Міністерство освіти і науки, молоді та спорту України

Донбаська державна машинобудівна академія

Укладач **О. І. Шеремет**

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«Системи та нові принципи керування електроприводами»

для студентів спеціальності 7.092203 всіх форм навчання

Затверджено
Декан ФАМІТ
_____С.В. Подлєсний

Затверджено
на засіданні
методичного семінару кафедри ЕСА
Протокол № 1 від 21 серпня 2012 р.

Краматорськ 2012

Міністерство освіти і науки, молоді та спорту України

Донбаська державна машинобудівна академія

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«Системи та нові принципи керування електроприводами»

для студентів спеціальності 7.092203 всіх форм навчання

Краматорськ 2012

Содержание

1 ОСНОВНЫЕ СВЕДЕНИЯ О ТЕОРИИ ФАЗЗИ-ЛОГИКИ И ФАЗЗИ-РЕГУЛИРОВАНИИ

Давним стремлением человечества было создание искусственного интеллекта. Интеллектуальность (разумность) человека определяется наличием комплексной способности, с помощью которой он распознает объекты, явления и соответственно ситуации адекватно на них реагирует.

Основными признаками интеллекта являются:

- адекватное осознание ситуации при неоднозначной или противоречивой информации;

- нахождение аналогий, способность к классификации;

- способность к изучению новых понятий и установлению связей между разными понятиями, способность к проведению анализа и синтеза.

Задачей теории искусственного интеллекта, как области прикладной математики, является формализация процессов распознавания, обобщения и познания, перенос их на язык алгоритмов. При этом умственная деятельность сводится к обработке информации. Это означает, что если теория искусственного интеллекта исходит из того, что деятельность человеческого ума в определенной мере может быть рассмотрена независимо от функций мозга, то достаточно знать основные принципы умного поведения, чтобы записать их в виде алгоритмов, понятных для компьютера.

Таким образом, инженерно-научный аспект теории искусственного интеллекта состоит в проектировании сложных систем обработки информации [1]. Эта задача включает в себя разработку, как методов обработки информации, так и соответствующих устройств и программного обеспечения.

Выделяют несколько направлений развития искусственного интеллекта, основанного на применении компьютерных технологий [1]:

- фаззи-логика;

- экспертные системы;

- нейросети;

- эволюционное моделирование.

Перед разработчиками искусственного интеллекта стоит цель создания компьютерных систем, способных выполнять:

1) *решение сложных многомерных логических задач.*
2) *распознавание и понимание человеческой речи;*
3) *распознавание и обработка изображений;*
4) *обучение системы;*
5) *экспертные оценки;*
6) *создание качественных выводов.*

Решение первой, пятой и шестой задач тесно связано с фаззи-логикой. Английский термин **fuzzy** означает нечеткий, размытый и произносится «**фаззи»**.

Фаззи-логика – это логика нечетких множеств.

Нечеткая логика является многозначной логикой, что позволяет определить промежуточные значения для таких общепринятых оценок, как *да|нет, истинно|ложно, черное|белое* и т.п.

Теория фаззи-логики позволяет выражения подобные таким, как *слегка тепло* или *довольно холодно,* формулировать математически и обрабатывать на компьютерах.

Понятие нечеткой логики появилось в 1965 г. в работах Лотфи А. Задэ (Lotfi A. Zadeh), профессора Калифорнийского университета в Беркли [2].

1.1 Понятие нечетких множеств

Из классической математики известно понятие *четких (определенных) множеств* [3]. Рассмотрим, например, множество **A** всех чисел от 0 до 10.

Определим подмножество **A0** множества **x** всех действительных чисел от 5 до 7: **A0** = [5,7].

График характеристической функции множества **A0**, в зависимости от того, принадлежит данный элемент подмножеству **A** (что соответствует логической 1) или нет (что соответствует логическому 0), представлен на рисунке 1.1.

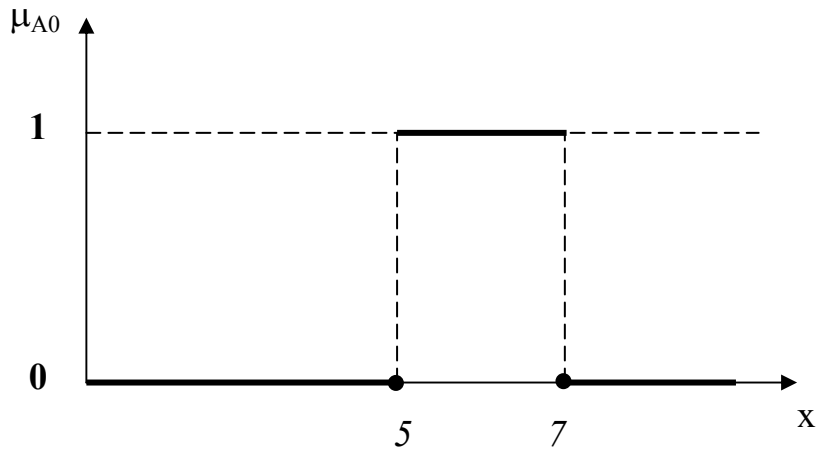


***Рисунок 1.1 – График характеристической функции μ_{A0} четкого подмножества** A0*

Рассмотрим пример нечеткого множества.

При механической обработке деталей на токарном станке для определенных материалов существуют оптимальные скорости резания. Пусть электропривод главного движения станка для конкретного материала детали должен обеспечить технологически удовлетворительную скорость резания (частоту вращения шпинделя) в диапазоне D = 20…40 об/мин, причем при частоте n_0 = 30 об/мин качество обработки – *идеальное*, а при частотах больших и меньших n_0 качество в указанном диапазоне не соответствует идеальному значению, но *удовлетворительное*. Причем, чем больше отклонение частоты от значения n_0, тем хуже качество обработки. При скоростях обработки, соответствующих частотам вращения шпинделя менее 20 и более 40 об/мин, качество обработки *неудовлетворительное*, а деталь бракуется.

Рассмотрим, как с помощью нечеткого множества определить такое выражение, как *удовлетворительные частоты.*

Обозначим нечеткое множество «*удовлетворительные частоты»* индексом **N.**

В первом примере мы кодировали все элементы рассуждения с помощью 0 или 1. Простой способ обобщить данную концепцию - ввести значения между 0 и 1. Реально можно даже допустить бесконечное число значений между 0 и 1, называемое единичным интервалом **I** = [0, 1].

В этом примере нечеткое множество удовлетворительных частот имеет разное отношение к качеству обработки детали.

Идеальная частота n_0 дает идеальное качество, имеющее код 1, частоты 20 и 40 об/мин дают неудовлетворительное качество обработки, т.е. нулевое (код 0). Промежуточные частоты в поддиапазоне $40 > n > 30$ об/мин, а также $30 > n > 20$ об/мин имеют разные отношения к качеству.

Зависимость качества от частоты может быть линейной или нелинейной.

В первом приближении примем эту зависимость линейной.

График характеристической функции μ_N множества **N** приведен на рисунке 1.2.

Таким образом, нечеткое множество – это множество, элементы которого принадлежат к нему с разной степенью, которая находится в интервале от 0 до 1.

Математически нечеткое множество **N** записывается таким образом:

$$\mathbf{N} = \{ [\ \mu_N(n_1)\ ] ... [\ \mu_N(n_n)\ ] \}. \qquad (1.1)$$

Это означает, что величины n_i принадлежат к нечеткому множеству **N** со степенью принадлежности $\mu_N(n_i)$. Как правило, для идентификации фаззи-множеств, например **A**, используют так называемую функцию принадлежности $\boldsymbol{\mu_A(x_i)}$. Функции принадлежности могут принимать разнообразные формы, наиболее распространенные из них приведены в таблице 1.1 [1].

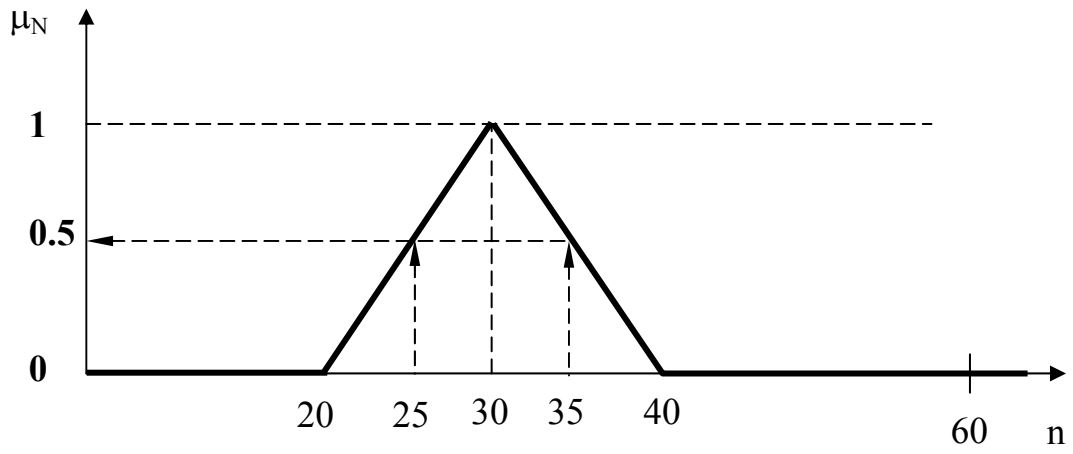


Рисунок 1.2 – График характеристической μ_N функции нечеткого множества N

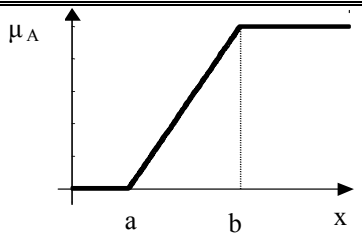
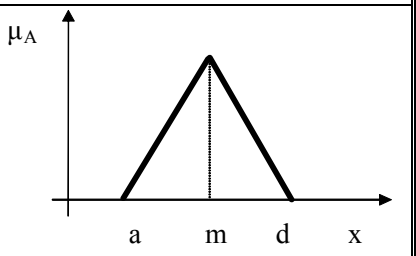
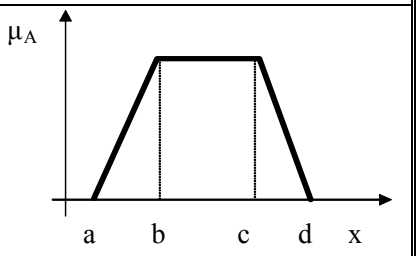
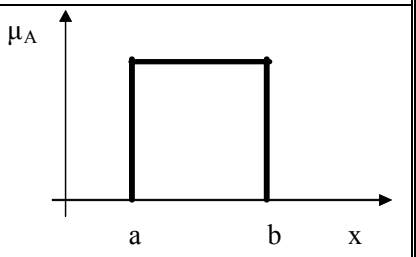
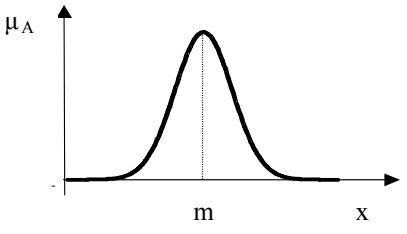
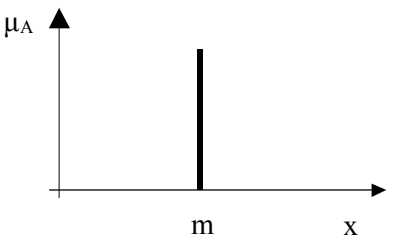
Аналогично действиям с обычными множествами при решении задач управления требуется определять **пересечение** (конъюнкцию), **объединение** (дизъюнкцию) и **отрицание** (инверсию) нечетких множеств [1].

В своей самой первой работе по нечетким множествам Л. А. Задэ предложил **оператор минимума** для **пересечения** и **оператор максимума** для **объединения** двух нечетких множеств.

Эти операции совпадают с обычными (*четкими*) объединением и пересечением, только рассматриваются степени принадлежности 0 и 1.

Логическая операция «**И**» реализует пересечение фаззи-множеств.

