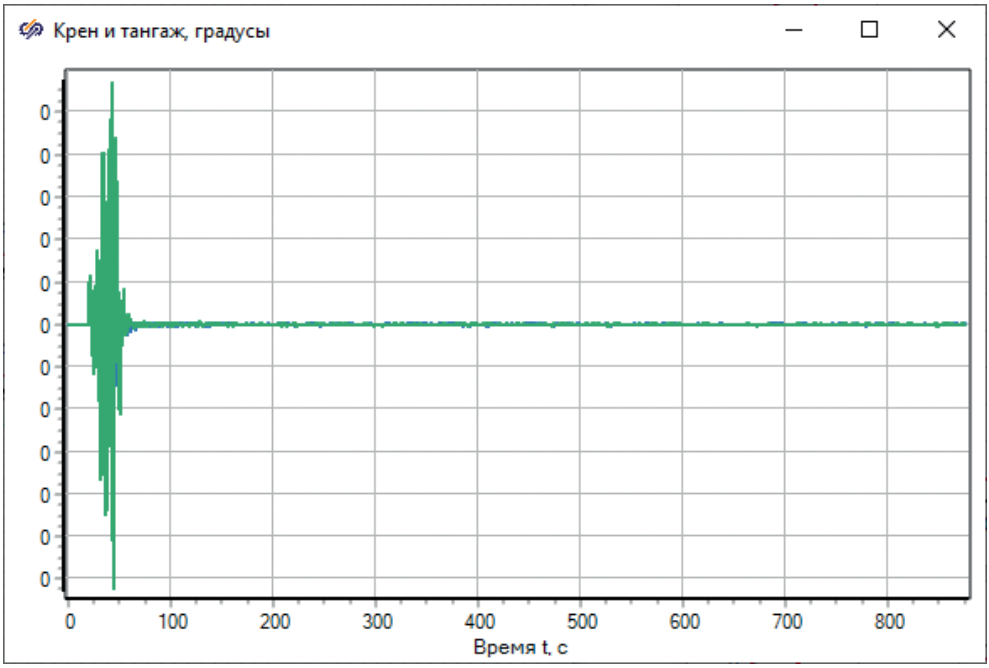


Рис. 9.3.11. Углы крена и тангажа

Если что-то пошло не так, ознакомьтесь со следующим подразделом…

9.3.4. Методы анализа и поиск ошибок (если что-то пошло не так…)

Если в каком-то месте вы опечатались или сделали что-то неверно – не так, как сказано выше в описании, – то рис. 9.3.10 и 9.3.11 у вас могут не получиться, и регуляторы либо будут работать не совсем верно, либо совсем не верно. Достаточно где-то перепутать плюс с минусом или ошибиться с коэффициентами, или еще что-то – система может пойти вразнос, и неопытному пользователю бывает на первых порах сложно разобраться, где и что происходит. Надо идти последовательно, от инициализации схемы и по шагам расчета, и внимательно смотреть, где и что насчитывается неверно.

Если что-то пошло не так и результат моделирования внезапно вышел за рамки разумного (например, могут получаться координаты или скорости или другие переменные равными ±1e20 или ±1e180 или подобным числам) – это говорит о том, что расчет «рассыпался», произошло либо деление на ноль, так как ошибка в численной схеме и численном интегрировании, либо еще какая-то недопустимая операция. В таком случае следует искать ошибку или в наборе схемы, или в задании начального состояния и заданных значений для констант, или в методе решения (попробовать уменьшить шаг и/или поменять метод интегрирования).

Также существуют определенные «приемы», которыми можно пользоваться для облегчения поисков ошибки. Если вы последовательно выполняли указа-

ния данной пошаговой методики, то ошибки возникнуть не должно. Но если она все же возникла, вам придется самостоятельно осуществить поиск «проблемного» места. К сожалению, количество возможных мест и видов ошибок велико, и невозможно привести в методике все случаи жизни и существующие способы устранения всех ошибок, однако возможно дать некоторые рекомендации. Приведем здесь основные из них.

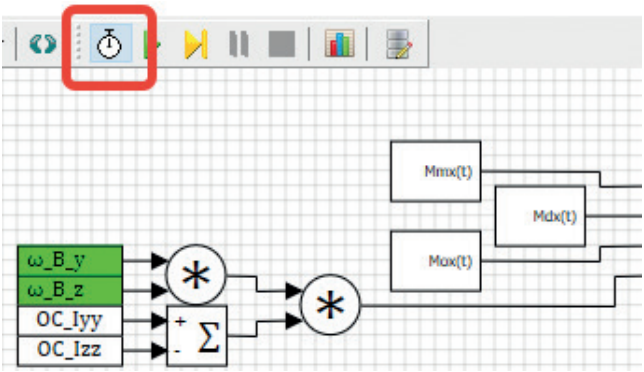


Рис. 9.3.12. Инициализация схемы

Во-первых, надо сохранить проект, закрыть его и переоткрыть заново (лучше даже с перезапуском SimInTech для чистоты эксперимента) и сделать инициализацию схемы (а не запуск ее на расчет) – это кнопка с секундомером, слева от кнопки **Пуск** (как показано на рис. 9.3.12). Если инициализация завершилась неуспешно, то система выдаст внизу отладочную информацию по топологии схемы – где не хватает линии связи, или не заданы те или иные свойства блоков, или опечатка в имени сигнала или еще что-то некорректное, что не позволяет инициализировать расчет, – это все надо устранить. Потом после успешной инициализации надо пройтись по основным смысловым местам схемы, основным уравнениям и убедиться, что по всем интеграторам на их выходах стоят корректные начальные значения (в нашей схеме должны быть везде нули, кроме интегратора высоты коптера – там должно быть 40, см. рис. 9.3.13).

Это можно сделать либо двойными щелчками мыши по нужным линиям связи, либо включив опцию отображения текущих значений по линиям связи – эта кнопка находится еще левее инициализации, там же в панели инструментов схемного окна. Также надо оценить все насчитанные правые части – это входы в интеграторы, они должны быть или нулевыми, или иметь определенный размер, укладывающийся в понятие «производная». На рис. 9.3.13 все правые части при инициализации нулевые, кроме правой части для угловой скорости вокруг вертикальной оси – но там очень малая величина, и при желании можно проследить по линиям связи, откуда она появилась. Не должно быть каких-то астрономических чисел. Если такое число увидели – значит, данная правая часть в уравнении посчиталась с ошибкой уже на инициализации, и по схеме по линиям связи надо пройти и найти источник этих больших чисел, который следует устранить.

Во-вторых, в SimInTech есть пошаговый режим расчета, когда расчет не продолжается дальше, а делается всего один шаг интегрирования и есть

возможность аккуратно пройтись снова по расчетной схеме и обнаружить место – первоисточник, где насчитались ошибочные значения (например, произошло деление на ноль, и результат получился равным плюс-минус бесконечности). В пошаговом режиме расчета также можно смотреть на графики и по ним обнаружить то место в схеме, в котором первой появляется проблема, из-за которой численная схема рассыпается. Сама схема, по сути дела, является графическим отладчиком в этот момент. Линии, на которых текущее значение больше нуля, подсвечиваются фиолетовым цветом (эта настройка сделана для дискретных алгоритмов по умолчанию, но можно этим пользоваться и при такой отладке).

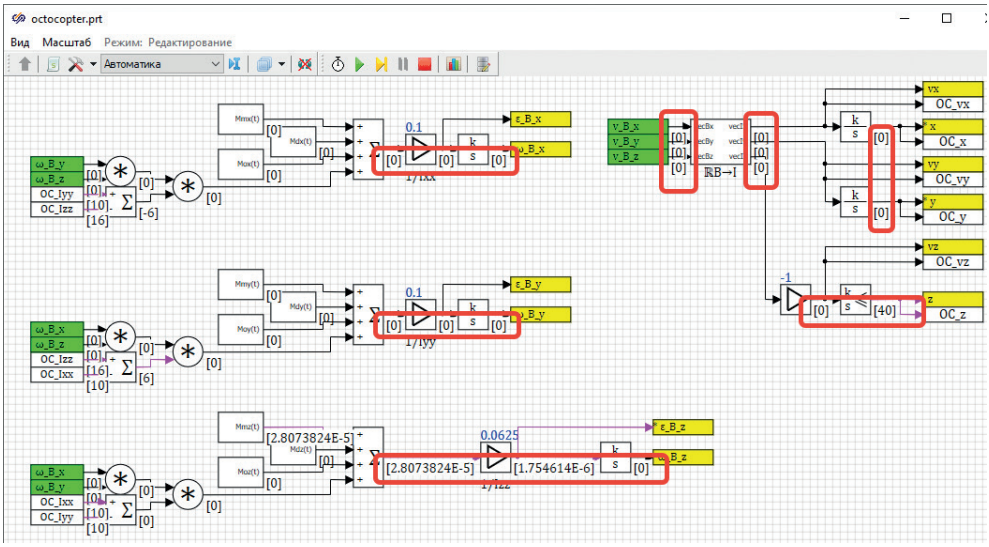


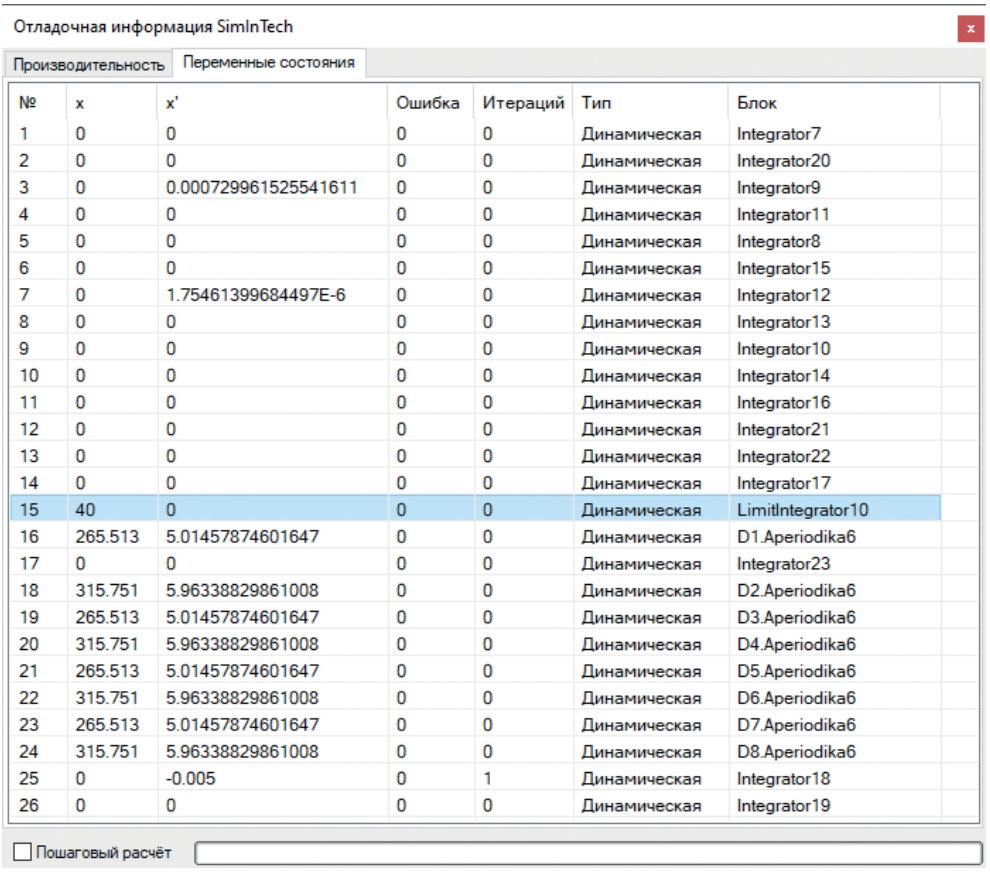
Рис. 9.3.13. Часть схемы при корректной инициализации

В-третьих, для больших схем, где трудоемко идти по всем участкам схемы, есть отладочный инструмент в пункте главного меню **Расчет → Отладочная информация**. Зайдя туда, можно увидеть текущие значения входов и выходов во все динамические звенья (в нашей задаче это интеграторы и апериодики первого порядка в ВМГ) и оперативно оценить, где кроется проблема. На рис. 9.3.14 представлено это окно, когда проблемы нет – все переменные состояния и их производные находятся «в пределах нормы». По текущим значениям можно определить, какая переменная за что отвечает, а двойным щелчком мыши по строке можно оперативно перейти к тому месту расчетной схемы, где насчитывается выделенная переменная состояния.

Как правило, ошибку деления на ноль или на околонулевые значения найти не трудно. Ошибку, если вы где-то перепутали плюс с минусом, бывает найти труднее, или она проявляется не всегда, а только при работе регулятора – а в те промежутки времени, когда он выдает на выходе 0, ошибка не будет видна. Бывают и более сложные ошибки, неявные, которые найти еще труднее.

Если быстро обнаружить ошибку не получается или она не вполне очевидна – например, может быть так, что переходной процесс идет неверно, но все

же в каких-то рамках и пределах физичности. Или взаимное влияние одной фазовой координаты на другую приводит к взаимной раскачке переменных (не вполне «физичной», а из-за ошибки в наборе уравнений). Тогда надо искать ошибку в частных случаях схемы. Это более сложный способ, но он тоже существует. В модели коптера у нас уже получилось более 20 дифференциальных переменных (одновременно решается 26 дифференциальных уравнений первого порядка на данный момент), а вообще, бывают модели, где таких уравнений могут быть сотни и тысячи, и проследить все без исключения взаимосвязи не реально.



№	x	x'	Ошибка	Итераций	Тип	Блок
1	0	0	0	0	Динамическая	Integrator7
2	0	0	0	0	Динамическая	Integrator20
3	0	0.000729961525541611	0	0	Динамическая	Integrator9
4	0	0	0	0	Динамическая	Integrator11
5	0	0	0	0	Динамическая	Integrator8
6	0	0	0	0	Динамическая	Integrator15
7	0	1.75461399684497E-6	0	0	Динамическая	Integrator12
8	0	0	0	0	Динамическая	Integrator13
9	0	0	0	0	Динамическая	Integrator10
10	0	0	0	0	Динамическая	Integrator14
11	0	0	0	0	Динамическая	Integrator16
12	0	0	0	0	Динамическая	Integrator21
13	0	0	0	0	Динамическая	Integrator22
14	0	0	0	0	Динамическая	Integrator17
15	40	0	0	0	Динамическая	LimitIntegrator10
16	265.513	5.01457874601647	0	0	Динамическая	D1.Aperiodika6
17	0	0	0	0	Динамическая	Integrator23
18	315.751	5.96338829861008	0	0	Динамическая	D2.Aperiodika6
19	265.513	5.01457874601647	0	0	Динамическая	D3.Aperiodika6
20	315.751	5.96338829861008	0	0	Динамическая	D4.Aperiodika6
21	265.513	5.01457874601647	0	0	Динамическая	D5.Aperiodika6
22	315.751	5.96338829861008	0	0	Динамическая	D6.Aperiodika6
23	265.513	5.01457874601647	0	0	Динамическая	D7.Aperiodika6
24	315.751	5.96338829861008	0	0	Динамическая	D8.Aperiodika6
25	0	-0.005	0	1	Динамическая	Integrator18
26	0	0	0	0	Динамическая	Integrator19

Рис. 9.3.14. Корректно почитанные переменные состояния и их производные

Например, по какой-то причине коптер может «заваливаться» и переворачиваться – один из углов ориентации растет и растет, несмотря на вроде бы корректную работу регулятора. Тогда можно «зафиксировать» другие переменные состояния в стационарных равновесных значениях и оставить считаться только интересующую нас дифференциальную переменную. Делается это путем замораживания линии связи или блока (исключения из расчета) у тех динамических блоков, значения которых мы хотим оставить неизменными. Например, можно выключить расчет всех дифференциальных уравнений, кроме

одного, выставив для всего времени моделирования x(t) = 0, y(t) = 0, z(t) = 40, φ(t) = 0, ψ(t) = 0, и все скорости так же занулить, а интегратор, который считает тангаж, оставить рассчитываться. Посмотреть после этого, как будет работать схема в таких условиях. Если все нормально, можно следующую – одну (!) дифференциальную переменную «включить» в расчет, запустить моделирование, оценить результат. И так по очереди, включая в цепь расчета следующие и следующие участки схемы, обнаружить, что очередное включение (задействование еще одного дифференциального уравнения) приводит к тому, что расчет начал рассыпаться или давать неверный результат – значит, в цепи блоков для данной переменной состояния есть ошибка. Следует внимательно пройтись по реализации этого уравнения и найти, понять, где сделано неправильно. Опять же, в этом способе отладки большую помощь оказывают графики, на которых видны не только текущие значения переменных, но и история расчета.

В качестве примера приведем такой вариант – например, нам показалось что изменение курса коптера рассчитывается неверно и это приводит к «проблемам» в другой части модели. Тогда можно временно вместо выхода интегратора подставить в блок «В память» нулевое значение, исключить из расчета блок и не принимать во внимание курс, насчитанный «честно» (см. рис. 9.3.15).

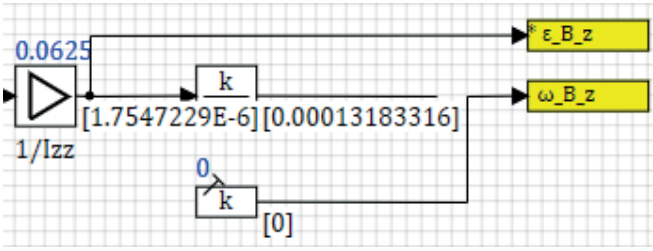


Рис. 9.3.15. Подстановка 0 вместо расчетного значения

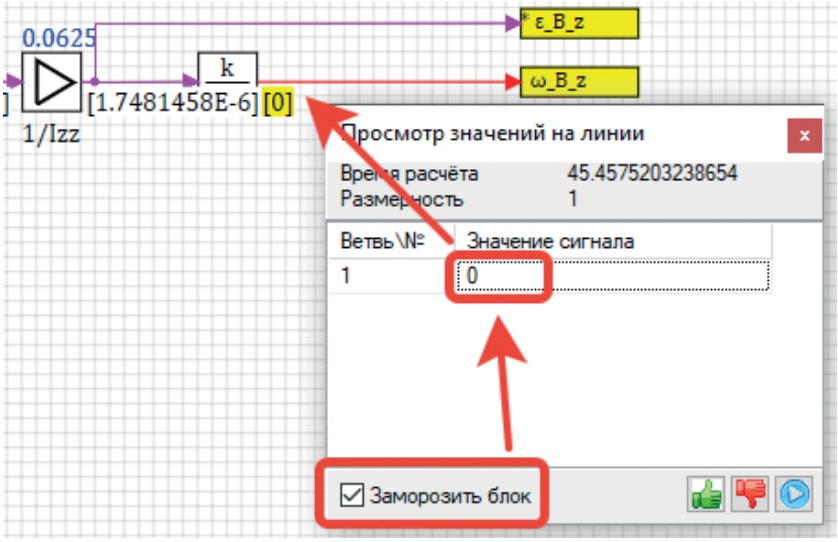


Рис. 9.3.16. Заморозка линии связи

Если надо в процессе расчета «перебить» текущее значение своим, то можно воспользоваться инструментом «заморозки» линии связи – для этого надо ее выделить двойным щелчком, включить галочку **Заморозить блок** (имеется в виду блок, из которого линия связи выходит), затем прописать 0 на строке с сигналом вместо текущего значения – иначе заморозится текущее значение – и продолжить расчет. Блок будет выдавать 0 в следующие блоки (куда входит линия связи). Если «заморозка» рис. 9.3.16 постоянная и потом надо ее вручную убирать, то заморозка рис. 9.3.17 – временная, при следующем расчете она сбросится, а блок разморозится.

Эти приемы отладки не сложны в освоении, но позволяют довольно быстро (по сравнению с отладкой программного кода, если бы модель была набрана в виде исходного кода на каком-либо языке программирования) найти ошибочную реализацию.

Дальнейшее пошаговое описание исходит из того, что все, что написано выше, у вас получилось проделать без сильных ошибок и без отклонений от методики. Иначе может получиться непредсказуемый результат.

9.3.5. Задание начальных условий, возмущений, анализ регуляторов

Если вы помните, в исходных уравнениях динамики мы оставляли равными нулю еще по одному слагаемому, которым предполагалось смоделировать внешнее возмущающее воздействие на коптер. Пришло время задействовать эти слагаемые. Идея заключается в следующем – путем создания внешнего возмущающего воздействия по какому-либо одному направлению или сразу по нескольким направлениям мы будем иметь возможность выводить коптер из состояния динамического равновесия (в работе должны быть регуляторы по трем каналам управления – высоты, крена и тангажа) и наблюдать, насколько успешно/быстро/качественно регулятор справляется с задачей стабилизации.

Однако самый простой способ сделать это (если бы мы не предусмотрели слагаемые) – задать ненулевые начальные условия в соответствующих интеграторах. Тем самым коптер будет стартовать либо из негоризонтального положения, либо с ненулевой угловой скоростью, либо с произвольной комбинацией углов ориентации и угловых скоростей. Линейные скорости нас пока мало интересуют, так как регулятора по направлениям мы еще не сделали. Пока что задача состоит в том, чтобы убедиться в устойчивой работе регулятора ориентации коптера при произвольных внешних воздействиях (конечно, воздействия должны быть не чрезмерными и не должны сильно наклонять коптер). Исходим из того, что углы наклона не должны превышать 25° (от горизонтали) на протяжении всех переходных режимов.

Давайте зададим начальную угловую скорость вращения, равную одному градусу в секунду (°/с) для скорости ω**_B_x** и 2 °/с для ω**_B_y**. Для этого достаточно вписать строки 1 × pi/180 и 2 × pi/180 в строки **Формула** для начальных условий двух интеграторов, вычисляющих эти скорости, как показано на рис. 9.3.17 (показано для одного из них). Физически это будет соответствовать, что коптер «стартует» не просто с высоты 40 м, а он при этом еще и «крутится» (например,

после незавершенного предыдущего маневра) вокруг горизонтальных осей с разной скоростью. После этого, если запустить на расчет модель, она даст, скорее всего, (если вы не меняли никакие настройки регуляторов) сильно колебательный расходящийся процесс, аналогичный рис. 9.3.18.

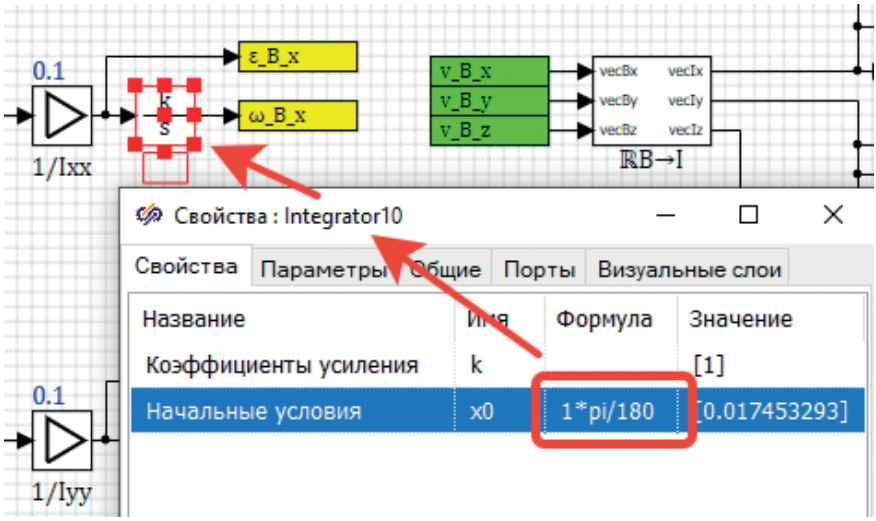


Рис. 9.3.17. Задание ненулевых начальных условий

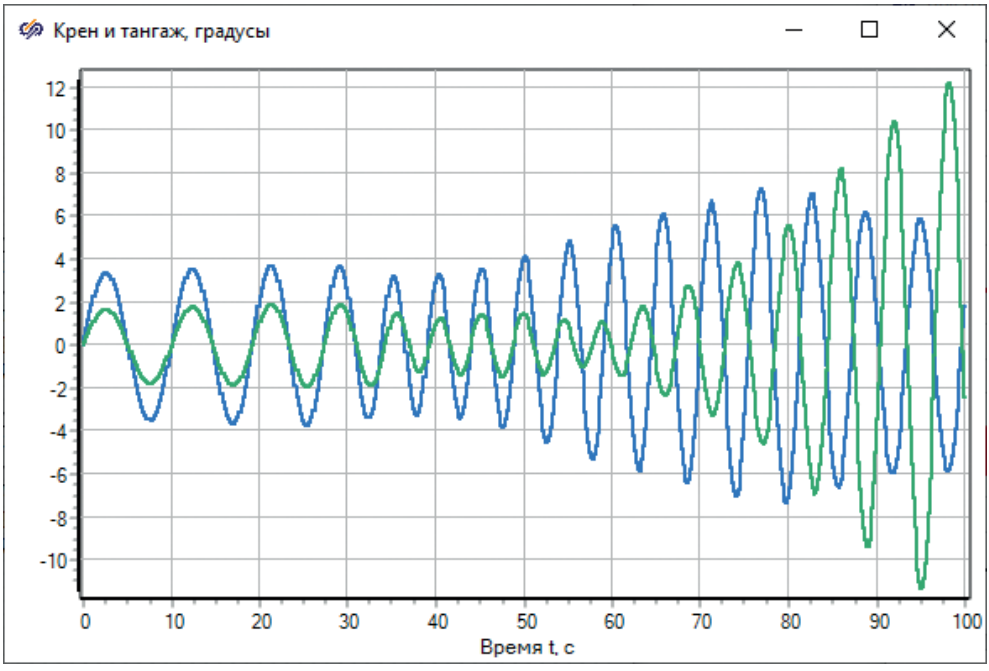


Рис. 9.3.18. «Плохая» отработка регуляторами начального возмущения

Видно, что регулятор пытается стабилизировать коптер, при этом размах качаний под одной из осей сильнее примерно в два раза (в начале переходного процесса), но текущие настройки регулятора не позволяют справиться с этими качаниями, и система идет по пути расходящегося колебательного процесса, т. е. система неустойчива. Вывод: регулятор настроен неудовлетворительно.

Измените коэффициент **0.5** в ветвях Д2 у регуляторов крена и тангажа (рис. 9.3.1 и 9.3.2) на значение, равное **3.0**, и повторите расчет. Сравните результат с рис. 9.3.19. Теперь регуляторы оказались способны взять ситуацию под контроль – т. е. усиление слежения за угловым ускорением в данном случае оказывает хорошее воздействие на устойчивость системы. Более точный подбор коэффициентов – дело будущего, когда модель объекта будет верифицирована на каком-либо реальном аппарате.

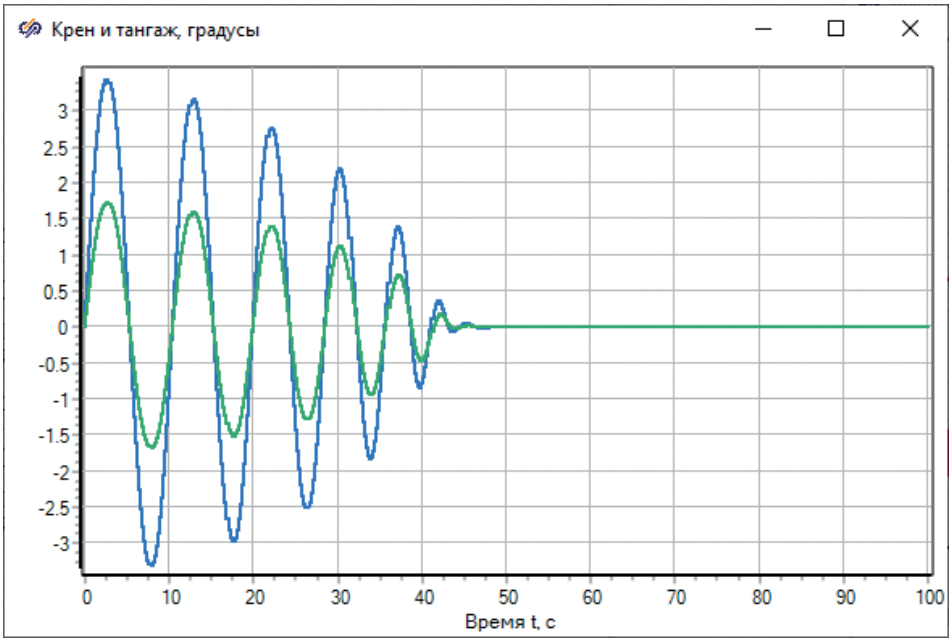


Рис. 9.3.19. Система стабильна

Полученный переходной процесс, представленный на рис. 9.3.19, еще далек от совершенства, можно еще улучшать и улучшать характеристики регулятора. Но, по крайней мере, система стала устойчивой, можно протестировать еще регулятор с другими начальными скоростями или задать скорости нулевыми, а ненулевым выставить один или два угла – и посмотреть, устойчив ли регулятор с такими начальными условиями, а также, многократно перезапуская модель и увеличивая начальный угол или скорость, найти предел устойчивости. Но это все имеет смысл делать на скорректированной модели октокоптера, когда все параметры в модели заданы верно – как коэффициент трения о воздух, так и моменты инерции, масса коптера, длины лучей установки ВМГ и т. д.

Перейдем теперь к реализации задания возмущающих воздействий. Сделаем это таким образом, чтобы разработчик или пользователь модели и регулято-

ров мог непосредственно в процессе моделирования влиять на возмущающие воздействия и задавать их. Например, примем величину внешнего возмущения (Fox(t), Foy(t), Foz(t), Mox(t), Moy(t), Moz(t)) равной ±1 Н, и ±1 Н×м для внешнего момента и наберем схему, которая по нажатию кнопки в соответствующие слагаемые исходных уравнений динамики будет записывать вместо нуля это значение возмущающей силы или момента. В принципе, уже и сейчас можно это делать – заходить в процессе расчета в субмодель возмущающей силы (момента) и в блоке константы записывать какое-то число вместо нуля – прямо в процессе расчета. Скорость расчета при этом лучше выставить синхронной с реальным временем в параметрах расчета проекта, как показано на рис. 9.3.20.

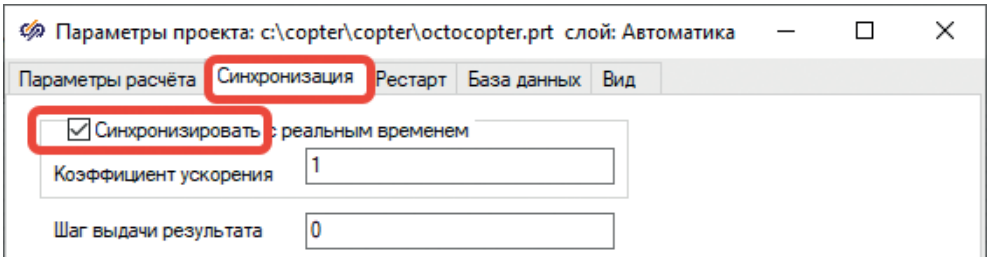


Рис. 9.3.20. Включение синхронизации в параметрах расчета

Но вариант изменения константы неудобен, и лучше сделать это все в отдельном месте в виде мини-пульта управления моделью. Посмотрите на вариант реализации, представленный на рис. 9.3.21. Показано задание только для двух моментов, сделайте аналогично для всех 6 внешних возмущающих моментов. Все блоки тут знакомы, кроме нового блока типа «Кнопка» (библиотека **Ключи**), который работает очень простым образом – на выход подается 0 или 1 в зависимости от настройки кнопки (начального состояния), а в процессе моделирования пользователь модели может нажимать на этот блок, и выходной сигнал блока будет скачком изменяться с 0 на 1 (или наоборот), тем самым изменяя дальнейший сигнал. Затем стоит усилитель, которым можно менять амплитуду воздействия, и сумматор для сложения двух разнонаправленных воздействий.

