## 1. Введение

Классический подход к строительству систем управления основан на физическом моделировании. Явно выделяют три метода построение модели.

- На основе эксперимента, наблюдая за процессом, и определяя его поведение в зависимости от входных значений, реально получить некую таблицу зависимости выходов от входов. Графически, такой способ эквивалентен нанесению дискретных точек на кривую входа-выхода, где на горизонтальной оси наносятся значения входов, а на вертикальной оси наносятся значения выходов. Поняв, как происходит такое взаимодействие, возможно построить управляющий контроллер. Недостатки такого способа очевидны: оборудования, доступного для экспериментов, может не оказаться; сама постановка эксперимента может «влететь в копеечку»; для значительнного количества входов и выходов метод окажется практически бесполезным; подходящих приборов для проведения измерений может не оказаться, или они вообще могут не существовать в природе.
- Классическая теория управления предполагает наличие некой идеализированной математической модели процесса, обычно в виде дифференциальных уравнений или разностных выражений. Для их анализа соответственно пользуются преобразованиями Лапласа и Z-преобразованием. Для того, чтобы получить математическую модель процесса, простую и годную для анализа, допускают некоторые упрощения, например, такие как линейность: полагают, что выход процесса всегда пропорционален входу. Линейный аппарат в этом случае весьма ценен, в силу хорошей изученности, простоты, и возможности получить вполне годную схему для анализа системы управления в целом. В добавок, в настоящее время не существует общей теории (по крайней мере, автору не известна) аналитического решения нелинейных дифференциальных уравнений, а потому не существует и достаточно полных средств для анализа нелинейных систем. Допускают обычно ещё одно упрощение, которое подразумевает, что параметры процесса не изменяются во времени, несмотря на то, что в реальных условиях параметры процесса, так или иначе, подвержены вариациям в силу воздействия на них внешней среды.
- Эвристические методы основаны на практическом опыте и неких производственных традициях и мастерстве. Эвристическое правило обычно имеет форму «если <условие> то <вывод>», а для типичного случая управления процессом «если <условие> то <действие>». Правила задают соответствие условий — выводам, таким образом, эвристические методы сходны с экспериментальным методом конструирования таблицы входов и соответствующих им выходов, но, вместо конкретных и чётких значений переменных, в них используются нечёткие лингвистические характеристики.

## 2. Нечёткие множества

Логика нечётких множеств пытается моделировать процесс принятия решений человеком на основании неполных или двусмысленных данных. В отличие от классической логики, полагающейся на точные значения или бинарные «истина-ложь», нечёткая логика способна обрабатывать неполные данные и даёт приблизительное решение для задач, которые сложно (или невозможно) решить другими методами. К сожалению, в теории нечётких множеств используется своеобразная терминология, часто не имеющая аналогов в других областях математики и техники. На сегодняшний день общего соглашения по использованию терминологии, к большому сожалению, не существует, что несколько осложняет освоение теории и вносит путаницу.

Вероятностная логика и нечёткая логика математически сходны: обе выполняют действия со значениями истинности, в области значений 0 и 1. Однако же интерпретация

их сущности различна: так нечёткая логика отражает «степень правдоподобности», а вероятностная логика отражает «вероятность, возможность события».

Переменные в математике, как правило, принимают некоторые числовые значения. В отличие от последних, в нечёткой логике обращаются с *лингвистическими переменными*, которые позволяют давать описания условий правил и выводов в «туманной» форме, близкой к естественной форме языка. Лингвистическая переменная, например «температура», может иметь значение такое как «холодно», «тепло», «жарко» и т. д. Лингвистические переменные изменяются под воздействием лингвистических модификаторов.

### 3. Введение в управление на основе нечёткой логики

Классический контроллер с отрицательной обратной связью показан на Рис. 1. Выход процесса $y(t)$ снимается датчиком или измерительным устройством, и по цепи обратной связи вычитается из опорного сигнала $r(t)$. Контроллер получает на вход ошибку между установившимся значением выхода процесса и желаемым значением процесса $e(t) = r(t) - y(t)$. Получив ошибку, контроллер вычисляет управляющий сигнал $u(t)$, и подаёт его через преобразователь на вход процессу.

Управление с нечёткой логикой это метод управления, основывающийся на нечёткой логике. Нечёткая логика, в отличие от обычной, ведёт счёт не с числами, а со словами. Управление с нечёткой логикой это управление на основе предложений, а не формул.