Таблица 1.1 – Функции принадлежности нечетких множеств[1]

Тип функции	Математическое определение	Графическое изображение
Монотонная (линейная)	$\mu_A(x) = \begin{cases} 0 & \text{при } x < a, \\ \dfrac{x-a}{b-a} & \text{при } a \le x \le b, \\ 1 & \text{при } b \le x \end{cases}$	
Треугольная	$\mu_A(x) = \begin{cases} 0 \text{ при } x < a \vee d \le x, \\ \dfrac{x-a}{m-a} \text{ при } a \le x \le m, \\ \dfrac{x-d}{m-d} \text{ при } m \le x \le d \end{cases}$	
Трапециевидная	$\mu_A(x) = \begin{cases} 0 \text{ при } x < a \vee d \le x, \\ \dfrac{x-a}{b-a} \text{ при } a \le x \le b, \\ 1 \text{ при } b \le x \le c, \\ \dfrac{x-d}{c-d} \text{ при } c \le x \le d \end{cases}$	
Прямоугольная	$\mu_A(x) = \begin{cases} 0 \text{ при } x < a \vee b < x, \\ 1 \text{ при } a \le x \le b \end{cases}$	
Функция Гаусса	$\mu_A(x) = e^{-a\cdot(x-m)^2} \quad a > 0$	
Синглетон-функция	$\mu_A(x) = \begin{cases} 1 \text{ при } x = m, \\ 0 \text{ при } x \ne m \end{cases}$	

Пересечение фаззи-множеств $\mathbf{A_1 \cap A_2}$ математически с учетом критерия минимума определяется:

$$\mu_{A_1 \cap A_2}(x) = \min\left\{ \mu_{A_1}(x), \mu_{A_2}(x) \right\}. \qquad (1.2)$$

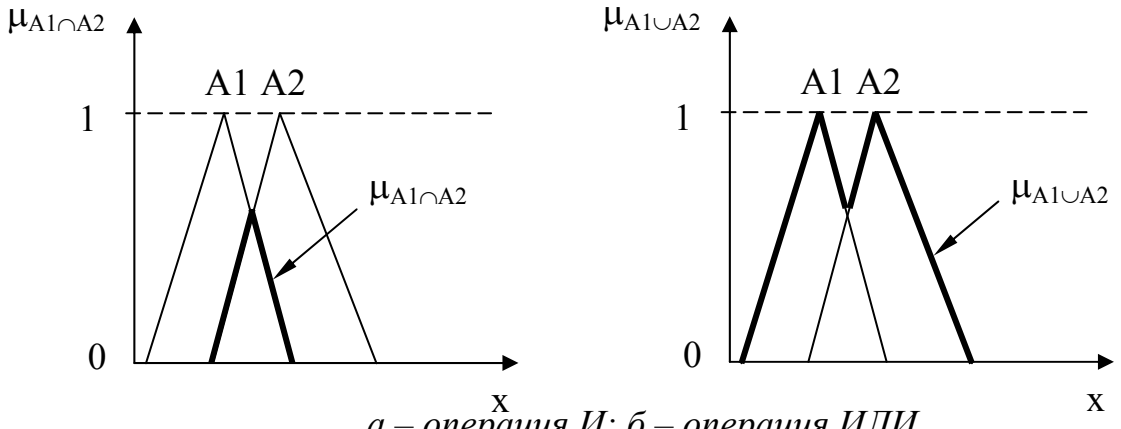
Логическая операция «**ИЛИ**» реализует объединение фаззи-множеств. Объединение двух фаззи-множеств $\mathbf{A_1 \cup A_2}$ математически с учетом критерия максимума определяется:

$$\mu_{A_1 \cup A_2}(x) = \max\left\{ \mu_{A_1}(x), \mu_{A_2}(x) \right\} \quad . \qquad (1.3)$$

Логическая операция «**НЕ**» реализует логическое отрицание фаззи-множеств. Логическое отрицание $\neg\mathbf{A}$ фаззи-множества **A** определяется:

$$\mu_{\neg A}(x) = 1 - \mu_A(x) \qquad (1.4)$$

Логические операции над фаззи-множествами могут быть рассмотрены как обобщения классических операций над множествами. На рисунках 1.3, 1.4 приведено графическое изображение представленных выше операций с нечеткими множествами.



а – операция И; б – операция ИЛИ

Рисунок 1.3 – Графики, иллюстрирующие логические операции с нечеткими множествами

В процессе дальнейшего развития фаззи-логики были определены также другие операторы для фаззи-множеств [1].

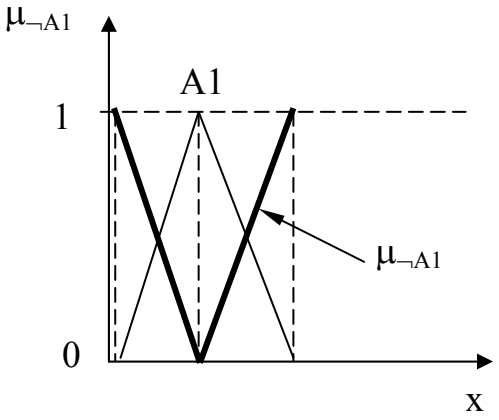


Рисунок 1.4 – График, иллюстрирующий логическую операцию НЕ с нечетким множеством A1

Рассмотрим применение фаззи-логики на примере управления электроприводом механизма передвижения тележки, на которой установлен перевернутый маятник (рисунок 1.5). Нижний конец маятника шарнирно закреплен на тележке, которая может двигаться только в двух направлениях: влево или вправо.

Задача управления электроприводом передвижения тележки состоит в поддержании перевернутого маятника в вертикальном положении.

Например, при падении маятника вправо привод должен перемещать тележку вправо и, наоборот, при падении маятника влево привод должен перемещать тележку влево. Система управления должна контролировать угол φ поворота маятника, скорость ω поворота рычага (см. рисунок 1.5) и по их соотношениям выдавать сигнал управления на установку соответствующего знака и значения скорости v тележки.

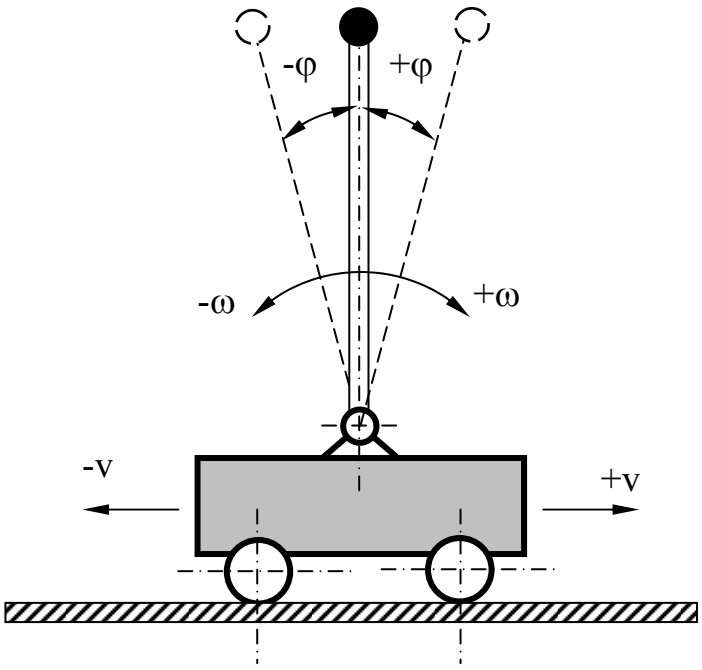


Рисунок 1.5 – Кинематическая схема перевернутого маятника

Для упрощения предполагается, что начальное положение мачты *около центра справа*, а угол более 45^0 в любом направлении по условию никогда не возникнет.

Практически эта задача решается системой управления с фаззи-контроллером, структура которой приведена на рисунке 1.6.

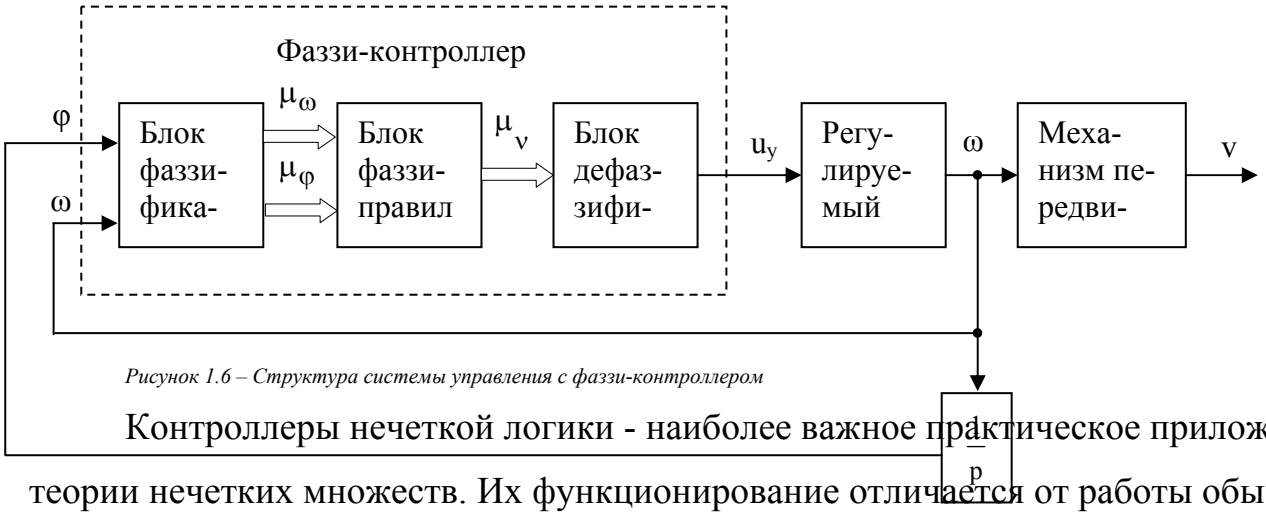


Рисунок 1.6 – Структура системы управления с фаззи-контроллером

Контроллеры нечеткой логики - наиболее важное практическое приложение теории нечетких множеств. Их функционирование отличается от работы обычных контроллеров.

Для создания алгоритма управления фаззи-контроллера вместо решения дифференциальных уравнений используются знания экспертов и логические правила обработки входных сигналов, представленных нечеткими множествами.

Знания экспертов могут быть выражены естественным образом с помощью лингвистических переменных, которыми описываются нечеткие множества.

Процесс *фаззификации* основан на создании и идентификации нечетких множеств в интервалах варьирования входных и выходных переменных фаззи-контроллера. Для этого диапазоны изменения входных и выходных переменных разбиваются на несколько поддиапазонов, которые представляют нечеткие множества:

- в области нуля;
- в области малых положительных и отрицательных изменений;
- в области средних положительных и отрицательных изменений (при необходимости);
- в области больших положительных и отрицательных изменений.

Соседние множества должны пересекаться. Тип функций принадлежности множеств выбирается в соответствии с таблицей 1.1.

На рисунке 1.7 приведены графики характеристических функций фаззи-множеств линейной скорости μ_{vi} .

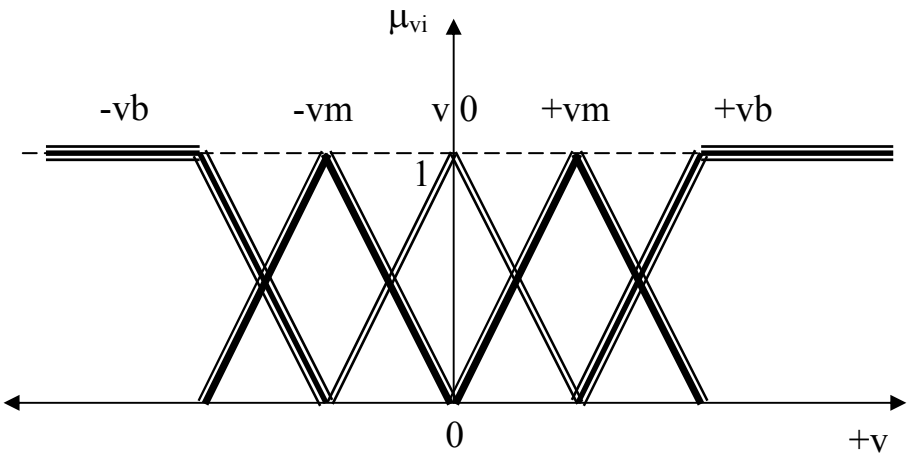


Рисунок 1.7 - Графики характеристических функций фаззи-множеств линейных скоростей μ_{vi}

На рисунке 1.8 приведены графики характеристических функций фаззи-множеств углов поворота маятника $\mu_{\varphi i}$.

