Министерство образования Российской Федерации

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Л.В. Бабко, В.С. Королев, Р.П. Строганов, В.Д. Ярмийчук**

# КОМПЬЮТЕРНОЕ УПРАВЛЕНИЕ ТЕХНИЧЕСКИМИ СИСТЕМАМИ

**Учебное пособие**

Санкт-Петербург
2011

УДК 681.51.01:004.9(075.8)

Компьютерное управление техническими системами. / Л.В. Бабко, В.С. Королев, Р.П. Строганов, В.Д. Ярмийчук. Учебное пособие. Изд. 3-е, перераб. и доп. Санкт-Петербургский государственный политехнический университет. СПб., 2011, 124 с.

Пособие предназначено для студентов, обучающихся по направлению 220400 – Управление в технических системах, профилю – Управление и информатика в технических системах и магистерской программе – Компьютерные интеллектуальные системы управления. Основное внимание уделено алгоритмизации задач управления и методам их решения. Рассмотрены особенности и структурно-функциональная организация систем управления на примерах задач программно-логического управления, применения промышленных регуляторов, а также типовых задач оптимизации и иерархического управления. Приведены постановки и примеры задач нечеткого управления, даны основные понятия нечетких множеств и аппарата нечеткой логики для построения систем управления на основе нечеткой технологии. Рассмотрены также особенности применения ЭВМ в системах управления.

Пособие ориентировано как на изучение теоретического курса, так и на использование при выполнении научно-исследовательских работ студентов в лаборатории.

# Содержание

# ВВЕДЕНИЕ

Под термином «компьютерное управление» здесь и далее будет пониматься управление объектами, технологическими процессами и системами с помощью вычислительных машин.

Как известно, основой процесса управления являются сбор и обработка информации в соответствии с поставленными задачами управления. Наилучшим средством получения и обработки информации на современном этапе являются вычислительные машины в различных формах их организации. Появление в конце XX века надежных и относительно дешевых компьютеров, снабженных «дружественным» интерфейсом, обусловило широкое их внедрение в различные сферы, в том числе и в управление техническими объектами и технологическими процессами.

При использовании ЭВМ в качестве управляющего устройства приходится решать три основных группы проблем: алгоритмические, программные и аппаратные.

В состав алгоритмических проблем входит формирование функциональных задачи управления и выбор или разработка методов их решения.

Программные проблемы включают выбор операционной среды реального времени, в которой будет функционировать управляющее устройство, и создание прикладного программного обеспечения, реализующего разработанные алгоритмы.

К аппаратным проблемам относится выбор архитектуры устройства управления и определение номенклатуры и количества его блоков, включая блоки сопряжения с объектом управления. В случае использования "готовой" ЭВМ - оценка ее ресурсов для решения рассматриваемой задачи.

Пособие имеет цель познакомить студентов с вопросами организации компьютерного управления. Круг задач, подлежащих решению при создании управляющего устройства на основе ЭВМ, необычайно широк. В данном пособии основное внимание уделено методам решения и алгоритмизации задач обработки информации с целью контроля или формирования управляющих воздействий на объект управления.

Достаточно подробно рассматриваются примеры алгоритмизации решения сложных типовых задач управления, таких как задачи оптимизации, иерархического управления, а также примеры обоснования и формализации задач нечеткого управления, интерес к которым в последнее время существенно вырос.

Наконец, кратко описаны организация и программное обеспечение устройств связи с объектом в системах управления, использующих персональные компьютеры.

# 1. ОСОБЕННОСТИ КОМПЬЮТЕРНОГО УПРАВЛЕНИЯ

Для характеристики особенностей компьютерного управления будем сравнивать его с аналоговым управлением, когда входная и выходная информация представлены в форме непрерывных физических величин, а ее обработка осуществляется элементами аналоговой вычислительной техники.

Первой отличительной особенностью компьютерного управления является его гибкость, так как реализация того или иного управления производится в форме алгоритмов, представляемых в компьютерах в виде комплексов легко заменяемых программных средств.

При этом:

- нет принципиальных препятствий реализации алгоритмов любой сложности;
- изменение алгоритмов управления с целью, например, улучшения функционирования объекта не требует изменения технических средств управляющего устройства, а может потребовать лишь изменения программы;
- появляется возможность использования предшествующего опыта управления, аккумулированного в форме баз данных или знаний.

Второй отличительной особенностью компьютерного управления является использование цифровой формы представления информации, что позволяет:

- производить обработку данных с заранее прогнозируемой точностью,
- сравнительно просто выполнять логические операции и формировать соответствующие заключения.

Третья особенность – квантование сигналов по времени и уровню. Квантование обусловлено особенностями представления и обработки информации в цифровых вычислительных машинах, т.к. современные ЭВМ в основном строятся по фон-неймановскому принципу, предусматривающему последовательный режим обработки информации в едином устройстве.

На рис. 1.1 показаны источники квантования информации при использовании компьютера для управления объектом непрерывного типа, снабженного аналого-цифровым (АЦП) и цифро-аналоговым преобразователями (ЦАП).

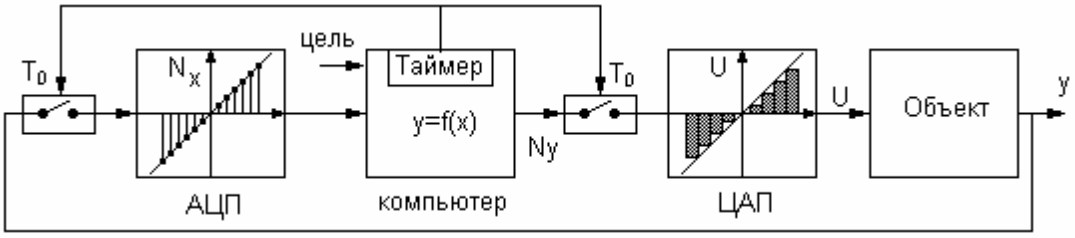



Рис. 1.1. Структурная схема системы прямого компьютерного управления объектом непрерывного типа

В связи с тем, что алгоритмы обработки информации компьютерами обсуждаются в последующих разделах пособия, здесь остановимся лишь на рассмотрении особенностей, обусловленных квантованием информации по времени и уровню.

**Квантование по времени**

Квантование по времени формирует из непрерывного сигнала последовательность импульсов, амплитуда которых соответствует значениям непрерывного сигнала в дискретные моменты времени, задаваемые тактом квантования $T_0$. Как правило, такт квантования выбирается постоянным. Его величина зависит не только от динамики самого сигнала, но в системах управления определяется также и динамическими характеристиками элементов системы в целом.

Если отвлечься от специфических требований, предъявляемых в каждом конкретном случае к качеству переходных процессов, то замена непрерывного сигнала импульсным может быть выполнена на основе теоремы Шеннона-Котельникова. Ее содержание можно сформулировать следующим образом.

Для того чтобы непрерывный сигнал со спектром, ограниченным максимальной частотой $\omega_{max}$, можно было точно восстановить по последовательности его дискретных значений, необходимо, чтобы частота квантования $\omega_0$ удовлетворяла условию $\omega_0 \geq 2\omega_{max}$. Если условие теоремы Шеннона-Котельникова не выполняется, то возможны сильные искажения непрерывного сигнала. Характерным в этом смысле является эффект, получивший название «транспонирование частоты». Рассмотрим его на следующем примере.

Пусть производится квантование гармонического сигнала частоты $\omega_1$, причем частота квантования $\omega_0$ выбрана из условия $2\omega_1 > \omega_0 > \omega_1$

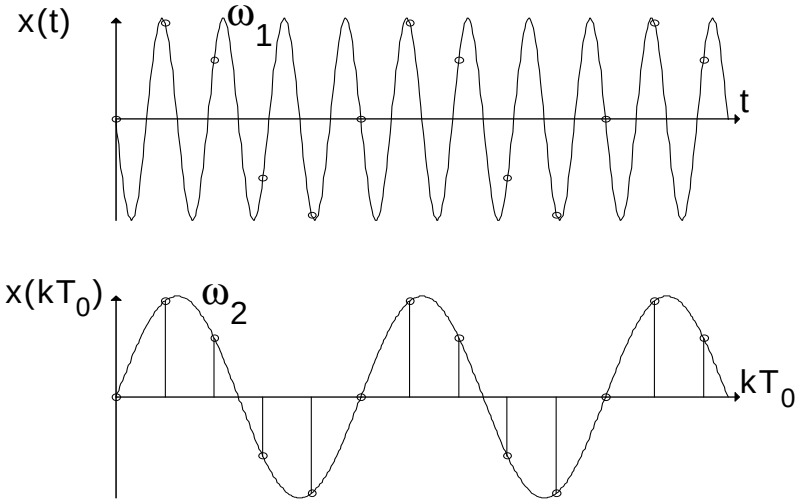



Рис. 1.2.Транспонирование частоты

Как следует из приведенного на рис. 1.2 процесса, в результате не-выполнения условий теоремы Шеннона-Котельникова генерируется

низкочастотный сигнал, частота которого $\omega_2 = \omega_0 - \omega_1$, а амплитуда в пределе может быть равна амплитуде входного сигнала.

Такой низкочастотный транспонированный сигнал может стать причиной появления гармонических колебаний в системе.

Несоблюдение условий теоремы Шеннона-Котельникова приводит к возникновению и другого эффекта, получившего название "поглощение частоты" [1,2 ].

Важнейшим следствием квантования сигналов является необходимость применения математического аппарата, учитывающего дискретизацию времени. Последнее обусловлено тем, что при квантовании информация известна только в дискретные моменты времени, кратные $T_0$ (рис. 1.3)

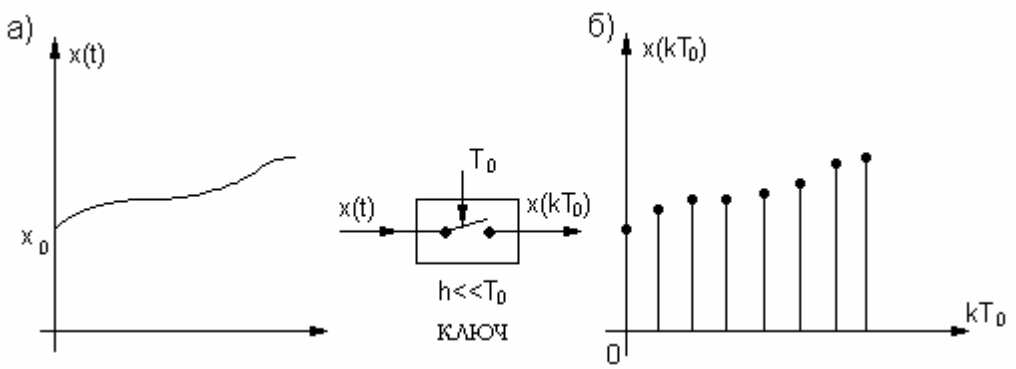



Рис. 1.3

Если квантование по времени смоделировать в форме ключа, у которого время замкнутого состояния $h << T_0$, то математическую модель квантования можно представить в следующей форме

$$x(t)=\begin{cases} x(kT_0) & \text{при } t = kT_0, \\ 0 & \text{при } kT < t < (k+1)T_0 \end{cases} \qquad (1.1)$$

k=0,1,2,...

Описание типа (1.1) является нелинейным и не удобным для анализа квантованных процессов. Возникающие трудности можно преодолеть при использовании различных способов аппроксимации сигнала между точками квантования. Наибольшее распространение получила ступенчатая аппроксимация.

Модели процессов в форме дифференциальных уравнений заменяются моделями в форме алгебраических уравнений для дискретных моментов времени.

Пусть, например, непрерывная система задана следующими линейными векторно-матричными уравнениями состояния:

$$\left.\begin{aligned}\frac{dx(t)}{dt} &= Ax(t)+Bu(t),\\ y(t) &= Cx(t),\end{aligned}\right\} \qquad (1.2)$$

где x(t) – вектор координат состояния

u(t) – вектор управления

y(t) – вектор выходных переменных

A, B и C – матрицы параметров системы и измерителя.

Пусть $t_k$ – к-ый момент квантования (k=0,1,2,…,n).

Для стационарной системы ее состояние в момент $t_k<t<t_{k+1}$ (при известном состоянии в момент $t_k$) можно записать в виде

$$x(t)=e^{A(t-t_k)}x(t_k)+\int_{t_k}^{t} e^{A(t-\tau)}Bu(\tau)d\tau$$

В следующий (очередной) момент квантования $t_{k+1}$ состояние системы определяется уравнением

$$x(t_{k+1})=e^{A(t_{k+1}-t_k)}x(t_k)+\int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\tau)}Bu(\tau)d\tau$$

Если $u(t_к)$=const на интервале $t_{k+1}-t_к=T_0$, то уравнения, описывающие квантованную по времени систему, примут вид

$$\left.\begin{aligned}x(t_{k+1}) &= \phi(t_{k+1},t_k)x(t_k)+\Gamma(t_{k+1},t_k)u(t_k)\\ y(t_{k+1}) &= Cx(t_{k+1})\end{aligned}\right\} \qquad (1.3)$$

где $\phi(t_{k+1},t_k) = e^{A(t_{k+1}-t_k)} = e^{AT_0}$,

$$\Gamma(t_{k+1},t_k) = \int_0^{t_{k+1}-t_k} e^{A\tau}Bd\tau = \int_0^{T_0} e^{A\tau}d\tau\cdot B$$

Уравнения системы (1.3) являются алгебраическими. Более подробно способы их получения и использования будут рассматриваться в дальнейшем. Здесь же отметим, что уравнения (1.3) не приближенные, а точные для моментов времени $t_k$, $t_{k+1}$,... .

*Квантование по уровню*

Квантование по уровню связано с представлением данных в ЭВМ с ограниченной разрядностью. В результате при преобразовании непрерывных данных в АЦП возникает погрешность их представления цифровым кодом, обусловленная необходимостью округления или усечения входных переменных до значения, определяемого весом младшего разряда преобразователя.

Оценку погрешности округления или усечения можно определить следующим образом.

Обозначим: ΔA - вес младшего разряда преобразователя; $\delta_x$ - среднеквадратическую погрешность квантования.

Тогда при округлении, в предположении о равновероятностном характере появления события (округления), описываемом постоянным значением плотности распределения $f(x)=\frac{1}{\Delta A}=$ const (рис. 1.4), дисперсию шума квантования можно определить следующим образом.

$$D(x)=\int_{-\infty}^{\infty} f(x)x^2dx = \int_{-\frac{1}{2}\Delta A}^{\frac{1}{2}\Delta A} \frac{1}{\Delta A} x^2 dx = \frac{1}{12}\Delta A^2$$

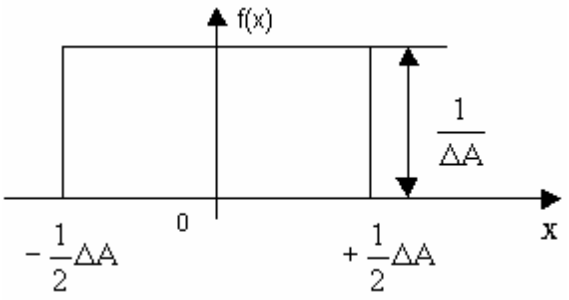



Рис. 1.4

В результате:

а) при округлении $\sigma_x = \sqrt{D(x)} = \frac{1}{2\sqrt{3}}\Delta A$

б) при усечении $\sigma_x = \frac{1}{\sqrt{3}}\Delta A$.

Очевидно, что можно решить и обратную задачу – определить разрядность АЦП при заданной среднеквадратичной погрешности квантования $\sigma_x$.

Количество квантов, необходимых для представления максимального значения $|x|_{max}$

$$N_{max} = \frac{|x|_{max}}{\Delta A} = \frac{|x|_{max}}{2\sqrt{3}\sigma_x}.$$

При использовании двоичной системы счисления $N_{max} = 2^{n_{АЦП}} - 1$.

В результате разрядность АЦП можно определить следующим образом:

$$n_{АЦП} = E_1\left\{\log_2\left(\frac{|x|_{max}}{2\sqrt{3}\sigma_x}+1\right)\right\}$$ - при округлении

или

$$n_{АЦП} = E_1\left\{\log_2\left(\frac{|x|_{max}}{\sqrt{3}\sigma_x}+1\right)\right\}$$ - при усечении,

где $E_1$ – функция округления числа до большего целого значения.

Влияние квантования при преобразовании цифровых значений в аналоговые с помощью ЦАП можно оценить следующим образом. Пусть при преобразовании используется фиксирующий элемент нулевого порядка. В этом случае управляющий (выходной) сигнал имеет ступенчатый характер.

Между тактами $T_0$ он не изменяется, принимая значения, имеющиеся в начале очередного такта (рис. 1.5а).

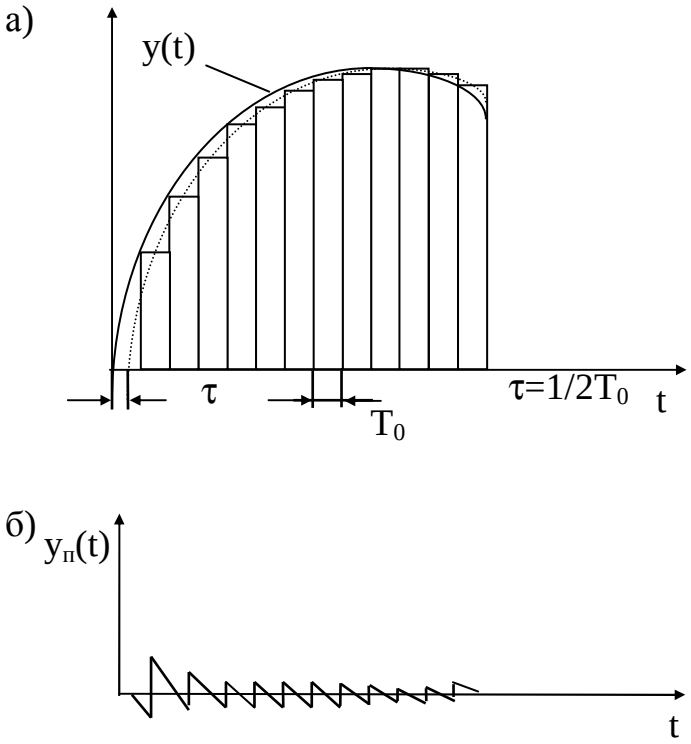



Рис. 1.5

Зависимость $y(кT_0)$ можно представить в виде двух процессов:

$y(кT_0)=y_0(t)+y_П(t)$.

Зависимость $y_0(t)$ получена путем интерполяции исходной ступенчатой кривой $y(кT_0)$ непрерывной кривой, проходящей через середины верхних сторон прямоугольников.

На рис.1.5б изображена зависимость $y_П(t)$ в форме пилообразной функции и представляющая помеху квантования сигнала цифро-аналоговым преобразователем. Как правило, при достаточно большом числе разрядов и инерционном объекте «пилообразный» шум не может оказать существенного влияния на процессы управления в системе.

В заключение отметим, что процессы квантования оказывают влияние и на точность вычисления управляющих воздействий процессором.

Последнее также связано с округлением результатов вычисления в процессоре из-за ограниченной разрядной сетки. Влияние этого эффекта подробно описано в [5,6].

# 2. СТРУКТУРНО-ФУНКЦИОНАЛЬНАЯ ОРГАНИЗАЦИЯ СИСТЕМ УПРАВЛЕНИЯ

Пусть в самом общем виде для любой системы управления выполнены два условия – определены цели управления и известны органы, способные принимать решения, обеспечивающие достижение этих целей (решающие органы – РО). Тогда, в зависимости от их структурно-функциональной организации можно выделить три типа систем управления:

одноуровневые одноцелевые системы;

одноуровневые многоцелевые системы;

многоуровневые многоцелевые системы.

Рассмотрим более подробно структурную организацию и особенности каждого из перечисленных выше типов систем управления.

### Одноуровневые одноцелевые системы

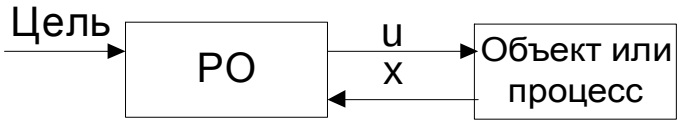



Рис. 2.1

В таких системах (рис. 2.1) цель управления выбирается единой; решающий орган (РО) строится так, чтобы принимаемые решения обеспечивали достижение этой цели. При этом решаемая управлением задача может быть как относительно простой, так и достаточно сложной.

К таким задачам могут относиться как задачи контроля, стабилизации, слежения, так и задачи оптимизации, адаптации и т.д.

Характерным для одноуровневых одноцелевых систем управления является одно замечательное свойство – бесконфликтность. Иногда говорят, что одноцелевые одноуровневые системы обладают свойством концептуальной простоты.

### Одноуровневые многоцелевые системы

Системы этого типа (рис. 2.2.) обладают следующими особенностями. Они жизнеспособны только в том случае, когда различные цели, поставленные перед различными решающими органами не противоречивы, т.е., если в системе невозможны конфликтные ситуации. Если конфликтные ситуации возможны, то они должны быть разрешены заранее (до начала функционирования системы). Как правило, исключение конфликтных ситуаций достигается введением извне в

систему некоторых дополнительных условий (компромиссов), позволяющих многоцелевую задачу свести к одноцелевой, в которой при корректных компромиссах удается перейти к бесконфликтным условиям решения задачи.

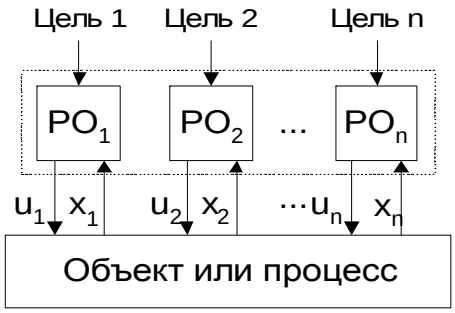



Рис. 2.2

**Многоуровневые многоцелевые системы**

В отличие от систем второго типа, в многоцелевых многоуровневых системах возможные конфликтные ситуации призван разрешить специальный орган вышестоящего уровня. Например, в двухуровневой системе, показанной на рис.2.3, таким органом будет $РО_2$. Как и решающие органы первого (нижнего) уровня, $РО_2$ действует в соответствии с заданной для него целью управления. Реализация этой цели должна обеспечить эффективность функционирования системы управления в целом. Поэтому ее принято называть глобальной целью, в отличие от целей подсистем нижнего уровня, называемых локальными целями.

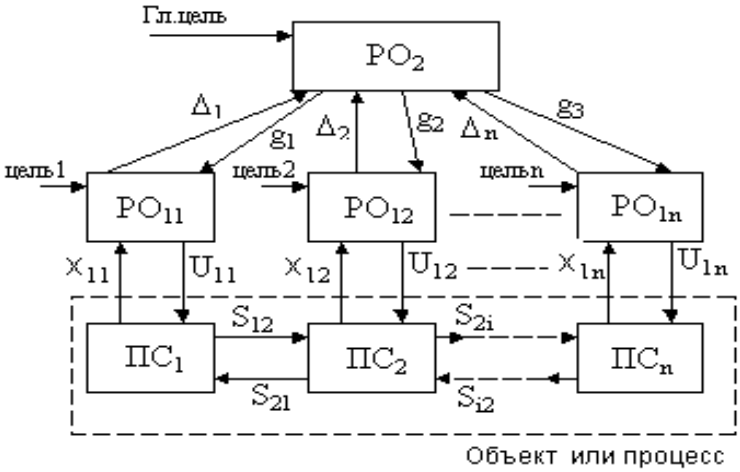



Рис. 2.3

Очевидно, что для решения общей задачи решающий орган $РО_2$ должен быть наделен свойствами доминирования над целями, стоящими перед решающими органами нижнего уровня. Такими свойствами не обладает ни один из решающих органов систем управления первого и второго типов. Поэтому

системы третьего типа принято выделять в самостоятельный класс, названный системами иерархического управления. (Термин "иерархия" происходит от греческого – "служебная лестница").

Наличие в иерархических системах управления вышестоящего уровня позволяет отказаться от априори заданных компромиссных решений и наделить решающие органы нижестоящих уровней свободой выбора локальных решений. При этом предполагается, что принятые на нижнем уровне решения не должны противоречить глобальной (общей) цели управления объектом или процессом. На $РО_2$ возлагается решение задачи коррекции принимаемых нижним уровнем решений.

Таким образом, из краткого рассмотрения принципов построения и особенностей работы современных систем управления, в концептуальном отношении наибольшую сложность представляют системы третьего типа – системы иерархического управления. Естественно, что и возможности таких систем решать сложные задачи управления существенно шире, чем систем первого и второго типов.

## 2.1. Одноуровневое одноцелевое управление

Как было отмечено выше, практическая реализация одноуровневого одноцелевого управления может быть различной. В рамках приведенной ранее классификации это наиболее распространенный тип управления.

Исторически одноуровневые одноцелевые системы управления раньше других были реализованы и хорошо разработаны теоретически, что объясняется концептуальной простотой их организации

Наибольшее распространение для управления техническими системами получили следующие виды одноцелевого одноуровневого управления:

- программно-логическое;
- стабилизационное, включая поддержание заданного значения и слежение за изменением задания;
- оптимальное, включая и различные виды адаптации.

Как правило, все эти виды управления реализуются в форме управления с обратными связями. В дальнейшем в пособии в качестве примеров будет рассмотрено ограниченное число методов и алгоритмов реализации каждого из перечисленных выше видов управления.

Следует подчеркнуть, что алгоритмизация является первым этапом разработки любой компьютерной системы управления. Поэтому большинство рассматриваемых далее примеров заканчивается формулировкой алгоритма.

### 2.1.1. Программно-логическое управление

Программным управлением принято называть управление, реализующее изменение координат состояния объекта или процесса в соответствии с заранее заданной программой. Одной из разновидностей программного управления является программно-логическое управление (ПЛ-управление). Требуемый алгоритм управления реализуется в зависимости от состояния и событий в объекте управления, которые задаются в форме ряда условий, определяющих последовательность выполнения операций управления.

Реализацию ПЛ-управления рассмотрим на примере управления лифтом в многоэтажном доме. Для упрощения примем, что в лифтом может быть выполнена только одна команда вызова или назначения. Команды, поступившие в момент выполнения предыдущей команды, игнорируются.

Синтез алгоритма системы ПЛ-управления лифтом производится в несколько этапов.

*1 этап*. Составляется таблица состояний, в которых может находиться объект и формализованное их описание в виде соответствующих логических условий.

*2 этап*. На основе таблицы состояний определяются возможные переходы из одного состояния в другое. Составляется направленный граф состояний и переходов, а также логические уравнения, описывающие условия реализации переходов.

*3 этап*. Определяется необходимое информационное обеспечение работы системы, а также требуемые по технологии временные задержки и блокировки.

*4 этап*. На основании полученных результатов составляется формализованное описание алгоритма реализации ПЛ-управления средствами вычислительной техники.

С учетом ограниченного объема пособия и того, что требования к информационному обеспечению системы управления в реальных условиях могут быть различными, третий этап синтеза далее не будет рассматриваться. Его предлагается выполнить читателям самостоятельно.

Пусть в системе ПЛ-управления лифтом предусмотрены следующие информационные сигналы (рис. 2.4):

$E_л$ – сигнал положения лифта относительно этажей,

$E_н$ – сигнал наружного вызова,

$E_в$ – сигнал внутреннего вызова (сигнал назначения),

$A1$ – сигнал включения основного двигателя лифта,

$\overline{A1}$ – двигатель отключен,

$A2$ – сигнал реверса (движение вверх - $A2$; движение вниз - $\overline{A2}$)

$A3$ – сигнал закрытия дверей (двери закрыть - $A3$; двери открыть - $\overline{A3}$)

$S$ – сигнал загрузки лифта (лифт загружен - $S$; лифт не загружен - $\overline{S}$),

$N_л$ – номер этажа, на котором остановился лифт,

$N_н$, $N_в$ – соответственно номера этажа наружного вызова и этажа назначения,

$\Delta = N_н - N_л > 0$ или $\Delta = N_в - N_л > 0$ – направление движения кабины вверх,

$\Delta = N_н - N_л < 0$ или $\Delta = N_в - N_л < 0$– направление движения кабины вниз,
$\Delta = 0$ –кабина лифта находится на этаже вызова – остановки.

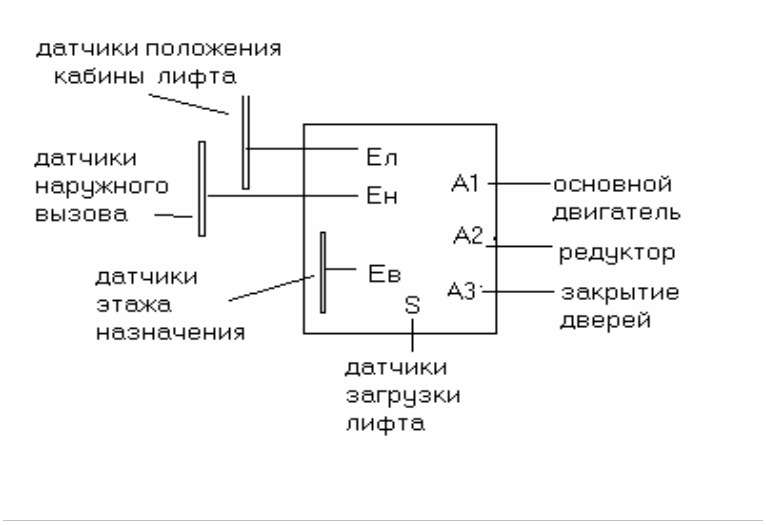



Рис. 2.4

Алгоритмизация задачи ПЛ-управления начинается с составления направленного графа состояний и переходов. Вершинами графа изображаются состояния, в которых может находиться лифт, а дугами – переходы из одного состояния в другое. Условия перехода записываются над дугами.

Граф состояний и переходов для рассматриваемой системы приведен на рис. 2.5, а в таблице 2.1 указаны вершины графа и соответствующие им состояния лифта.

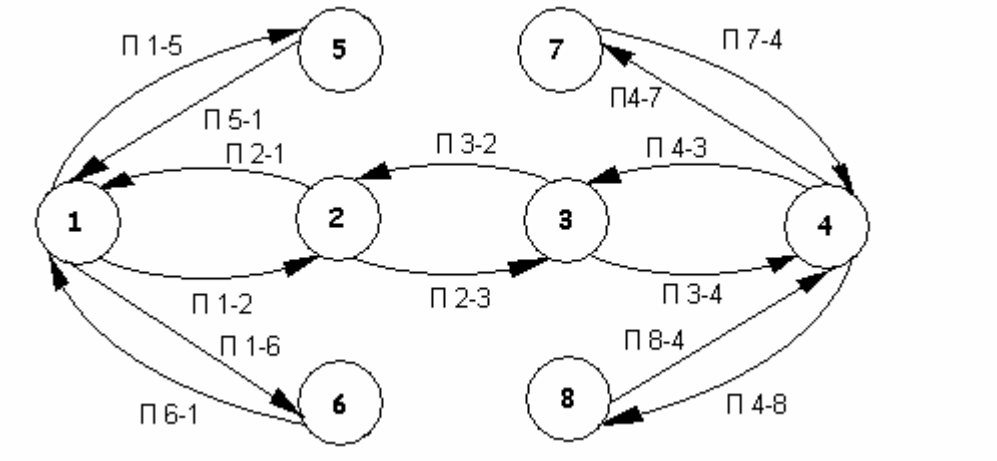



Рис. 2.5

Таблица 2.1

| № вершин | Состояния |
|---|---|
| 1 | Лифт стоит, не загружен, двери закрыты – С1 = ($\overline{A1}\cdot\overline{S}\cdot A3$) |
| 2 | Лифт стоит, не загружен, двери открыты – С2 = ($\overline{A1}\cdot\overline{S}\cdot\overline{A3}$) |
| 3 | Лифт стоит, загружен, двери открыты – С3 = ($\overline{A1}\cdot S\cdot\overline{A3}$) |
| 4 | Лифт стоит, загружен, двери закрыты – С4 = ($\overline{A1}\cdot S\cdot A3$) |
| 5 | Лифт движется вверх, не загружен, закрыт - С5 = ($A1\cdot A2\cdot\overline{S}\cdot A3$) |
| 6 | Лифт движется вниз, не загружен, закрыт - С6 = ($A1\cdot\overline{A2}\cdot\overline{S}\cdot A3$) |
| 7 | Лифт движется вверх, загружен, закрыт - С7 = ($A1\cdot A2\cdot S\cdot A3$) |
| 8 | Лифт движется вниз, загружен, закрыт - С8 = ($A1\cdot\overline{A2}\cdot S\cdot A3$) |

Условия перехода лифта из вершины 1 в вершину 2 можно записать в виде $П_{1\text{-}2} = Е_н\cdot\overline{A1}\cdot\overline{S}\cdot\bar{A}3$, что соответствует ситуации, когда поступает сигнал наружного вызова $Е_н$, лифт стоит на этаже вызова ($\Delta=0$), не загружен, необходимо лишь открыть двери.

Переход из вершины 2 в 3 реализуется следующим образом:

$П_{2\text{-}3} = Е_н\cdot\overline{A1}\cdot S\cdot\overline{A3}$, что соответствует загрузке лифта (пассажир вошел в кабину).

Переход из вершины 3 в 4 $П_{3\text{-}4} = \overline{A1}\cdot S\cdot Е_в\cdot A3$ означает, что вошедший в кабину пассажир нажал кнопку этажа назначения и двери лифта закрылись.

Переход $П_{4\text{-}7} = A3\cdot Е_в\cdot S\cdot A2\cdot A1$ – определено направление движения лифта вверх ($\Delta > 0$) и включен основной двигатель.

Переход $П_{4\text{-}8} = Е_в\cdot A3\cdot S\cdot A2\cdot A1$ означает выполнение таких же условий, что и при переходе $П_{4\text{-}7}$, но отличается от него движением вниз ($\Delta<0$).

Переход $П_{7\text{-}4} = П_{8\text{-}4} = Е_В\cdot\overline{A1}\cdot S\cdot A3$ – пассажир достиг этажа назначения ($\Delta=0$).

Переход $П_{4\text{-}3} = Е_в\cdot\overline{A1}\cdot S\cdot\overline{A3}$ – после достижения заданного этажа необходимо открыть двери.

Переход $П_{3\text{-}2} = \overline{A1}\cdot\overline{S}\cdot\overline{A3}\cdot\overline{S}$ – пассажир покинул лифт.

Переход $П_{2\text{-}1} = \overline{S}\cdot A3\cdot\overline{A1}$ – закрытие дверей лифта (пустого).

Переход $П_{1\text{-}5} = Е_н\cdot\overline{S}\cdot A3\cdot A2\cdot A1$ – получен вызов с этажа, расположенного выше кабины.

Переход $П_{1\text{-}6} = Е_н\cdot\overline{S}\cdot A3\cdot\overline{A2}\cdot A1$ – получен вызов с этажа, расположен-ного ниже кабины.

Переход $П_{5\text{-}1} = П_{6\text{-}1} = Е_н\cdot\overline{S}\cdot\overline{A1}\cdot A3$ – пустой лифт достиг этажа, с которого был получен вызов ($\Delta=0$).

Ниже приведен пример алгоритма ПЛ-управления. Как отмечалось ранее, характерной особенностью программно-логического управления является реализация старт-стопного режима управления исполнительными органами с контролем текущего состояния объекта в конкретных точках. В нашем примере этими точками являются вершины графа состояния. В алгоритме, для остановки его работы оператором, дополнительно введен сигнал z (z = 1 –стоп, z = 0 – разрешение работы).

Дальнейшее уточнение алгоритма предполагает введение необходимых блокировок (например, таких, как контроль перегрузки кабины лифта, задержки закрытия дверей и др.), а также учет требуемых информационных сообщений, индикации состояний и т.п. Эти операции выполняются на последующих этапах алгоритмизации. Их предлагается проделать самостоятельно.

*Алгоритм ПЛ-управления лифтом*

*Шаг 1*. Исходное положение лифта - $C1=\overline{A1}\cdot\overline{S}\cdot A3=1,\quad Z=0$.

*Шаг 2*. Если $E_н$=1, то переход к шагу 3, иначе - ждать.

*Шаг 3*. Определить значения $\Delta = N_н - N_л$= ( > 0, < 0, = 0)?

*Шаг 4*. Если Δ > 0,то переход к шагу 5, иначе - к шагу 7.

*Шаг 5*. Переход $П_{1–5}=E_н\,A1\cdot A2\cdot\overline{S}\cdot A3$ в состояние С5; если С5 = 1, то переход к шагу 6, иначе – ждать.

*Шаг 6*. Переход $П_{5–1}=E_н\cdot\overline{A1}\cdot\overline{S}\cdot A3$ в состояние С1; если С1 = 1, то переход к шагу 11, иначе - ждать.

*Шаг 7*. Если Δ < 0, то переход к шагу 8; иначе - к шагу 10.

*Шаг 8*. Переход $П_{1–6}=E_н\cdot A1\cdot\overline{A2}\cdot A3\cdot\overline{S}$ в состояние С6, если С6 = 1, то переход к шагу 9, иначе – ждать.

*Шаг 9*. Переход $П_{6–1}=E_н\cdot A1\cdot\overline{A2}\cdot A3\cdot\overline{S}$ в состояние С1; если С1 = 1, то переход к шагу 11, иначе - ждать.

*Шаг 10*. Если Δ = 0 при $E_н$ = 1, то переход к шагу 11.

*Шаг 11*. Переход $П_{1–2}=E_н\cdot\overline{A1}\cdot\overline{S}\cdot\overline{A3}$ в состояние С2; если С2 = 1, то переход к шагу 12, иначе – ждать.

*Шаг 12*. Загрузка кабины лифта – S = 1 и переход к шагу 13.

*Шаг 13* Переход $П_{2–3}=E_н\cdot S\cdot\overline{A1}\cdot\overline{A3}$ в состояние С3; если С3=1, то переход к шагу 14, иначе – ждать.

*Шаг 14*. Если $E_в$ = 1,то переход к шагу 15, иначе – ждать.

