Применение нечеткой логики в системах автоматического управления

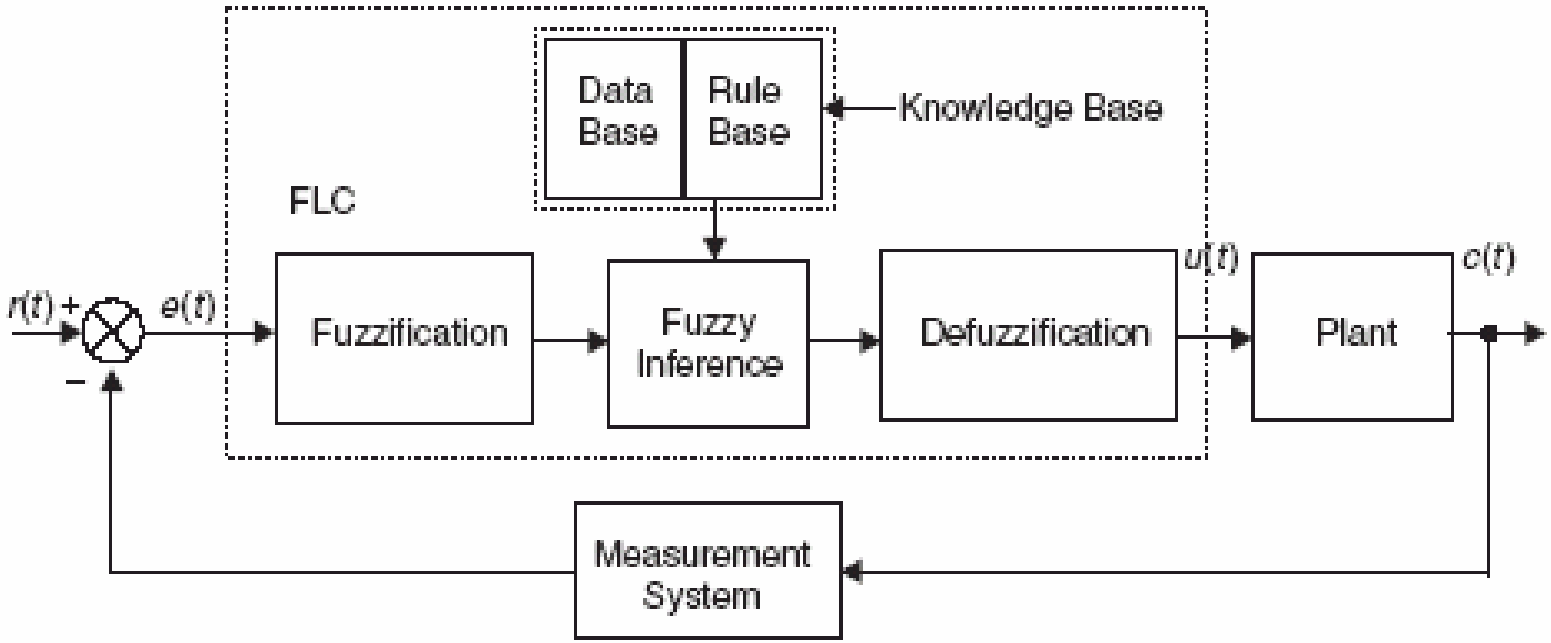
Лекция 3-2

Представление знаний в информационных системах

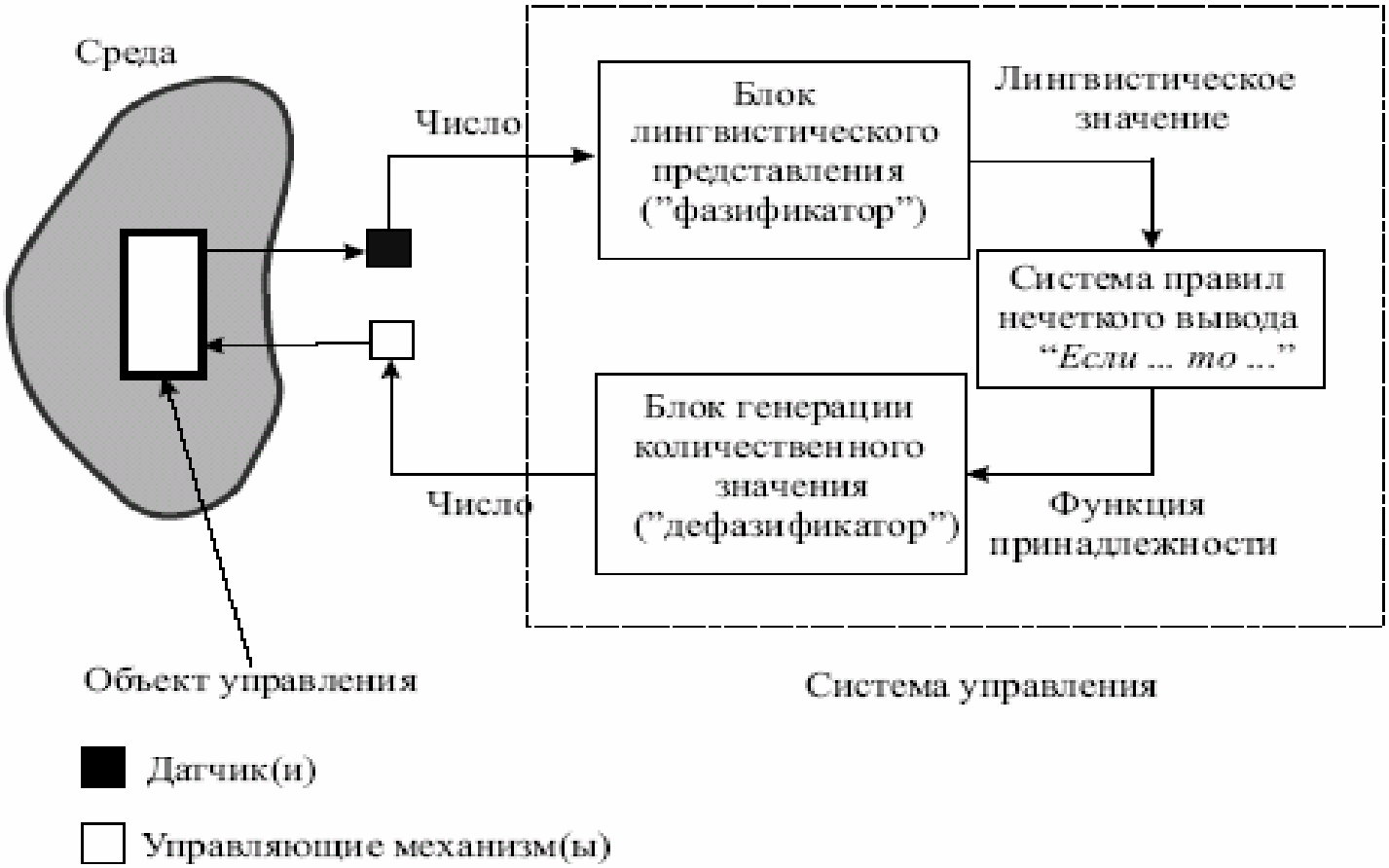
Этапы обработки информации при использовании нечеткой логики

