МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
имени академика С.П. КОРОЛЁВА»

А.В. КУЗНЕЦОВ, Г.М. МАКАРЬЯНЦ

Синтез нечёткого регулятора при помощи пакета прикладных программ системы Matlab

Методическое пособие

Самара 2016

УДК: 681.51

Авторы: А.В. Кузнецов, Г.М. Макарьянц

Синтез нечёткого регулятора при помощи пакета прикладных программ системы Matlab: методическое пособие / [*А.В. Кузнецов, Г.М. Макарьянц*]. – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2016. – 59 с.: ил.

В методическом пособии описана работа в программном пакете Matlab/Simulink. Приведены примеры настройки различных видов регуляторов.Рассмотрены методики создания ПИД-регулятора, fuzzy-регулятора и ПИД-регулятора с настройкой коэффициентов по fuzzy. Пособие предназначено для студентов технических специальностей и направлений.

УДК СГАУ: 681.51

Содержание

1. МАТЕМАТИЧЕСКОЕ ОПИСАНИЕ

1.1. Объект управления

В качестве объекта управления выступает малоразмерный газотурбинный двигатель, имеющий следующие характеристики:

Максимальная частота вращения, об/с	n_{max}	2000
Номинальная частота вращения, об/с	n	1420
Минимальная частота вращения, об/с	n_{min}	200
Расход топлива на номинальной частоте вращения, кг/с	G_t	0,009
Постоянная времени двигателя, с	$T_д$	1,45
Транспортная задержка, с	τ	0,04

Запишем приведённую модель в виде апериодического звена первого порядка[1, 2]:

$$W_д(s) = \frac{K_д}{T_д s + 1} = \frac{157777}{1{,}45s + 1};$$

$$K_д = n / G_t = 1420 / 0{,}009 = 157777.$$

Насос-дозатор имеет следующие характеристики:

Напряжение на электродвигателе насоса, В	$V_н$	10
Максимальный расход топлива через насос, кг/с	G_t	0,02
Постоянная времени насоса, с	$T_н$	0,2

Передаточная функция насоса-дозатора:

$$W_н(s) = \frac{K_н}{T_н s + 1} = \frac{0{,}002}{0{,}2s + 1};$$

$$K_н = G_t / V_н = 0{,}02 / 10 = 0{,}002.$$

У ШИМ (широтно-импульсная модуляция) есть параметр опорной частоты ν и коэффициент усиления $K_{ШИМ}$. Обычно частота ШИМ выбирается исходя из границы слышимости человеческим ухом и составляет около 24

кГц, однако при компьютерном моделировании в качестве ограничения выступает шаг моделирования по времени – при его уменьшении будут увеличиваться требования к вычислительным ресурсам компьютера. Для увеличения точности моделирования сигнала ШИМ его несущая частота принимается равной 100 Гц, а шаг по времени моделирования 0,0001 с.

$\nu = 100;$

$$K_{ШИМ} = V / U = 10 / 2000 = 0{,}005,$$

где $V_н$ – напряжение при котором насос-дозатор выдаёт максимальный расход топлива, *В*; U – максимально допустимая частота вращения ротора электродвигателя насоса, *об/с*, ν – опорная частота ШИМ, *Гц*.

ШИМ преобразует входной сигнал в соответствующий ему набор импульсов, при этом, чем больше значение входного сигнала, тем больше заполненность сигнала по широте импульсов. Пример работы ШИМ приведён на рисунке 1.

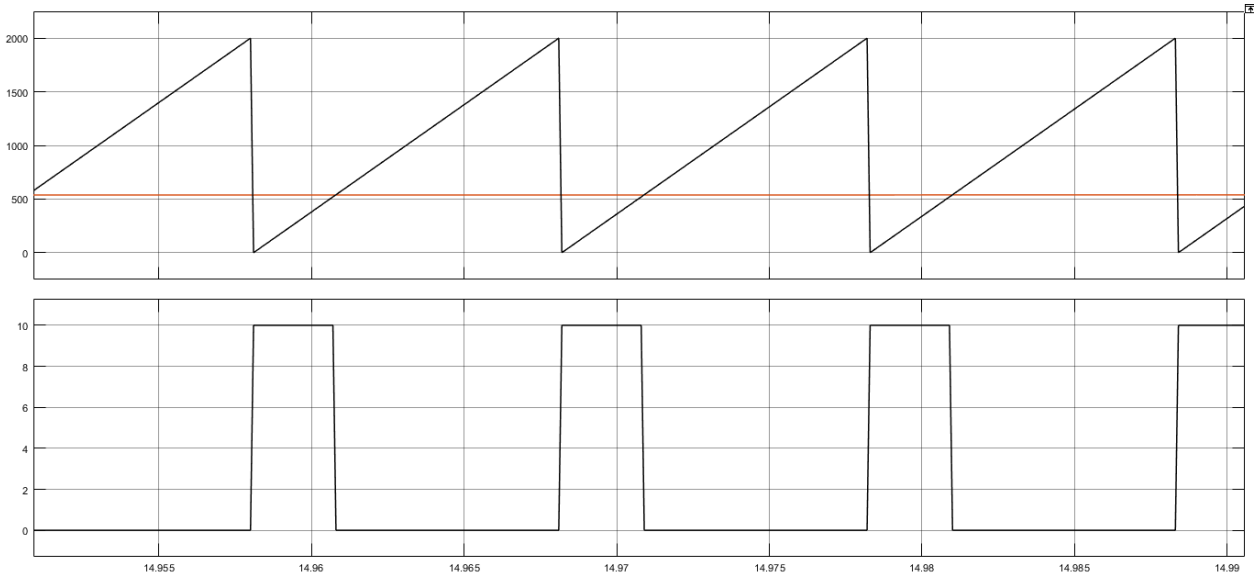


Рисунок 1 –Преобразование сигнала с помощью ШИМ

Во время работы алгоритма генерируется пилообразный сигнал (рисунок 1, сверху, чёрный) заданной частоты (100 Гц). Этот сигнал сравнивается с сигналом, поступающим на вход в блок ШИМ от регулятора (красный): если сигнал от регулятора больше, чем сгенерированный в алгоритме, то на

выход подаётся логическая 1, которая соответствует напряжению питания (10 В), иначе – 0 (рисунок 1, снизу).

Для синтеза ПИД-регулятора нам необходимо получить передаточную функцию всего объекта управления целиком:

$$W_o(s) = K_{ШИМ} \cdot W_н(s) \cdot W_o(s) =$$

$$= 0{,}005 \cdot \frac{0{,}002}{0{,}2s+1} \cdot \frac{157777}{1{,}45s+1} \approx \frac{1{,}578}{0{,}29s^2+1{,}65s+1}.$$

Общий вид передаточной функции второго порядка[1, 2]:

$$W(s) = \frac{K}{T^2 s^2 + 2T\xi s + 1}.$$

Таким образом, постоянная времени полученной передаточной функции $T = \sqrt{0{,}29} = 0{,}5385$.

1.2. Синтез ПИД-регулятора

Описание структуры ПИД-регулятора

Классический ПИД-регулятор [3] состоит из трёх основных элементов – пропорциональной, интегральной и дифференциальной составляющих с соответствующими коэффициентами, участвующими в его настройке K, T_i, T_d:

$$U = K\left(e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right).$$

Непрерывные переменные удобно использовать для анализа и синтеза ПИД-регуляторов. Для технического воплощения необходимо перейти к дискретной форме уравнений, поскольку основой всех регуляторов является микроконтроллер, контроллер или компьютер, которые оперируют с переменными, полученными из аналоговых сигналов после их дискретизации по времени и квантования по уровню.

Переход к дискретным переменным в уравнениях аналогового регулятора выполняется путем замены производных и интегралов их дискретными

аналогами.Существует множество способов аппроксимации производных и интегралов их дискретными аналогами, которые изложены в курсах численных методов решения дифференциальных уравнений. В ПИД-регуляторах наиболее распространенными являются простейшая аппроксимация производной конечной разностью и интеграла – конечной суммой.Рассмотрим интегральный член ПИД-регулятора [3]:

$$I(t) = \frac{K}{T_i}\int_0^t e(t)dt.$$

Продифференцировав обе части по времени, получим:

$$\frac{dI(t)}{dt} = \frac{K}{T_i}e(t).$$

Заменяя дифференциалы в этом выражении конечными разностями (правыми разностями), получим:

$$\frac{I_i - I_{i-1}}{\Delta t} = \frac{K}{T_i}e_i,$$

где индекс *i*обозначает, что данная величина взята в момент времениt_i(обратим внимание, что здесь и ниже*индексі*вT_i*обозначает не номер временного шага, а интегральный коэффициент*ПИД-регулятора)[3]. Из последнего выражения получим:

$$I_i = I_{i-1} + \frac{K \cdot \Delta t}{T_i}e_i.$$

По аналогии получим дифференциальный элемент ПИД-регулятора в конечных разностях:

$$D(t) = KT_d\frac{de(t)}{dt};$$

$$D_i = \frac{KT_d}{\Delta t}\left(e_i - e_{i-1}\right).$$

Определение параметров ПИД-регулятора

Определим начальные параметры ПИД-регулятора $К_p$ (пропорциональный коэффициент), T_i (постоянная интегрирования), T_d (постоянная дифференцирования) методом Зиглера-Никольса с транспортным запаздыванием по формулам:

$$K_p = \frac{1{,}2T}{K\tau} = \frac{1{,}2 \cdot 0{,}5385}{1{,}578 \cdot 0{,}04} = 10{,}24;$$

$$T_i = 2 \cdot \tau = 2 \cdot 0{,}04 = 0{,}08;$$

$$T_d = 2 \cdot \tau = 0{,}5 \cdot 0{,}04 = 0{,}02.$$

2. МОДЕЛИРОВАНИЕ ОБЪЕКТА УПРАВЛЕНИЯВ MATLAB

Использованные при моделировании блоки стандартной библиотеки Simulink приведены в ПРИЛОЖЕНИИ 1.Общий вид модели приведён на рисунке 2. Блоки PID, FuzzyPID и ШИМ представляют собой подсистемы.

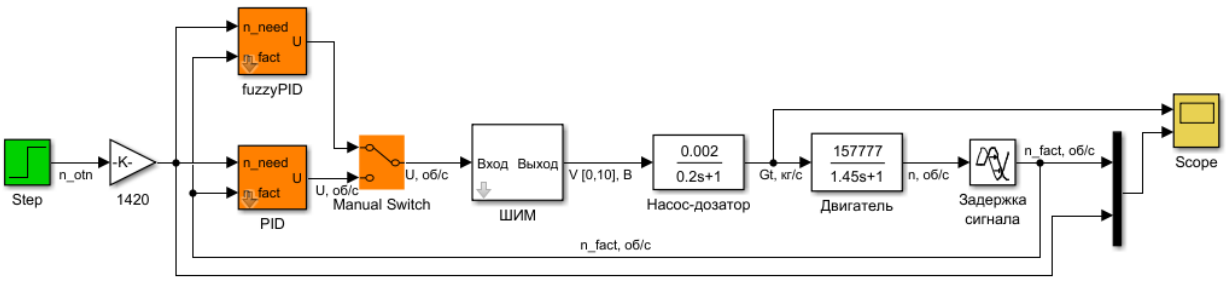


Рисунок 2 – Модель объекта управления с вариациями системы управления

Реализация полученной модели объекта управленияв MATLAB/Simulinkприведена на рисунке 3. Насос-дозатор и двигатель представлены блоками TranferFcn, а задержка – TransportDelay.

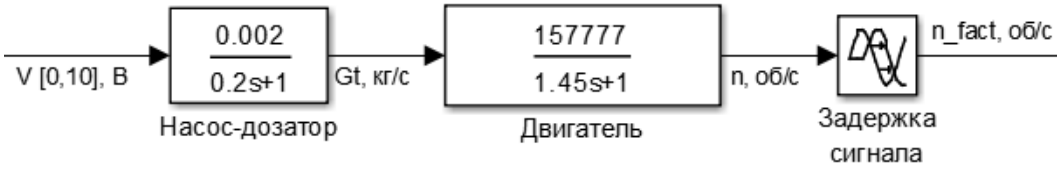


Рисунок 3 – Модель объекта управления

Моделирование ШИМ

Модель блока ШИМпредставляет собой подсистему с внутренней структурой, показанной на рисунке 4. Она работает следующим образом: входной постоянный сигнал «Constant2» преобразуется интегратором в линейно-изменяющийся. По достижении выходным сигналом значения равного «Constant3», блок «RelationalOperator» вырабатывает логический сигнал, по которому происходит сброс выходного сигнала интегратора до начального значения равного нулю. В результате на выходе интегратора формируется пилообразный сигнал, изменяющийся от 0 до Xmax = 2000 с частотой freq = 100 *Гц*. Этот сигнал вычитается из непрерывного входного сигнала. Если значение отрицательно (входной сигнал больше пилообразного), то на выход блока ШИМ идёт значение Umax = 10, иначе – 0.

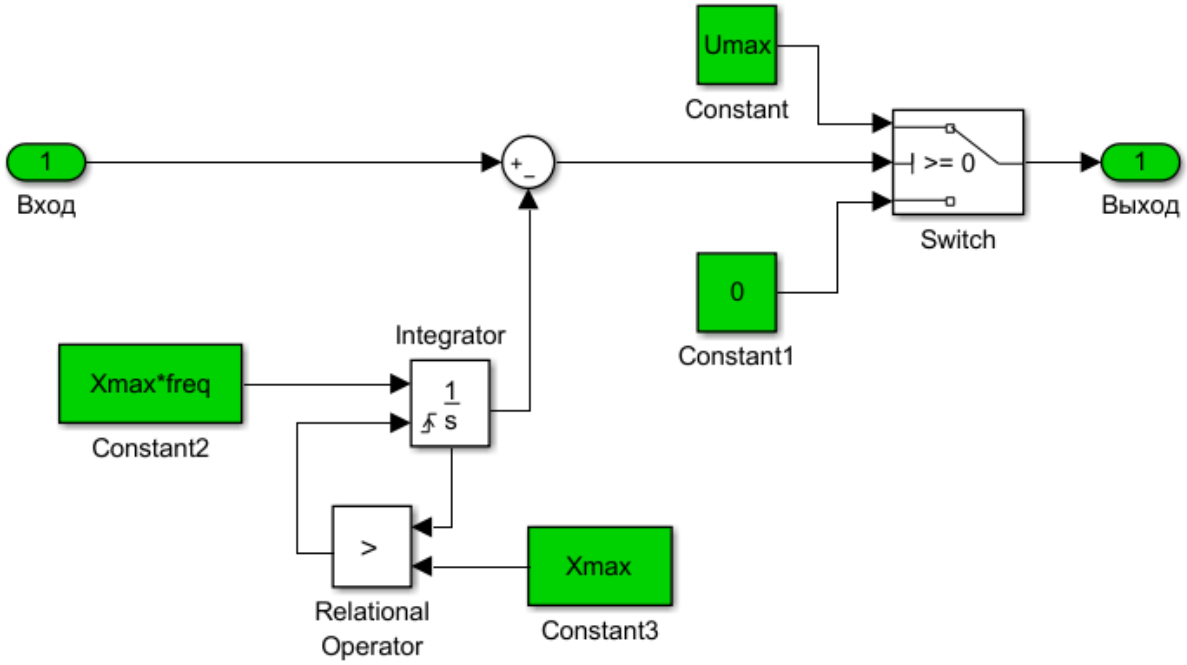


Рисунок 4 – Внутренняя структура блока ШИМ

Для подсистем есть возможность накладывать «маску», в которой могут быть прописаны все необходимые константы. При этом будет осуществлёна возможность удобного редактирования этих констант, вместо того, чтобы искать все вхождения этой переменной. Для этого на подсистеме нужно нажать правой кнопкой мыши и во всплывающем меню выбрать Mask ->CreateMask (либо, если она уже существует EditMask) (рисунок 5).

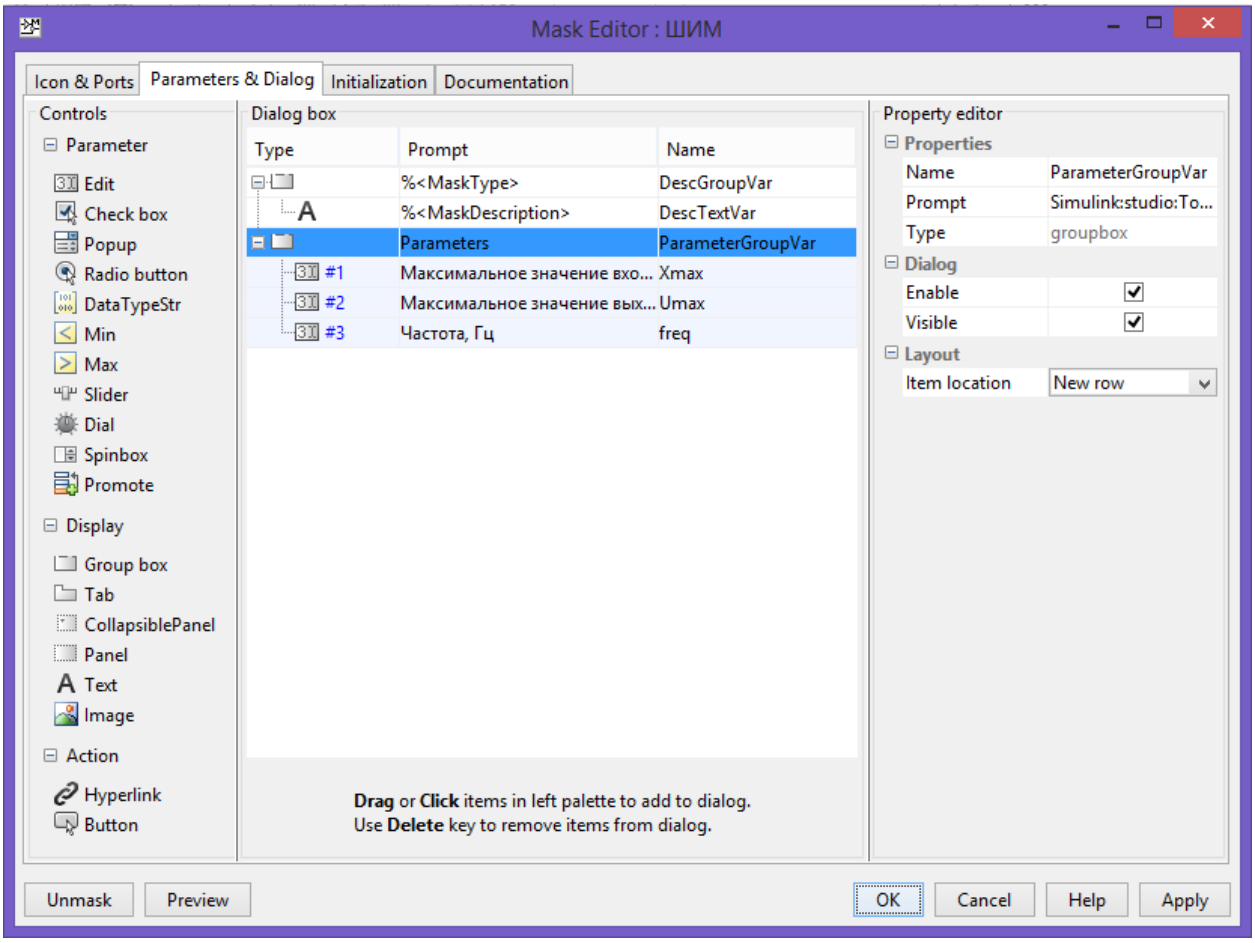


Рисунок 5 – Окно редактора «маски»

В открывшимся окне (рисунок 5) во вкладке Parameters&Dialogв левой колонке с названием Controlsнаходятся элементы, выносимые в «маску». Для наших целей достаточно будет поля редактирования Edit (добавляется в список параметров щелчком левой кнопки мыши). В центральной части окна в колонке Promptуказывается название параметра или его описание, а в колонке Name– название переменной, которая будет использоваться в подсистеме.

