Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Кузбасский государственный технический университет
имени Т. Ф. Горбачева»

Кафедра информационных и автоматизированных
производственных систем

Составители
В. Ю. Садовец, С. А. Кизилов

МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В РОБОТОТЕХНИКЕ

Методические указания к лабораторным работам для студентов очной формы обучения

Рекомендованы учебно-методической комиссией направления
подготовки 15.04.04 «Автоматизация технологических процессов
и производств» в качестве электронного издания
для использования в учебном процессе

Кемерово 2016

Рецензенты:

Чичерин И. В., кандидат технических наук, доцент, заведующий кафедрой информационных и автоматизированных производственных систем, председатель учебно-методической комиссии направления подготовки 15.04.04 «Автоматизация технологических процессов и производств»

Любимов О. В., кандидат технических наук, доцент кафедры информационных и автоматизированных производственных систем

Садовец Владимир Юрьевич
Кизилов Сергей Александрович

Методы искусственного интеллекта в робототехнике [Электронный ресурс]: методические указания к лабораторным работам для студентов направлений подготовки: 15.04.04 «Автоматизация технологических процессов и производств» (профиль «Роботы и робототехнические системы») очной формы обучения / сост. В. Ю. Садовец, С. А. Кизилов; КузГТУ. – Электрон. дан. – Кемерово, 2016. – Систем. требования : Pentium IV ; ОЗУ 256 Мб ; Windows XP, мышь. – Загл. с экрана.

Лабораторные работы выполняются на персональном компьютере с использованием программных комплексов MATLAB, предназначенных для научных и инженерных расчетов.

Каждая лабораторная работа включает все требуемые коды на языке MATLAB. В ходе выполнения лабораторных работ магистранты исследуют основные свойства и функциональные возможности базовых интеллектуальных технологий, которые в настоящее время наиболее активно используются при создании интеллектуальных систем управления сложными динамическими объектами.

Лабораторный практикум может быть изменен по согласованию с научным руководителем и заведующим кафедрой выполнением работ в ходе научно-исследовательской практики.

ЛАБОРАТОРНАЯ РАБОТА № 1

ИССЛЕДОВАНИЕ И СИНТЕЗ ФИЛЬТРА КАЛМАНА

Цель работы: изучение математических основ фильтра Калмана. Моделирование и исследования основных параметров, влияющих на качество фильтрации.

Работа выполняется на программе MATLAB.

Время выполнения работы – 4 учебных часа.

Теоретические основы фильтра

Пусть имеется линейная система, которая может быть описана следующими уравнениями

$$\begin{cases} X_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k + z_k \end{cases}$$

где A, B и C – матрицы коэффициентов; k – индекс такта времени; x отражает состоянием системы; u – известный вход системы; y – измеренный выход системы; w – шум процесса; z – шум измерений. Каждая из этих величин (в общем случае) является вектором.

С помощью алгоритма фильтра Калмана требуется в реальном времени построить оптимальную оценку состояния системы, основываясь на измерениях y, содержащих собственные погрешности. При этом y рассматривается в качестве измеренного многомерного выходного сигнала системы, имеющего шум, а вектор состояния – неизвестный многомерный сигнал, который требуется найти. Условием оптимальности построенной оценки состояния является минимум ее средней квадратичной ошибки.

Для корректного построения данного фильтра требуется, чтобы матожидания z и w были равны нулю. Также необходимо чтобы сигналы w и z были независимыми случайными величинами и не имели между собой корреляция.

Тогда матрицы ковариации S_w и S_z описываются следующим образом:

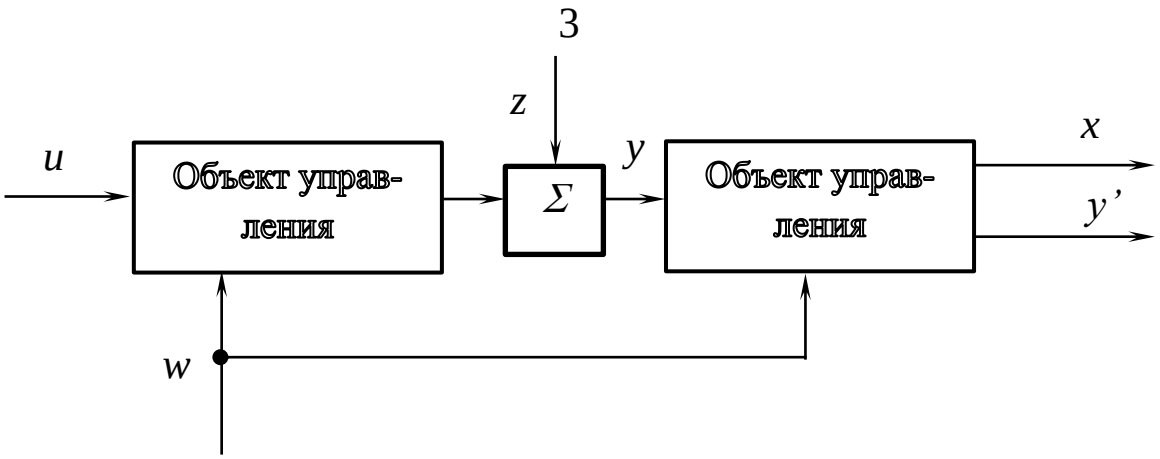

Рис. 1.1. Система фильтрации сигнала

- ковариация процесса: $S_w = E(w_k w_k^T)$,
- ковариация процесса: $S_z = E(z_k z_k^T)$,

где w^T и z^T транспонированные *w* и *z* значения векторов случайного шума, а оператор *E( )* означает ожидаемое значение.

Для описания математического аппарата фильтра Калмана существует несколько вариантов ниже представлен один из возможных

$$K_k = AP_k C^T (CP_k C^T + S_z)^{-1}$$
$$X_{k+1} = (Ax_k + Bu_k) + K_k(y_{k+1} - Cx_k)$$
$$P_{k+1} = AP_k A^T + S_w - AP_k C^T S_z^{-1} CP_k A^T$$

Анализ выражения для *K* показывает, что величина шума S_z, обратно пропорциональна величине *K* (матричный коэффициент усиления Калмана), тем самым это определяет степень учета измеренного значения при расчете следующего x.

Второе выражение определяет искомый сигнал.

Выражение для P_{k+1} называют корректирующей частью, она представляет собой коррекцию состояния системы согласно нашим измерениям [8,9].

Задание

1. Используя теоретический материал и листинг программы из примера, реализовать и исследовать фильтр для ОУ, имею-

щего передаточную функцию согласно варианту при изменяющихся на порядок параметр шума.

2. Реализовать модель в Simulink.

Пример

```
%КОД 1 Реализация фильтра Калмана
measnoise = 10; % амплитуда шума
accelnoise = 0.2; % ускорение шума
a = [1 dt; 0 1]; % переходная матрица
b = [dt^2/2; dt]; % входная матрица
c = [1 0]; % матрица измерений
x = [0; 0]; % вектор начального состояния
xhat = x; % начальное значение
Sz = measnoise^2; % значение ошибки ковариации
Sw = accelnoise^2 * [dt^4/4 dt^3/2; dt^3/2 dt^2]; % ошибка измерений
P = Sw; % начальное значение поправочного коэффициента
% инициализация массивов
pos = []; % значение реальной позиции
poshat = []; % предполагаемые значения позиции
posmeas = []; % измеренные значения позиции
vel = []; % массив реальной скорости
velhat = []; % массив предполагаемой скорости
for t = 0 : dt: duration,
% Пусть ускорение будет постоянным и равно 1
u = 1;
% моделирование линейной системы
ProcessNoise = accelnoise * [(dt^2/2)*randn; dt*randn];
x = a * x + b * u + ProcessNoise;
% моделирование шума
MeasNoise = measnoise * randn;
y = c * x + MeasNoise;
% Экстраполяция оценки в реальном времени
xhat = a * xhat + b * u;
% Новая поправка
Inn = y – c * xhat;
```

```
% расчет значения ковариации
s = c * P * c’ + Sz;
% Расчет коэффициента Калмана
K = a * P * c’ * inv(s);
% Корректировка нового значения.
xhat = xhat + K * Inn;
% Расчет корректирующего коэффициента.
P = a * P * a’ – a * P * c’ * inv(s) * c * P * a’ + Sw;
% сохранение значения для вывода на график
pos = [pos; x(1)];
posmeas = [posmeas; y];
poshat = [poshat; xhat(1)];
vel = [vel; x(2)];
velhat = [velhat; xhat(2)];
end
close all;
t = 0 : dt : duration;
figure;
plot(t,pos, t,posmeas, t,poshat);
grid;
xlabel(‘Время (sec)’);
ylabel(‘Значения (feet)’);
title(‘Figure 1 – значения (Реальное, Измеренное, и Предполага-
емое)’)
figure;
plot(t,pos-posmeas, t,pos-poshat);
grid;
xlabel(‘Time (sec)’);
ylabel(‘Значение ошибок (feet)’);
title(‘Figure 2 – Измеренное значение ошибки и предполагаемое
значение ошибки’);
```

Содержание отчета:

Графические зависимости сигнала с фильтрацией и без нее, разность полученной оценки с помощью фильтра и реального сигнала. Структура и значения параметров фильтра.