- Нечеткие рассуждения
  - На основе фаззификации – преобразование численного значения в символьное нечеткое значение
- Четкое принятие решения
  - дефаззификация – преобразование нечеткого символьного значения в число

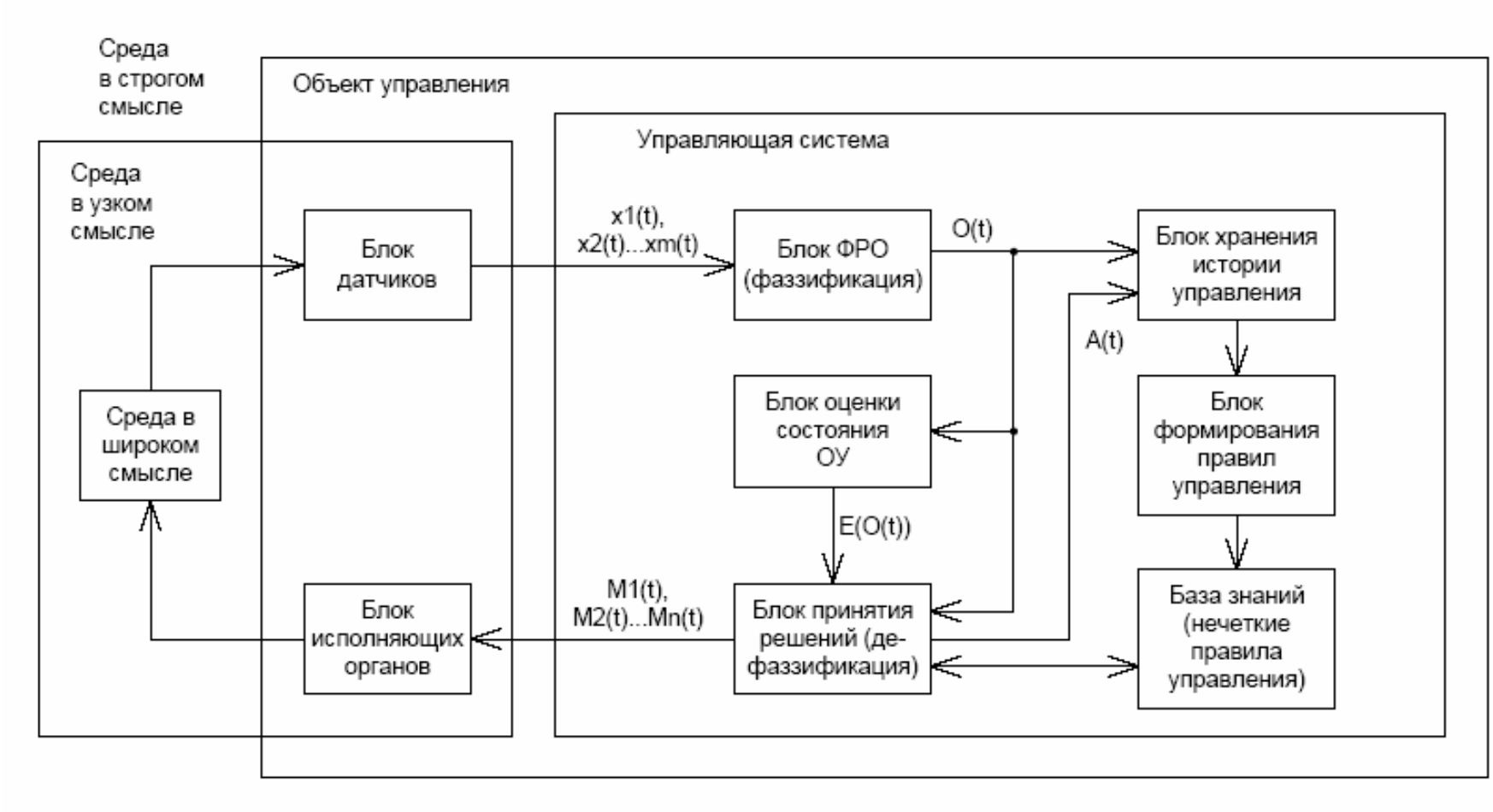
Структура системы управления на основе нечеткой логики



