В. В. Ковальчук

# Лабораторний практикум

# (SciLab)

**УДК 65.23-80+658.56 (078.5)**
**ББК 681.0.03+30.607я**

Рекомендовано Міністерством освіти і науки Україні як навчальний посібник для студентів вісших учебніх закладів

---

**Рецензенти**

**В.А.Дроздов**, доктор фізико-математичних наук, професор, Кременецкий обласний гуманітарно-педагогічний інститут
**В.Л.Костенко**, доктор технічних наук, професор, Одеський національний політехнічний університет

---

**Ковальчук В.В. Лабораторний практикум (SciLab).** Рукопис. – Одеса: ОККТ ОДЕКУ, 2013. – 164 с.

Запропоновано лабораторний практикум для освоєння пакету **Skilab** — системи комп'ютерної математики, яка призначена для виконання інженерних і наукових обчислень.

# Зміст

# Вступ

Пропонований навчальний посібник «Інформаційно-вимірювальні системи. Лабораторний практикум з SciLab» дозволяє практично закріпити отримані теоретичні знання з дисципліни інформаційно-вимірювальні системи, яку освоюють майбутні інженери-метрологи. Вивчення його співпадає з початком роботи студентів над виконанням курсових і выпусковых робіт по вибраній спеціальності, періодом активної участі студентів в науково-дослідній роботі кафедр.

Матеріали допомоги допоможуть студентам правильно орієнтуватися в складній структурі взаємозв'язків між окремими ланками процесу накопичення, збереження і обробки інформації, а також при побудові конкретних моделей, використовуваних в інформаційно-вимірювальних технологіях.

Досвід показує, що наукова робота студентів істотно підвищує мотивацію до вивчення загальних і спеціальних дисциплін за фахом, сприяє формуванню теоретичних і практичних навиків, необхідних фахівцеві-дослідникові, розширюючи науковий кругозір і здібності до проведення методологічного аналізу і критичного розуміння досягнень сучасної науки.

Основні завдання навчального посібника стосуються важливих питань математичного моделювання інформаційно-вимірювальних систем. Детально представлений пакет лабораторного практикуму, який буде вельми корисний фахівцям в області інформаційно-вимірювальних технологій. Матеріал поданий на тлі сучасного аналізу методології обробки результатів наукових досліджень в області інформаційно-вимірювальних систем.

Навчальний посібник може бути допоміжним довідником не тільки для студентів вищих учбових закладів по напряму технічного регулювання, але і молодим ученим – магістрантам, аспірантам.

# ТЕМА 1

# SKILAB І ЙОГО ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ

**Skilab** — це система комп'ютерної математики, яка призначена для виконання інженерних і наукових обчислень, таких як:

- вирішення нелінійних рівнянь і систем;
- вирішення завдань лінійної алгебри;
- вирішення завдань оптимізації;
- диференціювання і інтеграція;
- обробка експериментальних даних (інтерполяція і апроксимація, метод найменших квадратів);
- вирішення звичайних диференціальних рівнянь і систем.

Крім того, Skilab надає широкі можливості по створенню і редагуванню різних видів графіків і поверхонь.

Не дивлячись на те, що система Skilab містить достатню кількість вбудованих команд, операторів і функцій, відмінна її риса — це гнучкість. Користувач може створити будь-яку нову команду або функцію, а потім використовувати її нарівні з вбудованими. До того ж, система має достатньо могутню власну мову програмування високого рівня, що говорить про можливість вирішення нових завдань.

Пакет Scilab є вільно поширюваною (разом з початковими кодами) системою комп'ютерної математики. До недавнього часу він розроблявся дослідницькими інститутами INRIA і ENPC (обидва знаходяться у Франції), а з травня 2003 р. підтримку продукту узяв на себе спеціально створений для цієї мети Scilab Consortium, з Web-узла якого можна завантажити останню версію програми і повний комплект документації (на момент підготовки матеріалу була доступна версія 2.7.2). Scilab випускається для операційних систем Windows (будь-яка 32-розрядна версія), найбільш популярних Unix/Linux і не потребує великих системних ресурсів: інсталяційний модуль має розмір до 20 MB, а для установки потрібний небагато чим більше 40 MB.

Пакет не випадково має назву, співзвучну з Matlab -- одній з найбільш могутніх комерційних СКМ. У обох застосувань немало загального -- від інтерфейсу і принципу взаємодії з користувачем через командний рядок до

синтаксису мови. Таким чином, Scilab можна розглядати як полегшений варіант Matlab, який, втім, зберігає основні можливості останнього.

Scilab є типовим командним інтерпретатором і структурно складається з інтерпретуючої системи, що приймає команди користувача і що повертає результати, і двох бібліотек: власних функцій і додаткових -- на мовах З і Fortran.

### 1.1. Основні функціональні можливості

Хоча Scilab є безкоштовним продуктом, його обчислювальні можливості, забезпечені приблизно тысячью вбудованих функцій, цілком відповідають СКМ професійного рівня. Навряд чи має сенс перераховувати навіть головні з них, тому зупинимося тільки на ключових моментах.

Пакет підтримує основні елементарні і безліч спеціальних функцій, вживаних в математиці, зокрема - для різного виду згладжувань і апроксимацій, еліптичні інтеграли, функції Бесселя. Scilab містить також могутній набір засобів для роботи з поліномами - як звичайними, так і матричними. Наприклад, є оператори для створення полінома із заданим корінням або коефіцієнтами, обчислення коріння полінома (до сотого ступеня), ділення двох поліномів, знаходження найбільшого загального дільника і найменшого загального кратного декількох поліномів і виконання десятків інших важливих операцій над поліномами.

Особливістю пакету є те, що він призначений майже виключно для реалізації чисельних методів і за умовчанням оперує з будь-якими значеннями як з числами з плаваючою крапкою. Якщо ми введемо який-небудь цілочисельний вираз, скажемо суму 2 + 3, Scilab поверне результат у вигляді числа з плаваючою крапкою. Для того, щоб система сприймала подібні вирази як цілі числа, необхідно використовувати спеціальні команди. Також відзначимо, що, хоча додаток підтримує окремі символьно-аналітичні операції, їх круг украй обмежений. Наприклад, можна знайти символьну суму і твір матриць, виконати деякі аналітичні операції над многочленами алгебри, але відсутня підтримка символьних операцій аналізу, загальних перетворень алгебри і інших аналітичних обчислень, реалізованих в багатьох інших СКМ.

Як і Matlab, Scilab має розвинені інструменти для створення і маніпулювання масивами (векторами, матрицями і ін.), підтримуються і інші складні структури (списки), об'єднуючі послідовності даних довільного типу.

Для операцій з матрицями в Scilab використовується переважно природна математична нотація, відповідно до якої сума матриць A і B записується як A + У, твір -- як A * У, транспонування як Аї т.д. Такий спосіб робить спілкування з системою зручним і природним. Є оператори для генерування матриць спеціального вигляду -- що складаються з одиниць або нулів, заповнених елементами з випадкової вибірки, розподіленої по рівномірному або нормальному закону, і ін.

Scilab дозволяє також обмінюватися даними з іншими застосуваннями, хоча в цілому можливості імпорту/експорту системи не можна назвати особливо багатими. Підтримуються формати документів Matlab і Maple, структурований текст і TEX.

Функції системи, що відносяться до деяких прикладних областей математики і техніки, зібрані в додаткові пакети розширень (так звані toolboxes). Одні з них застосовуються достатньо широко (як, скажімо, методи чисельного вирішення краєвих завдань для систем диференціальних рівнянь, лінійне і квадратичне програмування), інші мають вузьку специфічну спрямованість. З останніх назвемо пакети для цифрової обробки сигналів, аналізу динамічних систем, оптимізації із спеціальними обмеженнями.

### 1.2. Програмування в Scilab

Кожен користувач математичних пакетів знає, що, наскільки б широкі не були їх можливості, при вирішенні реальних завдань доводиться створювати власні алгоритми і процедури. Для цих цілей Scilab має в своєму розпорядженні могутню вбудовану мову, широкий набір конструкцій, що володіє, для організацій циклів, умовних переходів, операцій введення/виводу. Природно, що за допомогою цієї мови можна дістати доступ до всіх внутрішніх можливостей додатку.

У даній СКМ реалізована виключно гнучка концепція процедурного програмування -- процедурам (званим в Scilab функціями) дозволяється повертати цілі масиви результатів довільного типу, а список аргументів навіть не зобов'язаний бути фіксованим. Програмувати можна безпосередньо в робочому середовищі вводячи необхідних операторів прямо в командний рядок, але зручнішим способом є застосування зовнішніх файлів (зазвичай з розширеннями *.sci або *.sce), які завантажуються в додаток спеціальною командою або за допомогою меню. Такий файл створюється в

будь-якому текстовому редакторові, але краще використовувати вбудований -- Scilab Pad, що дозволяє негайно завантажити код в систему. Процедури також часто об'єднуються в бібліотеки і зберігаються в бінарному форматі, що компілює (розширення *.bin) -- саме так зберігаються стандартні функції цієї СКМ.

Scilab має повноцінні засоби відладки, які дають можливість встановлювати контрольні крапки, отримувати інформацію змінних по ходу виконання процедури і так далі Проте потрібно мати на увазі, що відладчик Scilab не інтерактивний ті ж контрольні крапки необхідно розмістити за допомогою спеціальних операторів в тілі програми, а управління процесом відбувається через командний рядок.

Важливою особливістю програмних можливостей Scilab є підтримка взаємодії з кодом на мовах З і Fortran. Наприклад, можна чисельно вирішити краєву задачу для диференціальних рівнянь, деякі компоненти яких обчислюються в зовнішніх модулях. Цього добиваються декількома способами, але найчастіше використовують динамічний зв'язок Scilab з програмами, що компілюють.

Спеціальна, така, що поставляється з цією СКМ функція maple2scilab, дозволяє перетворити більшість об'єктів Maple в коди на мові Scilab для подальшого застосування. Підтримується також програмний інтерфейс між Scilab і Tk-Tcl мовою програмування, що інтерпретується, нерідко використовуваною для науково-технічних розрахунків.

### 1.3. Графічні функції

Без перебільшення можна сказати, що графіка Scilab, не дивлячись на безкоштовний статус пакету, виконана на цілком професійному рівні. Основою її реалізації є концепція пристрою, в який виводиться потік, що створюється графічними функціями -- це може бути екран або зовнішні файли у форматах PostScript, Xfig, GIF або PMM. Для управління виводом є близько десятка команд, що ініціалізували пристрій, здійснюють вивід, закривають сесію і так далі Якщо як графічний пристрій використовується вікно Scilab, то багато хто з цих команд доступний прямо в меню.

Scilab уміє малювати графіки функцій на площині (криві) або в тривимірному просторі (поверхні), основні геометричні шаблони (прямокутники, еліпси), будувати деякі спеціальні діаграми (гістограми,

графіки, вживані в теорії автоматичного контролю, і т. д.) і багато що інше. Передбачена широка настройка властивостей, допускається визначати кольори, заливки, точку огляду (у тривимірному випадку), відображати сітки і управляти десятками інших характеристик. Графіки можна виводити в одному вікні, зберігати в зовнішніх файлах, наносити на них написи пояснень, задаючи колір, розмір і гарнітуру шрифту.

### 1.4. Середовище Skilab

Після запуску Skilab на екрані з'явитися основне вікно додатку. Вікно містить меню, панель інструментів і робочу область. Ознакою того, що система готова до виконання команди, є наявність знаку запрошення -->, після якого розташований активний (миготливий) курсор. Робочу область із знаком запрошення зазвичай називають командним рядком. Введення команд в Skilab здійснюється з клавіатури. Натиснення клавіші Enter примушує систему виконати команду і вивести результат .

Зрозуміло, що всі виконувані команди не можуть одночасно знаходитися у полі зору користувача. Тому, проглянути ту інформацію, яка покинула видиму частину вікна, можна, якщо скористатися стандартними засобами перегляду, наприклад смугами прокрутки або клавішами переміщення курсора Page up, Page down.

Клавіші «Стрільця вгору» і «Стрілка вниз» також управляють курсором, проте в Skilab вони мають інше призначення. Ці клавіші дозволяють повернути в командний рядок раніше введені команди або іншу вхідну інформацію, оскільки вся ця інформація зберігається в спеціальній області пам'яті. Так, якщо в порожньому активному командному рядку натиснути клавішу те з'явиться остання команда, що вводиться, повторне натиснення викличе передостанню і так далі. Клавіша виводить команди в зворотному порядку. Таким чином, можна сказати, що вся інформація в робочій області знаходиться або в зоні перегляду або в зоні редагування.

Важливо знати, що в зоні перегляду не можна нічого виправити або ввести. Єдина допустима операція, окрім перегляду, це виділення інформації за допомогою миші і копіювання її в буфер обміну, наприклад, для подальшого приміщення в командний рядок.

Зона редагування — це фактично командний рядок. У ній діють елементарні прийоми редагування: > переміщення курсора управо на один

символ; < переміщення курсора вліво на один символ; Ноті — перемещение курсора в початок рядка; End — перемещение курсора в кінець рядка; de1 — видалення символу після курсора; Backspace — видалення символу перед курсором.

Крім того, існують особливості введення команд. Якщо команда закінчується крапкою з комою «;», то результат її дії не відображається в командному рядку. Інакше, за відсутності знаку «;», результат дії команди відразу ж виводиться в робочу область .

Документ, що відображає роботу користувача з системою Skilab, що містить рядки введення, виводу і повідомлення про помилки, прийнято називати сесією. Значення всіх змінних, обчислені протягом поточної сесії, зберігаються в спеціально зарезервованій області пам'яті, званої робочим простором системи. За бажання визначення всіх змінних і функцій, що входять в поточну сесію можна зберегти у вигляді файлу, саму сесію зберегти не можна.

# ТЕМА 2

# ОСНОВНІ КОМАНДИ ГОЛОВНОГО МЕНЮ SKILAB

Головне меню системи містить команди, призначені для роботи з файлами, настройки середовища, редагування команд поточної сесії і отримання довідкової інформації. Крім того, за допомогою головного меню можна створювати, редагувати, виконувати відладку і запускати на виконання так звані файли-сценарії Skilab, а також працювати з графічними додаткам пакету.

## 2.1. Робота з файлами

Пункт меню File  призначений для роботи з файлами. Розглянемо призначення представлених в нім команд:

New Scilab — відкриває нове вікно Skilab, фактично пакет запускається повторно;

Exec. — запуск на виконання створеної раніше Skilab-программы (файли з розширенням sce або sci );

Open — відкриває вікно для завантаження створеного раніше файлу, малюнка або моделі;

Load — відкриває вікно для завантаження файлів, інформація в яких зберігається у вигляді машинних код; при їх відкритті в пам'ять комп'ютера завантажуються визначені раніше змінні і функції;

Save — збереження всіх визначених в даній сесії змінних і функцій у вигляді файлу з розширенням saw  або bin;

Change Directory — зміна поточного каталога, виводить вікно настройки шляхів файлової системи;

Get change directory — виводить в командний рядок ім'я поточного каталога;

Print setup... — выводит вікно настройки параметрів друку;

Print — друк поточної сесії;

Exit— вихід з системи Skilab.

Редагування команд поточної сесії

Редагування команд поточної сесії

Пункт меню Edit містить наступні команди:

Select All — виділення всіх команд поточній сесії;

Copy — копіювання виділеного об'єкту в буфер;

Paste — вставка об'єкту з буфера;

Empty Clipboard — очищення буфера обміну;

History — група команд, призначених для редагування командного рядка.

***Настройка середовища***

Команди настройки середовища пакету представлені в меню Preferences:

Language — пропонує вибрати із списку мову інтерфейсу (англійський, французький) ;

Colors — дозволяє встановити колір шрифту (Text), колір фону (Background) або кольори, прийняті за умовчанням (Default System Colors);

Toolbar (F3) — выводит або видаляє панель інструментів;

Files Association — пропонує встановити типи підтримуваних файлів;

Choose Font — виконує настройки шрифту (гарнітура, зображення, розмір);

Clear History — очищає робочий простір;

Clear Command Window (F2) — очищає робоче вікно;

Consol (F12) — активізує консольне застосування.

Відзначимо, що редактор SciPad має можливість роботи з безліччю вікон (пункт меню Windows), володіє прийнятими для текстових редакторів прийомами редагування (пункт меню Edit) і пошуку (пункт меню Search). Крім того, можна виконати настройку середовища редактора SciPad (пункт меню Options), викликати довідкову інформацію (пункт меню Help) і здійснити відладку програми, набраної в редакторові (пункт меню Debug).

Вийти з режиму редагування можна, просто закривши вікно SciPad або виконавши команду File - Exit.

### 2.2. Основи роботи в SKILAB

Текстовий коментар в Scilab — це рядок, що починається з символів //. Використовувати текстові коментарі можна як в робочій області, так і в тексті файлу-сценарію. Рядок після символів // не сприймається як команда, і натиснення клавіші Enter приводить до активізації наступного командного рядка.

Для виконання простих арифметичних операцій в Scilab застосовують наступних операторів: + складання, - віднімання * множення / ділення зліва направо \ ділення справа наліво ~ піднесення до ступеня.

Обчислити значення арифметичного виразу можна, якщо ввести його в командний рядок і натиснути клавішу Enter.

Якщо обчислюваний вираз дуже довгий, то перед натисненням клавіші Enter слід набрати три або більш за крапки. Це означатиме продовження командного рядка. Якщо символ крапки з комою «;» вказаний в кінці виразу, то результат обчислень не виводиться, а активізується наступний командний рядок.

У робочій області Scilab можна визначати змінні, а потім використовувати їх у виразах. Будь-яка змінна до використання у формулах і виразах повинна бути визначена. Для визначення змінної необхідно набрати ім'я змінної, символ «=» і значення змінної. Тут знак рівності — це оператор привласнення,, дія якого не відрізняється від аналогічних операторів мов програмування. Тобто, якщо в загальному вигляді оператора привласнення записати як

имя_переменной = значение_выражения

то в змінну, ім'я якої вказане зліва, буде записано значення виразу, вказаного справа.

Ім'я змінної не повинне співпадати з іменами вбудованих процедур, функцій і вбудованих змінних системи і може містити до 24 символів. Система розрізняє великі і малі букви в іменах змінних. Тобто ABC, abc, Abc, aBc — це імена разных змінних. Вираз в правій частині оператора привласнення може бути числом, арифметичним виразом, рядком символів або символьним виразом. Якщо мова йде про символьній або строковій змінній, то вираз в правій частині оператора привласнення слід брати в одинарні лапки.

Якщо символ «;» в кінці виразу відсутній, то як результат виводиться ім'я змінної і її значення. Наявність символу «;» передає управління

наступному командному рядку. Якщо команда не містить знаку привласнення, то за умовчанням обчислене значення привласнюється спеціальній системній змінній ans. Причому набутого значення можна використовувати в подальших обчисленнях, але важливо пам'ятати, що значення ans змінюється після кожного виклику команди без оператора присваиванияРезультат останньої операції без знаку привласнення зберігається в змінній ans. Інші системні змінні в Scilab починаються з символу %:

%i — мнимая одиниця (V — 1);

%pi — число п = 3.141592653589793;

%e — число e = 2.7182818;

%inf — машинный символ нескінченності (ті);

%NaN — неопределенный результат (0/0,те/те і т. п.);

%eps — условный нуль

%eps=2.220E-16.

## 2.3. Елементарні математичні функції

Пакет Scilab забезпечений достатньою кількістю всіляких вбудованих функцій, знайомство з якими відбуватиметься в наступних розділах. Тут приведемо тільки елементарні математичні функції, використовувані найчастіше (табл. 2.1).

Таблиця 2.1.

Елементарні математичні функції

| Функція | Опис функції |
|---|---|
| Тригонометричні | |
| sin(x) | синус числа x |
| cos(x) | косинус числа x |
| tan(x) | тангенс числа x |
| cotg(x) | котангенс числа x |
| asin(x) | арксинус числа x |
| acos(x) | арккосинус числа x |
| atan(x) | арктангенс числа x |
| Експоненціальні | |
| exp(x) | Експонента числа x |
| log(x) | Натуральний логарифм числа x |
| Інші | |
| sqrt(x) | корінь квадратний з числа x |
| abs(x) | модуль числа x |
| log10(x) | десятковий логарифм від числа x |
| log2(x) | логарифм по підставі два від числа x |

## 2.4. Функції, визначені користувачем

У першому розділі ми вже згадували про файли-сценарії і навіть створювали невелику програму, яка вирішувала конкретне квадратне рівняння. Але в цю програму неможливо було передати вхідні параметри, тобто це був звичайний список команд, що сприймається системою як єдиний оператор.

Функція, як правило, призначена для неодноразового використання, вона має вхідні параметри і не виконується без їх попереднього завдання. Розглянемо декілька способів створення функцій в Skilab.

Перший спосіб — це застосування оператора deff, який в загальному вигляді можна записати так:

deff('[имя1...,имяN] = имя_функции(переменная_1...,переменная_И)', 'имя1=выражение1;...;имяЫ=в^1ражениеЫ')

де имя1...,имяN — список вихідних параметрів, тобто змінних, яким буде привласнений кінцевий результат обчислень

имя_функции — ім'я з яким ця функція викликатиметься

переменная_1...,переменная_И — входные параметри.

Другий спосіб створення функції це застосування конструкції вигляду: function[имя1...,имяN]=имя_функции(переменная_1...,переменная_М) тіло функції endfunction

де имя1...,имяN — список вихідних параметрів, тобто змінних, яким буде привласнений кінцевий результат обчислень; имя_функции — ім'я з яким ця функція викликатиметься, переменная_1 ...,переменная_М — вхідні параметри.

Всі імена змінних усередині функції, а також імена із списку вхідних і вихідних параметрів сприймаються системою як локальні, тобто вважаються визначеними тільки усередині функції.

Взагалі кажучи, функції в Scilab грають роль підпрограм. Тому доцільно набирати їх тексти в редакторові і зберігати у вигляді окремих файлів. Причому ім'я файлу повинне обов'язково співпадати з ім'ям функції. Розширення файлам-функціям зазвичай привласнюють sci або sce .

Звернення до функції здійснюється так само, як і до будь-якої іншої вбудованої функції системи, тобто з командного рядка. Проте функції, що зберігаються в окремих файлах, повинні бути заздалегідь завантажені в систему, наприклад, за допомогою оператора exec(имя_файла) або командою головного меню File - Exec..., що, загалом, одне і те ж.

Для роботи з безліччю даних зручно використовувати масиви. Наприклад, можна створити масив для зберігання числових або символьних даних. В цьому випадку замість створення змінної для зберігання кожного даного досить створити один масив, де кожному елементу буде привласнений порядковий номер.

Таким чином, масив — множинний тип даних, що складається з фіксованого числа елементів. Як і будь-якій іншій змінній, масиву повинне бути привласнене ім'я.

Змінну, що є просто списком даних, називають одновимірним масивом, або вектором. Для доступу до даним, що зберігаються в певному елементі масиву, необхідно вказати ім'я масиву і порядковий номер цього елементу, званий індексом.

Якщо виникає необхідність зберігання даних у вигляді таблиць, у форматі рядків і стовпців, то необхідно використовувати двовимірні масиви (матриці). Для доступу до даним, що зберігаються в такому масиві, необхідно вказати ім'я масиву і два індекси: перший повинен відповідати номеру рядка, а другий — номеру стовпця, в яких зберігається необхідний елемент. Значення нижньої межі індексації в skilab рівне одиниці. Індекси можуть бути тільки цілими позитивними числами.

***Контрольні питання:***

*1. Перерахувати основні команди меню в середовищі Skilab*

*2 Перерахувати функції за допомогою яких вводяться змінні*

*3 Перерахувати основні функції за допомогою яких вводяться елементарні математичесские функції*

# ТЕМА 3
# ВВЕДЕННЯ І ФОРМУВАННЯ МАСИВОВ І МАТРИЦЬ

Задати одновимірний масив в Scilab можна таким чином:

name=Xn:dX:Xk

де name — ім'я змінної, в яку буде записаний сформований масив, Xn — значення першого елементу масиву, Xk — значення останнього елементу масиву, dX — крок, за допомогою якого формується кожен наступний елемент масиву, тобто значення другого елементу складе Xn+dX, третього Xn+ dX+dX і так далі до Xk.

Якщо параметр dX в конструкції відсутній, це означає, що за умовчанням він приймає значення, рівне одиниці, тобто кожен наступний елемент масиву рівний значенню попереднього плюс один:

name=Xn:Xk

Змінну, задану як масив, можна використовувати в арифметичних виразах і як аргумент математичних функцій. Результатом роботи таких операторів є масиви:

Введення елементів матриці також здійснюється в квадратних дужках, при цьому елементи рядка відділяються один від одного пропуском або комою, а рядки розділяються між собою крапкою з комою

name=[x11, x12 ..., x1n; x21, x22 ..., x2n; ...; xm1, xm2 ..., xmn;]

Звернутися до елементу матриці можна, вказавши після імені матриці, в круглих дужках через кому, номер рядка і номер стовпця на перетині яких елемент розташований:

name(индекс1, индекс2)

Далі приведений приклад завдання матриці і звернення до її елементів:

Лістинг 3.5. Приклад звернення до елементів матриці

--> A=[1 2 3;4 5 6;7 8 9] A =

--> A(1,2)~A(2,2)/A(3,3) ans = 3.5556

Важливу роль при роботі з матрицями грає знак двокрапки «:». Указуючи його замість індексу при зверненні до масиву, можна діставати доступ до груп його елементів

Дії над матрицями

Для роботи з матрицями і векторами в Skilab передбачені наступні операції:

+ — складання;

- — віднімання;

' — транспонирование ;

— матричне множення ;

*— множення на число;

^— піднесення до ступеня ;

\ — левое ділення;

/ — праве ділення; .

* — поелементне множення матриць;

.^—поэлементное піднесення до ступеня;

.\ — поелементне ліве ділення;

./— поелементне праве ділення.

Крім того, якщо до деякого заданого вектора або матриці застосувати математичну функцію, то результатом буде новий вектор або матриця тієї ж розмірності, але елементи будуть перетворені відповідно до заданої функції

Для роботи з матрицями і векторами в Scilab існують спеціальні функції. Розглянемо найчастіше використовувані з них. Функції визначення матриць:

matrix(A [,n,m]) — преобразует матрицю A в матрицю іншого розміру

ones(m,n) — создает матрицю одиниць з m рядків і n стовпців;

zeros(m,n) — создает нульову матрицю з m рядків і n стовпців;

eye(m,n) — формирует одиничну матрицю з m рядків і n стовпців;

rand(n1,n2...nn[,fl]) — формирует багатовимірну матрицю випадкових чисел. Необов'язковий параметр p — це символьна змінна, за допомогою якої можна задати тип розподілу випадкової величини ('uniform' — равномерное, 'normal' — гауссовское); rand(m,n) — формирует матрицю m на n випадкових чисел; rand(M) — формирует матрицю випадкових чисел, розмір якої співпадає з розміром матриці M; результат функції rand() — случайное скалярне число;

sparse([i1 j1;i2 j2;...;in jn],[n1,n2...,nn]) — формирует розріджену матрицю. Для створення матриці такого типу необхідно вказати індекси її ненульових элементов— [i1 j1,i2 j2...,in jn], і їх значення — [n1,n2...,nn]. Індекси одного елементу відділяються один від одного або пропуском, або комою, а пари індексів — відповідно крапкою з комою, значення елементів розділяються комами. При спробі проглянути матрицю подібного типу користувачеві буде надано повідомлення про її розмірність, а також значення ненульових елементів і їх місцеположення в матриці;

full(M) — виведення розрідженої матриці M у вигляді таблиці;

hypermat(D[,V]) — створення багатовимірної матриці з розмірністю, заданою вектором D і значеннями елементів, що зберігаються у векторі V (використання параметра V необов'язково);

diag(V[,k]) — повертає квадратну матрицю з елементами V на головній або на к-й діагоналі; функція diag(A[,k]), де A — раніше певна матриця, як результат видасть вектор-стовпець, що містить елементи головної або к-ой діагоналі матриці A;

cat(n, A, B [C ...]) — объединяет матриці A і B або всі вхідні матриці, при n=1 по рядках, при n=2 по стовпцях; те ж що [A; B] або[A, B];

tril(A[,k]) — формирует з матриці A нижню трикутну матрицю, починаючи з головної або з к-й діагоналі;

triu(A[,k]) — формирует з матриці A верхню трикутну матрицю, починаючи з головної або з к-й діагоналі;

sort(X) — виконує впорядковування масиву X; якщо X — матриця, сортування виконується по стовпцях;

size(V[,fl]) — определяет розмір масиву V; якщо V — двовимірний масив, то size(V,1) або size(V,'r') визначають число рядків матриці V, а size(V,2) або size(V,'c') — число стовпців;

sum(X[,fl]) — вычисляет суму елементів масиву X, має необов'язковий параметр fl. Якщо параметр fl відсутній, то функція sum(X) повертає скалярне значення, рівне сумі елементів масиву. Якщо fl='r' або fl=1, що те ж саме, то функція поверне рядок, рівний поелементній сумі стовпців матриці X. Якщо fl='c' або fl=2, то результатом роботи функції буде вектор-стовпець, кожен елемент якого рівний сумі елементів рядків матриці X. Окремий випадок застосування функції sum — це обчислення скалярного твору векторов1;

prod(X[,fl]) — вычисляет твір елементів масиву X, працює аналогічно функції sum;

max(M[,fl]) — вычисляет найбільший елемент в масиві M, має необов'язковий параметр fl. Якщо параметр fl відсутній, то функція max(M) повертає максимальний елемент масиву M; якщо fl='r', то функція поверне рядок максимальних елементів стовпців матриці M; якщо fl='c', то результатом роботи функції буде вектор-стовпець, кожен елемент якого рівний максимальному елементу відповідних рядків матриці M. Функція [x, nom]=max(M[,fl]) поверне значення максимального елементу x і його номер в масиві nom;

min(M[,fl]) — вычисляет найменший елемент в масиві M, працює аналогічно функції max;

mean(M[,fl]) — вычисляет середнє значення масиву M; якщо M двовимірний масив, то mean(M,1) або mean(M,'r') визначають середнє значення рядків матриці M, а mean(M,2) або mean(M,'c') — среднее значення стовпців;

median(M[,fl]) — вычисляет медіану масиву M, працює аналогічно функції mean;

det(M) — вычисляет визначник квадратної матриці M;

rank(M[,tol]) — вычисление рангу матриці Mс точністю tol

norm(M[,fl]) — вычисление норми квадратної матриці M; тип норми визначається необов'язковій строковій змінній fl, за умовчанням fl=2. Функції

norm(M) і norm(M,2) еквівалентні і обчислюють другу норму матриці Мпервая нормаопределяется функцією norm(M,1). Функції norm(M,'inf') і norm(M,'fro') обчислюють відповідно нескінченну і евклидову норми Якщо V — вектор, то результатом роботи функції norm(V,1) буде сума модулів всіх елементів вектора V. За допомогою функції norm(V,2) можна обчислити модуль вектора V Значення norm(V,'inf') рівне модулю максимального елементу вектора по модулю;

cond(M) — вычисляет число обумовленості матриці M по другій нормі;

Врес(M) — вычисляет власні значення і власні вектори квадратної матриці M.

inv(A) — вычисляет матрицю, зворотну до A;

pinv(A[,tol]) — вычисляет псевдозворотну матрицю для матриці A з точністю tol (необов'язковий параметр);

linsolve(A,b) — решает систему лінійних рівнянь алгебри виду A • x — Ь = 0.

rref(A) — здійснює приведення матриці A до трикутної форми, використовуючи метод виключення Гауса;

qr(M) — выполняет розкладання матриці M на ортогональну і верхню трикутну матриці;

У Scilab можна задавати символьні матриці, тобто матриці, елементи яких мають строковий тип. При цьому необхідно пам'ятати, що строкові елементи повинні бути поміщені в подвійні або одинарні лапки.