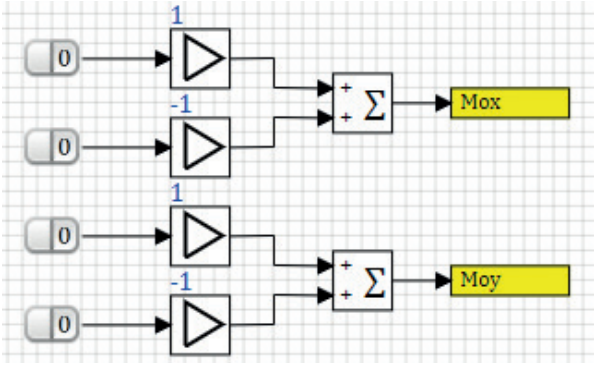


Рис. 9.3.21. Задание внешнего возмущающего момента по оси x

Чтобы полученный сигнал передать в уравнения динамики, поставлен блок типа «В память», а в нужных местах математической модели коптера – в субмоделях Mox(t), Moy(t) и т. д. – поставьте ответные блоки типа «Из памяти», как показано на рис. 9.3.22 (показано для одной субмодели, так надо сделать 6 раз, убрав всего из схемы 6 блоков типа «Константа», которые там стояли ранее).

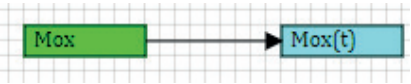


Рис. 9.3.22. Использование возмущающего момента в уравнении динамики

Теперь давайте сделаем график внешних возмущающих воздействий (рис. 9.3.23) в субмодели «Графики», и, запуская на расчет снова модель – пусть даже с ненулевыми начальными условиями по скоростям, – можно нажимать на кнопки внешнего воздействия и наблюдать за раскачкой коптера. Чтобы у блока типа «Временной график» появилось 8 портов, надо зайти в его свойства (через правую кнопку мыши) и задать там 8 вместо 1.

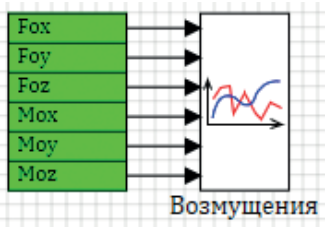


Рис. 9.3.23. Новый график – внешних возмущающих воздействий

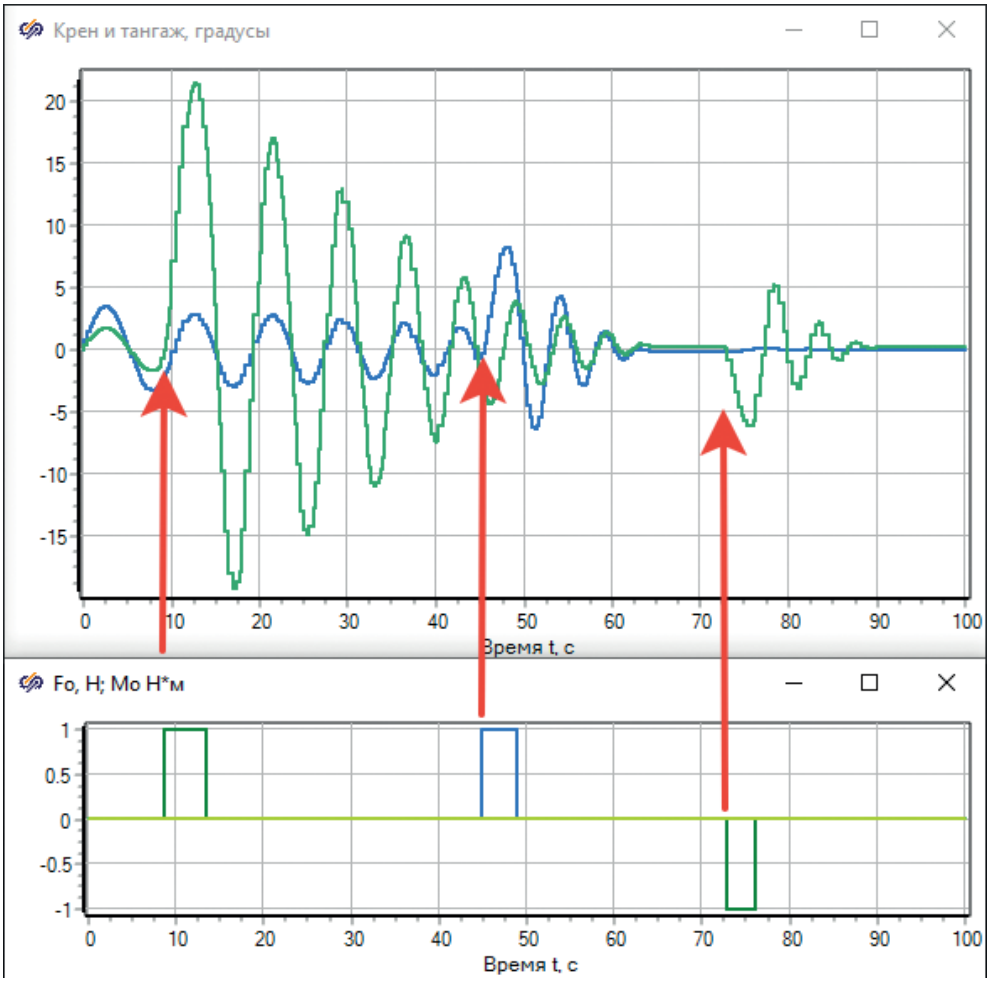


Рис. 9.3.24. Влияние внешнего возмущающего момента на коптер

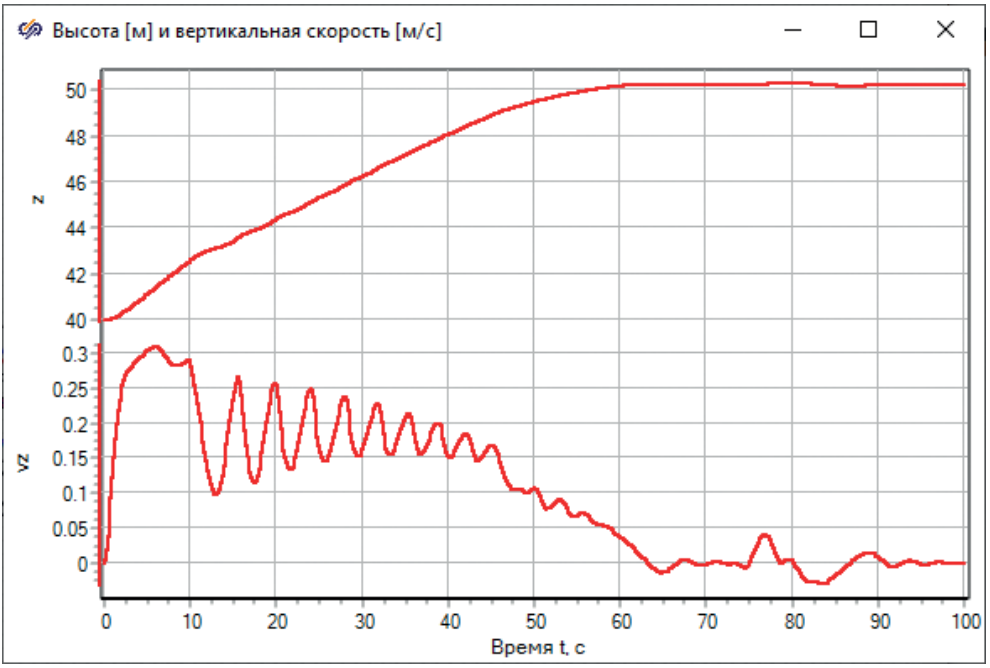


Рис. 9.3.25. Колебания вертикальной скорости и неравномерный взлет

На рис. 9.3.24 представлены совместные графики одного из переходных процессов – в районе 10-й и 45-й- секунды подается воздействие по крену и тангажу, в районе 75-й секунды подается отрицательное воздействие по крену, а на графике углов ориентации видно влияние этих воздействий.

Если посмотреть в этом же переходном процессе на график вертикальной скорости и высоты (рис. 9.3.25), то можно увидеть «просадки» и колебания по вертикальной скорости – в те моменты, когда коптер начал сильно наклоняться, вертикальная сила тяги немного уменьшалась и взлет происходил не так равномерно, как это было без сильных колебаний по высоте (до 10-й секунды). Таким совместным анализом разных графиков можно наблюдать и оценивать верность и физичность набранной модели. Приведем еще график работы двигателей (рис. 9.3.26) в этом же режиме – видна работа регуляторов и «микширование» частоты вращения электродвигателей, направленное на стабилизацию коптера. При этом группы двигателей (четные и нечетные) работают в окрестности своей «базовой» частоты вращения.

Вы можете самостоятельно промоделировать еще несколько режимов, произвольно задавая внешние воздействия (можно уменьшать или увеличивать их силу и продолжительность), и наблюдать по графикам за тем, как модель и регуляторы отрабатывают подаваемые возмущения. При разработке модели это полезно делать, тестируя таким образом модель, – это позволяет лучше понимать динамику объекта, находить скрытые ошибки и недостатки и пределы устойчивости регуляторов таким «опытным» путем, методом прямого моделирования.

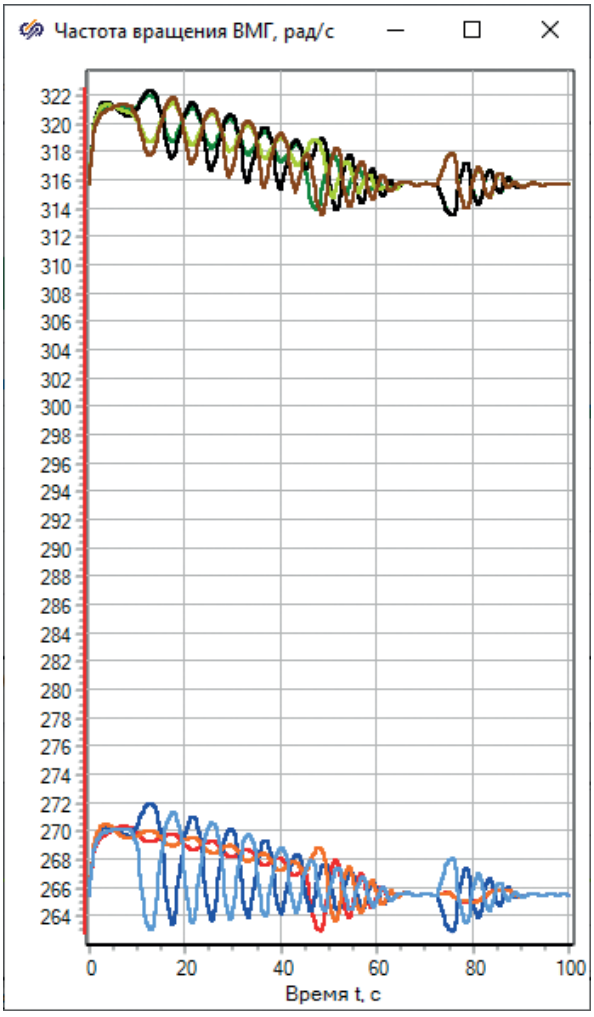


Рис. 9.3.26. Регулирование ВМГ

Если вы посмотрите на график горизонтальных скоростей vx и vy, там тоже можно увидеть колебания – когда коптер наклоняется в ту или иную сторону, суммарный вектор силы тяги поворачивается и, отклоняясь от вертикали, создает еще боковую тягу – и коптер летит в ту или иную сторону. Поскольку сейчас он раскачивается на регуляторах довольно длительное время, этого хватает на то, чтобы сместиться в горизонтальной плоскости на существенное расстояние от начальной точки. Так, отклонения от начала полета в горизонтальной плоскости в нашем случае составили –2.3 и 5.5 м по осям X и Y соответственно (рис. 9.3.27). При этом коптер набрал некоторую горизонтальную скорость, и к 100-й секунде он еще не пришел к равновесию – хотя колебания в целом завершились, но сам коптер продолжает полет…

Для лучшего понимания переходного процесса давайте построим еще три графика типа «фазовый портрет», на котором уже не будет координаты типа «время», а будет в координатных плоскостях X–Z, X–Y и в пространстве X–Y–Z выведена траектория полета коптера.

Точнее, будет выведена траектория его центра масс в пространстве. Сформируйте графики, как показано на рис. 9.3.28. Это блоки типа «Фазовый портрет» и «Трехмерный график» из библиотеки «Вывод данных». Если запустить коптер снова с начальными угловыми скоростями 1 и 2°/с и оставить без внешних возмущающих воздействий, то за 100 с полета он с теми же затухающими колебаниями поднимется с 40 до 50 м (при этом совершая эллиптические облеты вокруг некой равновесной точки), а затем после затухания колебаний с остаточной горизонтальной скоростью наступит более-менее прямолиней-

ная траектория полета, как показано на рис. 9.3.29 и 9.3.30.

Трехмерный график исследуйте самостоятельно… Геометрически, графики рис. 9.3.29 и 9.3.30 являются «проекциями» трехмерной траектории полета октокоптера.

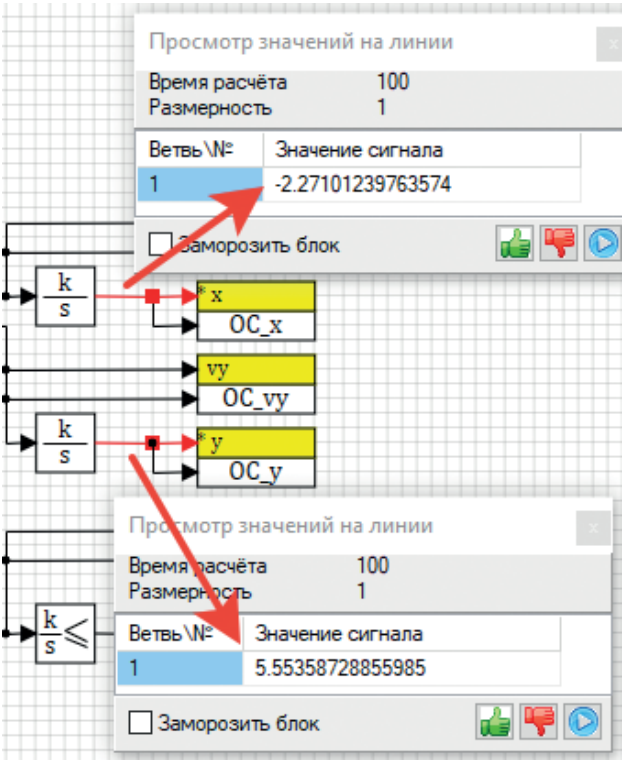


Рис. 9.3.27. Координаты на 100-й секунде полета

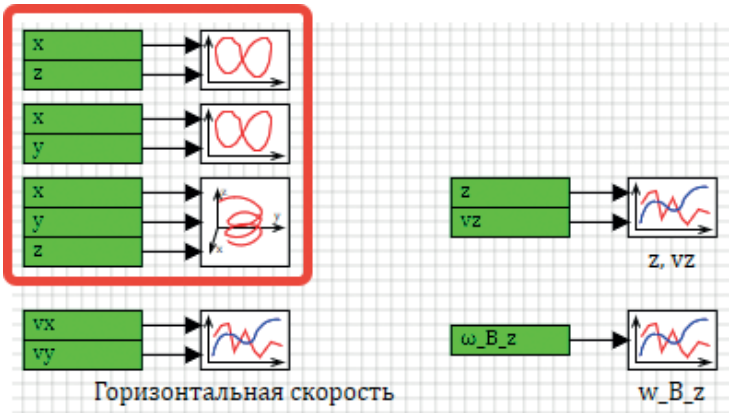


Рис. 9.3.28. Новые графики типа «фазовый портрет»

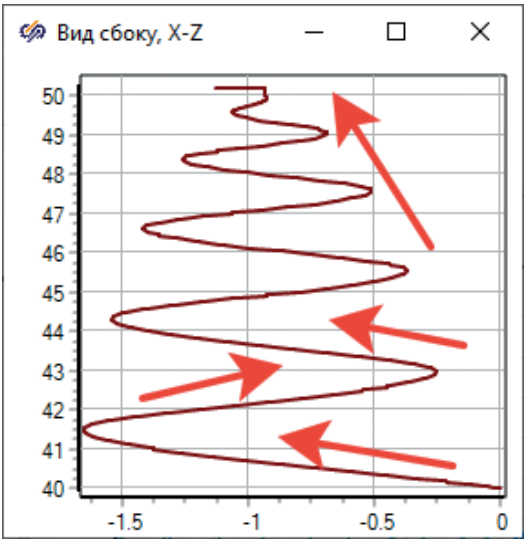


Рис. 9.3.29. Фазовый портрет траектории полета в координатах X–Z

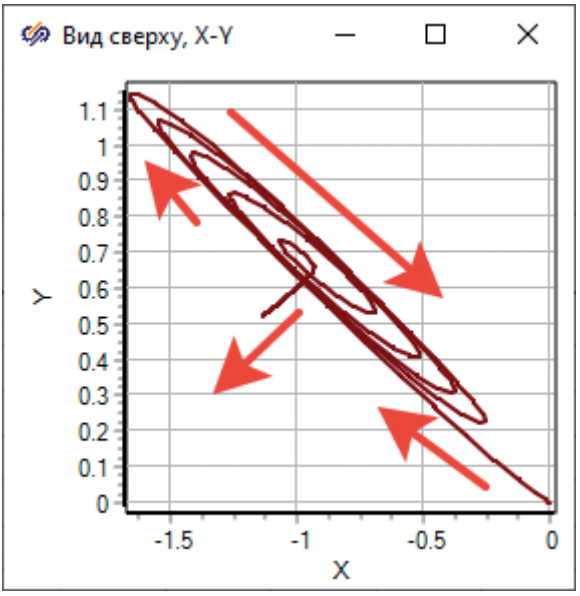
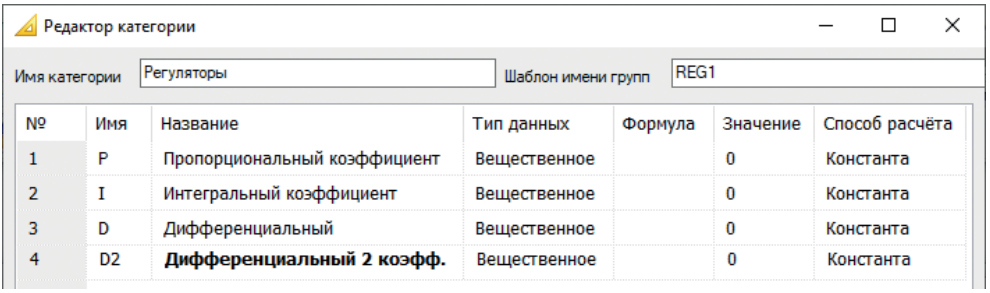


Рис. 9.3.30. Фазовый портрет траектории полета в координатах X–Y

9.3.6. Применение базы сигналов для задания коэффициентов регуляторов

Выполним еще одну доработку в регуляторах – ранее мы при создании базы сигналов создали две категории «ВМГ» и «Коптер», однако такой объект (или составная часть модели), как регулятор – тоже можно описать в понятиях категории и группы сигналов.

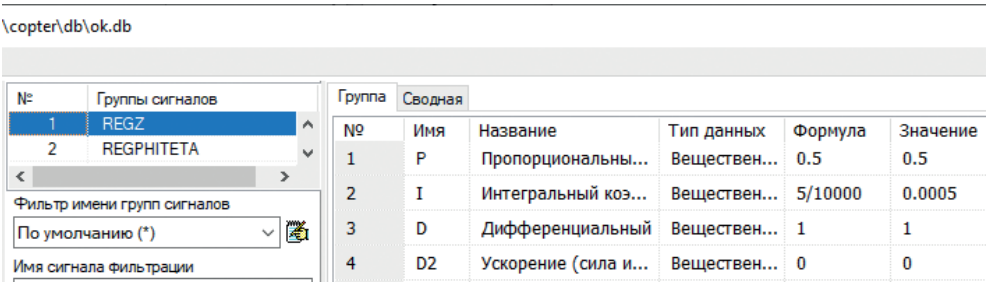
Поскольку мы используем ПИД-регулятор и в регуляторе высоты, и в регуляторах крена и тангажа (последние два практически идентичны даже по схеме), то настроечные коэффициенты ветвей регуляторов, заданные пока что числами на схеме, можно вынести в константы базы сигналов – для того чтобы в дальнейшем при итерациях настройки регуляторов это можно было делать централизованно в интерфейсе базы сигналов, а не перемещаясь по схеме и выискивая там нужные блоки. Давайте добавим по 4 сигнала к каждому регулятору с именами **P**, **I**, **D**, **D2**. Для этого нужно перейти в редактор базы сигналов через главное меню (**Инструменты → База данных...**), создать новую категорию «Регуляторы» и добавить там 4 сигнала, как показано на рис. 9.3.31.



№	Имя	Название	Тип данных	Формула	Значение	Способ расчёта
1	P	Пропорциональный коэффициент	Вещественное		0	Константа
2	I	Интегральный коэффициент	Вещественное		0	Константа
3	D	Дифференциальный	Вещественное		0	Константа
4	D2	**Дифференциальный 2 коэфф.**	Вещественное		0	Константа

Рис. 9.3.31. Категория «Регуляторы»

Далее, в этой категории создайте две группы сигналов с именами **REGZ**, **REGPHITETA** – тем самым мы добавили к модели 8 новых констант с именами REGZ_P, REGZ_I, REGZ_D, REGZ_D2 и т. д. Затем задайте числа для этих 8 коэффициентов регуляторов аналогично рис. 9.3.32 и 9.3.33.



№	Имя	Название	Тип данных	Формула	Значение
1	P	Пропорциональны...	Веществен...	0.5	0.5
2	I	Интегральный коэ...	Веществен...	5/10000	0.0005
3	D	Дифференциальный	Веществен...	1	1
4	D2	Ускорение (сила и...	Веществен...	0	0

Рис. 9.3.32. Коэффициенты регулятора высоты

После задания констант ими можно воспользоваться в модели – перейдите к регуляторам и в соответствующих блоках – в свойствах усилителей и интеграторов – задайте коэффициенты уже не численно, а при помощи этих новых констант аналогично рис. 9.3.34. Это надо проделать в 11 блоках – три блока у регулятора высоты, и по 4 блока у регуляторов крена и тангажа. После чего, уже меняя коэффициенты в базе данных, они распространятся на расчетную схему.

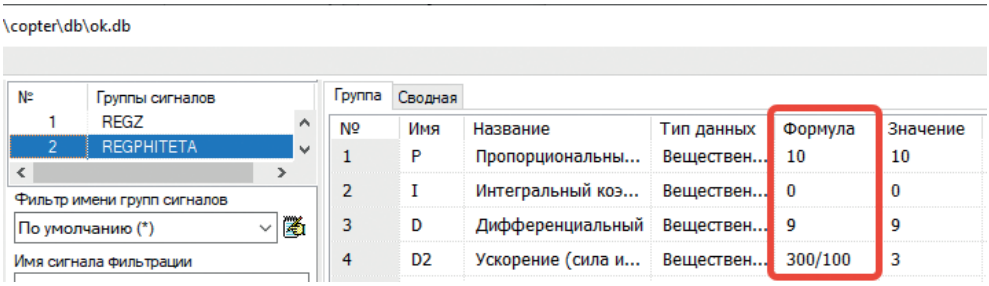


Рис. 9.3.33. Коэффициенты регуляторов крена и тангажа

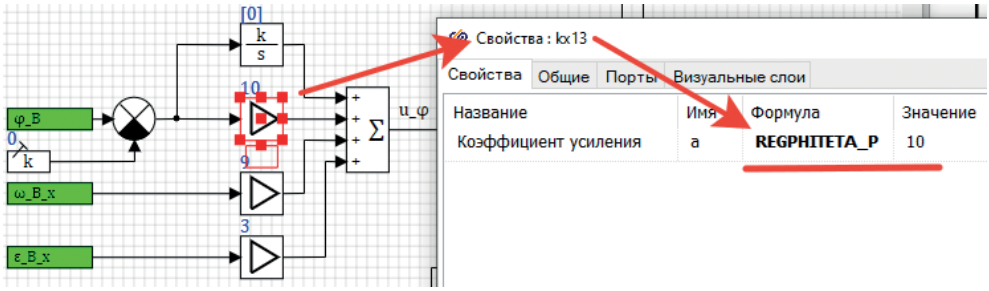


Рис. 9.3.34. Использование констант в настройке регуляторов

Поскольку регуляторы крена и тангажа одинаковы, то теперь для подстройки их коэффициентов достаточно менять одну группу сигналов – и эти изменения будут применяться автоматически к двум участкам расчетной схемы, что удобно.

9.4. Структуризация схемы

Давайте, прежде чем идти далее, приведем схему к аккуратному виду – все, что относится к регулятору, перенесите в свою субмодель с подписью «Регуляторы», все, что относится к модели коптера, – в субмодель «Модель октокоптера». Создайте еще одну субмодель «Управление» и перенесите туда недавно созданные кнопки и формирование возмущающих воздействий, как представлено на рис. 9.4.1.

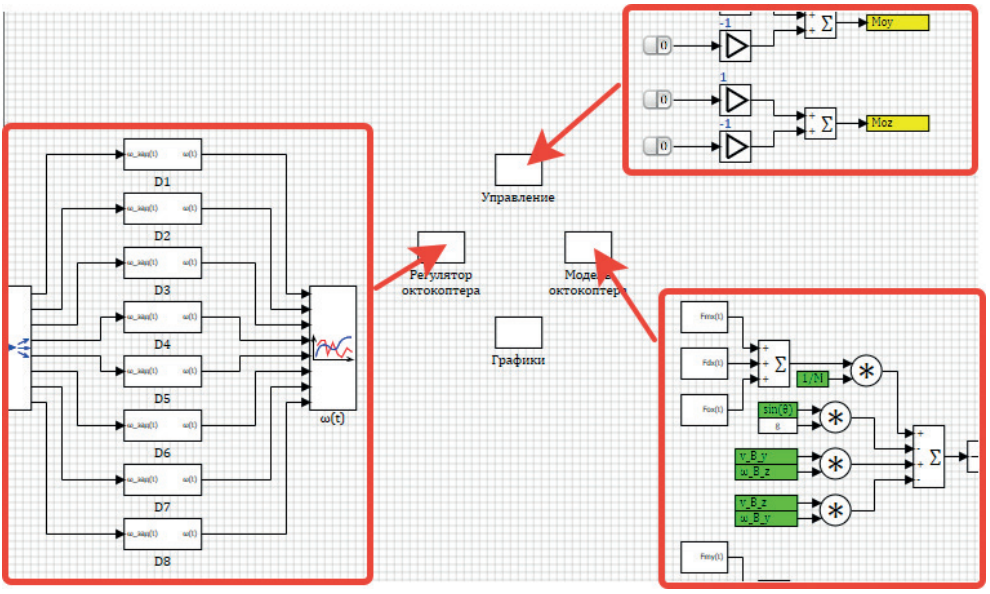


Рис. 9.4.1. Перенос разных (по смыслу) частей схемы в свои субмодели

Переносить проще всего, выделив участок схемы рамкой, потом вырезать кусок схемы, перейти в нужную субмодель и там вставить вырезанную ранее схему.

В итоге на верхнем уровне в вас останется 4 субмодели, а все смысловые – математические – блоки будут размещены на втором уровне вложенности. Можно пойти далее и, например, каждое из уравнений динамики перенести в свою субмодель, на третий уровень вложенности. А каждый из каналов регулирования – тоже в свои субмодели. Можете сделать это самостоятельно.

9.4.1. Рамка, штамп

Кратко рассмотрим еще вариант оформления расчетной схемы – добавление рамки или штампа с основной надписью.

На расчетной схеме SimInTech можно произвольным образом при помощи графических элементарных объектов (так называемых графических примитивов) создавать те или иные графически изображения. За счет механизма визуальных «слоев» созданные изображения можно вынести на свой слой и, сделав его неактивным, перенести графические нерасчетные объекты на свой слой, чтобы они не мешались при дальнейшей разработке схемы, а только лишь для оформления схемы.

Делается это таким образом. В свободном месте расчетной схемы (в свободном месте от расчетных блоков) разместите примитив типа «Прямоугольник» (рис. 9.4.2) и задайте размер этого примитива, равный 1280×912 (рис. 9.4.3). Размер именно такой обусловлен практическим опытом, что при распечатке схемы на принтере формата А3 такая рамка умещается на лист с учетом полей и настроек принтера. На разных принтерах это может варьироваться в некоторых пределах. Также ширина и высота кратны 8 – это стандартный шаг модельной сетки в SimInTech.