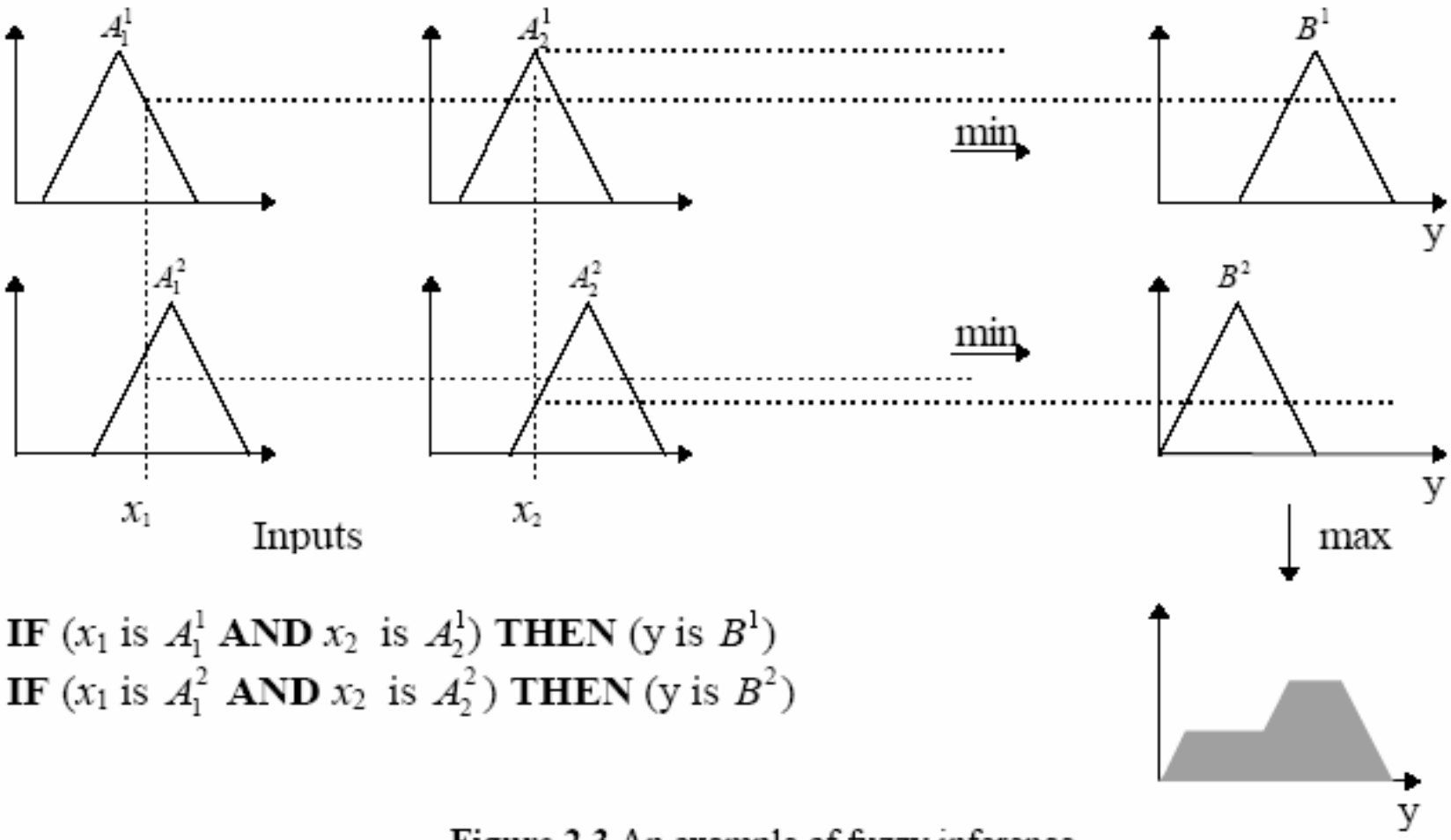
Структура системы управления на основе нечеткой логики (2)