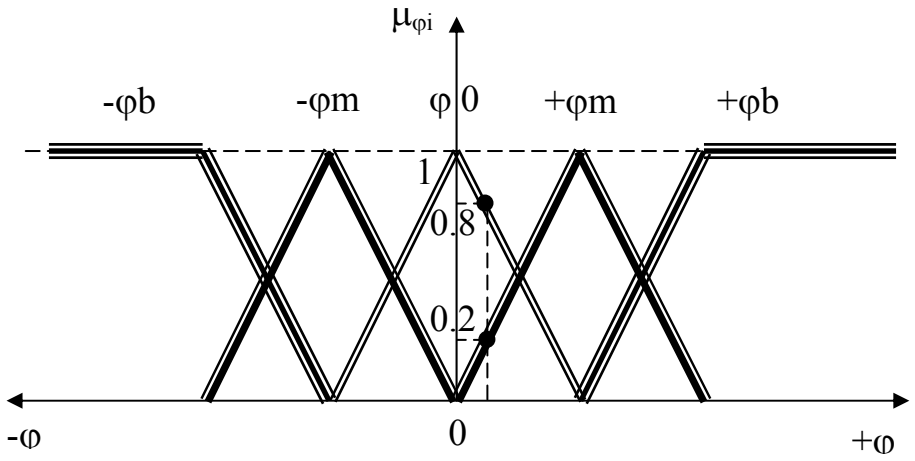


Рисунок 1.8 - Графики характеристических функций фаззи-множеств углов поворота маятника $\mu_{\varphi i}$

На рисунке 1.9 приведены графики характеристических функций фаззи-множеств скоростей поворота маятника $\mu_{\omega i}$.

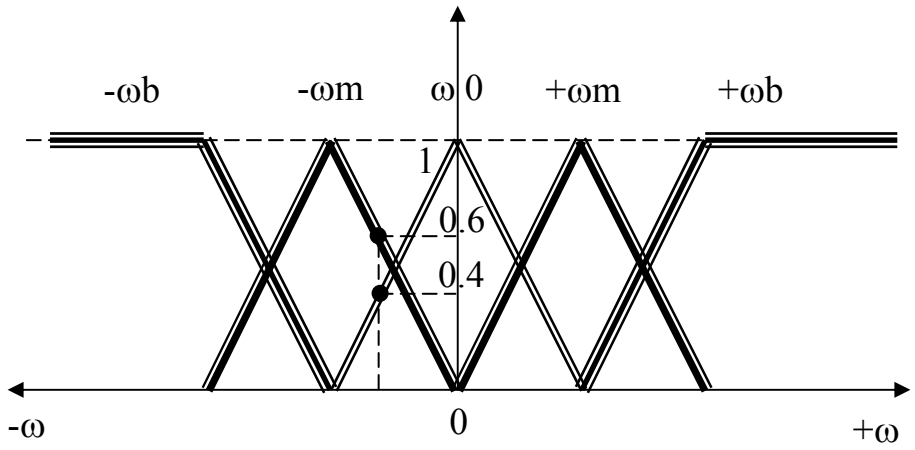


Рисунок 1.9 - Графики характеристических функций фаззи-множеств скоростей поворота маятника $\mu_{\omega i}$

Логическая обработка информации в фаззи-контроллере производится по специальным лингвистическим правилам, устанавливаемым программистом фаззи-контроллера на основе логических рассуждений.

Например, маятник находится в центре (угол равен нулю) и не поворачивается (угловая скорость равна нулю). Очевидно, что это желаемое положение и ничего предпринимать не надо (линейная скорость равна нулю).

Рассмотрим второй случай: маятник находится в центре, но движется с *малой* скоростью в *отрицательном* направлении. Для удержания маятника в центре необходимо перемещать тележку с *малой отрицательной* скоростью.

Рассмотрим третий случай: маятник находится справа, но движется с *малой* скоростью в *положительном* направлении. Чтобы вернуть маятник в вертикальное положение, необходимо тележку перемещать с *малой положительной* линейной скоростью.

Получаем три правила:

1 Если угол равен нулю И угловая скорость равна нулю, то линейная скорость должна быть равна нулю.

2 Если угол равен нулю И угловая скорость равна малой отрицательной, то линейная скорость должна быть равна малой отрицательной.

3 Если угол малый положительный И угловая скорость равна нулю, то линейная скорость должна быть равна малой положительной.

Аналогично устанавливаются другие правила, соответствующие всевозможным совпадениям входных фаззи-множеств. Число этих правил в приведенном примере равно числу сочетаний из десяти по два.

Правила сводятся в таблице 1.2.

Таблица 1.2 - Пример заполнения таблицы фаззи-правил

	Углы поворота					
Скорости поворота		-φb	-φm	φ0	+φm	+φb
	+ωb	v0	v0	+vb	+vb	+vb
	+ωm	v0	v0	+vm	+vm	+vb
	ω0	-vb	-vm	v0	+vm	+vb
	-ωm	-vb	-vm	-vm	v0	v0
	-ωb	-vb	-vb	-vb	v0	v0

Задачей создания алгоритма работы фаззи-контроллера по фаззи-правилам является преобразование качественного экспертного знания в закон регулирования, который может быть реализован технически.

Входными данными для этого процесса являются фаззи-правила и степени принадлежности, по которым устанавливается определенное логическое высказывание. На основе результатов фаззификации и фаззи-правил в блоке фаззи-правил происходит формирование выходного фаззи-множества.

Процесс обработки нечеткой информации в блоке фаззи-правил можно условно поделить на три этапа:

1) *агрегацию*;
2) *импликацию*;
3) *аккумуляцию*.

В процессе агрегации выбираются разные элементарные высказывания, которые принадлежат к одному фаззи-правилу, и определяется степень принадлежности к нему.

Каждая *условная часть* правила состоит из нескольких элементарных высказываний, которые с помощью логических связей сводятся к единому комплексному высказыванию.

Операторами *агрегации* могут быть любые из приведенных выше операторов, но в технике регулирования используется, как правило лишь И-оператор [1]. И-оператор в общем виде может быть реализован по критерию минимума*:*

$$E_i(\underline{x}) = \min\left\{E_{1,i}(x_1),...,E_{m,i}(x_m)\right\} = \min_{k=1}^{m} E_{k,i}(x_k), \qquad (1.5)$$

где $E_{k,i}$ – элементарные высказывания (фаззи-множества).

Процесс *агрегации* (принадлежности к первому правилу) для произвольно принятых численных значений входных переменных φ_1 и ω_1 поясняется рисунком 1.10.

В процессе *импликации* для каждого активного фаззи-правила в зависимости от численного значения степени принадлежности определяются фаззи-множества $\mathbf{V}_k$.

Для осуществления *импликации* используются операторы композиции. Часто в качестве оператора *импликации* в технике регулирования используется *И-оператор Мамдани* [1]. Графическое представление *импликации* с помощью *И-оператора* приведено на рисунке 1.11.

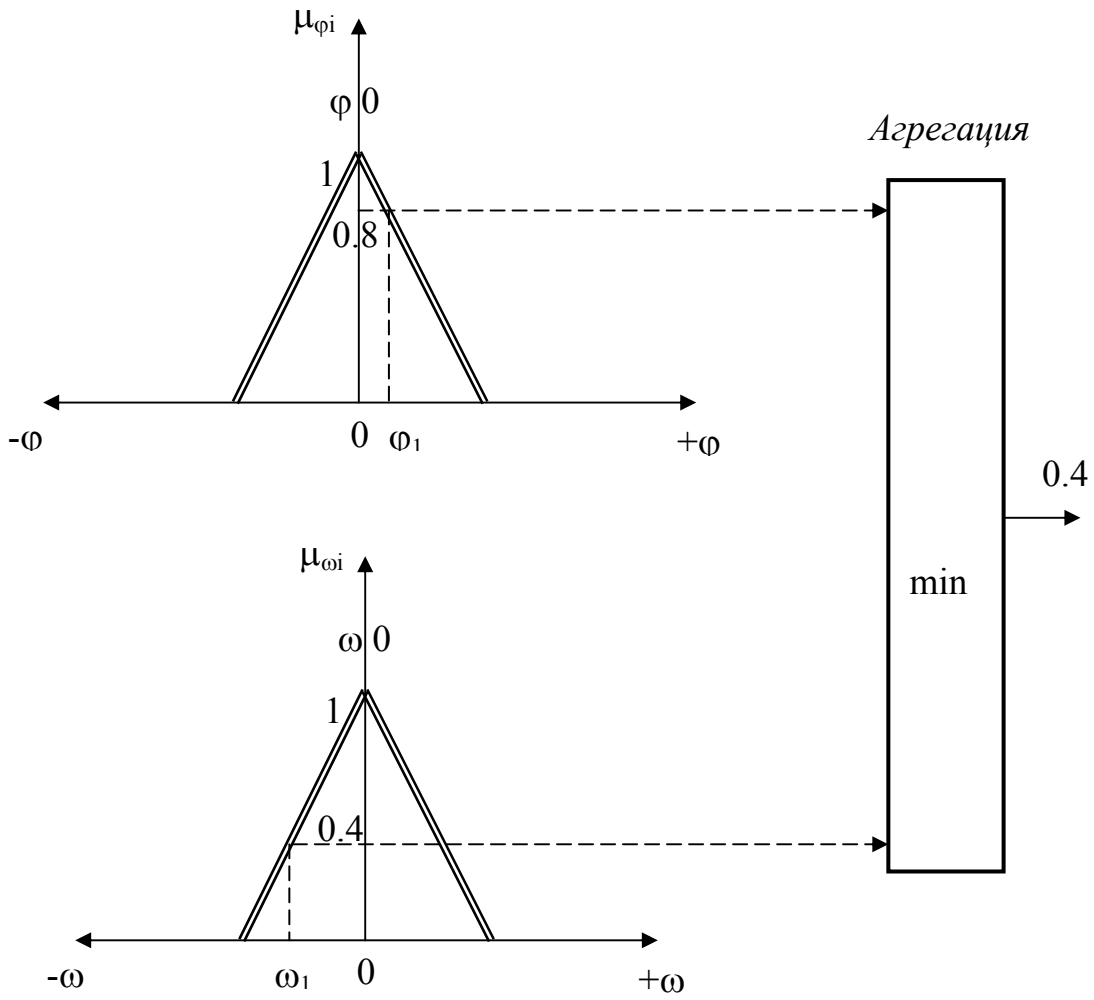


Рисунок 1.10 – Графики, поясняющие процесс агрегации

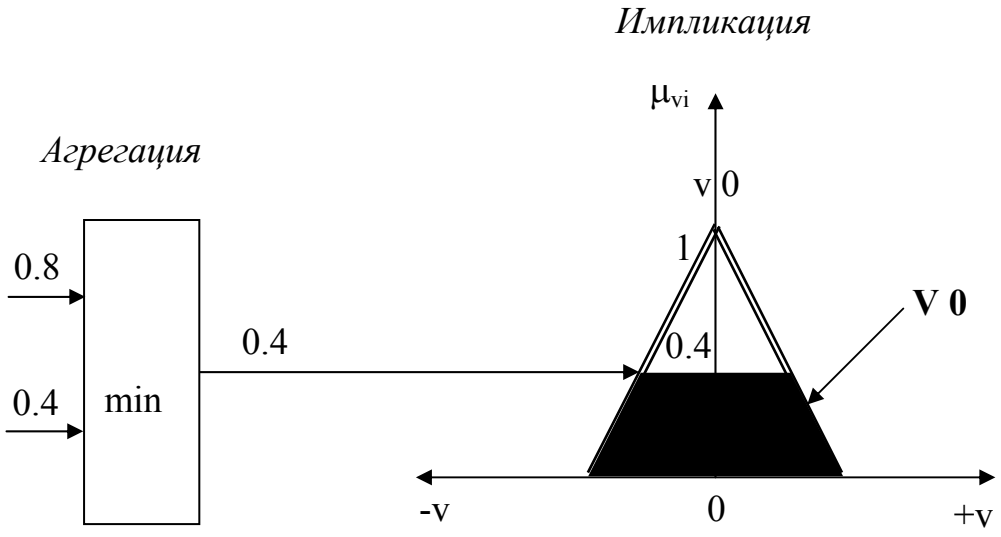


Рисунок 1.11 - Графическое представление импликации с помощью

И-оператора Мамдани

Результатом использования правила 2 для реальной угловой скорости ω_1 является импликация (рисунок 1.12).