Значение переменной можно редактировать либо сразу (правая колонка, при нажатии на определённый элемент появляется поле для более тонкого редактирования), либо ввести нужное значение при открытии «маски» после окончания редактирования (рисунок 6).

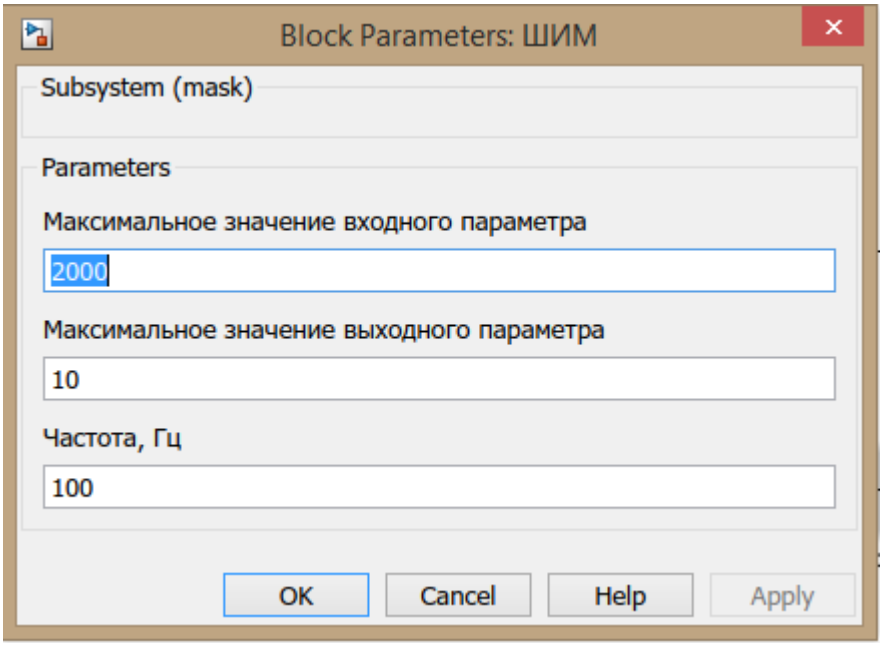


Рисунок 6 – Блок параметров «маски»

3. СОЗДАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ С PID-РЕГУЛЯТОРОМ

Математическая модель регулятора

Полученная в MATLAB/Simulinkструктура ПИД-регулятора приведена на рисунке 7.

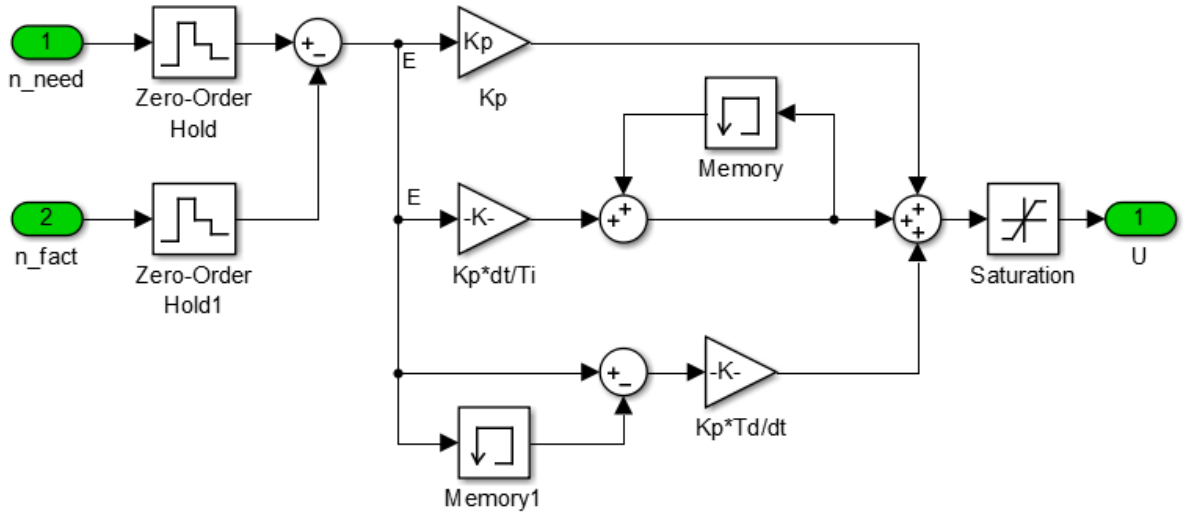


Рисунок 7 – Внутренняя структура ПИД-регулятора

В модели присутствуют вспомогательные элементы. Zero-OrderHold– блок, отвечающий за дискретизацию входных параметров с требуемой частотой. Физически это можно представить как интервал времени, с которым система управления опрашивает датчики. Saturation задаёт ограничения по выходному параметру. Следует заметить, что частота дискретизации в настройках Simulink,по которой программа рассчитывает дифференциальные уравнения,отличается от той частоты дискретизации, которую мы задаём при расчёте наших уравнений в конечных разностях. Чтобы использовать нужную частоту необходимо в блоках Memory, которые отвечают за единичную задержку на время, равное частоте дискретизации системы, поставить галочку на «Inheritsampletime».

Верхней границей выходной величины регулятора будет U_{max} = 2000, а нижней – U_{min} = 200 *об/с*, которые соответствуют максимально допустимой физической частоте вращения ротора двигателя и частоте вращения малого газа, соответственно. Период дискретизации Δt для нахождения производной и интеграла примем равным периоду дискретизации всей нашей системы (задаётся в блоках Zero-OrderHold) и равному 0,1 с. Период опроса датчиков за-

висит от скорости переходных процессов в объекте – чем он меньше, тем чаще надо опрашивать датчики.

Настройка полученной модели

График полученного переходного процесса приведён на рисунке 8.

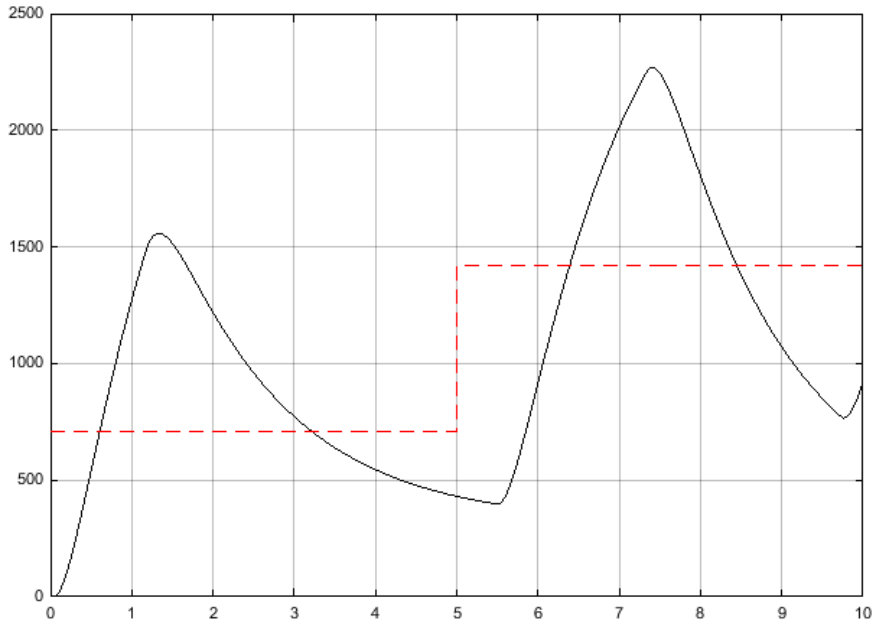


Рисунок 8 – Переходный процесс по частоте вращения ротора двигателя

На рисунке пунктиром показана величина, которую мы задаём и к которой регулятор должен привести объект управления. Как видно из рисунка, процесс далёк от приемлемого. Попробуем разобраться в причинах такого поведения нашей системы.

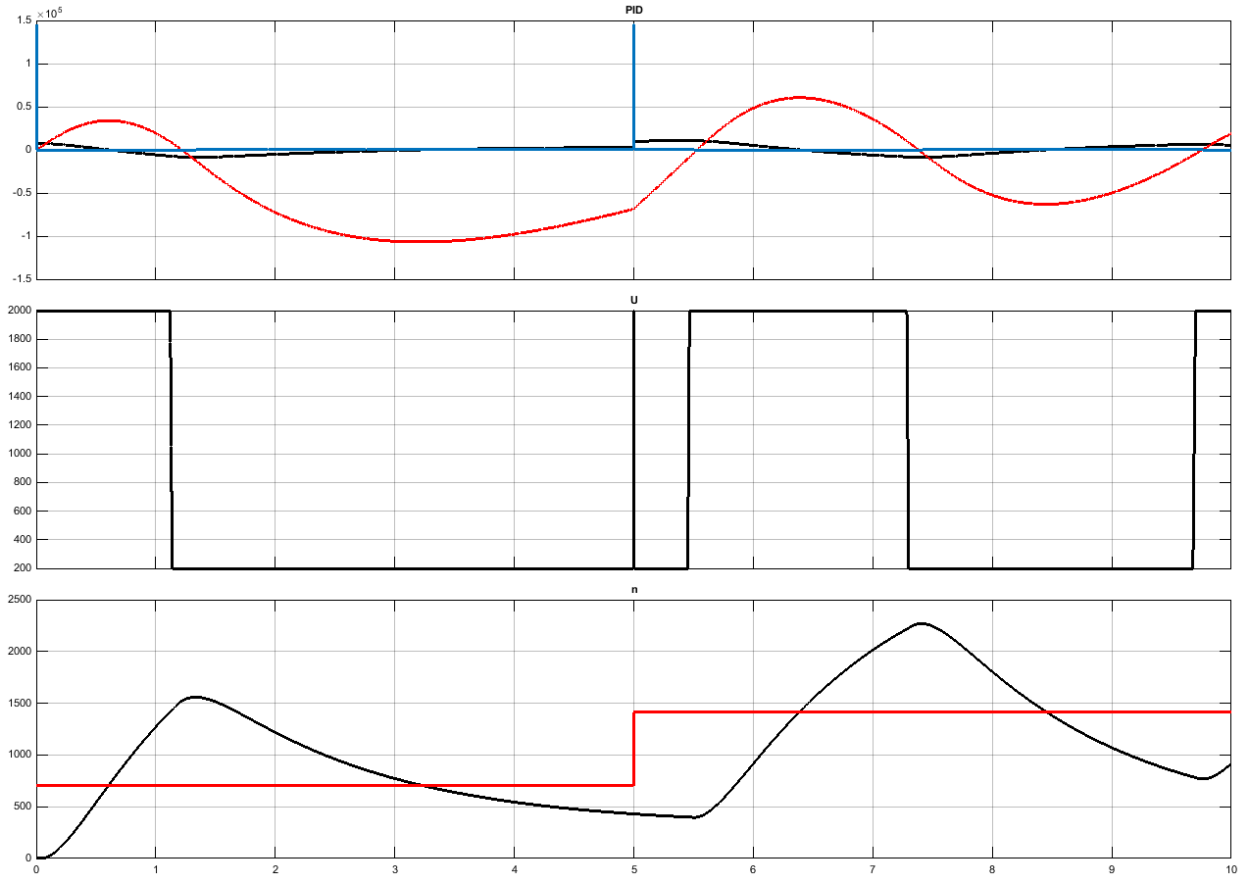


Рисунок 9 – Отклик системы на возмущения

На рисунке 9 на верхнем графике приведены переходные процессы для каждого из элементов ПИД-регулятора: чёрный – пропорциональная составляющая, красный – интегральная, синий – дифференциальная. На среднем графике приведено значение выходной величины регулятора с учётом ограничений на выходе. На нижнем графике приведена частота вращения ротора двигателя. Исходя из среднего рисунка видно, что основную часть выходного сигнала ПИД-регулятора составляет интегральная составляющая.

На установившемся режиме работы и при малых возмущениях большинство систем с ПИД-регуляторами являются линейными. Однако процесс выхода на режим практически всегда требует учета нелинейности типа "ограничение". Эта нелинейность связана с естественными ограничениями на мощность, скорость, частоту вращения, угол поворота, площадь поперечного сечения клапана, динамический диапазон, и т. п. Контур регулирования в системе, находящейся в насыщении (когда переменная достигла ограниче-

ния), оказывается разомкнутым, поскольку при изменении переменной на входе звена с ограничением его выходная переменная остается без изменений.

Наиболее типовым проявлением режима ограничения является так называемое "интегральное насыщение", которое возникает в процессе выхода системы на режим в регуляторах с ненулевым коэффициентом перед интегратором. Интегральное насыщение приводит к затягиванию переходного процесса, как это видно из рисунков 8 и 9.

Суть проблемы интегрального насыщения состоит в том, что если сигнал на входе объекта управления Uвошел в зону насыщения (ограничения), а сигнал рассогласования $e(t) \neq 0$, интегратор продолжает интегрировать, т.е. сигнал на его выходе растет, но этот сигнал не участвует в процессе регулирования и не воздействует на объект вследствие эффекта насыщения. Система управления в этом случае становится эквивалентной разомкнутой системе, сигнал на входе которой равен уровню насыщения управляющего сигналаU[3]. Для двигателя таким ограничением сверху выступает U_{max} – максимально допустимая частота вращения ротора, а ограничением снизу – U_{min} – минимальная частота вращения ротора двигателя, при которой происходит устойчивая работа двигателя.

Одним из способов компенсации интегрального насыщения является дополнительная обратная связь, которая отслеживает состояние исполнительного устройства, входящего в насыщение, и компенсирует сигнал, подаваемый на вход интегратора. Структура системы с таким компенсатором показана на рисунке 10.

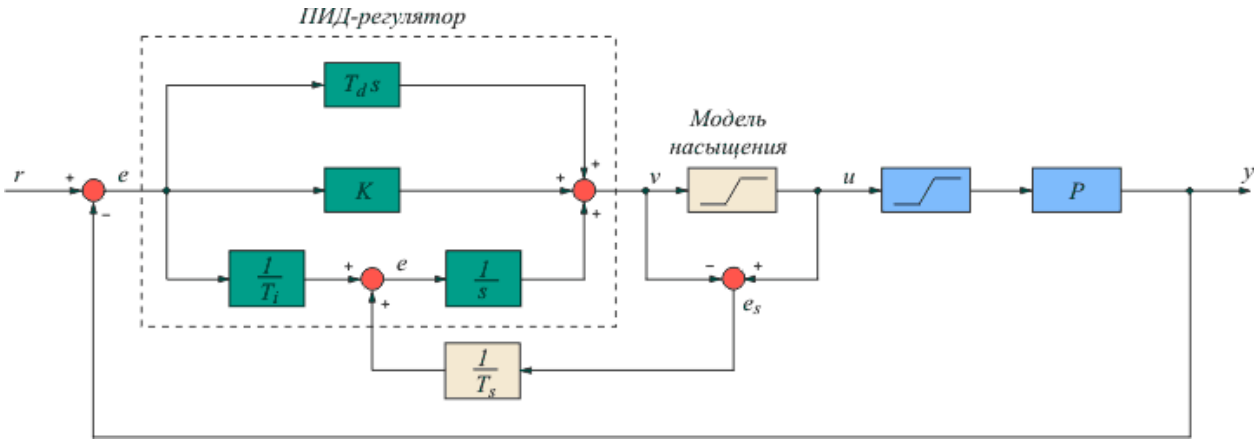


Рисунок 10 – Компенсация интегрального насыщения [3]

Принцип ее работы состоит в следующем. В системе вырабатывается сигнал рассогласования между входом и выходом исполнительного устройства $e_s = u - v$. Сигнал на выходе исполнительного устройства либо измеряют, либо вычисляют, используя математическую модель. Если $e_s = 0$, это эквивалентно отсутствию компенсатора и получаем обычный ПИД-регулятор. Если же исполнительное устройство входит в насыщение, то $v > u$ и $e_s < 0$. При этом сигнал на входе интегратора уменьшается на величину ошибки e_s, что приводит к замедлению роста сигнала на выходе интегратора, уменьшению сигнала рассогласования и величины выброса на переходной характеристике системы. Постоянная времени T_s определяет степень компенсации сигнала рассогласования.Обычно эту величину принимают равной T_i.