Символьні матриці можна складати (результат складання — конкатенація відповідних рядків) і транспонувати Крім того, операції складання і множення можна проводити над окремими елементами символьних матриць за допомогою функцій addf(a,b) і mulf(a,b).

***Контрольні питання:***

1 *Перерахувати операції які передбачені для роботи з матрицями і векторами*
2. *Перерахувати команди необхідні для роботи з матрицями і векторам*

# ТЕМА 4

# ПОБУДОВА ДВОМІРНИХ ГРАФІКІВ

У цьому розділі читач може познайомитися з графічним апаратом Scilab для побудови двовимірних графіків. Двовимірними рахуватимемо такі графіки, в яких положення кожної крапки задається двома величинами.

## 4.1 Функція plot

Розгляд графіків почнемо з простих функцій вигляду y = f (x), для побудови яких в Scilab існує функція plot. У попередніх версіях Scilab (по третю версію Scilab включно) функція plot призначена для побудови графіка однієї функції y = f (x). Звернення до неї має вигляд:

plot(x,y[xcap,ycap,caption])

Тут x — масив абсцис; y — масив ординат; xcap, ycap, caption — підписи осей X, Y і графіка відповідно.

Приклад Завдання. Побудувати графік функції = sin(cos(x)) за допомогою функції plot.

Хай x змінюється на інтервалі [—2π; 2π] з кроком 0,1. Сформуємо масив X. Обчислюючи значення функції y = sin(cos(x)) для кожного значення масиву X, створимо масив Y. Потім скористаємося функцією plot(x,y) для побудови кривої і виведемо з її ж допомогою підпису координатних осей 'X', 'Y', а також ім'я графіка 'plot function y=sin(cos(x))'

У простому випадку звернення до функції має вид plot(y), як масив x виступає масив номерів точок масиву y

Побудова декількох графіків в одній системі координат

При простому зверненні до функції plot(x,y) створюється вікно з ім'ям Scilab Graphic (0), в якому буде побудований графік функції y(x) на заданому інтервалі. Якщо ж повторно звернутися до функції plot, буде створено нове графічне вікно, і в нім буде побудований новий графік.

Для побудови декількох графіків в одній системі координат можна звернутися до функції plot таким чином:

plot(x1,y1,x2,y2...xn,yn)

де xl, y1 — масиви абсцис і ординат першого графіка; x2, y2 — массивы абсцис і ординат другого графіка;

xn, yn — массивы абсцис і ординат n-ого графіка. Завдання 4.3.

Побудувати в одних координатних осях графіки функцій = sin( X), z = cos(x) і v = exp(cos(x)).

Визначимо інтервал зміни x — [-6,28;6,28], крок — 0,02. Тепер сформуємо масиви значень функцій Y, Z, V.

Для побудови заданих кривих в одних координатних осях необхідно як аргументи функції plot попарно, через кому, вказати ім'я масиву першого аргументу і ім'я масиву першої функції, ім'я масиву другого аргументу і ім'я масиву другої функції і так далі В нашому випадку звернення до функції plot матиме вид plot(x,y,x,z,x,v)

Побудова графіків декількох функцій в одних координатних осях за допомогою команди plot в Scilab 4

```
x=-6.28:0.02:6.28;y=sin(x/2);

z=cos(x);v=exp(cos(x));

plot(x,y,x,z,x,v);
```

Побудувати декілька графіків в одному вікні можна і за допомогою короткого запису функції plot(x,y), але перед зверненням до функцій plot(x2,y2), plot(x3,y3) ..., plot(xn,yn) викликати команду mtlb_hold('on'), вона заблокує режим створення нового вікна.

Допустимо, що x належить інтервалу [—2п; 2п] і змінюється з кроком 0,1. Створимо масив X. Оскільки x є аргументом для всіх чотирьох функцій, його в обігу до функції plot можна не указувати. Також необов'язково формувати для кожної функції свій масив значень. Досить вказати в квадратних дужках через крапку з комою їх математичні вирази, і ці масиви автоматично будуть створені як проміжний етап побудови кривих функцій (див. лістинг 4.3 і рис 4.3).

Побудова графіків декількох функцій в одних координатних осях за допомогою команди plot в Scilab 3

```
x=-2*%pi:0.1:2*%pi;
```

plot([sin(cos(x));cos(sin(x));exp(sin(x));exp(cos(x))]);

Як видно з мал. 4.3, функція plot в Scilab 3 не дозволяє побудувати повноцінні графіки декількох функцій. Тому в Scilab 4 вона була значно модифікована, і її можливості тепер значно розширені. Просте ж звернення до функції plot в четвертій версії додатку залишилося колишнім plot(x,y).

У Scilab 4 можна виводити декілька графіків в одному вікні, не суміщаючи їх в одних координатних осях. Наприклад, якщо графічне вікно повинне містити 4 самостійних графіка, все вікно розбивається на 4 області, а потім в кожну з них виводиться графік функції.

Для формування області в графічному вікні служить команда plotframe:

plotframe(rect, tics [,grid, title, x-leg, y-leg, x, y, w, h])

де rect — вектор [xmin, ymin, xmax, ymax], який визначає межі зміни x і y-координат області;

tics — вектор [nx, mx, ny, my], який визначає кількість ліній сітки по осі X (mx) і Y (my), величини nx і ny повинні визначати число подин- тервалов по осях X і Y;

grid — логическая змінна, яка визначає наявність (%t) або відсутність координатної сітки (%f). Цей параметр слід указувати для обох осей, наприклад [%t,%t];

bound — логічна змінна, яка при значенні true дозволяє ігнорувати параметри tics(2) і tics(4).

title — заголовок, який буде виводиться над графічною областю;

x-leg, y-leg — подписи осей графіка X і Y;

x, y — координати верхнього лівого кута області в графічному вікні, w — ширина, h — висота вікна. Значення x, y, w, h вимірюються у відносних одиницях і лежать в діапазоні [0,1].

### 4.2 Функція plot2d

Наступною функцією, яка може бути використана для побудови двовимірних графіків, є функція plot2d. У загальному вигляді звернення до функції має вигляд:

plot2d([logflag],x,y'[key1=value1,key2=value2...,keyn=valuen]

де logflag — рядок з двох символів, кожен з яких визначає тип осей (п — нормальна вісь ! — логарифмическая вісь), по умолчанию— «пп»;

x — массив абсцис;

y — масив ординат або матриця, кожен стовпець яких містить масив ординат чергового графіка — у випадку, якщо необхідно побудувати графіки декількох функцій yl, y2 ..., yn, коли всі вони залежать від однієї і тієї ж змінної x. При цьому кількість елементів в масиві x і y повинно бути однаковим. Якщо x і y — матриці одного розміру, то кожен стовпець матриці y відображається щодо відповідного стовпця матриці x;

keyi=valuei — последовательность значень властивостей графіка key1=value1, key2=value2 ..., keyn=valuen

що визначають його зовнішній вигляд. Можливі значення властивостей графіка будуть детально описані нижчим.

Слід зазначити, що зовсім не обов'язково використовувати повну форму запису функції plot2d зі всіма її параметрами. У простому випадку до неї можна звернутися стисло, як і до функції plot.

### 4.3 Оформлення графіків за допомогою функції plot

Встановити бажаний вигляд і колір графіка можна, використовуючи повну форму звернення до функції plot:

plot(x1, yl, si, x2, y2, s2 ..., xn, yn, sn)

де xl, x2 ..., xn — массивы абсцис графіків; yl, y2 ..., yn — массивы ординат графіків;

si, s2 ..., sn — рядок, що складається з трьох символів, які визначають відповідно колір лінії, тип маркера і тип лінії графіків (див. табл. 4.14.3), в рядку можуть використовуватися один, два або три символи одночасні в будь-якій бажаній комбінації.

Таблиця 4.1.

Символи, що визначають колір лінії графіка

| **Символ** | **Опис** |
|---|---|
| U | жовтий |
| т | рожевий |
| з | блакитний |
| г | червоний |
| g | зелений |
| b | синій |
| V | білий |
| до | чорний |

Щоб графік простіше «читався», зручно виводити сітку — додаткові осі для показника X і показника Y. У Scilab це можна зробити за допомогою команди xgrid(color), де color визначає id кольори лінії сітки. Якщо залишити дужки порожніми, за умовчанням промальовує сітка чорного кольору.

Заголовок графіка, побудованого функцією plot, можна вивести командою xtitle:

Таблиця 4.2.

Символи, що визначають тип маркера

| **Символ** | **Опис** |
|---|---|
| . | крапка |
| o | кружок |
| x | хрестик |

| | |
|---|---|
| + | знак «плюс» |
| * | зірочка |
| s | квадрат |
| d | ромб |
| v | трикутник вершиною вниз |
| Δ | трикутник вершиною вгору |
| < | трикутник вершиною вліво |
| > | трикутник вершиною управо |
| p | п'ятикутна зірка |

xtitle(title, xstr, ystr) де title — название графіка; xstr — название осі X; ystr — назва осі Y.

legend(leg1, leg2 ..., legn [pos] [boxed])

де leg1 — ім'я першого графіка, leg2 — ім'я другого графіка, legn — ім'я n-го графіка;

pos — месторасположение легенди: 1—у верхньому правому куті (за умовчанням), 2 — у верхньому лівому кутку, 3 — в нижньому лівому кутку, 4 — в нижньому правому кутку, 5 — визначається користувачем після зображення графіка;

boxed — логічна змінна, яка визначає, промальовувати (значення за умовчанням — %t) чи ні (значення %f) рамку навколо легенди.

style — визначає масив числових значень квітів графіка. Кількість елементів масиву співпадає з кількістю графіків, що зображаються. Можна скористатися функцією color, яка по назві (м^гО'имя кольору")) або коду rgb (color(r,g,b)) кольору формує потрібний id (код) кольору.

Повний перелік всіх доступних при форматуванні відтінків з їх RGB- id можна знайти в статті вбудованої довідкової системи Scilab «Color_list». Проте слід врахувати, що в статті перед id-цвета опущені «0.».

rect—значение параметра keyn=valuen функції plot2d — це вектор [xmin, ymin, xmax, ymax], що визначає розмір вікна навколо графіка.

Тут xmin, ymin — положение верхньої лівої вершини; xmax — ширина вікна; ymax — висота вікна.

axesflag — значення параметра кеуп=уа1иеп функції plot2d — визначає наявність рамки навколо графіка. Необхідно виділити наступні базисні значення цього параметра:

0— немає рамки;

1—изображение рамки, вісь у зліва (за умовчанням);

3 — зображення рамки, вісь у справа;

5 — изображение осей, що проходять через крапку (0,0).

Для того, щоб визначити число основних і проміжних ділень координатних осей, в skilab існує параметр пах. Якщо параметр axesflag=1 (за умовчанням), це масив з чотирьох значень: [nx, Nx, ny, Ny].

Тут Nx (Ny) — число основних ділень з підписами під віссю X (У); nx (ny) — число проміжних ділень.

leg — значення параметра keyn=valuen функції plot2d — рядок, що визначає легенди для кожного графіка:

"leg1@leg2@leg3@...@legn"

де leg1 — легенда першого графіка ..., legn — легенда n-го графіка

### 4.4 Побудова точкових графіків

Функцію plot2d можна використовувати для побудови точкових графіків. В цьому випадку звернення до функції має вигляд:

plot2d(x,y,d)

де d — негативне число, що визначає тип маркера

Таблиця 4.3

Числа визначають тип маркера

| **Число** | **Опис** |
|---|---|
| -0 | крапка |
| -1 | плюс |
| -2 | хрестик |
| -3 | плюс, вписаний в коло |
| -4 | закрашений ромб |
| -5 | незакрашений ромб |
| -6 | трикутник вершиною вгору |
| -7 | трикутник вершиною вниз |
| -8 | плюс, вписаний в ромб |
| -9 | кружок |
| -10 | зірочка |
| -11 | квадрат |
| -12 | трикутник вершиною управо |
| -13 | трикутник вершиною вліво |
| -14 | п'ятикутна зірка |

### 4.5 Побудова графіків у вигляді ступінчастої лінії

Для зображення графіка у вигляді ступінчастої лінії у BoiIaь існує функція plot2d2(x,y). Вона повністю співпадає по синтаксису з функцією plot2d. Головна відмінність полягає в тому, що X і У можуть бути незалежними один від одного функціями, важливо лише, щоб масиви X і У були розбиті на однакову кількість інтервалів

Побудова графіків в полярній системі координат

Полярна система координат складається із заданої фіксованої точки Про — полюса, концентричних кіл з центром в полюсі і променів, що виходять з крапки Про, один з яких ОХ — полярна вісь.

Має в своєму розпорядженні будь-якої крапки М в полярних координатах можна задати позитивним числом р = ОМ (полярний радіус), і числом у, рівним величині кута ХОМ (полярний кут).

У BoiIaь для формування графіка в полярній системі координат необхідно сформувати масиви значень полярного кута і полярного радіусу, а потім звернутися до функції роїагрIo'Ь:

ро1агр1о'Ь(ii,го,[кеу1=уа1ие1,кеу2=уа1ие2...,кеуп=уа1иеп])

де іі — полярный кут; го — полярний радіус;

кеуп=уа1иеп — последовательность значень властивостей графіка

Завдання функції у(х) за допомогою рівності х = /(J) і у = $(J) називають параметричним, а допоміжну величину J — параметром.

Для побудови графіка функції, заданої параметрично, необхідно визначити масив J, визначити масиви х = /(J), у = #(J) і побудувати графік функції(х), використовуючи функції р1сЪ(х,у) або р1ot2d(x,y)

### 4.6 Режим форматування графіка

У Scilab зовнішній вигляд графіка можна змінювати, використовуючи можливості графічного вікна, в якому він відображається. Перехід до режиму форматування здійснюється командою Edit - Figure properties меню графічного вікна.

Командою меню графічного вікна Edit - Figure properties викликаємо вікно форматування отриманого графіка Figure Editor

Ліва частина вікна — Object Browser — це поле проглядання об'єктів, доступних для форматування. Клацання по об'єкту Figure(1) (Графічне вікно) робить його активним, а в правій області вікна — Object Properties — з'являються властивості активного об'єкту, які можуть бути змінені.

Спочатку в полі Object Browser завжди відображаються два об'єкти: Figure (Графічне вікно) і його дочірній об'єкт Axes (Oci). Значок «плюс» біля об'єкту указує на те, що він містить об'єкти нижчого порядку.

Якщо клацнути по значку «плюс» у об'єкту Axes (Oci ), з'явиться об'єкт — Compound(1) (Група) також із значком «плюс». Об'єкт Compound(1) містить побудовані в одних координатних осях графіки функцій y = sin(2x) і y2 = sin(3x) — відповідно Polyline(2) і Polyline(1)

### 4.7 Форматування об'єкту Figure (Графічне вікно)

Нагадаємо, що об'єкт Figure — це графічне вікно і власне графік, що відображається в нім. Для зміни властивостей графічного вікна необхідно виділити Figure(1) в полі Object Browser вікна форматування.

закладка Style вікна форматування Figure Editor для об'єкту Figure(1). Тут можна змінити значення наступних властивостей:

Visibility (відображення графіка) — переключатель, що приймає значення «on» і «off». За умовчанням встановлений стан «on» — графік виводиться на екран.

Figure name (ім'я графіка) — це послідовність символів, які виводяться в рядку заголовка графічного вікна. За умовчанням графічному вікну привласнюється ім'я Scilab Graphic (%d), де %d — це порядковий номер графіка (Figure id). Для першого графічного вікна Figure id рівний 0, для другого — 1, для третього — 2 і так далі Проте ви можете ввести будь-яке бажане ім'я. Наприклад, замінити Scilab Graphic (%d) на Grafic y=f(x) і натиснути клавішу Enter. Заголовок вікна буде змінений

X position, Y position — ці поля визначають положення графічного вікна на моніторі в пікселях по горизонталі і вертикалі відповідно. Крапка з координатами [0;0] — верхний лівий кут екрану.

X size, Y size — це відповідно ширина і висота графічного вікна в пікселях.

X axis size, Y axis size — ці значення визначають розмір осей X і Y.

Back. color (колір фону) — каждому положенню повзунка відповідає свій номер кольору (RGB-id). Доступні 35 відтінків (від -2 — білий до 32 — желто- гарячий).

Встановимо повзунок в положення 15. Колір фону стане зеленим Колірна палітра може бути змінена користувачем на закладці Colormap

Встановлюючи значення для червоного (RED), зеленого (GREEN) і синього (BLUE) кольорів, можна змінювати кожен відтінок незалежно, не викликаючи змін інших квітів в палітрі. Наприклад, вектор [0 0 0] задає чорний колір, [0.230 0.230 0.250] - блакитний, [0.85 0.107 0.47] - малиновий.

У рядку Colormap (Nx3 double array) можна задати RGB-id для всієї палітри.

Повний перелік всіх доступних при форматуванні відтінків з їх RGB- id можна знайти в статті вбудованої довідкової системи Scilab «Color_list». Проте слід врахувати, що в статті перед id-цвета опущені «0.».

На закладці Mode в області Object Properties для об'єкту Figure (див. мал. 4.30) можна встановити наступні властивості:

Auto resize — властивість, яка дозволяє змінювати розмір графіка. Коли цей режим включений (положення перемикача «on»—по умовчанню), ми можемо змінювати розмір графічного вікна, перетягуючи його межі за допомогою миші, і при цьому автоматично змінюватиметься розмір графіка, що відображається у вікні. У вимкненому положенні графік зберігатиме свої розміри

Pixmap — режим растрового зображення. У вимкненому положенні (за умовчанням) зображення формується безпосередньо на екрані. При включеному режимі (положення перемикача «on») графік створюється як растрове зображення і прямує в графічне вікно командою show_pixmap(). Слід зазначити, що режим Pixmap використовується при створенні анімованих графіків для згладжування переходів між кадрами.

Pixel drawing mode — властивість, яка визначає спосіб формування зображення на екрані. За умовчанням встановлений режим «сміттю». В цьому випадку точно виконується необхідна операція побудови графіка. Проте часто необхідно нанести зображення що вже існує, при цьому колір знов побудованого графіка повинен чітко виділятися. Для цього існує набір режимів: «clear», «and», «andReverse», «andInverted», «noop», «xor», «or», «nor», «equiv», «invert», «orReverse», «copyInverted», «orInverted», «nand», «set».

Rotation style — это властивість застосовна лише до тривимірних графіків. Режим за умовчанням «unary» призначений для обертання виділених графіків, при включеному режимі «(multiple» обертаються всі тривимірні графіки

### 4.8 Форматування об'єкту Axes (Осі графіка)

Для зміни властивостей об'єкту Axes (Осі графіка) необхідно виділити його в полі Object Browser вікна форматування. В області Object Properties доступні для модифікації властивості будуть згруповані на декількох закладках. Закладки X, Y і Z ідентичні, з тією лише різницею, що дозволяють встановлювати бажаний зовнішній вигляд відповідно для осей X, Y і Z. Тому ми розглянемо лише закладку X (див. мал. 4.33).

На закладці X всі властивості розділено на дві області: Label Options (Ової- ства підписів осей) і Axis Options (Властивості осей). В області Label Options можна встановити:

Label — собственно підпис осі — будь-яка послідовність символів;

Visibility —видимость—переключатель, що приймає значення «on» і «off». За умовчанням осі графіка виводяться на екран (положення «on»).

Fill mode — режим заливки — перемикач, що приймає значення «on» і «off» (за умовчанням). Для того, щоб визначити колір фону навколо підпису осі, необхідно встановити стан «on».

Auto position — автоматичне визначення положення підпису осі графіка. За умовчанням встановлено значення «on» — підпис виводиться внизу, по центру осі. Проте положення підпису можна визначити і самостійно, для цього в полі Position задаються координати у вигляді вектора [x, y]. При цьому перемикач Auto position автоматично прийме значення «off».

Auto Rotation — режим автоматичного обертання підпису осі. За умовчанням цей режим відключений (стан перемикача «off»).

Font angle — кут повороту підпису осі. Можна встановити одне з пропонованих значень: 0, 90, 180 і 270 градусів, а також будь-який довільний кут повороту напису в останньому полі (див. мал. 4.33).

Fore/Back colors — цвет символів і колір фону підпису осі відповідно - встановлюються за допомогою повзунка, кожному положенню якого відповідає певний колір. Всього доступні 35 квітів. Докладніше за див. параграф 4.6.

Font size — размер символів підпису осі, можливі значення від 0 до 6. За умовчанням для шрифту встановлений розмір 1.

Font style — стиль зображення символів підпису осі. За умовчанням встановлений стиль Helvetica (рубаний).

Location — розташування осі графіка. Для осі X можливі наступні значення цієї властивості: bottom — знизу, top — зверху, middle — посередині; а для осі Y : left — зліва, right — справа, middle — посередині.

Grid color — колір ліній сітки графіка, що встановлюється за допомогою повзунка. У положенні -1 лінії сітки графіка відсутні, в положенні 0 виводяться чорні лінії, крім того доступні ще 32 кольори. Для того, щоб відображалися лінії сітки для осей X і Y, необхідно встановити властивість Grid color і на закладці X, і на закладці Y.

Data bounds — ограничение даних. Для кожної осі можна зменшити діапазон початкових даних, по яких формується графік, зробивши його детальнішим.

Scale — масштаб осі графіка. Існує два автоматичні режими: lin (лінійний) і log (логарифмічний). Натиснення на кнопку Ticks (Зарубки) приводить до появи вікна модифікації ділення осі Edit Axes Ticks.

З її допомогою можна встановити наступні властивості зарубок координатних осей:

Visibility — відображення — перемикач, що приймає значення «on» і «off». За умовчанням зарубки на осі графіка виводяться на екран (положення «on»).

Auto ticks — режим автоматичного ділення осі, за умовчанням також включений (значення перемикача «on»). Проте існує можливість самостійно визначити крок, з яким буде розбита вісь, його потрібно ввести в поле Steb by і натиснути Enter. При цьому перемикач Auto ticks автоматично прийме значення «off».

Sub ticks — проміжні зарубки. У цьому полі потрібно ввести число зарубок, які виводитимуться між основними діленнями осі. Слід зазначити, що проміжні зарубки не підписуються.

У вікні Edit Axes Ticks формується таблиця основних зарубок (без зарубок Sub ticks). Перший стовпець Locations задає положення зарубки, а другий Labels — підпис зарубки.

Для зручності редагування таблиці вікно забезпечене кнопками Insert, Delete, Apply, Quit. Кнопка Insert дозволяє вставити у вікно готову таблицю зарубок (або її фрагмент) за допомогою буфера обміну. Вставка проводиться починаючи з позиції активного осередку. Кнопка Delete дозволяє видаляти не тільки активний осередок, але і весь рядок, яким вона належить. Кнопка Apply підтверджує зміни, а Quit служить для виходу з вікна Edit Axes Ticks.

Закладка Style вікна форматування осей графіка Axes Editor надає можливість змінювати наступні властивості лінії осі і підписів зарубок:

Visibility — відображення — перемикач, що приймає значення «on» (за умовчанням) і «off». У положенні «off» графік взагалі не виводиться у вікно.

Font style — стиль зображення символів підписів зарубок на осі. За умовчанням встановлений стиль Helvetica.

Font color — повзунок, кожне положення якого визначає колір символів підписів зарубок. За умовчанням встановлений в положенні -1 — черный колір.

Font size — розмір символів підписів зарубок на осі, можливі значення від 0 до 6. За умовчанням для шрифту встановлений розмір 1.

Fore. color — повзунок, кожне положення якого визначає колір власне координатній осі. За умовчанням встановлений в положенні -1 — черный колір.

Back. color — повзунок, кожне положення якого визначає колір заливки фону графіка. За умовчанням встановлений в положенні -2 — білий колір.

Thickness — товщина лінії координатної осі, визначувана повзунком з положеннями від 1 до 30. За умовчанням для товщини лінії встановлено значення 1.

Line style — стиль зображення лінії. Можливі 6 режимів: solid — суцільна лінія, решта режимів — варіації пунктирів.

Закладка Aspect вікна форматування Axes Editor (див. мал. 4.39) дозволяє змінювати наступні властивості:

Auto clear — если перемикач встановлений в положення «on», графічне вікно автоматично очищатиметься кожного разу перед побудовою нового графіка. Якщо ж цей режим відключений (за умовчанням), графіки накладатимуться в одних координатних осях відповідно до режиму Auto scale.

Auto scale — режим оновлення меж координатних осей графіка. В стані перемикача «on» (за умовчанням) новий графік змінить межі попереднього графіка, щоб сформуватися на всьому заданому інтервалі, але в тому ж масштабі, що і попередній графік. При відключеному режимі Auto scale новий графік буде побудований в межах осей попереднього графіка і, можливо, відображатиме лише частину заданого інтервалу.

Перемикач Boxed на закладці Aspect вікна форматування Axes Editor визначає, обмежувати графік прямокутною рамкою (положення «on» за умовчанням) або виводити тільки координатні осі (положення «off»).

Isoview — ця властивість використовується для того, щоб встановити однаковий масштаб для всіх осей графіка. За умовчанням встановлений стан перемикача «off».

Tight limits — якщо цей режим включений, осі графіка змінюються так, щоб точно відповідати значенню властивості Data bounds закладок X, Y і Z. При значенні «off» (за умовчанням) осі можуть збільшити початковий інтервал, щоб було простіше вибрати масштаб осі і нанести на неї зарубки.

Cube scaling — эта опція застосовна лише до тривимірних графіків. При стані перемикача «on» початкові дані обмежуються так, щоб поверхня помістилася в куб розміром 1. Це дозволяє наочніше зобразити 3D-график в тих випадках, коли масштаб координатних осей дуже різниться від однієї осі до іншої. За умовчанням встановлений стан перемикача «off».

dip state — режим того, що кадрується (обрізання) графіка. Можливий один з наступних станів перемикача: «off » означає, що створюване зображення не кадрується; «clipgrf » (за умовчанням) — от створюваного зображення обрізається область, що знаходиться поза межами осей; «on» — от створюваного зображення обрізається область, що знаходиться поза межами, заданими властивістю Glip box.

dip box — прямокутна область, яка відображатиметься після обрізання зображення. Спочатку в полях X і Y задаються координати верхньої лівої точки прямокутника (upper-left point coordinates), потім ширина і висота — поля W і H.

Margins — цю властивість встановлює відстань від межі графічного вікна до області графіка: Left (лівий край), Right (правий), Top (верхній), Bottom (нижний). Значення повинне знаходитися в інтервалі [0 : 1]. За умовчанням кожному полю привласнено значення 0.125.

Axes bounds — цю властивість задає частина графіка, яка виводитиметься в координатних осях. Left і Up визначають положення верхній лівий кут, Width і Height — ширину і висоту фрагмента графіка. Значення повинне знаходитися в інтервалі [0:1]. Фрагмент, що за умовчанням відображається, задається матрицею [0011].

Закладка View Point вікна форматування осей графіка Axes Editor дозволяє встановити лише одне свойство—угол, під яким спостерігач бачить графік. За умовчанням встановлені значення кутів повороту спостерігача (Rotation angles) 0 і 270.

### 4.9 Форматування об'єкту Polyline (Лінія графіка)

Для переходу до форматування лінії графіка необхідно вибрати об'єкт Polyline в полі проглядання об'єктів Object Browser вікна Polyline Editor. Доступні для зміни властивості в області Object Properties згруповані на трьох закладках: Style, Data, Clipping.

Закладка Style вікна форматування Polyline Editor (див. мал. 4.43) дозволяє встановити значення наступних властивостей:

Visibility — отображение — перемикач, що приймає значення «on» (за умовчанням) і «off». У положенні «off» лінія графіка не відображається у вікні.

Fill mode — режим заливки — перемикач, що приймає значення «on» і «off» (за умовчанням). Для того, щоб визначити колір фону області, яку обмежує крива, перемикач необхідно встановити в стан «on».

Closed — если включити цю властивість, лінія графіка стане замкнутою.

Polyline style — стиль відображення графіка. Можливі наступні значення: interpolated — суцільна плавна лінія; staircase — ступінчаста лінія; barplot — смужчаті області; arrowed — лінія, що складається з послідовності стрілок, розмір стрілки можна встановити в полі Arrow size; filled — закрашені області; bar — смужчаті області, обмежені суцільною плавною лінією.

Line — стилі зображення лінії графіка. Доступні 6 стилів: solid — суцільна, останні — варіації пунктирної лінії. Тут же із списку можна вибрати бажану товщину кривої: від 1 до 30.

Foreground і Background — властивості, що встановлюють відповідно колір лінії графіка і заливку області, яка обмежується кривою, при цьому перемикач Fill mode повинен бути встановлений в положення «on».

Interp color vector — вектор, що визначає заливку кожного сегменту графіка.

Mark mode — режим, що дозволяє будувати точкові графіки (положення перемикача «on»). За умовчанням ця властивість відключена.

Mark style — стиль маркера — можливі наступні значення: dot — крапка; plus — знак «плюс»; cross — хрестик; star — плюс, вписаний в коло; filled diamond — закрашений ромб; diamond — ромб; triangle up — трикутник вершиною вгору; triangle down — трикутник вершиною вниз; diamond plus — плюс, вписаний в ромб; circle — кружок; asterisk — зірочка; square — квадрат; triangle right — трикутник вершиною управо; triangle left — трикутник вершиною вліво; pentagram — п'ятикутна зірка.

Mark size — розмір маркера — встановлювані значення можуть змінюватися від 0 до 30pt.

Mark foreground — повзунок, кожне положення якого визначає колір заливки маркера.

***Контрольні питання:***

1. *Перерахувати основні команди меню в середовищі Skilab*
2. *Перерахувати функції за допомогою яких вводяться змінні*
3. *Перерахувати основні функції за допомогою яких вводяться елементарні математичесские функції*

# ТЕМА 5
# ПОБУДОВА ТРИВИМІРНИХ ГРАФІКІВ В SKILAB

У справжньому розділі будуть розглянуті основні можливості 8еПаЬ по створенню тривимірних графіків — об'ємних і просторових. При цьому до тривимірних віднесемо всі графіки, положення кожної точки яких задається трьома величинами.