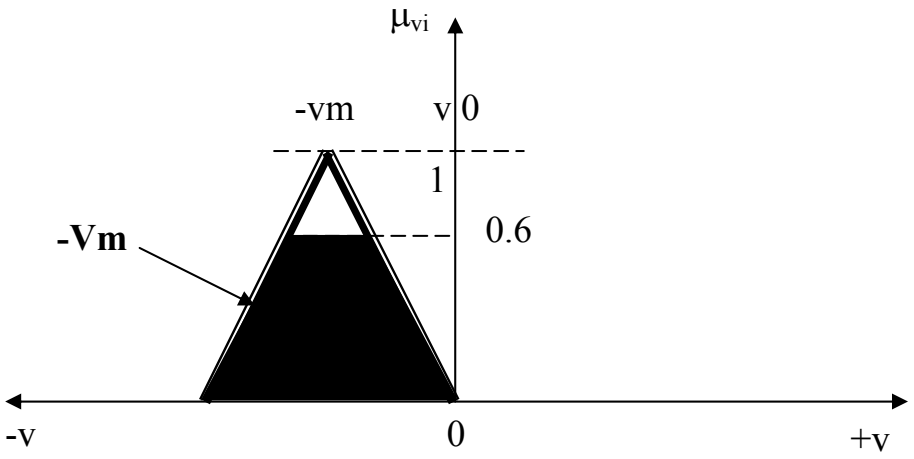


Рисунок 1.12 – Импликация в соответствии с фаззи-правилом 2

Результатом использования правила 3 для реальной угловой скорости ω_1 является импликация (рисунок 1.13).

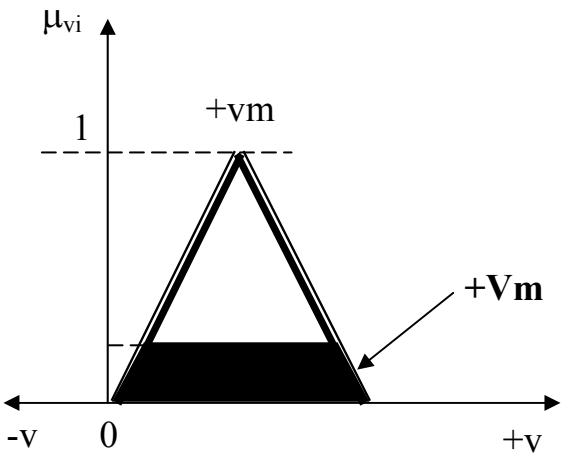


Рисунок 1.13 – Импликация в соответствии с фаззи-правилом 3

Последним этапом процесса обработки нечеткой информации в блоке фаззи-правил является *аккумуляция*.

Задачей аккумуляции является объединение результатов *импликации*.

Результатом *аккумуляции* является фаззи-множество $\mathbf{V_\Sigma}$, которое получается путем объединения отдельных фаззи-множеств: **V0, -Vm, +Vm** . Функция принадлежности результирующего фаззи-множества получается с помощью оператора аккумуляции *Akk*:

$$\mu_V(v) = \underset{i=1}{\overset{n}{Akk}}\{V0(v),-Vm(v),+Vm(v)\}. \qquad (1.6)$$

Аккумулированное фаззи-множество $\mathbf{V_\Sigma}$ представляет собой окончательный результат процесса обработки информации в блоке фаззи-правил.

Таким образом, результатом обработки нечетких множеств углов и угловых скоростей по правилам эксперта является нечеткое множество скорости $\mathbf{V_\Sigma}$, представленное на рисунке 1.14.

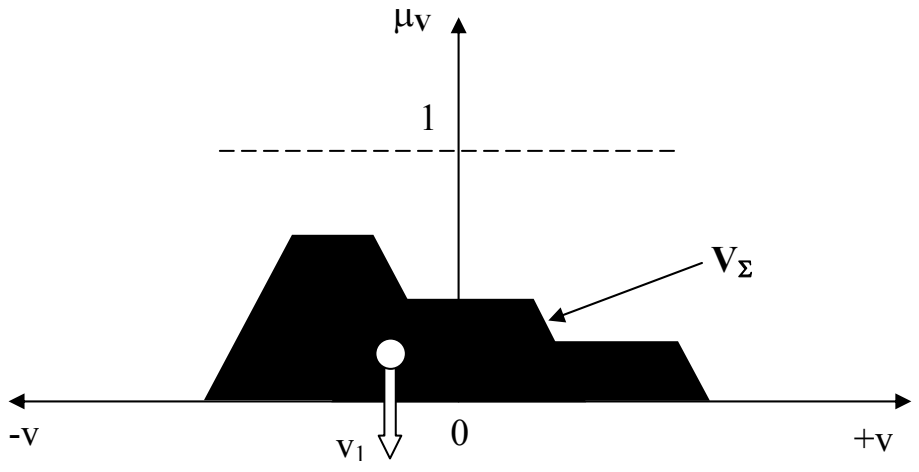


Рисунок 1.14 – График аккумулированного нечеткого множества $\boldsymbol{V_\Sigma}$

Аккумулированное фаззи-множество $\mathbf{V_\Sigma}$ не может быть использовано для управления, поэтому необходимо преобразовать его в однозначный управляющий сигнал.

Это преобразование выполняется в блоке *дефаззификации* (см. рисунок 1.6).

Существует большое количество методов *дефаззификации*, целью которых является определение такой однозначной величины, которая бы как можно точнее представила аккумулированное фаззи-множество.

Наиболее распространен гравитационный метод, который основан на определении центра веса площади функции принадлежности (см. рисунок 1.14).

Центрирующее значение линейной скорости перемещения тележки для реальных значений φ_1 и ω_1 определится в блоке *дефаззификации* по алгоритму

$$v_1 = \frac{\int v \cdot \mu(v) dv}{\int \mu(v) dv} \quad . \tag{1.7}$$

2 МОДЕЛИРОВАНИЕ СИСТЕМ ФАЗЗИ-РЕГУЛИРОВАНИЯ С ПОМОЩЬЮ ПРОГРАММНОГО ПАКЕТА MATLAB/SIMULINK

Важным этапом разработки систем управления электроприводов с фаззи-регулятором является их моделирование.

Многие проектировщики пользуются при моделировании программным пакетом Matlab и моделирующей программой Simulink, которая входит в его состав. Simulink дает возможность создавать наглядные математические модели систем регулирования и исследовать их функционирование.

Для моделирования фаззи-блоков в программном пакете Matlab/Simulink был разработан пакет Fuzzy Logic Toolbox, который представляет набор функций, созданных в среде Matlab. Пакет Fuzzy Logic Toolbox включает в себя средства создания и редактирования фаззи-блоков в границах структуры Matlab. Fuzzy Logic Toolbox дает возможность создавать фаззи-блоки трех типов [4]:

- систему инференции Мамдани (*Mamdani's fuzzy inference method*) – фаззи-блок, входные и выходные функции принадлежности которого могут иметь любую из приведенных в таблице 1.1 форму);
- систему инференции Сугено (*Sugeno's fuzzy inference method*) – фаззи-блок, входные и выходные функции принадлежности которого могут быть лишь постоянными или изменяющимися линейно);
- адаптивную нейро-фаззи-систему инференции (*adaptive neuro-fuzzy inference system*) – фаззи-блок, установка параметров функций принадлежности которого происходит на основе заданных желаемых значений входных и выходных величин фаззи-блока с помощью методов тренировки нейросети.

В данной работе рассматривается моделирование систем лишь первого типа.

Созданные с помощью Fuzzy Logic Toolbox фаззи-системы могут быть промоделированы программой Simulink. Недостатком Fuzzy Logic Toolbox является отсутствие возможности использования в системе инференции Мамдани функций

принадлежности с переменными параметрами. Моделировать в Matlab/Simulink такой адаптивный фаззи-блок можно, лишь написав функцию пользователя, которая содержит алгоритм обработки информации в среде Matlab или C, и поместив эту функцию Simulink-модели с помощью блока **Matlab Fcn** (функции Matlab) или **S-Function** (S-функции).

Средства, которые входят в состав Fuzzy Logic Toolbox, можно поделить на две группы. К первой группе относятся редактор фаззи-блоков (**FIS Editor**), редактор правил (**Rule Editor**) и редактор функций принадлежности (**Membership Function Editor**). Эти средства служат для установления входных и выходных переменных фаззи-блока, их функций принадлежности и фаззи-правил. Вторая группа представляет собой *read-only Tools*, то есть инструменты, которые не разрешают вносить изменения в фаззи-систему, а используются только для диагностики. К этой группе относятся просмотрщик трехмерной функции (**Surface Viewer**) и просмотрщик правил (**Rule Viewer**).

Для того, чтобы вызвать Fuzzy Logic Toolbox, необходимо набрать в командной строке Matlab *fuzzy* и нажать клавишу *Enter*. После этого открывается окно, вид которого представленный на рис. 2.1.

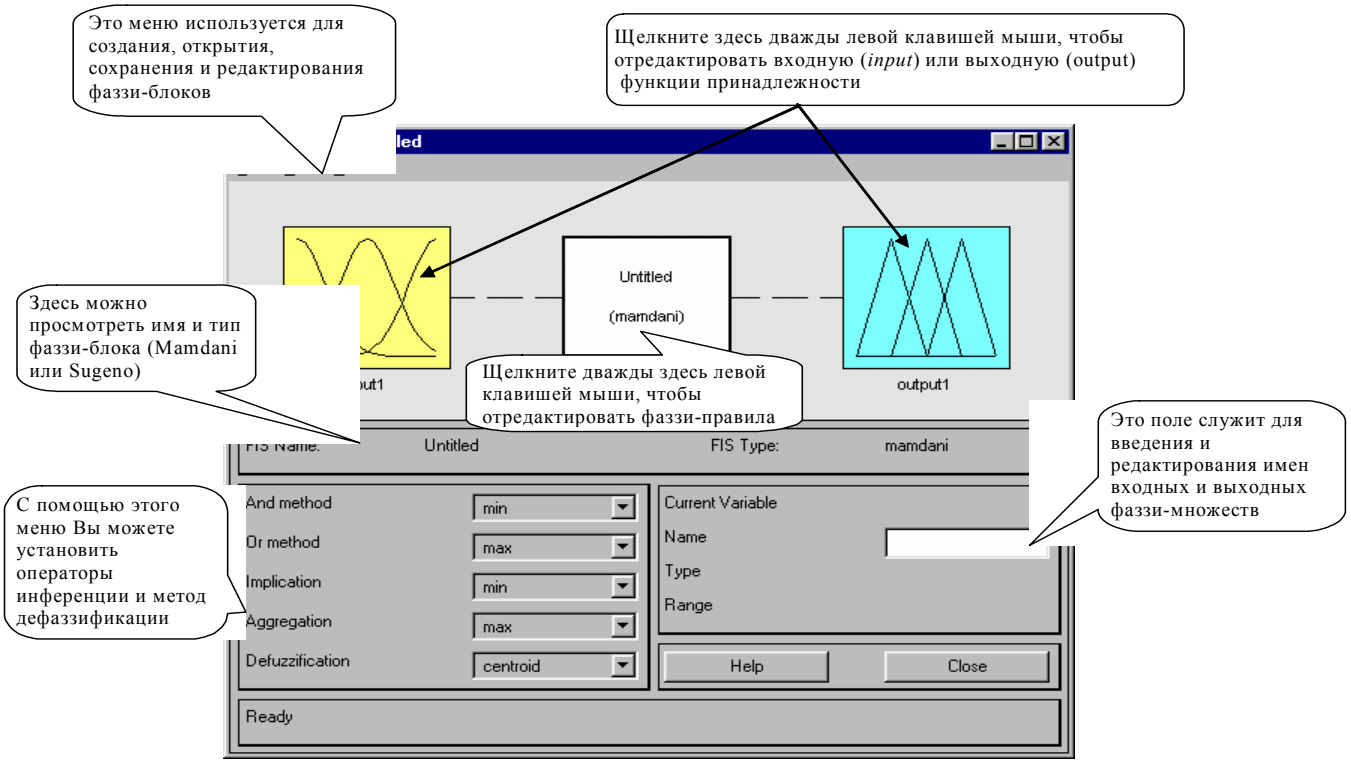


Рисунок 2.1 – Редактор фаззи-блоков

В этом окне Вы можете создать собственный фаззи-блок. В том случае, если фаззи-блок содержит более чем одну входную или выходную переменные, надо выбрать в меню соответственно **Edit→Add input** или **Edit→Add output**.