В некоторых регуляторах вход u устройства сравнения e_s выделяют как отдельный вход, называемый "вход слежения", что бывает удобно при построении сложных систем управления и при каскадном соединении нескольких регуляторов.

Новый вид интегральной составляющей, выделенной для удобства в отдельную подсистему, представлен на рисунке 11, а переходный процесс модифицированной системы – на рисунке 12.

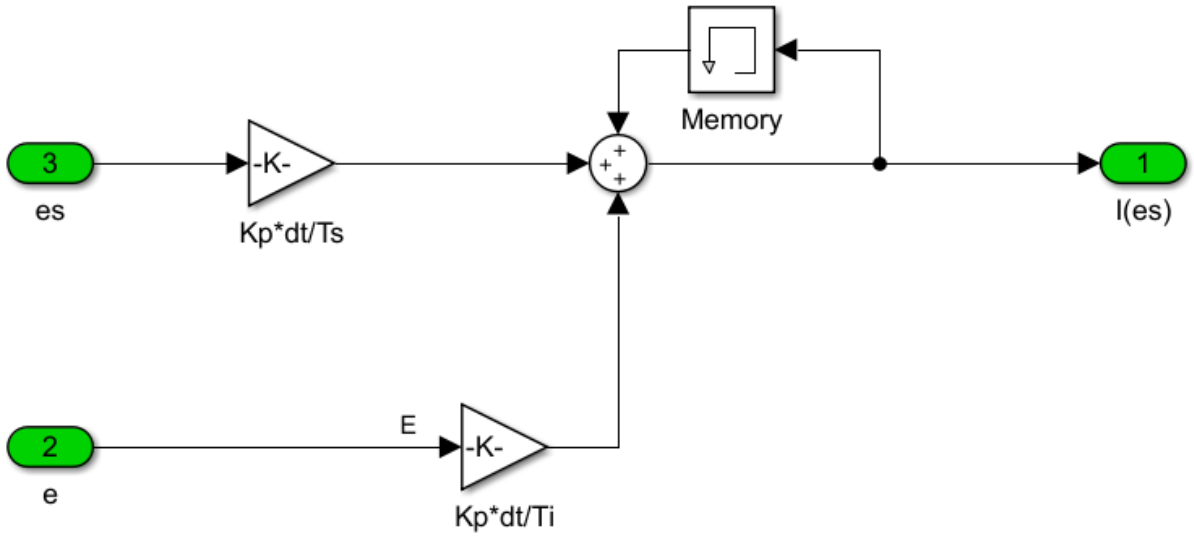


Рисунок 11 – Коррекция интегральной составляющей

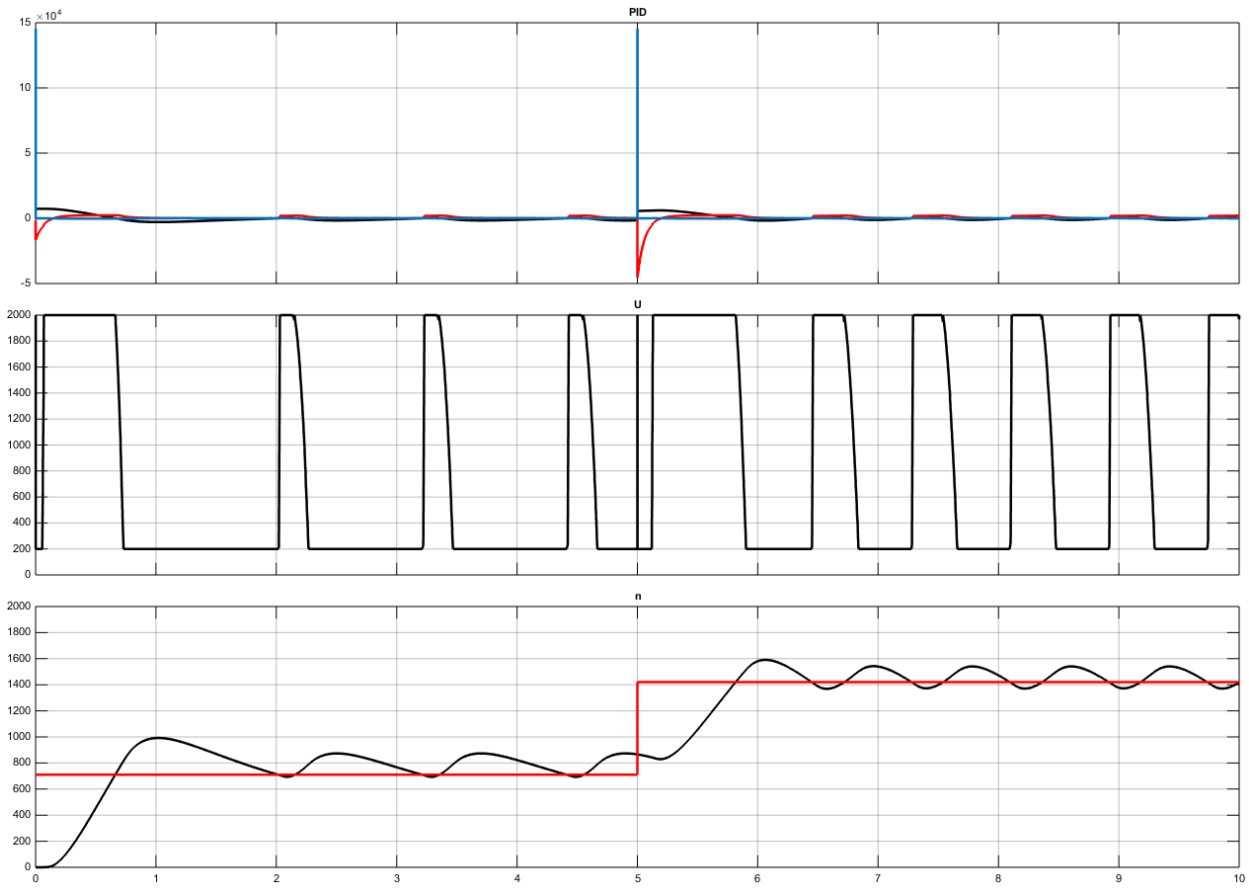


Рисунок 12 – Переходный процесс без интегрального насыщения

После избавления от интегрального насыщения переходный процесс стал гораздо ближе к необходимому. Однако, из верхнего графика видно, что у нас происходит падение интегральной составляющей при каждомступенча-

том управляющем воздействии. Это вызвано скачкообразным ростом дифференциальной составляющей, а вместе с ней и общей суммой всех трёх слагаемых. При этом полученная сумма превышает ограничивающее значение и начинает вычитаться из выхода ограничителя. Таким образом, получившееся отрицательное значение заводится в модифицированную интегральную составляющую, что вызывает её резкое падение. Есть несколько способов избавиться от этого эффекта – ввести передаточную функцию, вместо коэффициента усиления, в линию обратной связи от ограничителя к интегратору, либо просто ограничить дифференциальную составляющую. Недостатком первого способа является дополнительная задача корректировки коэффициента передаточной функции и возрастающая неустойчивость системы. У второго способа заметных недостатков нет, поэтому воспользуемся им. В качестве ограничивающих значений используем максимальное значение ограничения на выходе из регулятора. Получаем ограничение на дифференциальной составляющей, равным [U_{max}, $-U_{max}$]. Переходный процесс с модифицированной дифференциальной составляющей представлен на рисунке 13.

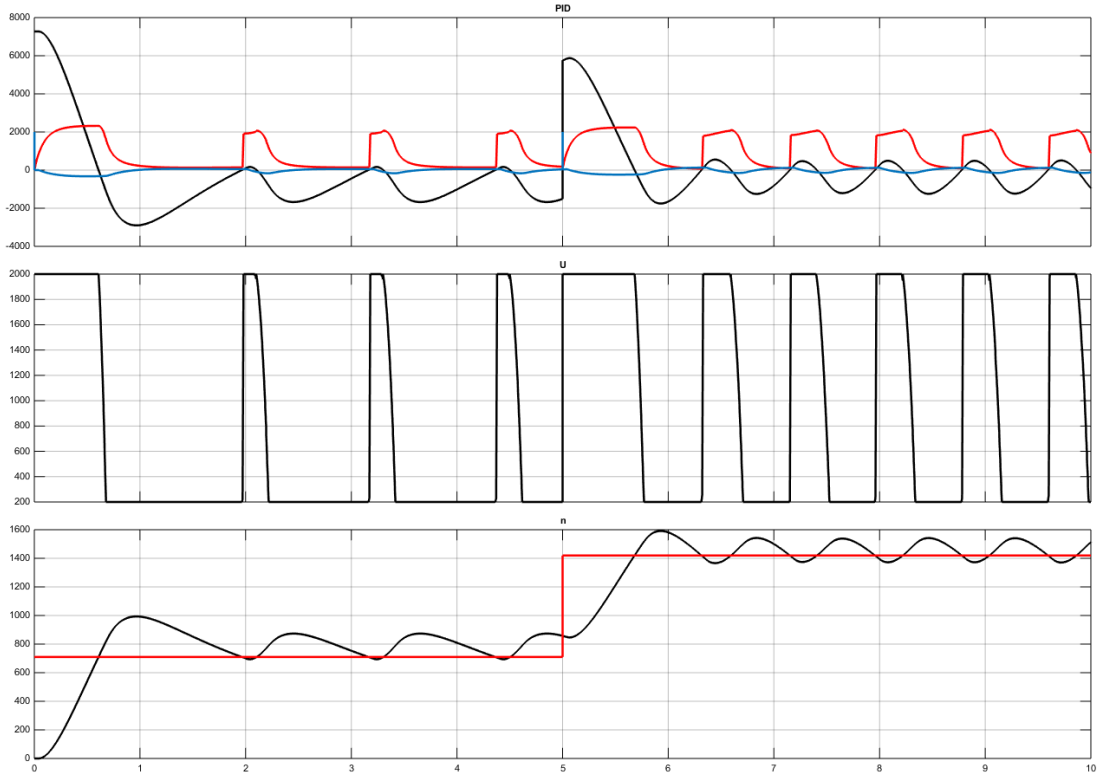


Рисунок 13 – Переходный процесс с модифицированной дифференциальной составляющей

Корректировка коэффициентов PID

Для дальнейшей настройки нужно корректировать коэффициенты ПИД-регулятора из следующих соображений.

а) Увеличение K_p ведёт к ускорению переходного процесса, уменьшению устойчивости и уменьшению статической ошибки.

б) Уменьшение K_p ведёт к замедлению переходного процесса, увеличению устойчивости (уменьшение колебательности) и увеличению статической ошибки.

в) Увеличение K_i (исходя из приведённой структуры ПИД-регулятора $K_i \sim 1/T_i$, значит это правило справедливо для уменьшения T_i) приводит к ускорению выхода объекта не требуемый режим и устраняет статическую ошибку, так же приводит к увеличению колебательности процесса.

г) Уменьшение K_i (увеличение T_i) приводит к увеличению времени выхода на заданный режим (при этом так же устраняет статическую ошибку), но уменьшает его колебательность.

д) Увеличение K_d ($K_d \sim T_d$, значит справедливо для увеличения T_d) приводит к повышению устойчивости процесса и его быстродействию, но при этом регулятор становится восприимчивым к высокочастотным шумам.

е) Уменьшение K_d (уменьшение T_d) приводит к уменьшению устойчивости и быстродействияпроцесса, но при этом высокочастотные помехи оказывают меньшее влияние на процесс управления.

При ручной настройке коэффициентов ПИД-регулятора настраивать следует последовательно каждый коэффициент по следующей схеме:

1) Начинать следует с K_p. После предварительной настройки по методу Зиглера-Никольса получаем устойчивый колебательный процесс, следовательно, этот коэффициент больше не изменяем. Если же процесс получился колебательным и расходящимся или апериодическим, то в первом случае нужно уменьшать K_p, а во втором - увеличивать до получения колебаний постоянной амплитуды и периода.

2) При настройке $K_t(T_i)$ необходимо увеличивать T_i (уменьшается K_i) до тех пор, пока увеличение этогокоэффициента не перестанет оказывать на процесс никакого действия. В нашем случае T_i = 1,2 (рисунок 14).

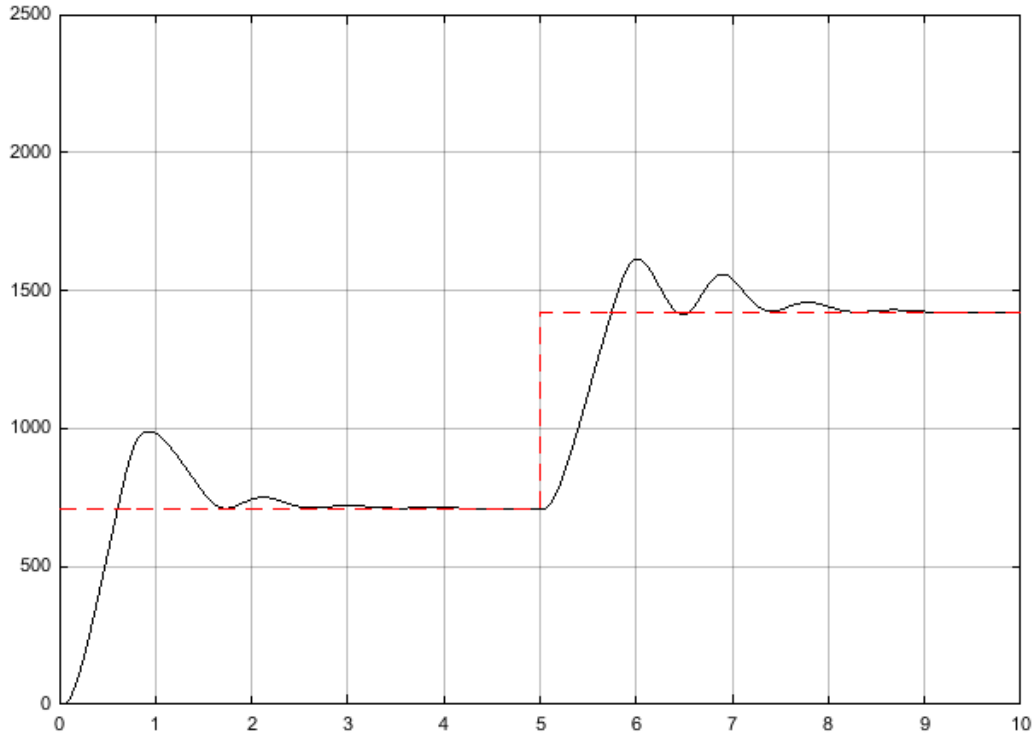


Рисунок 14 –Переходный процесс по частоте вращения ротора после настройки интегральной составляющей

3) Остаётся подобрать K_d (T_d) для уменьшения заброса регулируемой величины и получения характеристикинужного вида. Подбор этого коэффициента осуществляется согласно пунктам д) и е) (рисунок 15).

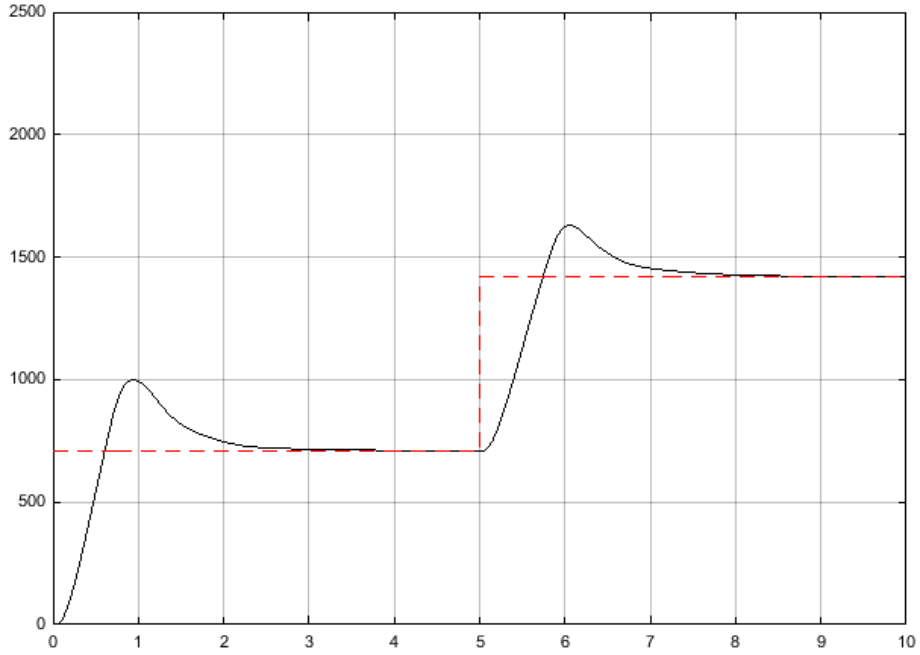


Рисунок 15 – Переходный процесс по частоте вращения ротора после настройки дифференциальной составляющей

4. СОЗДАНИЕ СИСТЕМЫ УПРАВЛЕНИЯ CFUZZY-РЕГУЛЯТОРОМ

4.1Нечеткая логика в регуляторах

Нечеткое управление (управление на основе методов теории нечетких множеств) [4 – 6] используется при недостаточном знании объекта управления, но наличии опыта управления им, в нелинейных системах, идентификация которых слишком трудоемка, а также в случаях, когда по условию задачи необходимо использовать знания эксперта. Примером может быть доменная печь или ректификационная колонна, для которых впервые был создан нечёткий регулятор, математическая модель которых содержит много эмпирических коэффициентов, изменяющихся в широком диапазоне и вызывающих большие затруднения при идентификации [4]. В то же время квалифицированный оператор достаточно хорошо управляет такими объектами, пользуясь показаниями приборов и накопленным опытом.

ПИД-регуляторы с нечеткой логикой в настоящее время используются в коммерческих системах для наведения телекамер при трансляции спортивных событий, в системах кондиционирования воздуха, при управлении авто-

мобильными двигателями. В качестве примера применения нечёткой логики в управлении авиационными двигателями можно привести двигатели JetCat, в которых в последнее время применяются блоки управления, основанные на нечёткой логике.

