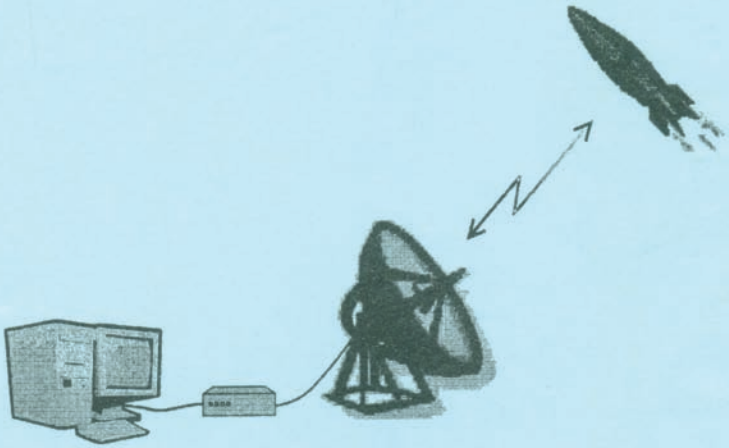
Глонь О.В.

Дубовой В.М.

Мітюшкін Ю.І.

КОМП'ЮТЕРИЗОВАНІ СИСТЕМИ КЕРУВАННЯ



Міністерство освіти і науки України
Вінницький національний технічний університет

О.В. Глонь
В.М. Дубовой
Ю.І. Мітюшкін

КОМП'ЮТЕРИЗОВАНІ СИСТЕМИ КЕРУВАННЯ

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напряму підготовки 0914 – "Комп'ютеризовані системи, автоматика і управління" всіх спеціальностей. Протокол № 10 від 26 травня 2005 р.

Вінниця ВНТУ 2005

УДК 681.518
Г 54

Рецензенти:

Р.Н. Квєтний, доктор технічних наук професор
В.О. Поджаренко, доктор технічних наук професор
В.М. Лисогор, доктор технічних наук професор

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Глонь О.В., Дубовой В.М., Мітюшкін Ю.І.

Г54 Комп'ютеризовані системи керування. Навчальний посібник. – Вінниця: ВНТУ, 2005. – 157с.

В посібнику узагальнюються питання моделювання процесів керування, а також практичної реалізації відповідних систем, що працюють під керуванням сучасних засобів комп'ютерної техніки. Посібник розроблений у відповідності з планом кафедри та програмами дисциплін "Комп'ютерні системи керування", "Багатомашинні комплекси та мережі ЕОМ", "Вступ в теорію систем".

УДК 681.518

ЗМІСТ

ВСТУП

Мета будь-якого виробництва – забезпечення багатогранних матеріальних та духовних потреб людства. Людство зростає чисельно та розвивається інтелектуально у геометричній прогресії. Зростають за обсягом, складністю та якістю його потреби. Відповідно сучасне виробництво в усіх сферах характеризується високою складністю та різноманіттям технологічних процесів. Воно вже неможливе без створення систем керування, які забезпечують його ефективність, надійність та безпеку. Системи керування повинні відповідати „принципу адекватної складності" Ешлі, який стверджує, що складність системи керування повинна бути не меншою за складність керованого процесу. Практично необмежене зростання складності керованих процесів викликало необхідність застосування в системах керування засобів, складність яких теж є теоретично необмеженою. Такими засобами є комп'ютери. Їх складність зумовлена нескінченним різноманіттям алгоритмів, під керуванням яких вони працюють. Але незважаючи на це потенційно нескінченне різноманіття, комп'ютеризовані системи керування (КСК) створюються на основі досить обмеженого числа загальних принципів. Ознайомленню з такими загальними принципами і присвячений цей посібник.

Предметом розгляду посібника є узагальнення переважної більшості спеціальних курсів, які викладаються студентам відповідної спеціальності. Всі ці курси можна розділити на три цикли: системотехніка КСК, застосування комп'ютерів в КСК, технічні засоби КСК. У посібнику не повторюється матеріал інших дисциплін, але він містить відповідні розділи, метою яких є з'ясування ролі окремих дисциплін у загальній системі знань про КСК.

Ще однією метою посібника є узагальнена точка зору на КСК незалежно від їх призначення, а саме для керування технологічними процесами, науковими та виробничими дослідженнями, соціальними та організаційними процесами. Така точка зору ґрунтується на уявленні про єдину структуру задачі керування, складовими якої є:

- аналіз об'єкта керування;
- мета та критерії керування;
- обмеження на параметри та характеристики системи;
- алгоритм (закон) керування;
- технічні засоби керування;
- взаємодія з персоналом.

Глибоке розуміння всіх зазначених проблем буде сприяти розвитку інженерного підходу до розв'язання складних проблем керування у різноманітних сферах і галузях.

Посібник призначений для підготовки бакалаврів і магістрів спеціальності "Системи управління і автоматики", а також буде корисним студентам і фахівцям інших споріднених спеціальностей.

1. ЗАГАЛЬНА ХАРАКТЕРИСТИКА КСК

Комп'ютерні системи керування можуть бути як простими, так і дуже складними комплексами, які включають програмно-алгоритмічні засоби, людей, які визначають мету та приймають стратегічні рішення. Але загальні принципи побудови таких систем досить близькі.

1.1 Основні поняття КСК

При описанні та дослідженні КСК звичайно використовуються певні поширені поняття теорії та техніки керування.

Система – це сукупність взаємопов'язаних між собою складових частин, яка характеризується спільною метою функціонування.

Керування – це процес цілеспрямованої зміни характеристик об'єкта керування.

Елемент системи – це проста складова частина системи.

Елементи системи пов'язані між собою зв'язками. *Зв'язки* між елементами можуть бути:

- фізичні;
- логічні;
- інформаційні.

Сукупність елементів системи та зв'язків між ними утворюють структуру системи.

Основними елементами системи керування є *об'єкт керування, керуюча частина* і *людина-оператор*, яка задає *мету та критерії керування, обмеження* на параметри та характеристики системи, *алгоритм (закон) керування.*

Системи керування характеризуються у статичному режимі структурою зв'язків та властивостями її елементів. У динамічному режимі система характеризується множиною станів.

Стан системи – це сукупність значень її параметрів та характеристик у конкретний момент часу.

Послідовність переходу системи з одного стану в інший називається *процесом*. В системах керування всі процеси визначається взаємодією процесу керування з процесом у об'єкті керування.

1.2 Узагальнена структурна схема КСК

Відповідно до виділених основних понять узагальнену структуру КСК можна представити у структурному і функціональному аспектах як

взаємодію трьох основних складових КСК та процесів в них, як показано на рис. 1.1.

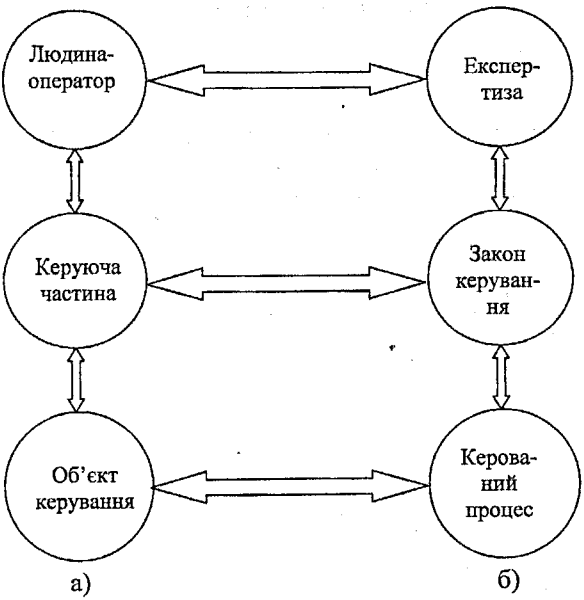


Рис. 1.1. Узагальнене представлення СК:
а) – основні структурні елементи; б) – відповідні функції

У комп'ютеризованих СК керуюча частина базується на використанні комп'ютера або контролера. Якщо виділити його з керуючої частини, то узагальнена структура буде мати вигляд, як на рис. 1.2. Розглянемо роль основних елементів КСК.

Характеристики і властивості КСК визначаються в першу чергу особливостями об'єкта керування. Отже відповідно до типу об'єкта КСК можна розділити на три основні категорії:

- технічні системи (ОК є машини, апарати, технологічні лінії тощо);
- виробничі системи (ОК є підприємство, підрозділ підприємства);
- організаційні системи (ОК є установи, суспільні організації, військові підрозділи тощо).

Керовані процеси в ОК можна розділити на:

- матеріальні, енергетичні, інформаційні. Відповідно до цього визначається фізичний принцип дії виконавчої частини. Матеріальні процеси відбуваються переважно у верстатах, технологічних лініях, хімічних реакторах тощо. Енергетичні процеси пов'язані з отриманням, перетворенням,

транспортуванням та використанням енергії. Вони відбуваються переважно в енергосистемах, на транспорті тощо. Інформаційні процеси передбачають отримання, передавання, обробку, зберігання та використання інформації. Вони відбуваються переважно в організаційних системах. Найскладнішим з точки зору керованих процесів є виробничі системи. В них найчастіше відбуваються і потребують керування всі три типи процесів;

- безперервної, дискретної, циклічної дії. Характер розвитку керованого процесу у часі безпосередньо впливає на закон керування.

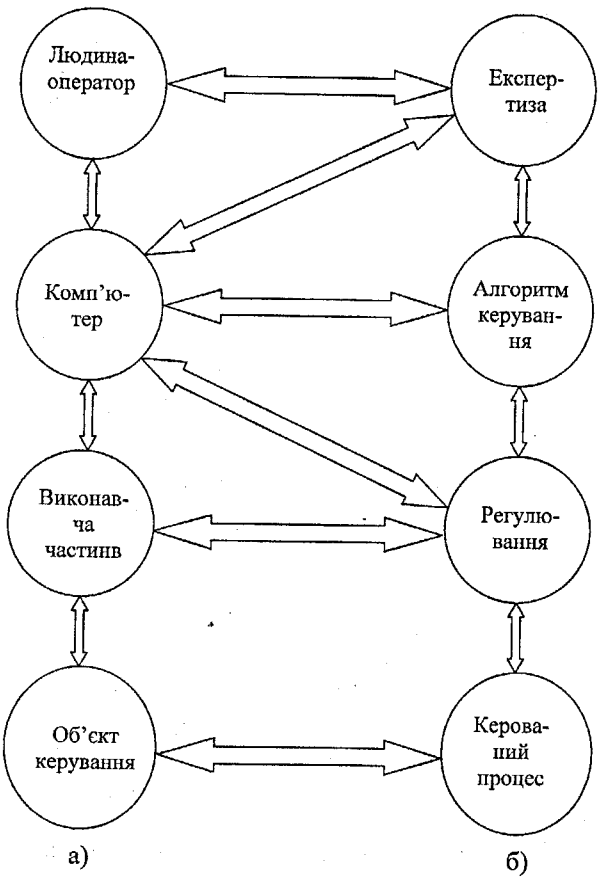


Рис. 1.2. Узагальнене представлення комп'ютеризованої СК:
а) – структурне; б) – функціональне

Роль людини-оператора в КСК теж залежить від характеру об'єкта керування. Переважно на неї покладається розв'язання задач, що важко

формалізуються, прийняття рішень в умовах невизначеності; стратегічне керування. Частина таких задач максимальна в організаційних системах і мінімальна в технічних системах.

1.3 Принципи функціонування комп'ютеризованих систем керування

У відповідності до трьох головних напрямків застосування КСК (керування технічними системами та технологічними процесами, керування підприємствами та виробництвами, керування організаційними системами) в основі побудови структурних схем КСК лежать різні початкові дані і відповідно будуються різні структурні схеми. При цьому в усіх схемах використовуються два головних принципи:

- принцип централізованого керування;
- принцип розподіленого керування.

При централізованому керуванні весь алгоритм керування реалізується на одному комп'ютері (рис. 1.3,а).

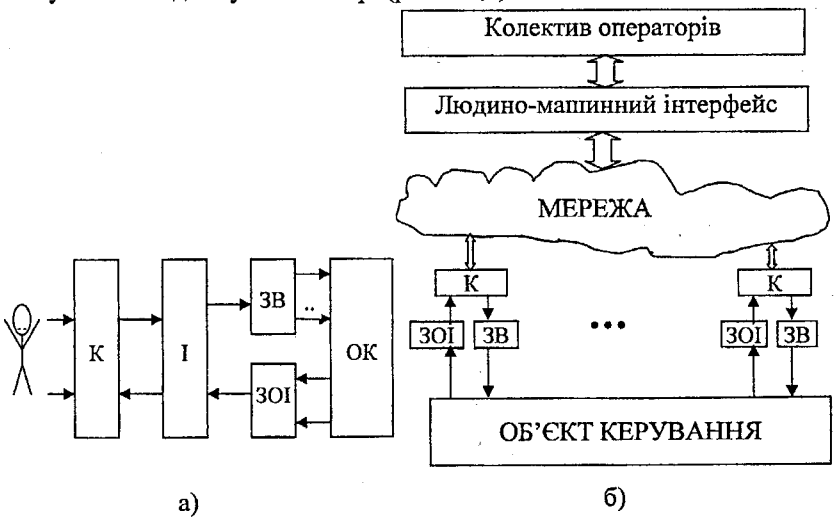


Рис. 1.3. КСК на основі базових принципів керування:
а – централізованого, б – розподіленого.
ОК – об'єкт керування; ЗВ – засоби впливу на об'єкт; ЗОІ – засоби отримання інформації; І – інтерфейс; К – комп'ютер (контролер)

Найчастіше КСК бувають розподіленими і містять не один, а багато комп'ютерів (контролерів). У цьому випадку структурна схема КСК тісно

пов'язана зі структурою комп'ютерної мережі, яка об'єднує всі комп'ютери (контролери) системи (рис.1.3,б).

Розподілені системи керування можуть бути побудовані за трьома принципами :

- ієрархічний принцип – використовується у випадках, коли необхідна чітка централізація усіх процесів;
- лінійний принцип – характеризується мінімальною довжиною зв'язків і високою швидкодією;
- кільцевий принцип – характеризується простотою реалізації і високою надійністю.

Відповідні структурні схеми зображені на рис. 1.4.

В основі побудови структурної схеми системи керування лежить система задач, для розв'язання яких призначена КСК.

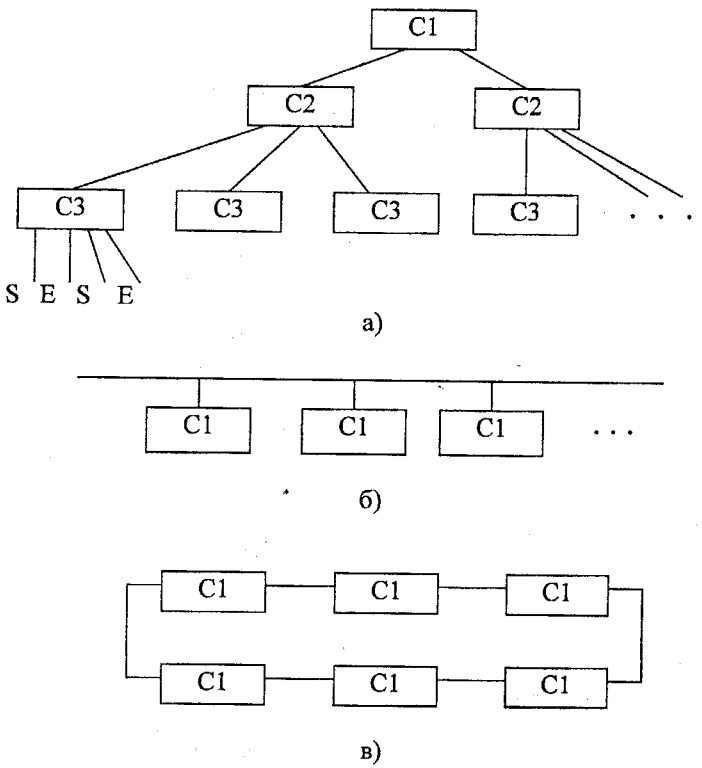


Рис. 1.4. Базові архітектури розподілених систем
а) ієрархічна, б) лінійна, в) кільцева

КСК технічними системами і технологічними процесами

Аналіз задач, що вирішуються комп'ютером в автоматичних системах, дозволяє виділити ряд основних функцій, які виконує комп'ютер. Ці функції можуть бути розділені на групи, які представлені у табл. 1.1. Функціями комп'ютера, що пов'язані з прийомом та попередньою обробкою сигналів сенсора, є лінеаризація амплітудної характеристики сенсора і масштабування сигналів. Для реалізації цих функцій використовуються системи з керуванням від центрального процесора і розподілені системи збору даних, засновані на периферійних мікропроцесорах.

Лінеаризація амплітудних характеристик сенсорів і масштабування сигналів звичайно поєднуються з їх аналого-цифровим перетворенням. Виникаючі в процесі роботи АЦП похибки автоматично коректуються з використанням програмних засобів комп'ютера.

На комп'ютер покладаються також функції стиснення інформації методами адаптивної телеметрії та їх завадостійке кодування. Така обробка інформації проводиться, як правило, в системах автоматичного керування з лініями зв'язку, які мають обмежену пропускну здатність та відкриті для проникнення завад.

Використання в сенсорах контролерів дозволяє надати їм деякі "інтелектуальні" функції, зв'язані з адаптацією сенсорів до мінливих умов спостерігання сигналів з метою їх завадостійкої фільтрації, а також з адаптивним стисненням даних, розпізнаванням зображень і класифікацією сигналів, масштабними та іншими перетвореннями складних багатовимірних зображень.

Ці та багато інших функцій виконують комп'ютери також в процесі перетворення сигналів та формування керуючих впливів. Приймання сигналів, які поступають відкритими лініями зв'язку, потребує їх фільтрації та відновлення, вимірювання інформативних параметрів, виявлення спотворень. Важливою функцією комп'ютера, що пов'язана з багатьма застосуваннями, є ідентифікація сигналів, прогнозування їх параметрів.

Комп'ютер надає відмінні можливості для організації в системах керування самоконтролю і самовідновлення втрачених функцій. Оснащені спеціальними програмами комп'ютери здатні виконувати сервісні функції, пов'язані з вирішенням задач обслуговуючого персоналу.

Приведений перелік функцій комп'ютера є далеко неповним. Функції комп'ютерів постійно розширюється і ускладнюється, що забезпечує зростання показників ефективності роботи автоматичних систем.

В основі побудови структурної схеми КСК технологічним процесом лежить структура технологічного процесу, перелік контрольованих та керованих параметрів процесу, вимоги до розташування засобів контролю та керування у просторі.

Таблиця 1.1

Основні функції комп'ютера

Елемент КСК	Функції комп'ютера
1. Сенсор	Лінеаризація амплітудної характеристики сенсора. Масштабування сигналів. Корекція похибок аналого-цифрового перетворення. Стиснення і завадостійке кодування даних. Фільтрація сигналів із завад
2. "Інтелектуальний" сенсор	Адаптивна фільтрація сигналів. Вимірювання параметрів сигналів в присутності завад. Поворот і масштабування багатовимірних зображень. Виділення параметрів неузгодженості. Адаптивне стиснення даних, компактне представлення зображень. Розпізнавання зображень і класифікація сигналів
3. Блок обробки и перетворення сигналів	Завадостійкий прийом, відновлення сигналів, зображення їх параметрів. Масштабні, геометричні та інші перетворення сигналів. Накопичення даних, статистичний експрес-аналіз. Прогнозування сигналів, параметрів сигналів і траєкторій руху об'єктів. Розпізнавання образів, класифікація сигналів. Ідентифікація процесів і систем. Перетворення координат об'єктів. Виділення параметрів неузгодженості. Забезпечення заданих динамічних властивостей САУ. Реєстрація параметрів процесів та об'єктів. Спряження з іншими системами.
4. Блок формування команд керування	Реалізація оптимальних законів керування і регулювання в одноконтурних і багатоконтурних системах. Завадостійке кодування. Передача сигналів керування до виконавчих механізмів.
5. Прилад контролю і перевірок	Індикація станів системи. Пошук та локалізація несправностей . Самовідновлення функцій САУ. Реалізація сервісних функцій.

Характерною рисою комп'ютерних систем керування є використання персонального комп'ютера (або спеціалізованого контролера) для обробки даних у реальному масштабі часу. Узагальнена структурна схема САУ наведена на рис. 1.5.

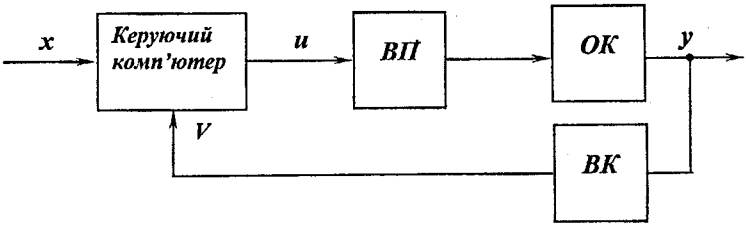


Рис. 1.5. Автоматична КСК технічною системою:
ОК – об'єкт керування, ВП – виконавчий пристрій,
ВК – вимірювальний канал

При побудові централізованих систем керування багатовимірними об'єктами задача полягає у визначенні способів збору вхідних даних контролера і передавання керуючих впливів від контролера. В основі визначення цього лежить кількість та типи сигналів, що повинні передаватися, і швидкість передавання, тобто характеристики інтерфейсу.

Найчастіше використовуються два головних типи інтерфейсів:

- радіальний;
- лінійний.

В радіальній системі керування кожний сенсор та виконавчий пристрій зв'язані з керуючим комп'ютером окремою лінією зв'язку. Це дозволяє використовувати прості сенсори та виконавчі пристрої. Всі задачі узгодження характеристик сигналів в такій системі покладаються на інтерфейси (рис. 1.6).

В лінійній системі кожний сенсор та виконавчий пристрій повинен мати вбудований мікроконтролер для забезпечення зв'язку, оскільки кожний сенсор та виконавчий пристрій повинен мати свою адресу у лінійній мережі і відповідати певними діями на запити контролера (рис. 1.7).

Системи керування підприємствами та виробництвами

Функціональні задачі, що розв'язуються за допомогою КСК на підприємствах пов'язані переважно з реєстрацією та координацією виробничо-господарської діяльності.

Центральною частиною КСК виробництвом є база даних, яка використовується як для реєстрації всіх господарських операцій, так і для обміну даними між окремими функціональними задачами.

Структура КСК виробництвом грунтується на структурі підприємства та інформаційних потоків, які використовують при розв'язанні основних функціональних задач.

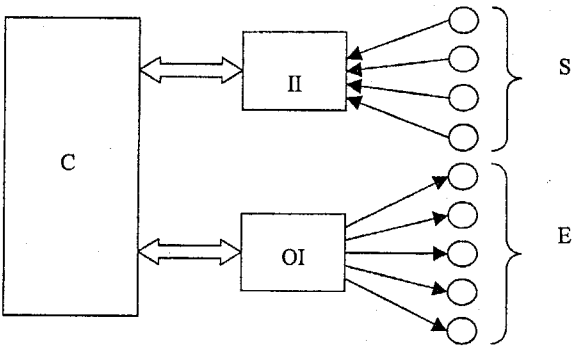


Рис. 1.6. Структура системи з радіальним інтерфейсом
С – контролер (комп'ютер); ІІ – вхідний інтерфейс; ОІ – вихідний інтерфейс; S – сенсори; Е – виконавчий пристрій

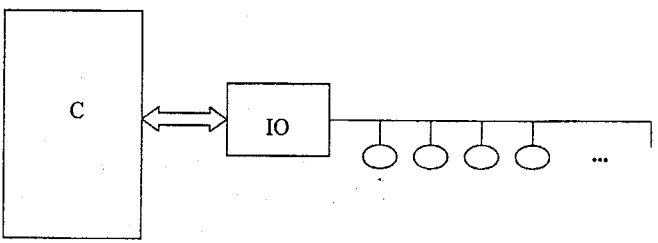


Рис. 1.7. Структура системи з лінійними інтерфейсом
ІО – інтерфейс подвійної дії (комбінований вхід – вихід)

Структура підприємства зазвичай ієрархічна з деякими додатковими зв'язками.

Якщо технологічні процеси на виробництві автоматизовані, то КСК технологічними процесами можуть включатися до КСК виробництвом як окремі підсистеми.

Для того, щоб побудувати систему керування при розв'язанні даної задачі необхідно з'ясувати:

- хто розв'язує дану задачу (який підрозділ);
- хто постачає початкові дані для розв'язання цієї задачі;
- хто споживає результати розв'язання задачі.

Враховуючи отримані дані, будується таблиця інформаційних потоків при розв'язанні конкретної функціональної задачі з графами:

- номер по порядку;
- джерело інформації;
- споживач інформації;
- середня кількість інформації на одне повідомлення;
- середня частота передавання повідомлення.

На основі таблиці інформаційних потоків будується структурна схема системи так, щоб мінімізувати загальне інформаційне навантаження на мережу.

КСК організаційними системами

Структури організаційних систем і ступінь їх автоматизації дуже різні. Так наприклад, збройні сили як організаційна система побудовані за ієрархічним принципом. З іншого боку органи і установи державної влади мають значно складнішу структуру через наявність багатьох гілок влади. Кожна з цих гілок має свою ієрархію, але є також і взаємодія між гілками по горизонталі.

Значна територіальна розсіяність багатьох організаційних систем приводить до необхідності використовувати для зв'язку між окремими підрозділами мережі загального користування (телефонні канали, Internet тощо). В результаті фізичні зв'язки між комп'ютерами, що обслуговують ці підрозділи, не є визначальними для структури відповідної КСК. Значно важливішими є логічні і інформаційні зв'язки між підсистемами.

В основі побудови структурної схеми КСК організаційними системами лежить аналіз функціональних задач КСК.

Аналіз функціональних задач КСК подається у вигляді таблиці з графами:

1) назва задачі;
2) початкові дані :
   - вид даних;
   - кількість даних;
   - постачальник даних.
3) результати :
   - вид результату;
   - кількість даних, які отримуються в результаті;
   - хто їх використовує.
4) як часто розв'язується ця задача.

Аналіз функціональних задач КСК показує величини інформаційних потоків при розв'язуванні задач КСК.

КСК організаційною системою є комп'ютерною мережею, в якій порядок з'єднання комп'ютерів визначається на основі врахування таких факторів:

- мінімізація завантаження каналів зв'язку в ході розв'язування функціональних задач.
- захищеність системи в цілому та окремих її елементів від несанкціонованого доступу.
- забезпечення надійності системи.

1.4 Методологія проектування комп'ютеризованих систем керування та автоматики

Проектування КСК є першим етапом їх життєвого циклу. Як правило, у проектних організаціях цей процес є досить формалізованим, за кожний його етап відповідають окремі працівники, які передають один одному результати проектування у вигляді стандартних документів.

Сучасні САУ з ЕОМ повинні задовольняти загальні вимоги. Основними з цих вимог є простота и зручність використання, гнучкість, живучість и економічність.

Простота і зручність використання пов'язані з необхідністю освоєння систем без залучення дефіцитних, висококваліфікованих спеціалістів. Терміни навчання спеціалістів і освоєння ними техніки повинні бути мінімальними.

Гнучкість систем характеризується їх здатністю до модернізації. Відомо, що в процесі експлуатації властивості керованого об'єкта, його структура можуть змінюватися. Це вимагає зміни алгоритму и внесення поправок в програми керування. Якщо система керування не передбачає таких можливостей, то з прогресивного фактора вона стає фактором консервативним.

Поняття "живучість", що є дещо ширшим, ніж поняття "надійність", зв'язують із збереженням працездатності системи не тільки в нормальних умовах експлуатації, але і при зовнішніх аварійних впливах. При цьому допускається деяке погіршення якості керування. Живучість систем звичайно забезпечується введенням резервування, діагностування і тестування, правильною побудовою структури і вишукуванням більш надійних методів вимірювання та керування.

Економічність обумовлюється малими капітальними вкладеннями та малими експлуатаційними витратами. Відомо, що в складі системи керування відносна вартість технічних засобів з кожним роком зменшується, а частка вартості проектування і програмування збільшується, досягаючи 60–90% загальних витрат на систему. Деякі зарубіжні фірми представляють користувачам ЕОМ навіть безкоштовно, вимагаючи оплати тільки математичного забезпечення. В зв'язку з цим виключно актуальним становиться широке використання систем

автоматичного проектування (САПР), що знижують трудомісткість и тривалість проектування програмного забезпечення.

Основні етапи проектування КСК.

1. *Змістовний опис об'єкта керування.* Цей етап здійснюється у тісній співпраці з замовниками КСК. Опис об'єкта здійснюється у загальновідомій термінології на основі технічної документації про об'єкт, експертних даних персоналу тощо.
2. *Формальний опис об'єкта.* Суть цього етапу у створенні математичної моделі об'єкта. Для розв'язання задач керування необхідна модель, яка пов'язує параметри, що визначають ефективність функціонування об'єкта з зовнішніми впливами, в тому числі з параметрами, на які може впливати КСК (керованими параметрами).
3. *Формулювання мети та обмежень системи.* Аналогічно опису об'єкта мета та обмеження формулюються спочатку змістовно, а потім формалізуються. Бажано, щоб мета була єдиною (однокритеріальна задача), а на решту показників були накладені обмеження. Але часто мета формулюється у вигляді багатьох критеріїв, які входять в протиріччя.
4. *Обрання алгоритму (закону) керування.* Для лінійних систем це здебільшого нескладний формальний процес. Але інколи при високих вимогах до якості керування, відмовостійкості, нелінійних системах тощо виникає необхідність застосування оптимальних, інваріантних, адаптивних, інтелектуальних та інших складних законів керування.
5. *Обрання технічних засобів системи керування.* Цей етап здебільшого ґрунтується на аналізі вартості, технологічності, надійності серійних засобів на основі технічної документації виробників. Задача одночасно проста, оскільки таких засобів і інформації про них достатньо, і складна через те, що потрібно зазвичай дуже велике різноманіття засобів.
6. *Розробка програмного забезпечення КСК.* Як вже зазначалося, цей етап має найбільшу трудомісткість при створенні сучасних систем керування. Тому якісне проведення цього етапу є визначальним для досягнення високих якісних показників КСК. Для запровадження системного підходу до проектування програмного забезпечення КСК використовують методи та інструментальні засоби системного аналізу (System Analysis).
7. *Розробка технічної документації.* Виконується з застосуванням сучасних систем автоматизації проектування.
8. *Розробка методик та допоміжних матеріалів для забезпечення взаємодії системи керування з персоналом.* Для забезпечення

системи взаємодії системи керування з персоналом розробляються спеціальні документи, проводиться навчання і тестування персоналу.

9. *Супроводження виробництва та впровадження.* Оскільки проект КСК у багатьох аспектах грунтується на експертних оцінках, то майже завжди доводиться коректувати як вихідні дані, так і прийняті проектні рішення вже на стадії виробництва та впровадження.

1.5 Методи моделювання КСК

1.5.1 Види моделей КСК

В залежності від способу опису КСК моделі розділяються на:

- вербальні;
- формальні;
- алгоритмічні;
- фізичні.

Вербальні моделі використовують словесний опис об'єкта. Такі моделі часто використовують в нетехнічних галузях, а також на початковому етапі моделювання в техніці.

Формальні моделі використовують опис об'єкта моделювання у вигляді формул і представляються системою математичних співвідношень.

Алгоритмічні моделі представляють об'єкт у вигляді послідовності дій, які дозволяють отримати його необхідну характеристику.

Фізична модель представляє об'єкт-оригінал іншим об'єктом такої ж (масштабні моделі) або іншої (аналогові моделі) фізичної природи. Основою фізичного моделювання є теорія подібності. При фізичному моделюванні зберігаються особливості проведення експерименту в натурі. Фізичне моделювання застосовують при дослідженні систем, для яких вихідні дані відомі з обмеженою точністю або неможливо дати точний математичний опис їх функціонування, а отримання експериментальних характеристик пов'язано з надмірними труднощами та затратами.

В *аналогових моделях* фізична природа моделі і об'єкта різні, а їх математичні описи подібні і, крім того, подібні рівняння, які описують їх окремі елементи. В моделі-аналозі реакції на збурення подібні реакціям на аналогічні збурення об'єкта. Моделі-аналоги складаються з окремих блоків, моделюючих фізичні елементи, а не з блоків, виконуючих окремі математичні операції. Кожному фізичному параметру в об'єкті однозначно відповідає деякий елемент в моделі-аналозі. Найчастіше використовують електронні моделі при дослідженні поведінки систем, конструювання та безпосереднє вивчення яких пов'язано з надмірними труднощами та витратами.

Масштабна модель – це аналогова модель, в якій між параметрами

об'єкта і моделі однакової фізичної природи існує однозначна відповідність, а також відповідність між функцією збудження та реакцією. В масштабній моделі кожний елемент їх в масштабі повторює відповідний елемент об'єкта.

В залежності від типу простору, в якому розглядається об'єкт виділяють моделі:

- геометричні;
- структурні;
- функціональні;
- інформаційні.

Геометричні моделі відображають форму та розташування об'єкта моделювання та його складових частин. Геометричними моделями є різні креслення механізмів, будівель тощо.

Структурна модель представляє об'єкт моделювання з точки зору його складу та взаємозв'язку частин між собою та з зовнішнім середовищем. Структурні моделі найчастіше існують у формі різних структурних та принципових схем.

Функціональні моделі описують процеси, що відбуваються в об'єкті моделювання.

Інформаційна модель – система даних про об'єкт та опис потоків даних в процесі його функціонування.

В залежності від наявності відображення змін стану об'єкта у часі моделі поділяються на:

- моделі статики;
- моделі динаміки.

Моделі статики відображують стан та функціонування об'єкта без врахування їх змін у часі. Як правило, вони подаються у вигляді функціональних залежностей, рівнянь та систем рівнянь.

Моделі динаміки відображають поведінку об'єкта у часі. Моделі динаміки багатші за моделі статики, оскільки останні можуть розглядатися як частковий випадок для певного фіксованого моменту часу. Відповідно і форм представлення моделей динаміки значно більше (диференціальні рівняння, операторні рівняння, спектральні представлення тощо).

За ступенем та характером невизначеності моделі поділяються на:

- детерміновані;
- стохастичні;
- нечіткі;
- узагальнені.

Детерміновані моделі не враховують можливі відхилення характеристик об'єкта від номінальних значень.

Стохастичні моделі розглядають поведінку системи в умовах дії випадкових впливів та випадкової зміни параметрів системи. Інколи розглядають також випадкові зміни структури системи, зумовлені

ненадійністю зв'язків між підсистемами та іншими причинами, моделюючи їх ймовірнісними графами.

Нечіткі моделі використовують у випадках, коли окремі параметри системи задані експертом з кінцевим ступенем впевненості.

Узагальнені моделі використовуються при моделюванні систем, в яких частина параметрів задані достовірно, частина отримана в результаті статистичної обробки певних випадкових процесів, а частина задана експертним методом.

В залежності від способу представлення складного об'єкта (системи) розрізняють моделі:

- агрегатні;
- комплексні.

Агрегатна модель складної системи складається з моделей окремих підсистем та опису їх взаємодії. Якщо розглядати для прикладу деяку систему керування, то моделі підсистем представляються окремими рівняннями, що зв'язують вихідні сигнали з вхідними і параметрами підсистеми, а агрегатна модель буде представлятися системою цих рівнянь. Взаємодія підсистем тут враховується тим, що вихідні сигнали однієї підсистеми є вхідними для іншої і в агрегатній моделі мають однакове позначення.

Комплексна модель представляє систему в цілому, не розділяючи її на підсистеми і окремі внутрішні процеси. Комплексна модель може бути отримана з агрегатної шляхом зведення системи рівнянь до одного рівняння, що зв'язує вхідні і вихідні сигнали системи, методом підстановки.

В залежності від способу отримання результатів моделювання розрізняють математичні моделі:

- аналітичні;
- імітаційні.

Аналітичне моделювання – знаходження характеристик об'єкта на основі формальної або алгоритмічної моделі шляхом виконання певних математичних перетворень: розв'язання рівнянь та систем рівнянь тощо.

Імітаційне моделювання – проведення на ЕОМ чисельних експериментів з математичною моделлю, що описує поводження складної системи на протязі певного періоду часу. Застосовується, як правило, в тих випадках, коли аналітичні способи дослідження моделі відсутні, а їх пошук потребує дуже великих витрат. Алгоритми імітаційного моделювання можуть враховувати як детермінованість, так і стохастичність, зв'язки і залежності, що характеризують об'єкт моделювання. Найбільшого розповсюдження отримали стохастичні методи імітаційного моделювання, оскільки ки для більшості складних систем відомі лише усереднені значення параметрів. Оскільки імітаційне моделювання представляє собою експеримент, важливе значення має застосування методів планування експерименту.

1.5.2 Представлення моделей графами

Граф – це наочне графічне представлення взаємозв'язку елементів деякої множини об'єктів. Таке дуже загальне визначення наштовхує на думку, що методи і алгоритми теорії графів можуть використовуватись для розв'язування дуже великої кількості задач. Адже поняття множини – це фундамент сучасної математики, а це означає, що практично будь-яка математична модель може бути представлена у термінах графів. Теорія графів дуже багата на алгоритми вирішення найрізноманітніших задач. Питання полягає лише в доцільності такого представлення – може існує більш простий і ефективний шлях вирішення конкретної задачі?

При створенні графової моделі слід в першу чергу визначитися, у якому просторі вона створюється. В залежності від природи елементів множини графи можуть розглядатися у різних просторах:

в евклідовому просторі – наприклад, карта автомобільних доріг, план розташування комп'ютерів у мережі;

в просторі станів (у часі) – алгоритм (відображує зміну та зв'язок станів комп'ютера), мережевий графік (відображує зміну та зв'язок станів технологічного процесу);

в просторі відношень – комп'ютерна мережа (відображує інформаційний зв'язок комп'ютерів), "любовний трикутник" (відображує відносини між людьми), схема взаємодії підрозділів підприємства.

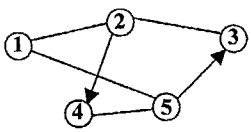
Графова модель G{V, P} складається з двух множин – множини V об'єктів (вершин, вузлів) і множини P зв'язків (ребер). Односторонні зв'язки зображуються направленими ребрами, які називають дугами (рис. 1.8). Граф з дугами називають орієнтованим або орграфом.

У програмуванні найчастіше використовують два представлення графа – наочне (графічне) та формальне у вигляді матриці суміжності (елемент a_{ij} =1, якщо між i і j вершинами є зв'язок та a_{ij} =0, якщо немає).

Шлях в орграфі між вершинами V_n і V_m – це послідовність вершин $\{V_n...V_i, V_{i+1}...V_m\}$ таких, що кожна пара послідовних вершин $\{V_i, V_{i+1}\}$ суміжна.

Ланцюг – шлях в неорієнтованому графі.

Цикл – ланцюг, початок і кінець якого збігаються.



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	0
3	0	1	0	0	0
4	0	0	0	0	1
5	1	0	1	1	0

а) б)

Рис. 1.8. Графова модель:
а) – зображення; б) – матриця суміжності

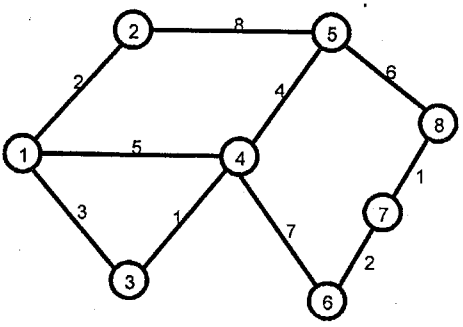
Контур – цикл в орграфі.

Петля – цикл, який складається з однієї вершини.

Дерево – зв'язаний граф без циклів.

Графи, в яких кожному ребру відповідає певне число (наприклад, відстань між комп'ютерами в комп'ютерній мережі, пропускна здатність каналів зв'язку, час виконання операцій у технологічному процесі тощо) називається зваженим. Інколи такі графи називають мережевими.

При описі зважених графів в алгоритмах використовується матриця відстаней. Відсутність зв'язку між вершинами в такій матриці позначається комп'ютерним аналогом нескінченності – достатньо великим числом (заздалегідь більшим за суму довжин всіх ребер). Приклад графа і відповідної матриці відстаней наведений на рис. 1.9.



$$\begin{bmatrix} B & 2 & 3 & 5 & B & B & B & B \\ 2 & B & B & B & 8 & B & B & B \\ 3 & B & B & 1 & B & B & B & B \\ 5 & B & 1 & B & 4 & 7 & B & B \\ B & 8 & B & 4 & B & B & B & 6 \\ B & B & B & 7 & B & B & 2 & B \\ B & B & B & B & B & 2 & B & 1 \\ B & B & B & 6 & B & B & 1 & B \end{bmatrix},$$

Рис. 1.9. Зважений граф

де B – велике число, наприклад, $B=1000$.

Більшість задач на графах пов'язані з пошуком шляхів, циклів та дерев у графі, які задовольняють певні умови задачі. Такі задачі розв'язуються шляхом повного чи обмеженого перебору комбінацій вершин. Кількість варіантів таких комбінацій, а також певні правила їх утворення розглядає комбінаторика.

1.6 Об'єкти керування

Головною частиною будь-якої системи керування є звичайно об'єкт керування, оскільки саме заради нього і створюється система. Від типу об'єкта, його характеристик та особливостей залежить не тільки результат керування, а й сама принципова можливість такого керування.