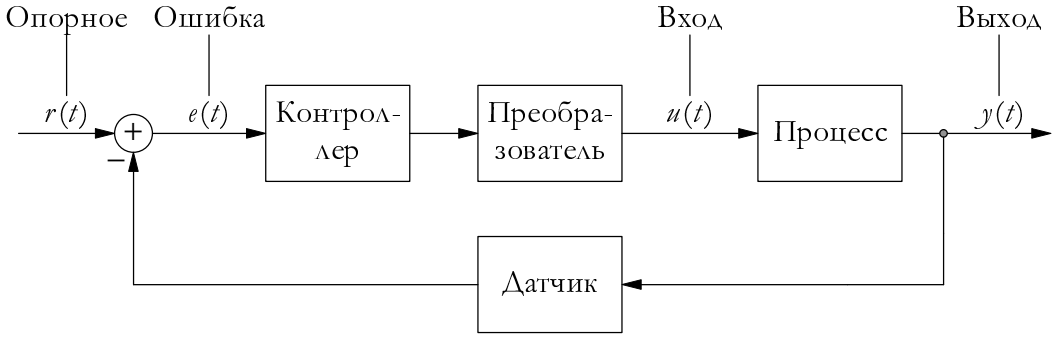


Рис. 1

Как правило, контроллер на нечёткой логике включает в себя эмпирические, т. е. полученные на основе опыта, правила, на основании которых и осуществляется управление. Например, типовой fuzzy-контроллер может содержать следующие правила:

1. Если error есть Negative и change-in-error есть Negative, то output есть Zero
2. Если error есть Negative и change-in-error есть Zero, то output есть NM
3. Если error есть Negative и change-in-error есть Positive, то output есть NH

...

Этот набор правил называется *базой правил*. Правила задаются в привычном виде «если <условие> то <вывод>». Левая часть правила называется *условием*, а правая часть называется *выводом*. В правилах никогда не бывает ветвления вроде «если ... то ... иначе ...», они всегда состоят только из условия и вывода. Контроллер должен выполнить вычисления условий в каждом правиле из базы. Значение Negative в условиях правил это лингвистический термин, обозначающий отрицательное значение входного сигнала. Выходные значения NM и NH соответственно сокращения от «Negative Medium» и «Negative High».

В данном случае, задачей контроллера является вычисление условий правил и выдача сигнала управления на основании входных сигналов error (ошибка) и change-in-error (изменение ошибки). Перечислим особенности контроллеров, управляющих процессом на основе нечёткой логики:

- база правил хранится в более-менее естественной языковой форме;
- в отличие от классических контроллеров, формульного описания в базе не содержится;
- база правил в большей степени понятна неспециалистам в области теории управления;
- пользователю легче самому добавлять и изменять правила, конструировать новые, основываясь на эмпирическом заключении.

Управление с нечёткой логикой может осуществляться как по схеме *прямого управления* так и по схеме *управления с упреждением*. Последняя, в случае классического контроллера, потребовала бы хорошей модели процесса, которую часто получить весьма затруднительно, дорого или вообще невозможно. В этом случае может быть использована нечёткая модель процесса, основывающаяся на эмпирических правилах.

Стабильность систем управления на основе нечёткой логики остаётся открытым вопросом. Для классических контроллеров, например ПИД-регуляторов, существуют методы вроде частотного анализа, которые дают оценку стабильности управления линейной модели «регулятор+процесс» по его амплитудно-фазовой частотной характеристике (АФЧХ), или же аналитически, по полюсам передаточной функции. Контроллеры с нечёткой логикой обладают значительной нелинейностью, поэтому классические методы анализа стабильности линейных систем к ним напрямую неприменяемы. Через аппроксимации кусочно-линейными функциями можно получить линейные характеристики таких контроллеров, и анализировать устойчивость на кусочных участках.

Большую практическую ценность несут в себе источники знаний для заполнения базы правил. На основании опыта, можно выделить несколько таких источников.