В цілому процес побудови графіка функції виду Z(x, y) можна розділити на три етапи:

1. Створення в області побудови графіка прямокутної сітки. Для цього формуються прямі лінії, паралельні координатним осям і уз-, де:

$$x_i = x_0 + ih, \quad h = \frac{x_n - x_0}{n}, \quad i = 0, 1, \ldots, n,$$
$$y_j = y_0 + jh, \quad h = \frac{y_k - y_0}{k}, \quad j = 0, 1, \ldots, k.$$

Обчислення значень функції $z_{ij} = f(x_i, y_j)$ у всіх вузлах сітки.

Звернення до функції побудови тривимірних графіків.

## 5.1 Функції plot3d і plot3d1

У Skilab поверхню можна побудувати за допомогою функцій plot3d або plot3d1. Їх відмінність полягає в тому, що plot3d будує поверхню і заливає її одним кольором, а plot3d1 — поверхность, кожен осередок якої має колір, залежний від значення функції в кожному відповідному вузлі сітки

Звернення до функцій наступне:

plot3d(x,y,z[theta,alpha,leg,flag,ebox][keyn=valuen]),
plot3d1(x,y,z[theta,alpha,leg,flag,ebox][keyn=valuen])

тут x — вектор-стовпець значень абсцис; y — вектор-стовпець значень ординат; z — матриця значень функції;

theta, alpha — действительные числа, які визначають в градусах сферичні координати точки зору на графік. Просто кажучи, це кут, під яким спостерігач бачить поверхню, що відображається;

leg — підписи координатних осей графіка — символи, відокремлювані знайомий @. Наприклад, 'X@Y@Z'.

flag — масив, що складається з трьох цілочисельних параметрів: [mode,type,box]. Тут

mode — встановлює колір поверхні (див. табл. 5.1). За умовчанням рівний 2—цвет заливки синій, прямокутна сітка виводиться.

Таблиця 5.1

Значення параметра mode

| **Значення** | **Опис** |
|---|---|
| > 0 | поверхня має колір «mode», виводиться прямокутна сітка |
| 0 | виводиться прямокутна сітка, заливка відсутня (белыйц вет) |
| < 0 | поверхня має колір «mode», відсутня прямокутна сітка |

type — позволяет управляти масштабом графіка (див. табл. 5.2), за умовчанням має значення 2;

ebox — визначає межі області, в яку виводитиметься поверхня, як вектор [^т^, xmax, уш1п, ymax, zmin, zmax] . Цей параметр може використовуватися тільки при значенні параметра type=1.

Таблиця 5.2

Значення параметра type

| **Значення** | **Опис** |
|---|---|
| 0 | застосовується спосіб масштабування, як у раніше созданногографика |
| 1 | межі графіка указуються уручну за допомогою параметра ebox |
| 2 | межі графіка визначають початкові дані |

box — визначає наявність рамки навколо графіка, що відображається, За умовчанням рівний 4.

Таблиця 5.3

Значення параметра box

| **Значення** | **Опис** |
|---|---|
| 0 і 1 | немає рамки |
| 2 | тільки осі, що знаходяться за поверхнею виводиться рамка і підписи осей виводиться рамка, осі і їх підписи |

keyn=valuen — послідовність значень властивостей графіка key1=value1, key2=value2 ..., keyn=valuen

Таким чином, функції plot3d (plot3d1) як параметри необхідно передати прямокутну сітку і матрицю значень у вузлах сітки.

Як видно з прикладу, використання лише функції plot3d для графічного зображення показників, залежних від двох незалежних змінних, достатньо складно. У Scilab існує декілька команд, покликаних полегшити процедуру створення прямокутної сітки, — це genfac3d і eval3dp. Простою з них по синтаксису є функція genfac3d:

[xx,yy,zz]=genfac3d(x,y,z)

Тут xx, yy, zz — результирующая матриця розміром (4, n — 1 x m — 1), де xx(:,i), yy(:,i) і zz(:,i) — координаты кожному з осередків прямокутної сітки;

x — вектор x-координат розміру m;

y — вектор y-координат розміру n;

Для формування графіка звернемося до функції plot3d (див. мал. 5.1). Недоліком команди genfac3d є те, що вона все-таки не спрощує роботу з функцією plot3d, якщо поверхня задається функцією від двох змінних. У такому разі необхідно використовувати команду eval3dp:

[Xf,Yf,Zf]=eval3dp(fun,p1,p2)

Xf,Yf,Zf результуюча матриця розміром (4,n — 1 x m — 1), де xx(:,i)

yy(:,i) і zz(:,i) — координаты кожному з осередків прямокутної сітки;

fun — функція, визначена користувачем, яка задає тривимірний графік;

p1 — вектор розміру т;

p2 — вектор розміру n.

### 5.2 Функції meshgrid, surf і mesh

Для формування прямокутної сітки вперше в Scilab 4.0 з'явилася функція meshgrid. Звернення до неї має вигляд:

[X, Y [Z]] = meshgrid(x, y [z])

тут (x, y [z]) — массивы 2 (3) початкових параметрів X, Y (Z), що указуються через кому;

[X, Y [Z]] — матрицы у разі 2 і масиви у разі 3 вхідних величин.

Після формування сітки вивести в неї графік можна за допомогою функції surf або mesh. Так само, як і у випадку з функціями plot3d і plot3d1, surf будує поверхню, заливаючи кожен осередок кольором, який залежить від конкретного значення функції у вузлі сітки, а mesh заливає її одним кольором.

Таким чином, mesh є повним аналогом функції surf із значенням параметрів Color rnode^^eKC білого кольору в поточній палітрі квітів і Color flag=0.

***Звернення до функцій має вигляд:***

surf([X,Y],Z[color,keyn=valuen]) mesh([X,Y],Z[color])

тут X,Y — массивы, задаючі прямокутну сітку; Z — матриця значень функції;

color — матриця дійсних чисел, що встановлюють колір для кожного вузла мережі;

keyn=valuen — послідовність значень властивостей графіка key1=value1, key2=value2 ..., keyn=valuen

що визначають його зовнішній вигляд.

Звичайно, в тому випадку, якщо прямокутна сітка була побудована командою meshgrid, необхідності указувати параметри X,Y немає. У найпростішому випадку до функції surf можна звернутися так— surf(z).

У Scilab можна побудувати графіки двох поверхонь в одній системі координат, для цього, як і для двовимірних графіків, слід використовувати команду mtlb_hold('on'), яка блокує створення нового графічного вікна при виконанні команд surf або mesh

### 5.3 Функції plot3d2 і plot3d3

Функції plot3d2 і plot3d3 є аналогами функції plot3d, тому мають такий же синтаксис:

plot3d2(x,y,z[theta,alpha,leg,flag,ebox][keyn=valuen]),
plot3d3(x,y,z[theta,alpha,leg,flag,ebox][keyn=valuen])

Ці функції призначені для побудови поверхні, яка задається набором граней. Т. е. якщо функція plot3d за вхідними даними зможе побудувати лише плоскі грані, що окремо стоять один від одного, то plot3d2 (plot3d3) проінтерпретує взаємне розташування цих граней у вигляді цілісного геометричного тіла.

Відмінність функцій plot3d2 і plot3d3 схоже з відмінністю дії функцій plot3d і plot3d1, а також surf і mesh. Plot3d2 будує поверхню, при цьому виводить сітку і заливає всі осередки одним з квітів, за умовчанням — синім.

Plot3d також виводить сітку, проте залишає всі осередки без заливки (тобто білими)

### 5.4 Функція contour

У Scilab, окрім побудови об'ємних графіків, також реалізована можливість створення просторових моделей об'єктів. На практиці часто виникає необхідність побудови карт в ізолініях значень показника, де X, Y-координаты задають положення конкретної крапки, що вивчається, на площині, а Z-координата — зафіксовану величину показника в цій крапці. Крапки з однаковими значеннями показника сполучають так звані ізолінії — лінії однакових рівнів значень досліджуваної величини.

Для побудови ізоліній в Scilab існує функція contour. Звернення до неї має вигляд:

contour(x,y,z,nz[theta,alpha,leg,flag,ebox,zlev])

Тут x, y — массивы дійсних чисел;

z — матриця дійсних чисел - значення функції, що описує поверхню Z (x, y);

nz — параметр, який встановлює кількість ізоліній. Якщо nz — ціле число, то в діапазоні між мінімальним і максимальним значеннями

функції Z(x,y) через рівні інтервали будуть проведені nz ізоліній. Якщо ж задати nz як масив, то ізолінії проводитимуться через всі вказані в масиві значення;

theta, alpha — действительные числа, які визначають в градусах сферичні координати кута огляду спостерігача. Просто кажучи, це кут, під яким спостерігач бачить поверхню, що відображається;

leg — підписи координатних осей графіка — символи, відокремлювані знайомий Наприклад, 'X@Y@Z'.

flag — масив, що складається з трьох цілочисельних параметрів: [mode,type,box]. Тут

mode — встановлює спосіб і місце нанесення ліній рівня

type — позволяет управляти масштабом графіка за умовчанням має значення 2;

box — визначає наявність рамки навколо графіка, що відображається, За умовчанням рівний 4;

ebox — визначає межі області, в яку виводитиметься поверхня, як вектор [xmin,xmax,ymin,ymax,zmin,zmax] . Цей параметр може використовуватися тільки при значенні параметра type=1;

zlev — математичний вираз, який задає план (горизонтальну проекцію заданої поверхні) для побудови ізоліній. За умовчанням співпадає з рівнянням, що описує площину, — в цьому випадку може не указуватися.

Слід зазначити, що функції contour рівняння поверхні Z(x, y) зручніше передавати як параметр як функцію, визначену користувачем.

Нагадаємо, що функції в Scilab створюються за допомогою команди deff: deff('[s1,s2...]=newfunction(e1,e2...)

де s1,s2... — список вихідних параметрів, тобто змінних, яким буде привласнений кінцевий результат обчислень;

newfunction — имя створюваної функції, воно використовуватиметься для її виклику;

e1,e2... — входные параметри.

Другий спосіб створення функції - це примение конструкції вигляду:

function <lhs_arguments>=<function_name><rhs_arguments>

‹тело_функции>

endfunction

де lhs_arguments — список вихідних параметрів; function_name — имя створюваної функції; rhs_arguments — входные параметри.

### 5.5 Функція contourf

У Scilab існує функція contourf, яка не просто зображає поверхню на горизонтальній площині у вигляді ізоліній, але і заливає інтервали між ними кольором, залежно від конкретного рівня значень показника.

Звернення до функції має вигляд:

contourf (x,y,z,nz[style,strf,leg,rect,nax])

Тут x, y — массивы дійсних чисел;

z — матриця дійсних чисел — значення функції, що описує поверхню Z(x, y);

nz — параметр, який встановлює кількість ізоліній. Якщо nz — целое число, то в діапазоні між мінімальним і максимальним значеннями функції Z(x, y) через рівні інтервали будуть проведені nz ізоліній. Якщо ж задати nz як масив, то ізолінії проводитимуться через всі вказані в цьому масиві значення;

style — масив того ж розміру, що і nz — встановлює колір для кожного інтервалу рівнів значень;

strf — строка, що складається з трьох чисел, — «csa». Тут з (Captions) встановлює режим відображення підписів графіка (див. табл. 5.5); s (Scaling) — режим масштабування (див. табл. 5.6); a (Axes) — определяет положення осей графіка (див. табл. 5.7).

leg — легенда графіка, підпис кожній з кривих — символи, відокремлювані знайомий По умолчанию— « ».

rect — вектор [xmin, ymin, xmax, ymax], який визначає межі зміни x і y координат графічної області вікна;

Таблиця 5.4

Значення параметра з (Captions) рядка strf

| **Значення** | **Опис** |
|---|---|
| 0 | немає підписів |
| 1 | відображаються підписи, задані параметрів leg |

nax — це масив з чотирьох значень [nx, Nx, ny, Ny], що визначає число основних і проміжних ділень координатних осей графіка. Тут Nx (Ny) — число основних ділень з підписами під віссю X (У); nx (ny) — число проміжних ділень.

Таблиця 5.5

Значення параметра s (Scaling) рядка strf

| **Значення** | **Опис** |
|---|---|
| 0 | масштабування за умовчанням |
| 1 | встановлюється параметром |
| 2 | масштаб залежить від мінімального і максимального значення вхідних даних |
| 3 | виводяться ізометричні осі, виходячи із значень параметра |
| 4 | виводяться ізометричні осі, виходячи з вхідних даних |
| 5 | розширення осей для якнайкращого вигляду, виходячи із значень параметра |
| 6 | розширення осей для якнайкращого вигляду, виходячи з вхідних даних |

Для наочності приведемо графік поверхні Z = sin(x) • cos(y), побудований функцією plot3d1, і її зображення на горизонтальній площині, сформоване функцією contourf, в одному графічному вікні. З цією метою звернемося до команди subplot, якою розіб'ємо графічне вікно на дві області для виведення графіків.

Використовуючи feval, обчислимо значення функції Z = sin(x) • cos(y) і побудуємо її графік за допомогою plot3d1, вказавши кути огляду спостерігача — 80 і 15, а також, викликавши команду xtitle, виведемо підпис графика— «plot3d1».

Тепер сформуємо проекцію поверхні на горизонтальну площину за допомогою функції contourf. Як параметри передаємо їй X, Y і Z-координати, число ізоліній (10), 10 : 20 — масив, що визначає колір кожного інтервалу між ізолініями, а також значення рядка strf="121" (1 — режим відображення підписів; 2 — вибір масштабу залежить від мінімального і максимального значення вхідних даних; 1 — режим відображення координатних осей, вісь Y знаходиться зліва).

Звернете увагу, в цьому завданні ми вперше створили шкалу кольору. Її використання часто полегшує читання графіка. Для її виводу в Scilab існує команда colorbar (n, m), тут n — мінімальне значення диапазона,m — максимальне значення.

Таблиця 5.6

Значення параметра a (Axes) рядки strf

| **Значення** | **Опис** |
|---|---|
| 0 | немає осей |
| 1 | виводяться осі, вісь У зліва |
| 2 | виводиться рамка навколо графіка без ділень |
| 3 | виводяться осі, вісь У справа |
| 4 | осі центруються в графічній області вікна |
| 5 | осі виводяться так, щоб вони перетиналися в крапці (0;0). |

### 5.6 Функція hist3d

Для побудови тривимірних гістограм в Scilab використовується функція hist3d:  hist3d(f[theta,alpha,leg,flag,ebox])

Тут f — матриця (m : n), задаюча гістограму f (i, j) = F(x(i),y(j)).

***Контрольні питання:***

1. *Перерахувати основні функції для побудови двомірних графіків в середовищі Skilab*
2. *Перерахувати основні функції для побудови тривимірних графіків в середовищі Skilab*
3. *Назвати функцію побудови гістограми в середовищі Skilab*

# ТЕМА 6

# НЕЛІНІЙНІ РІВНЯННЯ І СИСТЕМИ В SCILAB

Якщо нелінійне рівняння достатнє складне, то відшукання його коріння — процес нетривіальний. Розглянемо, якими засобами володіє Scilab для вирішення цього завдання.

## 6.1 Рівняння алгебри

Будь-яке рівняння P(x)= 0, де P(x) — это многочлен, відмінний від нульового, називається рівнянням алгебри або поліномом. Всяке рівняння алгебри відносне x можна записати у вигляді aoxn+aixn-1 + • •+an- 1 x+an = 0, де ao = 0, n ^ 1 і aj — коефіцієнти рівняння алгебри n-й ступеня. Наприклад, лінійне рівняння це рівняння алгебри першого ступеня, квадратне — другий, кубічне, — третьою і так далі.

Вирішення рівняння алгебри в Scilab складається з двох етапів. Необхідно задати поліном P(x) за допомогою функції poly, а потім знайти його коріння, застосувавши функцію roots.

Отже, визначення поліномів в Scilab здійснює функція

poly(a, "x["fl"])

де a — це число або матриця чисел, x — символьна змінна, f — необов'язкова символьна змінна, що визначає спосіб завдання полінома. Символьна змінна f може приймати тільки два значення — «roots» або «coeff» (відповідно «r» або «c»). Якщо f=c, то буде сформований поліном з коефіцієнтами, що зберігаються в параметрі a. Якщо ж f=r, то значенні

араметра a сприймаються функцією як коріння, для якого необхідно розрахувати коефіцієнти відповідного полінома. За умовчанням f=r.

## 6.2 Трансцендентні рівняння

Рівняння /(x) = 0, в якому невідоме входить в аргумент трансцендентних функцій, називається трансцендентним рівнянням. До трансцендентних рівнянь належать показові, логарифмічні і

тригонометричні. У загальному випадку аналітичне вирішення рівняння (x) = 0 можна знайти тільки для вузького класу функцій. Найчастіше доводиться вирішувати це рівняння чисельними методами.

Чисельне вирішення нелінійного рівняння проводять в два етапи. На початку відокремлюють коріння рівняння, тобто знаходять достатньо тісні проміжки, в яких міститься тільки один корінь. Ці проміжки називають інтервалами ізоляції кореня, визначити їх можна, зобразивши графік функції /(x) або будь-яким іншим методом. На другому етапі проводять уточнення відокремленого коріння, або, інакше кажучи, знаходить коріння із заданою точністю

### 6.3 Вирішення звичайних диференціальних рівнянь

Диференціальні рівняння і системи мають нескінченну безліч рішень, які відрізняються один від одного константами. Для однозначного визначення рішення потрібно задати додаткові початкові або граничні умови. Кількість таких умов повинна співпадати з порядком диференціального рівняння або системи. Залежно від виду додаткових умов

У диференціальних рівняннях розрізняють: завдання Коші — всі додаткові умови задані в одній (частіше початковою) точці інтервалу; краєве завдання — додаткові умови вказані на межах інтервалу.

Велика кількість рівнянь може бути вирішене точно. Проте є рівняння, а особливо системи рівнянь, для яких точне рішення записати не можна. Такі рівняння і системи вирішують за допомогою чисельних методів. Чисельні методи також застосовують в тому випадку, якщо для рівнянь з відомим аналітичним рішенням потрібно знайти числове значення при певних початкових даних.

Для вирішення диференціальних рівнянь і систем в Scilab передбачена функція

[y,w,iw]=ode([type],y0,t0,t [,rtol [,atol]],f [,jac] [,w,iw])

для якої обов'язковими вхідними параметрами є: y0 — вектор початкових умов; t0 — початкова точка інтервалу інтеграції; t — координати вузлів сітки, в яких відбувається пошук рішення; f —

зовнішня функція, що визначає праву частину рівняння або системи рівнянь ; y — вектор рішень .

Таким чином, для того, щоб вирішити звичайне диференціальне рівняння виду -Jy = f (t, y), y(to)= Yo, необхідно викликати функцію y=ode(y0,t0,t,f).

Розглянемо необов'язкові параметри функції ode:

type — параметр, за допомогою якого можна вибрати метод рішення або тип вирішуваного завдання, вказавши один з рядків: adams — применяют при вирішенні диференціальних рівнянь або систем методом прогнозу-корекції Адамса; stiff — указують при вирішенні жорстких завдань; rk — використовують при вирішенні диференціальних рівнянь або систем методом Рунге — Кутта четвертого порядку; rkf — указують при виборі п'ятиетапного методу Рунге — Кутта четвертого порядку; fix — той же метод Рунге — Кутта, але з фіксованим кроком;

rtol, atol — відносна і абсолютна погрішності обчислень, вектор, розмірність якого співпадає з розмірністю вектора y, за умовчанням rtol=0.00001, atol=0.0000001, при використанні параметрів rkf і fix - rtol=0.001, atol=0.0001;

jac — матриця, що є якобиан правої частини жорсткої системи диференціальних рівнянь, задають матрицю у вигляді зовнішньої функції виду J=jak(t,y);

w, iw — вектори, призначені для збереження інформації про параметри інтеграції, які застосовують для того, щоб подальші обчислення виконувалися з тими ж параметрами.

***Контрольні питання:***

1. *Перерахувати основні функції вирішення інтегралів в середовищі Skilab*
2. *Перерахувати функції вирішення звичайних диференціальних рівнянь*
3. *Перерахувати основні функції рішення трансцендентні рівняння*

# ТЕМА 7
# ПРОГРАМІРОВАНІЄ В SKILAB

У Scilab вбудована могутня мова програмування з підтримкою об'єктів. У цьому розділі буде описані можливості структурного програмування, наступний розділ буде присвячений візуальному програмуванню в середовищі Scilab.

Як вже наголошувалося раннє, робота в Scilab може здійснюватися не тільки в режимі командного рядка, але і в так званому програмному режимі. Нагадаємо, що для створення програми (програму в Scilab іноді називають сценарієм) необхідне:

Викликати команду Editor з меню.

У вікні редактора Scipad набрати текст програми.

Зберегти текст програми за допомогою команди File - Save у вигляді файлу з розширенням sce, наприклад, file.sce .

Після цього програму можна буде викликати, набравши в командному рядку exec, наприклад, exec("file.sce"). Інші способи виклику — скористатися командою меню File - Exec... або, знаходячись у вікні Scipad, виконати команду Execute - Load into Scilab (Ctrl+L ).

Програмний режим достатньо зручний, оскільки він дозволяє зберегти розроблений обчислювальний алгоритм у вигляді файлу і повторювати його при інших початкових даних в інших сесіях. Окрім звернень до функцій і операторів привласнення, в програмних файлах можуть використовуватися оператори мови програмування Scilab (мову програмування Scilab називатимемо sci- мовою).

Основні оператори sci-языка

Вивчення sci-языка почнемо з функцій введення-виводу.

## 7.1 Функції введення-виводу в Scilab

Для організації простого введення в Scilab можна скористатися функціями

x=input('title');

або

x=x_dialog('title', 'stroka');

Функція input виводить в командному рядку Scilab підказку title і чекає, поки користувач введе значення, яке як результат повертається в змінну x. Функція x_dialog виводить на екран діалогове вікно з ім'ям title, після чого користувач може клацнути OK, і тоді stroka повернеться як результат в змінну x, або ввести нове значення замість stroka, яке і повернеться як результат в змінну x.

Функція input перетворює введене значення до числового типу даних, а функція x_dialog повертає строкове значення. Тому при використанні функції x_dialog для введення числових значень повертаний нею рядок слід перетворити в число за допомогою функції evstr. Можна запропонувати наступну форму використання функції x_dialog для введення числових значень:

x=evstr(x_dialog('title','stroka'));

Для виводу в текстовому режимі можна використовувати функцію disp наступної структури:

disp(b)

Тут b — ім'я змінної або укладений в лапки текст.

### 7.2 Оператор привласнення

Оператор привласнення має наступну структуру a=b

тут a — ім'я змінної або елементу масиву, b — значення або вираз. В результаті виконання оператора привласнення змінній a привласнюється значення виразу b.

### 7.3 Умовний оператор

Одним з основних операторів, що реалізовують галуження в більшості мов програмування, є умовний оператор if. Існує звичайна і розширена форми оператора if в Scilab. Звичайний if має вигляд

if умова операторы1 else

операторы2 end

Тут умова — логічний вираз, операторы1, операторы2 — оператори мови Scilab або вбудовані функції. Оператор if працює після наступного алгоритму: якщо умова істинна, то виконуються операторы1, якщо помилково — операторы2 .

У Scilab для побудови логічних виразів можуть використовуватися умовні оператори: &, and (логічне і) |, or (логічне або), not (логічне заперечення) і оператори відношення: < (менше), > (більше), == (рівно), ~=, <> (не рівно), <= (менше або рівно), >= (більше або рівно).

Часто при вирішенні практичних завдань недостатньо вибору виконання або невиконання однієї умови. В цьому випадку можна, звичайно, по вітці else написати нового оператора if, але краще скористатися розширеною формою оператора if.

if условие1

операторы1

elseif условие2

операторы2 elseif умова 3

операторы3

elseif умова n

операторьт

else

оператори end

В цьому випадку оператор if працює так: якщо условие1 істинно, то виконуються операторы1, інакше перевіряється условие2, якщо воно істинне, то виконуються операторы2, інакше перевіряється условие3 і так далі Якщо жодне з умов по вітках else і elseif не виконується, то виконуються оператори по вітці else.

### 7.4 Оператор альтернативного вибору

Ще одним способом організації розгалужень є оператор альтернативного вибору select наступної структури:

select параметр

case значение1 then операторы1 case значение2 then операторы2

else оператори end

Оператор select працює таким чином: якщо значення параметра рівне значениию1, то виконуються операторы1, інакше, якщо параметр рівний значениию2, то виконуються операторы2. Інакше, якщо значення параметра співпадає із значением3, то виконуються операторы3 і так далі Якщо значення параметра не співпадає ні з одним із значень в групах case, то виконуються оператори, які йдуть після службового слова else.

Звичайно, будь-який алгоритм можна запрограмувати без використання select, використовуючи тільки if, але використання оператора альтернативного вибору select робить програму компактнішої.

Оператор while

Оператор циклу while має вигляд

while умова

оператори

end

Тут умова — логічний вираз; оператори виконуватимуться циклічно, поки логічна умова істинна.

Оператор циклу while володіє значною гнучкістю, але не дуже зручний для організації «строгих» циклів, які повинні бути виконані задане число разів. Оператор циклу for використовується саме в цих випадках.

Оператор for

Оператор циклу for має вигляд

for x=xn:hx:xk

оператори

end

Тут x — ім'я скалярної змінної — параметра циклу, xn — початкове значення параметра циклу, xk — кінцеве значення параметра циклу, hx — крок циклу. Якщо крок циклу рівний 1, то hx можна опустити, і в цьому випадку оператор for буде таким.

for x=xn:xk

оператори

end

Виконання циклу починається з привласнення параметру стартового значення (ж = жп). Потім слідує перевірка, чи не перевершує параметр кінцеве значення (ж > жк). Якщо ж > жк, то цикл вважається завершеним, і управління передається наступному за тілом циклу операторові. Якщо ж ж ^ жк, то виконуються оператори в циклі (тіло циклу). Далі параметр циклу збільшує своє значення на h-ж (же = ж + h-ж). Після чого знову проводиться перевірка значення параметра циклу, і алгоритм повторюється.

Розглянемо можливості sci-языка для обробки масивів і матриць. Особливістю програмування завдань обробки масивів (одновимірних, двовимірних) на sci-языке є можливість як поелементної обробки масивів (як в будь-якій мові програмування), так і використання функцій Scilab для роботи масивами і матрицями.

Розглянемо основні алгоритми поелементної обробки масивів і матриць і їх реалізацію на sci-языке.

Розглянемо алгоритм знаходження суми, який полягає в наступному: спочатку сума рівна 0 (s = 0), потім до s додаємо перший елемент масиву і результат записуємо знову в змінну s, далі до змінної s додаємо другий елемент масиву і результат записуємо в s і далі

аналогічно додаємо до s решту елементів масиву. При знаходженні суми елементів матриці послідовно підсумовуємо елементи всіх рядків.

Алгоритм знаходження твору наступний: на першому етапі початкове значення твору рівне 1 (p = 1). Потім p послідовно умножається на черговий елемент, і результат записується в p і т. д

Пошук максимального (мінімального) елементу масиву (матриці)

Алгоритм рішення задачі пошуку максимуму і його номера в масиві наступний. Хай в змінній з ім'ям Max зберігається значення максимального елементу масиву, а в змінній з ім'ям Nmax — його номер. Припустимо, що перший елемент масиву є максимальним і запишемо його в змінну Max, а в Nmax — його номер (1). Потім всі елементи, починаючи з другого, порівнюємо в циклі з максимальним. Якщо поточний елемент масиву виявляється більше максимального, то записуємо його в змінну Max, а в змінну Nmax — поточне значення індексу i.

Алгоритм пошуку мінімального елементу в масиві відрізнятиметься від приведеного вище лише тим, що в операторові if знак поміняється з «>» на «<».

При пошуку мінімального (максимального) елементу матриці цикли по i і j починаються з 1. Якщо написати з двох, то при обробці елементів буде пропущений перший рядок або перший стовпець при порівнянні a,i,j з min.

### 7.5 Сортування елементів масиву

Сортування є процесом впорядковування елементів масиву в порядку зростання або убування їх значень. Наприклад, масив X з n елементів буде відсортований в порядку зростання значень його елементів, якщо

X [1] < X [2] < • < X [n], і в порядку убування, якщо

X[1]> X[2]> • > X[n].

Розглянемо найбільш відомий алгоритм сортування методом бульбашки. Порівнянний перший елемент масиву з другим, якщо

перший опиниться більше другого, то поміняємо їх місцями. Ті ж дії виконаємо для другого і третього, третього і четвертого ..., і-го і (і + 1) -го ..., (n — 1)-го і n-го елементів. В результаті цих дій найбільший елемент стане на останнє (n-е) місце. Тепер повторюваний даний алгоритм спочатку, але останній (n-й) елемент розглядати не будемо, оскільки він вже зайняв своє місце. Після проведення даної операції найбільший елемент масиву, що залишився, стане на (n — 1) -е місце, далі слід повторити алгоритм до тих пір, поки не упорядкуємо масив.

Алгоритм сортування по убуванню відрізнятиметься від приведеного заміною знаку «>» на «<».

### 7.6 Видалення елементу з масиву

Необхідно видалити з масиву x, що складається з n елементів, m-й по номеру елемент. Для цього досить записати (m + 1) -й елемент на місце елементу з номером m, (m + 2) -й, — на місце (m + 1) -го ..., n-й — на місце (n — 1) -го, після чого видалити останній n-й елемент

***Функція відкриття файлу mopen***

Як і в будь-якій іншій мові програмування, робота з файлом починається з його відкриття. Для відкриття файлу в sci-языке призначена функція mopen, яка має вигляд

[fd,err]=mopen(file,mode) file — строка, в якій зберігається ім'я файлу, mode — режим роботи з файлом:

'r' — текстовий файл відкривається в режимі читання

'rb' — двоичный файл відкривається в режимі читання

'w' — открывается порожній текстовий файл, який призначений тільки для запису інформації;

'wb' — відкривається порожній двійковий файл, який призначений тільки для запису інформації;

'a' — відкривається текстовий файл, який використовуватиметься для додавання даних в кінець файлу; якщо файлу немає, він буде створений;

'ab' — відкривається двійковий файл, який використовуватиметься для додавання даних в кінець файлу; якщо файлу немає, він буде створений;

'r+' — відкривається текстовий файл, який використовуватиметься в режимі читання і запису;

'rb+' — відкривається двійковий файл, який використовуватиметься в режимі читання і запису;

'w+' — створюваний порожній текстовий файл призначений для читання і запису інформації;

'wb+' — створюваний порожній двійковий файл призначений для читання і запису інформації;

'a+' — текстовий файл, що відкривається, використовуватиметься для додавання даних в кінець файлу і читання даних; якщо файлу немає, він буде створений;

'ab+' — двійковий файл, що відкривається, використовуватиметься для додавання даних в кінець файлу і читання даних; якщо файлу немає, він буде створений.