Чтобы отредактировать функции принадлежности, надо щелкнуть дважды левой клавишей мыши на иконке входной или выходной переменной или выбрать в меню **View→Edit Membershipsfunctions**.

После этого на экране появится окно редактора функций принадлежности (рис. 2.2), с помощью которого можно задать функции принадлежности определенной лингвистической переменной или отредактировать уже существующие.

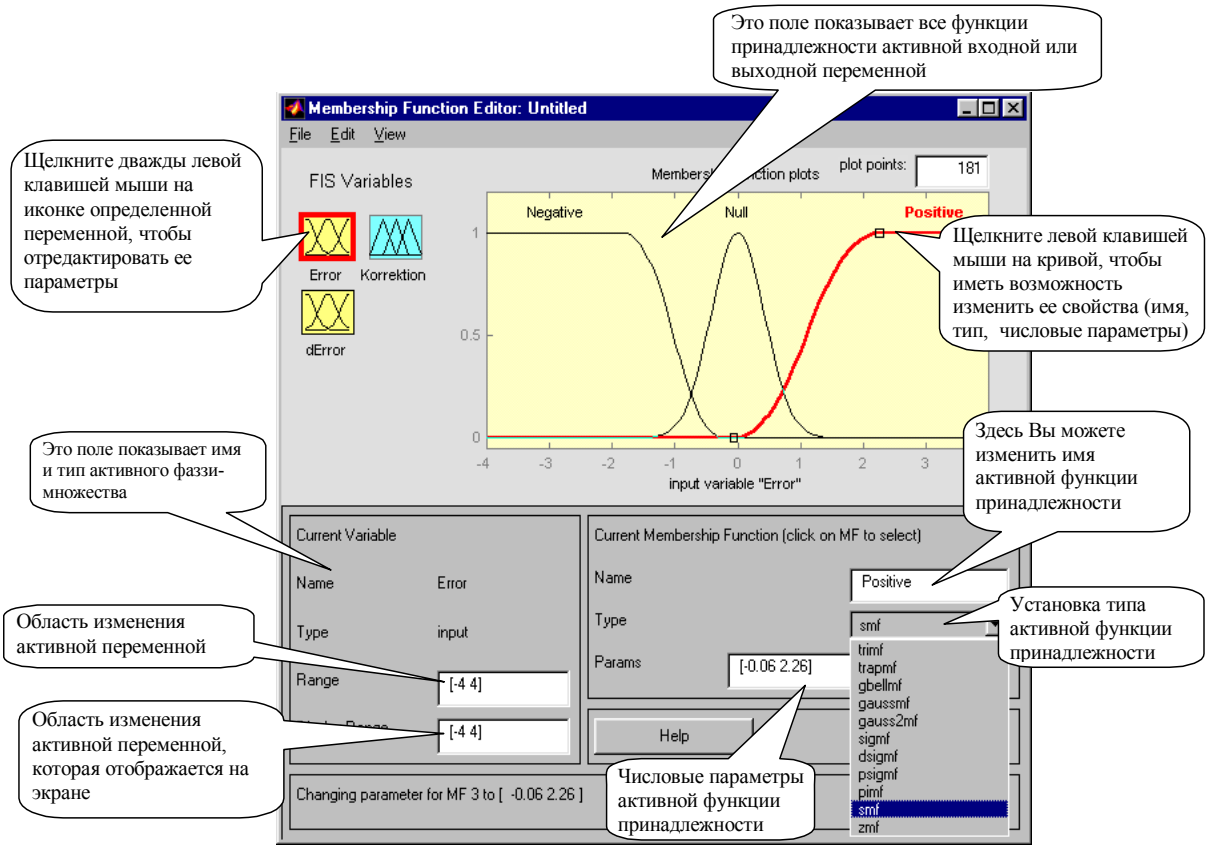


Рисунок 2.2 – Окно редактора функций принадлежности

Для того, чтобы ввести новую функцию принадлежности, необходимо выбрать в меню **Edit→Add MFs...** или **Edit→Add custom MFs...**. В первом случае открывается окно (рисунок 2.3):

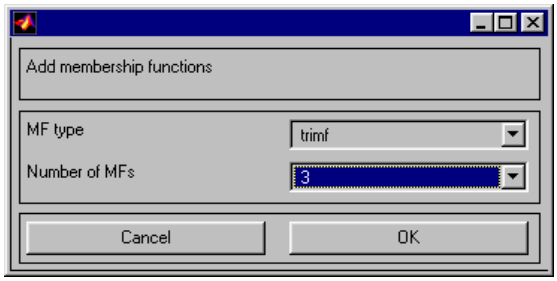


Рисунок 2.3 – Окно параметров стандартной функции принадлежности

Во втором случае открывается окно (рисунок 2.4):

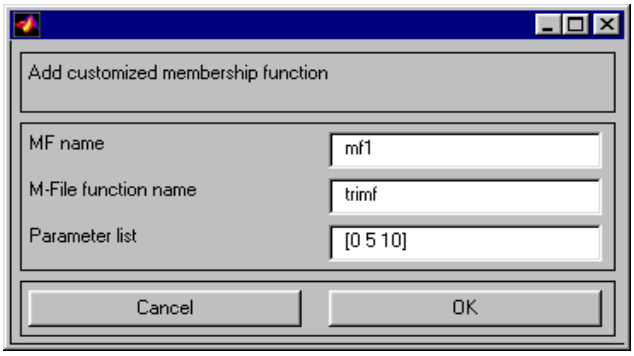


Рисунок 2.4 – Окно параметров функции принадлежности пользователя

В этом окне можно ввести любое имя , тип и границы функции принадлежности.

Для введения фаззи-правил используется редактор правил. Чтобы его открыть, надо выбрать в меню **View→Edit rules**. После этого открывается окно (рисунок 2.5). Складывание фаззи-правил проводится автоматически на основе описания входных и выходных лингвистических переменных, которое было сделано с

помощью редактора фаззи-блоков. Для того, чтобы образовать новое правило, надо нажать на кнопку **Add rule**, после чего выбрать значение каждой входной переменной, логическую связь между входными переменными и значение выходной переменной. В случае, если правило не содержит какой-то входной величины, надо избрать как входную величину **none**. Если переменная должна быть проинвертирована, надо поставить флажок напротив **not**.

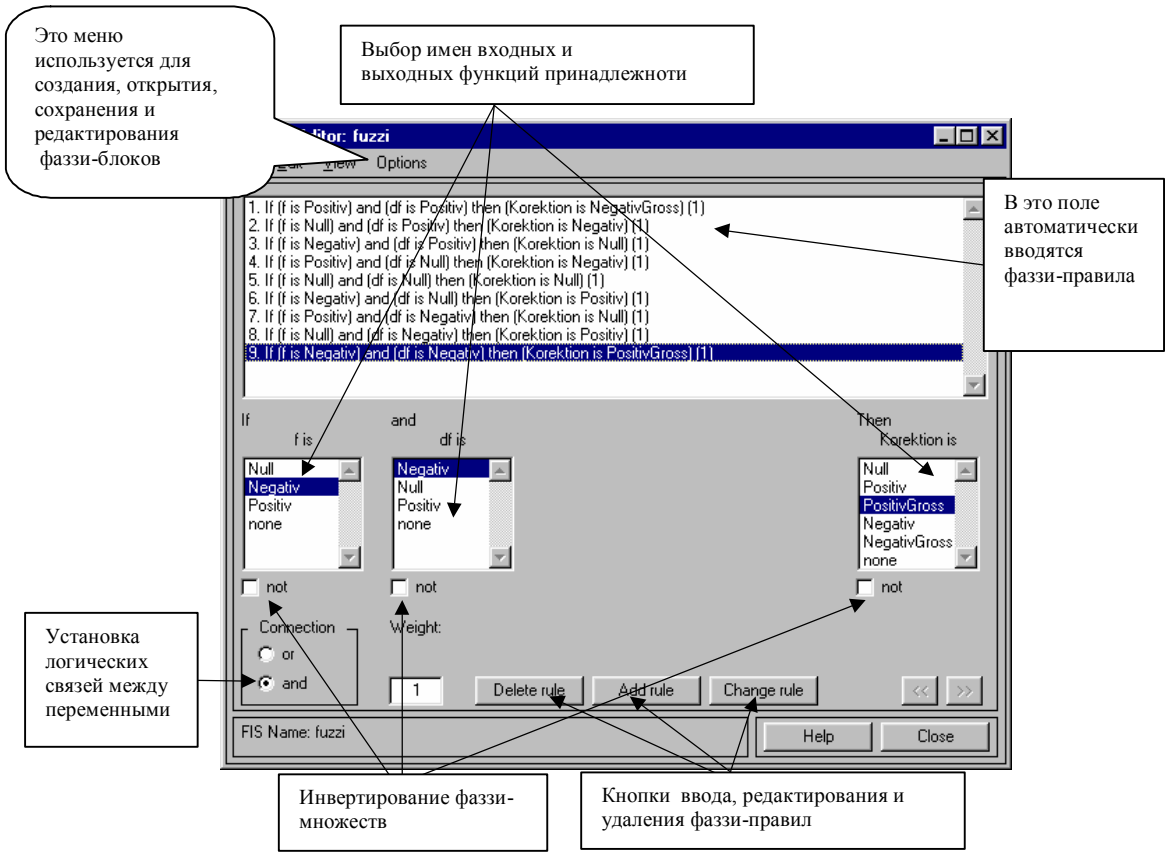


Рисунок 2.5 – Окно введения фаззи-правил

Диагностика разработанного фаззи-блока осуществляется с помощью просмотрщика трехмерной функции (**Surface Viewer**) и просмотрщика правил (**Rule Viewer**). Для того, чтобы вызвать окно просмотрщика правил, надо избрать в меню **View→View rules**, после чего на экране появится окно (рисунок 2.6):