- Опыт экспертов отрасли и знания из практики теории управления. Для составления базы правил таким способом опрашивают экспертов или операторов, имея на руках тщательно составленный вопросник.
- Действия операторов, регулирующих процесс. Нечёткие правила, состоящие из пар «если <...> то <...>» могут быть установлены на основании наблюдений за операторам и анализе журналов процесса, где записываются действия оператора и параметры сигналов.
- Нечёткая модель процесса, fuzzy-модель. Лингвистические правила в базе могут рассматриваться как модель, обратная модели процесса. Таким образом, правила нечёткой логики для управления могут быть получены инверсией fuzzy-модели процесса. Самообучающаяся система. Такой контроллер возможно сконструировать на основе нейронных сетей.

## 4. Структура контроллера с нечёткой логикой

В типовом контроллере с нечёткой логикой (Рис. 2) обычно выделяют несколько частей:

- Предварительная обработка (Preprocessing)
- Подготовка задачи для решения нечёткой логикой (Fuzzification)

- Машина логического вывода и база правил (Inference Engine + Rule Base)
- Получение решения задачи нечёткой логикой (Defuzzification)
- Заключительная обработка (postprocessing)

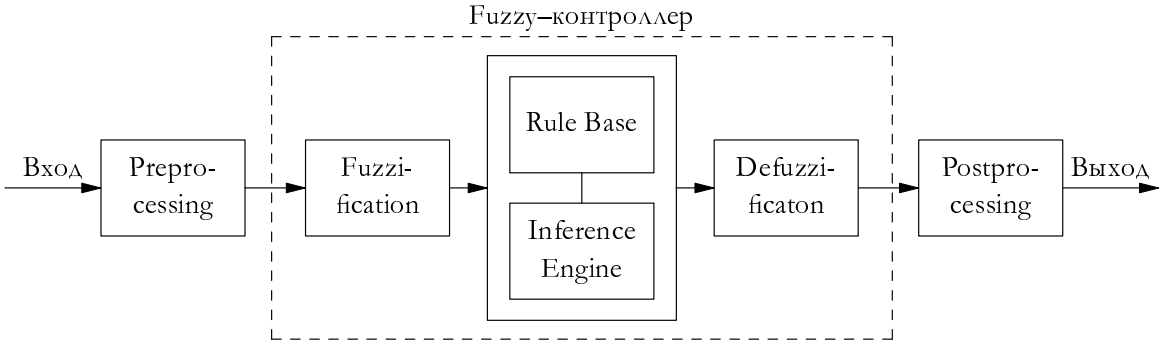


Рис. 2

### 4.1. Предварительная обработка

Предварительная обработка включает в себя:

- квантование и семплирование измерений;
- нормализацию и масштабирование;
- фильтрацию и удаление шумов;
- усреднение значений;
- дифференцирование и интегрирование дискретных величин.

### 4.2. Подготовка задачи для решения нечёткой логикой

Каждое из входных значений проходит стадию подготовки. Для этого с помощью *функций принадлежности* выполняется поиск для определения того, насколько полно условие в каждом из правил соответствует входным значениям. Для каждого лингвистического термина, связанного со входным значением, назначается вес, определяющий степень сходства входного значения с ним.

### 4.3. База правил

Правила в базе могут иметь несколько входов (или входных переменных) и несколько выходов (выходных переменных). Таким образом, могут решаться как проблемы SISO (single-input-single-output), так и MIMO (multiple-input-multiple-output). Типовая задача SISO — управление процессом, на основе значений ошибки, а также её производной и интеграла. Все примеры и правила в дальнейшем будут рассматриваться на примере контроллера с нечёткой логикой, изображённого на Рис. 3. Контроллер получает на вход два сигнала: сигнал ошибки и её производную.

#### 4.3.1. Формат правил

Обычно контроллер содержит правила в формате если-то, хотя такой способ представления не является единственным.

1. Если error есть Negative и change-in-error есть Negative, то output есть Zero
2. Если error есть Negative и change-in-error есть Zero, то output есть NM