Контрольные вопросы

1. В чем отличие фильтра Калмана от известных вам?
2. Может ли представленный метод распространяться на нелинейные системы?
3. Как зависит адекватность оценки от количества расчетных операций и вида входных сигналов?
4. Как зависит качество оценки от степени коррелированности сигналов шумов?

ЛАБОРАТОРНАЯ РАБОТА №2

ОПТИМИЗАЦИЯ МНОГОМЕРНЫХ ФУНКЦИЙ СРЕДСТВАМИ ГЕНЕТИЧЕСКОГО АЛГОРИТМА

Цель работы: Изучение особенностей применения генетических алгоритмов для нахождения минимума функций со многими экстремумами.

Работа выполняется на программе MATLAB.

Время выполнения работы – 6 учебных часов.

Теоретическое введение

Под оптимизацией функции следует понимать как нахождение экстремального значения какой-либо зависимости на всем рабочем диапазоне.

В отличие от точных методов математического программирования эволюционные методы позволяют находить решения, близкие к оптимальным, за приемлемое время, а в отличии от известных эвристических методов оптимизации характеризуются существенно меньшей зависимостью от особенностей приложения (т.е. более универсальны) и в большинстве случаев обеспечивают лучшую степень приближения к оптимальному решению.

Важнейшим частным случаем эволюционных методов являются генетические методы, основанные на поиске лучших решений с помощью наследования и усиления полезных свойств множества объектов определенного приложения в процессе имитации их эволюции. Общая задача непрерывной оптимизации может представляться как:

$$f(x) \to \min_{x \epsilon D}, \quad D \epsilon RN \tag{2.1}$$

где $D = \{x = (x_1,\ x_2, :, x_N) | x_i$ на $[a_i, b_i], i = 1,2, \dots, N\}$,

$f(x)$ – минимизируемая (целевая) скалярная многопараметрическая функция, которая может иметь несколько глобальных экстремумов, прямоугольная область D – область поиска, D – подмножество RN.

Предполагается, что о функции $f(x)$ известно лишь то, что она определена в любой точке области D. Никакая дополнительная информация о характере функции, и ее свойствах, (дифференцируемость, липшицируемость, непрерывность и т.д.) не учитывается в процессе поиска.

Под решением задачи (2.1) будем понимать вектор $x = (x_1, x_2, \dots, x_N)$. Оптимальным решением задачи (2.1) будем считать вектор x^*, при котором целевая функция $f(x)$ принимает максимальное значение. Исходя из предположения о возможной многоэкстремальности $f(x)$, оптимальное решение может быть не единственным.

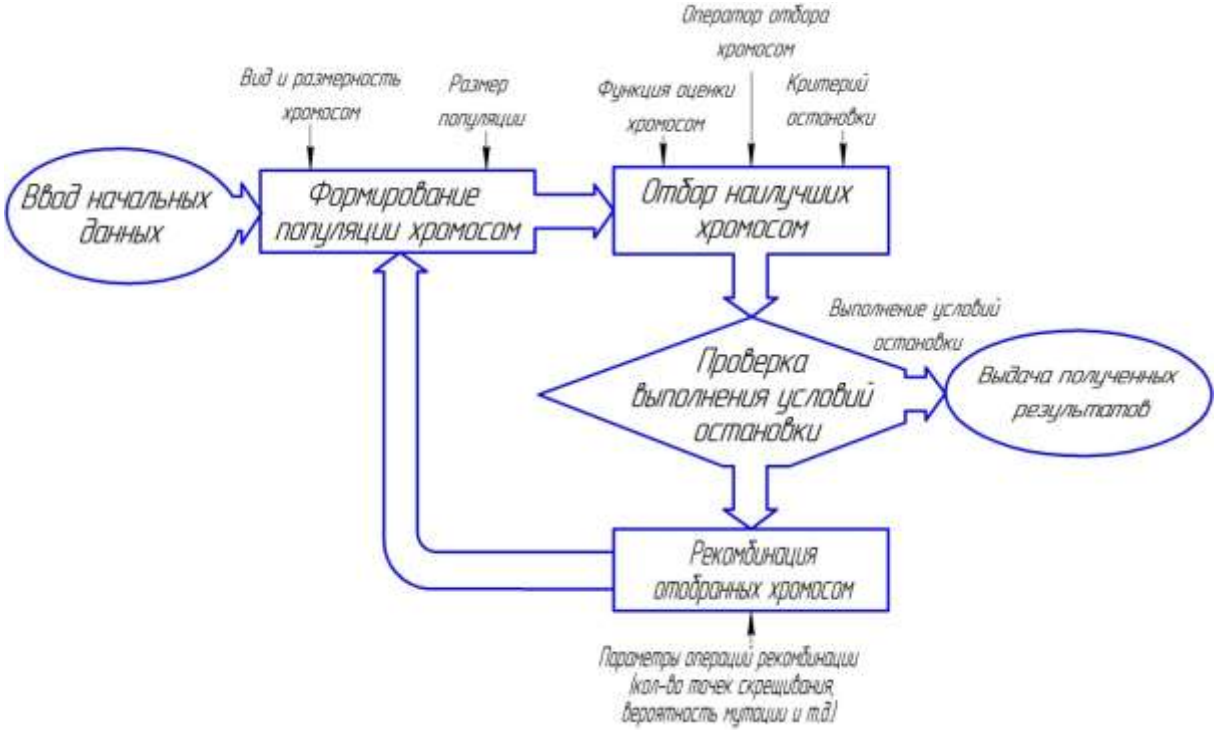

Рис. 2.1. Обобщенная структура стандартного генетического алгоритма

Стандартные генетические алгоритмы, изначально предложенные как эффективное средство для решения различных оптимизационных задач, строятся согласно обобщенной структуре, которая показана на рис. 2.1, и описываются набором следующих основных параметров:

$$GA = (P^T, d, l, s, r, f, t),$$

где $P^T = ({a_1}^T, \dots, {a_d}^T)$ – некоторое подмножество потенциальных решений, называемое популяцией хромосом;

$T = (0, 1, 2 \dots)$ – номер популяции;

d – размер популяции;

${a_1}^T$ – хромосома, представляющая собой последовательность генов, значения которых определяют закодированный вид потенциального решения некоторой поставленной задачи;

l – длина хромосомы, определяющая число генов каждой особи;

s – оператор отбора;

r – операторы рекомбинации хромосом;

f – функция оценки степени полезности (приспособленности) хромосом;

t – критерий остановки.

Практическое применение генетических алгоритмов к задачам оптимизации предполагает необходимость кодировки пространства поиска решений. Выбранный способ кодировки будет определять структуру хромосомы как набора генов, представляющего собой некоторую бинарную последовательность. При этом, уникальная кодовая комбинация, записанная в каждой отдельной хромосоме, может рассматриваться в качестве потенциального решения соответствующей оптимизационной задачи.

Методика выполнения работы

Методику выполнения работы покажем на примере оптимизации следующей функции

$$f(x) = \begin{cases} 3\sin(8x) - \exp\left(-\left(\frac{x}{20}\right)^2\right), & \text{для } x \le 20 \\ -\exp(-1) + (x-20)(x-22), & \text{для } x \le 20 \end{cases}$$

где x изменяется в диапазоне $[-10{:}\,25]$.

Для визуального отображения вида функции $f(x)$ необходимо создать следующий код в m-файле.

```
% Код 1
y = [];
x = -10:0.2:25; NT = length(x);
for t = 1:NT
if x(t)<20
y(t) = 3*sin(2*x(t))-exp(-(x(t)/20).^2);
else
y(t) = -exp(-1)+(x(t)-20)*(x(t)-22);
end
end
plot(x,y)
```

На получившемся рис. 2.2 можно визуально оценить точку глобального минимума и зафиксировать соответствующие значение координаты по x. Для вычисления точного значения минимума можно использовать функцию min:

$[ymin, imin] = min(y);$

$ymin, x(imin).$

Результат выводится в командной строке:

$ymin = -3.9983$

$x = -0.7900.$

Примечание

Применение функции $min(f(x))$ возможно, если известны все значения функции для всего пространства аргументов, т.е. при решении задачи минимизации полным перебором.