*Шаг 15*. Переход $П_{3–4}=E_в\cdot\overline{A1}\cdot S\cdot A3$ в состояние С4; если С4 = 1, то переход к шагу 16, иначе – ждать.

*Шаг 16*. Определить значение $\Delta = N_в – N_л$ = (> 0, < 0, = 0)? и переход к шагу 17.

*Шаг 17*. Если Δ > 0, то А2 = 0 и переход к шагу 18; иначе – к шагу 20.

*Шаг 18*. Переход $П_{4–7}=E_в\cdot A1\cdot A2\cdot A3\cdot S$в состояние С7 если С7 = 1, то переход к шагу 19, иначе – ждать.

*Шаг 19.* Переход $П_{7-4} = E_в \cdot \overline{A1} \cdot S \cdot A3$ в состояние С4; если С4 = 1, то переход к шагу 20, иначе - ждать.

*Шаг 20.* Если $\Delta < 0$ и $\overline{A2} = 1$, то переход к шагу 21; иначе - к шагу 23.

*Шаг 21.* Переход $П_{4-8} = E_в \cdot \overline{A2} \cdot A1 \cdot S \cdot A3$ в состояние С4; если С4 = 1, то переход к шагу 22, иначе - ждать.

*Шаг 22.* Переход $П_{8-4} = E_в \cdot \overline{A1} \cdot A3 \cdot S$ в состояние С4; если С4 = 1, то переход к шагу 23, иначе – ждать.

*Шаг 23.* Если $\Delta = 0$ , то С4 = 1 и переход к шагу 24.

*Шаг 24.* Переход $П_{4-3} = E_в \cdot S \cdot \overline{A1} \cdot \overline{A3}$ в состояние С3; если С3 = 1, то переход к шагу 25, иначе – ждать.

*Шаг 25.* Освобождение кабины лифта - $\overline{S}$ и переход к шагу 26.

*Шаг 26.* Если $S \neq 0$. то переход к шагу 14; иначе – к шагу 27.

*Шаг 27.* Переход $П_{3-2} = \overline{A1} \cdot \overline{S} \cdot \overline{A3}$ в состояние С2; если С2 = 1, то переход к шагу28, иначе – ждать.

*Шаг 28.* Переход $П_{2-1} = \overline{A1} \cdot \overline{S} \cdot \overline{A3}$ в состояние С1; если С1 = 1, то переход к шагу 29, иначе – ждать.

*Шаг 29.* Если Z= 0, то переход к шагу 2; иначе – к шагу 30.

*Шаг 30.* Конец.

*Примечания.*

*1. Выражение “* ***иначе – ждать****” означает возвращение к началу шага и используется для учета динамики процессов перехода от одного состояния к другому.*

*2. Знак операции конъюнкции* (∧) *в логических условиях, для упрощения их записи, заменен на знак* ( · **).**

### 2.1.2. Управление, реализуемое промышленными регуляторами

Промышленными регуляторами называют управляющие устройства как жесткой, так и нежесткой (программной) реализации, используемые в системах автоматического управления промышленными объектами различного назначения и сложности.

К таким системам помимо реализации технических характеристик предъявляются повышенные требования к удобству эксплуатации и способности сравнительно простой перенастройки управления при изменении параметров объекта или при полной его замене.

В большинстве промышленных САУ обеспечение требуемых показателей качества процесса управления достигается использованием обратной связи в форме пропорционально-интегрально-дифференциального (ПИД) закона или комбинацией составляющих этого закона. Основным достоинством таких регуляторов является малое количество настраиваемых параметров. Это коэффициенты передачи (усиления) по пропорциональной ($k_п$), интегральной ($k_и$) и дифференциальной ($k_д$) составляющим закона управления.

Уравнение ПИД - регулятора (в непрерывной форме) можно записать в виде:

$$u(t) = k_п e(t) + k_и \int_0^t e(\tau)d\tau + k_д \frac{de(t)}{dt} \qquad (2.1)$$

где $e(t) = y^з(t) - y(t)$ – рассогласование управляемой координаты y(t) и уставки $y^з$ (ошибка управления).

В дискретной форме при использовании интегрирования по методу прямоугольников и замене дифференцирования операцией взятия конечной разности уравнение (2.1) можно записать в виде:

$$u(kT_0) = u(k) = k_п e(k) + k_и T_0 \sum_{i=0}^{k-1} e(i) + \frac{k_д}{T_0}\big(e(k) - e(k-1)\big), \qquad (2.2)$$

где $k_и = \frac{1}{T_и}$; $k_д = \frac{1}{T_д}$

$T_0$ – "период" квантования;

$kT_0$ – дискретное время (при $T_0 = \text{const}$ $kT_0 \equiv k$);

$T_и$, $T_д$ – постоянные времени соответственно интегрирования и дифференцирования.

В рекуррентной форме ПИД-закон управления (2.2) имеет вид [6]:

$$u(k) = u(k-1) + q_1 e(k) + q_2 e(k-1) + q_3 e(k-2), \qquad (2.3)$$

где $q_1 = k_п + k_д/T_0$; $q_2 = k_и T_0 - k_п - 2k_д/T_0$; $q_3 = k_д/T_0$.

Начальное значение $u(0)$ в этом случае можно получить, положив $u(0) = q_1 e(0) = (k_п + k_д/T_0)e(0)$.

Рекуррентная форма для программных вычислений наиболее удобна, поэтому в дальнейшем она будет считаться основной.

В выражениях (2.2) и (2.3) присутствует параметр $T_0$ (дискретное время или "период" квантования). Он существенно влияет не только на качество процессов управления, но во многих случаях обуславливает устойчивость системы в целом.

Выбор периода (частоты) квантования сигналов, являющегося одним из параметров настройки системы управления, зависит от многих факторов. При его определении необходимо учитывать процесс прохождения сигнала через динамическую систему, каковой и является объект управления. Хотя основой для выбора $T_0$ может служить теорема Котельникова-Шеннона, в ряде случаев рекомендуются и другие правила [1]. Сводка этих правил приведена в таблице 2.2.

### Выбор настроек коэффициентов $к_п$, $к_и$, $к_д$ из условий обеспечения устойчивости дискретных систем с линейными стационарными объектами

Пусть линейный многомерный стационарный объект управления описывается векторно-матричным уравнением

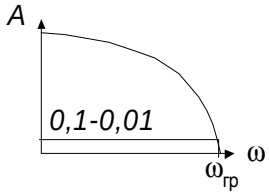
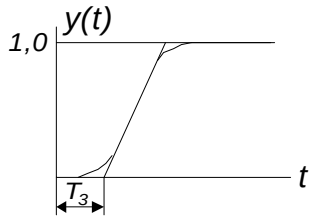
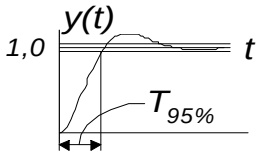
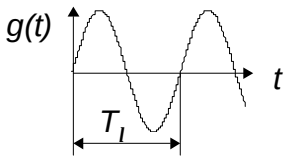
$$\dot{x}(t) = Ax(t) + Bu(t) \qquad (2.4)$$

и уравнением выхода

$y(t) = Cx(t)$,

где $x \in R^n$ - вектор состояния объекта; $y \in R^p$ - вектор выхода; $u \in R^m$ - вектор управления; $A \in R^{n \times n}$ - матрица параметров объекта; $B \in R^{n \times m}$ - матрица управления; $C \in R^{p \times n}$ - матрица выхода.

Таблица 2.2

| Класс или особенности процесса | Значение $T_0$ | Примечания |
|---|---|---|
| 1. Любой в низкочастотной области | $T_0 = \frac{\pi}{\omega_{гр}} = \frac{1}{2f_{гр}}$ $\omega_{гр}, f_{гр}$ - граничные частоты |   |
| 2. Апериодический с явно выраженным запаздыванием $T_з$ | $T_0 = \left(\frac{1}{4} \div \frac{1}{8}\right) T_з$ |   |
| 3. Возможны затухающие низкочастотные колебания | $T_0 = \left(\frac{1}{6} \div \frac{1}{15}\right) T_{95\%}$ |   |
| 4. Гармонический | $\frac{T_1}{12} \le T_0 \le \frac{T_1}{6}$ |   |

Пусть $y(t) = x(t)$, $y^з = 0$, а управление $u(t)$ формируется по ПИД-закону. Для анализа устойчивости необходимо сформировать дискретную модель объекта. Пусть на $T_0$ не накладывается условие малости. В этом случае дискретную модель объекта можно получить следующим образом.

Из теории управления известно, что вектор состояния объекта

$$x(t) = \Phi(t)x_0 + \int_0^t \Phi(t-\tau)Bu(\tau)d\tau, \qquad (2.5)$$

где $\Phi(t)$ – переходная матрица состояния системы (фундаментальная матрица).

Для линейных стационарных объектов

$$\Phi(t) = e^{At} = \exp At = \sum_{\nu=0}^{\infty} \frac{(At)^\nu}{\nu!} = \left(E + \frac{At}{1!} + \frac{A^2t^2}{2!} + \cdots\right);$$

В дискретных системах часто используется фиксатор нулевого порядка, при котором для $kT_0 \le t \le (k+1)T_0$ управление $u(t)$ постоянно, т.е. $u(t) = u(kT_0) = \text{const}$.

Если принять $x(kT_0)$ за начальное состояние объекта, то для $t > kT_0$ из (2.5) следует $x(t) = \Phi(t - kT_0)x(kT_0) + \int_{kT_0}^{t} \Phi(t-\tau)B d\tau \cdot u(kT_0)$.

Для момента времени $t = (k+1)T_0$

$$x\big((k+1)T_0\big) = \Phi\big((k+1)T_0 - kT_0\big)x(kT_0) + \int_{kT}^{(k+1)T_0} \Phi\big((k+1)T_0 - \tau\big)d\tau \cdot Bu(kT_0) =$$

$$= \Phi(T_0)x(kT_0) + \int_{kT_0}^{(k+1)T_0} \Phi\big((k+1)T_0 - \tau\big)d\tau \cdot Bu(kT_0) = e^{AT_0}x(k) + \int_{t=k}^{t=k+1} e^{A\left((k+1)T_0 - \tau\right)}d\tau \cdot Bu(k)$$ О

бозначим $e^{AT_0} = E + AT_0 + \frac{A^2T_0^2}{2!} + \cdots = H;$

$$\int_{t=k}^{t=k+1} e^{A\left((k+1)T_0 - \tau\right)}d\tau = \int_{t_k}^{t_{k+1}} e^{At_{k+1}} \cdot e^{-A\tau}d\tau = e^{At_{k+1}} \cdot (-A)^{-1}e^{-A\tau}\Big|_{t_k = kT_0}^{t_{k+1} = (k+1)T_0} = A^{-1} + A^{-1}e^{AT_0} =$$

$$= ET_0 + \frac{AT_0^2}{2!} + \frac{A^2T_0^3}{3!} + \cdots = F.$$

Тогда, приняв $kT_0 \equiv k, \quad kT_0 + T_0 \equiv k+1,$ получим

$$x(k+1) = Hx(k) + FBu(k); \quad x(k=0) = x_0. \qquad (2.6)$$

Уравнение (2.6) является алгебраическим уравнением и представляет модель объекта в дискретной форме.

При использовании ПИД-закона управления для k-го момента $u(k)$ можно представить следующим образом:

$$u(k) = -\left[k_п x(k) + k_{дд}\big(x(k+1) - x(k)\big) + k_{ид}\sum_{i=0}^{k-1} x(i)\right], \qquad (2.7)$$

где $x(k+1) - x(k)$ -правая разность, $k_{дд} = \frac{k_д}{T_0}$; $k_{ид} = k_и \cdot T_0$

Обозначим $u^u(k) = \sum_{i=0}^{k-1} x(i)$, тогда

$$u(k) = -\left[k_п x(k) + k_{дд}\big(x(k+1) - x(k)\big) + k_{ид}u^u(k)\right] \qquad (2.8)$$

Подставим (2.8) в (2.6) и после очевидных преобразований получим

$$\left.\begin{array}{c} x(k+1) = (E + FBk_{дд})^{-1}\left[H - FB(k_п - k_{дд}\right]x(k) - (E + FBk_{дд})^{-1}FBk_{ид}u^u(k) \\ u(k+1) = x(k) + u^u(k) \end{array}\right\} \qquad (2.9)$$

Пусть

$$\tilde{H} = (E + FBk_{дд})^{-1}\left[H - FB(k_п - k_{дд})\right],$$

$\tilde{B} = -(E + FBk_{дд})^{-1} FBk_{ид}$ .

Тогда систему (2.9) можно переписать в виде

$$\left.\begin{aligned} x(k+1) &= \tilde{H}x(k) + \tilde{B}u^{u}(k) \\ u^{u}(k+1) &= x(k) + u^{u}(k) \end{aligned}\right\} \qquad (2.10)$$

Обозначим

$$\hat{x}(k+1) = \begin{pmatrix} x(k+1) \\ ---- \\ u^{u}(k+1) \end{pmatrix}; \quad \hat{x}(k) = \begin{pmatrix} x(k) \\ --- \\ u^{u}(k) \end{pmatrix}; \quad \hat{H} = \begin{pmatrix} \tilde{H} & | & \tilde{B} \\ - & - & - \\ E & | & E \end{pmatrix}$$

Тогда (2.10) примет вид

$$\hat{x}(k+1) = \hat{H}\hat{x}(k) \; ; \quad \hat{x}(0) = \hat{x}_{0} . \qquad (2.11)$$

Уравнение (2.11) является линейным однородным алгебраическим уравнением и описывает собственное движение системы с ПИД-регулятором.

Как известно, для устойчивости такой системы необходимо, чтобы

$$|\lambda_{i}| < 1 \; ; \quad i = \overline{1,n} , \qquad (2.12)$$

где $\lambda_{i}$ - собственные числа блочной матрицы $\hat{H}$ , определяемые из условия

$$\det(\hat{H} - \lambda \cdot E) = 0 \qquad (2.13)$$

или

$$\begin{vmatrix} \tilde{H} - \lambda \cdot E & | & \tilde{B} \\ -- & - & -- \\ E & | & E - \lambda \cdot E \end{vmatrix} = 0 , \qquad (2.14)$$

где E -единичные матрицы соответствующих размерностей.

Подставив в (2.14) значения $\tilde{H}$ и $\tilde{B}$ и раскрыв определитель, получим

$$\left\{(E + FBk_{дд})^{-1}\left[H - FB(k_{п} - k_{дд})\right] - \lambda \cdot E\right\}(E - \lambda \cdot E) + (E + FBk_{дд})^{-1} FBk_{ид} = 0 \qquad (2.15)$$

Коэффициенты $k_{п}$, $k_{ид}$ и $k_{дд}$ при использовании векторно-матричной модели объекта являются векторами соответствующей размерности. При заданном $T_0$ они должны для обеспечения устойчивости системы выбираться

из условия (2.12). Для определения устойчивости можно воспользоваться билинейным преобразованием $\lambda = \frac{w+1}{w-1}$ и применить известные критерии устойчивости, например, критерий Гурвица.

Выбор коэффициентов настроек ПИД-регуляторов из условий обеспечения требуемого качества процессов, как правило, производится средствами моделирования с последующим обобщением результатов в форме рекомендаций по выбору настроек для определенных типовых объектов и желаемого вида переходных процессов [1].

**Модификация ПИД-законов управления**

Целью модификации является уменьшение возможных динамических воздействий со стороны регулятора на объект при резких изменениях задающих воздействий и сохранении основных технических характеристик ПИД-регулирования.

В зависимости от динамических свойств объекта модификацию можно выполнить различными способами.

***а) Исключение дифференцирования по ошибке (рассогласованию) с сохранением дифференцирования по выходной координате.***

На рис. 2.6 приведен фрагмент реализации этого способа модификации.

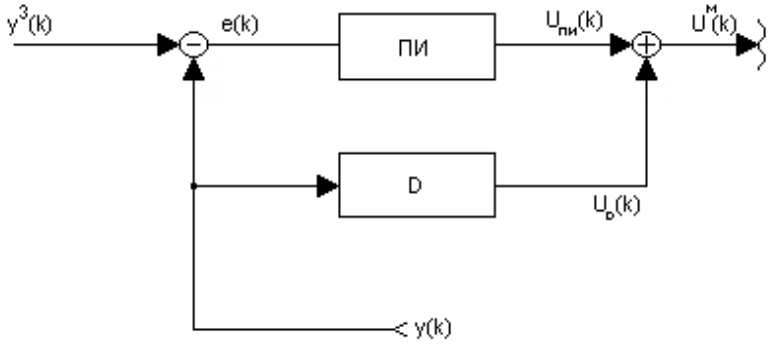



Рис. 2.6

Здесь $y^{з}(k)$ - изменение задающей координаты в $k$-ый момент времени, – дифференцирующая составляющая ПИД-закона по выходной координате,

$e(k) = y^{з}(k) - y(k)$ - рассогласование задающей и выходной координаты (ошибка);

Рекуррентное выражение для модифицированного управления $U^{м}(k)$ можно получить следующим образом:

$$u^{м}(k) = k_{п}e(k) + k_{и}T_0\sum_{i=0}^{k-1}e(i) - \frac{k_{д}}{T_0}\big(y(k) - y(k-1)\big)$$

$$u^{м}(k-1) = k_{п}e(k-1) + k_{и}T_0\sum_{i=0}^{k-2}e(i) - \frac{k_{д}}{T_0}\big(y(k-1) - y(k-2)\big)$$

$$u^{м}(k) - u^{м}(k-1) = k_{п}\big(e(k) - e(k-1)\big) + k_{и}T_0e(k-1) - \frac{k_{д}}{T_0}\big(y(k) - 2y(k-1) + y(k-2)\big)$$

$$u^{м}(k) = u^{м}(k-1) + k_{п}\big(e(k) - e(k-1)\big) + k_{и}T_0e(k-1) - \frac{k_{д}}{T_0}\big(y(k) - 2y(k-1) + y(k-2)\big)$$

Если обозначить $q_1^{м} = k_п$, $q_2^{м} = k_и T_0 - k_п$, $q_3^{м} = \frac{k_д}{T_0}$,

$D_y = y(k) - 2y(k-1) + y(k-2)$,

то $u^{м}(k) = u^{м}(k-1) + q_1^{м} e(k) + q_2^{м} e(k-1) - q_3^{м} D_y$,

причем $u^{м}(0) = q_1^{м} e(0) - q_3^{м} y(0)$.

***б) Исключение из ПИД-закона дифференциальной и пропорциональной составляющих.***

Реализация этого способа модификации приведена на рис. 2.7. Он применяется с той же целью, что и первый способ, но для еще менее инерционных объектов.

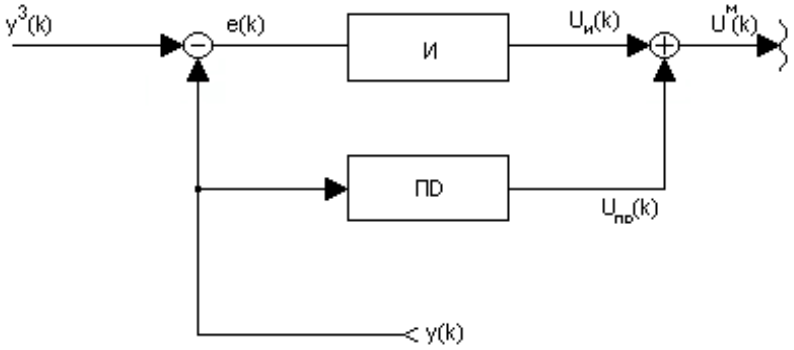



Рис. 2.7

Обозначим $u_и(k) = k_и T_0 \sum_{i=o}^{k-1} e(i)$ ; $u_{пд}(k) = -\left( k_п y(k) + \frac{k_д}{T_0}(y(k) - y(k-1)) \right)$,

тогда выражение для $u^{м}(k)$ можно записать следующим образом:

$$u^{м}(k) = u_и(k) + u_{пд}(k) = k_и T_0 \sum_{i=0}^{k-1} e(i) - \left( k_п y(k) + \frac{k_д}{T_0}(y(k) - y(k-1)) \right)$$

$$u^{м}(k-1) = u_и(k-1) + u_{пд}(k-1) = k_и T_0 \sum_{i=0}^{k-2} e(i) - \left( k_п y(k-1) + \frac{k_д}{T_0}(y(k-1) - y(k-2)) \right)$$

$$u^{м}(k) - u^{м}(k-1) = k_и T_0 e(k-1) - \left( k_п + \frac{k_д}{T_0} \right) y(k) + \left( k_п + 2\frac{k_д}{T_0} \right) y(k-1) - \frac{k_д}{T_0} y(k-2)$$

Обозначим $k_и^{д} = k_и T_0$, $q_1^{м} = k_п + \frac{k_д}{T_0}$, $q_2^{м} = k_п + 2\frac{k_д}{T_0}$; $q_3^{м} = \frac{k_д}{T_0}$,

тогда $u^{м}(k) = u^{м}(k-1) + k_{ид} e(k-1) - q_1^{м} y(k) + q_2^{м} y(k-1) - q_3^{м} y(k-2)$,

причем $u^{м}(0) = q_1^{м} y(0)$.

Кроме описанных способов модификации ПИД законов возможны и другие, например, вместо способа (а) – включение реального (инерционного) дифференцирующего звена.

### 2.1.3 Оптимальное управление динамическими системами

При синтезе оптимального управления необходимо решать следующие задачи:

– сформулировать критерий качества (оптимальности) работы системы;
– установить и формализовать ограничения на диапазон возможных (допустимых) изменений координат состояния и управления, доставляющего экстремум критерию оптимальности;
– разработать алгоритм, доставляющий экстремум принятому критерию качества.

Как известно, формирование критериев оптимальности является результатом анализа требований, предъявляемых к конкретной системе управления. Наибольшее распространение в системах оптимального управления получили критерии в виде функционалов квадратичной или линейной форм. В непрерывном времени это критерии вида:

1) $J_1 = \min_t \int_{t_0}^{t_f} dt = \min(t_f - t_0)$ – используется при решении задачи максимального быстродействия;

2) $J_2 = \min_u \int_{t_0}^{t_f} C|u(t)|dt$ – используется при решении задачи минимизации «расхода топлива», как задачи математического программирования;

3) $J_3 = \min_u \left\{ \int_{t_0}^{t_f} (x^T(t)Qx(t) + u^T(t)Ru(t))dt + x_f^T(t_f)Gx(t_f) \right\}$ – используется при решении задач, обеспечивающих минимизацию взвешенных расходов энергии на управление, отклонение траектории от оптимальной и поддержание объекта в конечном состоянии $x_f(t_f)$.

Задачи оптимального быстродействия достаточно подробно описаны в литературе. Поэтому ниже будут рассматриваться вопросы алгоритмизации задач второго и третьего типов.

### *а) Алгоритмизация задачи математического программирования*

Для дискретных систем управления критерий $J_2$ имеет вид:

$$J_2 = \min\left\{\sum_{i=0}^{n-1}\sum_{j=1}^{m} C_j \,|\, u_j(i)\,|\right\} \qquad (2.16)$$

где $i = \dfrac{t}{T_0}$, t- текущее время "наблюдения", $T_0$ - период квантования,

i - порядковый номер точки "наблюдения", m - максимальное количество точек "наблюдения", j - номер переменной в критерии; $C_j$ - весовые коэффициенты.

Так как обычно $T_0 = \text{const}$, то в выражение для критерия качества его не вводят, поскольку постоянный коэффициент изменяет только масштаб критериальной оценки.

Задача минимизации $J_2$ должна решаться при учете ограничений следующего вида

$$\left.\begin{array}{l} a)\ A_u u(i) \le b_u \\ b)\ A_x X(i) \le b_x \end{array}\right\},\quad i = \overline{1, n-1}$$
$$c)\ SX(n) \le b_s, \qquad (2.17)$$

где $A_u \in R^{m\times m},\ A_x \in R^{r\times r},\ S \in R^{r\times r}$ – постоянные матрицы,

$u \in R^m$ - вектор управления, $X \in R^r$ - вектор координат состояния,

$b_u \in R^m,\ b_x \in R^r,\ b_s \in R^r$ - постоянные векторы.

Неравенство a) описывает ограничение на управление; неравенство b) – ограничение на траекторию движения объекта; неравенство c) – ограничение, учитывающее расходы на поддержание объекта в конечной точке траектории.

Задача (2.16) с ограничениями (2.17) относится к типу задач нелинейного программирования, т.к. использует подынтегральную функцию модульной формы. Методом разности переменных [5,14] эта задача может быть сведена к задаче линейного программирования. Для этого необходимо модифицировать критерий $J_2$ и ограничения (2.17).

Как известно, в задачах линейного программирования переменные в критерии должны быть знакопостоянными, т. е. подчиняться условию $u_j \ge 0$, для $\forall j$. Метод разности переменных предусматривает замену переменных в критерии и в ограничениях следующего вида:

1) в критерии: $u_j(i) = u'_j(i) + u''_j(i)$ ;

2) в ограничении: $u_j(i) = u'_j(i) - u''_j(i)$,

причем $u'_j \ge 0,\ u''_j \ge 0,\quad i = \overline{0, n-1},\ j = \overline{1, m}$.

В результате задачу (2.16), (2.17) можно переформулировать следующим образом.

$$\text{Найти } u = \arg\min_{u}\left\{ J_2 = \sum_{i=0}^{n-1}\sum_{j=1}^{m} C_j[u'_j(i) + u''_j(i)] \right\} \qquad (2.18)$$

при ограничениях:

$$\left.\begin{aligned} &A_u[u'_j(i) - u''_j(i)] \le b_u, \\ &A_x x(i) \le b_x, \\ &Sx(n) \le b_s, \\ &u'_j(i) \ge 0;\ u''_j \ge 0. \end{aligned}\right\} \qquad (2.19)$$

Задача (2.18) с ограничениями (2.19) принадлежит к задачам линейного программирования и может быть решена симплекс-методом [7].

При большой размерности решение задачи (2.18) с ограничениями в реальном времени может потребовать высокого быстродействия вычислительных средств. В этом случае задачу минимизации "расхода топлива" можно сформулировать в упрощенной локально-оптимальной форме. При этом оптимизация критерия производится на каждом отдельном шаге (временном отрезке). Подробнее методы локально-оптимального управления для задач минимизации "расхода топлива" описаны в [6].

### *б) Алгоритмизация задач оптимального терминального управления в непрерывном времени*

Терминальным управлением называют управление с фиксированным конечным состоянием системы. Рассмотрим задачу синтеза оптимального терминального управления на примере линейного стационарного объекта с использованием интегрального квадратичного критерия $J_3$.

### Оптимизация управления в непрерывном времени

Пусть имеется многомерный линейный стационарный объект, описываемый векторно-матричным уравнением вида

$$\dot{x}(t) = Ax(t) + Bu(t), x(t_0) = x_0, \qquad (2.20a)$$

где $x \in R^n$ - вектор координат состояния объекта,

$u \in R^m$ - вектор управления,

$x_0$ - вектор начальных условий,

$A \in R^{n x n}$ - матрица параметров объекта,

$B \in R^{n \times m}$ - матрица параметров цепи управления объекта.

Необходимо синтезировать управление, обеспечивающее перевод объекта из некоторого начального состояния $x(t_0) \ne 0$ в заданное конечное состояние $x(t_f) = 0$ и минимизацию целевого функционала вида:

$$J(u(t), x(t)) = \min_u \int_{t_0}^{t_f} [x^T(t)Qx(t) + u^T(t)R_u u(t)]dt, \qquad (2.20б)$$

где Q и $R_u$ - симметрические неотрицательно определенные диагональные матрицы.

Функционал (2.20б) используется при синтезе оптимальных систем различного назначения. Главные причины его применения:

- хорошо разработанный математический аппарат синтеза,
- целевой функционал имеет достаточно ясный физический смысл.

Например, при Q=E и $R_u=E_1$, где E и $E_1$ единичные матрицы, слагаемые $\int_0^{t_f} x_i^2 dt$ характеризуют энергию i-ой компоненты вектора состояния, а слагаемые $\int_0^{t_f} u_k^2 dt$ - энергию k-й компоненты вектора управления.

Если Q≠E и $R_u \neq E_1$, то соответствующие компоненты $x_i$ и $u_k$ входят в критерий с различными весами.

Таким образом, минимизация критерия J(u(t)) характеризует управление, обеспечивающее минимизацию энергетических затрат на управление и минимизацию отклонения координат состояния от нулевого значения.

Очевидно, что меняя веса координат состояния и управления можно изменять значимость расходов на управление и величину и продолжительность отклонения координат состояния от заданного значения.

В теории управления описанная задача известна, как линейно-квадратичная (ЛК) проблема оптимального управления. Результатом ее решения является определение параметров обратной связи (регулятора), обеспечивающей наилучший в смысле минимального значения функционала (2.20б) переходный процесс, при любых начальных условиях $x(t_0)$.

Как известно, решение ЛК - проблемы оптимального управления для стационарных систем сводится к решению алгебраического векторно-матричного квадратного уравнения Риккати вида

$$A^T S + SA - SBR_u^{-1}B^T S + Q = 0, \qquad (2.21)$$

где S - квадратная положительно определенная симметрическая матрица.

В результате его решения находится оптимальная матрица $S^*$ и матрица оптимальных коэффициентов обратной связи

$$K^* = R_u^{-1}B^T S^* \qquad (2.22)$$

При этом оптимальное управление формируется в виде пропорциональной обратной связи по всем координатам состояния

$$u^*(t) = -K^* x(t). \qquad (2.23)$$

Примечание. При нестационарных объектах управления для определения S необходимо решать векторно-матричное нелинейное ***дифференциальное уравнение*** Риккати. В этом случае S*(t) является функцией времени, а оптимальные коэффициенты обратной связи зависят не только от параметров B и $R_u$, но и от времени, т.е. K*(t).

Первым этапом решения ЛК - проблемы оптимального управления линейными стационарными объектами является решение алгебраического векторно-матричного уравнения Риккати. Известно достаточно большое количество возможных вариантов его решения. Все машинные варианты базируются на различного рода итерационных алгоритмах. Ниже рассматривается один из возможных алгоритмов этого типа.

Итак, требуется решить итерационным способом векторно-матричную систему уравнений вида

$$\left.\begin{aligned} A^T S + SA - SBR_u^{-1}B^T S + Q = 0 \\ K = R_u^{-1}B^T S \end{aligned}\right\} \qquad (2.24)$$

Итерационная процедура решения системы (2.24) строится на использовании идеи преобразования нелинейного уравнения Риккати к линейному. Для этого второе уравнение системы (2.24) домножим слева на $R_u$ и транспонируем:

$$\left.\begin{aligned} R_u K = B^T S \\ K^T R_u = SB \end{aligned}\right\} \qquad (2.25)$$

С учетом (2.25) первое уравнение (2.24) можно переписать в виде

$A^T S + SA - K^T R_u R_u^{-1} R_u K + Q = 0$ или

$$A^T S + SA + K^T R_u K + Q = 0. \qquad (2.26)$$

При известных значениях К и $К^T$ матричное уравнение (2.26) является линейным. Для его решения требуется выбрать начальные значения коэффициентов обратной связи так, чтобы обеспечивалась устойчивость замкнутой системы.

Рассмотрим более подробно методику и этапы решения итерационным способом уравнения (2.26) в предположении, что К и $К^T$ известны, например, определены в результате выполнения предыдущего этапа.

Учитывая (2.23), уравнение (2.20а) перепишем следующим образом

$$\dot{x}(t) = Ax(t) + Bu(t) = (A - BK)x(t) = A_z x(t), \quad x(t_0) = x_0, \qquad (2.27)$$

где $A_z = A - BK$ - матрица параметров замкнутой системы.

Откуда $A = A_z + BK$ (2.28)

Подставим (2.28) в (2.26):

$$(A_z + BK)^T S + S(A_z + BK) - K^T R_u K + Q = 0$$

или $A_z^T S + SA_z + K^T B^T S + SBK - K^T R_u K = 0$

Учитывая (2.25), получим

$A_z^T S + SA_z + K^T R_u^T K + K^T R_u K - K^T R_u K + Q = 0$ или

$$A_z^T S + SA_z + K^T R_u K + Q = 0. \qquad (2.29)$$

Так как Q - симметрическая положительно-определенная матрица, а $K^T R_u K$ - квадратичная форма, у которой $R_u$ положительно-определенная симметрическая матрица (или скаляр), то $K^T R_u K+Q=W$ также является симметрической положительно-определенной матрицей.

В результате (2.29) преобразуется к уравнению, известному в теории управления как уравнение Ляпунова:

$$A_z^T S + S A_z = -W \qquad (2.30)$$

## Решение матричного уравнения Ляпунова методом прямого интегрирования

Решение уравнения Ляпунова для устойчивой матрицы замкнутой системы имеет вид:

$$S = \int_0^\infty e^{A_z^T t} W e^{A_z t}\, dt \qquad (2.31)$$

Для численного решения уравнения Ляпунова в форме (2.31) перепишем его следующим образом:

$$S = \int_0^\infty e^{A_z^T t} W e^{A_z t} dt = \lim_{h\to 0} h \sum_{k=0}^{\infty} e^{A_z^T kh} W e^{A_z kh} \qquad (2.32)$$

Численное решение требует вычисления матричных экспонент $e^{A_z h}$. Известно несколько вариантов их вычисления. Все они базируются на идее разложения матричной экспоненты в бесконечный ряд с последующей аппроксимацией его конечным числом членов.

Воспользуемся представлением матричной экспоненты в виде:

$$e^{A_z h} = \Lambda = \left[(12E - 6hA_z + h^2 A_z^2)\right]^{-1}\left[(12E + 6hA_z + h^2 A_z^2)\right] \qquad (2.33)$$

Основным достоинством такого представления является то, что при выборе h>0 матрица $(12E - 6hA_z + h^2 A_z^2)^{-1}$ является неособенной и всегда существует. Очевидно, что $e^{A_z hk} = \Lambda^k$.

С учетом сказанного выше, выражение (2.32) можно записать в виде:

$$S = h(W + \Lambda^T W \Lambda + (\Lambda^T)^2 W \Lambda^2 + ... + (\Lambda^T)^k W \Lambda^k + ...) \qquad (2.34)$$

Если частичные суммы ряда (2.34) обозначить $S_k$, то рекуррентную форму вычисления S можно получить следующим образом:

$$S_k = S_{k-1} + \Lambda^T Z_{k-1} \Lambda,$$

причем

$$Z_{k-1} = (\Lambda^T)^{k-1} W_0 \Lambda^{k-1}; \quad \text{если } k = 1, \quad \text{то } S_{k-1} = W_0, \quad Z_{k-1} = W_0. \qquad (2.35)$$

При реализации данной рекуррентной процедуры параметр k в формулах определяется требуемой точностью вычисления S. Следует отметить, что точность вычисления S зависит также от выбора шага h. На основании результатов вычислительных экспериментов рекомендуется [10] выбирать h из условия $h = \dfrac{1}{200\left|\lambda^+(A_z)\right|}$, где $\lambda^+(\mathbf{A_z})$ - доминирующее собственное значение матрицы $A_z$, т.е. $\lambda^+(A_z) \geq \lambda_i(A_z)$.

Описанный метод численного решения уравнения Ляпунова позволяет сформулировать алгоритм решения уравнения Риккати и определить матрицу оптимальных коэффициентов обратной связи, реализующей минимизацию интегрального квадратичного критерия.

В укрупненной форме этот алгоритм можно представить следующим образом.

Исходные данные:

-матрицы A и B; начальные значения коэффициентов обратной связи (матрица K(0**)**); матрицы Q и R; значения шага интегрирования h и точности вычисления ε; k-номер шага интегрирования; m - номер итерации (m = 0).

*Шаг 1*. m = m+1, k = 1.

*Шаг 2*.Подготовка данных для m-ой итерации

$$A_z(m) = A - BK(m-1);\ W(m) = Q + K^T(m-1)RK(m-1).$$

*Шаг 3*. Определить значение матричной экспоненты $e^{A_z h}$ в m-ой итерации

$$\Lambda 1(m) = (12E - 6hA_z(m) + h^2 A_z^2(m))^{-1}.$$

$$\Lambda 2(m) = (12E + 6hA_z(m) + h^2 A_z^2(m)).$$

$$\Delta(m) = \Lambda 1(m)\Lambda 2(m).$$

*Шаг 4*. Формирование исходных данных для m-ой итерации и к = 1

$W_0(m) = hW(m)$

$\Delta S_{k-1}(m) = W_0(m)$

$S_k(m-1) = W_0(m)$

*Шаг 5*. Определение матрицы Риккати в m-ой итерации

$$\Delta S_k(m) = \Delta^T(m) \cdot \Delta S_{k-1}(m) \cdot \Delta(m)$$

$$S_k(m) = S_{k-1}(m) + \Lambda^T(m) \cdot \Delta S_{k-1}(m) \cdot \Lambda(m)$$

*Шаг 6*. Если норма $\Delta S(m) > \varepsilon$ в к-ом цикле суммирования (2.34),

то $S_{k-1}(m) = S_k(m);\quad \Delta S_{k-1}(m) = \Delta S_k(m);\ k = k+1$

и перейти к шагу 5, иначе - к шагу 7.

*Шаг 7*. Если m = 1, то переход к шагу 9, иначе к шагу 8.

*Шаг 8*. Если $\sqrt{\| S_k(m) - S(m-1) \|^2} < \varepsilon$, то переход к шагу 10, иначе - к шагу 9

*Шаг 9*. $K(m-1) = R^{-1}B^T S_k(m)$; $S(m-1) = S_k(m)$ и переход к шагу 1.

*Шаг 10*. Присвоить $S^* = S_k(m)$ и $K^* = R_u^{-1}B^T S_k(m)$.

*Шаг 11*. Конец.

## Решение матричного уравнения Ляпунова путем сведения его к векторно-матричной форме

Неизвестным в уравнении Ляпунова $A_z^T S + SA_z = -W$ является матрица S. Учитывая специфическую структуру матриц S, W, а также известное положение о

равенстве матриц (матрицы считаются равными, если равны соответствующие их элементы), решение матричного уравнения можно свести к решению эквивалентного векторно-матричного уравнения вида

$$A_э S_b = B_b, \qquad (2.36)$$

где $A_э$ - матрица, элементы которой являются комбинациями элементов матриц $A_z^T$ и $A_z$, $S_b, B_b$ - векторы, составленные из элементов матрицы S, B уравнения Ляпунова.