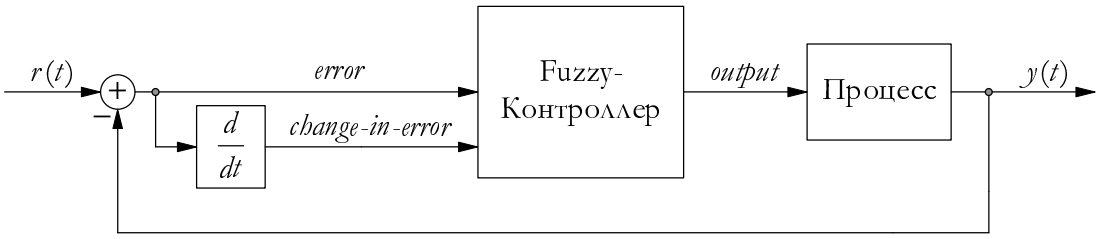


Рис. 3

3. Если error есть Negative и change-in-error есть Positive, то output есть NH
4. Если error есть Zero и change-in-error есть Negative, то output есть NM
5. Если error есть Zero и change-in-error есть Zero, то output есть Zero
6. Если error есть Zero и change-in-error есть Positive, то output есть PM
7. Если error есть Positive и change-in-error есть Negative, то output есть Zero
8. Если error есть Positive и change-in-error есть Zero, то output есть PM
9. Если error есть Positive и change-in-error есть Positive, то output есть PH

Здесь названия Positive, Negative и Zero а также NH, NM, PH, PM — имена нечётких множеств. Этот же набор правил может быть представлен компактно в реляционной форме.

| error | change-in-error | output |
|---|---|---|
| Negative | Negative | Zero |
| Negative | Zero | NM |
| Negative | Positive | NH |
| Zero | Negative | NM |
| Zero | Zero | Zero |
| Zero | Positive | PM |
| Positive | Negative | Zero |
| Positive | Zero | PM |
| Positive | Positive | PH |

Табл. 1

В Табл. 1 предполагается, что первые две колонки — входы, а самая правая колонка — выход. Тогда каждая строка будет определять правило. Такой формат представления базы правил неявно предполагает, что входы связаны логическим И, или же логическим ИЛИ, если правила относятся к одному и тому же выходу.

Существует ещё один формат в виде лингвистической таблицы.

| | | change-in-error | | |
|---|---|---|---|---|
| | | Negative | Zero | Positive |
| error | Negative | Zero | NM | NH |
| | Zero | NM | Zero | PM |
| | Positive | Zero | PM | PH |

Табл. 2

В Табл. 2 пустующая ячейка вывода означала бы неполное правило. Не смотря на компактность и удобство, представление в таком виде оказывается затруднительным, в случае если входов > 2.

### 4.4. Связующие

Предложения в базе правил составляют с помощью связующих союзов «и», «или» и частицы «не». В нечёткой логике они могут быть реализованы вычислительно по-разному.

Предназначение связующих — вычисление комбинаций из функций принадлежности в правилах.

Хотя в нашем примере встречается только связующее «и», правило могло бы выглядеть и как «Если error есть очень Negative и не есть Zero или change-in-error есть Zero, то <...>».

$$\mu(x) \text{ И } \mu(y) = \min(\mu(x), \mu(y)) \tag{1}$$

$$\mu(x) \text{ ИЛИ } \mu(y) = \max(\mu(x), \mu(y)) \tag{2}$$

$$\text{НЕ } \mu(x) = 1 - \mu(x) \tag{3}$$

Существует другой вариант для операций И и ИЛИ. На практике, с их использованием процесс регулировки будет «мягче».

$$\mu(x) \text{ И } \mu(y) = \mu(x) \cdot \mu(y) \tag{4}$$

$$\mu(x) \text{ ИЛИ } \mu(y) = \mu(x) + \mu(y) - \mu(x) \cdot \mu(y) \tag{5}$$

### 4.5. Модификаторы

Модификаторы по сути есть операторы, изменяющие функции принадлежности некоторым образом. Например, модификатор «очень» в предложении «очень Negative» изменяет нечёткое множество Negative. Модификатор «очень» может быть задан как возведение функции принадлежности в квадрат:

$$\text{очень } a = a^2.$$

При возведении в степень > 1 функция сжимается, а при возведении в степень < 1 растягивается. Модификаторы могут быть разной степени силы, «около», «приблизительно», «весьма»:

$$\text{чрезвычайно } a = a^3,$$

$$\text{слегка } a = a^{1/3}.$$

Целое семейство модификаторов генерируется как $a^p$, где $p$ любая степень, от 0 до ∞. Если $p = \infty$, то модификатор имеет смысл «в точности», так как он подавит все значения функции принадлежности, кроме точки, где она принимает значение 1. Результаты модификаторов, где $p = 1/2$ и $p = 3$, применённых к функции $e^{-x^2/2}$, показаны графически на Рис. 7. Модификатору при $p = 3$ резонно дать название «весьма».