Общая структура адаптивной системы управления с использованием нечеткой логики



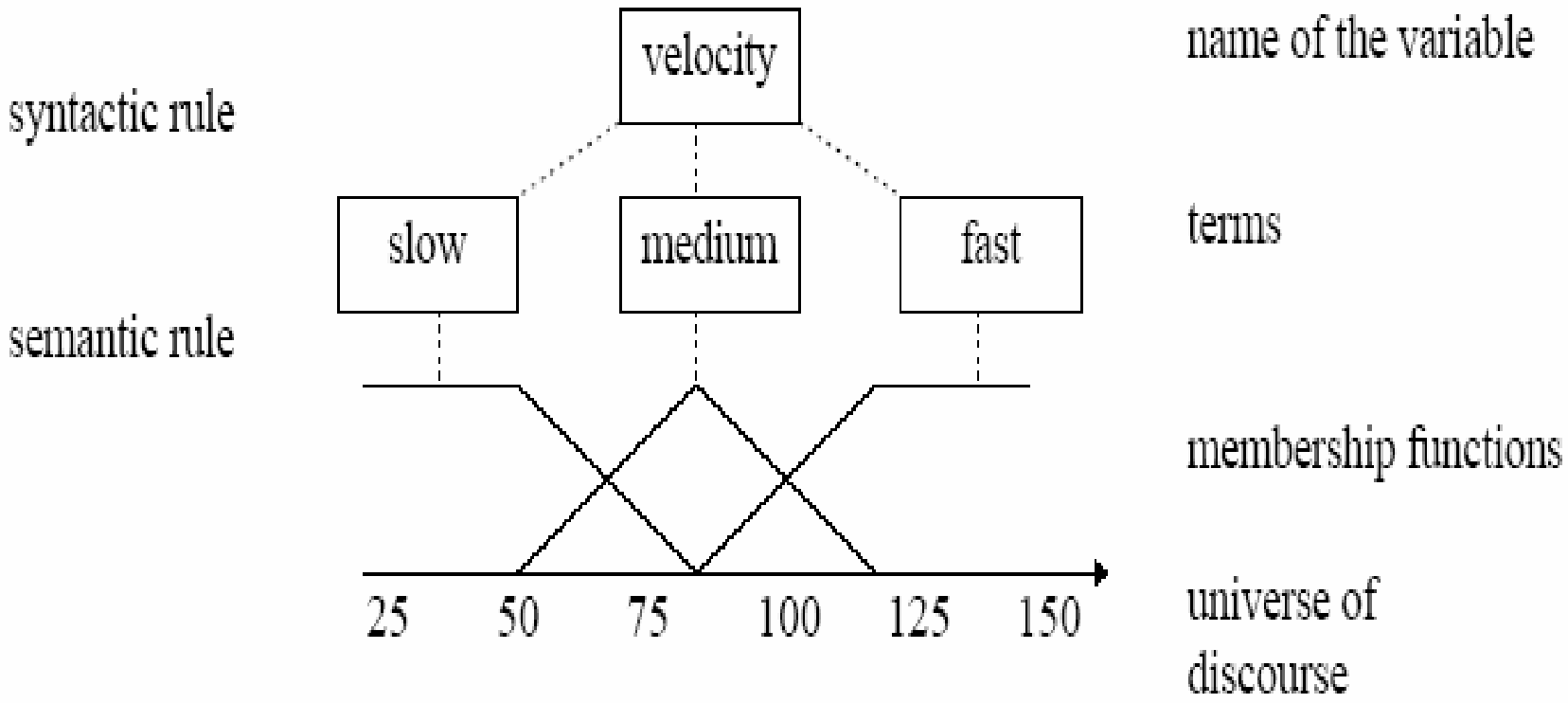
Пример нечеткого логического вывода



IF (x_1 is A_1^1 **AND** x_2 is A_2^1) **THEN** (y is B^1)
IF (x_1 is A_1^2 **AND** x_2 is A_2^2) **THEN** (y is B^2)

Figure 2.3 An example of fuzzy inference.

Пример лингвистической переменной



Методы дефаззификации

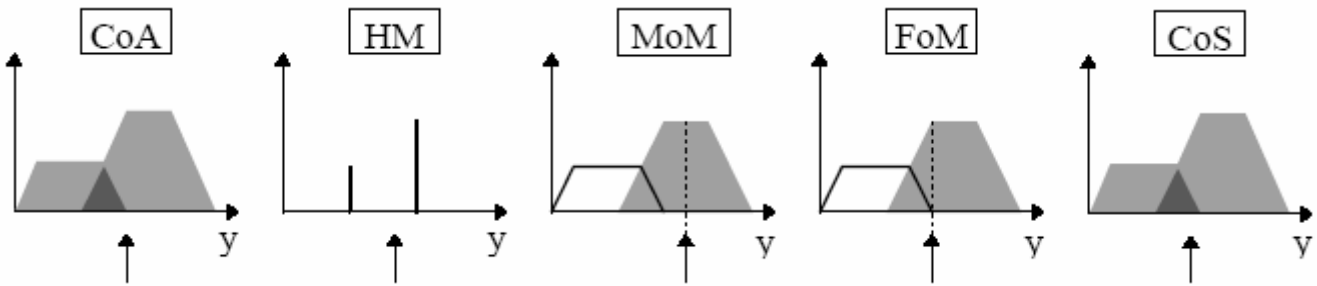
Центр гравитации

центр области

центр суммы

Средний максимум

Первый максимум



Особенности нечеткой логики (fuzzy logic)

- В нечеткой логике точные рассуждения рассматриваются как частный случай нечетких рассуждений
- В нечеткой логике нечто является чем-то определенным только в какой-то степени
- В нечеткой логике знание интерпретируется как набор гибких или нечетких ограничений на наборе нечетких переменных
- Вывод рассматривается как процесс распространения нечетких ограничений
- Любая логика может быть фаззифицирована

Какая разница между нечеткой логикой и обычными методами управления?

Нечеткая логика вводит простой, основанный на правилах вида IF X AND Y THEN Z подход к решению проблемы управления вместо попыток смоделировать систему математически.

Нечеткая логика основана на эмпирике (опыте) оператора, а не на понимании внутренностей системы.

Например, вместо того, чтобы оперировать такими высказываниями с температурой как "SP =500F", "T <1000F", или "210C <TEMP <220C", мы имеем дело с правилами типа "IF (process is too cool) AND (process is getting colder) THEN (add heat to the process)" или "IF (process is too hot) AND (process is heating rapidly) THEN (cool the process quickly)".

Эти высказывания неточны и в то же время описывают то, что действительно происходит.