Решение векторно-матричного уравнения (2.36) возможно одним из следующих способов.

1. С использованием обратной матрицы $A_э$, т.е.

$$S_э = A_э^{-1} B_b \qquad (2.37)$$

Решение возможно, если матрица $A_э$ не является особенной.

2. Применение метода Гаусса решения систем линейных алгебраических уравнений.

Отмеченные выше симметрические свойства матриц **S** и **W** позволяют сформировать из матрицы S вспомогательный вектор $S_b$, а из матрицы W - вспомогательный вектор $B_b$. Матрица параметров объекта $A_z$ модифицируется в матрицу $A_э$. Элементы эквивалентной матрицы $A_э$ составлены из элементов $A_z$.

*Пример.*

Рассмотрим переход от матричного уравнения Ляпунова к его векторно-матричному аналогу (2.37) на примере системы 2-го порядка.

Пусть

$$A_z = \begin{pmatrix} A_z(1,1) & A_z(1,2) \\ A_z(2,1) & A_z(2,2) \end{pmatrix}, \quad S = \begin{pmatrix} S(1,1) & S(1,2) \\ S(2,1) & S(2,2) \end{pmatrix},$$

$$W = \begin{pmatrix} W(1,1) & W(1,2) \\ W(2,1) & W(2,2) \end{pmatrix}$$

Из условий симметричности матриц S, W будем иметь

$S(2, 1) = S(1, 2)$ , $W(2, 1) = W(1, 2)$

В результате векторы $S_b$, $B_b$ можно представить в виде:

$$S_b^T = (S_b(1),\ S_b(2),\ S_b(3)) = (S(1,1),\ S(2,1),\ S(2,2))$$
$$B_b^T = (B_b(1),\ B_b(2),\ B_b(3)) = (-W(1,1), -W(2,1), -W(2,2)) \qquad (2.38)$$

Перемножив $A_z^T S$ и $S A_z$ и сложив их, т.е. сформировав левую часть уравнения Ляпунова с учетом (2.38), получим связь элементов уравнения (2.36) с элементами уравнения Ляпунова в форме следующего векторно-матричного уравнения:

$$\begin{pmatrix} 2A_z(1,1) & 2A_z(2,1) & 0 \\ A_z(1,2) & A_z(1,1)+A_z(2,2) & A_z(2,1) \\ 0 & 2A_z(1,2) & 2A_z(2,2) \end{pmatrix} \begin{pmatrix} S_b(1) \\ S_b(2) \\ S_b(3) \end{pmatrix} = \begin{pmatrix} B_b(1) \\ B_b(2) \\ B_b(3) \end{pmatrix}, \qquad (2.39)$$

где $B_b(1) = -W(1,1),\ B_b(2) = -W(2,1),\ B_b(3) = -W(2,2)$.

Учитывая малый порядок системы (n = 2), для ее решения можно использовать известные формулы Крамера. Для систем больших размерностей метод сведения матричного уравнения к векторно-матричному требует большой вычислительной работы и поэтому не эффективен.

В общем случае, если размерность исходных матриц представить в виде $A_z \in R^{n\times n},\ S \in R^{n\times n},\ W \in R^{n\times n}$, то после модификации получим $A_b \in R^{n_b\times n_b}$, $S_b \in R^{n_b}, B_b \in R^{n_b}$, где $n_b = (n\times n) - \sum_{i=1}^{n}(n-i)$ – размерность вспомогательных элементов.

Полученные значения элементов вектора $S_b$ используются в дальнейшем для формирования элементов матрицы Риккати:

$S(1,1) = S_b(1),\ S(2,1) = S_b(2),\ S(1,2) = S_b(2),\ S(2,2) = S_b$.

В результате, после формирования матрицы S можно определить оптимальный коэффициент обратной связи по известной формуле (2.22).

### *в) Алгоритмизация задач оптимального терминального управления в дискретном времени*

Рассмотрим линейный стационарный объект, описываемый уравнением

$$x(k+1) = Ax(k) + Bu(k),\ x(0) = x_0 \neq 0,$$

где $t = kT_0$, а $T_0$ - период дискретизации по времени $(T_0 = \text{const})$, k - порядковый номер дискреты.

Задача системы управления состоит в формировании такого управляющего воздействия $u(k)$, которое приводит систему в конечное состояние $x(N) = 0$ и минимизирует квадратичный критерий качества

$$J = x^T(N)Sx(N) + \sum_{k=0}^{N-1}[x^T(k)Qx(k) + u^T(k)Ru(k)].$$

Матрицы S и Q неотрицательно определены и симметрические; матрица R положительно определена и симметрическая.

Для синтеза оптимальной обратной связи будем использовать принцип оптимальности Беллмана. Этот принцип базируется на так называемой концепции инвариантного вложения, при использовании которой решение сложной исходной проблемы заменяется решением некоторого количества аналогичных более простых проблем. При этом решение реализуется в виде многошагового процесса последовательного решения одношаговых процессов оптимизации.

В данном случае для реализации оптимального управления свободным движением системы из исходного ненулевого начального состояния в нулевое конечное состояние многошаговый процесс решения начинают с последнего участка, координаты которого известны (рис. 2.8).

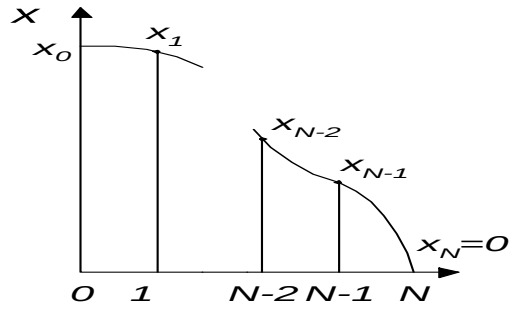


Рис. 2.8

Так как x(N) = 0, то матрица S в критерии может быть выбрана достаточно произвольно. Пусть для первого шага вычислений S = Q. Для последнего участка критерий J можно записать в виде

$$J = \min_{u(k);k=0,1,...,N-2} \left[ \min_{u(N-1)} \left\{ x^T(N)Qx(N) + \sum_{k=0}^{N-1} (x^T(k)Qx(k) + u^T(k)Ru(k)) \right\} \right] \qquad (2.40)$$

Первая часть в этом выражении является условной записью, отражающей многошаговый процесс минимизации критерия. Вторую часть выражения можно расписать более подробно:

$$\begin{aligned} \min_{u(N-1)} \{...\} &= \sum_{k=0}^{N-1} x^T(k)Qx(k) + \sum_{k=0}^{N-2} u^T(k)Ru(k) + \\ &+ \min_{u(N-1)} \left\{ x^T(N)Qx(N) + u^T(N-1)Ru(N-1) \right\} \end{aligned} \qquad (2.41)$$

В этом выражении предполагается, что управление u(N-1) на последнем участке постоянно и формируется в начале участка (точка N-1). Под действием этого управления объект переходит из точки x(N-1) в конечную точку x(N)=0. Первые два члена в этом выражении не зависят от управления u(N-1), поэтому они вынесены за знак минимизации. Обозначим вклад управления в критерий J на участке [N-1, N] через $J_{N-1,N}$.

Тогда из предыдущего выражения следует, что:

$$J_{N-1,N} = \min_{u(N-1)} \left\{ x^T(N)Qx(N) + u^T(N-1)Ru(N-1) \right\} \qquad (2.42)$$

Запишем ограничения, определяющие движение объекта по траектории на последнем ее участке:

$$\left.\begin{aligned} x(N) &= Ax(N-1) + Bu(N-1) \\ x^T(N) &= x^T(N-1)A^T + u^T(N-1)B^T \end{aligned}\right\} \qquad (2.43)$$

Подставим их в уравнение:

$$\begin{aligned} J_{N-1,N} \min_{u(N-1)} \{ & x^T(N-1)A^TQAx(N-1) + 2x^T(N-1)A^TQBu(N-1) + \\ & + u^T(N-1)B^TQBu(N-1) + u^T(N-1)Ru(N-1)\} = \\ = x^T(N-1)A^TQAx(N-1) + & \min_{u(N-1)} \{2x^T(N-1)A^TQBu(N-1) + \\ & + u^T(N-1)B^TQBu(N-1) + u^T(N-1)Ru(N-1)\} \end{aligned} \qquad (2.44)$$

Условие минимизации критерия $J_{N-1,N}$:

$$\frac{\partial}{\partial u(N-1)}\left\{J_{N-1,N}\right\}=0, \quad \text{и} \quad \frac{\partial^2 J_{N-1,N}}{\partial (u(N-1))^2}>0 \qquad (2.45)$$

$$\frac{\partial J_{N-1,N}}{\partial u(N-1)}=2B^TQAx(N-1)+2(B^TQB+R)u(N-1)=0$$

$$\frac{\partial^2 J_{N-1,N}}{\partial (u(N-1))^2}=2(B^TQB+R)>0.$$

Из первого уравнения следует, что

$u^*(N-1)=-(B^TQB+R)^{-1}B^TQA)x(N-1)$.

Обозначим $(B^TQB+R)^{-1}B^TQA)=K^*(N-1)$. В результате управление на участке N-1, N

$u^*(N-1)=-K^*(N-1)x(N-1)$

Второе неравенство удовлетворяется, так как R>0, а $Q\geq 0$.

Перепишем уравнение (2.44), выразив $J_{N-1,N}$ в виде функции текущих начальных условий x(N-1):

$$\min_{u(N-1)}\{...\}=\sum_{k=0}^{N-1}x^T(k)Qx(k)+\sum_{k=0}^{N-2}u^T(k)Ru(k)+x^T(N-1)(P_{N-1,N})x(N-1)=$$

$$=\sum_{k=0}^{N-2}(x^T(k)Qx(k)+u^T(k)Ru(k))+x^T(N-1)(P_{N-1,N}+Q)x(N-1),$$

где $P_{N-1,N}=A^TQA-K^T(N-1)(B^TQB+R)K(N-1)$.

Обозначим

$P_{N-1}=P_{N-1,N}+Q$ ,

$J_{N-1}=x^T(N-1)P_{N-1}x(N-1)$.

На следующем (втором от конца) шаге, задачу минимизации критерия J можно записать в виде

$$J=\min_{\substack{u(k)\\k=0,1,2,...,N-3}}...\left[\min_{u(N-2)}\left\{\sum_{k=0}^{N-2}(x^T(k)Qx(k)+u^T(k)Ru(k))+J_{N-1}\right\}\right]$$

По аналогии с (2.41) на части траектории $(N-2)\rightarrow N$ критерий J будет иметь вид

$$J=\min_{u(N-2)}\{...\}=\sum_{k=0}^{N-2}x^T(k)Qx(k)+\sum_{k=0}^{N-3}u^T(k)Ru(k)+$$

$$+\min_{u(N-2)}\left\{u^T(N-2)Ru(N-2)+x^T(N-1)P_{N-1}x(N-1)\right\}$$

Обозначим $J_{N-2,N}=\min_{u(N-2)}\{u^T(N-2)Ru(N-2)+x^T(N-1)Qx(N-1)+J_{N-1,N}\}$

Используя уравнение состояния объекта на предпоследнем шаге $x(N-1)=Ax(N-2)+Bu(N-2)$, выражение для $J_{N-2,N}$ можно записать в виде

$$J_{N-2,N} = x^T(N-2)A^T P_{N-1} Ax(N-2) +$$
$$+ \min_{U(N-2)} \{u^T(N-2)(R + B^T P_{N-1} B)u(N-2) + 2u^T(N-2)B^T P_{N-1} Ax(N-2)\}.$$

Используя условие оптимизации (2.45), на втором шаге получим

$$u^*(N-2) = -(R + B^T P_{N-1} B)^{-1} B^T P_{N-1} Ax(N-2) = -K(N-2)x(N-2),$$

где $K(N-2) = (R + B^T P_{N-1} B)^{-1} B^T P_{N-1} A$ - значение матричного коэффициента усиления на втором шаге.

После преобразований $J_{N-2,N}$, аналогичных преобразованиям $J_{N-1,N}$ на первом шаге, получим

$$J_{N-2,N} = x^T(N-2) P_{N-2,N} x(N-2),$$

где

$$P_{N-2,N} = A^T P_{N-1} [E - B(R + B^T P_{N-1} B)^{-1} B^T P_{N-1}] A =$$
$$= A^T P_{N-1} A - K_{N-2}^T (R + B^T P_{N-1} B) K_{N-2}.$$

Обозначим

$$P_{N-2} = P_{N-2,N} + Q,$$
$$J_{N-2} = J_{N-2,N} + x^T(N-2)Qx(N-2) = x^T(N-2)P_{N-2}x(N-2).$$

Последнее выражение отражает вклад в критерий J от минимизации управления на предпоследнем участке траектории, представленный в виде квадратичной формы от ***текущих начальных условий*** для предпоследнего участка. Можно показать, что матрица $P_{N-j}$ является эквивалентом матрицы Риккати (с точностью, определяемой h = To).

Преобразования, аналогичные произведенным выше, можно сделать и для последующих шагов вычислений. В результате для участков траектории с номерами от N-1 до 0 рекуррентные формулы для вычисления оптимальных коэффициентов $K_{N-j}$, матрицы Риккати $P_{N-j}$, управления $u_{N-j}$ и критерия $J_{N-j}$ имеют вид:

$$K_{N-j} = (R + B^T P_{N-j+1} B)^{-1} B^T P_{N-j+1} A,$$
$$P_{N-j} = Q + A^T P_{N-j+1} A + K_{N-j}^T (R + B^T P_{N-j+1} B) K_{N-j},$$
$$u_{N-j} = -K_{N-j} x_{N-j},$$
$$J_{N-j} = x_{N-j}^T P_{N-j} x_{N-j}. \qquad (2.46)$$

Начальное значение для $P_{N-j+1}$ при j=1, $P_{N-j+1} = P_N = Q$.

При j = N min J = $J_0 = x^T(0)\, P_0\, x(0)$.

Для структурно устойчивой системы ряды

$K_{N-1}, K_{N-2}, \ldots, K_1, K_0,$

$P_{N-1}, P_{N-2}, \ldots, P_1, P_0,$

$J_{N-1}, J_{N-2}, \ldots, J_1, J_0$ сходятся, а оптимальные значения имеют вид:

$K^*=K_0$; $P^*=P_0$; $J^*=J_0$.

Полученные значения $K^*$ используются для реализации оптимальной линейной обратной связи $u^*(k) = - K^* x(k)$.

### 2.1.4. Восстановление неизмеряемых координат состояния

Как было показано ранее, ЛК-оптимальное управление строится в форме пропорционального управления по каждой из координат состояния объекта.

Поскольку в большинстве случаев все координаты не могут быть измерены непосредственно, возникает задача восстановления неизмеряемых координат состояния. Построить устройства, которые на основе измерения доступных выходных переменных вырабатывали бы точные значения координат состояния, практически невозможно. Эту задачу удается решить приближенно, получая вместо вектора состояния x его оценки €. Соответствующие устройства называются наблюдателями (идентификаторами, оценивателями) состояния. Как правило, восстановление неизмеряемых координат производится путем использования специальной модели, соответствующей координатам состояния объекта. На вход модели подается то же воздействие u, что и на вход реального объекта. Выход модели сравнивается с выходом объекта. Возникающая в результате сравнения разность $y(k) - €(k)$ умножается на некоторую матрицу H и используется для коррекции результатов моделирования.

Описанная идея иллюстрируется структурной схемой системы управления с наблюдателем Люенбергера (рис.2.9).

Пусть динамический объект описывается следующей системой линейных векторно-матричных уравнений

$$\left.\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) \end{aligned}\right\}$$

Уравнение наблюдателя, в соответствии с приведенной на рис. 2.9 схемой и с учетом уравнения выхода, можно записать в виде:

$$\begin{aligned} €(k+1) &= A€(k) + Bu(k) + He(k) = \\ &= A€(k) + Bu(k) + H[y(k) - C€(k)] = A€(k) + Bu(k) + HC[x(k) - €(k)] \end{aligned} \qquad (2.47)$$

Обозначим ошибку оценивания вектора состояния (невязку)

$\delta x(k) = x(k) - €(k)$.

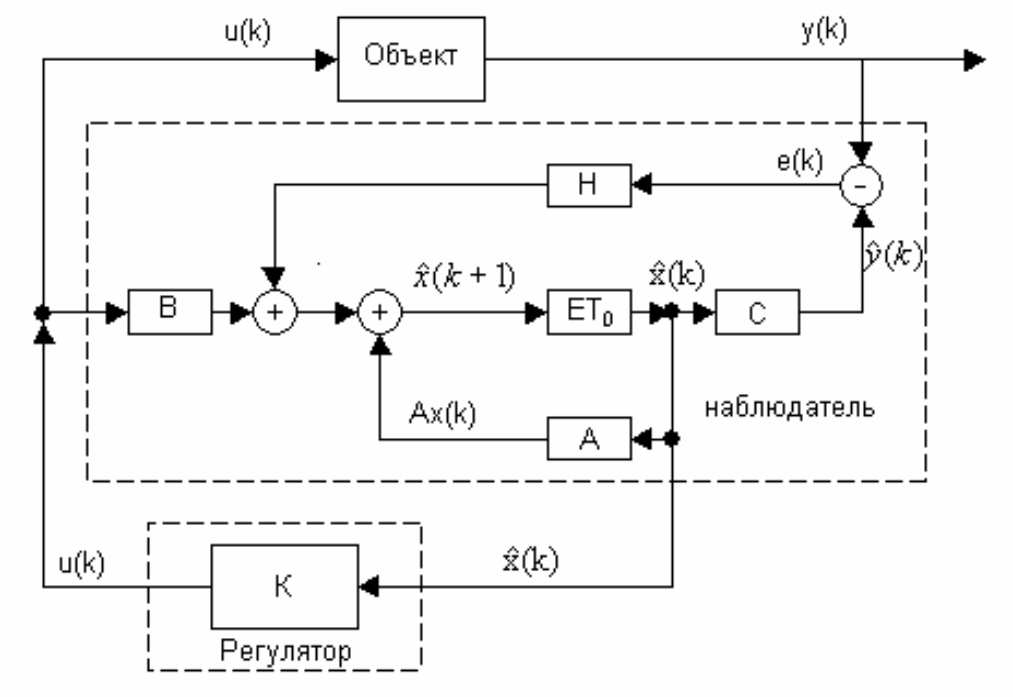



Рис. 2.9

Для $t = (k+1)T_0$ невязку можно записать в виде:

$$\delta x(k+1) = x(k+1) - \hat{x}(k+1) =$$

$$= Ax(k) + Bu(k) - A\hat{x}(k) - Bu(k) - HC[x(k) - \hat{x}(k)] = (A - HC)\delta x(k)$$

Из полученного выражения следует, что изменение невязки описывается однородным векторно-матричным уравнением. Ее поведение, а также динамические свойства всей системы управления существенно зависят от выбора матрицы H.

Для обеспечения асимптотической устойчивости контура невязки $\lim\limits_{k\to\infty}\delta x(k) \to 0$ необходимо, чтобы собственные числа $\lambda_i$ матрицы A–HC лежали внутри единичного круга, т.е. для уравнения det W=0, где W=λ – (A – HC), выполнялось условие $|\lambda_i| < 1, \quad i = \overline{1,n}$, при этом $\lambda = \mathrm{diag}[\lambda_i]$.

Наблюдатель представляет собой динамическую систему, связанную с объектом и нужен не сам по себе, а для создания обратной связи по вектору состояния (или по его оценке). Поэтому обеспечения устойчивости наблюдателя недостаточно. Однако наблюдатель и регулятор могут проектироваться порознь. Выбором матрицы H можно добиться, чтобы скорость затухания ошибки оценивания была не меньше заданной. В частности, в [15] показано, как выбрать оптимальное в смысле минимизации квадратичного критерия значение H.

### 2.1.5. Оценка неизмеряемых координат состояния объекта при наличии случайных помех (дискретный фильтр Калмана)

Фильтр Калмана обеспечивает получение оценки неизмеряемых координат объекта в случае, когда как сам объект, так и измеритель выходных координат находится под воздействием случайных помех.

Пусть имеется линейный стационарный объект, уравнение состояния которого в дискретном времени

$$x(k+1)=Ax(k)+F(v(k)+u(k)), \qquad (2.48)$$

а уравнение измерителя выходных координат

$$y(k+1)=Cx(k+1)+w(k+1), \qquad (2.49)$$

где

x(k) - вектор координат состояния, $x \in R^n$;

v(k) - входной векторный случайный сигнал (белый шум) с ковариационной матрицей V, $V \in R^m$;

u(k) - входной детерминированный сигнал, $u \in R^m$;

y(k) - вектор измеряемых выходных координат; $y \in R^r$;

w(k) - вектор шума измерений с ковариационной матрицей W, $w \in R^r$ (белый шум);

A - матрица параметров системы, $A \in R^{nxn}$;

F - матрица входа, $F \in R^{nxm}$;

C - матрица измерений; $C \in R^{rxn}$.

Требуется получить оценку вектора состояния $\hat{\mathbf{x}}(k+1)$ на основе измерений y(k+1), содержащих случайные погрешности в виде белого шума w(k+1).

Калмановская фильтрация использует идею прогнозирования состояния объекта в (k+1)-й момент времени на основе оценки координат состояния в k-й момент времени и текущего измерения выходных координат - y(k+1).

Прогнозируемое значение выходных координат сравнивается с измеряемым значением, и невязка (отличие прогнозируемого и измеренного значений) используется для коррекции оценки координат состояния в текущий момент времени. При этом коррекция производится в форме аддитивного сигнала, получаемого путем умножения невязки на корректирующий коэффициент (коэффициент Калмана). Корректирующий коэффициент вычисляется на каждом шаге и обеспечивает получение оценки $\hat{\mathbf{x}}(k+1)$ в виде:

$$\hat{\mathbf{x}}(k+1) = A\hat{\mathbf{x}}(k) + K(k+1)[y(k+1) - \tilde{\mathbf{y}}(k+1)] \qquad (2.50)$$

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Новая оценка | = | Оценка предсказания на предыдущем шаге | + | Коэффициент коррекции | [Новое измерение | – | Предсказанное измерение] |

На рис. 2.10 приведена структурная схема фильтра Калмана, реализующего описанную выше идею. Очевидно, что основной проблемой получения $\hat{\mathbf{x}}(k+1)$ является вычисление корректирующего коэффициента K(к+1).

Для ее решения необходима следующая априорная информация:

- параметры матриц A, F и C;
- $M\{v(k)\} = \bar{v}$ - математическое ожидание случайного входного сигнала;
- $COV\{v(k)\} = M\{(v(i)-\bar{v})(v(i)-\bar{v})^T\} = V\delta_{ij}$ - ковариация входного сигнала (шума) v(i);
- $M\{w(k)\} = \bar{w}$ – математическое ожидание шума измерения;
- $COV\{(w(k)\} = M\{(w(i)-\bar{w})(w(i)-\bar{w})^T\} = W\delta_{ij}$- ковариация шума измерения, $\delta_{ij}$ - функция Кронекера:

$$\delta_{ij} = \begin{cases} 1 & \text{при} \quad i = j, \\ 0 & \text{при} \quad i \neq j. \end{cases}$$

При i = j ковариация переходит в дисперсию V и W. В дальнейшем термин ковариация будет использоваться в этом смысле.

Матрицы A и F являются дискретными эквивалентами соответствующих непрерывных матриц.

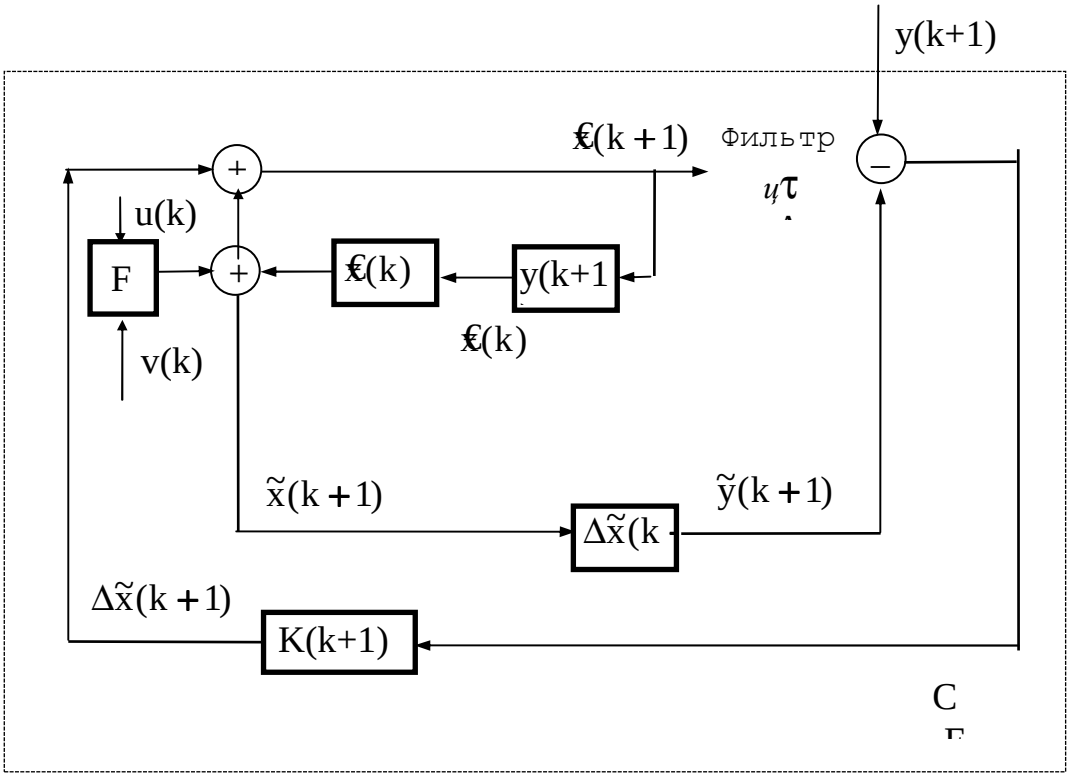



Рис. 2.10

Для определения оптимального коэффициента коррекции воспользуемся методом взвешенного усреднения двух независимых векторных случайных величин $x_1$ и $x_2$ с минимальной дисперсией.

При этом оценка среднего значения

$\hat{x} = x_1 + K_1(x_2 - x_1)$, где $K_1$-коэффициент взвешивания.

Пусть $K_1 = KC$, тогда

$\hat{x} = x_1 + KC(x_2 - x_1) = x_1 - KCx_1 + KCx_2$ .

Обозначим $y_2 = Cx_2$. Тогда оценку $\hat{x}$ можно записать в виде

$\hat{x} = (E - KC)x_1 + Ky_2$, где E - единичная матрица.

Обозначим дисперсию оценки $\hat{x}$ через P:

$P = M\{(\hat{x} - \bar{\hat{x}})(\hat{x} - \bar{\hat{x}})^T\}$, где $M\{\cdot\}$ - второй момент; $\bar{\hat{x}}$ – математическое ожидание оценки. Тогда $\bar{\hat{x}} = M\{(E - KC)x_1 + Ky_2\} = (E - KC)\bar{x}_1 + K\bar{y}_2$.

Обозначим дисперсию случайной величины $x_1$ –

$Q = M\{(x_1 - \bar{x}_1)(x_1 - \bar{x}_1)^T\}$ , а дисперсию $y_2$ – $Y = M\{(y_2 - \bar{y}_2)(y_2 - \bar{y}_2)^T\}$.

Учитывая независимость случайных величин $x_1$ и $y_2$, дисперсию оценки $\hat{x}$ можно после некоторых преобразований получить в виде:

$$P = (E - KC)Q(E - KC)^T + KYK^T.$$

Для получения средневзвешенной оценки двух случайных величин с минимальной дисперсией необходимо, чтобы коэффициент взвешивания K выбирался из условия

$$\frac{dP}{dK} = 0 \quad \text{и} \quad \frac{d^2P}{dK^2} > 0.$$

Перепишем P в следующем виде:

$P = (Q - KCQ)(E - C^TK^T) + KYK^T = Q - KCQ - QC^TK^T + KCQC^TK^T + KYK^T =$

$= Q - QC^TK^T - QC^TK^T + KCQC^TK^T + KYK^T.$

Так как P и Q - диагональные матрицы, то и составляющие их являются также диагональными матрицами.

Поэтому $(KCQ)^T = QC^TK^T = KCQ$.

В результате получим:

$P = Q - 2QC^TK^T + KCQC^TK^T + KYK^T$,

$\frac{dP}{dK} = -2QC^T + 2KCQC^T + 2KY = 0$, следовательно, $K(CQC^T + Y) = QC^T$ и

$$K^* = QC^T(CQC^T + Y)^{-1} \qquad (2.51)$$

Учитывая значение $K^*$, выражение для P можно переписать в виде

$P = Q - 2QC^TK^T + K(CQC^T + Y)K^T =$

$= Q - 2QC^TK^T + QC^T(CQC^T + Y)^{-1}(CQC^T + Y)K^T =$

$= Q - QC^TK^T = Q(E - C^TK^T)) = Q - KCQ.$ (2.52)

Очевидно, что условие $\frac{d^2P}{dK^2} > 0$ выполняется, так как дисперсия $Y \geq 0$, а $CQC^T$ является квадратичной формой, причем матрица Q неотрицательно определена.

Применительно к динамическому объекту полученные выше выражения можно интерпретировать следующим образом. Пусть $x_1$ соответствует предсказанному значению $\tilde{x}$, а $y_2$ - измеренному значению y. Дисперсию предсказания в этом случае можно записать следующим образом:

$$Q(k+1) = M\{(\tilde{x}(k+1)\} - \bar{\tilde{x}}(k+1))\ (\tilde{x}(k+1) - \bar{\tilde{x}}(k+1))^T\},$$

где $\bar{\tilde{x}}(k+1) = M\{\tilde{x}(k+1)\} = M\{A€(k) + F(v(k) + u(k))\}$.

Пусть для упрощения дальнейшего вывода u(k)=0. Тогда

$\bar{\tilde{x}}(k+1) = A \cdot M\{\tilde{x}(k+1) - \bar{\tilde{x}}(k+1) = A€_0(k) + Fv_0(k), \{€(k)\} + FM\{v(k)\} = A\bar{€}(k) + F\bar{v}(k)$,

$(\tilde{x}(k+1) - \bar{\tilde{x}}(k+1))^T = (€_0(k))^T A^T + (v_0(k))^T F$,

где $\hat{x}_0$ и $v_0$ - центрированные значения соответственно € и v:

$€_0(k) = €(k) - \bar{€}(k)$,

$v_0(k) = v(k) - \bar{v}(k)$.

Учитывая независимость € и v, выражение для дисперсии предсказания можно записать в виде

$Q(k+1) = M\{(A€_0(k) + Fv_0(k))\ ((€_0(k))^T + (v_0(k))^T F^T)\} =$

$AP(k)\ A^T + 0 + 0 + Fv(k)F^T = A(P(k))^T A^T + Fv(k)F^T$.

Дисперсия сигнала измерения:

$Y(k+1) = M\{(y(k+1) - \bar{y}(k+1))(y(k+1) - \bar{y}(k+1))^T\}$.

Так как $y(k+1) = Cx(k+1) + w(k+1)$,

то $\bar{y}(k+1) = CM\{x(k+1)\} + M\{w(k+1)\} = C\bar{x}(k+1) + \bar{w}(k+1)$.

В результате

$Y(k+1) = M\{w_0(k+1) \cdot w_0^T(k+1)\} = W(k+1)$,

а выражения для коэффициента коррекции (2.51) и оценки (2.52) получим в виде:

$K(k+1) = Q(k+1)C^T[CQ(k+1)C^T + W(k+1)]^{-1}$,

$€(k+1) = A€(k) + F\bar{v} + K(k+1) \cdot (y(k+1) - CA€(k) - CF\bar{v})$

Учитывая (2.52), дисперсию оценки можно записать в виде:

$P(k+1) = Q(k+1) - K(k+1)CQ(k+1)$.

Если u≠0 и его можно измерить, то вычисление $\tilde{x}(k+1)$ требуется производить, используя модификацию выражения (2.48) следующим образом:

$\tilde{x}(k+1) = A€(k) + F(\bar{v} + u(k))$.

В результате получим

$€(k+1) = A€(k) + F(\bar{v} + u(k)) + K(k+1)[y(k+1) - CA€(k) - CF(\bar{v} + u(k))]$.

**Алгоритм реализации дискретного фильтра Калмана для линейного стационарного объекта**

Исходные данные:

- параметры объекта и измерителя (матрицы A, F и C);
- характеристики случайных сигналов:
  $\bar{v}$ - математическое ожидание входного сигнала v(k);
  V- ковариационная матрица (дисперсия) входного сигнала v(k);
  W - ковариационная матрица (дисперсия) помехи измерения w(k) ($\bar{w}$ =0);

- N - объем выборки;
- u(k) - неслучайный входной сигнал;
- P(0), $\hat{\mathbf{x}}(0)$ - начальные значения соответственно ковариационной матрицы оценки и оценки вектора состояния (P(0)=0, $\hat{\mathbf{x}}(0)=0$).

*Шаг 1.* Формирование предсказываемого значения $\tilde{x}(k+1) = A\hat{x}(k) + F(u(k) + \bar{v})$.

*Шаг 2.* Получение невязки измерения

$$e(k+1) = y(k+1) - C\tilde{x}(k+1).$$

*Шаг 3.* Вычисление ковариационной матрицы оценки предсказания $\tilde{x}(k+1)$:

$Q(k+1) = AP(k)A^T + FVF^T$.

*Шаг 4.* Определение коэффициента коррекции

$$K(k+1) = Q(k+1)C^T [CQ(k+1)C^T + W]^{-1}.$$

*Шаг 5.* Формирование корректирующего сигнала

$$\Delta\tilde{x}(k+1) = K(k+1) \cdot e(k+1).$$

*Шаг 6.* Формирование оценки

$$\hat{x}(k+1) = \tilde{x}(k+1) + \Delta\tilde{x}(k+1).$$

*Шаг 7.* Вычисление ковариационной матрицы оценки

$$P(k+1) = Q(k+1) - K(k+1)CQ(k+1).$$

*Шаг 8.* P(k) = P(k + 1), $\hat{\mathbf{x}}(k) = \hat{\mathbf{x}}(k+1)$, k = k + 1.

*Шаг 9.* Если k < N, то переход к шагу 1; иначе – к шагу 10.

*Шаг 10.* Конец.

## 2.2. Одноуровневое многоцелевое управление

Пусть имеется объект или процесс, управление в котором должно быть организовано таким образом, чтобы обеспечивалось достижение экстремального значения нескольких критериев оптимальности.

Одна из основных проблем при решении многокритериальных задач – концептуальная и состоит в формулировке критерия оптимальности. Пока не решена эта проблема, хорошо разработанные методы теории оптимизации не применимы, так как многоцелевая задача оптимизации математического смысла не имеет.

Решение подобных задач возможно лишь в том случае, когда многокритериальная задача может быть сведена к однокритериальной, т.е. когда будет решен вопрос, что для данной конкретной задачи считать оптимальным решением. Для многокритериальных задач не существует наилучшего решения сразу по всем критериям. Очевидно, что преодоление проблемы неопределенности цели возможно лишь при введении извне в задачу некоторых условий, называемых компромиссами.

Существует несколько вариантов таких компромиссных решений. Рассмотрим те из них, которые получили наибольшее распространение при решении задач оптимизации управления в статике.

### 2.2.1. Сведение многокритериальной задачи к однокритериальной по принципу свертки критериев

Чаще всего используется линейная свертка критериев

$$F(x) = \sum_{i=1}^{k} \omega_i f_i(x),$$

где k – число критериев;

$x$ – вектор аргументов целевых функций $f_i(x)$ и $F(x)$;

$\omega_i$ – весовые коэффициенты, $\omega_i = \dfrac{\partial F(x)}{\partial f_i(x)}$.

Если $f_i(x)$ выразить в форме безразмерных величин, то $\sum_{i=1}^{k} \omega_i = 1.$

Таким образом, способ линейной свертки эквивалентен взвешиванию критериев. Основной проблемой при использовании метода линейной свертки является определение значений весовых коэффициентов $\omega_i$. Наибольшее распространение получил метод экспертных оценок. В дальнейшем будем считать, что $\omega_i$ известны (получены методом экспертных оценок). Вычисление

частных критериев определяется видом конкретной задачи и здесь не рассматривается.

### 2.2.2. Использование метода уступок

В методе уступок все критерии ранжируются в порядке их важности, т.е. их можно записать в виде

$f_1(x) \succ f_2(x) \succ .... \succ f_k(x)$, где $\succ$ – знак предпочтения.

Решение $F[f_1(x), f_2(x), ..., f_k(x)]$ считается оптимальным, если найдено экстремальное значение последнего критерия в ряду предпочтений, при условии гарантированных значений предшествующих критериев. На практике в большинстве случаев решение таких задач сводится к поиску условного экстремума, т. е. решению оптимизационной задачи с ограничениями, задающими области работоспособности реальной системы.

Метод уступок является расширением известных методов поиска условного экстремума, при котором в систему ограничений вводятся также и ограничения на допускаемое изменение (уступка) от экстремального значения (k - 1) критериев в ряду предпочтений.

В самом общем виде алгоритм решения многокритериальной задачи в одноуровневых системах по методу уступок можно представить следующим образом.

*Шаг 1*. Решается задача поиска экстремального значения целевой функции (критерия) $f_1(x)/\varphi_1(x) \geq 0$. Оставшиеся (k - 1*)* критериев не рассматриваются.

*Шаг 2*. Делается "уступка" по первому критерию путем уменьшения найденного экстремального значения $f^*(x)$ в $m_1$ раз, т.е. принимается $f_1(x) = m_1 f_1^*(x)$, $(m_i < 1, i = \overline{1, n})$.