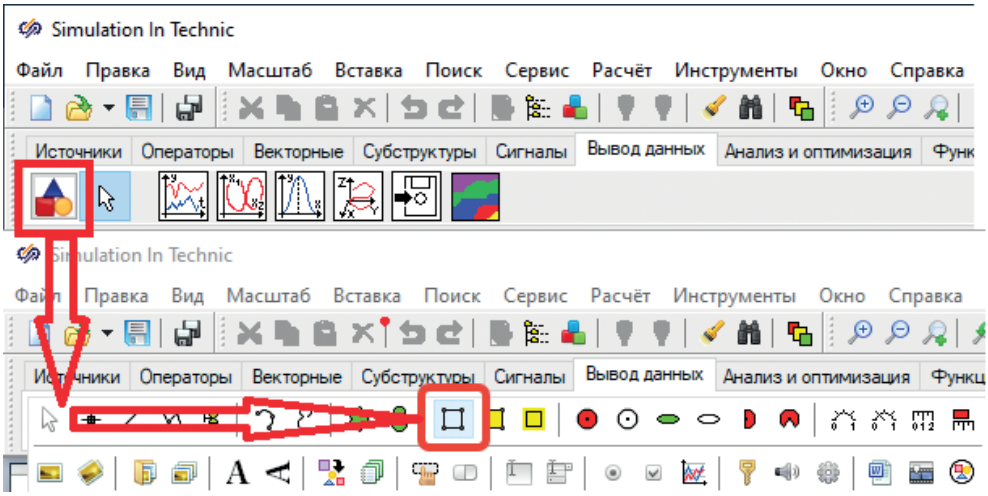


Рис. 9.4.2. Графический примитив «Прямоугольник»

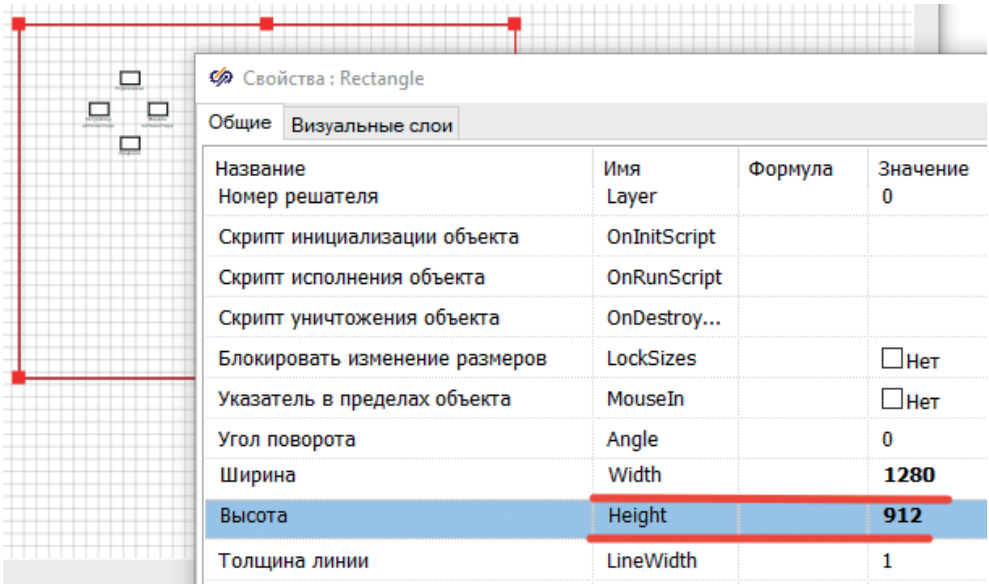


Рис. 9.4.3. Прямоугольник размера 1280×912 на расчетной схеме

Далее, разместите примитив типа «Прямая» в нижней части прямоугольника от левого края до правого края, создав прообраз основной надписи, и еще один примитив типа «Текст», вписав в него строку, например «Октокоптер», и задав размер шрифта, равный 20 пунктам. Все примитивы – это тоже блоки (только не расчетные), и каждый из них обладает рядом свойств, доступных по правой кнопке на примитиве, и далее свойства редактируются так же, как и для всех остальных блоков.

После формирования «штампа» выделите сразу все три размещенных объекта и выберите в главном меню пункт **Правка → Собрать в группу** (рис. 9.4.4).

Далее, когда из трех примитивов получится группа объектов (как бы блок с графическим изображением, составленным из трех примитивов), надо зайти в редактор изображения этого блока и задать там тоже вручную размеры графической области тем же размером 1280×912 (рис. 9.4.5) и переместить потом графические примитивы – уже в графическом редакторе – так, чтобы они точно вписывались в эту область (именно это состояние показано на рис. 9.4.5). Закрыть редактор графического изображения с сохранением изменений. После этого, возможно, изображение на схемном окне изменится, и надо будет снова установить размер объединенной группы блоков в 1280×912. Подробнее это будет объяснено дальше, а пока что попробуйте самостоятельно проделать эти эволюции. На выходе вы должны получить группу из трех примитивов одним блоком на схеме с размером 1280×912.

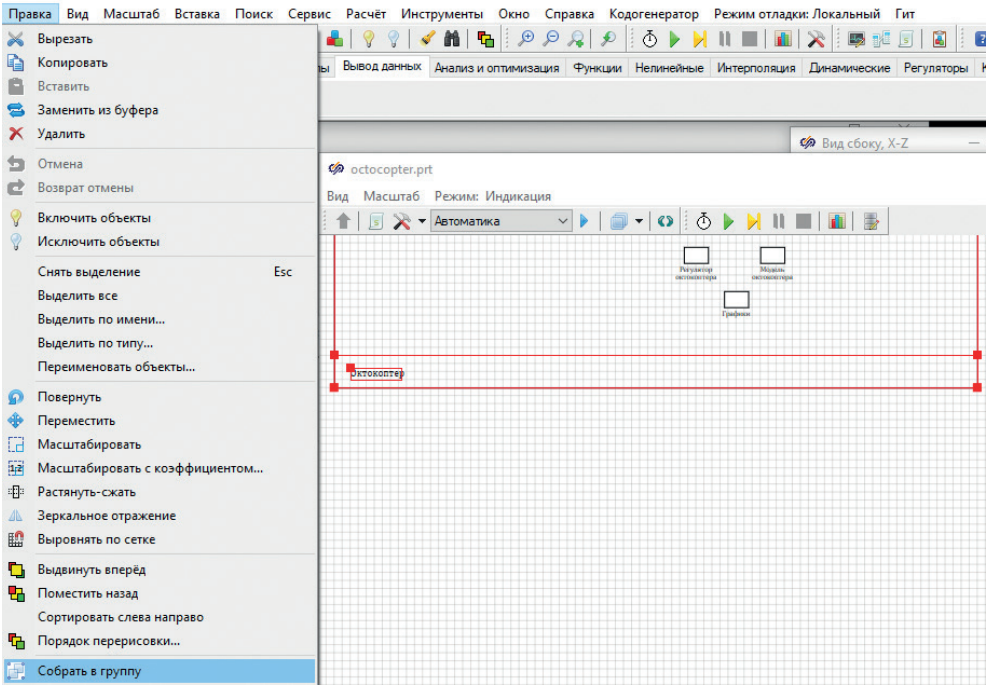


Рис. 9.4.4. Объединение трех графических примитивов в группу

После успешного объединения и задания размеров мы сделали в самом простом варианте нерасчетный блок, который изображает рамку и основную надпись (можно было бы сделать более сложный вариант). Если у вас не получилось, – можете взять из библиотеки блоков подготовленную заранее рамку формата А3 (библиотека «Форматы ГОСТ», она находится почти в самом конце списка библиотек). Дальнейшие манипуляции возможны и с типовым блоком. Типовые блоки сделаны сходным образом.

После размещения на схеме блока зайдите через панель инструментов схемного окна в слои (рис. 9.4.6) и для последнего 16-го визуального слоя задайте имя рамки и сделайте его неактивным, как показано на рисунке. Это означает, что все блоки, которые будут отнесены к этому слою, будут оставаться видны

на схеме, но к ним нельзя будет никак обратиться – ни выделить мышкой, ни переместить, ни поменять свойства и т. д. Как раз то, что нужно для нашего блока-рамки.

Далее, выделите новый блок и зайдите в свойства блока, во вкладку **Визуальные слои**, и там (рис. 9.4.7) снимите галочку с 1-го слоя и поставьте галочку напротив 16-го слоя. После этого с блоком на схеме нельзя будет ничего сделать… Разместите 4 субмодели верхнего уровня поверх блока-рамки. Это будет как бы первый «лист» модели.

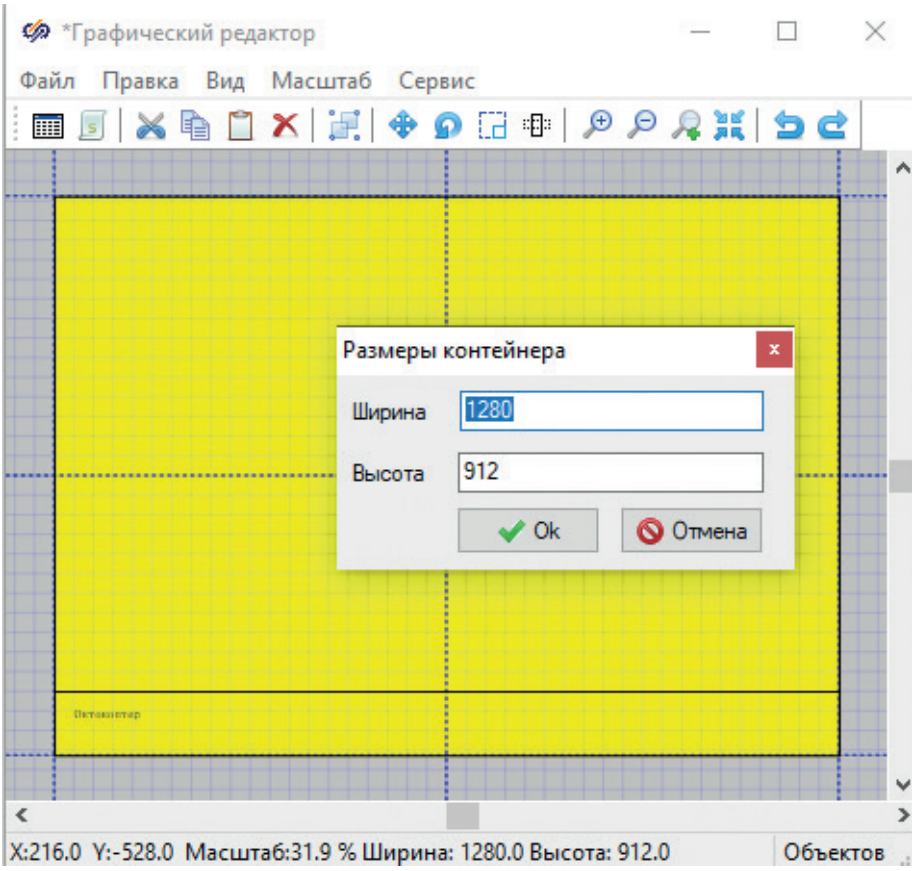


Рис. 9.4.5. Рамка вписана в графическую область изображения блока

Для того чтобы рамку распространить на остальные субмодели, надо снова сделать активным 16-й визуальный слой проекта, скопировать в буфер обмена блок-рамку. Далее, сделать текущим (в слоях проекта) именно 16-й слой и дальше N раз вставить блок внутри нужных субмоделей – например, 4 раза, а может быть, и внутри субмоделей 3-го уровня. Вставлять и размещать на схеме следует таким образом, чтобы расчетные блоки оказывались внутри рамки. После того как все блоки были вставлены, надо в слоях проекта сделать текущим снова 1-й слой, а 16-й слоя сделать неактивным. На этом все, вы сделали базовое «оформление» для листов математической модели.

Дальнейшая работа с рамками должна проходить также – делаете активным 16-й слой, меняете что-то в плане рамок, потом слой снова выставляете неактивным. В принципе, иногда такой прием работы бывает удобен и с расчетными блоками – если какую-то часть проекта надо «заморозить», чтобы она уже не менялась (например, некая защита от случайных нажатий на расчетную схему при интенсивной работе с ней).

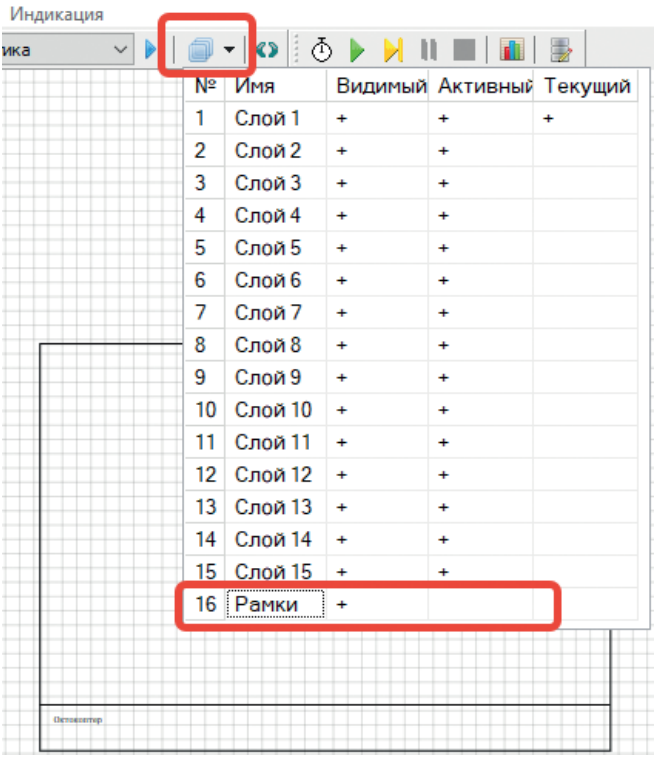


Рис. 9.4.6. Визуальные слои проекта

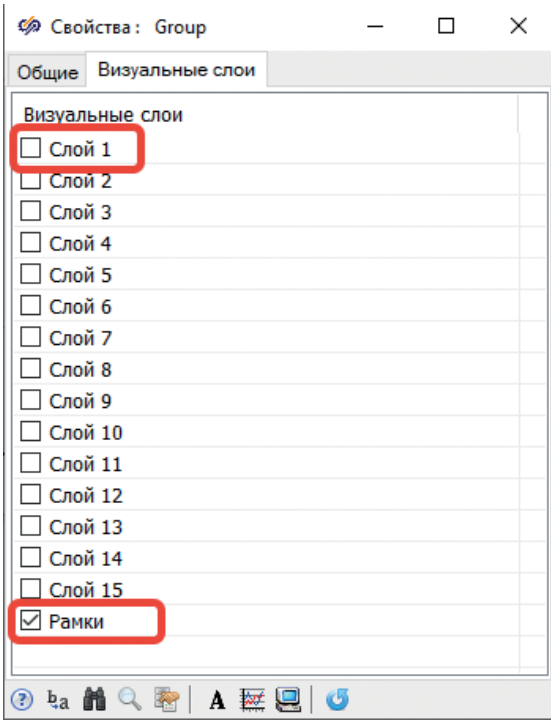


Рис. 9.4.7. Визуальные слои блока

Если вы еще не разделили модель объекта на следующие уровни вложенности, проще будет это сделать следующим образом. Сделать это желательно, так как на один лист модель, скорее всего, и у вас тоже не умещается. Например, разнесем 6 уравнений в свои 6 субмоделей.

Для этого проще сначала подготовить одну субмодель с рамкой внутри, подписать ее, например, как ωBx, и далее раскопировать этот – пустой еще – блок 6 раз. Получив таким образом 6 субмоделей с пустыми листами. Далее поменяйте подписи каждой субмодели на ωBx, ωBy, ωBz, vBx, vBy, vBz и затем аккуратно перенесите путем вырезания и вставки каждое из уравнений в свою субмодель, см. рис. 9.4.8.

После этого вся модель уместится на одном листе, при этом надо помнить, что 6 уравнений выполнены на следующем уровне вложенности, на котором есть еще 4-й уровень вложенности для слагаемых уравнений сил и моментов (это мы делали ранее). Уровень вложенности может быть неограничен, а вся структура проекта представлена в интерфейсе **Поиск → Структура** главного меню. Иногда бывает удобнее там осуществлять навигацию по схеме, если корректно заданы имена субмоделей (в данной учебной модели мы этого не делаем, так как количество субмоделей не очень велико, и навигацию можно успешно осуществлять непосредственно на схеме).

Многие пользователи не уделяют внимания оформлению расчетной схемы – действительно, на математику и результаты расчета оформление не влияет. Но, как правило, верное и лаконичное оформление позволяет: а) находить скрытые ошибки, проводя ревизию схем; б) гораздо проще через время работать с моделью, когда подробности бывают забыты; в) с оформленной схемой удобно работать и другому человеку, который не является разработчиком.

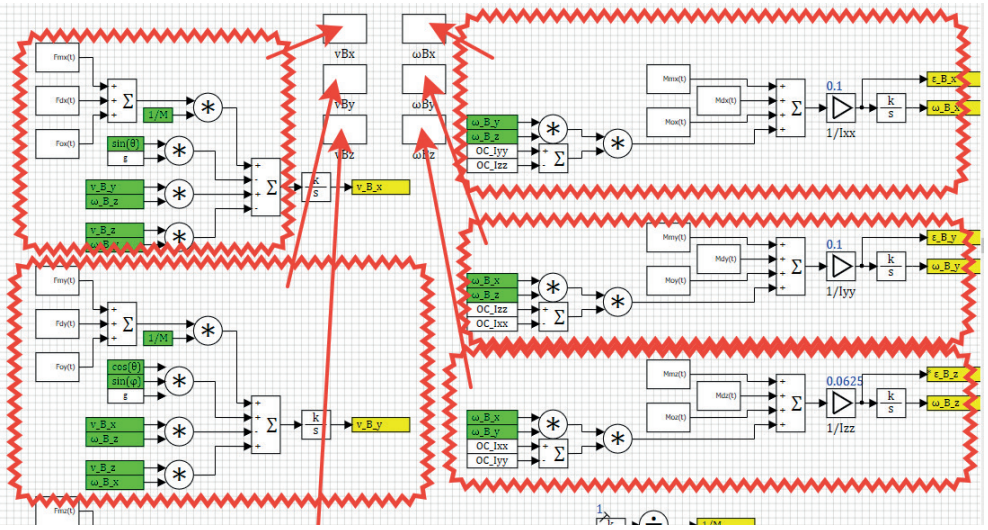


Рис. 9.4.8. Перенос каждого уравнения в свою субмодель

9.5. Регулятор по направлениям X и Y

Теперь, когда коптер у нас стабилизирован (правда, пока с низким качеством, на три с плюсом, но все же), можно перейти к построению регулятора положения коптера в пространстве, чтобы пилот мог задавать конечные координаты, а коптер сам летел к ним.

При этом все было бы хорошо и можно было бы регулятор сделать довольно просто, если бы не два обстоятельства: направление осей X и Y в системах **B** и **I** далеко не всегда совпадает – это выполняется только при нулевом угле курса $\psi(t) = 0$. Также способность ВМГ сдвигать коптер по горизонтальным осям очень мала (см. коэффициенты матрицы A+, первые две колонки представлены с очень большими числами, т. е. для создания даже малого бокового усилия без наклона коптера требуется очень сильно менять частоты вращения двигателей ВМГ, что приведет к низкому качеству переходного процесса – низкие скорости, малые запасы устойчивости и способность сопротивляться внешним возмущениям – ветру и т. п.).

Поэтому в идею регулятора по направлениям заложены два дополнения – вначале, в зависимости от текущего курса коптера (этот угол показывает, насколько повернута система B относительно системы I, если смотреть сверху),

произведем пересчет заданных координат из системы I в систему B – потому что оператору коптера удобнее работать в неподвижной системе I, а регулятор управляет коптером в системе отсчета, связанной с ним. Второе дополнение – по насчитанным Δx_B и Δy_B мы не будем формировать дополнительные $\Delta\omega$ для двигателей, а вычислим соответствующий угол наклона по крену или тангажу, который передадим на вход в регулятор как заданный угол, вместо нуля, который там сейчас задан константой. Таким образом, перемещение коптера в горизонтальной плоскости будет осуществляться через каналы регулирования по крену и тангажу.

9.5.1. Новые сигналы

Добавьте в базу сигналов, в шаблон категории «Коптер» три новых вещественных сигнала с именами xzad, yzad, zzad – это будут переменные под заданную координату в пространстве, которую оператор будет задавать с пульта управления в дальнейшем. В модели при этом должны появиться три новых сигнала с именами **OC_xzad**, **OC_yzad** и **OC_zzad**.

Если делать все правильно, то пришлось бы набирать схему преобразования вектора-рассогласования текущих координат коптера (измеренных) от координат заданных, как показано на рис. 9.5.1 и 9.5.2, а также реализовать матрицу преобразования $\boldsymbol{R}_{IB}$ – это транспонированная матрица $\boldsymbol{R}_{BI}$ (см. рис. 3.1.1 и 7.13.2). Но нам потребуется только часть от этой схемы, так как в обучающем курсе пойдем по облегченной схеме – предположим, что курс коптера всегда нулевой, т. е. регулятор курса всегда держит коптер в том же направлении, в котором расположена и инерциальная система координат. Тогда нам потребуется только часть схемы рис. 3.1.1, а именно – вычисление x_I_зад и y_I_зад. Схему лучше набирать сразу внутри субмодели регулятора.

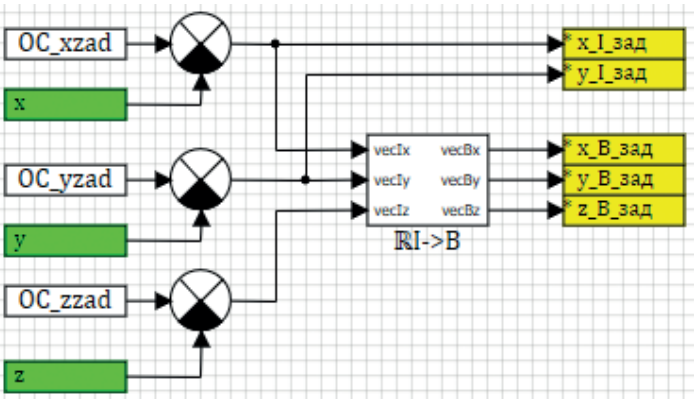


Рис. 9.5.1. Получение рассогласования координат в системе координат, связанной с коптером

Матрицу преобразования (если ее делать) лучше выполнить внутри своей субмодели аналогично тому, как ранее мы делали другую матрицу $\boldsymbol{R}_{BI}$.

9.5.2. Вычисление заданных углов наклона

Путем несложных выкладок [1] для предположения, что ψ(t) = 0, можно получить следующие формулы для вычисления угла наклона:

$$\varphi_{зад} = \arcsin\left(y_{задI}\right),$$
$$\theta_{зад} = \arcsin\left(\frac{x_{задI}}{\cos\left(\varphi_{зад}\right)}\right)$$

где $y_{задI}$ и $x_{задI}$ – рассогласования заданных значений координат, измеренных в инерциальной системе отсчета. Выражения под арксинусом следует дополнительно ограничить некоторой величиной (полученной исходя из запасов управляемости коптера по крену и тангажу), например диапазоном $\left[-\frac{\pi}{16};+\frac{\pi}{16}\right]$, для того чтобы заданный угол был небольшим и у регуляторов крена и тангажа оставался бы достаточный запас на стабилизацию коптера.

Если такие величины $\varphi_{зад}$ и $\theta_{зад}$ подать вместо нулевого значения на вход регуляторам крена и тангажа, то регулятор будет дополнительно доворачивать коптер на небольшой угол для полета вдоль своих осей X и Y. Если учитывать еще и возможность изменения курса, то формулы получатся сложнее, но смысл управления будет тем же самым.

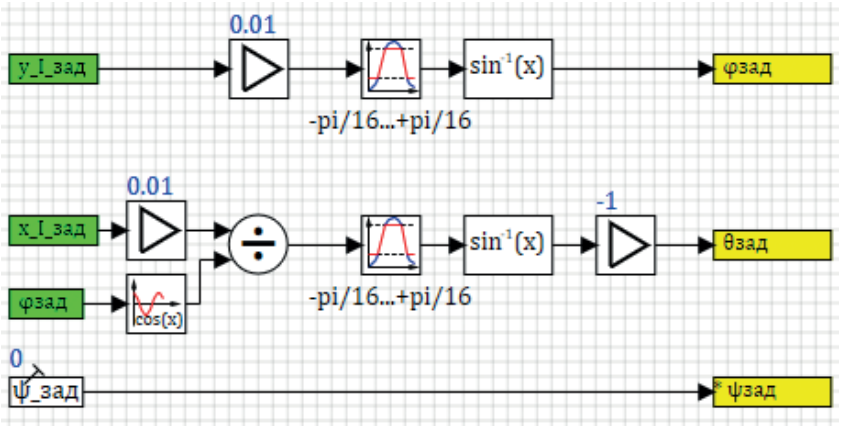


Рис. 9.5.2. Формирование заданных углов наклона

Реализация такого управления показана на рис. 9.5.2, –1 использована, чтобы привести в соответствие направления осей и поворота коптера вокруг оси Y. Усилители с коэффициентами 0.01 поставлены и подобраны таким образом, чтобы максимальный угол наклона формировался только при рассогласованиях более 20 м. Это настроечные вещи и подбираются под конкретную модель аппарата.

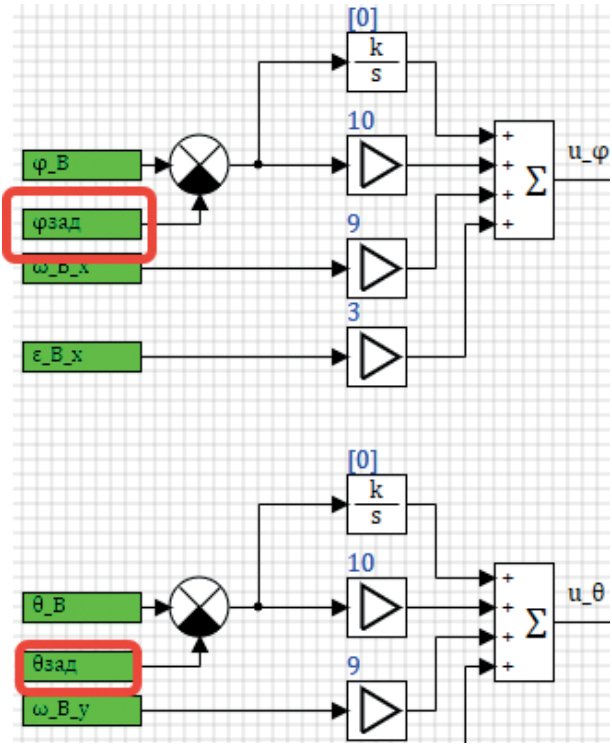


Рис. 9.5.3. Доработка регуляторов крена и тангажа

Таким образом, эта часть схемы будет формировать заданные углы крена и тангажа в размере от –11.25 до +11.25° при отклонении коптера от заданной позиции от –20 до +20 м. Далее, вычисленные заданные углы надо подать на входы регуляторов крена и тангажа вместо 0, как показано на рис. 9.5.3. После этого, чтобы наши новые идеи начали работать правильно, надо «заморозить» вычисление курса коптера (пока у нас еще не реализован регулятор курса), поставив там вместо производной 0, как показано на рис. 9.5.4.

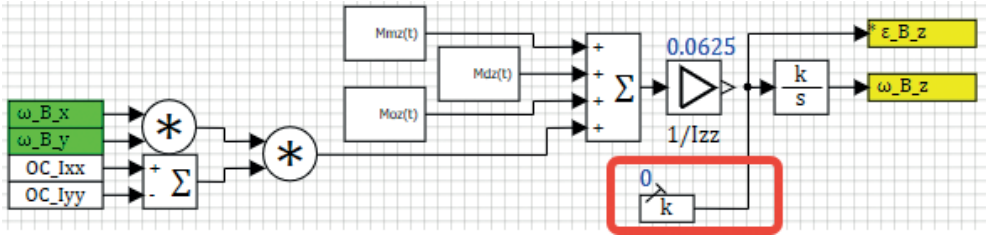


Рис. 9.5.4. Обнуление курса

После этих манипуляций можно запустить схему на расчет и увидеть, что вместо эллипсов вокруг непонятного (случайно получившегося) центра коптер после начального возмущения – заданного ненулевыми угловыми скоростями – примерно также, как и раньше, отклонится от начала координат, но постепенно будет возвращаться к началу координат, так как заданные координаты по осям пока что нулевые, см. рис.9.5.5.

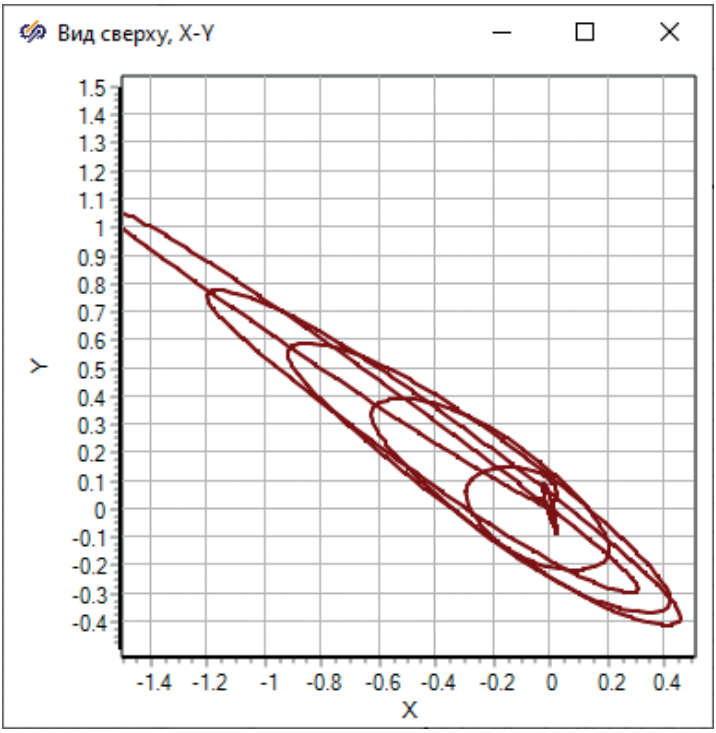


Рис. 9.5.5.

Наблюдение за поведением коптера по графикам представляет собой некоторую сложность, так как по плоским и даже по трехмерной фазовой траектории не очень удобно и наглядно следить за аппаратом. В SimInTech есть возможность создавать техническую анимацию, и в следующей главе мы ее реализуем, чтобы за коптером было удобнее наблюдать и отлаживать регуляторы. А пока что будем довольствоваться графиками. Можете самостоятельно пронаблюдать за изменившимся поведением коптера. Также будет полезно задать конечную точку (заданную) с другими координатами, например +10 м по оси X и +15 м по оси Y, и пронаблюдать за тем, как движется октокоптер (см. рис. 9.5.6).

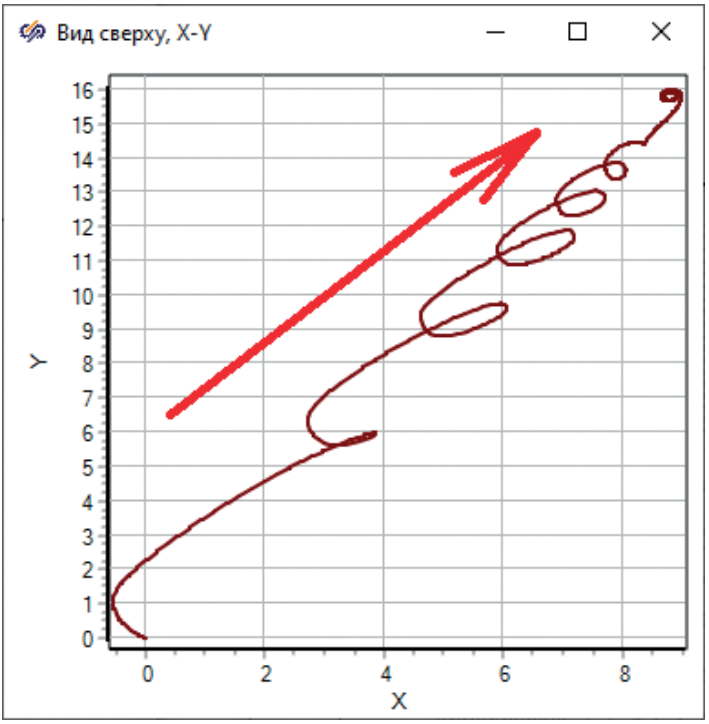


Рис. 9.5.6. Движение коптера в сторону точки с координатами (+10, +15)

При этом внимательный читатель увидит, что коптер, помимо того, что у него проблемы с раскачиваниями (они идут от не вполне отлаженных регуляторов крена и тангажа), переместился не совсем точно, значит, где-то в регуляторе допущена ошибка, но это уже другая история…

9.6. Регулятор курса

Регулятор курса структурно похож на регулятор крена и тангажа: хотя управляемость по курсу у данной конструкции невелика, однако ее будет достаточно для успешного управления, так как не предполагается каких-либо сильных возмущений по курсу коптера (закручивающих его вокруг вертикальной оси). Будем предполагать, что заданный угол по курсу будет нулевым, для того чтобы упрощенная модель регуляторов по направлениям работала корректно.