1.6.1 Класифікація об'єктів керування

Відповідно до трьох основних напрямків застосування КСК можна виділити такі типи об'єктів керування:

- технологічне обладнання;
- виробництво;
- організаційна система.

За характером протікання технологічних процесів об'єкти керування поділяються на циклічні, неперервно-циклічні і безперервні.

За характером зміни значення вихідної величини об'єкта при дії на його вхід ступінчатого сигналу виділяють об'єкти із самовирівнюванням і без самовирівнювання.

За кількістю вхідних і вихідних величин і їх взаємозв'язку об'єкти поділяються на одновимірні (один вхід і один вихід) і багатовимірні. Останні можуть бути багатозв'язними – коли спостерігається взаємний вплив каналів регулювання один на одного або незв'язні – взаємозв'язок між каналами малий.

Статичні характеристики об'єкта керування встановлюють зв'язок між сталими значеннями входу і виходу об'єкта. За видом статичних характеристик об'єкти поділяються на лінійні і нелінійні (у тому числі екстремальні). В останніх статична характеристика може бути гладкою, лінеаризованою в околі заданої точки або носити істотно нелінійний характер. Більшість систем регулювання відноситься до класу систем автоматичної стабілізації режиму роботи об'єкта відносно його робочої точки (відносно номінального режиму роботи). У цьому випадку в процесі роботи відхилення змінних відносно робочої точки будуть малі, що дозволяє використовувати лінійні моделі об'єкта керування. Для системи автоматичної стабілізації необов'язкове визначення повної статичної характеристики об'єкта. Досить знати лише динамічний коефіцієнт підсилення в околі робочої точки. У той же час на деяких об'єктах керування необхідне знання всієї статичної характеристики процесу. Якщо вона носить нелінійний характер, то з метою стабілізації загального коефіцієнта підсилення системи у замкнутий контур включають додаткову нелінійність, зворотну статичній характеристиці об'єкта.

Реальні об'єкти можуть бути зосереджені або розподілені, тому регульована величина залежить не тільки від часу, але і від поточних координат точки контролю. Тому повний опис об'єкта керування буде складатися із системи диференціальних рівнянь з частковими похідними. При використанні точкового методу вимірювання одним сенсором, система диференціальних рівнянь з частковими похідними переходить у систему рівнянь зі звичайними похідними. Це істотно спрощує побудову математичної моделі об'єкта, дозволяючи визначити його передавальну функцію. Однак при наявності безлічі датчиків, розподілених, наприклад,

по довжині об'єкта, може виникнути необхідність використання безлічі керуючих сигналів (розподілене керування).

Об'єкти можуть бути як стаціонарні, так і нестаціонарні. У нестаціонарних об'єктах параметри змінюються з часом (дрейфують). Прикладами таких об'єктів можуть бути хімічний реактор з каталізатором, активність якого падає з часом або аерокосмічний апарат, маса якого при вигорянні палива зменшується.

Разом з динамічною частиною W(p) у структурі об'єкта можуть міститися різні запізнення в сигналах керування, вимірювання і стану (рециклу – у промислових об'єктах під рециклом розуміється повернення частини продукту з виходу об'єкта на його вхід з метою повторної переробки) (рис. 1.10).

Більшість промислових об'єктів керування мають запізнення. Наявність запізнення пояснюється кінцевою швидкістю поширення речовини і сигналів в технологічних об'єктах (транспортне запізнення).

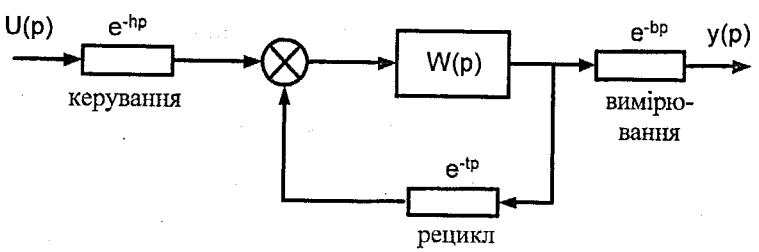


Рис. 1.10. Об'єкт керування с запізненням

В об'єктах керування можуть змінюватися не тільки параметри, але і структура й алгоритм. У лінійних об'єктах зміни структури приводять до змін порядку передавальної функції. У нелінійних об'єктах зміни структури можуть приводити до ще більш глибоких змін математичної моделі.

Зміни алгоритму функціонування звичайно спостерігаються в об'єктах з елементами інтелекту і пов'язані з процесами адаптації і навчання.

В залежності від інтенсивності випадкових збурень, діючих на об'єкт, вони поділяються на стохастичні і детерміновані. У реальних умовах часто точно невідома ні точка прикладення збурення F, ні його характер.

1.6.2 Обстеження об'єктів керування

Обстеження об'єктів керування здійснюється на початковому етапі проектування КСК. Воно призначене для визначення типу об'єкта, мети керування, обмежень та основних характеристик об'єкта, необхідних для

розробки його математичної моделі. Джерелами обстеження є змістовний опис об'єкта експертами, технічна документація об'єкта (якщо вона є), особисті спостереження та дослідження розробника КСК.

Обстеження об'єкта здійснюється у відповідності до заздалегідь розробленого плану і оформлюється у вигляді відповідних документів.

Процес обстеження зазвичай носить ітераційний характер, оскільки на першому етапі важко скласти вичерпний план обстеження. Після первинного обстеження та аналізу результатів план обстеження уточнюється і обстеження доповнюється. Інколи доводиться повертатися до обстеження об'єкта і на наступних етапах проектування КСК, якщо результати проектування не задовольняють вимоги.

1.6.3 Аналіз та моделі об'єктів керування

Аналіз результатів обстеження об'єкта керування здійснюється на першому етапі розробки моделі об'єкта.

Метою аналізу є виявлення:

- мінімальної множини параметрів стану, які вичерпно характеризують стан об'єкта з точки зору мети керування та обмежень;
- множини керованих змінних, вплив на які можливий і достатній для досягнення мети керування;
- множини збурень, які є суттєвими з точки зору мети керування та обмежень.

Методологія аналізу залежить від способу отримання та форми представлення результатів обстеження. Найгрунтовніше проробленa методика факторного аналізу, яка застосовується при поданні результатів обстеження у статистичній формі.

Відомо, що лише при наявності досить точної математичної моделі об'єкта можна спроектувати високоякісну систему керування цим об'єктом. Причому, відповідно до принципу Ешбі, складність пристрою керування повинна бути не нижче складності об'єкта керування.

Тому основною метою побудови математичної моделі об'єкта керування є визначення структури об'єкта, його статичних і динамічних характеристик. Особливо важливе визначення структури для багатовимірних і багатозв'язних об'єктів керування. У той же час для локальних об'єктів керування визначення структури може бути зведене до визначення порядку диференціального рівняння, що описує об'єкт. Крім того, оцінюються вхідні сигнали і збурення, що діють на об'єкт (їхні статистичні характеристики, точки прикладення, максимальні амплітуди). Значення цих характеристик дозволяє вибрати структуру регулятора і розрахувати параметри його настройки, орієнтуючись також на критерій якості роботи цієї системи.

Існують аналітичні, експериментальні і комбіновані методи одержання математичного опису об'єктів керування.

Аналітичні методи базуються на використанні рівнянь, які описують процеси, що протікають у досліджуваному об'єкті керування з використанням законів тієї предметної області, до якої відноситься об'єкт. Наразі для багатьох класів об'єктів керування отримані їх математичні моделі.

Експериментальні методи припускають проведення серії експериментів на реальному об'єкті керування. Обробивши результати експериментів, оцінюють параметри динамічної моделі об'єкта, задавшись попередньо її структурою.

Найбільш ефективними виявляються комбіновані методи побудови математичної моделі об'єкта, коли, використовуючи аналітично отриману структуру об'єкта, її параметри визначають під час натурних експериментів.

Переваги аналітичних методів:

- не вимагають проведення експериментів на реальному об'єкті;
- дозволяють визначити математичний опис ще на стадії проектування системи керування;
- дозволяють врахувати всі основні особливості динаміки об'єкта керування, наявність нелінійностей, нестаціонарність, розподілені параметри і т.д.;
- забезпечують одержання універсального математичного опису, придатного для широкого класу аналогічних об'єктів керування.

Недоліки:

- складність отримання досить точної математичної моделі, що враховує всі особливості реального об'єкта;
- перевірка адекватності моделі і реального процесу вимагають проведення натурних експериментів;
- багато математичних моделей мають ряд важкооцінюваних у числовому вигляді параметрів (наприклад, константи швидкостей хімічних реакцій).

Методи експериментального визначення динамічних характеристик об'єктів керування

При розрахунку настроювань регуляторів локальних систем широко використовуються досить прості динамічні моделі промислових об'єктів керування. Зокрема, використання моделей інерційних ланок першого або другого порядку з запізнюванням для розрахунку настроювань регуляторів забезпечує в більшості випадків якісну роботу системи керування.

В зв'язку з цим виникає задача визначення чисельних значень параметрів динамічних моделей промислових об'єктів керування.

В залежності від вигляду перехідної характеристики моделі задаються найчастіше одним із трьох видів передавальної функції об'єкта керування:

- у вигляді передавальної функції інерційної ланки першого порядку

$$W_0(p)=\frac{Ke^{-\tau p}}{Tp+1}; \tag{1.1}$$

де - K,T,τ коефіцієнт підсилення, постійна часу і запізнювання, що повинні бути визначені в околі номінального режиму роботи об'єкта.

- другого порядку з запізнюванням

$$W_0(p)=\frac{Ke^{-\tau p}}{(T_1p+1)(T_2p+1)}; \tag{1.2}$$

- для об'єкта керування без самовирівнювання передавальна функція має вигляд

$$W_0(p)=\frac{Ke^{-\tau p}}{p}. \tag{1.3}$$

Експериментальні методи визначення динамічних характеристик об'єктів керування поділяються на два класи:

- методи визначення часових характеристик об'єкта керування.
- методи визначення частотних характеристик об'єкта керування.

Часові методи визначення динамічних характеристик поділяються, у свою чергу, на активні і пасивні.

Активні методи допускають подання на вхід об'єкта пробних тестових сигналів, якими є:

- регулярні функції часу (ступінчастий або прямокутний імпульси, гармонічний сигнал змінної частоти, періодичний двійковий сигнал);
- пробні сигнали випадкового характеру (білий шум, псевдовипадковий двійковий сигнал).

В залежності від виду пробного сигналу обирають відповідні методи обробки вихідного сигналу об'єкта керування. Так, наприклад, при поданні ступінчастого керуючого сигналу знімають криву розгону об'єкта, а при поданні гармонічного сигналу змінної частоти знімають амплітудно-частотну і фазо-частотну характеристики.

Перевагами активних методів є:

- досить висока точність одержання математичного опису;
- відносно мала тривалість експерименту.

Варто враховувати, що активні методи приводять до порушення нормального ходу технологічного процесу. Тому проведення активного експерименту не завжди можливе.

У пасивних методах на вхід об'єкта не подаються ніякі пробні сигнали, а лише фіксуються вхідні сигнали і стани об'єкта в процесі його нормального функціонування. Отримані реалізації масивів даних вхідних і вихідних сигналів обробляються статистичними методами. За результатами обробки одержують параметри передавальної функції об'єкта. Однак, такі методи мають ряд недоліків:

- мала точність одержуваного математичного опису, (тому що відхилення від нормального режиму роботи малі);
- необхідність накопичення великих масивів даних з метою підвищення точності (тисячі точок);
- якщо експеримент проводиться на об'єкті, охопленому системою регулювання, то спостерігається ефект кореляції (взаємозв'язку) між вхідним і вихідним сигналами об'єкта через регулятор. Такий взаємозв'язок знижує точність математичного опису.

При знятті кривої розгону необхідно виконати ряд умов:

- якщо проектується система стабілізації, то крива розгону повинна зніматися в околі робочої точки процесу;
- криві розгону необхідно знімати як при позитивних, так і негативних стрибках керуючого сигналу. З вигляду кривих можна судити про ступінь асиметрії об'єкта. При невеликій асиметрії розрахунок настроювань регулятора рекомендується вести за усередненим значенням параметрів передавальних функцій;
- при наявності зашумленого виходу бажано знімати кілька кривих розгону з наступним одержанням усередненої кривої;
- при знятті кривої розгону амплітуда пробного вхідного сигналу повинна бути, з одного боку, досить велика, щоб чітко виділялася крива розгону на тлі шумів, а, з іншого боку, вона повинна бути досить мала, щоб не порушувати нормального ходу технологічного процесу.

Знявши криву розгону і оцінивши характер об'єкта керування (із самовирівнюванням або без) можна визначити параметри відповідної передавальної функції. Передавальну функцію вигляду (1.1) рекомендується застосовувати для об'єктів керування з явно вираженою домінуючою постійною часу. Перед початком обробки криву розгону рекомендується пронормувати (діапазон зміни нормованої кривої 0 - 1) і виділити з її початкової ділянки величину чистого часового запізнювання.

Приклад. Дано нормовану криву розгону об'єкта, у якої заздалегідь виділена величина чистого запізнювання $\tau_3 = 3хв$ (рис. 1.11.).

Динамічний коефіцієнт підсилення K об'єкта визначається як відношення збільшення вихідного сигналу до збільшення вхідного в околі робочої точки.

Визначення динамічних характеристик об'єктів по кривій розгону можна робити двома методами.

1. Метод дотичної до точки перегину кривої розгону.

У даному випадку точка перегину відповідає переходу кривої від режиму прискорення до режиму уповільнення темпу наростання вихідного сигналу. Постійна часу Т і динамічне запізнювання τ_∂ визначаються відповідно до графіка рис.1.11., тобто $\tau = \tau_з + \tau_\partial$.

2. Формульний метод дозволяє аналітично обчислити величину динамічного запізнювання і постійної часу за формулами

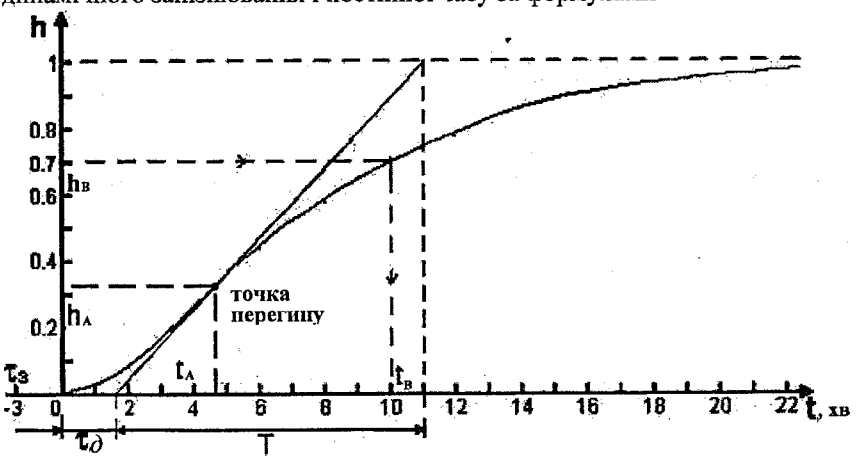


Рис. 1.11. Графік кривої розгону

$$\tau_d = \frac{t_B \ln(1-h_A) - t_A \ln(1-h_B)}{\ln(1-h_A) - \ln(1-h_B)}, \quad T = -\frac{t_A - \tau_d}{\ln(1-h_A)},$$

де значення h_A, береться в околі точки перегину кривої, а значення h_B приймається рівним 0,7. За цими значеннями визначаються і моменти часу t_A та t_B.

Методику визначення параметрів динамічної моделі (1.2) об'єкта без самовирівнювання розглянемо на прикладі кривої розгону рис. 1.12.

Для об'єкта без самовирівнювання коефіцієнт підсилення визначається як відношення сталої швидкості зміни вихідної величини до величини стрибка вхідного сигналу. У нашому прикладі величина

динамічного запізнювання τ в об'єкті визначається так, як показано на рис.1.12.

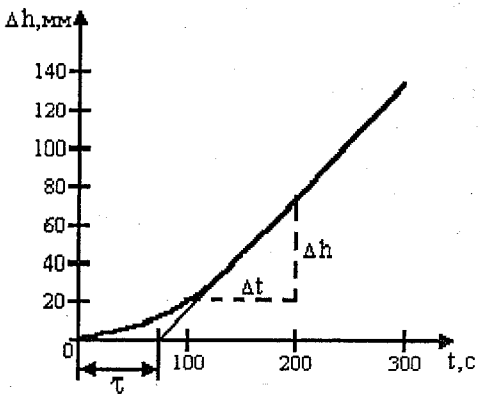


Рис. 1.12. Графік характеристики розгону об'єкта без самовирівнювання

Частотні методи визначення динамічних характеристик

Ці методи припускають, що на вхід об'єкта подається періодичний сигнал з відомою частотою й амплітудою. Якщо цей сигнал формується за допомогою прямокутних імпульсів, то еквівалентна амплітуда синусоїдального сигналу буде більше амплітуди прямокутного імпульсу в $4/\pi$ рази, що повинно враховуватися при розрахунку частотних характеристик.

Частотні методи визначення динамічних характеристик об'єкта припускають наявність двох етапів, на яких визначаються.

1. Амплітудно-фазова характеристика об'єкта.
2. Передавальна функція об'єкта.

Амплітудно-фазова характеристика (АФХ) об'єкта несе більшу інформацію про об'єкт, ніж його крива розгону. Однак при визначенні динамічних характеристик об'єкта за допомогою частотних методів варто враховувати, що вони більш трудомісткі і вимагають наявності спеціальної апаратури.

Поняття про статистичні методи визначення динамічних характеристик об'єкта

У статистичних методах визначення динамічних характеристик об'єкта керування передбачається, що вхідний і вихідний сигнали об'єкта є реалізаціями випадкових процесів. У зв'язку з цим їх обробка повинна проводитися статистичними методами. Вхідний сигнал може бути або штучно сформований від генератора шуму, або бути природним шумовим компонентом в умовах нормальної експлуатації об'єкта.

Пробний вхідний сигнал повинен містити у своєму спектрі безліч гармонік з випадковою амплітудою і фазою. Це дозволяє прискорити процес визначення параметрів об'єкта керування. Цю вимогу задовольняє сигнал типу "білого шуму". Але в реальних умовах такий сигнал не відтворюваний, тому обмежують частотний спектр випадкового сигналу, прикладом якого є псевдовипадковий двійковий сигнал. Генератор шуму настроюють таким чином, щоб він генерував безліч гармонік, що оцінюють найбільш істотну ділянку АФХ об'єкта керування.

За отриманими масивами вхідних і вихідних даних визначають автокореляційну $R_{uu}(\tau)$ і взаємнокореляційну $R_{uy}(\tau)$ функції. Кореляційні функції характеризують тісноту імовірнісного зв'язку між двома сусідніми ординатами процесу, зсунутими на час τ, де $0 \le \tau \le \infty$. За кореляційними функціями знаходять відповідні спектральні щільності $S_{uu}(\omega)$ та $S_{uy}(\omega)$:

$$S_{uu}(j\omega) = \int_0^\infty R_{uu}(\tau)e^{-j\omega\tau}d\tau \text{ та } S_{uy}(j\omega) = \int_0^\infty R_{uy}(\tau)e^{-j\omega\tau}d\tau.$$

Знаючи спектральні щільності, можна визначити модуль частотної передавальної функції об'єкта

$$|W(\omega)| = \frac{S_{uy}(\omega)}{S_{uu}(\omega)}.$$

Контрольні питання та завдання

1. Назвіть основні складові задачі керування.
2. Які операції, пов'язані з керуванням, можуть бути виконані за допомогою комп'ютерів?
3. Що спільного і чим відрізняються САУ і АСУ?
4. Порівняйте переваги і недоліки централізованих і розподілених систем керування.
5. Назвіть основні етапи проектування КСК.
6. Які типи моделей використовуються на кожному етапі проектування КСК?
7. На якому етапі проектування КСК здійснюється обстеження об'єкта керування?
8. В чому переваги і недоліки пасивного і активного експериментів?

Література

1. Молчанов А.А. Моделирование и проектирование сложных систем. – К.: Вища школа, 1988.
2. Коршунов Ю.М. Математические основы кибернетики. – М.: Энергия, 1972. – 376 с.
3. Дубовой В.М. Моделювання систем контролю та керування: Навчальний посібник. – Вінниця: ВНТУ, 2005. – 175 с.
4. Сигорский В.П. Математический аппарат инженера. – К.: Техника, 1975.
5. Основи дискретної математики: Підручник. /Ю.В.Капітонова та ін. – К.: Наукова думка, 2002. – 580с.
6. Мокін Б.І., Мокін В.Б. Математичні методи ідентифікації електромеханічних процесів: Навчальний посібник.–Вінниця: УНІВЕРСУМ-Вінниця, 1999.
7. Справочник по теории автоматического управления/ Под ред. А.А. Красовского. – М.: Наука, 1987. – 712с.
8. Иванова В.М., Калинина В.Н., Нешумова Л.А., Решетникова И.О. Математическая статистика.– М.: Высш. школа, 1981. – 371с.
9. Вентцель Е.С., Овчаров Л.А. Теория случайных процессов и ее инженерные приложения. – М.: Наука, 1991. – 384с.
10. Автоматизированные системы управления технологическими процессами (справочник)/Под ред. Б.Б.Тимофеева. – К.: Техніка, 1983. – 351с.
11. Адлер Ю.П. и др. Планирование эксперимента при поиске оптимальных условий. – М.: Наука, 1976. – 280с.

2. МЕТОДИ КОМП'ЮТЕРНОЇ ОБРОБКИ СИГНАЛІВ В КСК

Обробка сигналів широко використовується у КСК технологічними процесами і технічними об'єктами. Сигнал є головним носієм інформації, а обробка сигналів здійснюється з метою виділення цієї інформації з параметрів сигналу, усунення дії завад, перетворення його форми тощо.

Особливість обробки сигналів в КСК полягає у необхідності здійснення цієї операції переважно у реальному масштабі часу. Зазвичай КСК працюють безперервно на протязі досить довгого часу. Відповідно кожен сигнал на протязі цього часу відображується великою кількістю даних, яка разом з тим постійно збільшується. Оскільки постійна обробка такої кількості даних вимагає все більше і більше часу, то найчастіше обмежуються обробкою „останніх N даних". З надходженням кожного нового даного це поняття змінюється – нове дане додається до реалізації сигналу, а найстаріше відкидається. Така операція реалізується за допомогою буфера даних типу „черга", як показано на рис. 2.1. Програмна реалізація черги наведена у Додатку 1.

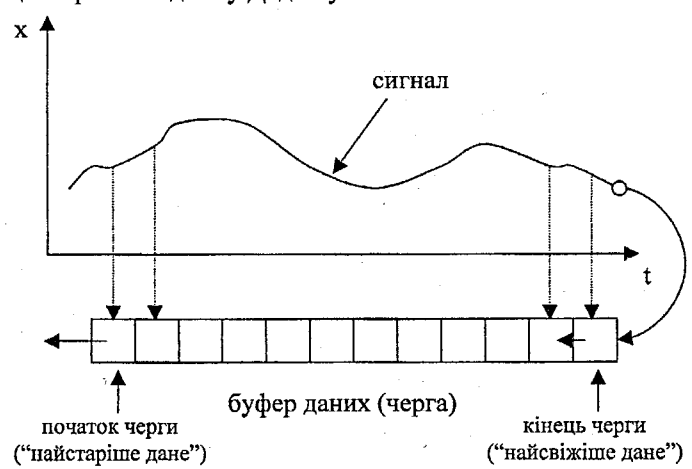


Рис. 2.1. Занесення значень сигналу *x(t)* до черги

Загальна схема алгоритму обробки сигналу за допомогою черги наведена на рис. 2.2.

В алгоритмі рис. 2.2. циклічна процедура обробки сигналу здійснюється з надходженням кожного значення. Цей процес відбувається весь час, поки працює система керування.

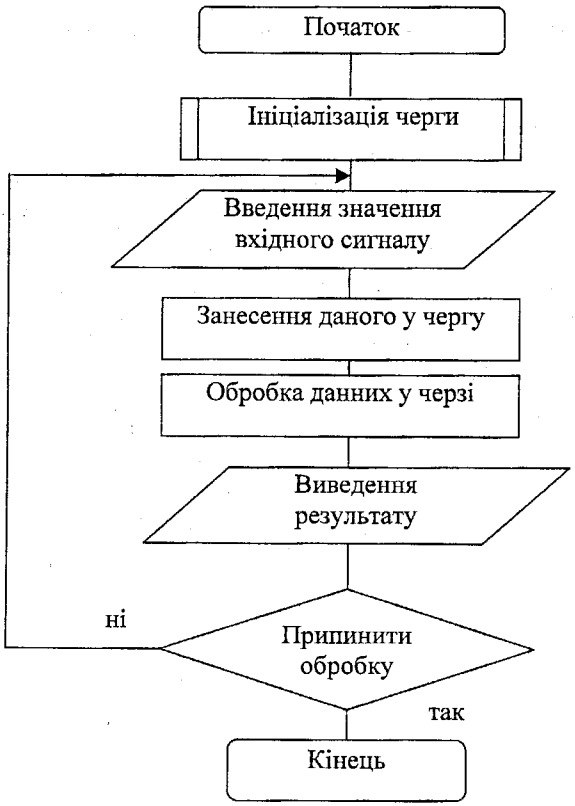


Рис. 2.2. Схема алгоритму обробки сигналу за допомогою черги

2.1 Статистична обробка у реальному часі

Статистична обробка сигналів у системах керування може здійснюватися на основі множини реалізацій випадкового процесу або на основі послідовності значень однієї реалізації у часі. У першому випадку отримують різні середні (моменти): середнє арифметичне, середнє квадратичне, дисперсію, асиметрію, ексцес тощо, а також гістограму закону розподілення імовірностей. У другому випадку отримують ті ж самі характеристики, а також усереднені характеристики зміни параметрів сигналу у часі: кореляційну функцію (коефіцієнт кореляції) і пов'язану з нею спектральну щільність потужності. Але усереднення у часі можливе лише для ергодичних процесів.

Оцінювання моментів у реальному масштабі часу

Якщо вибірка даних про сигнал *x(t)* занесена до буфера довжиною *N*, то обчислення вибіркового середнього m_x, вибіркової дисперсії D_x і вибіркового середнього квадратичного відхилення σ_x здійснюється за допомогою звичайних формул математичної статистики:

$$m_x = \frac{\sum_{i=1}^{N} x_i}{N}, \quad D_x = \frac{\sum_{i=1}^{N} (x_i - m_x)^2}{N-1}, \qquad \sigma_x = \sqrt{D_x}.$$

З наведених співвідношень видно, що всі дані реалізації сигналу враховуються з однаковою вагою. Це доречно у випадках стаціонарного сигналу. Але при наявності тренду (повільної зміни середніх значень) доцільно найновішим даним давати більшу вагу, а старішим – меншу. Це може бути здійснене за допомогою рекурсивних формул, застосування яких не потребує використання буфера даних:

$$m_x = \frac{N \cdot m_x + x}{N+1}, \quad D_x = \frac{(N-1) \cdot D_x + (x - m_x)^2}{N},$$

де x – чергове значення сигналу, N-константа. Враховуючи реальну фізичну природу сигналу початкові значення m_x та D_x слід взяти нульовими. Значення константи N визначає кількість даних, які суттєво впливають на результат.

Очевидно, при розрахунку середнього вага найновішого значення буде $1/(N+1)$, а кожного попереднього зменшується з коефіцієнтом $N/(N+1)$.

Тоді вага даного x_i, буде $\left(\frac{N}{N+1}\right)^i$. Якщо вважати даними, які суттєво впливають на результат, такі, що за величиною збігаються з останнім прийнятим даним, то константу N можна знайти з рівняння

$$\left(\frac{N}{N+1}\right)^i = 0{,}1.$$

Якщо, наприклад, ми хочемо здійснювати усереднення на основі останніх 20 даних, то

$$\left(\frac{N}{N+1}\right)^{20} = 0{,}1,$$

звідки

$$N = \frac{\sqrt[20]{0{,}1}}{1 - \sqrt[20]{0{,}1}}.$$

Програма статистичної обробки даних рекурсивним методом наведена у додатку.

2.2 Кореляційна обробка, визначення інтервалу кореляції

Кореляційна функція показує зв'язок між двома сигналами або між рознесеними у часі частинами одного сигналу.

Автокореляційна функція:

$$R_{xx}(\tau)=\frac{1}{T}\int_0^T[X(t)-m_x][X(t-\tau)-m_x]dt.$$

Взаємокореляційна:

$$R_{xy}(\tau)=\frac{1}{T}\int_0^T[X(t)-m_x][Y(t-\tau)-m_y]dt.$$

За допомогою кореляційної обробки даних здійснюється вимірювання затримки сильно защумлених сигналів, спектральний аналіз, ров'язуються задачі ідентифікації тощо.

При використанні дискретних відліків

$$R_{xx}(\tau)=\frac{1}{K-1}\sum_{i=N-K+1}^{N}(x_i-m_x)(x_{i-\tau}-m_x),$$

$$R_{xy}(\tau)=\frac{1}{K-1}\sum_{i=N-K+1}^{N}(y_i-m_y)(x_{i-\tau}-m_x),$$

де $N=2K,\quad \tau=0\ldots K$.

Очевидно, для обчислення взаємної кореляційної функції необхідно використати дві черги вхідних даних: для значень X і для значень У. Схема розрахунку наведена на рис. 2.3.

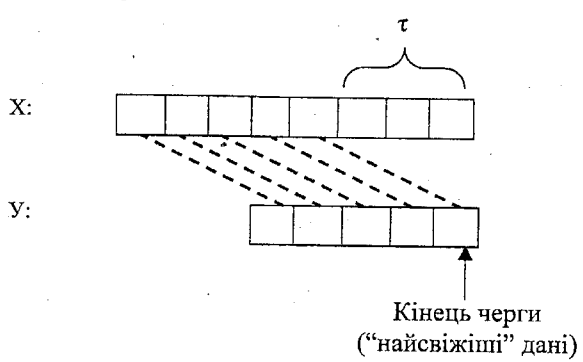


Рис. 2.3. Схема розрахунку кореляційної функції

На рис. 2.3 пунктирними лініями показані пари даних, які перемножаються при обчисленні. З схеми видно, що при зміні τ від 0 до k довжина черги x повинна бути $2k$.

При розрахунках взаємнокореляційної функції слід враховувати причинно-наслідкові зв'язки, а саме те, що причина завжди передує наслідку. Якщо x – вхідний сигнал системи, тобто є причиною, а y – вихідний, тобто є наслідком, то відповідно у розрахункових формулах повинні використовуватися значення y у момент i, а x у момент $(i - \tau)$.

Програма кореляційної обробки наведена у додатку.

2.3 Цифрова фільтрація

Цифрова фільтрація сигналів є важливою частиною будь-яких автоматизованих систем через необхідність боротьби з завадами. Способи фільтрації можна розділити на такі:

- лінійна;
- нелінійна.

2.3.1 Лінійна фільтрація

Лінійні фільтри розрізняють в першу чергу за виглядом частотної характеристики. Основні типи наведені на рис. 2.4.

Всі лінійні фільтри є інерційними елементами. Тому для обчислення результату фільтрації вони використовують не одне, а декілька останніх значень вхідного сигналу. Ступінь інерційності фільтра (тобто кількість вхідних даних, що враховуються) визначається сталою часу і порядком фільтра та інтервалом надходження вхідних даних. Особливість таких алгоритмів фільтрації полягає в утворенні черги вхідних даних.

Довжина черги відповідає умові

$$N \geq 5\, k\, (T_\phi / \tau),$$

де N – кількість елементів черги;

k – порядок фільтра;

T_ϕ – стала часу фільтра;

τ - інтервал надходження даних.

Синтез алгоритмів фільтрації складає окрему і часто складну математичну задачу. У загальному випадку лінійний фільтр описується передавальною функцією W(p) у вигляді раціонального дробу.

Передавальну функцію можна отримати шляхом розкладання ЛАЧХ фільтра на прості складові, які є характеристиками елементарних динамічних кіл (аперіодичного, інтегрального, диференційного, коливального) і перемноження відповідних передавальних функцій.

Побудуємо дискретну модель лінійного фільтра, який представляється передавальною функцією загального типу:

$$W(p) = \frac{Y(p)}{X(p)} = \frac{a_n p^n + \ldots + a_1 p + a_0}{b_m p^m + \ldots + b_1 p + b_0}, \qquad (2.1)$$

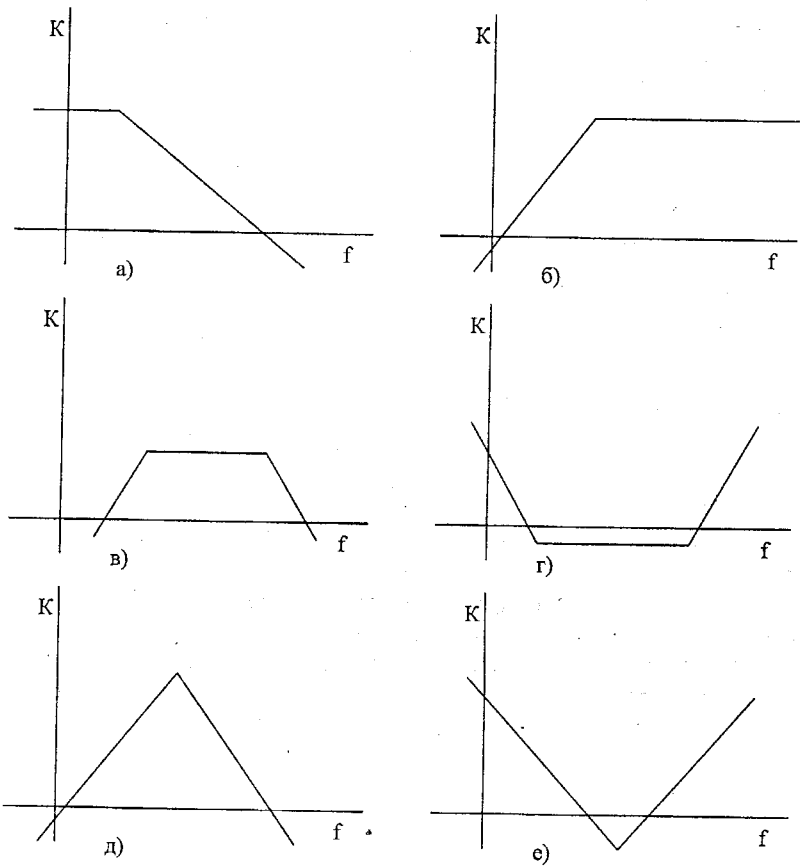


Рис. 2.4. Лінійні фільтри:

а) – нижніх частот; б) – верхніх частот;
в) – смуговий; г) – смуговий загороджувальний;
д) – вибірковий; е) – вибірковий загороджувальний

де Y(p), X(p) – зображення за Лапласом вихідного та вхідного сигналів лінійної системи. З (2.1) знаходимо:

$$b_m p^m y + \ldots + b_1 py + b_0 y = a_n p^n x + \ldots + a_1 px + a_0 x. \qquad (2.2)$$

Відповідно до перетворення Лапласа:

$$p^k Z(p) \overset{L}{\leftrightarrow} \frac{d^k Z(t)}{dt^k}. \qquad (2.3)$$

При дискретній реалізації цифрової фільтрації аналогом похідних є відповідні різниці:

$$z'(t_0) = \frac{z_0 - z_{-1}}{\Delta t},$$

$$z''(t_0) = \frac{z'(t_0) - z'(t_{-1})}{\Delta t} = \frac{z_0 - 2z_{-1} + z_{-2}}{\Delta t^2},$$

$$z'''(t_0) = \frac{z''(t_0) - z''(t_{-1})}{\Delta t} = \frac{z_0 - 3z_{-1} + 3z_{-2} - z_{-3}}{\Delta t^3}$$

тощо. У загальному випадку:

$$z^{(k)}(t_0) = \frac{1}{\Delta t^k} \sum_{i=0}^{k} (-1)^i C_k^i z_{-i}, \qquad (2.4)$$

де t_0 – момент надходження останнього даного, поточний момент часу;

Δt – інтервал дискретизації.

Підставляючи (2.4) в (2.2), отримаємо дискретний вираз рівняння фільтра

$$\sum_{j=0}^{m} \left[\frac{b_i}{\Delta t^j} \sum_{i=0}^{j} (-1)^i C_j^i y_{-i} \right] = \sum_{j=0}^{n} \left[\frac{a_i}{\Delta t^j} \sum_{i=0}^{j} (-1)^i C_j^i x_{-i} \right]. \qquad (2.5)$$

Виділимо з лівої частини рівняння (2.5) значення вихідної величини у поточний момент часу.

$$\sum_{j=0}^{m} \frac{b_i}{\Delta t^j} \left[y_0 + \sum_{i=1}^{j} (-1)^i C_j^i y_{-i} \right] = \sum_{j=0}^{n} \left[\frac{a_i}{\Delta t^j} \sum_{i=0}^{j} (-1)^i C_j^i x_{-i} \right]$$

або

$$y_0 = \frac{\sum_{j=0}^{n} \left[\frac{a_i}{\Delta t^j} \sum_{i=0}^{j} (-1)^i C_j^i x_{-i} \right] - \sum_{j=1}^{m} \left[\frac{b_i}{\Delta t^j} \sum_{i=1}^{j} (-1)^i C_j^i y_{-i} \right]}{\sum_{j=0}^{m} \frac{b_i}{\Delta t^j}}.$$

Змінюючи порядок підрахунку сум у чисельнику отримуємо

$$y_0 = \frac{\sum_{j=0}^{n} \left[\sum_{i=0}^{j} (-1)^i C_j^i \frac{a_i}{\Delta t^j} \right] x_{-i} - \sum_{j=1}^{m} \left[\sum_{i=1}^{j} (-1)^i C_j^i \frac{b_i}{\Delta t^j} \right] y_{-i}}{\sum_{j=0}^{m} \frac{b_i}{\Delta t^j}}$$

або
$$y_0 = \sum_{i=0}^{n} K_{x_i} x_{-i} + \sum_{i=1}^{m} K_{y_i} y_{-i}, \qquad (2.6)$$

де $K_{x_i} = \dfrac{(-1)^i \sum\limits_{j=i}^{n} C_j^i \dfrac{a_j}{\Delta t^j}}{\sum\limits_{j=0}^{m} \dfrac{b_j}{\Delta t^j}},$

$$K_{y_i} = \frac{(-1)^{i+1} \sum\limits_{j=i}^{m} C_j^i \dfrac{b_j}{\Delta t^j}}{\sum\limits_{j=0}^{m} \dfrac{b_j}{\Delta t^j}}.$$

Вираз (2.6) є дискретною моделлю лінійного фільтра з передавальною функцією (2.1). Модель (2.6) рекурсивна, оскільки поточне значення вихідної величини Y обчислюється з використанням попередніх значень. Початкові значення змінних рекурсивного виразу:

$$\forall x_i = 0, i = 0,-1,...,-n; \quad \forall y_i = 0, i = 0,-1,...,-m.$$

У окремих випадках:

1. $W(p) = \dfrac{1}{Tp}$ (2.7)

Тут n=0, m=1, a_0=1, b_1=T, b_0=0

$$y_0 = \frac{x_0 - \left[\dfrac{T}{\Delta t}(-1)^i y_{-1}\right]}{\dfrac{T}{\Delta t}} = y_{-1} + \frac{1}{T} X \Delta t. \qquad (2.8)$$

Очевидно, передавальна функція (2.7) відповідає інтегратору, а вираз (2.8) є кусково-ступінчастою апроксимацією інтегрування.

2. $W(p) = \dfrac{1}{Tp+1}$ (2.9)

Тут n=0, m=1, a_0=1, b_1=T, b_0=1

$$y_0 = \frac{x_0 - \left[\dfrac{T}{\Delta t}(-1) y_{-1}\right]}{1+\dfrac{T}{\Delta t}} = \frac{\dfrac{T}{\Delta t} y_{-1} + x_0}{1+\dfrac{T}{\Delta t}}. \qquad (2.10)$$

Очевидно, передавальна функція (2.9) відповідає аперіодичному ланцюгу, а вираз (2.10) є рекурсивною сумою із зменшенням вагових коефіцієнтів "застарілих даних". Якщо позначити $T/\Delta t = k$, то рекурсивна дискретна модель аперіодичного ланцюга матиме вираз

$$y = \frac{ky + x}{k+1}. \qquad (2.11)$$

Аперіодичний ланцюг є типовим фільтром нижніх частот першого порядку, який є базовим для більшості алгоритмів фільтрації.

3. $W(p) = Tp$ (2.12)

Тут n=1, m=0, a_0=0, a_1=T, b_0=1

$$y_0 = \frac{T}{\Delta t}(x_0 - x_{-1}) = T\frac{x_0 - x_{-1}}{\Delta t}. \qquad (2.13)$$

Очевидно, передавальна функція (2.12) відповідає диференціатору, а вираз (2.13) є дискретним аналогом похідної.