Нечеткая логика описывает поведение оператора при управлении.

Как нечеткая логика работает?

Нечеткая логика использует некоторые численные параметры для того, чтобы оценивать ошибку и скорость изменения ошибки,
Но точные значения этих величин обычно не требуется.
Например, простая система управления температурой может использовать один датчик, показания которого вычитаются из управляющего сигнала для вычисления ошибки и затем происходит дифференциация по времени для вычисления скорости изменения ошибки.
Ошибка может вычисляться в градусах F и маленькая ошибка может ассоциироваться с 2F, а большая с 5F.
Скорость изменения ошибки может измеряться в град/мин с маленькой скоростью в 5F/min и большой в 15F/min.
Эти величины не должны быть симметричными и могут быть настроены для лучшего представления поведения при управлении.
Обычно нечеткая логика позволяет системе работать с первого раза без всякой настройки.

Первый в мире Fuzzy Logic Controller

В Англии в 1973 в University of London, профессор Mamdani и студент S. Assilian пытались стабилизировать скорость маленького парового двигателя, построенного студентом.

Они имели оборудование для этого – мини-ЭВМ PDP-8 и обычные средства для цифрового управления.

Но они не могли управлять двигателем как им хотелось.

Наблюдалось перерегулирование и требуемая скорость достигалась только после серии колебаний, т.е. управление скоростью было очень медленным

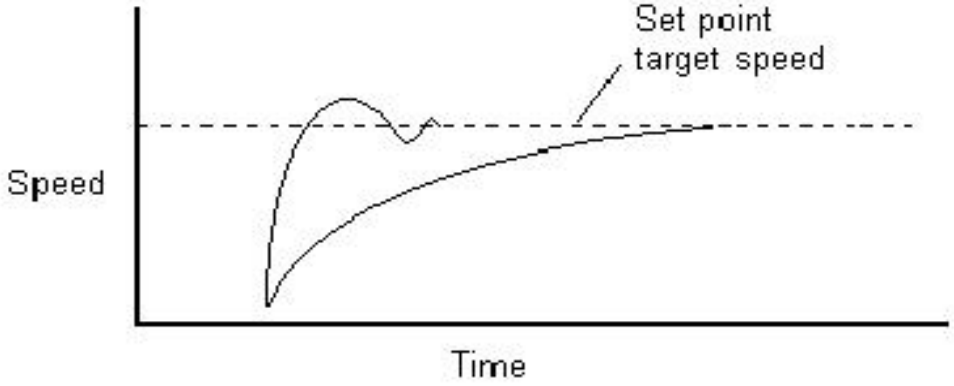


Figure 1. System Response Without Fuzzy Logic Controller

Применив нечеткую логику они получили лучшие результаты

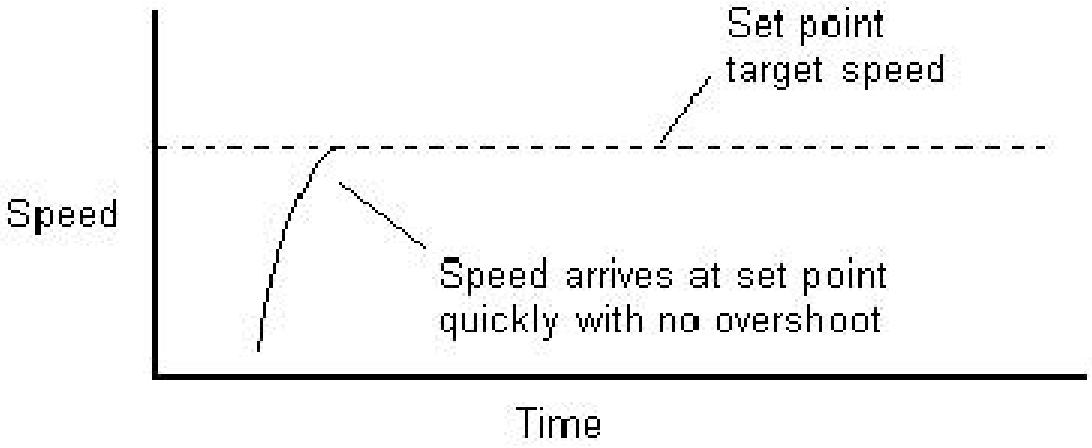


Figure 2. System Response Using Fuzzy Logic Controller

Для того, чтобы получить fuzzy logic control system на персональном компьютере, необходимо:

- Определить входы.
- Описать состояния и действия системы с помощью нечетких правил на естественном языке.
- Написать программу для выполнения этих правил над всеми входами и для определения выходов. Правила становятся операторами "If-Then" в программе. (Как показано дальше, при включении в управление обратной связи, использование треугольников для представления функций принадлежности может помочь визуализировать и вычислять результаты выполнения правил)
- В программе используется взвешенное среднее для объединения разных действий, вызванных разными правилами, в одно действие управляемой системы. (В случае, если имеется только один выход, объединения не требуется)

Функциональная схема системы управления скоростью электродвигателя.
Двигатель управляется контроллером, управляемым программой, реализованной на BASIC в ПК