Внешний вид регулятора курса представлен на рис. 9.6.1.

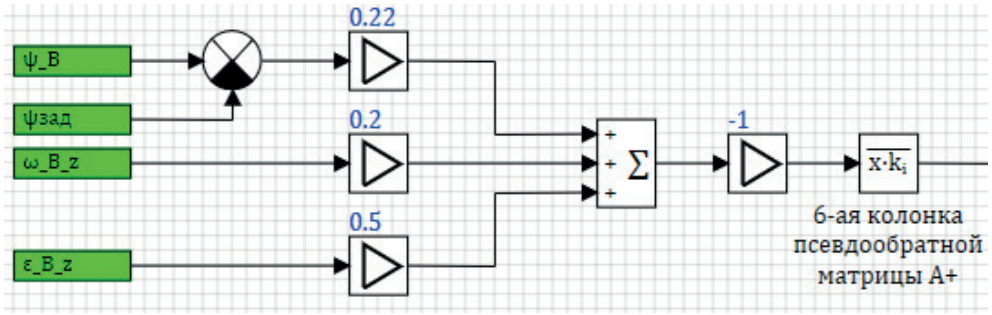


Рис. 9.6.1. Регулятор курса

Заданный угол (нулевой) был определен на рис. 9.5.2, остальные входные сигналы берутся из математической модели динамики коптера. Коэффициенты П-ветви, Д-ветви и Д2-ветви задайте через базу сигналов аналогично тому, как мы это делали ранее. Для этого надо добавить группу сигналов с новым именем, например **REGPSI** в категорию «Регуляторы», и присвоить сигналам **REGPSI_P** = 0.22, **REGPSI_D** = 0.2, **REGPSI_D2** = 0.5, см. рис. 9.6.2, а затем эти сигналы подставить в блоки-усилители на схеме рис. 9.6.1. В размножитель следует вставить вектор-столбец (6-й) из матрицы A+, равный [–67.8562, 67.8562, –67.8562, 67.8562, –67.8562, 67.8562, –67.8562, 67.8562].

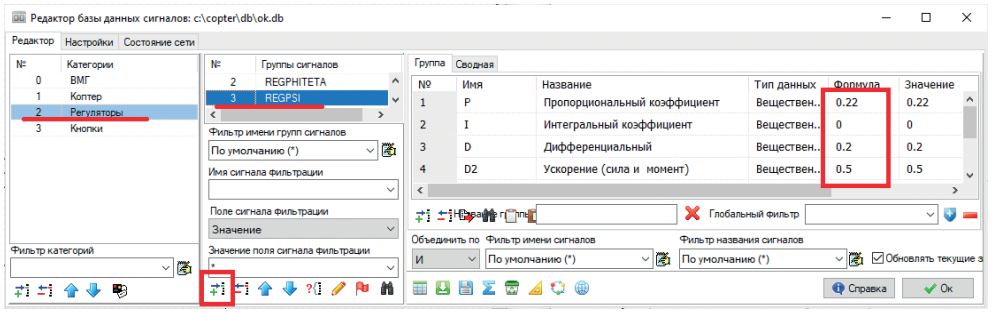


Рис. 9.6.2. Новая группа сигналов для регулятора курса

После набора схемы рис. 9.6.1 следует выходной вектор соединить, сложить с другими выходами регуляторов по другим направлениям. Для этого в сумматоре надо добавить еще один входной порт, расширив на один коэффициент свойство коэффициентов усиления, и завести выход регулятора курса на новый входной порт сумматора.

Дальше надо убрать «заморозку» с угла курса и угловой скорости, которую мы делали ранее для тестирования регуляторов по направлению, и попробовать запустить заново режим тестового моделирования, задав для начала координаты X и Y нулевыми, а потом можно посмотреть, что будет, если их задать ненулевыми. Проведите это тестирование самостоятельно.

Теперь, когда все степени свободы коптера «разморожены», т. е. честно считаются все уравнения динамики, и регуляторы по всем каналам управления также «в строю», можно переходить к следующему большому этапу – верификации

математической модели коптера и дальнейшей настройке регуляторов. Верификацию можно выполнить только на базе экспериментов – поэтому мы зададимся некоторыми из коэффициентов, опираясь на измерения из работы [1], а настройка регуляторов будет заключаться в дальнейшем подборе коэффициентов (после доводки модели) и, возможно, в корректировке схем регуляторов по направлениям и по высоте – добавим к регуляторам ограничитель скорости. Это можно делать уже сейчас, по графикам, но делать это будет гораздо удобнее, если реализовать для коптера небольшую анимацию, чтобы визуально видеть текущее состояние коптера – высоту, положение X–Y в плане (на виде сверху) и текущие углы наклона (крен и тангаж) на виде сбоку. Поэтому дальнейшая модификация регуляторов будет рассмотрена после 5-го раздела.

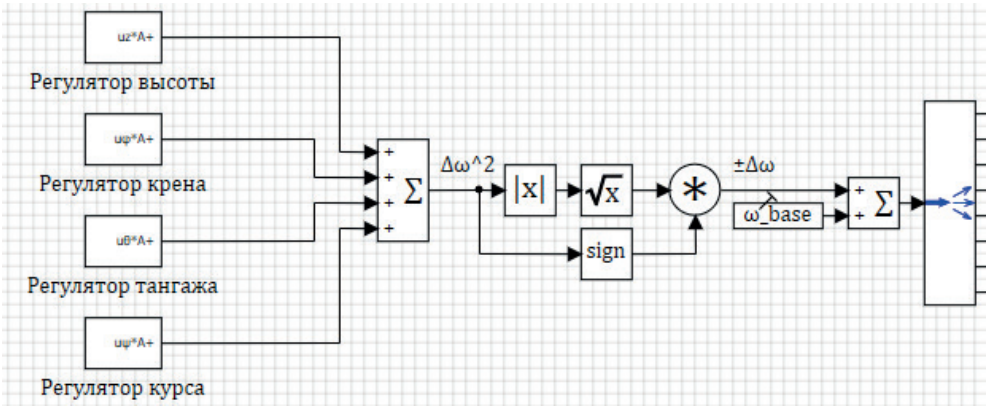


Рис. 9.6.3. Внешний вид регуляторов по направлениям

Сейчас можно только еще немного структурировать регулятор, убрав каждый из регуляторов (каждое из направлений) в свою субмодель, тогда результат у вас будет похож на рис. 9.6.3.

Напомним, что всего здесь на самом деле 6 регуляторов, просто регуляторы по направлениям X и Y «работают» через задание угла для регуляторов крена и тангажа.

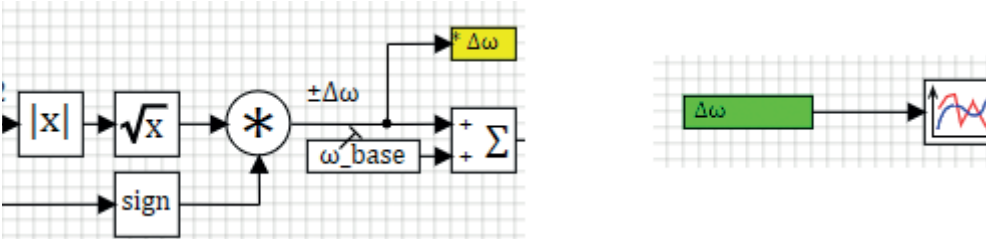


Рис. 9.6.4. Новый график для анализа переходных процессов

Видно, что все регуляторы задают некоторую «добавку» к угловым частотам вращения двигателей относительно базовых частот вращения, которые мы задали постоянной величиной. Для анализа режимов может быть полезным построить график именно этих добавок – чтобы видеть, насколько сильно тот или

иной канал регулирования «отклоняет» винтомоторную группу от ее равновесной частоты вращения. Для этого можно нужную линию связи блоком «В память» определить, а в субмодели, где мы строим графики, – создать нужный график, см. рис. 9.6.4.

Разделение модели на субмодели преследует еще одну цель – если разъединить модель коптера и его систему управления (регуляторы) таким образом, чтобы весь обмен сигналами между ними шел через базу данных, то возможно разделение одного проекта на два – в первом будет считаться модель динамики объекта, возможно, с моделью двигателей ВМГ, а во втором будет считаться исключительно модель алгоритмической части системы управления, т. е. это будет прообразом «прошивки», или функционального программного обеспечения, для полетного контроллера. И в SimInTech есть все возможности как для такого разделения проекта на несколько проектов, затем запуска этого всего как пакета проектов, так и для прямой генерации кода на выбранную целевую систему. Код генерируется стандартный, на языке Си, и может быть создан с применением нужного шаблона, и его можно скомпилировать почти под любое современное оборудование, подробнее об этом смотрите в справочной системе к SimInTech.

Пульт управления и анимация

10

В среде SimInTech расчетная схема одновременно является и математическим «исходником» для расчета, и графическим пространством, в котором можно размещать те или иные объекты – блоки, графические примитивы, группы объектов и т. п.

У каждого объекта на расчетной схеме, или на изображении блока, есть свои свойства, в том числе координаты, цвет, толщина линии и многие другие. Свойства объектов можно связывать с расчетными переменными – сигналами модели. И настроить все так, что при изменении какого-либо сигнала в модели у графического объекта будет изменяться цвет или его положение на схеме, или объект будет становиться невидимым, или еще появится какое-то динамическое поведение, связанное с изменениями в модели. На базе этого принципа строится так называемая техническая анимация – мы можем взять из модели координаты коптера, его углы положения и настроить некоторые графические объекты так, чтобы они «отображали» в анимированном виде текущее расположение коптера в пространстве (точнее, на плоскости).

10.1. Пример анимации

Прежде чем переходить к сложным вещам – графическим примитивам и к громоздким скриптам, анимирующим их в процессе моделирования, – давайте рассмотрим сам принцип создания анимации. Первое: у каждой страницы расчетной схемы SimInTech есть страничка скрипта (см. рис. 10.1.1), которая на встроенном языке программирования (подробнее см. справку) позволяет писать программы как математического плана, так и предназначенные для манипуляций с графическими объектами. У нас (например) изменяется высота коптера с 40 до 50 м во всех тестовых расчетах, которые мы делали ранее, – это сигнал OC_z. Давайте возьмем этот сигнал и в соответствии с ним сделаем анимацию у субмодели «Модель коптера», как если бы эта субмодель у нас была самим коптером и должна «взлететь» на расчетной схеме на +10 м от своего начального положения. Этот пример исключительно для демонстрации графических возможностей и принципа анимации.

Зайдите в свойства этой субмодели, поменяйте у нее свойство **Имя** на SubCopter, а значение свойства Points скопируйте в буфер обмена (см. рис. 10.1.2) – это начальное положение субмодели, и ее положение кодируется четырьмя па-

рами координат (X,Y) на схеме. X – горизонталь, Y – вертикаль, направленная вниз. У вас, скорее всего, конкретное значение свойства Points будет другим – это не важно, главное скопируйте как у вас. В моем случае оно равно [(–848 , 16),(–824 , 16),(–848 , 0),(–848 , 36)].

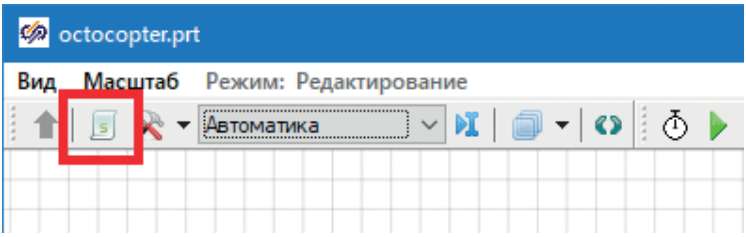


Рис. 10.1.1 Скрипт проекта

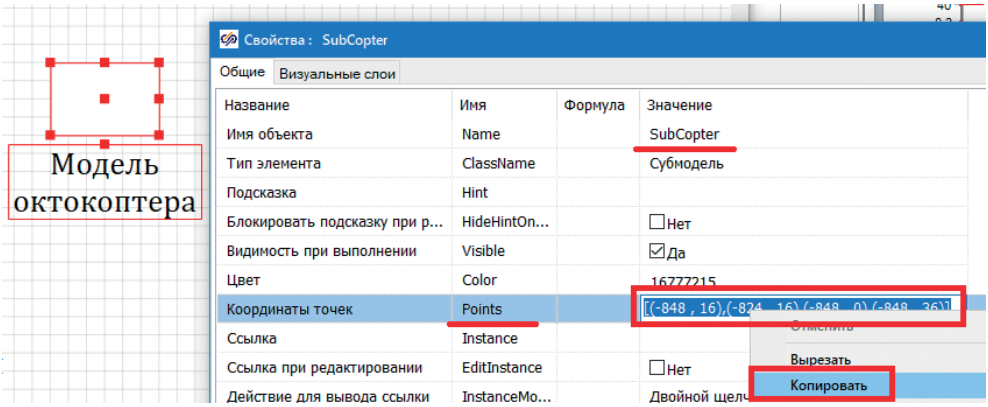


Рис. 10.1.2. Свойства субмодели «Модель октокоптера»

Имя объекта может быть, в принципе, любым, лучше не начинать его с цифры и использовать английские буквы, хотя это не строго. Итак, у нас есть объект с именем SubCopter и начальной позицией, которую мы скопировали в буфер обмена. Чтобы реализовать анимацию, надо в скрипте сделать следующую относительно простую запись, где pts0 надо приравнять тому вектору, который вы скопировали из свойств субмодели (это ее начальные координаты):

```
initialization
 pts0 = [(-848 , 16),(-824 , 16),(-848 , 0),(-848 , 36)];
end;
SubCopter.Points = pts0 + [(0,40-OC_z), (0,40-OC_z), (0,40-OC_z), (0,40-OC_z)];
```

После этого выставьте в параметрах расчета синхронизацию с реальным временем (чтобы увидеть анимацию), запустите режим на расчет и увидите, что субмодель по мере расчета медленно поднимается и должна подняться на +10 пунктов – это чуть больше одной ячейки модельной сетки на схеме, см. рис. 10.1.3. Изначально она была на той же высоте, что и субмодель «Регулятор октокоптера». После окончания расчета координаты объекта вернутся к исходным, поскольку новые координаты, которые были насчитаны в процессе

моделирования, не сохранены в свойствах блока – для этого есть специальная процедура **storeposition**, которой сейчас мы не воспользовались.

Разберем представленный выше скрипт. В секции инициализации определена константа с именем pts0 (точнее, переменная – в SimInTech практически нет отличий между константами и переменными, определенными в скриптах). И эта переменная присваивается только один раз, при инициализации расчетной схемы. Все, что записано между **initialization** … **end**, исполняется однократно. Далее, на каждом такте перерисовки расчетной схемы свойству Points у блока SubCopter (обращение к свойству идет через точку) присваиваются те же координаты точек pts0 с одинаковым смещением по второй координате, равным –(OC_z-40). –––40 нужно для того, чтобы убрать начальное значение, а еще один минус – затем, что высота коптера возрастает при его взлете вверх, а на экране координата точек Y должна уменьшаться при этом, чтобы изображение блока перемещалось вверх на экране.

Разберем еще более лаконичный вид записи – из приведенного скрипта видно, что выражение (0, 40-OC_z) нам пришлось записать 4 раза. На практике бывает и большее количество. Рационально его определить своей переменной, например vec_z = (0, 40-OC_z), затем построить массив из 4 таких векторов и далее уже приплюсовать массив к начальным координатам. Скрипт при этом может преобразоваться к следующему:

```
initialization
 pts0 = [(-848 , 16),(-824 , 16),(-848 , 0),(-848 , 36)];
end;
vec_z = (0,40-OC_z);
pts_z = [vec_z, vec_z, vec_z, vec_z]
SubCopter.Points = pts0 + pts_z;
```

Теперь, если надо будет исправить смещение – это достаточно сделать в одном месте один раз. И еще один момент – использование служебного символа # может сократить запись pts_z. Можно записать:

```
pts_z = 4#vec_z;
```

и это будет эквивалентной записью (4 раза по vec_z).

Представленный пример, конечно же, синтетичен, но он показывает саму суть анимации – через скрипт и переменные, рассчитываемые в модели, изменяем графическое изображение объекта в схемном окне. Изменять можно не только координаты, а цвет объекта, толщину линии, видимость, изображаемый текст и т. д.

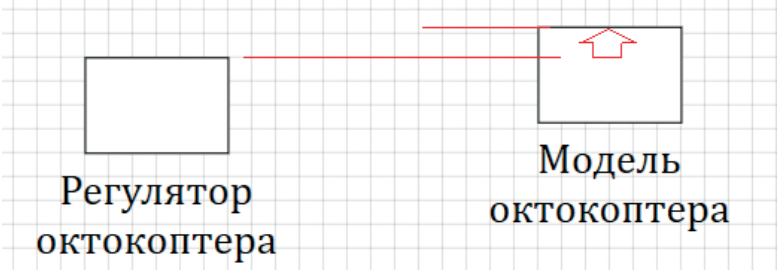


Рис. 10.1.3. Субмодель «поднялась» из-за скрипта анимации для нее

10.2. Плоская анимация в окне анимации

В примере, который мы выполнили, анимированным оказался объект в пределах расчетной схемы. Это не всегда удобно (чаще всего – неудобно). Поэтому в SimInTech предусмотрен отдельный механизм, который называется «Окно анимации». Это окно находится в менеджере данных и является одним из типовых объектов для пост-обработки результатов расчета наряду с графиками и другими объектами. Если вы вызовете пункт меню **Расчет → Менеджер данных...**, то увидите перечень созданных вами графиков, похожий на рис. 10.1.4.

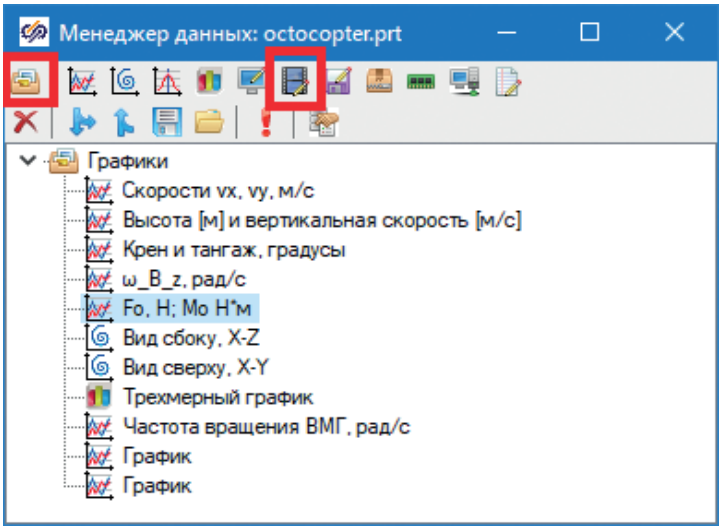


Рис. 10.1.4. Менеджер данных, создание нового окна анимации

Если в менеджере данных нажать кнопку **Окно анимации** (выделена на рис. 10.1.4), то появится новый объект в менеджере данных. Лучше предварительно создать новую категорию и назвать ее «Анимация» и уже в ней создать новое окно анимации, чтобы оно не путалось с графиками. Создание категории делается кнопкой **Создать категорию** в левом верхнем углу, удаление объекта – кнопкой под ней. В итоге у вас должна получиться картинка, похожая на рис. 10.1.5, с новой категорией и окном анимации в ней. Если двойным щелчком нажать на **Окно анимации**, то рядом с окном менеджера данных появится пустое (чистое) окно анимации. Визуально оно точно такое, как и окно расчетной схемы, только в пределах этого окна невозможно расставлять расчетные блоки и набирать расчетную схему. В этом окне доступны только графические примитивы и скрипт, осуществляющий манипуляции с объектами окна анимации и, возможно, какие-то вспомогательные вычисления. Кроме того, все глобальные переменные проекта здесь тоже доступны напрямую, поэтому скрипт можно писать примерно аналогично тому, как если бы его писали в скрипте схемного окна.

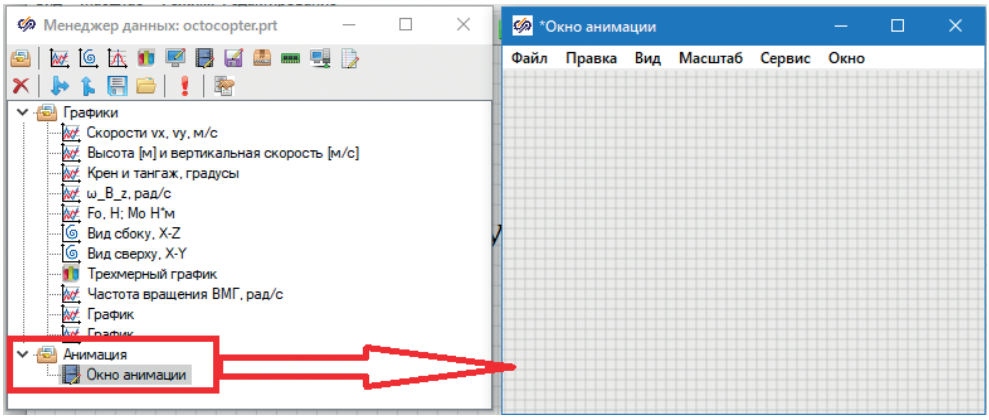


Рис. 10.1.5. Новое окно анимации

10.2.1. Постановка задачи

Определим, что будем создавать в окне анимации и с какой целью. Для отладки регуляторов нам будет интересен вид спереди на коптер (в осях X–Z) и вид сверху (оси X–Y). Было бы неплохо также иметь и вид сбоку (оси Y–Z), но так как регуляторы крена и тангажа у нас реализованы одинаково, то это может быть избыточным. Ограничимся двумя видами. При этом мы будем видеть изменение высоты коптера, его координаты X, на виде сверху также и координаты Y. Если сделать еще анимацию поворота коптера, то будем визуально наблюдать угол тангажа (на виде спереди) и курса коптера (на виде сверху).

Определим также, что 1 м длины будет равен одному делению модельной сетки (8 пунктов внутренних единиц). То есть 10 м – это будет 80 пунктов, а 100 м – 800 пунктов.

Смысл анимации очень простой – в модели будут насчитываться какие-то текущие координаты коптера в зависимости от режима полета, а также углы ориентации. На виде спереди мы будем менять координаты изображения коптера в соответствии с сигналами OC_x, OC_z. На виде сверху – в соответствии с сигналами OC_x, OC_y. При изменении угла тангажа будем вращать изображение на виде спереди, как если бы курс всегда был нулевым (что, вообще говоря, неверно), но в приближении учебной задачи путь будет так, для простоты, а при изменении угла курса будем вращать изображение на виде сверху.

10.2.2. Плоская анимация

Область «полета» коптера давайте ограничим кубом со стороной 100 м и разрисуем «координатную сетку» в этих пределах с шагом 10 м. По высоте от 0 до +100 м, по направлениям X–Y от –50 до +50 м. Для двух видов это будут квадраты, причем сделаем так, что начало координат модельной сетки (для простоты) будет находиться в середине нижней стороны квадрата для вида спереди.

В панели примитивов (самая левая кнопка в палитре блоков, рядом со стрелкой) выберите примитив «Прямая» и разместите ее произвольным образом в окне анимации, как показано на рис. 10.2.1.

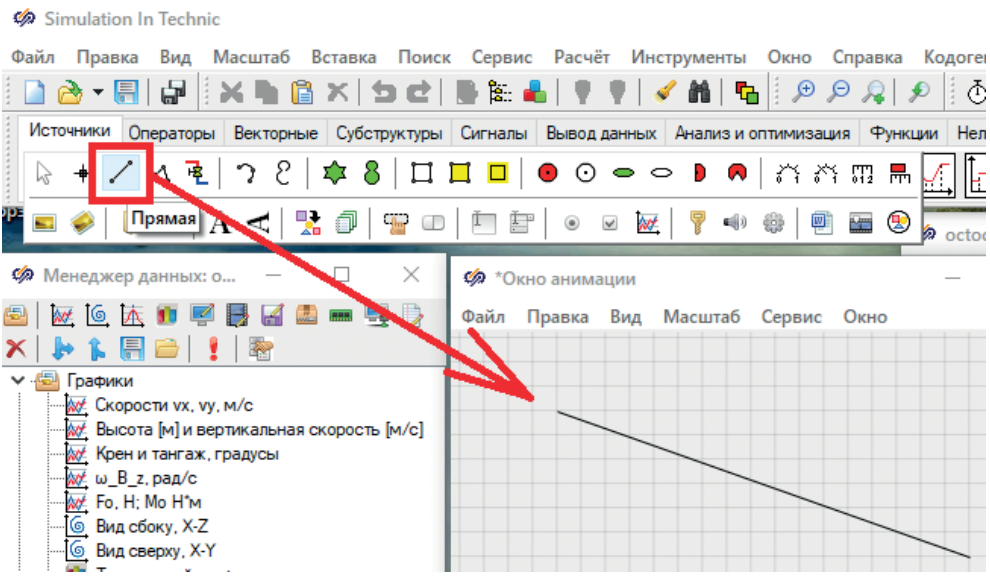


Рис. 10.2.1. Отрезок прямой из окна примитивов

Далее, перейдите в свойства размещенного объекта (свойства будут содержать только общие графические свойства, как и у расчетных блоков на расчетной схеме), найдите там свойство Points – и так как отрезок прямой характеризуется двумя точками, исправьте эти точки на значение [(–400, 0), (400 , 0)] – этот отрезок будет представлять собой уровень поверхности земли. Возможно, при вводе этого значения отрезок уйдет из вида – найдите его, отмасштабировав (колесом мыши, например) окно анимации. Далее надо подготовить еще 10 горизонтальных отрезков такой же длины, но размещенных на 80 пунктов выше друг от друга, т. е. их координаты будут равны:

```
[(-400 , -80),(400 , -80)],
[(-400 , -160),(400 , -160)],
…
```

и т. д. до [(–400 , –800),(400 , –800)].

Минус здесь нужен, так как у окна анимации вертикальная ось направлена вниз. Разместить эти отрезки можно как вручную, так и копированием первого отрезка и смещая его на нужное количество ячеек модельной сетки вверх или каким-то еще способом. В результате должна получиться картина, как на рис. 10.2.2, причем начало координат – в середине нижнего отрезка.

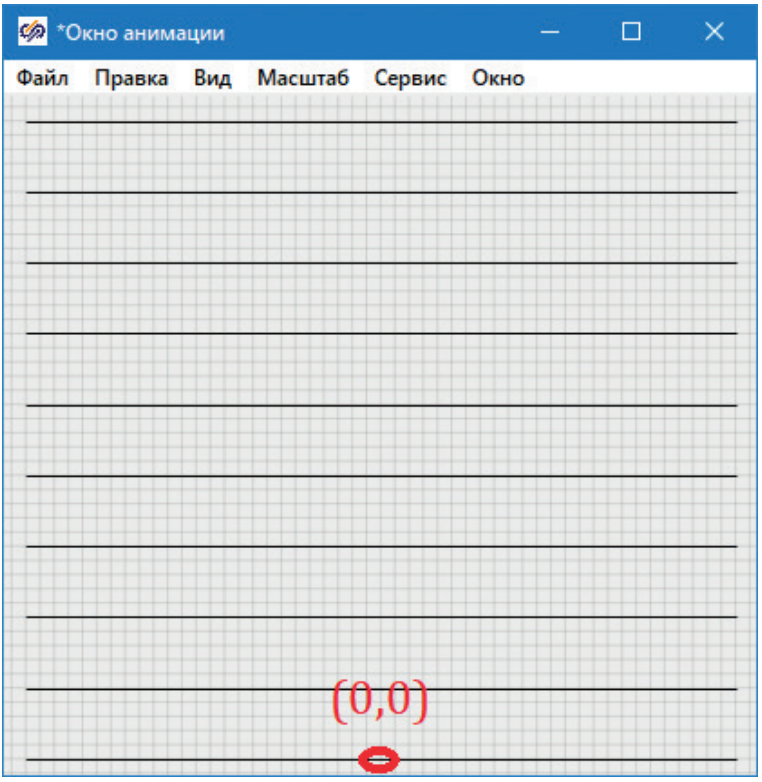


Рис. 10.2.2. Начало координатной сетки

По аналогии сформируйте вертикальные отрезки, потом подпишите их при помощи примитива «Текст» значениями 0 м, +10 м, +20 м и т. д., а затем скопируйте набранные 22 отрезка для того, чтобы быстрее сделать «вид сверху», вставьте несколько ниже первого получившегося квадрата и исправьте там надписи (или сделайте заново). Результат с точностью до художественно-оформительских навыков должен совпасть с рис. 10.2.3. Обратите внимание на такие свойства у графического примитива, как **Шрифт** (пришлось выставить размер, равный 32, чтобы числа было видно на скриншоте), **Стиль выравнивания** и **Положение точки вставки**. В дальнейшем нам еще потребуется свойство **Формат числа** и некоторые другие.

Линии, отображающие координатную сетку, можно отобразить другим цветом и, возможно, пунктиром, чтобы они не так сильно выделялись – это тоже настраивается через окно свойств. Дальше, чтобы размещенная «статическая» картинка не мешалась при рисовании следующих объектов, лучше отнести размещенные графические примитивы на свой слой (например, 16-й) и сделать его видимым, но неактивным. Для этого надо нажать **Ctrl**+**A**, чтобы выделить все объекты на окне анимации, далее через правую кнопку вызвать окно свойств и в нем перейти во вкладку **Визуальные слои**. В этой вкладке снять галочку с 1-го слоя и поставить галочку на 16-м – он будет называться «Рамки», если вы делали часть методики, посвященной оформлению. Но ак-

тивность этого слоя у окна анимации своя (окно анимации имеет свои слои, расчетная схема – свои). Надо перейти еще в пункт меню **Вид → Визуальные слои...** и там сделать неактивным слой «Рамки», см. рис. 10.2.4.