Приклад лінійної фільтрації наведено у додатку

2.3.2 Нелінійна фільтрація

На відміну від лінійних фільтрів параметри нелінійного фільтра залежать від значень вхідних даних. Найчастіше нелінійні фільтри використовуються для покращення фільтрації в умовах значних імпульсних завад. Прикладом застосування нелінійних фільтрів є подавлення шуму у паузах корисного сигналу. Це є особливо важливим в системах з автоматичним регулюванням підсилення. В таких системах у паузах коефіцієнт підсилення зростає, в результаті чого шуми у паузах підсилюються. Для запобігання цьому здійснюють нелінійну фільтрацію, при якій інерційність фільтра при малих сигналах зростає

$$T = T_0\left(\frac{1}{1+x} + 1\right).$$

Ще одним прикладом застосування нелінійних фільтрів є фільтрація великих викидів вхідного сигналу.

Якщо взяти за основу лінійний фільтр першого порядку (8**), то при великих відхиленнях вхідного сигналу від середнього значення фільтр роблять більш інерційним (збільшують k)

$$k = k_0 + k_1|x - y|.$$

Приклад програми нелінійної фільтрації наведений у додатку.

2.3.3 Спектральний аналіз. Швидке перетворення Фур'є

Дуже важливою операцією обробки сигналів є знаходження їх спектрів. Спектр сигналу – це результат подання його у вигляді суми (інтегралу) ортогональних функцій. Найчастіше за систему базисних ортогональних функцій приймають гармонічні (синусоїдальні) функції. Гармонічний спектр сигналу отримується в результаті перетворення Фур'є.

Перетворення Фур'є буває двох типів: дискретне й неперервне. Неперервне використовується в аналітичних дослідженнях, дискретне – у всіх інших випадках.

Неперервне перетворення Фур'є – це перетворення, яке застосовується до функції $x(t)$, заданої на інтервалі $(-\infty;+\infty)$. В результаті одержується функція $S(f)$:

$$S(f)=\int_{-\infty}^{+\infty}x(t)e^{2\pi ft}dt.$$

Також існує зворотне перетворення, яке дозволяє за зображенням $S(f)$ відновити початкову функцію $x(t)$:

$$x(t)=\int_{-\infty}^{+\infty}S(f)e^{-2\pi ft}df.$$

Очевидно, що відображення $S(f)$ є комплексною функцією дійсного аргументу, однак і $x(t)$ може приймати як дійсні, так і комплексні значення.

На практиці ми зазвичай працюємо з дискретними даними. Досить часто у нас заданий не аналітичний вираз перетворюваної функції, а лише набір її значень на деякій сітці (як правило, на рівномірній). В такому разі доводиться робити припущення, що за межами цієї сітки функція дорівнює нулю, і апроксимувати інтеграл інтегральною сумою:

$$S_n\equiv\sum_{k=0}^{N-1}x_k\exp\left(\frac{kn}{N}2\pi i\right)\quad n\in\left[-\frac{N}{2},\frac{N}{2}\right].$$

В разі рівномірної сітки ця формула спрощується. Також на рівномірній сітці зазвичай відмовляються від кроку, щоб одержати безрозмірну формулу.

Визначене таким чином дискретне перетворення Фур'є зберігає практично усі властивості неперервного.

На проведення дискретного перетворення Фур'є потрібно порядка N^2 операцій. Але можна обмежитись і значно меншою кількістю операцій. Спосіб, за допомогою якого ДПФ обчислюється за $Nlog_2N$ операцій, має назву швидкого перетворення Фур'є (ШПФ). Обмеженням ШПФ є те, що воно працює лише з наборами даних в кількості, яка дорівнює степені двійки.

Формули прямого й зворотного дискретних перетворень Фур'є відрізняються лише множником перед сумою і знаком показника експоненти. Фактично, для проведення прямого й зворотного перетворень можна застосувати один і той же алгоритм, вказуючи при цьому напрямок перетворення для зміни відповідних параметрів.

В основі ШПФ закладено ідею про те, що для парного N можна розділити перетворення Фур'є від усього масиву на суму перетворень від елементів з парними й від елементів з непарними номерами:

$$H_n=\sum_{k=0}^{N-1}h_k\exp\left(\frac{kn}{N}2\pi\right)=\sum_{k=0}^{N-1}h_kW^{kn}=\ldots$$

$$\ldots = \sum_{k=0}^{N/2-1} h_{2k} W^{2kn} + \sum_{k=0}^{N/2-1} h_{2k+1} W^{(2k+1)n} = H_n^e + W^n H_n^o,$$

де $W = \exp\left(\frac{2\pi}{N}\right)$.

Перший доданок в сумі відповідає ДПФ від елементів з парними номерами, другий – від елементів з непарними. Якщо N не просто парне, а ще й кратне чотирьом, то можна рекурсивно застосувати цей процес до одержаних доданків. Якщо N є степенем двійки, то процес можна звести до обчислення ДПФ від одного числа. Цей процес повторюється $log_2 N$ раз для всіх N значень перетворення.

Процедура ШПФ наведена у додатку. На вході процедура отримує параметри:

- m – кількість значень функції. Повинно дорівнювати степені двійки;
- a – масив з 2·m дійсних чисел, нумерація елементів від 0 до 2·m-1. Якщо проводиться пряме перетворення, цей масив задає nn значень функції (не забуваємо, що функція комплексна). Якщо перетворення зворотне, то тут зберігаються відповідні частоти (більш детально формат зберігання наведено нижче, на рисунку);
- *InverseFFT* – напрямок перетворення. Якщо проводиться пряме перетворення, то параметр встановлюється в False, в разі зворотного перетворення – в True.

На виході:

- a – містить результат перетворення. При проведенні прямого перетворення тут знаходяться частоти (див. рисунок), при зворотному перетворенні – комплексні значення функції.

Схема даних, що використовуються у ШПФ, наведена на рис. 2.5. Нехай відомо N значень функції, розташованих з інтервалом *Delta*. Значення зберігаються парами дійсних чисел; перше число в парі відповідає дійсній частині, а друге – уявній, як показано на рисунку. Кожному значенню x_k відповідає певний момент часу t_k.

Після проведення перетворення Фур'є ми одержуємо масив комплексних значень (пар дійсних чисел), в якому зберігаються коефіцієнти S_n. Існує залежність між номером i та відповідною йому частотою f_i, між коефіцієнтом S_i і значенням перетворення Фур'є (неперервного) на відповідній частоті f_i:

$$f_i = \frac{i}{N\Delta}, \; \Delta S_i \approx S(f_i).$$

При проведенні зворотного перетворення формат параметрів не змінюється, лише на вході задається частотне представлення, а на виході отримуємо часове.

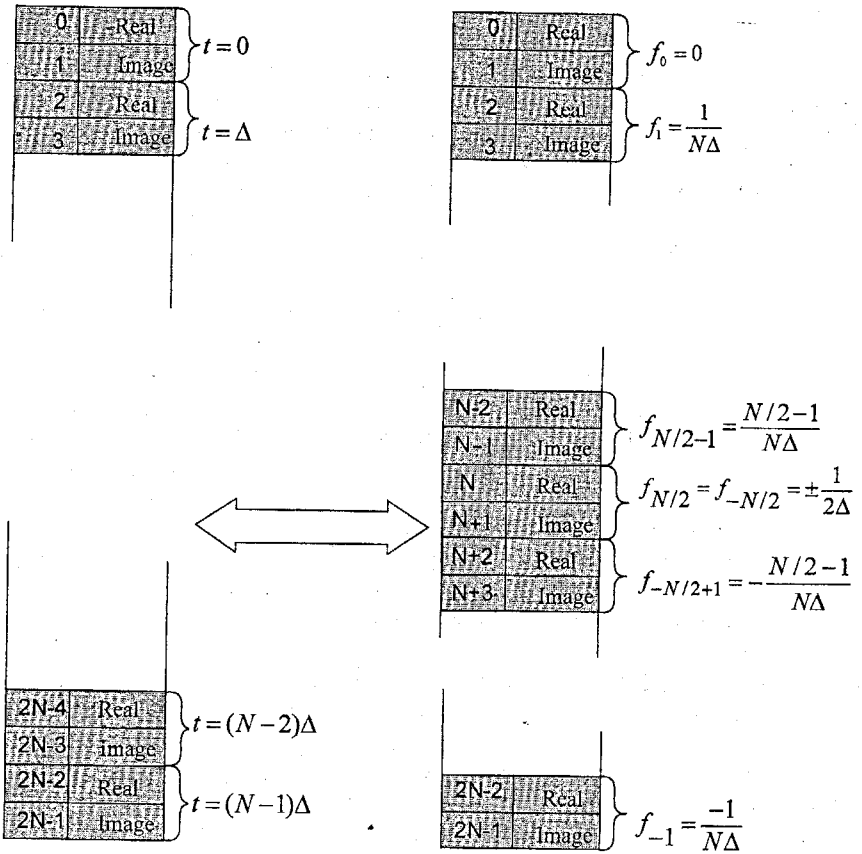


Рис. 2.5. Схема даних, що використовуються у ШПФ

2.4. Екстраполяція сигналу

Екстраполяція даних використовується при побудові адаптивних систем з прогнозуванням керованого процесу. Алгоритм екстраполяції залежить від вибору екстраполюючої функції. Але у будь-якому випадку

для екстраполяції n-го порядку необхідно зберігати n даних у буфері зі структурою черги.

Задача екстраполяції полягає у визначенні значення функції $y(x)$ в точці, що не належить відрізку $[x_0, x_n]$, на якому визначені вузли інтерполяції. Для екстраполяції використовуються ті ж вирази, що і при інтерполяції, але з більшою похибкою. Так, для гладких функцій екстраполяція доцільна при x, що виходять за зазначені межі не більш ніж на $h/2$, де h - крок розташування вузлів.

При інтерполяції Лагранжа розв'язок задачі подається у вигляді

$$P(x) = \sum_{1}^{n} y_k L_k(x),$$

де

$$L_k(x) = \frac{(x-x_1)(x-x_2)...(x-x_{k-1})(x-x_{k+1})...(x-x_n)}{(x_k-x_1)(x_k-x_2)...(x_k-x_{k-1})(x_k-x_{k+1})...(x_k-x_n)}.$$

Контрольні питання та завдання

1. Назвіть основні види обробки сигналів в КСК.
2. Які моментні характеристики випадкових сигналів найчастіше визначаються в КСК?
3. Охарактеризуйте чергу як базову структуру для обробки даних в динамічній системі.
4. Що таке кореляційна функція, як вона обчислюється і для чого застосовується?
5. Як здійснюється перетворення Фур'є аналітично і чисельно?

Література

1. Макс Ж. Методы и техника обработки сигналов при измерениях. В 2-х томах – М.: Мир, 1983.
2. Бесекерский В.А., Изранцев В.В. Системы автоматического управления с микро-ЭВМ. – М.: Наука, 1987. – 320с.
3. Дубовой В.М. Моделювання систем контролю та керування: Навчальний посібник. – Вінниця: ВНТУ, 2005. – 175 с.
4. Иванова В.М., Калинина В.Н., Нешумова Л.А., Решетникова И.О. Математическая статистика.– М.: Высш. школа, 1981. – 371с.
5. Вентцель Е.С., Овчаров Л.А. Теория случайных процессов и ее инженерные приложения. – М.: Наука, 1991. – 384с.
6. Теория передачи сигналов. / Зюко А.Г. и др. – М.: Связь, 1980.

3. АЛГОРИТМИ КЕРУВАННЯ

Закон керування – це правило обчислення керуючого сигналу u на основі уставки (задавального сигналу) x і параметрів стану об'єкта y. При комп'ютерній реалізації закон керування називають алгоритмом.

Головне призначення закону керування – переведення об'єкта керування з поточного стану y в заданий стан x при умові забезпечення необхідних показників якості.

Закони керування розділяють на лінійні і нелінійні.

Нелінійні закони розділяються на:

- релейне керування;
- оптимальне керування;
- адаптивне керування.

Фактори, які впливають на вибір закону керування:

1) природа об'єкта:
   - технічна (різні електричні, механічні, гідравлічні, теплові та інші пристрої);
   - організаційні об'єкти (підприємства, установи);
   - виробничі (технологічні процеси, конвеєрні лінії тощо);
   - соціальні об'єкти.
2) лінійність об'єкта;
3) інерційність об'єкта;
4) детермінованість об'єкта;
5) вид показників якості керування;
6) наявність та вид обмежень.

3.1 Загальна структура алгоритму

Комп'ютеризоване керування передбачає не тільки комп'ютерну реалізацію законів керування, але й виконання великої кількості допоміжних функцій:

- задання параметрів закону керування;
- задання уставки;
- здійснення зв'язку з рештою підсистем за допомогою прийнятих протоколів;
- реєстрація (ведення протоколу) процесу керування;
- забезпечення довідок у відповідності до стандартних запитів про роботу системи;
- відображення роботи системи у вигляді таблиць, графіків, діаграм;
- виконання діагностичних функцій.

Виконання зазначених функцій може бути організоване кількома способами. Найпростіший з них – об'єднання всіх функцій в одному послідовному алгоритмі з циклічним опитуванням сенсорів та клавіатури.

Узагальнений алгоритм керування при використанні режиму роботи типу циклічного опитування вхідних даних.

```
uses crt;
var
 c:char;
begin
 задання початкових умов(x);
 repeat
  repeat
   читання y з порта;
   обробка даних, обчислення u;
   передавання u через порт;
  until keypressed;
  c:=readkey; if c=#0 then c:=readkey;
  case c of
   код1: процедура1 введення x;
   код2: процедура2;
   ...
  end
 until c=#27
end.
```

Такий алгоритм простий, але допускає втрату даних при виході з режиму *приймання-обробка-передавання* в режим роботи з клавіатурою.

Більш ефективним є алгоритм, в якому процедури *приймання-обробка-передавання* працюють резидентно по перериванню від порту, а решта процедур знаходяться в окремій програмі або в декількох програмах. В такому алгоритмі цикл керування здійснюється у фоновому режимі, не заважаючи оператору користуватися сервісними процедурами.

При використанні резидентних процедур, встановлених різними програмами, виникає необхідність обміну даними між ними. Спосіб обміну даними залежить від операційної системи, під керуванням якої працює програмне забезпечення КСК. Найпростіший і універсальний спосіб обміну – використання всіма програмами спільної бази даних колективного доступу. Але постійне звертання до файлів уповільнює виконання програм.

Ще один нескладний спосіб зв'язку між програмами може використовуватися при роботі під керуванням MS-DOS. Для цього можна використати область зв'язку між застосуваннями, яка виділяється в BIOS у просторі адрес $0000:$04F0 - $0000:$04FF. Якщо цієї області не вистачає,

то для зв'язку виділяють буфер у динамічній пам'яті, а його адресу заносять у область зв'язку BIOS, наприклад, так

```
{buf – вказівник на буфер; M – розмір буфера; i – розмір використаної частини буфера}
begin
 getmem(buf,M);
 i:=0;
 memw[$0000:$04F0]:=Seg(i);
 memw[$0000:$04F2]:=Ofs(i);
 memw[$0000:$04F4]:=Seg(buf^);
 memw[$0000:$04F6]:=Ofs(buf^);
end;
```

Тоді читання даних з буфера здійснюється за покажчиком, який отримується так

```
SgBuf:=memw[$0000:$04F4];
OsBuf:=memw[$0000:$04F6];
Buf:=ptr(SgBuf,OsBuf);
```

При програмуванні під Windows обмін даними між застосуваннями (програмою приймання/передавання даних і сервісними програмами) здійснюється через систему "подій" і "повідомлень".

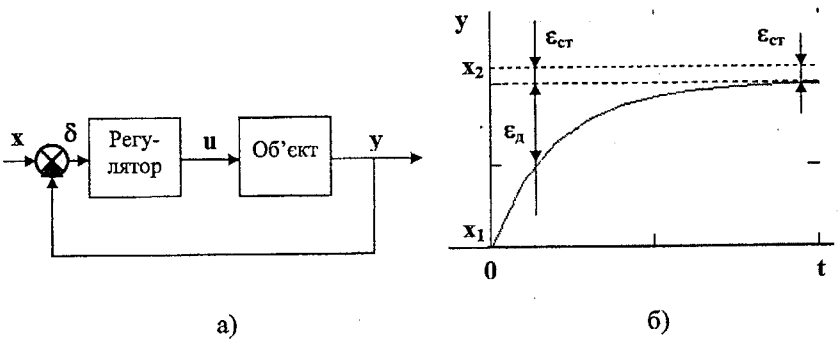
3.2 Реалізація лінійних законів керування

Лінійні закони використовуються в системах автоматичного керування технологічними процесами для керування лінійними об'єктами без обмежень з необхідною точністю. Точність керування визначається похибкою. Розглянемо складові загальної похибки керування на основі найпростішої системи керування. На рис. 3.1,а наведена структурна схема системи, а на рис. 3.1,б часові діаграми вхідного та вихідного сигналів (уставки та стану об'єкта).

Внаслідок інерційності системи при зміні уставки в момент $t = 0$ з $x = x_1$ на $x = x_2$ стан системи у поступово змінюється, наближаючись до заданого значення. В ході перехідного процесу ця інерційність приводить до наявності динамічної похибки єд. Але наявність в будь-якій системі керування певної нечутливості та втрат енергії (аналогічно механічному сухому тертю) приводить до того, що навіть в усталеному режимі існує певна похибка єст, яку називають статичною. Загальна похибка

$$\delta = \varepsilon_{cm} + \varepsilon_{\partial}.$$

Для зменшення тої чи іншої складової загальної похибки використовують різні лінійні закони керування. Найпоширенішими є типові лінійні закони керування:



а) б)

Рис. 3.1. Найпростіша лінійна САУ

1. Пропорційний – П;
2. Інтегральний – І;
3. Диференціальний – Д;
4. Пропорційно-інтегральний – ПІ;
5. Пропорційно-диференціальний (форсуючий) – ПД;
6. Пропорційно-інтегрально-диференціальний – ПІД.

Розглянемо їх на прикладі системи, що має структуру рис. 3.1,а і аперіодичний об'єкт керування з передавальною функцією

$$W_0 = \frac{1}{Tp+1}.$$

Найпростішим є пропорційний закон $u = k(x-y)$ тобто з передавальною функцією регулятора

$$W_p = k.$$

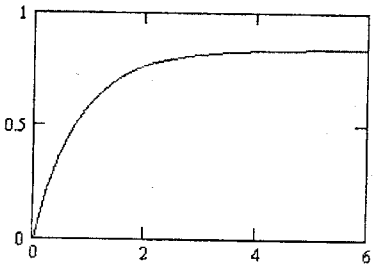
Тоді передавальна функція системи керування

$$W = \frac{k\dfrac{1}{Tp+1}}{1+k\dfrac{1}{Tp+1}} = \frac{k}{Tp+1+k} = \frac{\dfrac{k}{k+1}}{\dfrac{T}{k+1}p+1}.$$

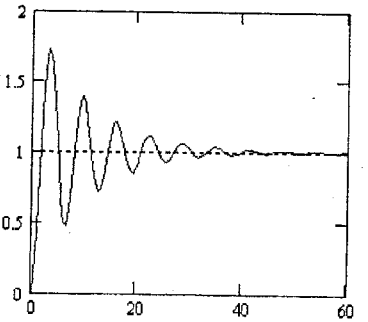
Приклад перехідної характеристики при k=5 і T=5 наведений на рис. 3.2,а.

Така система має значні похибки як статичну, так і динамічну.

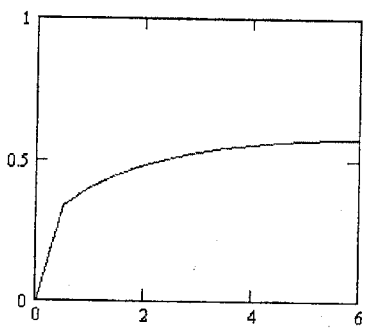
Для усунення статичної похибки використовується інтегральний закон керування



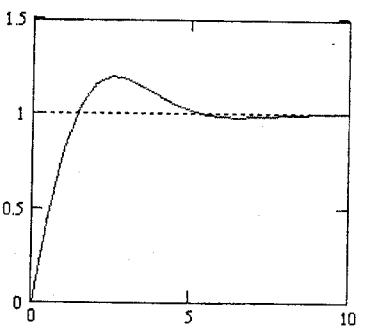
а)



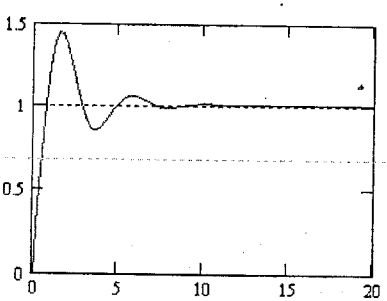
б)



в)



г)



д)

Рис. 3.2. Перехідні характеристики систем з лінійними законами керування

$$u = k\int_0^t (x-y)\,dt.$$

Передавальна функція регулятора, що реалізує такий закон

$$W_p = \frac{k}{p}.$$

Тоді передавальна функція системи керування

$$W = \frac{\frac{k}{p}\cdot\frac{1}{Tp+1}}{1+\frac{k}{p}\cdot\frac{1}{Tp+1}} = \frac{k}{Tp^2+p+k} = \frac{1}{\frac{T}{k}p^2+\frac{1}{k}p+1}.$$

Приклад перехідної характеристики при k=5 і T=5 наведений на рис.3.2,б. З рисунку добре видно, що усуваючи статичну похибку, інтегральний закон керування суттєво погіршує динамічні характеристики системи.

Для одночасного покращання як статичних, так і динамічних характеристик КСК використовуються комбіновані закони керування.

Для зменшення динамічної похибки використовується пропорційно-диференціальний (форсуючий) закон

$$u = k_1(x-y) + k_2\frac{d(x-y)}{dt}.$$

Передавальна функція регулятора, що реалізує такий закон

$$W_p = k_1 + k_2 p.$$

Тоді передавальна функція системи керування

$$W = \frac{(k_1+k_2p)\cdot\frac{1}{Tp+1}}{1+(k_1+k_2p)\cdot\frac{1}{Tp+1}} = \frac{k_2p+k_1}{(T+k_2)p+(k_1+1)} = \frac{\frac{k_2}{k_{1+1}}p+\frac{k_1}{k_{1+1}}}{\frac{T+k_2}{k_{1+1}}p+1}.$$

Приклад перехідної характеристики при k_1=5, k_2=5 і T=5 наведений на рис. 3.2,в. З рисунку добре видно, що диференціальна складова зменшує динамічну та збільшує статичну похибку системи.

Пропорційно-інтегральний закон $u = k_1(x-v) + k_2\int_0^t (x-v)\,dt$ зменшує статичну похибку, не погіршуючи динаміку системи.

Передавальна функція регулятора, що реалізує такий закон $W_p = k_1 + k_2/p$.

Тоді передавальна функція системи керування

$$W=\frac{\left(k_1+{k_2}/{p}\right)\cdot\frac{1}{Tp+1}}{1+\left(k_1+{k_2}/{p}\right)\cdot\frac{1}{Tp+1}}=\frac{k_1p+k_2}{Tp^2+(k_1+1)p+k_2}=\frac{\frac{k_1}{k_2}p+1}{\frac{T}{k_2}p^2+\frac{k_1+1}{k_2}p+1}.$$

Приклад перехідної характеристики при k_1=5, k_2=5 і T=5 наведений на рис. 3.2,г. З рисунку видно, що пропорційно-інтегральний закон усуває статичну похибку, не збільшуючи динамічну похибку системи.

Найкращим (але найскладнішим) є пропорційно-інтегрально-диференціальний закон $u=k_1(x-v)+k_2\int_0^t(x-v)dt+k_3\frac{d(x-v)}{dt}$.

Передавальна функція регулятора, що реалізує такий закон, $W_p=k_1+{k_2}/{p}+k_3p$.

Тоді передавальна функція системи керування

$$W=\frac{\left(k_1+{k_2}/{p}+k_3p\right)\cdot\frac{1}{Tp+1}}{1+\left(k_1+{k_2}/{p}+k_3p\right)\cdot\frac{1}{Tp+1}}=\frac{k_3p^2+k_1p+k_2}{(T+k_3)p^2+(k_1+1)p+k_2}=\frac{\frac{k_3}{k_2}p^2+\frac{k_1}{k_2}p+1}{\frac{T+k_3}{k_2}p^2+\frac{k_1+1}{k_2}p+1}.$$

Приклад перехідної характеристики при k_1=5, k_2=15, k_3=1 і T=5 наведений на рис. 3.2,д. З рисунку видно, що пропорційно-інтегрально-диференціальний закон усуває статичну похибку і зменшує динамічну похибку системи.

3.3 Оптимальні системи

В задачах оптимального керування до основної «технологічної» мети добавляється мета оптимізації, яка математично виражається як вимога забезпечення екстремуму деякого показника якості. Показник якості подається функціоналом, тобто змінною, чисельне значення якої залежить від виду деяких функцій, які є аргументами функціоналу.

3.3.1 Неперервні системи

В оптимальних системах керування всі параметри системи, які входять до задачі оптимізації, заздалегідь відомі. Для неперервних систем можна розрахувати оптимальні зміни в часі керуючого впливу в процесі переводу системи з початкового стану у кінцевий.

Узагальнена структурна схема оптимальної системи наведена на рис.3.3.

В цьому випадку блок оптимізації розраховує як повинен змінюватися X.

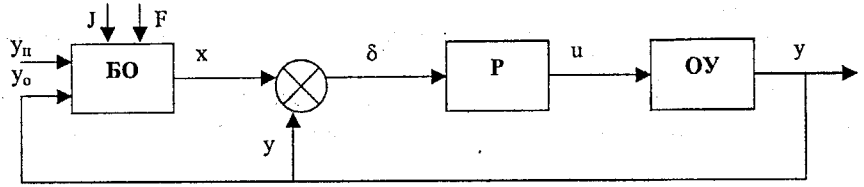


Рис. 3.3. Узагальнена структурна схема оптимальної системи керування
БО – блок оптимізації (розраховує як треба змінити у часі уставку),
Р – регулятор,
ОУ – об'єкт керування.

В цілому постановка задачі оптимального керування включає:

1. Критерій оптимізації (цільову функцію) J;
2. Початковий та кінцевий стани системи y_0, y_n;
3. Обмеження на керуючі впливи та параметри стану системи F, в тому числі:
   - обмеження типу нерівності, які пов'язані з кінцевими фізичними можливостями системи керування (наприклад, швидкість авто у місті не повинна перевищувати 90км/год);
   - обмеження типу рівностей або рівнянь, які є фактично математичною моделлю системи.

Задачі оптимального керування в залежності від характеру початкового та кінцевого станів системи поділяються на 3 типи:

1. Задачі з закріпленими кінцями; коли кінцеві стани є повністю визначеними кінцевими станами. Наприклад необхідно пересунути вантаж з одного місця на інше з мінімальними витратами енергії.
2. Задачі з рухомими кінцями.
   У цих задачах початковий та кінцевий стани задаються множиною значень. Наприклад, необхідно перейти з певного початкового місця з фіксованими координатами до ... за мінімальний час.
3. Задачі з вільними кінцями.

Ці особливості задач впливають на постановку задачі оптимального керування у вигляді відповідного типу обмежень.

У практичних задачах керування в загальному випадку, якщо об'єкт описується системою диференціальних рівнянь першого порядку, можуть зустрічатися функціонали, що залежать від змінних $x, \dot{x}, u, \dot{u}, x(t_0), x(t_f)$, t_0, t_f, t тобто

$$J = J\left[x(t), u(t), \dot{x}(t), \dot{u}(t), x(t_0), x(t_f), t_0, t_f, t\right].$$

В окремих задачах звичайно у функціонал входять лише деякі із

цих змінних.

Функціонал J називають показником якості, критерієм оптимальності, цільовою функцією.

Як критерії оптимальності часто використовують лінійні функціонали виду $J = \sum(a_i x_i + b_i u_i)$ (найпоширеніші в економічних задачах) або квадратичні $J = \int_{t_0}^{t_f} \sum_i \left(a_i x_i^2 + b_i u_i^2\right) dt$ (у динамічних задачах керування). Ці функціонали складають так, щоб вони більш-менш точно відображали фактичні втрати або вигоду.

Отже, задача оптимального керування має таке формулювання. *Знайти функції u*(t) і x*(t), що мінімізують (максимізують) функціонал*

$$J = J\left[x(t), u(t), \dot{x}(t), \dot{u}(t), x(t_0), x(t_f), t_0, t_f, t\right].$$

при умовах

$$F_i(x, \dot{x}, u, \dot{u}, t) = 0, i = 1, ..., n;$$

$$\int_{t_0}^{t_f} f_{n+j}(x, u, t) dt = B_j, j = 1, ..., r;$$

$$x(t_0) \in X_0, x(t_f) \in X_f$$

і обмеженнях на координати і керування $x \in X_t; u \in U_t$.

Керування u*(t) і траєкторія x*(t), які є розв'язком сформульованої задачі, називають оптимальними керуванням і траєкторією відповідно.

Спосіб розв'язання задачі оптимізації залежить від виду критерію оптимальності і обмежень. Аналітичні методи оптимізації використовуються досить рідко через складність комп'ютерної реалізації. Звичайно перевагу віддають пошуковим алгоритмам.

Якщо закон керування відомий, а критерій оптимальності неперервний та диференційований, то використовуються градієнтні методи оптимізації: градієнтний метод, метод найшвидшого спуску, метод Ньютона. У випадку, коли критерій оптимальності не задовольняє умову диференційованості, використовують стохастичні алгоритми.

Задачі на екстремум функціоналів, коли закон керування невідомий, розглядаються у варіаційному численні й називаються варіаційними задачами. Залежно від вигляду функціонала, що задає критерій оптимальності, розрізняють варіаційні задачі Лагранжа, Больца й Майєра.

У задачі Лагранжа функціонал має вигляд певного інтеграла. Сформулюємо задачу із закріпленими кінцями, обмеженнями типу рівностей і фіксованим часом (моменти часу t_0 і t_f задані).

Знайти оптимальне керування u(t) (і, якщо треба, оптимальну траєкторію x*(t)), що мінімізує (максимізує) функціонал*

$$J = \int_{t_0}^{t_f} f_0(x, \dot{x}, u, \dot{u}, t)dt,$$

при умовах

$$F_i(x, \dot{x}, u, \dot{u}, t) = 0, i = 1, ..., n;$$

$$\int_{t_0}^{t_f} f_{n+j}(x, u, t)dt = B_j, j = 1, ..., r;$$

$$\varphi_k(x, u, t) = 0, k = 1, ..., l;$$

$$x(t_0) = x^0, x = (t_f) = x^f,$$

де x^0 і x^f - задані вектори.

Розв'язок $u^*(t)$ і $x^*(t)$ знаходять використовуючи рівняння Ейлера - Лагранжа.

Складемо функцію Лагранжа:

$$L = \psi_0 f_0 + \sum_{j=1}^{r} \xi_j f_{n+j} + \sum_{k=1}^{l} \lambda_k \varphi_k + \sum_{i=1}^{n} \psi_i F_i,$$

де $\psi_0, \xi_i, \lambda_k, \psi_i$ – невизначені множники Лагранжа. Рівняння Ейлера - Лагранжа мають вигляд:

$$\left.\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{x}_i} - \frac{\partial L}{\partial x_i} = 0, i = 1, ..., n; \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{u}_j} - \frac{\partial L}{\partial u_j} = 0, j = 1, ..., m. \end{aligned}\right\}.$$

Припускають, що оптимальне керування належить до класу кусково-неперервних функцій.

Для знаходження оптимального керування й траєкторії рівняння Ейлера-Лагранжа потрібно розв'язати разом з рівняннями об'єкта й обмежень. Розв'язки зазначених рівнянь називаються *екстремалями.*

Якщо граничні значення $x(t_0), x(t_f), t_0$ і t_f не фіксовані, то критерій оптимальності може залежати від цих змінних. У цьому випадку в досить загальному вигляді критерій оптимальності можна записати так:

$$j = g_0\left[x(t_0), x(t_f), t_0, t_f\right] + \int_{t_0}^{t_f} f_0(x, u, t)dt.$$

Варіаційну задачу з таким оптимізуючим функціоналом називають задачею Больца. Якщо $f_0 \equiv 0$, тобто функціонал залежить тільки від граничних значень, одержуємо задачу Майєра, а при $g_0 \equiv 0$ – уже розглянуту задачу Лагранжа. У теоретичному відношенні всі три задачі (Больца, Майєра й Лагранжа) еквівалентні в тому розумінні, що можуть

бути перетворені одна в іншу заміною змінної.

Узагальненням класичного методу множників Лагранжа для визначення екстремуму при наявності обмежень типу рівностей на випадок, коли з'являються обмеження типу нерівностей, забезпечує теорема Куна-Таккера.

Вектор $\bar{x}^0$ *тоді і тільки тоді є розв'язком задачі і*

$$\min\left\{\frac{f(\bar{x})}{\varphi_j(\bar{x})} \le 0,\ x_i \ge 0\right\},$$

де $f(\bar{x})$ *і* $\varphi_j(\bar{x})$ *— випуклі функції, коли існує вектор* $\bar{\lambda}^o$ *такий що*

$$\bar{x}^0 \ge 0, \quad \bar{\lambda}^0 \ge 0; \quad F\left(\bar{x}^0, \bar{\lambda}\right) \le F\left(\bar{x}^0, \bar{\lambda}^O\right) \le F\left(\bar{x}, \bar{\lambda}^0\right).$$

У літературі вказуються способи узагальнення класичних методів шляхом введення ряду додаткових умов, але багато років складність розв'язування приводила до пошуків нових шляхів розв'язування некласичних задач. Одним з таких нових методів, що отримав широке застосування у світовій практиці, є метод, що ґрунтується на використанні принципу максимуму Л.С. Понтрягіна. Цей принцип особливо зручний у застосуванні для розв'язання таких задач, де керування є кусково-сталою функцією. Найчастіше принцип максимуму використовують у задачах, де на керування накладені обмеження типу нерівностей вигляду $u_i \le a_i$, а функціонали лінійні стосовно вхідного в них керування.

Принцип максимуму формулюється таким чином.

Якщо поведінка системи описується системою диференціальних рівнянь

$$\frac{dX}{dt} = f(X,U),$$

де U(t) належить обмеженій замкнутій області Ω*, а фазова траєкторія* $X(_н)$ *належить обмеженій, але відкритій області S, то керування U(t) вважається оптимальним, якщо при переході із заданої початкової точки* $X(t_н)=(x_{1н}, x_{2н},\ldots,x_{nн})$ *у задану кінцеву точку* $X(t_к)=(x_{1н}, x_{2до},\ldots,x_{nк})$ *функціонал*

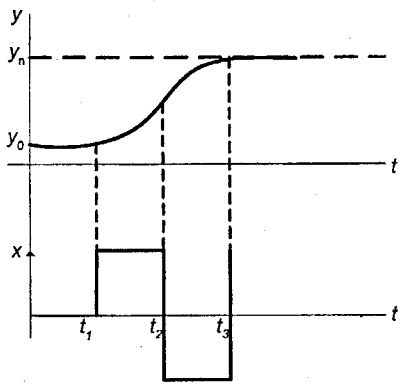
$$J = \int_{t_н}^{t_к} f_0(x,u)dt$$

досягає екстремуму.

З принципу максимуму випливає, що оптимальне керування в системі, на яку накладається k обмежень, складається з k інтервалів постійності.

Приклад оптимального керування при $k=2$ (найчастіше це обмеження величини впливу на об'єкт керування $-U_{max} \le U \le +U_{max}$) наведений на рис. 3.4,а.

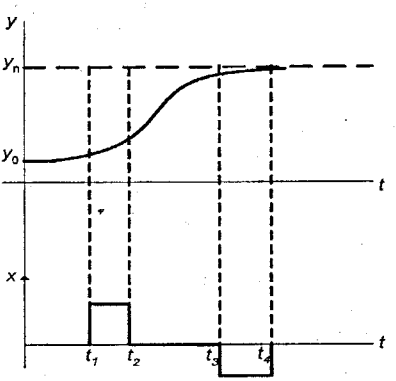
Приклад для k=3 (на додаток до попереднього накладене обмеження швидкості руху $\frac{dy}{dt} < y_{max}$) наведений на рис. 3.4,б.



а)

(t_1, t_2) – інтервал розгону

(t_2, t_3) – інтервал гальмування



б)

(t_1, t_2) – інтервал розгону

(t_2, t_3) – інтервал сталої швидкості

(t_3, t_4) – інтервал гальмування

Рис. 3.4. Оптимальне керування за "принципом максимуму" Понтрягіна

Для систем, точний математичний опис яких невідомий, а стани й критерії оптимальності задаються наборами експериментально заданих значень у дискретні моменти часу, використовують принцип оптимальності, який був сформульований Р. Белманом так: оптимальна поведінка має таку властивість, що які б не були первісний стан і розв'язок в початковий момент часу, наступні розв'язування повинні становити оптимальну поведінку щодо стану, що утворюється в результаті першого розв'язку.

Формулювання стосується широкого класу систем, тому в ній фігурують такі поняття, як поведінка й вирішення. У динамічних системах, які описуються диференціальними рівняннями, стан характеризується набором чисельних значень координат $x=(x_1,...x_n)$ у кожен момент часу, поведінка відповідає процесу зміни координат у часі $x(t)$, а вирішення — вибору оптимального керування $u^*(t)$.

Система задовольняє принцип оптимальності, якщо вона має марковські властивості: її поведінка на будь-якому кінцевому відрізку часу

$t_1 \le t \le t_f$ повністю визначається керуванням на цьому відрізку і станом системи в початковий для цього відрізка момент часу t_1.

Порядок розв'язання задачі оптимального керування методом динамічного програмування:

1. Складають рівняння Белмана $\min_{u \in V} \left[f_0(x,u) + \langle grad\, S(x), f(x,u) \rangle \right] = 0$ і з умови мінімуму його лівої частини знаходять керування, що залежить від поки ще невідомої функції Белмана $S(x)$, що має неперервні часткові похідні за всіма своїми аргументами.
2. Знаходять функцію Белмана, підставивши знайдене керування в рівняння Белмана й розв'язавши його.
3. Підставляють функцію Белмана у вираз для керування. В результаті зазначеної процедури визначають керування як функцію від фазових координат і часу.

3.3.2 Дискретні системи

Велика кількість задач керування подається графовими моделями. Як приклади можна навести задачу знаходження оптимального маршруту передавання інформації в комп'ютерній мережі або руху транспорту мережею шляхів; задачу оптимального керування трубопровідною мережею тощо. Представлення моделей графами описане в розділі 1.

Метод прямого перебору

Для пошуку оптимального маршруту може бути використаний метод перебору всіх варіантів. Метод перебору є найпростішим і найменш ефективним. Він дозволяє знайти розв'язок у будь-якому випадку (якщо розв'язок взагалі існує), але кількість варіантів, які необхідно перебрати, найчастіше надто велика. Легко довести, що при кількості вершин графа системи n відповідна кількість варіантів буде $n!$.

Метод гілок та границь

Метод дозволяє скоротити кількість варіантів у порівнянні з методом перебору. Застосовується в задачах, де критерій пошуку є монотонно зростаючою функцією кількості ребер.

Ідея методу: знаходиться перший шлях, що задовольняє умову задачі, він приймається за границю. Далі робимо крок назад і шукаємо іншу гілку, яка веде до мети. При пошукові іншої гілки з додаванням до неї чергового ребра перевіряємо, чи не став вже цей частковий шлях довшим за початковий (порівнюємо з границею). Якщо став, то далі нарощувати його немає сенсу – він вже заздалегідь гірший, і всі наступні варіанти (гілки), що мають своїм початком цей частковий шлях, не розглядаються.

Алгоритм Дейкстри

Алгоритм Дейкстри дозволяє знайти найкоротші шляхи від заданої вершини до кожної з решти вершин графа.

Ідея алгоритму: якщо безпосередній шлях від вершини V_i до V_j більший за шлях $Vi \rightarrow Vk \rightarrow V_j$, то шлях $Vi \rightarrow V_j$ заміняють на шлях $Vi \rightarrow Vk \rightarrow V_j$.

Програма, що працює за алгоритмом Дейкстри, наведена у додатку.

Задачі керування потоками в мережах

У вузькому розумінні мережею називають зважений граф, в якому задані пропускні спроможності ребер c(x,y), де x – початкова вершина ребра, y – кінцева вершина. Прикладами таких мереж є водогінна мережа, комп'ютерна мережа, мережа автомобільних доріг тощо. У таких мережах через ребра проходять потоки (води, даних, автомобілів тощо). Значення потоків визначаються функцією потоку f(x,y), причому

$$f(x,y) \le c(x,y). \tag{3.1}$$

В мережі існує три типи вузлів:

- витік S, для якого $\sum_x f(S,x) \ge \sum_x f(x,S)$; (3.2)

- стік T, для якого $\sum_x f(T,x) \le \sum_x f(x,T)$; (3.3)

- проміжні вузли V, для яких $\sum_x f(V,x) = \sum_x f(x,V)$ (3.4)

Очевидно $\sum_x f(S,x) \ge \sum_x f(x,T)$, тобто скільки потоку вийшло з витоку, стільки ж увійшло до стоку.

Значна кількість задач керування потоками зводиться до задачі про максимальний потік.

Знайти такий розподіл потоків у мережі, який забезпечить максимальний сумарний потік з витоку до стоку при виконанні умов (3.1) - (3.4).

Для розв'язання задачі про максимальний потік найчастіше використовується алгоритм Форда-Фалкерсона.