### 4.6. Области исследования

Элементы нечётких множеств берутся из *областей исследования* (или областей рассуждения), мы далее будем называть их просто *областями*.

Перед тем, как конструируются функции принадлежности, принимают во внимание области для входных и выходных значений.

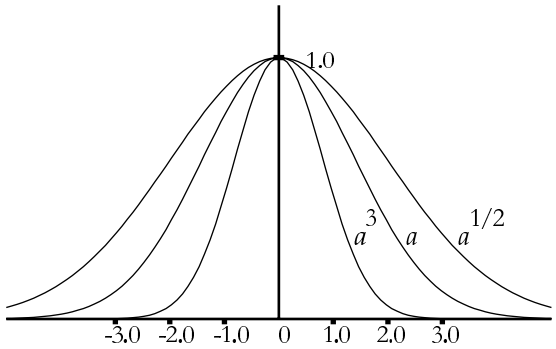


Рис. 7

Например, для правила

Если error есть Negative и change-in-error есть Positive, то output есть NH

функции принадлежности лингвистических терминов Negative и Positive должны быть определены для всей области входных сигналов error и change-in-error.

Разработчик контроллера должен определить, будет ли функция принадлежности дискретной или же непрерывной. В случае, если выбирается дискретная функция принадлежности, она реализуется в виде массива значений. Этот массив индексируется значением дискрета входного сигнала.

## 5. Функции принадлежности

Каждый элемент области входит в какое-либо нечёткое множество в некоторой степени, возможно даже и нулевой. Степень принадлежности для каждого элемента области и есть описание нечёткого множества. В нечётких множествах степени принадлежности назначаются таким образом, что переходы из одной степени принадлежности в другую для элементов области выполнялись плавно. Функция, которая задаёт привязку каждому элементу множества из области, называется функцией принадлежности, и обозначается как $\mu(x)$.

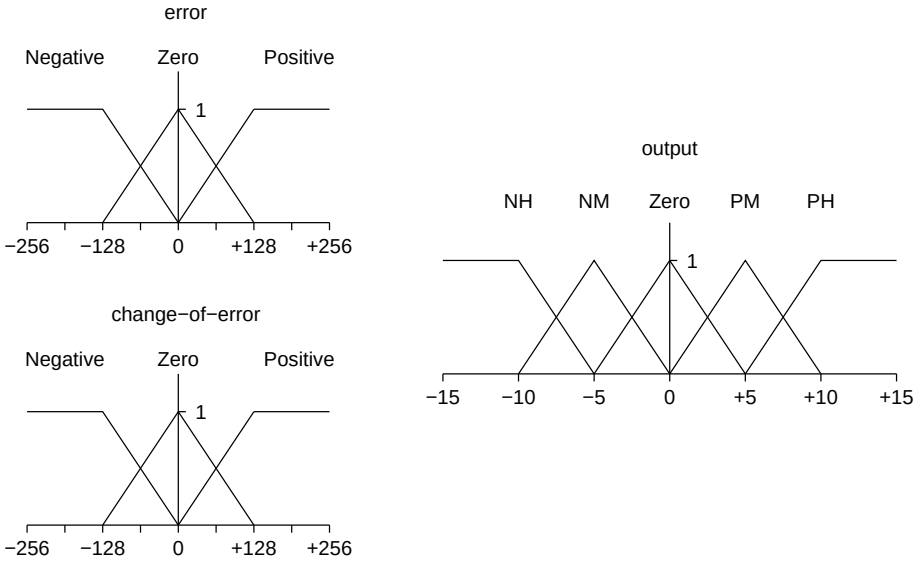


Рис. 4

Здесь резонно возникают два практических вопроса.

1. Каким образом следует определять форму множеств?
2. Сколько множеств достаточно определить для области?

С точки зрения теории нечётких множеств, множества могут принимать любую форму, и самой теории нет дела до их целевого назначения. Исходя же из практических соображений, существует несколько эмпирических правил:

- множество лингвистического термина должно быть достаточно широким для того, чтобы нивелировать шумы измерений;
- некоторая часть каждого множества должна пересекаться с соседними множествами, иначе возможны неопределённые состояния.