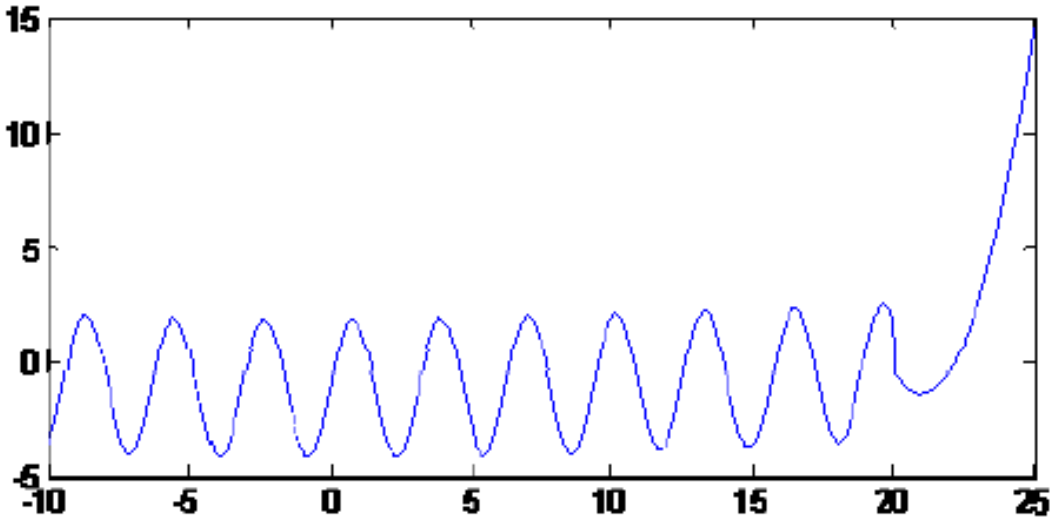

Рис. 2.2. Вид функции $f(x)$

При использовании оптимизатора GATOOL необходимо создать функцию (отдельный m -файл) со следующим кодом, вычисляющим значение функции $f(x)$:

```
%код 2
function y = func_1(x)
if x<20
y = 3*sin(2*x)-exp(-(x/20).^2);
else
y = -exp(-1)+(x-20)*(x-22);
end
```

которую требуется сохранить в рабочую (или текущую) папку под именем func_1.

Открыть мастера «генетического алгоритма» можно набрав в командной строке Matlab gatool и нажать ENTER. В появившемся окне мастера «генетического алгоритма» в строке «***Fitness function***» задать имя функции для оптимизации со следующим префиксом **@func_1**.

В строке «***Number of variables***» задать число переменных $(x_1, \dots, x_{n)}$ по которым осуществляется поиск, в данном случае 1.

В строке «***Initial Range***» задать диапазон поиска [-10; 25].

После этого можно запустить алгоритм поиска с помощью кнопки Start.

Результат выполнения оптимизации отображаются в окне Final Poin. Для заданного примера результат будет x= -0.77887.

Значение функции y = -3.99823.

Провести анализ соответствия найденных координат ГА и координат, найденных с помощью функции min перебором всех значений совпадают с точностью в четвертом знаке.

Задание

Найти минимум двухмерной функции Z

$$Z = 2 \times N \times \sin(2 \times pi \times N \times X) \times \exp(-X^2 - Y^2);$$

где параметры изменяются в диапазоне X=[-2:2] и У=[-2:2],

N – номер варианта.

1. Записать код вычисления минимума функции по сетке X,Y (полным перебором).

```
%Код 3
N = 8; % N=номер варианта;
dx = 0.01;
[X,Y] = meshgrid(-2:dx:2); % задание матриц значений
Z = 2*N*sin(2*N*pi*(X)).* exp(-X.^2 – Y.^2); % задание функции
surf(X,Y,Z) % построение графика
[Zmin] = min(min(Z)) % минимум полным перебором
```

Выполнить его и записать значение Z_{min}.

2. Создать m-файл с функцией расчета Z от параметров для gatool.

```
% код 4
function Z = func_1(x) % объявление функции
N = 8; % Номер варианта
Z = 2*N*sin(2*N*pi*(x(1))).* exp(-x(1).^2 – x(2).^2); % Расчет функции
```

где $X = X(1), Y = X(2)$. Сохранить в файл рабочую (или текущую) папку под именем «**func_1.m**».

3. Открыть мастера «генетического алгоритма» можно набрав в командной строке Matlab gatool и нажать ENTER. В появившемся окне мастера «генетического алгоритма» в строке «***Fitness function***» задать имя функции для оптимизации со следующим префиксом **@func_1**. В строке «***Number of variables***» задать число переменных (x1,…,xn) по которым осуществляется поиск, в данном случае 2. В строке «***Initial Range***» задать диапазон поиска [-2 -2; 2 2]. После этого можно запустить алгоритм поиска с помощью кнопки Start.

4. Повторить поиск ГА три раза. Оценить найденные ГА значений минимума функции Z по сравнению с полным перебором по сетке.

5. Исследовать на примерах поиска каждый параметр ГА и формат его задания, при этом необходимо изучить процесс влияния каждого из них на конечный результат.

6. Сгенерировать m-файл (File->Generate m-file) с настроенными параметрами (мутация скрещивание, миграция и т.п.

Отчет должен содержать:

тексты М-файлов, параметры настройки ГА, временные графики процесса оптимизации требуемых величин, а также график оптимизируемой функции с искомыми координатами.

Контрольные вопросы

1. Как осуществляется кодировка величин в хромосому?
2. Как влияют операции ГА на процесс оптимизации?
3. В чем отличие и особенность ГА от обычных (перебор, пузырьковый метод и т.д.)?
4. Что такое функции полезности (на паре примеров)?

ЛАБОРАТОРНАЯ РАБОТА № 3

ПРЯМОЕ УПРАВЛЕНИЕ НА НЕЧЕТКОЙ ЛОГИКЕ

Цель работы: Научить создавать и использовать системы НЛВ в качестве регуляторов прямой цепи в системах управления.

Показать возможности нечетких регуляторов по сравнению с линейными законами управления на примере П-регулятора.

Работа выполняется на программе *MATLAB.*

Время выполнения работы – 6 учебных часов.

Методика выполнения работы

Для моделирования системы с нечетким контроллером (блок Fuzzy Logic Controller) необходимо в Simulink создать модель, приведенную на рис. 3.1, и систему нечеткого логического вывода (НЛВ). В блоке Fuzzy Logic Controller в качестве параметра “Fis file or structure” необходимо задать имя переменной (например, b), которая должна содержать описание и параметры системы НЛВ. Далее требуется сформировать систему НЛВ и записать ее в объект с именем b.

Коэффициент GE предназначен для перевода сигнала ошибки к нормализованному диапазону [-1 1], в котором обычно и функционирует система НЛВ. Коэффициент GU предназначен для перевода нормализованного сигнала с выхода системы НЛВ в рабочий диапазон сигнала управления. Произведение GE*GU определяют значение общего коэффициента усиления.

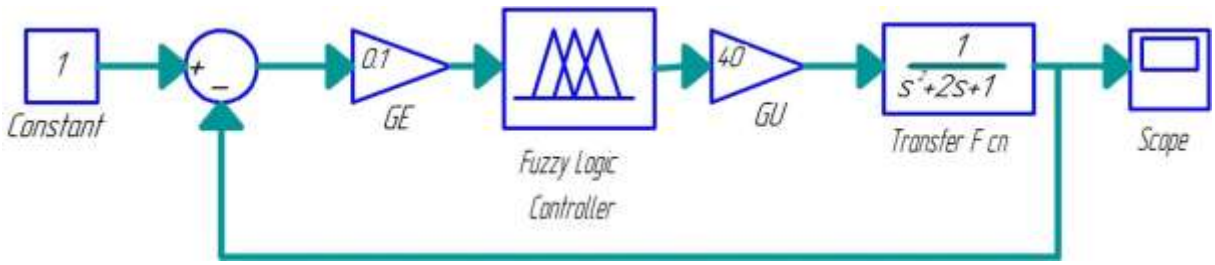

Рис. 3.1. Система управления с нечетким регулятором

Для примера рассмотрим построение линейного и нелинейного пропорционального закона управления с помощью системы НЛВ. В качестве типа НЛВ выберем алгоритм Сугено.