Функція шореп повертає ідентифікатор відкритого файлу fd і код помилки егг. Ідентифікатор файлу — ім'я (код), по якому описані нижче функції звертатимуться до реального файлу на диску. У змінною егг повертається значення 0 у разі вдалого відкриття файлу. Якщо егг = 0, то це означає, що файл відкрити не вдалося.

### 7.7 Функція читання даних з текстового файлу mfscanf

При прочитуванні даних з файлу можна скористатися функцією mfscanf наступного вигляду:

A=mfscanf(f, sl)

Тут f — ідентифікатор файлу, який повертається функцією шорєп, sl — рядок форматів вигляду

%[ширина][.точность]тип

Функція mfscanf працює таким чином: з файлу з ідентифікатором f прочитуються в змінну A значення відповідно до формату sl. При читанні числових значень з текстового файлу слід пам'ятати, що два числа вважаються розділеними, якщо між ними є хоч би один пропуск, символ табуляції або символ переходу на новий рядок.

При прочитуванні даних з текстового файлу користувач може стежити, чи досягнутий кінець файлу, за допомогою функції шєо^^ ^ — ідентифікатор файлу), яка повертає одиницю, якщо досягнутий кінець файлу, і нуль інакше

Таблиця 7.1

Значення параметрів рядка перетворення

| **Параметр** | **Призначення** |
|---|---|
| **Прапор** | |
| - | Вирівнювання числа вліво. Права сторона доповнюється пропусками. За умовчанням вирівнювання управо. |
| + | Перед числом виводиться знак «+» або «-» |
| Пропуск | Перед позитивним числом виводиться пропуск, перед негативним — « - » |
| # | Виводиться код системи числення: 0 — перед вісімковим, 0x (0X) — перед шістнадцятиричним числом. |
| **Ширина** | |
| n | Ширина поля виводу. Якщо п позицій недостатньо, то поле виводу розширюється до мінімально необхідного. Незаповнені позиції заповнюються пропусками. |
| On | Те ж, що і п, але незаповнені позиції заповнюються нулями. |

| **Точність** | |
|---|---|
| нічого | Точність за умовчанням |
| n | Для типів e, E, f виводити n знаків після десяткової крапки |
| **Тип** | |
| з | При введенні символьний тип char, при виводі один байт. |
| d, i | Десяткове із знаком |
| i | Десяткове із знаком |
| o | Вісімкове int unsigned |
| u | Десяткове без знаку |
| x, X | Шістнадцятиричне int unsigned, при x використовуються символи a-f, при X—a-f. |
| f | Значення із знаком виду [-]dddd.dddd |
| e | Значення із знаком виду [-]d.ddde[+\|-]ddd |
| E | Значення із знаком виду [-]d.ddddE[+\|-]ddd |
| g | Значення із знаком типу e або f залежно від значення і точності |
| G | Значення із знаком типу E або F залежно від значення і точності |
| s | Рядок символів |
| **Модифікатор (типу)** | |
| h | Для d, i, про, u, x, X коротке ціле |
| l | Для d, i, про, u, x, X довге ціле |

### 7.8 Функція закриття файлу mclose

Після виконання всіх операцій з файлом він повинен бути закритий за допомогою функції mclose наступної структури:

mclose(f)

Тут f — идентификатор файлу, що закривається. За допомогою функції mclose('all') можна закрити відразу всі відкриті файли, окрім стандартних системних файлів.

Таблиця 7.2

Деякі спеціальні символи

| Символ | Призначення |
|---|---|
| \b | Зрушення поточної позиції вліво |
| \n | Переклад рядка |
| \r | Переклад в початок рядка, не переходячи на новий рядок |
| \t | Горизонтальна табуляція |
| \' | Символ одинарної лапки |
| \" | Символ подвійної лапки |
| \? | Символ «?» |

### 7.9 Функції в Scilab

У Scilab нескладно оформляти власні функції. Структура функції в Scilab наступна:

function [y1,y2...,yn]=ff(x1,x2...,xm)

оператори

endfunction

Тут x1, x2 ..., xm — список вхідних параметрів функції; y1, y2 ..., yn — список вихідних параметрів функції, ff — ім'я функції.

Якщо функція, що викликається, знаходиться не в поточному файлі, то перед її викликом слід завантажити файл, в якому знаходиться функція за допомогою функції exec наступної структури.

exec('file'-1).

Тут file — имя файлу на диску, в якому знаходиться функція, що викликається. Розглянемо приклад з використанням функцій і файлів.

### 7.10 Створення графічних застосувань в середовищі Scilab

Scilab дозволяє створювати не тільки звичайні програми для автоматизації розрахунків, але і візуальні застосування, які запускатимуться в середовищі Scilab. Основним об'єктом в середовищі Scilab є графічне вікно.

***Робота з графічним вікном***

Для створення порожнього графічного вікна служить функція figure. F=figure();

В результаті виконання цієї команди буде створено графічне вікно з ім'ям objfigurel. За умовчанням перше вікно отримує ім'я objfigurel, друге, — objfigure2 і так далі Покажчик на графічне вікно записується в змінну F. Розмір і положення вікна на екрані комп'ютера можна задавати за допомогою параметра

'position'[x y dx dy]

де x, y — положення верхнього лівого кута вікна (по горизонталі і вертикалі відповідно) щодо верхнього лівого кута екрану;

dx — розмір вікна по горизонталі (ширина вікна) в пікселях;

dy — розмір вікна по вертикалі (висота вікна) в пікселях.

Параметри вікна можна задавати одним з двох способів.

1. Безпосередньо при створенні графічного вікна задаються його параметри. В цьому випадку звернення до функції figure має вигляд

F=figure('Свойство1', 'Значеніє1', 'Свойство2', 'Значеніє2' ..., 'Свойство^, 'Значеніє^)

тут 'Свойство1' — название першого параметра, Значеніє1 — его значення, 'Свойство2'—названіє другого параметра, Значеніє2— значеніє другого параметра і так далі

Динамічне створення інтерфейсних елементів. Опис основних функцій

У Scilab використовується динамічний спосіб створення інтерфейсних компонентів. Він полягає в тому, що на стадії виконання програми можуть створюватися (і віддалятися) ті або інші елементи управління (кнопки, влучні, прапорці і т. д.) і їх властивостям привласнюються відповідні значення.

Для створення будь-якого інтерфейсного компоненту із заданими властивостями використовується функція uicontrol, що повертає покажчик на формований компонент:

C=uicontrol(F, 'Style', 'тип_компонента', 'Свойство_1', Значеніє_1, 'Свойство_2', Значеніє_2..., 'Свойство_к',значеніє_к);

Тут C — покажчик на створюваний компонент;

F — покажчик на об'єкт, усередині якого створюватиметься компонент (найчастіше цим компонентом буде вікно); перший аргумент функції uicontrol не є обов'язковим, і якщо він відсутній, то батьком (власником) створюваного компоненту є поточний графічний об'єкт — поточне графічне вікно;

'Style' — служебная рядок Style, указує на стиль створюваного компоненту (символьне ім'я);

'тип_компонента' — визначає, до якого класу належить створюваний компонент, це може бути PushButton, Radiobutton, Edit, StaticText, Slider, Panel, Button Group, Listbox або інші компоненти, ця властивість указуватиметься для кожного з компонентів;

'Свойство_к', Значеніє_к — определяют властивості і значення окремих компонентів, вони будуть описані нижчим конкретно для кожного компоненту.

У існуючого інтерфейсного об'єкту можна змінити ті або інші властивості за допомогою функції set:

set(C,'Свойство_1',Значение_1, 'Свойство_2', Значеніє_2 ..., 'Свойство_к', Значеніє_к)

Тут С — покажчик на динамічний компонент, параметри якого мінятимуться. С може бути і вектором динамічних елементів, в цьому випадку функція set задаватиме значення властивостей для всіх об'єктів С(i);

'Свойство_к', Значеніє_к — изменяемые параметри і їх значення.

Набути значення параметра компонентів можна за допомогою функції get наступної структури:

get(C,'Свойство')

Тут С — покажчик на динамічний інтерфейсний компонент, значення параметра якого необхідно дізнатися;

'Властивість' — имя параметра, значення якого потрібно дізнатися. Функція повертає значення параметра.

Далі ми поговоримо про особливості створення різних компонентів.

### 7.11 Командна кнопка

Командна кнопка типу PushButton створюється за допомогою функції uicontrol, в якій параметру 'Style' необхідно привласнити значення 'pushbutton'. За умовчанням вона не забезпечується ніяким написом, має сірий колір і розташовується в лівому нижньому кутку фігури, Напис на кнопці можна встановити за допомогою властивості String.

### 7.12 Влучна

Наступним найбільш часто використовуваним компонентом є мітка — текстове поле для відображення символьної інформації. Для визначення мітки значення параметра 'Style' у функції uicontrol повинно мати значення 'text'. Компонент призначений для виведення символьного рядка (або декількох рядків). Текст, що виводиться на мітку, — значення параметра 'String' — може бути змінений тільки з програми

Однією з основних властивостей мітки є горизонтальне вирівнювання тексту, яке визначається властивістю HorizontalAlignment. Ця властивість може приймати одне з наступних значень:

left — выравнивание тексту по лівому краю;

center — выравнивание тексту по центру (значення за умовчанням);

right — выравнивание по правому краю.

Як приклад розглянемо вікно, що містить 4 текстових поля з різними значеннями властивості HorizontalAlignment.

### 7.13 Компоненти Перемикач і Прапорець

Розглянемо ще два компоненти — перемикач і прапорець, які дозволяють перемикатися між станами або вимикати одну з властивостей.

У прапорця властивість 'Style' приймає значення 'checkbox', у перемикача властивість 'Style' повинна бути встановлене в 'radiobutton'

### 7.14 Компонент вікно редагування

Інтерфейсний елемент вікно редагування (у того компоненту властивість 'Style' повинна приймати значення 'edit') може використовуватися для введення і виведення символьної інформації. Текст, що набирає у вікні редагування, можна коректувати. При роботі з компонентом можна використовувати операції з буфером обміну.

Процедура введення, що завершується натисненням клавіші Enter, генерує подію CallBack.

Рядок введення визначається параметром 'String', який визначає текст, що знаходиться в компоненті. Для нормального функціонування компоненту цей параметр необхідно обов'язково задавати при визначенні компоненту за допомогою функції uicontrol. Змінити значення цієї властивості можна за допомогою функції set, а рахувати його значення — за допомогою функції get.

Текст, що вводиться, може бути притиснутий до лівого або правого краю вікна введення, якщо задати відповідне значення властивості HorizontalAlignment (по аналогії з компонентом «Влучна»). Якщо текстом, що вводиться, є числове значення, яке повинне бути використане в роботі програми, то вміст властивості 'String' переводиться в числовий формат за допомогою функції eval (можна було скористатися і функцією evstr) (буде розглянуто далі на прикладі квадратного рівняння).

### 7.15 Списки рядків

Інтерфейсний компонент «список рядків» в простому випадку можна розглядати як вікно з масивом рядків в нім. Якщо довжина списку перевищує висоту вікна, то для переміщення за списком може використовуватися вертикальна смуга прокрутки, яка генерується автоматично.

### 7.16 Вирішення диференціальних рівнянь в приватних похідних

Математичні моделі фізичних і інших процесів описуються за допомогою диференціальних рівнянь в приватних производных1. Аргументами функцій цих рівнянь є просторові координати x, y, г і час Загальні відомості про диференціальні рівняння в приватних похідних приведені в першому параграфі розділу. У skilab, як і в більшості математичних пакетів, немає засобів для безпосереднього вирішення рівнянь математичної фізики. Проте можливості пакету досить, для реалізації методу сіток вирішення диференціальних рівнянь в приватних

похідних. У подальших параграфах цього розділу і описана реалізація методу сіток для вирішення параболічних, гіперболічних і еліптичних рівнянь в skilab.

Загальні відомості про диференціальні рівняння в приватних похідних

Лінійним рівнянням в приватних похідних другого порядку називається співвідношення між функцією u(x,y) (або u(x,t)) і її приватними похідними вигляду [1]:

$$L(u) = A(x,y)\frac{\partial^2 u}{\partial x^2} + 2B(x,y)\frac{\partial^2 u}{\partial x \partial y} + C(x,y)\frac{\partial^2 u}{\partial y^2} + \\ + D(x,y)\frac{\partial u}{\partial x} + E(x,y)\frac{\partial u}{\partial y} + G(x,y)u(x,y) = F(x,y) \quad (7,1)$$

ХВ літературі ці рівняння часто називають рівняннями математичної фізики.

Якщо змінна функція u залежить від x і t, то рівняння може бути записане таким чином:

$$L(u) = A(x,t)\frac{\partial^2 u}{\partial x^2} + 2B(x,t)\frac{\partial^2 u}{\partial x \partial t} + C(x,t)\frac{\partial^2 u}{\partial t^2} + \\ + D(x,t)\frac{\partial u}{\partial x} + E(x,t)\frac{\partial u}{\partial t} + G(x,t)u(x,t) = F(x,t)$$

(7,2)

У випадку якщо в правій частині рівняння F = 0, то рівняння називаються однорідними, інакше — неоднорідними .

Якщо B2 — 4AC < 0, те рівняння відноситься до класу еліптичних рівнянь, якщо B2 — 4AC > 0, то це — гіперболічне рівняння, якщо B2 — 4AC = 0 — параболічне рівняння. У разі, коли B2 — 4AC не має постійного знаку, то це — рівняння змішаного типу.

За допомогою перетворення змінних x, y (або x, t) рівняння можна привести до вигляду, коли B = 0. В цьому випадку дуже просто визначається тип рівняння. Якщо A і З мають один і той же знак, то

рівняння (7.2) — еліптичне рівняння, якщо разные, то гіперболічне, а якщо A або З рівне 0, то рівняння відноситься до параболічних .

До класичних еліптичних рівнянь відносяться :

- рівняння Лапласа $\Delta u=0$[1], яке використовується для опису магнітних і стаціонарних теплових полів;
- рівняння Пуассона $\Delta u=f$, яке застосовується в електростатиці, теорії пружності і т. д.;
- рівняння Гельмгольца Пуассона $\Delta u$+си=$f$, що описує сталі коливальні процеси.

### 7.17 Вирішення завдань лінійного програмування

Ще одним оптимізаційним завданням, що часто зустрічається в практиці, є завдання лінійного програмування. Знайомство із завданнями лінійного програмування почнемо на прикладі завдання про оптимальний раціон.

Завдання про оптимальний раціон. Є чотири види продуктів харчування: П1, П2, П3, П4. Відома вартість одиниці кожного продукту С1, С2, С3, С4. З цих продуктів необхідно скласти харчовий раціон, який повинен містити не менше b1 одиниць білків, не менше b2 одиниць вуглеводів, не менше b3 одиниць жирів. Причому відомо, що в одиниці продукту П1 міститься an одиниць білків, a12 одиниць вуглеводів і a13 одиниць жирів і так далі (див. таблицю 7.3 ).

Потрібно скласти харчовий раціон, щоб забезпечити задані умови при мінімальній вартості.

Таблиця 7.3

Вміст білків, вуглеводів і жирів в продуктах

| Елемент | білки | вуглеводи | жири |
|---|---|---|---|
| П1 | *a11* | *a12* | *a13* |
| П2 | *a21* | *a22* | *a23* |

| | | | |
|---|---|---|---|
| П3 | *a31* | *a32* | *a33* |
| П4 | *a41* | *a42* | *a43* |

Хай x1, x2, x3, x4 — количества продуктів П1, П2, П3, П4. Загальна вартість раціону рівна

$$L = C1x1 + C2x2 + C_3X_3 + C_4X_4 = \sum CiXi \qquad (7.3)$$

Сформулюємо обмеження на кількість білків, вуглеводів і жирів у вигляді нерівностей. У одній одиниці продукту П1 міститься оц одиниць білків, в X1 одиницях *a11x1* —, в $_{X2}$ одиницях продукту П2 міститься *a21x2* одиниць білка і так далі Отже, загальна кількість білків у всіх чотирьох типах продукту рівна xз і повинно бути не менше b1. Отримуємо перше обмеження:

$$a11x1 + a21x2 + a31x3 + a41x4 > b1 \qquad (7.4)$$

Аналогічні обмеження для жирів і вуглеводів мають вигляд:

$$a12x1 + a22x2 + a32x3 + a42x4 > b1 \qquad (7.5)$$

$$a13x1 + a22x2 + a32x3 + a42x4 > b1 \qquad (7.6)$$

Взявши до уваги, що xl, x2, xз, x4 — положительные значення, отримаємо ще чотири обмеження:

$$x1 > 0,\ x2 > 0,\ x3 > 0,\ x4 > 0 \qquad (7.7)$$

Таким чином, завдання про раціональний раціон можна сформулювати таким чином: знайти значення змінних $_{Xl,\ X2}$, xз, $_{X4}$, що задовольняють системі обмежень при яких лінійна функція (13.1) приймала б мінімальне значення.

Завдання про оптимальний раціон є завданням лінійного програмування, функція називається функцією мети, а обмеження системою обмежень завдання лінійного програмування.

У завданнях лінійного програмування функція мети L і система обмежень є лінійними.

У загальному випадку завдання лінійного програмування можна сформулювати таким чином. Знайти такі значення $x_1$, $x_2$..., $x_n$, що задовольняють системі обмежень, при яких функція мети L досягає свого мінімального (максимального) значення:

$$\sum a_{ij}x_j \le fb_i \; i=1..., m \qquad x_i \ge 0 \qquad (7.7)$$

$$L = C1x1 + C2x2 + ... + Cпхп = \sum CiXi \qquad (7.8)$$

Для вирішення завдань лінійного програмування в skilab призначена

функція linpro наступної структури:

[x,kl,f]=linpro(c,A,b[,ci,cs][,k][,x0])

Тут з — масив (вектор-стовпець) коефіцієнтів при невідомих функції мети, довжина вектора n співпадає з кількістю невідомих X.

A — матриця при невідомих з лівої частини системи обмежень, кількість рядків матриці рівна кількості обмежень m, а кількість стовпців співпадає з кількістю невідомих п.

b — массив (вектор-стовпець), містить вільні члени системи обмежень, довжина вектора m.

*ci* — массив (вектор-стовпець) розмірності п містить нижню межу змінних (*cij* ] Xj); якщо така відсутня, указують [].

*cs* — массив (вектор-стовпець) довжиною п, містить верхню межу змінних (csj ]$X_j$); якщо така відсутня, указують [].

*до* — цілочисельна змінна, використовується, якщо в систему обмежень окрім нерівностей входить і рівність, в матриці вони знаходитимуться в до перших рядках, що залишилися *l* рядків займуть нерівності, тобто $m = до + l$.

x0 — вектор-стовпець початкових наближень довжиною n.

***Контрольні питання:***

1. *Назвати способи введення готової програми ski в робочу область Skilab*

*2. Перерахувати команди які можна використовувати в середовищі Skilab*

*3. Назвати команди для створення графичесского вікна в середовищі Skilab*

*4. Як викликати вікно програмирования в середовищі Skilab*

**Додатки**

# ЛАБОРАТОРНИЙ ПРАКТИКУМ

## Лабораторна робота №1

## Знайомство з системою програмування Scilab

### Зміст частини 1:

• Як запустити Scilab?

• Як покинути сесію Scilab?

• Де знаходяться відомості про ініціалізацію Scilab?

• Як змінити шрифт у вікні Scilab?

• Як помістити частину тексту з вікна Scilab в текстовий редактор або який-небудь інший пакет?

• Як редагувати командний рядок?

• Як синтаксично перенести довгий командний рядок на наступний рядок?

• Чи розрізняються програмою рядкові і заголовні букви?

• Як подивитися демонстраційні приклади?

• Як дізнатися ім'я поточного каталога в Scilab?

• Як змінити поточний каталог в Scilab?

• Як запустити програму на виконання?

• Як отримати довідку?

• Як розуміти формули синтаксису команд?

• Як дізнатися версію пакету Scilab?

• Як зробити коментар?

• Як використовувати Scilab як простого калькулятора?

• Як отримати список поточних імен змінних ?

• Як записати результат сесії Scilab у файл?

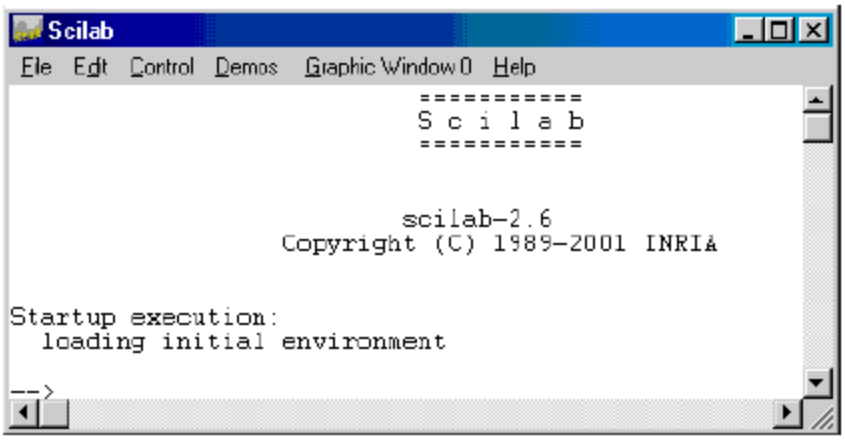
• Як виконувати операції?

• Як використовувати файл сценарію?

# 1. Перші кроки

## Як запустити Scilab?

З командного рядка наберіть scilab\bin\runscilab.exe В результаті отримаєте головне робоче вікно Scilab із запрошенням -->:




Зауваження: У версіях Scilab для операційних систем Linux і Windows можуть бути невеликі відмінності в меню і підміню верхній панелі управління вікна.

## Як покинути сесію Scilab?

Набрати exit і натиснути [Enter] або набрати quit і натиснути [Enter] або вибрати на панелі управління вікна File-Exit

## Де знаходяться відомості про ініціалізацію Scilab?

Початкові установки Scilab знаходяться у файлі \scilab\scilab.star. Напевно краще за нього не чіпати...

## Як змінити шрифт у вікні Scilab?

За допомогою верхньої панелі вікна виконати Edit-Choose Font і далі виберіть бажану гарнітуру і розмір шрифту.

## Як помістити частину тексту з вікна Scilab в текстовий редактор або який-небудь інший пакет?

Виділите потрібний текст мишею, потім [Edit][Copy to Clipboard]. Потім у Вашому пакеті виконаєте в потрібному місці [Paste] . Примітка: Звична заміна вибору режим сміттю як ctrl-c і режиму Paste як ctrl-v тут не спрацьовує. Проте, у вікні допомоги (Help) ці "гарячі" кнопки працюють.

## Як редагувати командую рядок?

Подивитеся Edit-History, або можна набрати текст в будь-якому Windows-приложении і перенести його через Clipboard у вікно Scilab. ( Примітка: російська мова підтримується) Рекомендації режиму Edit-History знадобляться, мабуть, тільки при роботі під платформою LINUX. Для набору тексту російською мовою виберіть у вікні scilab режим Edit-ChooseFont і далі потрібний шрифт, наприклад, Courier.

## Як синтаксично перенести довгий командний рядок на наступний рядок?

У місці перенесення додати дві крапки. Приклад. plotframe(rect,tics[%t,%t].. ["My plot with grids and automatic bounds","x","y"],[0,0.5,0.5,0.5]) замість plotframe(rect,tics[%t,%t]["My plot with grids and automatic bounds","x","y"][0,0.5,0.5,0.5])

## Чи розрізняються програмою рядкові і заголовні букви?

Так, і не тільки в LINUX, але і в Windows. Якщо ви визначили значення величини "x", значення величини "X" їй не рівне! Приклад. ->x=5; X=3; =>y=x+X+x y = 13. //то є 5+3+5

## Як подивитися демонстраційні приклади?

Виберіть в рядку меню Demos і слідуйте подальшим вказівкам. Це відповідає виконання в командному режимі exec('SCI/demos/alldems.dem'); Демонстрації відкриватимуться в новому вікні. Після закінчення проглядання його слід закрити.

## Як дізнатися ім'я поточного каталога в Scilab?

За допомогою меню File - GetCurrentDirectory. Це еквівалентно командному рядку -->getcwd() Результат: ans = D:\scilab\bin Аналогічний результат дає і команда pwd(). Затверджується, що pwd друкує поточний каталог Scilab, а getcwd повертає його. (Я не відчула різниці). Синтаксис цих команд: pwd x=pwd() x=getwcd() Приклади. pwd x=pwd()

## Як змінити поточний каталог в Scilab?

За допомогою команди chdir. Синтаксис ierr=chdir('path-name') Параметри ierr : ціле число, рівне 1, якщо команда не може змінити каталог помилка, і 0 в решті всіх випадків. path-name : назва нового каталога Приклад. f= chdir("d:\my_scilab") Результат: f=0.

Зауваження: 1) Якщо ми хочемо вказати ім'я підкаталогу пакету Scilab, то місце знаходження самого пакету можна не указувати, замінивши його на

конструкцію SCI+. Приклад. chdir(SCI+"/demos") еквівалентно chdir("c:/scilab/demos")

2) Якщо каталог не існує, у мене чомусь з'являється повідомлення про помилку і ierr=240 (Error 998). У якому ж випадку ierr буде =1?

## Як запустити програму на виконання?

За допомогою меню File-Exec або командою exec.

## Як отримати довідку (help)?

Є три підміню в меню Help.

1. Режим Help-HelpDialog. Допустимо, ми хочемо дізнатися синтаксис операції "exec". Нижня частина таблиці дозволяє вибрати розділ, що цікавить нас. Це буде глава "Scilab Programming". Тепер виберіть з верхнього вікна операцію "exec"и натисніть кнопку Show. В результаті отримаємо довідкову інформацію по цій операції.
2. Режим Help-Topic дозволяє отримати довідку прямо по назві команди.
3. Режим Help-Apropos дозволяє отримати довідку по ключовому слову, що задається. Використовується, коли Ви забули назву потрібної команди.

Ще одна можливість: у командному рядку після запрошення --> набрати help <ім'я команди>. Еквівалентно Help-Topic. Приклад. help exec Зауваження: Для того, щоб дізнатися в яких каталогах присутні відповідні тексти для help, служить змінна %helps. Значення %helps за умовчанням встановлюється у файлі scilab.star.

## Як розуміти формули синтаксису команд?

Повний синтаксис команд можна отримати за допомогою команди help з параметрами. Параметри, вказані в квадратних дужках не є обов'язковими. Приклад. Хай ми хочемо дізнатися, як користуватися командою обчислення синуса sin. Виконаєте help sin. У формулі синтаксису [t]=sin(x) параметр x - обов'язковий, а параметр t може не бути і присутнім. Іноді синтаксис команди не укладається в одну формулу і представляється декілька варіантів її застосування. Як приклад подивитеся help rand.

## Як дізнатися версію пакету scilab?

Виконаєте команду version=getversion() Результатом може бути:

version = scilab-2.6

## Як зробити коментар?

Коментарі повинні починатися також, як і в мові З, з конструкції //. Це застосовується зазвичай при виконанні програм в Scilab. Приклад. --> // It is plot of 2 functions --> z=3456; //It is my number

## Як використовувати Scilab як простого калькулятора?

Ввести цифровий вираз і натиснути [Enter]. Результатом буде: ans=<ответ> Приклад 1. 2+3^2 ans = 11. Приклад 2. a=1; b=7; x=a+b

Буде отримана відповідь x = 8. Зауваження: Якщо в кінці рядка коштує знак "крапки з комою"(;), то ця операція виконується не відразу, а після набору всієї послідовності команд до тієї, після якої вже не коштує крапка з комою.

## Як отримати список поточних імен змінних ?

Використовуйте команду who. Замість who можна писати who(). Результат той же. Приклад. a=3; b=5;

who your variables are...

b a startup ierr demolist

%scicos_display_mode scicos_pal

%scicos_menu %scicos_short

%helps MSDOS home PWD TMPDIR percentlib soundlib xdesslib utillib tdcslib siglib s2flib roblib optlib metalib elemlib commlib polylib autolib armalib alglib intlib mtlblib WSCI SCI

%F %T %z %s %nan %inf $

%t %f %eps %io %i %e using 5866 elements out of 1000000. and 48 variables out of 1791

Змінні, вказані в другому рядку і нижче, є вбудованими. who('local') дозволяє побачити всі імена вбудованих змінних. Наприклад, видно, що існує змінна з ім'ям %e. Виконання команди %e [Enter] дасть нам її значення

%e = 2.7182818

## Як записати результат сесії Scilab у файл?

Спосіб 1. Ми хочемо зберегти отримані у вікні Scilab запису у файлі. Для цього виконаємо команду save('my_file'). У простому випадку для цього можна використовувати меню File-Save. В результаті файл запишеться в поточний каталог. За умовчанням це каталог scilab\bin\. Розширення створюваного файлу за умовчанням *.bin. Отриманий файл надалі можна завантажити за допомогою команди load. Спосіб 2. За допомогою команди diary. Це журнал для запису копії поточної сесії Scilab в текстовий файл. Синтаксис diary('file-name') Приклад. diary('D:/slon/my_example.txt') a=5; // Далі йдуть команди сесії Scilab. b=6 c=a*b+3 diary(0) //Это ознака кінця запису сесії Все, що знаходиться в головному вікні Scilab між командою diary('file-name'), що відкриває запис сесії, і командою diary(0), що закриває запис, буде записано у файл. Зауваження: Зміст графічних вікон при цьому не зберігаються.

## Як виконувати операції?

Якщо ми хочемо, щоб кожна операція виконувалася безпосередньо відразу, слідує після запрошення -> набрати операцію і клавішу введення [Enter]. Можна набирати операції послідовно, розділяючи їх знаком ";", а потім виконати [Enter]. Тоді на екрані буде виведений тільки кінцевий результат. Можна писати після запрошення в одному командному рядку декілька операцій, розділяючи їх крапкою з комою. Тоді результати тих конструкцій, які закінчуються знаком ";" не виводитимуться на екран. Ми побачимо кінцевий результат виконання послідовності декількох командних рядків, остання з яких не закінчується крапкою з комою. Результати проміжних операцій на екран виводитися не будуть.

Приклад 1. -->a=2 a = 2. -->b=3 b = 3. -->c=a+b з = 5.