Количество множеств, необходимых для покрытия области, в общем определяется самой областью, однако, в практическом смысле, здесь важен опыт, полученный экспериментально или известный операторам. Практики рекомендуют следующий подход (см. Рис. 4):

- Начинать следует с треугольной формы множеств. Все функции принадлежности для конкретного входа или выхода должны быть симметричными треугольниками одинаковой ширины. Самое левое и правое множество должны быть наклонными с горизонтальным «плечом».
- Пересечение должно составлять, как минимум, 50 процентов. Ширина должна выбираться таким образом, чтобы каждый элемент области принадлежал, как минимум, двум множествам, за исключением, может быть, крайнего левого и правого множеств.

В строгом математическом смысле, нечёткое множество $A$ есть набор упорядоченных пар

$$A = \{(x, \mu(x))\}, \tag{6}$$

где $x$ — элемент области, а $\mu(x)$ — его степень принадлежности области $A$. Пара из $(x, \mu(x))$ называется *нечётким одноточечным множеством*, или *синглтоном*. Одноточечное нечёткое множество состоит из одного значения, и заменяет множество одним числом. Например

1 Если error есть Positive, то output есть 10 Вольт

2 Если error есть Zero, то output есть 0 Вольт

3 Если error есть Negative, то output есть -10 Вольт

Есть несколько преимуществ в использовании синглтонов в выводе:

- вычисления становятся проще;
- появляется возможность в точности выдать максимальное значение выходного сигнала;
- оператору составление правил и их чтение, возможно, покажется интуитивнее.

Если положить, что нечёткие множества для output в примере на Рис. 4 были заменены на синглтоны, а максимальные значения выходного сигнала составляют $-12.5$ и $+12.5$, то мы бы получили форму множеств такую, как изображено на Рис. 5.

## 6. Принятие решений

В примере базы правил, девять правил определяют, что выходной сигнал есть некая комбинация сигналов error и change-in-error. В каждой строке правил, для каждого условия, машина логического вывода находит значения функций принадлежности.

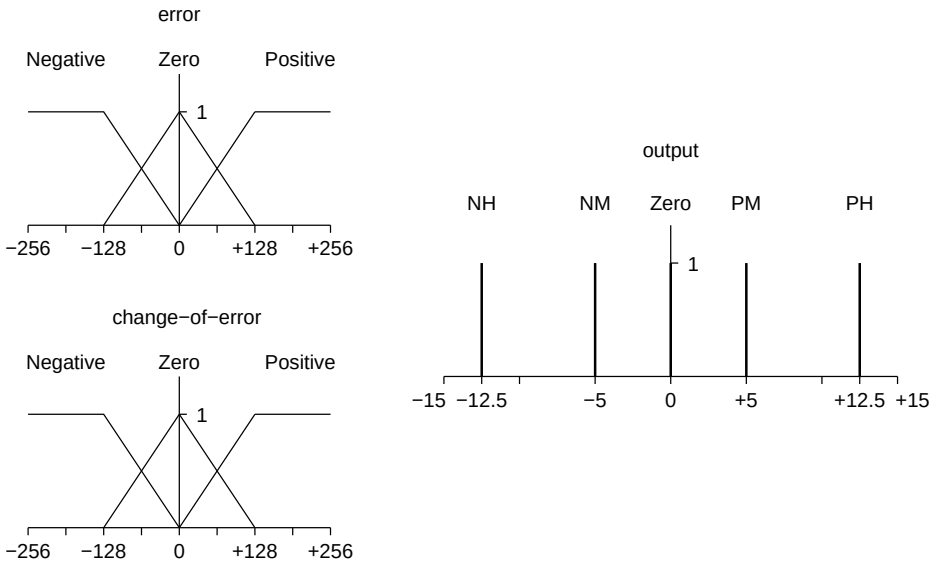


Рис. 5

### 6.1. Объединение

Операция объединения применяется тогда, когда вычисляется *степень соответствия* $\alpha_k$ для правила $k$. Пусть, например, для правила 1 в его условии мы получили значения функций принадлежности для error и change-of-error соответственно $\mu_{e1}$ и $\mu_{\Delta e1}$. Тогда их объединение, как следует из (1), это

$$\alpha_1 = \mu_{e1} \quad \text{И} \quad \mu_{\Delta e1} = \min(\mu_{e1}, \mu_{\Delta e1}). \tag{7}$$

Для всех прочих правил, операции объединения выполняются точно таким же образом. Объединение по сути формирует из нескольких входов один выход. Значение операции объединения называют степенью соответствия правила. Это значение масштабирует нечёткое множество, или синглтон, если это множество состояло из одного скаляра, заданные в выводе правила. Пример объединения для правила 2 показан на Рис. 5.