Для задания системы НЛВ Сугено необходимо: определить термы и их параметры для входной переменной (её назовем “Error“), значения выходной переменной (её назовем “Control“), базу правил, связывающую термы входной переменной со значениями выходной переменной.

Создать линейный закон управления можно на основании простой базы правил, содержащей три следующие правила.

ЕСЛИ Error = NB, ТО Control = NB

ЕСЛИ Error = Z, ТО Control = Z

ЕСЛИ Error = PB, ТО Control = PB.

Исходя из данных правил, можно определить, что входная переменная Error имеет три терма {“NB”–отрицательное большое значение, “Z”–нулевое значение, “PB”– положительное большое значение}. Для нормализованного случая значения (универсум) входной переменной должны находиться в диапазоне [-1 1]. Для соблюдения линейности итогового преобразования выберем треугольные функции принадлежности (ФП) для этих термов с параметрами Z = {-1, 0, 1} , PB = {0, 1, 2}, NB = {-2, -1, 0}, где первое значение левый край, втрое – центр, третье – правый край.

Выходная переменная Control имеет три значения, для нормализованного вида Z = 0, PB = 1, NB = -1, смысловое описание которых аналогично переменной Error.

Быстрее и надежнее создавать системы НЛВ, используя мфайл (функции командной строки), а редактировать и исследовать, используя пользовательский графический интерфейс (GUI). Вызов оболочки осуществляется с помощью команды **fuzzy**. Создадим в MATLAB описанную выше систему НЛВ.

```
% КОД 1.// Создание объекта системы НЛВ типа Сугено
b=newfis('fis', 'sugeno', 'prod', 'probor', 'prod', 'sum', 'wtaver');
% добавление входной лингвистической переменной Error
b=addvar(b,'input','Error',[-1 1]);
% создание ФП (термов) для входной ЛП Error
b=addmf(b,'input',1,'NB','trimf',[-2 -1 0]);
b=addmf(b,'input',1,'Z' ,'trimf',[-1 0 1]);
b=addmf(b,'input',1,'PB','trimf',[0 1 2]);
% добавление выходной ЛП Control
b=addvar(b,'output','Control',[-1 1]);
```

```
% создание ФП (термов) для выходной ЛП Control
b=addmf(b,'output',1,'NB','constant',[-1]);
b=addmf(b,'output',1,'Z','constant',[0]);
b=addmf(b,'output',1,'PB,'constant',[1]);
```

В системе НЛВ типа Сугено есть только ФП для входной переменной Error, их можно посмотреть (рис. 3.2) с помощью оператора plotmf. plotmf(b,'input',1) % рисуем полученные ФП (термы)

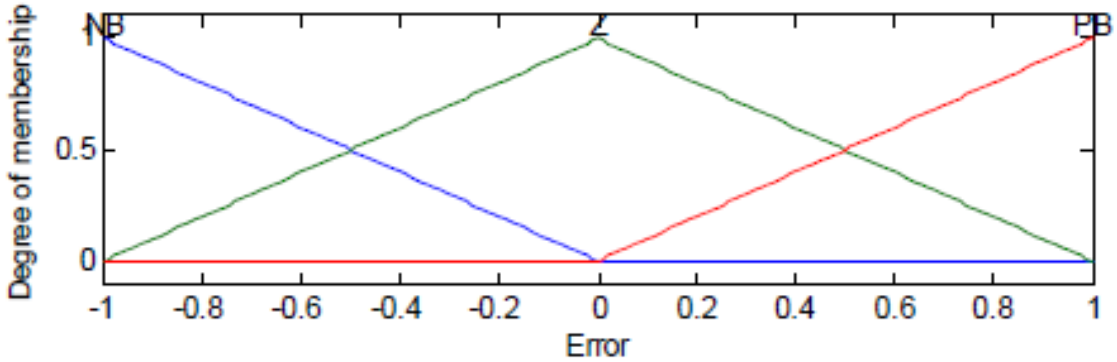

Рис. 3.2. ФП для входной переменной

Создание базы правил производится с помощью функции **b=addrule(b,ruleList)**. Количество строк матрицы правил **ruleList** равно количеству добавляемых правил, т.е. каждая строка матрицы соответствует одному правилу. Количество столбцов матрицы

равно m+n+2, где m (n) – количество входных (выходных) ЛП системы НЛВ.

Первые m столбцов соответствуют входным переменным, т.е. задают ***посылку*** правил (ЕСЛИ…). Элементы этих столбцов содержат порядковые номера термов, используемых для лингвистической оценки соответствующих входных переменных. Номера термов определяются очередностью их добавления в НЛВ.

Следующие n столбцов соответствуют выходным переменным, т.е. задают ***заключение*** правил (ТО). Элементы этих столбцов содержат порядковые номера термов, используемых для лингвистической оценки соответствующих выходных переменных.

Предпоследний столбец матрицы содержит весовые коэффициенты правил. Значения весовых коэффициентов должны быть в диапазоне [0, 1].

Последний столбец матрицы задает логические связки между переменными внутри правил. Значение 1 соответствует логической операции И, а значение 2 – логической операции ИЛИ.

Если только один вход, то значение данный параметр не имеет. Таблица соответствующих привил приведена ниже.

```
% КОД 2.// Создание базы правил
ruleList=[ 1 1 1 1; 2 2 1 1; 3 3 1 1];
b=addrule(b,ruleList); % дабавление базы правил в систему
```

Посмотреть и провести редактирование базы правил можно с помощью функции ruleedit(b)**.**

Посмотреть итоговое преобразование вход/выход системы НЛВ можно с помощью функции surfview(b)**.** Из рисунка 3.3а. видно, что получившееся преобразование – линейно. Редактирование и исследование всей системы НЛВ можно провести с пользовательского графического интерфейса, выполнив в командной строке fuzzy(b).

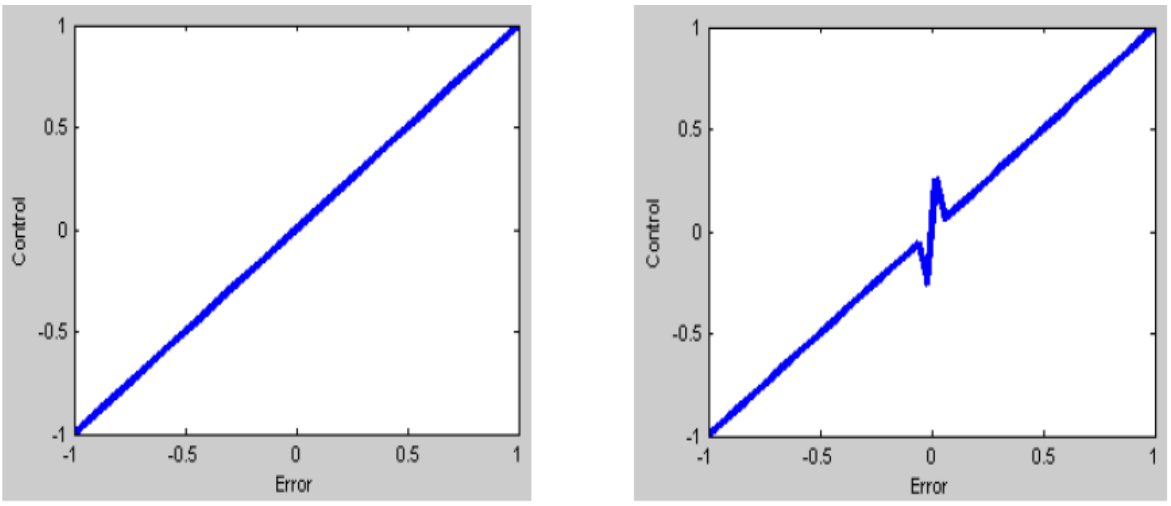

Рис. 3.3. Итоговые преобразования вход выход для базы из трех правил (а) и пяти правил (б)

Промоделировав систему управления с линейным П-регулятором (т.е. запустив модель в Simulink), можно увидеть (рис. 3.4), что ошибка в установившемся режиме очень большая (20 %).