1. Почати з тривіальної функції потоку f(x,y)=0, яка задовольняє умови (3.1)-(3.4).
2. Знайти збільшуючий ланцюг мінімальної довжини, тобто ланцюг з S в T, у якому можна збільшити потік. Для цього в усіх ребрах I ланцюга, спрямованих в напрямку $S \rightarrow T$ повинна виконуватись умова $f(x,y)<c(x,y)$, а в усіх ребрах R, спрямованих в напрямку $S \leftarrow T$ повинна виконуватись умова $f(x,y)>0$;
3. Знайти величину збільшення потоку $\Delta=\min[(c-f)_I, f_R]$, і в усіх ребрах I збільшити потік на Δ, а в усіх ребрах R зменшити на Δ;
4. Знайти наступний збільшуючий ланцюг і повторити п.3.

Програма, що реалізує алгоритм Форда-Фалкерсона, наведена у додатку.

Лінійне програмування

Більшість задач оптимального керування складними системами описується лінійними моделями, критеріями якості і обмеженнями.

Задачами лінійного програмування (ЛП) називають оптимізаційні задачі, в яких обмеження і цільова функція є лінійними.

Постановка задачі лінійного програмування.

Необхідно знайти такі значення змінних $x_1, x_2, \ldots, x_n$, які обертають в максимум цільову функцію $Z = \sum_{j=1}^{n} c_j \cdot x_j$ при обмеженнях:

$$\begin{array}{ccccccccccc} a_{11}\cdot x_1 & + & a_{12}\cdot x_2 & + & \ldots & + & a_{1n}\cdot x_n & \le & b_1 \\ a_{21}\cdot x_1 & + & a_{22}\cdot x_2 & + & \ldots & + & a_{2n}\cdot x_n & \le & b_2 \\ \ldots & & \ldots & & \ldots & & \ldots & & \ldots \\ a_{m1}\cdot x_1 & + & a_{m2}\cdot x_2 & + & \ldots & + & a_{mn}\cdot x_n & \le & b_m \end{array}, \qquad (3.5)$$

де $x = (x_1, \ldots, x_n)$ – вектор невідомих; $b = (b_1, \ldots b_m)$ – вектор обмежень;
$c = (c_1, \ldots, c_n)$ – вектор коефіцієнтів цільової функції.

Приклад 1.

Розв'язати задачу лінійного програмування: $Z = x_1 - 3x_2 - 2x_3 \rightarrow max$ *при обмеженнях*

$x_1 - x_2 - 2x_3 < 5,$
$2x_1 - 3x_2 + x_3 < 3,$
$2x_1 - 5x_2 + 6x_3 < 5.$

Для ефективного розв'язання ЛП використовують симплекс-метод. Застосування симплекс-методу вимагає представлення обмежень в стандартній формі:

$$\begin{array}{ccccccccccc} a_{11}\cdot x_1 & + & a_{12}\cdot x_2 & + & \ldots & + & a_{1n}\cdot x_n & + & x_{n+1} & = b_1 \\ a_{21}\cdot x_1 & + & a_{22}\cdot x_2 & + & \ldots & + & a_{2n}\cdot x_n & + & x_{n+2} & = b_2 \\ \ldots & & \ldots & & \ldots & & \ldots & & \ldots & \ldots \\ a_{m1}\cdot x_1 & + & a_{m2}\cdot x_2 & + & \ldots & + & a_{mn}\cdot x_n & + & x_{n+m} & = b_m \end{array}, \qquad (3.6)$$

$$x_i \ge 0,\ i = \overline{1, n+m},\ b_j \ge 0,\ j = \overline{1, m}.$$

Для розв'язування задачі за симплексом необхідно перейти від стандартної форми (3.6) до канонічної (3.7), в якій m змінних входять з одиничними коефіцієнтами в одне рівняння системи, а з нулевими – в інші. Ці змінні називаються базисними, а решта (n-m) називають небазисними.

$$\sum_{l=1}^{n} a_{1l} x_l + x_{n+1} = b_1$$

$$\sum_{l=1}^{n} a_{2l} x_l + x_{n+2} = b_2,\ x_i \ge 0,\ b_j \ge 0. \qquad (3.7)$$

$$\ldots$$

$$\sum_{l=1}^{n} a_{ml} x_l + x_{n+m} = b_m$$

Для переходу від системи обмежень у вигляді нерівностей до системи обмежень в вигляді рівностей використовують надлишкові змінні. Якщо знак нерівності $\leq$, то надлишкову змінну додають, інакше – віднімають.

Алгоритм симплекс-методу:

1. Перейти від стандартної форми до канонічної.

Приклад 1.

$Z=x_1-3x_2-2x_3+0x_4+0x_5+0x_6$

$x_1-x_2-2x_3+x_4=5,$

$2x_1-3x_2+x_3+x_5=3,$

$2x_1-5x_2+6x_3+x_6=5.$

2. Заповнити первинну симплекс-таблицю 3.1.

Таблиця 3.1

Симплекс-таблиця

		c_j						
		c_1	...	c_n	c_{n+1}	...	c_{n+m}	
c_B	*Базис*	x_1	...	x_n	x_{n+1}	...	x_{n+m}	*Сталі,* b_j
0	x_{n+1}	a_{11}	...	a_{1n}	a_{1n+1}	...	a_{1m}	b_1
...	...	...	...	...	...	...	...	...
0	x_{n+m}	a_{m1}	...	a_{mn}	a_{mn+1}	...	$a_{m\,m+n}$	b_m
$\bar{c}$ *-рядок*		c_1	...	c_n	0		0	Z

В табл. 3.1 c_B – значення оцінок змінних c_j базису; $\bar{c}$ -рядок – вектор відносних оцінок, який розраховується за правилом скалярного добутку (3.8); Z – значення цільової функції $Z = \bar{c}_B \cdot \bar{b}^T$;

$$\bar{c}_j = c_j - c_B \bar{P}_j, \qquad (3.8)$$

де $\bar{P}_j$ – j-ий стовпець в канонічній системі, що відповідає базису, який розглядається.

Таблиця 3.2

Первинна симплекс-таблиця

		c_j						
		1	-3	-2	0	0	0	
c_B	*Базис*	x_1	x_2	x_3	x_4	x_5	x_6	*Сталі,* b_j
0	x_4	1	-1	-2	1	0	0	5
0	x_5	2	-3	1	0	1	0	3
0	x_6	2	-5	6	0	0	1	5
$\bar{c}$ *-рядок*		1	-3	-2	0	0	0	0

3. Якщо всі оцінки $\bar{c}_j$ недодатні, то поточний допустимий базисний

розв'язок є оптимальним і наступний крок 7. Інакше крок 4.

У нас є одна додатна відносна оцінка змінної x_1, тому переходимо до кроку 4.

4. Ввести в базис небазисну змінну x_s з найбільшим значенням $\bar{c}_j$.

Вводимо в базис змінну x_1, табл.3.2.

5. Вивести з базису змінну, для якої відношення b_i/a_{is} буде мінімальним.

Розраховуємо відношення b_i/a_{is} і вибираємо базисну змінну з мінімальним значенням співвідношення, це змінна x_5, тобто цю змінну ми замінюємо на x_1.

Таблиця 3.3

Значення співвідношень b_i/a_{is}

Базис	x_1	*Сталі,* b_i	b_i/a_{is}
x_4	*1*	*5*	*5*
x_5	***2***	***3***	***1,5***
x_6	*2*	*5*	*2,5*

6. Виконати елементарні перетворення (помножити одне рівняння системи на певне число і додати до іншого, щоб одержати нову базисну змінну) для одержання нового допустимого базисного розв'язку і записати нову симплекс-таблицю. Далі крок 3.

Шляхом елементарних перетворень робимо змінну x_1 базисною, щоб вона входила з одиничним коефіцієнтом в одне рівняння системи, а з нулевими – в інші. Для цього рядок з x_1 ділимо на 2 і вносимо його в нову симплекс-таблицю, а потім, наприклад, віднімаємо від першого рядка другий і т.д.

Таблиця 3.4

Друга симплекс-таблиця.

c_B	*Базис*	c_j						*Сталі,* b
		1	*-3*	*-2*	*0*	*0*	*0*	
		x_1	x_2	x_3	x_4	x_5	x_6	
0	x_4	*0*	*-0.5*	*-2.5*	*1*	*-0.5*	*0*	*3.5*
1	x_1	*1*	*-1.5*	*0.5*	*0*	*0.5*	*0*	*1.5*
0	x_6	*0*	*-2*	*5*	*0*	*-1*	*1*	*5*
$\bar{c}$ *-рядок*		*0*	*-1.5*	*-2.5*	*0*	*-0.5*	*0*	*1.5*

Оскільки всі відносні оцінки недодатні, то задача розв'язана.

Отже, оптимальний базисний розв'язок $x_1=1,5$, $x_2=x_3=x_5=0$, $x_4=3,5$, $x_6=5$, оптимальне значення цільової функції дорівнює 1,5.

7. Кінець.

3.4. Прогнозуючі системи. Адаптивні системи

Адаптивні системи – відрізняються від оптимальних тим, що параметри системи та інші її характеристики не повністю відомі або можуть змінюватися. Адаптація – це пристосування до зміни параметрів.

В залежності від того, що саме змінюється в системі, розрізняють три типи адаптивних систем:

1. Системи із змінними параметрами (системи із самоналагоджуванням);
2. Системи із змінною структурою (системи із самоорганізацією);
3. Системи із змінним алгоритмом (системи із самонавчанням).

Необхідність використання трьох типів адаптивних систем зумовлена принципом адекватної складності Ешбі: доведено, що складність системи керування повинна бути не меншою за складність керованого об'єкта.

Існує два способи компенсації недостатньої інформації про стани системи:

1. Ідентифікація системи (активна або пасивна).
2. Пошукова адаптація.

Активна ідентифікація – це визначення параметрів та математичної моделі системи в цілому за допомогою спеціальних тестових сигналів в штучно створених умовах.

Пасивна ідентифікація – це визначення параметрів системи у процесі її нормального функціонування шляхом вимірювання та математичної обробки вхідних впливів та вихідних станів.

Адаптація може здійснюватися в середньому, тобто на основі результатів статистичної обробки або на основі миттєвих значень сигналів і параметрів стану.

Узагальнена структурна схема адаптивної системи наведена на рис.3.5.

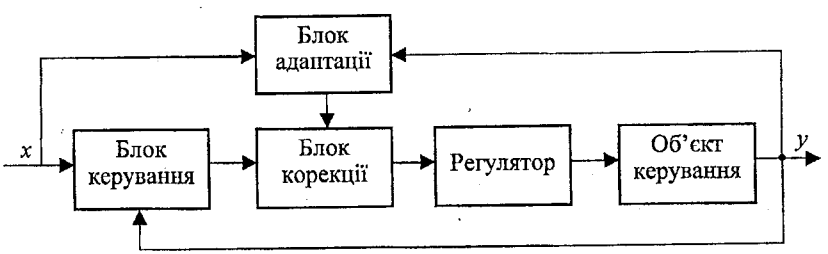


Рис. 3.5. Узагальнена схема адаптивної КСК

Узагальнений алгоритм адаптації на основі пасивної статистичної ідентифікації наведений на рис. 3.6.

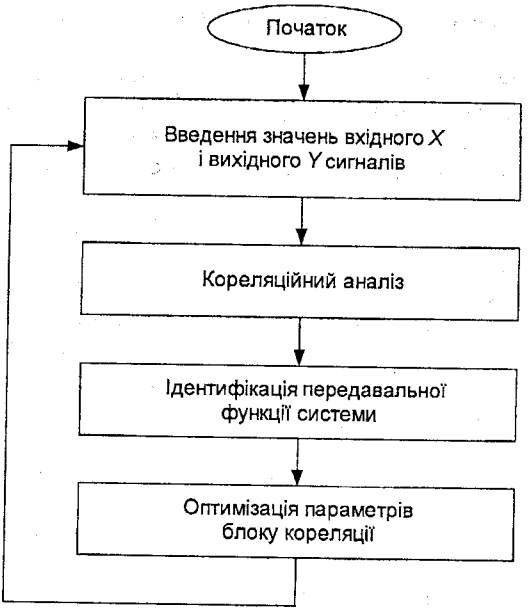


Рис. 3.6. Алгоритм статистичної адаптації

У лінійних системах з високою швидкодією кореляційний аналіз та ідентифікація найчастіше здійснюються за допомогою спеціалізованого сигнального процесора за схемою рис. 3.7.

Ідентифікація за схемою рис. 3.7. ґрунтується на відомому співвідношенні

$$Suy(w)/Suu(w)=W_0(w).$$

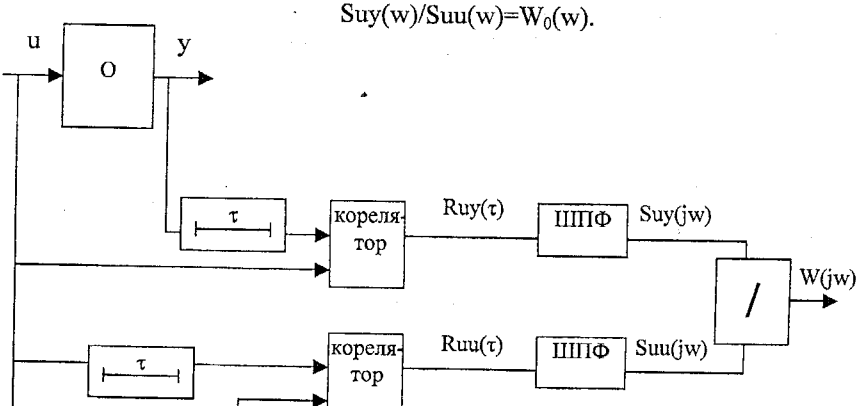


Рис. 3.7. Схема статистичної ідентифікації

Найсуттєвіша відмінність між оптимальними та адаптивними системами полягає в тому, що при необхідності перевести систему з початкового стану в кінцевий в оптимальній системі задача оптимізації розв'язується один раз, оскільки умови задачі заздалегідь відомі. В адаптивній системі під час керування уточнюються характеристики системи і відповідно задача оптимізації розв'язується багатократно.

Але враховуючи, що будь-яка ідентифікація не може бути абсолютно точною, відповідно і адаптивна система наближається до оптимальної, але ніколи абсолютно оптимальною не стане.

3.5. Керування розподіленими системами

У розподілених системах (об'єктах) крім традиційних задач керування виникає ряд специфічних задач. Більшість з них можуть бути подані як задачі на графах, які зводяться до класичних задач керування маршрутом руху за певним критерієм і керування потоками у мережі.

Розподілені системи поділяються на системи з розподіленими і з зосередженими параметрами. *Систему з розподіленими параметрами* можна розділити на агрегати, які характеризуються тими ж параметрами, що і система в цілому; кожен агрегат можна знов розділити на ще менші агрегати і т.д. *Система з зосередженими параметрами* складається з агрегатів, які або є неподільними, або характеризуються іншими параметрами і функціонують інакше, ніж система в цілому. Зустрічаються також комбіновані системи, в яких частина агрегатів може розглядатися як підсистеми з розподіленими параметрами, а частина як з зосередженими.

Перехід при моделюванні систем з розподіленими параметрами до границі їх подрібнення на найменші агрегати приводить до моделей у вигляді диференціальних рівнянь. Моделі розподілених систем також повинні враховувати *затримки при передаванні впливів* від одного агрегата до іншого. В системах з розподіленими параметрами це забезпечується використанням *диференціальних рівнянь у часткових похідних.* Найвідомішими моделями такого типу є система рівнянь Максвелла, хвильове рівняння тощо.

Моделі розподілених систем широко використовуються у різних галузях математичної фізики. Найвідомішими прикладами таких моделей є модель статики механічної системи при просторово розподіленому навантеженні, модель розповсюдження тепла у просторі, модель дифузії, моделі геофізичних процесів тощо.

В системах з зосередженими параметрами затримки описуються передавальними функціями

$$W_з(p) = e^{-p\tau}, \qquad (3.9)$$

де τ - часова затримка.

В дискретних системах одним з найсуттєвіших є питання синхронізації процесів в агрегатах, що взаємодіють. В синхронних системах для цього в моделі відображують кількість тактів, що витрачається на функціонування агрегатів. Найчастіше такі моделі представляються часовими діаграмами.

Сьогодні практично всі великі програмні системи є розподіленими. *Розподілена система* – система, у якій обробка інформації зосереджена не на одній обчислювальній машині, а розподілена між декількома комп'ютерами.

Існує п'ять основних характеристик розподілених систем.

1. *Спільне використання ресурсів.*
2. *Відкритість.* Це можливість розширення системи шляхом додавання нових ресурсів.
3. *Паралельність.*
4. *Масштабованість.* Під масштабованістю розуміється можливість додавання нових властивостей і методів.
5. *Відмовостійкість.* Розподілені системи у випадку помилки можуть підтримувати часткову функціональність.

Розподілені системи володіють і рядом недоліків.

1. *Складність.* Набагато складніше зрозуміти й оцінити властивості розподілених систем в цілому, їх складніше проектувати, тестувати і обслуговувати.
2. *Безпека.* У розподіленій системі набагато складніше підтримувати безпеку.
3. *Керованість.* Помилки на одній ділянці можуть поширитися непередбаченим чином на інші.
4. *Непередбачуваність.* Реакція розподілених систем на деякі події непередбачена і залежить від повного завантаження системи, її організації.

Однією з найважливіших при побудові моделей розподілених систем є проблема *спостережуваності.* Визначення стану системи в процесі моделювання здійснюється за допомогою системи сенсорів, як показано на рис. 3.8.

При дослідженні спостережуваності визначається мінімальна кількість та розташування точок контролю, які забезпечують можливість визначення стану системи з необхідною точністю. На жаль, не завжди існує такий набір, для якого система контролю може бути технічно реалізована. У такому випадку модель розподіленої системи будується з використанням певних припущень і матиме відповідну невизначеність.

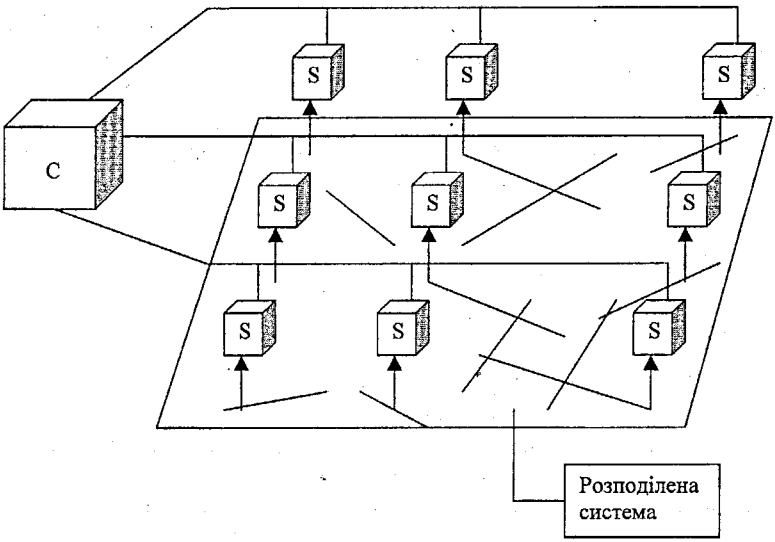


Рис. 3.8. Схема дослідження розподіленої системи:
S – сенсори, C – контролер

3.6. Керування в умовах невизначеності

Системи керування завжди працюють в умовах певної невизначеності.

Врахування невизначеності приводить до суттєвого ускладнення систем керування. Але у багатьох випадках нехтування невизначеністю приводить не просто до певних похибок, а до втрати керованості об'єкта взагалі.

Невизначеність у СК може мати переважно стохастичну чи нечітку природу.

3.6.1. Статистичні методи прийняття рішень

В умовах стохастичної невизначеності ефективне використання статистичних методів прийняття рішень.

Для того, щоб вибрати те або інше правило прийняття рішення, необхідно керуватися певними критеріями. Існує кілька таких критеріїв: критерій максимуму правдоподібності, критерій максимуму апостеріорної ймовірності, критерій ідеального спостерігача, критерій Неймана - Пірсона, критерій мінімального ризику (критерій Байєса) і мінімаксний

критерій.

Розглянемо їх детальніше.

Критерій максимуму правдоподібності формулюється в такий спосіб: приймається те значення параметра X, для якого функція правдоподібності $L(X)$ максимальна.

Відповідно до даного критерію методика прийняття рішення зводиться до таких кроків: обчислюються функції правдоподібності $L(x_1)$ і $L(x_0)$, визначається відношення правдоподібності λ й залежно від того, більше, дорівнює або менше λ одиниці, приймається відповідне рішення:

- якщо $\lambda = \dfrac{L(x_1)}{L(x_0)} > 1$, то $x = x_1$;

- якщо $\lambda = \dfrac{L(x_1)}{L(x_0)} \le 1$, то $x = x_0$.

Практична перевага даного критерію полягає в тому, що при його застосуванні не потрібно знання апріорних імовірностей $p(x_1)$ і $p(x_0)$.

За критерієм максимуму апостеріорної ймовірності процедура прийняття рішення така ж, як і відповідно до критерію максимуму правдоподібності, але відмінність полягає в тому, що в першому випадку відношення правдоподібності порівнюється з одиницею, а у другому випадку – з відношенням апріорних імовірностей $p(x_0)/p(x_1)$. При наявності апріорних даних $p(x_1)$ і $p(x_0)$ доцільно застосовувати критерій максимуму апостеріорної ймовірності, тому що при цьому є можливість користуватися додатковою інформацією. Слід зазначити, що критерій максимуму правдоподібності є оптимальним з інформаційної точки зору.

Відповідно до **критерію ідеального спостерігача** (критерій Котельникова) приймається та гіпотеза, при якій забезпечується мінімум загальної помилки прийняття рішення. Отже, умова оптимального рішення за критерієм ідеального спостерігача має вигляд

$$p_{пом} = p(x_0)\alpha + p(x_1)\beta = \min,$$

де $p_{пом}$ – загальна безумовна ймовірність помилкового рішення;

α – умовна ймовірність помилки першого роду;

β – умовна ймовірність помилки другого роду.

Критерій Неймана-Пірсона враховує, що помилки першого й другого родів не однаково небезпечні, причому помилка першого роду приводить до таких наслідків, що її ймовірність необхідно обмежити деякою дуже малою величиною. Помилку другого роду бажано при цьому забезпечити мінімальною.

Виходячи з цього, критерій Неймана-Пірсона можна сформулювати в такий спосіб: найкращим рішенням є таке, при якому забезпечується найменша ймовірність помилки другого роду при заданій допустимій імовірності помилки першого роду.

Критерій мінімального ризику (критерій Байєса) враховує не тільки нерівномірність помилок першого й другого родів але й ті наслідки, до яких приводять ці помилки. Для врахування цих наслідків введені вагові коефіцієнти (коефіцієнти ціни помилок) r_{10} й r_{01}.

Усереднена величина

$$r = r_{10} p(x_0)\alpha + r_{01} p(x_1)\beta$$

називається середнім ризиком.

Відповідно до критерію мінімального ризику правило вибору рішення формулюється у такий спосіб: приймається те рішення, яке забезпечує мінімальний ризик.

Мінімаксний критерій є спеціальним випадком критерію мінімального ризику, коли апріорні ймовірності $p(x_1)$ і $p(x_0)$ не задані. Ідея мінімаксного критерію полягає в тому, що забезпечується мінімум ризику при найгіршому співвідношенні апріорних імовірностей.

3.6.2. Експертні методи прийняття рішень

Експертна система (система заснована на знаннях) – це програма для комп'ютера, яка оперує зі знаннями у певній предметній галузі з метою видачі рекомендацій чи рішень проблем [Джексон П. Экспертные системы].

Знання, якими володіє спеціаліст у будь-якій області, можна розділити на формалізовані (точні) і неформалізовані (неточні). Останні є результатом узагальнення багаторічного досвіду роботи і інтуїції фахівців. Вони звичайно є різноманітними евристичними прийомами і правилами.

До задач, які вимагають використання евристичних методів відносяться такі:

- алгоритмічне розв'язання задачі невідоме (хоч, можливо, і існує) чи не може бути використане через обмеження ресурсів ЕОМ (часу, пам'яті);
- задача не може бути подана в числовому виразі (потрібне символьне подання);
- мета задачі не може бути виражена у термінах точно визначеної цільової функції.

Експертні системи (ЕС) грунтуються на знаннях, можливості їх є наслідком їх бази знань (БЗ), яка постійно нарощується.

Експертні системи в КСК використовуються для вирішення різного роду проблем:

- ЕС, які виконують інтерпретацію, як правило, використовує інформацію від сенсорів для опису ситуації;
- ЕС, які здійснюють прогноз, визначають ймовірні наслідки заданих ситуацій;
- ЕС виконують діагностування, використовуючи опис ситуацій,

характеристики поведінки чи знання про конструкції компонент, щоб встановити імовірні причини неправильного функціонування системи;

- ЕС, які зайняті плануванням, визначають певну послідовність дій;
- ЕС, які виконують налагодження, пропонують певні дії для виправлення неправильної поведінки об'єкта.

Типова ЕС складається з таких основних компонентів (рис. 3.9): інтерпретатора, робочої пам'яті (РП), бази знань (БЗ), компонентів отримання знань пояснювального та діалогового.

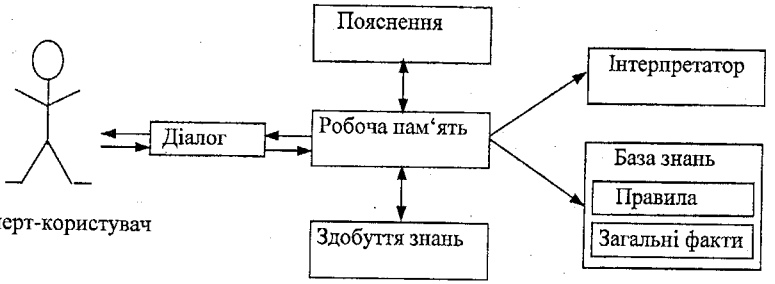


Рис. 3.9. Схема узагальненої експертної системи

Робоча пам'ять призначена для зберігання вихідних і проміжних даних задачі, яка розв'язується в даний момент.

База знань в ЕС призначена для зберігання довгострокових даних, які описують область, що розглядається, і правил, які описують доцільні перетворення даних цієї області.

Інтерпретатор, використовуючи вихідні дані з РП і знання з БЗ, формує таку послідовність правил, застосування яких до вихідних даних приводить до розв'язання задачі.

Компонент здобування знань автоматизує процес наповнення ЕС знаннями, який здійснюється експертом.

Пояснювальний компонент пояснює, як система отримала розв'язок задачі (чи чому вона не отримала розв'язку) і які знання вона при цьому використовувала, що полегшує експерту тестування системи і збільшує довіру користувача до отриманого результату.

Діалоговий компонент орієнтований на організацію дружнього спілкування зі всіма категоріями користувачів як в ході розв'язання задач, так і здобуття знань, пояснення результату роботи.

Експертна система працює в двох режимах: отримання знань і розв'язання задач.

Подання знань найчастіше здійснюється з використанням *правил*.

Правила подаються у вигляді тверджень типу ЯКЩО – ТО. Коли чистина правила ЯКЩО задовольняє факти, то дія, вказана в частині ТО, виконується. Дії правил можуть полягати в модифікації набору фактів в базі знань, наприклад в додаванні нового факту. Нові факти, які додані до бази знань, самі можуть бути використані для зіставлення з частинами правил ЯКЩО.

Семантична мережа застосовується для подання знань мережевої структури. Семантичні мережі складаються з точок, які називаються вузлами, та дуг, які їх зв'язують і описують відношення між ними. Вузли в семантичній мережі відповідають об'єктам, концепціям чи подіям. Дуги можуть бути визначені різними методами, які залежать від виду представлених знань.

Елементи нижчого рівня в мережі можуть наслідувати властивості елементів вищого рівня. Це заощаджує пам'ять, оскільки інформацію про подібні вузли не потрібно повторювати в кожному вузлі мережі. Замість цього вона може розміщуватись в одному центральному вузлі мережі.

Фрейм за своєю організацією в більшості схожий з семантичною мережею. Фрейм – це мережа вузлів і відношень, організованих ієрархічно, де верхні вузли представляють загальні поняття, а нижні вузли окремий випадок цих понять.

Висновки на основі баз знань можуть здійснюватися на основі чіткої або нечіткої логіки.

При застосовуванні чіткої логіки процес зіставлення з фактами частин ЯКЩО правило породжує те, що називається ланцюгом висновків. Ланцюги висновків експертної системи можуть бути пред'явлені користувачеві, що допомагає зрозуміти, як система досягає своїх висновків.

Нечіткий алгоритм керування складається з перетворення вхідних змінних нечіткого регулятора в його вихідні змінні за допомогою таких взаємозалежних процедур:

- перетворення вхідних фізичних змінних нечіткого регулятора, одержуваних від вимірювальних датчиків з об'єкта керування в безрозмірні відносні змінні;
- фазифікація, тобто перетворення відносних значень на нечітку форму;
- нечіткий логічний висновок щодо доцільного керування;
- дефазифікація, тобто перетворення нечіткого результату на чітку форму;
- перетворення вихідних безрозмірних відносних змінних нечіткого регулятора у фізичні керуючі змінні.

Нечітким логічним висновком називається одержання висновку у вигляді нечіткої множини, що відповідає поточним значенням входів, з використанням нечіткої бази знань і нечітких операцій.

Основу нечіткого логічного висновку становить композиційне правило Заде: якщо відомо нечітке відношення $\widetilde{R}$ між вхідною (x) і вихідною (y) змінними, то при нечіткому значенні вхідної змінної $x = \widetilde{A}$, нечітке значення вихідної змінної визначається так:

$$y = \widetilde{A} \circ \widetilde{R},$$

де $\circ$ – максмінна композиція.

Нечіткий логічний висновок Мамдані

Нечіткий логічний висновок за алгоритмом Мамдані виконується за нечіткою базою знань:

$$\bigcup_{p=1}^{k_j} \left(\bigcap_{i=1}^{n} x_i = a_{i,jp} \text{ з вагою } w_{jp} \right) \to y = d_j,\ j = \overline{1,m},$$

у якій значення вхідних і вихідних змінних заданих нечіткими множинами.

Введемо такі позначення, необхідні для подальшого викладу матеріалу:

$\mu_{jp}(x_i)$ – функція належності входу x_i нечіткому терму $a_{i,jp}$, тобто,

$$a_{i,jp} = \int_{\underline{x_i}}^{\overline{x_i}} \mu_{jp}(x_i)/x_i,\ x_i \in \lfloor \underline{x_i}, \overline{x_i} \rfloor.$$

$\mu d_j(y)$ – функція належності виходу y нечіткому терму d_j, тобто

$$d_j = \int_{\underline{y}}^{\overline{y}} \mu d_j(y)/y,\ y \in \lfloor \underline{y}, \overline{y} \rfloor.$$

Ступінь належності вхідного вектора $X^* = (x_1^*, x_2^*, \ldots x_n^*)$ нечітким термам d_j з бази знань розраховується в такий спосіб:

$$\mu_{d_j}(X^*) = \bigvee_{p=1,k_j} w_{jp} \cdot \bigwedge_{i=1,n} \left[\mu_{jp}(x_i^*) \right],\ j = \overline{1,m},$$

де $\vee(\wedge)$ – операція з s-норми* (t-норми), тобто із множини реалізацій логічної операції АБО (І). Найбільше часто використовуються такі реалізації: для операції АБО – знаходження максимуму і для операції І – знаходження мінімуму.

У результаті одержуємо таку нечітку множину $\widetilde{y}$, що відповідає вхідному вектору X^*:

$$\widetilde{y} = \frac{\mu_{d_1}(X^*)}{d_1} + \frac{\mu_{d_2}(X^*)}{d_2} + \ldots + \frac{\mu_{dm}(X^*)}{d_m}.$$

Нехай, наприклад, дані про незалежні нечіткі аргументи оцінені *експертами* і занесені до таблиці 3.5. Якісні фактори можуть бути оцінені

термами {"відсутній", "малий", "нижче середнього", "середній", "вище середнього", "великий"}, яким для зручності ставиться у відповідність шкала з діапазону (1- 6).

Таблиця 3.5

Початкові дані

Номер правила	Вхідні дані					Результат
j	x_1	x_2	x_3	x_4	x_5	y
1	5	4	4	5	5	4
2	2	3	1	5	4	2
3	3	3	2	4	5	3
4	5	5	5	5	4	4
5	2	3	1	5	3	1
6	2	3	2	4	3	2
7	4	4	4	6	2	6
8	4	3	4	5	3	3
9	4	4	3	3	2	5
10	3	3	5	3	4	3

Припустимо, що всі експерти, які надавали наведені дані, мають однаковий *ступінь впевненості* в кожному з них, який можна подати у вигляді трикутної функції належності

$$\mu_i(x)=\begin{cases}\dfrac{D/2-x_i+x}{D/2}, & \max\left(1,x_i-D/2\right)\le x\le x_i\\ \dfrac{D/2+x_i+x}{D/2}, & x_i\le x\le \min\left(D,x_i+D/2\right),\end{cases}$$

де D – діапазон *шкали* (у нашому випадку D=6).

Нехай тепер задана інша *експертна оцінка* аргументів табл. 3.6.

Таблиця 3.6

Експертна оцінка аргументів

x_{10}	x_{20}	x_{30}	x_{40}	x_{50}	y
3	4	2	2	5	?

Визначимо для цього набору аргументів *функцію належності* результату.

Користуючись даними таблиці 3.5, знаходимо $\mu_y(y=1...6)$.

$$\mu_y(y=1)=\max_{j:y_j=1}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=\max_{j=5}[\min(\mu_{x_{15}=2}(x_{10}=3),\mu_{x_{25}=3}(x_{20}=4),$$

$$\mu_{x_{35}=1}(x_{30}=2),\mu_{x_{45}=5}(x_{40}=2),\mu_{x_{55}=3}(x_{50}=5))]=\max_{j=5}\left[\min(0.7,0.7,0.7,0,1)\right]=$$

$$=\max[0]=0;$$

$$\mu_y(y=2)=\max_{j:y_j=2}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=\max_{j=2,6}\{\min[\mu_{x_{12}=2}(x_{10}=3),\mu_{x_{22}=3}(x_{20}=4),$$

$$\mu_{x_{32}=1}(x_{30}=2),\mu_{x_{42}=5}(x_{40}=2),\mu_{x_{52}=4}(x_{50}=5)],\min[\mu_{x_{16}=2}(x_{10}=3),\mu_{x_{26}=3}(x_{20}=4),$$

$$\mu_{x_{36}=2}(x_{30}=2),\mu_{x_{46}=4}(x_{40}=2),\mu_{x_{56}=3}(x_{50}=5))]\}=$$

$$=\max_{j=2,6}\left[\min_{j=2}(0.7,0.7,0.7,0,0.7),\min_{j=6}(0.7,0.7,1,0.3,0.3)\right]=\max[0,0.3]=0.3.$$

Аналогічно

$$\mu_y(y=3)=\max_{j:y_j=3}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=\max_{j=3,8,10}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=$$

$$=\max\left[\min_{j=3}(0.3,0.7,1,0.3,1),\min_{j=8}(0.7,0.7,0.3,0,0.3),\min_{j=10}(1,0.7,0,0.7,0.7)\right]=$$

$$=\max[0.3,0,0]=0.3;$$

$$\mu_y(y=4)=\max_{j:y_j=4}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=\max_{j=1,4}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=$$

$$=\max\left[\min_{j=1}(0.3,1,0.3,0,1),\min_{j=4}(0.3,0.7,0,0,0.7)\right]=\max[0,0]=0;$$

$$\mu_y(y=5)=\max_{j:y_j=5}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=\max_{j=9}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=$$

$$=\max\left[\min(0.7,1,0.7,0.7,0)\right]=\max[0]=0;$$

$$\mu_y(y=6)=\max_{j:y_j=6}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=\max_{j=7}\left[\min_{i=1}^{5}\mu_{x_{ij}}(x_{i0})\right]=$$

$$=\max\left[\min(0.7,1,0.3,0,0)\right]=\max[0]=0.$$

Таким чином, функція належності має вигляд, показаний на рис.3.10.

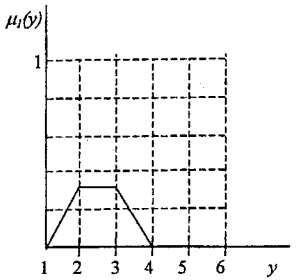


Рис. 3.10. Функція належності результату моделювання

З графіка видно, що найможливіший рівень значення функції для даного прикладу є 2.5, тобто між "малим" і "нижче середнього". Конкретний результат моделювання отримується за допомогою операції *дефазифікації*. Ця операція може здійснюватися різними способами і давати, відповідно, різні результати. Так, наприклад, за результат можна прийняти значення, якому відповідає максимум функції належності або середнє значення, яке отримується аналогічно математичному очікуванню.

Особливістю цієї нечіткої множини є те, що універсальною множиною для нього є терм-множина вихідної змінної y. Такі нечіткі множини називаються нечіткими множинами другого порядку.

Для переходу від нечіткої множини, заданого на універсальній множині нечітких термів $\{d_1, d_2, ..., d_m\}_k$ до нечіткої множини на інтервалі $\lfloor \underline{y}, \bar{y} \rfloor$ необхідно: 1) "зрізати" функції належності $\mu d_j(y)$ на рівні $\mu d_j(X^*)$; 2) об'єднати (агрегувати) отримані нечіткі множини. Математично це записується в такий спосіб:

$$\tilde{y} = \underset{j=1,m}{agg}\left(\int_{\underline{y}}^{\bar{y}} \min\left(\mu_{d_j}(X^*), \mu_{d_j}(y)\right) / y \right),$$

де agg - агрегування нечітких множин, що найбільше часто реалізується операцією знаходження максимуму.

Чітке значення виходу y, що відповідає вхідному вектору X^* визначається в результаті дефазифікації нечіткої множини $\tilde{y}$. Найчастіше застосовується дефазифікація за методом центра ваги:

$$y = \frac{\int\limits_{\underline{y}}^{\bar{y}} y \cdot \mu_{\tilde{y}}(y)dy}{\int\limits_{\underline{y}}^{\bar{y}} \mu_{\tilde{y}}(y)dy},$$

де $\int$ – тут символ інтеграла.

Нечіткий логічний висновок Сугено

Нечіткий логічний висновок за алгоритмом Сугено (іноді говорять алгоритм Такаги-Сугено) виконується за нечіткою базою знань:

$$\bigcup_{p=1}^{k_j}\left(\bigcap_{i=1}^{n} x_i = a_{i,jp} \text{ з вагою } w_{jp}\right) \to y = b_{j,0} + b_{j,1} \cdot x_1 + b_{j,2} \cdot x_2 + \ldots + b_{j,n} \cdot x_n,\ j = \overline{1,m},$$

де $b_{j,i}$ – деякі числа.

База знань Сугено аналогічна базі знань Мамдані за винятком висновків правил d_j, які задаються не нечіткими термами, а лінійною функцією від входів: $d_j = b_{j,0} + \sum\limits_{i=1,n} b_{j,i} \cdot x_i$. Правила в базі знань Сугено є свого роду перемикачами з одного лінійного закону "вхід – вихід" на інший, теж лінійний. Границі підобластей розмиті, отже, одночасно можуть виконуватися кілька лінійних законів, але з різними ступенями. Ступінь належності вхідного вектора $X^* = \left(x_1^*, x_2^*, \ldots x_n^*\right)$ до значень $d_j = b_{j,0} + \sum\limits_{i=1,n} b_{j,i} \cdot x_i$ розраховується в такий спосіб:

$$\mu_{d_j}\left(X^*\right) = \bigvee_{p=1,k_j} w_{jp} \cdot \bigwedge_{i=1,n} \left[\mu_{jp}\left(x_i^*\right)\right],\ j = \overline{1,m},$$

де $\vee(\wedge)$ – операція з s-норми (t-норми), тобто із множини реалізацій логічної операції АБО (І). У нечіткому логічному висновку Сугено найчастіше використовуються такі реалізації трикутних норм: імовірнісне АБО як s-норма й добуток як t-норма.

У результаті одержуємо таку нечітку множину $\tilde{y}$, що відповідає вхідному вектору X^*:

$$\tilde{y} = \frac{\mu_{d_1}\left(X^*\right)}{d_1} + \frac{\mu_{d_2}\left(X^*\right)}{d_2} + \ldots + \frac{\mu_{dm}\left(X^*\right)}{d_m}.$$

Контрольні питання та завдання

1. Для чого використовується
   - П-регулятор?
   - ПІ-регулятор?
   - І-регулятор?
   - ПД-регулятор?
   - ПІД-регулятор?
2. Сформулюйте постановку задачі оптимального керування.
3. Сформулюйте та поясніть принцип максимуму Понтрягіна.
4. У чому сутність методу пошуку в ширину? Його геометрична інтерпретація.
5. Яка структура даних використовується у методі пошуку в ширину?
6. У чому сутність методу пошуку в глибину? Його геометрична інтерпретація.
7. Яка структура даних використовується у методі пошуку в глибину?
8. У чому відмінність пошуку з поверненням від пошуку в глибину?
9. Наведіть приклади потоків різної фізичної природи.
10. У чому ідея алгоритму отримання максимального потоку?
11. Адаптивні системи. Що таке активна та пасивна ідентифікація?
12. Назвіть основні особливості розподілених систем.
13. Що таке нечіткий алгоритм?
14. Охарактеризуйте основні статистичні методи прийняття рішень.
15. Що таке експертні методи та системи?
16. Дайте означення поняттям "база знань", "фрейм", "семантична мережа".
17. Поясніть чим відрізняються методи нечіткого висновку Заде, Мамдані і Сугено.
18. Що таке робастне керування? Назвіть його переваги.