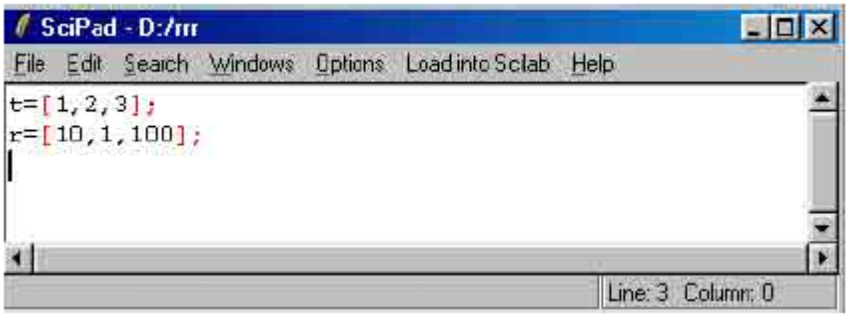
Приклад 2. -->a=2;

-->b=3; -->c=a+b; -->d=c*10 d = 50. Якщо ми хочемо дізнатися, чому було рівне значення змінної "c", ця можливість нами не втрачена: -->c з = 5. Зауваження по синтаксису: Якщо командний рядок дуже довгий, то її можна розбити на дві, додаючи в місці розбиття дві крапки (..).

Приклад. Два записи, що приводять до ідентичного результату: 1) -->plot2d(x[sin(x) sin(2*x) sin(3*x)].. -->[1,2,3],"111","L1@L2@L3"[0-2,2*%pi,2][2,10,2,10]) 2) -->plot2d(x,[sin(x) sin(2*x) sin(3*x)],[1,2,3],"111","L1@L2@L3"[0-2,2*%pi,2][2,10,2,10])

## Як використовувати файл сценарію?

Сценарій є зручним засобом введення команд в пакет Scilab . У файлі сценарію можна записати ряд послідовних операцій, які відразу підвантажуватимуться в Scilab блоком . Це альтернатива набору команд безпосередньо в головному вікні Scilab в режимі командного рядка. У версії 2.7 додатково створений спеціальний редактор SciPad для створення файлу сценарію. Для виклику цього редактора слід виконати з основного вікна Scilab за допомогою меню верхньої панелі File-Editor. В результаті отримаєте вікно редактора Scipad:




Після набору тексту сценарію можливі два способи дії: 1) За допомогою меню верхньої панелі вікна Load into Scilab текст сценарію відразу буде завантажений в сесію Scilab. Цей шлях можливий тільки для версії 2.7 і використання редактора SciPad. 2) За допомогою команд меню верхньої панелі вікна Scipad-редактора File - Save записати у файл сценарію з розширенням sce, який може бути завантажений пізнішим в сесію Scilab з головного вікна за допомогою команд меню File-Exec для файлів з розширенням sce або sci і для завантаження функцій (з расширеним sci) за допомогою команд меню File-Getf. У ранішій версії пакету 2.6 файл сценарію можна створити в довільному текстовому редакторові і використовувати спосіб 2) для введення його в Scilab.

**Зміст частини 2:**

- Спеціальні константи
- Скалярні величини
- Числові матриці
- Скалярні величини

- Вектора
- Як створити двовимірну матрицю?
- Як створити багатовимірну матрицю?
- Які існують функції формування масивів "спеціального вигляду"?
- Як сформувати одиничну матрицю (всі елементи нульові окрім діагональних)?
- Як швидко створити матрицю, всі елементи якої =1?
- Як швидко створити матрицю, всі елементи якої =0?
- Як створити рядок, елементи якого будуть послідовними цілими числам від N до M?
- Генератор випадкових чисел
- Які операції можна виконати над числовими матрицями?
- Як знайти суму елементів матриці?
- Як знайти твір елементів матриці?
- Нечислові (символьні) матриці
- Які дії можна здійснювати над символьними матрицями?
- Поліноми і полиномные матриці (уявлення алгебри)
- Як визначити поліном алгебри?
- Як отримати полиномную дріб?
- Як проводити операції в символьному вигляді?
- Як створити символьну матрицю, елементи якої є поліномами?
- Булеві матриці
- Цілочисельні матриці
- Які типи цілочисельних даних (integer) визначені в Scilab?

- Як перетворити дані з одного типу змінних в іншій?
- Тип даних "список"
- Як дізнатися тип змінної?
- Як дізнатися тип об'єкту Scilab?
- N-мерные масиви
- Як визначити N-мерный масив?
- Як визначити розмір масиву?
- Функції (макроси)
- Як створити функцію засобами пакету Scilab?
- Бібліотеки
- Об'єкти
- Операції над матрицями
- Індексування
- Як здійснюється індексування елементів в матрицях?
- Точність обчислень і визначення формату виведення числового результату

**Спеціальні константи**

**Скалярні величини**

Визначені в Scilab стандартні скалярні змінні починаються із знаку %**.** Частина спеціальних змінних зумовлена. Вони захищені і не можуть бути видалені користувачем (але можуть бути перевизначені).

**%i** Уявна одиниця: sqrt(-1)=%i =i

**%pi** Число Pi =3.1415927

**%e** Число e = 2.7182818

**%eps** Це умовний нуль, тобто таке максимальне число, що 1+%eps=1 %eps=2.220E-16

**%inf** Беськонечность=inf

**%nan** NotANumber: неопределено= -Nan

**%s** Змінна, значення якої рівне "s",т.е.%s=s або s=poly(0,"s")

**%t** Булеве true=T

**%f** Булеве false=F

**~%t** Заперечення true=F

**~%f** Заперечення false=T

*Зауваження:* Користувач може визначати свої скалярні величини і без знаку **%**.

**Числові матриці**

**Скалярні величини**

Можуть бути дійсні і уявні.

Приклад уявної величини.

a=5-2*%i

Результат:

a=

5.-2.i

Дозволені дії над уявними числами.

**Вектора**

Знаки **!** замінюють зображення круглих дужок для матриць.

Вектор-рядок

->v=[1-3%i,7]

v =

! 1. - 3. i 7. !

*Зауваження:* v=[1+3] є число 4, а то ж саме, але з пропуском перед "+" v=[1 +3] - ужевектор-строка. Якщо ж пропуск був після знаку "+", то це знову число.Задана операція транспонування: <вектор-строка>'

Приклад.

b=v'

Результат:

b =

! 1. !

! 4. !

! 8. !

Вектор-стовпець: елементи повинні бути розділені за допомогою крапки з комою.

-->v=[4;3+%i;3.33]

v =

! 4. !

! 3. + i !

! 3.33 !

Аналогічний вектор-рядок повинен бути закінчена знаком '.

-->v=[4;3+%i;3.33]'

v =

! 4. 3. - i 3.33 !

Для вектора (рядки) x[n], де x[0]=xmin, x[n]=xmax, а x[i]=x[i-1]+delta, допустимо

наступне завдання даних: x=[xmin:delta:xmax], де xmax, xmin і delta можуть бути

арифметическимими виразами.

Приклад.

x=[0:0.1:2*%pi]

У Scilab визначені арифметичні операції над векторами.

**Як створити двовимірну матрицю?**

**Спосіб 1.**

Пряме завдання матриці.

Приклад.

b=[11 22 33;21 22 23;31 32 33]

Результат:

b =

! 11. 22. 33. !

! 21. 22. 23. !

! 31. 32. 33. !

**Спосіб 2.**

Составление матриці з декількох підматриць.

Приклад.

a=[1 2;3 4]

a =

! 1. 2. !

! 3. 4. !

b=[5 5;7 8]

b =

! 5. 5. !

! 7. 8. !

c=[9 10;11 12]

з =

! 9. 10. !

! 11. 12. !

d=[a,b,c]

d =

! 1. 2. 5. 5. 9. 10. !

! 3. 4. 7. 8. 11. 12. !

Спосіб 3.

У Scilab існує тип змінних "розріджена матриця" (sparse). Розріджена матриця - це матриця, більшість елементів якої рівні 0. Використання такого типу змінних для великих матриць економить машинну пам'ять і підвищує швидкодія.

Для завдання матриць такого типу служить команда **sparse**. Щоб конвертувати

розріджену матрицю в звичний вигляд служить команда **full**.

Приклад.

```
sp=sparse([1,2;4,5;3,10],[1,2,3])
sp =
( 4, 10) sparse matrix
( 1, 2) 1.
( 3, 10) 3.
( 4, 5) 2.
r=full(sp)
r =
! 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. !
! 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. !
! 0. 0. 0. 0. 0. 0. 0. 0. 0. 3. !
! 0. 0. 0. 0. 2. 0. 0. 0. 0. 0. !
```

**Як створити багатовимірну матрицю?**

Використовуйте команду **hypermat**.

**Які існують функції формування масивів "спеціального вигляду"?**

У Scilab є наступні стандартні функції:

**ones** - формування масиву одиниць

**eye** - формування одиничної матриці

**zeros** - формування масиву нулів

**rand** - формування масиву елементів, розподілених по рівномірному закону (з

допомогою генератора випадкових чисел)

**:** - формування векторів і підматриць

Докладніше дивися нижче.

**Як сформувати одиничну матрицю (всі елементи нульові окрім**

**діагональних)?**

За допомогою команди **eye**.

Синтаксис

**X=eye(m,n)**

**X=eye(A)**

**X=eye()**

Параметри

**A,X** : матриці

**m,n** :целые числа

**m=<число рядків>, n=<число стовпців>**

Матриці A і X можуть бути неквадратними.

**eye(<число строк>,<число стовпців>)** служить для визначення матриці, укоторой всі елементи рівні нулю, окрім елементів головної діагоналі, значення яких рівне "1".

Якщо <число строк>=<число стовпців>, то це - одинична матриця.

Приклад 1.

eye(2,3)

ans =

! 1. 0. 0. !

! 0. 1. 0. !

Приклад 2.

Обращаніє заданої матриці v довільного розміру в матрицю з головною диагональю=1 і рештою елементів, рівних 0.

v=[2 -3.5 7]

a=eye(v)

Результат:

a =

! 1. 0. 0. !

Тобто розмір матриці зберігся, а значення елементів змінилося.

Приклад 3.

**eye()** задає матрицю з невизначеними розмірами. Вони будуть визначені, коли до цієї

матриці буде додана матриця з конкретними розмірами.

r=eye();

q=[1 2 3;10 20 30];

w=q+r

Результат:

w =

! 2. 2. 3. !

! 10. 21. 30. !

**Як швидко створити матрицю, всі елементи якої =1?**

Використовуйте команду **ones**. Простий випадок: **ones(<число строк>,<число стовпців>).**

Приклад.

-->ones(2,3)

ans =

! 1. 1. 1. !

! 1. 1. 1. !

Для створення вектора-рядка служить **ones(1,<число стовпців>)**

Для створення вектора-стовпця служить **ones(<число строк>,1)**

Для створення багатовимірної матриці, всі елементи якої =1, служить **ones(m1,m2..,mn)**

де **m1, m2..mn**- цілі позитивні числа.

**Як швидко створити матрицю, всі елементи якої =0?**

Використовуйте команду **zeros**, аналогічну команді **ones**.

Для двовимірної матриці слід використовувати конструкцію **zeros(<число строк>,<число**

**стовпців>)**

**Як створити рядок, елементи якого будуть послідовними цілими**

**числам від N до M?**

Потрібно виконати команду **N:M**.

Приклад.

a=2:5

Результат:

a =

! 2. 3. 4. 5. !

Конструкція **a=(2:5)** дасть еквівалентний результат.

**Генератор випадкових чисел**

Команда **rand** генерує послідовність випадкових чисел.

Синтаксис

**rand(m1,m2.. [,key])**

**rand(x [, key])**

**rand()**

**rand(key)**

**rand("seed" [,n])**

**rand("info")**

Параметри

**mi** : integers

**key** : символьна змінна (character string), що приймає значення "Uniform"и

"normal". Значення "uniform" відповідає рівномірному розподілу, а "normal" - гаусу.

**x** : матриця. До уваги приймається тільки її розмір.

Приклади використання.

1) rand(m1,m2) Утворення випадкової матриці з m1 рядків і m2 стовпців (m1 на m2).

2) rand(m1,m2..,mn) Утворення випадкової матриці розміром m1 на m2.. на mn.

3) rand(A) Утворення випадкової матриці такого ж розміру, якого була матриця A.

Матриця rand(A) комплексна, якщо матриця A була комплексна.

s=1:4; a=rand(s) a = ! .9184708 .0437334 .4818509 .2639556 !

4) rand() без аргументів дає випадкове скалярне число, випадковим чином

що змінюється при наступному виклику.

5)rand("normal") або rand("uniform") дозволяють задати розподіл випадкових

чисел.

6)rand("info") повертає значення змінної **key.**

7)rand("seed" [,n]) дозволяє реинициализировать генератор за рахунок зміни стартового числа.

**Які операції можна виконати над числовими матрицями?**

У Scilab визначені основні операції над матрицями:

A+B Складання

A-B Віднімання

c*A Множення на число

A*B Множення матриці на матрицю

A' Транспонування (поміняти місцями рядки і стовпці)

**matrix** Перетворення вектора або матриці в матрицю (вектор) іншого розміру

Існують і інші операції. Синтаксис таких операцій дивитеся в описі, але

дія таких операцій на перший погляд мало зрозуміло, і, ймовірно, не варто на них зупинятися. Можливо, є ще якісь нестандартні операції. Вивчайте їх самостійно.

**Як знайти суму елементів матриці?**

За допомогою команди **sum**.

Синтаксис

**y=sum(x)**

**y=sum(x,'r')** або **y=sum(x,1)**

**y=sum(x,'c')** або **y=sum(x,2)**

Параметри

**x** : вектор або матриця (дійсна, комплексна, розріджена (sparse) або

поліноміальна)

**y** : скаляр або вектор

sum(x) дасть суму всіх елементів вектора або матриці x .

y=sum(x,'r') або, еквівалентно, y=sum(x,1) є рядок, рівний поелементній сумі

стовпців.

(y(j)= sum(x(i,j),i=1,m)).

y=sum(x,'c') або, еквівалентно, y=sum(x,2) є стовпець, рівний поелементній сумі

рядків.

(y(i)= sum(x(i,j),j=1,n))).

Приклад.

-->a=[10,20;300,400]

a =

! 10. 20. !

! 300. 400. !

-->b=sum(a)

b =

730.

-->c=sum(a,1) //це еквівалентно c=sum(a,"r")

з =

! 310. 420. !

-->c=sum(a,2) //це еквівалентно c=sum(a,"c")

з =

! 30. !

! 700. !

**Як знайти твір елементів матриці?**

За допомогою команди **prod**.

Синтаксис

**y=(x)**

**y=prod(x,'r')** або **y=prod(x,1)**

**y=prod(x,'c')** або **y=prod(x,2)**

Параметри

**x** : вектор або матриця (дійсна, комплексна, розріджена (sparse) або

поліноміальна)

**y** : скаляр або вектор

Приклад.

-->a=[10,20;300,400]

a =

! 10. 20. !

! 300. 400. !

-->help prod

-->prod(a)

ans =

24000000.

-->prod(a,1) //це еквівалентно c=prod(a,"r")

ans =

! 3000. 8000. !

-->prod(a,2) //це еквівалентно c=prod(a,"c")

ans =

! 200. !

! 120000. !

**Нечислові (символьні) матриці**

Символьні матриці - це матриці, елементи яких мають строковий тип (strings).

Строкові елементи повинні бути поміщені в одинарних або в подвійні лапки (це байдуже).

Приклад.

-->a=["2+3" 'bb";'cc' 'dd"]

a =

!2+3 bb !

! !

!cc dd !

Символьниє матриці можна складати один з одним. При цьому відбувається злиття відповідних елементів.

Приклад:

->a=["a","b";"c","d"];

-->b=["x1","y1";"z1","w1"];

-->a+b

ans =

!ax1 by1 !

! !

!cz1 dw1 !

**Які дії можна здійснювати над символьними матрицями?**

Над символьними матрицями в Scilab дозволені наступні операції:

Символьне складання: команда **addf**

Символьне множення: команда **mulf**

Вирішення символьних лінійних систем: команда **solve**

Символьна тріангуляція: команда **trianfml**

Є і інші команди: **trisolve**, **evstr**. Синтаксис дивитеся за допомогою системи "Help".

Приклад.

-->w1="a";

-->w2="b";

-->c=w1+w2 // Це злиття

з =

ab

-->s=addf(w1,w2) // Це символьне складання

s =

a+b

**Поліноми і полиномные матриці (алгебра**

**уявлення)**

**Як визначити поліном алгебри?**

**Спосіб 1.**

Використовуйте команду **poly**.

Синтаксис

**[p]=poly(a,"x" ["flag"])**

Параметри

**a** : матриця або дійсне число

**x** : символьна змінна

"**flag"** : строкова змінна, що приймає значення "roots" або "coeff". По

умовчанню приймає значення "roots".

Можна писати скорочено: "flags"= "r" або "c".

Прапор "c" означає, що параметр **a** матиме сенс коефіцієнтів полінома.

Слід додати, що образ думки розробників украй дивний...

Ми б рекомендували все-таки прапор "c". Що відбувається з прапором "r" зовсім незрозуміло.

Звернете увагу на наступний приклад:

x=poly(5,"y")

Це еквівалентно x=poly(5,"y","r") або x=poly(5,"y","roots")

Результат:

x=

-5+y

x=poly(5,"y","c") дасть просто 5. Це еквівалентно x=poly(5,"y","coeff").

Спосіб, що рекомендується:

-->poly([1 2 3],"x","c")

ans =

1 + 2x + 3x2

*Примітка:* знак ступеня на екрані не містить "^", і виглядає так, як ніби це число

у іншому рядку, що приводить до сильного подиву... (Не лякайтеся)

**Спосіб 2.**

Цей спосіб є наочнішим.

x=poly(0,"x");

// Цей рядок зазвичай забувають. Вона визначає символьну змінну x="x".

1+2*x+3*x2

***Зауваження:***

Scilab може працювати тільки з однією змінною, заданою цим методом.

Виконання програми

a=poly(0,"a");

b=poly(0,"b");

c=a+b

дає в результаті помилку:

!--error 4

undefined variable : %p_a_p

Вираз же d=a*7+12 дозволений...

**Як отримати полиномную дріб?**

Створіть чисельник і знаменник дробу за допомогою команди **poly** і скористайтеся

знаком ділення /

Приклад.

-->poly([1 -1],"x","c")/poly([1 2 3],"x","c")

ans =

1 - x

---

1 + 2x + 3x2

**Як проводити операції в символьному вигляді?**

Так само, як і в чисельному вигляді. Дуже зручно для виконання завдань алгебри в

середній школі.

Приклад.

Хай ми хочемо помножити один поліном на іншій, наприклад (1-x)*(1+2x2).

p1=poly([1 -1],"x","c");

p2=poly([1 0 2],"x","c");

p1*p2

Результат:

ans =

1 - x + 2x2 - 2x3

**Як створити символьну матрицю, елементи якої є**

**поліномами?**

Приклад.

s=poly(0,"s");

p=1+2*s+s^2;

M=[p,p-1;p+1,2]

Результат:

M =

! !

! 1 + 2s + s2 2s + s2 !

! !

! 2 + 2s + s2 2 !

Можна знайти детермінант цієї матриці в символьному вигляді за допомогою команд

**det(M)**,detr(M) або **determ(M)**.

**Булеві матриці**

Булеві (логічні) матриці - це матриці, елементами яких є Булеві

константи.

Визначення Булевих констант:

**%t=T TRUE**

**%f=F FALSE**

Синтаксис для визначення Булевих матриць такий же, як і для звичайних матриць.

Над Булевими змінними визначені наступні операції:

**== equal** Рівність

**~** Логічне заперечення

**|** Логічне "або" (or)

**&** Логічне "и" (and )

Приклад 1.

Застосування команди оператора "==" до вектор-рядка.

-->[1,2]==[1,3]

ans =

! T F !

Приклад 2.

Визначимо, які елементи вектор-рядка більше 2.

a=[1 2 4 5 1.5];

a>2

ans =

! F F T T F !

Приклад 3. Порівняння Булевих матриць.

A=[%t,%f,%t,%f,%f,%f];

B=[%t,%f,%t,%f,%t,%f];

A | B

ans =

! T F T F T F !

A&B

ans =

! T F T F F F !

Якщо q=%t, тоді значення q буде рівне T, а ~q матиме значення F.

**Цілочисельні матриці**

**Які типи цілочисельних даних (integer) визначені в Scilab?**

У Scilab визначено 6 типів даних типу integer:

32 bit signed integer (sub-type 4)

32 bit unsigned integer (sub-type 14)

16 bit signed integer (sub-type 2)

16 bit unsigned integer (sub-type 23)

8 bit signed integer (sub-type 2)

8 bit unsigned integer (sub-type 12)

**Як перетворити дані з одного типу змінних в іншій?**

**Оператори конверсії в дані цілого типу:**

**y=int8(X)** : повертає значення з області [-128,127]

**y=uint8(X)** : повертає значення з області [0,255]

**y=int16(X)** : повертає значення з області [-32768,32767]

**y=uint16(X)** : повертає значення з області [0, 65535]

**y=int32(X)** : повертає значення з області [-2147483648,2147483647]

**y=uint32(X)** : повертає значення з області [0, 4294967295]

Параметри

**X** : матриця з дійсних або цілих чисел

**y** : матриця з цілих чисел, закодованих з 1, 2 або 4 байт.

Приклад.

x=[0.11 3.2 27.5 187];

int16(x)

ans =

!0 3 27 187 !

int8(x)

ans =

!0 3 27 -69!

Замість 187 ми отримали -69, оскільки максимальне значення допустимого числа в цьому

типі =127 (тобто <187)

**Оператори конверсії в дані дійсного типу:**

**double** - конверсія з будь-якого типу даних integer в дійсне число.

Синтаксис

**y=double(X)**

Параметри

**X** : дійсна або ціла матриця

**y** : матриця з дійсних чисел. Число вважається окремим випадком матриці.

**Як дізнатися тип змінної?**

За допомогою оператора **type**

Синтаксис

**[i]=type(x)**

Параметри

**x** : об'єкт Scilab

**i** : ціле

**type(x)** повертає ціле число, що характеризує тип змінної "**x**" відповідно нижчеприведеному списку значень, що приймаються:

1 : дійсна або комплексна постійна матриця

2 : поліноміальна матриця

4 : Булева матриця

5 : розріджена (sparse) матриця (дивитеся її визначення за допомогою **help sparse**

8 : ціла матриця з 1, 2 або 4 байт

10 : матриця з символьних змінних

11 : нескомпилированнвя функція

13 : скомпільована функція

14 : бібліотека функцій

15 : список (list)

16 : типизованный список (tlist)

128 : покажчик (pointer)

Приклад.

-->type(5.63)

ans =

1.

**Як дізнатися тип об'єкту Scilab?**

За допомогою оператора **typeof**. Детально дивися в описі.

**N-мерные масиви**

**Як визначити N-мерный масив?**

Можливі декілька варіантів рішення.

**Спосіб 1.**

Масив з n1 рядків і n2 стовпців може бути визначений таким чином:

n1=3;n2=4;z=9;

d(n1,n2)=9

Результат:

d =

! 0. 0. 0. 0. !

! 0. 0. 0. 0. !

! 0. 0. 0. 9. !

Аналогічно можна визначити багатовимірний масив. Він буде весь нульовим окрім

елементу з максимальними індексами по всіх змінних. Задамо його рівним числу

наприклад =77.

x(n1,n2,n2,n3,n4)=77

Для визначення нульової матриці:

d(n1,n2)=0

**Спосіб 2.**

Визначимо матрицю з цілими елементами, рівними послідовним цілим числам, з

допомогою команди **hypermat**.

hypermat([2,3],7:12)

ans =

! 7. 9. 11. !

! 8. 10. 12. !

**Спосіб 3.**

Цей спосіб найбільш природний.

Задаватимемо матрицю з клавіатури відрядкового. Число рядків попереднє не

задається.

Приклад.

-->V=[11 12 13;

-->21 22 23;

-->31 32 33;

-->41 42 43]

Результат:

V =

! 11. 12. 13. !

! 21. 22. 23. !

! 31. 32. 33. !

! 41. 42. 43. !

**Як визначити розмір масиву?**

**Спосіб 1.**

За допомогою команди **size(d)**, де **d** - масив, який ми хочемо визначити.

Для 2-мірного масиву в результаті отримаємо **[nr,nc]=size(x)**, де **nr**- число рядків, а **nc**-

число стовпців.

Приклад.

d(3,4)=9

d =

! 0. 0. 0. 0. !

! 0. 0. 0. 0. !

! 0. 0. 0. 9. !

size(d)

ans =

! 3. 4. !

Можна отримати для двовимірного масиву число рядків і як size(s,1) або size(d,"r")

а число стовпців як size(s,2) або size(d,"c").

Загальне число елементів масиву можна визначити за допомогою оператора **lenght(<ім'я**

**об'єкту>)**. Воно буде рівне твору всіх размерностей масиву. Для приведеного

вище за приклад lehgth(d) буде рівне 12.

**Спосіб 2.**

За допомогою оператора **hypermat**.

Синтаксис

**M=hypermat(dims [,v])**

Параметри

**dims** : вектор розмірів гіперматриці

**v** : вектор введення даних гіперматриці Значення його за умовчанням рівне

zeros(prod(dims),1), тобто нульовий вектор.

Приклади.

->a=hypermat([2,3])

a =

! 0. 0. 0. !

! 0. 0. 0. !

-->q=hypermat([2,3],7:12)

q =

! 7. 9. 11. !

! 8. 10. 12. !

N-мерная матриця закодована у вигляді списку mlist з двома полями. Для визначення

розмірності цієї матриці здійснимий:

-->q.dims

ans =

! 2. 3. !

Для визначення змісту цієї матриці:

-->q.entries

ans =

! 7. !

! 8. !

! 9. !

! 10. !

! 11. !

! 12. !

Щоб визначити значення конкретного елементу цієї матриці

-->q.entries(5)

ans =

11

*Зауваження:* Мабуть, елементи нумеруються з одним індексом послідовно "стовпець за стовпцем".

Приклад.

Побудуємо матрицю M з 2-х рядків і 3-х стовпців і задамо її елементи:

-->M=hypermat([2,3]);

-->M.entries(1)=11;

-->M.entries(2)=21;

-->M.entries(3)=12;

-->M.entries(4)=22;

-->M.entries(5)=13;

-->M.entries(6)=31;

В результаті побудована матриця:

M =

! 11. 12. 13. !

! 21. 22. 31. !

**Функції (макроси)**

Функції (макроси) в Scilab схожі на тих, що ви вже зустрічали в інших мовах програмування.

Функції можуть мати аргумент, самі бути аргументом іншої функції, бути членом

списку, брати участь в операціях порівняння, викликатися рекурсивно.

Функція починається із слова function **i** закінчується словом endfunction. Зазвичай

функції визначені в текстовому файлі, набраному в зовнішньому редакторові (наприклад, в Windows в редактоope Wordpad або в "блокноті") і завантажуються в Scilab з допомогою

команди **exec("filename")**. Можна створювати функції і усередині Scilab. Замість подвійних лапок можна писати одинарні. Те ж саме можна виконати за допомогою меню **File operation** (**Load**, **getf**, **Exec** ). Надалі буде показано як завантажувати функції у файл

"filename" і компілювати їх.

Перший рядок функції може бути наступним:

**function[y1...,yn]=my_name(x1...,xk)**,

де **yi** - вихідні змінні і **xi** - вхідні змінні. Детально про використання макросів вивчатимемо пізніше.

**Як створити функцію засобами пакету Scilab?**

За допомогою команди **deff.**

Детально дивися в розділі 3 розділ "Завантаження функцій".

Приклад 1.

Створимо функцію з ім'ям **fun** двох аргументів **t** і **y**, результатом якої буде

тривимірний вектор, перший елемент якого рівний **t+y**, другий елемент рівний **t-y**, а

третій елемент рівний **t*y**.

```
deff('[w]=fun(t,y)',[
'w(1)=t+y;';
'w(2)= t-y;';
'w(3)= t*y;'])
//Викличемо цю функцію
```

q=fun(5,7);

Результат:

q =

! 12. !

! - 2. !

! 35. !

Приклад 2.

Створимо функцію, що обчислює координати сфери радіусу **r** і посторим її.

x=(0:0.5:10);

y=(0:0.5:10);

r=10;

//рівняння для сфери x^2+y^2+z^2=r^2 Обчислимо z=sqrt(r^2-x^2-y^2);

deff('[z]=surf(x,y)','z=sqrt(r^2-x^2-y^2)');

fplot3d1(x,y,surf);

***Зауваження:*** Складні функції краще створювати на мовах Fortran або C, а потім

линковать разом з пакетом Scilab.

**Бібліотеки**

Бібліотеки - це колекції стандартних функцій, які можуть або автоматично

завантажуватися в Scilab, або викликатися і завантажуватися користувачем. Бібліотеки

створюються командою **lib**. Приклади бібліотек дані в директорії /macros/. Файли з

розширенням .sci написані в ASCII кодах і містять текст функцій. Файли з

розширенням **.bin** є функціями, що вже відкомпілювалися. Що відкомпілювалися

функції бібліотеки автоматично викликаються в Scilab при першому виклику. Щоб

побудувати власну бібліотеку слід використовувати команду **genlib**.

У стандартний пакет Scilab входять бібліотеки прикладних, що часто використовуються

функцій:

• Бібліотека Control (Classical, LQG, H-infinity...).

• Лінійна алгебра

• Пакет оптимізації LMI (вирішення лінійних матричних нерівностей) оптимізації.

• Бібліотека обробки сигналів (Signal processing).

• Бібліотека моделювання (вирішення диференціальних рівнянь і ін.).

• Бібліотека оптимізації (дифференцируемость і недифференцируемость

вирішення LQ).

• Інтерактивна бібліотека Scicos моделювання динамічних систем.

• Бібліотека Metanet (аналіз і оптимізація мереж).

**Об'єкти**

У Scilab визначені наступні об'єкти:

"**constant"** якщо об'єкт дійсна або комплексна матриця

"**polynomial"** якщо об'єкт дійсна або поліноміальна матриця

"**function"** якщо об'єкт функція

"**string"** якщо об'єкт матриця з символьних змінних

"**boolean"** якщо об'єкт булева матриця

"**list"** якщо об'єкт список

"**rational"** якщо об'єкт раціональна матриця (transfer matrix)

"**state-space"** якщо об'єкт "state-space" модель (дивися syslin)

"**sparse"** якщо об'єкт розріджена(sparse) матриця

"**boolean sparse"** якщо об'єкт розріджена (sparse) булева матриця

Ці типи узяті з help для операції typeof моєї версії scilab, у файлі intro.pdf

приводяться об'єкти з невеликим різночитанням.

Про всі об'єкти, визначені в Scilab, можна дізнатися, подивившись опис функції

**typeof**, яка служить для визначення типу об'єкту, що цікавить нас.

**Операції над матрицями**

У Scilab дозволені наступні базові операції над матрицями:

**[ ]** визначення матриці, "зчеплення" (concatenation)

**;** роздільник рядків

**( )** розширення m=a(k)

**( )** вставка a(k)=m

**'** транспонування

**+** складання

**-** віднімання

***** множення

\ ліве ділення

/ праве ділення

^ піднесення до ступеня

**.*** поелементний спосіб множення

**.** поелементний спосіб лівого ділення

**../** поелементний спосіб правого ділення

**.^** поелементний спосіб піднесення до ступеня

**.*.** тензорне множення (кронекерное множення)

**Індексування**

Для повного огляду можливостей індексування подивитеся Help по операціях

**extraction** і **insertion**.

**Як здійснюється індексування елементів в матрицях?**

Індексування матриці може здійснюватися вибором рядків і стовпців, або використовуючи булеві індекси, або за допомогою символу **$**.