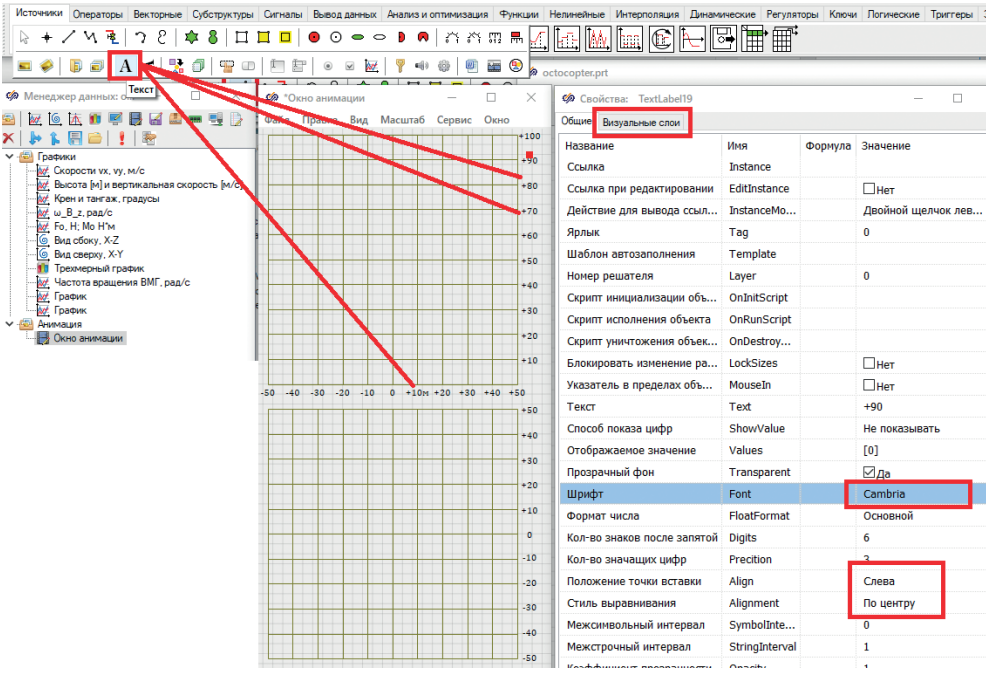
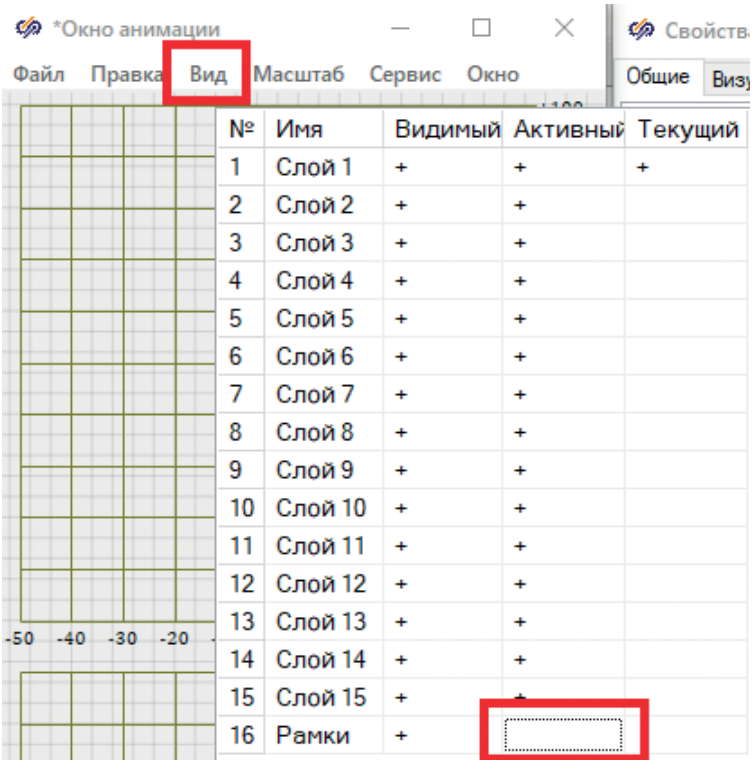


Рис. 10.2.3. Координаты для вида спереди и вида сверху готовы

После этого все размещенные объекты останутся видимыми, но неактивными для нажатий кнопкой мыши, т. е. станут как бы «подложкой» или калькой для дальнейших художеств.

Отобразим октокоптер на виде спереди достаточно простым (условным) изображением. Вспомним, что размах лучей для 1-й и 5-й ВМГ был принят равным примерно 1.4 м, т. е. это 1.5 деления модельной сетки, а диаметр винтов примем 1 м – тогда раму октокоптера и ВМГ номер 1,3,5 можно изобразить несколькими отрезками, как показано на рис. 10.2.5. Сделайте толщину этих отрезков равной 2 или 3, чтобы коптер выделялся сильнее на фоне сетки, и разместите все это в точке (0 м, 40 м), так как начальная высота в модели равна 40 м. Далее, можно анимировать коптер как по частям (отдельно задавая координаты для каждого графического примитива), так и объектом в целом – для этого надо сгруппировать размещенные примитивы. Поскольку коптер у нас будет двигаться как единое целое, лучше пойти по второму варианту. Для этого выделите все 7 размещенных отрезков (охватывающей рамкой) и далее в контекстном меню (по правой кнопке мыши) выберите пункт **Действия → Сгруппировать**. После выполнения этого действия вместо 7 примитивов на окне анимации будет доступен один блок, графическое изображение которого повторяет нарисованное ранее (графическое изображение можно исправлять далее уже у этого блока или разгруппировать его). У блока есть свое имя в свойствах – задайте его равным

Copter1, чтобы дальше из крипта обращаться к нему по этому имени. Также скопируйте его начальные координаты в буфер обмена (см. рис. 10.2.7) и, перейдя в скрипт, запомните их в виде переменной с именем (например) copter1pts0.



№	Имя	Видимый	Активный	Текущий
1	Слой 1	+	+	+
2	Слой 2	+	+	
3	Слой 3	+	+	
4	Слой 4	+	+	
5	Слой 5	+	+	
6	Слой 6	+	+	
7	Слой 7	+	+	
8	Слой 8	+	+	
9	Слой 9	+	+	
10	Слой 10	+	+	
11	Слой 11	+	+	
12	Слой 12	+	+	
13	Слой 13	+	+	
14	Слой 14	+	+	
15	Слой 15	+	+	
16	Рамки	+		

Рис. 10.2.4. Слой рамки – делаем неактивным

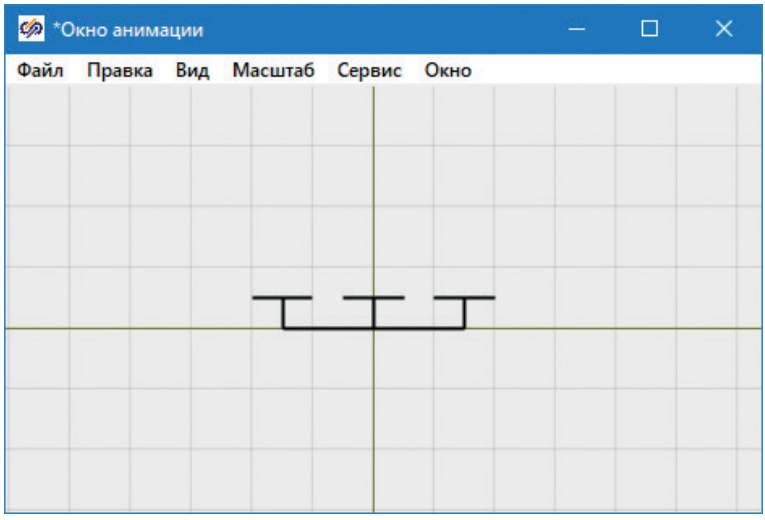


Рис. 10.2.5. Октокоптер, вид спереди

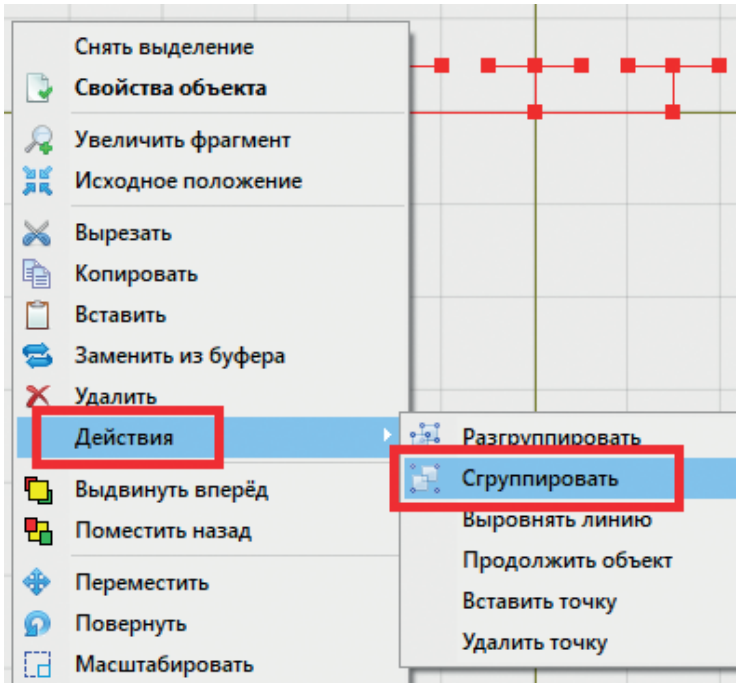


Рис. 10.2.6. Группировка объектов в один блок

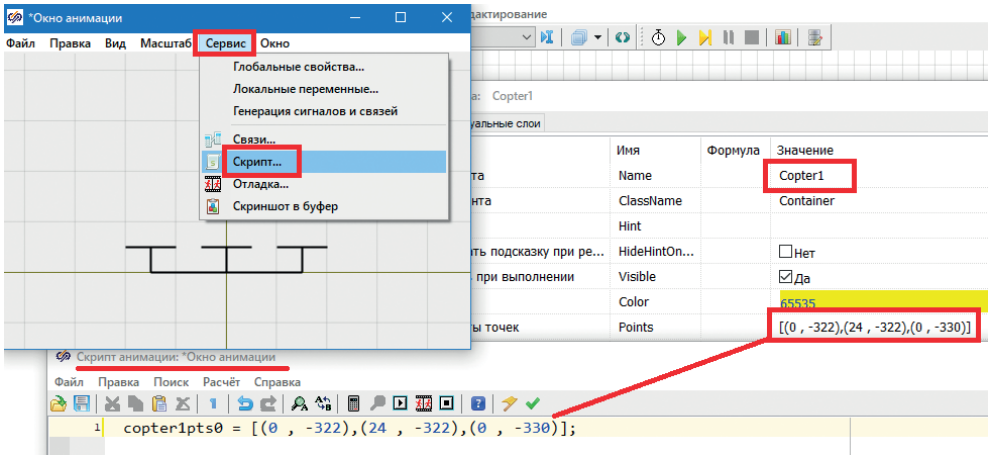


Рис. 10.2.7. Начальные координаты группы Copter1

После этих предварительных манипуляций наконец пришло время реализовать сам (относительно простой) скрипт анимации для вида спереди. Сначала нам надо сформировать вектор, на который мы хотим перемещать дан-

ный блок (группу линий Copter1). Вектор назовем vec1, а его значение будет равно (OC_x, –(OC_z-40)) (минус из-за направления осей). Затем этот вектор в количестве 3 шт. надо добавить к свойству Points группы Copter1. Сделаем это, протестируем, а затем перейдем к углу тангажа.

Правильный скрипт должен быть таким (с точностью до начальных координат, но они у вас должны получиться такими же, если разместили все объекты верно):

```
const copter1pts0 = [(0 , -322),(24 , -322),(0 , -330)];
vec1 = (OC_x, -(OC_z-40)); //м
pts1 = 3#vec1;
pts1 = 8*pts1; //перевод в пункты модельной сетки
Copter1.Points = copter1pts0 + pts1;
```

Обратите внимание, что вместо слов **initialization**…**end**; мы использовали здесь ключевое слово **const**. Это альтернатива, если надо только задать начальные значения той или иной переменной (константе).

Если окно анимации с этим скриптом (смысл скрипта, думаем, понятен, из предыдущих объяснений с синтетическим примером про блок на расчетной схеме) запустить на исполнение вместе с расчетом модели, то вы можете увидеть, что объект Copter1 будет совершать некоторые эволюции и достигнет +50 м через время – как это раньше мы и видели на графиках. Также, возможно, он будет совершать некоторые колебания вправо-влево из-за ненулевых начальных условий в модели. В итоге, повторив линию фазового портрета X–Z, коптер должен дойти до 50-м высоты и там «зависнуть».

Давайте немного переделаем скрипт анимации, потому что сейчас он не совсем корректен, а сделан для частного случая начальных условий, равных +40 м по высоте. Остановите расчет, переместите группу Copter1 в начальную точку (0,0), отметьте, что координаты группы поменялись – стали более простыми, и переделайте скрипт на следующий. Убедитесь, что все работает корректно и с таким скриптом коптер должен так же взлетать, как и ранее, с отметки +40 м на +50 м, возможно, с раскачкой по оси X из-за плохого качества регулирования (на самом деле, оно специально оставлено таким, чтобы было проще отлаживать анимационные вещи, в том числе и наклоны коптера в следующем подразделе):

```
const copter1pts0 = [(0 , 0),(24 , 0),(0 , -8)];
vec1 = (OC_x, -OC_z); //м
pts1 = 3#vec1;
pts1 = 8*pts1; //перевод в пункты модельной сетки
Copter1.Points = copter1pts0 + pts1;
```

Если вам не нравится редкое (заметное глазу) обновление анимационной картинки – перерисовка окна анимации, можете перейти в пункт меню **Вид → Опции…** и выставить там период перерисовки, равный 50 или 25 мс (см. рис. 10.2.8).

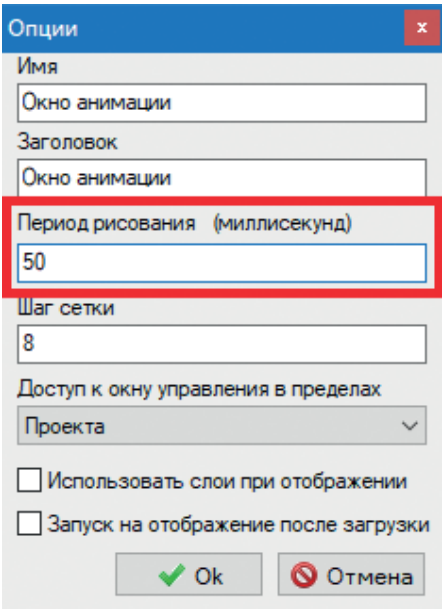


Рис. 10.2.8. Период перерисовки окна анимации

10.2.3. Поворот объекта

Перейдем к следующему пункту – реализовать на сделанном виде спереди поворот коптера в соответствии с его углом тангажа (напомним, это угол поворота относительно оси Y, т. е. как раз на виде X–Z мы должны видеть этот угол, если курс = 0°). Поскольку в рамках учебных допущений мы принимаем, что курс у нас не отклоняется сильно от 0°, то плоская анимация будет справедлива для этого частного случая – объект надо наклонять именно на φ° в плоскости X–Z, и это будет относительно «честным» отображением проекции коптера на эту плоскость.

Примечание: если бы мы нарисовали коптер как набор линий, то нам пришлось бы вычислять для каждой линии кроме плоского смещения (реализованного в предыдущем подразделе) еще и вращательное смещение по углу тангажа и через тригонометрические функции, фактически делать плоскую матрицу поворота в плоскости X–Z. Если вы посмотрите на свойства примитива типа «Линия», то ничего, кроме Points, там не найдется. Но для блока или для группы линий (когда мы их сгруппировали) есть еще свойство с именем Angle – и задание его ненулевым как раз и выполнит эту работу для нас. Таким образом, для того чтобы изображение октокоптера вращалось, достаточно присвоить свойству Copter1.Angle нужное значение угла поворота, в радианах. А в модели угол у нас как раз (случайно) и считается в радианах.

Поэтому все, что нужно сделать, – добавить в скрипт всего лишь одну строку (главное, не ошибиться со знаком!). Попробуйте сначала сделать так:

```
Copter1.Angle = OC_tet;
```

На анимации увидите нефизичность полета – коптер наклоняется вправо, а летит влево…

Значит, верная запись такая: Copter1.Angle = –OC_tet;

Если проследить за тем, куда направлена ось Y в системе координат B (на зрителя, на нас), то станет понятно (даже очевидно), зачем тут нужен знак минус…

Итак, снабдив группу объектов Copter1 не только плоским смещением, но и углом поворота, мы получили реализованную плоскую анимацию на виде спереди (см. рис. 10.2.9). Отметим, правда, что она будет справедлива только для нулевого курса. Для общего случая здесь придется «навернуть» побольше тригонометрии, но принцип анимирования будет оставаться тем же самым. Добавим еще некоторую интерактивную информацию, которая будет перемещаться вместе с изображением коптера, рядом с ним.

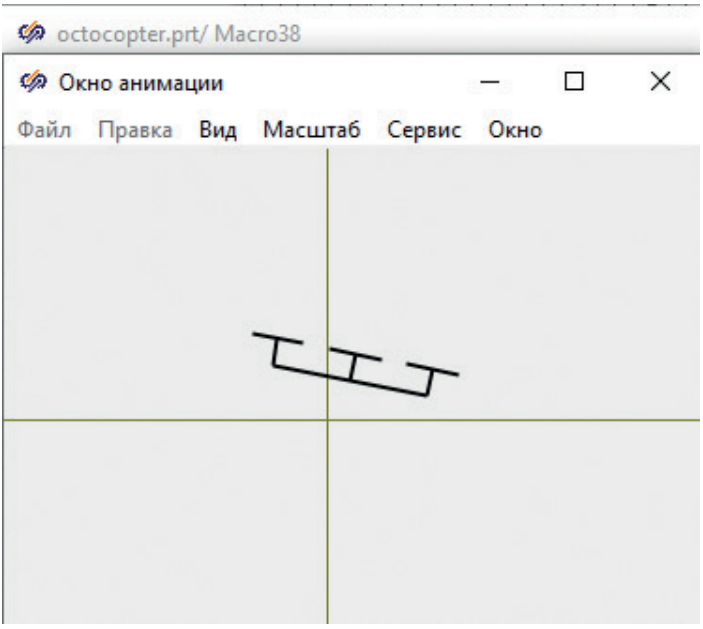


Рис. 10.2.9. Анимация с поворотом

10.2.4. Текст с анимацией

Например, нам было бы интересно рядом с «летающим» изображением октокоптера видеть его текущую высоту, вертикальную скорость и углы крена и тангажа – для этого надо разместить примитив типа «Текст» рядом с коптером и через скрипт задать для него текст (изменяющийся в процессе моделирования) и смещение точки вставки этого текста, аналогичное смещению самого коптера. Текст хотим иметь, например, в таком виде:

z: 45.46 м
vz: –0.0290 м/с
φ: 0.00°
θ: 0.00°

Разместите один графический примитив типа «Текст» (в панели примитивов это кнопка с большой буквой **A**) справа-снизу от коптера, см. рис. 10.2.10,

и задайте там начальный текст, как приведено выше. Ничего страшного, что один текст сейчас разместился поверх другого, – во-первых, координатные оси и метки у нас сделаны неактивными, а во-вторых, при анимации коптер будет вверху вместе с текстом.

Зайдите в свойства текста (правой кнопкой мыши), задайте имя этому объекту CopterLabel, запомните начальные координаты точки вставки (в нашем случае они равны [(16, 8)]) и проверьте, что начальный (статичный) текст задан примерно из 4 строк (на самом деле начальный текст не особо нужен – только для предпросмотра, как он будет выглядеть в дальнейшем рядом с коптером). Координаты [(16, 8)] получились, так как мы вставили текст на две ячейки модельной сетки направо (2 × 8) и на одну вниз (1 × 8) от начала координат, где расположен коптер.

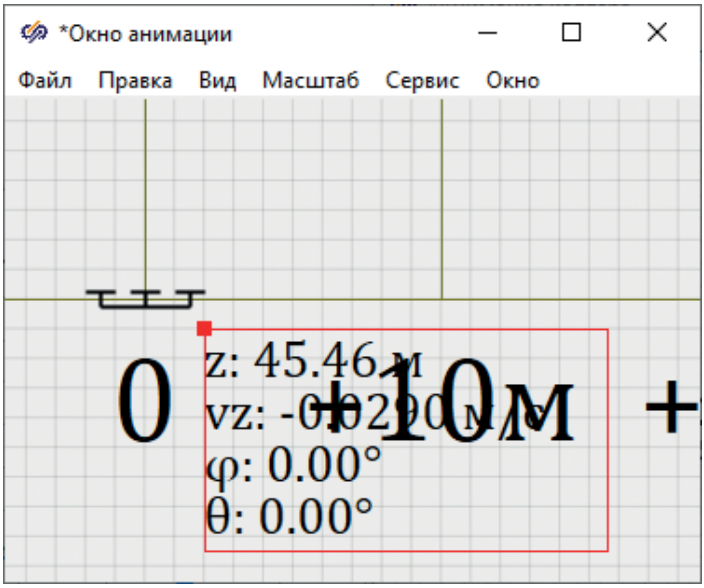


Рис. 10.2.10. Текстовая подпись для анимации

Затем зайдите в скрипт окна анимации и наберите там следующие строки:

```
CopterLabel.points = [(16 , 8)] + 8 * [(OC_x , -OC_z)];
CopterLabel.Text =
 «z: « + floattostrf((OC_z),3,8,2) + “ м» + Chr(13) +
 “vz: “+ floattostrf((OC_vz),3,8,4) + “ м/с» + Chr(13) +
 “φ: « + floattostrf((OC_phi*180/pi),3,8,2) + “°” + Chr(13) +
 “θ: « + floattostrf((OC_tet*180/pi),3,8,2) + “°”;
```

В этом скрипте должно быть все понятно – функцией **Chr**(13) осуществляется перенос строки и возврат каретки (в Windows работает один этот символ…), функцией **floattostrf**() реализуется форматирование и вывод в виде текста действительных чисел. Для скорости мы отвели четыре значащих цифры после запятой, у остальных по две, а для углов выполнен еще перевод из радиан в градусы.

Далее, если запустить модель на расчет, вы должны увидеть все так же поднимающийся и качающийся коптер, но уже с «плывущим» рядом с ним отладочным сообщением с краткой текущей «телеметрией», измеренной в модели (один из моментов представлен на рис. 10.2.11). С такой картинкой уже будет гораздо удобнее и понятнее отлаживать регуляторы.

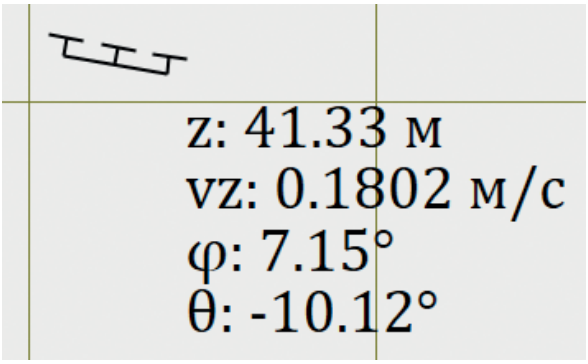


Рис. 10.2.11. Коптер с краткой информацией о полете

При желании можете самостоятельно сделать и вид коптера в осях Y–Z, а мы перейдем к виду сверху…

Если вы обратите внимание, в перечне свойств у объекта типа TextLabel (наш объект CopterLabel относится к этому типу) есть свойства ShowValue и Values. Они используются в другом режиме работы этого блока – без скрипта. Можно вписать статичный однострочный или многострочный текст, напротив каждой строки которого блок будет сам отображать значение, указанное в свойстве Values (в колонке **Формула**). Отображать он его будет или слева, или справа – это ограничивает данный способ использования блока. Через скрипт, как сделали мы, можно произвольным образом форматировать выводимый текст. С другой стороны, этим способом можно обходиться без написания скрипта. Например, если поставить еще один блок типа TextLabel на окно анимации и задать ему свойства, как показано на рис. 10.2.12, то блок будет сам считывать из базы данных значение OC_z и отображать на окне анимации строку «z: 41.3» (без кавычек) как сумму статичного текста, указанного в свойстве Text, и значения переменной, указанной в свойстве Values. Аналогично можно делать и многострочное отображение.

Блокировать изменение размеров	LockSizes		☐Нет
Указатель в пределах объекта	MouseIn		☑Да
Текст	Text		**z:**
Способ показа цифр	ShowValue		**Показывать справ**
Отображаемое значение	Values	**OC_z**	[41.328143]

Рис. 10.2.12. Другой способ отображения текущих значений

10.2.5. Вид сверху

Вид сверху попробуйте нарисовать самостоятельно, разместив 4 объекта типа Line и 8 окружностей на концах этих отрезков, как показано на рис. 10.2.13. Постарайтесь соблюсти размеры – чтобы длинные лучи имели длину в 3 ячейки модельной сетки, как это было для эскиза коптера на виде X–Z.

Потом при объединении всех нарисованных примитивов в группу SimInTech может не очень верно определить границы всех примитивов, а охватывающую рамку сделать шире, чем нужно, а также центр изображения может не совпасть с центром группы (как это показано на рис. 10.2.13 слева – есть небольшое смещение и охватывающая рамка не точна). В этом случае следует зайти в редактирование изображения группы (двойным щелчком мыши по ней) и там совместить центры. Делается это следующим образом – в графическом редакторе можно нажать **Ctrl**+**A**, чтобы выделить все объекты, и затем подвинуть их так, чтобы перекрестие совпало с началом координат (внутри изображения группы), см. рис. 10.2.14.

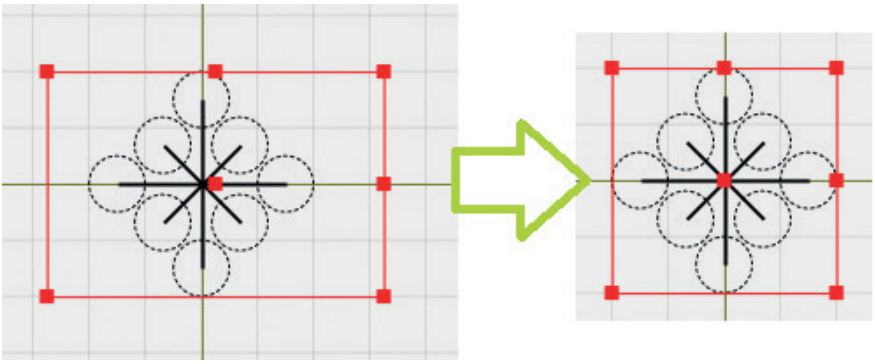


Рис. 10.2.13. Вид сверху, до и после корректировки

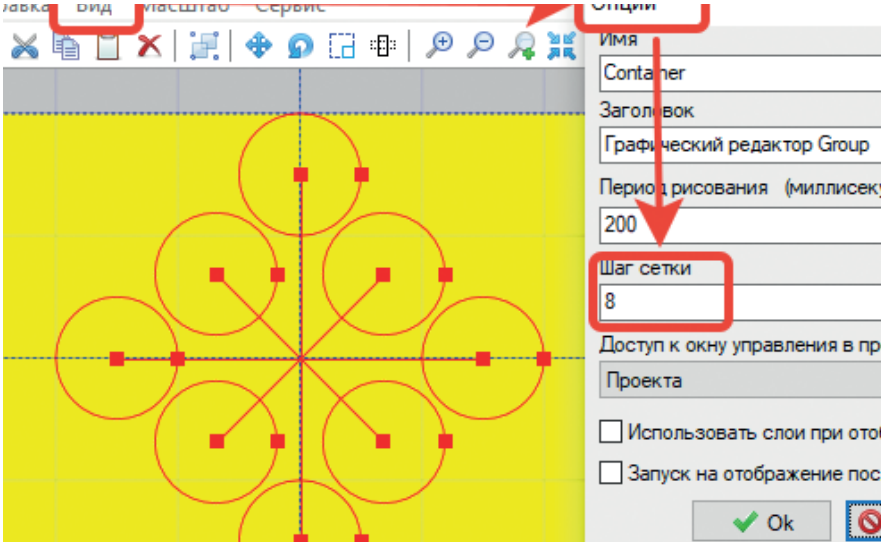


Рис. 10.2.14. Редактирование изображения группы (или блока)

Если шаг сетки (равный по умолчанию 8 пунктам) не позволяет совместить центры, то надо зайти в пункт меню **Вид→ Опции** и там поставить шаг, равный двум или одному пункту, и затем повторить совмещение.

Далее, так как изображение квадратно и его размер составляет 4×4 клетки (с шагом 8), т. е. размер равен 32×32, лучше такой размер и выставить как размер изображения. Для этого надо зайти в пункт меню **Масштаб → Задать размеры контейнера** и там вписать число 32 два раза (рис. 10.2.15). После этого желтая область должна четко охватить нарисованное изображение коптера, и окно редактирования графического изображения группы можно закрыть с сохранением этого изображения.

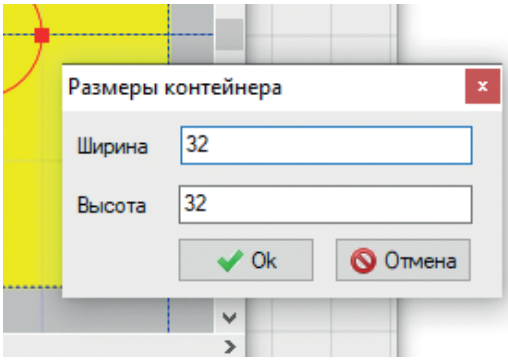


Рис. 10.2.15. Задание размера изображения

Далее, группе на окне анимации надо также скорректировать размер на схеме, чтобы он составлял 32×32 пикселя (как показано на рис. 10.2.13, справа). В принципе, эти манипуляции не очень обязательны, так как внутренний размер изображения необязательно должен совпадать с размером блока (или группы) на схеме (в окне анимации). Но для простоты изложения и написания дальнейшего скрипта будет лучше, чтобы они совпадали.

Итак, мы получили блок с изображением коптера сверху размером 32×32 пункта, который находится в начале координатной сетки, нарисованной ранее как подложка. Но это не начало координат модельной сетки. В нашем случае координаты этого блока (свойство Points) имеют следующие значения: [(0 , 480),(16 , 480),(0 , 464)]. У вас может отличаться, но числа 0 и 16 должны совпадать, так как вид сверху должен быть нарисован строго под видом спереди, т. е. координата X должна быть нулевой. 16 – это половина горизонтального размера блока.

Задайте имя этому блоку (свойство Name), например как Copter2 – по аналогии с изображением коптера на виде спереди.

Теперь нам нужно в скрипте сформировать перемещение и поворот этого изображения в соответствии с переменными OC_x, OC_y и OC_psi. Попробуйте сделать это сначала самостоятельно, по аналогии с видом спереди здесь все будет очень похоже, а ниже приведен один из вариантов такого скрипта:

```
const copter2pts0 = [(0 , 480),(16 , 480),(0 , 464)];

vec2 = (OC_x, OC_y); //м
pts2 = 3#vec2;
pts2 = 8*pts2; //перевод в пункты модельной сетки
Copter2.Points = copter2pts0 + pts2;
Copter2.Angle = OC_psi;
```

Как вы можете видеть, все довольно сходно со скриптом, написанным ранее, и должно нормально работать. Проверьте направление оси Y и нарисованные координаты на виде сверху, чтобы они совпадали по направлению. Так как угол курса мы зафиксировали как нулевой, то коптер пока что поворачиваться вокруг вертикальной оси не будет. Но если в модели исправить на честное вычисление угла курса, то изображение на виде сверху будет закручиваться.

Можно здесь также доделать и отображение базовой «телеметрии», соответствующей виду сверху. Это можете проделать самостоятельно.

Такими относительно несложными действиями мы получили наглядную модель коптера, которая визуально отражает многие из основных параметров полета коптера, а именно: высоту и вертикальную скорость, позицию X–Y на местности, горизонтальные скорости (их видно хотя бы визуально), угол тангажа и курса октокоптера. Это позволит более удобным способом провести отладку регуляторов.