Література

1. Бесекерский В.А., Изранцев В.В. Системы автоматического управления с микро-ЭВМ. – М.: Наука, 1987. – 320с.
2. Дубовой В.М., Квєтний Р.Н. Програмування персональних комп'ютерів систем керування. – Вінниця: ВДТУ, 1999.
3. Дубовой В.М. Моделювання систем контролю та керування: Навчальний посібник. – Вінниця: ВНТУ, 2005. – 175 с.
4. Воронов А.А. Теория автоматического управления. Ч.II. Теория нелинейных и специальных систем автоматического управления. – М.: Высш. школа, 1977. - 288с.
5. Батищев Д.И. Методы оптимального проектирования. – М.: Радио и связь, 1984.
6. Чураков Е.П. Оптимальные и адаптивные системы. – М.: Энергоатомиздат, 1987. – 256 с.

7. Дехтярев Ю. И. Методы оптимизации.- М.:Советское радио, 1980. –13с.
8. Реклейтис Г., Рейвиндран А., Рэгсдел К. Оптимизация в технике. Кн.1.- М.: Мир, 1986.- 320с.
9. Бондарев В.М. и др. Основы программирования. – Харьков: Фолио, 1997.
10. Простое и сложное в программировании. – М.: Наука, 1988.
11. Антонов В.А., Пришвин А.М. Адаптивные системы автоматического управления / Под ред. В.Б. Яковлева. – Л.: Издательство Ленингр. ун-та, 1984. - 204с.
12. Кику А.Г. и др. Адаптивные системы идентификации. – К.: Техника, 1975. - 326с.
13. Чураков Е.П. Оптимальные и адаптивные системы. - М.: Энергоатомиздат, 1987. - 256с.
14. Мартыненко Н.А., Пустыльников Л.М. Конечные интегральные преобразования и их применение к исследованию систем с распределенными параметрами. - М.: Наука, 1985. - 312с.
15. Кузьмин И.В., Кедрус В.А. Основы теории кодирования. – 2-е изд., перераб. и доп. – К.: Вища шк. Головное изд-во, 1986. – 238с.
16. Пугачев В.С. Теория вероятностей и математическая статистика. - М.: Наука, 1979. – 326с.
17. Лапа В.Г. Математические основы кибернетики. - К.: Вища шк.,1974.- 452с.
18. Кофман А. Введение в теорию нечетких множеств.-М.:Радио и связь, 1982. – 432с
19. Аверкин А.Н., Батыршин И.З., Блишун А.Ф., Силов В.Б., Тарасов В.Б. Нечеткие множества в моделях управления и искусственного интеллекта / Под ред. Поспелова Д.А. – М.: Наука, 1986. –312с.
20. Алтунин А.Е., Семухин М.В. Модели и алгоритмы принятия решений в нечетких условиях: Монография. Тюмень: Издательство Тюменского государственного университета, 2000. – 352 с.
21. Борисов А.Н., Алексеев А.В., Крумберг О.А. и др. Модели принятия решений на основе лингвистической переменной – Рига : Зинатне, 1982. - 256с.
22. Жуковин В.Е. Многокритериальные модели принятия решений с неопределенностью. - Тбилиси: Мецниереба, 1983. – 104с.
23. Мелихов А.Н., Бернштейн Л.С., Коровин С.Я. Ситуационные советующие системы с нечеткой логикой. М.: Гл. ред. физ.-мат. Лит., 1990. – 272с.
24. Методы и системы принятия решений. Системы, основанные на знаниях. Под ред. А.Н. Борисова. - Рига. РПИ, 1989. – 175с.
25. Нечеткие множества и теория возможностей. Последние достижения: Пер. с англ./ Под ред. Р. Р. Ягера. – М.: Радио и связь,1986. – 408с.
26. Стогов Г.В., Макшанов А.В., Мусаев А.А. Устойчивые методы обработки результатов измерений. – 1982. - №8. - С.3-46.

4. ЕЛЕМЕНТИ ТА ТЕХНІЧНІ ЗАСОБИ КСК

Технічні засоби систем автоматичного керування (САК) і автоматизованих систем (АСК) суттєво відрізняються. В основі комплексу технічних засобів АСК є комп'ютерні мережі і різноманітні засоби людино-машинного інтерфейсу. В той же час САК більше орієнтовані на взаємодію з об'єктом керування.

Структуру комплексу технічних засобів типової комп'ютеризованої САК показано на рис. 4.1.

На вхід системи керування від об'єкта керування (ОК) надходять вхідні сигнали Y, які сприймаються вимірювальними каналами. Основними складовими цієї підсистеми є різного роду сенсори та перетворювачі сигналів, а також мікроконтролери. При цьому перетворення аналогових фізичних величин, що надходять від ОК, в цифровий код, який підлягає попередній обробці мікроконтролером, здійснюється аналого-цифровими перетворювачами (АЦП). Зв'язок ЕОМ із зовнішніми пристроями забезпечують зовнішні інтерфейси.

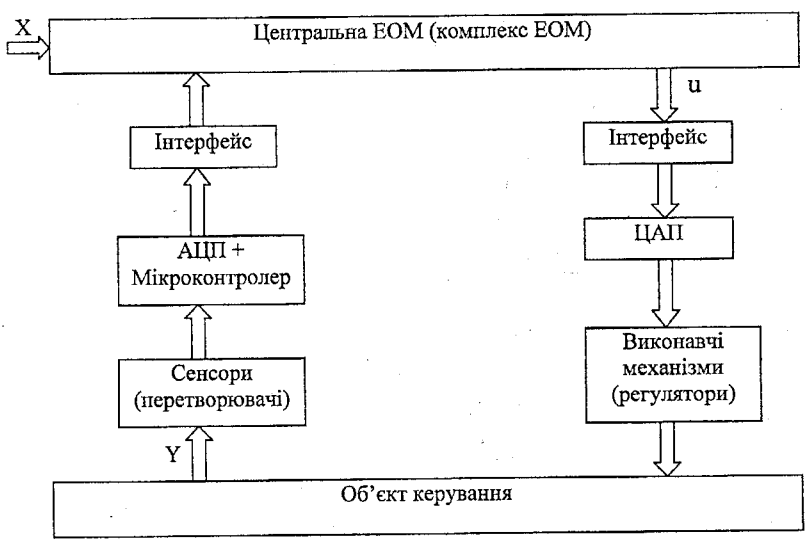


Рис. 4.1. Структура комплексу технічних засобів комп'ютеризованої САК

Мета керування досягається виробленням центральною ЕОМ вектора керуючих сигналів *u* та їх дією на об'єкт через виконавчу підсистему, до якої входять спеціальні виконавчі механізми та регулятори.

Оскільки початковий вектор u представляється у двійковому коді, його перетворення у вектор аналогових керуючих величин здійснюється за допомогою цифроаналогових перетворювачів (ЦАП).

Розглянемо основні типи технічних засобів, що складають САК.

4.1. Вимірювальні перетворювачі

У кожному з елементів САУ здійснюється перетворення інформації за допомогою тих чи інших перетворювачів. У перетворювачах інформації фізичний сигнал одного виду – вхідний параметр x, перетворюється на фізичний сигнал іншого виду – вихідний параметр y. Прикладом перетворювача інформації може бути термоелектроперетворювач (термопара), який перетворює температуру (вхідний сигнал x=t°) на термоелектрорушійну силу (вихідний параметр $y = e_t$).

Найважливішим типом перетворювачів комп'ютерних систем автоматичного керування є сенсори, які призначені для визначення стану об'єкта керування і зовнішніх впливів.

Сенсори можна поділити за родом вхідної фізичної величини:

- сенсори електричних і магнітних величин;
- сенсори механічних величин (в тому числі сенсори гідравлічних і пневматичних величин);
- сенсори теплових величин;
- сенсори оптичних величин;
- сенсори концентрації та складу речовин.

Сенсори, що використовуються у КСК, розподіляються на стандартні та нестандартні, серійні та унікальні.

Стандартні сенсори – сенсори, що занесені у державний реєстр засобів вимірювальної техніки.

Серійні сенсори – сенсори, які масово випускаються фірмами-виробниками.

Унікальні сенсори – сенсори, що створені для окремої системи.

За вихідними сигналами сенсори поділяються на:

- уніфіковані;
- неуніфіковані.

Уніфіковані сенсори – сенсори, вихідні сигнали яких відповідають певному стандарту.

Провідні фірми-виробники Analog Device, Dallas Semiconductors, Texas Instruments, Philips на сьогоднішній день випускають сенсори з цифровими вихідними сигналами. Вони віддають перевагу певним стандартам (M-Bus, CAN тощо), призначеним для створення систем контролю і автоматизації.

Системи M-BUS – це система протоколів та інтерфейсів. Система M-Bus складається з універсальних апаратних елементів і програмних

модулів, спроектованих для керування й контролю за допомогою стандартного керуючого обладнання. Система включає в себе центральні блоки, вузли передачі даних, модулі для підключення обладнання, сенсори (температури, вологості, задимлення, вібрації, доступу). Основні характеристики каналу передачі даних M-Bus:

- можливість використання різних схем розведення кабелю (стандартними є дві виті пари), з'єднання більш ніж з 250 пристроями;
- захист каналу передачі даних від неправильного підключення;
- електроживлення кінцевих пристроїв здійснюється безпосередньо через канал передачі даних M-Bus.
- ці системи дуже низькошвидкісні, що зумовлює застосування проводових ліній зв'язку (звичайні проводи);
- живлення сенсорів та інших периферійних пристроїв здійснюється через ті ж лінії зв'язку, по яких передаються інформативні сигнали. Це дає можливість зменшити кількість проводів і спростити систему.

CAN – система, яка придатна для об'єднання в мережу таких «інтелектуальних» пристроїв, як сенсори та силові приводи з вбудованими мікроконтролерами. На сьогоднішній день існує більш ніж 50 схем контролерів (як анонсованих, так і доступних), що підтримують протокол CAN, від більш як 15 виробників. Використання CAN-технологій в більшості європейських марок автомобілів протягом більш ніж 10 років свідчить про безумовну ефективність схем CAN.

Максимальна швидкість передавання встановлена на рівні 1 Мбіт/с. Цей рівень є придатним для мереж з відстанню між вузлами до 40 м. Для більших відстаней швидкість передавання повинна бути зменшена: для дистанцій до 500 м допустимо порядку 125 Кбіт/с, а для передавання на відстань до 1 км – 50 Кбіт/с.

Мережі CAN можуть використовуватися в якості вбудованих комунікаційних систем для мікроконтролерів, тобто як відкриті системи зв'язку для інтелектуальних пристроїв. Ця система, яка спочатку була розроблена для використання в автомобілях, з розвитком впроваджується у великі промислові системи керування, оскільки ці галузі мають багато схожих рис. В обох випадках спільними є головні вимоги: низька вартість, можливість працювати в умовах посиленого зовнішнього електромагнітного впливу, високий рівень забезпечення функціонування в режимі реального часу і простота використання.

4.2. Цифроаналогові й аналого-цифрові перетворювачі

Цифроаналогові й аналого-цифрові перетворювачі, виконані на базі інтегральних схем, мають малі габарити, високу надійність, точність і

швидкодію. Цифроаналоговий перетворювач (ЦАП) це пристрій, у якого вихідний аналоговий сигнал $U_{вих}$ пов'язаний з цифровим вхідним сигналом C співвідношенням:

$$U_{вих} = U_{ет} C, \tag{4.1}$$

де $U_{ет}$ – еталонна напруга.

Сигнал C являє собою число, що містить певну кількість двійкових розрядів, і може бути представлений у вигляді:

$$C = a_0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + ... + a_n \cdot 2^n, \tag{4.2}$$

де $a_0, a_1, a_2, ..., a_n$ – коефіцієнти відповідних двійкових розрядів, які приймають дискретні значення 1 або 0; n – загальна кількість двійкових розрядів.

На рис. 4.2 зображено схему ЦАП з кроковим резистивним струмозадавальним колом *R-2R*. Тут використовуються резистори з двома номінальними опорами. Співвідношення струмів у паралельних колах задовольняє двійковий закон:

$$I_1 = 2I_2 = 4I_3 = ... = 2^{n-1} I_n. \tag{4.3}$$

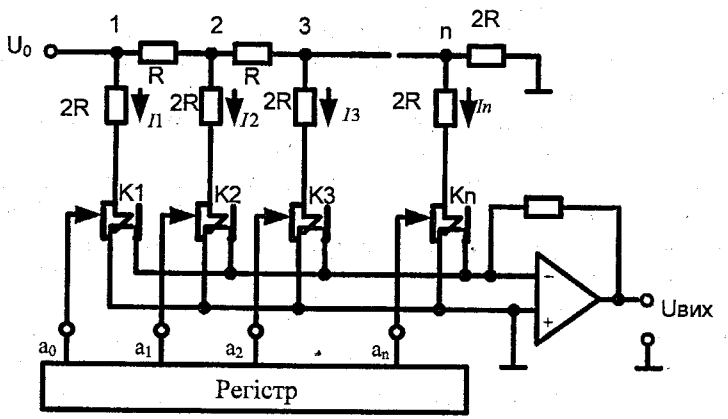


Рис. 4.2. ЦАП з резистивною сіткою

Аналого-цифрові перетворювачі (АЦП) виконують операцію, зворотну ЦАП, тобто перетворюють вхідний аналоговий сигнал у цифровий вихідний сигнал, що представляє собою n-розрядне двійкове число.

На рис. 4.3 представлено одну з схем АЦП. До її складу входять схема порівняння (СП), реверсивний лічильник (РЛ) і ЦАП.

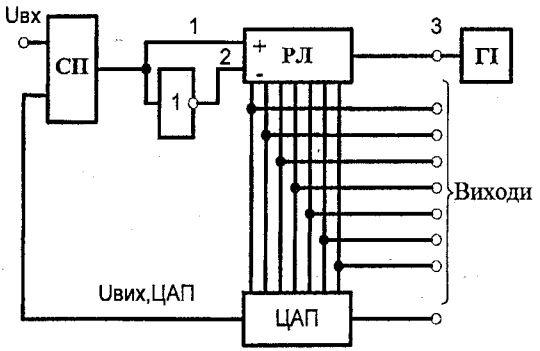


Рис. 4.3. АЦП на основі ЦАП і ревёрсивного лічильника

Для керування лічильником РЛ використовується генератор імпульсів (ГІ). При надходженні імпульсів від ГІ на вхід *3* лічильника РЛ стани тригерів, що входять до його складу, змінюються в залежності від сигналів, що надходять на його входи 1 і 2 від схеми порівняння СП. Схема СП порівнює вхідну напругу $U_{вх}$ з вихідною напругою $U_{вих}$ ЦАП. При $U_{вх} < U_{вих.ЦАП}$. Лічильник РС здійснює підрахунок імпульсів, що надходять від ГІ, у прямому напрямку (підсумовування), при $U_{вх} > U_{вих.ЦАП}-$ в зворотному (віднімання). Таким чином, при $U_{вх} > U_{вих.ЦАП}$ напруга на виході ЦАП зростає, а при $U_{вх} < U_{вих.ЦАП}$ зменшується. Таким чином, в даній схемі РЛ в комбінації *з ЦАП* ніби стежить за значенням $U_{вх}$. Вихідні сигнали лічильника відповідають значенню $U_{вх}$ в цифровій формі.

АЦП послідовної дії в деяких випадках мають недостатню швидкодію. Максимальний час перетворення визначається кількістю розрядів і частотою генератора імпульсів, яка, в свою чергу, вибирається з умови узгодження періоду імпульсів з часом перемикання ключів у ЦАП і спрацьовування схеми порівняння.

Якщо визначальним фактором при побудові АЦП є швидкодія, то застосовується схема паралельної дії. Як видно з наведеної на рис. 4.4. структурної схеми такого перетворювача, весь діапазон можливих значень $U_{вх}$ від 0 до $U_{ет}$ (еталонної напруги) розбитий на 2^n ділянок. Для визначення ділянки діапазону, у якому знаходиться значення $U_{вх}$, використовується 2^n-1 схем СП. Вихідні сигнали схем порівняння надходять на входи шифратора (кодера), на виходах якого в залежності від

кількості схем, що спрацювали, з'являється сигнал у цифровій формі, який відповідає значенню $U_{вх}$ з похибкою не більше $U_{em}/(2^n)$.

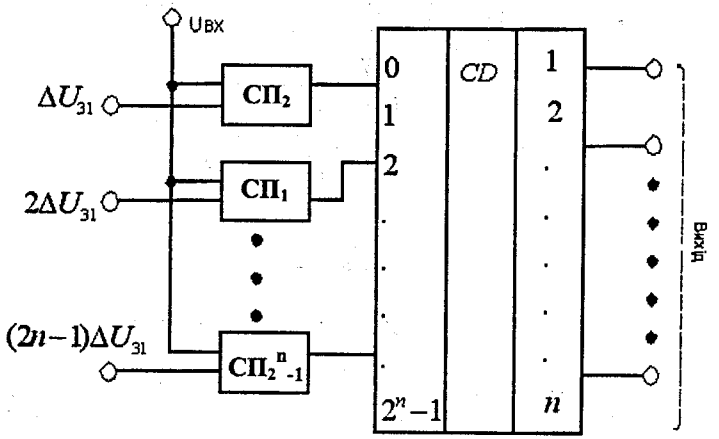


Рис. 4.4. Структурна схема аналого-цифрового перетворювача паралельної дії

Проміжне положення за швидкодією і складністю виконання між АЦП послідовної і паралельної дії займають схеми АЦП, що працюють за принципом порозрядного зрівноважування або послідовного наближення.

4.3. Контролери

Контролери є найважливішою частиною сучасних комп'ютеризованих систем автоматичного керування. Конструктивне виконання контролерів суттєво відрізняється в залежності від призначення і функцій. Це може бути як окремий блок з власним живленням, вбудованими АЦП і ЦАП, портами для введення/виведення потужних аналогових сигналів, великою оперативною і електропрограмованою пам'яттю тощо. Приклад конструктивного виконання такого контролера наведений на рис. 4.5.

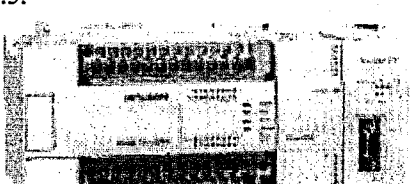


Рис. 4.5. Типовий контролер САУ

Контролери також можуть бути виконані у вигляді окремої мікросхеми, які вбудовуються у сенсори, інтерфейси, підсистеми зв'язку тощо.

Вбудовані мікроконтролери (embedded microcontrollers) або просто мікроконтролери представляють собою однокристальні системи, орієнтовані на виконання, в першу чергу, функцій керування різними пристроями, звідки й отримали таку назву. Використовуються вони досить широко в різних сферах – від сучасної побутової техніки (холодильники, пральні машини, кухонні комбайни тощо) до найскладніших систем керування технологічними процесами та робототехнічними комплексами.

Мікроконтролери сімейства MCS-96/196

Сімейство цих 16-розрядних мікроконтролерів випускається фірмою Intel з 1984 р. Кількість моделей мікроконтролерів сімейства MCS-96 – понад 50. В цьому сімействі можна виділити три основні групи kMOH-мікроконтролерів, кожна з яких має свою оригінальну архітектуру.

Представниками першої групи є мікроконтролери 8XC196KB/KC/ /KD. Вони характеризуються можливістю здійснення прийому й видачі керуючих сигналів у реальному масштабі часу за допомогою спеціалізованого блока швидкого введення/виведення HSIO (High Speed Input/Output).

До другої групи належать мікроконтролери 8XC196KR/KT/KQ/JR/ /JQ/NP/NT/NU/CA/CB, що здійснюють обробку керуючих сигналів у реальному масштабі часу за допомогою спеціального процесора подій EPA (Event Processor Array).

Третю групу становлять мікроконтролери 8XC196MC/MD/MH. Вони оснащені додатковими блоками, що забезпечують генерування трифазних імпульсних сигналів для керування електродвигунами.

На практиці використовуються моделі мікроконтролерів переважно трьох модифікацій, які відрізняються реалізацією постійного запам'ятовуючого пристрою (ПЗП). У першій модифікації ПЗП відсутній взагалі. Друга модифікація містить внутрішній ПЗП ємністю від 4 до 32 Кбайт, який програмується і записується за допомогою фотошаблона в процесі виготовлення мікроконтролера. Третя модифікація містить ПЗП, який одноразово програмується користувачем (OTP ROM) за допомогою програматора.

Більшість мікроконтролерів мають внутрішні ПЗП ємністю до 56 Кбайт та регістрові ОЗП даних ємністю до 1,5 Кбайт. Деякі моделі характеризуються додатковим внутрішнім ОЗП команд ємністю до 512 байт. Загальна ємність пам'яті, що може адресуватися, становить 64 Кбайт або 1 Мбайт. Більшість моделей мають внутрішній 8- або 10-розрядний АЦП з аналоговими входами в кількості від 4 до 14. Всі мікроконтролери містять також по два 16-розрядних лічильники-таймери. Вони забезпечують обробку в реальному масштабі часу сигналів, що надходять на входи мікро-контролера, і формують необхідні вихідні сигнали за допомогою

спеціалізованих блоків HSIO або процесорів подій EPA, які мають від 8 до 20 таких входів/виходів. До складу мікроконтролерів входять також 3-4 порти паралельного введення/виведення даних і 1-2 послідовних двонаправлених порти обміну.

kMOH-технологія, за якою виготовляються більшість мікроконтролерів, забезпечує їх функціонування з максимальною тактовою частотою від 16 до 50 МГц. Споживана потужність не перевищує 500 мВт на максимальній частоті.

Мікроконтролери сімейства MCS-96 мають ідентичну архітектуру, їх відмінності зводяться до різного типу та ємності пам'яті й реалізації периферійних пристроїв. Різні моделі контролерів використовують одне і те ж процесорне ядро і відповідно систему команд та способи адресації.

Найперспективнішими на даний час вважаються мікроконтролери 8XC196NP/NU. Їх процесорне ядро передбачає використання користувачем ємності пам'яті, що адресується, 1 Мбайт, внутрішнього ПЗП до 4 Кбайт і набору периферійних пристроїв: таймерів, процесора подій, універсального послідовного порту. Ці моделі знайшли широке застосування завдяки своїй низькій вартості, досить широкому набору виконуваних функцій, які можуть доповнюватись відповідно до вимог користувачів шляхом підключення зовнішніх пристроїв.

Модель 8XC196NU має порівняно з 8XC196NP удвічі вищу швидкодію за таких самих функціональних можливостей. Така швидкодія є максимальною для мікроконтролерів сімейства MCS-96.

Основними складовими мікроконтролера 8XC196NP є центральний процесор, блок регістрів ємністю 1 Кбайт та блок керування пам'яттю (БКП), який здійснює вибір команд із зовнішньої чи внутрішньої пам'яті, організовує 4-байтну чергу команд (ЧК) і забезпечує звернення до зовнішньої пам'яті даних. Внутрішня пам'ять команд характерна тільки для моделі 8XC196NP. Вона реалізується у вигляді розміщеного на кристалі ПЗП ємністю 4 Кбайт.

Структурну схему мікроконтролерів 8XC196NP наведено на рис. 4.6.

Мікроконтролер розміщується в 100-вивідному корпусі типу QFP чи SQFP, має чотири 8-розрядні порти P1-P4 для зв'язку із зовнішніми пристроями. Порт P4 має виводи тільки від 4 молодших розрядів. Старші розряди P4 зарезервовані для наступних моделей сімейства. Тому P4 використовується як 4-розрядний порт.

Виводи порту P1 можуть бути використані для виконання спеціальних функцій, зокрема для подачі тактових і керуючих сигналів для таймерів 1 і 2, а також введення/виведення сигналів, які керують функціонуванням процесора подій (ПРП).

Зовнішні запити на переривання надходять на входи EXTINT0 і EXTINT1 порту P2 та на два входи EXTINT2, EXTINT3 порту P3. Ці запити разом із внутрішніми обробляються контролером переривань (КПеР) згідно

з їх пріоритетом. У структурі мікроконтролера є також спеціалізований блок обслуговування переривань – периферійний сервер (ПСР), який виконує специфічні види обслуговування: пересилання окремих слів чи цілих блоків, формування імпульсних сигналів заданої частоти і тривалості на виводах EPA0-EPA3. Виводи порту P2 можуть використовуватися також для послідовного введення і виведення даних через блок послідовного обміну (БПО), а також для обміну сигналами, які забезпечують зайняття магістральної шини іншими пристроями (HOLD, HOLDA, BREQ), і видачі синхроімпульсів (CLK OUT).

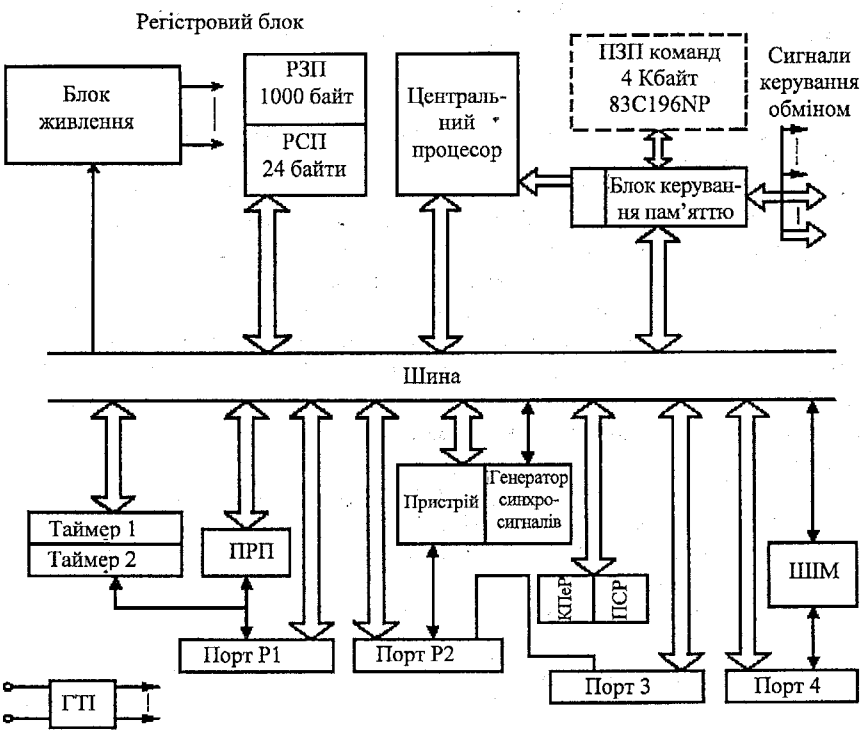


Рис. 4.6. Структура мікроконтролера 8XC196NP

Виводи порту P3 використовуються для виведення сигналів дозволу вибірки CS0-CS5, які надходять з БКП і дають змогу мікроконтролеру звертатися до кількох типів зовнішньої пам'яті, які відрізняються ємністю, швидкодією, розрядністю, режимом звернення.

Три виводи порту P4 використовують як виходи ШІМ, що забезпечує формування послідовності імпульсів заданої частоти і тривалості.

Мікроконтролер може працювати із зниженим енергоспоживанням. Для цього передбачені режими холостого ходу і відключення живлення. Передбачений також режим повного відключення мікроконтролера від зовнішніх ліній зв'язку. При цьому всі лінії, крім ліній синхронізації, живлення і «землі», переводяться у відключений (високоімпедансний) стан. Цей режим використовується для підключення схемного емулятора в процесі налагодження мікропроцесорної системи, а також при тестуванні друкованої плати, на якій розміщено мікроконтролер, чи інших мікросхем, розташованих на цій платі.

Для синхронізації мікроконтролера є можливість використовувати зовнішнє джерело тактових імпульсів чи внутрішній генератор, який функціонує при підключенні кварцового резонатора.

Мікроконтролери сімейства MCS-51/151/251

Дане сімейство мікроконтролерів нараховує майже півсотні мікроконтролерів.

Група контролерів 80C51FA/FB/FC пов'язана з введенням в архітектуру сімейства модуля PCA (programmable counter array) і сторожового таймера WDT (watchdog timer). Модуль PCA призначений для виконання різних операцій обчислення і визначення тривалості інтервалів чАСК, в тому числі при широтно-імпульсній модуляції. Сторожовий таймер забезпечує перезавантаження процесора в разі зависань. «Старші» мікроконтролери (80C51GB) мають вбудований 8-розрядний АЦП та шість паралельних портів. Кардинальне збільшення продуктивності 8-розрядних мікроконтролерів було досягнуто фірмою Intel за рахунок розробки сімейств MCS-151 і MCS-251.

Внутрішня структура мікроконтролера 8051АН, що є вихідною для сімейства MCS-51, містить такий набір функціональних модулів:

- 8-розрядний АЛП з апаратною реалізацією операцій типу множення;
- внутрішню пам'ять програм ємністю 4 Кбайт і ОЗП даних ємністю 128 байт;
- чотири універсальних програмованих паралельних 8-розрядних порти введення/виведення з можливістю реалізації певних альтернативних функцій;
- два 16-розрядних програмованих лічильники-таймери;
- дуплексний послідовний порт.

Ці апаратні пристрої та сукупність функцій, що реалізуються мікроконтролерами сімейства 8051, є ефективним засобом збирання, обробки інформації і керування об'єктами. Структура мікроконтролерів має напівзакритий характер. Аналогом мікросхеми 8051АН є мікросхема К1816ВЕ51.

Архітектура мікроконтролерів MCS-251 є вдосконаленою архітектурою MCS-51. В її основу покладено систему команд попереднього сімейства і набір блоків введення/виведення, зокрема три таймери-лічильники, послідовний порт, блок PCA і сторожовий таймер. Збільшення продуктивності досягається за рахунок введення нових команд, збільшення адресного простору, використання механізму конвеєризації в центральному процесорі, стиснення циклів обміну по магістралі.

Центральний процесор мікроконтролерів MCS-251 побудований на основі конвеєра команд і регістрового файлу. Це дає змогу виконувати декілька команд за один машинний такт, тоді як процесор мікроконтролера 8XC51FX виконує одну команду за шість тактів. Система команд процесора контролерів MCS-251 доповнена інструкціями, які оперують 16- і 32-розрядними операндами.

Найважливішими особливостями архітектури мікроконтролерів MCS-251 є:

- конвеєр команд і регістровий файл;
- розширений набір команд;
- внутрішній ОЗП ємністю до 1 Кбайт;
- стек до 64 Кбайт;
- можливості порядкової вибірки команд, введення стану очікування на магістралі, формування 17-розрядної адреси;
- внутрішня пам'ять з одноразовим записом ємністю до 16 Кбайт;
- три 16-розрядних таймери-лічильники;
- програмований послідовний порт;
- блок PCA з п'ятьма модулями порівняння-захоплення;
- виділений сторожовий таймер.

Основу структури сімейства мікроконтролерів MCS-251 становить ядро (core). Воно визначає найважливіші параметри архітектури, зокрема набір команд, тактові частоти процесора, механізми конвеєризації та переривань. Ядро є спільною частиною для всіх мікроконтролерів цього сімейства. Відмінність між окремими типами мікроконтролерів сімейства полягає в різному складі внутрішніх блоків інтерфейсу, ємності внутрішньої пам'яті програм, функціях окремих ліній введення/виведення. Ядро архітектури MCS-251 містить центральний процесор, блок синхронізації, блок обробки переривань, інтерфейс внутрішньої магістралі та інтерфейс локальної шини, який обслуговує внутрішні блоки введення/виведення.

Крім ядра, мікроконтролери містять внутрішні блоки введення/виведення, інтерфейс магістралі та внутрішню пам'ять. Внутрішні блоки введення/виведення призначені для виконання типових функцій інтерфейсу із зовнішніми пристроями. Вони містять сторожовий таймер, три таймери-лічильники, блок PCA та послідовний порт.

Лінії введення/виведення мікроконтролерів 8XC251SB об'єднані в чотири 8-розрядних порти загального призначення. Кожна лінія має

засувку. Апаратні драйвери виконують альтернативні функції, які залежать від режиму роботи і номера лінії. Ці мікроконтролери мають маскований ПЗП ємністю 16 Кбайт, на кристалі міститься репрограмований ЗП тієї самої ємності. Великий розмір цих матриць дає можливість у багатьох випадках відмовитися від використання зовнішньої пам'яті програм.

Мікроконтролери сімейства AVR

Сьогодні на світовому ринку електронних компонентів великим попитом користуються розробки корпорації Atmel (США), серед яких особливе місце займають пристрої енергонезалежної пам'яті високої швидкодії й мінімального питомого енергоспоживання, мікроконтролери загального призначення, а також мікросхеми програмованої логіки. Саме 8-розрядні високопродуктивні мікроконтролери, що об'єднуються під маркою AVR, є тим напрямком діяльності Atmel, що розвивається найбільш активно, починаючи з 1996 р., коли почалося їх серійне виробництво.

На сьогоднішній день Atmel масово випускає три основні підсімейства AVR – "tiny", "classic" та "mega", які орієнтовані на різні області застосування:

- інтелектуальні автомобільні сенсори, іграшки, ігрові приставки, материнські плати персональних комп'ютерів, контролери захисту доступу в мобільних телефонах, детектори диму та вогню, різна побутова техніка, пульти дистанційного керування (tiny);
- модеми, зарядні пристрої, вироби класу Smart Cards і засоби читання для них, супутникові навігаційні системи для визначення знаходження рухомих об'єктів, мережні карти, промислові системи контролю і керування (classic);
- цифрові мобільні телефони, принтери і їх ключові контролери, контролери факсимільних пристроїв, ксероксів, сучасних дискових накопичувачів, CD-ROM, CD-RW тощо (mega).

В загальному випадку мікроконтролер сімейства AVR представляє собою 8-розрядну однокристальну мікро-ЕОМ зі спрощеною системою команд RISC (Reduced Instructions Set Computers). Виготовляються вони на основі високоякісних CMOS-технологій, а енергонезалежні пристрої пам'яті, що входять до їх складу, виконуються на базі Flash і EEPROM-технологій. Характерна риса при цьому – низьке споживання енергії при високій тактовій частоті.

Базова структура мікроконтролерів AVR є єдиною для усіх типів. Узагальнена структурна схема показана на рис. 4.7.

Основними структурними блоками мікроконтролера є:

- генератор тактового сигналу (GCK);

- процесор (CPU);
- постійний запам'ятовуючий пристрій для зберігання програми, виготовлений за Flash-технологією (FlashROM);
- оперативний статичний запам'ятовуючий пристрій для зберігання даних (SRAM);

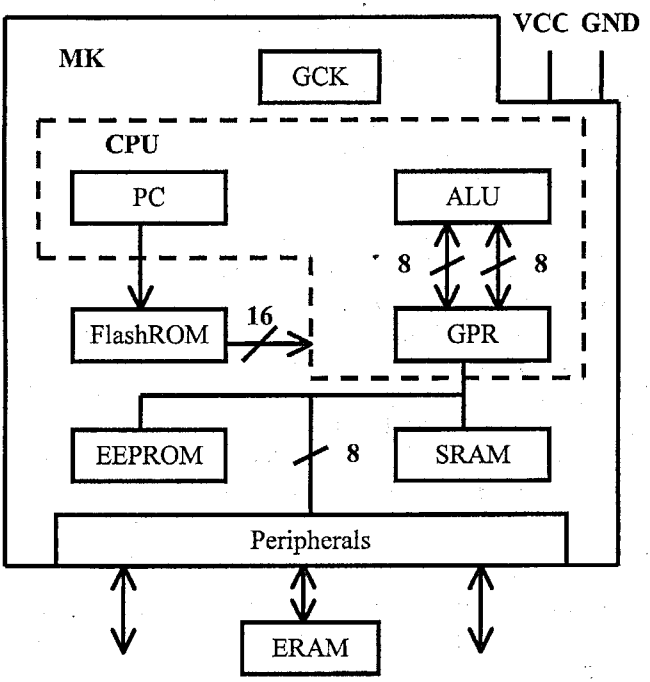


Рис. 4.7. Структурна схема типового AVR-мікроконтролера

- постійний запам'ятовуючий пристрій для зберігання даних, виготовлений на базі EEPROM-технології (EEPROM);
- набір периферійних пристроїв для введення/виведення даних, керуючих сигналів і виконання інших функцій.

До мікроконтролерів деяких типів також може підключатись зовнішній запам'ятовуючий пристрій для зберігання даних (ERAM). Виводи VCC і GND призначені для підключені джерела живлення.

До складу процесора (CPU) входять:

- лічильник команд (PC);
- арифметико-логічний пристрій (ALU);
- блок регістрів загального призначення (GPR – General Purpose Registers) та ряд інших елементів.

Особливе місце займають регістри спеціальних функцій, або регістри введення-виведення (I/O Registers, IOR), які призначені для:

- керування роботою мікроконтролера та його окремих пристроїв;
- визначення стану мікроконтролера та його окремих пристроїв;
- введення та виведення даних з мікроконтролера і його окремих пристроїв, а також ряд інших функцій.

Зупинимось коротко на призначенні та особливостях основних структурних елементів AVR-мікроконтролера.

Генератор тактового сигналу. AVR-мікроконтролери належать до класу пристроїв синхронного типу, оскільки всі дії, що в них виконуються, жорстко прив'язані до імпульсів тактового сигналу. Структура мікроконтролерів є повністю статичною і забезпечує можливість роботи при тактовій частоті від 0 Гц.

В якості генератора тактового сигналу (GCK) в різних типах мікроконтролерів можуть використовуватись:

- внутрішній генератор із зовнішнім кварцовим або керамічним резонатором (XTAL);
- внутрішній RC-генератор (IRC);
- внутрішній генератор із зовнішнім RC-ланцюжком (ERC);
- зовнішній генератор (EXT).

Процесор. До основних функцій процесора (CPU) належать формування адреси чергової команди, вибір команди з пам'яті і організація її виконання. Команди при цьому можуть представлятися у форматі "слово" (16 біт) або "два слова".

Окрім елементів, показаних на рис. 1, до складу процесора також входять регістр стану мікроконтролера (SREG), регістр-вказівник стека (SP або SPL і SPH для мікроконтролерів з великою ємністю SRAM) та ряд інших допоміжних елементів. Призначення та функції цих складових є типовими для більшості процесорів різних сімейств.

Запам'ятовуючий пристрій FlashROM. Цей постійний запам'ятовуючий пристрій призначений для зберігання кодів команд програми і констант. Одна його комірка пам'яті складається з 16 розрядів, а отже, може зберігати код команди у форматі "слово", половину коду команди формату "два слова" або коди двох констант. При цьому адреси команд надходять до FlashROM з лічильника команд, а адреси констант – зі спеціальної пари регістрів загального призначення. В процесі програмування запис кодів до FlashROM здійснюється побайтно. В залежності від кількості виводів мікроконтролера, байти можуть вводитися паралельно або послідовно (20 виводів і більше) або лише послідовно (8 виводів).

Запам'ятовуючий пристрій SRAM. Це оперативний запам'ятовуючий пристрій статичного типу, який призначений для зберігання даних, що отримуються в процесі роботи мікроконтролера. При

вимкненні напруги живлення мікроконтролера дані, що знаходилися в SRAM, втрачаються.

Запам'ятовуючий пристрій EEPROM. Це постійний запам'ятовуючий пристрій, який призначений для зберігання даних, що записуються при програмуванні мікроконтролера і отримуються в процесі виконання програми. При вимкненні напруги живлення мікроконтролера дані, що знаходилися в EEPROM, зберігаються.

Зовнішній запам'ятовуючий пристрій ERAM призначений для зберігання байтів даних.

Периферійні пристрої AVR-мікроконтролерів характеризуються певним різноманіттям. Коротко охарактеризуємо найбільш типові серед них.

Паралельний порт введення-виведення (Port, P) призначений для введення і виведення даних. В AVR-мікроконтролерах одночасно можуть використовуватись від одного до шести портів, кожен з яких може мати від трьох до восьми виводів.

Послідовний порт введення-виведення SPI (Serial Peripheral Interface) призначений для введення і виведення байтів в процесі обміну даними з іншими пристроями, оснащеними портом такого ж типу. Тактовий сигнал порту керує процесом обміну, а пристрій, що ініціалізує обмін і виробляє тактовий сигнал, є ведучим (master). Пристрій, що здійснює обмін при надходженні тактового сигналу, є веденим (slave). Процес видачі й прийняття байтів здійснюється обома пристроями одночасно, послідовно біт за бітом, для чого використовуються три шини обміну. До одного ведучого пристрою можуть підключатися декілька ведених, а функції ведучого й веденого в процесі роботи можуть змінюватись.

Послідовний порт введення-виведення UART (Universal Asynchronous Receiver-Transmitter) призначений для передавання і приймання байтів даних за допомогою двопровідних ліній зв'язку (наприклад, через інтерфейс RS-232 або "струмова петля"). Процес приймання і передавання може здійснюватись одночасно. Дані, призначені для передавання, формуються у вигляді кадру з десяти або одинадцяти бітів, до складу якого входять старт-біт ("0"), вісім бітів байту і стоп-біт ("1"). Між старшим бітом байту і стоп-бітом може поміщатись додатковий біт.