Поскольку информация, полученная от оператора, выражена словесно, для ее использования в ПИД-регуляторах применяют лингвистические переменные и аппарат теории нечетких множеств, который был разработан Л. Заде в 1965 году [7]. Основная идея этой теории состоит в следующем. Если в теории четких множеств некоторый элемент (например, температура 50 град.) может принадлежать множеству (например, множеству "температура горячей воды$T_{гор}$) или не принадлежать ему, то в теории нечетких множеств вводится понятиефункции принадлежности, которая характеризует степень принадлежности элемента множеству. При этом говорят, например, температура 50 град. принадлежит множеству $T_{гор}$со степенью принадлежности 0,264. Функцию принадлежности можно приближенно трактовать как вероятность того, что данный элемент принадлежит множеству [8], однако такая интерпретация, хотя и является для инженеров более понятной, не является математически строгой, поскольку существующая теория нечетких множеств не оперирует понятием вероятности.

В 1974 году Мамдани [9] показал возможность применения идей нечеткой логики для построения системы управления динамическим объектом, а годом позже вышла публикация [10], в которой описывался нечеткий ПИ-регулятор и его применения для управления парогенератором. С тех пор область применения нечетких регуляторов постоянно расширяется, увеличивается разнообразие их структур и выполняемых функций.

Нечеткая логика в ПИД-регуляторах используется преимущественно двумя путями: для построения самого регулятора и для организации подстройки коэффициентов ПИД-регулятора (рисунки 16, 17). Оба пути могут использоваться в ПИД-контроллере одновременно. Далее мы будем использовать регулятор на основе только нечёткой логики.

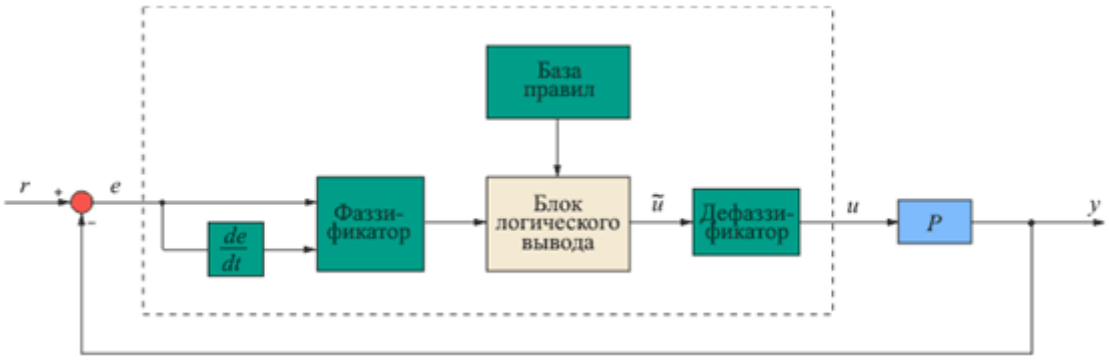


Рисунок 16 – Структура нечеткого регулятора

Принципы построения нечеткого регулятора

Для применения методов нечеткой логики прежде всего необходимо преобразовать обычные четкие переменные в нечеткие. Процесс такого преобразования называетсяфаззификацией(от английского "fuzzy" – "нечеткий"). Он иллюстрируется на рисунке 17.

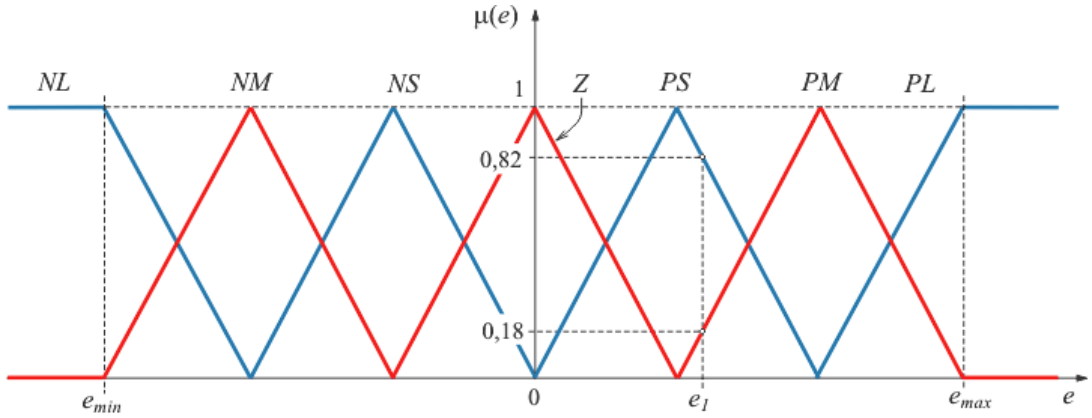


Рисунок 17 – Деление области изменения переменной *e*на множества *NL, NM, NS*и т.д. с функциями принадлежности *μ(e)* треугольной формы

Диапазон изменения переменной*e*разбивается на множества (подмножества)*NL, NM, NS, Z, PS, PM, PL,*в пределах каждого из которых строится функция принадлежности переменной*e*каждому из множеств. Нарисунке 17 функции принадлежности имеют треугольную (наиболее распространенную) форму, хотя в общем случае они могут быть любыми, исходя из смысла решаемой задачи [6]. Количество множеств также может быть произвольным.

Для нечетких множеств существует общепринятая система обозначений: *N* – отрицательный (*Negative*); *Z* – нулевой (*Zero*); *P* – положительный (*Positive*); к этим обозначениям добавляют буквы *S* (малый, *Small*), *M* (средний, *Medium*), *L* (большой, *Large*). Например, *NL* - отрицательный большой; *NM* – отрицательный средний (*NegativeMedium*); *PL* - положительный большой. Количество таких переменных (термов) может быть любым, однако с увеличением их количества существенно возрастают требования к опыту эксперта, который должен сформулировать правила для всех комбинаций входных переменных.

Если величина ошибки *e*на входе нечеткого регулятора (рисунок 16) равна e_1(рисунок 17), то соответствующее значение нечеткой переменной будет равно *PS*со степенью принадлежности подмножеству*PS*, равной $\mu(e_1) = 0{,}82$, или равно *PM*со степенью принадлежности$\mu(e_1) = 0{,}18$. Степень принадлежности ошибкиe_1 другим множествам (*Z, PL, NS*и др.) равна нулю. Таким образом, величина ошибкиe_1оказалась преобразованной в нечеткие переменные.

Для выполнения функции регулирования над нечеткими переменными должны быть выполнены операции, построенные на основании высказываний оператора,сформулированных в виде нечетких правил. Совокупность нечетких правил и нечетких переменных используется для осуществления нечеткого логического вывода, результатом которого является управляющее воздействие на объект управления (рисунок 16).

Нечеткий вывод выполняется следующим образом. Предположим, что область изменения ошибки *e* разделена на множества*N, Z, P*, область изменения управляющего воздействия – на множества *NL, NM, Z, PM, PL* и что с помощью эксперта удалось сформулировать следующие правила работы регулятора [11]:

Правило 1: если $e = N$ и $de/dt = P$, то $\tilde{u} = Z$

Правило 2: если $e = N$ и $de/dt = Z$, то $\tilde{u} = NM$

Правило 3: если $e = N$ и $de/dt = N$, то $\tilde{u} = NL$

Правило 4: если $e = Z$ и $de/dt = P$, то $\tilde{u} = PM$

Правило 5: если $e = Z$ и $de/dt = Z$, то $\tilde{u} = Z$

Правило 6: если $e = Z$ и $de/dt = N$, то $\tilde{u} = NM$

Правило 7: если $e = P$ и $de/dt = P$, то $\tilde{u} = PL$

Правило 8: если $e = P$ и $de/dt = Z$, то $\tilde{u} = PM$

Правило 9: если $e = P$ и $de/dt = N$, то $\tilde{u} = Z$.

Для вычисления результатов действия правил необходимо использовать логические операции пересечения (И), которая показывает выполнение правил одновременно, и объединения (ИЛИ), которая показывает выполнение хотя бы одного из правил. Таким образом, вычисляя результат каждого из правил, а затем их совокупный результат, мы получаем результат нечёткого вывода по всем правилам.

Для нахождения результата пересечения множеств применяется функция принадлежности, например, N и P (см. Правило 1) находится как [6]:

$$\mu_{e \cap de/dt} = \min(\mu_e, \mu_{de/dt}),$$

т.е. каждое значение функции принадлежности пересечения множеств равно наименьшему значению из двух, стоящих в круглых скобках.

Функция принадлежности для объединения (логическая операция ИЛИ) тех же множеств имеет вид [6]:

$$\mu_{e \cup de/dt} = \max(\mu_e, \mu_{de/dt}).$$

Функции принадлежности, полученные при пересечении или объединении множеств, могут быть определены различными способами, в зависимости от смысла решаемой задачи. В [6] приводится 10 различных определений функции принадлежности для пересечения множеств, но не сказано, какое из них нужно выбрать для решения конкретной задачи. Используют, в частности, более понятную операцию нахождения функций принадлежности в случае пересечения и объединения множеств, имеющую аналогию с правилами умножения и сложения вероятностей:

$$\mu_{e\cap de/dt} = \mu_e \cdot \mu_{de/dt}$$

$$\mu_{e\cup de/dt} = \mu_e + \mu_{de/dt} - \mu_e \cdot \mu_{de/dt}.$$

Однако применение первых двух способов нахождения функции принадлежности обычно более предпочтительно, т.к. при этом сохраняется большинство правил, разработанных для обычных множеств [5].

Функции принадлежности для каждого из множеств *NL, NM, Z, PM, PL,* входящих в нечеткую переменную $\tilde{u}$ в правилах, получаются в виде [6]:

$$\mu_{П1}(\tilde{u}) = \min\{\mu_{u1}(\tilde{u}), \min(\mu_{e1}(e), \mu_{de/dt1}(de/dt))\},$$

$$\mu_{П2}(\tilde{u}) = \min\{\mu_{u2}(\tilde{u}), \min(\mu_{e2}(e), \mu_{de/dt2}(de/dt))\},$$

…

$$\mu_{П9}(\tilde{u}) = \min\{\mu_{u9}(\tilde{u}), \min(\mu_{e9}(e), \mu_{de/dt9}(de/dt))\}.$$

Здесь каждое из 9-ти уравнений соответствует одному из правил. Результирующая функция принадлежности управляющего воздействия $\tilde{u}$, полученная после применения всех 9-ти правил, находится как объединение функций принадлежности всех правил:

$$\mu(\tilde{u}) = \max\{\mu_{П1}(\tilde{u}), \mu_{П2}(\tilde{u}), \ldots, \mu_{П9}(\tilde{u})\}.$$

Теперь, когда получена результирующая функция принадлежности управляющего воздействия u, возникает вопрос, какое конкретно значение управляющего воздействия нужно выбрать. Если использовать вероятностную интерпретацию теории нечетких множеств, то становится понятно, что такое значение можно получить по аналогии с математическим ожиданием управляющего воздействия в виде:

$$u = \frac{\int_{u\min}^{u\max} \tilde{u}\,\mu(\tilde{u})\,d\tilde{u}}{\int_{u\min}^{u\max} \mu(\tilde{u})\,d\tilde{u}}.$$

Такой способ дефаззификации является наиболее распространенным, но не единственным.

Для построения нечетких регуляторов обычно используют П, И, ПИ и ПД ПД+И, ПИ+Д и ПИД-законы регулирования [12]. В качестве входных сигналов для системы нечеткого вывода используют сигнал ошибки, приращение ошибки, квадрат ошибки и интеграл от ошибки [12]. Реализация нечеткого ПИД регулятора вызывает проблемы, поскольку он должен иметь трехмерную таблицу правил в соответствии с тремя слагаемыми в уравнении ПИД-регулятора, которую чрезвычайно сложно заполнить, пользуясь ответами эксперта. Большое количество структур ПИД-подобных нечетких контроллеров можно найти в статье [12].

4.2. Структура регулятора с нечётким логическим выводом

Довольно часто, особенно в нейросетевых и фаззи-регуляторах, используют уравнение ПИД-регулятора в виде зависимости приращения управляющей величины от ошибки регулирования и ее производных (без интегрального члена). Такое представление удобно, когда роль интегратора выполняет внешнее устройство, например, обычный или шаговый двигатель. Угол поворота его оси пропорционален значению управляющего сигнала и времени. В фаззи-регуляторах при формулировке нечетких правил эксперт может сформулировать зависимость управляющей величины от величины производной, но не может - от величины интеграла, поскольку интеграл "запоминает" всю предысторию изменения ошибки, которую человек помнить не может (рисунок 18).

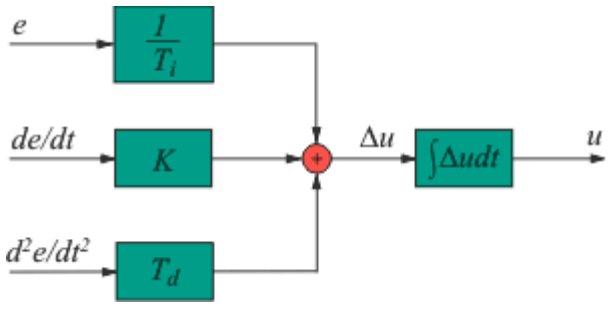


Рисунок 18 – Инкрементная форма ПИД-регулятора [3]

Инкрементная форма регулятора удобна для применения в микроконтроллерах, поскольку в ней основная часть вычислений выполняется с приращениями, для представления которых можно использовать слово с малым количеством двоичных разрядов.

Инкрементный нечёткий регулятор

Нечёткий регулятор (рисунок 19), также как и рассмотренный ранее ПИД-регулятор, работает в дискретной форме. Однако существенным отличием является выбор диапазона изменений рассогласования требуемой и фактической частоты вращения, а так же производной этого рассогласования , величины которых подбирались исходя из переходного процесса, управляемого ПИД-регулятором. В нашем случае диапазон рассогласования *E* был принят [-500;500], а его производной *dE/dt* – [-1000;1000]. Диапазоны заданы в ограничительных блоках Saturation. Введение ограничений по величинам ошибки рассогласования и её производной необходимо для работы регулятора в области корректных значений.

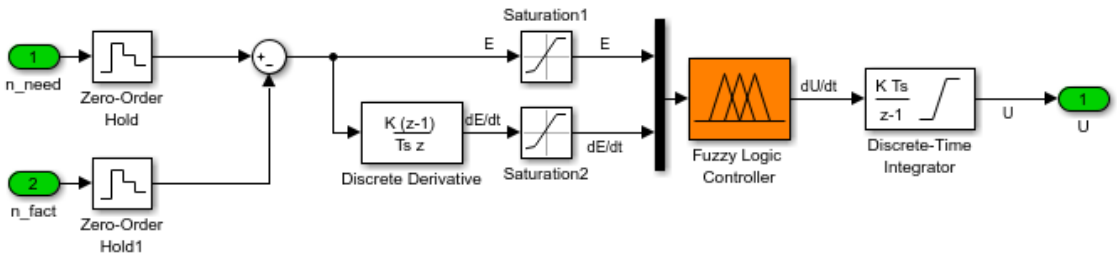


Рисунок 19 – Нечёткий регулятор

Как видно из рисунка 19, на выходе из нечёткого логического контроллера выходит сигнал производной управляющего воздействия, который затем интегрируется в дискретном интеграторе с тем же временем дискретизации, что и в блоках Zero-OrderHold. Диапазон выходных значений *dU/dt* примем равным [-2000; 2000]. Все приведённые выше ограничения будут использоваться при создании нечёткого регулятора, описанного далее. Для работы с блоком Fuzzy Logic Controller (рисунок 20 сверху) необходимо в окне имени модели указать ссылку на файл, в котором был сгенерирован нечёткий

регулятор. На рисунке 21 приведены настройки блока дискретного интегратора.

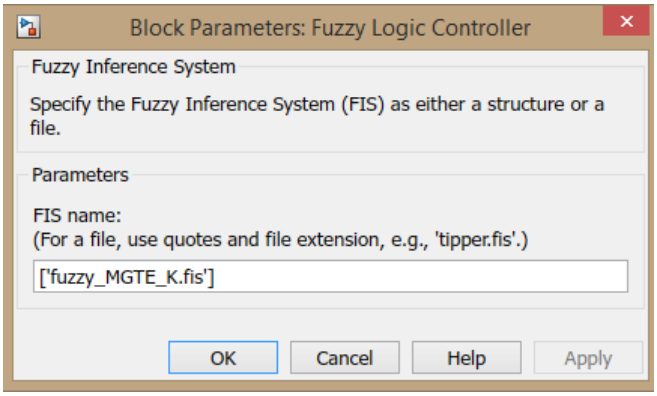


Рисунок 20 – Настройка блока Fuzzy Logic Controller

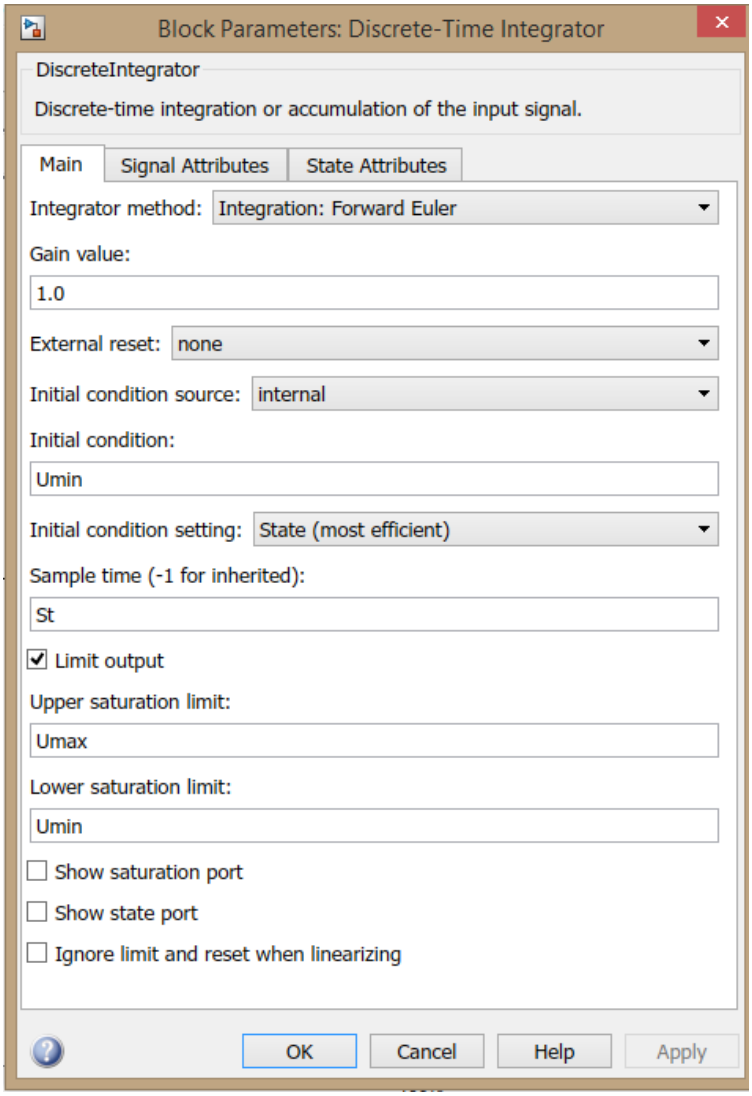


Рисунок 21 – Настройка блока дискретного интегратора

Создание базы правил для инкрементного нечёткого регулятора

Основные принципы построения базы правил этого регулятора:

1. Чем больше величина ошибки и величина её производной, тем выше скорость нарастания управляющего воздействия.