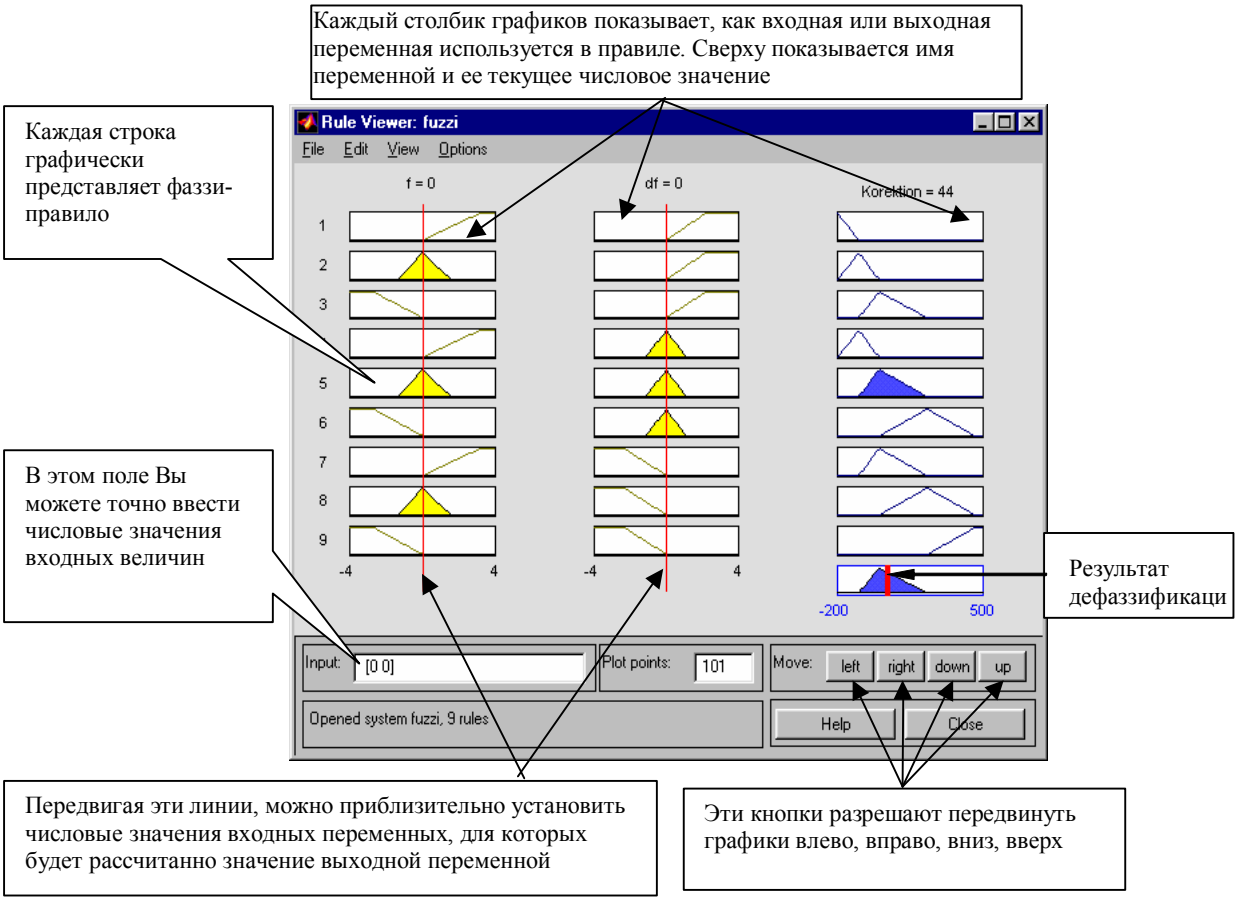


Рисунок 2.6 – Окно тестирования фаззи-правил

Просмотрщик правил дает возможность для любой комбинации числовых значений входных переменных определить соответствующее числовое значение выходной переменной.

Просмотрщик трехмерной функции дает возможность представить зависимость выходной переменной от входных в виде трехмерного графика. Это разрешает проанализировать работу фаззи-блоков, которые имеют две входные и одну выходную переменную. В случае, если фаззи-блок имеет более чем одну выход-

ную переменную или более чем две входные переменные можно избрать такую комбинацию двух входных и одной выходной переменной, которая разрешает наилучшим образом представить передающие свойства фаззи-блока. Для того, чтобы вызвать просмотрщик трехмерной функции, необходимо избрать в меню **View** команду **View surface**, после чего на экране появится окно (рисунок 2.7):

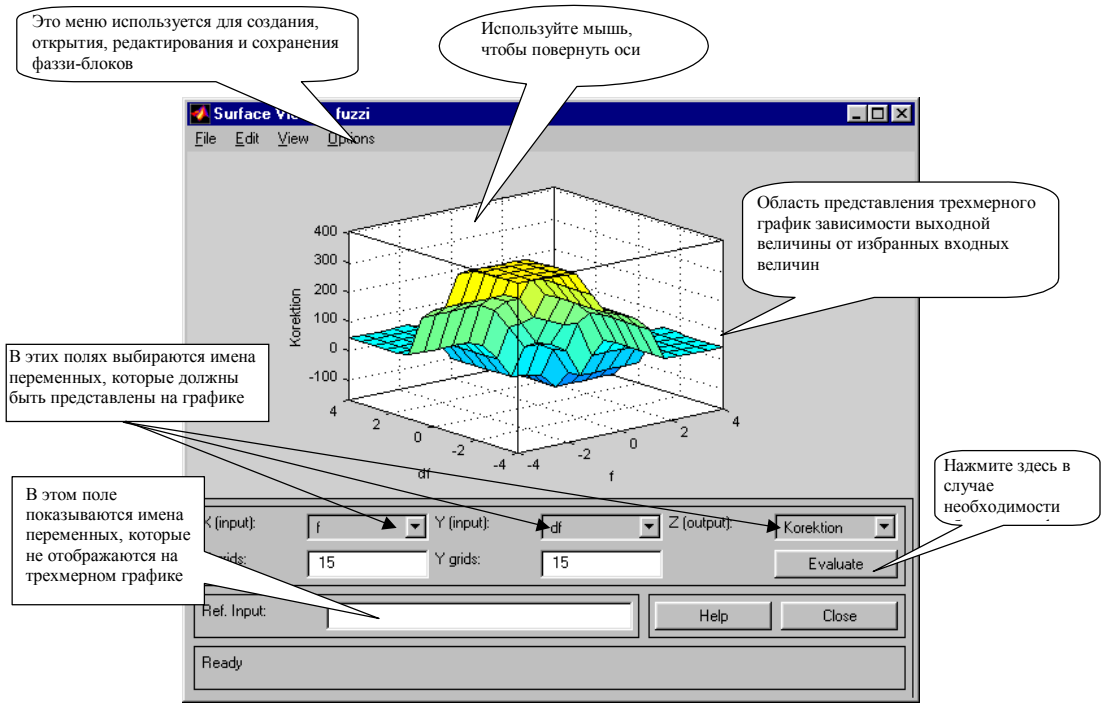


Рисунок 2.7 – Окно просмотрщика трехмерного графика, передающего функции фаззи-блока

Для просмотра установки фаззи-блока в более удобной форме, с возможностью копирования информации в буфер обмена данных для дальнейшего редактирования с помощью других программ, был разработан ряд функций вывода на экран (**System Display Functions**).

Обзор этих функций представлен в таблице 2.1.

Таблица 2.1 – Функции вывода установок фаззи-блока на экран

Имя и параметры функции	Описание функций
plotfis (fismat)	Создает в окне диаграмму фаззи-блока, который помещается в рабочем пространстве Matlab под именем *fismat*. На диаграмме показываются входные и выходные переменные, их вид, количество фаззи-правил и тип системы инференции
plotmf (fismat,varType,varIndex)	Выводит на экран график функции принадлежности лингвистической переменной фаззи-блока *fismat*. В списке параметров указываются тип переменной (*Input* или *Output*) и ее индекс
gensurf (fismat)	Выводит на экран трехмерный график передающей функции фаззи-блока *fismat*
getfis (fismat)	Выводит на экран установки фаззи-блока *fismat* – тип фаззи-блока, количество входных и выходных переменных, их имена, тип дефаззификации, тип операторов инференции

Для подключения разработанного фаззи-блока в Simulink-модель, которая содержит фаззи-регулятор, и моделирования этой системы регулирования, надо сначала убедиться, что фаззи-блок, созданный с помощью редактора фаззи-блоков, сохранен в рабочем пространстве Matlab под именем, которое используется при параметризации Simulink-блока **Fuzzy-Logic-Controller**

Для сохранения параметров созданного фаззи-блока в рабочем пространстве Matlab надо избрать в меню редактора фаззи-блоков **File,** включить команды **Save to workspace** и **Save to workspace as**. В случае, если фаззи-блок был спроек-

тирован и сохранен на жестком диске или другом элементе памяти раньше, эту операцию можно выполнить непосредственно в рабочем пространстве Matlab. Для этого в командной строке Matlab необходимо набрать команду: *fuz = readfis('Controller'),* где *fuz* – имя FIS-матрицы, которая отображает фаззи-блок в рабочем пространстве Matlab;

Controller - имя файла разработанного фаззи-блока.

После этого надо открыть Simulink-библиотеку **fuzblock**. В результате открывается окно (рисунок 2.8):

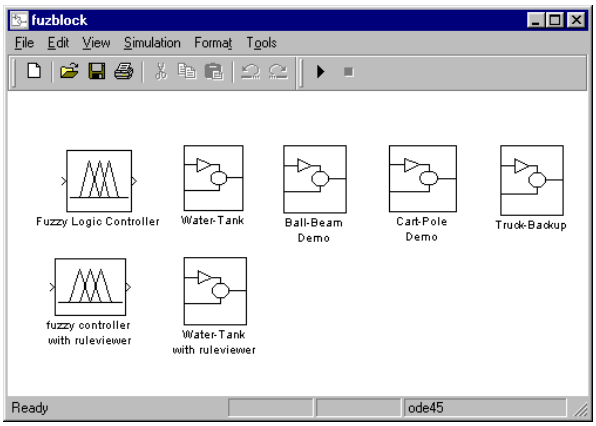


Рисунок 2.8 – Simulink-библиотека фаззи-блоков

Из этой библиотеки в Simulink-модель методом «перетаскивания» копируется один из таких блоков:

- Fuzzy Logic Controller;

- Fuzzy Controller with ruleviewer.

Использование блока **Fuzzy Controller with ruleviewer** дает возможность в процессе моделирования наблюдать за изменениями входных и выходных величин фаззи-блока с помощью просмотрщика правил.

Избранный блок необходимо идентифицировать. Для этого надо дважды щелкнуть левой клавишей мыши на иконке блока на поле Simulink – модели сис-

темы управления. Если используется блок **Fuzzy Logic Controller**, то после этого на экране появится окно (рисунок 2.9):

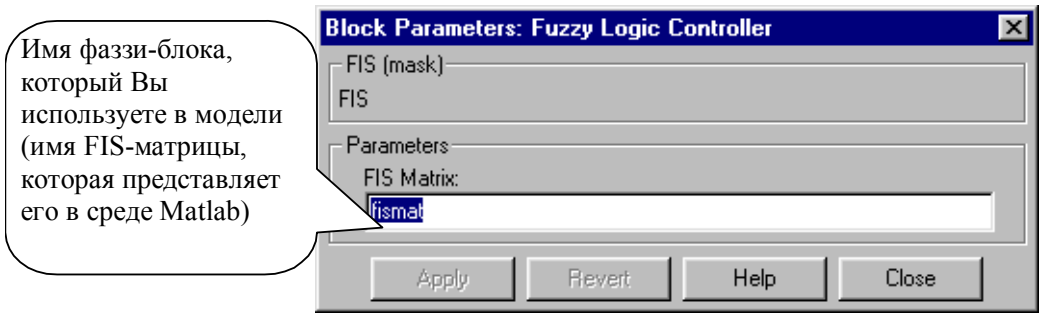


Рисунок 2.9 – Окно параметров блока ***Fuzzy Logic Controller***

Блок **Fuzzy Controller with ruleviewer** имеет довольно сложную структуру, которая представлена на рисунке 2.10.

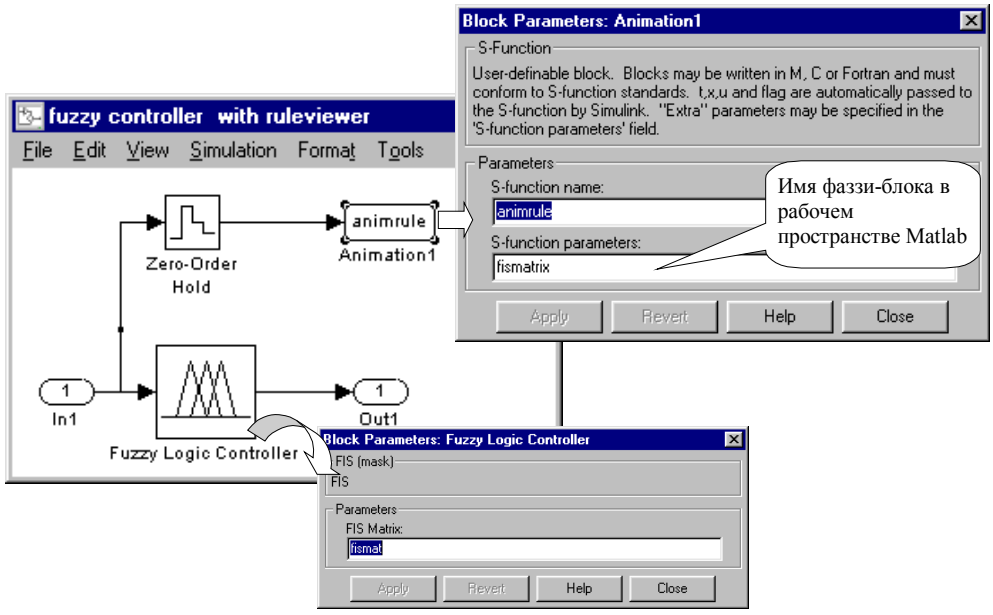


Рисунок 2.10 – Структура блока ***Fuzzy Controller with ruleviewer***

Параметризация блока **Fuzzy Controller with ruleviewer** осуществляется, как показано на рисунке 2.10.