10.3. Набор пульта управления

Пульт управления, в отличие от анимации, работает еще и наоборот – действия оператора пульт должен в том или ином виде передавать в математическую модель. Как правило, на пультах управления кроме отображающих приборов (частично мы это сделали в анимации – отобразили высоту, скорость, углы наклона коптера) размещены и органы управления. Большинство из них дискретны – это кнопки (включить, выключить), перекидные ключи – двухпозиционные, многопозиционные, рычаги и т. п. Но бывают также и аналоговые, (непрерывные), к которым более всего подходит вещественный тип данных. Это могут быть «крутильные» рукоятки, джойстики, мышь/трэк-бол, рули и подобные устройства.

При моделировании или макетировании пультов управления при помощи графических примитивов SimInTech можно создать практически любой виртуальный пульт управления, в том числе и фотореалистичный (при наличии соответствующих фотографий можно сделать библиотеку блоков для типовых элементов того или иного реального пульта и воссоздать его практически точно в том же виде, в котором он представлен и в аппаратной форме). В настоящей методике описаны только некоторые базовые элементы, позволяющие вывести на пульт ту или иную «телеметрию» (примерно аналогично тому, как мы это сделали выше для анимации), а также элемент типа «сенсор» и «кнопка», которые позволяют с пульта передавать в модель (в контроллер) новые заданные значения для полета коптера.

Идея базового пульта управления коптером в следующем – так как стабилизация (ориентация) коптера отдана на откуп регулятору и у нас как бы есть датчики текущего положения коптера не только по высоте, но и по горизонтальным направлениям в инерциальной системе (по осям X и Y), то мы можем организовать с пульта задание конечной точки в пространстве (x_заданное, y_заданное, z_заданное), в которую коптеру предписано лететь. Изменяя задание с пульта, мы будем направлять коптер по тому или иному маршруту, последовательно меняя заданные координаты. По каждому направлению сделаем 4 кнопки, автоматически сбрасывающиеся. Каждая из кнопок при однократном нажатии будет изменять для данного направления заданную координату на +5 м, +1 м, –1 м и –5 м соответственно. Например, если коптер находится на высоте +50 м, то для перемещения его на высоту +70 м надо будет 4 раза последовательно нажать на кнопку **z+5**.

Выпишем все 12 кнопок, которые будут на пульте:

x+5м, x+1м, x-1м, x-5м,
y+5м, y+1м, y-1м, y-5м,
z+5м, z+1м, z-1м, z-5м.

10.3.1. Категория «Кнопки» базы сигналов

Для моделирования кнопок и других активных элементов пульта, как правило, в базу сигналов проекта добавляются сигналы, отображающие состояние этих элементов. Для дискретной кнопки – так как она может быть только в состоянии «нажата» или «не нажата» – достаточно будет одной переменной типа «логическая» или «целочисленная» по каждой кнопке. В реальной жизни бывает еще состояние «не определено», т. е. промежуточное состояние кнопки, или «нет питания» и др. – это сейчас не рассматриваем. Ранее мы создавали в базе сигналов категорию «Кнопки».

Зайдите в редактирование шаблона для этой категории и добавьте там один сигнал с именем **Down** целого типа данных (см. рис. 10.3.1), с начальным значением 0 и описанием «нажата» – это будет сигнал, отображающий состояние кнопки в модели, и при изменении этого сигнала с 0 на 1 в модели будет изменяться заданное значение координаты полета.

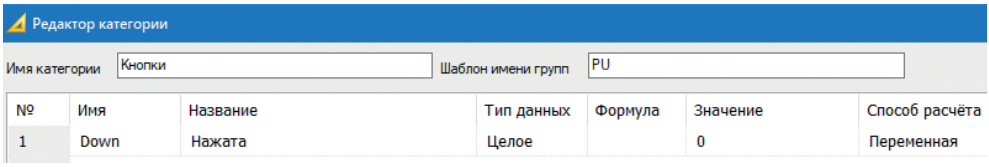


Рис. 10.3.1. Шаблон категории «Кнопки»

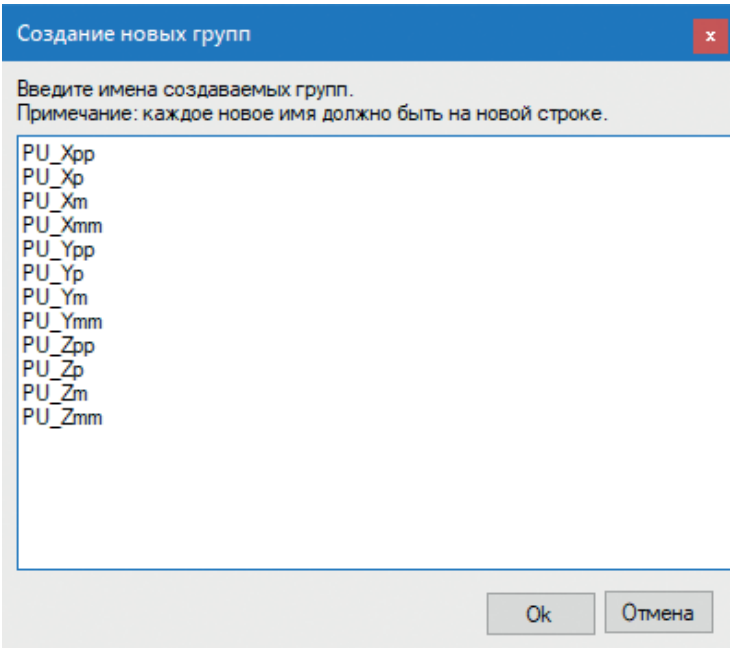


Рис. 10.3.2. Новые группы сигналов в категории «Кнопки»

После добавления шаблонного сигнала, добавьте в категорию «Кнопки» 12 групп сигналов с именами PU_Xp, PU_Xm, PU_Xpp, PU_Xmm и аналогично по другим направлениям, см. рис. 10.3.2. Таким образом, мы добавили 12 новых сигналов в модель и кроме создания пульта управления в модели нужно будет добавить «обработчик» этих сигналов – по нажатию каждой кнопки должно что-то происходить в модели.

10.3.2. Алгоритм пульта управления

Обработка нажатия кнопки может быть реализована множеством способов. Один из них представлен на рис. 10.3.3 – и даже в таком простом варианте есть ряд хитростей и защит от случайностей работы оператора или пульта. Например, кнопка (у реального пульта) может «залипнуть», т. е. зафиксироваться в нажатом положении на неограниченный срок времени, или может произойти обрыв связи, и нажатое положение останется в одном на неограниченный срок… Во-вторых, кроме пульта управления, заданное значение для регулятора в общем случае может задавать и еще какой-то алгоритм или защита (например, заканчивается заряд аккумулятора, и высоту полета надо снижать до безопасных 5–3 м независимо от того, что делает оператор и т. п.). В общем, алгоритм, который воспринимает нажатия кнопки и делает приращение или декремент у заданной координаты, должен работать не всегда, а только в редкие моменты нажатия кнопок, иметь более низкий приоритет по сравнению с защитами и блокировками и т. д. и т. п.

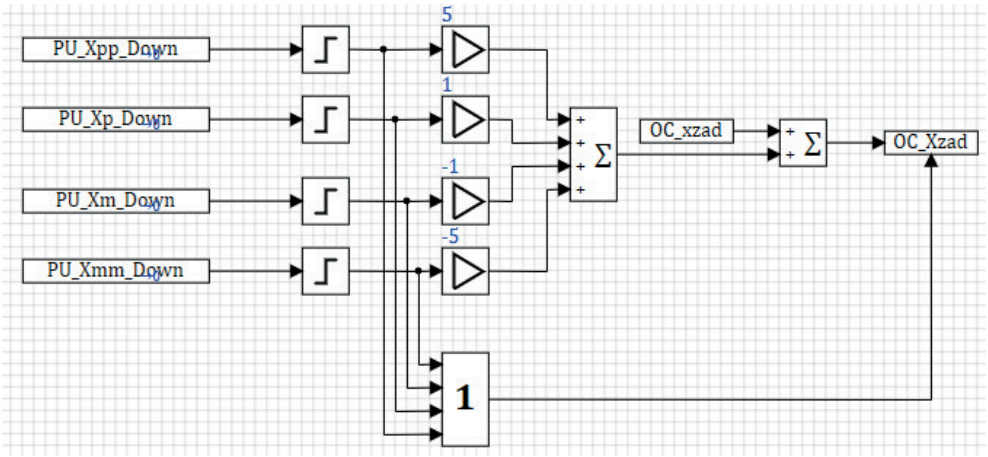


Рис. 10.3.3. Обработка нажатия кнопок на пульте управления по каналу X

Предлагаемый вариант реализует следующее – если не нажата ни одна кнопка (это большая часть времени работы модели), то на выходе в сигнал OC_Xzad будет передаваться этот же сигнал (он будет суммироваться с нулем), но запись в базу данных происходить не будет, так как для блока «Запись в список сигналов» включена опция условной записи – только когда на нижний порт будет поступать 1, то блок будет записывать сигнал в базу данных и переопределять его (см. рис. 10.3.3 и 10.3.5 – там включена эта опция, порт условия, как правило, располагается внизу блока). На остальных линиях связи в данном алгоритме будут нулевые значения. В случае нажатия какой-либо кнопки будет осуществляться вычисление возникновения «фронта» этого сигнала (только при нажатии кнопки), и в момент изменения 0 -> 1 будет формироваться единичный импульс. После этого импульс будет умножен на соответствующий коэффициент (в метрах) и просуммирован с текущим значением сигнала OC_Xzad. Дополнительно по возникновению любого импульса будет сформировано условие (логическая единица) для передачи его на порт условной записи.

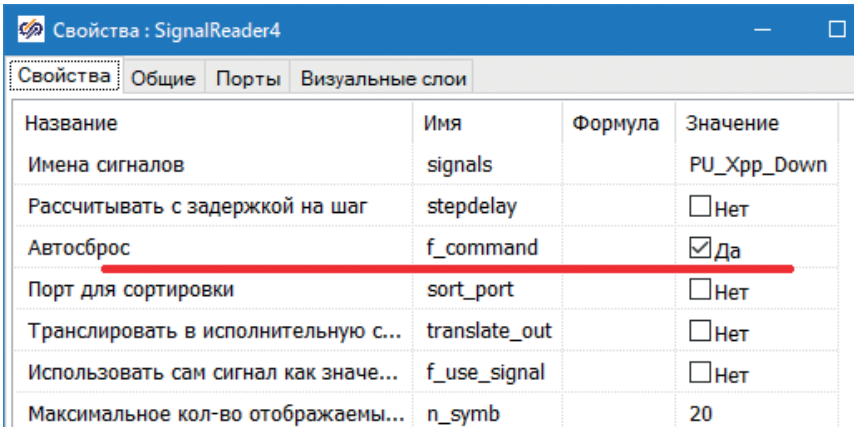


Рис. 10.3.4. Автосброс сигнала у блока «Чтение сигнала из списка»

Для предотвращения двойного и многократных нажатий кнопок (залипания кнопки) здесь применен блок импульса по фронту. Также у блоков типа «Чтение сигнала» в свойствах блока есть опция с именем «Автосброс» (см. рис. 10.3.4) – при ее включении блок сам сбрасывает прочитанный сигнал снова в 0 (изображение блока при этом дополняется стрелкой и нулем – на рис. 10.3.3 показан этот нюанс). Но при залипании кнопки пульта в этом случае возможно как раз «многократное» нажатие кнопки – блок будет сбрасывать в ноль сигнал, а на следующем такте работы системы пульт будет снова записывать 1, и цикл нажатия повторится. Вариант с импульсом по фронту – надежнее. Автосброс применяют, как правило, в программных дискретных алгоритмах, а не при разработке человеко-машинного интерфейса. Также можно применять и другие комбинации блоков, импульсов и алгоритмов, засекающих факт нажатия кнопки. Для наших целей достаточно представленного на рис. 10.3.3 варианта.

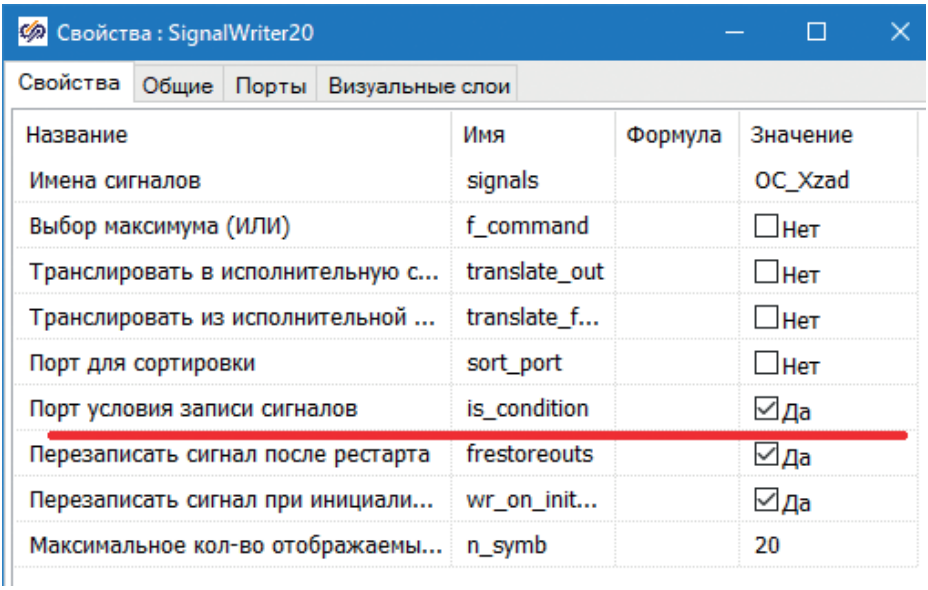


Рис. 10.3.5. Условная запись для блока «Запись в список сигналов»

Для проверки корректности набранного алгоритма нужен пульт, но пока его еще нет – вы можете в процессе моделирования зайти в базу сигналов, перейти в категорию «Кнопки» и там попробовать задать один какой-либо сигнал, отражающий нажатие кнопки. При этом сигнал сразу же сбросится в 0, если вы включили автосброс, а значение сигнала OC_Xzad будет изменено на 10 или 5 м. Можно попробовать сделать это многократно чтобы убедиться в верности набранного алгоритма обработки нажатий кнопок.

Реализуйте по аналогии еще два таких же алгоритма для направления Y и Z.

10.3.3. Проект пульта управления как «клиента» к модели динамики

В данном подразделе будет рассмотрена и применена одна из важных возможностей среды SimInTech – создание распределенной (по сети и вычислительным узлам) комплексной математической модели. В частности, мы сделаем пульт управления как отдельный проект – без математической модели, в котором просто будут представлены некоторые отображающие элементы и некоторые активные (сенсорные) элементы, и настроим этот проект таким образом, чтобы он мог асинхронно подключаться и отключаться от работающей математической модели и ее базы данных сигналов.

Комплексный проект архитектурно будет дополнен таким образом: основной файл проекта, содержащий модель коптера и его систему управления и регулирования, уже набранный выше, вместе с базой данных сигналов будет настроен как «сервер», допускающий подключение к себе и отключение других проектов в процессе моделирования. В основном проекте будет установлена синхронизация с реальным временем и задано конечное время расчета, равное бесконечности (например, 1e9 с). В базе данных сигналов будет установлено, что проект является сервером и будет «принимать» соединения от клиентов по такому-то порту (сетевой адрес определяется настройками сетевой карты в операционной системе). При этой настройке проект можно запускать на расчет на неограниченное время – и так как он устойчиво считает, можно про него «забыть».

Второй проект – с пультом управления – будет настроен как «клиент» к основной модели, будет использовать (если сравнивать их по именам) те же сигналы, которые прописаны в базе данных математической модели, и не будет ничего вычислять. Другими словами, он не будет являться необходимой частью для корректного расчета модели и будет допускать подключение и отключение от модели в любой момент. При этом единственная возможность влиять на расчет – это в подключенном состоянии изменить с 0 на 1 значение одной из 12 переменных, остальная часть модели будет как бы «защищена» от влияния извне – т. е. пульт не должен никак манипулировать другими переменными модели.

Чтобы пульт использовал те же переменные, как и модель, проще всего его настроить на тот же файл базы данных, который был сделан и для модели. Второй вариант – взять файл базы от модели и оставить там только те переменные, которые нужны для отображения и задания с пульта.

Ранее мы создали папку Control – создайте новый файл проекта и сохраните его там с именем **pult.prt**, см. рис. 10.3.6. Поскольку мы хотим использовать ту же базу данных, которая уже у нас имеется в проекте, то для того, чтобы не настраивать заново ее подключение, более быстрый способ создания нового файла – это взять существующий файл проекта, предварительно сохранить на диск еще раз, затем удалить все блоки в нем и сохранить его под именем **Control\pult.prt** – тогда все настройки базы данных сохранятся как надо. Если, конечно, вы придерживались той же структуры каталогов, какая была ранее предложена в настоящей методике.

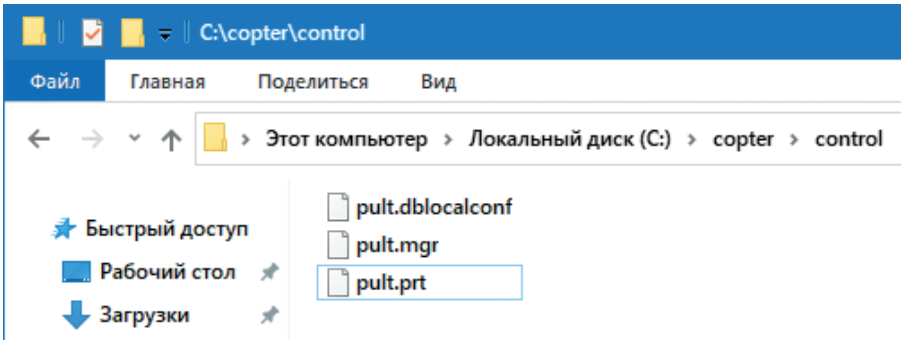


Рис. 10.3.6. Файл проекта пульта управления

В итоге, если перейти в параметры расчета, там должна быть настроена база данных, как показано на рис. 10.3.7.

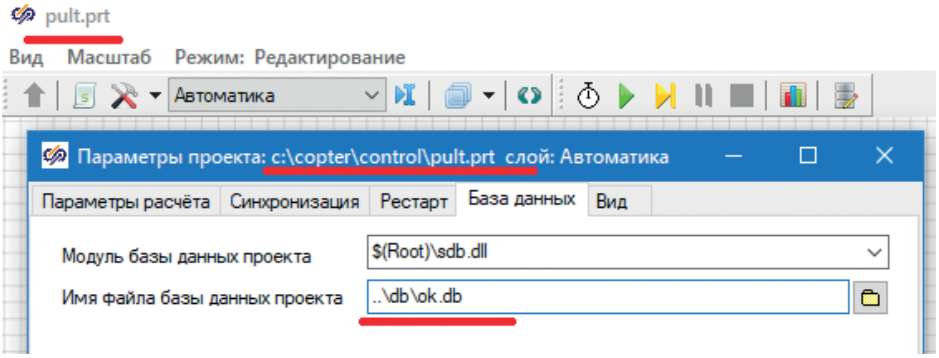


Рис. 10.3.7. Настройки базы данных для проекта пульта управления

После настройки базы данных для пульта управления сохраните проект и откройте его заново, чтобы база данных корректно загрузилась. После этого перейдите в настройки базы данных (**Инструменты** → **База данных** → **Настройки**) и задайте там настройки, как показано на рис. 10.3.8, – это настройки для асинхронного клиента к базе данных основного проекта – к файлу проекта с математической моделью.

Здесь есть ряд тонкостей. Так как мы будем запускать и модель, и пульт на одной и той же машине, то IP-адрес удаленного сервера обмена надо задать как 127.0.0.1 – это стандартный адрес локального (текущего) хоста. Если бы мы запускали модель на одном компьютере, а пульт на другом, здесь следовало бы поставить реальный IP того вычислительного узла, на котором будет запускаться модель коптера.

Порт удаленного сервера обмена можно оставить как 19000, поскольку в серверной настройке (чуть далее) по умолчанию будет настроен этот же порт.

Галочку **Включить удаленный обмен** надо включить – тогда при инициализации текущего проекта модуль базы данных будет пытаться присоединиться как клиент по адресу и порту, указанному выше.

Галочку **Синхронизировать модельное время** надо выключить, так как предполагается, что пульт управления будет подключаться и отключаться к/от модели в произвольный момент времени и не будет содержать вычислений – т. е. синхронизировать время не нужно. Эта опция используется только для моделей, распределенных по вычислительным узлам в тех случаях, где нужна синхронизация моделей по модельному времени, чтобы шел синхронный расчет разных частей модели.

Также установите (включите) еще три галочки ниже – чтобы клиент передавал свои данные на сервер (нажатия кнопок), принимал данные с сервера (текущие параметры коптера, вычисленные в модели) и выполнял асинхронные запросы данных – для снижения «заторможенности» пульта управления.

Порт приема данных надо здесь заменить на другой – установите его в 19001 (это поле слева, на рис. 10.3.8 оно указано неверно – там стоит еще значение по умолчанию).

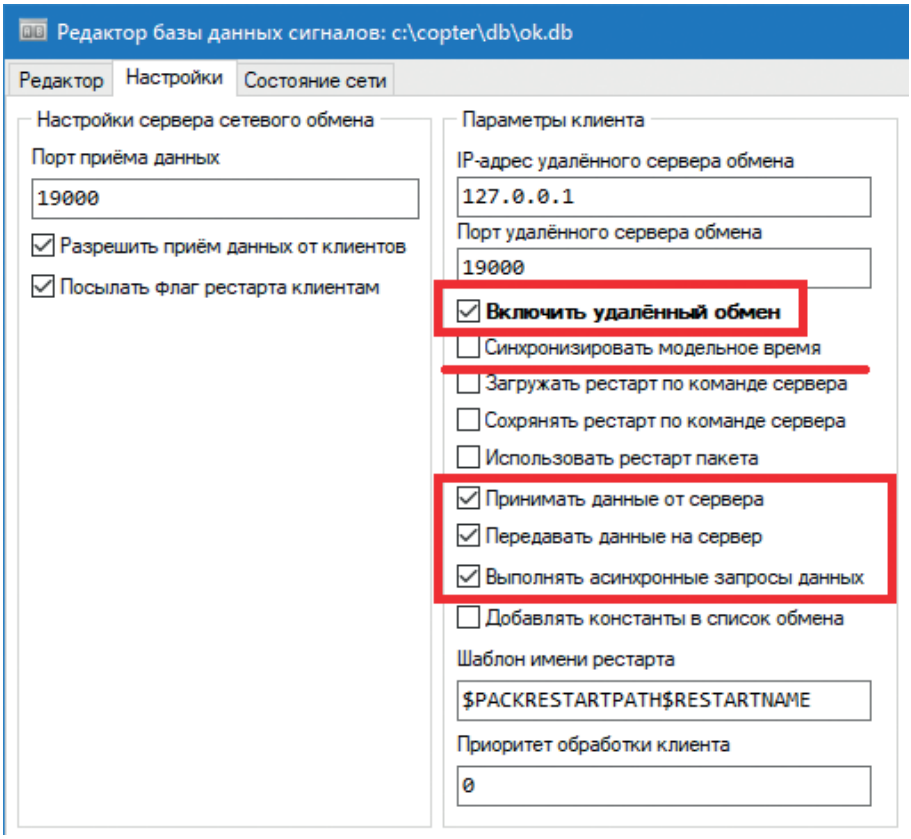


Рис. 10.3.8. Настройки базы данных как клиента (для файла pult.prt)

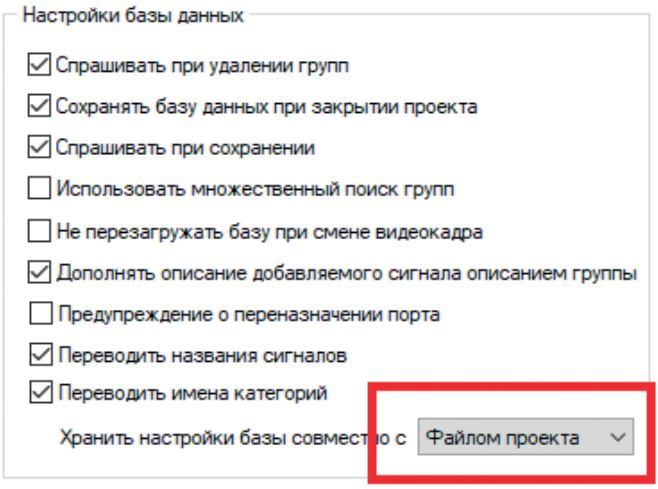


Рис. 10.3.9. Настройки базы данных разные для разных проектов

И есть еще один нюанс – так как мы используем одну и ту же базу данных, но с разными настройками, то настройки для проектов надо хранить отдельно – установите эту опцию для каждого из проектов **pult.prt** и **octocopter.prt**. Проверьте, что настройки не сбились при этом и при открытии пульта управления база настроена как клиент, а при открытии модели – как сервер (отличие будет в том что галочка **Включить удаленный обмен** будет выключена, а порт 19001 будет прописан только у клиента в левой части настроек).

После этих настроек, если вы попробуете запустить проект **pult.prt** на расчет, то система выдаст предупреждение, что модуль базы данных, настроенных как клиент, не может установить подключение с сервером, будет выдана ошибка вида: «[Ошибка]: «Не удалось соединиться с сервером обмена: Socket Error # 10061 Connection refused».».

Инициализация тем не менее произойдет, но сигналы базы данных будут оставаться константами, так как никто их в пустом проекте не рассчитывает.

Остановите расчет этого проекта, откройте файл **octocopter.prt**, настройте его на расчет с синхронизацией модельного времени, конечное время выставьте относительно большим (1e10, например), сохраните файл и запустите на расчет. Сверните файл (расчет при этом будет продолжать идти).

После этого вернитесь к пустому файлу проекта **pult.prt** и запустите его на расчет (предварительно проверьте, что здесь тоже скорость расчета синхронизирована с реальным временем и конечное время стоит большим) – при этом ошибки не должно быть, и, если в процессе такого расчета открыть интерфейс базы данных, через файл пульта управления, там можно будет увидеть все рассчитывающиеся в модели переменные. Но они считаются реально на другой «машине», к которой данным файлом мы подключились по сети. Такой же результат можно было бы добиться и с двумя компьютерами в одной сети, запустив на одном математическую модель, а на другом – пульт управления.

Примечание: пункт главного меню SimInTech **Инструменты → База данных…** работает в зависимости от того, какой из открытых проектов сейчас является активным. В нашем случае, когда открыты оба проекта, вы можете открыть два интерфейса базы данных, причем они будут с разными настройками и, вообще говоря, с разными текущими значениями переменных. Например, вы можете открыть оба интерфейса базы данных, затем нажать кнопку **Стоп** на проекте **pult.prt** и увидеть, что в окне базы данных, которое относится к этому проекту, текущие значения расчетных переменных перестанут изменяться – так как проект **pult.prt** отключится в момент остановки от сервера и перестанет получать обновления по переменным. При последующем пуске этого проекта обновление переменных (например, высоты, других координат коптера) возобновится. Таким образом, каждый из проектов имеет в памяти свой экземпляр базы данных, а обмен между ними происходит периодически и только по выбранным для чтения и записи переменным. Эти перечни можно увидеть на вкладке **Состояние сети** в интерфейсе базы данных.

Примечание 2: в больших проектах, где размер базы сигналов может достигать нескольких мегабайт и десятков мегабайт, бывает целесообразно уменьшать размер перечня сигналов для пульта – тогда специальным образом готовится база данных, таким образом, чтобы можно было легко отделить расчетные переменные от тех, которые нужны пульту (например, как мы сделали, выделив отдельно категорию «Кнопки»), и для проекта с макетом пульта управления готовится свой файл базы данных, в котором остаются только используемые пультом управления. То есть база данных для пульта является частным случаем (выборкой) из базы данных проекта. Вообще говоря, даже это не является обязательным – в отдельных случаях файл базы данных для проекта-клиента может быть совсем другим, совпадающим с сигналами базы данных проекта-сервера по каким-то переменным, а остальные сигналы могут быть и другими, которых не существует в проекте-сервере. Но в этом случае обмен сигналами будет возможен только по тем переменным, которые существуют в обеих базах сигналов – и на сервере, и на клиенте (т. е. по тому подмножеству, по которым списки сигналов пересекаются).

10.3.4. Примитивы для пульта

Для набора пульта управления используются разные графические примитивы, и это больше художественно-оформительская работа, чем математическая. С точки зрения модели и математики: есть графические примитивы, которые считывают переменные из модели и каким-то образом их либо отображают, либо используют при отображении (для изменения видимости, расположения, цвета и т. п.). Есть графические элементы, которые, наоборот, задают те или иные переменные в модели. Либо могут быть графические примитивы, сочетающие эти два направления передачи данных (например, сложные отображающие приборы с кнопками).

Для начала при помощи графических примитивов типа «Линия», «Полилиния», «Текст» и «Кнопка» сформируйте макет пульта, аналогичный рис. 10.3.10.

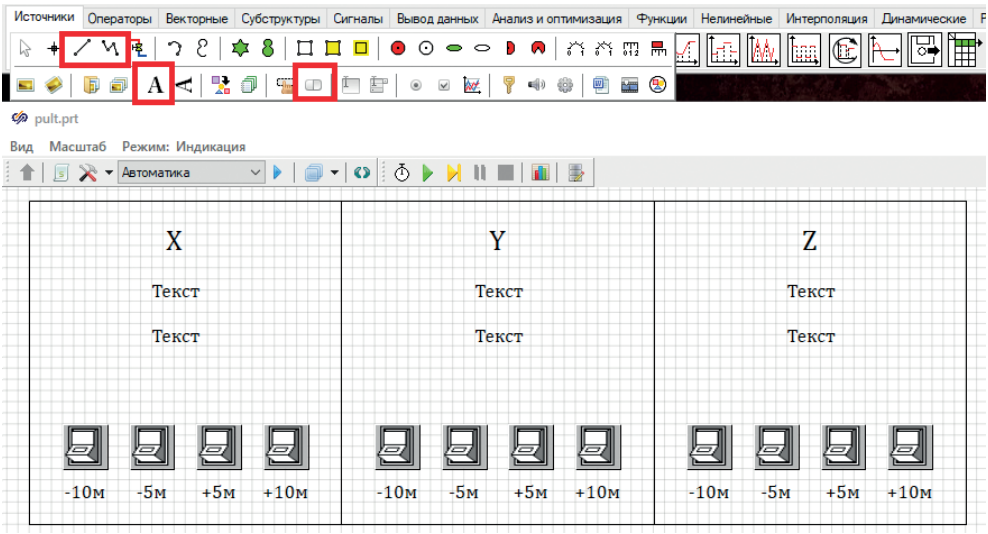


Рис. 10.3.10. Макет пульта управления октокоптером (начало)

Смысл примерно такой – по каждому из направлений будем выводить текущую координату (это первый текст) и заданное положение (координату) – второй текстовой надписью. Нажатием кнопок будем изменять заданное положение и визуально наблюдать здесь же на пульте и, возможно, на анимации, сделанной ранее, насколько быстро и успешно регуляторы и коптер справляются с «заданием».