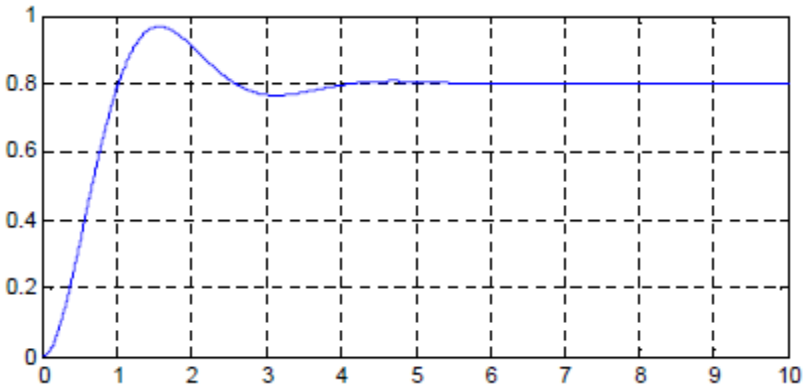

Рис. 11.4. Переходный процесс в системе с линейным П-регулятором

Для коррекции поведения системы необходимо добавить правила и соответствующие ФП, увеличивающие общий коэффициент усиления вблизи нуля. Сформулировать данные правила можно следующим образом.

ЕСЛИ Error = NS, ТО Control = NS

ЕСЛИ Error = PS, ТО Control = PS.

ФП NS (Negative Small) и PS (Positive Small) должны действовать только в близи нуля, можно задать им следующие значения PS = {0, 0.02, 0.05}, NS = {-0.05, -0.02, 0}. Зададим новые значения выходной переменной PS = 0.5, NS = -0.5. Добавляем соответствующие термы и уточняющие правила в систему НЛВ.

```
% КОД 3.// добавляем новые ФП (термы)
b=addmf(b,'input',1,'NS','trimf',[-0.05 -0.02 0]);
b=addmf(b,'input',1,'PS','trimf',[0 0.02 0.05]);
b=addmf(b,'output',1,'NS','constant', [-0.5]);
b=addmf(b,'output',1,'PS','constant', [0.5]);
ruleList=[ 4 4 1 1; 5 5 1 1]; % создаем дополнительные правила
b=addrule(b,ruleList); % дабавление базы правил в систему
```

Можно нарисовать все ФП (термы) входной переменной (рис. 3.5) plotmf(b,'input',1)

Посмотреть итоговое преобразование вход/выход системы НЛВ с пятью правилами можно с помощью функции **surfview(b)** (для правильного отображения характеристики требуется увеличить число точек X grids до 50). Из рисунка 3.3б. видно, что получившееся преобразование – нелинейно вблизи нуля, что и требовалось.

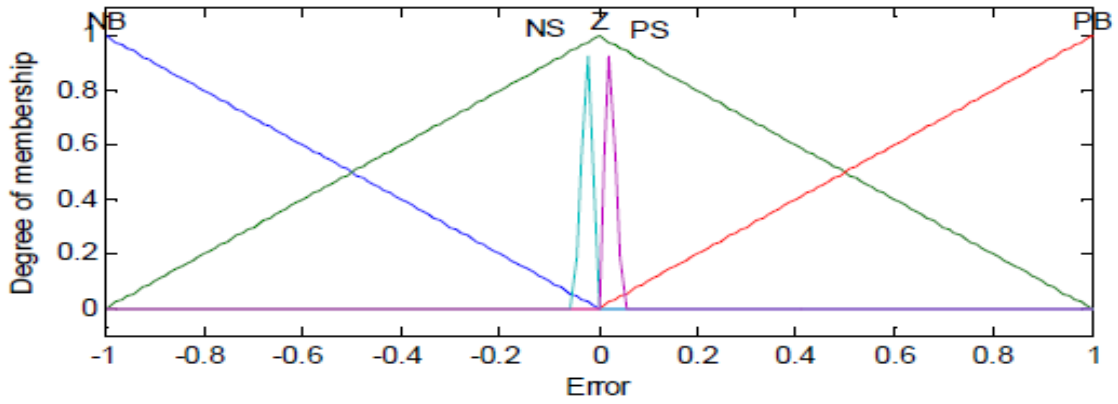

Рис. 3.5. ФП для входной переменной

Моделирование, полученной системы управления с нелинейным П-регулятором, подтверждает предположение об увеличении точности в установившемся режиме (рис. 3.6).

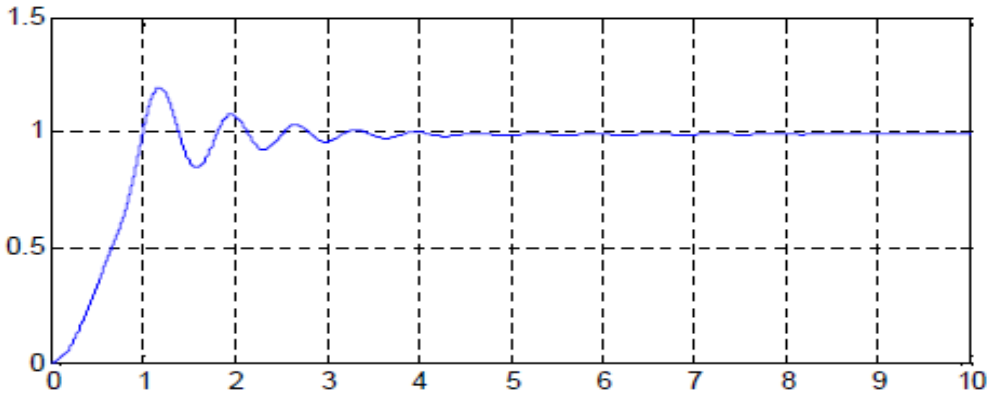

Рис. 3.6. Переходный процесс в системе
с нелинейным П-регулятором

Попытки увеличить точность в системе с линейным П-регулятором приводят к возникновению большого перерегулирования, не уменьшая время регулирования. На рис 3.7 приведены переходные процессы в системе с линейным П-регулятор (тонкая линия) с общим коэффициентом усиления Кп = 40 (ошибка в установившемся режиме равна 2,4 %, а перегулирование 55%) и в системе с нечетким П-регулятором с общим коэффициентом усиления Кп = 3 (ошибка в установившемся режиме равна 1,3 %, а перегулирование 18%).

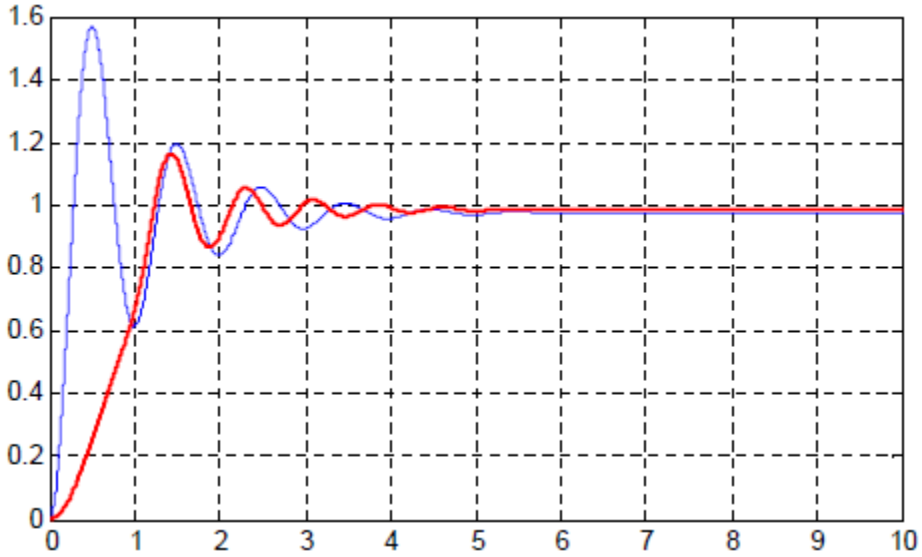

Рис. 3.7. Переходный процесс в системе
с нелинейным П-регулятором

Таблица 3.1

	1	2	3	4	5	6	7	8	9	10
K	1	1	1	1.5	1.5	1.5	1.5	2	2	2
T	1	1.5	2	0.8	1	1.5	2	1	1.5	2

Задание

1. Создать модель системы управления с нечетким контроллером в Simulink в соответствии с рис. 11.1. Задать передаточную функцию W(s) = K/(T2s+2Ts+1). Параметры взять из таблицы 11.1. Задать GE = 0.1 и GU=30. В блоке Fuzzy Logic Controller в качестве параметра “Fis file or structure” необходимо задать имя переменной b.

2. Создать м-файл, реализующий линейный П-регулятор на системе НЛВ и состоящий из кодов 1-2. Выполнить его (создать переменную b).