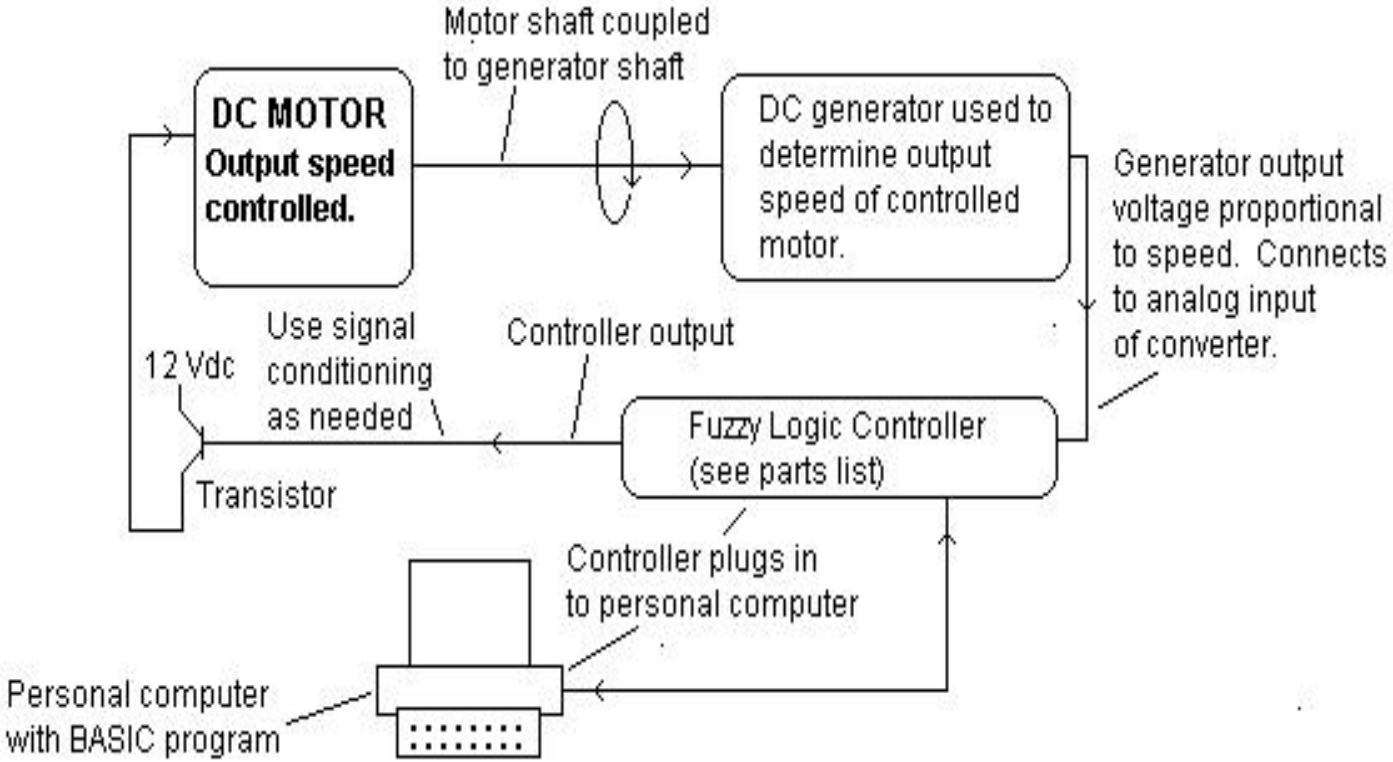


Figure 3. Motor Speed Control System

Компоненты

- IBM or compatible personal computer equipped to run Microsoft Quick BASIC. IBM is a registered trademark of IBM Corporation. Microsoft and Quick BASIC are registered trademarks of Microsoft, Inc.
- 8 channel input; 8 bit, analog/digital converter with 8 on-off, digital output channels and one 8 bit digital/analog output channel.
- Signal conditioner (transistor amplifier to adjust levels as needed)
- Transistor - 2N3053.
- DC motor, 1.5 V to 3.0 V, 100 ma., 1100 Rpm to 3300 Rpm.

Этапы разработки этой системы:

1. Определение входов. Примеры: Температура для управления системой кондиционирования в доме. Скорость автомобиля для системы управления движением. В нашем случае, вход – скорость в оборотах в мин. Ошибка между требуемой скоростью 2,420 об/мин определяется в программе. Ошибка может быть положительной и отрицательной. Напряжение генератора, пропорциональное скорости, преобразуется в контроллере в цифровое значение для компьютера.
2. Определение выхода. Для кондиционера это включение и выключение вентилятора. Для управление движением автомобиля это регулирование дросселя. В нашем случае только один выход – напряжение, подаваемое на транзистор, управляющий электродвигателем.
3. Определение требуемой величины. Например, 70 градусов F для температуры в доме, или 60 Miles per hour для автомобиля. Для нашего случая цель - 2,420 оборотов в мин.
4. Выбор словесного описания для правил.

Для управления паровым двигателем профессор Mamdani использовал следующие слова:

•Positive Big
•Positive Medium
•Positive Small
•Almost No Error
•Negative Small
•Negative Medium
•Negative Big

Наша система проще и можно использовать только три описания состояния входа:

Input Status Word Descriptions

•Too slow
•About right
•Too fast

Для выходов:

Output Action Word Descriptions

•Speed up
•Not much change needed
•Slow down

RULES:

Rule 1: If the motor is running too slow, then speed it up.

Rule 2: If motor speed is about right, then not much change is needed.

Rule 3: If motor speed is to fast, then slow it down.

5. Описание связей между входами и выходами в виде диаграммы (Figure 4)

Диаграмма строится с использованием треугольников для описания функций принадлежности используемых значений лингвистических переменных. Могут быть и другие фигуры, но треугольники удобнее.

Ширина треугольников может быть разной.

Узкие треугольники обеспечивают более строгое управление. Узкие треугольники обычно используются в центре.