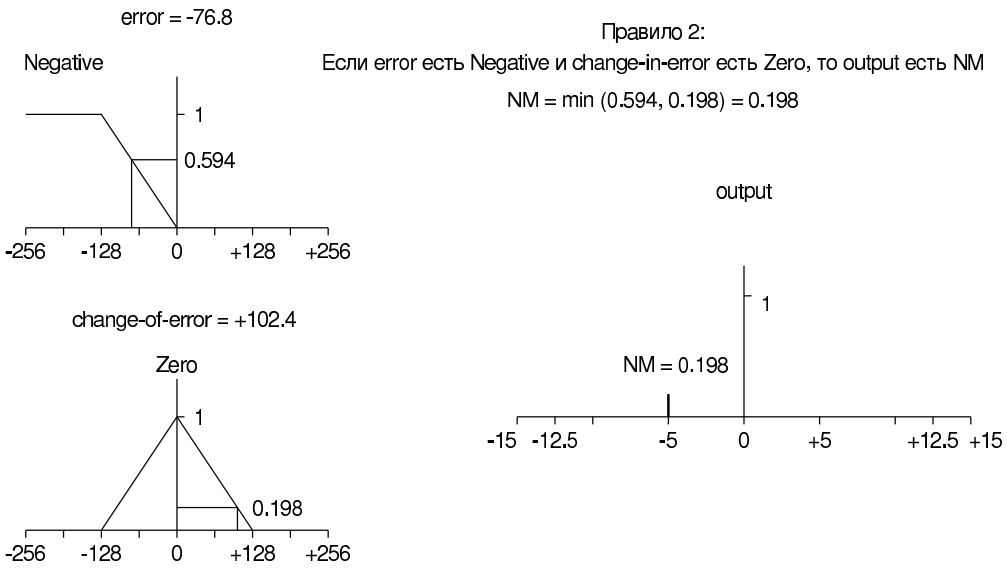


Рис. 6

### 6.2. Активация

Активация правила это получение вывода, возможно, уменьшенного на степень соответствия. В дополнение к этому, правило $k$ может априори иметь весовой коэффициент $\omega_k \in [0, 1]$, называемый *степенью доверия*.

$$a_k^* = \omega_k \cdot a_k \tag{8}$$

Степень доверия задаётся или оператором при конфигурировании контроллера, или же некоторой обучающейся программой, которая пытается адаптироваться к некоторым отношениям сигналов входа и выхода.

### 6.3. Суммирование

Все нечёткие множества, полученные в выводе из правил, аккумулируются с помощью оператора max. Если функции принадлежности в выводе правил являются синглтонами, то в результате получим сумму множеств:

$$\alpha_1 \cdot s_1 + \alpha_2 \cdot s_2 + \ldots + \alpha_n \cdot s_n,$$

где $\alpha_k$ — степень соответствия для правила $k$, а $s$ — значение синглтона, как функции принадлежности, в выводе правила $k$.

### 6.4. Получение решения

Нечёткое множество, полученное после суммирования, должно быть преобразовано каким-то образом в число, для того, чтобы его можно было послать процессу. Существует несколько методов.

#### Центр масс

Выходное значение (скаляр) $u$ есть центр масс для нечёткого множества:

$$u = \frac{\sum_i \mu(x_i)\, x_i}{\sum_i \mu(x_i)}. \tag{9}$$

Здесь $x_i$ — пробегает все значения дискретной области, а $\mu(x_i)$ — значение функции принадлежности. Эта формула может быть интерпретирована как вычисление средневзвешенного значения элементов множества. В случае непрерывных функций, знаки сумм заменяются на интегралы.

#### Центр масс для синглтонов

Если функции принадлежности в выводах правил являются синглтонами, то центр масс $u$ для них рассчитывается по формуле

$$u = \frac{\sum_i \alpha_i\, s_i}{\sum_i \alpha_i}. \tag{10}$$

Здесь $\alpha_i$ — степень соответствия для $i$-го правила, а $s_i$ значение скаляра синглтона в выводе $i$-го правила.