3. Запустить модель в Simulink и получить переходный процесс. Изменяя значение GU от 30 до 500 найти переходный процесс, обеспечивающий наилучшее качество (перегулирование и время регулирование) при условии, что точность в установившемся режиме должна быть меньше 5 %. И для этого значения GU определить качество управления.

4. Добавить в м-файл код 3, реализующий нелинейный П-регулятор на системе НЛВ. Выполнить его (создав параметр b).

3. Запустить модель в Simulink и получить переходный процесс. Изменяя значение GU от 30 до 500 найти переходный процесс, обеспечивающий наилучшее качество (перегулирование и время регулирование) при условии, что точность в установившемся режиме должна быть меньше 5 %. И для этого значения GU определить качество управления.

Отчет по заданию

Схема модели в Simulink, значение GU и наилучшие показатели качества для линейного П-регулятора, значение GU и наилучшие показатели качества для нелинейного П-регулятора, сравнение качества управления линейного и нелинейного П-регулятора.

Контрольные вопросы

1. Какие требования предъявляются к функциям принадлежности?
2. Как создать систему НЛВ Сугено с линейной функцией преобразования?
3. Какие отличия между системами НЛВ Мамдани и Сугено?
4. В чем отличие системы НЛВ реализующей П и ПД- регулятор?

ЛАБОРАТОРНАЯ РАБОТА № 4

РАБОТА FUZZY LOGIC С БЛОКАМИ SIMULINK

Цель работы: изучить методику интеграции инструментов *Fuzzy Logic Toolbox* с блоками *Simulink* в программе *MatLab*.

Работа выполняется на программе *MatLab*.

Время выполнения работы – 6 учебных часов.

Теоретическая часть

Системы нечёткого вывода, созданные тем или иным образом с помощью пакета *Fuzzy Logic Toolbox*, допускают интеграцию с инструментами пакета *Simulink*, что позволяет выполнять моделирование систем в рамках последнего. Рассмотрим это на примере контроля уровня воды в баке [3].

Пример 1. Контроль уровня воды в баке.

На рис. 4.1 изображен объект управления в виде бака с водой, к которому подходят две трубы: через одну трубу, снабженную краном, вода втекает в бак, через другую – вытекает. Подачу воды в бак можно регулировать, больше или меньше открывая кран. Расход воды является неконтролируемым и зависит от диаметра выходной трубы (он фиксирован) и от текущего уровня воды в баке. Если понимать под выходной (регулируемой) переменной уровень воды, а под регулирующим элементом – кран, то можно отметить, что подобный объект регулирования, с точки зрения его математического описания, является динамическим и существенно нелинейным.

Определим цель управления здесь как установление уровня воды в баке на требуемом (изменяющемся) уровне и попробуем решить соответствующую задачу управления средствами нечеткой логики.

Очевидно, в регулятор, обеспечивающий достижение цели управления, должна поступать информация о несоответствии (разности) требуемого и фактического уровней воды, при этом

данный регулятор должен вырабатывать управляющий сигнал на регулирующий элемент (кран).

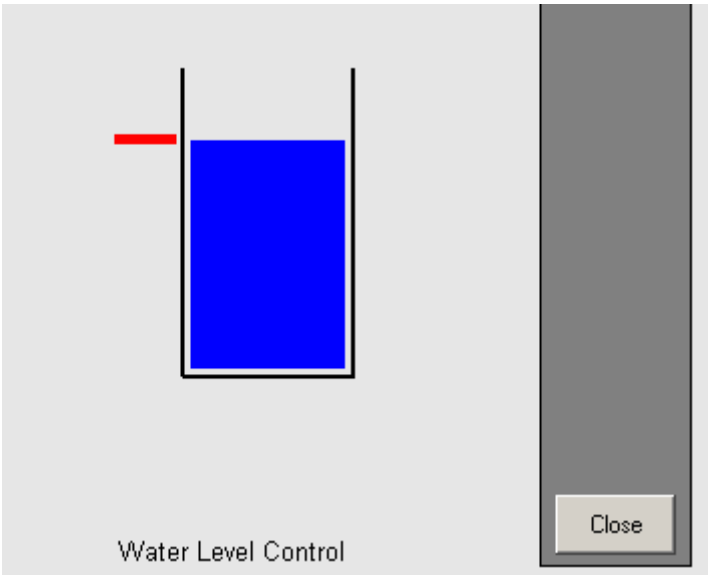

Рис. 4.1. Схематическое представление объекта управления (бака с водой)

В первом приближении функционирование регулятора можно описать набором из следующих правил:

1. If (level is okay) then (valve is no_change) (1)
2. If (level is low) then (valve is open_fast) (1)
3. If (level is high) then (valve is closefast) (1)
4. If (level is okay) and (rate is positive) then (valve is close_slow) (1)
5. If (level is okay and (rate is negative) then (valve is openslow) (1),

что в переводе означает:

1. Если (уровень соответствует заданному), то (кран без изменения) (1)
2. Если (уровень низкий), то (кран быстро открыть) (1)
3. Если (уровень высокий), то (кран быстро закрыть) (1)
4. Если (уровень соответствует заданному) и (его прирост – положительный), то (кран надо медленно закрывать) (1)
5. Если (уровень соответствует заданному) и (его прирост – отрицательный), то (кран надо медленно открывать) (1)

Модель системы управления уровнем воды в баке с нечетким регулятором, основанным на приведенных правилах, является одной из демонстрационных моделей пакетов *Fuzzy Logic* и *Simulink.*

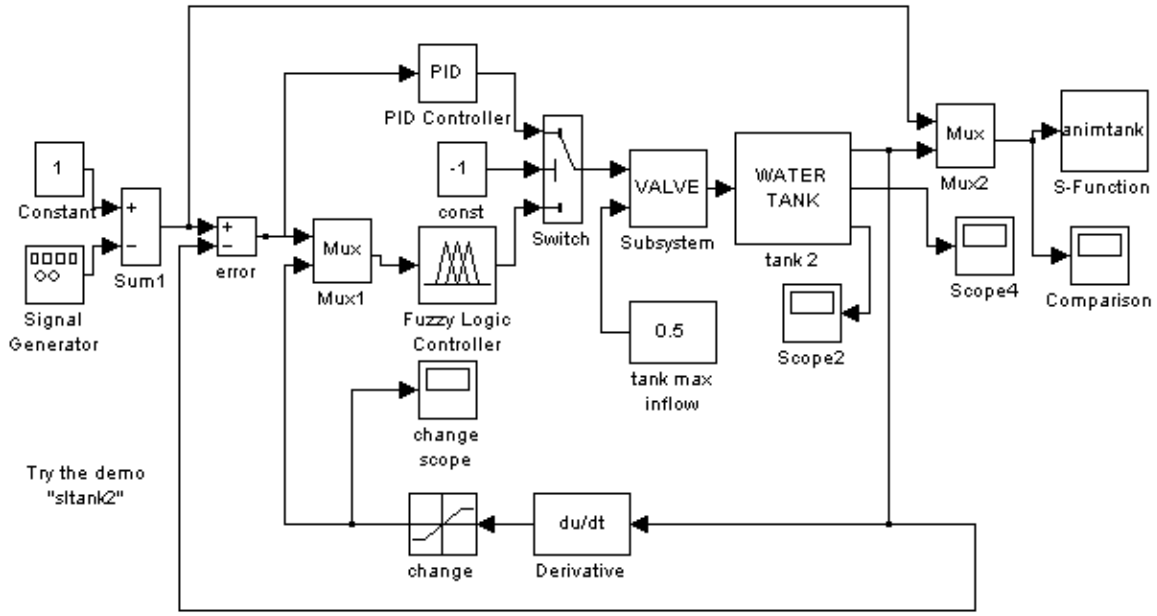

Рис. 4.2. Блок-диаграмма модели системы управления уровнем воды в баке нечётким регулятором

Ее можно вызвать из командной строки командой **sltank,** что приведет к открытию окна *Simulink* с блок-диаграммой указанной модели (рис. 4.2).

Одновременно блок *Fuzzy Logic Controller* будет сопоставлен с системой нечеткого вывода, записанной в файле *tank.fis*.

Для изучения процесса функционирования системы необходимо нажать кнопку ***Start*** панели инструментов блок-диаграммы модели и дважды щелкнуть левой кнопкой мыши по блоку *Comparison* (Сравнение). В появившемся окне этого блока будут показаны изменяющиеся во времени сигналы заданного (последовательность импульсов прямоугольной формы) и фактического уровней воды (рис. 4.3).