3 СИСТЕМА УПРАВЛЕНИЯ СЛЕДЯЩИМ ЭЛЕКТРОПРИВОДОМ (СЭП) С ФАЗЗИ-КОНТРОЛЛЕРОМ

Система управления СЭП должна обеспечивать заданную точность регулирования положения рабочего органа без перерегулирования.

Функциональная схема трехконтурной САР СЭП [5] приведена на рисунке 3.1.

Функциональная схема включает:

- силовой трансформатор TV;

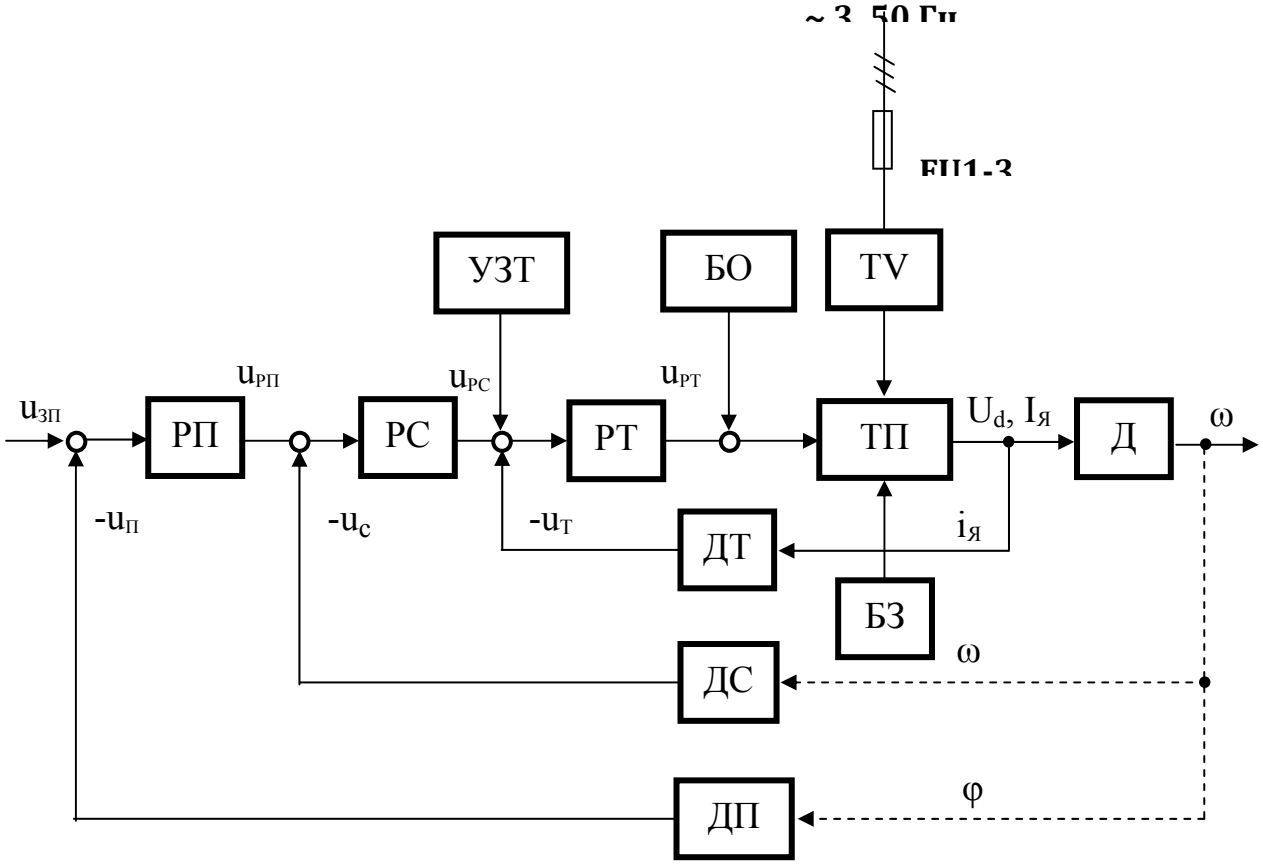


Рисунок 3.1 - Функциональная схема системы регулирования положения

- тиристорный преобразователь ТП;
- электродвигатель постоянного тока Д;
- регулятор тока РТ;
- регулятор скорости РС;
- регулятор положения РП;
- датчик тока ДТ;
- датчик скорости ДС;
- датчик положения ДП;
- узел зависимого токоограничения УЗТ;
- блок ограничения минимального угла управления БО;
- блок защит БЗ.

Структурная схема системы регулирования положения, разработанная на основе функциональной схемы, приведена на рисунке 3.2.

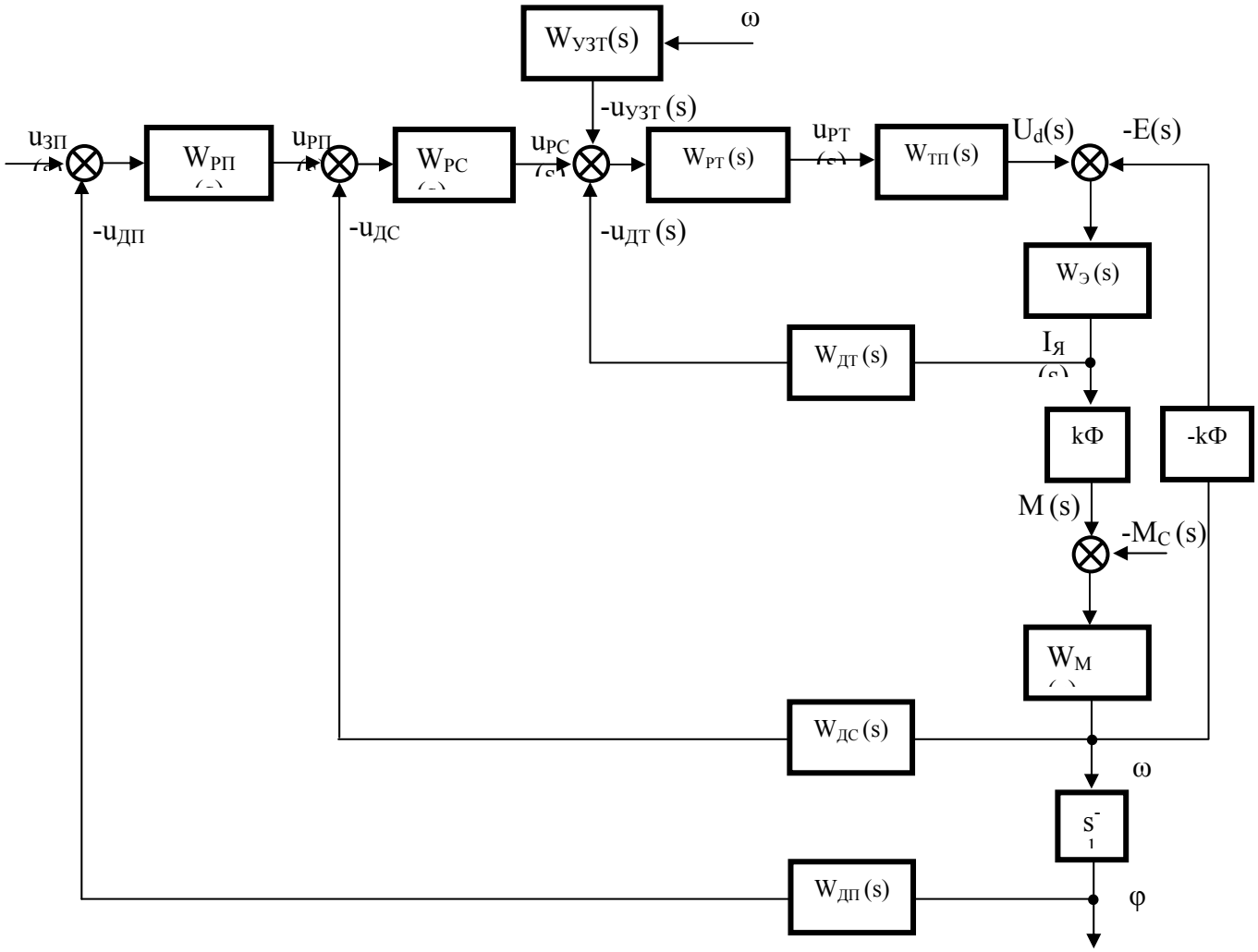


Рисунок 3.2 - Структурная схема системы регулирования положения

На рисунке 3.3 в качестве примера приведена структурная схема модели следящего электропривода в MatLab Simulink.

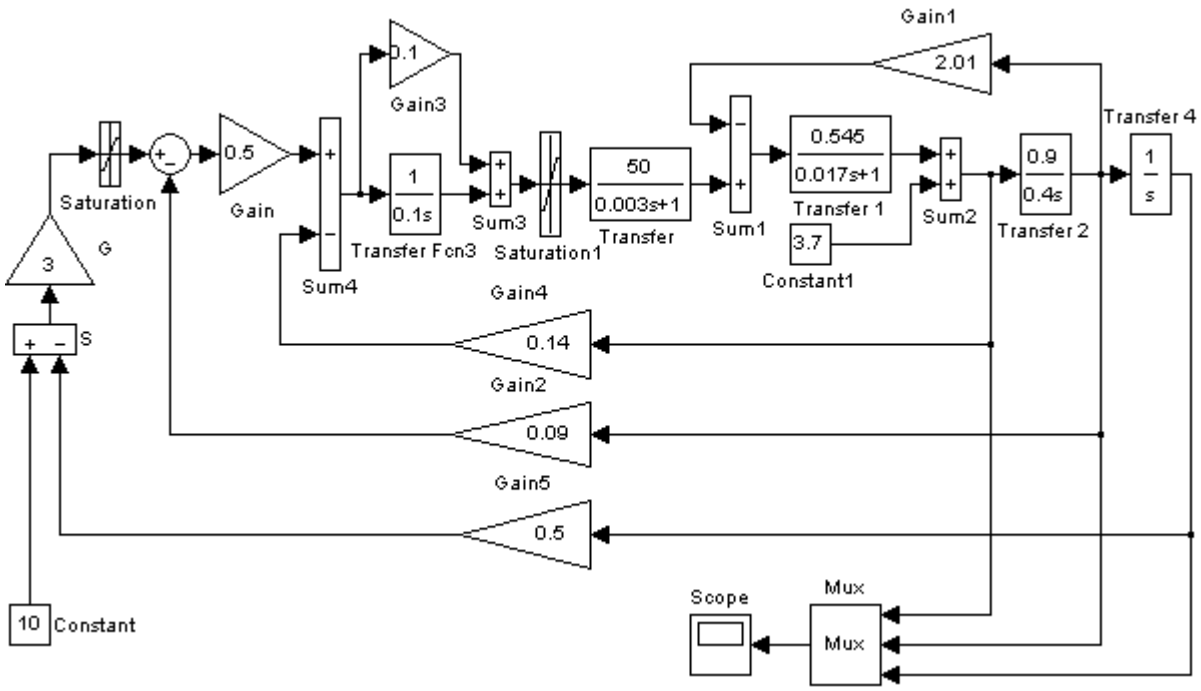


Рисунок 3.3 – Схема структурной модели СЭП в MatLab Simulink

В зависимости от значений коэффициентов передачи САР по прямому каналу управления график переходного процесса изменения угла поворота φ(t) может иметь разный вид.

Время регулирования СЭП зависит от статического коэффициента передачи регулятора положения. Увеличить быстродействие системы можно за счет его увеличения.

В качестве иллюстрации на рисунке 3.4 приведены графики переходного процесса i(t), ω(t), φ(t) для исходного (увеличенного) значения коэффициента передачи регулятора положения.

На рисунке 3.4 видно, что имеет место перерегулирование основной координаты – положения, это недопустимо для систем регулирования положения.