*Шаг 3*. В задачу поиска экстремума вводится второй критерий $f_2(x)/\varphi_2(x) \geq 0$, а первый критерий принимается равным $f_1(x) = m_1 f_1^*(x) = c_1 = \text{const}$. Решается задача поиска условного экстремума $f_2(x)\Big/{\varphi_2(x) \geq 0 \atop f_1(x) \geq c_1}$ и определяется $f_2^*(x)$ - экстремум функции при фиксированном значении критерия $f_1(x)$.

*Шаг 4*. Делается уступка по второму критерию $f_2^*(x)$ в форме $f_2(x) = m_2 f_2^*(x)$.

*Шаг 5*. В задачу поиска условного экстремума $f_3(x)/\varphi_3(x) \geq 0$ вводится и фиксируется второй критерий $f_2(x) = m_2 f^*(x) = c_2$ и находится

экстремальное значение критерия $f_3(x)\Big/\begin{matrix}\varphi_3(x)\ge 0\\ f_1(x)\ge c_1\\ f_2(x)\ge c_2\end{matrix}$, равное $f_3^*(x)$.

*Шаг 6*. Решение заканчивается, когда будут учтены все k критериев.

Очевидно, что основной проблемой метода уступок является выбор значений коэффициентов уступок $m_i$. Вследствие их неопределенности система ограничений может оказаться несовместной. В таких случаях необходимо изменить коэффициенты уступок и повторить описанную выше процедуру поиска оптимального решения.

### 2.2.3. Использование метода равных и наименьших отклонений

Пусть все критерии $f_i(x)$, $i=\overline{1,k}$ равнозначны. Задача заключается в том, чтобы найти такое решение, при котором отклонение от экстремального значения каждого из критериев $f_i^*$ было бы равным и минимальным. Другими словами, требуется найти координаты некоторой точки 0, отстоящей от точек экстремального значения функций $f_1^*, f_2^*, f_3^*$ на равное и наименьшее расстояние.

Математически задачу обеспечения равных отклонений можно записать в виде выполнения условий:

$$\left|\frac{f_1(x)-f_1^*(x)}{f_1^*(x)}\right|=\left|\frac{f_2(x)-f_2^*(x)}{f_2^*(x)}\right|=\ldots=\left|\frac{f_k(x)-f_k^*(k)}{f_k^*(x)}\right|. \qquad (2.53)$$

В случае ранжирования критериев условие равенства отклонений можно записать в виде:

$$\beta_1\left|\frac{f_1(x)-f_1^*(x)}{f_1^*(x)}\right|=\beta_2\left|\frac{f_2(x)-f_2^*(x)}{f_2^*(x)}\right|=\ldots=\beta_k\left|\frac{f_k(x)-f_k^*(k)}{f_k^*(x)}\right|,$$

где $\beta_i$ - весовые коэффициенты, $0<\beta_i\le 1$, $(i=\overline{1,k})$.

Пусть, например, многокритериальная задача решается при условии максимизации частных критериев. Тогда условие (2.53) для любых двух критериев можно переписать в следующем виде:

$$\alpha_1 f_1(x)=\alpha_i f_i(x),\ i=\overline{2,k}, \qquad (2.54)$$

где $\alpha_i=\dfrac{1}{f_i^*(x)}$; $i=\overline{1,k}$ или $\alpha_1 f_1(x)-\alpha_i f_i(x)=0$.

Для случая, когда один критерий максимизируется, а другой минимизируется, условие (2.54) записывается в виде:

$$\alpha_1 f_1(x)+\alpha_i f_i(x)=2. \qquad (2.55)$$

Так как для решения задачи необходимо стремиться к минимуму отклонения от оптимума, т.е. к минимизации выражения

$$\left|\frac{f_i(x)-f_i^*(x)}{f_i^*(x)}\right|=\left|\frac{\Delta f_i(x)}{f_i^*(x)}\right|, \qquad (2.56)$$

то задачу поиска компромисса, обеспечивающего равные и минимальные отклонения от оптимального значения каждого из критериев, можно решить, введя в рассмотрение так называемую замещающую задачу.

Замещающая задача строится из основной путем использования только одного из критериев исходной задачи и расширения системы ограничений путем добавления к ограничениям исходной задачи (например, задачи линейного программирования) ограничений вида:

$$\left.\begin{array}{l}\sum\limits_{i=1}^{k} c_i x_i - f_1(x)=0\\ \ldots\ldots\ldots\ldots\ldots\ldots\\ \sum\limits_{i=1}^{k} c_i x_i - f_k(x)=0\end{array}\right\},$$

$$\alpha_1 f_1(x)-\alpha_i f_i(x)=0,$$

$$\alpha_1 f_1(x)+\alpha_j f_j(x)=2,$$

$$\left.\begin{array}{l} i\in f_i(x)\Rightarrow \max\\ j\in f_j(x)\Rightarrow \min\end{array}\right\}$$

Построение замещающей задачи можно проиллюстрировать на следующем примере [11].

*Пример*. Найти методом равных и минимальных отклонений решение одноуровневой многокритериальной задачи линейного программирования следующего вида:

$$\max f_1(x_1,x_2)=x_1+2x_2,$$

$$\min f_2(x_1,x_2)=x_1+x_2$$

при ограничениях $x_1+2x_2\ge 6$;

$$x_1\le 4,\ x_2\le 4;\ x_1\ge 0,\ x_2\ge 0.$$

Симплекс-методом находим решение $f_i^*(x_1,x_2), i=1,2$ по каждому из критериев.

Учитывая разную направленность критериев $f_1(x_1,x_2)$ и $f_2(x_1,x_2)$, дополнительное условие равных относительных отклонений следует записать в виде:

$$\alpha_1 f_1(x_1,x_2)+\alpha_2 f_2(x_1,x_2)=2,$$

где $\alpha_1=\dfrac{1}{f_1^*(x_1,x_2)}$, $\alpha_2=\dfrac{1}{f_2^*(x_1,x_2)}$.

В результате замещающую задачу в канонической форме можно сформулировать следующим образом:

найти $\max\{f_1(x_1, x_2) = x_1 + 2x_2\}$ при ограничениях

$$x_1 + 2x_2 - x_3 = 6, \quad x_1 + x_4 = 4, \quad x_2 + x_5 = 5,$$

$$x_1 + 2x_2 - f_1(x_1, x_2) = 0,$$

$$x_1 + x_2 - f_2(x_1, x_2) = 0,$$

$$\alpha_1 f_1(x_1, x_2) + \alpha_2 f_2(k_1, x_2) = 2,$$

$$x_i \geq 0, \quad i = 1,3,$$

$$f_k(x_1, x_2) \geq 0, \quad k = 1,2.$$

Очевидно, что при другой формулировке исходной задачи будет изменяться и форма замещающей задачи.

## 2.3. Многоуровневое многоцелевое управление

Основным недостатком одноуровневого многоцелевого управления можно считать необходимость привнесения извне дополнительных компромиссных решений, позволяющих свести многокритериальное управление к однокритериальному. Этот недостаток можно устранить или смягчить переходом к многоуровневому управлению. Принятие компромиссных решений в этом случае возлагается на вышестоящий уровень, имеющий соответствующий решающий орган.

При создании систем многоцелевого многоуровневого управления необходимо решать две главные проблемы: – структурную проблему (или проблему стратификации) и проблему координации.

Структурная проблема заключается в оптимальной декомпозиции общей задачи управления на ряд подзадач, решаемых на различных уровнях. Формализация решений задач декомпозиции выходит за рамки данного пособия и здесь не рассматривается. Вторую проблему – проблему координации и методы ее решения, не снижая общности, будем рассматривать на примере двухуровневых систем управления в статике
(рис. 2.3).

### *Координация. Основные понятия и определения*

В многоуровневых системах априори предполагается, что подсистемы обладают определенной свободой выбора. Цель координации или координирующего управления заключается в обеспечении согласованных действий подсистем нижнего уровня для достижения глобальной цели, стоящей перед управляемым объектом или процессом. Таким образом, координатор

должен обладать правом вмешиваться в действия решающих органов управления локальными подсистемами.

Вследствие этих особенностей многоуровневые многоцелевые системы принято также называть иерархическими системами управления. Очевидно, что иерархическая система работоспособна тогда, когда она координируема управляющими воздействиями от верхнего уровня.

Под координируемостью понимается такое действие координирующего элемента, когда при наличии соответствующих сигналов могут быть решены локальные задачи на нижестоящем уровне.

Применительно к двухуровневой системе можно отметить следующее:

– двухуровневая система координируема, если задачи, решаемые на уровне нижестоящих решающих органов, могут быть скоординированы относительно глобальной задачи,

– нижестоящие решающие органы координируемы, если могут быть скоординированы решаемые ими задачи [4].

Таким образом, ответ координируема ли иерархическая система, может быть получен тогда, когда получены результаты решения задач нижестоящих решающих органов, т.е. после вмешательства координирующего органа. Такое вмешательство возможно двумя способами:

– вмешательство до принятия решения локальными решающими органами;

– вмешательство после принятия решения локальными решающими органами.

В первом случае на основании имеющейся информации координатор формирует воздействия на локальные решающие органы, которые должны принимать решения с учетом полученной от координатора информации.

Во втором случае, координатор анализирует принятые локальными решающими органами решения и, если необходимо, корректирует их, но уже на следующем цикле принятия решения.

### 2.3.1. Принципы координации подсистем в многоцелевой иерархической системе управления

В зависимости от способов организации координирующего управления принято различать следующие виды координации подсистемы:

– координация по принципу прогнозирования взаимодействий;

– координация по принципу оценки взаимодействий;

– координация по принципу согласования (развязывания) взаимодействий.

***Координация по принципу прогнозирования взаимодействий***

В этом случае реализуется описанный выше способ вмешательства координатора до принятия решения в действия решающих органов локальных подсистем.
В самом общем виде его можно представить следующим образом. На рис. 2.11 приведена структурная схема двухуровневой системы, в которой он реализован. Здесь приняты следующие обозначения:
$РО_2$ и $РО_{11,}$ $РО_{12}$ - соответственно координатор и решающие органы подсистем 1 и 2,
ГЦ и $ЛЦ_1$, $ЛЦ_2$ - соответственно глобальная целевая функция и локальные целевые функции подсистем 1 и 2,
$s_1$ и $s_2$ - связующие переменные подсистем,
$\hat{s}_1$ и $\hat{s}_2$ - прогнозируемые значения связующих переменных ($\hat{s}_1 = g_1$; $\hat{s}_2 = g_2$),
$\varepsilon_1, \varepsilon_2$ - соответственно отклонения связующих переменных от прогнозируемых значений:
$\varepsilon_1 = \hat{s}_1 - s_1(u_1, z_1)$, $\varepsilon_2 = \hat{s}_2 - s_2(u_2, z_2)$,
$U_1$, $U_2$ – управляющие воздействия, формируемые решающими органами $РО_{11}$ и $РО_{12}$ соответственно,
$y_1$, $y_2$ – координаты "внутреннего" состояния подсистем 1 и 2,
$z_1$, $z_2$ – воздействия подсистем друг на друга.

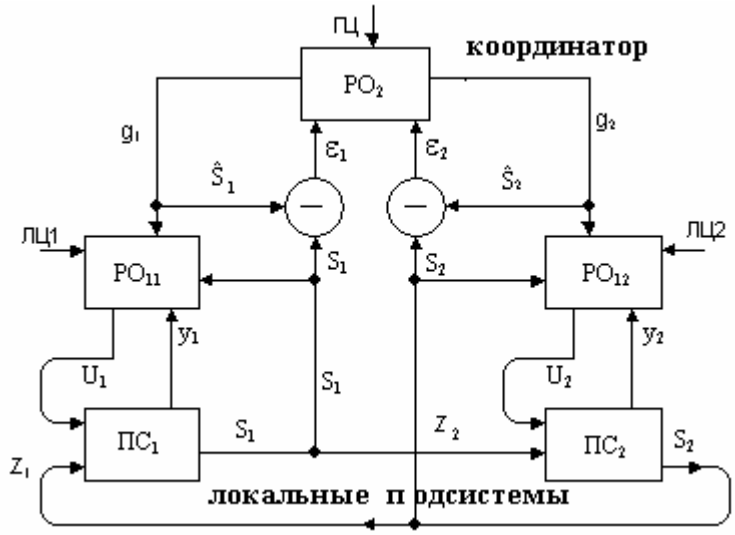



Рис. 2.11

Работа координатора на принципе прогнозирования взаимодействий происходит следующим образом. Исходя из ГЦ и имея информацию о состоянии всей системы, координатор прогнозирует желательные значения связующих переменных $\hat{s}_1$ и $\hat{s}_2$. Эти значения передаются в решающие органы подсистем $РО_{11}$ и $РО_{12}$. Решающие органы в соответствии с информацией о текущем

состоянии $y_i$ и заданным (прогнозируемым) значением связующих переменных $\hat{s}_i$ формируют управляющие воздействия $u_i$.

Отклонения текущих значений связующих переменных от прогнозируемых значений $\varepsilon_1, \varepsilon_2$ являются сигналом продолжения коррекции текущих значений связующих переменных $\hat{s}_1$ и $\hat{s}_2$.

Таким образом, координатор при использовании координации по принципу прогнозирования выступает в роли "диктатора", требуя обязательного учета спрогнозированных значений связующих переменных при решении своих локальных целей.

### *Координация по принципу согласования (развязывания) взаимодействий*

Координация на принципе согласования взаимодействий относится к типу координаций после принятия решений решающими органами локальных подсистем.

В общем виде структурная организация такой координации показана на рис. 2.12.

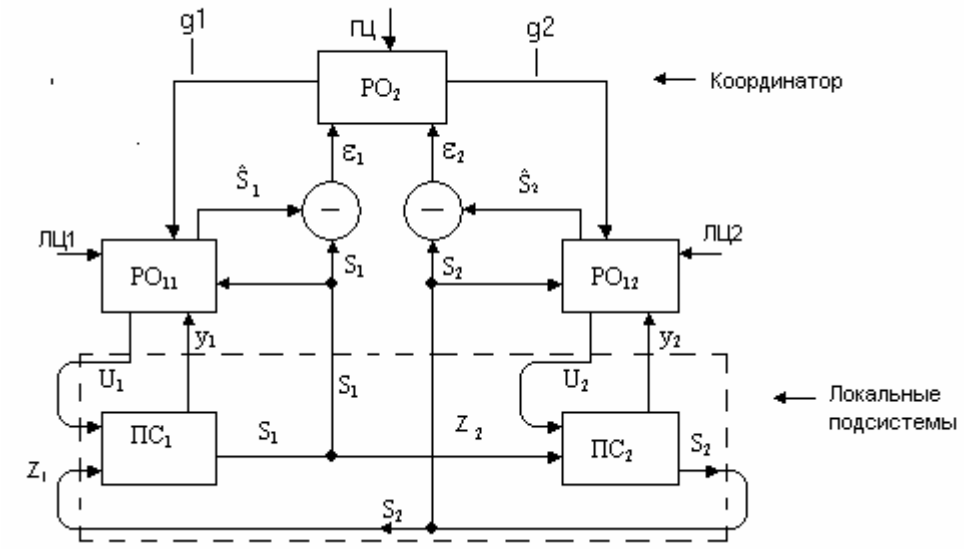



Рис. 2.12

Система, использующая координацию на принципе согласования взаимодействий, работает следующим образом. Решающие органы локальных подсистем на основании имеющейся у них информации и заданных локальных целей управления прогнозируют свои связующие переменные $\hat{s}_1$ и $\hat{s}_2$. Отклонения текущих значений $s_i$ от прогнозируемых $\hat{s}_1$ $\varepsilon_1 = \hat{s}_1 - s_1(u_1, z_1)$ и $\varepsilon_2 = \hat{s}_2 - s_2(u_2, z_2)$ передаются в координатор ($РО_2$), который в соответствии со своей глобальной целью формирует координирующие воздействия $g_1$ и $g_2$. Эти воздействия модифицируют управление $u_1$ и $u_2$, так, чтобы, с одной стороны, достигалась

глобальная цель, а с другой стороны, реализовывались цели локальных подсистем.

При этом формирование координирующих воздействий на очередном шаге в самом общем виде можно представить следующим образом:

$$g_1(k+1) = g_1(k) + \varphi_1(\varepsilon_1(k)),$$
$$g_2(k+1) = g_2(k) + \varphi_2(\varepsilon_2(k))$$

В результате роль координатора сводится к согласованию принимаемых решающими органами локальных подсистем решений с тем, чтобы достигались не только локальные цели, но и глобальная цель.

### *Координация по принципу оценки взаимодействий*

Этот вид координации является модификацией принципа прогнозирования взаимодействий, когда ставится задача поддержания связующих переменных в некоторой допустимой области. Структурная организация такого типа координации подобна координации по принципу прогнозирования взаимодействий (рис. 2.11). Ее отличает лишь то, что с координатора вместо прогнозируемых значений $\varepsilon_i$ на РО локальных подсистем передаются значения допустимой области $\Omega\varepsilon_i$, в которой может находиться связующая переменная. Очевидно, что координация по принципу оценки взаимодействий может существенно расширить область возможного применения координации, использующей прогнозирование взаимодействий.

### 2.3.2. Реализация координирующих управлений

Реализация координирующих управлений в любой иерархической системе возможна тремя различными способами:

– путем модификации целей (целевых функций локальных подсистем);

– путем модификации образов (ограничений, определяющих взаимодей-ствия подсистем по связующим переменным);

– путем модификации целей и образов.

*Примечание. Для лучшего понимания способов формирования координирующих воздействий их описание, приводимое ниже, предельно упрощено.*

Для синтеза координирующих управлений, в первую очередь, необходимо ввести в рассмотрение формализованную модель подсистем. На рис. 2.13 приведен возможный вариант такой модели в форме статической модели типа вход-выход.

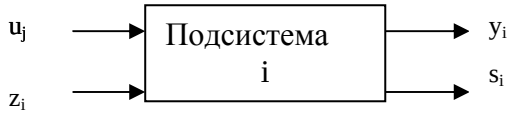



Рис. 2.13

Входные сигналы:

$u_i$ – управляющее воздействие решающего органа РОi;

$z_i$ – воздействие на i -ую подсистему других подсистем по связующим переменным:

$$z_i = \left( \sum_{j=1}^{k} c_{ji}\, s_j \right),\ i \notin j,\ k \le n-1;$$

$c_{ji}$ – весовые коэффициенты (коэффициенты влияния).

Выходные сигналы:

$y_i$ – вектор внутренних координат состояния подсистемы $y_i = \gamma_i(u_i, z_i)$;

$s_i$ – вектор связующих переменных i - ой подсистемы $s_i = \varphi_i(u_i, z_i)$.

Если, в первом приближении, при формировании связующих переменных не учитывать $y_i$, то целевую функцию i - ой подсистемы можно записать в виде - $f_i(u_i, z_i)$.

Пусть функции $f_i(u_i, z_i)$, $i = \overline{1, n}$ – гладкие (дифференцируемые) и выпуклые. В результате задача оптимального управления i-ой подсистемой может быть представлено в форме задачи поиска условного экстремума:

найти $\text{ext}\{f_i(u_i, z_i)\}$, при ограничениях: $s_i - \varphi_i(u_i, z_i) = 0,\ z_i - \sum_j c_{ji} s_j) = 0.$

Как известно, эта задача может быть сведена к классической задаче Лагранжа поиска безусловного экстремума вида:

$$L_i(u, z, \mu, \rho) = f_i(u_i, z_i) + \mu_i[s_i - \varphi_i(u_i, z_i)] + \rho_i(z_i - \sum_{j \neq i} c_{ji} s_j), \qquad (2.57)$$

где $\mu_i$, $\rho_i$ – неопределенные множители Лагранжа.

Нахождение экстремума лагранжиана (2.57) сводится к решению системы алгебраических уравнений:

$$\left.\begin{aligned}
&\frac{\partial L_i(\cdot)}{\partial u_i} = L_{u_i} = \frac{\partial f_i(\cdot)}{\partial u_i} - \mu_i \frac{\partial \varphi_i(\cdot)}{\partial u_i} = 0 \\
&\frac{\partial L_i(\cdot)}{\partial z_i} = L_{z_i} = \frac{\partial f_i(\cdot)}{\partial z_i} - \mu_i \frac{\partial \varphi_i(\cdot)}{\partial z_i} + \rho_i = 0 \\
&\frac{\partial L_i(\cdot)}{\partial s_i} = L_{s_i} = \mu_i - \rho_i \sum_j c_{ji} \frac{\partial s_j}{\partial s_i} = 0 \\
&\frac{\partial L_i(\cdot)}{\partial \mu_i} = L_{\mu_i} = s_i - \varphi_i(\cdot) = 0 \\
&\frac{\partial L_i(\cdot)}{\partial \rho_i} = L_{\rho_i} = z_i - \sum_j c_{ji} s_j = 0
\end{aligned}\right\} \qquad (2.58)$$

При совместности уравнений системы (2.58) ее решение позволяет найти оптимальные значения переменных $u_i$, $z_i$, $s_i$, $\mu_i$, $\rho_i$ .

### *Решение задач координации по методу согласования взаимодействий путем модификации целей с нулевой суммой*

Пусть глобальная целевая функция задана в виде суммы локальных целевых функций подсистем. Так как конфликты в иерархических системах управления (ИСУ) могут возникнуть из-за несогласованного изменения связующих переменных $s_i$ отдельных подсистем, то для координации необходимо найти способ так изменять (модифицировать) локальные целевые функции (ЛЦ), чтобы это изменение приводило к изменению связующих переменных в нужном направлении. Глобальная целевая функция (ГЦ) при этом не должна изменяться, т.е. $\sum_{i=1}^{n} \Delta f_i(\cdot) = 0$, где $\Delta f_i(\cdot)$ - изменение ЛЦ i-ой подсистемы, производимое координирующим воздействием.

Для рассматриваемой задачи

$$F_{гц}(\bullet) = \text{ext}\ \{\sum_i f_i(u_i, z_i)\} \quad i = 1,2, \ldots, n$$

с ограничениями $S_i - \varphi_i(u_i, z_i) = 0$; $Z_i - \sum_{j \neq i} C_{ji} S_j = 0$ и полученного для нее

лагранжиана

$$L(u, z, \mu, \rho) = \left\{ \sum_{i=1}^{n} f_i(u_i, z_i) + \sum_{i=1}^{n} \mu_i (s_i - \varphi_i(u_i, z_i)) + \sum_{i=1}^{n} \rho_i (z_i - \sum_{j \neq i} c_{ji} s_j) \right\}$$

таким параметром может являться неопределенный множитель Лагранжа $\rho_i$. Этот выбор обусловлен тем, что $z_i$, наряду с $u_i$, является входной переменной для i-ой подсистемы.

В слагаемом $\rho_i (z_i - \sum_j c_{ji} s_j) = \rho_i z_i - \rho_i \sum_j c_{ji} s_j$ представим вторую его часть следующим образом:

$$\rho_i \sum_{j \neq i} c_{ji} s_j = (\sum_{j \neq i} \rho_j c_{ij}) s_i, \qquad (2.59)$$

где $c_{ij}$ - коэффициент влияния i-ой подсистемы на j-ую подсистему.

Выражение (2.59) означает, что i-я подсистема берет от других подсистем столько, сколько она через связующую переменную $s_i$ отдает другим подсистемам (условие не накапливания продукта внутри подсистемы) [4]. Математически такой прием делает задачу поиска экстремума лагранжиана каждой из подсистем сепарабельной.

Рассмотрим, как этот прием можно использовать в данном способе координации. Как было показано выше, ГЦ в форме лагранжиана можно представить в виде:

$$L_{гц}(\cdot)=\sum_{i=1}^{n}\left[f_i(u_i,z_i)+\mu_i(s_i-\varphi_i(u_i,z_i))+\rho_i z_i-\sum_j \rho_j c_{ij} s_i\right]=\sum_{i=1}^{n} L_i(u_i,z_i,\mu_i,\rho_i)$$

где $L_i(\cdot)$ - лагранжиан (целевая функция без ограничений) $i$-й подсистемы:

$$L_i(\cdot)=\left[f_i(u_i,z_i)+\mu_i(s_i-\varphi_i(u_i,z_i)+\rho_i z_i-\sum_j \rho_j c_{ij} s_i\right] \qquad (2.60)$$

В случае координации путем модификации локальных целевых функций с нулевой суммой, необходимо обеспечить выполнение условия $\sum_{i=1}^{n}\Delta f_i(\cdot)=0$ или более слабого - $\sum_{i=1}^{n}\Delta f_i(\cdot)\le\varepsilon$, где $\varepsilon$ - малое число. Очевидно, что если это условие не выполняется, то необходимо изменение локальных целевых функций $\Delta f_i(\cdot)$:

$$\Delta f_i(\cdot)=\rho_i z_i-\sum_j \rho_j c_{ij} s_i .$$

Из последнего выражения следует, что формирование координирующего воздействия возможно за счет изменения $\rho_i\ (i=\overline{1,n})$, так как оно приводит к изменению ЛЦ.

В локальных подсистемах для нахождения экстремума ЛЦ необходимо найти экстремум лагранжиана (2.60). Для этого необходимо решить следующую систему уравнений:

$$\left.\begin{aligned}
&\frac{\partial L_i(\cdot)}{\partial u_i}=L_{ui}=\frac{\partial f_i(\cdot)}{\partial u_i}-\mu\frac{\partial \varphi_i(\cdot)}{\partial u_i}=0\\
&\frac{\partial L_i(\cdot)}{\partial z_i}=L_{zi}=\frac{\partial f_i(\cdot)}{\partial z_i}-\frac{\partial \varphi_i(\cdot)}{\partial z_i}+\rho_i=0\\
&\frac{\partial L_i(\cdot)}{\partial s_i}=L_{si}=-\mu_i+\sum_j \rho_j c_{ij}=0\\
&\frac{\partial L_i(\cdot)}{\partial \mu_i}=L_{\mu i}=\varphi_i(u_i z_i)-s_i=0
\end{aligned}\right\} \qquad (2.61)$$

$$\frac{\partial L_i(\cdot)}{\partial \rho_i}=z_i=0 \qquad (2.62)$$

Условие (2.62) не может использоваться для поиска экстремума. Оно в такой постановке бессмысленно. Поэтому вернемся к другой форме вычисления $\rho_i$.

Как было показано ранее,

$$L_{\rho i}=\frac{\partial L_{гц}(\cdot)}{\partial \rho_i}=z_i-\sum_j c_{ji} s_j=0 . \qquad (2.63)$$

Но $L_{\rho i}(\cdot)$ является градиентом по $\rho_i$ глобальной целевой функции $L_{гц}(\cdot)$. Его можно использовать, применив метод наискорейшего спуска, для реализации

многошагового процесса поиска значений $\rho_i (i = \overline{1,n})$, обеспечивающих выполнение условия

$$\sum_{i=1}^{n} \Delta f_i(\cdot) = \left| \sum_{i=1}^{n} (z_i - \sum_j c_{ji} s_j) \rho_i \right| \le \varepsilon. \qquad (2.64)$$

При этом $\rho_i(k) = \rho_i(k-1) \pm \gamma L_{\rho i} = \rho_i(k-1) \pm \gamma \dfrac{\partial L_{гц}(\cdot)}{\partial \rho_i(k-1)}$,

где $\gamma$- коэффициент пропорциональности, а числа k и k–1 обозначают номера шагов вычисления нового значения $\rho_i$, обеспечивающего выполнение условия (2.64). В дальнейшем будем предполагать, что итерационный процесс вычисления $\rho_i(k)$ сходится.

### *Алгоритм одного цикла работы ИСУ при координации по методу согласования взаимодействий путем модификации целей.*

Исходные данные:
- целевые функции $f_i(z_i, u_i)$ и F(u,z);
- структура ИСУ и коэффициенты влияния $c_{ij}$ и $c_{ji}$;
- начальные значения $\rho_i$, $z_i$, $s_i$, $u_i$, (i =1…n);
- значения $\varepsilon$ и $\gamma$.

*Шаг 1.* На уровне подсистемы решается экстремальная задача Лагранжа:

$$\left.\begin{aligned} \frac{\partial L_i(\cdot)}{\partial u_i} &= 0 \\ \frac{\partial L_i(\cdot)}{\partial z_i} &= 0 \end{aligned}\right\}$$
$$\left.\begin{aligned} \frac{\partial L_i(\cdot)}{\partial \mu_i} &= 0 \\ \frac{\partial L_i(\cdot)}{\partial s_i} &= 0 \end{aligned}\right\}$$

и определяются оценочные значения $\hat{u}_i, \hat{z}_i, \hat{s}_i$.

*Шаг 2.* Значения $\hat{z}_i, \hat{s}_i$ направляются в координатор.

*Шаг 3.* На основании исходных данных и полученных оценок в $РО_2$ вычисляется $\sum_i \Delta f_i(\cdot) = \sum_i \rho_i (\hat{z}_i - \sum_j c_{ji} \hat{s}_j)$ или ее эквивалент (2.59).

*Шаг 4*. Если $\left|\sum_i \Delta f_i(\cdot)\right| > \varepsilon$, то для каждой подсистемы вычисляются значения $\Delta\rho_i(k) = \pm\gamma(z_i - \sum_j c_{ji} s_j)$ и $\rho_i(k) = \rho_i(k-1) + \Delta\rho_i(k)$ и переход к шагу 5, иначе переход к шагу 6.

*Шаг 5*. Передача вектора $\rho(k)$ в решающие органы подсистем $РО_{1i}$ и переход к шагу 1.

*Шаг 6*. Формирование значений $z_i^* = z_i$, $s_i^* = s_i$, $u_i^* = u_i$.

*Шаг 7*. Конец одного цикла координации.

Схему информационных связей в ИСУ, реализующую координацию по методу согласования взаимодействий путем модификации целей с нулевой суммой можно представить как показано на рис. 2.14.

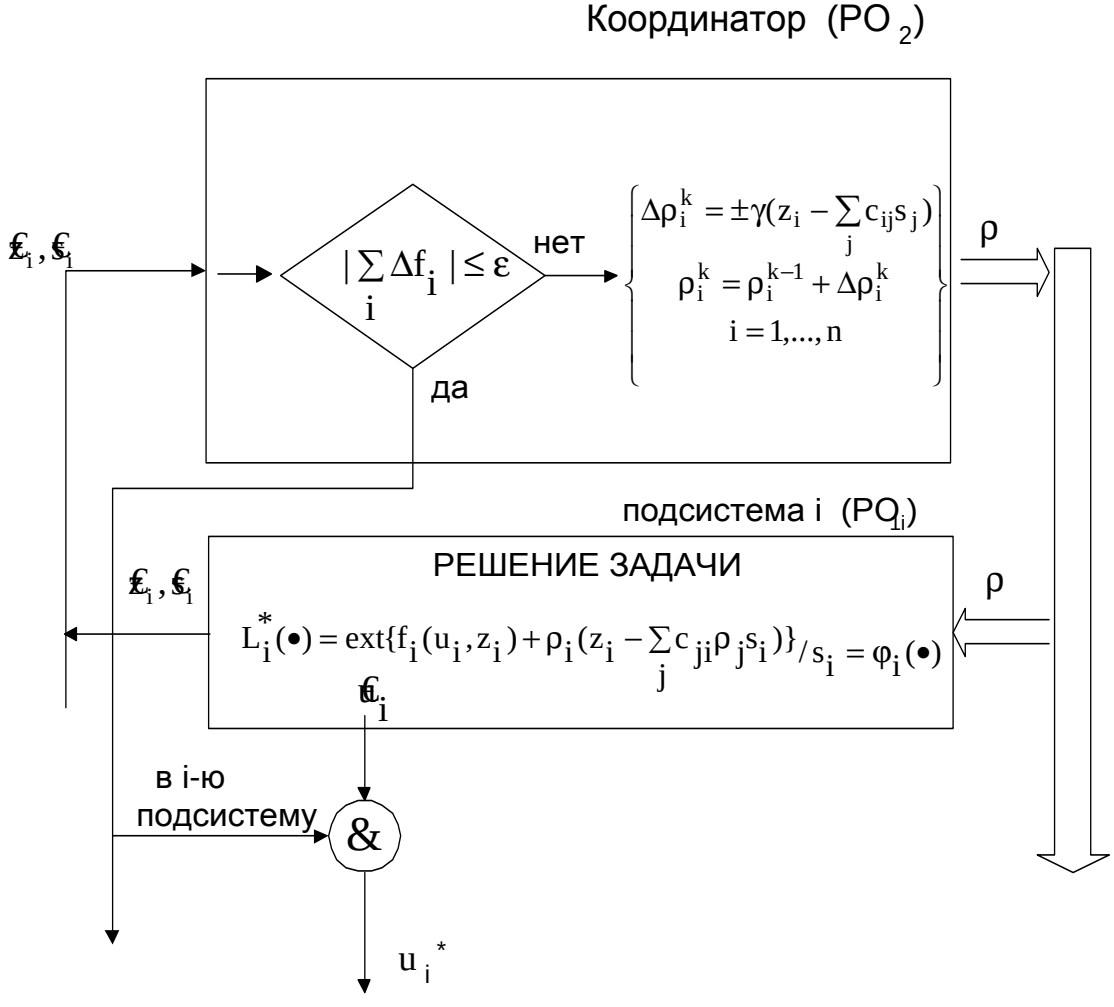



Рис 2.14

***Решение задачи координации по способу прогнозирования взаимодействий методом модификации образов (ограничений)***

Как указывалось ранее, при использовании координации по принципу прогнозирования взаимодействий реализуется идея вмешательства координатора в работу РО подсистем до принятия ими решений. Эту идею можно осуществить не модификацией целей, а модификацией образов.

При этом на верхнем уровне для каждой из подсистем определяются желательные для оптимизации глобальной целевой функции значения связующих переменных на входе ($z_i$) и на выходе ($s_i$) (Рис. 2.13). Значения $z_i$ и $s_i$ передаются на нижний уровень (в $РО_{1i}$). На нижнем уровне решается задача оптимизации локальной целевой функции с учетом заданных значений связующих переменных $s_i$.

Проиллюстрируем реализацию этого принципа координации на примере рассмотренной ранее ИСУ.

Как уже указывалось, для этой задачи глобальную целевую функцию можно записать в форме лагранжиана:

$$L(u,z,\mu,\rho)=\sum_i[f_i(u_i,z_i)+\mu_i(s_i-\varphi_i(u_i,z_i)+\rho_i(z_i-\sum_{j\neq i}C_{ji}s_j)]$$

В квадратных скобках записано выражение лагранжиана $L_i(u_i,z_i,\mu_i,\rho_i)$ для i-ой подсистемы:

$$L_i(u_i,z_{i,}\mu_{i,},)=f_i(u_i,z_i)+\mu_i(s_i-\varphi_i(u_i,z_i))+\rho_i(z_i-\sum_{j\neq i}C_{ji}s_j)$$

В результате условия оптимизации, определяемые на уровне подсистем, можно записать следующим образом:

$$\left.\begin{aligned}
&\frac{\partial L_i(\cdot)}{\partial u_i}=L_{u_i}=\frac{\partial f_i(\cdot)}{\partial u_i}-\mu_i\frac{\partial\varphi_i(\cdot)}{\partial u_i}=0\\
&\frac{\partial L_i(\cdot)}{\partial z_i}=L_{z_i}=\frac{\partial f_i(\cdot)}{\partial z_i}-\mu_i\frac{\partial\varphi_i(\cdot)}{\partial z_i}+\rho_i=0\\
&\frac{\partial L_i(\cdot)}{\partial \mu_i}=L_{\mu_i}=s_i-\varphi_i(\cdot)=0\\
&\frac{\partial L_i(\cdot)}{\partial \rho_i}=L_{\rho_i}=z_i-\sum_j C_{ji}s_j
\end{aligned}\right\} \qquad (2.65)$$

Отметим, что в (2.65) отсутствует уравнение $\frac{\partial L_i}{\partial s_i}=L_s$. Это обусловлено тем, что метод модификации образов предполагает задание из координатора в решающие органы подсистем значений $s_{iз}$=const. Поэтому, из системы (2.65) следует исключить уравнение $L_{si}=0$, так как оно не имеет смысла.

Следует отметить, что при $s_i$ = var, $\frac{\partial L_i(\cdot)}{\partial s_i}=L_{si}(\mu,\rho)$, т.е. градиент $L_s$, является функцией неопределенных множителей Лагранжа $\mu_i$ и $\rho_i$. Следовательно,

изменяя значения этих множителей, можно в координаторе воздействовать на изменение связующих переменных $s_i$.

Так как система (2.65) является не полной, на уровне подсистем можно получить лишь оценочные значения $\mu_i$ и $\varphi_i$. В координаторе, в связи с тем, что на этом уровне можно изменять связующие переменные, возможно определить оценочные значения градиента $L_{si}(\mu_i, \varphi_i)$. Для этого, как и при модификации целей, следует использовать итерационную процедуру вычисления градиента

$$L_{si}(k) = \frac{\partial L_{гц}(k)}{\partial s_i(k)} = \mu_i(k) - \sum_j \varphi_j c_{ij}.$$

В результате значения связующих переменных $s_i(k)$ на k - ом шаге итерации можно определить следующим образом:

$s_i(k) = s_i(k-1) + \Delta s_i(k)$, где $\Delta s_i(k) = \gamma L_{si}(k)$.

Значения $s_i(k)$ используются в следующем цикле итерационного процесса в качестве заданных значений связующих переменных $s_{iз}$ и передаются в решающие органы подсистем.

Схему информационных связей для ИСУ данного типа (аналогичную изображенной на рис. 2.14) предлагается построить самостоятельно. При этом следует учесть, что с координатора на подсистемы должны поступать сигналы $s_{iз}$, а с подсистем на координатор – $\mu_i, \varphi_i$.

***Алгоритм одного цикла работы ИСУ по методу прогнозирования взаимодействий путем модификации образов.***

Исходные данные:

- структура ИСУ и коэффициенты влияния $c_{ij}$ и $c_{ji}$;
- локальные функции подсистем $f_i(u_i, z_i)$ и ограничения:

$s_i = \varphi_i(u_i, z_i)$ и $z_i = \sum_{j \neq i} c_{ji} s_j$, $i = 1, \ldots, n$;

- глобальная целевая функция $F(u, z)$;
- точность вычисления $\varepsilon$.