2. Чем ниже величина ошибки и её производной, тем выше скорость убывания управляющего воздействия.

Реализация этих принципов приведена в таблице 1.

Таблица 1 – База правил нечёткого логического вывода

dU/dt			**Величина ошибки, E**						
			← +			**0**	**- →**		
			LP	**MP**	**SP**	**Z**	**SN**	**MN**	**LN**
Производная ошибки, dE/dt	**↑ +**	**LP**	LP	LP	LP	LP	MP	SP	Z
		MP	LP	LP	MP	MP	Z	Z	SN
		SP	LP	MP	MP	MP	Z	SN	MN
	0	**Z**	LP	MP	SP	Z	SN	MN	LN
	- ↓	**SN**	MP	SP	Z	MN	MN	MN	LN
		MN	SP	Z	Z	MN	MN	LN	LN
		LN	Z	SN	MN	LN	LN	LN	LN

Реализация нечёткого логического вывода в MatLab/Simulink

Для использования функций нечёткой логики в MatLab должен быть установлен пакет FuzzyLogicToolbox. Для новичка в нечёткой логике гораздо удобнее будет пользоваться графическим интерфейсом, но, всё же, после краткого знакомства с ним лучше учиться писать напрямую в скриптовых файлах MatLab, так как это сэкономит много времени в процессе настройки регулятора. Приведённые в предыдущем пункте правила уже прошли некоторую отработку, поэтому покажем уже готовый код скрипта (приложение 2). Разберём этот скрипт.

Знаком % в MatLab обозначаются комментарии, т.е. участки кода, которые игнорируются компилятором. Если строка начинается с %, то она вся

игнорируется. Если участок текста находится %{ *между*%} этими символами, то он так же игнорируется.

В приложении 2 перед каждой функцией присутствует участок с комментарием, отражающем синтаксис этой функции. Нечёткие переменные *E, dE/dt, dU*разделены на 7 термов и указаны в порядке от наибольшего отрицательного до наибольшего положительного [LN; MN; SN; Z; SP; MP; LP]. Нумерация термов внутри нечёткой переменной осуществляется сверху вниз в неявном виде по умолчанию.

Правила задаются в виде матрицы, где строкам соответствуют номера правил. По столбцам нумерация осуществляется в следующем порядке: сначала указываются входные переменные (*E, dE*), затем выходные(*dU*), затем столбец с весовыми коэффициентами для каждого правила (в нашем случае везде 1) и в последнем столбце указывается логическое действие между термами (для операции «И» ставится 1, для операции «ИЛИ» - 0).

Для упрощения составления набора правилможно составлять их в Excelили другом аналогичном редакторе в виде таблицы, такой, например, как таблица 1, а затем, с помощью небольшой функции преобразовать строковые значения в соответствующие им численные. Пример такого преобразования в одной из ячеек Excel:

```
=ЕСЛИ(D19="LP";7;ЕСЛИ(D19="MP";6;ЕСЛИ(D19="SP";5;ЕСЛИ(D
19="Z";4;ЕСЛИ(D19="SN";3;ЕСЛИ(D19="MN";2;ЕСЛИ(D19="LN";
1;"ошибка"))))))
```

После этого просто создать ссылки на ячейки в таблице, выполненной в виде матрицы правил MatLab. Таким образом, при изменении правил в исходной таблице останется лишь скопировать их численные эквиваленты в виде базы правил в файл скрипта. Преобразованная база правил в числовом эквиваленте представлена в таблице 2.

Таблица 2 – База правил в числовом эквиваленте

dU/dt	Величина ошибки, E

			← +			0	- →		
			LP	MP	SP	Z	SN	MN	LN
Производная ошибки, dE	↑ +	LP	7	7	7	7	6	5	4
		MP	7	7	6	6	4	4	3
		SP	7	6	6	6	4	3	2
	0	Z	7	6	5	4	3	2	1
	↓ -	SN	6	5	4	2	2	2	1
		MN	5	4	4	2	2	1	1
		LN	4	3	2	1	1	1	1

Заметим так же, что в Excel можно автоматически раскрашивать ячейки. Это доступно по следующему пути (Office 2007): Главная ->Стили ->Условное форматирование. Это позволяет более наглядно представить некоторую поверхность правил.

Работа нечёткого регулятора и его улучшение

Переходный процесс при ступенчатом изменении сигнала уставки, приведён на рисунке 20.

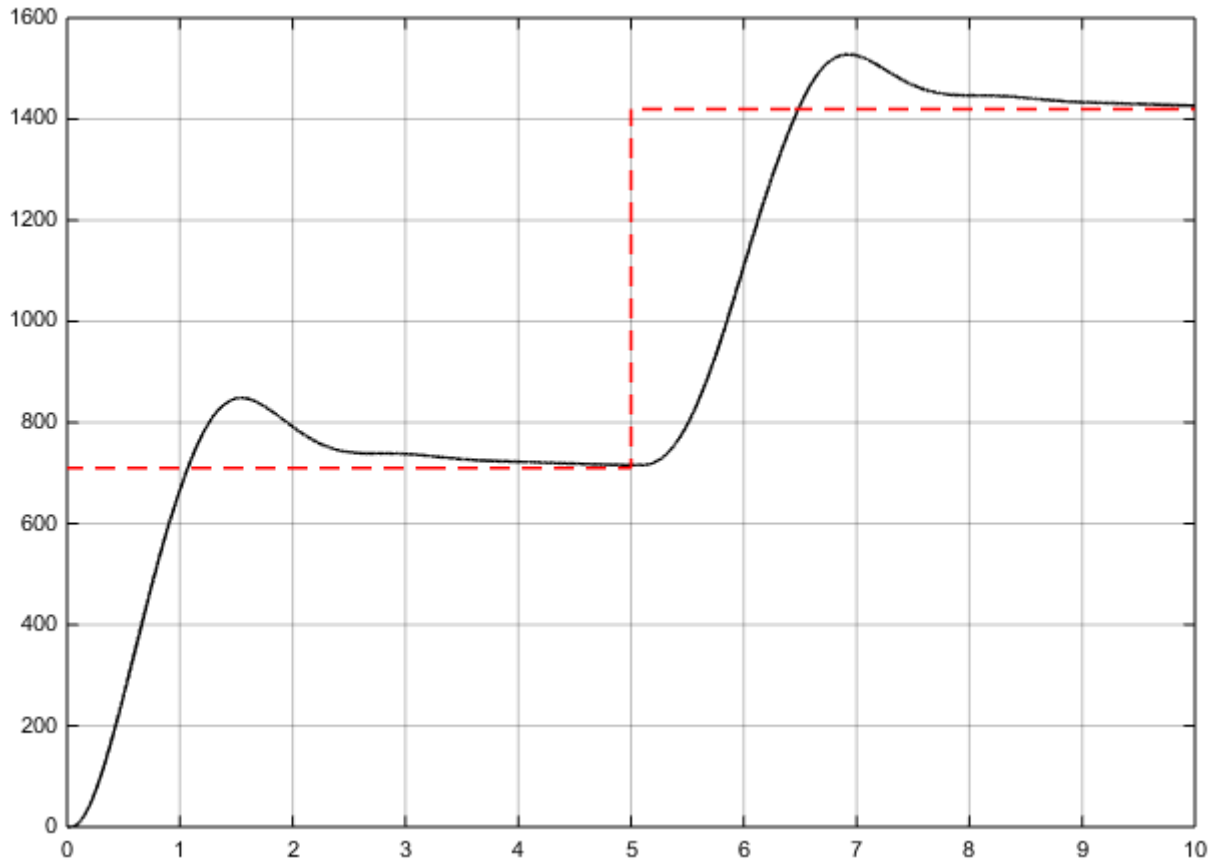


Рисунок 20 – Переходный процесс с нечётким регулятором

Можно немного улучшить его работу, если сделать так, чтобы на околонулевых значениях ошибки и её производной переменная *dU* менялась меньше, чем на высоких. Для этого можно сделать термы выходной переменной *dU*не симметричными, а сам диапазон увеличить, например до [-10000; 10000], однако, такой способ приводит к колебательностипроцесса. В качестве другого способа можно предложить добавить дополнительную нечёткую переменную *K*,которая будет домножатьсяна выходную величину (рисунок 21).

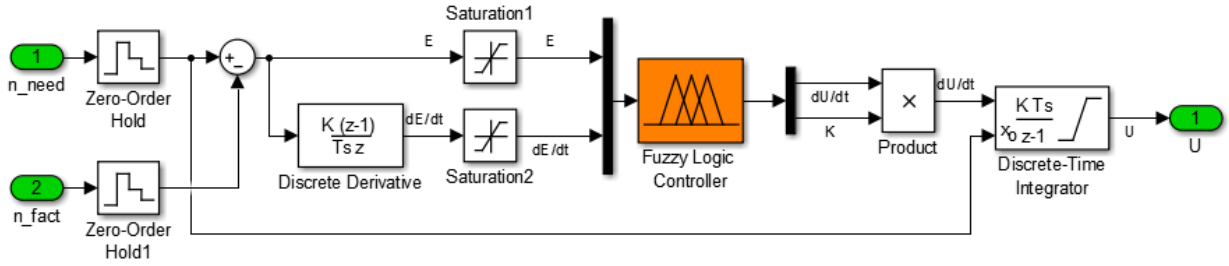


Рисунок 21 – Модифицированный нечёткий регулятор

При этом изменим диапазон первой выходной переменной *dU*на [-1000; 1000], а выходной диапазон переменной К – [1; 10]. Правила для *dU* оставим неизменными, а для *K* будем исходить из того, что чем больше E или *dE/dt* по модулю, тем она больше. Нечёткая переменная К имеет 4 терма [ON; TW; TH; FO], которые названы исходя из первых двух букв названий английских цифр от 1 до 4 (приложение 3). Правила для К приведены в таблице 3.

Таблица 3 – Правила вывода К

К			Величина ошибки, E						
			← +			0	- →		
			LP	MP	SP	Z	SN	MN	LN
Производная ошибки, dE	↑ +	LP	FO	FO	FO	FO	FO	FO	FO
		MP	FO	TH	TH	TH	TH	TH	FO
		SP	FO	TH	TW	TW	TW	TH	FO
	0	Z	FO	TH	TW	ON	TW	TH	FO
	- ↓	SN	FO	TH	TW	TW	TW	TH	FO
		MN	FO	TH	TH	TH	TH	TH	FO

		LN	FO	FO	FO	FO	FO	FO	FO

По аналогии с указанными ранее правилами сделаем перевод из строк в числа (таблица 4):

```
=ЕСЛИ(E6 = "ON"; 1; ЕСЛИ(E6 = "TW";2;ЕСЛИ(E6 = "TH"; 3;
ЕСЛИ(E6 = "FO"; 4; "ошибка"))))
```

Таблица 4 – Числовые отображения правил К

К			Величина ошибки, E						
			← +			0	- →		
			LP	MP	SP	Z	SN	MN	LN
Производная ошибки, dE	↑ +	LP	4	4	4	4	4	4	4
		MP	4	3	3	3	3	3	4
		SP	4	3	2	2	2	3	4
	0	Z	4	3	2	1	2	3	4
	↓ -	SN	4	3	2	2	2	3	4
		MN	4	3	3	3	3	3	4
		LN	4	4	4	4	4	4	4

Код скрипта приведён в приложении 3. Переходный процесс при этом представлен на рисунке 22.

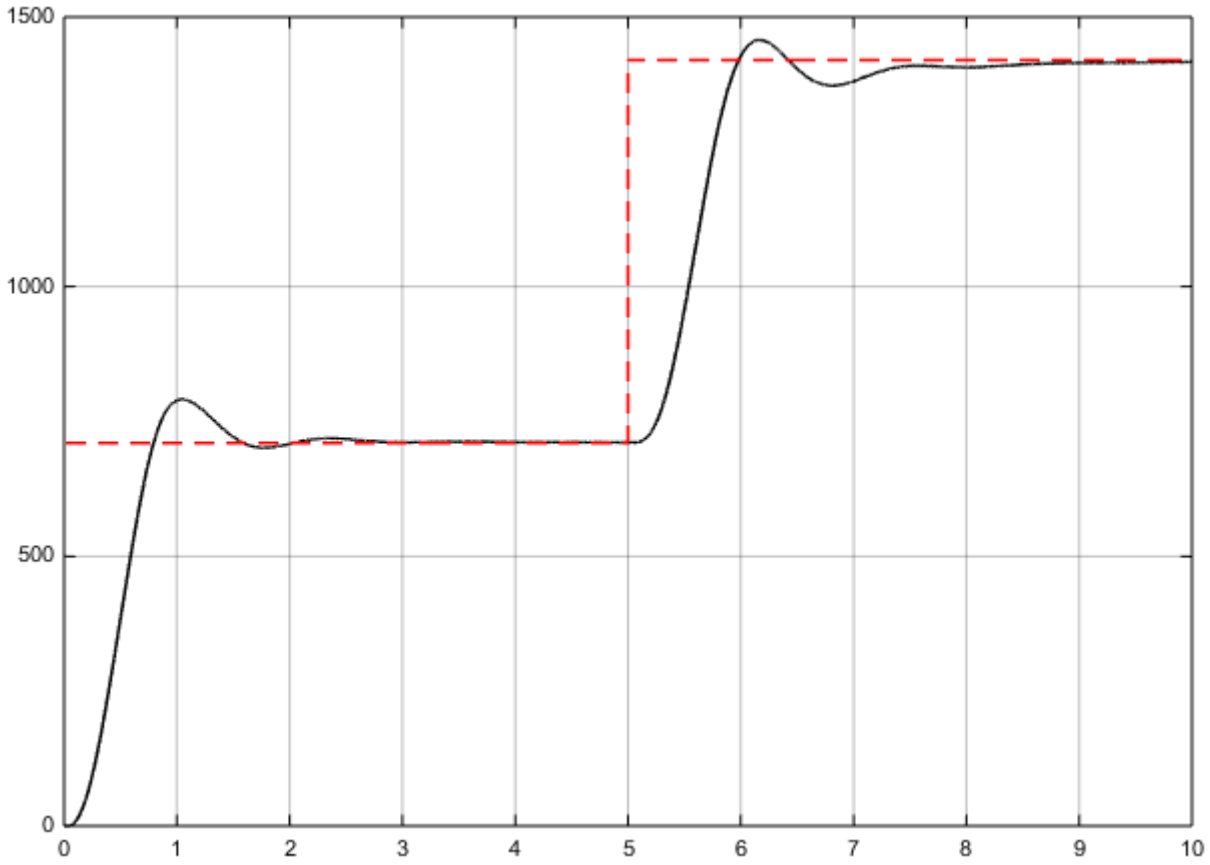


Рисунок 22 – Переходный процесс модифицированного нечетко регулятора

5. НАСТРОЙКА КОЭФФИЦИЕНТОВ PID-РЕГУЛЯТОРА С ПОМОЩЬЮ FUZZY

5.1. Применение нечеткой логики для подстройки коэффициентов ПИД-регулятора

Настройка регулятора, выполненная методом, изложенным в разделе 3, не является оптимальной и может быть улучшена с помощью дальнейшей подстройки. Подстройка может быть выполнена оператором на основании правил (раздел 3) или автоматически, с помощью блока нечеткой логики (рисунок 23). Блок нечеткой логики (фаззи-блок) использует базу правил подстройки и методы нечеткого вывода. Фаззи-подстройка позволяет уменьшить перерегулирование, снизить время установления и повысить робастность ПИД-регулятора [13].

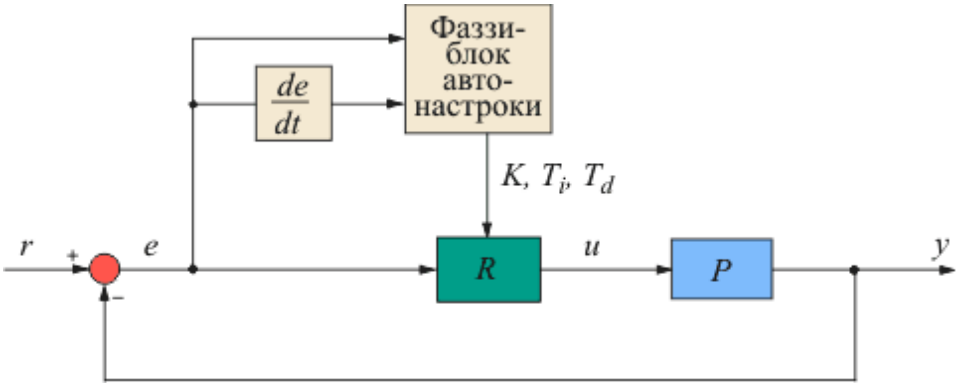


Рисунок 23 – Структура ПИД-регулятора с блоком автонастройки на основе нечеткой логики

Процесс автонастройки регулятора с помощью блока нечеткой логики начинается с поиска начальных приближений коэффициентов регулятора K, T_i, T_d. Это делается обычно методом Зиглера-Никольса, исходя из периода собственных колебаний в замкнутой системе и петлевого усиления. Далее формулируется критериальная функция, необходимая для поиска оптимальных значений параметров настройки методами оптимизации.