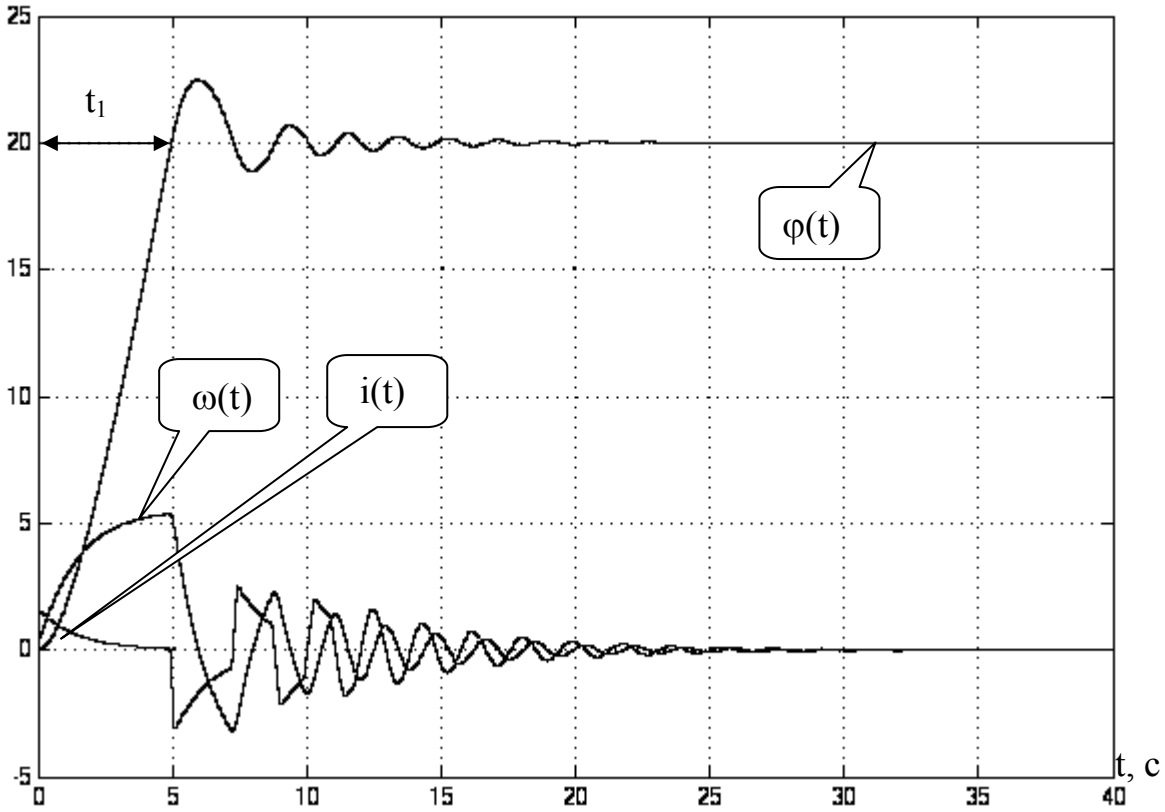


Рисунок 3.4 – Графики переходных функций i(t), ω(t), φ(t) исходной системы управления СЭП

На рисунке 3.5 приведены графики переходного процесса i(t), ω(t), φ(t) после коррекции коэффициента передачи регулятора положения ($k_п = 0.04$).

Переходный процесс φ(t) на рисунке 3.5 показывает, что при малых коэффициентах передачи регулятора положения перерегулирование отсутствует, но время регулирования достигает 25 с. При больших коэффициентах передачи регулятора положения время первого установления задания уменьшается в несколько раз, но возникает перерегулирование, которое недопустимо (см. рисунок 3.4).

Идея применения фаззи-контроллера состоит в ускорении времени регулирования СЭП.

Задачей, которую должен решать фаззи-контроллер, является исключение динамической ошибки регулирования при обеспечении заданного быстродействия.

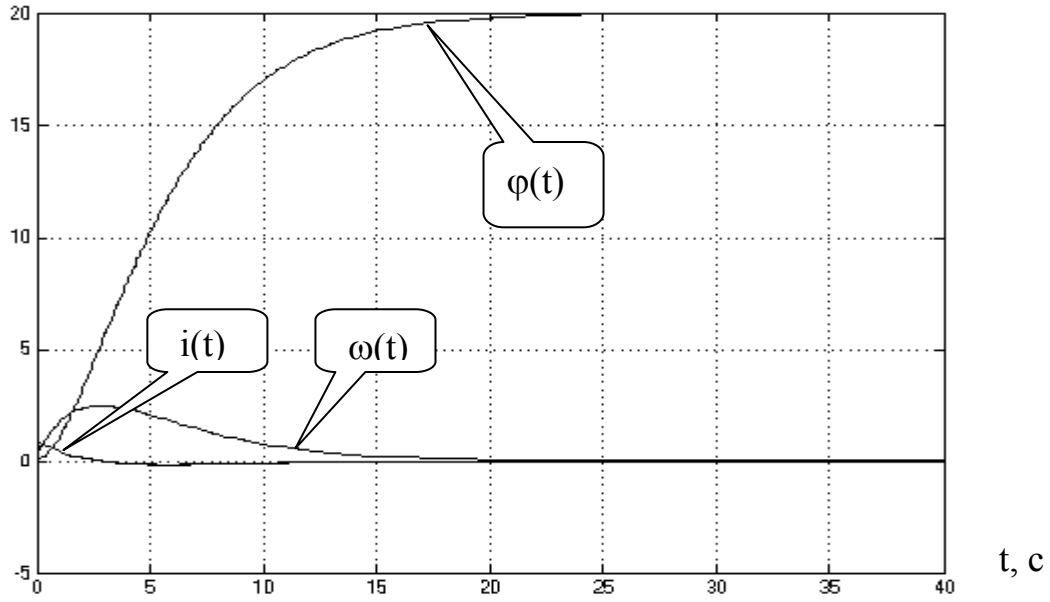


Рисунок 3.5 – Графики переходных функций i(t), ω(t), φ(t) СУЭП после коррекции k_n

При этом фаззи-контроллер должен обеспечивать время регулирования, равное времени t_1 первого достижения системой управления задания $\varphi_з$ (см. рисунок 3.4).

Целью параллельной коррекции с помощью фаззи-контроллера является получение заданной переходной функции $\varphi_з(t)$, общий вид которой приведен на рисунке 3.6.

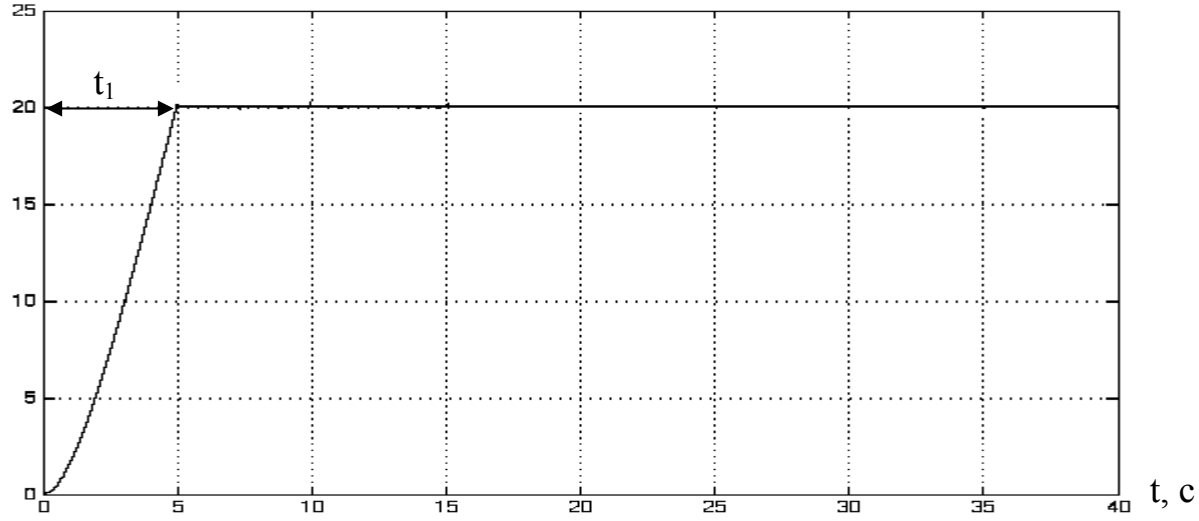


Рисунок 3.6 – График желаемого переходного процесса $\varphi_з(t)$ СУЭП после фаззи-коррекции

Подключение фаззи-контроллера в режиме параллельной коррекции регулятора положения показано на рисунке 3.7.

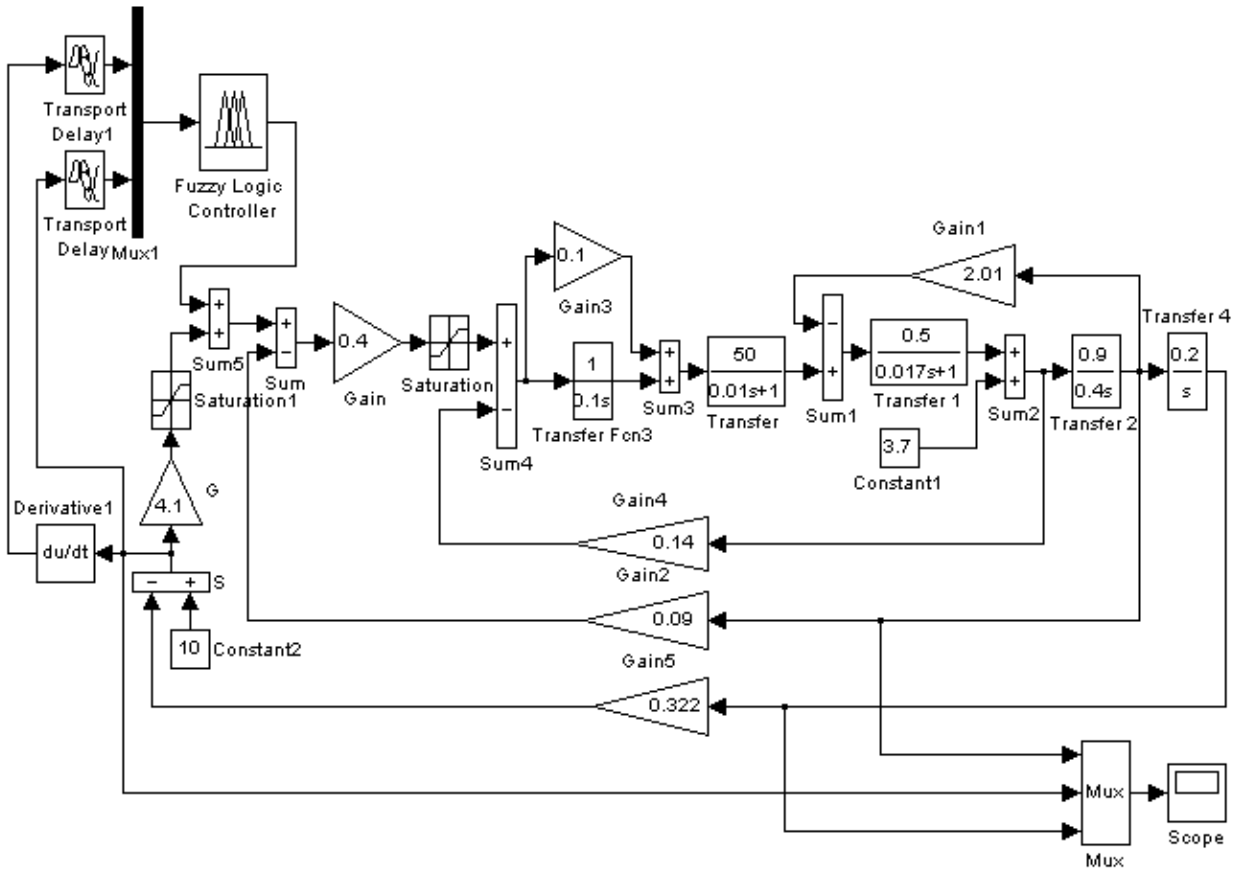


Рисунок 3.7 – Схема модели СЭП с фаззи-коррекцией регулятора положения