Для лучшего выравнивания рекомендуется пользоваться привязкой к сетке, а также выравниванием по центру для текстовых надписей (где это применимо и целесообразно).

Чтобы элементами типа «Текст» вывести то или иное значение – можно пойти по варианту, аналогичному анимации, – т. е. задать каждому такому блоку свое имя, затем перейти в скрипт и задавать там свойства блоков <имя примитива>.Text = "…";

Этот вариант нормален, но несколько избыточен, если нам требуется в примитиве «Текст» вывести значение одной действительной переменной и статичный текст рядом с ней (например, единицы измерения) – для этого проще воспользоваться встроенными свойствами в примитив с именами «Текст», «Способ показа цифр» и «Отображаемое значение», см. рис. 10.3.11.

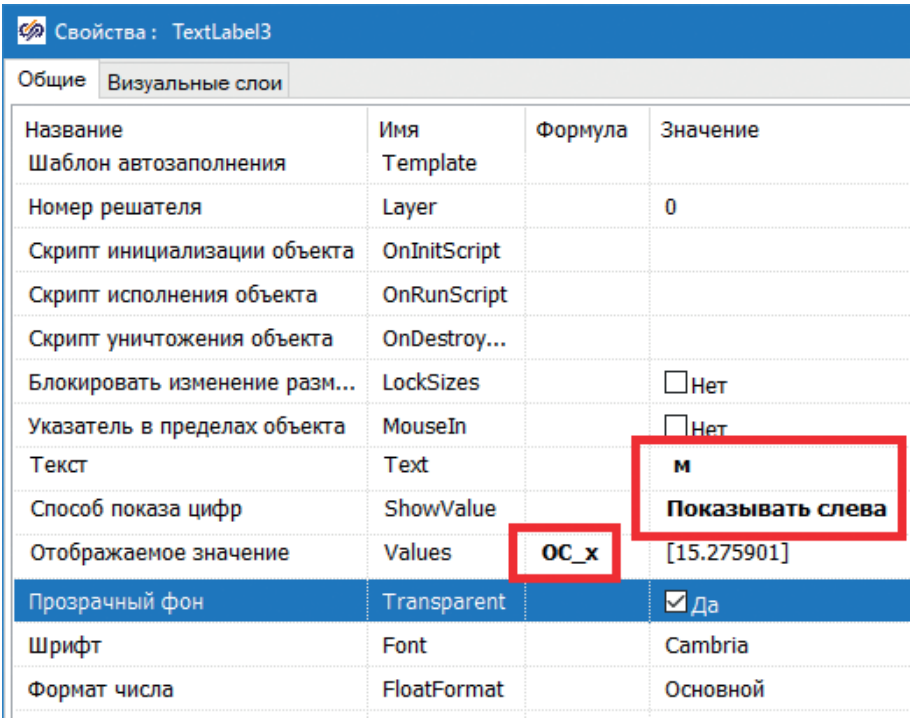


Рис. 10.3.11. Вывод текущих и заданных координат

В свойство «Текст» добавьте статичный текст, состоящий из символа пробела и буквы «м». Пробел нужен, чтобы между отображаемым числом и единицей измерения был пропуск.

Способ показа цифр поставьте в **Показывать слева**, чтобы единица измерения отображалась справа. В колонку **Формула** для **Отображаемого значения** впишите (в 6 графических примитивах), соответственно, сигналы OC_x, OC_y,OC_z, OC_xzad, OC_yzad, OC_zzad, как показано на рис. 10.3.12.

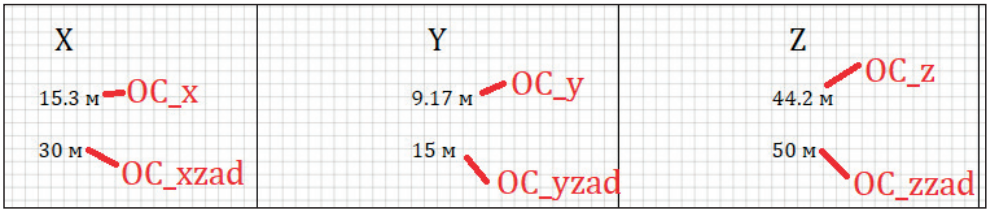


Рис. 10.3.12. Выводимые величины по каналам управления

Для того чтобы выведенные величины были более понятны оператору, лучше задать еще одну поясняющую подпись – например, слева от этих чисел со статичным текстом «текущее положение», «заданное положение». Также можно увеличить или задать другой шрифт, цвет и другие атрибуты для выводимых чисел. В итоге пульт, математически тот же самый, может трансформи-

роваться во что-то внешне совсем другое, например может стать похожим на рис. 10.3.13 (изменен цвет и размер шрифта, а также выведено три значащих цифры после запятой в выводимых числах).

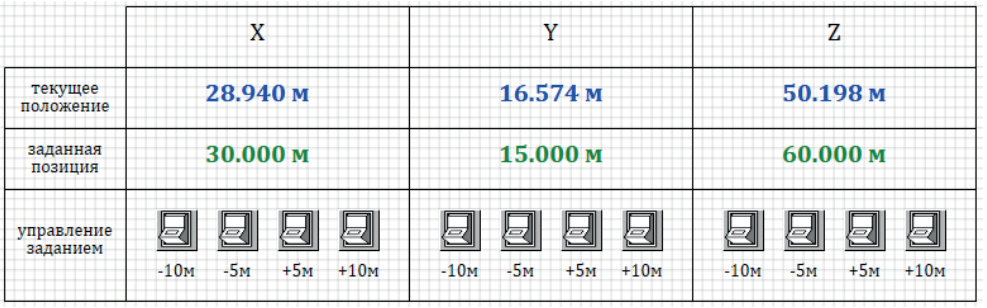


Рис. 10.3.13. Модификации внешнего вида пульта

10.3.5. Тестирование пульта, настройка внешнего вида и кнопок

Если вы до сих пор все сделали правильно, то теперь возможна такая проверка: запускаете на расчет модель, затем запускаете на расчет файл пульта управления, он подключается к модели и начинает с определенной периодичностью отображать текущее положение коптера и заданную позицию.

Если вас не устраивает такт обновления, можно это изменить в пункте меню **Вид → Интервал перерисовки**.

Среда SimInTech предоставляет относительно богатые возможности по формированию видеокадра (макета пульта управления) – можно скрыть полосы прокрутки, можно убрать панели инструментов и модельную сетку, можно также скрыть обрамление окна Windows и в последующем запускать пульт управления из командной строки с нужными опциями (подробнее см. в справке).

Также важно понимать, что пульт управления нарисован в таком же схемном окне, в котором формируется и расчетная схема. Однако в режиме пульта нажатия кнопки мыши на объектах должны обрабатываться по-другому, не так как в режиме редактирования. За это переключение отвечает опция **Индикация/редактирование** в панели инструментов схемного окна (рядом с названием расчетного слоя «Автоматика»). Видеокадр должен работать и запускаться в режиме индикации. В этом режиме (в частности) примитив типа «Кнопка» работает как кнопка (нажимаясь и отжимаясь по нажатию мышки), а не как блок с доступом к его свойствам.

Давайте перейдем к заданию воздействия на модель при помощи кнопок. Для этого задайте каждой кнопке свое уникальное имя, желательно заданное по какой-нибудь системе, например: BXmm, BXm, BXp, BXpp, BYmm, BYm, BYp, BYpp, BZmm, BZm, BZp, BZpp.

Далее, у каждой кнопки есть свойство с именем Down – оно будет становиться равным 1, если кнопка нажата, и равно 0, если кнопка не нажата. По данному свойству и предлагается написать скрипт. Ниже приведен этот скрипт (одинаковый, в принципе, для каждого направления), который надо вписать в скрипт

проекта **pult.prt**. Прежде чем писать скрипт, выставьте у каждой кнопки свойство Fixed в «нет», для того чтобы кнопка не могла фиксироваться (сама) в нажатом положении.

```
if BXmm.Down then PU_Xmm_Down = 1
    else PU_Xmm_Down = 0;
if BXm.Down then PU_Xm_Down = 1
    else PU_Xm_Down = 0;
if BXp.Down then PU_Xp_Down = 1
    else PU_Xp_Down = 0;
if BXpp.Down then PU_Xpp_Down = 1
    else PU_Xpp_Down = 0;

if BYmm.Down then PU_Ymm_Down = 1
    else PU_Ymm_Down = 0;
if BYm.Down then PU_Ym_Down = 1
    else PU_Ym_Down = 0;
if BYp.Down then PU_Yp_Down = 1
    else PU_Yp_Down = 0;
if BYpp.Down then PU_Ypp_Down = 1
    else PU_Ypp_Down = 0;

if BZmm.Down then PU_Zmm_Down = 1
    else PU_Zmm_Down = 0;
if BZm.Down then PU_Zm_Down = 1
    else PU_Zm_Down = 0;
if BZp.Down then PU_Zp_Down = 1
    else PU_Zp_Down = 0;
if BZpp.Down then PU_Zpp_Down = 1
    else PU_Zpp_Down = 0;
```

Как вы можете видеть, в этом скрипте для каждой переменной на каждом такте выполнения скрипта определено значение – 1 или 0, т. е. на каждом такте синхронизации в базу данных модели будет передано новое число. В самой модели – если вы поставили опцию «Автосброс» для блоков чтения сигналов (сигналов типа PU_Xmm_Down) при каждом такте работы и приходящей единицы, блоки чтения будут сбрасывать значение опять в ноль, и если кнопка будет зажата несколько тактов, то придет вновь 1, блок импульса по фронту сформирует еще один импульс, и заданное значение еще раз изменится. Короче говоря, скрипт уже реализует функцию автосброса, и если автосброс включен еще и в модели, то они будут избыточно дублировать друг друга и, накладываясь, реализовывать ситуацию «накрутки» заданного значения все время, пока кнопка нажата (пока нажата кнопка мыши). Чтобы этого избежать – надо либо в модели выключить автосброс, либо здесь, в скрипте, не писать вторые строки else … , так как переменная будет сама сбрасываться в 0. Лучше выключить в модели автосброс у блоков чтения сигналов, так как это довольно специфичная опция, которая редко используется и здесь не требуется – лучше формировать значение переменной в одном месте, а не в разных, для асинхронно работающих машин/моделей.

В итоге после написания такого скрипта и выключенной опции автосброса в принимающих блоках «Чтение сигналов» в модели нажатие кнопок на пульте (при корректно запущенной на расчет модели и подключенном пульте к ней) будет приводить к инкременту или декременту на 1 или 5 м для заданного значения координаты. Это в свою очередь приведет к изменению рассогласования в соответствующем канале управления регулятора и формированию управляющих команд на электродвигатели ВМГ и соответствующему перемещению коптера в «виртуальном» пространстве модели.

Если расположить удобным образом окно анимации, то вы получите «виртуальный октокоптер», которым можно управлять по направлениям, но пока еще с довольно плохо работающим регулятором. Перейдем к донастройке регуляторов, но прежде еще скорректируем изображение кнопок пульта.

10.3.6. Кнопки пульта

Кнопка (графический примитив) – это типовой объект в среде SimInTech, с двумя явно выделенными состояниями: «нажата» и «не нажата». Каждому из состояний соответствует свое графическое изображение, задаваемое свойством «Растровое изображение», также у кнопки есть ряд других свойств (см. рис. 10.3.14), которые могут пригодиться при создании макетов других пультов управления.

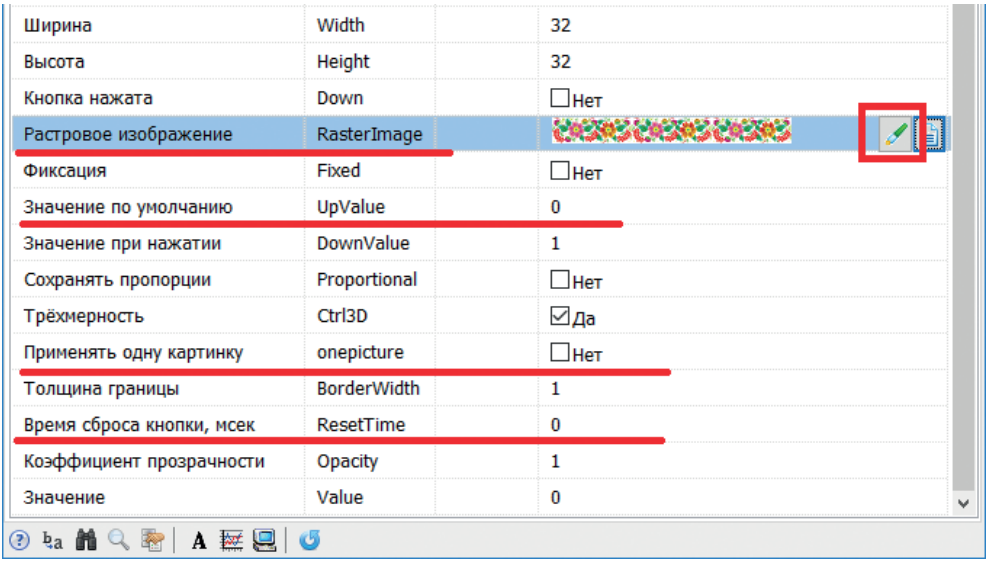


Рис. 10.3.14. Свойства графического примитива типа «Кнопка»

Сделано так, что графическое изображение кнопки – это растровое изображение, сохраняемое в проекте в стандартном формате BMP операционной системы Windows, причем размер этого изображения по ширине в пикселях должен быть в два раза больше, чем ширина блока в пунктах модельной сетки SimInTech, а высота должна совпадать. В левой части (половине) изображения должно храниться изображение одного из состояний кнопки, в правой – дру-

гого. При несовпадении размеров будет проведено масштабирование, поэтому будет лучше, чтобы размеры совпадали.

Для замены стандартного изображения на другое надо либо подготовить новый файл с картинкой нужных размеров, либо «вытащить» из проекта текущее изображение (для этого нажать выделенную на рис. 10.3.14 кнопку в свойстве растрового изображения, и в папке проекта появится новый bmp-файл) и отредактировать его. Предположим, вы достали из проекта изображение кнопки и «раскрасили» его по-другому или вообще заменили на что-то свое (см. рис. 10.3.15) – тогда при замене стандартной картинки пульт управления будет выглядеть по-другому, например как на рис. 10.3.16.

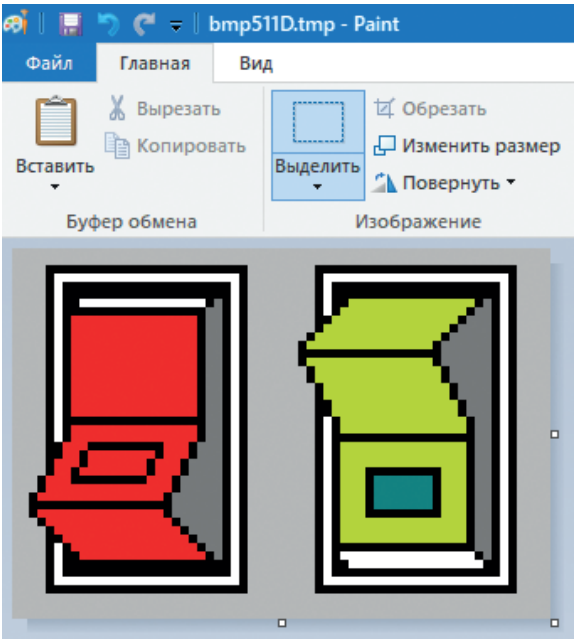


Рис. 10.3.15. Новое изображение кнопки

Таким образом, совмещая «дизайнерские» приемы, скрипт для анимирования и придания интерактивности графическим примитивам и их многочисленные особенности, возможности отображения результатов расчета и воздействия на модель, можно набирать достаточно разнообразные макеты и виртуальные пульты управления техническими объектами.

Если требуется сделать многопозиционный ключ управления, то применяют графический примитив типа «Сенсор», который реагирует на нажатие кнопки мыши на ту или иную область, и пишут скрипт, который по нажатию на отслеживаемую сенсором зону изменяет изображение в той или иной области пульта управления (или меняющий выводимое на экран изображение у примитива типа «Набор изображений»). Подробнее информацию можно почерпнуть из справочной системы SimInTech, из демонстрационных примеров и у разработчиков ПО напрямую через письменные запросы или в Telegram-канале вопросов и ответов.

X	Y	Z
17.550 м	16.348 м	50.084 м
19.000 м	15.000 м	60.000 м
 -5м -1м +1м +5м	 -5м -1м +1м +5м	 -5м -1м +1м +5м

Рис. 10.3.16. Обновленный пульт

10.4. О трехмерной анимации

В составе ПО SimInTech есть модуль трехмерной анимации, который позволяет сделать анимацию не только в плоскости, а в пространстве, в отдельном окне, где ряду объектов (предварительно подготовленных в том или ином 3D-пакете, в формате *.obj или *.stl) можно задавать координаты и углы поворота путем встроенных функций в скриптовом языке программирования. Пример можно посмотреть в папке демонстрационных примеров **C:\SimInTech\Demo\Визуализация и анимация\3D-визуализация**, там же можно увидеть и примеры применения модифицированного примитива типа «Кнопка» и др.

Например, в папке **Модель манипулятора (Кинематика)** есть макет пульта управления манипулятором с измененными изображениями кнопок, хотя математически все они выполняют ту же функцию передачи 0 или 1 в математическую модель объекта управления, как и в предыдущем подразделе.

Не описывая подробно, пример анимации для нашей модели коптера может быть реализован скриптом, приведенным ниже:

```
// === begin 3d sample animation ===
const sF = 10; // масштабный коэффициент
initialization
 window_id1 = Viewer3DCreate;
 Viewer3DSetAxesVisibility(window_id1, True);
 Viewer3DSetGridVisibility(window_id1, True);
 Viewer3DSetWindowSize(window_id1, 1400, 600);
 Viewer3DSetWindowPosition(window_id1, 130, 130);
 Viewer3DSetBackGround(window_id1, “Data\hipparcos_9.0.stars”, cl_Black);
 object_id11 = Viewer3DPlotFromFile(window_id1, “Data\v5.stl”, 0, 0, 0);
 Viewer3DSetScale(window_id1, object_id11, 1.0, 1.0, 1.0);
 Viewer3DSetColor(window_id1, object_id11, cl_Red);
 Viewer3DSetPosition(window_id1, object_id11, OC_X*sF, OC_y*sF, OC_z*sF);
 Viewer3DSetPitch(window_id1, object_id11, OC_PHI*180/pi);
 Viewer3DSetTurn(window_id1,  object_id11, OC_TET*180/pi);
 Viewer3DSetRoll(window_id1,  object_id11, OC_PSI*180/pi);
```

```
 Viewer3DSetCameraPosition(window_id1, 10*sF, 50*sF, 0*sF); //x,z,y
 for (i = 0, 100) begin
 object_id = Viewer3DPlotParallelepiped(window_id1, 1000, 1, 8, 0, -100, -i*10);
 if i mod 2 = 0 then
 Viewer3DSetColor(window_id1, object_id, cl_Black)
 else
 Viewer3DSetColor(window_id1, object_id, cl_White);
 end;
end;
Viewer3DSetPosition(window_id1, object_id11, OC_X*sF, OC_y*sF, OC_z*sF);
Viewer3DSetPitch(window_id1, object_id11, OC_PHI*180/pi);
Viewer3DSetTurn(window_id1,  object_id11, -OC_TET*180/pi);
Viewer3DSetRoll(window_id1,  object_id11, OC_PSI*180/pi);
// === end of 3d sample animation ===
```

Приведенный скрипт будет, в зависимости от модельных сигналов положения и ориентации коптера, рисовать в пространстве трехмерный объект в нужной точке и с нужными углами наклона. Объект изначально должен быть изготовлен в каком-либо трехмерном редакторе и сохранен в файл v5.stl – он загружается функцией `Viewer3DPlotFromFile()`.

Доработка и отладка регуляторов октокоптера

Модель октокоптера на данном этапе завершена в первом приближении. Однако остался ряд недоработок, которые мы специально оставили на последний этап работы, когда их будет удобнее делать с наличием анимации и пульта управления. Недоработок на самом деле еще много, поскольку в настоящей методике мы делали все по минимуму, но основные из них – три, и они следующие: а) в модели сопротивления воздуха коэффициенты сопротивления были заданы слишком большими, что повышало устойчивость модели, б) коэффициенты ПИД-регуляторов подобраны очень приблизительно, так чтобы коптер оставался устойчивым, но качество переходного процесса относительно низкое (долгое время переходного процесса, наличие перерегулирования и колебательности), в) в регуляторах скрыта ошибка, из-за которой коптер не приходит точно в заданную позицию, и эту ошибку читателю предлагается сначала найти самостоятельно, прежде чем двигаться далее.

После создания всех вспомогательных (но нужных) элементов к математической модели – графиков, анимации коптера (хотя бы вида спереди и сверху) и пульта управления для задания новых координат, в которые предписывается «лететь» октокоптеру – можно вернуться к отладке регуляторов и тщательнее настроить их коэффициенты, а также, возможно, модифицировать некоторые регуляторы, чтобы достичь лучшего качества управления. Отметим, что это можно было бы делать и ранее, без анимации и без пульта управления, просто с наличием этих составных частей модели процесс отладки будет более наглядным. В описании трудно представить весь рабочий процесс отладки, однако вы самостоятельно можете его проводить так, как вам удобнее, одновременно наблюдая за переходным процессом по графикам на анимации и задавая новые задания коптеру с пульта управления.

Также как самостоятельное упражнение можете выполнить перенос кнопок, задающих внешнее воздействие на коптер с расчетной схемы на пульт управления, чтобы все воздействия на модель были собраны водном месте, как и графики.

Вообще говоря, примененные ПИД-регуляторы как вид регуляторов лучше всего подходят для поддержания какой-то измеряемой величины в заданной позиции при относительно небольших отклонениях этой величины от заданного значения. Например, регулятор уровня в баке: если уровень может изменяться от 0 до 2 м и его нужно поддерживать в значении +1 м, то для такого регулятора мы имеем максимально возможное отклонение параметра плюс-минус 1 м от заданного. В такой ситуации ПИД-регулятор подойдет лучше всех. Для

коптера ситуация несколько сложнее – например, заданная координата может отстоять от текущей на сотни и тысячи метров, а точность позиционирования должна быть, например, плюс-минус 10 см или даже еще меньше. В этом случае ПИД-регулятор, настроенный на малые отклонения, даст плохой переходной процесс на больших расстояниях, а настроенный на большие отклонения даст плохой результат в окрестности заданной координаты. Для углов ориентации ПИД-регулятор хорош. Но, кроме этого, могут быть наложены еще ограничения на максимальную скорость перемещения и другие дополнительные требования… Квалификации автора настоящей методики не хватает, чтобы дать ответ в общем виде, какой именно вид и структуру регулятора следует применить в модели коптера для достижения наилучшего качества управления, тем более в общем виде, когда параметры объекта управления до конца не определены, однако если оставлять ПИД-регулятор как основу, то можно наметить следующее направление модификации регуляторов: оставить ПИД-регулятор как основной способ регулирования коптера, когда он находится в некоторой окрестности заданной точки (скажем, когда рассогласование по координате не более 10 м), и переключать ПИД-регулятор координаты в ПД-регулятор скорости при отклонениях, больших чем 10 м. То есть идея в следующем – когда до точки назначения лететь надо на значительную дистанцию и долго, коптеру надо набрать постоянную (максимально допустимую) скорость в нужную сторону и поддерживать ее, а когда он достигнет (примерно) точки назначения, там уже включать ПИД-регулятор, который точно доведет то заданной точки с уже уменьшенной скоростью и точным позиционированием.

11.1. Сопротивление воздуха

Скорректируем коэффициент, определяющий сопротивление воздушной среды движению воздуха. Вообще говоря, этот коэффициент лучше получить практически как результат эксперимента, но теоретически можно дать оценку его величине. Если принять плотность воздуха равной 1.2 кг/м^3, коэффициент формы $C_D = 1$ (как для куба), а длину стороны этого куба задать равную 2 м (размах лучей коптера), тогда площадь сечения будет равна 4 м^2, а коэффициент сопротивления для вычисления момента сопротивления будет примерно равен -0.5×1.2 кг/м^3 $\times 1 \times 4$ м^2 $\times 1$ м $= 2.4$ кг (формулы смотрите в разделе 3.2). Это верхняя оценка для коэффициента сопротивления перед квадратом угловой скорости при вычислении момента сопротивления воздуха при вращении вокруг осей X и Y, вокруг оси Z она будет в 8 раз больше, так как лучей всего 8 при вращении вокруг этой оси. Для коэффициента перед квадратом линейной скорости (при вычислении силы сопротивления воздуха) оценка будет такая же, так как длину луча мы приняли равную 1 м.

Поскольку приближение, что коптер – это куб со стороной 2 м, довольно грубое, можно снизить этот коффициент в несколько раз. Примем для простоты, что он будет равен 1 (кг) для силы сопротивления воздуха и для момента сопротивления воздуха при вращении коптера.

Подставьте вместо –50 (того числа которое стоит в модели) величину –1 в субмоделях для вычисления Fdx, Fdy, Fdz, а также в субмоделях для вычисления Mdx, Mdy. В субмодели Mdz поставьте –8. Для примера см. рис. 11.1.1.

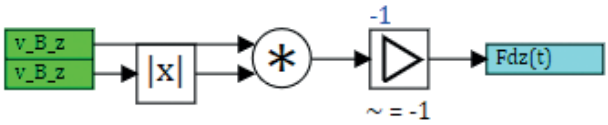


Рис. 11.1.1. Скорректированный коэффициент сопротивления воздуха

После этого если вы запустите модель на расчет, то увидите качественно другое поведение модели – регуляторы будут пытаться привести коптер к заданному положению, но так как воздух стал гораздо менее сопротивляющимся, то перерегулирование увеличится, и у системы появится большая колебательность, и вообще, система станет неустойчивой – регуляторы будут раз за разом раскачивать коптер все сильнее и сильнее вокруг заданной точки. Для примера, в нашем случае были заданы координаты, равные +15м по обеим осям, и горизонтальные скорости получились в начале переходного процесса вроде бы устойчивыми, но потом начался расходящийся процесс, как показано на рис. 11.1.2. Конечно, это неудовлетворительный результат, а регуляторы требуют доработки.

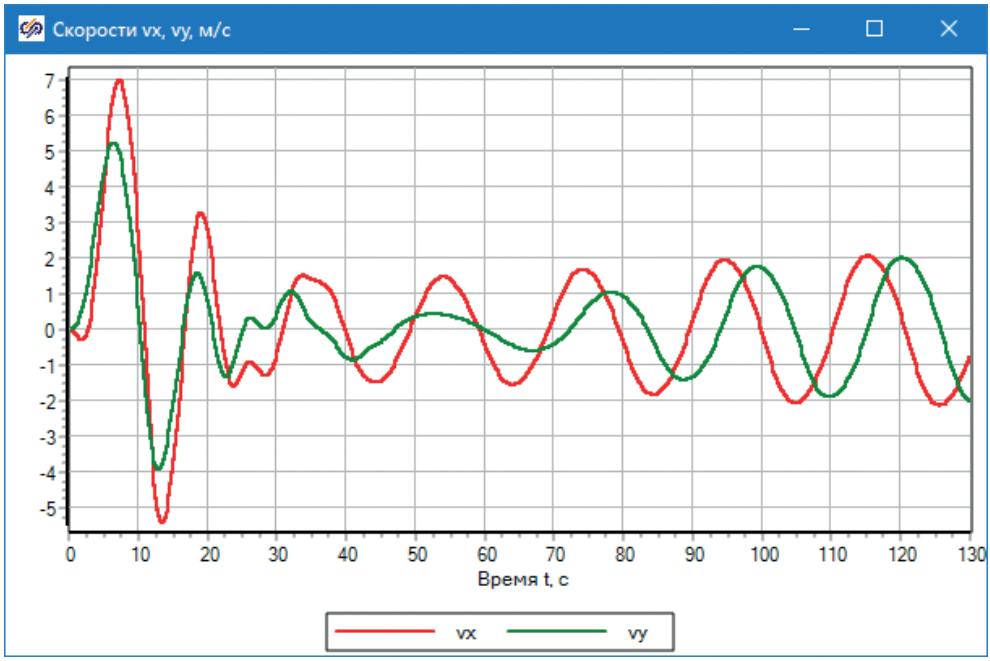


Рис. 11.1.2. Неустойчивое регулирование

11.2. Корректировка регуляторов

Каков принцип настройки регуляторов? Чаще всего принцип следующий – или на основе предыдущего опыта в сходных проектах, либо путем оптимизации вручную, либо оптимизации автоматически некоторыми из стандартных методов (метод градиентного спуска, симплекс-метод и др.). Задача заключается в том, чтобы подобрать N коэффициентов регуляторов для достижения некоторых из параметров регулирования. Мы будем делать это вручную и на глаз, определяя каждый раз моделированием, достаточно ли хорош переходной процесс.

Например, сейчас нас не устраивают регуляторы крена и тангажа. В модели они сделаны одинаковым способом, т. е. имеют одни и те же коэффициенты, заданные через базу данных как REGPHITETA_P, REGPHITETA_I, REGPHITETA_D и REGPHITETA_D2. Будем изменять с некоторым шагом настроечные коэффициенты регуляторов по очереди и смотреть, как это скажется на качестве переходного процесса.

В пошаговом режиме это довольно трудно описать, попробуйте самостоятельно задать для REGPHITETA_P следующие значения: 1, 2, 5, 7, 10 и провести последовательно 5 расчетов с заданием по осям OC_xz = 5 м и OC_yz = 5 м, конечным временем 2 мин (можно поставить с некоторым запасом 130 с) и оценить, какой коэффициент усиления для П-ветви регулятора наиболее оптимален при неизменных значениях других коэффициентов. Идея в том, что на 5 м по обеим осям коптер должен перелететь за время менее минуты, достаточно точно и без сильного перерегулирования. Сначала вы можете наблюдать за полетом и неспособностью регуляторов справиться с задачей при моделировании в реальном масштабе времени (с включенной синхронизацией с реальным временем), наблюдая за полетом коптера на анимации и одновременно на графиках, при коэффициенте REGPHITETA_P=1, далее для быстроты можно выключить синхронизацию с реальным временем и повторять моделирования в ускоренном режиме, анализируя по получающимся графикам. По анимации понять, что происходит в модели, – проще, поэтому мы ее и сделали, но по графикам – быстрее. Поэтому при настройках регуляторов используют, как правило, фазовые координаты и временные графики переходных процессов. Результаты 5 моделирований представлены на рис. 11.1.3…11.1.7 (значения остальных коэффициентов REGPHITETA_I=0.1, REGPHITETA_D=9 и REGPHITETA_D2=3). Для того чтобы графики отличались по осям, задана начальная скорость по оси Y, равная –0.5 м/с, т. е. в начальный момент времени коптер горизонтален и летит в неправильном направлении по оси Y).

Также внесите еще одну корректировку – по сравнению с рис. 9.5.2, поставьте коэффициент пропорциональности, равный 0.001 (а не 0.01, как на рис. 9.5.2) при формировании заданного угла наклона, чтобы заданный угол не был таким большим – при сниженной вязкости воздуха коптеру не надо так сильно наклоняться, как ранее.