В процессе настройки регулятора используют несколько шагов [14]. Сначала выбирают диапазоны входных и выходных сигналов блока автонастройки, форму функций принадлежности искомых параметров, правила нечеткого вывода, механизм логического вывода, метод дефаззификации и диапазоны масштабных множителей, необходимых для пересчета четких переменных в нечеткие.

Поиск параметров регулятора выполняется методами оптимизации. Для этого выбирается целевая функция как интеграл от суммы квадратов ошибки регулирования и времени установления. В критерий минимизации иногда добавляют скорость нарастания выходной переменной объекта.

В качестве искомых параметров (параметров, которые надо найти) выбирают положение максимумов функций принадлежности (рисунок 17) имасштабные коэффициенты на входе и выходе фаззи-блока. К задаче оптимизации добавляют ограничения на диапазон изменения позиций функций

принадлежности. Оптимизация критериальной функции может быть выполнена, например, с помощью генетических алгоритмов.

Следует отметить, что в случаях, когда информации достаточно для получения точной математической модели объекта, традиционный регулятор всегда будет лучше нечеткого потому, что при синтезе нечеткого регулятора исходные данные заданы приближенно.

5.2. Разработка нечёткого ПИД-регулятора с подстройкой коэффициентов

Создание базы правил

Ещё раз повторим правила подборки коэффициентов ПИД-регулятора.

а) Увеличение K_p ведёт к ускорению переходного процесса, уменьшению устойчивости и уменьшению статической ошибки.

б) Уменьшение K_p ведёт к замедлению переходного процесса, увеличению устойчивости (уменьшение колебательности) и увеличению статической ошибки.

в) Увеличение K_i (исходя из приведённой структуры ПИД-регулятора $K_i \sim 1/T_i$, значит это правило справедливо для уменьшения T_i) приводит к уменьшению статической ошибки и к увеличению его колебательности.

г) Уменьшение K_i (увеличение T_i) приводит к увеличению статической ошибки, но уменьшает его колебательность.

д) Увеличение K_d ($K_d \sim T_d$, значит справедливо для увеличения T_d) приводит кповышению устойчивости процесса и его быстродействию, но при этом регуляторстановится восприимчивым к высокочастотным шумам.

е) Уменьшение K_d (уменьшение T_d) приводит к уменьшению устойчивости и быстродействияпроцесса, но при этом высокочастотные помехи оказывают меньшее влияние на процесс управления.

Таким образом, выведем следующие закономерности для корректировки параметров:

1. Чем больше по модулю ошибка рассогласования и её производная, тем больше должна быть K_pи тем меньшеK_iи K_d.

2. Чем меньше модулю ошибка рассогласования и её производная, тем меньше должна быть K_pи тем большеK_iи K_d.

Нечёткие логические переменные в этом регуляторе представляют собой набор термов от нуля до некоторого положительного значения (коэффициенты ПИД-регулятора не могут быть отрицательными) [Z, SP, MP, LP] (приложение 4).

Таким образом, полученные таблицы правил (таблицы 5 – 6) являются симметричными относительно нуля величины ошибки рассогласования и её производной. Кроме того, таблицы правил для интегральной K_iи дифференциальной K_d составляющих одинаковы.

Таблица 5 – Правила для K_p

P			Величина ошибки, E						
			← +			0	- →		
			LP	MP	SP	Z	SN	MN	LN
Производная ошибки, dE	↑ +	LP	LP	LP	MP	SP	MP	LP	LP
		MP	LP	MP	SP	SP	SP	MP	LP
		SP	MP	SP	SP	Z	SP	SP	MP
	0	Z	SP	SP	Z	Z	Z	SP	SP
	↓ -	SN	MP	SP	SP	Z	SP	SP	MP
		MN	LP	MP	SP	SP	SP	MP	LP
		LN	LP	LP	MP	SP	MP	LP	LP

Таблица 6 – Правила для K_iи K_d

I, D	Величина ошибки, E		
	← +	0	- →

			LP	MP	SP	Z	SN	MN	LN
Производная ошибки, dE	↑ +	LP	Z	Z	SP	MP	SP	Z	Z
		MP	Z	SP	MP	MP	MP	SP	Z
		SP	SP	MP	MP	LP	MP	MP	SP
	0	Z	MP	MP	LP	LP	LP	MP	MP
	↓ -	SN	SP	MP	MP	LP	MP	MP	SP
		MN	Z	SP	MP	MP	MP	SP	Z
		LN	Z	Z	SP	MP	SP	Z	Z

Как и ранее, с помощью небольшого скрипта Excel преобразуем буквенные обозначения в числовые:

```
=ЕСЛИ(E7="Z";1;ЕСЛИ(E7="SP";2;ЕСЛИ(E7="MP";3;ЕСЛИ(E7="LP";4;"ошибка"))))
```

В таблицах 7 и 8 приведены правила в числовом виде для коэффициентов ПИД-регулятора.

Таблица 7 – Правила для K_p в числовом виде

P			Величина ошибки, E						
			← +			0	- →		
			LP	MP	SP	Z	SN	MN	LN
Производная ошибки, dE	↑ +	LP	4	4	3	2	3	4	4
		MP	4	3	2	2	2	3	4
		SP	3	2	2	1	2	2	3
	0	Z	2	2	1	1	1	2	2
	↓ -	SN	3	2	2	1	2	2	3
		MN	4	3	2	2	2	3	4
		LN	4	4	3	2	3	4	4

Таблица 8 – Правила для K_i и K_d в числовом виде

I, D	Величина ошибки, E

			← +			0	- →		
			LP	MP	SP	Z	SN	MN	LN
Производная ошибки, dE	↑ +	LP	1	1	2	3	2	1	1
		MP	1	2	3	3	3	2	1
		SP	2	3	3	4	3	3	2
	0	Z	3	3	4	4	4	3	3
	↓ -	SN	2	3	3	4	3	3	2
		MN	1	2	3	3	3	2	1
		LN	1	1	2	3	2	1	1

Моделирование с нечётким ПИД-регулятором

Подбор максимального значения для каждого из коэффициентов производится аналогично обычному ПИД-регулятору. Переходный процесс полученного регулятора приведён на рисунке 24.

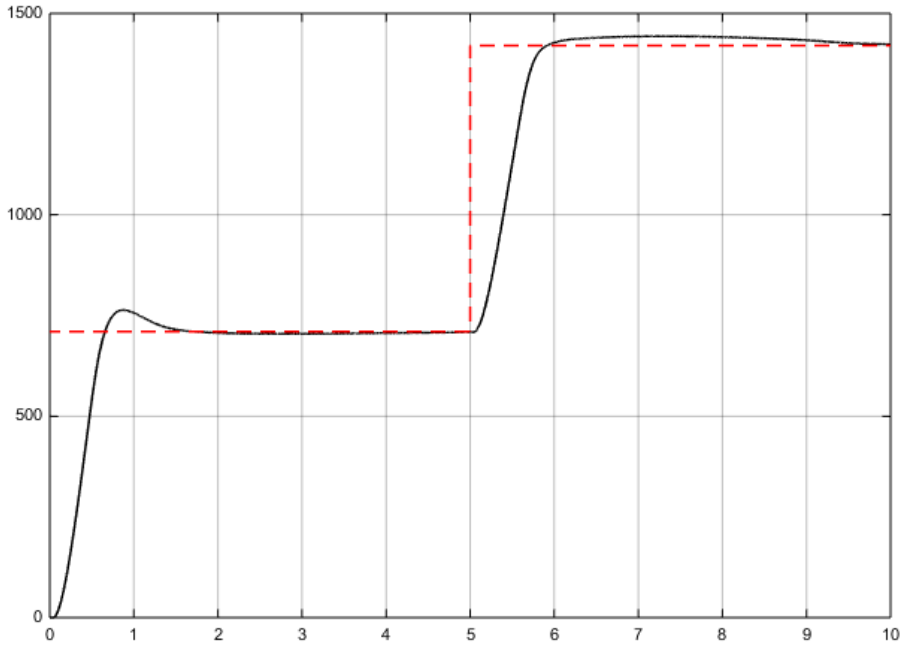


Рисунок 24 – Переходный процесс нечёткого ПИД-регулятора

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

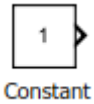
1. Исследование переходных и частотных характеристик типовых звеньев САР: метод. указания / сост. Гимадиев А.Г., Крючков А.Н., Быстров Н.Д., Ермилов М.А. – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2013. – 52 с.: ил.
2. Бесекерский, В.А. Теория систем автоматического управления [Текст] / В.А. Бесекерский, Е.П. Попов. – Санкт-Петербург, Профессия, 2003 – 752 с., ил
3. http://bookasutp.ru/
4. Методы робастного, нейро-нечеткого и адаптивного управления. Ученик/ Под ред. Н. Д. Егупова, изд. 2-е. М.: Изд-во МГТУ им. Бауман, 2002, 744 с.
5. Усков А.А., Кузьмин А.В. Интеллектуальные технологии управления. Искусственные нейронные сети и нечеткая логика. М.: "Горячая линия-Телеком", 2004, 143 с.
6. Рутковская Д, Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия-Телеком, 1006, 383 с.
7. Zadeh L.A. Fuzzy sets. - Information and Control. 1965, №8, p.338-353
8. Ротач В.Я. Теория автоматического управления. М.: МЭИ, 2004, 400 с.
9. Mamdani E. H. Application of fuzzy algorithms for control of simple dynamic plant. - Proc. Inst. Elect. Eng. Contr. Sci., vol. 121, 1974, p. 1585–1588.
10. Mamdani E. H., Assilian S. An experiment in linguistic synthesis with a fuzzy logic controller. - Int. J. Man-Mach. Stud., vol. 7, 1975, p. 1–13.
11. Astrom K.J., HagglundT.. Advanced PID control. - ISA - The Instrumentation, Systems, and Automation Society, 2006, 460 p.

12. Mann G.K.I., Bao-Gang Hu., Gosine R.G. Analysis of direct action fuzzy PID controller structures. - IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 29, Issue 3, Jun 1999, p. 371 - 388.

13. Yesil E.; Guzelkaya M.; Eksin I. Internal model control based fuzzy gain scheduling technique of pid controllers. - World Automation Congress, 2004. Proceedings. Vol. 17, 28 June - 1 July 2004, p.501 - 506.

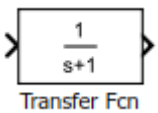
14. Hsuan -Ming Feng. A self-tuning fuzzy control system design. - IFSA World Congress and 20th NAFIPS International Conference, 2001. Joint 9th. Vol. 1, 25-28 July 2001, p. 209 - 214 vol.1.

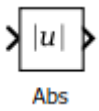
ПРИЛОЖЕНИЕ 1

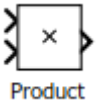
Sources	
	**Источник постоянного сигнала Constant** *Назначение*: Задает постоянный по уровню сигнал. *Параметры:* 1. **Constantvalue** - Постоянная величина. 2. **Interpretvectorparametersas 1-D –** Интерпретировать вектор параметров как одномерный (при установленном флажке). Данный параметр встречается у большинства блоков библиотеки**Simulink**. В дальнейшем он рассматриваться не будет. Значение константы может быть действительным или комплексным числом, вычисляемым выражением, вектором или матрицей.
	**Блок входного порта Inport** *Назначение:* Создает входной порт для подсистемы или модели верхнего уровня иерархии. *Параметры:* • **Portnumber –** Номер порта. • **Portdimensions –** Размерность входного сигнала. Если этот параметр равен **–1**, то размерность входного сигнала будет определяться автоматически. • **Sampletime –**Шаг модельного времени. • **Data type –** Типданныхвходногосигнала: **auto, double, single, int8, uint8, int16, uint16, int32, uint32** или **boolean.** • **Signaltype –** Тип входного сигнала: 1. **auto –** Автоматическое определение типа. 2. **real –**Действительный сигнал. 3. **complex-** Комплексный сигнал. • **Interpolatedata** (флажок) – Интерполировать входной сигнал. В случае, если временные отсчеты входного сигнала считываемого из рабочей области **MATLAB** не совпадают с модельным временем, то блок будет выполнять интерполяцию входного сигнала. При использовании блока **Inport** в подсистеме данный параметр не доступен.
	**Генератор ступенчатого сигнала Step** *Назначение*: Формирует ступенчатый сигнал. *Параметры*: 1. **Steptime** - Время наступления перепада сигнала (с).

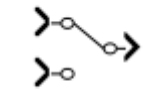
	2. **Initialvalue** - Начальное значение сигнала. 3. **Finalvalue** - Конечное значение сигнала. Перепад может быть как в большую сторону (конечное значение больше чем начальное), так и в меньшую (конечное значение меньше чем начальное). Значения начального и конечного уровней могут быть не только положительными, но и отрицательными (например, изменение сигнала с уровня –5 до уровня –3).
Sinks	
 Display	**Цифровой дисплей Display** *Назначение*: Отображает значение сигнала в виде числа. *Параметры:* • **Format** – формат отображения данных. Параметр **Format** может принимать следующие значения: 1. **short –** 5 значащих десятичных цифр. 2. **long –** 15 значащих десятичных цифр. 3. **short_e –**5 значащих десятичных цифр и 3 символа степени десяти. 4. **long_e –** 15 значащих десятичных цифр и 3 символа степени десяти. 5. **bank –** "*денежный*" формат. Формат с фиксированной точкой и двумя десятичными цифрами в дробной части числа. • **Decimation** – кратность отображения входного сигнала. При **Decimation = 1** отображается каждое значение входного сигнала, при **Decimation = 2** отображается каждое второе значение, при **Decimation = 3** – каждое третье значение и т.д. • **Sampletime** – шаг модельного времени. Определяет дискретность отображения данных. • **Floatingdisplay** (флажок)– перевод блока в "*свободный*" режим. В данном режиме входной порт блока отсутствует, а выбор сигнала для отображения выполняется щелчком левой клавиши "мыши" на соответствующей лини связи. В этом режиме для параметра расчета**Signalstoragereuse** должно быть установлено значение **off** (вкладка **Advanced** в окне диалога **Simulationparameters…**).
 Out1	**Блок выходного порта Outport** *Назначение*: Создает выходной порт для подсистемы или для модели верхнего уровня иерархии. *Параметры:* • **Portnumber –** номер порта. • **Outputwhendisabled –** вид сигнала на выходе подсистемы, в случае если подсистема выключена. Используется для управляемых подсистем. Может принимать значения (выбираются из списка):

	1. **held** – выходной сигнал подсистемы равен последнему рассчитанному значению. 2. **reset** – – выходной сигнал подсистемы равен значению задаваемому параметром**Initialoutput**. • **Initialoutput -** значение сигнала на выходе подсистемы до начала ее работы и в случае, если подсистема выключена. Используется для управляемых подсистем.
 Scope	**Осциллограф Scope** *Назначение*: Строит графики исследуемых сигналов в функции времени. Позволяет наблюдать за изменениями сигналов в процессе моделирования.
Continuous	
 Derivative	**Блок вычисления производной Derivative** *Назначение:* Выполняет численное дифференцирование входного сигнала. *Параметры:* Нет. Для вычисления производной используется приближенная формула Эйлера: $\frac{du}{dt} = \frac{\Delta u}{\Delta t}$, где Δu – величина изменения входного сигнала за время Δt, Δt – текущее значение шага модельного времени. Значение входного сигнала блока до начала расчета считается равным нулю. Начальное значение выходного сигнала также полагается равным нулю. Точность вычисления производной существенно зависит от величины установленного шага расчета. Выбор меньшего шага расчета улучшает точность вычисления производной.
 Integrator	**Интегрирующий блок Integrator** *Назначение:* Выполняет интегрирование входного сигнала. *Параметры:* • **Externalreset** – Внешний сброс. Тип внешнего управляющего сигнала, обеспечивающего сброс интегратора к начальному состоянию. Выбирается из списка: 1. **none** – нет (сброс не выполняется), 2. **rising** - нарастающий сигнал (передний фронт сигнала), 3. **falling** - спадающий сигнал (задний фронт сигнала), 4. **either** – нарастающий либо спадающий сигнал,