Приклад 1.

->A=[1 2 3;11 22 33]

Результат

A =

! 1. 2. 3. !

! 11. 22. 33. !

Приклад 2.

Хай ми хочемо привласнити величині m значення одне з елементів матриці A.

-1->m=A(2)

Тут індексування проводилося в одну лінію: по першому стовпцю зверху-вниз, а потім по- другому стовпцю зверху-вниз і так далі. Така нумерація не зовсім зручна.

m =

11.

Зрозуміліший спосіб m=A(,)

m=A(1,2) // дасть для наший матриці 2

->A([2 1],2)

дасть

ans =

! 22. !

! 2. !

Що це означає, не знаю...

-->A(:,3) //дає зміст всього третього стовпця

ans =

! 3. !

! 33. !

->A(:) //дає зміст всієї матриці у вигляді вектор-столбца.

Можливі і більш екзотичні незрозумілі комбінації:

A(:,3:-1:1)

A([%t %f %f %t])

A(1:2,$-1)

A($:-1:1,2)

A($)

Вивчайте це самі, якщо хочете (і якщо думаєте, що це може Вам в житті стати в нагоді)...

Приклад.

Заповнимо матрицю однаковими елементами.

Крок 1

-->x="fox"

Зараз **x** - строкова змінна, рівна "fox". А зараз створимо матрицю, всі елементи

якими будуть "fox".

Крок 2

-->f=x([1 1;1 1;1 1])

Результат:

f =

!fox fox !

! !

!fox fox !

! !

!fox fox !

*Зауваження:* Існують багато багатих можливостей для створення і індексування

матриць. На перший погляд деякі з них малозрозумілі і неясно, де застосовні, це

буде ясно надалі.

**Точність обчислень і визначення формату виводу**

**числового результату**

На перший погляд здається, що пакет Scilab має недостатню високу точність

обчислення, а саме 8 значущих цифр.

Приклад.

a=12345.6789012345

Результат:

a =

12345.679

Але це не так: 8 значущих цифр - це формат для виведення числа на екран за умовчанням. Насправді Scilab "знає" число набагато точніше. Для того, щоб контролювати кількість розрядів числа, що виводяться, на друк можна застосувати, наприклад, команду **printf** із заданим форматом. Формат виводу задається по тих же правилах, що і для мови С. Наприклад, формату **f** відповідає синтаксична формула

**f double; [-]m.dddddd, d's = precision (default 6)**

Приклад.

r=0.1234567890123456789

Результат:

r =

.1234568

printf("%1.12f",r);

Результат:

0.123456789012

***Зауваження:*** Застосування формату **%f** без вказівки формату виводить 6 знаків після

комі.

Приклад.

x=12345678.12345

Результат:

x =

12345678.

printf("%f",x);

Результат:

12345678.123450

Чому рівна величина, невідмітна по трохи від нуля в пакеті Scilab мені зовсім неясно,но для дійсних чисел вона, мабуть, ніяк не більше, ніж **e-16** (а можливо і менше)!

**Оформлення протоколу**

***По ходу виконання роботи документувати всі виконувані дії і вносити їх до протоколу.***

**Лабораторна робота №2**

**Знайомство з системою програмування Scilab (частина 2)**

# Програмування

## Зміст:

- Використання функцій
- Структура функцій
- Завантаження функцій
- Глобальні і локальні змінні
- Спеціальні команди для функцій
- Визначення операцій на нових типах даних
- Відладка (debbuging) Scilab-функции

## Використання функцій

У пакеті є можливість використовувати функції. Це дозволяє створювати інтегровані в Scilab спеціалізовані програми і використовувати бібліотеки. Засоби програмування (Tools): цикли, умовні конструкції. Дозволено застосування вкладених циклів. У умовних конструкціях і циклах використовуються наступні оператори порівняння:

== рівно

< менше ніж

- більш ніж
- <= менше або рівно чим
- >= більше або рівно чим
- <> або
- ~= не рівно
- Визначені наступні типи циклів:

for

while

1) Цикл for Синтаксис for variable=n1:step:n2,<тело цикла>,end Як індекс циклу можуть виступати змінні різних типів (вектора, списки і ін.). Зауваження: Якщо крок циклу step=1, його можна не писати: x=1;for k=1:4,x=x*k,end еквівалентно x=1;for k=1:1:4,x=x*k,end.

Приклад циклу for.

Обчислення факторіалу

-->x=1;for k=1:4,x=x*k,end x = 1. x = 2. x = 6. x = 24.

Зауваження: Оскільки усередині циклу в даному прикладі стоять як роздільник коштують коми (","), виводяться всі проміжні значення "X". Якщо використовувати конструкцію з роздільником "крапка з комою" (";"), то ніяких повідомлень ні буде і по спеціальному запиту можна буде дізнатися кінцеве значення "X".

x=1;for k=1:4;x=x*k;end

2)Цикл while

Синтаксис

while <тіло циклу> end

Приклад циклу while. -->x=1; while x<14, x=x*2,end x = 2. x = 4. x = 8. x = 16. Цикли while і for можуть бути перервані за допомогою оператора break.

Приклад. -->a=0; for i=1:5:100, a=a+1; if i>10 then break, end;end -->a a = 3. Зауваження: Якщо ми хочемо перервати ззовні виконання циклу (наприклад, видно, що відбулося зациклення), використовуйте функціонального меню управління вікна: Control - Abort.

Подивитеся також команди return і pause.

Допустимі наступні умовні конструкції:

1) if-then-else

2) select-case

1) Конструкція if-then-else Синтаксис if expr1 then statements elseif expri then statements .... else statements end, end Зауваження: Число символьних знаків, використовуваних для визначення тіла умовної конструкції (if while for або select/case) повинно бути обмежено 16k.

Приклад. i=2 for j = 1:3, if i == j then a(i,j)= 2; elseif abs(i-j)== 1 then a(i,j)= -1; else a(i,j)= 0; end, end Результат: a = ! 0. 0. 0. ! ! - 1. 2. - 1. ! Там, де, наприклад, в мові C++ використовуються фігурні дужки { } для розділення тексту, в Scilab використовується кома.

Синтаксис конструкції " select-case" select expr0, case expr1 then instructions1, case expr2 then instructions2 ... case exprn then instructionsn, [else instructions], end

Приклад. x=-1 select x, case 1,y=x+5,case -1,y=sqrt(36),end Результат: y=6

Структура функцій

Функції грають роль підпрограм. Найзручніше набирати функції в текстовому редакторові і зберігати їх в окремих файлах (зовнішні функції), але можна їх використовувати і безпосередньо в системі Scilab.

Синтаксис function[y1...,yn]=foo(x1...,xm) ..... тіло функції .... endfunction

де foo - ім'я фунции, xi - вхідні аргументи функції (їх m штук), yi - вихідні аргументи функції (їх n штук).

Приклад. Обчислення факторіалу. function [x]=fact(k) k=int(k) if k<1 then k=1, end x=1; for j=1:k,x=x*j;end endfunction

Наберемо цей текст в будь-якому текстовому редакторові і збережемо його у файлі з ім'ям fact.sci. Розширення *.sci є для Sclab "рідним", але не обов'язковим. Потім слід викликати ці файли з Scilab допомогою команд getf(filename) або exec(filename-1); Ті ж операції можна провести за допомогою команд меню File-getf або File-exec. Для нашого прикладу цей виклик буде: getf('D:\zzz\fact.sci'); До виклику функції бажано перевірити, чи не була вже завантажена така функція раніше. Для цього: exists('fact') Результат: ans = 0. Так само можна було використовувати для цього команду who. Після завантаження функції з допомогою -->exec('D:\zzz\fact.sci'); -->exists('fact') отримаємо: ans = 1. Тепер ми можемо користуватися цією функцією: -->x=fact(5) x = 120

**Завантаження функцій**

Для завантаження функцій застосовуються команди exec, deff і getf. При визначенні функції function[y1...,yn]=foo(x1...,xm) вхідні і вихідні параметри xi і yi можуть бути будь-якими об'єктами Scilab, окрім самих цих функцій. Якщо функція заздалегідь не завантажена в Scilab за допомогою getf(filename) або exec(filename-1), то вона і не виконується. Завантажуваний файл може містити декілька функцій. Функції можна визначати і безпосередньо в сеансі Scilab, використовуючи конструкцію function/endfunction або використовуючи функцію deff. Це корисно, якщо ми хочемо визначити функцію як вихідний параметр іншої функції. Спосіб 1. Виклик функції за допомогою exec.

Синтаксис exec(fun [,mode]) exec(path [,mode]) ierr=exec(path,'errcatch' [,mode]) ierr=exec(fun,'errcatch' [,mode]) Параметри fun : ім'я файлу, в якому знаходиться Scilab функція (функція визначена безпосередньо в Scilab) path : рядок, що містить повний шлях до файлу, що містить функцію ierr : ціле, 0 або номер помилки mode : цілий скаляр, що може приймати наступні вирази: 0 : за умовчанням -1 : нічого не друкувати 1 : повідомлення (echo) для кожного командного рядка 2 : prompt --> друкує 3 : echoes + prompts

4 : stops зупинка перед кожним "prompt" 7 : stops + prompts + echoes : корисно для демонстрацій.

Спосіб 2. Виклик функції за допомогою deff Синтаксис deff('[s1,s2...]=newfunction(e1,e2....)',text [,opt]) Параметри e1,e2..., : вхідні змінні s1,s2..., : вихідні змінні text : матриця з символьних значень (character strings) opt : необов'язкова змінна типу character string. Можливі значення opt: 'c' : функція скомпільована для більшої ефективності (за умовчанням) 'n' : функція не скомпільована Приклад. deff('[x]=myplus(y,z)','x=y+z') deff('[x]=mymacro(y,z)',['a=3*y+1'; 'x=a*z+y']) Зауваження: на перший погляд це виглядає достатньо незручно. Спосіб 3. Виклик функції за допомогою getf Синтаксис getf(file-name [,opt]) Параметри filename : Ім'я файлу (Scilab string) opt : необов'язкова змінна типу character string. Можливі значення opt: 'c' : функція скомпилированна для більшої ефективності (за умовчанням) 'n' : функція нескомпільована "p" : завантажувана функція скомпільована і приготована для профілізації. Приклад. getf('SCI/macros/xdess/plot.sci') Зауваження: Метод виклику функцій за допомогою getf вважається застарілим, переважно користуватися exec.

## Глобальні і локальні змінні

Змінні можуть бути локальними і глобальними. 1) Локальні змінні Якщо значення змінних у функції не визначені (і не є вхідними параметрами), то їх значення беруться як змінні із зухвалого середовища.

Приклад. Наберемо в будь-якому текстовому редакторові в ASCII кодах: function [y1,y2]=f(x1,x2) y1=x1+x2; y2=x1-x2; endfunction Запишемо цю функцію у файл D:\zzz\fact.sci. exec('D:\zzz\fact.sci') // завантаження функції [y1,y2]=f(1,1) Результат: y2 = 0. y1 = 2. Краще все ж таки при виклику користуватися не іменами змінних, використовуваних у визначенні функції, а вводити нові. Приклад. [a,b]=f(1,1) При цьому x1 x2 залишаються локальними змінними і після виклику функції залишаються невідомими для

Scilab. 2)Глобальные змінні Можуть бути визначені за допомогою команди global. Синтаксис global('nam1'...,'namn') global nam1 ... namn Параметри nam1..., namn : імена змінних Приклад. global a b з a=1;b=2;c=3; Для знищення глобальних змінних служить команда clearglobal. clearglobal() знищує всі глобальні змінні. clearglobal nam1 .. namn знищує глобальні змінні із заданими іменами. clearglobal nam1 .. namn clearglobal('nam1' ..,'namn') Щоб дізнатися, які глобальні змінні визначені в даній сесії, служить команда who('global'). Приклад. who('global') Подивитеся, які з глобальних змінних залишилися за допомогою who('global'). clearglobal('a') clearglobal() знищить всі змінні і результатом who('global') буде: ans = []

## Спеціальні команди для функцій

| | |
|---|---|
| argn | повертає число вхідних і вихідних аргументів у виклику функції |
| error | використовується для друку повідомлень про помилку або розгалужень на випадок детектування помилок. |
| pause | мода очікування. Використовується для відладки. |

| | |
|---|---|
| abort | служить для виходу з режиму очікування |
| warning | очікування повідомлень |
| break | переривання циклу |
| return і resume | служать для передачі локальних змінних з функції в основне тіло програми |

Зауваження: Всі ці службові команди використовуються виключно усередині функцій. Приклад. function [z]=foo(x,y) if x==0 then, error('division by zero'); end, z=x/y endfunction Результат застосування функції foo: -->foo(6,2) ans = 3. -->foo(6,0) !--error 27 division by zero... at line 5 of function foo called by : foo(6,0)

## Визначення операцій на нових типах даних

Існує можливість визначити фундаментальні операції для нових типів даних. Дивися help overloading. Користувач може дати власне визначення, наприклад, операції множення. Нові оператори, визначені користувачем, повинні починатися із знаку % і складатися з 3 або 4 полів. Дивитеся таблицю для визначення можливих імен на стор. 68 файлу intro.pdf або за допомогою help overloading. Залишимо розгляд цієї можливості на майбутнє.

**Відладка (debbuging) Scilab-функции**

Найбільш простій спосіб відладки Scilab-функции полягає у введенні у функцію однієї із спеціальних команд (дивися розділ 3.2.4). Зупинити виконання функції можна за допомогою оператора abort.Удобно вставляти у функцію переривання. Подивитеся команди setbpt, delbpt і disbpt. Для відстежування помилок також зручні errclear і errcatch. Експерти в Scilab можуть використовувати функцію debug(i), де i=0...,4) означають рівень відладки.

# Графіка

**Зміст:**

- Як намалювати 3D гістограму?
- Побудова змішаних графіків з 2D і 3D поверхонь
- Зображення декількох малюнків в одному графічному вікні

## Графічне вікно

Графічні застосування в пакеті Scilab виводяться в окремому вікні, званому графічним. Графічне вікно можна створити за допомогою меню головного вікна Scilab Graphic Window "x" - Set (Create) Window, де "x" є поточним номером графічного вікна. За умовчанням цей номер рівний нулю і створюється графічне вікно з номером "0". Номер графічного вікна вказаний в його заголовку.

Допустимо відкрити одночасно декілька графічних вікон, але активним з них є тільки одне, зване поточним. Якщо ми хочемо створити графічне вікно з іншим номером вікна, наприклад "4", то ми винні спочатку його пронумерувати за допомогою команди меню Graphic Window0-(increase current num), виконавши його стільки раз, щоб в меню головного вікна замість напису Graphic Window0 з'явився напис Graphic Window4. Потім створимо графічне вікно за допомогою меню Raise (Create) Window або Set (Create) Window. У заголовку створеного графічного вікна тоді замість ScilabGraphic0 буде написано ScilabGraphic4. Заголовок поточного графічного вікна можна змінити за допомогою команди xname. Якщо ви хочете зробити активним, наприклад, вікно з номером 3, слід встановити його номер в головному вікні за допомогою Graphic Window0-(increase current num). Потім зробити його активним за допомогою Graphic Window "x" - Set (Create) Window. Тепер воно стало поточним і всі графічні застосування зображатимуться в нім. Команда x=winsid(), де x -вектор-строка, повертає список номерів всіх існуючих вікон. Для очищення поточного графічного вікна використовується команда меню Graphic Window- Clear Current Window.

У графічному вікні є своя панель меню. Меню File має підміню і служить для роботи з файлами: дозволяє записувати вміст графічного вікна у файл в рідній форматі .scg (File - Save), конвертувати в інші формати(File - Export), завантажувати у вікно картинки у форматі .scg з диска (File - Load), друкувати на принтері і ін. Команди меню 2D Zoom і UnZoom служать для збільшення або зменшення масштабу в графічному вікні. Команда меню 3D Rot дозволяє обертати 3D-объекты.

Як очистити графічне вікно?

Спосіб 1. С допомогою меню Graphic Window - Clear Current Window. Це аналог команди xbasc(). Спосіб 2. За допомогою команди xbasc. При очищенні вікна асоціативний запис при цьому теж знищується. Синтаксис xbasc([window-id]) Параметри window-id : цілий скаляр або вектор, вказуючий на номери тих вікон, які очищатимуться. Без вказівки параметрів очищається поточне вікно. Команда xbasc(1:3) знищить зображення у вікнах з номерами 1, 2 і 3. Команда xbasc([1,3,5]) знищить зображення у вікнах з номерами 1, 3 і 5. Якщо одне з цих вікон не існує, воно буде автоматично створено, що є достатньо несподіваним. Знищується тільки зображення, самі вікна продовжують існувати. Приклад. plot(1:10) xbasc() // Вікно очищається plot(1:5) // Зображається новий графік в нових осях. Спосіб 3. За допомогою команди xclear. Синтаксис xclear([window-id]) Параметри window-id : цілий скаляр або вектор, вказуючий на номери тих вікон,которые очищатимуться. Відмінність цієї команди від xbasc: вікно очищається, а відповідні команди запису залишаються В результаті зображення графіка, намальованого до застосування функції xclear() залишиться, а зверху накладеться графік, побудований після застосування функції xclear().

У Scilab можливі різні графічні пристрої для засилання зображень у вікна або на друкуючий пристрій. За умовчанням для виводу призначено вікно ScilabGraphic0.

Присутні наступні драйвери:

- X11 вивід на екран комп'ютера
- Rec вивід на екран комп'ютера із записом всіх графічних команд. Цей драйвер використовується за умовчанням.
- Wdp
- Pos - вивід у форматі Postscript
- Fig - вивід у форматі Xfig
- GIF - вивід у форматі GIF

## 2D графіка

Як побудувати простий одновимірний графік y=f(x)?

За допомогою команди plot. Синтаксис plot(x,y[xcap,ycap,caption]) plot(y) Параметри x,y: два вектори однакового розміру xcap,ycap,caption: рядки або строкові матриці Команда plot зображає параметр y як функцію від

параметра x. Якщо y задано у вигляді формули від x, то перший параметр у формулі можна опустити. Параметри xcap і ycap є відповідними найменуваннями осей координат графіка x і y. За умовчанням у мене значення параметра ycap знаходиться зверху праворуч від вертикальної осі, а значення параметра xcap знаходиться зверху праворуч від кінця горизонтальної осі, що не дуже красиво. Параметр caption є заголовком графіка. У мене він пишеться зверху над графіком. Дивися приклад 4(нижче). Перший аргумент x може бути опущений, якщо y заданий як функція від x і є вектором. Інакше використовуйте функцію plot2d. Нагадаємо способи завдання векторів:

1) Можна задати аргумент трьома значеннями: мінімальним, кроком і максимальним значенням: a=(amin:step:amax) x=(2:0.2:3); Можна визначити аргумент і без круглих дужок: x=0:0.1:2*%pi;

2)Указать всі значення: x=[2 2.2 2.4 2.6 2.8 3.0]; Приклад 1. x=[1 2 3 4 5 6]; y=x^2; plot(y) Зауваження: Оскільки аргумент x функції y заданий неявно, те використання команди plot з одним параметром все ж таки небажано.

Приклад . Хай ми хочемо побудувати криву залежність[i] від x[i]. Хай ці значення задані експериментально. Введемо їх для прикладу самі.

x=[1 2 3 4 5 6 7 8]; y=[1 4 9 16 25 36 49 64]; plot(x,y)

В результаті отримаємо графік параболи.

Приклад .

Хай значення вектора x задане у вигляді дискретних крапок, а y задано у вигляді аналітичної залежності від x: x=[1 2 3 4 5 6 7 8]; y=(x+1)^2-x+7; plot(x,y) В результаті отримаємо графік параболи.

Приклад. Формування заголовка і найменувань осей координат. x=0:0.1:2*%pi; plot(x,sin(x),"sin","time","plot of sinus")

Приклад. Два графіки в одних осях координат.

x=0:0.1:2*%pi; plot(cos(x)); plot(sin(x)) або правильніше (і тоді вони будуть різного кольору) використовувати конструкцію: x=0:0.1:2*%pi; plot([sin(x);cos(x)]) Зауваження: Функцію від одного аргументу можна (і краще) зобразити за допомогою команди plot2d.

## Як намалювати двовимірний графік?

С помощью команды plot2d. Обычно применяется для линейных графиков. Для уничтожения предыдущего содержания окна используйте команду xbasc(). Посмотрите демонстрационный пакет plot2d() Существуют аналогичные функции более высокого уровня: plot2d1 эквивалентна команде plot2d. plot2d2 : аналогична команде plot2d, но выглядит как гистограмма без вертикальных делений на прямоугольники (огибающая гистограммы). Является кусочно-непрерывной функцией. plot2d3 : аналогична команде plot2d, но кривая изображается в виде столбчатой диаграммы, у которой прорисованы только вертикальные линии. plot2d4 : аналогична команде plot2d, но кривая изображается стрелочками. Посмотрите демонстрационный пакет plot2d2(), plot2d3(), plot2d4().Синтаксис функции plot2d plot2d([x],y) plot2d([x],y,[opt_args]) plot2d([logflag],x,y,[style,strf,leg,rect,nax]) Параметры x,y : две матрицы или два вектора Если y является вектором, то x должен быть вектором той же размерности. Если значение x не задано, оно полагается равным вектору [1:]. Это означает, что если y=[2.2,4.4,9.3], то команда plot2d(y) будет считать, что x=[1,2,3]. Замечание: Применяя такие конструкции, очень легко получить в дальнейшем ошибку. Если y является матрицей, то x может быть: 1) вектором размера, равного размерности строки матрицы y Каждый столбец матрицы y будет изображаться относительно вектора x. 2) матрица того же размера, что и y. Каждый столбец y изображается относительно соответствующего столбца x. 3) если x не задан, он полагается равным вектору [1:[row dimension of y]].

logflag : строка формирующаяся с помощью characters h (для горизонтальной оси) и v (для вертикальной оси). Каждая из них может принимать значение "n" или "l". "l" означает логарифмический масштаб, а "n" - обычный. "ll" означает, что обе оси логарифмические. По умолчанию значение "nn".

Примеры. x=[0:0.1:2*%pi]'; plot2d(sin(x))

xbasc() // несколько графиков в одних осях plot2d(x,[sin(x) sin(2*x) sin(3*x)])

//оси справа xbasc() plot2d(x,sin(x),1,"183","sin(x)")

## Які ще є варіанти для зображення 2D-графика?

1)Команда plot2d2 - 2D графік у вигляді ступінчастої ламаної лінії. Повний синтаксис дивися за допомогою help plot2d2.

Команда plot2d2() дасть докладний демонстраційний приклад.

2)Команда plot2d3 - аналогічна команді plot2d, але крива зображається у вигляді стовпчастої діаграми, у якої промальовували тільки вертикальні лінії. Приклад. x=[0:0.1:2*%pi]'; xbasc(); plot2d3(x,sin(x))

Команда plot2d3() дасть докладніший демонстраційний приклад. 3)Команда plot2d4 аналогічна команді plot2d, але крива зображається стрілками від

однієї точки графіка до іншої. Приклад. x=[0:0.1:2*%pi]'; xbasc(); plot2d4(x,sin(x))

Команда plot2d4() дасть докладний демонстраційний приклад. Класичну стовпчасту гістограму можна отримати послідовно застосувавши команди plot2d2 і plot2d3.

Приклад. x=[0:0.1:2*%pi]'; plot2d2(x,sin(x)); plot2d3(x,sin(x))

Як намалювати гістограму?

За допомогою команд histplot і hist3d.
Приклад.
histplot([-6:0.2:6],rand(1,2000,'n'),[1-1],'011" ',[-6,0,6,0.5],[2,12,2,11]);

## 3D графіка

Як зобразити 3D поверхню?

Спосіб 1. За допомогою команди plot3d. Команда створює 3D графік по крапках, заданих матрицями x, y і z.

Спосіб 2. За допомогою команди plot3d1. Команда створює 3D графік по крапках, заданих матрицями x, y і z за допомогою рівнів кольору. Річ загалом надмірна: величина координати z додатково ще і пофарбована, залежно від значення z, що приймається.

Спосіб 3. За допомогою команди fplot3d. Це аналог команди fplot3d, але поверхня, що зображається, задана за допомогою зовнішньої функції. Спосіб 4. За допомогою команди fplot3d1. Це аналог команди plot3d1, але поверхня, що зображається, задана за допомогою зовнішньої функції. Синтаксис цих команд дивися за допомогою help. Приклад 1. t=-%pi:0.3:%pi; plot3d(t,t,sin(t) '*cos'(t),35,45,'X@Y@Z[2,2,4]);

Приклад 2. t=-%pi:0.3:%p i;plot3d1(t,t,sin(t) '*cos'(t),35,45,'X@Y@Z[2,2,4]);

Приклад 3. deff('[z]=surf(x,y)','z=sin(x)*cos(y)'); t=-%pi:0.3:%pi; fplot3d(t,t,surf,35,45,"X@Y@Z"); Приклад 4. deff('[z]=surf(x,y)','z=sin(x)*cos(y)'); t=-%pi:0.3:%pi; fplot3d1(t,t,surf,35,45,"X@Y@Z");

Як зобразити просторову криву у вигляді ізоліній на площині?

Спосіб 1. За допомогою команди contour. Просторова крива задається як z=f(x,y). Приклад. contour(1:5,1:10,rand(5,10),5);

Спосіб 2. За допомогою команди contour2d. Приклад. contour2d(1:10,1:10,rand"(10,10),5,1:5,"011" ",[0,0,11,11]);

Спосіб 3. За допомогою команди fcontour. Аналог команди contour, але просторова крива задана зовнішньою функцією. Приклад. deff('[z]=surf(x,y)','z=x**2+y**2'); fcontour(-1:0.1:1,-1:0.1:1,surf,10); Спосіб 4. За допомогою команди fcontour2d. Аналог команди countour2d, але просторова крива задана зовнішньою функцією.

Як зобразити 3D поверхню на 2D площині у вигляді заливки изообластей кольором?

Спосіб 1. За допомогою команди grayplot. Приклад. t=-%pi:0.1:%pi; m=sin(t) '*cos(t); grayplot(t,t,m);

Спосіб 2. За допомогою команди fgrayplot. Аналог grayplot, але просторова крива задана зовнішньою функцією. Приклад. deff('[z]=surf(x,y)','z=x**2+y**2'); fgrayplot(-1:0.1:1,-1:0.1:1,surf);

Спосіб 3. За допомогою команди Sgrayplot. Це те ж, що і grayplot, але кольори переходять в один одного поступово, без різкої межі (smooth). Приклад. t=-%pi:0.1:%pi; m=sin(t) '*cos(t); Sgrayplot(t,t,m);

Як намалювати 3D гістограму?

За допомогою команди hist3d Приклад. hist3d(10*rand(10,10))

**Зображення декількох малюнків в одному графічному вікні**

За допомогою команди subplot. Графічне вікно розділяється на "матрицю" з окремих областей ("підвікон"), в кожній з яких можна буде зобразити окремий графік. Синтаксис subplot(m,n,p) subplot(mnp) Параметри m, n, p : позитивні цілі числа mnp : ціле число. (Та ж конструкція, що і m, n, p, але з пропущеними комами) Графічне вікно розбивається на матрицю (m на n) підвікон. Поточним вибирається вікно з номером p.

Приклад. subplot(221) plot2d() subplot(222) plot3d() subplot(2,2,4) hist3d()
Зауваження: У вікні з номером "3" ми спеціально нічого не намалювали.

## Оформлення протоколу

***По ходу виконання роботи документувати всі виконувані дії і вносити їх до протоколу.***

Протокол по даній роботі повинен містити (стисло) наступні елементи Scilab:

1) цикли

2) умовні переходи

3) функції

4) 2D графіка

5) 3D графіка

## Лабораторна робота №3

## Дослідження системно-топологічних характеристик систем

**1.Мета роботи:** Вивчити методи, що дозволяють визначати структурні характеристики систем і давати їм кількісну оцінку.

**2. Загальні положення.**

Структуру системи прийнято зображати відповідним графом, що дозволяє використовувати при аналізі структурно-топологических характеристик системи добре розвинений апарат теорії графів.

У матричному виді граф можна задати матрицею суміжності вершин $A = |a_{ij}|$ елементи якої визначаються так:

$$a_{ij} = \begin{cases} 1, \text{если существует путь из вершины i в вершину j,} \\ 0 \text{ в противном случае} \end{cases}.$$

В даному випадку мова йде про неорієнтованому графові.

**2.1. Загальний аналіз структури**. Дана характеристика дозволяє виявити наявність обривів в структурі, ізольовані, висячі і тупикові вершини. Для цього по матриці суміжності графа $A = |a_{ij}|$ для кожної вершини *до (k*=$\overline{1,n}$,*n* – число вершин графа) визначається вектор $v(k) = (v_k, v^k)$ з компонентами:

$$v_k = \sum_{j=1}^{n} a_{kj}, \quad v^k = \sum_{i=1}^{n} a_{ik}.$$

Тобто *vk* є сума елементів *к-ой* рядка, а *vk* – *к-го* стовпця матриці суміжності. Величина визначає число ребер, що виходять з вершини *до*, а – число ребер, що входять в неї. Коли *vk* = *vk* =0, вершина *до* буде ізольованою, якщо *vk* =0 – тупиковою, при *vk* =0 – висячей.

Зв'язність структури для неорієнтованого графа відповідає виконанню умови

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} \geq n-1, i \neq j.$$

**2.2. Структурна надмірність.** Це структурний параметр, що відображає перевищення загального числа зв'язків над мінімально необхідним, він позначається *R.* Структурна надмірність вважається таким чином:

$$R = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij} \cdot \frac{1}{n-1} - 1.$$

Дана структурна характеристика використовується для оцінки економічності і надійності аналізованої системи. Для систем з максимальною надмірністю (повний граф, кліка) *R>0*, для систем з мінімальною надмірністю *R=0,* для систем незв'язкових *R<0*. Таким чином, система з більшою надмірністю *R* потенційно надійніша.