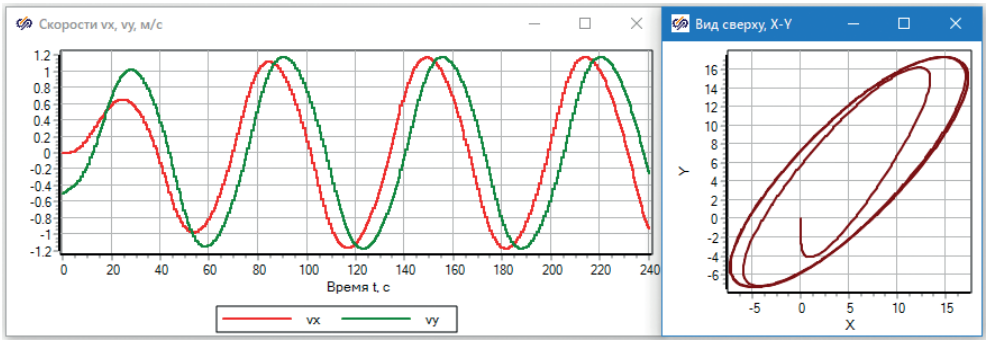


Рис. 11.1.3. REGPHITETA_P=1

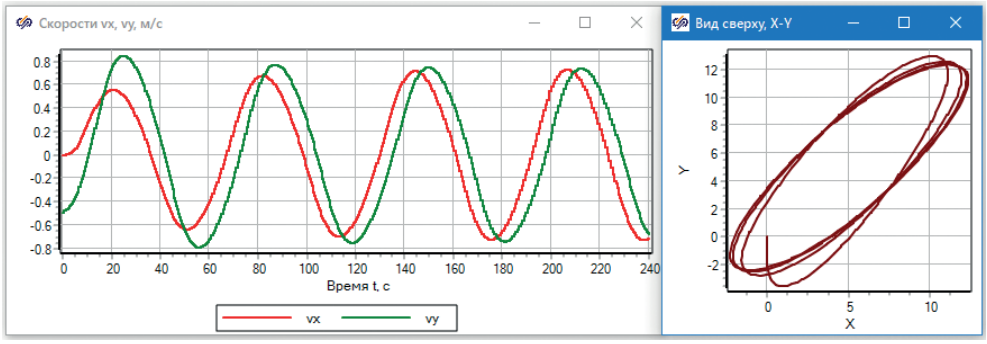


Рис. 11.1.4. REGPHITETA_P=2

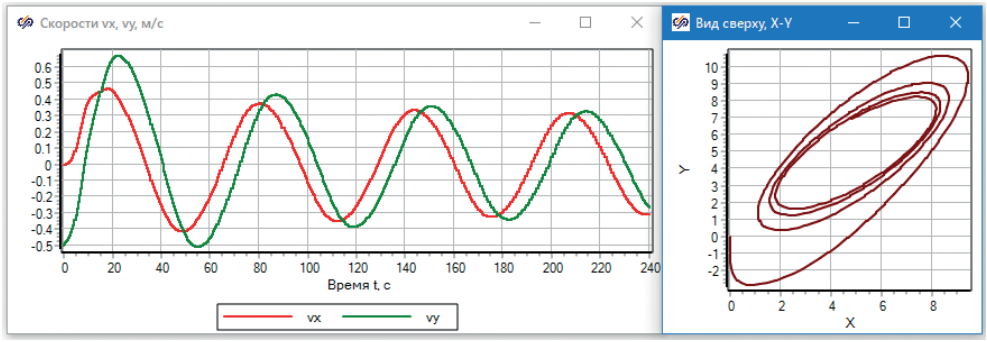


Рис. 11.1.5. REGPHITETA_P=5

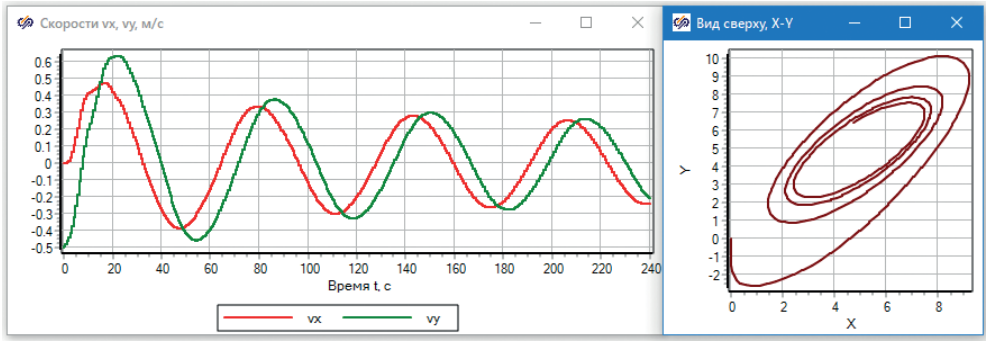


Рис. 11.1.6. REGPHITETA_P=7

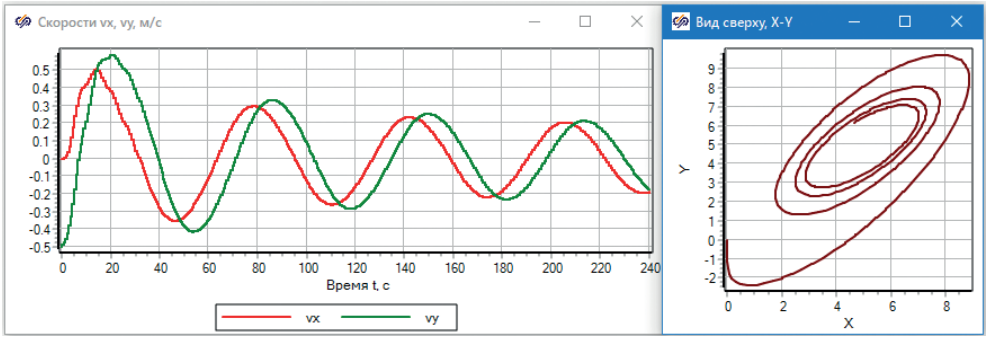


Рис. 11.1.7. REGPHITETA_P=10

Вы также можете провести тестовые моделирования при других значениях P, как меньших 1, так и больших 10 – и в принципе, можно найти некий оптимальный коэффициент, который даст меньшую величину перерегулирования при заданных постоянными остальных коэффициентах.

Аналогичную процедуру можно (и нужно) провести с другими коэффициентами регуляторов, и, возможно, не один раз – найдя оптимальные значения других коэффициентов, можно вернуться снова к P и еще раз подобрать оптимум – вполне возможно, что при другом наборе значений оптимум для P будет тоже другим. И таким итерационным способом можно найти оптимальный набор коэффициентов, при котором будет достигнуто максимальное качество переходного процесса.

Однако в модели коптера в текущей конфигурации вам вряд ли удастся найти такой оптимальный набор коэффициентов…

11.3. Введение еще одной обратной связи в регуляторы крена и тангажа

Дело вот в чем – ранее с сильной «вязкостью» воздуха сама модель силы сопротивления воздуха тоже была как бы частью регулирующего механизма, и коптер тоже совершал колебательные движения вокруг точки равновесия, но не очень сильные по амплитуде. Когда мы снизили в модели в 50 раз коэффициент сопротивления воздуха, диссипация энергии снизилась (снизилось демпфирование в модели), и коптер перестал эффективно «тормозиться» при ненулевой поступательной скорости движения, а регулятор, который у нас есть, – принципиально не способен заранее «затормозить» коптер до достижения заданной точки, так как у него отсутствует нужная обратная связь. При любых коэффициентах ПИД-регулятора коптер все равно будет летать с перерегулированием и не сможет точно остановиться в заданной точке.

Действительно, ветви регуляторов тангажа и крена, работающие по рассогласованию текущего и заданного положения, направлены на движение к заданной точке до тех пор, пока коптер не достигнет заданной точки, и по инерции он пролетает мимо заданного положения. Например, коптеру надо лететь вперед на +5 м и, пока он не достигнет этой отметки, регулятор будет его наклонять вперед. Но для того, чтобы коптер остановился в отметке +5 м, его надо заранее – пока он еще не прилетел туда – наклонить назад.

Такой «работой на опережение» в ПИД-регуляторах занимается дифференциальная ветвь, но в наших регуляторах мы этого не предусмотрели для регуляторов направлений, так как задействовали для каналов управления по осям X и Y регуляторы углов наклона. В них, конечно, есть Д-ветви, но они работают по угловой скорости для стабилизации угла наклона, а не по линейной.

Добавьте еще одну обратную связь в регуляторы крена и тангажа, заведя в них линейные скорости vy и vx, сначала с минимальным коэффициентом, например 0.01 и –0.01, как показано на рис. 11.1.8 и 11.1.9. Минус нужен, так как направление, вращение по углу тангажа и углу крена не одинаково (не одинаково совпадает с направлением осей X и Y). Для этого надо в сумматорах прописать вектор из 5 единиц, поставить на схему еще по одному усилительному звену и звенья типа «Из памяти». Строго говоря, здесь надо было бы брать проекции вектора скорости в системе координат B, но, так как мы пока имеем дело с коптером, у которого нулевой курс, можно взять и сигналы vx, vy из инерциальной системы координат.

После этого если вы промоделируете тот же переходной процесс, то он не должен сильно отличаться от предыдущих итераций, но можно заметить слабое сокращение (затухание) колебаний около заданной точки. Постепенно увеличивайте коэффициент при vx, vy – 0.02, 0.03 и т. д.

Через некоторое время при значениях коэффициента усиления порядка 0,06 и выше можно будет заметить (постепенный) качественный переход от колебательного характера переходного процесса к апериодическому. Для примера см. рис. 11.1.10 – это переходной процесс с коэффициентом усиления = 0.1 для линейной скорости.

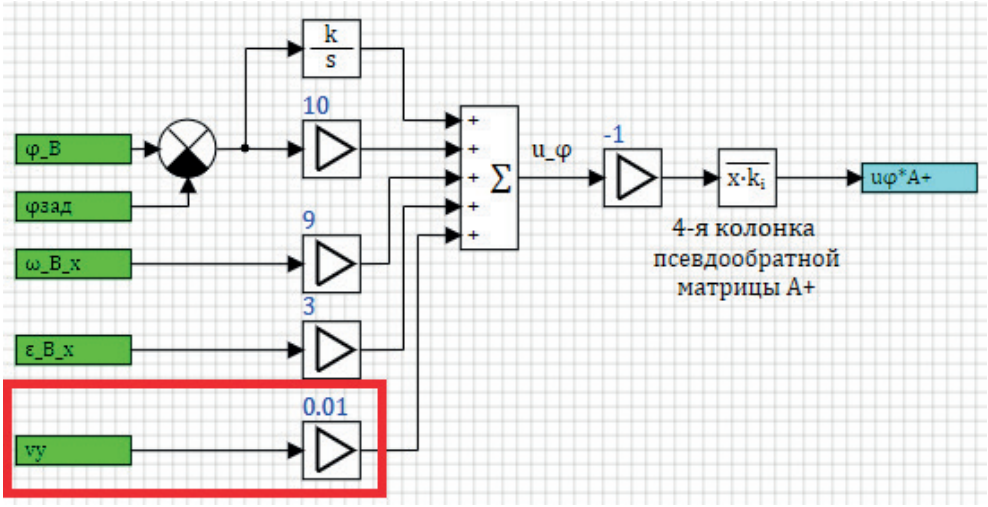


Рис. 11.1.8. Обратная связь по скорости vy для регулятора крена

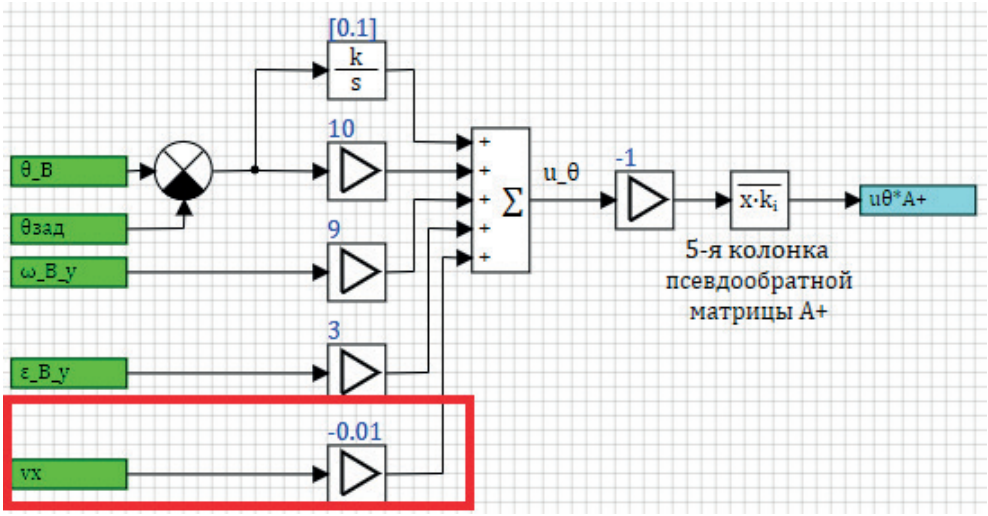


Рис. 11.1.9. Обратная связь по скорости vx для регулятора тангажа

Видно, что процесс носит уже явно апериодический характер, а не колебательный, есть некоторое перерегулирование и на начальном этапе колебательные процессы. Однако сама система уже является явно устойчивой, и дальше итерационным способом, который был описан в предыдущем подразделе, можно постепенно найти оптимальные или близкие к оптимальным значения коэффициентов усиления, меняя их в пределах от нуля до примерно 10...30. Проделайте это сначала самостоятельно. Удобнее при этом расположить на экране рядом и окно анимации, и базу данных, и графики, держа открытыми 2-3 характерных графика.

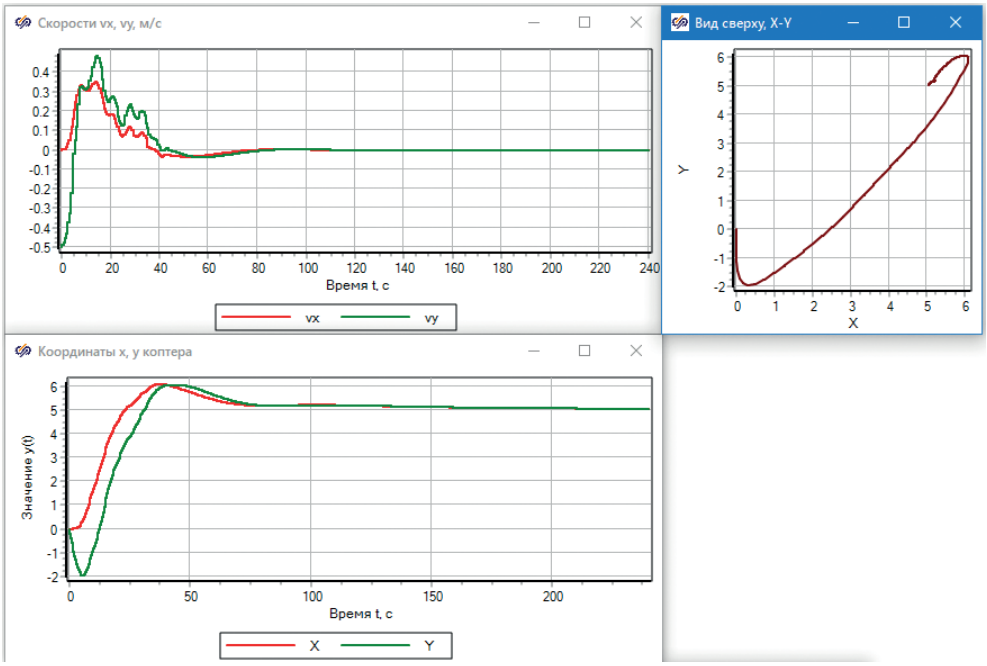


Рис. 11.1.10. Переходной апериодический процесс

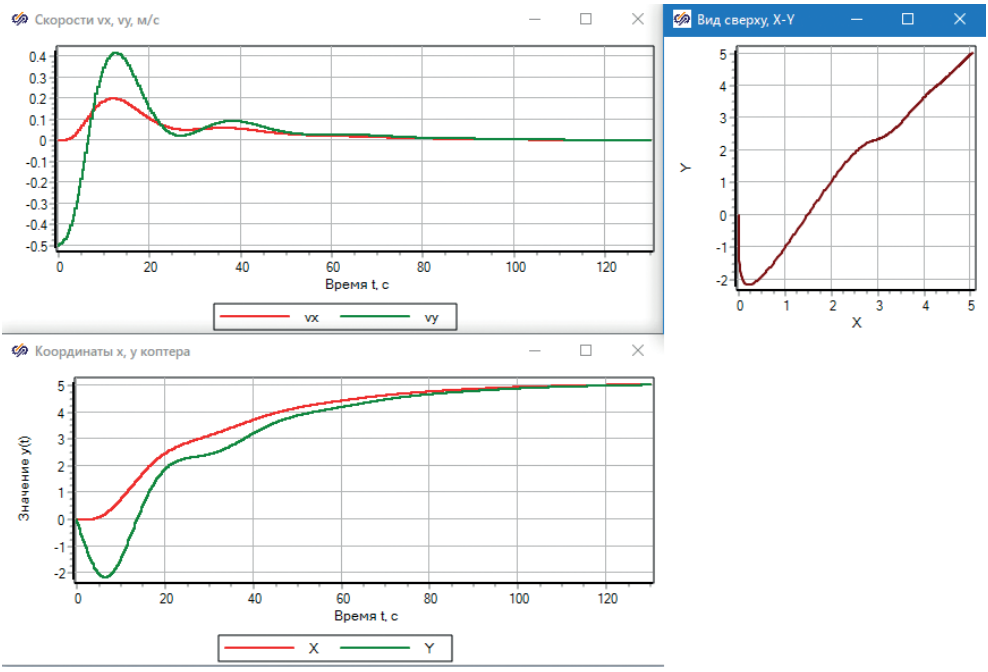


Рис. 11.1.11. Переходной процесс, уже относительно качественный

В качестве иллюстрации приведем еще график переходного процесса, который получается при следующих коэффициентах регулятора: REGPHITETA_P = **5**, REGPHITETA_I = **0.003**, REGPHITETA_D = **25**, REGPHITETA_D2 = **3**, коэффициент при vx, vy равен **0.15**, а коэффициент x_I_зад, y_I_зад равен **1/1000**. Это не самые оптимальные коэффициенты, однако они дают относительно плавный переходной процесс, длящийся менее 2 мин и без перерегулирования, см. рис. 11.1.3. Даже при начальном возмущении по горизонтальной скорости вдоль оси Y регулятор приводит коптер в позицию +5 м примерно за те же 2 мин, как и по оси X. Конечно, 2 мин для коптера – это очень длительное время, и регулятор требует дальнейшей доработки для получения большего быстродействия, при сохранении устойчивости и плавности работы, однако для пошаговой методики и введения в математическое моделирование коптера полученный результат более чем удовлетворителен.

11.4. Что дальше?

Одна из возможных доработок регуляторов – это разделение регулятора на регулятор положения и регулятор скорости. Например, при отклонении от заданной точки более 3–5 м можно переключать регулятор на другой способ, при котором он будет работать не на минимизацию рассогласования текущей позиции от заданной, а на поддержание заданной постоянной скорости. Вариант такого регулятора представлен на рис. 11.1.12, полученный переходной процесс – на рис. 11.1.13. Представленный вариант не самый лучший, можно даже сказать, что неудовлетворительный, так как регулятор скорости входит в противоречие с регулятором угла наклона и некоторыми другими ветвями регулятора.

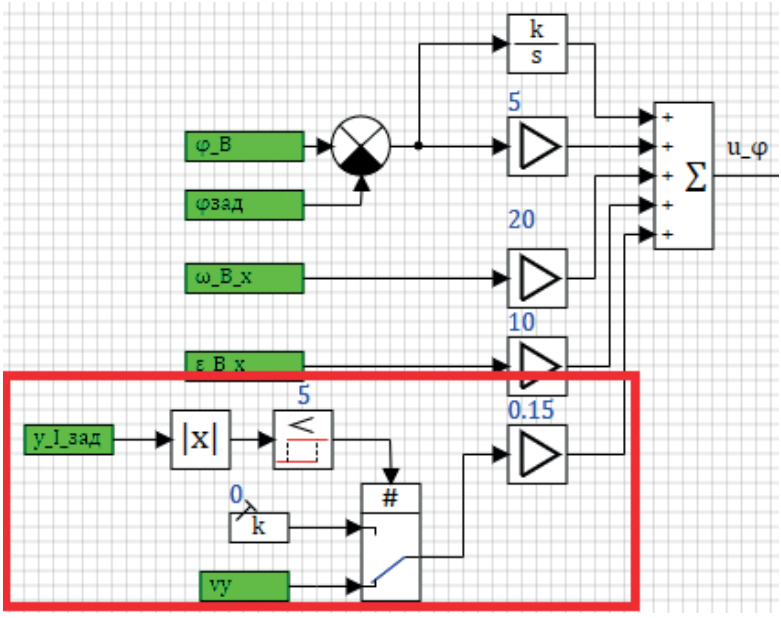


Рис. 11.1.12. Регулятор скорости в нулевом приближении

Идея этой коррекции заключается в следующем: когда рассогласование между заданной и текущей позицией менее 5 м, регулятор работает, как и раньше – приводит коптер в заданную позицию с некоторым «торможением», заранее до заданной точки. Если рассогласование более 5 м, то вместо скорости vy подается постоянная величина (ноль), и коптер летит с максимальной скоростью, которую ему удается развить при максимальном угле наклона и сопротивлении воздуха. Углом наклона можно «подбирать» максимум скорости или, наоборот – задаться некоторой максимальной скоростью и в зависимости от нее задавать максимальный допустимый угол наклона коптера.

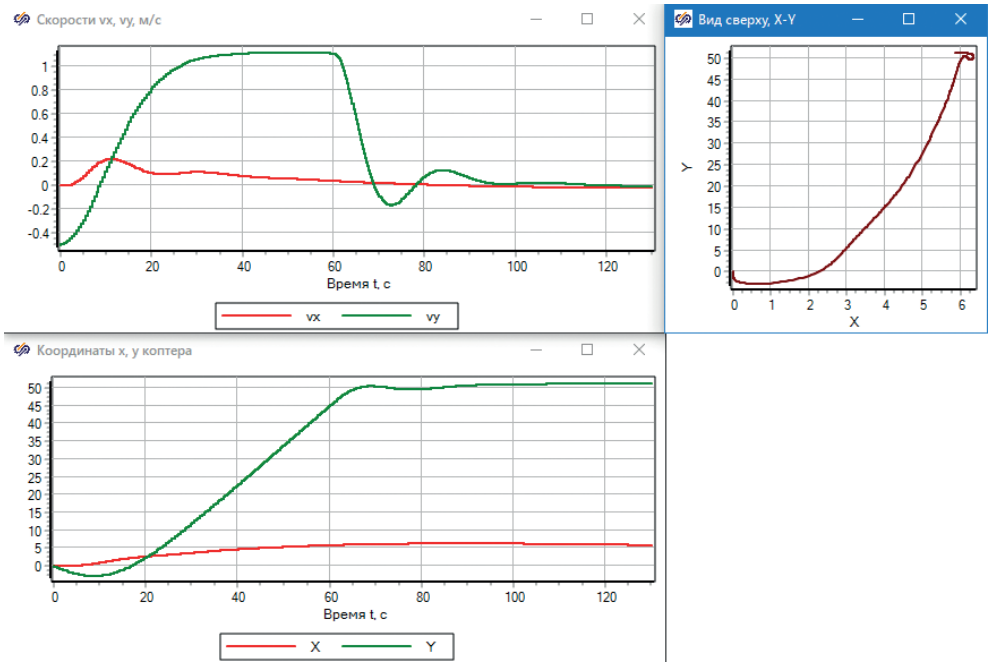


Рис. 11.1.13. Полет на +50 м по оси Y с постоянной скоростью

По графикам переходного процесса видно что первые 20 с коптер разгонялся, затем происходил полет с примерно постоянной (по оси Y) скоростью около 1 м/с (в основном выход на максимальную скорость произошел за счет сопротивления воздуха), а в момент, когда коптер пришел на позицию 45 м, регулятор переключился на «тормозящий» вариант, и коптер довольно быстро пришел на заданную позицию, хотя и с некоторым перерегулированием – это вопрос к доработке максимального угла наклона или других характеристик регулятора. Например, при рассогласовании более 5 м, скорее всего, надо отключать И-ветвь регулятора, чтобы там не накапливался большой интеграл, – эту ветку надо включать тоже в области плюс-минус 5 м от заданной позиции. Также регулятор требует и других изменений, но эти вопросы уже выходят за рамки настоящей пошаговой методики.

На рис. 11.1.14 показаны углы крена и тангажа при этом режиме – видно, что в процессе полета с постоянной скоростью коптер был наклонен на мак-

симально допустимый угол (в нашей модели он сейчас задается в диапазоне плюс-минус полградуса).

На текущей версии модели и регуляторов возможно придумать еще сценарии проверки и доводки – можно воздействовать внешним возмущающим воздействием по направлениям и смотреть на анимации и графиках реакцию октокоптера и регуляторов на такие возмущения. Сами возмущения можно менять по величине и найти «предел» возможности коптера по сопротивлению внешним нагрузкам.

Можно менять массу коптера до расчета или непосредственно в процессе моделирования, тогда будет виден переходной процесс в варианте, что коптер принял (или сбросил) дополнительный груз.

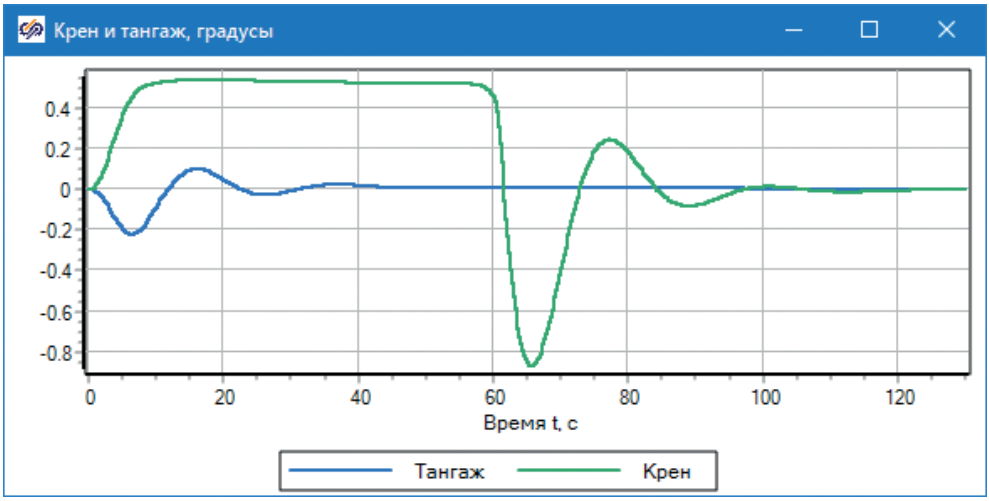


Рис. 11.1.14. Тангаж и крен при регулировании скорости

Можно также наметить другие идеи по дальнейшей доработке представленной модели – сделать модель датчиков, модель электродвигателей и их пультов управления;, скорректировать модель с учетом специфики работы задатчика оборотов (то, что частота вращения регулируется не плавно, а некоторыми шагами по шкале оборотов), можно структурно в SimInTech регулятор вынести в свой проект и сделать пакет проектов, приближаясь к проекту, который в дальнейшем будет являться «прошивкой» полетного контроллера. На пульте управления можно реализовать некоторый набор приборов, вывод другой телеметрии, в модели можно добавить модель воздуха, ветровой нагрузки и возмущений и т. д. и т. п.

На этом учебно-демонстрационная модель завершена.

Заключение

Среда SimInTech позволяет просто, в едином интерфейсе и достаточно эффективно создавать как модели динамики технических объектов (например, октокоптера), так и модель алгоритмической части его системы управления, включая модели исполнительных механизмов и датчиков, панель управления и элементы анимации. Имея перед собой реализованную модель и модель системы управления, можно удобно анализировать совместное поведение регуляторов и объекта, уточнять модель и проектировать регуляторы, достигая требуемого качества управления и регулирования.

Регуляторы, представленные в настоящей методике, специально не сделаны «хорошими» (хотя они и обеспечивают приемлемое качество регулирования), чтобы не раскрывать некоторые ноу-хау. Например, вы можете самостоятельно внимательно изучить рис. 5.3.1 и увидеть, что регулятор, представленный на нем, отличается от типового ПИД-регулятора, описанного в пошаговой методике. По сути, в версии, представленной на рис. 5.3.1, реализовано два регулятора (положения и скорости), которые переключаются из одного в другой. Аналогичные усовершенствования можно (и нужно) добавлять к регуляторам по другим направлениям

На базе полученной модели динамики при добавлении в нее более подробной модели электродвигателя и аккумуляторной системы можно получить расчетную модель для вычисления максимального времени полета и энергозатрат при варьировании сценария полета и способа регулирования.

Список литературы

1. Design, Modeling and Control of an Octocopter, Oscar Oscarson, Royal Institute of Technology 2015.
2. Development, Modelling and Control of a Multirotor Vehicle, Markus Mikkelsen, Umeå University 2015.
3. *Савицкий А. В.*, *Павловский В. Е.* Модель квадрокоптера и нейросетевой алгоритм управления // Препринты ИПМ им. М. В. Келдыша. 2017. № 77. 20 с. Режим доступа: doi:10.20948/prepr-2017-77 URL: http://library.keldysh.ru/preprint.asp?id=2017-77.

Книги издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «Планета Альянс» наложенным платежом,
выслав открытку или письмо по почтовому адресу:
115487, г. Москва, 2-й Нагатинский пр-д, д. 6А.
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.
Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: **www.a-planeta.ru**.
Оптовые закупки: тел. **(499) 782-38-89**.
Электронный адрес: **books@alians-kniga.ru**.

Щекатуров Александр Михайлович

Методика моделирования динамики октокоптера

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Зам. главного редактора *Сенченкова Е. А.*
Корректор *Абросимова Л. А.*
Верстка *Луценко С. В.*
Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.
Гарнитура «PT Serif». Печать цифровая.
Усл. печ. л. 18,53. Тираж 200 экз.

Веб-сайт издательства: www.dmkpress.com

Ссылка на дополнительные материалы книги

На примере задачи моделирования динамики октокоптера представлено пошаговое описание процесса реализации такой модели в SimInTech, моделирования системы управления и регулирования для октокоптера. Выполнен последовательный переход от теоретической проработки модели к её реализации в виде конкретной расчетной схемы во входо-выходных соотношениях.

Книга предназначена для студентов технических специальностей, выпускников вузов или инженеров, желающих самостоятельно научиться математическому моделированию динамики в среде SimInTech на примере беспилотного летательного аппарата коптерного типа. Она может быть полезна также и тем, кто хочет самостоятельно разработать свою динамическую модель беспилотного летательного аппарата.

ISBN 978-5-97060-933-0

Москва 2021