	5. **level** – не нулевой сигнал (сброс выполняется если сигнал на управляющем входе становится не равным нулю); В том случае, если выбран какой-либо (но не **none**), тип управляющего сигнала, то на изображении блока появляется дополнительный управляющий вход. Рядом с дополнительным входом будет показано условное обозначение управляющего сигнала. • **Initialconditionsource** — Источник начального значения выходного сигнала. Выбирается из списка: 1. **internal** – внутренний 2. **external** – внешний. В этом случае на изображении блока появляется дополнительный вход, обозначенный x_0, на который необходимо подать сигнал задающий начальное значение выходного сигнала интегратора. • **Initialcondition** — Начальное условие. Установка начального значения выходного сигнала интегратора. Параметр доступен, если выбран внутренний источник начального значения выходного сигнала. • **Limitoutput** (флажок) — Использование ограничения выходного сигнала. • **Uppersaturationlimit** — Верхний уровень ограничения выходного сигнала. Может быть задан как числом, так и символьной последовательностью **inf**, то есть + ∞. • **Lowersaturationlimit** — Нижний уровень ограничения выходного сигнала. Может быть задан как числом, так и символьной последовательностью **inf**, то есть - ∞. • **Showsaturationport** — управляет отображением порта, выводящего сигнал, свидетельствующий о выходе интегратора на ограничение. Выходной сигнал данного порта может принимать следующие значения: 1. **Ноль**, если интегратор не находится на ограничении. 2. **+1**, если выходной сигнал интегратора достиг верхнего ограничивающего предела. 3. **-1,** если выходной сигнал интегратора достиг нижнего ограничивающего предела. • **Showstateport** (флажок) — Отобразить/скрыть порт состояния блока. Данный порт используется в том случае, если выходной сигнал интегратора требуется подать в качестве сигнала обратной связи этого же интегратора. На пример, при установке начальных условий через внешний порт или при сбросе интегратора через порт сброса. Выходной сигнал с этого порта может использоваться также для организации взаимодействия с управляемой подсистемой. • **Absolutetolerance** — Абсолютная погрешность.
	**Блок передаточной функции TransferFcn** *Назначение:* Блок передаточной характеристики **TransferFcn** задает передаточную функцию в виде отношения полиномов: $$H(s) = \frac{y(s)}{u(s)} = \frac{num(s)}{den(s)} = \frac{num(1)s^{nn-1} + num(2)s^{nn-2} + \ldots + num(nn)}{den(1)s^{nd-1} + den(2)s^{nd-2} + \ldots + den(nd)},$$

	где ***nn*** и ***nd*** – порядок числителя и знаменателя передаточной функции, ***num*** – вектор или матрица коэффициентов числителя, ***den*** – вектор коэффициентов знаменателя. *Параметры:* 1. **Numerator** — вектор или матрица коэффициентов полинома числителя 2. **Denominator** -вектор коэффициентов полинома знаменателя 3. **Absolutetolerance** — Абсолютная погрешность. Порядок числителя не должен превышать порядок знаменателя.
Math Operations	
	**Блок вычисления модуля Abs** *Назначение:* Выполняет вычисление абсолютного значения величины сигнала. *Параметры:* • **Saturateonintegeroverflow** (флажок) – Подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно.
	**Усилители Gain и MatrixGain** *Назначение:* Выполняют умножение входного сигнала на постоянный коэффициент. *Параметры:* 1. **Gain** – Коэффициент усиления. 2. **Multiplication –** Способ выполнения операции. Может принимать значения (из списка): - **Element-wise K*u–** Поэлементный. - **Matrix K*u –** Матричный. Коэффициент усиления является левосторонним операндом. - **Matrix u*K –** Матричный. Коэффициент усиления является правосторонним операндом. 3. **Saturateonintegeroverflow** (флажок) – Подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно. Блоки усилителей **Gain** и **MatrixGain** есть один и тот же блок, но с разными началь-

	ными установками параметра **Multiplication**.
	**Блок вычисления математических функций MathFunction** *Назначение:* Выполняет вычисление математической функции. *Параметры:* 1. **Function –** Вид вычисляемой функции (выбирается из списка): **exp –** Экспоненциальная функция **log –** Функция натурального логарифма **10^u –** Вычисление степени **10** **log10 –** Функции логарифма **magnitude^2 –** Вычисление квадрата модуля входного сигнала **square –** Вычисление квадрата входного сигнала **sqrt –** Квадратный корень **pow –** Возведение в степень **conj –** Вычисление комплексно-сопряженного числа **reciprocal –** Вычисление частного от деления входного сигнала на **1** **hypot –**Вычисление корня квадратного из суммы квадратов входных сигналов (гипотенузы прямоугольного треугольника по значениям катетов) **rem –** Функция, вычисляющая остаток от деления первого входного сигнала на второй **mod** – Функция, вычисляющая остаток от деления с учетом знака **transpose –** Транспонирование матрицы **hermitian –** Вычисление эрмитовой матрицы. 2. **Outputsignaltype –** Тип выходного сигнала (выбирается из списка): **auto –** Автоматическое определение типа **real –**Действительный сигнал **complex-** Комплексный сигнал.
	**Блок умножения Product** *Назначение:* Выполняет вычисление произведения текущих значений сигналов. *Параметры:* 1. **Numberofinputs –** Количество входов. Может задаваться как число или как список знаков. В списке знаков можно использовать знаки *** (**умножить) и / (разделить). 2. **Multiplication –** Способ выполнения операции. Может принимать значения (из списка): **-Element-wise –** Поэлементный. **- Matrix –** Матричный. 3. **Saturateonintegeroverflow** (флажок) – Подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно. Если параметр **Numberofinputs** задан списком, включающим кроме знаков умноже-

	ния также знаки деления, то метки входов будут обозначены символами соответствующих операций.
 Sum	**Блок вычисления суммы Sum** *Назначение:* Выполняет вычисление суммы текущих значений сигналов. *Параметры:* 1. **Iconshape –** Форма блока. Выбирается из списка. **- round –** окружность, **-rectangular –** прямоугольник. 2. **List of sign** – Списокзнаков. В списке можно использовать следующие знаки: **+ (**плюс), - (минус) и **|** (разделитель знаков). 3. **Saturateonintegeroverflow** (флажок) – Подавлять переполнение целого. При установленном флажке ограничение сигналов целого типа выполняется корректно. Количество входов и операция (сложение или вычитание) определяется списком знаков параметра **Listofsign**, при этом метки входов обозначаются соответствующими знаками. В параметре **Listofsign** можно также указать число входов блока. В этом случае все входы будут суммирующими. Если количество входов блока превышает **3**, то удобнее использовать блок **Sum** прямоугольной формы.
Signal Routing	
 Manual Switch	**Блок ручного переключателя ManualSwitch** *Назначение:* Выполняет переключение входных сигналов по команде пользователя. *Параметры:* Нет. Командой на переключение является двойной щелчок левой клавишей “мыши” на изображении блока. При этом изображение блока изменяется, показывая, какой входной сигнал в данный момент проходит на выход блока. Переключение блока можно выполнять как до начала моделирования, так и в процессе расчета.
 Mux	**Мультиплексор (смеситель) Mux** *Назначение:* Объединяет входные сигналы в вектор. *Параметры:*

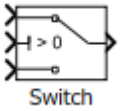
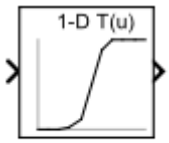
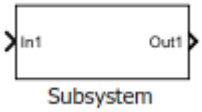
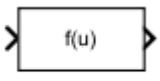
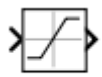
	1. **NumberofInputs** - Количество входов. 2. **Displayoption** - Способ отображения. Выбирается из списка: ○ **bar** - Вертикальный узкий прямоугольник черного цвета. ○ **signals** - Прямоугольник с белым фоном и отображением меток входных сигналов. ○ **none** - Прямоугольник с белым фоном без отображения меток входных сигналов. Входные сигналы блока могут быть скалярными и (или) векторными.
 Switch	**Блок переключателя Switch** *Назначение:* Выполняет переключение входных сигналов по сигналу управления. *Параметры:* **Threshold** – Порог управляющего сигнала. Блок работает следующим образом: Если сигнал управления, подаваемый на средний вход больше, чем величина порогового значения**Threshold**, то на выход блока проходит сигнал с первого (верхнего) входа. Если сигнал управления станет меньше, чем пороговое значение, то на выход блока будет поступать сигнал со второго (нижнего) входа.
Lookup Tables	
 1-D Lookup Table	**Блок одномерной таблицы Look-UpTable** *Назначение:* Задает в табличной форме функцию одной переменной. *Параметры:* 1. **Vectorofinputvalues –** Вектор значений входного сигнала. Может быть задан в виде дискретных значений (например, **[1 2 7 9]**), либо в виде непрерывного диапазона (например, **[0:10]**). Элементы вектора или граница диапазона могут быть заданы в виде вычисляемого выражения, например **[tan(5) sin(3)]**. 2. **Vectorofoutputvalues –** Вектор выходных значений, соответствующий вектору входных значений. Блок работает в соответствии со следующими правилами: 1. Если входной сигнал равен одному из элементов вектора входных значений (**Vectorofinputvalues**), то выходное значение блока будет равно соответствующему элементу вектора выходных значений (**Vectorofoutputvalues**). Например, пусть вектор входных значений равен **[0 1 2 5]**, а вектор выходных значений **[-5 –10 3 100],** тогда при входном сигнале равном **1** выходной сигнал будет равен **–10.** 2. Если входной сигнал не совпадает ни с одним из элементов вектора входных значений, то блок выполняет линейную интерполяцию между двумя ближайшими к нему элементами. 3. Если входной сигнал выходит за границы вектора входных значений, то блок выполняет линейную экстраполяцию по двум крайним элементам.

	График функции, заданный с помощью настроек блока отображается на его пиктограмме. Входной сигнал блока может быть векторным. В этом случае блок выполняет поэлементную операцию.
Ports & Subsystems	
 Subsystem	**Виртуальная и монолитная подсистемы Subsystem** и **AtomicSubsystem** Доступ к окну параметров подсистемы осуществляется через меню **Edit** командой **Block Parameters…** *Параметры:* 1. **Showportlabels –** Показать метки портов. 2. **Treatasatomicunit** (флажок) – Считать подсистему монолитной. Таким образом, блоки виртуальной и монолитной подсистем – это один и тот же блок, отличающийся значением данного параметра. 3. **Access –** Доступность подсистемы для изменений. Выбирается из списка: ○ **ReadWrite –** Пользователь может открывать и изменять подсистему. ○ **ReadOnly –** Пользователь может открывать подсистему только для просмотра. ○ **NoReadOrWrite –** Пользователь не может открывать и изменять подсистему. 4. **Nameoferrorcallbackfunction –** Имя функции используемой для обработки ошибок возникающих в данной подсистеме. Остальные параметры подсистемы доступны при разработке приложений с использованием **Real-TimeWorkshop** и рассмотрены в документации на это приложение. Находящийся в библиотеке блок **Subsystem** (или **AtomicSubsystem**) содержит входной и выходной порты и линию связи между ними. После того как блок подсистемы скопирован из библиотеки в модель, он становится доступным для редактирования.
User-Defined Functions	
 Fcn	**Блок задания функции Fcn** *Назначение:* Задает выражение в стиле языка программирования **C .** *Параметры:* **Expression –** Выражение, используемое блоком для вычисления выходного сигнала на основании входного. Это выражение составляется по правилам, принятым для описа-

	ния функций на языке **C**. В выражении можно использовать следующие компоненты: 1. Входной сигнал. Входной сигнал в выражении обозначается **u**, если он является скаляром. Если входной сигнал – вектор, необходимо указывать номер элемента вектора в круглых скобках. Например, **u(1)** и **u(3)** – первый и третий элементы входного вектора. 2. Константы. 3. Арифметические операторы (**+ – * /**). 4. Операторы отношения (**= = != ><>= <=**). 5. Логические операторы (**&& | | !**). 6. Круглые скобки. 7. Математические функции: **abs, acos, asin, atan, atan2, ceil, cos, cosh, exp, fabs, floor, hypot, ln, log, log10, pow, power, rem, sgn, sin, sinh, sqrt, tan,** и **tanh.** 8. Переменные из рабочей области. Если переменная рабочей области является массивом, то ее элементы должны указываться с помощью индексов в круглых скобках. Напрмер, **A(1,1)** - первый элемент матрицы **A**. Операторы отношения и логические операторы возвращают значения в виде логического нуля (**FALSE**) или логической единицы (**TRUE**). Операторы, допускаемые к использованию в выражении, имеют следующий приоритет (в порядке убывания): 1. ( ) 2. + – (унарные) 3. Возведение в степень 4. ! 5. / 6. + – (бинарные) 7. ><<= >= 8. = != 9. && 10. | | Блок не поддерживает матричные и векторные операции. Выходной сигнал блока всегда – скаляр.
Discontinuities	
 Saturation	**Блок ограничения Saturation** *Назначение:* Выполняет ограничение величины сигнала. *Параметры:* 1. **Upperlimit** - Верхний порог ограничения. 2. **Lowerlimit** - Нижний порог ограничения. 3. **Treatasgainwhenlinearizing (флажок)** - Трактовать как усилитель с коэффициентом передачи равным 1 при линеаризации.

	Выходной сигнал блока равен входному если его величина не выходит за порог ограничения. По достижении входным сигналом уровня ограничения выходной сигнал блока перестает изменяться и остается равным порогу.
Logic and Bit Operations	
 Relational Operator	**Блок вычисления операции отношения RelationalOperator** *Назначение:* Блок сравнивает текущие значения входных сигналов. *Параметры:* **RelationalOperator –** Тип операции отношения (выбирается из списка): • = = - Тождественно равно. • ~ = - Не равно. • < - Меньше. • < = - Меньше или равно. • > = - Больше или равно. • > - Больше. В операции отношения первым операндом является сигнал, подаваемый на первый (верхний) вход блока, а вторым операндом – сигнал, подаваемый на второй (нижний) вход. Выходным сигналом блока является **1**, если результат вычисления операции отношения есть **“ИСТИНА”** и **0,** если результат – **“ЛОЖЬ”**. Входные сигналы блока могут быть скалярными, векторными или матричными. Если оба входных сигнала – векторы или матрицы, то блок выполняет поэлементную операцию сравнения, при этом размерность входных сигналов должна совпадать. Если один из входных сигналов – вектор или матрица, а другой входной сигнал – скаляр, то блок выполняет сравнение скалярного входного сигнала с каждым элементом массива. Размерность выходного сигнала, в этом случае, будет определяться размерностью векторного или матричного сигнала, подаваемого на один из входов. Для операций = = (тождественно равно) и ~ = (не равно) допускается использовать комплексные входные сигналы. Входные сигналы также могут быть логического типа (**boolean**).

ПРИЛОЖЕНИЕ 2

```
%% Создание нечёткой системы
%{
Syntax: a =
newfis(fisName,fisType,andMethod,orMethod,impMethod,aggMethod,defuzzMethod)

a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'my_file')
%}
fMGTE = newfis('fuzzy_MGTE');

%% Созданиенечёткихлогическихпеременных
%{
 Syntax: a = addvar(a,'varType','varName',varBounds)
 a = addmf(a,'varType',varIndex,'mfName','mfType',mfParams)
%}

E = 500;
% Нечёткаялогическаяпеременная E
fMGTE = addvar(fMGTE,'input','E',[-E E]);
fMGTE = addmf (fMGTE,'input',1,'LN','trimf',[-E      -E      -E*0.66]);
fMGTE = addmf (fMGTE,'input',1,'MN','trimf',[-E      -E*0.66 -E*0.33]);
fMGTE = addmf (fMGTE,'input',1,'SN','trimf',[-E*0.66 -E*0.33  0     ]);
fMGTE = addmf (fMGTE,'input',1,'Z' ,'trimf',[-E*0.33  0       E*0.33]);
fMGTE = addmf (fMGTE,'input',1,'SP','trimf',[ 0       E*0.33  E*0.66]);
fMGTE = addmf (fMGTE,'input',1,'MP','trimf',[ E*0.33  E*0.66  E     ]);
fMGTE = addmf (fMGTE,'input',1,'LP','trimf',[ E*0.66  E       E     ]);

dE = 1000;
% Нечёткая логическая переменная dE/dt
fMGTE = addvar(fMGTE,'input','dE/dt',[-dEdE]);
fMGTE = addmf (fMGTE,'input',2,'LN','trimf',[-dE      -dE      -dE*0.66]);
fMGTE = addmf (fMGTE,'input',2,'MN','trimf',[-dE      -dE*0.66 -dE*0.33]);
fMGTE = addmf (fMGTE,'input',2,'SN','trimf',[-dE*0.66 -dE*0.33  0      ]);
fMGTE = addmf (fMGTE,'input',2,'Z' ,'trimf',[-dE*0.33  0        dE*0.33]);
fMGTE = addmf (fMGTE,'input',2,'SP','trimf',[ 0        dE*0.33  dE*0.66]);
fMGTE = addmf (fMGTE,'input',2,'MP','trimf',[ dE*0.33  dE*0.66  dE     ]);
fMGTE = addmf (fMGTE,'input',2,'LP','trimf',[ dE*0.66  dEdE     ]);

dU = 1000;  %ед/с
% Нечёткая логическая переменная dU
fMGTE = addvar(fMGTE,'output','dU',[-dUdU]);
fMGTE = addmf (fMGTE,'output',1,'LN','trimf',[-dU      -dU      -dU*0.66]); %
1
fMGTE = addmf (fMGTE,'output',1,'MN','trimf',[-dU      -dU*0.66 -dU*0.33]); %
2
fMGTE = addmf (fMGTE,'output',1,'SN','trimf',[-dU*0.66 -dU*0.33  0      ]); %
3
fMGTE = addmf (fMGTE,'output',1,'Z' ,'trimf',[-dU*0.33  0        dU*0.33]); %
4
fMGTE = addmf (fMGTE,'output',1,'SP','trimf',[ 0        dU*0.33  dU*0.66]); %
5
fMGTE = addmf (fMGTE,'output',1,'MP','trimf',[ dU*0.33  dU*0.66  dU     ]); %
6
fMGTE = addmf (fMGTE,'output',1,'LP','trimf',[ dU*0.66  dUdU     ]); % 7

%% Созданиебазыправил
%{
Syntax: a = addrule(a,ruleList)
```

```
Create the fuzzy inference system. For this example, load the tipper FIS and
clear the existing rules for the FIS.

a = readfis('tipper');
a.rule=[];
Create the rule list.

rule1 = [1 1 1 1 1];
rule2 = [1 2 2 1 1];
ruleList = [rule1;rule2];
Add the rules list to the FIS.

a = addrule(a,ruleList);
Verify that the rules were added.

showrule(a)
ans =

1. If (service is poor) and (food is rancid) then (tip is cheap) (1)
2. If (service is poor) and (food is delicious) then (tip is average) (1)
%}

%            E  dEdU   w  and          №
ruleList = [7   7   7   1   1   ;   %   1
            7   6   7   1   1   ;   %   2
            7   5   7   1   1   ;   %   3
            7   4   7   1   1   ;   %   4
            7   3   6   1   1   ;   %   5
            7   2   5   1   1   ;   %   6
            7   1   4   1   1   ;   %   7
            6   7   7   1   1   ;   %   8
            6   6   7   1   1   ;   %   9
            6   5   6   1   1   ;   %   10
            6   4   6   1   1   ;   %   11
            6   3   5   1   1   ;   %   12
            6   2   4   1   1   ;   %   13
            6   1   3   1   1   ;   %   14
            5   7   7   1   1   ;   %   15
            5   6   6   1   1   ;   %   16
            5   5   6   1   1   ;   %   17
            5   4   5   1   1   ;   %   18
            5   3   4   1   1   ;   %   19
            5   2   4   1   1   ;   %   20
            5   1   2   1   1   ;   %   21
            4   7   7   1   1   ;   %   22
            4   6   6   1   1   ;   %   23
            4   5   6   1   1   ;   %   24
            4   4   4   1   1   ;   %   25
            4   3   2   1   1   ;   %   26
            4   2   2   1   1   ;   %   27
            4   1   1   1   1   ;   %   28
            3   7   6   1   1   ;   %   29
            3   6   4   1   1   ;   %   30
            3   5   4   1   1   ;   %   31
            3   4   3   1   1   ;   %   32
            3   3   2   1   1   ;   %   33
            3   2   2   1   1   ;   %   34
            3   1   1   1   1   ;   %   35
            2   7   5   1   1   ;   %   36
            2   6   4   1   1   ;   %   37
            2   5   3   1   1   ;   %   38
            2   4   2   1   1   ;   %   39
            2   3   2   1   1   ;   %   40
            2   2   1   1   1   ;   %   41
```

```
2   1   1   1   1   ;   %   42
            1   7   4   1   1   ;   %   43
            1   6   3   1   1   ;   %   44
            1   5   2   1   1   ;   %   45
            1   4   1   1   1   ;   %   46
            1   3   1   1   1   ;   %   47
            1   2   1   1   1   ;   %   48
            1   1   1   1   1   ];  %   49

fMGTE = addrule(fMGTE, ruleList);

%% Записьнечёткойсистемывфайл
%{
 Syntax:
writefis(fismat)
writefis(fismat,'filename')
writefis(fismat,'filename','dialog')
%}

writefis(fMGTE,'fuzzy_MGTE')
```