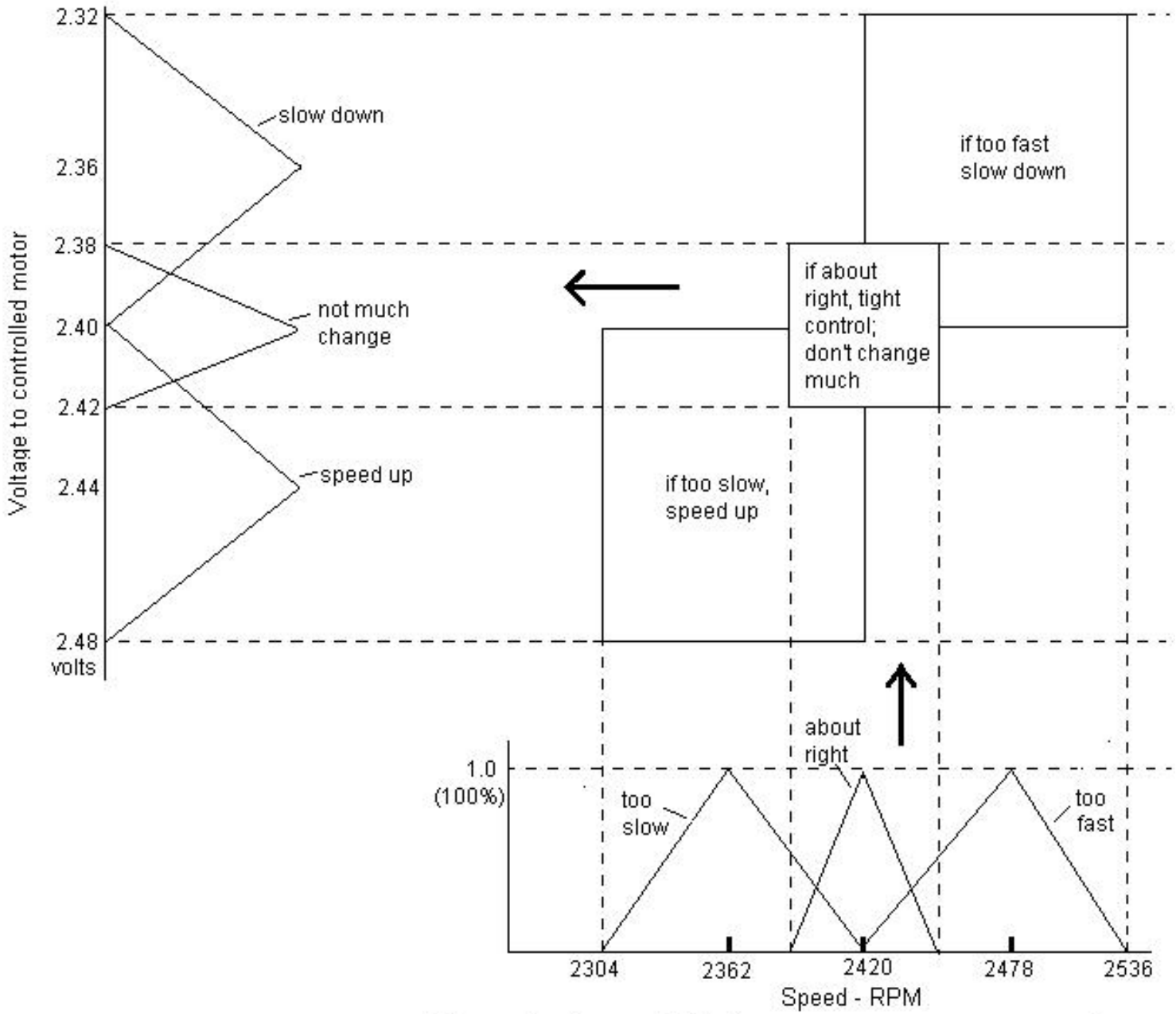


Figure 4 Cause-Effect

6. Figure 4 получается из ранее приведенных правил (Rules) и результатом является напряжение, подаваемое на контроллер скорости:
   a. Если скорость About right то Not much change необходимо для контроллера.
   b. Если скорость Too slow то для того, чтобы увеличить ее, надо подать Speed up.
   c. Если скорость Too fast то необходимо Slow down.
7. Определить выход – напряжение для подачи на транзистор

Предположим, что скорость увеличилась с нужной 2,420 Rpm до 2,437.4 Rpm, 17.4 Rpm сверх требуемой. Необходимо уменьшить скорость до 2,420 Rpm. Интуитивно ясно, что надо немного уменьшить напряжение, подаваемое на транзистор. На диаграмме внизу вертикальная линия – фактическая скорость.