Послідовний порт введення-виведення TWSI (Two-Wire Serial Interface) призначений для обміну байтами даних з іншими пристроями по двопровідній лінії типу IC (Integrated Circuit), при цьому до шини обміну одночасно можуть бути підключені до 127 пристроїв.

Таймер-лічильник загального призначення (General Purpose Timer/Counter) може працювати в двох режимах. В режимі таймера формується запит переривання після завершення заданого часового інтервалу, в режимі лічильника здійснюється перелік заданої кількості подій. Основним його елементом є базовий лічильник, який працює в режимі підсумовувального лічильника. Зазвичай в AVR-мікроконтролерах

використовуються від одного до трьох таймерів-лічильників такого типу, тобто типу А. В залежності від розрядності лічильника і наявності додаткових функцій всього можуть використовуватись п'ять типів лічильників загального призначення (відповідно типу A, B, C, D і E).

Сторожовий таймер (WatchDog Timer, WDT) призначений для ліквідації збою при виконанні програми шляхом перезапуску мікроконтролера при виявленні збою.

Аналого-цифровий перетворювач (Analog-to-Digital Converter) формує десятирозрядний двійковий код числа, пропорційного величині напруги аналогового сигналу на вході мікроконтролера. В AVR-мікроконтролерах до АЦП можуть підключатися від чотирьох до восьми входів мікроконтролера.

Апалоговий компаратор (Analog Comparator, AC) порівнює аналогові сигнали, що надходять на два входи мікроконтролера, і формує спеціальний запит переривання, якщо різниця значень цих сигналів змінює знак на протилежний.

Програмовний апаратний модулятор (Programmable Hardware Modulator, PHM) призначений для формування імпульсного сигналу на спеціальному виводі для живлення світлодіодних індикаторів. При цьому тривалість імпульсу і скважність сигналу задаються програмно.

Блок переривань (Interrupt Unit, IU) забезпечує перехід до виконання програми переривання при надходженні відповідного запиту. При цьому встановлюється чи переривання за даним запитом дозволено, а також порівнюються пріоритети декількох переривань, що надходять одночасно. Запити до блоку переривань можуть надходити як від зовнішніх пристроїв, так і від джерел, розташованих у внутрішніх пристроях мікроконтролера.

4.4. Загальна характеристика інтерфейсів

Розрізняють два великі класи інтерфейсів – *зовнішні інтерфейси*, що з'єднують окремі пристрої, віддалені один від одного на значні відстані, а також *внутрішні інтерфейси*, призначені для швидкого зв'язку на короткі відстані. Другий клас даних пристроїв, судячи з назви, застосовується для внутрішнього зв'язку компонентів, що складають певну конфігурацію ЕОМ. В контексті аналізу технічних засобів систем керування більший інтерес представляють саме зовнішні інтерфейси про які далі і піде мова.

За способом передачі інформації інтерфейси підрозділяються на *паралельні* і *послідовні*. У *паралельному інтерфейсі* всі біти переданого слова (байта) виставляються і передаються по відповідних провідниках одночасно. В сучасних персональних ЕОМ традиційно використовується паралельний інтерфейс Centronics, реалізований LPT-портами. У

послідовному інтерфейсі біти передаються один за одним, зазвичай по одній лінії. COM-порти персональних ЕОМ забезпечують послідовний інтерфейс у відповідності зі стандартом RS-232C.

Для інтерфейсу, що з'єднує (фізично або логічно) два пристрої, розрізняють три можливих режими обміну - *дуплексний, півдуплексний* і *симплексний. Дуплексний режим* дозволяє по одному каналу зв'язку одночасно передавати інформацію в обох напрямках. Він може бути *асиметричним*, якщо пропускна здатність у протилежних напрямках має істотну різницю або *симетричним. Півдуплексний режим* дозволяє передавати інформацію в обох напрямках по черзі, при цьому інтерфейс має засоби переключення напрямку передавання. *Симплексний (однорічний) режим* передбачає тільки один напрямок передачі інформації (у зустрічному напрямку передаються тільки допоміжні сигнали інтерфейсу).

Іншим важливим параметром інтерфейсу є *допустиме віддалення пристроїв*, що з'єднуються. Воно обмежується як частотними властивостями кабелів, так і завадостійкістю інтерфейсів. Частина завад виникає від сусідніх ліній інтерфейсу – так звані перехресні перешкоди, захистом від яких може бути застосування кручених пар проводів для кожної лінії. Інша група завад спричиняється спотворенням рівнів сигналів.

З появою шин USB і FireWire в якості характеристики інтерфейсу почали використовувати *топологію з'єднання*. Для інтерфейсів RS-232C і Centronics практично завжди застосовується двоточкова топологія "комп'ютер – пристрій" (або "комп'ютер – комп'ютер"). Інтерфейсні шини USB і FireWire реалізують *деревоподібну топологію,* у якій зовнішні пристрої можуть бути як кінцевими, так і проміжними (роздільниками). Ця топологія дозволяє підключати безліч пристроїв до одного порту USB або FireWire.

Важливою властивістю інтерфейсу є *гальванічна розв'язка.* Схеми заземлення пристроїв, що з'єднуються інтерфейсом із COM- або LPT-портом комп'ютера, виявляються зв'язаними зі схемами заземлення комп'ютера (а через інтерфейсний кабель і між собою). Якщо між ними до підключення інтерфейсу була різниця потенціалів, то в інтерфейсах без розв'язки по спільному провіднику потече струм зрівноваження, що є небажаним з цілого ряду причин.

Паралельні інтерфейси

Паралельні інтерфейси характеризуються тим, що в них для передачі кожного біту інформації використовуються окремі сигнальні лінії, і всі біти передаються одночасно. Передача даних може бути як однонаправленою (Centronics), так і двонаправленою (Bitronics).

Інтерфейс Centronics і LPT-порт

Для підключення зовнішніх пристроїв за інтерфейсом Centronics комп'ютери обладнуються портами паралельного інтерфейсу, звідси –

назва LPT-порт (LinePrinTer – лінійний принтер). Хоча зараз через цей порт підключаються не тільки лінійні принтери, назва "LPT" залишилася.

Поняття Centronics характеризує набір сигналів, протокол взаємодії і 36-контактний роз'єм.

Протокол обміну Centronics програмно реалізується через традиційний порт SPP (Standard Parallel Port), який є однонаправленим портом.

Адаптер паралельного інтерфейсу є набором регістрів, розташованих у просторі введення/виведення. Порт має зовнішню 8-бітну шину даних, 5-бітну шину сигналів стану і 4-бітну шину керуючих сигналів.

BIOS підтримує до чотирьох (іноді до трьох) LPT-портів (LPT1-LPT4), забезпечуючи через них зв'язок із зовнішнім пристроєм по інтерфейсу Centronics. Спеціальний сервіс BIOS дозволяє здійснювати виведення символа, ініціалізацію інтерфейсу і пристрою, а також опитування його стану.

Недоліки стандартного порту частково усуваються шляхом використання нових типів портів, що вперше почали з'являтися в комп'ютерах PS/2.

Двонапрямлений порт 1 (Type 1 parallel port) – інтерфейс, введений у PS/2. Крім стандартного режиму, може працювати в режимі введення або двонаправленому режимі. Може використовуватися й у звичайних комп'ютерах.

Порт із прямим доступом до пам'яті (Type 3DMA parallel port) застосовувався в PS/2 моделей 57, 90, 95. Був введений для підвищення пропускної здатності і розвантаження процесора при виведенні інформації на друк. Програмі, що працює з портом, достатньо лише розмістити в пам'яті блок даних, що підлягають виведенню, а сам процес виведення здійснюється згідно з протоколом Centronics без участі процесора.

Пізніше з'явилися інші адаптери LPT-портів, що реалізують протокол обміну Centronics апаратно – Fast Centronics. Не будучи стандартизованими, такі порти різних виробників вимагали використання власних спеціальних драйверів.

Застосування паралельних інтерфейсів і LPT-портів

Паралельні інтерфейси застосовуються в системах, що керуються ЕОМ різних сімейств і класів. Обмежимося розглядом IBM PC-сумісних комп'ютерів.

Досить поширеним застосуванням LPT-порту є підключення принтера і плотера. Практично всі принтери можуть працювати з портом у режимі SPP, але застосування розширених режимів дає свої переваги:

- *двонаправлений режим* (Bi-Di) не підвищує продуктивність, але служить для повідомлення про стан і параметри принтера.
- *швидкісні режими* (Fast Centronics) підвищують продуктивність принтера, але можуть при цьому потребувати якісного кабелю. Від самого принтера при цьому не вимагається будь-яких додаткових "інтелектуальних" здібностей;

- *режим ECP* – потенційно найефективніший, має системну підтримку у всіх версіях Windows. На деяких принтерах реалізований не повністю (може бути відсутня апаратна компресія). Вимагає використання кабелю, який за частотними властивостями відповідає IEEE1284.

Послідовні інтерфейси

Послідовний інтерфейс для передачі даних використовує одну сигнальну лінію, по якій інформаційні біти передаються один за одним послідовно. Послідовна передача дозволяє скоротити кількість сигнальних ліній і збільшити відстань зв'язку. В ряді послідовних інтерфейсів застосовується гальванічна розв'язка зовнішніх (зазвичай вхідних) сигналів від схеми заземлення пристрою, що дозволяє з'єднувати пристрої, які знаходяться під різними потенціалами.

Інтерфейс RS-232C призначений для підключення апаратури, що передає або приймає дані (ППД – пристрій призначення або ПДД – пристрій-джерело даних; DTE – Data Terminal Equipment), а також до спеціальної апаратури передавання (АПД; DCE – Data Communication Equipment). У ролі ППД може виступати комп'ютер, принтер, плотер та інше обладнання. У ролі АПД зазвичай виступає модем. Кінцевою метою підключення є з'єднання двох пристроїв ППД.

Стандарт RS-232C передбачає використання 9-ти або 25-ти контактного з'єднувача. Призначення ліній 9-ти контактного з'єднувача наведено в таблиці 4.1.

Таблиця 4.1

Призначення ліній 9-ти контактного з'єднувача

Сигнал	Контакт	Призначення
PG	4	Заземлення на металевий корпус
TXD	2	Лінія передачі даних
RXD	8	Лінія прийому даних
RTS	3	Сигнал запиту на передачу даних від передавача даних
CTS	7	Сигнал підтвердження на обмін інформації від приймача даних
DSR	6	Сигнал підтвердження на обмін інформації від передавача даних
DTR	1	Сигнал запиту на приймання даних від приймача даних
SG	5	Загальна лінія для усіх провідників (сигналів)
---	9	---

Розповсюдженим варіантом послідовного інтерфейсу є струмова петля. У ній головним електричним сигналом є не рівень напруги відносно

загального проводу, а струм у двопровідній лінії, що з'єднує приймач і передавач.

Струмова петля зазвичай використовує гальванічну розв'язку вхідних ланцюгів приймача від схеми пристрою. При цьому джерелом струму в петлі є передавач (цей варіант називають активним передавачем). Можливо також живлення від приймача (активний приймач), при цьому вихідний ключ передавача може бути також гальванічно розв'язаний з іншою схемою передавача. Існують також спрощені варіанти без гальванічної розв'язки. Струмова петля з гальванічною розв'язкою дозволяє передавати сигнали на відстані до декількох кілометрів.

Інтерфейс RS-485 є стандартною модифікацією інтерфейсу RS-232C з струмовою петлею.

Основні технічні характеристики інтерфейсів RS-232 та RS-485 наведені в таблиці 4.2.

Таблиця 4.2

Основні технічні характеристики інтерфейсів RS-232 та RS-485

Технічні характеристики	Інтерфейси	
	RS-232	RS-485
Швидкість передачі, біт/с	20	10^4
Довжина магістралі	15	1200
Число ліній	1	2
Спосіб обміну інформацією	Послідовний	Послідовний
Режим обміну	Симплексний	Симплексний
Кількість ПП	1	1(10)

COM-порт

Послідовний інтерфейс COM-порт (Communication Port – комунікаційний порт) з'явився в перших моделях IBM PC. Він реалізований на мікросхемі асинхронного приймача-передавача Intel 8250, тому дотепер у всіх PC-сумісних комп'ютерах для послідовного інтерфейсу застосовують мікросхеми приймача-передавача, сумісні з i8250. У ряді вітчизняних PC-сумісних комп'ютерів для послідовного інтерфейсу застосовувалася мікросхема КР580ВВ51 – аналог i8251. Ця мікросхема є універсальним синхронно-асинхронним приймачем-передавачем (UART – Universal Asynchronous Receiver-Transmitter).

За своїми параметрами COM-порт відповідає стандарту RS-232C. Інтерфейс RS-232C широко розповсюджений у різних пристроях керування і терміналах. При цьому COM-порт може використовуватися як двонаправлений інтерфейс, що має 3 програмнокеровані вихідні лінії і 4 вхідні лінії, по яких передаються двополярні сигнали. Їх використання визначається розробником.

Під'єднання до комунікаційного порту здійснюється як мінімум за допомогою трьох ліній, як показано на рис. 4.8.

Такий спосіб з'єднання за допомогою 3-х проводів називається *нуль-модем*.

Для реалізації інтерфейсу в периферійних пристроях використовується мікросхема – універсальний асинхронний приймач-передавач.

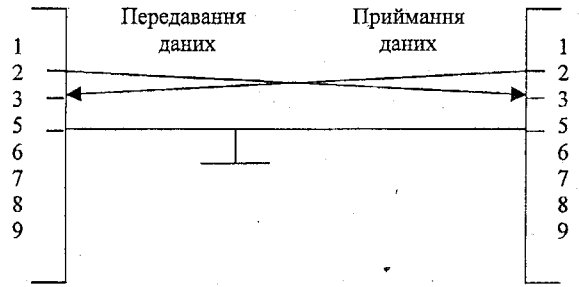


Рис . 4.8. З'єднання COM-портів за схемою нуль-модема

Найчастіше КСК-утворення системи здійснюється таким способом.

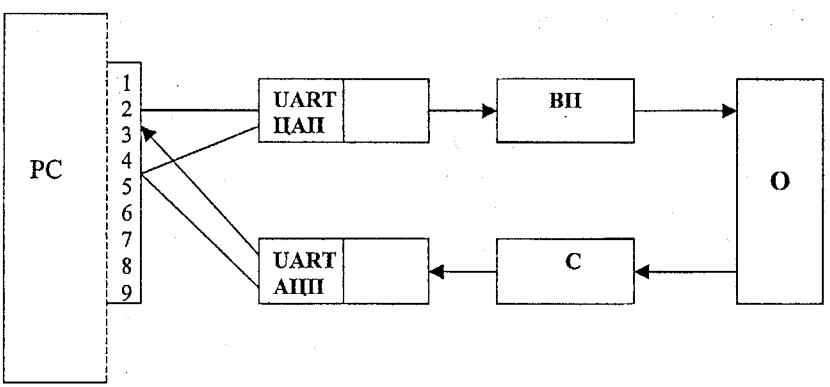


Рис. 4.9. Схема приєднання САК до комп'ютера через COM-порт

Програмне керування комунікаційним портом здійснюється за допомогою спеціальних регістрів, які є у складі комунікаційного порту.

COM-порт використовують для безпровідникових комунікацій із застосуванням випромінювачів і приймачів інфрачервоного діапазону – IR (Infra Red) Connection. Цей інтерфейс дозволяє здійснювати зв'язок між парою пристроїв, віддалених на відстань, що сягає декількох метрів. Розрізняють інфрачервоні системи низької (до 115,2 Кбіт/с), середньої (1,152 Мбіт/с) і високої (4 Мбіт/с) швидкості. Системи з низькою швидкістю служать для обміну короткими повідомленнями, високошвидкісні – для обміну файлами між комп'ютерами, підключення до комп'ютерної мережі, виведення на принтер, проекційний апарат і т. п.

Досягнення більш високих швидкостей обміну дасть можливіть передавати "живе відео". У 1993 році створена асоціація розробників систем інфрачервоної передачі даних IrDA (Infrared Data Association), покликана забезпечити сумісність обладнання різних виробників. В наш час діє стандарт IrDA 1.1. Існують власні системи фірм Hewlett Packard – HP-SI (Hewlett Packard Slow InfraRed) і Sharp – ASK (Amplitude Shifted Keyed IR).

Основні характеристики інтерфейсів такі:

- IrDA SI (Slow Infra Red), HP-SI - 9,6-115,2 Кбіт/с;
- IrDA MI (Middle Infra Red) - 1,2 Мбіт/с;
- IrDA FI (Fast Infra Red) – 4 Мбіт/с;
- Sharp ASK - 9,6-57,6 Кбіт/с.

Інфрачервоні випромінювачі не створюють завад у радіочастотному діапазоні і забезпечують конфіденційність передачі. Ir-промені не проходять через стіни, тому зона прийому обмежується невеликим, легко контрольованим простором. Інфрачервоний інтерфейс мають також деякі моделі принтерів.

Застосування послідовних портів

Регістри комунікаційних портів займають діапазон адрес:

- 2F8 до 2FE : COM2;
- 3F8 до 3FE : COM1.

Головним регістром приймання-передавання є регістр 3F8 (2F8). Решта регістрів – це керуючі регістри.

Процес керування, приймання та передавання даних полягає у запису відповідних байтів у регістри комунікаційних портів.

Програмне забезпечення обміну даними через com ґрунтується на 3-х головних процедурах:

- ініціалізації (настроювання порта);
- прийманні байта;
- передаванні байта.

Обмін даними і відповідно побудова цих процедур може здійснюватись 2 способами:

- способом опитування;
- способом передавання.

При використанні способу опитування, процесор періодично контролює стан порту і, якщо після попереднього контролю надійшли дані, то приймає їх.

Недоліки цього способу:

- якщо у проміжку між двома контролями надійшло декілька даних, то прийнято буде лише останнє, а попередні будуть втрачені;
- якщо не надійшло жодного з даних, то процесор виконує зайву роботу, контролюючи стан портів.

Перевага: простота програмної реалізації.

Рекомендується використовувати цей спосіб при рівномірній та заздалегідь відомій періодичності надходження даних.

При способі переривання – при надходженні даних у порт, порт генерує сигнал переривання, який надходить до процесора, процесор переривае свою роботу і запускає процедуру обробки переривання, яка полягає у прийманні даних і розміщенні їх у спеціальному буфері для наступної обробки іншою програмою.

Перевага способу переривання: висока ефективність, нічого не втрачається, процесор не виконує зайвої роботи.

Недолік: складність програмної реалізації.

Алгоритми приймання/передавання даних через порти грунтуються на чотирьох процедурах: встановлення необхідних параметрів порту, передача байта, приймання байта, перевірка стану порту. Ці процедури зазвичай однакові, але алгоритм їх виконання залежить від режиму приймання даних – за перериванням або за опитуванням.

Програми приймання/передавання символів у режимі опитування (табл. 4.3): на першому компютері програма зчитує символи з клавіатури та передає їх через COM1 зі швидкістю 9600 біт/с по 8 біт, без контролю парності; на другому компютері програма приймає дані через COM2 зі швидкістю 9600 біт/с байтами по 8 біт, без контролю парності, і виводить їх на екран.

Програма приймання даних через COM2 у режимі переривання зі швидкістю 9600 біт/с байтами по 8 біт, без контролю парності, і виведення їх на екран (передавання виконується аналогічно) наведена у додатку.

Використання портів під керуванням Windows

При створені програмного забезпечення, яке використовує обмін даними через порти під керуванням операційної системи Windows, виникають додаткові труднощі. Вони викликані тим, що Windows працює у захищеному режимі, який не дозволяє безпосередньо звертатися до регістрів COM (і будь-якого іншого обладнання). Тому для операцій із зовнішніми пристроями використовуються API-функції бібліотеки Windows.

Для того, щоб викликати API-функції потрібно підключити файл < Windows .h>.

Відкриття порту

Для відкриття COM порту використовуємо функцію CreateFile із такими параметрами.

```
HANDLE hPort = CreateFile(_T("COM1"),
                          GENERIC_READ | GENERIC_WRITE,
                          0,
                          NULL,
                          OPEN_EXISTING,
                          0,
                          NULL);
```

Таблиця 4.3

Програми приймання/передавання символів у режимі опитування

ПЕРЕДАВАЧ	Напрям потоку даних	ПРИЙМАЧ
uses crt; *var c : char;* *Procedure InitCom1;* *begin* *port[$3FB]:=$80;* *port[$3F8]:=12;* *port[$3F9]:=0;* *port[$3FB]:=3;* *end;* *Procedure* *SendByte(b:byte);* *begin* *port[$3F8]:=b;* *end;* *BEGIN* *InitCom1;* *repeat* *c:=readkey;* *SendByte(ord(c))* *until c=#27* *END.*	→	*var b : byte;* *Procedure InitCom2;* *begin* *port[$2FB]:=$80;* *port[$2F8]:=12;* *port[$2F9]:=0;* *port[$2FB]:=3;* *end;* *Function ChekCom2:boolean;* *Begin* *ChekCom2:=(port[$2FD] and 1) = 1* *End;* *Procedure ReadCom2(var b:byte);* *begin* *repeat until ChekCom2;* *b:=port[$2F8]* *end;* *BEGIN* *InitCom2;* *repeat* *ReadCom2(b);* *write(chr(b))* *until b=27* *END.*

Перший параметр цієї функції є ім'я порту. Другий параметр це рівень доступу до об'єкта, для запису і для читання. В нашому випадку ми відкриваємо доступ для читання і для запису відповідно прапорці GENERIC_READ і GENERIC_WRITE.

Третій параметр це тип сумісного використання. В нашому випадку він не використовується, тому виставляємо його в 0 (можуть бути параметри: **FILE_SHARE_DELETE** – якщо цей прапорець виставлений, інший процес не має права видалити файл; **FILE_SHARE_READ** – інший процес не має права на читання; **FILE_SHARE_WRITE** – інший процес не має права на запис).

Четвертий параметр не використовується, виставляємо його в NULL.

П'ятий параметр визначає як відкрити файл – виставлямо прапо-

рець OPEN_EXISTING (можуть бути параметри: **CREATE_ALWAYS** – створює новий файл, якщо файл є, то він буде перезаписаний; **CREATE_NEW** – створює новий файл, функція поверне помилку якщо файл створений; **OPEN_ALWAYS** – якщо файл створений, відкриє його, якщо файлу немає, то створить його; **TRUNCATE_EXISTING** – відкриває файл і стирає всі дані файлу, якщо файлу немає то функція поверне помилку).

Шостий параметр – це атрибути файлу (системний, нормальний, прихований (hide), архівний (arhived)). Для портів цей параметр не використовується, виставляємо його в 0.

Останній параметр – це тимчасовий файл. Його також не будемо використовувати, тому виставляємо його в NULL.

Ця функція поверне HANDLE порту: якщо виникне помилка, то функція поверне значення INVALID_HANDLE_VALUE. HANDLE – тип даних Windows.

Конфігурація COM порту

Перед тим як читати або записувати дані в порт, потрібно його настроїти, а саме, виставити максимальний час читання і запису одного байта в порт і з порту (за допомогою структури COMMTIMEOUTS). Також потрібно виставити для COM порту такі параметри, як кількість стопових бітів, кількість бітів в байтові, перевірку на парність і т.п., які визначає структура DCB.

Для початку визначимо структуру COMMTIMEOUTS за допомогою функції GetCommTimeouts

```
COMMTIMEOUTS comtime;              // означення змінної
GetCommTimeouts(hPort, & comtime); // виклик функції
```

Потім модифікуємо її таким чином

```
comtime.ReadIntervalTimeout = 10; /* час читання одного байта в мс*/
comtime.ReadTotalTimeoutConstant  = 10;
comtime.ReadTotalTimeoutMultiplier = 10;
comtime.WriteTotalTimeoutConstant  = 10;
comtime.WriteTotalTimeoutMultiplier = 10;
```

і виставляємо часові параметри для COM порту

```
SetCommTimeouts(hPort, &comtime);
```

Тепер потрібно настроїти інші параметри порту, а саме: стопові біти перевірку на парність, швидкість передачі даних, кількість бітів в байтові. Для цього використовуємо структуру DCB.

Таблиця 4.4

Параметри структури COMMTIMEOUTS

Параметр	Опис параметра
ReadIntervalTimeout	Максимальний час дозволений для затримки між прибуттям двох байтів на комунікаційній лінії, в мілісекундах. Значення 0 означає що цей параметр не використовується
ReadTotalTimeoutMultiplier	Використовується для обчислення часу для операції читання, в мілісекундах. Для кожної операції читання це значення множиться на кількість байтів, які потрібно прочитати
ReadTotalTimeoutConstant	Константа використовується для обчислення повного часу читання, в мілісекундах. Для кожної операції читання це значення додається до добутку ReadTotalTimeoutMultiplier на кількість байтів, які потрібно прочитати
WriteTotalTimeoutMultiplier	Використовується для обчислення часу запису, в мілісекундах. Для кожної операції запису це значення множиться на кількість байтів, які потрібно записати
WriteTotalTimeoutConstant	Константа використовується для обчислення загального часу запису, в мілісекундах. Для кожної операції запису це значення додається до добутку WriteTotalTimeoutMultiplier на кількість байтів, які мають бути записані

```
DCB PortDCB;                        // означення змінної
GetCommState(m_hPort, &PortDCB);    // виклик функції
```

Модифікуємо її таким чином:

```
PortDCB.BaudRate = CBR_19200;  // Швидкість передачі даних 19200 бод
PortDCB.fBinary = TRUE;        // Двійковий режим
PortDCB.fParity = FALSE;       // Не дозволяти перевірку на парність
PortDCB.fOutxCtsFlow = FALSE;  // Немає сигналу CTS в потоці данних
PortDCB.fOutxDsrFlow = FALSE;  // Немає DSR в вихідному потоці
                               // даних
```

```
PortDCB.fDtrControl = 0;            // DTR_CONTROL_ENABLE;
PortDCB.fDsrSensitivity = FALSE;    // Виключити чутливість до DSR
PortDCB.fTXContinueOnXoff = TRUE;   // Продовжити передавати дані,
                                    // якщо вхідний буфер заповнений
PortDCB.fOutX = FALSE;  // Немає XON/XOFF сигналів в вихідному
                        // потоці даних.
PortDCB.fInX = FALSE;   // Немає XON/XOFF сигналів в потоці
                        // вхідних даних
PortDCB.fErrorChar = FALSE; // Заборонити заміну байтів з помилками
PortDCB.fNull = FALSE;          // Не блокувати 0 символи
PortDCB.fRtsControl = 0;        //RTS_CONTROL_ENABLE;
PortDCB.fAbortOnError = FALSE;  // Не припиняти передачі даних або
                                // читання коли виникла помилка
PortDCB.ByteSize = 8;           // Кількість бітів в байтові
PortDCB.Parity = NOPARITY;
PortDCB.StopBits = ONESTOPBIT; // Кількість стопових бітів 1
```

Виставляємо параметри до COM порту

```
SetCommState(hPort, &PortDCB); // виклик функції
```

Запис і читання порту

Для запису і читання порту використовується функції відповідно WriteFile і ReadFile. Розглянемо запис 100 байт в COM порт.

```
1. char buffer[100];
2. setmem(buffer, 1, 100);
3. DWORD dwWritten = 0, dwTotalWritten = 0;
4. BOOL bWrite = TRUE;
5. while (bWrite)
6. {
7.      bWrite=WriteFile(hPort,buffer+dwTotalWritten,100-
dwTotalWritten,&dwWritten, NULL);
8.      dwTotalWritten+= dwWritten;
9.      if (dwTotalWritten == 100) break;
9. }
```

В 1-ому рядку виділяємо буфер на 100 байт.

В 2-ому рядку заповнюємо буфер даними (для прикладу – одиницями).

В 3-ому рядку оголошуємо змінні **dwWritten** – скільки байтів записано за один виклик функції **WriteFile. dwTotalWritten** – скільки всього байтів записано.

В 4-ому рядку оголошуємо змінну, яка прийме значення FALSE, якщо в функція **WriteFile** виникне помилка.

В 7-ому рядку викликаємо функцію **WriteFile** яка записує дані до порту. Перший параметр функції – це HANDLE відкритого раніше порту.

Другий параметр – це вказівник на буфер. Третій параметр передає кількість байтів, які потрібно записати до порту; четвертий параметр повертає кількість записаних байтів. Якщо функція не змогла записати всі дані, в цей параметр вона запише кількість записаних байтів. Якщо ця функція не записала всі дані, знову викликаємо її, але вже з урахуванням тих даних, які записані. Так повторюємо, поки всі дані не будуть записані. П'ятий параметр не використовуємо, записуємо в нього NULL.

В 8-ому рядку підсумовуємо кількість записаних байтів.

В 9-ому рядку перевіряємо, чи записали ми всі дані до порту; якщо записали, то виходимо з циклу.

Розглянемо читання даних з COM порту.

```
10. DWORD dwRead = 0, dwTotalRead = 0;
11. BOOL bRead = TRUE;
12. while (bRead)
13. {
14.     bRead=ReadFile(hPort, buffer+ dwTotalRead, 100- dwTotalRead,
&dwRead, NULL);
15.     dwTotalRead+= dwRead;
16.     if (dwTotalRead == 100) break;
17.}
```

В 10-му рядку оголошуємо змінні: **dwRead** – зберігає кількість прочитаних байтів за один виклик **ReadFile. dwTotalRead** – зберігає кількість прочитаних байтів за всі виклики функції.

В 11-ому рядку оголошуємо змінну **bRead,** яка прийме значення FALSE, якщо в функції **ReadFile** виникне помилка.

В 14-ому рядку викликаємо функцію **ReadFile** з такими параметрами:

1-й параметр - HANDLE відкритого раніше COM порту;

2-й параметр – буфер, в який будуть записуватися прочитані з порту байти;

3-й параметр – кількість байтів, які потрібно прочитати;

4-й параметр – кількість байтів, які функція прочитала; якщо функція прочитала не всі байти, то її потрібно ще раз викликати з урахуванням прочитаних байтів;

5-й параметр не використовується, передаємо в нього NULL

В 15-ому рядку підсумовуємо кількість прочитаних байтів.

В 16-ому рядку перевіряємо кількість прочитаних байтів; якщо прочитані всі байти, то виходимо з циклу.

Завершення роботи з портом

Для завершення роботи з портом потрібно викликати функцію CloseHandle:

CloseHandle(hPort),

де **hPort** – це HANDLE порту, відкритого за допомогою функції CreateFile.

4.5. Виконавчі пристрої

Виконавчим пристроєм (ВП) системи керування називається пристрій, призначений для зміни стану об'єкта керування відповідно до заданого закону керування.

Зважаючи на специфіку роботи цих пристроїв, до них висуваються такі вимоги:

- забезпечення необхідних вихідних зусиль у всіх режимах роботи;
- здатність витримувати короткочасні перевантаження, що виникають в системі;
- забезпечення високих швидкостей і прискорень по переміщенню об'єкта керування;
- плавне регулювання в широких межах;
- невелика інерційність.

Окрім перерахованих специфічних вимог, до ВП висувають також загальнотехнічні вимоги: невеликі маса і габарити, висока надійність при механічних і кліматичних впливах.

Виконавчі пристрої використовують різні фізичні принципи дії в залежності від типу процесу, яким вони керують:

- для керування механічними процесами (рух, тиск, орієнтація в просторі) використовуються переважно електромеханічні, електрогідрав-лічні та електропневматичні ВП;
- для керування тепловими процесами використовуються різні нагрівачі (термоелектронагрівачі – ТЕН, регульовані газові пальники тощо);
- для керування електричними процесами використовуються електричні прилади з регульованими характеристиками (транзистори, варикапи, негатрони тощо);
- для керування світловими процесами використовуються нелінійні оптичні елементи – модулятори.

Класифікація електромеханічних ВП представлена на рис. 4.10.

Як електромеханчні ВП зазвичай використовуються регульовані електродвигуни постійного і змінного струму, а також нерегульовані електродвигуни в поєднанні з керованими електромагнітними муфтами. Кожний з перерахованих типів пристроїв має певні переваги і недоліки, що необхідно враховувати при виборі області їх застосування.

Електричні двигуни постійного струму

В системах керування застосовують в основному двигуни постійного струму з незалежним збудженням та із збудженням від постійного магніту.

Керування здійснюється подачею регульованої напруги на обмотку якоря або на обмотку збудження. Але оскільки при керуванні по колу збудження не можна забезпечити хороших пускових якостей і достатньої

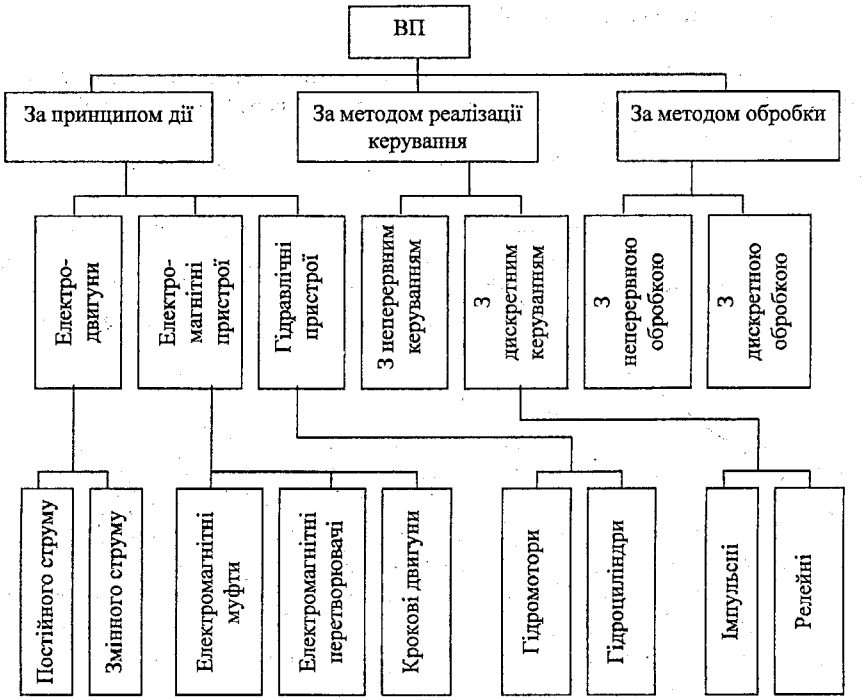


Рис. 4.10. Класифікація виконавчих пристроїв

жорсткості механічної характеристики в широкому діапазоні регулювання, то це керування застосовують рідко. При керуванні по колу якоря сигнал керування (I_y) надходить на якірну обмотку від підсилювача потужності (рис. 4.11, а).

За допомогою даної схеми здійснюється безперервне регулювання швидкості двигуна М.

Схема мостового вихідного каскаду на транзисторах (рис. 4.11,*б)* з включеним в діагональ двигуном дозволяє здійснювати як безперервне, так і дискретне керування, залежно від режиму роботи транзисторів. Дана схема застосовується для керування двигунами потужністю не більше 100 Вт, що обумовлено великими тепловими втратами, пов'язаними з безперервним режимом роботи транзисторів.

Імпульсний метод регулювання дозволяє полегшити режим роботи транзисторів і підвищити потужність вихідного каскаду до декількох кіловат. В цьому випадку дана схема працює в режимі перемикання, посилаючи в двигун серію імпульсів. Надходження імпульсу розганяє двигун, а його відсутність – гальмує. При цьому середня швидкість вала двигуна визначається співвідношенням довжин імпульсу і паузи. Для

усунення коливань швидкості (отримання безперервного обертання вала) частоту перемикань роблять якомога більшою. Імпульсне регулювання забезпечує нормальний тепловий режим роботи двигуна, оскільки він нагрівається тільки протягом часу дії імпульсу.

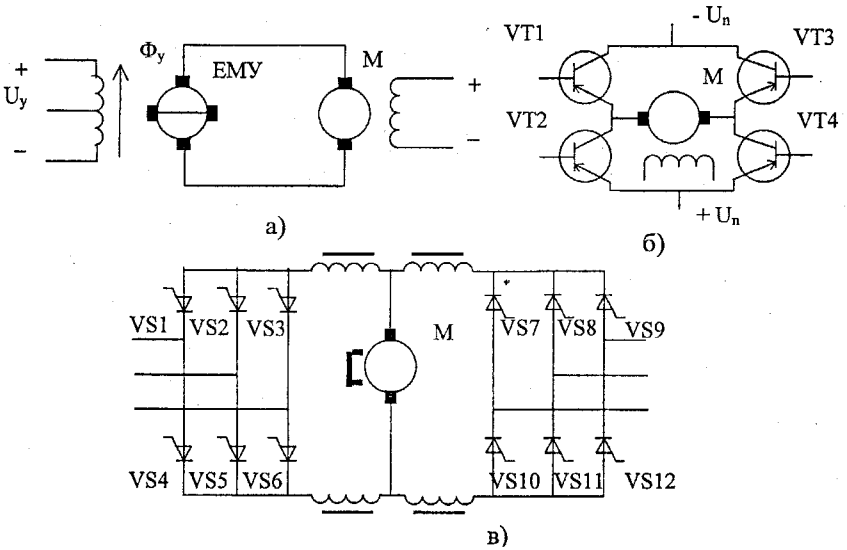


Рис. 4.11. Схеми керування ВД постійного струму

Мостова схема на тиристорах (рис. 4.11, в) дозволяє здійснювати дискретне регулювання швидкості двигуна при комутації струмів до декількох сотень ампер і напругах до тисячі вольт. Регулювання частоти обертання двигуна відбувається за рахунок регулювання часу включення тиристорів, тобто часу підключення двигуна до джерела живлення. Схема застосовується для керування ВД потужністю до 100 кВт.

Електричні двигуни змінного струму

З числа двигунів змінного струму для роботи в системах керування найбільш придатними є малоінерційні асинхронні двигуни з хорошими регулювальними характеристиками. Такі переваги, як відсутність колектора і щіток, малий момент інерції, простота керування і узгодження з підсилювачами змінного струму, роблять асинхронні двигуни незамінними в системах високої швидкодії, а також в системах, що працюють у вибухонебезпечних умовах. Невеликі розміри цих двигунів дають можливість використовувати їх в пристроях, в яких ставляться жорсткі вимоги до розмірів і маси. Крім того, ці двигуни простіші й економічніші.

Основний недолік асинхронних ВД, яким обмежено їх

розповсюдження – низький ККД і невелика потужність (до 100-150 Вт). Збільшення потужності спричиняє значне зростання розмірів і вимагає інтенсивного охолоджування.

Розрізняють методи неперервного і дискретного керування частотою обертання асинхронного двигуна. Неперервне керування частотою досягається шляхом зміни амплітуди керуючої напруги, дискретне – шляхом зміни часу підключення обмоток двигуна до джерела живлення.

Неперервний метод керування може бути здійснений за допомогою схем з використанням електронних підсилювачів змінного струму і магнітних підсилювачів. На рис. 4.12, *а* зображена схема регулювання асинхронного двигуна за допомогою транзисторного підсилювача, виконаного на транзисторах VT1 і VT2.

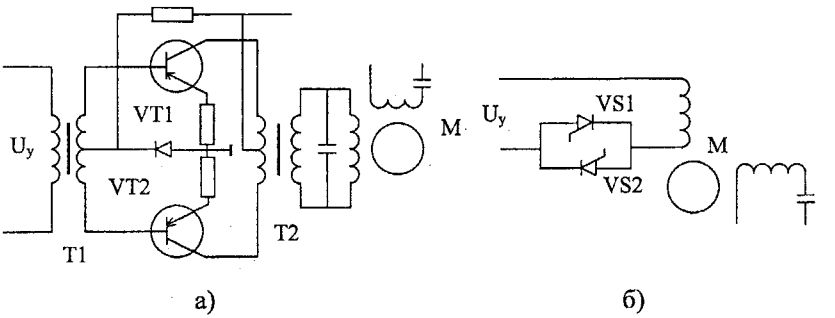


Рис. 4.12. Схеми керування асинхронним двигуном

Дискретний метод керування асинхронним двигуном можна пояснити за допомогою схеми керування на тиристорах (рис. 4.12, б).

Електромагнітні муфти

На відміну від електродвигуна, електромагнітна муфта не створює обертового моменту, а виступає як передавальна ланка між нерегульованим приводним двигуном і навантаженням за рахунок механічного тертя або зчеплення.

За типом основного робочого (виконавчого) органу муфти поділяють на фрикційні (ЕФМ) і порошкові (ЕПМ). У фрикційних муфтах основними виконавчими органами є диски, які при подачі керуючого сигналу притягуються і за рахунок сили тертя, що виникає між ними, створюють механічний зв'язок.

В порошкових муфтах основним виконавчим органом є феромагнітний порошок, що заповнює порожнину між ведучою і веденою частинами муфти. За відсутності сигналу керування магнітний порошок

знаходиться у вільному стані, зчеплення відсутнє, і обертання не передається. При подачі в обмотку електромагніту сигналу керування порошок намагнічується і твердне, створюючи між ведучою і веденою частинами момент зчеплення. Ведена частина починає обертатися, причому переданий момент залежить від струму керування.

До переваг ЕПМ слід віднести:

- високе значення відношення обертового моменту до моменту інерції веденої частини муфти, що забезпечує великі прискорення – до 100000 рад/с^2 (у двигунів постійного струму такої ж потужності – до 20000 рад/с^2);
- незначну потужність керування за рахунок високої магнітної провідності виконавчого органу і, як наслідок, велике підсилення за потужністю;
- лінійну залежність передаваного моменту від сигналу керування;
- висока швидкодія;
- можливість використання простих, дешевих і надійних в роботі нерегульованих двигунів.