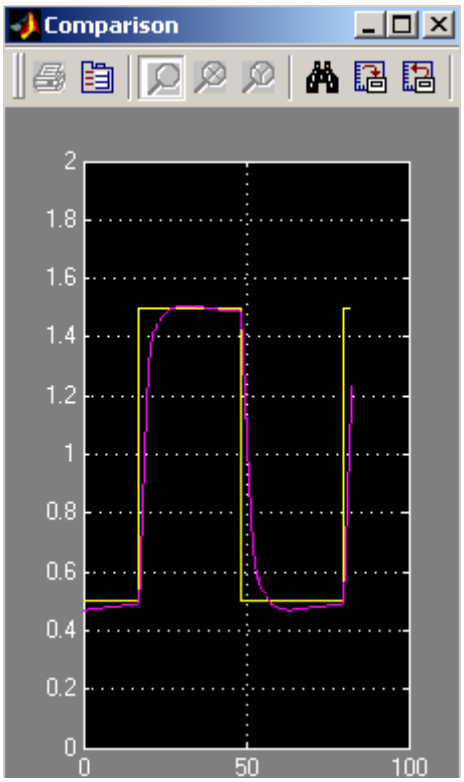

Рис. 4.3. Результаты моделирования системы управления с нечётким регулятором.

Как видно из графиков, представленных на рис. 3, переходный процесс в системе имеет апериодическую форму и заканчивается достаточно быстро, т.е. качество регулирования следует признать хорошим.

Построение нечеткой модели с использованием блоков Simulink

Для построения собственной моделирующей системы с использованием средств нечеткой логики и блоков *Simulink* рекомендуется скопировать блок *Fuzzy Logic Controller* из рассмотренной системы *sltank* (или какого-либо другого демонстрационного примера *MATLAB*) и поместить его в блок-диаграмму разрабатываемой системы. Из командной строки командой **fuzblock** можно также открыть библиотеку нечетких блоков пакета Simulink, которые могут быть использованы точно так же, при необходимости – с дополнительным редактированием.

Отметим, что функционирование указанных блоков осуществляется с использованием системной S-функции **sffis.mex**. Запись этой функции такова:

output = sffis(t, x, u, nag, nsmat),

где **output** – выход нечеткого регулятора,

t, x и **flag** – стандартные аргументы системной функции Simulink,

fismat – идентификатор (имя) нечеткой системы вывода, и входной сигнал (вектор входных сигналов) регулятора.

Данная функция оптимизирована для работы в среде *Simulink.*

Демонстрационные примеры работы с пакетом Fuzzy Logic Toolbox.

Для ознакомления с пакетом Fuzzy Logic Toolbox можно использовать следующие функции (команды) в режиме командной строки:

defuzzdm – обзор методов приведения к четкости (дефазификации),

fcmdemo – демонстрация алгоритма кластеризации *Fuzzy c-means* (2-D графика),

fuzdemos – демонстрация графического интерфейса пакета *Fuzzy Logic*,

gasdemo – демонстрация использования аппарата гибридных сетей для решения задачи о выборе автомобиля, наиболее экономичного по расходу топлива,

juggler – демонстрация системы жонглирования мячом с использованием нечеткого регулятора,

invkine – демонстрация нечеткого управления движением робота-манипулятора,

irisfcm – демонстрация алгоритма кластеризации Fuzzy c-means,

noisedm – демонстрация решения задачи фильтрации на основе методов нечеткой логики,

slbb – демонстрация задачи «шар на качелях»,

slcp – демонстрация нечеткой системы управления перевернутым маятником,

sltank – демонстрация системы управления уровнем воды в баке с нечетким регулятором,

sltankrule – то же, что в предыдущем случае, но с дополнительным просмотром нечетких правил,

sltbu – демонстрация функционирования нечеткой системы управления грузовиком.

С каждым из этих примеров связан *M-файл, fis-файл* или/и блок-диаграмма *Simulink*, запуск которых производится с помощью одной из указанных команд. Текст пояснений в примерах – на английском языке. Данные примеры доступны и через главное меню *MATLAB* (пункт *Help/Examples and Demos*, раздел *Toolboxes/Fuzzy Logic*). Дополнительную информацию можно получить, используя команду **help fuzzy.**

Программа работы

Составьте модель системы регулирования с использованием средств нечеткой логики для объектов, представленных в табл. 4.1. Представьте схему управления и результаты моделирования.

Таблица 4.1

Вариант	**Система регулирования**
1	Система автоматической стабилизации космического аппарата в инерциальной системе координат
2	Система автоматической подачи резца при точении детали (копирование с образца)
3	Система управления солнечными батареями космического аппарата
4	Система управления телевизионной антенной
5	Система стабилизации количества кислорода в атмосфере космической станции
6	Система автоматического управления освещенностью помещения
7	Система управления ориентацией искусственного спутника Земли относительно земной поверхности

8	Система стабилизации курса пассажирского самолета
9	Система управления транспортным роботом в цехе
10	Система стабилизации давления газа в баллоне

Содержание отчета

- цель работы;
- задание;
- краткое описание действий по пунктам;
- графики по всем пунктам программы;
- выводы по работе.

ЛАБОРАТОРНАЯ РАБОТА №5

РАЗРАБОТКА ЭКСПЕРТНОЙ СИСТЕМЫ УПРАВЛЕНИЯ МОБИЛЬНЫМ ОБЪЕКТОМ

Цель работы: лабораторной работы является получение практических навыков в создании экспертных систем для конкретных задач с использованием оболочки экспертной системы Expert 2.0.

Для успешного освоения приведенного ниже материала необходимо ознакомление с основными понятиями экспертных систем и с «Руководством пользователя оболочки экспертной системы Expert 2.0».

Время выполнения работы – 6 учебных часов.

Описание имитатора «ARENA».

«ARENA» — компьютерный пошаговый имитатор действий мобильных роботов. На поле размером 10 на 10 позиций (рис. 5.1) располагаются 3 робота – голубой, зеленый и синий. Каждый робот имеет 5 жизненных ресурсов. Существует 5 вариантов действий робота за один шаг имитации: перемещение вперед, поворот вправо, поворот влево, выстрел, бездействие. Управление роботом заключается в выборе одного из этих действий в зависимости от ситуации на поле.

Стратегия действий роботов должна быть направлена на выживание робота при одновременном уничтожении других роботов. Каждое попадание в робота уменьшает на единицу количество его жизненных ресурсов. Дальность выстрела робота – 5 клеток. Направление выстрела и движения вперед указывается стрелкой в той позиции, в которой находится робот соответствующего цвета. Студентом разрабатывается экспертная система, которая анализирует ситуацию на поле и управляет роботом путем выбора одного из возможных действий.