*Шаг 1.* Определяются заданные значения $s_i = s_{iз}$, $i = \overline{1,n}$ для данного цикла вычисления.

*Шаг 2* Значения $s_{iз}$ передаются в $РО_{1i}$ всех подсистем.

*Шаг 3.* На уровне подсистем решается задача $\text{ext}\, f_i(u_i, z_i)$ при ограничениях $s_i = s_{iз}$ и $z_i = z_{iз} = \sum_j c_{ij} s_j$ и определяются значения $\mu_i, \varphi_i$.

*Шаг 4.* Значения $\mu_i(k), \varphi_i(k)$ передаются в координатор, где на их основе определяются $\Delta s_i(k)$ и $s_i(k)$.

*Шаг 5.* Если для какого-либо $\Delta s_i(k)$ не выполняется условие $|\Delta s_i(k) - \Delta s_i(k-1)| \le \varepsilon$, то $s_i(k) = s_i(k-1)$ и вычисляются новые значения $s_i(k)$, которые направляются в $РО_{1i}$ в качестве новых значений $s_{iз}$ и переход к шагу 3; иначе - к шагу 6.

*Шаг 6* Передача сигнала управления $u_i(k)$ в исполнительные органы подсистемы.

*Шаг 7* . Конец одного цикла координации.

*Примечание. Начальные значения* $s_{iз}$ *определяются исходя из стратегии решения задачи на верхнем уровне.*

### *Координация с использованием модификации целей и образов*

При координации с использованием модификации образов контролируется состояние связующих подсистемы переменных, т.е. обмен между подсистемами и непосредственно не регламентируется значение как локальных, так и глобальной целевых функций. Метод координации с использованием модификаций целей и образов призван устранить этот недостаток.

Пусть, как и в рассмотренных ранее способах, решающие органы подсистем реализуют управление в форме решения локальной оптимизационной задачи нахождения экстремума $\text{ext}\{f_i(u_i, z_i)\}$ при ограничениях на $s_i$ и $z_i$. На верхнем уровне решается глобальная оптимизационная задача в виде

$$F_{гц}(u, z) = \text{ext} \sum_{i=1}^{n} \{f_i(s_i, z_i) \,/\, s_i = \varphi_i(u_i, z_i);\ z_i = \sum_{j \ne i} C_{ji} s_j\}$$

Требуется, используя модификацию целей (с нулевой суммой) и образов, сформировать координирующее воздействие на локальные подсистемы так, чтобы контролировать как значения связующих переменных, так и значение глобальной целевой функции.

Пусть в координаторе получено требуемое значение связующих переменных $s_{iз}$ (вмешательство до принятия решения). Как было показано выше, при модификации целей в решающие органы подсистем необходимо направить (в качестве заданных координатором) значения неопределенных множителей $\rho_{iз}$, а при модификации образов - $s_{iз}$. В результате, при модификации целей и образов в решающих органах нижнего уровня можно варьировать значения только трех переменных – $\alpha_i, \beta_i, \gamma_i$ .

При переходе к поиску безусловного экстремума локальных целевых функций, задаваемых в форме лагранжиана $L_i(u_i, z_i, \mu_i)$, возможно получить лишь оценочные значения $\alpha_i, \beta_i, \gamma_i$, решив следующую систему уравнений:

$$\frac{\partial L_i(\cdot)}{\partial \alpha_i} = L_{ui} = \frac{\partial f_i(\cdot)}{\partial \alpha_i} - \frac{\partial \varphi_i(\cdot)}{\partial \alpha_i} = 0,$$

$$\frac{\partial L_i(\cdot)}{\partial x_i} = L_{zi} = \frac{\partial f_i(\cdot)}{\partial x_i} - \mu_i \frac{\partial \varphi_i(\cdot)}{\partial x_i} + \rho_i = 0, \qquad (2.66)$$

$$\frac{\partial L_i(\cdot)}{\partial \mu_i} = L_{\mu i} = s_{i.} - \varphi_i(x_i, x_i) = 0.$$

Значения градиентов $L_{si}$ и $L_{\rho i}$ определяются на верхнем уровне (в координаторе) путем организации многошагового итерационного процесса их вычисления, т.к. в координаторе, в отличие от нижнего уровня, они могут варьироваться.

$$\frac{\partial L(k)}{\partial s_i(k)} = L_{si}(k) = \mu_i(k) - \sum_{j \neq i} C_{ij}\mu_j(k-1)$$

$$\frac{\partial L(k)}{\partial \rho_i(k)} = L_{\rho i}(k) = x_i(k) - \sum_{j \neq i} C_{ji}x_i(k-1) \qquad (2.67)$$

После определения на к – ом шаге итерации значений градиентов (2.67), изменения значений $s_{iз}(k-1)$ и $\rho_{iз}(k-1)$ можно определить в виде:

$$\Delta s(k) = \gamma_s L_{si}(k) = \gamma_s(\mu_i(k) - \sum_{j \neq i} C_{ij}\mu_j(k-1))$$

$$\Delta\rho_i(k) = \gamma_{i\rho} L_{\rho i}(k) = \gamma_{\rho i}(x_i(k) - \sum_{j \neq i} C_{ji}x_i(k-1)) \qquad (2.68)$$

Задаваемые с координатора значения $s_{iз}$ и $\rho_{iз}$, с учетом (2.68), определяются следующим образом:

$$s_{iз}(k) = s_i(k-1) + \Delta s_i(k)$$

$$\rho_{iз}(k) = \rho_i(k-1) + \Delta\rho_i(k) \qquad (2.69)$$

Полученные значения $s_{iз}(k)$, $\rho_{iз}(k)$ передаются на нижний уровень для коррекции в очередном цикле итераций переменных $x_i, x_i$, и $\mu_i$.

На верхнем уровне также вычисляется значение

$$\sum_{i=1}^{n} \Delta f_i(\cdot) = \sum \mu_i(k)(x_i(k) - \sum_{j \neq l} C_{ji}x_j(k)).$$

Остановка итерационного процесса в очередном цикле модификации производится после получения условий: $L_{si} = 0, \quad L_{\rho i} = 0, \quad \sum_{i=1}^{n} \Delta f_i(\cdot) = 0$.

### *Укрупненный алгоритм одного цикла формирования координирующих воздействий с использованием модификации целей и образов*

Исходные данные:
- структура и параметры ИСУ;
- локальные и глобальная целевые функции;
- начальные значения переменных.

*Шаг 1.* Для m - го цикла координации на верхнем уровне определяются векторы желательных значений связующих переменных $S_3$, $\rho_3$.

*Шаг 2.* Значения $s_{i3}$ и $\rho_{i3}$ передаются в решающие органы всех подсистем.

*Шаг 3.* На уровне подсистем решаются задачи определения оценок $\beta_i$, $\varepsilon_i$, $\alpha_i$.

*Шаг 4.* Значения $\varepsilon_i$, $\beta_i$ передаются на верхний уровень, а значение $\alpha_i$ запоминается.

*Шаг 5.* В координаторе организуется итерационная процедура вычисления значений

$$\Delta S_i(k) = \gamma_{si}(\beta_i(k) - \sum_{j\neq i} C_{ij}\alpha_j(k-1)) \text{ и } \Delta\rho_i(k) = \gamma_{\rho i}(\varepsilon_i(k) - \sum_{j\neq i} C_{ji}s_j(k-1)).$$

*Шаг 6.* В координаторе определяются значения $\rho_i(k) = \rho_i(k-1) + \Delta\rho_i(k-1)$,

$$S_i(k) = S_i(k-1) + \Delta S_i(k) \text{ и } \Delta f_i(k) = \sum_{i=1}^{n}(\varepsilon_i(k) - \sum_{j\neq i} C_{ji}S_i(k)).$$

*Шаг 7.* Проверяется выполнение условия окончания процесса m-го цикла координации в виде анализа логического уравнения следующего вида:
$(\sum \Delta f_i(k)) \vee (\forall \Delta s_i(k)) = 0$, где $\forall$- квантор общности (для всех), $\vee$-логический оператор.

*Шаг 8.* Если условие предыдущего пункта не выполняется, решается задача поиска оптимального значения глобальной целевой функции на k-ом шаге итерационного процесса с целью определения желательных значений связующих переменных $s_3$ и $\rho_3$ для (к+1) цикла итерационного процесса и выполняется переход к шагу 2, иначе - к шагу 9.

*Шаг 9.* Процесс модификации заканчивается; разрешается передача на исполнительные органы оптимальных управляющих воздействий $u_i^*(k)$.

*Шаг 10* Конец одного цикла координации модификацией целей и образов.

*Примечание. Как показывают исследования* [4], *рассмотренные выше способы формирования координирующих управлений могут быть реализованы при условии достаточно широкого диапазона возможных изменений целевых функций подсистем.*

Как следует из приведенных выше примеров, задача формирования координирующих воздействий реализуется в виде итерационной процедуры с заранее неизвестным временем ее окончания. В ряде применений такое решение из-за неопределенности времени, затрачиваемого на завершение итераций, неприемлемо. В таких случаях необходимо применение других алгоритмов получения координирующего управления.

### 2.3.3. Вариант координации по принципу прогнозирования взаимодействий с использованием линейного программирования

Рассмотрим распределенную систему производства однородного продукта. Система состоит из ряда связанных между собой подсистем, имеющих соответствующие управляющие органы.

Необходимо организовать управление системой таким образом, чтобы решались задачи как отдельными подсистемами, так и общая глобальная задача, заключающаяся в том, что в случае дефицита в каких либо подсистемах им была бы оказана помощь другими недефицитными подсистемами. В случае, когда связи между подсистемами обладают ограниченной пропускной способностью, эта помощь, с одной стороны, должна быть организована наиболее экономным способом, а с другой – не создавать перегрузку связей между подсистемами. Подобные постановки задач характерны, например, для электроэнергетических объединений [12].

Структура такой системы в двухуровневом упрощенном варианте изображена на рис. 2.15. Внутренняя структура подсистемы может быть представлена источниками производства и потребления продукта, связанными между собой внутрисистемными линиями (связями), также обладающими ограниченными пропускными способностями.

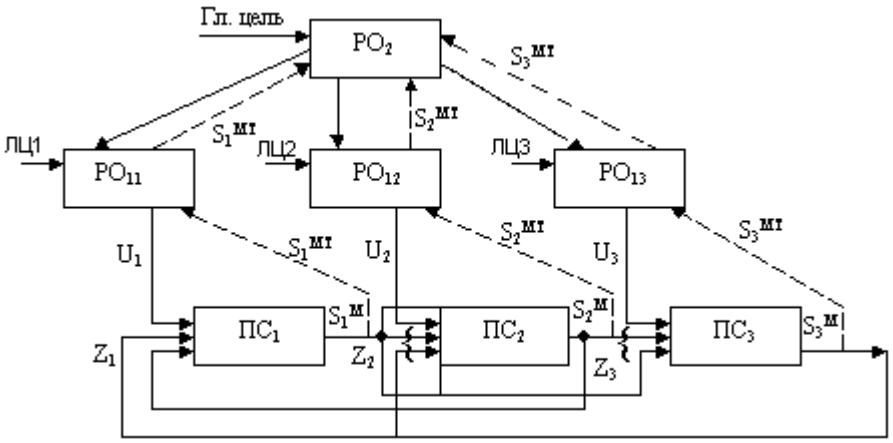



Рис. 2.15

Для i-ой подсистемы введем следующие обозначения:

$s_j^{Bi}$ - значения связующих переменных по $j$- ой связи;

$u_k^{Bi}$ - управление, подаваемое на k-ый источник продукции с $PO_{1i}$;

$u_{пр}^{i}$ - прогнозируемое значение агрегированного управления в $i$- ой подсистеме, необходимое для ликвидации перегрузки по связи $s_i^M$;

$z_i = \sum_j s_j^M$ - воздействие на $i$- ую подсистему по связям от других подсистем.

Задачу верхнего уровня (координатора $PO_2$) можно сформулировать следующим образом. Найти $u_i^{пр} = \arg\min \sum_{i=1}^{3} c_i \left| u_i^{пр} \right|$ при ограничениях

$$\Delta s_j^M = \sum_{i=1}^{3} \alpha_{ij}^M u_i^{пр} \le \bar{s}_j^M ,$$

где $u_i^{пр}$ - прогнозируемое изменение управления в $i$-ой подсистеме, обеспечивающее ликвидацию перегрузки по связям $i$-ой подсистемы с другими подсистемами;

$\alpha_{ij}^M$ - коэффициенты влияния (чувствительности) изменения управления в $i$-ой подсистеме на связующие переменные $s_i$;

$\Delta s_j^M$ - изменение связующей переменной $s_j^M$, обеспечивающее ликвидацию возникшей перегрузки $\Delta \bar{s}_j^M$.

На нижнем уровне в описанной ситуации $РО_{1i}$ решает следующую задачу.

Найти $u_k^i = \arg\min \sum_{k=1}^{mi} c_k^i \left| u_k^i \right|$, при ограничениях $\sum_{k=1}^{mi} u_k^i = u_i^{пр}$,

$$\Delta s_j^{Bi} = \sum_{k=1}^{mi} \alpha_{kj}^{Bi} (u_k^i + \xi_k^i z_i) \le \Delta \bar{s}_j^{Bi}$$

где $u_k^i$ - изменение управления на внутреннем $k$-ом источнике, обеспечивающее неперегрузку $j$-ой внутренней связи $s_j^{Bi}$;

$\xi_k^i$ - коэффициент, учитывающий влияние изменения связующих переменных $z_i$ на $j$-ую связь внутри $i$-ой подсистемы, приведенное к изменению управления $k$-го внутреннего источника;

$\alpha_{kj}^{Bi}$ - коэффициент влияния изменения управления на каждом внутреннем источнике $i$-ой подсистемы на $j$-ую внутреннюю связующую переменную;

$\Delta z_i$ - изменение связующих переменных на входе $i$-ой подсистемы при изменении управления в других подсистемах.

Поставленная задача как на нижнем, так и на верхнем уровне может быть сведена к задаче линейного программирования [12, 13].

При этом прогнозированные на верхнем уровне значения $u_{пр}^i$ и $\Delta s_j^M$ передаются на нижний уровень в качестве директивных значений и должны обязательно учитываться при принятии решений $РО_1$ нижнего уровня.

### 2.3.4. Последовательная (улучшающая) координация

Всякая координация имеет целью улучшение работы системы. Например, задачу координации можно поставить следующим образом. Имеется сложная система, состоящая из нескольких подсистем, связанных друг с другом через связующие переменные. Каждая подсистема управляется в соответствии со своими заранее определенными целями. Можно ли улучшить характеристики системы, вводя координирующее управление? Опыт показывает, что можно. Причем в такой постановке координирующее управление значительно проще реализовать, так как не требуется получать оптимальное управление. Улучшения характеристик системы можно достигнуть как в одном цикле координации, так и

последовательно улучшая их во времени. Формализацию процесса улучшающей координации в обобщенном виде можно представить следующим образом.

В [4] показано, что в самом общем случае связующие переменные $s_i$ являются функциями общего управления ($s_i = \Psi_i(U)$, где $U = (u_1, u_2, \ldots, u_n)^T$), рассматриваемого на некотором пространстве ограничений.

Исходя из описанных ранее проблем, оценочная функция эффективности системы иерархического не оптимального управления также должна быть функцией общего управления. Пусть, например, функция оценки эффективности ИСУ будет иметь смысл уменьшения потерь - $G(U)$. В этом случае эффективным будет считаться такое управление, которое уменьшает $G(U)$ и находится в допустимой области его изменения. Математически такую задачу в общем виде можно записать как задачу оценки значения функции $G(U)$ при ограничениях $AU \le B$, где данное нестрогое неравенство учитывает ограничения на возможный диапазон изменений управления.

Пусть, как и в рассмотренных выше ИСУ, оценочная функция состояния системы является аддитивной функцией потерь ее подсистем, т.е.

$$G(U) = \sum_{i=1}^{n} \{ g_i(U) \diagup \alpha_j^T U \le b_j \}, \qquad (2.70)$$

где $b_{j-}$ – j-я строка системы ограничений,

$g_j(U)$ – оценочная функция потерь i – ой подсистемы,

$\alpha_j^T$ – j-я строка матрицы A.

При принятой оценке эффективности системы условие улучшения ее состояния на к + 1-м шаге по сравнению с к - м шагом можно записать в виде

$$G(U(k+1)) < G(U(k)) \qquad (2.71)$$

Если условие (2.71) не выполняется, то управление на к+1 шаге не может считаться более эффективным по сравнению с предыдущим и его лучше не производить.

Для улучшения эффективности системы под воздействием управления необходимо:

- определить направление возможного изменения состояния системы,
- решить, насколько следует изменить это состояние по отношению к текущему.

Определения.

1. Направление изменения состояния системы под действием управления считается *возможным*, если сделанный в этом направлении шаг не выводит систему за границу допустимой области,
2. Направление будем называть *приемлемым*, если выбранное направление приводит к выполнению условия (2.71).

Для решения задачи определения возможного и приемлемого изменения состояния системы наибольшее распространение получают методы, известные из теории математического программирования как методы *возможного направления*, модифицируемые для задач оценки эффективности.

Пусть для рассматриваемой задачи перевода системы из одного состояния в другое, управление строится в виде

$$U(k+1) = U(k) + dr(k)$$

где $U(k)$ - “старое” управление;

r(k)-единичный вектор, определяющий направление перехода к новому состоянию;

d – длина шага в выбранном направлении.

Возможное направление можно оценить, например, исходя из условия выполнения ограничения $\alpha_j^T U(k+1) \le b_j$ или

$$\alpha_j^T (U(k) + dr(k)) = \alpha_j^T U(k) + \alpha_j^T dr(k) \le b_j \qquad (2.72)$$

Пусть в (2.72) $\alpha_j^T U(k) = b_j$. Для того, чтобы в этой ситуации выбранное направление было возможным, необходимо выполнить условие $\alpha_j^T dr(k) < 0$.

Наиболее логично выбрать направление изменения состояния, определяемое антиградиентом в исходной точке, т.е.

$$dr(k) = -\gamma \nabla G(U(k)) \qquad (2.73)$$

Потребуем, чтобы с координатора в решающие органы подсистем передавалась информация $\beta_i(k)$, модифицирующая оценочную функцию потерь.

$$g_I(u_i(k+1)) = g_i(u_i(k), \beta_i(k)), \qquad (2.74)$$

где $\beta_i$ - модифицирующий параметр.

Выражение (2.74) означает, что для получения нового значения оценочной функции можно воспользоваться управлением $u_i(k)$ и модифицирующим параметром $\beta_i(k)$. Например, в случае линейной модификации, $\beta_i$ имеет смысл коэффициента пропорциональности $\Delta u_i(k) = \beta_i(k) u_i(k)$. Пусть функция потерь i-ой подсистемы имеет следующий вид: $g_i(u_i(k+1)) = u_i^2(k) + \Delta u_i(k)$.

Учитывая (2.72), получим $\Delta u_i(k) = C_j^T dr(k) = C_j^T(-\gamma_i \nabla G(U(k))) = \beta_i(k) u_i(k)$.

Отсюда следует

$$\beta_i(k) = \frac{\Delta u_i(k)}{u_i(k)} = -\frac{C_j^T(\gamma_{ii} \nabla G(U(k)))}{u_i(k)}.$$

Для оценки эффективности выбранного направления и длины шага следует проверить выполнение условия $G(U(k+1)) < G(U(k))$. Если это условие не выполняется, то изменения состояния не производится ($\beta_i(k) = 0$).

Приведенный пример улучшающей координации – это один из возможных вариантов ее организации. С другими можно ознакомиться, например, в [4].

Рассмотренные выше примеры реализации координирующего управления не охватывают всех возможных способов его организации. Они имеют цель познакомить с одним из наиболее перспективных направлений компьютерного управления в ИСУ - оптимального координирующего управления и его разновидностей. Реализация такого управления практически невозможна без использования вычислительной техники, так как требует обработки большого объема информации в ограниченное время.

# 3. НЕЧЕТКОЕ УПРАВЛЕНИЕ

В рассматривавшихся ранее системах управления формирование управляющих воздействий производится на основе анализа объективных, строгих количественных данных о состоянии объекта и заданной цели управления. Однако в некоторых случаях получить информацию с достаточной строгостью невозможно – она изначально является нечеткой. Причины нечеткости могут быть различными. Это и объективные условия функционирования сложных, "плохо определенных" систем, и субъективные оценки операторов и, наконец, то, что называют “проклятием размерности”, т.е. сложность системы, не позволяющая создать удовлетворительную модель системы.

Решение задач обработки информации и управления в таких условиях стало возможным при использовании принципов и техники компьютерного управления. Например, алгоритмы нечеткого управления, как правило, строятся с использованием экспертных оценок или накопленного опыта, реализуемого в форме специализированных баз знаний. Для их хранения и оперативного использования необходима организация в составе управляющего устройства запоминающих блоков с прямым доступом. Достаточно просто эта задача решается средствами компьютерной техники. Поэтому в нечетких системах обработки информации и управления находят применение как узкоспециализированные “нечеткие” компьютеры, так и компьютеры общего назначения [8].

Синтез управляющих устройств на базе компьютерной техники позволяет решать не только задачи хранения базы знаний, но также существенно расширить области применения нечетких систем за счет возможности реализовывать более сложные алгоритмы обработки и “четкой” информации.

Остановимся более подробно на таких причинах появления нечеткости как субъективные оценки операторов. Они во многих случаях наиболее ясно иллюстрируют причины появления нечеткости в управлении.

Выполняя задачи управления и контроля различными машинами и технологическими процессами, человек часто использует субъективные алгоритмы. Например, управляя автомобилем при маневрировании в ограниченном пространстве, водитель использует такую информацию, как “немного правее (левее)”, “чуть-чуть ближе”, “можно ближе”.

Очевидно, что эти понятия и масса им подобных не отличаются четкостью. Тем не менее, несмотря на это, водитель обычно успешно заканчивает маневрирование. Понятия типа “немного”, “левее” и тому подобные являются субъективными оценками. Они отражают особенности качественного, а не количественного восприятия событий человеком и выражаются в словесной форме.

При решении, например, задач автоматизации управления оценка явлений и процессов на лингвистическом уровне требует разработки и лингвистических

методов обработки и анализа информации. Обычная двузначная математическая логика для этих целей не подходит, так как она не способна отражать нечеткие формулировки и явления. Для решения возникающих проблем американский математик Л. Заде в 1965 году предложил новый математический аппарат нечетких множеств и нечеткой логики. Появление этого аппарата, обеспечивающего возможность формализации и обработки лингвистических данных с целью получения соответствующих выводов, позволило приступить к созданию систем управления, обладающих искусственным интеллектом.

Как и в природных системах, в системах с искусственным интеллектом существенное расширение их возможностей может быть достигнуто использованием предшествующего опыта, сохраняемого в базах знаний. Для технических систем последнее возможно лишь при использовании компьютеров в качестве устройств обработки информации.

Таким образом, нечеткие системы управления изначально стали развиваться как системы компьютерного управления. В простейших случаях нечеткое управление можно реализовать нечеткими контроллерами [8], а также с помощью контроллеров общего применения.

## 3.1. Организация нечеткого управления

Для лучшего понимания проблем нечеткого управления и методов их решения средствами нечеткой логики в первую очередь рассмотрим два типа возможных структур систем нечеткого управления.

На рис. 3.1 приведена структура системы ручного (неавтоматического) нечеткого управления объектом. В этой системе оператор, анализируя состояние объекта, формирует качественную оценку процесса. На основании этой оценки в дальнейшем производится преобразование ее в некоторый количественный (числовой) эквивалент. Полученные числовые данные с учетом цели управления и имеющейся базы знаний соответствующим алгоритмом нечеткого (лингвистического) управления преобразуются во множество возможных значений управляющего воздействия.

Так как исполнительный орган системы управления должен реагировать на конкретные значения управляющего сигнала, то в следующем функциональном блоке множество возможных значений преобразуется в конкретное (четкое) значение управляющего сигнала u.

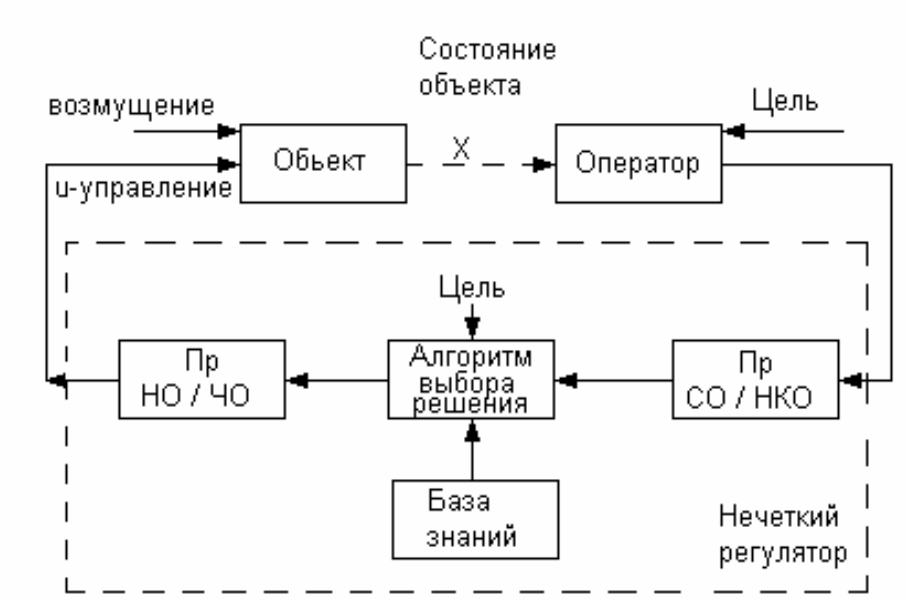



Рис. 3.1. Структура системы ручного нечеткого управления

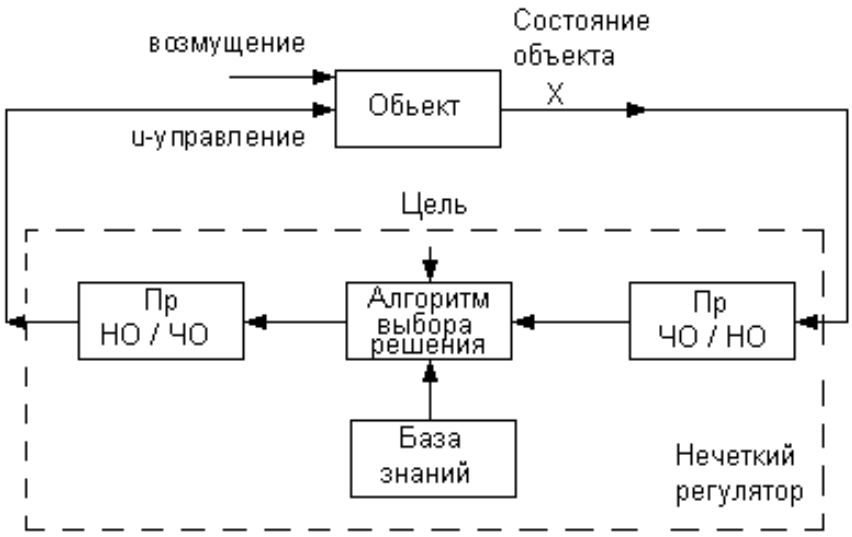
Пр СО/НКО – преобразование субъективной оценки в нечеткую качественную оценку; Пр НО/ЧО – преобразование нечетких оценок в четкие сигналы управления

На рис. 3.2 приведена структура системы автоматического нечеткого управления объектом. Особенностью этой структуры по сравнению с предыдущей является то, что координаты состояния объекта измеряются конкретными датчиками и являются вполне четкими значениями.

Не рассматривая пока причины появления нечеткости в данной системе управления, отметим, что в ней должно быть предусмотрено преобразование четких (измеренных) значений в нечеткие. Остальные функциональные блоки данной системы аналогичны ранее рассмотренным.

Операцию преобразования четких значений в нечеткие иногда называют фаззификацией (от слова fuzzy – нечеткий, размытый, неопределенный), а обратное преобразование - дефаззификацией.

В заключение следует сказать, что структура как первой, так и второй системы в некоторой (чисто внешней) форме похожа на обычную (алгоритмическую) систему управления. Как будет показано ниже, “начинка” функциональных блоков и алгоритмы обработки существенно отличаются. Эти отличия носят концептуальный характер и основываются на представлении реального мира в форме “размытых” процессов.




Пр ЧО/НО – преобразование четкой оценки в нечеткую

Рис. 3.2. Структура автоматической системы нечеткого управления.

## 3.2 Основные понятия и определения теории нечетких множеств

Теория нечетких множеств базируется на теории обычных (четких) множеств и является ее расширением. Нечеткие множества (как и четкие) могут характеризоваться мощностью и характеристической функцией. Напомним, что характеристическая функция $\chi_A(x)$ в полном пространстве X равна 1, если $x \in A$ и равна 0, если $x \notin A$. Таким образом, наличие или отсутствие заданного свойства A у элемента x характеризуется числами 1 или 0.

В реальном мире довольно часто встречаются события, явления и процессы, принадлежность которых к тому или иному классу определить чрезвычайно сложно, так как разделяющие их границы обычно размыты.

В 1965г. Л. Заде предложил расширить двузначную оценку на многозначную, причем неограниченную, в интервале [0, 1]. Для такого типа переменных Л. Заде ввел специальный термин Fuzzy Set – нечеткое множество (размытое множество). Таким образом, нечеткое множество – это расширение двузначной оценки 0,1 до неограниченной многозначной оценки (больше 0 и меньше 1) на множестве действительных чисел [0, 1].

Вместо характеристической функции Л. Заде предложил понятие функции принадлежности. Для ее представления используются обозначения $\mu_A(x)$ или $m_A(x)$. Запись вида $\mu_A : X \to [0, 1]$ означает задание функции принадлежности на замкнутом множестве действительных чисел [0, 1].

Так как функция принадлежности является одним из фундаментальных понятий нечетких множеств, остановимся на нем более подробно.

Пусть на полном множестве X={x} рассматривается подмножество A(x). Рассмотрим случай, когда субъективная оценка принадлежности x к подмножеству A определена функцией принадлежности $\mu_A(x) = 0,7$. Последнее означает, что тот, кто давал оценку, лишь на 70% уверен, что данное значение x можно отнести к подмножеству A.

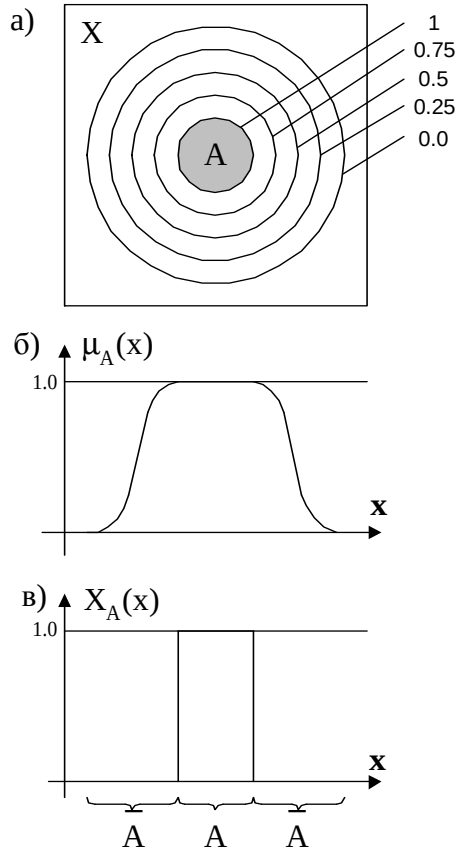



Рис. 3.3

Этот пример можно иллюстрировать диаграммой Виенна и графиком функции принадлежности $\mu_A(x)$ (рис. 3.3а, б). На диаграмме окружности изображают равные значения оценки принадлежности объектов x, обладающих свойством A. Ниже для такого же случая, но при четких оценках, приведен график, изображающий характеристическую функцию $\chi_A(x)$ (рис. 3.3в).

Из графиков $\mu_A(x)$ и $\chi_A(x)$ следует, что «четкие» множества являются частным (предельным) случаем нечетких множеств.

Таким образом, любое нечеткое множество полностью определяется функциями принадлежности, которые количественно характеризуют степень принадлежности того или иного элемента множества к рассматриваемому признаку.

**Операции над нечеткими множествами**

Операции над нечеткими множествами – это в первую очередь, операции над функциями принадлежности, так как только они являются числовыми

эквивалентами, отражающими степень принадлежности к тому или иному нечеткому множеству.

Операции над нечеткими множествами аналогичны операциям над четкими множествами.

1. *Операция вложения* – $\subset$ . (множество $A \subset B$)

$\mu_A(x) \le \mu_B(x)$, для $\forall \rangle \in X$

2. *Дополнение*

$\mu_{\overline{A}}(x) = 1 - \mu_A(x)$, для $\forall x \in X$

3. *Произведение (пересечение) нечетких множеств*

$\mu_{A \cap B}(x) = \mu_A(x) \bigcap \mu_B(x)$, для $\forall\, x \in X$

4. *Сумма (объединение) нечетких множеств*

$\mu_{A \cup B}(x) = \mu_A(x) \bigcup \mu_B(x)$, для $\forall x \in X$

Правила (законы) преобразования нечетких множеств аналогичны соответствующим правилам преобразования четких множеств. Для нечетких множеств не выполняется лишь закон комплементарности, т.е.

$$A(x) \bigcap \overline{A}(x) \neq \varnothing, \quad A(x) \bigcup \overline{A}(x) \neq X \text{ для } \forall A(x) \in X, \quad \overline{A}(x) \in X.$$

Для нечетких множеств можно записать

$$A(x) \bigcap \overline{A}(x) \supset \varnothing \text{ и } A(x) \bigcup \overline{A}(x) \subset X.$$

Т.е. пересечение нечетких множеств A и $\overline{A}$ не равны пустому множеству. Пустое множество вложено в пересечение множеств A(x) и $\overline{A}(x)$.

Аналогично объединение нечетких множеств A(x) и $\overline{A}(x)$ не образует полное множество X, а является вложенным в него, т.е. является его подмножеством.

Эти утверждения иллюстрируются графиками изменениями функций принадлежности на рис. 3.4.

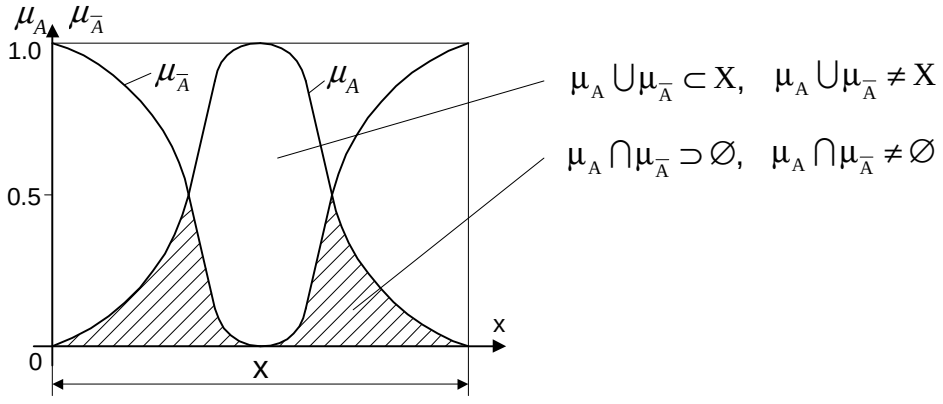



Рис. 3.4

Невыполнение закона комплементарности приводит для операций с нечеткими множествами к появлению "незамкнутой" алгебры. При этом оказывается возможным, кроме операций $\subset$, $-$, $\bigcap$, $\bigcup$ , ввести неограниченное число и других операций над нечеткими множествами.

Примерами таких операций могут быть следующие:

1. *Степень нечеткого множества* – $\alpha$

$$\mu_{A^\alpha}(x) = \{\mu_A(x)\}^\alpha \text{ для } \forall x \in X.$$

Графики для наиболее часто употребляемых степеней $\alpha$ = 0,5; 1; 2 показаны на рис. 3.5, из которого следует, что степень $\alpha > 1$ сужает диапазон изменения x, т.е. уменьшает "размытость" границ множества. Наоборот $\alpha < 1$ увеличивает размытость границ принадлежности элементов множества конкретному подмножеству.

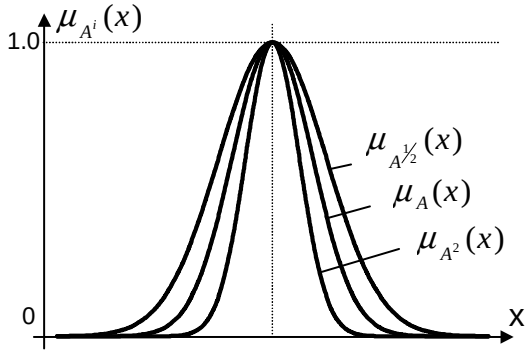



Рис. 3.5

2. *Операция пересечения нечетких множеств* (операция произведения или взятия минимума) может быть расширена введением следующих модификаций:

а) операция алгебраического произведения множеств – A•B

$$\mu_{A\bullet B}(x) = \mu_A(x) \cdot \mu_B(x) \text{ для } \forall x \in X.$$

б) операция граничного произведения множеств – A⊙B

$$\mu_{A\bullet B}(x) = (\mu_A(x) + \mu_B(x) - 1) \vee 0, \text{ для } \forall x \in X$$

в) операция драстического произведения множеств – AΔB

$$\mu_{A\Delta B}(x) = \begin{cases} \mu_A(x), & \text{если } \mu_B(x)=1 \\ \mu_B(x), & \text{если } \mu_A(x)=1 \\ 0, & \text{в других случаях для } \forall x \in X. \end{cases}$$

3. *Операция объединения нечетких множеств* (взятие максимума) может быть расширена введением следующих модификаций:

а) операция алгебраического суммирования нечетких множеств – A+B

$$\mu_{A+B}(x) = \mu_A(x) + \mu_B(x) - \mu_A(x)\mu_B(x), \text{ для } \forall x \in X$$

б) операция получения граничной суммы – A⊕B

$$\mu_{A\oplus B}(x) = (\mu_A(x) + \mu_B(x)) \wedge 1, \text{ для } \forall x \in X.$$

в) операция получения драстической суммы – A∇B

$$\mu_{A\nabla B}(x) = \begin{cases} \mu_A(x), & \text{если } \mu_B(x) = 0, \\ \mu_B(x), & \text{если } \mu_A(x) = 0, \\ 1, & \text{в других случаях для } \forall x \in X. \end{cases}$$

4. *Разность нечетких множеств* – A-B

$\mu_{A-B}(x) = (\mu_A(x) - \mu_B(x)) \vee 0$ для $\forall x \in X$.