Недоліками муфт є:

- складність конструкції механічних вузлів муфт;
- значний нагрів виконавчого органу муфт при ковзанні і необхідність в спеціальних заходах для охолодження;
- вплив зовнішніх чинників на характеристики муфт внаслідок непостійності магнітних властивостей порошку;
- невисокий ККД, обумовлений непродуктивною витратою енергії на обертання приводного двигуна за відсутності сигналу керування (ККД = 0,5).

Гідравлічні виконавчі пристрої

Як гідравлічні ВП в системах керування зазвичай застосовують гідроприводи (ГП) з об'ємним або дросельним регулюванням. Об'ємне регулювання ГП засновано на зміні робочого об'єму гідромашини (насоса або гідродвигуна) при повороті її регулюючого органу відносно початкового положення, при якому робочий об'єм дорівнює нулю. При дросельному регулюванні використовується закономірність зміни витрат робочої рідини при її протіканні через регульовану щілину дросельного розподільного пристрою.

Гідропривод з об'ємним регулюванням

ВП такого типу зазвичай складається з регульованого насоса *Н* і нерегульованого гідродвигуна *М* (рис. 4.13, а), двох регульованих гідромашин (рис. 4.13, б) або одного регульованого гідродвигуна. В ГП з об'ємним регулюванням застосовують різні аксіально-поршневі гідромашини з похилим блоком або шайбою. Насос *Н* може працювати від будь-якого

механічного або електричного двигуна. Залежно від кута повороту, регулюючий орган насоса змінює подачу в одну або іншу гідролінію, що з'єднує насос з гідродвигуном М, за рахунок чого змінюється кутова швидкість гідродвигуна. Поворот регулюючого органу гідромашини на певний кут здійснюється за допомогою спеціальних механізмів керування (МК) залежно від керуючого сигналу, що надходить на вхід МК. Для забезпечення поворотно-поступального руху об'єкта регулювання замість гідродвигуна застосовують гідроциліндр. Регульований гідромотор може працювати від гідравлічного джерела живлення з постійним тиском в напірній гідролінії.

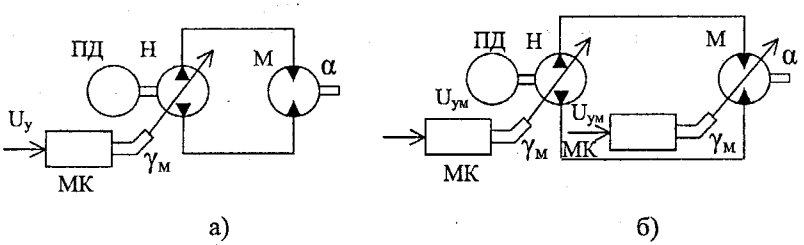


Рис. 4.13. Типові схеми гідроприводу із об'ємним регулюванням

В порівнянні з електромашинним приводом, ГП з об'ємним регулюванням потужністю від 0,3 до 100 кВт має кращі енергомасові характеристики, і має велику потужність при менших масових і габаритних параметрах машин. Мала інерційність рухомих частин гідродвигунів, дозволяє одержати більш високу швидкодію привода і кращі показники якості перехідних процесів.

ГП дозволяє регулювати кутову швидкість гідродвигуна при зміні моменту навантаження в широкому діапазоні. При цьому ГП має велику жорсткість, тобто зміна кутової швидкості гідродвигуна при дії навантаження відбувається повільно.

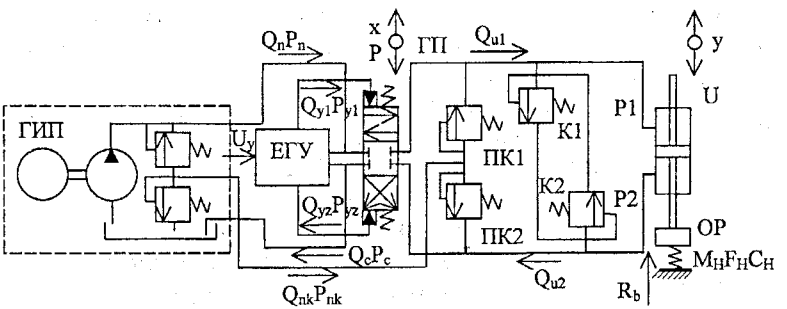
Гідропривод з дросельним регулюванням

ГП цього типу (рис. 4.14, *а)* складається з дросельного розподільника *Р,* гідроциліндра *Ц,* сполученого з об'єктом регулювання, запобіжних К1, *К2* і підживлювальних *ПК1, ПК2* клапанів. Розподільник *Р* може бути непроточного або напівпроточного типу.

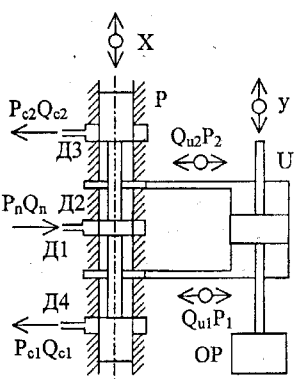
ГП з дросельним регулюванням в порівнянні з ГП з об'ємним регулюванням має ряд переваг: більш компактний, має меншу мАСК і розміри, кращі динамічні характеристики за рахунок меншої інерційності рухомих частин розподільника і гідроциліндра; зручний при розміщенні на виробах, оскільки дозволяє встановлювати електрогідравлічні підсилювачі безпосередньо на гідродвигуні.

Недоліками ГП є невеликий ККД (< 0,3), велике тепловиділення і мала жорсткість механічної характеристики.

Зазвичай ГП з дросельним регулюванням використовують для забезпечення високої швидкодії і точності систем керування відносно малої потужності (від 0,5 до 10 кВт) або при короткочасній її роботі.



а)



б)

Рис. 4.14. ГП з дросельним регулюванням
а – функціональна схема;
б – розподільник

Пневматичні виконавчі пристрої

Використовуються дуже широко, особливо для пневматичного інструменту.

Переваги:

- висока надійність;
- хороші динамічні характеристики (швидкість, прискорення, які може отримати інструмент);

- нежорсткість характеристики.
- простота забезпечення кожного робочого місця стиснутим повітрям.

Недоліки:

- складність отримання високого тиску та великого об'єму повітря;
- небезпечність великих об'ємів стиснутого повітря.

Цікавою особливістю пневматичних ВП є наявність великої системи допоміжних пневматичних малогабаритних пристроїв, які забезпечують перемикання, логічне керування та інші операції над потоками повітря. Ця система пневматичних елементів отримала назву "пневмоніка". Елементи пневмоніки використовуються у вибухонебезпечних умовах, де застосування електричних ВП обмежене.

Контрольні питання та завдання

1. Які основні складові входять до типової структури комплексу технічних засобів комп'ютеризованої системи керування?
2. Які типи сенсорів використовуються в КСК? В чому їх особливості?
3. В чому полягає принцип роботи цифроаналогового перетворювача? Чим характерні найпоширеніші схеми ЦАП?
4. В чому полягає призначення та принцип роботи аналого-цифрового перетворювача? Назвіть особливості конструктивних різновидів схем АЦП послідовної дії.
5. Охарактеризуйте властивості схем АЦП паралельної дії та порозрядного зрівноважування.
6. Що представляє собою типовий сучасний мікроконтролер? В яких основних галузях використовуються мікроконтролери?
7. Дайте характеристику основних параметрів мікроконтролерів сімейства MCS-96.
8. В чому особливості мікроконтролерів сімейства MCS-51/151/251?
9. Дайте визначення інтерфейсу. В чому особливості зовнішніх та внутрішніх інтерфейсів, послідовних та паралельних?
10. Які основні параметри та режими роботи характерні для сучасних інтерфейсів?
11. В чому полягають особливості інтерфейсів Centronics і LPT-порту? Як використовуються паралельні інтерфейси?
12. Охарактеризуйте основні різновиди послідовних інтерфейсів.
13. Які основні вимоги висуваються до виконавчих механізмів? На які основні класи поділяються виконавчі механізми?
14. В чому полягає принцип дії електродвигунів постійного та змінного струму?
15. В чому полягає призначення електромагнітних муфт?
16. Назвіть основні типи гідравлічних виконавчих механізмів, особливості їх функціонування, переваги та недоліки

Література

1. Локазюк В.М. Мікропроцесори та мікро-ЕОМ у виробничих системах: Посібник. – К.: Академія, 2002. – 368 с.
2. Автоматика і автоматизація технологічних процесів: Підручник / Д.Б. Головко, К.Г. Рего, Ю.О. Скрипник. – К.: Либідь, 1997. – 232 с.
3. Элементы автоматических устройств: Учебник для вузов / В.Л. Фабрикант, В.П. Глухов, Л.Б. Паперно, В.Я. Путниньш. – М.: Высшая школа, 1981. – 400 с.
4. Гук. М. Интерфейсы ПК: справочник. – СПб.: Питер Ком, 1999. – 416 с.
5. В.М.Дубовой, Р.Н.Квєтний. Програмування персональних комп'ютерів систем управління. – Вінниця: ВДТУ, 1999. – 110 с.
6. Основы проектирования и расчета следящих систем: Учебник для техникумов / В.И. Смирнова, Ю.А. Петров, В.И. Разинцев. – М.: Машиностроение, 1983. - 295 с

5. РОЗПОДІЛЕНІ КСК

Переважна більшість сучасних КСК є розподіленими системами, в основі яких лежать комп'ютерні мережі. Причиною цього є відносно низька вартість контролерів, комп'ютерів та іншого мережного обладнання, що дозволяє використовувати їх у комплексі з будь-яким сенсором, виконавчим пристроєм тощо. Правила побудови комп'ютерних мереж визначаються так званими протоколами.

5.1. Протоколи відкритих мереж

В наш час використовується досить велика кількість мережевих протоколів, причому в рамках однієї і тієї ж мережі визначається відразу декілька з них. Прагнення до максимального упорядкування і спрощення процесів розробки, модернізації і розширення мереж визначило необхідність уведення стандартів, що регламентують принципи і процедури організації взаємодії абонентів комп'ютерних мереж. З цією метою була розроблена так звана *Еталонна модель взаємодії відкритих систем*, яка складається із семи рівнів. Кожний з рівнів представляє визначену групу функцій, необхідних для роботи комп'ютерної мережі.

Схема взаємодії рівнів в процесі обміну даними наведена на рис.5.1

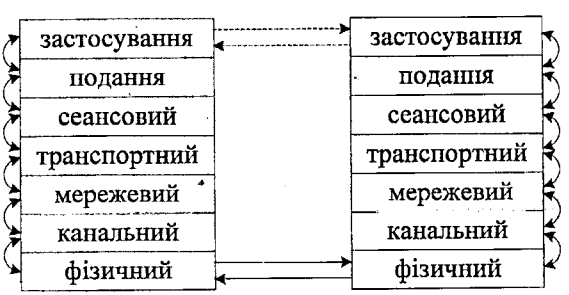


Рис. 5.1. Взаємодія мережевих протоколів

Основним, з погляду користувача, є *прикладний рівень*. Цей рівень забезпечує виконання прикладних процесів користувачів. Поряд із прикладними протоколами, він визначає протоколи передачі файлів, віртуального термінала, електронної пошти.

З точки зору користувача, програми практичного використання КСК обмінюються даними на рівні застосування (пунктирні лінії на рис. 5.1). Але насправді за цим обміном прихований процес, коли процедура

рівня застосування викликає процедуру нижчого рівня (рівня подання даних) і передає їй пакет даних, який призначений для передавання. Рівень подання даних викликає процедуру сеансового рівня і т.д. Безпосередній зв'язок між комп'ютерами (контролерами) здійснюється на найнижчому фізичному рівні. У комп'ютері, який отримав дані, вони потрапляють до програми користувача лише після того, як пройдуть від процедури до процедури всі рівні знизу догори.

Операції, які виконує процедура нижчого рівня, коли її викликає процедура верхнього рівня, називаються „сервісом", який надає нижчий протокол верхньому.

Шостий рівень протоколів називається *представницьким* рівнем подання даних. Він визначає єдиний для всіх систем синтаксис інформації, яка передається. Необхідність даного рівня обумовлена різною формою подання інформації в мережі передачі даних і комп'ютерах. Цей рівень відіграє важливу роль у забезпеченні "відкритості" систем, дозволяючи їм спілкуватися між собою незалежно від їхньої внутрішньої мови.

П'ятий рівень називають *сеансовим,* тому що основним його призначенням є організація сеансів зв'язку між прикладними процесами різних робочих станцій. На цьому рівні створюються порти для прийому і передачі повідомлень і організовуються з'єднання — логічні канали між процесами. Необхідність протоколів цього рівня визначається відносною складністю мережі передачі даних і прагненням забезпечити досить високу надійність передачі інформації.

Четвертий, *транспортний* рівень (рівень наскрізної передачі) служить для передачі даних між двома взаємодіючими відкритими системами й організації процедури сполучення абонентів мережі із системою передачі даних. На цьому рівні визначається взаємодія робочих станцій — джерела й адресата даних, організується і підтримується логічний канал (транспортне з'єднання) між абонентами.

Третій, *мережевий* рівень, призначений для маршрутизації інформації і керування мережею передачі даних. На відміну від попередніх, цей рівень у більшому ступені орієнтований на мережу передачі даних. Тут вирішуються питання керування мережею передачі даних, у тому числі маршрутизація і керування інформаційними потоками.

Канальний рівень забезпечує функціональні і процедурні засоби для встановлення, підтримки і розривання з'єднань на рівні каналів передачі даних. Процедури канального рівня забезпечують виявлення і, можливо, виправлення помилок, що виникають на фізичному рівні.

Фізичний рівень забезпечує механічні, електричні, функціональні і процедурні засоби організації фізичних з'єднань при передаванні бітів даних між фізичними об'єктами.

Чотири нижніх рівні утворюють транспортну службу комп'ютерної мережі, яка забезпечує передачу ("транспортування") інформації між робочими станціями, звільняючи більш високі рівні від цих задач.

У свою чергу, три верхніх рівні, що забезпечують логічну взаємодію прикладних процесів, функціонально поєднуються в абонентську службу.

У рамках еталонної моделі також визначаються послуги, що повинні забезпечувати її рівні. Послуги представляють собою функції, які виконуються на відповідному рівні еталонної моделі.

Зокрема, фізичний рівень повинен забезпечувати такі види послуг, як встановлення й ідентифікація фізичних з'єднань, організація послідовностей передачі бітів інформації, оповіщення про закінчення зв'язку.

Канальний рівень забезпечує організацію потрібної послідовності блоків даних і їх передачу, керування потоками між суміжними вузлами, ідентифікацію кінцевих пунктів канальних з'єднань, виявлення і виправлення помилок, оповіщення про помилки, що не виправлені на канальному рівні.

Мережевий рівень у числі основних послуг здійснює ідентифікацію кінцевих крапок мережних з'єднань, організацію мережних з'єднань, керування потоками блоків даних, забезпечення послідовностей доставки блоків даних, виявлення помилок і формування повідомлень про них, роз'єднання мережних з'єднань.

Транспортний рівень забезпечує встановлення і роз'єднання транспортних з'єднань, формування блоків даних, забезпечення взаємодії сеансових з'єднань із транспортними з'єднаннями, керування послідовністю передачі блоків даних, забезпечення цілісності блоків даних під час передачі, виявлення й усунення помилок, повідомлення про невиправлені помилки, надання пріоритетів у передачі блоків, передачу підтверджень про прийняті блоки, ліквідацію безвихідних ситуацій.

На сеансовому рівні надаються послуги, зв'язані з обслуговуванням сеансів і забезпеченням передачі даних у діалоговому режимі, встановленням сеансового з'єднання, обміном даними, керуванням обміном, синхронізацією сеансового з'єднання, повідомленнями про виняткові ситуації, відображенням сеансового з'єднання на транспортний рівень, завершенням сеансового з'єднання.

Представницький рівень забезпечує вибір вигляду подання даних, інтерпретацію і перетворення інформації, яка передається до вигляду, зручному для прикладних процесів, перетворення синтаксису даних, формування блоків даних.

При побудові КСК не завжди є необхідність у використанні всіх 7 рівнів протоколів. Якщо комп'ютерна мережа має нескладну архітектуру, яка не потребує складної маршрутизації і керування, протоколи сеансного,

транспортного і мережевого рівнів можуть не використовуватися або забезпечувати лише найпростіший сервіс. Тоді з рівня подання даних буде викликатися зразу процедура канального рівня.

5.2.Мережеве обладнання

Обладнання, яке використовується для утворення розподілених КСК, дуже різні. Частково це питання розглянуто у попередньому розділі. Як вже зазначалося раніше, основою побудови розподілених систем є комп'ютерні мережі.

Найчастіше комп'ютерні мережі розвиваються поступово, шляхом додавання нових комп'ютерів, приєднання незалежних сегментів, заміни застарілого обладнання і програмного забезпечення на більш досконале, додавання нових функціональних можливостей. Відповідно, на кожному етапі розвитку мережу утворюють дуже різнорідні елементи. Узагальнена схема комп'ютерної мережі зображена на рис. 5.2.

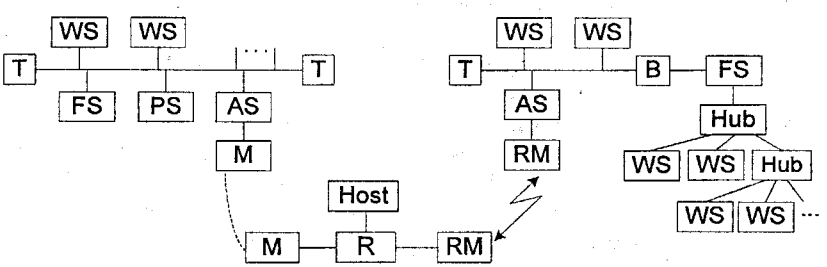


Рис. 5.2. Мережеве обладнання

WS – робоча станція, FS – файловий сервер, PS – сервер друку, AS – сервер доступу, T – термінатор, R – маршрутизатор, B – міст, Host – головний сервер, Hub – комутатор, M – модем, RM – радіомодем, ——— - кабельна лінія, -------- -телефонна лінія, ⟷ - радіоефір

Пасивне обладнання

Пасивне обладнання в основному визначається середовищем передачі даних в локальних мережах. Це може бути мідь (Copper) та оптоволокно (Fiber). *Мідний кабель* давно використовується як середовище передавання сигналів в різних діапазонах частот.

Основними електричними параметрами мідного кабелю є:

- хвильовий опір (імпеданс);

- полоса пропускання (максимальна частота сигналу, на якій затухання сигналу ще прийнятне);
- питоме затухання сигналу.

В засобах комунікацій застосовують декілька видів мідного кабелю.

Коаксіальний кабель (Coaxial) є центральною сигнальною жилою, що оточена ізоляцією та одним чи декількома шарами екрана. Відрізки кабелю з'єднуються коаксіальними роз'ємами. Коаксіальний кабель в комунікаціях найчастіше застосовується як розподілене середовище передачі – до одного сегмента (сукупності електрично безпосередньо з'єднаних відрізків) безпосередньо підключається багато абонентів. Такий спосіб з'єднання називається шинним.

Скручена пара (Twisted Pair) є парою скручених провідників, що утворюють одну сигнальну лінію. Завдяки скрутці зменшується паразитна ємність кабелю, його зовнішнє випромінювання та чутливість до електромагнітних завад. Скручені пари можуть бути як неекранованими – UTP (Unshielded Twisted Pair), так і екранованими – STP (Shielded Twisted Pair). Максимальна довжина сегмента витої пари не повинна перевищувати 100 м.

Для витої пари широко застосовуються модульні роз'єми типу RJ-45: розетки та вилки.

Оптоволоконний кабель (Fiber Optic Cable) містить одну чи декілька ниток оптоволокна, кожна з яких міститься в декількох оболонках, що забезпечують механічну міцність кабелю. Промінь світла розповсюджується по кабелю, відбиваючись від стінок центрального волокна. Існують багатомодові (Multi Mode) та одномодові (Single Mode) кабелі, що відрізняються траєкторіями проходження світлових променів.

Переваги оптоволокна такі. Крім високої пропускної здатності, що обчислюється гігабітами на секунду та великої допустимої довжини сегментів (ділянок кабелю без проміжного активного обладнання), що обчислюється кілометрами, оптоволокно забезпечує гальванічну розв'язку з'єднуваних вузлів з будь-якою необхідною напругою ізоляції. Воно нечутливе до електромагнітних перешкод і саме не є джерелом перешкод, забезпечує високу захищеність інформації від несанкціонованого прослуховування.

Але є й недоліки. Оптоволокно більш чутливе, ніж мідь до впливу зовнішнього середовища – від перепадів температур воно може тріскатись, від високого рівня радіації мутнішати, в результаті чого зросте затухання сигналу в кабелі.

Встановлення роз'єму до оптоволоконного кабелю може займати багато часу. Після розділення кабелю потрібно полірувати зріз на кінці з'єднувального роз'єму. Якість полірування контролюють з допомогою

мікроскопа, а при невдалому зрізі (сколі) можливо, що доведеться повторювати всю операцію спочатку. Є з'єднувачі, що не вимагають полірування – в них скол волокна занурюється в гель, коефіцієнт заломлення якого точно збігається з тим самим параметром волокна. Зварювання оптоволокна, що застосовується для з'єднання відрізків кабелю, вимагає, крім високої кваліфікації спеціаліста, ще й застосування вартісного обладнання.

Активне обладнання для Ethernet

Ethernet – це найпопулярніша мережева технологія, що представляє архітектуру мереж з розподіленим середовищем та широкою передачею. Це означає, що всі вузли мережі отримують пакет одночасно.

Мережеві адаптери для ПК випускаються для шин ISA, EISA, MCA, PCI, PC Card, VLB. Існують адаптери, що під'єднуються до стандартного LPT-порту ПК, перевага яких – відсутність потреб в системних ресурсах (порти, переривання і т.ін) та легкість під'єднання (без відкриття комп'ютерів). Їх недолік – при обміні вони значно завантажують процесор.

Основні властивості адаптерів такі:

- швидкість передачі – 10 чи 100 Мбіт на секунду, багато видів адаптерів мають обидва режими;
- можливість повного дуплексу для середовищ з розділеними лініями приймача та передавача (вита пара чи оптоволокно) – в багатозадачних системах дозволяє теоретично подвоїти пропускну здатність (при підтримці цього режиму на іншій стороні).

Repeater (повторювач) в мережах Ethernet на коаксіалі використовується як засіб подолання обмежень довжини кабелю і кількості під'єднаних вузлів (за електричними характеристиками). Класичний повторювач з внутрішніми термінаторами додається між кінцями сусідніх сегментів. Повторювач з зовнішніми термінаторами може під'єднуватись до Т-конекторів (чи трансіверів) в довільних місцях сегментів.

Hub (хаб) є обов'язковим (крім двоточкової мережі) з'єднувальним елементом мережі на витій парі та засобом розширення топологічних, функціональних та швидкісних можливостей для будь-яких середовищ передачі. Найпростіші хаби є багатопортовими повторювачами. Деякі порти хабів можуть мати набір роз'ємів BNC, RJ-45, AUI, забезпечуючи вибір середовища передачі. До порту хабу можна під'єднувати як окремий вузол, так і інший хаб чи сегмент коаксіалу. Хаби з набором різнотипних портів дозволяють об'єднувати сегменти мереж з різними кабельними системами.

Intelligent hub (інтелектуальний хаб) має більш складну архітектуру з вбудованим мікроконтролером, що дозволяє керувати мережею (звичайно на основі засобів SNMP). В хабі знаходиться апаратно-програмний SNMP-агент, що веде базу даних про стан ресурсів.

Менеджер, що керує хабом, взаємодіє з агентами по мережі. Керування хабу забезпечує можливість централізованого керування та діагностики стану вузлів мережі, захист від несанкціонованого доступу, сегментування мережі для розподілення трафіку.

Stackable hub (нарощуваний хаб) має спеціальні засоби з'єднання декількох хабів в стек, що виступає в ролі єдиного цілого. При цьому звичайно інтелектуальність одного хабу робить інтелектуальним весь стек. Відстань між хабами в стеку може бути коротким (локальний стек) чи довгим, до сотень метрів (розподілений стек, більш гнучкий елемент для оптимізації кабельної системи).

Full-Duplex hub (повнодуплексний хаб) – варіант комутуючого, в якого порт може одночасно приймати та передавати пакети.

Модемний зв'язок

Під'єднання до комп'ютера або контролера пристроїв, які знаходяться на значних відстанях (1 км і більше) здійснюється за допомогою модемів. Термін "модем" походить від комбінації двох процесів: модуляції і демодуляції. Використання модуляції дозволяє збільшити завадостійкість зв'язку і, відповідно, швидкість і дальність передавання.

Модемні канали зв'язку використовують переважно телефонні лінії. При передаванні даних у межах міста в утворенні каналу зв'язку бере участь і станційне обладнання, яке здійснює комутацію, фільтрацію і підсилення сигналів. Структурна схема системи з модемним зв'язком показана на рис. 5.3. Це накладає відбиток на характеристики модемних каналів – збільшує дальність передавання, але обмежує смугу частот модуляції і, відповідно, швидкість передавання. Останнім часом використовують також радіомодеми, але для їх застосування необхідно отримувати спеціальний дозвіл в установах нагляду за використанням (та "засміченням") ефіру.

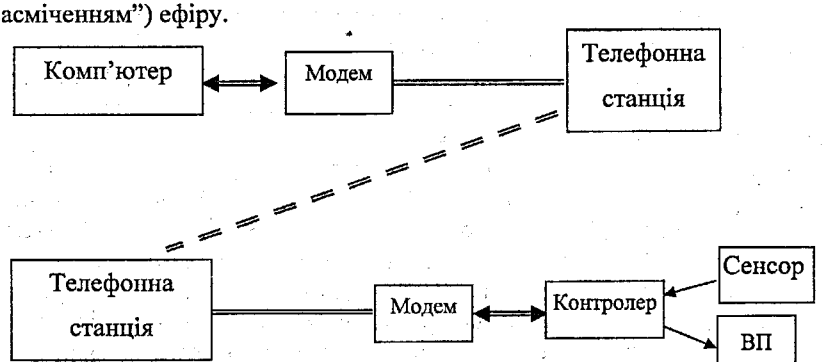


Рис. 5.3. Схема зв'язку між двома комп'ютерами через модеми

Телефонна лінія зв'язку між модемами може бути виділеною і комутованою. Комутована лінія означає, що для встановлення зв'язку через станцію необхідно набрати телефонний номер іншого модема (це робить модем, який ініціює зв'язок за командою комп'ютера), причому таким способом можна здійснювати з'єднання з різними модемами по черзі. Виділена лінія означає, що необхідна комутація між парою здійснена на станції заздалегідь і назавжди.

Модеми розділяються на зовнішні і внутрішні. Зовнішні модеми мають окреме джерело живлення і підключаються до комп'ютера (контролера) через комунікаційний порт. Внутрішні модеми виготовляються у вигляді вставної плати до комп'ютера або мікросхеми, що встановлюється безпосередньо на материнській платі, але для керування внутрішнім модемом все одно виділяється комунікаційний порт.

З рис. 5.3 видно, що модем має два канали зв'язку: з комп'ютером та з іншим модемом через телефонну мережу. Відповідно обмін даними через ці лінії здійснюється за своїми окремими протоколами.

Обмін даними і командами між комп'ютером і модемом здійснюється за допомогою допоміжних ліній комунікаційного порту та at-команд. За основу протоколу обміну даними взято модеми фірми Hayes, а модеми, які підтримують цей протокол, називають Hayes-сумісними. На жаль, не всі модеми підтримують повний набір команд. Детальна інформація про команди кожного модема наводиться у його паспорті.

АТ-команди є послідовностями ASCII-кодів символів, які починаються з символів at. Команди можна поділити на три групи:

- команди налаштовування параметрів модема;
- команди встановлення з'єднання між модемами і переходу в режим передавання інформації;
- команда повернення з режиму передавання інформації в командний режим.

Основні at-команди наведені у додатку.

Налаштовування параметрів модема може здійснюватися тимчасово (на один сеанс роботи) і постійно – шляхом запису відповідних кодів у регістри модема.

Встановлення з'єднання відбувається за ініціативою одного з модемів, який здійснює виклик за командою

*atd*параметри.

Інший модем відповідає на виклик або автоматично (якщо в ньому встановлений режим автоматичної відповіді), або за командою

ata.

Після цього модеми виконують процедуру узгодження параметрів зв'язку (Hand Shaking) і переходять у режим обміну інформацією. Це означає, що всі дані, які надходять з комп'ютера на модем, зразу передаються на інший модем, а звідти на інший комп'ютер. З цього

моменту керування потоками даних здійснюється за допомогою протоколів вищого рівня, наприклад, протоколу "Z-modem".

Після закінчення обміну інформацією модеми повертаються у командний режим. Для цього в кінці потоку даних ставиться Escape-послідовність

+++.

Для того, щоб ця послідовність була правильно розпізнана, до і після неї повинні бути зроблені паузи у потоці даних не менше 1с.

Після повернення у командний режим може бути подана команда розриву з'єднання

*ath*0.

На цьому сеанс зв'язку закінчується.

Обмін даними між модемами на фізичному і частково канальному рівні здійснюється у відповідності до протоколів, стандартизованих CCITT (від французького Comite Consultatif Telegraphique et Telephonique). Це протоколи серії V: V23, V32, V32bis, V42 тощо. Кожен протокол визначає можливі стандартні швидкості передавання бітів, спосіб модуляції, спосіб захисту від завад, алгоритм стискання інформації. Як правило, модеми з більшим номером забезпечують більшу швидкість та надійність передавання.

5.3. Алгоритми передачі даних

Використання стандартних протоколів передавання даних рекомендоване і забезпечує високу ефективність і надійність зв'язку. Але в КСК, в яких використовуються мікроконтролери, часто використовується базове програмне забезпечення (Windows, Linux тощо), для якого написані готові процедури реалізації стандартних протоколів самотужки, використовуючи тільки систему програмування певного контролера – задача надто складна і трудомістка. Тому розробникам КСК часто доводиться використовувати спрощені алгоритми обміну даними.

Приймання та передавання багатобайтових даних

У системах керування найчастіше передаються не поодинокі байти, а багатобайтові дані: цілого типу зі знаком (2 байти) та реального типу з плаваючою точкою (6 байтів). У наступному прикладі наведені процедури прийому та передачі таких даних. У цих процедурах тип вхідного/вихідного даного позначений TData. Головна ідея побудови цих алгоритмів – суміщення у пам'яті необхідного даного з масивом байтів і приймання/передача масива байтів (тим самим приймаються/передаються окремі байти багатобайтового даного).

```
Procedure InDat(var y:TData);
 Var
   a : array [1..sizeof(TData)] of byte absolute y;
   i : 1..sizeof(TData);
 Begin
  for i:=1 to sizeof(TData) do
    begin
     repeat until (port[$3FD] and 1) <> 0;
     a[i] := port[$3F8]
    end
 End.
Procedure OutDat(x:TData);
 Var
   a : array [1..sizeof(TData)] of byte absolute x;
   i : 1..sizeof(TData);
 Procedure MyDelay(t:byte);
  var
   L : boolean;
   n : byte;
  begin
   for n:=1 to t do
     repeat (port[$3FD] and 1) <> 0 until true;
  end;
 Begin
  for i:=1 to sizeof(TData) do
    begin
     port[$3F8] := a[i];
     MyDelay(3)          { Затримка для узгодження швидкості передачі та
                           прийому }
    end
 End.
```

Забезпечення достовірності передавання даних

Процес передавання даних практично завжди відбувається в умовах дії завад, які спотворюють дані. Так, наприклад, якщо інтенсивність завад складає 10^{-3}, то спотворюється лише 1 біт з тисячі. Але це означає, що при передаванні лише 125 байтів з великою ймовірністю спотвориться 1 байт. Якщо інформація, що передається, не припускає спотворення (наприклад, інформація комерційного обліку), то така система зв'язку не могла б використовуватись.

Для запобігання помилок використовують різні алгоритми, наприклад, алгоритми з підрахунком контрольної суми, наведений на рис. 5.4.

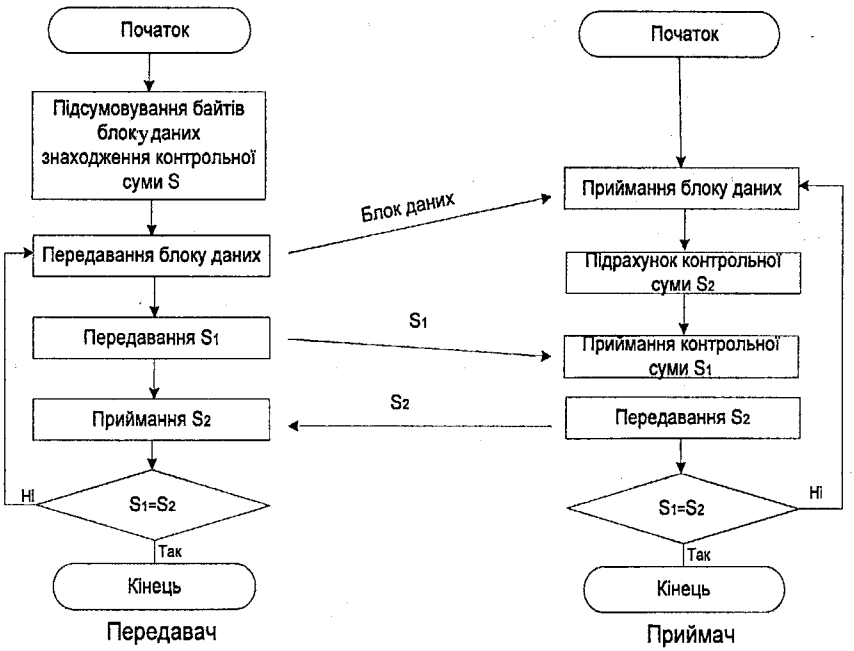


Рис. 5.4. Алгоритм зв'язку з обміном контрольними сумами

Для забезпечення високої швидкості передавання даних необхідно підібрати оптимальний розмір блоку даних. В стандартних протоколах передавання даних „Y- модем" і „Z- модем" розмір блоку даних адаптивно змінюється: при появі помилок зменшується, а при відсутності – збільшується.

Підвищити достовірність даних можна також використанням кодів з виправленням помилок, наприклад, циклічних(CRC).

5.4. Структурна оптимізація КСК

Структурна оптимізація розподіленої КСК здійснюється методами теорії графів. Критерієм оптимальності є вартісна функція, яка кожному зв'язку між елементами КСК ставить у відповідність певну вартість (наприклад, вартість прокладання кабеля). Ця вартість є вагою ребер в графі.

Задачу структурної оптимізації КСК звичайно спрощують,

обмежившись певним класом структур:

- лінійною;
- кільцевою;
- ієрархічною.

Оптимізація лінійної КСК

З позицій теорії графів задача оптимізації лінійної КСК є задачею знаходження найкоротшого шляху в графу, який проходить через всі вершини. Приклад розв'язання задачі зображений на рис. 5.5.

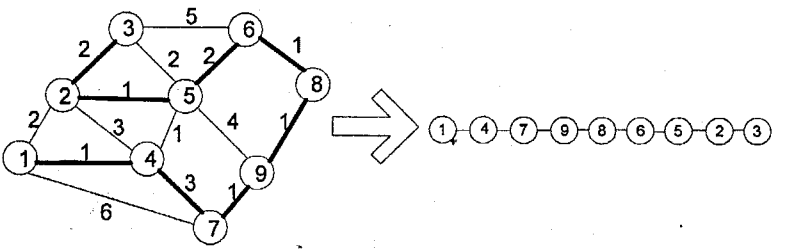


Рис. 5.5. Приклад побудови лінійної КСК

Для розв'язання задачі можуть використовуватися різні алгоритми, наприклад, алгоритм „гілок і границь", наведений у додатку.

Оптимізація кільцевої КСК

Задача оптимізації кільцевої КСК є *задачею комівояжера.*

Комівояжер повинен вийти з певного міста, відвідати один раз в невідомій послідовності міста 2,3, 4, .., n та повернутися в перше місто. Відстань між всіма містами відома. В якій послідовності необхідно обходити міста, щоб замкнутий шлях комівояжера був найкоротшим.

Тур комівояжера може бути описаний циклічною перестановкою $t=(j_1, j_2, j_3, . ., j_n, j_1)$, причому всі $j_1, j_2, j_3, . ., j_n$ – різні номери.

Відстані між парами вершин Cij утворюють матрицю С.

Задача полягає в тому, щоб мінімізувати функціонал:

$$L = L(t) = C_{j_k j_{k+1}}.$$

Приклад розв'язання задачі комівояжера наведений у додатку.

Оптимізація ієрархічної КСК

Задача оптимізації ієрархічної КСК є задачею побудови мінімального остовного дерева.

Деревом графа називається такий зв'язний підграф, який не має циклів. Остовне дерево містить всі вершини графа. Мінімальне остовне дерево з усіх остовних дерев має найменшу сумарну довжину ребер.

Доведено, що жадібний алгоритм (Алгоритм Пріма-Краскала)

дозволяє знайти мінімальне остовне дерево. Ідея алгоритму: спочатку вибирають найкоротше ребро, а його кінцеві вершини включають до дерева – це початок дерева. Потім приєднують до нього найкоротше ребро з тих, що виходять з вершин, які належать до дерева, а входять у вершину, яка не належить до дерева. І так далі, поки всі вершини не увійдуть до складу дерева (рис. 5.6.).

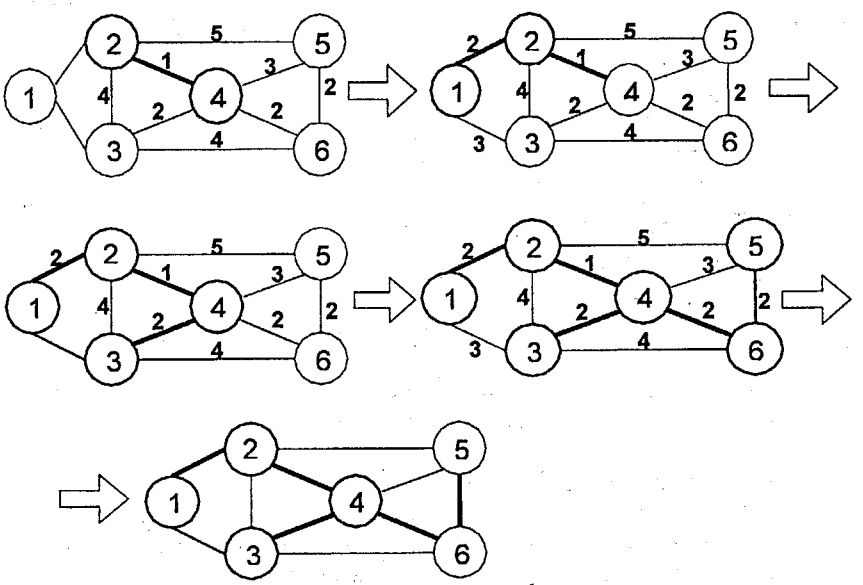


Рис. 5.6. Реалізація алгоритму Пріма-Краскала

Приклад реалізації алгоритму Пріма-Краскала наведений у додатку.

Контрольні запитання і завдання

1. Охарактеризуйте причини розробки еталонної 7-рівневої моделі взаємодії відкритих мереж.
2. Охарактеризуйте структуру 7-рівневої моделі, особливості та призначення кожного рівня.
3. Назвіть основні види пасивного обландання комп'ютерних мереж, їх особливості, переваги та недоліки.
4. Які технічні елементи комп'ютерних мереж належать до активного обладнання? Охарактеризуйте призначення основних активних мережних засобів: мережевий адаптер, повторювач, хаб, модем.
5. Які методи підвищення достовірності передавання даних використовуються в КСК?

6. Запропонуйте структуру блока даних при передаванні файлів між двома комп'ютерами.
7. Якому співвідношенню, на Вашу думку, повинні відповідати швидкості передавання даних від комп'ютера до модема і від модема до іншого модема?
8. Складіть програму оптимізації лінійної мережі з 12 комп'ютерів, відстані між якими задані графом.
9. Складіть програму оптимізації лінійної мережі з 10 комп'ютерів, відстані між якими задані графом.
10. Складіть програму оптимізації лінійної мережі з 15 комп'ютерів, відстані між якими задані графом.

Література

1. Кулаков Ю.А., Омелянский С.В. Компьютерные сети. Выбор, установка, использование и администрирование. – К.: Юниор, 1999. – 544 с.
2. Гук М. Аппаратные средства IBM PC. Энциклопедия. – СПб.: Питер, 1999. – 816 с.
3. Флинт Д. Локальные сети ЭВМ: архитектура, принципы построения, реализация: Пер. с англ. – М.: Финансы и статистика, 1986. – 359 с.
4. Холидей К. Секреты ПК. – К.: Диалектика, 1995. - 416 с.
5. В.М.Дубовой, Р.Н.Квєтний. Програмування персональних комп'ютерів систем управління. – Вінниця: ВДТУ, 1999. – 110 с.
6. Форсюк В.В. Модемний словник. Методичні вказівки для користувачів. – К.: Євроіндекс, 1994. – 80 с.
7. Бондарев В.М. и др. Основы программирования. – Харьков: Фолио, 1997.
8. Свами М.Н., Тхуласираман К. Графы, сети и алгоритмы: Пер. с англ. – М.: Мир, 1984. – 454с.
9. Седжвик Р. Фундаментальные алгоритмы на С. Алгоритмы на графах: Пер. с англ. – СПб.: ДиаСофтЮП, 2003. – 480с.