Структурна надмірність доповнюється іншим параметром, що враховує нерівномірність зв'язків в структурі $\varepsilon^2$. Рівномірний розподіл зв'язків в структурі неорієнтованого графа, що має m ребер і n вершин, характеризується середнім ступенем вершини $\overline{\rho} = 2m/n$. Тоді, ввівши поняття відхилення $\rho_i - \overline{\rho}$ де $\rho_i$ - дійсний ступінь і-й вершини даного графа (нагадаємо, що ступенем вершини *з(i)* називається число ребер, інцидентних даній вершині ), можна визначити среднеквадратическое відхилення заданого розподілу вершин від рівномірного:

$$\varepsilon^2 = \sum_{i=1}^{n}(\rho_i - \overline{\rho})^2 = \sum_{i=1}^{n}\rho_i^2 - 2\overline{\rho}\sum_{i=1}^{n}\rho_i + 4m^2/n = \sum_{i=1}^{n}\rho_i^2 - 2\cdot 2m/n \cdot 2m = 4m^2/n = \sum_{i=1}^{n}\rho_i^2 - 4m^2/n.$$

Показник $\varepsilon^2$ характеризує недовикористання можливостей даної структури, що має *m* ребер і *n* вершин, в досягненні максимальної зв'язності.

**2.3. Структурна компактність.**

Для кількісної оцінки структурної компактності вводиться параметр, що відображає близькість елементів між собою. Близькість двох елементів *i* і *j* визначатимемо як мінімальну довжину шляху $d_{ij}$. Тоді величина $Q = \sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij}\,(i \neq j)$ відображає загальну структурну близькість елементів між собою в системі. Дл оцінки структурної компактності використовується відносний показник

$Q_{rel} = \frac{Q}{Q_{\min}} - 1$ де $Q_{\min} = n(n-1)$ - мінімальне значення компактності для структури типу «повний граф».

Структурну компактність можна характеризувати і іншим показником – діаметром структури: $d = \max(d_{ij})$. Враховуючи переважаюче значення

інформаційних зв'язків в системах, можна сказати, що як величина $Q_{rel}$ так і $d$ інтегральний оцінюють інерційність процесів в системі, а при рівних значеннях $\varepsilon^2$ і *R* їх зростання відображає збільшення розділяючих зв'язків в системі, характеризуючи тим самим зниження загальної надійності.

Ступінь централізації в структурі. Для кількісної оцінки цього показника використовується індекс центральности:

$$\delta = (n-1)(2z_{\max} - n)\frac{1}{z_{\max}^{n-2}},$$

де $z_{\max}$ - максимальне значення величини

$$z_i = \frac{Q}{2}(\sum_{j=1}^{n} d_{ij}, i = 1,2,...,n; i \neq j.$$

Для структур, що мають максимальний ступінь централізації д=1, для структур з рівномірним розподілом зв'язків д=0.

Ранг елементу. Ця характеристика дозволяє розташувати елементи в порядку їх значущості. Чим більше ранг елементу, тим більше зв'язаний він з іншими елементами системи, і тим важче будуть наслідку при зміні якості його функціонування. Ранг елементу визначається по наступній формулі:

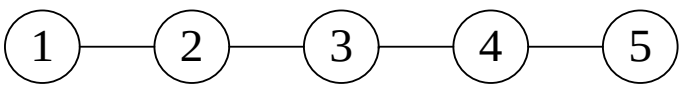
$$r_i = \frac{\sum_{j=1}^{n} a_{ij}^{(K)}}{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{(k)}},$$

де $a_{ij}^{(K)}$ - елементи матриці суміжності *A,* зведеною в ступінь *k=3-4*.

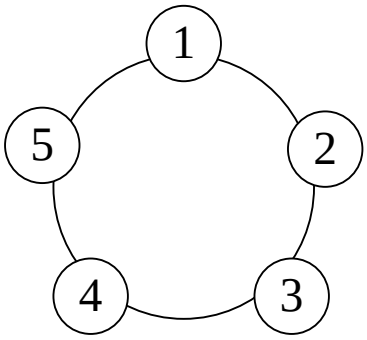
**2.4. Завдання на лабораторну роботу.**

2.4.1. У роботі досліджуються типові структури:

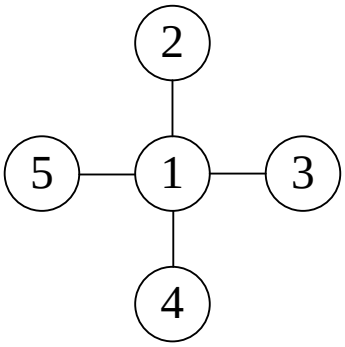
а) послідовна



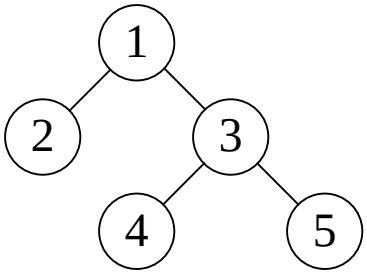

б) кільце



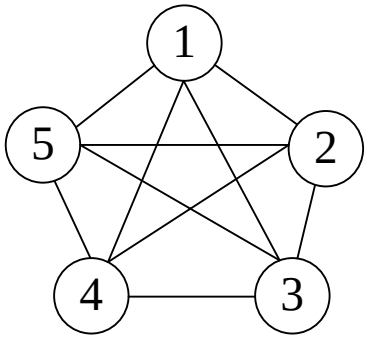

в) радіальна



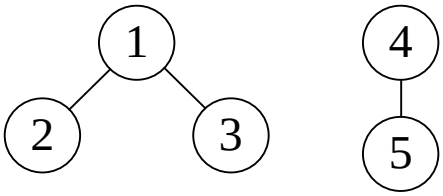

г) деревовидна




д) типу «повний граф»




е) незв'язкова.




Число елементів в кожній структурі рівне п'яти.

2.4.2. Для кожної структури виконати наступний аналіз:

- побудувати матрицю суміжності

- провести загальний аналіз структури

- обчислити показники структурної надмірності

- обчислити показники структурної компактності

- обчислити індекс центральности.

Всі розраховані показники звести в таблицю.

2.4.3. Для кожної структури обчислити ранги елементів.

2.4.4. Повторити аналіз для довільного графа, заданого викладачем для кожної бригади окремо.

**2.5. Виводи.** Зробити змістовні виводи по роботі в цілому (пояснити всі отримані чисельні результати)(всі проміжні виводи можуть бути віднесені в остаточні виводи по роботі).

**3. Методичні рекомендації.**

3.1. Робота виконується в системі програмування SciLab.

3.2. При формуванні матриць суміжності рекомендується задати нульову матрицю відповідної розмірності (наприклад, ***zeros(5,5)***), а потім у відповідні позиції матриці вносити одиничні елементи.

3.3. Для обчислення суми елементів матриці рекомендується користуватися оператором ***sum(a)***.

3.4. Для виділення стовпця (рядки) матриці рекомендується користуватися операторами ***a, a(1:).***

**Контрольні питання.**

1) Що таке структура системи?
2) Які відомі типові структури?
3) Які переваги і недоліки кожній із структур?
4) Привести приклади реальних систем з відповідними структурами.

# Лабораторна робота №4

## Дослідження системно-топологічних характеристик систем
## Частина 2

**1.Мета роботи:** Вивчити методи, що дозволяють упорядкувати структури систем.

**2. Загальні положення.**

**2.1. Впорядкування графа системи.** Метою впорядкування графа системної структури є розбиття безлічі вершин графа на непересічні множини, впорядковані так, що якщо вершина входить в підмножину з номером *i*, то наступна за нею вершина в підмножину з номером, великим *i*. Отримані непересічні множини називаються рівнями. Таким чином, по сенсу впорядкування графа системи є побудова з початкового графа ієрархічного графа. Впорядкування дає можливість побачити ієрархію в системі і, отже, скористатися цією ієрархією для ефективного управління системою.

Алгоритм впорядкування зводиться до наступного.

1) У підмножину нульового рівня *N0* включаються всі вершини *i*, у яких

$G\text{-}1(i)=\overline{0}$ (порожня множина). (нагадаємо, що для орієнтованого графа вводяться дві множини – безліч вершин, в які можна безпосередньо потрапити з вершини *i*, воно позначається *G (i)* і називається безліччю правих инциденций, аналогічна безліч всіх вершин, з яких можна потрапити у вершину *i*, називається безліччю лівих инциденций і позначається *G-1(i)*). Проводиться послідовна нумерація вершин: 1,2 ., *l*.

2) У підмножину першого рівня *N1* включаються всі вершини *i*, у яких

$G\text{-}1(i)\subset N0.$ Проводиться послідовна нумерація вершин: *l+1, l+2 .,l+r.*

3) У підмножину другого рівня *N2* включаються всі вершини *i*, у яких

$G\text{-}1(i)\subset(N0\cup N1)$. Проводиться послідовна нумерація вершин: *l+r+1 .,l+r+p.*

4) У підмножину третього рівня *N3* включаються всі вершини *i*, у яких

$G\text{-}1(i)\subset(N0\cup N1\cup N2)$, після чого також проводиться нумерація вершин.

Даний процес продовжується до тих пір, поки не будуть перенумеровані всі вершини графа.

**2.2. Виділення комплексів (зв'язних підструктур).**

У структурі системи завжди можна виділити зв'язні підструктури, які для технічних систем називають комплексами. Для цього існують два алгоритми.

1) Алгоритм з використанням матриці шляхів на графі.

Будується квадратна матриця *P*, число рядків і стовпців якої рівно числу елементів в структурі. Якщо на графі є шлях будь-якої довжини з вершини *i* у вершину *j*, то на перетині *i-й* рядка і *j-го* стовпця ставиться одиниця, інакше – нуль. Далі на підставі матриці *P* будується допоміжна матриця *S*:

$$S_{ij} = \begin{Bmatrix} 1, \textit{если} p_{ij} = p_{j1} = 1 \\ 0, \textit{впротивномслучае} \end{Bmatrix}.$$

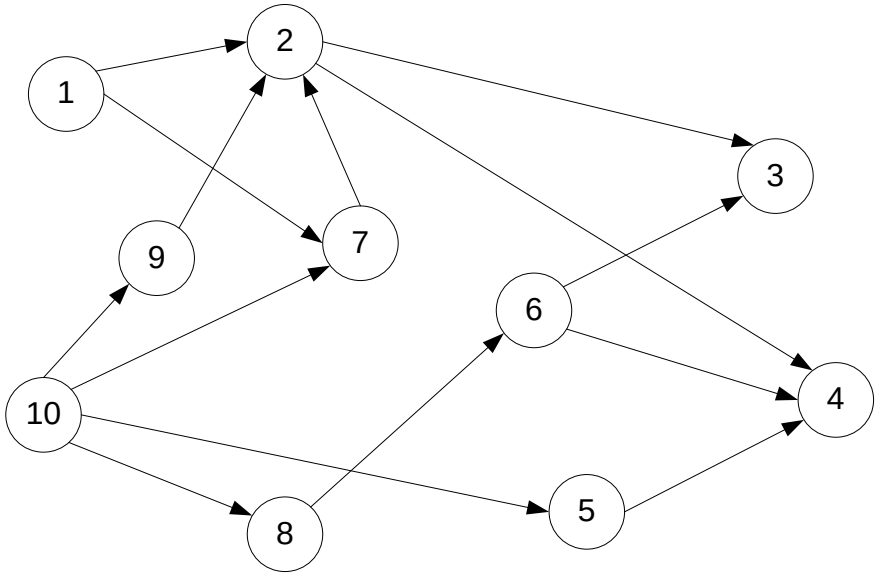
На підставі матриці *S* визначаються комплекси, що входять до складу структури. Якщо в *i-й* рядку є тільки один ненульовий елемент *Sii* (на головній діагоналі), то елемент з номером *i* можна розглядати окремо від решти елементів. Рядки матриці *S*, які мають окрім діагонального інші ненульові елементи – комплекси (підструктури). Ненульові елементи показують вершини графа, які входять в комплекс.

Приведений алгоритм вимагає багато ручної праці.

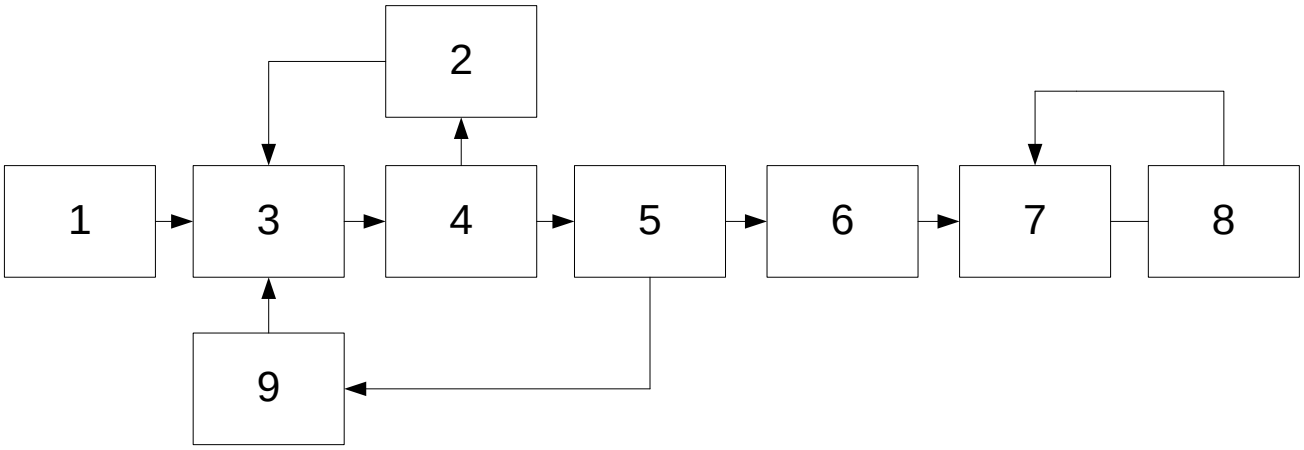
2) Альтернативний алгоритм. Розглядаються три будь-які вершини графа *i,j,m*.Если існує шлях(будь-якої довжини!) з вершини *i* у у вершину *j* і з *j* в *i*, то обидві ці вершини належать комплексу *до*. Для приєднання вершини *m* до комплексу *до* необхідно проаналізувати: чи є шлях з будь-якої вершини комплексу (наприклад, *i*) у вершину *m* і зворотний шлях з вершини *m* в будь-яку вершину комплексу *до* (наприклад, в *i*). Якщо ці два шляхи існують, то вершина *m* належить комплексу *до.*

**2.4. Завдання на лабораторну роботу.**

2. Для структури, граф, якою приведений на малюнку, провести впорядкування структури. Побудувати матриці суміжності до впорядкування і після впорядкування, з'ясувати, чим вони відрізняються.




2.4.2. Для структури, заданої графом, приведеним на малюнку, виділити комплекси двома алгоритмами.




**2.5. Виводи.** Зробити змістовні виводи по роботі в цілому (зобразити графи до аналізу і після аналізу, пояснити всі отримані чисельні результати)(всі проміжні виводи можуть бути віднесені в остаточні виводи по роботі).

**Контрольні питання.**

1) Що таке ієрархія?
2) Що дає впорядкування структури?
3) Що таке множина правих і лівих инциденций?
4) Для чого потрібно виділяти в структурі комплекси (підструктури) ?

# Лабораторна робота №5

## Дослідження системно-топологічних характеристик систем
## Частина 3

**1.Мета роботи:** Вивчити методи, що дозволяють оцінити складність структури і максимальні шляхи в ній.

**2. Загальні положення.**

**2.1. Складність структури системи.** Якщо функціонування системи уявити собі, як процес переробки вхідних дій у вихідні, направлений від входів до виходів, то природно припустити, що вивчати властивості цього процесу буде тим важче, чим різноманітніше шляхи, що ведуть від входу системи до її виходу.

Тоді для оцінки складності структури пропонується показник складності

$$\rho = \frac{1}{m_1 m_2} \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \rho_{ij} - 1,$$

де $m_1$ і $m_2$ - число висячих і тупикових шляхів в графі структури $\rho_{ij}$ - число різних шляхів, ведучих з *i-й* висячої структури в *j-ю* тупикову.

Згідно останньому виразу величина $\rho$ чисельно рівна середньому числу шляхів, ведучих з висячої вершини в тупикову, зменшеному на одиницю. У структурі з мінімальною складністю з кожної висячої вершини в тупикову можна потрапити єдиним шляхом (рис.1).

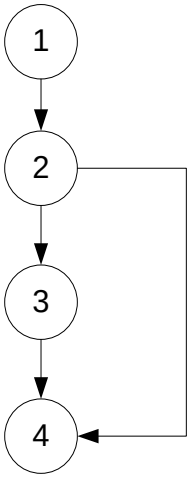


Рис 1

Для такої структури $\rho$=0. Граф на рис.2 містить значно менше вершин, чим граф на рис.1. Проте він структурно складніше, тому для нього показник складності $\rho$=1.

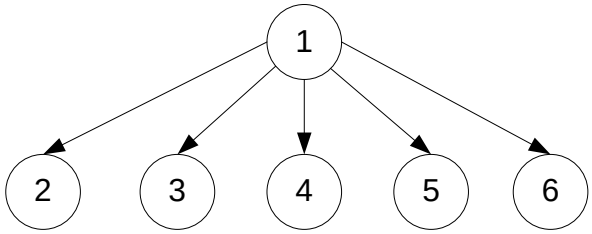


Рис 2

Загальний підхід, що дозволяє обчислити $\rho$ для будь-якого орієнтованого графа, полягає в наступному.

Ш а г 1. Початковий граф представляється у вигляді багаторівневого ієрархічного графа.

Ш а г 2. Ієрархічний граф, отриманий на кроці 1, перетвориться в еквівалентний йому граф, що не містить суміжних вершин на одному рівні ієрархії.

Ш а г 3. Отриманий в результаті граф представляється сукупністю підграфів.

Ш а г 4. Перемножуючи матриці інцидентності підграфів, отримуємо матрицю $W = \left|\rho_{ij}\right|$ розмірності $m_1$ на $m_2$ .Суммируя її елементи, обчислюють показник $\rho$.

Приклад. Обчислимо значення $\rho$ для графа, зображеного на рис.3.

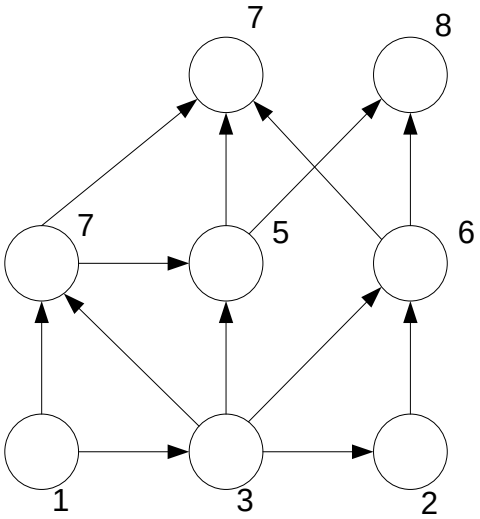


Рис 3

Перш за все зобразимо його у вигляді ієрархічного графа, в якому рівень ієрархії кожної вершини визначається мінімальним числом ребер, що пов'язує її з висячими вершинами. Тоді отримаємо трирівневий граф, показаний на мал. 4.

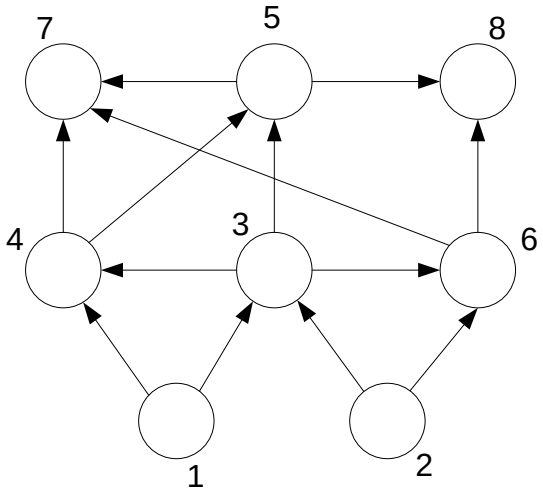



Рис 4

Проте у нього є ребра, що зв'язують вершини одного рівня. Щоб позбавитися від цих ребер, всі інцидентні ним вершини продублюємо фіктивними вершинами, розташованими на один рівень ієрархії вище. На мал. 5 фіктивних вершин позначені номерами з штрихом.

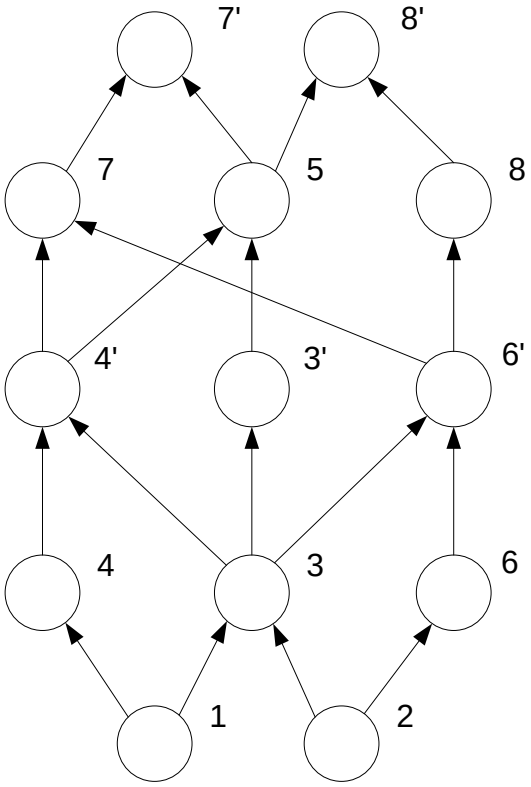



Рис 5

Введення фіктивних вершин дозволяє замінити зв'язки між вершинами одного рівня еквівалентними ним зв'язками між вершинами сусідніх рівнів ієрархії. Так ребро графа(3,4) на мал. 4 замінюється ребром (3,4’) на рис.5. Аналогічно проводиться заміна ребер (3,6) (5,7). (5,8) на ребра (3,6’) (5,7’) (5,8’). В результаті можна перейти до кроку 3.

Отриманий п'ятирівневий граф описується чотирма підграфами з матрицями инциденций:

$$W_1 = \begin{array}{c|ccc} & 4 & 3 & 6 \\ \hline 1 & 1 & 1 & 0 \\ 2 & 0 & 1 & 1 \end{array}, W_2 = \begin{array}{c|ccc} & 4' & 3' & 6' \\ \hline 4 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 \end{array}, W_3 = \begin{array}{c|ccc} & 7 & 5 & 8 \\ \hline 4' & 1 & 1 & 0 \\ 3' & 0 & 1 & 0 \\ 6' & 1 & 0 & 0 \end{array}, W_4 = \begin{array}{c|cc} & 7' & 8' \\ \hline 7 & 1 & 0 \\ 5 & 1 & 1 \\ 8 & 0 & 1 \end{array}.$$

Перемножуючи ці матриці, отримаємо:

$$W = W_1W_2W_3W_4 = \begin{array}{c|ccc} & 4 & 3 & 6 \\ \hline 1 & 1 & 1 & 0 \\ 2 & 0 & 1 & 1 \end{array} * \begin{array}{c|ccc} & 4' & 3' & 6' \\ \hline 4 & 1 & 0 & 0 \\ 3 & 1 & 1 & 1 \\ 6 & 0 & 0 & 1 \end{array} * W_3 * W_4 = \begin{array}{c|ccc} & 4' & 3' & 6' \\ \hline 1 & 2 & 1 & 1 \\ 2 & 1 & 1 & 2 \end{array} * \begin{array}{c|ccc} & 7 & 5 & 8 \\ \hline 4' & 1 & 1 & 1 \\ 3' & 0 & 1 & 1 \\ 6' & 1 & 0 & 1 \end{array} * W_4 =$$

$$= \begin{array}{c|ccc} & 7 & 5 & 8 \\ \hline 1 & 3 & 3 & 1 \\ 2 & 3 & 2 & 2 \end{array} * \begin{array}{c|cc} & 7' & 8' \\ \hline 7 & 1 & 0 \\ 5 & 1 & 1 \\ 8 & 0 & 1 \end{array} = \begin{array}{c|cc} & 7' & 8' \\ \hline 1 & 6 & 4 \\ 2 & 5 & 4 \end{array}.$$

Звідси ясно, що $\rho_{17} = 6, \rho_{18} = 4, \rho_{27} = 5, \rho_{28} = 4,$ а оскільки для початкового графа (рис.3) $m_1 = m_2 = 2,$ то $\rho$ =0,25(6+4+5+4)-1=3,75.

**2.2. Визначення максимального шляху на графі системи.**

Для графа без контурів визначення максимального шляху реалізується на підставі наступного функціонального рівняння:

$$q_s^{макс}(a_1a_j) = \max_{a_i \in G^{-1}(a_j)} \left[ q_s^{макс}(a_1a_j) + q(a_ia_j) \right],$$

где $q_s^{макс}(a_1a_j)$ - максимальное значение функции на путях *S* из некоторой начальной вершины $a_1$ в вершину $a_j$ (аналогично $q_s^{макс}(a_1a_i)$), $G^{-1}(a_j)$ - множество левых инциденции для вершины $a_j$, $q(a_ia_j)$ - значение функции на дуге $(a_ia_j)$.

Приклад. Для графа на рис.6 знайдемо максимальний шлях з вершини $a_1$ у вершину $a_7$.

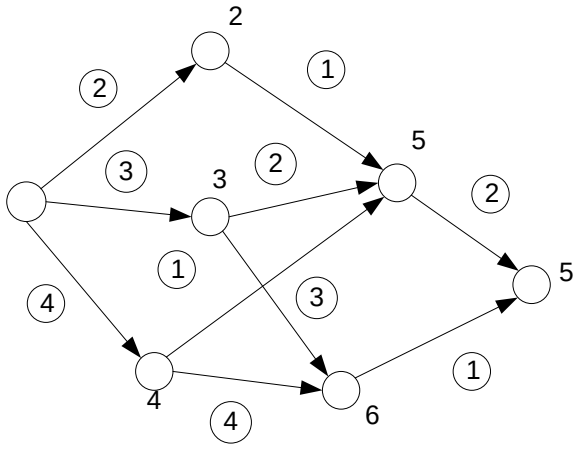



Рис 6

Для вершини $a_1$ приймаємо $q_s^{макс}(a_1a_1)=0$. Для вершин $a_2,a_3,a_4: q_s^{макс}(a_1a_2)=2,$ $q_s^{макс}(a_1a_3)=3, q_s^{макс}(a_1a_4)=4.$ Для вершини $a_5: q_s^{макс}(a_1a_5)=макс(2+1,3+2,4+1)=5.$ Для вершини $a_6: q_s^{макс}(a_1a_6)=макс(3+3,4+4)=5.$ Для вершини $a7: q_s^{макс}(a_1a_7)=макс(5+2,8+1)=9.$ Значення функції на максимальному шляху рівне дев'яти, а сам шлях виділений жирною лінією.

**2.4. Завдання на лабораторну роботу.**

2.4.1. Розрахувати показник складності $\rho$ для графів на рис.1, рис.2, повторити розрахунок для графа на рис.3-5, провести розрахунок для графа, заданого викладачем.

2.4.2. Повторити розрахунок максимального шляху на графі рис.6 і провести розрахунок для довільного графа, заданого викладачем.

**2.5. Виводи.** Зробити змістовні виводи по роботі в цілому (зобразити досліджувані графи, пояснити всі отримані чисельні результати)(всі проміжні виводи можуть бути віднесені в остаточні виводи по роботі).

**Контрольні питання.**

5) Що таке ієрархія?
6) Що таке складність структури?
7) Що таке множина правих і лівих инциденций?
8) Для чого потрібно виділяти в структурі максимальні шляхи ?_

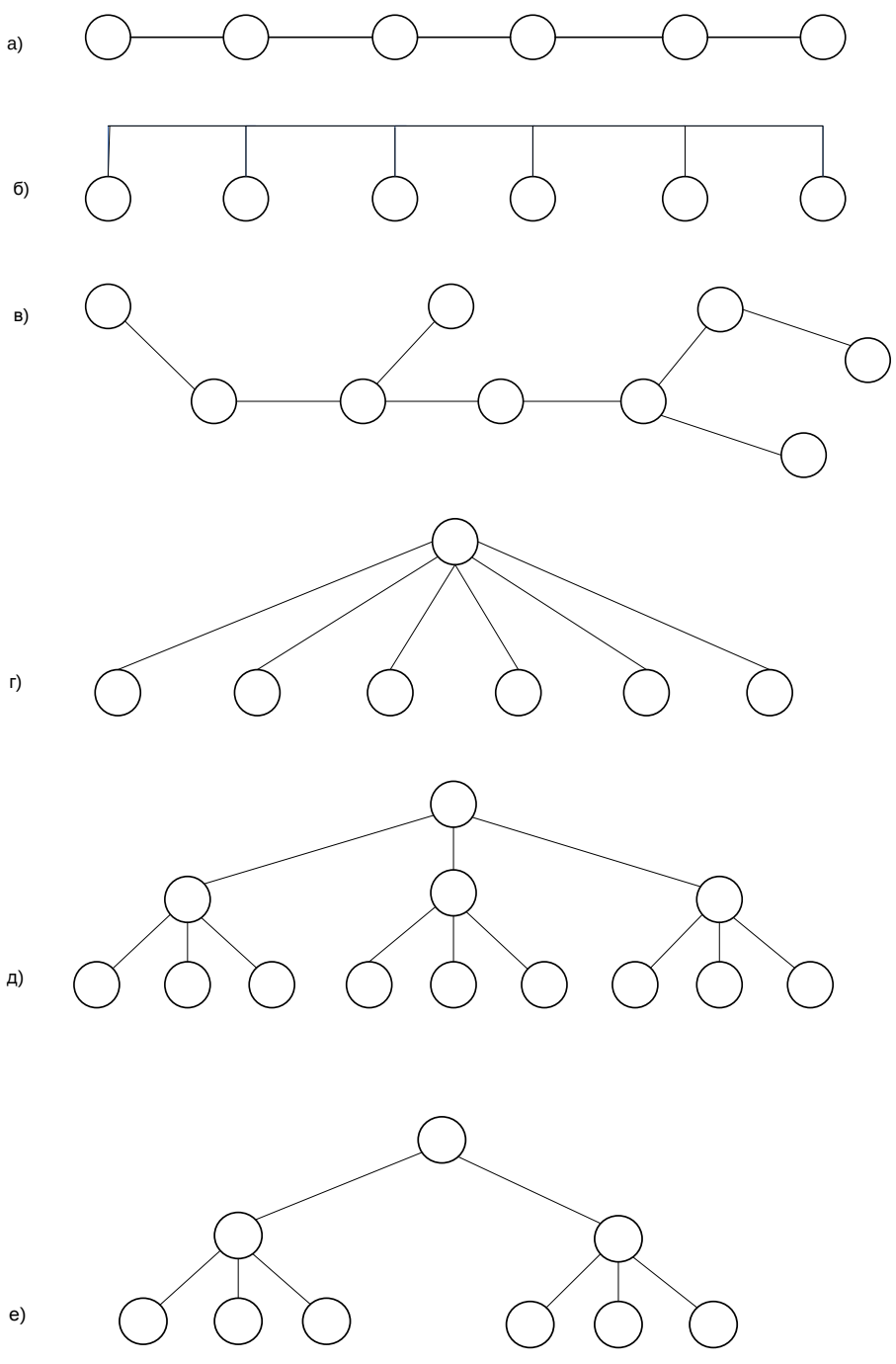
## Лабораторна робота №6

# Побудова мережевих структур комп'ютерних систем систем

**1.Мета роботи:** Вивчити методи і алгоритми, що дозволяють побудувати необхідну системну структуру.

**2. Загальні положення.**

**2.1.** Достатньо часто в системах використовуються деревовидні структури, тобто такі конфігурації, в яких від одного вузла (вершини) до іншого можна пройти єдиним шляхом (рис.1).