5. *Абсолютная разность нечетких множеств* – |A-B|

$\mu_{|A-B|} = |\mu_A(x) - \mu_B(x)|$.

Как отмечалось выше, возможны и другие модификации операций над нечеткими множествами.

## 3.3. Нечеткая логика

В обычной (двузначной) логике операции над множествами и их элементами реализуются на основе некоторого набора базисных (элементарных) логических функций. Эти базисные функции (одного и двух аргументов) используются для синтеза сложных алгоритмов обработки информации.

Как известно, булева алгебра строится на использовании базиса трех элементарных функций (инверсии, конъюнкции и дизъюнкции). Базис булевой алгебры, обладая функциональной полнотой, в то же время является неминимальным. В нечеткой логике такую систему из ограниченного числа базисных функций создать невозможно, так как в нечеткой логике аргументы могут на интервале [0, 1] принимать бесконечно большое число различных значений. Можно предполагать, что законы двузначной логики являются предельным случаем логики нечеткой, так как на концах интервала [0, 1] элементы нечеткой и двузначной логик совпадают.

### Расширение операций булевой алгебры на область нечетких аргументов

В отличие от обычной булевой алгебры базисные операции нечеткой логики называются следующим образом.

1. Операция инверсии ("НЕ") – нечеткое отрицание обозначается $\Theta$.

2. Операция конъюнкции ("И") – называется t-нормой (триан-гуляционной нормой) и обозначается T.

3. Операция дизъюнкции ("ИЛИ") называется s-нормой и обозначается S

Операции нечеткой логики, в отличие от обычной логики, невозможно описать с помощью таблиц истинности, так как аргументы в нечеткой логике могут принимать множество различных значений. Поэтому в дальнейшем они будут поясняться с помощью специальных аксиом, а затем образы этих операций будут иллюстрироваться графически.

1. Нечеткое отрицание (инверсия).

Нечеткое отрицание, как и свой прототип, представляет унарную операцию отрицания, дающую в ответе нечеткую оценку в интервале [0, 1] .

*Аксиомы нечеткого отрицания*

A1. $0^{\Theta} = 1$ – сохранение свойства двузначного "НЕ".

A2. $(x^{\Theta})^{\Theta} = x$ для $\forall x \in [0,1]$ – правило двойного отрицания.

А3. $x_1<x_2$ , $x_1^{\Theta} > x_2^{\Theta}$ – нечеткое отрицание выполняет инверсию в смысле строгого неравенства количественных оценок. Для качественной оценки – меняет "хорошие" и "плохие" оценки местами.

Нечеткое отрицание может быть интерпретировано операцией $x^{\Theta} = 1 - x$, для $\forall x \in X, x \in [0,1]$.

Легко показать, что приведенные выше аксиомы соблюдаются. Например, А3:

если $x_1 < x_2$, то $x_1^{\Theta} = 1 - x_1 > 1 - x_2 = x_2^{\Theta}$, для $X \in [0,1]$, причем $1 - x = x^{\Theta}$ – строго и монотонно убывающая функция.

2. Нечеткое произведение (t - норма)

Нечеткое произведение является бинарной операцией $x_1 \in X$, $x_2 \in X$, определенной на интервале [0, 1] в форме $x_1 T x_2$.

*Аксиомы t - нормы*

Т1. $xT1 = x$; $xT0 = 0$, для $\forall x \in [0, 1]$ – сохраняет свойства двузначного «И» и справедливо в форме граничных условий.

Т2. $x_1 T x_2 = x_2 T x_1$ – свойство коммутативности, для $\forall x \in [0,1]$.

Т3. $x_1 T (x_2 T x_3) = (x_1 T x_2) T x_3$ – свойство дистрибутивности для $\forall x \in [0,1]$.

Т4. $x_1 \le x_2$, то $x_1 T x_3 \le x_2 T x_3$ – свойство упорядоченности, утверждающее, что введение третьей оценки при известных отношениях первых двух не изменяет порядок оценок при объединении каждой из оценок с новой оценкой x.

Графическая интерпретация *нечеткого логического произведения* (конъюнкции) $x_1 T x_2 = x_1 \wedge x_2$ для $\forall x \in X \in [0, 1]$ показана на рис. 3.6.

В отличие от двузначной логики, для которой результаты операции $x_1 \wedge x_2$ определены вершинами единичного куба, в нечеткой логике области определения t-нормы $x_1 T x_2 = x_1 \wedge x_2$ образованы равнобедренными треугольниками с общей вершиной А (на рис. 3.6 заштрихованы). Другим отличием t-нормы нечеткой логики от аналогичной операции конъюнкции $x_1 \wedge x_2$ булевой алгебры является то, что она трактуется как операция взятия min**,** т.е. определения минимального числового значения среди двух аргументов $x_1$ и $x_2$. Например, если $x_1 = 0{,}1$ и $x_2 = 0{,}1$, то операция $x_1 T x_2 = x_1 \wedge x_2 = 0{,}1 \wedge 0{,}1 = 0{,}1$, а не 0,01.

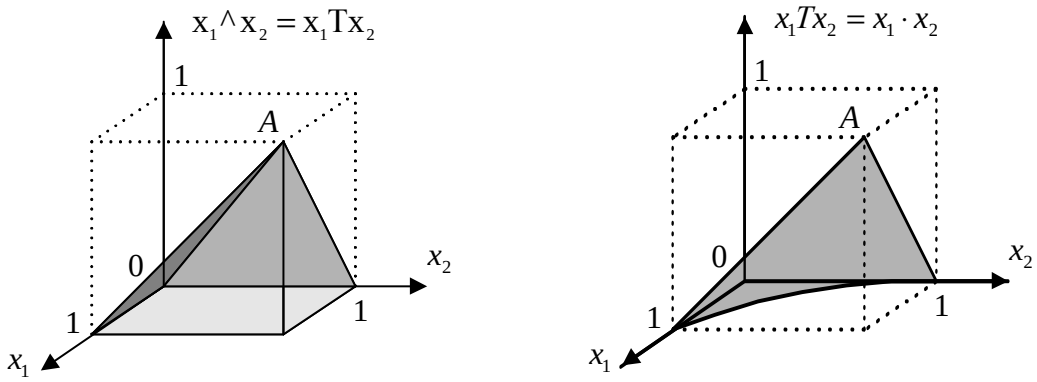


Рис. 3.6

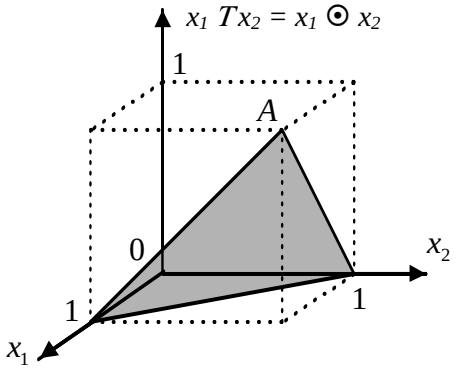



Рис. 3.7

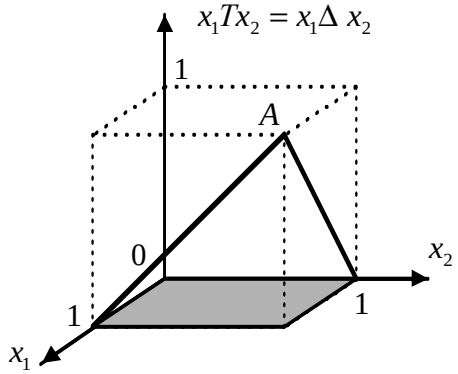



Рис. 3.8

Рис. 3.9

Графическая интерпретация *алгебраического произведения* $x_1Tx_2 = x_1 \bullet x_2$ – это криволинейная плоскость с вершиной в точке А (рис. 3.7).

Например:

а) при $x_1 = 0{,}1$; $x_2 = 0{,}5$; $x_1 \bullet x_2 = 0{,}05$

б) при $x_1 = 1$; $x_2 = 1$; $x_1 \bullet x_2 = 1$ (точка А)

в) при $x_1 = 0{,}5$; $x_2 = 1$; $x_1 \bullet x_2 = 0{,}5$.

График t-нормы в форме *граничного произведения* показан на рис. 3.8.

$x_1Tx_2 = x_1 \bullet x_2 = (x_1 + x_2 - 1) \vee 0$ – это треугольник $x_1 = x_2 = 1$ с вершиной А и треугольник 1,0,1 для $x_1Tx_2 = 0$.

Рассмотрим некоторые примеры.

А: а) при $x_1 = 1$, $x_2 = 0$ $x_1Tx_2 = 0$;

б) при $x_1 = 0{,}5$, $x_2 = 0{,}5$ $x_1Tx_2 = 0$;

в) при $x_1 = 0$, $x_2 = 1$ $x_1Tx_2 = 0$.

Это линия $x_1 = 1$ и $x_2 = 1$ на рис. 3.8.

В: г) при $x_1 = 0{,}5$, $x_2 = 1$ $x_1Tx_2 = 0{,}5$;

д) при $x_1 = 1$, $x_2 = 1$ $x_1Tx_2 = 1$.

Это линия А и $x_2 = 1$ на рис. 3.8

С: е) при $x_1 = 0{,}4$, $x_2 = 0{,}3$ $x_1Tx_2 = 0(-0{,}3) \vee 0 = 0$;

ж) при $x_1 = 0{,}1$, $x_2 = 0{,}1$ $x_1Tx_2 = 0(-0{,}8) \vee 0 = 0$.

Последний результат обусловлен тем, что операция $\vee$ – взятие max.

График t-нормы в форме *драстического произведения* показан на рис. 3.9.

$$x_1Tx_2 = x_1 \Delta x_2 = \begin{cases} x_2 & \text{при } x_1 = 1, \\ x_1 & \text{при } x_2 = 1, \\ 0 & \text{в других случаях.} \end{cases}$$

Графическая интерпретация драстического произведения – это заштрихованный прямоугольник и наклонные линии $(1-A)\big/_{x_1} = \text{const} = 1$ и $(1-A)\big/_{x_2} = \text{const} = 1.$

Сравнивая области определений различных способов расширения операций нечеткого произведения, можно записать следующее условие [8]:

$$x_1 \wedge x_2 \geq x_1 x_2 \geq x_1 \bullet x_2 \geq x_1 \Delta x_2 \geq 0 \qquad (3.1)$$

Из (3.1) следует, что логическая t-норма среди всех t-норм обладает наибольшими, а драстическая - наименьшими значениями.

Так как операция логического нечеткого произведения является оценкой минимальной связи нечетких значений $x \in X$, то этот минимум можно еще уменьшить, переходя от логической к алгебраической, граничной и драстической t-нормам.

3. Нечеткое суммирование (s - норма)

Нечеткое суммирование (взятие max) является бинарной операцией для $x_1 \in X$ и $x_2 \in X$, определяемой на интервале [0, 1] в форме s-нормы.

*Аксиомы s-нормы:*

S1. $x S \mathbf{1} = 1,\ x S \mathbf{0} = x$ для $\forall x \in [0,1].$

S2. $x_1 S x_2 = x_2 S x_1$ для $\forall x \in [0,1].$

S3. $x_1 S (x_2 S x_3) = (x_1 S x_2) S x_3$ для $\forall x \in [0,1].$

S4. $x_1 \leq x_2 \to x_1 S x_3 \leq x_2 S x_3$ для $\forall x \in [0,1].$

1. *Логическая сумма* (логическая s-норма) нечетких аргументов $x_1 S x_2 = x_1 \vee x_2$ для $\forall x \in X$ определяется в форме операции нахождения max. Это наиболее часто используемая операция s-нормы.

2. *Алгебраическая сумма* - « + ».

$x_1 S x_2 = x_1 + x_2 = x_1 + x_2 - x_1 x_2$ для $\forall x \in X.$

3. *Граничная сумма* - «$\oplus$ ».

$x_1 S x_2 = x_1 \oplus x_2 = (x_1 + x_2) \Lambda 1$ для $\forall x \in X,$

где $\Lambda$ - операция взятия min.

4. *Драстическая сумма* - «$\nabla$ »

$$x_1 S x_2 = x_1 \nabla x_2 = \begin{cases} x_2, & \text{если } x_1 = 0 \\ x_1, & \text{если } x_2 = 0 \\ 1, & \text{в других случаях.} \end{cases} \quad \text{для } \forall x \in X$$

Графическая интерпретация операций s-нормы (так же как и операций t-нормы) может быть показана на единичном кубе. Ее предлагается выполнить читателю самостоятельно. Можно отметить лишь то, что изображение s-нормы инверсно по отношению к изображению t-нормы.

Для различных s-норм нечеткой логики можно записать следующие неравенства [8]:

$$x_1 \vee x_2 \le x_1 + x_2 \le x_1 \oplus x_2 \le x_1 \nabla x_2 \le 1. \qquad (3.2)$$

Из (3.2) следует, что расширение областей определения можно достигнуть переходом от логической к алгебраической, граничной и драстической s-нормам.

В заключение запишем правило де Моргана для t-норм и s-норм:

$(x_1 S x_2)^{\Theta} = x_1^{\Theta} T x_2^{\Theta}$ для $\forall x \in [0,1]$.

$(x_1 T x_2)^{\Theta} = x_1^{\Theta} S x_2^{\Theta}$ для $\forall x \in [0,1]$.

Как показывает имеющийся на данное время опыт, наибольшее применение в системах нечеткого управления находят логические операции типа операторов min/max, реже алгебраические и граничные операции и еще реже драстические операции.

### 3.4. Нечеткие выводы

Процесс обработки нечетких данных с целью получения некоторого результата принято называть нечеткими выводами. Нечеткие выводы строятся на использовании лингвистических, то есть сформулированных в виде предложений на общепринятом языке, правил обработки входной информации и информации, имеющейся в базе знаний.

База знаний строится на основе продукционной и декларативной моделей знаний. Продукционными моделями знаний называют модели, построенные на использовании одной или нескольких предпосылок (обоснований ситуации), которые используются для формирования заключений (выводов).

В самом общем виде продукционные модели знаний используют конструкцию: ЕСЛИ {(... или ...) и (... или ...)}, ТО ...}. Левая часть продукционного правила (ЕСЛИ ...) рассматривается как конъюнкция элементарных условий (предпосылок), а правая часть (ТО ...) - как некоторое множество элементарных действий («включить ...», «выключить ...» и т.д.). Таким образом, построенная на этих принципах база знаний может быть представлена в форме множества пар «ситуация – действие».

В базу знаний включаются и так называемые декларативные модели знаний в форме утверждений (заключений) без обоснования. В основу формирования заключений положены различного рода силлогизмы, т.е. дедуктивные логические заключения, состоящие из двух типов посылок (большой и малой) и самого заключения. При этом посылки могут использовать отмеченные выше продукционные и декларативные модели знаний. Возможны следующие типы силлогизмов:

I тип. Как первая, так и вторая предпосылка строится на использовании продукционной модели знаний.

II тип. Первая («большая») предпосылка строится на продукционной модели знаний, а вторая («малая») предпосылка – на декларативной модели знаний. (Этот тип силлогизма известен в лингвистике как модус поненс.)

III тип. Первая предпосылка использует продукционную модель знаний, а вторая – декларативную, но отрицающую знания, задаваемые первой посылкой. При этом заключение формируется с отрицанием первой части предпосылки.

Рассмотрим формализацию описанных выше силлогизмов.

Обозначим A – множество, определяющее ту или иную предпосылку, а B – множество возможных заключений. Формализация силлогизма – это описание причинно-следственных отношений, связывающих нечеткие множества A и B.

Пусть $R = A \rightarrow B$ обозначает описанное выше отношение нечетких множеств (нечеткое отношение). Нечеткое отношение R можно рассматривать как нечеткое множество на прямом (декартовом) произведении $X \times Y$ полного пространства предпосылок X и полного пространства заключений Y.

В результате процесс получения некоторого заключения $B_1$ по данным наблюдений $A_1$ и известном нечетком отношении $R = A \rightarrow B$ можно записать в виде

$$B_1 = A_1 \bullet R = A_1 \bullet (A \rightarrow B),$$

где $\bullet$ – знак композиционного правила (свертка) нечеткого вывода;

$\rightarrow$ – знак нечеткой импликации.

Напомним, что операция импликации известна как одна из элементарных бинарных операций булевой алгебры. Здесь она расширена на область нечеткой логики.

Следует отметить, что возможно формирование заключений (выводов) двух типов.

I тип – *восходящий вывод (заключение).* Его структура:

Предпосылки – продукционная (если x есть A, то y есть B) и декларативная: x есть $A_1$.

Заключение: y есть $B_1$

II тип – *нисходящий вывод (заключение).*

Предпосылки – продукционная (если x есть A, то y есть B) и декларативная: y есть $B_1$.

Заключение: x есть $A_1$.

В первом типе из предпосылок делается вывод (заключение) по факту (y есть $B_1$). Во втором типе из предпосылки и декларативного вывода делается заключение в форме принадлежности входных аргументов к множеству данных наблюдений (x есть $A_1$). Второй тип заключений адекватен ситуации, когда по выходным признакам делается заключение о причинах (задача диагностики).

### 3.4.1. Способы записи операций нечеткой импликации при восходящих выводах

Областью определения нечеткой импликации $A \rightarrow B = R$, является интервал [0, 1]. Как известно, в булевой алгебре импликация $x_1 \rightarrow x_2 = \overline{x}_1 \vee x_2$, но $\overline{x}_1 = 1 - x_1$, тогда

$$x_1 \to x_2 = (1 - x_1) \vee x_2. \qquad (3.3)$$

Выражение (3.3) является исходным и для представления нечеткой импликации с той лишь разницей, что вместо аргументов x должны использоваться соответствующие функции принадлежности.

Если $x_1 \in A$, а $y \in B$, причем $0 \le x \le 1$, $0 \le y \le 1$, то

$$\mu_R(x, y) = \mu_{A \to B}(x, y) = (1 - \mu_A(x)) \vee \mu_B(y). \qquad (3.4)$$

Существует и другая форма представления импликации в булевой алгебре - формула импликации Лукашевича: $x_1 \to x_2 = (1 - x_1 + x_2) \wedge 1$. В терминах расширенных операций формула Лукашевича есть уже упоминавшаяся ранее граничная сумма, так как, если $\bar{x}_1 = 1 - x_1$, получим

$$x_1 \to x_2 = \bar{x}_1 \oplus x_2 = (1 - x_1 + x_2) \wedge 1$$

Применительно к нечеткой логике, заменив аргументы их функциями принадлежности, получим

$$\mu_R(x, y) = \mu_{A \to B}(x, y) = (1 - \mu_A(x) + \mu_B(y)) \wedge 1 \qquad (3.5)$$

Ряд авторов предлагают и другие способы формализации заключения. Например, рассмотрим случай формирования заключения, когда предпосылка (в смысле булевой алгебры) является ложной ($x_1$=0). В этом случае можно сделать любое заключение. Но в соответствии со здравым смыслом при ложной предпосылке и результат (заключение) следует считать ложным. Последнее обстоятельство привело к тому, что для описания отношений было предложено не ограничиваться функцией импликации.

В частности, для этих целей в ряде случаев используется t-норма в форме логического произведения, т.е.

$$x_1 \to x_2 = x_1 \wedge x_2. \qquad (3.6)$$

Очевидно, что запись отношения в форме (3.6) не является импликацией, а соответствует формированию min. Применительно к нечеткой логике выражение (3.6) должно записываться через функции принадлежности.

Рассмотрим, как используется описанная выше методика формализации нечеткого заключения в виде реализации композиционного правила $B_1 = A_1 \bullet B = A_1 \bullet (A \to B)$ на уровне функций принадлежности $\mu_{B_1}(x, y)$, $\mu_{A_1}(x)$, $\mu_A(x)$, $\mu_B(y)$.

При формализации свертки будем использовать наличие min/max операции t- и s-норм.

$$\mu_{B_1}(x, y) = \bigvee_{x \in X} (\mu_{A_1}(x) \wedge \mu_R(x, y)) = \bigvee_{x \in X} (\mu_{A_1}(x) \wedge (\mu_A(x) \wedge \mu_B(y))).$$

Так как операция V - взятия max берется по x, а $\mu_B(y) \notin x$, то $\mu_{B_1}(x, y)$ можно записать:

$$\mu_{B_1}(x, y) = [\bigvee_{x \in X} (\mu_{A_1}(x) \wedge \mu_A(x))] \wedge \mu_B(y) = \bigvee_{x \in X} \mu_{A_1 \cap A}(x) \wedge \mu_B(y) \qquad (3.7)$$

Полученное выражение можно трактовать следующим образом. Функция принадлежности $\mu_{B_1}(x, y)$, описывающая нечеткое множество возможных

результатов, определяется как min/max композиция множеств A, $A_1$ и B. При этом композиция выполняется в два этапа. На первом этапе с помощью операции max определяется композиция множеств A и $A_1$. На втором этапе с использованием операции min определяется композиция множеств A, $A_1$ и B, т.е. нечеткое множество результатов.

### 3.4.2. Нисходящие нечеткие выводы

Нисходящие выводы – это выводы, позволяющие по заключению идентифицировать причины появления такого заключения (явления).

Обозначим X - полное пространство предпосылок ($x_1$, $x_2$,..., $x_m$), а Y - полное пространство заключений ($y_1$,$y_2$,...,$y_n$).

При нисходящих выводах требуется найти $x_i \in X, i = \overline{1,m}$, если Y = {$y_1$,$y_2$,..., $y_n$} известно из базы знаний эксперта.

Очевидно, что между $x_i$ и $y_j$ существует причинно-следственная связь. Эту связь для каждой пары значений x и y обозначим через $r_{ij}$. Полное описание этой связи можно представить в виде матрицы $R = [r_{ij}], i = \overline{1,m}, j = \overline{1,n}$; область определения $r_{ij}$ - [0, 1].

При формализации причинно-следственных связей (отношений) x и y будем использовать (как и в случае восходящих выводов) min/max операторы.

Пусть

$$R = \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \dots & \dots & \dots & \dots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{pmatrix},$$

Y = ($y_1$, $y_2$,..., $y_n$)$\in$ B – (множество заключений),

X = ($x_1$, $x_2$,..., $x_m$)$\in$ A – (множество факторов).

Учитывая выражение для композиции $B_1 = A \bullet B$, можно записать

$$(y_1, y_2, \dots, y_n) = (x_1, x_2, \dots, x_m) \bullet \begin{pmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ r_{21} & r_{22} & \dots & r_{2n} \\ \dots & \dots & \dots & \dots \\ r_{m1} & r_{m2} & \dots & r_{mn} \end{pmatrix}.$$

Это векторно-матричное уравнение с пока не определенным правилом свертки X×Y. Применим операцию транспонирования к обеим частям этого уравнения:

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} r_{11} & r_{21} & \dots & r_{m1} \\ r_{12} & r_{22} & \dots & r_{m2} \\ \dots & \dots & \dots & \dots \\ r_{1n} & r_{2m} & \dots & r_{mn} \end{pmatrix} \bullet \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{pmatrix} \qquad (3.8)$$

Для решения уравнения (3.8) воспользуемся известным правилом композиции нечетких выводов с использованием операторов min/max.

Учитывая, что композиция определяется декартовым (простым) произведением, в результате получим:

$$\left.\begin{array}{l} y_1 = (r_{11} \wedge x_1) \vee (r_{21} \wedge x_2) \vee \dots \vee (r_{m1} \wedge x_m) \\ y_2 = (r_{12} \wedge x_1) \vee (r_{22} \wedge x_2) \vee \dots \vee (r_{m2} \wedge x_m) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ y_n = (r_{1n} \wedge x_1) \vee (r_{2n} \wedge x_2) \vee \dots \vee (r_{mn} \wedge x_m) \end{array}\right\} \qquad (3.9)$$

Система уравнений (3.9) описывает в min/max операторах связь между выходом (заключением) и входом (предпосылками) при известных членах матрицы R. Из уравнений (3.9), зная левую часть $y_j$ и $r_{ij}$, необходимо найти такие значения $x_i (i = \overline{1,m})$, которые при выполнении операции $(r_{ij} \wedge x_i) \to \min$ обеспечивали бы равенство левой и правой частей каждого из уравнений $y_j = \underset{i}{V}(r_{ij} \underset{j}{\wedge} x_i)$. Так как из возможных конъюнкций (min) выбирается максимальная (операция V), то при n≥m удается определить возможные диапазоны изменения $x_i$ и указать наиболее вероятные причины появления той или иной ситуации на выходе диагностируемой системы.

### 3.5. Способы преобразования четких данных в нечеткие (фаззификация)

Процесс преобразования исходных четких значений в соответствующие значения функций принадлежности (т.е. нечеткие значения) получил название фаззификации. Приведем пример.

Можно оценить температуру воздуха тремя оценками: «холодно», «нормально» и «жарко». Пусть функции принадлежности к каждому из множеств (A, B и C) показаны на рис. 3.10.

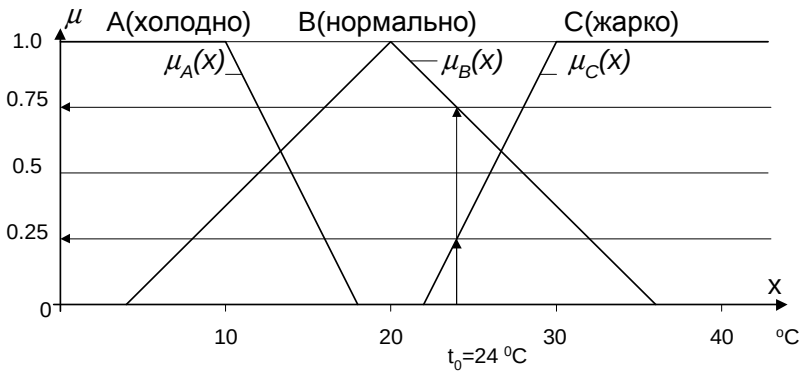



Рис. 3.10

Предположим, что измеренная температура t = 24°C. В этом случае температура будет охарактеризована следующими значениями функций принадлежности :

$\mu_A$ (t = 24°C) = 0 («холодно»),

$\mu_B$ (t = 24°C) = 0,75 («нормально»),

$\mu_C$ (t = 24°C) = 0,25 («жарко»).

Таким образом, в данном примере принадлежность к нечеткой оценке «нормально» оказалась наибольшей (0,75). Но эту же температуру можно отнести и к «жарко» (с оценкой 0,25). Из этого примера видно, что преобразование четких исходных данных в нечеткие при наличии функций принадлежности производится достаточно просто.

Если измерения производятся неточно («на глазок»), то они должны в базе знаний фигурировать также в форме соответствующих функций принадлежности. Например, если функция принадлежности измерения расстояния экспертом представлена кривой $\mu_A(x)$, а функция принадлежности измерения оператором в виде $\mu_{A_1}(x)$ (рис. 3.11), то операцию получения результирующего нечеткого значения входной переменной можно представить следующим образом.

Рассмотрим композицию множеств A и $A_1$ в форме $\mu_{A\bigcap A_1}(x) = \mu_A(x) \wedge \mu_{A_1}(x)$. Определим максимальное значение композиции A и $A_1$ в виде $\alpha = \underset{x\in X}{\vee}(\mu_A(x) \underset{x\in X}{\wedge} \mu_{A_1}(x))$, что и будет характеризовать нечеткое значение (рис. 3.11).

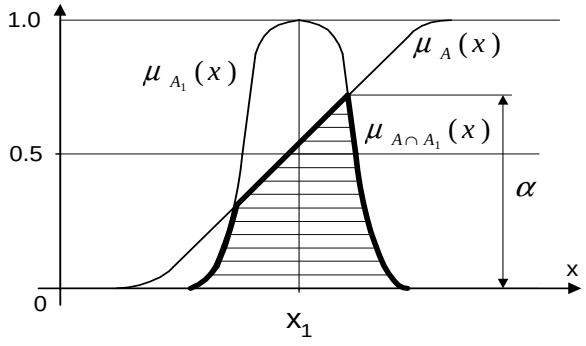



Рис. 3.11

Форма функций принадлежности $\mu(x)$ может быть самой разнообразной. Для систем управления наибольшее распространение получили функции принадлежности следующих стандартных видов:

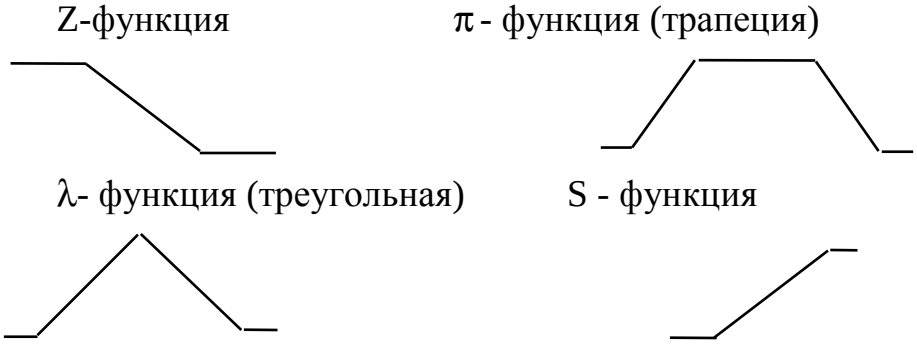



Рис. 3.12

Однако для решения специальных задач могут использоваться и нелинейные зависимости более сложной формы.

## 3.6. Способы построения базы знаний и лингвистических правил обработки нечетких данных

Существуют четыре способа построения базы знаний и лингвистических правил обработки нечетких данных.

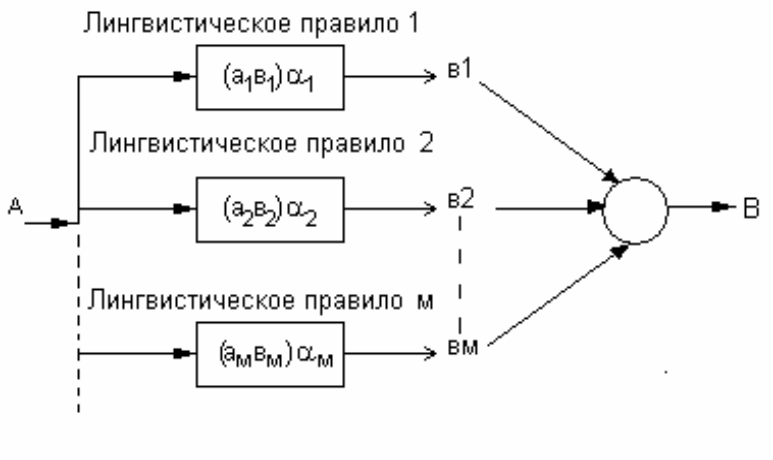
*Первый способ* основан на использовании опыта и знаний эксперта. При этом в словесной форме фиксируются ответ квалифицированного оператора и знания инженера по управлению, которые обобщаются в виде соответствующих функций принадлежности и лингвистических правил управления.

*Второй способ* базируется на разработке моделей действий оператора на существующем оборудовании. Способ используется в тех случаях, когда от экспертов не удается получить описания их действий. Оператор запоминает свои действия в форме манипуляции рук (ног), но представить их на словесном уровне затрудняется.

*Третий способ* использует идею обучения на реальном оборудовании или адекватной модели. Обучение ведется от простого к сложному. В результате достигается постепенное совершенствование правил управления. Этот метод находит наибольшее применение при разработке систем нечеткого управления роботами.

*Четвертый способ* предполагает разработку нечеткой модели работы оборудования. Он используется в случаях, когда априори предполагается создание такой модели. При этом если модель создается в форме лингвистических описаний типа «если …, то …», то правила нечеткого управления легко выводятся теоретически, исходя из правил обработки нечетких множеств и нечеткой логики.

Вне зависимости от способа формирования, в базе знаний обычно хранится целая система знаний об объекте или процессе. Кроме того, в нечетких суждениях, описываемых в предпосылках и заключениях каждого правила, может иметься несколько вариантов. В общем виде база знаний может состоять из набора лингвистических правил (рис. 3.13).




A – нечеткое множество предпосылок;
B – нечеткое множество заключений (управлений).

Рис. 3.13

Пусть в рассмотренном ранее примере с измерением температуры воздуха применяется нечеткий регулятор температуры с воздействием на вентиль подачи топлива.

База данных может состоять из трех лингвистических правил:

ЛП1: Если $x \in A$ = «жарко», то установить вентиль в положение «убавить».

ЛП2: Если $x \in A$ = «нормально», то оставить без изменения положение вентиля.

ЛП3: Если $x \in A$ = «холодно», то вентиль перевести в положение «прибавить».

Пусть функции принадлежности заключений (управляющей переменной) $\mu(y)$, определяющих множество возможных положений вентиля подачи топлива, имеют вид, показанный на рис. 3.14. Так как при «точных» измерениях температуры (t = 24°C) действуют два лингвистических правила ЛП1 («жарко») и ЛП2 («нормально») с различными количественными значениями соответствующих функций принадлежности (рис. 3.11) $\mu_B(x) = 0{,}75 = \alpha_{ЛП2}$ и $\mu_C(x) = 0{,}25 = \alpha_{ЛП1}$, то эти значения имеют смысл «весов» соответствующих предпосылок. Если спроектировать (редуцировать) функции принадлежности $\alpha_{ЛП1}$ и $\alpha_{ЛП2}$ на функции принадлежности заключений (множества М и К на рис. 3.14), то можно получить множества $К_1$ и $М_2$, определяющие возможные положения

вентиля подачи топлива.

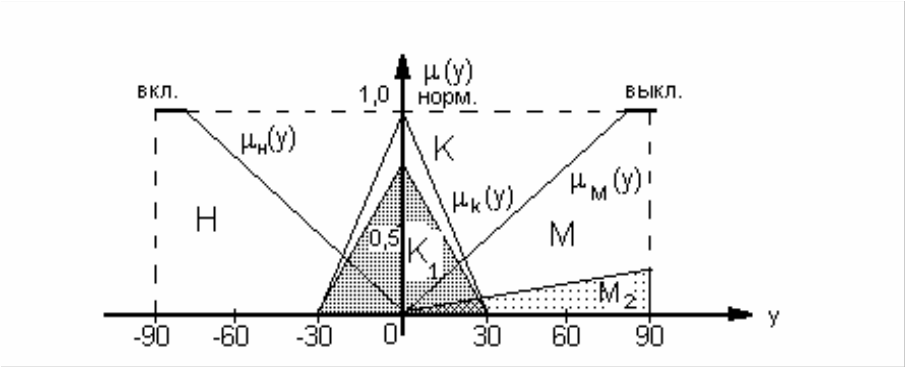



Рис. 3.14

Показанные на рис. 3.15 в виде заштрихованных областей множества $К_1$ и $М_2$ получены простым умножением каждого элемента множеств К и М на значения $\alpha_{ЛП1}$ и $\alpha_{ЛП2}$, т.е.

$$\mu_{ЛП1}(y)=\alpha_{ЛП1}\,M(y);$$
$$\mu_{ЛП2}(y)=\alpha_{ЛП2}\,K(y). \qquad (3.10)$$

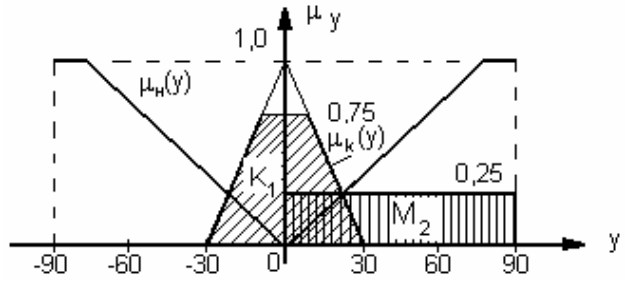



Рис. 3.15

Для уменьшения вычислительных затрат при редуцировании множеств предпосылок на множество заключений используют метод отсечек на уровне максимальных значений $\alpha_{ЛПj}$ (рис. 3.15). В общем случае нечетких измерений (см. рис. 3.11) отсечка на максимальном уровне $\alpha$ композиций множеств А и $А_1$ позволяет сравнительно просто и быстро определить композицию множеств предпосылок и заключений в форме (3.10).

### 3.7. Преобразование нечетких значений в чёткие (дефаззификация)

Как отмечалось ранее, полученное множество заключений в реальных системах управления должно быть преобразовано в конкретное значение управляющего сигнала u. Для решения задачи дефаззификации предложено несколько методов. Наибольшее распространение при практических разработках получили метод центра тяжести и метод центра максимумов.

*Метод центра тяжести* основан на идее вычисления центра тяжести площади, ограниченной результирующей функцией принадлежности заключений, учитывающей все лингвистические правила, участвующие в формировании

заключения. В соответствии с этим методом управление определяется в виде:

$$u = y_{цт} = \int_{-\infty}^{+\infty} \mu_{D1}(y) y dy \Big/ \int_{-\infty}^{+\infty} \mu_{D1}(y) dy ,$$

где $\mu_{D1}$ - результирующая функция принадлежности выводов (заключений).

Для рассмотренного выше примера $\mu_{D1}(y) = \max\{\alpha_{лп1} M(y), \alpha_{лп2} K(y)\}$. Метод центра тяжести логичен, так как при композиции ряда лингвистических правил формирования заключений учитывается значимость каждого правила.