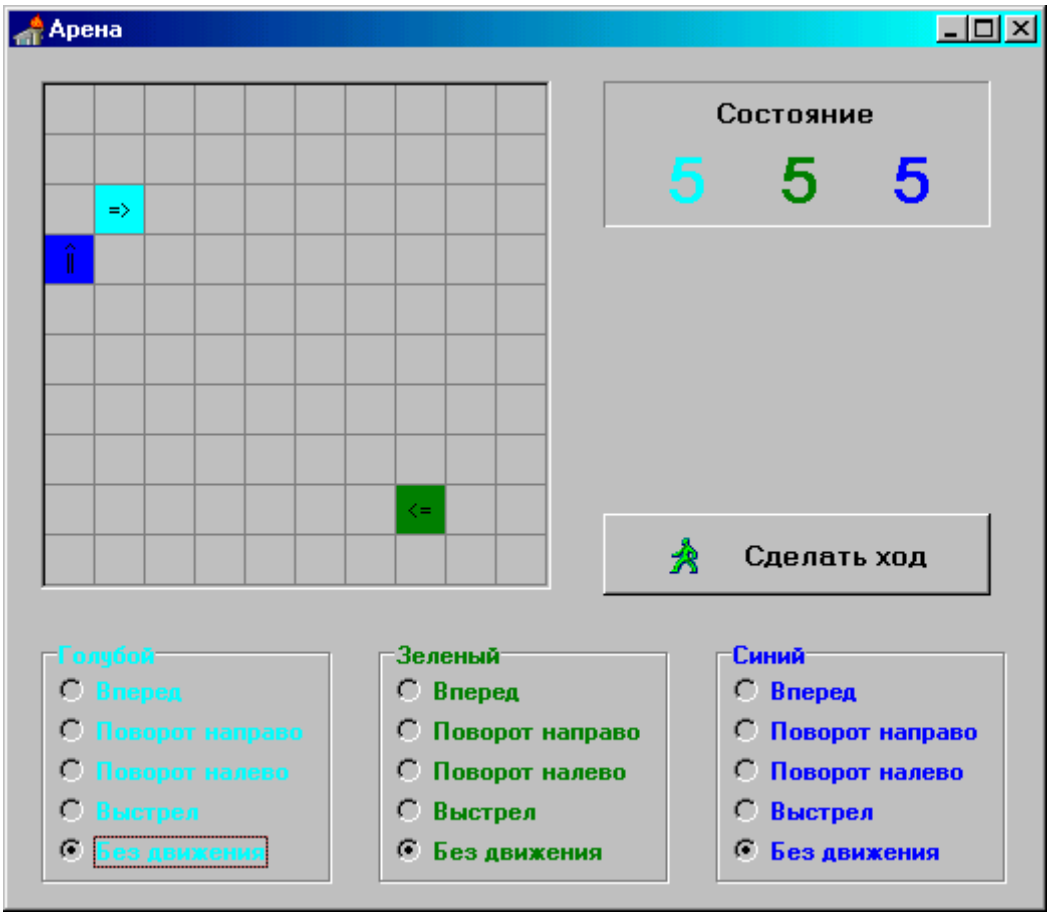

Рис. 5.1. Окно имитатора «ARENA»

Имитация действий происходит следующим образом. Запущенные на одном компьютере три экспертных системы (по одной для каждого робота) после анализа ситуации указывают необходимые действия для своих роботов. Эти действия устанавливаются при помощи мыши в окне имитатора, затем нажимается кнопка «сделать ход».

Происходит поочередное выполнение указанной операции каждым роботом. При этом голубой робот отрабатывает свое действие первым, затем зеленый и синий. Начальное положение и ориентация роботов задаются имитатором случайным образом.

Информация о ситуации на поле.

В корневом каталоге жесткого диска располагаются файлы с информацией о состоянии процесса имитации действий роботов, в которые после каждого хода заносится различная информация для каждого из роботов (таблица 5.1).

После букв в названиях файлов идет цифра, определяющая цвет робота, к которому относится информация в файле: 1 – голубой, 2 – зеленый, 3 – синий. Например в файле O2.dat хранится текущая ориентация зеленого робота, а в файле O3.dat – ориентация синего робота.

Таблица 5.1.

Имя файла	Смысловое значение	Диапазон значений
D.dat	Количество оставшихся жизненных ресурсов	0…5
LA.dat	Предыдущий ход (0 – вперед, 1 – поворот направо, 2 – поворот налево, 3 – выстрел, 4 – без движения)	0…4
O.dat	Ориентация робота на поле: 1 – направлен вверх, 2 – вправо, 3 –вниз, 4 – влево	1…4
PLA.dat	Пред- предыдущий ход робота	0…4
R.dat	Дистанция до объекта впереди робота(если по курсу движения робота никаких объектов нет, то число 10)	1…10
RA.dat	Результат выполнения последнего действия	0…1
RS.dat	Признак попадания выстрела в робота (1 – в робота попал выстрел противника, 0 – на данном ходе попадании не было)	0…1
X.dat	Позиция робота по горизонтали (1 – слева, 10 – справа)	1…10
Y.dat	Позиция робота по вертикали (1 – внизу, 10 – вверху)	1…10

Результат выполнения последнего действия робота (файл RA.dat) содержит 1, если последнее указанное ему действие робот выполнил, и 0 – если выполнить это действие не удалось (например нельзя сделать ход вперед, если впереди стоит другой робот).

Ход выполнения работы.

В процессе выполнения лабораторной работы необходимо решить следующие задачи:

1. Разработать стратегию управления одним из роботов, приводящую к победе этого робота.
2. Составить набор правил, реализующих эту стратегию.
3. Составить алгоритм действий на основе созданного набора правил и входных данных.
4. С помощью оболочки ЭС Expert 2.0 реализовать полученный алгоритм в виде структуры БЗ.
5. Обучить все узлы создаваемой экспертной системы.
6. Проверить работоспособность созданной ЭС.

Проверка выбранной стратегии и работоспособности созданной ЭС.

Для проверки ества выбранной стратегии и работоспособности созданной ЭС на одном компьютере запускается программа «ARENA» и три оболочки ЭС «EXPERT 2.0» с тремя разными проектами, реализующими различные стратегии поведения роботов. Каждая экспертная система управляет одним из роботов. Перед каждым ходом все три экспертные системы выдают свои рекомендации по управлению роботами, эти варианты действий устанавливаются в программе «ARENA» и нажимается кнопка «Сделать ход». Затем «ARENA» отображает на экране монитора изменение ситуации на поле и обновляет данные игрового процесса в соответствующих файлах на жестком диске. После этого последовательность действий повторяется до гибели одного из роботов.

Пример выполнения работы (управление синим роботом).

Описание стратегии:

Если впереди есть противник и он находится на расстоянии выстрела, то робот делает выстрел, если расстояние до противника больше 6, робот приближается к нему;

Если впереди противника нет, то робот перемещается в левый нижний угол, и поочередно поворачивается влево и вправо (т.е. горизонтально или вертикально) и ждет появления противника.

Для работы ЭС необходима следующая информация: дистанция до объекта впереди, ориентация и координаты синего робота на поле. Для синего робота это, соответственно, файлы R3.dat, O3.dat, X3.dat, Y3.dat.

Создание экспертной системы.

1. Запускается программа «EXPERT 2.0», создается новый проект (см. руководство пользователя).
2. Добавляются в проект 4 элемента «Ввод из файла», в которых указываются полные имена файлов R3.dat, O3.dat, X3.dat, Y3.dat. Числовые значения, содержащиеся в эих файлах, будут являться входными данными для узлов экспертной системы.
3. Добавляются в проект основные элементы системы – узлы. Создается структура базы знаний в соответствии со стратегией управления роботом. Для этого устанавливаются все необходимые связи между элементами структуры базы знаний.
4. Обучаются узлы базы знаний.
5. Добавляются в проект элементы «Вывод на экран» для выдачи результатов экспертизы на монитор, проверяются связи проекта.

Экспертная система готова к проведению экспертизы.

Содержание отчета

В отчете должна быть указана цель работы, описана стратегия управления роботом, приведен алгоритм действий, реализующий эту стратегию, составленная на основе этого алгоритма структура базы знаний, таблицы с базами примеров для обучения узлов экспертной системы.

Библиографический список

1. Дьяконов В. П. MATLAB 6/6.1/6.5 + SIMULINK 4/5 в математике и моделировании. – М.: СОЛОН-Пресс, 2003. – 565 с.
2. Круглов В. В. Нечёткая логика и искусственные нейронные сети: учеб. пособие / В. В. Круглов, М.И. Дли, Р. Ю. Голунов. – М.: Изд-во Физико-математической литературы, 2001. – 224 с.
3. Matlab 6.5 SP1/7/7 SP1/7 SP2 + Simulink 5/6. Инструменты искусственного интеллекта и биоинформатика. Серия «Библиотека профессионала». – М.: СОЛОН-ПРЕСС, 2006. – 456 с.
4. Барский А. Б. Нейронные сети: распознавание, управление, принятие решений. – М.: Финансы и статистика, 2004. – 175 с.
5. Еремин Д. М. Искусственные нейронные сети в интеллектуальных системах управления: учеб. пособие / Д. М. Еремин, И. Б. Гарцеев. – М.: МИРЭА, 2004. – 75 с.
6. Круглов В. В. Искусственные нейронные сети. Теория и практика / В. В. Круглов, В. В. Борисов. – М.: Горячая линия-Телеком, 2002.
7. Борисов В. В. Нечеткие модели и сети / В. В. Борисов, В. В. Круглов, А. С. Федулов. – М.: Горячая линия-Телеком, 2007. – 283 с.
8. Хайкин С. Нейронные сети: Полный курс: пер. с англ. – М.: Вильямс, 2008. – 1103 с.
9. Редько В. Г. Эволюция, нейронные сети, интеллект: Модели и концепции эволюционной кибернетики – М.: ЛИБРОКОМ, 2011. – 220 с.
10. http://matlab.exponenta.ru/fuzzylogic/book1/index.php (С. Д. Штовба "Введение в теорию нечетких множеств и нечеткую логику")
11. http://matlab.exponenta.ru/fuzzylogic/book5/index.php (А. П. Ротштейн "Интеллектуальные технологии идентификации")