Достоїнствами деревовидних структур є їх висока економічність, що забезпечується мінімальною зв'язністю, спрощені алгоритми маршрутизації, оскільки з одного вузла в іншій можна прийти єдиним шляхом. Недолік деревовидних структур полягає в їх невисокій надійності, оскільки у разі відмови одному зв'язку вся структура розчленовується на дві незв'язні області.

Структура системи *S*, що містить в своєму складі безліч вузлів $a_i, i = 1,..., N$ вважається побудованою, якщо визначений набір дуг $x_{ij} \in (0,1)$ що забезпечує з'єднання всіх пар вузлів $a_i$ і $a_j$ у єдину мережу.

**2.2.** Побудова структури деревовидної конфігурації довільної форми.

Завдання формулюється таким чином.

Задано: безліч вузлів $\{a_i\}, i = 1,.., N$ мережі і повний граф зв'язків $m_{ij}$ якими можуть бути сполучені вузли. Потрібний: знайти деревовидну зв'язну мережу, яка має мінімальну сумарну средневзвешенную довжину.

Математичне формулювання:

$$Q = \min_{x_{ij}} \sum_{i=1}^{N} \sum_{j=1}^{n} m_{ij} x_{ij},$$

де $m_{ij}$ - зважена довжина зв'язку між *i-м* і *j-м* вузлами

$$x_{ij} = \begin{cases} 1, m_{ij} \in S, \\ 0, m_{ij} \notin S \end{cases}$$

- шукані булеві змінні.

Найефективніше таке завдання вирішується по алгоритму Прима. У основу алгоритму Прима покладено два принципи.

А. всякий ізольований вузол з'єднується з найближчим сусідом (найближчим сусідом вузла є той, який знаходиться від нього на відстаней, не більшому, ніж будь-який інший вузол).

Б. Всякий ізольований фрагмент з'єднується з найближчим сусідом найкоротшою дугою (ізольований фрагмент- це підмножина вузлів, зв'язана дугами, але що не охоплює всіх заданих вузлів).

Розглянемо процедуру реалізації алгоритму Прима.

1. Введення початкових даних: $N$ -кількість вузлів, що сполучаються $M$ - матриця зважених відстаней розмірністю $N \times N$ у якій елементи , - значення цільової функції $\Phi = \{0\}$- список вузлів, що входять у фрагмент $\|x_{ij}\| = 0$ - матриця, в яку зберуться результати розрахунку структури.

2. Проглядаючи початкову матрицю $M$ по рядках і по стовпцях, визначаємо мінімальну дугу $m_{kl}$.

3. Заносимо в список $\Phi$ вузлів, що входять у фрагмент, номери вузлів *k,l,* відповідних індексам рядків і стовпців знайденого елементу $m_{kl}$. Елемент матриці $x_{kl} = x_{lk} = 1.$Обчислюємо поточне значення цільової функції $Q = Q + m_{kl}$.

4. Викреслюємо з початкової матриці $M$ ті стовпці, які увійшли до списку вузлів $\Phi$ що входять у фрагмент, і отримуємо скоректовану матрицю $M'$.

5. З скоректованої матриці $M'$ виділяємо рядки, номери яких входять в список $\Phi$. Проглядаємо вказані рядки поелементно, знаходимо мінімальний елемент $m_{kl}$.

6. Заносимо в список вузлів $\Phi$ що входять у фрагмент, номери стовпця l знайденого мінімального елементу $m_{kl}$ відповідних значенням $x_{kl} = x_{lk} = 1.$ Обчислюємо $Q = Q + m_{kl}$.

7. Якщо число елементів $|\Phi|$ записаних в списку $\Phi$ вузлів, що входять у фрагмент, рівно числу вузлів $N$ мережі, то переходиться до п.8, якщо $|\Phi| < N$ то до п.4.

8. Вихід, виведення результатів розрахунку матриці *X*, значення цільової функції $Q$ і побудова графа мережі.

Не дивлячись на простоту, процедура Прима дозволяє відшукати глобальний максимум, що зазвичай рідко вдається.

Приклад.

1. Задана початкова матриця розмірністю 5*5:

$$m = \begin{Vmatrix} - & 37 & 29 & 75 & 78 \\ 37 & - & 21 & 61 & 42 \\ 29 & 21 & - & 49 & 52 \\ 75 & 61 & 49 & - & 57 \\ 78 & 42 & 52 & 57 & - \end{Vmatrix}.$$

2. $m_{23} = 21$.

3. $\Phi = \{2,3\}$, $x_{23} = x_{32} = 1$, $Q$=21.

4. Викреслюємо 2-й і 3-й стовпці матриці $M$ і отримуємо $M'$.

5. $m_{31} = 29$ (мінімальний елемент 2-го і 3-го рядків матриці).

6. $\Phi = \{2,3,1\}$, $x_{31} = x_{13} = 1$, $Q$=50.

7. $|\Phi|$=3, переходимо до п.4.

4'. Викреслюємо 1-3-й стовпці матриці $M$.

5'. $m_{25} = 42$.

6'. $\Phi = \{2,3,1,5\}$, $x_{25} = x_{52} = 1$, $Q$=92.

7' і 5''. $m_{34} = 49$.

6''. $\Phi = \{2,3,1,5,4\}$, $x_{34} = x_{43}$=1, $Q$=141.

7 і 8. Результат

$$X = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 \\ 3 & 1 & 1 & 0 & 1 & 0 \\ 4 & 0 & 0 & 1 & 0 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \end{array}.$$

**3.** Повторити розрахунок дл приведеного прикладу допомогою Scilab, побудувати граф деревовидної структури і провести розрахунок для довільної матриці, заданої викладачем.

**2.5. Виводи.** Зробити змістовні виводи по роботі в цілому (зобразити структурні графи, пояснити всі отримані чисельні результати)(всі проміжні виводи можуть бути віднесені в остаточні виводи по роботі).

**Контрольні питання.**

9) Що таке деревовидна структура?
10) Які переваги деревовидної структури?
11) Які недоліки деревовидної структури?
12) Коротка схема алгоритму Прима.

# Лабораторна робота №7

## Аналіз принципу необхідної різноманітності Ешбі

**1.Мета роботи:** Вивчити принцип необхідної різноманітності і його особливості.

**2. Загальні положення.**

Принцип необхідної різноманітності Ешбі може бути сформульований так:

$$H(X)_{ТРЕБ} \geq H(Y),$$

що означає: необхідна різноманітність системи управління повинна бути не менше різноманітності керованої системи.

**2. Завдання на лабораторну роботу.**

2.1. Визначити необхідну ентропію системи управління $H(X)_{ТРЕБ}$ за умови, що об'єкт управління $Y$ може знаходитися в двох станах $\{y_1, y_2\}$.

Зокрема, хай об'єкт управління $Y$ - це системний адміністратор локальної обчислювальної мережі. Стан $y_1$ означає належне виконання ним посадових обов'язків, стан $y_2$ - відхилення від правильного виконання службових обов'язків.

Вірогідність знаходження $p_i$ у різних станах такі:

1. $p_1 = 0{,}5$ для стану $y_1$ $p_2 =$

2. $p_1 = 0{,}9$ для стану $y_1$ $p_2 =$

3. $p_1 = 0{,}1$ для стану $y_1$ $p_2 =$

Розрахувати необхідну ентропію системи управління для кожного випадку.

2.2. Локальна обчислювальна мережа може знаходитися в трьох станах: $y_1$ - повністю працездатному $y_2$ - частково працездатному (наприклад, відомо, що зроблена вірусна атака, але тип вірусу знаходиться у стадії виявлення і знищення) $y_3$ - непрацездатному (наприклад, через відсутність електроенергії і відмови джерела безперебійного живлення сервера мережі).

Для тих наборів вірогідності станів, заданих викладачем, розрахувати необхідну ентропію системи управління і зробити виводи.

**2.5. Виводи.** Зробити  змістовні виводи по роботі в цілому (

**Контрольні питання.**

5) Назвіть п'ять компонентів управління?
6) Перерахуєте сім типів управління?
7) Перерахуєте аксіоми управління?
8) Принцип необхідної різноманітності Ешбі.

## Лабораторна робота №8

## Метод аналізу ієрархій

**1.Мета роботи:** Оволодіти на практиці методом аналізу ієрархій.

**2. Загальні положення.**

Метод аналізу ієрархій (ТРАВЕНЬ) розроблений Т. Сааті для вирішення многокритериаль-ных складних проблем. Постановка завдання для ТРАВЕНЬ полягає зазвичай в наступному.

Дано: 1) загальна мета рішення задачі;2) N критеріїв оцінки альтернатив; 3) n альтернатив, з яких треба здійснити вибір.

Перший етап рішення задачі полягає в її структуризації у вигляді ієрархії з декількома рівнями (зокрема, з трьома): мета – критерії – альтернативи.

Уровень1 Мета

Уровень2 Критерій 1 Критерій 2 ... Критерій і ... Критерій N

Рівень 3 Альтернатива 1 ... Альтернатива і ... Альтернатива n

На другому рівні виконуються попарні порівняння для кожного рівня. Попарні порівняння переводяться в матрицю парних порівнянь за допомогою наступної таблиці.

Шкала відносної важливості

| Рівень важливості | Кількісне значення |
|---|---|
| Рівна важливість | 1 |
| Помірна перевага | 3 |
| Істотна або сильна перевага | 5 |
| Значна перевага | 7 |
| Дуже велика перевага | 9 |

| | |
|---|---|
| Проміжні значення | 2,4,6,8 |

На кожному рівні ієрархії (окрім першого) будується так звана матриця парних порівнянь.

$$A=\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}=\begin{vmatrix} w_1/w_1 & w_1/w_2 & \dots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \dots & w_2/w_n \\ \dots & \dots & \dots & \dots \\ w_n/w_1 & w_n/w_2 & \dots & w_n/w_n \end{vmatrix}.$$

Якщо елемент порівняння $B_i$ домінує, тобто є кращим, ніж елемент $B_j$ то експерт визначає ступінь домінування за приведеною вище шкалою і відповідне значення в балах привласнюється елементу $a_{ij}$ а значення 1/ $a_{ij}$ - елементу $a_{ji}$ тобто матриця парних порівнянь назад симетрична, тобто елементи, симетричні щодо головної діагоналі, є зворотними по відношенню один до одного. Елементи головної діагоналі – одиниці.

Далі на підставі матриці парних порівнянь формуються локальні пріоритети шляхом визначення головних власних векторів матриці *A* і нормалізації результату. Обчислення головного власного вектора $x=\{x_1, x_2, \dots, x_n\}$ позитивної квадратної матриці *A* здійснюється на підставі визначення $Ax=\lambda_{\max} x$ де $\lambda_{\max}$ - максимальне власне число матриці *A. Вичисленіє* спектру матриці (тобто набору її власних чисел і векторів) – складна обчислювальна процедура. Тому в ТРАВЕНЬ власні вектора обчислюються приблизно на підставі граничної формули $\lim\limits_{k\to\infty}\frac{A^k e}{e^T A^k e}=x$ _

де *e* – одиничний вектор. Насправді (у практичних розрахунках ТРАВЕНЬ) пріоритети (власні вектори) вважають як середнє геометричне стовпців матриці парних порівнянь з подальшою нормалізацією:

$$x_i=\sqrt[n]{(\prod_{j=1}^{n} a_{ij})}\ /\ \sum_{i=1}^{n}(\sqrt[n]{\prod_{j=1}^{n} a_{ij}}).$$

Далі вважається індекс узгодженості, який дає можливість перевірити, наскільки послідовний був експерт, заповнюючи матрицю парних порівнянь.

$Ic = \frac{\sum_{j=1}^{n}(x_j \times \sum_{i=1}^{n} a_{ij}) - n}{n-1} = \frac{\lambda_{\max} - n}{n-1}$ звідки оцінка $\lambda_{\max} = \sum_{j=1}^{n}(x_j \times \sum_{i=1}^{n} a_{ij})$.

Обчислений індекс узгодженості *Ic* порівнюється із значенням, яке виходить за умови випадкового вибору кількісних значень з шкали 9, 8, 7.,1/8, 1/9.

Значення індексу узгодженості для випадкових матриць

| Розмір матриці | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Випадкова узгодженість | 0 | 0 | 0,58 | 0,90 | 1,12 | 1,24 | 1,32 | 1,41 | 1,45 | 1,49 |

Відношення узгодженості є приватне від ділення індексу узгодженості на відповідне значення випадкової узгодженості, узяте з таблиці: *C=ic/rand(Ic,n)*. Якщо набутого значення менше 10%, то рівень узгодженості вважається задовільним.

Далі розраховуються глобальні пріоритети. Як це робиться, буде ясно з прикладу.

**Приклад (складений самим Т.Сааті, автором ТРАВЕНЬ для журналу ”Системні дослідження та інформаційні технології” №1, 2002). Вибір чоловіка для Оксани.**

Оксана має трьох кандидатів в чоловіки. Критерії її вибору: вік, зовнішність, інтелігентність і економічний стан.

1.Тарас, 30 років, інженер з хорошим доходом, зацікавлений в кар'єрі і, створенні сім'ї і її процвітанні. Довгоносий, благородний, любить Оксану.

2. Іван, 37 років, багатообіцяючий художник, але доходи його нестабільні, але у нього є багато пропозицій. Красивий, духовний, пашить здоров'ям.

3. Григорій, 25 років, молода людина з процвітаючої бизнес-семьи з блискучим майбутнім. Недалекий і злегка дурнуватий.

Ієрархія виглядає таким чином.

**Мета: Кращий чоловік для Оксани**

**Вік** **Зовнішність** **Інтелігентність** **Забезпеченість**

**Тарас** **Іван** **Григорій**

Володіючи ТРАВЕНЬ, Оксана склала матрицю парних порівнянь для першого рівня ієрархії – критеріїв.

Порівняння критеріїв по ступеню їх важливості для досягнення мети

| | Вік | Зовнішність | Інтелігентність | Забезпеченість | Пріоритети |
|---|---|---|---|---|---|
| Вік | 1 | 1/7 | 1/5 | 1/3 | 0,062 |
| Зовнішність | 7 | 1 | 1/2 | 2 | 0,327 |
| Інтелігентність | 5 | 2 | 1 | 2 | 0,429 |
| Забезпеченість | 3 | 1/2 | 1/2 | 1 | 0,182 |

Відношення узгодженості С=4%

Далі Оксана оцінює кандидатів по кожному з критеріїв на третьому рівні ієрархії.

| Вік | Тарас | Іван | Григорій | Пріоритети |
|---|---|---|---|---|
| Тарас | 1 | 3 | 1/3 | 0,258 |
| Іван | 1/3 | 1 | 5 | 0,105 |
| Григорій | 3 | 1/5 | 1 | 0,637 |

Відношення узгодженості С=4%

| Зовнішність | Тарас | Іван | Григорій | Пріоритети |
|---|---|---|---|---|
| Тарас | 1 | 1/5 | 2 | 0,166 |
| Іван | 5 | 1 | 7 | 0,740 |
| Григорій | 1/2 | 1/7 | 1 | 0,094 |

Відношення узгодженості С=1%

| Інтелігентність | Тарас | Іван | Григорій | Пріоритети |
|---|---|---|---|---|
| Тарас | 1 | 1/5 | 3 | 0,188 |
| Іван | 5 | 1 | 7 | 0,731 |
| Григорій | 1/ 3 | 1/7 | 1 | 0,081 |

Відношення узгодженості С=6%

| Забезпеченість | Тарас | Іван | Григорій | Пріоритети |
|---|---|---|---|---|
| Тарас | 1 | 5 | 1/4 | 0,237 |
| Іван | 1/5 | 1 | 1/8 | 0,064 |
| Григорій | 4 | 8 | 1 | 0,699 |

Відношення узгодженості С=9%

Нарешті, в останній таблиці ваги кандидатів умножаються на ваги критеріїв і складаються.

| Критерії / Кандидати | Вік (0,062) | Зовнішність (0,327) | Інтелігентність (0,429) | Забезпеченість (0,182) | Підсумкови пріоритет |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| Тарас | 0,256 | 0,166 | 0,188 | 0,237 | 0,195 |
| Іван | 0,105 | 0,740 | 0,731 | 0,064 | 0,537 |
| Григорій | 0,637 | 0,094 | 0,081 | 0,699 | 0,232 |

Отже, підсумковий вектор пріоритетів: (0,195; 0,537; 0,232). Аналіз ТРАВЕНЬ закінчений. Оксана повинна вийти заміж за Івана, який переміг завдяки красивій зовнішності і інтелігентності.

**2. Завдання на лабораторну роботу.**

2.1. Користуючись матричним апаратом *Scilab*, запрограмувати ТРАВЕНЬ і повторити розрахунки з чоловіком для Оксани.

2.2.Два підприємства: X (комерційна компанія, що працює на зовнішньому ринку) і Y (державне підприємство з вельми скромним бюджетом) вибирають, якій з трьох пропонованих на ринку операційних систем оснастити свій програмно-апаратний комплекс. Для вибору необхідною підприємству операційної системи задані критерії оцінки:

A1 – вартість

A2 - доступність замовленої розробки

A3 – підтримка режиму жорсткого реального часу

A4 – наявність навченого персоналу.

Дані про операційні системи

| | A1 | A2 | A3 | A4 |
|---|---|---|---|---|
| ОС-1 | 150 у.о. | Немає | Немає | Так |
| ОС-2 | 3000 у.о. | Немає | Так | Немає |
| ОС-3 | 40000 у.о. | Так | Так | Немає |

На підприємствах X і Y після бурхливих дебатів експерти вказали відносні ваги критеріїв.

Підприємство X.

| | A1 | A2 | A3 | A4 |
|---|---|---|---|---|
| A1 | 1 | 9 | 7 | 3 |
| A2 | 1/9 | 1 | 1/5 | 1/9 |
| A3 | 1,7 | 5 | 1 | 1/7 |
| A4 | 1/3 | 9 | 7 | 1 |

Підприємство Y.

| | A1 | A2 | A3 | A4 |
|---|---|---|---|---|
| A1 | 1 | 1/5 | 1/9 | 1/3 |
| A2 | 5 | 1 | 1/3 | 5 |
| A3 | 9 | 3 | 1 | 7 |
| A4 | 3 | 1/5 | 1/7 | 1 |

Знайти пріоритети критеріїв. Поставивши себе на місце експертів на обох підприємствах побудувати матриці парних порівнянь на нижньому рівні. Побудувати вектор остаточних пріоритетів. Зробити вибір операційної системи для підприємств X і Y.

2.3. **Тільки для студентів, що претендують на високий бал.** Скласти і вирішити ТРАВЕНЬ своє завдання, бажано, щоб вона мала відношення до спеціальності, хоча вона може стосуватися і повсякденного життя (вибір дорогої покупки і тому подібне)

**3. Виводи.** Зробити **змістовні** виводи по роботі в цілому.

**Контрольні питання.**

9) Що таке ієрархія?
10) Що таке принцип Кестлера (принцип дволикого Януса)?
11) Для чого потрібні ієрархії?
12) Переваги і недоліки ієрархій.

## Лабораторна робота №9

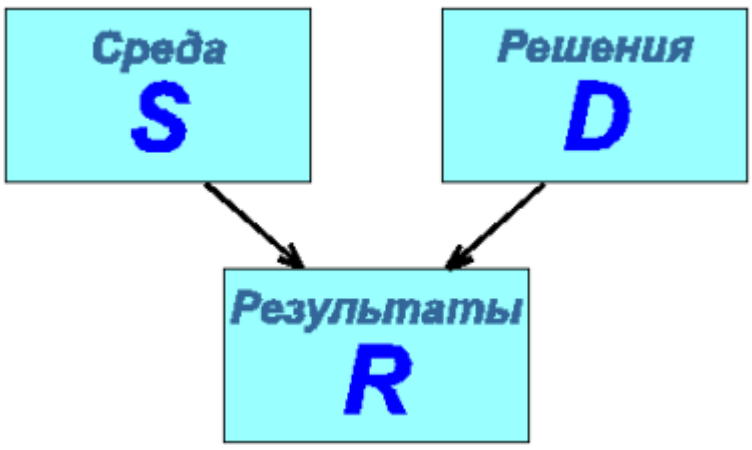
## Основні поняття теорії риски

**1.Мета роботи:** Ознайомитися з основними поняттями теорії риски.

**2. Загальні положення.**

Теорія риска є розділом системного аналізу, присвяченим ухваленню рішень в умовах імовірнісної неопределенности. У основі її лежать поняття риска меры і цены риски, відношення індивідуума до ризику. Поняття риски має безліч відтінків і нюансів. Кількість сенсів, що вкладаються в це слово, мабуть, ненамного менше кількості людей, що його вживають. Страховики називають ризиком об'єкт страхування, а також розмір можливих страхових збитків (відшкодувань), пов'язаних з цим об'єктом. Оскільки останній випадковий, математично риск описується випадковою величиною. Абсолютно інакше йде справа у інвесторів. Деякі з них на питання "що таке ризик" відповідають: ризик - це дисперсия (іноді - стандартное отклонение) прибутковості інвестиційного інструменту. У всіх цих випадках мається на увазі кількісна характеристика ризикованої випадкового об'єкту, яку в загальній теорії називають мерой риска. Міра риски – це "кількість риски", поміщена в даному риске; функціонал на просторі рисок. Прикладами мерів риски можуть служити очікувана корисність, значення під кривій щільності вірогідності риски. У класичних завданнях теорії риски як мери риски використовувалися також дисперсия і стандартное отклонение. Окремим випадком міри риски є цена риска. Кожна міра риски задає своє поняття детерминированного эквивалента.

На наступному малюнку в спрощеній формі представлена схема ухвалення рішення в умовах риски.



Тут *S* - безліч станів середовища, *D* - безліч можливих рішень, *R* - безліч всіляких результатів. На результат робить вплив як наше рішення, так і стан середовища. Таким чином, математична модель даної ситуації є відображення *M: S x D --> R*, що зіставляє стану середовища *s* і рішенню *d* результат *r=M(s,d)*. Стан середовища є, як правило, невизначеним, і описується в рамках теорії риски якою-небудь імовірнісною моделлю: говорять, що на *S* задане вероятностное распределение. За допомогою відображення *M* воно при кожному вирішенні *d* з *D* породжує розподіл на *R*. Таким чином, кожному рішенню відповідає свій розподіл на безлічі результатів, і вибір оптимального рішення зводиться до вибору "якнайкращого" розподілу на *R*.

## 2.Приклад

Розглянемо наступний простий приклад. Пікнік можна провести на відкритому повітрі в лісі, або удома. На природі, звичайно, краще, але якщо піде дощ, то пікнік буде зіпсований. В даному прикладі середовище може знаходитися в одному з двох станів: "дощ", "сухо". Безліч рішень також складається з двох елементів: "ліс" і "будинок". Хай распределелеление на *S* задано так: вірогідність того, що піде дощ, рівна 0.3 (і, значить, вірогідність сухої прекрасної погоди рівна 0.7). Хай безліч результатів складається з чотирьох елементів ("огидно", "погано", "средненько", "відмінно"), а відображення *M* влаштоване таким чином:

- *M(дощ, ліс)=отвратительно,*
- *M(дощ, будинок)=плохо,*
- *M(сухо, ліс)=отлично,*
- *M(сухо, будинок)=средненько.*

Якщо ми виберемо рішення провести пікнік в лісі, то на безлічі результатів буде породжено розподіл, приведений в наступній таблиці:

| Значення | огидно | погано | средненько | відмінно |
|---|---|---|---|---|
| Вірогідність | 0.3 | 0 | 0 | 0.7 |

Рішення ж провести пікнік будинку породить такий розподіл:

| Значення | огидно | погано | средненько | відмінно |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Вірогідність | 0 | 0.3 | 0.7 | 0 |

Ухвалення оптимального рішення в даному випадку означає вибір якнайкращого з приведених вище розподілів. Стандартна процедура вибору полягає в приписуванні кожному з результатів числового значення, що трактувало як його "корисність", з подальшою максимізацією очікуваної (середньою) корисності. Якщо ми оцінимо корисність результатів так, як описано в наступній таблиці:

| Значення | Корисність |
|---|---|
| огидно | 0 |
| погано | 2 |
| средненько | 5 |
| відмінно | 10 |

то набудемо наступних значень для очікуваної корисності рішень:

- *u(будинок)= 0.3*2 + 0.7*5 = 4.1*
- *u(ліс)= 0.3*0 + 0.7*10 = 7.*

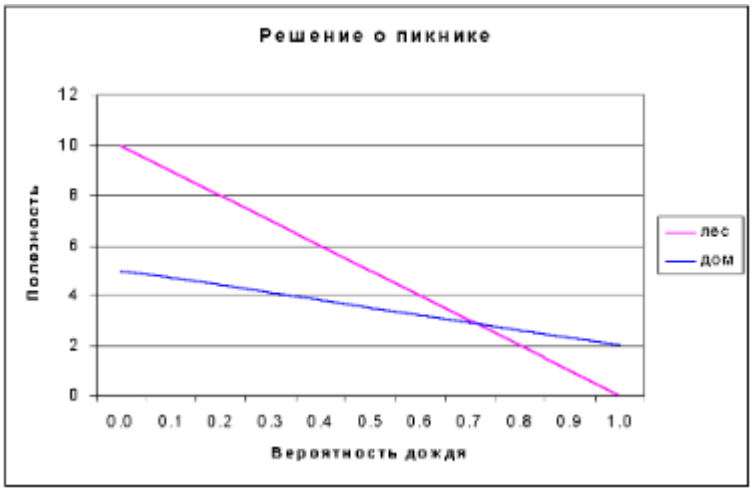
В даному випадку рішення провести пікнік в лісі має велику очікувану корисність, воно і приймається.

Цікаво прослідкувати, як зміниться вирішення при зміні інформації про можливі стани середовища. Хай вірогідність дощу рівна 0.8 (і, отже, вірогідність сухої погоди рівна 0.2). Тоді обчислення очікуваних полезностей дає

- *u(будинок)= 0.8*2 + 0.2*5 = 2.6*
- *u(ліс)= 0.8*0 + 0.2*10 = 2*

і оптимальним є вже вирішення про проведення пікніка будинку.

У описаній схемі на вибране рішення робить вплив не тільки розподіл на безлічі станів середовища, але і значення корисності, що приписуються кожному з результатів. На наступному малюнку показані графіки залежності корисності рішень від вірогідності дощивши.




3. **Завдання на лабораторну роботу.** Відкрити Exel-файл, що додається, і міняючи параметри завдання про пікнік досліджувати можливі варіанти рішення.Зробити **змістовні** виводи по роботі в цілому.

## Література


1. Campbell S. Modeling and Simulation in Scilab/Scicos. — New York: Springer, 2006. —ISBN 9780387278025
2. Campbell Stephen L., Chancelier Jean-Philippe, Nikoukhah Ramine. Modeling and Simulation in Scilab/Scicos with ScicosLab 4.4
3. Велічко о.М., Коломієць Л.В. Гордієнко Т.Б. Основі метрології:теорія та практика.Підручник. Одеса:вмв, 2009.-391 с.
4. Основі метрології та вимірювальної техніки : Підручник: У 2т./М. Дорожовець, В. Мотало, Би. Стадник, В. Василюк, Р.Борек, А. Ковальчик; за ред. Б.Стадника.-львів: Відавніцтво національного університету «Львівська політехніка», 2005.- Т.1. Основі метрології.-532с.
5. Основі метрології та вимірювальної техніки : Підручник: У 2т./М. Дорожовець, В. Мотало, Би. Стадник, В. Василюк, Р.Борек, А. Ковальчик; за ред. Б.Стадника.-львів: Відавніцтво національного університету «Львівська політехніка», 2005.- Т.1. Вімірювальна техніка.-656с.
6. Коломієць Л.В., Воробієнко П.П., Козаченко м.Т., Налісній М.Б., Козаченко л.О., Грабовський о.В. Метрологія у галузі звІязку.Книга 1.Загальні електрорадіовимірювання.Посібник. – Одеса:вмв,2009.- 480 с.
7. Долінська Л.В., Ковальчук в.В. Вступ в теорію систем та теорію управління. – Одеса.:ВД «В.В.Бакаєв», 2010.-207 с.
8. Іванов в.С. Основи математичної статистики. – М.: ФІС, 1990. – 176 с.
9. Черепанов в.Ф. Експертні оцінки в педагогічних дослідженнях. – М.: Наука, 1988. – С. 11-123.
10. Цапенко м.П. Вимірювальні інформаційні системи: Навчань. Допомога для вузів.– М.: Енергоатоміздат, 1985. – 357 с.
11. Метрологічне забезпечення вимірювальних інформаційних систем (теорія, методологія, організація) / Е.Т. Удовіченко, А.А. Брагин, А.Л. Семенюк і ін. – М.: Вид-во стандартів, 1991. – 192 с.
12. Керівництво по виразу невизначеності вимірювання / Під ред. проф. Слаєва В.А.– Спб.: “Літас+”, 1999.- 126 с.
13. Селіванов м.Н., Фрідман а.Е., Кудряшова ж.Ф. Якість вимірювань: Метрологічна справ.книга. – Л.: Леніздат, 1987.– 295 с.
14. Новіцкий п.В. Основи інформаційної теорії вимірювальних пристроїв. – Л.: Енергія, 1968. – 248 с.
15. Килін с.Я. Квантова інформація. – Успіхи фізичних наук.-1999.-т.169,№5.-С.507-525
16. Харт Х.. Введення у вимірювальну техніку: Пер. з йому. – М.: Мір,1999. - 391 с.
17. Лєнков С.В., Перегудов Д.А, Хорошко в.А. Методи і засоби захисту інформації. У 2-х томах.-К.:Арий,2008.- Т.1. 464 с.

18. http://fogmaker.net/tekno/contents_cleverhome_0.html
19. http://www.joshushund.com/fotos/wien/wien_millennium-tower.jpg
20. http://en.wikipedia.org/wiki/Smart_Home#Standards_and_bridges
21. http://en.wikipedia.org/wiki/Home_automation_for_the_elderly_and_disabled
22. http://www.smartsensorsystems.com/index.htm
23. http://home.howstuffworks.com/smart-window3.htm
24. http://www.sensedu.com/
25. http://www.citytech.com