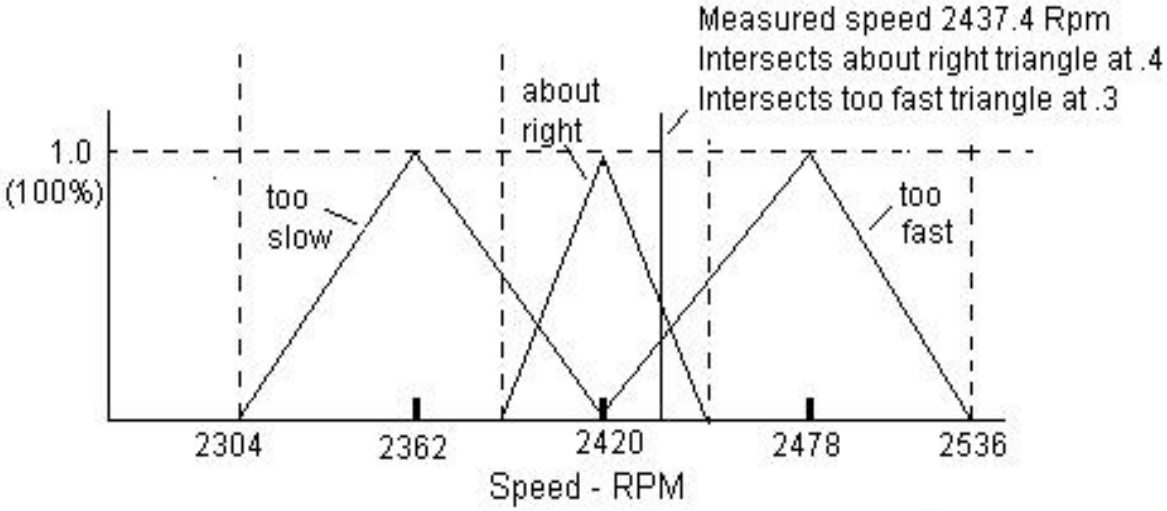


Figure 5. Speed above target value

Эта линия пересекает треугольник About right при значении функции принадлежности .4 и треугольник Too fast - при .3. :

Intersect point / 1 = 11.6/29 = .4

Intersect point / 2 = 17.4/58 = .3

Следующий этап – нарисовать выходные треугольники. Их высота определяется полученными .3 и .4 значениями.

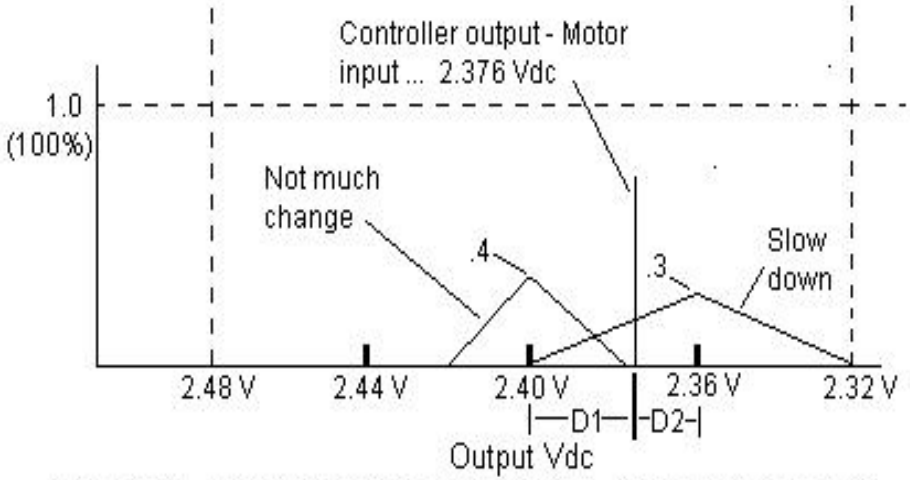


Figure 6. Determination of Control Voltage to Motor

The portion of the program for the above system which examines the input and performs the "triangle" calculations to arrive at a crisp output follows:

```
910 IF MS = 2420 THEN MIV = 2.4 : GOTO 5000 'MS-MEASURED SPEED, MIV-MOTOR INPUT VOLTAGE
920 IF MS < 2420 THEN 2000 ELSE 1000
1000 ' LINES 1010-1110; GREATER THAN 2420 RPM, SLOW DOWN
1010 IF MS > 2449 THEN MIV = 2.36 : GOTO 5000
1020 ' COMPUTE INTERSECT POINT, IPA, FOR 'ABOUT RIGHT' TRIANGLE
1030 IPA = (2449-MS) / 29
1040 IF IPA =< 0 THEN IPA = .0001
1050 ' COMPUTE INTERSECT POINT, IPS, FOR 'SLOW DOWN' TRIANGLE
1060 IPS = (MS-2420) / 58
1070 ' COMPUTE MOTOR (TRANSISTOR) INPUT VOLTAGE (MIV)
1080 AAR = .5 * .04 * IPA 'AAR - AREA OF 'ABOUT RIGHT' TRIANGLE
1090 ASD = .5 * .08 * IPS 'ASD - AREA OF 'SLOW DOWN' TRIANGLE
1100 D1 = .04 * (ASD / (ASD+AAR))
1110 MIV = 2.4 - D1 : GOTO 5000
2000 ' LINES 2010-2110; LESS THAN 2420 RPM, SPEED UP
2010 IF MS < 2362 THEN MIV = 2.44 : GOTO 5000
2020 ' COMPUTE INTERSECT POINT, IPA, FOR 'ABOUT RIGHT 'TRIANGLE
2030 IPA = (MS-2391) / 29
2040 IF IPA =< 0 THEN IPA = .0001
2050 ' COMPUTE INTERSECT POINT, IPF, FOR 'SPEED UP' TRIANGLE
2060 IPF = (2420-MS) / 58
2070 ' COMPUTE MOTOR INPUT VOLTAGE (MIV)
2080 AAR = .5 * .04 * IPA 'AAR - AREA OF 'ABOUT RIGHT 'TRIANGLE
2090 ASU = .5 * .08 * IPF 'ASU - AREA OF 'SPEED UP' TRIANGLE
2100 D1 = .04 * (ASU / (ASU+AAR))
2110 MIV = 2.4 + D1
5000 '
```