При использовании *метода центра максимумов* результирующее значение управления определяется по формуле:

$$u = y = \sum_{j=1}^{n} \mu_{лпj}^{M}(y) y_j^{M} \Big/ \sum_{j=1}^{n} \mu_{лпj}^{M} ,$$

где: $\mu_{ЛПj}^{M}(y)$ – максимальное значение функции принадлежности j-го лингвистического правила формирования заключений,

$y_j^{M}$ –значение выходной переменной (управления), соответствующее максимальному значению функции принадлежности $\mu_{ЛПj}^{M}(y)$.

Практическая реализация систем нечеткого управления здесь не рассматривается. С некоторыми из них можно познакомиться, например, в [8], [16], а также в журнальной литературе и в демонстрационных примерах системы Matlab.

Следует отметить лишь некоторые особенности систем управления, использующих методы обработки нечеткой информации. Как известно, одним из необходимых условий их работоспособности является обеспечение устойчивости. Как и “четкие”, нечеткие системы могут строиться в форме незамкнутых, так и замкнутых систем. Область применения разомкнутых систем (систем без обратной связи) – это, в основном, системы обработки нечеткой информации различного назначения, экспертные системы, старт-стопные системы программного и программно-логического управления и т.д.

Для таких систем практически не существует проблемы обеспечения устойчивости. В замкнутых системах для обеспечения их работоспособности необходимо выполнение условий устойчивости. В настоящее время при формировании управления, в связи с лингвистическим характером исходной информации, не существует прямых (специальных) методов анализа устойчивости замкнутых нечетких систем управления. Причиной такого положения, по нашему мнению, является лингвистический характер исходной информации. По-видимому, прямое решение проблем устойчивости нечётких систем следует искать, привлекая методы обучения и эволюции. Однако в ближайшем будущем, с целью инженерного применения, подобные методы вряд ли будут найдены.

Для анализа устойчивости замкнутых систем управления в различных работах по теории нечётких систем предлагается использовать хорошо разработанный аппарат анализа устойчивости обычных (чётких) систем.

Как правило, используется та или иная модификация второго метода Ляпунова. При этом, помимо известных проблем использования этого метода, возникает весьма сложная проблема построения формализованной модели нечеткой системы. В результате, анализ устойчивости замкнутых систем нечеткого управления превращается в задачу более сложную по сравнению анализом устойчивости четких систем аналогичного класса.

В заключение отметим, что нечеткие системы не заменяют обычные (алгоритмические) системы управления. Однако в условиях плохой определенности они дополняют обычные системы и в ряде случаев упрощают реализацию управления. Достоинством нечеткого подхода является его "естественность" и достаточно простые математические понятия. Но в любых случаях следует руководствоваться правилом: если существует более простое (и удовлетворительное!) решение – используйте его.

В последние годы происходит быстрый рост приложений нечеткой логики: от потребительских товаров (например, фотокамеры, моечные машины, микроволновые печи) до управления производственными процессами и механизмами (башенные краны, электропоезда), медицинской диагностической аппаратурой, систем поддержки принятия решений и т.д.

Методы проектирования нечетких систем в настоящее время еще далеки от завершения. Однако уже используются программные средства для разработки и отладки систем, основанных на нечетких выводах. К их числу относится FuzzyTECH –развитая система проектирования и отладки на основе нечеткой технологии [16], а также инструментарий нечеткой логики (Fuzzy Logic Toolbox) в среде MATLAB [17].

Дальнейшее развитие нечеткого подхода, по всей видимости, связано с использованием нечеткой логики в комбинации с нейронными сетями и генетическими алгоритмами. Нечеткая логика, нейронные сети и генетические алгоритмы могут рассматриваться как главные составляющие того, что называют мягкой обработкой.

# 4. ОСОБЕННОСТИ ТЕХНИЧЕСКОЙ И ПРОГРАММНОЙ РЕАЛИЗАЦИИ УПРАВЛЯЮЩИХ УСТРОЙСТВ В КОМПЬЮТЕРНЫХ СИСТЕМАХ УПРАВЛЕНИЯ

В компьютерных системах управления (КСУ) управляющие устройства строятся на базе:

-специализированных **контроллеров** различной степени сложности и универсальности;

-**компьютеров** общего назначения (в том числе и персональных) с использованием специальных плат - устройств сопряжения с объектом (УСО), физически и программно совместимых с компьютером;

-компьютеров, объединенных **в локальную сеть** и имеющих общую базу данных;

-**узкоспециализированных компьютеров**, ориентированных на решение конкретных функциональных задач контроля и управления.

Учитывая то, что контроллеры, локальные управляющие сети и узко-специализированные компьютеры изучаются в соответствующих курсах, ниже рассматриваются особенности управляющих устройств второй группы. Следует отметить, что вследствие распространенности и доступности, в частности, персональных компьютеров, включение в их состав специали-зированных аппаратно и программно совместимых плат УСО обеспечивает:

- возможность многофункционального использования компьютера,
- относительно простую, быструю и недорогую реализацию как систем сбора и обработки информации, так и систем управления.

Полагая, что читатели достаточно хорошо знакомы с принципами организации, архитектурой и программным обеспечением современных компьютеров, далее основное внимание уделим рассмотрению особенностей технической и программной реализации специализированных плат УСО и их использования для решения задач сбора и преобразования информации, а также взаимодействия с компьютером.

При выборе средств для реализации управляющих устройств КСУ прежде всего необходимо сформулировать требования к этим средствам.

## 4.1 Формирование требований к вычислительным средствам реализации алгоритмов систем автоматического управления

Одним из важных этапов создания СУ является определение требований к применяемым в них вычислительным программным и аппаратным средствам. Возможны общая и частная постановка задачи выбора вычислительных средств.

При общей постановке из номенклатуры имеющихся технических и программных средств управляющей вычислительной техники общепромышленного назначения необходимо выбрать такую конфигурацию и ОС

УВК, которые бы наилучшим образом в смысле выбранного критерия удовлетворяли предъявленным требованиям и ограничениям.

Частная постановка возникает при создании новой системы управления на базе имеющихся вычислительных средств.

Процесс выбора технических средств включает в общем случае три основных этапа.

### *1.1. Этапы выбора*

а) Первый этап

На первом этапе проверяется выполнение основных требований и ограничений.

1. Требование к быстродействию:

$$V \geq \frac{Nпп}{T_{рдоп}},$$

где $N_{пр}$ - объем вычислительной работы, необходимый для решения задачи управления, т.е. реализации разработанных алгоритмов;

$T_{рдоп}$ - допустимое по ТЗ время решения задачи.

Объем вычислений определяется обычно в приведенных эталонных операциях. Например, в операциях пересылки регистр-регистр или сложения двух чисел с фиксированной точкой, которые выполняются за один такт.

2. Требование к объему памяти:

$$Q \geq Q_{тр},$$

где $Q_{тр}$ - требуемый объем памяти ОЗУ или раздельно ОЗУ и внешней памяти, которые зависят от набора разработанных алгоритмов.

3. Требование к точности получаемых результатов:

$$R \geq R_{ТР},$$

где R – разрядность представления информации в устройствах комплекса (процессор, ОП, АЦП, ЦАП и другие модули);

$R_{тр}$ – требуемая разрядность, зависящая от алгоритмов, точности получения результатов и исходных данных, приводимых в ТЗ.

4. Требование по стоимости:

$$C \leq C_{доп},$$

где $C_{доп}$ – допустимые затраты на разработку управляющего комплекса.

5. Требование по надежности:

$$T \geq T_{доп},$$

где $T_{доп}$ – допустимое среднее время наработки на отказ по ТЗ.

6. Наличие в составе УВК всех устройств (УСО, модемы и т.д.).

7. Наличие ОС, обеспечивающей требуемый режим работы работы, например режим РВ.

б) Второй этап

На втором этапе производится уточненная проверка выполнения ограничений на параметры УВК с учетом устройств ввода – вывода для выбранного множества вариантов. Множество может оказаться пустым. В этом случае необходимо вернуться на первый этап.

в) Третий этап

На третьем этапе выбирается единственный вариант комплекса в соответствии с критериями. Например:

1. Критерий минимума приведенных затрат

$$J_1 = \min\{C + \tau_н w_э\},$$

где C – единовременные капитальные затраты,
$\tau_н$ - нормативный срок окупаемости (6 лет),
$w_э$ – эксплуатационные расходы за год.

2. Критерий максимальной производительности

$$J_2 = \max\left\{\frac{1}{T_P}\right\}.$$

3. Критерий наилучшего соответствия группе требований

$$J_3 = \max \sum_{1}^{r} \omega_i \beta_i,$$

где r – число важнейших требований,
$\omega_i$ - весовые коэффициенты (экспертные оценки),
$\beta_i$ - значение i – го параметра в относительных единицах.

Таким образом, для выбора технических средств необходимо решить следующие задачи:

- определить требуемый объем вычислительной работы $N_{пр}$;
- определить требуемый объем памяти $Q_{тр}$;
- рассчитать разрядность представления информации в различных модулях УВК, т.е. в АЦП, ЦАП, процессоре и других модулях.

### *1.2. Оценка объема вычислительной работы*

Эта величина необходима для оценки требуемого быстродействия УВК. На практике объем вычислительной работы оценивается тремя способами:

- по аналогии;

- аналитически;
- методом моделирования на ЭВМ.

Метод аналогий предполагает использование априорных сведений о требуемом объеме вычислений (практика, литература).

Метод аналитический требует составления по алгоритму машинной программы. Он заключается в подсчете требуемых для реализации заданных алгоритмов машинных операций по детальной схеме программы.

Метод моделирования дает наиболее достоверную оценку объема вычислительной работы и является наиболее простым. Он заключается в моделировании на имеющейся у разработчика вычислительной технике полученных алгоритмов и последующей коррекции результатов.

### *1.3. Оценка объема требуемой памяти*

В общем случае объем памяти как ПЗУ так и ОЗУ (включая и внешнюю) можно оценить следующим образом:

$$Q_{ТР} = Q_{ОС} + Q_{ЗАГР} = Q_{ОС} + (Q_{ПР} + Q_{ДАН} + Q_{ССР}),$$

где $Q_{ОС}$ - объем памяти, занимаемый ОС;

$Q_{ПР}$ - объем памяти, занимаемый программой;

$Q_{ДАН}$ - объем памяти для входных, выходных и промежуточных данных алгоритмов;

$Q_{ССР}$ - объем памяти, занимаемый вспомогательными алгоритмами необходимыми для вычислений (библиотеки, драйверы, утилиты).

Более подробно об определении $N_{ПР}$ и $Q_{ТР}$ смотри в (2).

### *1.4. Определение требуемой разрядности представления информации В УВК*

В различных модулях управляющих комплексов данные представляются в различных формах счисления. Так в блоках АЦП и ЦАП, как правило, используется режим представления информации с фиксированной точкой. В контроллерах также иногда используется этот режим.

Основным моментом при определении требуемой разрядности является условие получения результата с требуемой точностью.

В общем случае в УВК реализуется уравнение вида:

$$y = \varphi(x),$$

где x – вектор входных переменных, поступающий в УВК от датчиков ОУ, часть из которых поступает через АЦП;

y – вектор управления, передаваемый на ОУ через ЦАП;

$\varphi(x)$ - алгоритмизируемая функция преобразования x в y.

### *1.5. Уравнение баланса ошибок*

Получение значения управления с требуемой точностью означает выбор таукой разрядности представления информации в УВК, которая обеспечивает выполнение условия баланса ошибок:

$$\sigma_{Y} = \sqrt{\sigma_{М}^2 + \sigma_{Т}^2 + \sigma_{И}^2} \le \sigma_{ЕДДОП}$$

где $\sigma_{УДДО}$ - допустимое значение среднеквадратической погрешности управления;

$\sigma_{М}, \sigma_{Т}, \sigma_{И}$ - соответственно среднеквадратические методическая, трансформированная и инструментальная погрешности вычисления управления.

Среднеквадратические значения и абсолютные максимальные погрешности для нормального закона распределения связаны вероятностным соотношением по правилу 3–х $\sigma$. В соответствии с этим законом, появление максимальной ошибки $\Delta_{М} > 3\sigma$ не превышает 0,27%.

### *1.6. Методическая погрешность*

Эта погрешность определяется численным методом определения $y = \varphi(x)$ и может быть сделана достаточно малой. Если известен метод определения:

$$\varphi(x) = y \pm \Delta_{М},$$

где $\Delta_{М}$ - абсолютная погрешность метода.

Для нормального закона распределения $\sigma_{М} = \frac{\Delta_{М}}{\sqrt{3}}$. Для арифметических операций сложения и вычитания $\Delta_{М} = 0$ и $\sigma_{М} = 0$.

### *1.7. Инструментальная погрешность*

Инструментальная погрешность обусловлена конечной разрядностью операционного устройства машины и является следствием накопления погрешностей округления:

$$\Delta_{ОК\ МАХ} = 0{,}5\,\Delta A_{i},$$

где $\Delta A_i$ – ед. младшего разряда. Тогда:

$$\sigma_{КВ} = 0{,}5\,\Delta A_i \sqrt{3} = \frac{\Delta A_i}{2\sqrt{3}}.$$

Так как погрешности округлений в ряду последовательных округлений независимы, то результирующая инструментальная погрешность равна:

$$\sigma_{И} = \sqrt{\sigma^2_{ОК1} + \sigma^2_{ОК2} + \cdots + \sigma^2_{ОК}} = \sqrt{\frac{m}{12}}\Delta A_i,$$

где m – число округлений (умножений, делений).

Практически получить таким способом значение инструментальной ошибки очень сложно.

### *1.8. Трансформированная погрешность*

Эта погрешность определяется погрешностью входных переменных, их взаимовлиянием (т.е. корреляцией), а также видом функции $y=\varphi(x)$. Так как в нормальных режимах работы УВК погрешности относительно невелики, то, линеаризируя функцию $y=\varphi(x)$ разложением в ряд Тейлора и отбрасывая элементы выше первого порядка, можно записать:

$$y = \varphi(m_{X)} + \sum_1^n \frac{\varphi(x)}{\partial x_i}(x_i - m_{Xi}).$$

Пусть $x_i^o = x_i - m_{Xi}$ - центрированная случайная величина $x_i$. Тогда y можно представить в виде:

$$y = \varphi(x) \approx b_i + \sum_1^n a_i x_i^o = b_i + \sum_i^n \Delta y_i\ .$$

Математическое ожидание и дисперсия линейной функции y:

$$m_Y = M\left[ b_i + \sum_1^n a_i x_i \right] = b + \sum_1^n a_i M\left[x_i^o\right] = b;$$

$$D[y] = D\left[ b + \sum_1^n a_i x_i^o \right] = \sum_1^n D\left[a_i x_i^o\right] + 2\sum_{i\neq j} a_i a_j k_{ij} =$$

$$= \sum_1^n \left[ \frac{\partial \varphi(x)}{\partial x_i} \right]_{y=m_y} D(x_i^o) + 2\sum_{i\neq j} \frac{\partial \varphi(x)}{\partial x_i}\frac{\partial \varphi(x)}{\partial x_j}\sigma_{xi}\sigma_{xj}r_{ij}\ .$$

Здесь $\Delta y_i = a_i x_i^o$; $k_{ij}$ – корреляционная функция случайных величин $x_i$ и $x_j$; $r_{ij}$ – коэффициент корреляции. Так как, $\sigma_y = \sqrt{D(y)}$, то трансформированная ошибка определяется:

$$\sigma_T = \sqrt{\sum_1^n \left[\frac{\partial \varphi(x)}{\partial x_i}\right]^2 \sigma_{xi}^2 + 2\sum_{i\neq j} \frac{\partial \varphi(x)}{\partial x_i}\frac{\partial \varphi(x)}{\partial X_j} r_{ij}\sigma_{xi}\sigma_{xj}}.$$

Если входные переменные не коррелированы, т.е. $r_{ij}=0$, то

$$\sigma_T = \sqrt{\sum_1^n \left[\frac{\partial \varphi(x)}{\partial x_i}\right]^2 \sigma_{xi}^2}.$$

*2. Методика определения разрядной сетки устройств УВК*

Для выбора конкретных устройств УВК необходимо определить:

- разрядность представления информации на входе УВК, т.е. разрядность цифрового датчика или АЦП для аналогового сигнала;
- разрядность представления информации на выходе УВК, т.е. разрядность цифрового исполнительного органа или ЦАП для аналогового исполнительного органа;
- разрядность процессора, включая разрядность мантиссы и порядка представления информации в УВК.

### *2.1. Исходные данные для определения разрядности устройств УВК*

1. Алгоритм решения задачи, т. Е. поиск функции управления: $y = \varphi(x)$.
2. Максимальные значения входных и выходных переменных $x_{imax}$, $x_{imin.}$, $y_{imax}$, $y_{imin}$.
3. Среднеквадратические погрешности измерения входных переменных $\sigma_{BXi}$.
4. Среднеквадратические погрешности выходных переменных $\sigma_{ВЫХi.}$
5. Погрешность метода обработки входной информации и метода вычисления управлений.
6. Коэффициенты корреляции входных переменных $r_{ij}$.

### *2.2. Определение разрядности представления информации на входе и выходе УВК*

Пусть:

- максимальные значения входных переменных $x_{imax}$, $x_{imin}$;
- среднеквадратическая погрешность датчиков входных переменных $\sigma_{xi}$.

Значение ошибки снимаемых с датчиков величин не должно превышать ! младшего разряда цифрового датчика и, следовательно, цифрового входа УВК. При равновероятностном характере появления ошибки (события округления) величина младшего разряда цифрового датчика связана с погрешностью квантователя соотношением:

$$\Delta A_i = 2\sqrt{3}\sigma_{KBi}.$$

Тогда, максимальный код измеряемых величин

$$N_{макс} = 2^n - 1 = \frac{|x_{макс}|}{\Delta A_i}.$$

В этом случае разрядность информации, снимаемой с цифрового датчика

$$n_x = E_1\left[\log_2\left(\frac{|x_{iмак}|}{\Delta A_i}+1\right)\right] + n_{зн} == E_1\left[\log_2\left(\frac{|x_{iмак}|}{2\sqrt{3}\min\sigma_{xi}}+1\right)\right] + n_{зн},$$

где $E_1[z]$ – ближайшее большее целое для [z]; $n_{зн}=1$ – знаковый разряд.

Для аналоговых датчиков объекта управления разрядность АЦП

$$n_{АЦП} = E_1\left[\log_2\left(\frac{|x_{iмак}|}{2\sqrt{3}\rho_{АЦП}\min\sigma_{xi}}+1\right)\right] + n_{зн},$$

где $\rho_{АЦП} = \sigma_{АЦП}/\sigma_{xi}$ - коэффициент пропорциональности, учитывающий дополнительную погрешность, вносимую АЦП в канал измерения; $\rho_{АЦП} = 0{,}2 \div 0{,}4$. Это приводит к увеличению разрядности на $2 \div 3$ разряда, учитывающих погрешность АЦП.

Для цифрового исполнительного органа разрядность поступающей на него информации:

$$n_y = E_1\left[\log_2\left(\frac{|y_{iмак}|}{2\sqrt{3}\sigma_{ндоп}}+1\right)\right] + n_{зн}.$$

Аналогично определяется разрядная сетка для передачи кода в канале измерения, а также разрядность выходного регистра цифрового выхода:

$$n_y = E_1\left[\log_2\left(\frac{|y_{\text{имак}}|}{2\sqrt{3}\sigma_{\text{ндоп}}}+1\right)\right]+n_{\text{зн}}.$$

### *2.3. Определение разрядности представления информации в процессоре УВК*

Данный расчет включает в себя получение разрядности порядка $n_{\text{пор}}$ и мантиссы $n_{\text{м}}$ :

$$R_{\text{тр}} = n_{\text{пор}} + n_{\text{м}} .$$

а) Разрядность порядка

Разрядность порядка вычисляется из учета максимального значения порядков всех переменных :

$$n_{\text{пор}} = E_1\left[\log_2\left(\max\left(\left|P_{\text{пормакс}}\right|,\left|P_{\text{пормин}}\right|\right)\right)\right],$$

где $P_{\text{пор макс}}$, $P_{\text{пор мин}}$ – соответственно максимальное и минимальное значение порядка.

$$P_{\text{пормакс}} = E_1\left[\log_2 n_{\text{макс}}\right]$$
$$P_{\text{пормин}} = E_1\left[\log_2 n_{\text{мин}}\right],$$

где $n_{\text{макс}}$. $n_{\text{мин}}$ – соответственно максимальное и минимальное значения среди всех входных, выходных и промежуточных переменных и зависящих от вида функции $\varphi(x)$.

б) Разрядность мантиссы

Определение разрядности мантиссы производится из условия баланса ошибок и представляет наибольшие трудности. Оно включает несколько этапов.

1. Определение методической ошибки, если используется приближенный способ решения задачи. Если известна абсолютная погрешность метода (по литературе, из опыта и т.д.) - $\Delta_{\text{м}}$, то

$$\sigma_{\text{м}} = \frac{\Delta_{\text{м}}}{\sqrt{3}}.$$

2. Вычисление трансформированной ошибки производится аналитически или методом моделирования. В случае аналитического метода определения:

$$\sigma_T = \sqrt{\sum_1^n \left(\frac{\partial y}{\partial x_i}\right)\sigma_{x_i}^2 + 2\sum_{i=1}^n \sum_{i\neq j}^n \left|\frac{\partial y}{\partial x_i}\right|\left|\frac{\partial y}{\partial x_j}\right| r_{ij}\sigma_{x_i}\sigma_{x_j}} .$$

Во многих случаях функция $y = \varphi(x)$ задана алгоритмически и ее производные трудно или невозможно задать аналитически. Тогда частные производные определяются путем моделирования на ВМ методом припасовывания:

$$\frac{\partial y}{\partial x_i} \approx \frac{\varphi(x_1 \cdots x_i + \Delta x_i \cdots x_n) - \varphi(x_1 \cdots x_i \cdots x_n)}{\Delta x_i}.$$

В случае некоррелированных сигналов вторая составляющая в $\sigma_T$ отсутствует, так как $r_{ij}=0$ .

3. Из неравенства ошибок (условие баланса) находят допустимое значение инструментальной ошибки:

$$\sigma_{и_{доп}} = \sqrt{\sigma_{у_{доп}}^2 - \sigma_м^2 - \sigma_T^2}.$$

Зная $\sigma_{идоп}$, определяют допустимое число разрядов с недостоверной информацией из-за ошибок округления:

$$2^{n_{и_{доп}}} - 1 = \frac{A_{и_{доп}}}{\Delta A_i} = \frac{2\sqrt{3}\sigma_{и_{доп}}}{2\sqrt{3}\min\sigma_{x_i}} = \frac{\sigma_{и_{доп}}}{\min\sigma_{x_ш}} \quad и$$

$$n_{и_{доп}} = E\left[\log_2\left(\frac{\sigma_{и_{доп}}}{\min\sigma_{x_i}} + 1\right)\right],$$

где $E[Z]$ - ближайшее меньшее целое для значения Z.

4. Определение фактической инструментальной ошибки методом моделирования, выполняя вычисления в режиме с плавающей точкой с обычной и двойной точностью. Число разрядов с недостоверной иформацией из-за ошибок округления и определяющее инструментальную ошибку должно укладываться в разность между обычной и двойной точностью. В этом случае:

$$A_и = \frac{|y_{об} - y_{дв}|}{\Delta A_{мл_{об}}} = к[\text{ед. мл. дес. разряда}],$$

где $\Delta A_{мл_{об}}$ - вес младшего десятичного разряда результата вычисления $y_{об}$ с обычной (не двойной) точностью.

5. По найденной ошибке $\sigma_и$ находят число разрядов с недостоверной информацией $n_и$:

$$n_и = E_1[\log_2(k+1)].$$

6. В результате получают требуемую разрядность мантиссы:

$$n_м = n_х + \max\{0, n_и - n_{идоп}\} + 1,$$

где 1 – добавочный разряд для выявления переполнения разрядной сетки в модифицированном коде.

8. Результирующая разрядность представления информации в процессоре:

$$R_{тр} = n_{пор} + n_м .$$

## *3. Пример формирования требований к вычислительным средствам*

### *3.1. Исходные данные для формирования требований к УВК для реализации заданных алгоритмов*

1. Алгоритмизируемая функция управления

$$y = k \cdot x_1 \cdot x_2 , \qquad k = 1 \div 500 .$$

2. Число входных переменных

$$n = 3 .$$

3. Тип датчика входных переменных

а) для х – аналоговый,

б) для к – цифровой.

4. Диапазон изменения входных и выходных переменных

$$y_{min} = -5\ В, \quad y_{max} = +5\ В,$$
$$x_{1min} = -5\ В, \quad x_{2min} = -5\ В,$$
$$x_{1max} = +5\ В, \quad x_{2max} = +5\ В.$$

5. Среднеквадратическая погрешность входных и выходных переменных

$$\sigma_{вх1,2} = 10^{-3}, \quad \sigma_{вых} = 10^{-2} .$$

6. Примем погрешность метода $\Delta_м = 10^{-4}$.

7. Коэффициент корреляции $r_{ij} = 0,5$ .

8. Цикл управления $T_ц = 0,5$ с.

9. Режим работы - непрерывный.

10.Режим сохранения данных – накопление (сохранение) данных в течении месяца.

### *3.2. Формирование требований к УВК*

1. Требование по быстродействию.

Заданная длительность цикла вычисления управления $T_ц = 0,5$ с.

За время цикла необходимо:

- принять и преобразовать два входных сигнала – $x_1$ и $x_2$,
- ввести или подтвердить значение коэффициента – k,
- вычислить выходную величину (управление) – y ,
- выдать полученное управление через ЦАП на объект управления.

Практически любые современные вычислительные средства справятся за время $T_ц$ с поставленной задачей, т.е. требование по быстродействию не критично.

2. Требование к объему памяти.

$$Q_{тр} = Q_{ос} + Q_{загр} = Q_{ос} + (Q_{пр} + Q_{дан} + Q_{сс}) \approx Q_{ос} + Q_{дан}.$$

Требуется операционная система РВ (QNX, LINUX, модифицированный WINDOWS и т. д.) и, соответственно, $Q_{ос}$ . Объем требуемой памяти под все данные (входные, выходные и промежуточные) в основном определяется объемом информации, требуемого режимом хранения – $Q_{дан}$. Пусть мы регистрируем текущие значения величин - k, $x_1$, $x_2$ . Тогда

$$Q_{дан} \approx 3 \cdot 2 \cdot N_{см} = 6 \cdot N_{см} = 6 \cdot 60 \cdot 60 \cdot 24 \cdot 30 \approx 15 \cdot 10^6 \text{ (чисел)},$$

где $N_{см}$ – число секунд в месяце. Так как для хранения одного числа требуется 4 байта памяти, то

$$Q_{дан} \approx 15 \cdot 4 \cdot 10^6 = 60 \text{ Мб}.$$

3. Требования к разрядности представления информации в устройствах УВК.

а) Разрядность АЦП для преобразования информации с аналоговых датчиков

$$n_{АЦП} = E_1\left[\log_2\left(\frac{|x_{imax}|}{2\sqrt{3}\rho_{АЦП}\min\sigma_{xi}}+1\right)\right]+n_{зн}.$$

Возьмем $\rho$ =0,4; $|x_{imax}|=5$, $\sigma_{xi}=10^{-3}$, $n_{зн}=1$. Тогда

$$n_{АЦП} = E_1\left[\log_2\left(\frac{5\cdot 10^3}{2\sqrt{3\cdot 0{,}4}}+1\right)\right]+1=$$
$$= E_1\left[\log_2(3613+1)\right]+1=12+1=13.$$

б) Разрядность ЦАП или выходного цифрового регистра

$$n_{цап} = E_1\left[\log_2\left(\frac{|y_{max}|}{2\sqrt{3}\sigma_{уддо}}+1\right)\right]+n_{зн}.$$

Возьмем $\sigma_{удоп}=0{,}01$, $|y_{max}|=5$, $n_{зн}=1$. Тогда

$$n_{цап} = E_1\left[\log_2\left(\frac{5\cdot 100}{2\sqrt{3}}+1\right)\right]+1=$$
$$= E_1\left[\log_2(144+1)\right]+1=8+1=9.$$

в) Разрядность представления информации в процессоре.

- Разрядность порядка

$$n_{пор} = E_1\left[\log_2\left(\max\left(|P_{пор\,max}|,|P_{пор\,min}|\right)\right)\right];$$

$$P_{порmax} = E_1\left[\log_2(n_{max})\right];$$

$$n_{max} = k\cdot x_1\cdot x_2 = 10^2\cdot 5\cdot 5 = 2500;$$

$$P_{пор\,max} = E_1\left[\log_2 2500\right]=12;$$

Следовательно, $n_{пор} = E_1\left[\log_2 12\right]=4$.

## 4.2. Плата ввода/вывода аналоговых и дискретных сигналов

В зависимости от назначения и технической реализации выпускаемые в настоящее время различными фирмами серии плат УСО для установки их в компьютеры общего назначения можно разделить на две группы:

1. Платы без использования специализированных процессоров.
2. Платы с использованием специализированных процессоров.

В зависимости от выполняемых функций различают платы, обеспечивающие:

- прием и формирование дискретных сигналов,
- прием и преобразование аналоговых сигналов в цифровые,
- формирование и преобразование цифровых управляющих сигналов в аналоговые,
- выполнение всех перечисленных выше функций.

Платы, не имеющие в своем составе специализированных процессоров, являются простейшими и недорогими УСО. В этом случае для обработки информации и управления режимами ввода-вывода используется центральный процессор компьютера.

Платы, снабженные специальным (сигнальным) процессором, позволяют обеспечить управление операциями ввода-вывода независимо от загрузки центрального процессора компьютера. Такая организация платы УСО обеспечивает большее быстродействие как системы сбора информации, так и системы управления в целом. На сигнальный процессор также возлагаются задачи первичной обработки информации. Естественно, что по сравнению с платами первой группы они стоят дороже, что в ряде случаев может служить препятствием их использования в разрабатываемой системе управления.

Рассмотрим более подробно принципы построения и особенности использования плат УСО каждой из указанных выше групп на примере платы АЦП-ЦАП L-154 для IBM PC (ПЭВМ), плат приема и формирования дискретных сигналов PLC-720, PLCD-782, PLCD-785 и платы MC-3628/MC-3629, предназначенной для управления приводами трехкоординатного технологического оборудования.

## 4.1 Формирование требований к вычислительным средствам реализации алгоритмов систем автоматического управления

### 4.2.1. Технические характеристики и структура L-154

Плата L-154 предназначена для установки в персональных IBM совместимых компьютерах и представляет собой относительно простое, недорогое и надежное УСО без сигнального процессора. Использование ее оправдано в тех случаях, когда решаемые задачи не требуют ввода/вывода с высокой скоростью. Плата полностью доступна для программирования со стороны компьютера через порты ввода-вывода.

Плата выполнена в стандарте IBM PC XT/AT и устанавливается в любой из свободных разъёмов, расположенных непосредственно в компьютере. Подключение к внешним устройствам осуществляется через внешний разъём платы со стороны задней панели компьютера. L-154 является функционально полным комплексом, включающим в себя многоканальный 12-ти разрядный аналого-цифровой преобразователь с программируемым входным диапазоном сигнала по каждому каналу и частотой преобразования до 70 кГц на один канал.

Генерация прерываний, внутренняя синхронизация и работа в режиме реального времени обеспечивается тремя 16-ти разрядными таймерами с опорной кварцевой частотой 1 МГц. Предусмотрена возможность внешнего запуска АЦП и внешнего запуска третьего счетчика - таймера.

L-154 - имеет одноканальный 12-ти разрядный цифро-аналоговый преобразователь с временем установления 5 мкс, и амплитудой выходного напряжения ± 5,12 В.

Плата содержит три блока перемычек, определяющих адрес платы в адресном пространстве компьютера, номер линии прерывания, конфигурацию АЦП (выбор входного диапазона: ± 5,12 В, ± 2,56 В, ± 1,28 В; число входных каналов: 16 дифферениальных и 32 заземленных).

Структурная схема платы L-154 изображена на рис. 4.1. Схема содержит следующие функциональные элементы:

1) Аналого-цифровой преобразователь (АЦП).

Многоканальный режим преобразования реализуется путем включения во входную цепь преобразователя мультиплексора. Для возможности преобразования сигналов различных диапазонов во входной канал АЦП включен усилитель ОУ с изменяемым коэффициентом усиления.

2) Цифро-аналоговый преобразователь (ЦАП)- одноканальный с фиксированным диапазоном выходного сигнала (±5,12 В). ЦАП через порт ввода/вывода компьютера может получать информацию из компьютера и через внешний разъем платы передавать результаты преобразования на объект управления.

3) Таймеры. На плате установлена микросхема 580ВИ53 трехканального таймера. При помощи трех каналов (A, B, C) таймера можно осуществить программную синхронизацию процессов ввода-вывода и генерировать прерывания по шине IRQ в компьютере. Канал А тактируется от кварцевого генератора частотой 1 МГц. Канал В каскадно связан с таймером А, что позволяет работать плате с большими временными интервалами (до OxFFFF∗OxFFFF мкс). Счетный вход таймера С выведен на внешний разъем. Он может использоваться для программной синхронизации процессов ввода от внешних цифровых сигналов.

Установка режимов таймеров и опрос их текущего состояния достигается при записи соответствующего кода в регистр управления и опросом соответствующего счетчика.

4) ТТЛ входы и ТТЛ выходы - используются для ввода и вывода цифровых сигналов по 8 каналам.

5) Порты ввода-вывода компьютера - обеспечивают связь элементов платы через стандартную ISA шину IBM с основными блоками персонального компьютера.

Плата L-154 имеет специальный набор контактов, которые обеспечивают при ее установке в любой из свободных слотов внутри компьютера подключение на шину ISA.

### 4.2.2. Программное обеспечение

Для платы L-154 разработчиками поставляется готовая библиотека подпрограмм на языке Си и $C^{++}$. Библиотека позволяет использовать все возможности платы, не вдаваясь в тонкости программирования на уровне Ассемблера и портов ввода-вывода. Библиотека содержит законченный набор функций для работы с контроллером прерываний IBM PC, а также функции, позволяющие осуществить ввод-вывод аналоговой и цифровой информации в асинхронном режиме, вводить и выводить аналоговую информацию как в одноканальном, так и в многоканальном режимах, вводить и выводить данные в программном режиме и в режиме генерации прерываний.

Функции библиотеки по выполняемым ими действиям можно разделить на следующие группы:

- инициализации;
- завершения;
- начала вывода (конца ввода);
- информационные;
- изменения уставок;
- получения данных.

Функции определены в файле inp.c и объявлены в заголовочном файле inp.h.

#### Использование библиотеки

Для библиотеки ввода/вывода аналоговой информации требуется библиотека подпрограмм из состава программного обеспечения, поставляемого вместе с платой L-154. Библиотека написана на языке Турбо Ассемблер версии 3.1. Ее исходный текст находится в файле L-154drv.asm.

Для использования функций библиотеки ввода/вывода аналоговой информации необходимо:

- создать файл проекта для языка Си;
- добавить в него файл L-154drv.obj;
- добавить в него файл inp.obj (или inp.c);
- создать и добавить в него файл с основной программой обработки информации - xxx.c;
- добавить в начало файла строку #include"inp.h" (файл inp.h,

содержащий описание функций библиотеки, должен находиться в текущей директории, где создан файл проектов);

- если требуется использовать дополнительные функции библиотеки ввода/вывода (например, для сохранения вводимых и выводимых данных, а также результатов их обработки в файле), то нужно добавить в файл проекта файл results.c, а в файл с исходным текстом основной программы - строку #include"results.h";
- если требуется использование функций библиотеки L-154drv.asm, то добавить в начало файла строку #include"function.h".

Более подробно с программным обеспечением можно познакомиться в техническом описании и инструкции по эксплуатации фирмы-разработчика платы (фирма L-CARD).

Ниже остановимся лишь на особенностях реализации режима работы в реальном времени, так как этот режим обязателен для осуществления процесса управления.