ПІСЛЯМОВА

Створення комп'ютеризованих систем керування – це один з найсучасніших і найдинамічніших напрямків науково-технічного прогресу. При спостереженні цього бурхливого процесу може скластися враження, що розробники КСК „сп'яніли" від можливостей, які надають їм сучасні високі технології. КСК розповсюджуються в усіх напрямках: донизу, аж до дитячих іграшок, домашніх прасок, інтелектуальних вимикачів світла тощо, і доверху, аж до танцюючих роботів, самокерованих автомобілів, автоматичних космічних станцій тощо. І це не дивно, адже за кілька десятиріч вартість потужного процесора знизилася з мільйонів до одиниць доларів, а технічні характеристики в той же час покращились на стільки ж порядків. Сьогодні, наприклад, нікого не дивує кишенькова Flash-пам'ять на 4 Гігабайти, у яку можна записати кілька тисяч книжок, а лише 15 років тому досить сучасною вважалася ЕОМ СМ-1420 з обсягом оперативної пам'яті 500 кілобайт.

Технічні можливості сучасної елементної бази КСК постійно покращуються. Але фундаментальні принципи побудови КСК або залишаються незмінними, або розвиваються. З'являється можливість реалізувати в процесі керування такі алгоритми, які раніше розглядалися лише теоретично і не використовувалися через брак пам'яті і швидкодії.

Автори сподіваються, що підготовлений ними посібник сприятиме майбутнім фахівцям з комп'ютеризованих систем керування успішно застосовувати і розвивати принципи побудови КСК для створення нових систем і галузей їх застосування.

Приклади програм

1. Процедури статистичної обробки даних

```
Procedure Mx(x : real; var M : real);  {обчислення поточного середнього}
 Begin
  M:=(M*N+x)/(N+1)
 End;

Procedure Dx(M, x : real; var D : real); {обчислення поточної дисперсії}
 Begin
  D:=(D*(N-1)+(x-M)*(x-M))/N
 End;

Procedure Sx(D : real; var S : real);    {обчислення СКВ}
 Begin
  S:=sqrt(D)
 End;
```

2. Програма кореляційної обробки даних

```
Const N=50;                          {довжина черги}
Var
 x : array [1..N] of real;           {черга процесу x(t)}
 y : array [1..2*N] of real;         {черга процесу y(t)}
 R : array [1..N] of real;           {кореляційна функція}
 i, t : byte;

Procedure Init;                      {ініціалізація черги}
 Begin
  for i:=1 to N do x[i]:=0;
  for i:=1 to 2*N do y[i]:=0;
 End;

Procedure ReadXY;                    {занесення нових даних до черг}
 Var
  a,b : real;
 Begin
  for i:=1 to N-1 do x[i+1]:=x[i]; x[1]:=a;
  for i:=1 to 2*N-1 do y[i+1]:=y[i]; y[1]:=b;
 End;
```

```
Procedure Process;                    {обчислення кореляційної функції}
 Begin
  for t:=1 to N do
    begin
      R[t]:=0;
      for i:=1 to N do R[t]:=R[t]+x[i]*y[i+t]
    end;
 End;
```

3. Програма лінійної фільтрації даних

```
uses dos;
Const
 BufLength=10;
Type
 TData=record                  {значення сигналу x пов'язані з моментами їх
       x:real;                  надходження}
       t:real;
      end;
Var
 Buffer:array [1..BufLength] of TData;  {Черга вхідних даних}

Procedure Filter_LF_lin_i(x:TData; var y:real); {ідеальний лінійний
                                                 фільтр низьких частот W(p)=
                                                 1/Tp }
 Begin
  Buffer[2]:=Buffer[1];
  Buffer[1]:=x;
  y:=y+Buffer[1].x*((Buffer[1].t-Buffer[2].t))
 End;

Procedure Filter_LF_lin_r(N:real; x:TData; var y:real); {реальний фільтр
                                                         низьких частот W(p)=
                                                         1/(Tp+1)}
 Begin
  y:=(N*y+x.x)/(N+1)
 End;

Procedure Filter_HF_lin_i(x:TData; var y:real); {ідеальний лінійний
                                                 фільтр високих частот W(p)=
                                                 Tp }
 Begin
  Buffer[2]:=Buffer[1];
```

```
  Buffer[1]:=x;
  if (Buffer[1].t-Buffer[2].t)>0 then y:=(Buffer[1].x-Buffer[2].x)/(Buffer[1].t-
Buffer[2].t)
                    else if (Buffer[2].x-Buffer[1].x)=0 then y:=0
                                        else y:=1.0E+6*Buffer[1].x
 End;

Procedure Filter_HF_lin_r(N:real; x:TData; var y:real); {реальний фільтр
                                                    високих частот W(p)=
                                                    Tp/(Tp+1) }
 Begin
  Buffer[2]:=Buffer[1];
  Buffer[1]:=x;
  if (Buffer[1].t-Buffer[2].t)>0 then y:=(N*y+((Buffer[1].x-
                                    Buffer[2].x)/(Buffer[1].t-
                                    Buffer[2].t)))/(N+1)
                  else
                    if (Buffer[1].x-Buffer[2].x)=0 then y:=0
                                        else y:=1.0E+6*Buffer[1].x
 End;
```

4. Програма нелінійної фільтрації даних

```
Procedure Filter_NF_lin_r(K:real; x:byte; var y:byte);
```

{Реальний нелінійний фільтр низьких частот. **Інерційність такого фільтра пропорційна відхиленню вхідного даного від усередненого значення.***}*

```
 Var
  N:real;
 Begin
  N:=K*abs(x-y);
  y:=trunc((N*y+x)/(N+1))
 End;
```

5. Швидке перетворення Фур'є

В залежності від переданих параметрів, може виконуватись як пряме, так і зворотне перетворення.

Вхідні параметри:

nn – кількість значень функції. Повинна бути степенем двійки. Алгоритм не перевіряє правильність переданого значення.

a – array [0 .. 2*nn-1] of Real – значення функції. i-му значенню відповідають елементи a[2*i] (дійсна частина) і a[2*i+1] (уявна частина).

InverseFFT – напрямок перетворення (True – зворотне, False – пряме).

Вихідні параметри:

a – результат перетворення.

```
#include "ap.h"
void fastfouriertransform(ap::real_1d_array& a, int nn, bool inversefft);
void fastfouriertransform(ap::real_1d_array& a, int nn, bool inversefft)
{
  int ii, jj, n, mmax, m, j, istep, i, isign;
  double wtemp, wr, wpr, wpi, wi, theta, tempr, tempi;

  if( inversefft ) { isign = -1; } else { isign = 1; }
  n = 2*nn;
  j = 1;
  for(ii = 1; ii <= nn; ii++)
  { i = 2*ii-1;
    if( j>i )
    { tempr = a(j-1); tempi = a(j);
      a(j-1) = a(i-1); a(j) = a(i);
      a(i-1) = tempr; a(i) = tempi;  }
    m = n/2;
    while(m>=2&&j>m)
    { j = j-m;
      m = m/2; }
    j = j+m;
  }
  mmax = 2;
  while(n>mmax)
  { istep = 2*mmax;
    theta = double(2*ap::pi())/double(isign*mmax);
    wpr = -2.0*ap::sqr(sin(0.5*theta));
    wpi = sin(theta);
    wr = 1.0;   wi = 0.0;
    for(ii = 1; ii <= mmax/2; ii++)
    { m = 2*ii-1;
      for(jj = 0; jj <= (n-m)/istep; jj++)
      { i = m+jj*istep;
        j = i+mmax;
        tempr = wr*a(j-1)-wi*a(j); tempi = wr*a(j)+wi*a(j-1);
        a(j-1) = a(i-1)-tempr; a(j) = a(i)-tempi;
```

```
            a(i-1) = a(i-1)+tempr;  a(i) = a(i)+tempi;   }
          wtemp = wr;
          wr = wr*wpr-wi*wpi+wr;     wi = wi*wpr+wtemp*wpi+wi;
        }
        mmax = istep;
    }
    if( inversefft )
    {   for(i = 1; i <= 2*nn; i++)  { a(i-1) = double(a(i-1))/double(nn);  }    }
}
```

Наведений приклад реалізує сплайн-інтерполяцію і екстраполяцію з використанням полінома першого порядку. Асимптотичне значення функції $y(x)$, що описана в масиві y, визначається за допомогою лінійної екстраполяції з обчисленням першої похідної на кінцях відрізка вузлів інтерполяції x_0 та x_n за формулами численного диференціювання.

```
const
  n = 8;
  h = 0.1;
  x0 = 0;
  xn = 0.8;
  y : array [0..n] of real = (0,10,4,2,1.5,2,5,9,13);
  x  = -0. 1;
var
 y : real;
 i : integer;
begin
if x<=x0 then
 y:=y[0]+(x-x0)*(y[1]-y[0])/h
else
 if x>=xn then
  y:=y[n]+(x-xn)*(y[n]-y[n-1])/h
 else
  begin
  i:=trunc((x-x0)/h);
  y:=y[i]+(y[i+1]-y[i])*(x-i*h-x0)/h;
  end;
writeln('При x=',x:8:6,' y=',y:8:6);
end.
```

У наступній програмі дані приймаються і передаються через COM1. Комп'ютер виконує екстраполяцію степеневим поліномом.

```
uses crt,dos;
const N=3;
```

```
type Tdata=record
                d:byte;
                t:real
              end;
var c : char;
    x, y : Tdata;
    Bufer : array [1..N] of Tdata;
    i : byte;
Procedure InitCom1;
 begin
  port[$3FB]:=$80; port[$3F8]:=12; port[$3F9]:=0; port[$3FB]:=3;
 end;

Procedure SendByte(b:byte);
 begin
  port[$3F8]:=b;
 end;

Function ChekCom1:boolean;
 Begin
  ChekCom1:=(port[$3FD] and 1) = 1
 End;

Procedure ReadCom1(var b:Tdata);
 Var h,m,s,s100:word;
 begin
  repeat until ChekCom1;
  b.d:=port[$3F8];
  gettime(h,m,s,s100);   b.t:=3600.0*h+60.0*m+s+0.01*s100;
 end;

Procedure Ekstra(dt:real; x:Tdata; var y:Tdata);
 Var
  Y0,Y1,Y1_,Y2:real;
 Begin
  For i:=1 to N-1 do Bufer[i]:=Bufer[i+1]; Bufer[N]:=x;
  Y0:=Bufer[1].d;
  Y1:= (Bufer[1].d - Bufer[2].d) / (Bufer[1].t - Bufer[2].t);     {Перша
похідна}
  Y1_:= (Bufer[2].d - Bufer[3].d) / (Bufer[2].t - Bufer[3].t);
  Y2:= (Y1-Y1_) / ((Bufer[1].t - Bufer[3].t) /2);        {Друга похідна}
  y.d:=trunc(Y0 + Y1*dt+Y2*dt*dt / 2);              {Екстрапляційний
поліном}
```

```
  y.t:= Bufer[3].t + dt;
 End;

BEGIN
 InitCom1;
 Repeat
  Repeat
   ReadCom1(x);
   Ekstra (1, x, y);
   SendByte(y.d)
  Until keypressed;
  c:=readkey;
 until c=#27
END.
```

6. Процедури реалізації лінійних законів керування

```
Procedure P(k:real; x,y:Tdata; var y:Tdata);
 Begin
  y:=k*(x-y);
 End;

Procedure I( x,y:Tdata; var u:Tdata);
 Begin
  S:=S+(x-y)/dt;
  u:=S;
End;

Procedure D( x:Tdata; var y:Tdata);
 Begin
  U:=(x-y-x0+y0)/dt;
  x0:=x;
  y0:=y;
End;

Procedure PI(N:real; x:TData; var y:TData);
 Begin
  u:=(N*u+(x-y))/(N+1)
 End;

Procedure PD(k:real; x,y:Tdata; var u:Tdata);
 Begin
  u:=k*(x-y)+(x-y-x0+y0)/dt;
```

```
 x0:=x;
 y0:=y;
End;
Procedure PID(k1,k2:real; x,y:Tdata; var u:Tdata);
 Begin
 S:=S+(x-y)/dt;
 u:=k1*(x-y)+k2*(x-y-x0+y0)*dt+S;
 x0:=x;
 y0:=y;
End;
```

7. Пошук оптимального маршруту

У наступній програмі a – множина розглянутих вершин, b – масив відстаней від початкової вершини до кожної з решти вершин графа, c – масив вершин, через які проходять найкоротші шляхи.

```
Const
 N=8;
 M=999;
 D:array [1..N,1..N] of real=((0, 23, 12, M, M, M, M, M),
                              (23, 0, 25, M, 22, M, M, 35),
                              (12, 25, 0, 18, M, M, M, M),
                              (M, M, 18, 0, M, 20, M, M),
                              (M, 22, M, M, 0, 23, 14, M),
                              (M, M, M, 20, 23, 0, 24, M),
                              (M, M, M, M, 14, 24, 0, 16),
                              (M, 35, M, M, M, M, 16, 0));
Var
 a:set of 1..N;
 c,P:array[1..N]of integer;
 b:array[1..N]of real;
 V0,j,k,all,VN:integer;
 bmin:real;
Begin
 write('Початкова вершина: '); readln(V0);
 write('Кінцева вершина: ');readln(VN);
 writeln;
 for j:=1 to N do
  begin
   c[j]:=V0;
   b[j]:=D[V0,j]
  end;
 a:=[V0]; c[V0]:=0;
```

```
 while a<>[1..N] do
 begin
  bmin:=M;
  for k:=1 to N do
   if (not (k in a)) and (b[k]<bmin) then
    begin
     bmin:=b[k];
     j:=k
    end;
  a:=a+[j];
  for k:=1 to N do
   if b[k]>(b[j]+D[j,k]) then
    begin
     b[k]:=b[j]+D[j,k];
     c[k]:=j
    end;
 end;
 j:=1;
 P[j]:=VN;
 while c[P[j]]<>0 do
  begin
   inc(j);
   P[j]:=c[P[j-1]];
  end;
 for k:=j downto 2 do write(P[k],'-->'); writeln(P[1]);
 writeln('Lmin=',b[VN]:6:1);
 readln
 End.
```

Послідовність кроків алгоритму для випадку:

- початкова вершина - 3
- кінцева вершина - 8

1	a	0	0	1	0	0	0	0	0
	b	12	25	0	18	999	999	999	999
	c	3	3	0	3	3	3	3	3
2	a	1	0	1	0	0	0	0	0
	b	12	25	0	18	999	999	999	999
	c	3	3	0	3	3	3	3	3
3	a	1	0	1	1	0	0	0	0
	b	12	25	0	18	999	38	999	999
	c	3	3	0	3	3	4	3	3
4	a	1	1	1	1	0	0	0	0
	b	12	25	0	18	47	38	999	60
	c	3	3	0	3	2	4	3	2

5	a	1	1	1	1	0	1	0	0
	b	12	25	0	18	47	38	62	60
	c	3	3	0	3	2	4	6	2
6	a	1	1	1	1	1	1	0	0
	b	12	25	0	18	47	38	61	60
	c	3	3	0	3	2	4	5	2
7	a	1	1	1	1	1	1	0	1
	b	12	25	0	18	47	38	61	60
	c	3	3	0	3	2	4	5	2
8	a	1	1	1	1	1	1	1	1
	b	12	25	0	18	47	38	61	60
	c	3	3	0	3	2	4	5	2

Мінімальний шлях 3-->2-->8, Lmin= 60.0

8. Оптимальне керування потоками в мережі

```
Const
  N=5;
  PS:array[1..N,1..N]of real= ((0,15,3,0,2),     { Матриця пропускних
здатностей }
                               (5,0,4,7,3),
                               (3,4,0,2,0),
                               (0,7,0,0,8),
                               (0,3,0,8,0));
Var
  V0,VN,V:byte;
  Q:array[1..N*(N-1),1..2]of byte; { Черга для пошуку збільшуючого ланцюга
}
  BQ,EQ:byte;
  Found:boolean;
  SetV:set of 1..N;
  Way:array[1..N] of byte;  { Збільшуючий ланцюг }
  Potok:array[1..N,1..N] of real;      { Шукана матриця потоків}
  i,j : byte;
  dPmin:real;

Procedure MinIzm;  { Знаходження величини збільшення потоку }
  var
    dP:real;
  begin
    V:=N;
    if Potok[Way[V-1],Way[V]]>=0
```

```
      then dPmin:=PS[Way[V-1],Way[V]]-Potok[Way[V-1],Way[V]]
      else dPmin:=PS[Way[V],Way[V-1]]-Potok[Way[V],Way[V-1]];
  while Way[V-1]<>V0 do
   begin
    dec(V);
    if Potok[Way[V-1],Way[V]]>=0
       then dP:=PS[Way[V-1],Way[V]]-Potok[Way[V-1],Way[V]]
       else dP:=PS[Way[V],Way[V-1]]+Potok[Way[V-1],Way[V]];
    if dP<dPmin then dPmin:=dP
   end
 end;

Procedure CorPot; { Зміна потоку у збільшуючому ланцюгу }
 begin
  V:=N;
  while Way[V]<>V0 do
   begin
    if Potok[Way[V-1],Way[V]]>=0
       then Potok[Way[V-1],Way[V]]:=Potok[Way[V-1],Way[V]]+dPmin
       else Potok[Way[V-1],Way[V]]:=Potok[Way[V-1],Way[V]]-dPmin;
    Potok[Way[V],Way[V-1]]:=-Potok[Way[V-1],Way[V]];
    dec(V);
   end
 end;

Procedure Poisk;  { Пошук збільшуючого ланцюга }
Begin
 SetV:=[1..N];
 Found:=false;
 while (not Found) and (SetV<>[]) and (BQ<=EQ) do
  begin
   inc(BQ);
   while Q[BQ,1]=VN do inc(BQ);
   if BQ<=EQ then
    begin
     j:=BQ;
     while j>0 do
      begin
       SetV:=SetV-[Q[j,1]];
       j:=Q[j,2]
      end;
     for V:=1 to N do
      if (V in SetV) and (PS[Q[BQ,1],V]<>0) and (not Found)
```

```
        then begin
             inc(EQ);
             Q[EQ,1]:=V; Q[EQ,2]:=BQ;
             if V=VN then Found:=true;
            end;
   end;
  i:=N;
  j:=EQ;
  while j>0 do
   begin
    Way[i]:=Q[j,1];
    dec(i);
    j:=Q[j,2]
   end;
 end
End;

BEGIN
 writeln('Введіть вимік і стік потоку:');
 readln(V0,VN);
 BQ:=0;
 EQ:=1;
 Q[EQ,1]:=V0; Q[EQ,2]:=0;
 for i:=1 to N do
  for j:=1 to N do
    Potok[i,j]:=0; {Тривіальний розв'язок}
 while BQ<=EQ do
  begin
   Poisk;
   MinIzm;
   CorPot;
  end
END.
```

9. Програми керування через COM під Windows

```
#include <windows.h>
#include <TCHAR.h>
#include <stdio.h>
#include <conio.h>
#include <iostream>

#pragma once
```

```
using namespace std;

struct COMMDATA
{
	COMMDATA()
	{
		b55 = 0x55;
		bAA = 0xAA;
		bHi = 0;
		bLo = 0;
		bTl = b55 + bAA + bHi + bLo;
	}

	BOOL IsSummCorrect() const
	{
		return bTl == (b55 + bAA + bHi + bLo);
	}

	void BuildSumm()
	{
		bTl = b55 + bAA + bHi + bLo;
	}

	BYTE b55;
	BYTE bAA;
	BYTE bHi;
	BYTE bLo;
	BYTE bTl;
};

int main(int argc, _TCHAR* argv[])
{
	HANDLE hPort = CreateFile(_T("COM1"), GENERIC_READ | GENERIC_WRITE,
	0,
		NULL, OPEN_EXISTING, 0, NULL);
	if (INVALID_HANDLE_VALUE == hPort)  return -1;

	COMMTIMEOUTS timeout;
	timeout.ReadIntervalTimeout         = 10;
	timeout.ReadTotalTimeoutConstant    = 10;
	timeout.ReadTotalTimeoutMultiplier  = 10;
	timeout.WriteTotalTimeoutConstant   = 10;
	timeout.WriteTotalTimeoutMultiplier= 10;
	BOOL IsOk = SetCommTimeouts(hPort, &timeout);
	if (!IsOk) return -2;

	DCB  PortDCB;
	IsOk = GetCommState(hPort, &PortDCB);
	if (!IsOk) return -3;
```

```
// configuration Comm
PortDCB.BaudRate = CBR_19200;      // Current baud
PortDCB.fBinary = TRUE;            // Binary mode; no EOF check
PortDCB.fParity = FALSE;           // Enable parity checking
PortDCB.fOutxCtsFlow = FALSE;      // No CTS output flow control
PortDCB.fOutxDsrFlow = FALSE;      // No DSR output flow control
PortDCB.fDtrControl = 0;//DTR_CONTROL_ENABLE;
// DTR flow control type
PortDCB.fDsrSensitivity = FALSE;   // DSR sensitivity
PortDCB.fTXContinueOnXoff = TRUE;  // XOFF continues Tx
PortDCB.fOutX = FALSE;             // No XON/XOFF out flow control
PortDCB.fInX = FALSE;              // No XON/XOFF in flow control
PortDCB.fErrorChar = FALSE;        // Disable error replacement
PortDCB.fNull = FALSE;             // Disable null stripping
PortDCB.fRtsControl = 0;//RTS_CONTROL_ENABLE;
// RTS flow control
PortDCB.fAbortOnError = FALSE;     // Do not abort reads/writes on
// error
PortDCB.ByteSize =  8;             // Number of bits/byte, 4-8
PortDCB.Parity = NOPARITY;         // 0-4=no,odd,even,mark,space
PortDCB.StopBits =  0;                          // 0,1,2 = 1, 1.5, 2

IsOk = SetCommState(hPort, &PortDCB);
if (!IsOk) return -4;

// ----------------------------------------
// WRITE AND READ DATA
// ----------------------------------------

// Use bytes ...
BYTE b55 = 0x55; //
BYTE bAA = 0xAA; //
BYTE bHi = 1;   // hight byte
BYTE bLo = 2;   // Low byte
BYTE bTl = b55 + bAA + bHi + bLo; // Total byte
// ... or struct
COMMDATA dani; // It's my struct, see code above
dani.bHi = 2;
dani.bLo = 3;
dani.BuildSumm();

//while (true)
//{
        //char k = _getch();
        //if (k == 'q') break;
        // Read data from COM port
        DWORD dwRead = 0; // there are read bytes from a port
        IsOk = ReadFile(hPort, &b55, 1, &dwRead, NULL);
        IsOk = ReadFile(hPort, &bAA, 1, &dwRead, NULL);
        IsOk = ReadFile(hPort, &bLo, 1, &dwRead, NULL);
```

```
            IsOk = ReadFile(hPort, &bHi, 1, &dwRead, NULL);
            IsOk = ReadFile(hPort, &bTl, 1, &dwRead, NULL);
            // І ще можна так (розкоментувати якщо потрібно)
            // IsOk = ReadFile(hPort, &dani, sizeof(COMMDATA), &dwRead, NULL);

            Sleep(500);

            // begin process data
            // ...
            // -----------------
            // PROCEESS DATA
            // -----------------
            // Example:
            //  BYTE bTempTl = b55 + bAA + bLo + bHi;
            //  if (bTempTl != bTl) cout << " Error check control summ" << endl;
            // or
            //  if (!dani.IsSummCorrect())
            //  {
            //     Process error...
            //  }
            // -----------------
            // ...
            // end process data

            printf("Hight byte: %d\n", bHi /*dani.bHi*/);
            printf("Low bite: %d\n", bLo /*dani.bLo*/);

            // Write data to a COM port
            DWORD dwWritten = 0; // There are written bytes to a port
            IsOk = WriteFile(hPort, &b55, 1, &dwWritten, NULL);
            IsOk = WriteFile(hPort, &bAA, 1, &dwWritten, NULL);
            IsOk = WriteFile(hPort, &bHi, 1, &dwWritten, NULL);
            IsOk = WriteFile(hPort, &bLo, 1, &dwWritten, NULL);
            IsOk = WriteFile(hPort, &bTl, 1, &dwWritten, NULL);
            // І ще можна так (розкоментувати якщо потрібно)
            // IsOk = WriteFile(hPort, &dani, sizeof(COMMDATA), &dwWritten,
NULL);
        //}

        // Destroy port
        CloseHandle(hPort);
        return 0;
}
```

10. Програми приймання даних через COM у режимі переривання

Програма приймання даних через COM2 у режимі переривання зі швидкістю 9600 біт/с байтами по 8 біт, без контролю парності і виведення

їх на екран (передача виконується аналогічно прикладу 1).

```
{$M $800, 0, 0}
uses dos,crt;
const Maxbuf=100;
type
  TSaveComStatus = record
                     Upr,
                     MaskInt9,
                     MaskIntC,
                     MaskInt21 : Byte;
                     Speed     : Word;
                   end;

  TSaveIntrStatus = record
                     MaskInt21 : Byte;
                    end;

  TComDat = record
          Speed  : Word;
          reg    : Byte;
         end;
var
  Init         : TComDat;
  SaveComStatus  : TSaveComStatus;
  SaveIntrStatus : TSaveIntrStatus;
  SaveOldInt     : Procedure;
  Buf          : array [1..Maxbuf] of byte;
  Begbuf, Endbuf : byte;

procedure outcom(p:Byte);    { Вивéдення в COM1 }
  begin
   repeat until (port[$3FD] and $40) <> 0; {Очікування завершення
                                            попередньої передачі}
   port[$3F8]:= p;
  end;

{$F+}
{ Переривання COM'a: }
Procedure ReadComInt(flags,cs,ip,ax,bx,cx,dx,si,di,ds,es,bp:word;
                var b:byte); interrupt;
 Label 1;
 Function ChekInt:boolean;
```

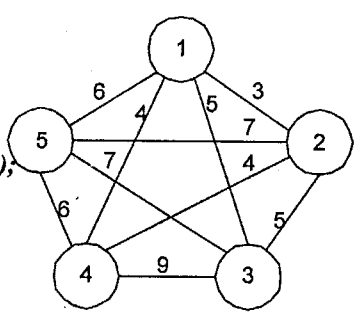
```
  begin
    ChekInt:=(port[$3FA] and 4) <> 0
  end;
 begin
   asm cli end;
   if ChekInt then
     begin
      B:=port[$3F8];
      {Занесення байта до кільцевого буфера}
      if (Endbuf=0) and (Begbuf=0) then begin inc(Endbuf); inc(Begbuf); goto 1;
                                                    end;
      if (Endbuf>=Begbuf) and (Endbuf<Maxbuf) then begin inc(Endbuf); goto
                                                          1; end;
      if (Begbuf>1) and (Endbuf=Maxbuf) then begin Endbuf:=1; goto 1; end;
      if (Begbuf>Endbuf+1) then begin inc(Endbuf); goto 1; end;
      1: Buf[Endbuf]:=B;
      outcom(B)   {эхо}
     end;
   asm sti end;
   port[$20]:=$20;
 end;
{$F-}

Procedure InitCom1( com: tcomDat );   { Ініціалізація COM'a }
  var in_b : byte;
     mask : byte;
  begin
   asm cli end;
   SaveIntrStatus.MaskInt21:= port[$21];  {збереження режиму контролера
                                           переривань}
   with SaveComStatus do
    begin
     Upr     := port[$3FB];   {збереження режиму прийому}
     MaskInt9 := port[$3F9];   {збереження дозволених переривань}
     MaskIntC := port[$3FC];   {збереження режимів модема}
     port[$3FB] := $80;
     Speed   := port[$3F9]*256+port[$3F8]; {збереження швидкості}
    end;

   port[$3FB] := $80;
   port[$3F8] := lo(com.Speed);
   port[$3F9] := hi(com.Speed);
   port[$3FB] := com.reg;
```

```
  getintvec($0C, @saveOldInt);        { COM2 - $0B }
  setintvec($0C, addr(ReadComInt));
  in_b:=port[$3F8];
  port[$3F9] := $01; {дозвіл переривання при наявності даних}
  port[$3FC] := $08;  {Встановлення в порту $3FC бітів 3,1,0}

  mask    := $10;    { COM2 - $08 }
  port[$21] := port[$21] and (not mask); { Дозвіл переривання контролера }
  in_b:=port[$3F8];
  asm sti end;
 end;

procedure closecom;    {Відновлення режимів при виході}
 begin
  setintvec($0C, addr(saveOldInt));
  port[$21] := SaveIntrStatus.MaskInt21;
  with SaveComStatus do
   begin
    port[$3F9] := MaskInt9;
    port[$3FC] := MaskIntC;

    port[$3FB] := $80;
    port[$3F9] := Hi(Speed);
    port[$3F8] := Lo(Speed);
    port[$3FB] := Upr;
   end;
 end;

Procedure InitRegCom(var Init:TcomDat);
 Begin
  Init.Reg    := $18 or $03;
  Init.Speed  := Round(115200/19200);
 End;

begin
 InitRegCom(Init);
 InitCom1(Init);
 Begbuf:=0;
 Endbuf:=0;
 repeat
  if Endbuf>0 then
```

```
  begin      { Виведення байта з буфера на екран і зсув початку буфера на
               одну позицію}
   gotoxy(1,1);
   write(chr(buf[Begbuf]));
    if Begbuf<>0 then write(buf[Begbuf]);
    if (Begbuf>Endbuf) and (Begbuf<Maxbuf) then begin inc(Begbuf); goto 2;
                                                        end;
    if (Begbuf>Endbuf) and (Begbuf=Maxbuf) then begin Begbuf:=1; goto 2;
end;
    if (Endbuf=Begbuf) then begin Endbuf:=0; Begbuf:=0; goto 2; end;
    if (Begbuf<Endbuf) then begin inc(Begbuf); goto 2; end;
    2:;
  end
 until keypressed;
 closecom
end.
```

11. Оптимізація лінійної мережі

```
Const
 N=5;
 Dist:array[1..N,1..N] of integer =((0,3,5,4,6),
                                    (3,0,5,4,7),
                                    (5,5,0,9,7),
                                    (4,4,9,0,6),
                                    (6,7,7,6,0));
Type
 IntSet = set of 1..N;
Var
 A,Amin:array [1..N] of integer;
 Lmin,j:integer;
Procedure Commi(S:IntSet; K:integer);
 Var
 L,i:integer;
Begin
 if S=[] then
  begin
   L:=0;
   for i:=1 to N-1 do L:=L+Dist[A[i],A[i+1]];
   if L<Lmin then
    begin
     Lmin:=L;
     Amin:=A
```



```
      end
    end
   else
    for i:=1 to N do
      if i in S then
        begin
          A[K]:=i;
          Commi(S-[i],K+1)
        end
End;
BEGIN
 Lmin:=Maxint;
 Commi([1..N],1);
 for j:=1 to N do writeln(Amin[j])
END.
```

12. Оптимізація ієрархічної мережі

Реалізація алгоритму Пріма-Краскала з використанням множини

```
Const
 N=6;
 B=1000;
 D:array [1..N,1..N] of real=((0, 1, 3, B, B, B),
                              (1, 0, B, B, B, B),
                              (1, B, 0, 1, 2, B),
                              (B, 1, 1, 0, B, 1),
                              (B, B, 5, B, 0, B),
                              (B, B, B, 1, 1, 0));
Var
 COL:set of 1..N;
 i,j,i0,j0,k:integer;
 L,Dmin:real;
 REB:array[1..N-1,1..2]of integer;

Begin
 COL:=[1..N];
  Dmin:=B;
  for i0:=1 to N-1 do
   for j0:=i0+1 to N do
    if (D[i0,j0]<Dmin) then
      begin
       Dmin:=D[i0,j0];
```

```
      i:=i0; j:=j0;
     end;
  REB[1,1]:=i; REB[1,2]:=j;
  COL:=COL-[i,j];
  L:=Dmin;
  k:=2;
 while k<=N-1 do
  begin
  Dmin:=B;
  for i0:=1 to N-1 do
   for j0:=i0+1 to N do
    if (D[i0,j0]<Dmin) and ((i0 in COL) xor (j0 in COL)) then
     begin
      Dmin:=D[i0,j0];
      i:=i0; j:=j0;
     end;
  L:=L+Dmin;
  REB[k,1]:=i; REB[k,2]:=j;
  if i in COL then COL:=COL-[i] else COL:=COL-[j];
  inc(k);
 end;
for k:=1 to N-1 do writeln(REB[k,1],' - ',REB[k,2]);
writeln('Lmin=',L:9:1)
End.
```

13. Оптимізація кільцевої мережі

(Задача комівояжера)

```
Const
 N=5;
 Dist:array[1..N,1..N] of integer =((0,3,5,4,6),
                                    (3,0,5,4,7),
                                    (5,5,0,9,7),
                                    (4,4,9,0,6),
                                    (6,7,7,6,0));
Type
 IntSet = set of 1..N;
Var
 A,Amin:array [1..N] of integer;
 L,Lmin,j:integer;
Procedure Commi(S:IntSet; K:integer);
 Var
  i:integer;
```

```
Begin
 if S=[] then
  begin
   L:=L+Dist[A[N],A[1]]
   if L<Lmin then
    begin
     Lmin:=L;
     Amin:=A
    end
  end
  else
   for i:=1 to N do
    if i in S then
     begin
      A[K]:=i;
      if K>1 then L:=L+Dist[A[K-1],A[K]];
      if L<Lmin then Commi(S-[i],K+1);
      if K>1 then L:=L-Dist[A[K-1],A[K]];
     end
End;
BEGIN
 Lmin:=Maxint;
 L:=0;
 Commi([1..N],1);
 for j:=1 to N do writeln(Amin[j])
END
```

Команди модема

ATA

Ця команда ініціює «стан відповіді». Після приблизно 2-секундної затримки модем починає надсилати в лінію сигнал відповіді. Якщо він не знаходить несучу за час, що визначається регістром S7, він «вішає слухавку» і повертається в холостий режим. Інакше модем «знімає слухавку» і входить в режим передавання даних (online).

Ця команда повинна бути останньою в командному рядку після неї.

ATD

За допомогою цієї команди модем може набирати телефонний номер (на звичайних комутованих лініях) або ініціювати виклик (на виділених лініях). Ця команда повинна бути останньою в рядку, оскільки модем не звертає уваги на команди, що з'явилися в командному рядку після неї.

Команда має безліч параметрів:

0-9 цифри, що використовуються при пульсовому наборі

ABCD#* додаткові символи, допустимі тільки при тоновому наборі

P перемикає модем на пульсовий метод набору номера (набір починається після паузи, тривалість якої задається в S6). Якщо цей метод використовується за замовчуванням, то до явної зміни підкомандою T, ATD еквівалентне ATDP.

T перемикає модем на тональний (DTMF) набір номера (набір починається після паузи, тривалість якої задається в S6).

, примушує модем зробити паузу, тривалість якої задається регістром S8 (стандартно – 2 сек).

W примушує модем чекати сигнал телефонної станції. Час очікування задається в регістрі S7 (стандартно – 30 сек). Якщо за цей час сигнал не буде почутий, повертає код результату NO DIALTONE.

@ модем чекає появи 5 сек тиші (відсутність сигналу). Час очікування задається в регістрі S7 (стандартно – 30 сек). Якщо за цей час вона не з'явиться – модем повертає код результату NO ANSWER.

Ще одна маловідома властивість модифікатора @ корисна, якщо поставити @ останнім символом в рядку набору – багатьом модемам він допомагає повернути код результат BUSY при появі сигналу «зайнято».

! модем повинен на короткий час (1/2 сек) відключитися від лінії, потім підключиться знову. За дією це рівнозначно короткочасному натисненню на важіль телефонного апарата.

; цей модифікатор поміщається тільки в кінці командного рядка і указує модему, що він повинен залишитися в командному режимі зразу ж після набору номера не відключаючись від лінії(«з піднятою трубкою»). Це може бути дуже корисним при наборі довгих номерів, що переповнюють буфер (40 символів).

R (Reverse) цей модифікатор ставиться тільки в кінці команди. Він дозволяє встановити зв'язок з модемом, який не уміє відповідати на дзвінки або з відключеною в даний момент автовідповіддю (ATS = 0). При отриманні несучої від віддаленого модему відбудеться перемикання телефонуючого модему в стан автовідповіді. Див. також команду ATR.

S ця підкоманда дозволяє набирати телефонний номер, збережений раніше командою AT&Z.

Окрім цих символів, при наборі номера можна використовувати роздільники - «(« »)» «пробіл» «-». І хоча роздільники ігноруються модемом, вони все ж таки займають місце в буфері команд.

Деякі модеми розуміють ще один модифікатор, «/» (слеш):

/ примушує модем витримати паузу в 125 мілісекунд (1/8 сек)

ATD

Параметри: 0,1

За замовчуваням: 0

Ця команда керує підключенням модему до лінії.

ATH0 відключається від лінії («повісити трубку»)

ATH1 підключається до лінії («зняти трубку»)

Аналогія з телефонною трубкою (присутня, до речі, в англійській назві команди) виникає через те, що за цими командами модем, відносно телефонної станції, робить саме такі дії.

ATL

Параметри: 0, 1, 2, 3

За замовчуваням: 2

Встановлює рівень гучності звуку вбудованого динаміка.

ATL0 низький рівень

ATL1 низький рівень

ATL2 середній рівень

ATL3 високий рівень

У деяких моделях модемів (як правило, внутрішніх) ці команди хоча і приймаються модемом (в цілях сумісності), але ні на що не впливають. У таких випадках передбачений ручний регулятор рівня гучності.

ATM

Параметри: 0, 1, 2, 3

За замовчуваням: 1

Керування роботою вбудованого в модем динаміка.

ATM0 динамік вимкнений

ATM1 динамік увімкнений тільки до виявлення несучою

ATM2 динамік увімкнений постійно

ATM3 як ATM1, але динамік вимкнений на час набору номера

Стандартне значення, мабуть, оптимальне психологічно. Але в деяких випадках кращий ATM2 (якщо це, звичайно, не заважає оточуючим), і ви можете чути пропадання несучої, а при деякому досвіді – навіть оцінювати якість лінії.

Модеми Inlel SatisFAXtion:

ATM4 як ATM1, але динамік увімкнений і під час retrain'ов

ATM5 як ATM3, але динамік увімкнений і під час retrain'ов

ATN

Параметри: 0, 1

За замовчуваням: 1

Ця команда (сумісно з регістром S37) дозволяє керувати методом вибору швидкості для з'єднання аж до її фіксації.

ATN0 вимагає, щоб швидкість з'єднання встановлювалася відповідно до регістра S37

ATN1 дозволяє handshake (процедуру ініціації зв'язку) на будь-якій швидкості, підтримуваній обома модемами. Вмикає automode detection.

Команда ATB ігнорується – модем намагатиметься встановлювати зв'язок тільки по CCITT.

ATO

Параметри: 0, 1 (рідше: 0-3)

За замовчуваням: 0

Ця команда служить для повернення з командного режиму в стані online. Припустимо, ви вийшли під час сеансу зв'язку в командний режим за допомогою «эскейп-ланцюжка» (тобто, секунда паузи, символи +++ і знову секунда паузи), тоді для повернення назад потрібно видати модему команду ATO.

Інший приклад – можна перемкнутися на модеми, не виходячи з голосового дзвінка. Для цього на одній із сторін потрібно видати модему команду ATO, на іншій – ATA.

ATO0 перейти в режим online

ATO1 перейти в online з ініціацією процедури настроювання на поточні параметри лінії зв'язку («retrain»)

Команда ATO1, окрім повернення в режим даних, примушує модем почати процедуру настроювання компенсатора (equalizer retrain). Це допомагає поліпшити якість зв'язку, якщо поява помилок пов'язана із зміною характеристик лінії.

У деяких модемах (Inlel SatisFAXtion, наприклад) реалізовані додаткові можливості:

ATO2 перейти в online з настроюванням на меншу швидкість передачі

ATO3 перейти в online з настроюванням на велику швидкість

Зауваження: модем може опинитися в командному режимі при встановленому з'єднанні з двох причин:

- після эскейп-ланцюжка;
- після попадання сигналу DTR, якщо модем був конфігурований командою AT&DI.

ATS

Ця команда дозволяє подивитися і змінити значення будь-якого S-регістра модема.

ATSr? – показати значення (десяткове!) регістра номер r

ATSr = n – переслати в регістр r число n

AT&L

Параметри: 0, 1

За замовчуваням: 0

Вибір типу лінії зв'язку:

AT&L0 робота на комутованих лініях зв'язку

AT&L1 робота на виділених двопровідних лініях зв'язку

Навчальне видання

Ольга Віталіївна Глонь
Володимир Михайлович Дубовой
Юрій Ігоревич Мітюшкін

Комп'ютеризовані системи керування

Навчальний посібник

Оригінал-макет підготовлено В.М. Дубовим

Редактор О.Д. Скалоцька

Навчально-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку 8.02.06р. Гарнітура Times New Roman
Формат 29,7х42 1/4 Папір офсетний
Друк різографічний Ум. друк. арк. 6.6
Тираж 75 прим.
Зам. № 2006-031

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