ПРИЛОЖЕНИЕ3

```
%% Созданиенечёткойсистемы
%{
 Syntax: a =
newfis(fisName,fisType,andMethod,orMethod,impMethod,aggMethod,defuzzMethod)

a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'my_file')
%}
fMGTE = newfis('fuzzy_MGTE');

%% Созданиенечёткихлогическихпеременных
%{
 Syntax: a = addvar(a,'varType','varName',varBounds)
 a = addmf(a,'varType',varIndex,'mfName','mfType',mfParams)
%}

E = 500;
% Нечёткаялогическаяпеременная E
fMGTE = addvar(fMGTE,'input','E',[-E E]);
fMGTE = addmf (fMGTE,'input',1,'LN','trimf',[-E      -E      -E*0.66]);
fMGTE = addmf (fMGTE,'input',1,'MN','trimf',[-E      -E*0.66 -E*0.33]);
fMGTE = addmf (fMGTE,'input',1,'SN','trimf',[-E*0.66 -E*0.33  0     ]);
fMGTE = addmf (fMGTE,'input',1,'Z' ,'trimf',[-E*0.33  0       E*0.33]);
fMGTE = addmf (fMGTE,'input',1,'SP','trimf',[ 0       E*0.33  E*0.66]);
fMGTE = addmf (fMGTE,'input',1,'MP','trimf',[ E*0.33  E*0.66  E     ]);
fMGTE = addmf (fMGTE,'input',1,'LP','trimf',[ E*0.66  E       E     ]);

dE = 1000;
% Нечёткая логическая переменная dE/dt
fMGTE = addvar(fMGTE,'input','dE/dt',[-dEdE]);
fMGTE = addmf (fMGTE,'input',2,'LN','trimf',[-dE      -dE      -dE*0.66]);
fMGTE = addmf (fMGTE,'input',2,'MN','trimf',[-dE      -dE*0.66 -dE*0.33]);
fMGTE = addmf (fMGTE,'input',2,'SN','trimf',[-dE*0.66 -dE*0.33  0      ]);
fMGTE = addmf (fMGTE,'input',2,'Z' ,'trimf',[-dE*0.33  0        dE*0.33]);
fMGTE = addmf (fMGTE,'input',2,'SP','trimf',[ 0        dE*0.33  dE*0.66]);
fMGTE = addmf (fMGTE,'input',2,'MP','trimf',[ dE*0.33  dE*0.66  dE     ]);
fMGTE = addmf (fMGTE,'input',2,'LP','trimf',[ dE*0.66  dEdE     ]);

dU = 1000;  %ед/с
% Нечёткая логическая переменная dU
fMGTE = addvar(fMGTE,'output','dU',[-dUdU]);
fMGTE = addmf (fMGTE,'output',1,'LN','trimf',[-dU      -dU      -dU*0.66]); %
1
fMGTE = addmf (fMGTE,'output',1,'MN','trimf',[-dU      -dU*0.66 -dU*0.33]); %
2
fMGTE = addmf (fMGTE,'output',1,'SN','trimf',[-dU*0.66 -dU*0.33  0      ]); %
3
fMGTE = addmf (fMGTE,'output',1,'Z' ,'trimf',[-dU*0.33  0        dU*0.33]); %
4
fMGTE = addmf (fMGTE,'output',1,'SP','trimf',[ 0        dU*0.33  dU*0.66]); %
5
fMGTE = addmf (fMGTE,'output',1,'MP','trimf',[ dU*0.33  dU*0.66  dU     ]); %
6
fMGTE = addmf (fMGTE,'output',1,'LP','trimf',[ dU*0.66  dUdU     ]); % 7

K = 10;
% Нечёткаялогическаяпеременная K
fMGTE = addvar(fMGTE,'output','K',[1 K]);
```

```
fMGTE = addmf (fMGTE,'output',2,'ON','trimf',[1          1          1*(K-
1)/4]); % 1
fMGTE = addmf (fMGTE,'output',2,'TW','trimf',[1          1*(K-1)/4 2*(K-
1)/4]); % 2
fMGTE = addmf (fMGTE,'output',2,'TH','trimf',[1*(K-1)/4 2*(K-1)/4 3*(K-
1)/4]); % 3
fMGTE = addmf (fMGTE,'output',2,'FO','trimf',[2*(K-1)/4 3*(K-1)/4 4*(K-
1)/4]); % 4

%% Созданиебазыправил
%{
Syntax: a = addrule(a,ruleList)
Create the fuzzy inference system. For this example, load the tipper FIS and
clear the existing rules for the FIS.

a = readfis('tipper');
a.rule=[];
Create the rule list.

rule1 = [1 1 1 1 1];
rule2 = [1 2 2 1 1];
ruleList = [rule1;rule2];
Add the rules list to the FIS.

a = addrule(a,ruleList);
Verify that the rules were added.

showrule(a)
ans =

1. If (service is poor) and (food is rancid) then (tip is cheap) (1)
2. If (service is poor) and (food is delicious) then (tip is average) (1)
%}

%            E  dEdUK   w  and          №
ruleList = [7   7   7   4   1   1   ;   %   1
            7   6   7   4   1   1   ;   %   2
            7   5   7   4   1   1   ;   %   3
            7   4   7   4   1   1   ;   %   4
            7   3   6   4   1   1   ;   %   5
            7   2   5   4   1   1   ;   %   6
            7   1   4   4   1   1   ;   %   7
            6   7   7   4   1   1   ;   %   8
            6   6   7   3   1   1   ;   %   9
            6   5   6   3   1   1   ;   %   10
            6   4   6   3   1   1   ;   %   11
            6   3   5   3   1   1   ;   %   12
            6   2   4   3   1   1   ;   %   13
            6   1   3   4   1   1   ;   %   14
            5   7   7   4   1   1   ;   %   15
            5   6   6   3   1   1   ;   %   16
            5   5   6   2   1   1   ;   %   17
            5   4   5   2   1   1   ;   %   18
            5   3   4   2   1   1   ;   %   19
            5   2   4   3   1   1   ;   %   20
            5   1   2   4   1   1   ;   %   21
            4   7   7   4   1   1   ;   %   22
            4   6   6   3   1   1   ;   %   23
            4   5   6   2   1   1   ;   %   24
            4   4   4   1   1   1   ;   %   25
            4   3   2   2   1   1   ;   %   26
            4   2   2   3   1   1   ;   %   27
            4   1   1   4   1   1   ;   %   28
            3   7   6   4   1   1   ;   %   29
```

```
    3   6   4   3   1   1   ;   %   30
    3   5   4   2   1   1   ;   %   31
    3   4   3   2   1   1   ;   %   32
    3   3   2   2   1   1   ;   %   33
    3   2   2   3   1   1   ;   %   34
    3   1   1   4   1   1   ;   %   35
    2   7   5   4   1   1   ;   %   36
    2   6   4   3   1   1   ;   %   37
    2   5   3   3   1   1   ;   %   38
    2   4   2   3   1   1   ;   %   39
    2   3   2   3   1   1   ;   %   40
    2   2   1   3   1   1   ;   %   41
    2   1   1   4   1   1   ;   %   42
    1   7   4   4   1   1   ;   %   43
    1   6   3   4   1   1   ;   %   44
    1   5   2   4   1   1   ;   %   45
    1   4   1   4   1   1   ;   %   46
    1   3   1   4   1   1   ;   %   47
    1   2   1   4   1   1   ;   %   48
    1   1   1   4   1   1   ];  %   49

fMGTE = addrule(fMGTE, ruleList);

%% Записьнечёткойсистемывфайл
%{
Syntax:
writefis(fismat)
writefis(fismat,'filename')
writefis(fismat,'filename','dialog')
%}

writefis(fMGTE,'fuzzy_MGTE')
```

ПРИЛОЖЕНИЕ4

```
%% Созданиенечёткойсистемы
%{
 Syntax: a =
newfis(fisName,fisType,andMethod,orMethod,impMethod,aggMethod,defuzzMethod)

a = newfis('tipper');
a = addvar(a,'input','service',[0 10]);
a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
writefis(a,'my_file')
%}
fMGTE = newfis('fuzzy_MGTE');

%% Созданиенечёткихлогическихпеременных
%{
 Syntax: a = addvar(a,'varType','varName',varBounds)
 a = addmf(a,'varType',varIndex,'mfName','mfType',mfParams)
%}

E = 500;
% Нечёткаялогическаяпеременная E
fMGTE = addvar(fMGTE,'input','E',[-E E]);
fMGTE = addmf (fMGTE,'input',1,'LN','trimf',[-E      -E      -E*0.66]);
fMGTE = addmf (fMGTE,'input',1,'MN','trimf',[-E      -E*0.66 -E*0.33]);
fMGTE = addmf (fMGTE,'input',1,'SN','trimf',[-E*0.66 -E*0.33  0     ]);
fMGTE = addmf (fMGTE,'input',1,'Z' ,'trimf',[-E*0.33  0       E*0.33]);
fMGTE = addmf (fMGTE,'input',1,'SP','trimf',[ 0       E*0.33  E*0.66]);
fMGTE = addmf (fMGTE,'input',1,'MP','trimf',[ E*0.33  E*0.66  E     ]);
fMGTE = addmf (fMGTE,'input',1,'LP','trimf',[ E*0.66  E       E     ]);

dE = 1000;
% Нечёткая логическая переменная dE/dt
fMGTE = addvar(fMGTE,'input','dE/dt',[-dEdE]);
fMGTE = addmf (fMGTE,'input',2,'LN','trimf',[-dE      -dE      -dE*0.66]);
fMGTE = addmf (fMGTE,'input',2,'MN','trimf',[-dE      -dE*0.66 -dE*0.33]);
fMGTE = addmf (fMGTE,'input',2,'SN','trimf',[-dE*0.66 -dE*0.33  0      ]);
fMGTE = addmf (fMGTE,'input',2,'Z' ,'trimf',[-dE/4     0        dE*0.33]);
fMGTE = addmf (fMGTE,'input',2,'SP','trimf',[ 0        dE*0.33  dE*0.66]);
fMGTE = addmf (fMGTE,'input',2,'MP','trimf',[ dE*0.33  dE*0.66  dE     ]);
fMGTE = addmf (fMGTE,'input',2,'LP','trimf',[ dE*0.66  dEdE     ]);

Kp = 10;
% НечёткаялогическаяпеременнаяKp
fMGTE = addvar(fMGTE,'output','Kp',[Kp*0.1 Kp]);
fMGTE = addmf (fMGTE,'output',1,'Z' ,'trimf',[Kp*0.1 Kp*0.1 Kp*0.4]);
fMGTE = addmf (fMGTE,'output',1,'SP','trimf',[Kp*0.1 Kp*0.4 Kp*0.7]);
fMGTE = addmf (fMGTE,'output',1,'MP','trimf',[Kp*0.4 Kp*0.7 Kp     ]);
fMGTE = addmf (fMGTE,'output',1,'LP','trimf',[Kp*0.7 KpKp     ]);

Ti = 1/0.5;
% НечёткаялогическаяпеременнаяTi
fMGTE = addvar(fMGTE,'output','Ti',[Ti*0.1 Ti]);
fMGTE = addmf (fMGTE,'output',2,'Z' ,'trimf',[Ti*0.1 Ti*0.1 Ti*0.4]);
fMGTE = addmf (fMGTE,'output',2,'SP','trimf',[Ti*0.1 Ti*0.4 Ti*0.7]);
fMGTE = addmf (fMGTE,'output',2,'MP','trimf',[Ti*0.4 Ti*0.7 Ti    ]);
fMGTE = addmf (fMGTE,'output',2,'LP','trimf',[Ti*0.7 TiTi    ]);

Td = 1;
% Нечёткаялогическаяпеременная Td
fMGTE = addvar(fMGTE,'output','Td',[Td*0.1 Td]);
fMGTE = addmf (fMGTE,'output',3,'Z' ,'trimf',[Td*0.1 Td*0.1 Td*0.4]);
```

```
fMGTE = addmf (fMGTE,'output',3,'SP','trimf',[Td*0.1 Td*0.4 Td*0.7]);
fMGTE = addmf (fMGTE,'output',3,'MP','trimf',[Td*0.4 Td*0.7 Td    ]);
fMGTE = addmf (fMGTE,'output',3,'LP','trimf',[Td*0.7 Td     Td    ]);

%% Созданиебазыправил
%{
Syntax: a = addrule(a,ruleList)
Create the fuzzy inference system. For this example, load the tipper FIS and
clear the existing rules for the FIS.

a = readfis('tipper');
a.rule=[];
Create the rule list.

rule1 = [1 1 1 1 1];
rule2 = [1 2 2 1 1];
ruleList = [rule1;rule2];
Add the rules list to the FIS.

a = addrule(a,ruleList);
Verify that the rules were added.

showrule(a)
ans =

1. If (service is poor) and (food is rancid) then (tip is cheap) (1)
2. If (service is poor) and (food is delicious) then (tip is average) (1)
%}
%           E  dEKpTi  Td   w  and          №
ruleList = [7   7   4   1   1   1   1   ;   %   1
            7   6   4   1   1   1   1   ;   %   2
            7   5   3   2   2   1   1   ;   %   3
            7   4   2   3   3   1   1   ;   %   4
            7   3   3   2   2   1   1   ;   %   5
            7   2   4   1   1   1   1   ;   %   6
            7   1   4   1   1   1   1   ;   %   7
            6   7   4   1   1   1   1   ;   %   8
            6   6   3   2   2   1   1   ;   %   9
            6   5   2   3   3   1   1   ;   %   10
            6   4   2   3   3   1   1   ;   %   11
            6   3   2   3   3   1   1   ;   %   12
            6   2   3   2   2   1   1   ;   %   13
            6   1   4   1   1   1   1   ;   %   14
            5   7   3   2   2   1   1   ;   %   15
            5   6   2   3   3   1   1   ;   %   16
            5   5   2   3   3   1   1   ;   %   17
            5   4   1   4   4   1   1   ;   %   18
            5   3   2   3   3   1   1   ;   %   19
            5   2   2   3   3   1   1   ;   %   20
            5   1   3   2   2   1   1   ;   %   21
            4   7   2   3   3   1   1   ;   %   22
            4   6   2   3   3   1   1   ;   %   23
            4   5   1   4   4   1   1   ;   %   24
            4   4   1   4   4   1   1   ;   %   25
            4   3   1   4   4   1   1   ;   %   26
            4   2   2   3   3   1   1   ;   %   27
            4   1   2   3   3   1   1   ;   %   28
            3   7   3   2   2   1   1   ;   %   29
            3   6   2   3   3   1   1   ;   %   30
            3   5   2   3   3   1   1   ;   %   31
            3   4   1   4   4   1   1   ;   %   32
            3   3   2   3   3   1   1   ;   %   33
            3   2   2   3   3   1   1   ;   %   34
            3   1   3   2   2   1   1   ;   %   35
```

```
    2   7   4   1   1   1   1   ;   %   36
    2   6   3   2   2   1   1   ;   %   37
    2   5   2   3   3   1   1   ;   %   38
    2   4   2   3   3   1   1   ;   %   39
    2   3   2   3   3   1   1   ;   %   40
    2   2   3   2   2   1   1   ;   %   41
    2   1   4   1   1   1   1   ;   %   42
    1   7   4   1   1   1   1   ;   %   43
    1   6   4   1   1   1   1   ;   %   44
    1   5   3   2   2   1   1   ;   %   45
    1   4   2   3   3   1   1   ;   %   46
    1   3   3   2   2   1   1   ;   %   47
    1   2   4   1   1   1   1   ;   %   48
    1   1   4   1   1   1   1   ];  %   49

fMGTE = addrule(fMGTE, ruleList);

%% Записьнечёткойсистемывфайл
%{
Syntax:
writefis(fismat)
writefis(fismat,'filename')
writefis(fismat,'filename','dialog')
%}
writefis(fMGTE,'fuzzy_PID_MGTE_U')
```