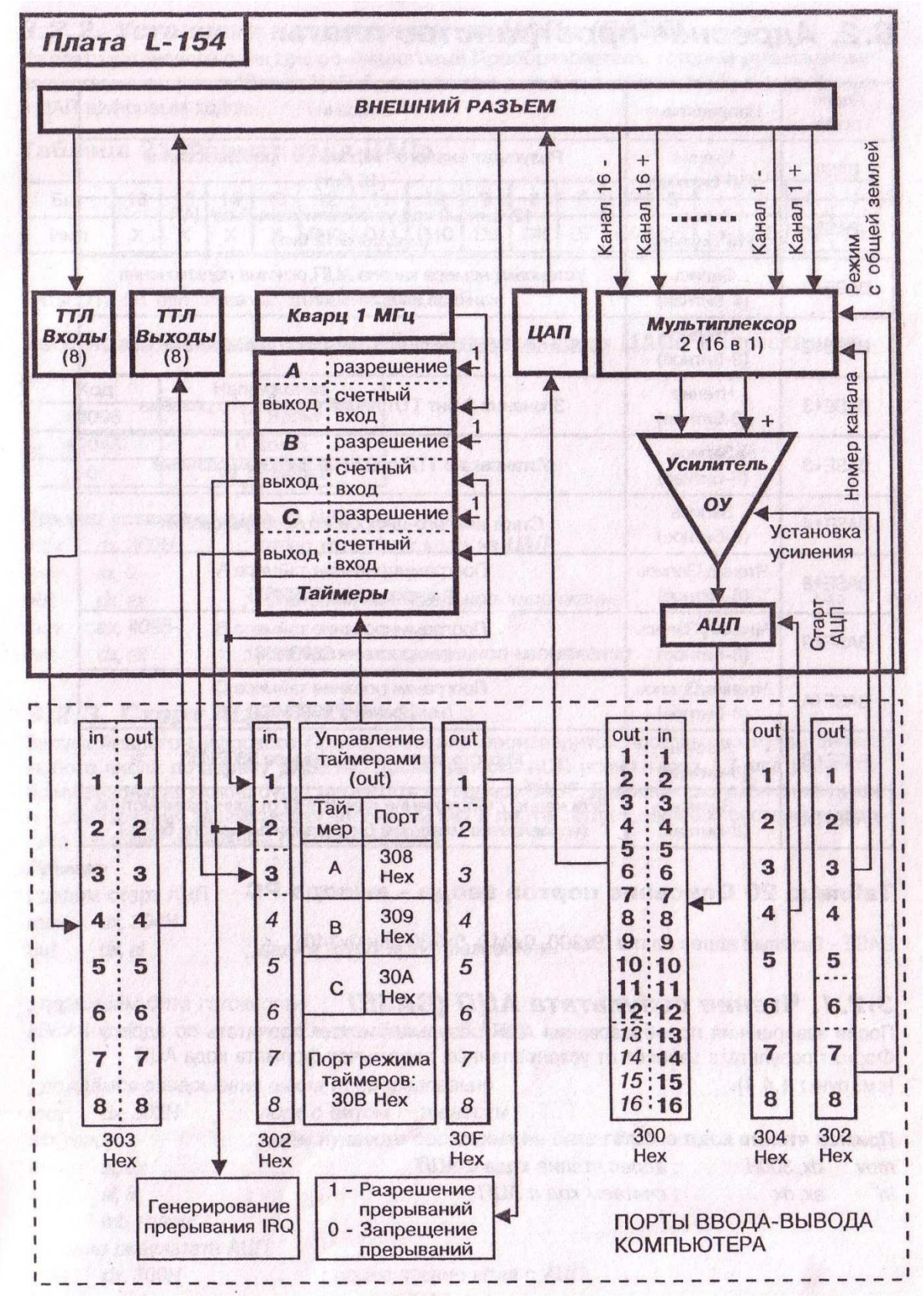

Плата L-154
ВНЕШНИЙ РАЗЪЕМ
Канал 16 -
Канал 16 +
Канал 1 -
Канал 1 +
Режим с общей землей
ТТЛ Входы (8)
ТТЛ Выходы (8)
Кварц 1 МГц
ЦАП
Мультиплексор 2 (16 в 1)
A разрешение
выход счетный вход
B разрешение
выход счетный вход
C разрешение
выход счетный вход
Таймеры
Усилитель ОУ
Номер канала
Установка усиления
АЦП
Старт АЦП
in out
303 Hex
in
302 Hex
Управление таймерами (out)
Тай-мер Порт
A 308 Hex
B 309 Hex
C 30A Hex
Порт режима таймеров: 30B Hex
out
30F Hex
out in
300 Hex
out
304 Hex
out
302 Hex
Генерирование прерывания IRQ
1 - Разрешение прерываний
0 - Запрещение прерываний
ПОРТЫ ВВОДА-ВЫВОДА КОМПЬЮТЕРА

Рис.4.1

### 4.2.3. Реализация платой L-154 режима реального времени

Плата L-154 позволяет производить ввод/вывод аналоговой информации в двух режимах:

- с использованием синхронизации от таймера;
- в режиме генерации прерываний IRQ.

Режим синхронизации от таймера является программным режимом, реализуемым библиотекой L-154drv из состава программного обеспечения платы. Он используется при анализе процессов на высоких частотах ввода (более 10 кГц). В этом режиме библиотекой накладываются ограничения ***снизу*** на интервал ввода (не более 65 мс), что не позволяет использовать его для управления "медленными" процессами.

Ввод по прерываниям, напротив, используется на "медленных" частотах ввода (до 10 кГц). Идея ввода в режиме генерации прерываний сводится к следующему. Плата генерирует прерывание в компьютер, в котором предварительно должен быть загружен драйвер-обработчик используемого прерывания. Драйвер-обработчик после каждого вызова выполняет какую-либо функцию - вводит информацию с АЦП в соответствующую программу обработки, выводит результаты обработки на ЦАП.

Ввод по прерываниям позволяет организовать выполнение двух задач - основной и фоновой. В основной задаче производится ввод/вывод аналоговой информации. В фоновой производятся вычисления (или иные действия) в асинхронном режиме, т.е. без синхронизации от таймера платы.

Ввод по прерываниям предусматривает выполнение следующих действий. С помощью перемычек на плате необходимо запрограммировать работу в требуемом режиме прерывания. Операция прерывания осуществляется вызовом функции INITINTR, входящей в состав библиотеки L-154 drv платы. После вызова этой функции плата генерирует прерывание с интервалом, указанным в качестве аргумента функции INITINTR. Драйвер-обработчик должен "позаботиться" о сбросе как контроллера прерываний компьютера, так и элементов управления прерыванием платы. После завершения использования платы в режиме генерирования прерываний необходимо вызвать функцию STOP_INTR библиотеки L-154 drv. Данная функция выключает режим генерации прерываний на плате и восстанавливает контроллер прерывания компьютера.

Следует отметить, что для реализации режима реального времени по методу прерывания сигналами таймера платы необходимо обеспечить освобождение одной из линий внешнего прерывания IRQ (IRQ3 ÷ IRQ6) от сигналов прерывания из других источников.

Необходимо также иметь в виду, что преобразование сигналов ЦАП можно реализовать в асинхронном режиме. При этом операции получения информации, реализацию алгоритма ее обработки и выполнение обратного преобразования результатов в аналоговую величину можно выполнить в одном и том же цикле работы системы управления.

### 4.2.4. Экспериментальные методы оценки точности и быстродействия устройств связи с объектом

В сопроводительных документах промышленных систем сбора данных и управления, как правило, указываются основные технические характеристики различных модулей УСО и условия их эксплуатации. Часто реальные условия работы УСО могут потребовать уточнения этих характеристик. Например, такое может случиться, если приходится использовать УСО в условиях повышенных помех или жестких требований соблюдения временных затрат при реализации режима реального времени.

Ниже описываются экспериментальные методы, позволяющие получить оценки точностных и временных характеристик УСО на месте их эксплуатации и без использования специальной дополнительной аппаратуры.

#### Оценка точности работы подсистем ввода/вывода аналоговой информации

Как правило, в промышленных УСО используются многоканальные АЦП и одноканальные ЦАП. В описываемом методе экспериментальной оценки точностных характеристик предполагается наличие как минимум двух каналов ввода аналоговой информации.

Один из каналов используется для подачи постоянного входного сигнала x ("прямой" сигнал), а другой канал - для подачи на вход АЦП сигнала, снимаемого с выхода ЦАП ("обратный" сигнал). При этом входной сигнал x, проходя через АЦП, преобразуется в цифровой эквивалент и в таком виде без изменения передается на вход ЦАП. С выхода ЦАП в виде аналогового сигнала он передается на вход второго канала АЦП и еще раз преобразуется в цифровую форму. Результаты как первого, так и второго преобразования запоминаются в памяти ЭВМ.

Точность работы подсистем как ввода, так и вывода оценивается вычислением среднеквадратических погрешностей. Для этого с целью получения представительной выборки предусматривается многократное его преобразование с последующим вычислением математического ожидания и дисперсии. Таким образом, в эксперименте используется метод кольца, внутри которого "вращается" испытуемый сигнал.

Если обозначить среднеквадратичные погрешности ввода и вывода информации соответственно через $\sigma_1$ и $\sigma_2$, а суммарную среднеквадратическую погрешность преобразования подсистем ввода/вывода по кольцу - $\sigma_\Sigma$ , то, при некоррелированности погрешностей $\sigma_1$ и $\sigma_2$, уравнение баланса ошибок можно записать в виде

$$\sigma_\Sigma^2 = 2\sigma_1^2 + \sigma_2^2.$$

При этом

$$\sigma_1 = \sqrt{D(x)} = \sqrt{\frac{1}{n-1}\sum_{1}^{n}[x(k) - \bar{x}]^2},$$

где x(k) – входной аналоговый сигнал для $t = kT_0$ ;

$T_0$ – период квантования;

n – объем выборки;

$\bar{x} = \frac{1}{n}\sum_{1}^{n} x(k)$ – математическое ожидание x.

Аналогичные зависимости можно получить и для сигнала, поступающего на вход второго канала. В результате оценку среднеквадратической погрешности канала вывода можно определить в виде

$$\sigma_2 = \sqrt{\left|\sigma_\Sigma^2 - 2\sigma_1^2\right|}.$$

Изменяя значения x на входе первого канала (АЦП), можно получить оценку точности работы подсистем ввода-вывода во всем диапазоне возможного изменения сигналов на входе АЦП и ЦАП.

Программа, реализующая описанный выше метод оценки точностных характеристик подсистем ввода-вывода, приведена в Приложении 1. Эта программа может использоваться для оценки точностных характеристик УСО на базе платы L-154 для IBM PC.

### Оценка быстродействия подсистемы ввода информации

Быстродействие является одной из важнейших характеристик УСО. Особенно оно актуально для многоканальных АЦП подсистем ввода информации, так как их реальное быстродействие существенно зависит от числа используемых каналов. В подсистемах вывода информации, как правило, используются одноканальные ЦАП и для оценки их реального быстродействия можно ориентироваться на данные, приводимые в технической документации на ЦАП подсистемы вывода. Поэтому ниже будет рассматриваться метод экспериментальной оценки быстродействия многоканальных АЦП в режиме прерываний от таймера платы.

Идея метода заключается в следующем. Пусть задан некоторый фиксированный интервал времени $T_c$. Пусть в течение этого времени выполняется в циклическом режиме некоторая вычислительная операция, для которой исходные данные задаются из программы. Зафиксируем количество операций этого типа, которое можно выполнить за время $T_c$. Обозначим их - N1. Тогда среднее время, затрачиваемое на выполнение этой вычислительной операции, будет $\tau_1 = T_c / N1$.

Следующий эксперимент проведем таким же образом, но исходную информацию для выполнения той же операции будем задавать через подсистему ввода/вывода.

Пусть в результате эксперимента за время $T_c$ мы зафиксировали N2 операций. Тогда можно записать $N2(\tau_1+\tau_2) = T_c$ ,
где $\tau_2$ - время, затрачиваемое на выполнение операций ввода.
В результате

$$\tau_2 = (T_c - N2 \cdot \tau_1)/N2 = (N1 - N2)\tau_1/N2 = \Delta T/N2,$$

где $\Delta T=\Delta N\tau_1$.

Для того чтобы части программы, оценивающие объем вычислительной работы без операций ввода/вывода и с ними, были идентичны, программу эксперимента необходимо компилировать с выключенной оптимизацией.

Программа, реализующая описанный выше метод оценки времени выполнения операций ввода/вывода, приведена в Приложении 2. В ней для $T_c$ =30 сек. выполняются определенные вычисления (1000 умножений) сначала без операций ввода через УСО, а затем - то же количество операций с вводом/выводом через УСО.

## 4.3. Платы ввода/вывода дискретных сигналов

Принцип построения и использование этих плат рассмотрим на примере плат PLC-720, PLCD-782, PLCD-785, программно и аппаратно совместимых с IBM PC. Эти платы изготавливаются фирмой Advantech Co, Ltd.

### Плата ввода дискретных сигналов PLCD-782

Платы этого типа предназначены для ввода в ЭВМ дискретных сигналов, характеризующих состояние конечных выключателей и других устройств, имеющих два устойчивых состояния. Плата рассчитана на прием дискретных сигналов с 16 датчиков. На плате имеются переключатели, с помощью которых на один и тот же контакт можно осуществлять ввод и потенциальных и оптоизолированных сигналов.

В платах этого типа выполняется преобразование сигнала, принимающего только два состояния, в информационное значение «1-0» и запись информационного значения в заданный разряд определенного входного регистра. Сигнал поступает с физического датчика и преобразуется с помощью микросхемы. Для сигналов типа “сухой контакт” входная цепь запитывается от источника напряжения, находящегося в устройстве управления. Входное напряжение в схему преобразования снимается с резистора, расположенного на плате приема дискретных сигналов. Для уменьшения помех (наводок) применяется оптронная развязка маломощных цепей системы управления от силовых цепей управляемого объекта.

Электрическая схема ввода одного дискретного сигнала приведена на рис. 4.2. Для ввода потенциальных сигналов с помощью перемычки замыкаются контакты 1 и 2 переключателей П1 и П2. При необходимости изолировать входную цепь от цепи ЭВМ замыкаются контакты 3 и 4 тех же переключателей.

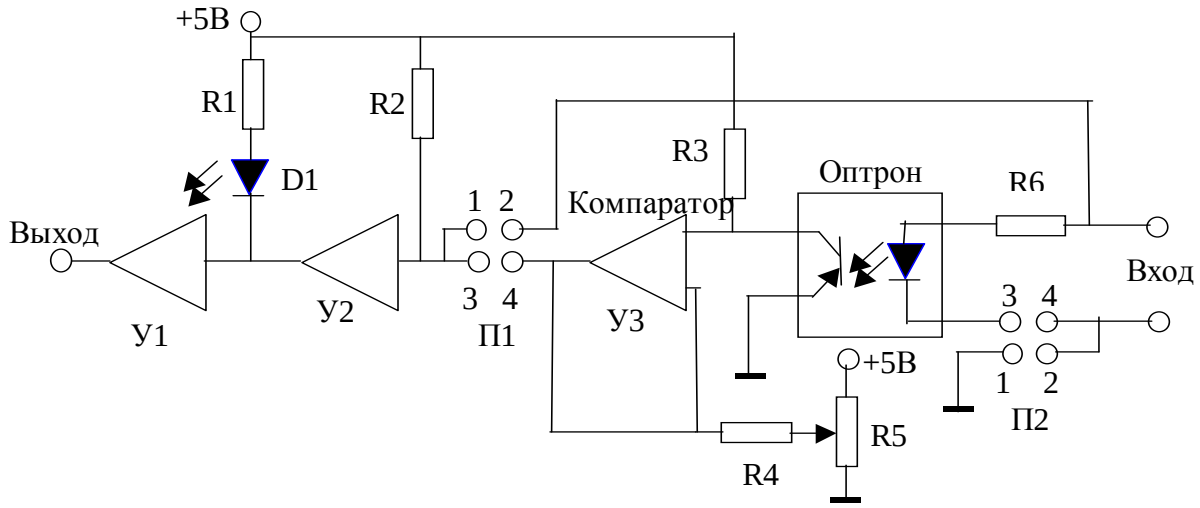



Рис. 4.2. Схема ввода дискретного сигнала

Таким образом, в состав платы входят схемы преобразования и записи информации во входные регистры и выходные регистры, откуда информация считывается в управляющую ЭВМ. (Обычно для ЭВМ эти регистры доступны только по считыванию информации).

**Плата формирования дискретных сигналов PLCD-785**

Эта плата предназначена для формирования управляющих воздействий на исполнительные механизмы (электромагниты, реле, пускатели и т.д.), информационное и контрольно-сигнальное оборудование объекта управления (электрические лампочки, ревуны, элементы мнемосхем и т.д.).

Общее количество выходных сигналов равно 16. С помощью электронных схем логические сигналы «0-1» преобразуются в физические электрические сигналы, уровень и мощность которых достаточны для срабатывания выходного реле. Это реле используется для гальванической развязки исполнительных силовых цепей от маломощных цепей устройства управления.

При этом необходимо предусмотреть меры для защиты выходных контактов реле от обгорания, которое может произойти при размыкании цепи, в которой имеется значительная индуктивность обмотки исполнительного механизма. Для этой цели параллельно выходным контактам подсоединяются искрогасящие RC-цепочки. Постоянная времени этих цепочек определяется двумя противоречивыми условиями: чем больше значение постоянной, тем меньше

перепад напряжения на разомкнутых контактах, но и тем больше запаздывание, вносимое этим каскадом. Следует иметь в виду, что последнее обстоятельство ведет к ухудшению динамических характеристик системы управления. Для каждого конкретного случая необходимо определять оптимальное значение этой постоянной.

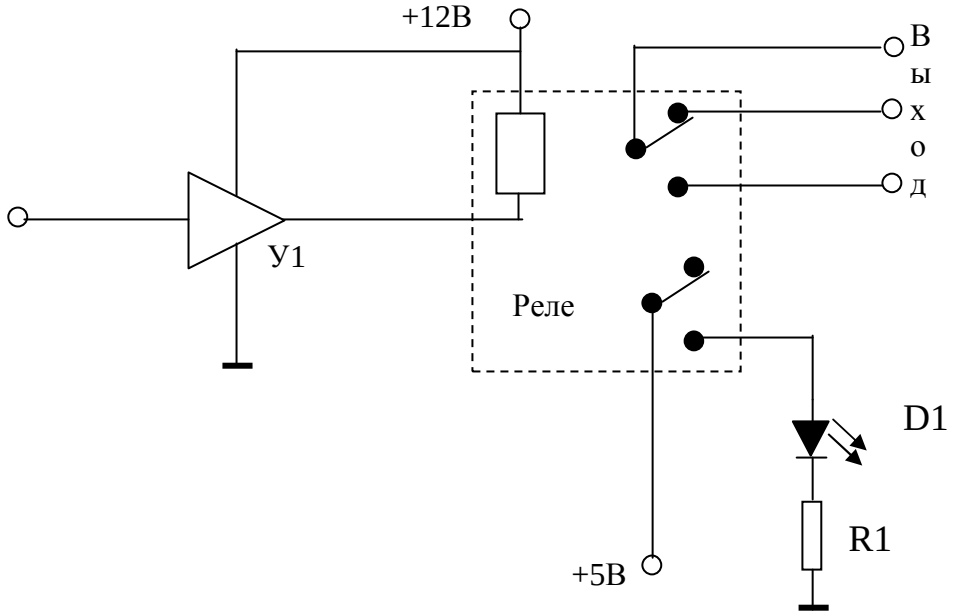



Рис. 4.3. Электрическая схема формирования выходного релейного сигнала.

Таким образом, в состав платы входят выходные регистры, в которые ЭВМ записывает коды сформированных управляющих воздействий, схемы преобразования логической управляющей информации в управляющие воздействия, выходные реле. (Выходные регистры доступны для ЭВМ только по записи). Электрическая схема формирования одного выходного релейного сигнала приведена на рис. 4.3.

**Коммутирующая плата PLC-720**

Эта плата предназначена для подсоединения с помощью плоских кабелей одной или двух плат PLCD-782 и одной или двух плат PLCD-785
к IBM PC. На плате расположен переключатель, с помощью которого устанавливается базовый адрес регистров плат ввода/вывода в адресном пространстве IBM PC. Помимо этого, в плате PLC-720 имеются три счетчика, которые при работе могут аппаратно и программно настраиваться и как счетчики событий, и как таймеры.

## 4.4. Платы управления приводами МС-3628/МС-3629

Плата МС-3628 (МС-3629) предназначена для управления 3-х координатным приводом, оборудованным инкрементными (например, фотоимпульсными) датчиками положения и относится к числу плат, в состав которых включены специализированные процессоры. Она программно и аппаратно совместима с процессорами Intel и может быть использована для построения устройств управления на базе IBM PC. Плата выпускается фирмой Omnitech Robotics.

В состав платы входят:

- два регистра для приема дискретных сигналов (8 сигналов уровня TTL и 8 гальванически изолированных сигналов),
- один регистр для выдачи дискретных сигналов (8 сигналов типа открытый коллектор),
- 3 специализированных процессора LM-628 для управления тремя координатными приводами,
- три 12-ти разрядных ЦАПа для формирования управляющих напряжений, подаваемых на усилители мощности координатных приводов.

(Модификация платы МС-3629 предназначена для управления координатными приводами с помощью ШИМ - усилителей).

Процессор LM-628 обладает широкими аппаратными и вычислительными возможностями. В его составе имеется интерфейс для связи с инкрементным фото-импульсным датчиком (ФИД) положения. В LM-628 аппаратными средствами повышается в 4 раза точность ФИД за счет того, что подсчитываются не импульсы с ФИД, а фронты импульсов прямой и смещенной последовательностей. Максимальная частота принимаемой последовательности импульсов равна 187,5 кГц. На плате имеется переключатель, который позволяет использовать датчики как с простыми, так и с дифференциальными выходными сигналами.

Процессор LM-628 может по командам от центрального процессора (ЦП) программно настраиваться на один из двух режимов работы: режим вывода рабочего органа в заданную точку или режим движения рабочего органа с заданной скоростью (режим стабилизации скорости). Разгон и торможение происходят с заданным ускорением автоматически.

Исходными данными для управления в первом режиме являются координаты заданной точки, заданные значения скорости и ускорения. Во втором режиме задаются только скорость и ускорение. Помимо этого, для каждого из режимов работы задаются допустимые пределы изменения сигнала рассогласования и координаты запретной зоны, в которой не может находиться рабочий орган. Все эти значения по командам поступают в LM-628 из центрального процессора ЭВМ. Если сигнал рассогласования превышает предельное значение или рабочий орган выходит в запретную зону, процессор LM-628 посылает в центральный процессор

сигнал прерывания, сопровождая его словом состояния, по которому центральный процессор может определить тип и причину прерывания.

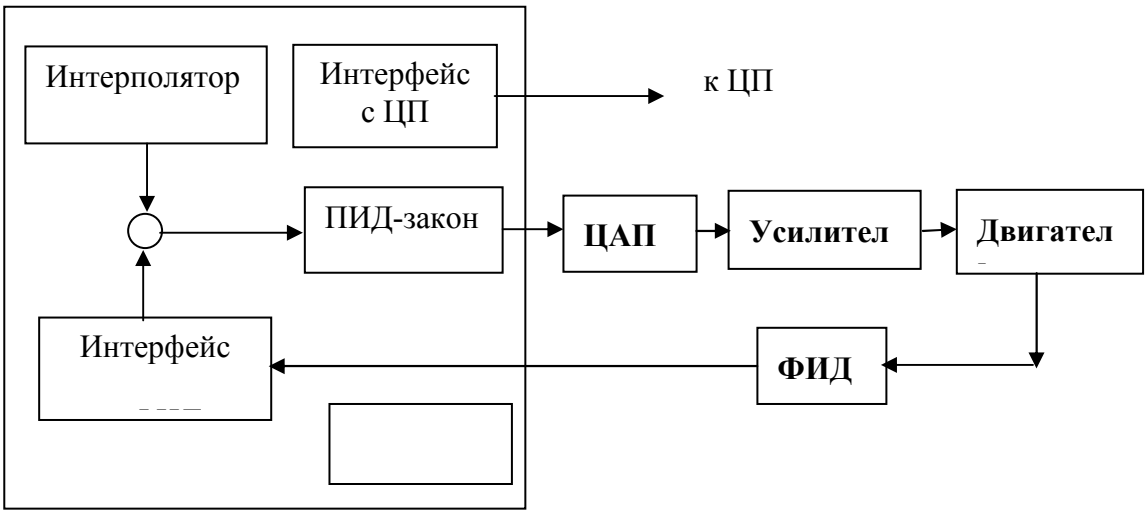



Рис. 4.4. Схема включения приводного процессора LM-628

Период смены управляющих воздействий составляет 341 мкс. С этим периодом в процессоре LM-628 вычисляется заданное значение координаты, куда должен переместиться рабочий орган (расчет координат промежуточных точек – интерполяция), а по показаниям ФИД определяется его истинное положение. В общем случае ПИД-закона управления вычисленное управляющее воздействие может иметь три составляющих:

- пропорциональную сигналу рассогласования (разности между заданным и истинным значениями координат),
- пропорциональную интегралу от сигнала рассогласования
- пропорциональную производной от сигнала рассогласования.

На различных участках управления значения коэффициентов пропорциональности могут программно изменяться. Они вводятся в LM-628 из центрального процессора. Также программно может изменяться и предел интегрирования, т.е. количество значений сигнала рассогласования, которое учитывается в интегральной составляющей. Значения координаты, скорости и ускорения имеют 32 двоичных разряда, коэффициенты ПИД - закона управления – 16 двоичных разрядов.

Управляющее воздействие может иметь либо 8, либо 12 двоичных разрядов. (Для LM-629 при использовании ШИМ - усилителя управляющее воздействие имеет 8 двоичных разрядов.)

Типовая схема включения приводного процессора LM-628 приведена на рис. 4.4.

Связь LM-628 с ЦП (рис. 4.5) осуществляется по восьмиразрядной шине данных и адресной шине с использованием сигналов записи, чтения, начальной

установки и прерывания. Сигнал по адресной шине поступает через дешифратор адреса, расположенный на плате МС-3628.

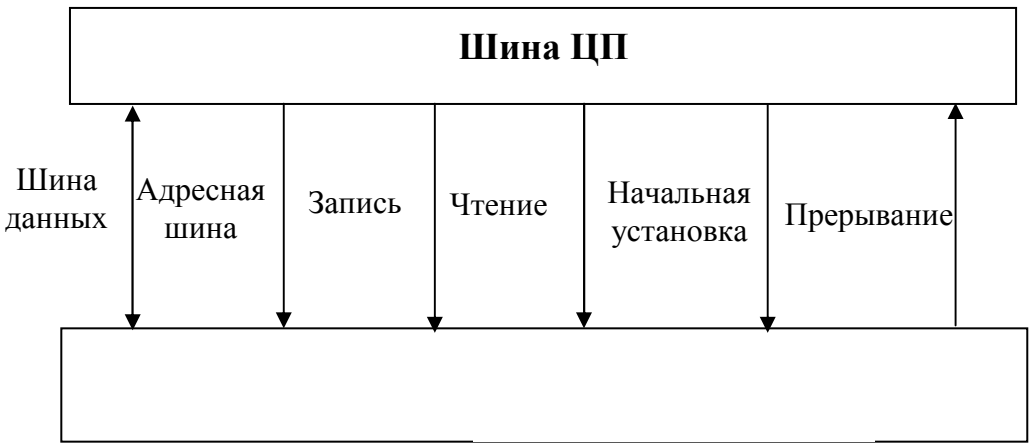



Рис. 4.5. Интерфейс между LM-628 и центральным процессором

Обмен между LM-628 и ЦП осуществляется по специальным командам. Эти команды можно разделить на 4 группы:

- команды установки: начальная установка LM-628, установка разрядности (8 или 12) управляющего воздействия, которая зависит от разрядности ЦАП;
- команды прерывания: запрет и разрешение прерывания и команды, соответствующие различным признакам прерывания;
- команды записи в LM-628 исходных данных для управления и коэффициентов ПИД - закона управления;
- команды чтения слова состояния LM-628, действительных и расчетных значений координаты и скорости.

Процессор LM-629 отличается от LM-628 только тем, что на его выходе формируется сигнал переменной скважности для управления ключами ШИМ - усилителя. Длина слова, формирующего скважность, составляет 8 двоичных разрядов.

На плате МС-3628 имеются переключатели, с помощью которых осуществляется привязка ее регистров к адресному полю центрального процессора. Адреса всех регистров МС-3628 расположены последовательно. Положение переключателей определяет базовый (начальный) адрес в адресном поле внешних регистров ЦП. С помощью второго переключателя определяется номер входа в ЦП, на который подается сигнал прерывания
с платы МС-3628. Прерывание ЦП может вызвать один из шести сигналов, которые соединены по схеме ИЛИ. Три сигнала формируются тремя приводными процессорами LM-628 (LM-629) по причинам, которые были указаны выше. Три других сигнала являются сигналами от дискретных датчиков оборудования (два

оптоизолированных сигнала и один сигнал уровня TTL). Конкретную причину прерывания ЦП определяет по слову состояния платы МС-3628.

Программирование платы МС-3628 может осуществляться тремя способами: с использованием языка BASIC, дополненного специальными командами; в среде Windows на одном из базовых языков этой среды с использованием специально разработанной библиотеки прикладных программ; на языке С++. Для последнего случая также разработан специальный компилятор и библиотека прикладных программ.

## 4.5. Особенности программного обеспечения компьютерных систем управления на базе ПЭВМ

В программном обеспечении компьютерной системы управления можно выделить две части: системное программное обеспечение (СПО) и прикладное программное обеспечение. Деление это условно и различные авторы часто проводят разделение программного обеспечения на составные части по-разному.

Системное программное обеспечение разрабатывается фирмой–изготовителем ЭВМ, на базе которой построена система управления, и чаще всего поставляется вместе с аппаратурой. Конфигурация СПО и его состав зависят от сложности объекта управления и целей и задач, поставленных перед системой управления. Характерной особенностью СПО для компьютерных систем управления является наличие в его составе операционной системы реального времени или средств внешней эмуляции обработки информации в режиме реального времени.

Как известно, любое СПО практически инвариантно к объекту управления и определяется вычислительными ресурсами системы управления (компьютера). В противоположность СПО прикладное программное обеспечение, реализующее алгоритмы управления, полностью определяется объектом управления. Поэтому говорить о составе прикладного программного обеспечения безотносительно к управляемому объекту бессмысленно. Более или менее универсальным является только программный модуль, формирующий аналоговые управляющие воздействия по выбранному П-, ПИ-, ПД- или ПИД - алгоритму.

В заключение сформулируем несколько требований к программным модулям, работающим в реальном масштабе времени. При разработке и программной реализации алгоритмов, рассчитывающих управляющие воздействия, приходится искать компромисс между точностью вычислений и временем реализации алгоритма, так как чем выше точность алгоритма, тем больше процессорного времени требуется для его реализации. Другим важным вопросом, который приходится решать разработчику прикладного программного обеспечения компьютерной системы управления, является распределение приоритетов между различными задачами, заявки на выполнение которых могут поступать в процессор одновременно. Общее количество таких задач при управлении сложными объектами может достигать нескольких десятков.

Имеется несколько общих правил при расстановке приоритетов и организации прерываний в компьютерных системах управления.

Первое правило: наивысший приоритет присваивается задачам, реализация которых определяет показатели качества работы объекта управления. Это, прежде всего, задачи, невыполнение которых ведет к аварийной ситуации или к необратимой потере информации, а также задачи, в результате выполнения которых вырабатываются управляющие воздействия на динамические устройства с малой инерционностью.

Второе правило заключается в том, что выполнение программных модулей, время реализации которых сравнительно мало (меньше или сравнимо со временем формирования управляющих воздействий на динамические устройства) не прерывается даже при поступлении заявки на более приоритетную задачу.

Во многих практических случаях приходится считаться с ограниченностью ресурсов устройств управления, что накладывает жесткие ограничения на суммарный объем программного обеспечения и особенно на время реализации программных модулей, работающих в масштабе реального времени. Это относится как к прикладному, так и к системному программному обеспечению и должно учитываться разработчиками ПО.

# ПРИЛОЖЕНИЯ

Приложение 1

```
// TPVV.C -Исходный текст программы оценки точности работы подсистемы
// ввода/вывода. Разработал Д.Кухарь
#include <stdio.h>
#include <math.h>
#include "inp.h"
#include "results.h"

double M1, M2, D1, D2, D3, SK1, SK3;
double sqr(double x);
int WriteFunc(FILE * f, int Mode);
double Out(void);
int SaveResultsToFile(char * FName);
void main(void)
{
      int   i, t;
      int   N1;
      InitializeFromFile("TPVV.DAT", Out);
      printf("Нажмите кнопку ПУСК АВК\n");
      WaitForStart();
      N1 = N();
      // Вычисляем мат.ожидание
      M1 = 0;
      M2 = 0;
      for (t = 0; t < N1; t++)
      {
            M1 += GetInpData(Channels[0], t);
            M2 += GetInpData(Channels[1], t);
      }
      M1 = M1 / N1;
      M2 = M2 / N1;
      // Вычисляем дисперсию
      D1 = 0;
      D2 = 0;
      for (t = 0; t < N1; t++)
      {
            D1 += sqr(GetInpData(Channels[0], t) - M1);
            D2 += sqr(GetInpData(Channels[1], t) - M2);
      }
      D1 = D1/(N1-1);
      D2 = D2/(N1-1);
      D3 = D2-2*D1;
      SK1 = sqrt(D1);
      SK3 = sqrt(D3);
      SaveResultsEx("con", WriteFunc);
      SaveResultsToFile("TPVV.OUT");
      Finalize();
}
double Out(void)
{
      return GetInput(Channtls[0]);
}
```

```
int SaveResultsToFile(char * FName)
{
      FILE  *f;
      f = fopen(FName, "wt");
      if (f == NULL) return FALSE;
      if (!WriteFunc(f, WRITEHEADER)) goto error;
      if (!WriteFunc(f, WRITEFOOTER)) goto error;
      if (fclose(f) == EOF) return FALSE;
      return TRUE;
      error:
      fclose(f);
      remove(FName);
      return FALSE;
}
int WriteFunc(FILE * f, int Mode)
{
      int res;
      switch (Mode)
      {
            case WRITEHEADER:
                  // Записываем заголовок в файле
                  res = fprintf(f, "Оценка точности работы подсистем\n"
                              "ввода/вывода аналоговой информации\n\n");
                  if (res == EOF) return FALSE;
                  return TRUE;
            case WRITEFOOTER:
                  // Записываем результат обработки
                  res = fprintf(f, "\n\n M1 = %10.6f\n M2 = %10.6f\n "
                            "D1 = %10.6f\n D2 = %10.6f\n D3 = %10.6f\n "
                            "SK1 = %10.6f\n SK3 = %10.6f\n", M1, M2, "
                            "D1, D2, D3, SK1, SK3);
                  if (res == EOF) return FALSE;
                  return TRUE;
            default:
                  return TRUE;
      }
}
double sqr(double x)
{
      return x * x;
}
```

```
// SPTEST.C - Программа оценки временных затрат на ввод/вывод
// аналоговой информации. Разработал Д.Кухарь

#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include "inp.h"

int   Chls[] = {1, 2, 3, 4};  // каналы для ввода
unsigned long *time;          // счетчик таймера ПК
int TestTime = 30;            // время тестирования в секундах

double Dummy(void)            // Dummy - возвращает 1 для вывода на ЦАП
{
      return GetInput(Chls[0]+GetInput(Chls[1])+
            GetInput(Chls[2])+GetInput(Chls[3]);
}
unsigned long PerformCalculation(unsigned long EndTime)
{
      int i, j;                     // переменные циклов
      unsigned long n = 0;
      while (*time < EndTime)
      {
            n++;
            for ( i = 0; i < 1000; i++)
                  j = i*3;
      }
      return n;
}
unsigned long GetStartTime(void)
{
      unsigned long StartTime;

      StartTime = *time;
      while (StartTime == *time);
      StartTime = *time;
      return StartTime;
}
void SaveResults(FILE *f, unsigned long n, unsigned long c)
{
      unsigned long diff = n - c;
      double     Ctime;

      fprintf(f,"Тестирование быстродействия подсистемы"
                "ввода/вывода\n\n");
      fprintf(f,"Без работы подсистемы ввода/вывода %ld циклов.\n", n);
      fprintf(f,"С работой подсистемы ввода/вывода %ld циклов.\n",c);
      Ctime = (30.0/n)*diff/TIME();
      fprintf(f, "На каждый опрос по 4 каналам приходится %f c, \n"
                 "частота = %f опросов в секунду.\n", (CTime, 1/Ctime);
}
void main()
{
      unsigned long c, n;  // число циклов и разница между ними
      int TickCount = (int) (TestTime * 18.2);
```

```
    FILE * f;

    // Обеспечиваем доступ к текущему значению таймера ПК
    time = (long unsigned *) MK_FP(0x0040, 0x006C);

    // измеряем число циклов ( без ввода/вывода )
    printf("Тестирование без подсистемы ввода/вывода"
           "(%d секунд)\n", TestTime);
    n = PerformCalculation(GetStartTime() + TickCount);

    // измеряем число циклов ( с вводом/выводом)
    if (!RTCInitialize(Chls, 4, 10, 4000, Dummy))
    {
         printf("Не хватает памяти\n");
         exit(1);
    }
    printf("Тестирование с подсистемой ввода/вывода"
           "(%d секунд)\n\n", TestTime);
    Start();
    c = PerformCalculation(GetStartTime() + TickCount);
    Stop();

    // Вывод результатов тестирования на экран
    SaveResults(stdout, n, c);

    // Вывод результатов тестирования в файл sptest.out
    f = fopen("sptest.out", "wt");
    if (f == NULL)
    {
         printf("Ошибка создания sptest.out\n");
         exit(1);
    }
    SaveResults(f, n, c);
    fclose(f);

    // Выход
    exit(0);
}
```

## Литература


1. Изерман Р. Цифровые системы управления. – М.: Мир, 1984. - 541с.
2. Острем К., Виттенберг Б. Системы управления с ЭВМ: - М.: Мир, 1987.-480с.
3. Дроздов Н.М., Мирошник И.В., Скорубский В.И. Системы автоматического управления с микро-ЭВМ. - Л.: Машиностроение, 1989. - 280с.
4. Месарович М., Мако Д., Такахара И. Теория иерархических многоуровневых систем. - М,: Мир, 1973. - 342с.
5. Строганов Р.П. Управляющие машины и их применение. - М.: Высш. шк., 1986.-240с.
6. Строганов Р.П., Давыдов В.Г. Васильев В.П. Алгоритмическое обеспечение задач управления в АСУ ТП. Учебное пособие. - Л.: ЛПИ, 1989 - 92с.
7. Применение ЭВМ в системах управления: Методические указания / СПбГТУ; Сост. Р.П.Строганов, 1993.- 60с.
8. Прикладные нечеткие системы / под редакцией Терано Т., Асаи К., Сугено М.-М.: Мир, 1993. - 386с.
9. Коневцова О.В. Система автоматического управления технологическими процессами при нечеткой исходной информации. Дипл. работа; СПбГТУ, 1994.
10. Андреев Ю.Н. Управление конечномерными линейными объектами. - М.: Наука, 1976.-424с.
11. Кузнецов А.В., Холод Н.И., Костевич Л.С. Руководство к решению задач по математическому программированию. - Минск: Вышэйшая школа, 1978.
12. Строганов Р.П. и др. Иерархическое управление внеплановыми нагрузками в энергообъединениях //Известия АН СССР, Энергетика и транспорт, №5, 1984.
13. Строганов Р.П., Нестеров С.А., Ярмийчук В.Д. Необходимые условия координируемости сетевых объектов по методу прогнозирования взаимодействий //Труды СПбГТУ № 442, 1992.
14. Табак Д., Куо Б. Оптимальное управление и математическое программирование. - М.:- Наука, 1978.
15. Воронов А.А. Введение в динамику сложных управляемых систем. - М.: Наука, 1985.-352с.
16. Леонтьев А.Г. Микропроцессорные электромеханические системы. Учебное пособие. -СПб.: СПбГТУ, 1998.-110с.
17. Matlab. Fuzzy Logic Toolbox Users Guide. The MathWorks, Inc., 1998.