Ніжинський державний університет
імені Миколи Гоголя

В. С. Фетісов

МАТЕМАТИЧНА СИСТЕМА SCILAB

Навчально-методичний посібник

2-ге видання,
перероблене і доповнене

Ніжин – 2022

УДК 681.3.068 519.67
Ф45

Рекомендовано Вченою радою
Ніжинського державного університету імені Миколи Гоголя
(НДУ ім. М. Гоголя)
Протокол № 4 від 27.10. 2022 р.

Рецензенти:

Віра М. Б. **–** доцент кафедри інформаційних технологій, фізико-математичних та економічних наук навчально-наукового інституту природничо-математичних, медико-біологічних наук та інформаційних технологій Ніжинського державного університету імені Миколи Гоголя, кандидат фізико-математичних наук;

Лисенко І. М. **–** доцент кафедри інформаційних технологій, фізико-математичних та економічних наук навчально-наукового інституту природничо-математичних, медико-біологічних наук та інформаційних технологій Ніжинського державного університету імені Миколи Гоголя, кандидат фізико-математичних наук

Фетісов В. С.

Ф45 Математична система Scilab: навч.-метод. посібн. 2-ге вид., перероб. і доп. Ніжин: НДУ ім. М. Гоголя, 2022. 82 с.

У посібнику розглядаються основи практичної роботи з математичною комп'ютерною системою для виконання чисельних розрахунків Scilab.

Розглянути основні прийоми роботи з програмою: виконання обчислень, роботи зі змінними, використання функцій, побудова двовимірних і тривимірних графіків. Наводяться найбільш вживані стандартні функції системи, розглядаються операції над даними. Окремими розділами подані розв'язування завдань лінійної алгебри та операції математичного аналізу.

Посібник містить також лабораторні завдання для самостійного виконання.

УДК 681.3.068 519.67

Загальні відомості

Scilab – це математична система для виконання чисельних розрахунків, яка за принципом роботи схожа із відомою математичною системою *MATLAB*. Засоби пакета дозволяють також виконувати деякі символьні перетворення, наприклад виконувати операції з поліномами.

Принципової відмінністю *Scilab* (так саме, до речі, як і *MATLAB*) від інших математичних систем є те, що основним елементом даних у системі є масив, тобто система орієнтована на роботу з даними, поданими у табличному вигляді. Система виконує операції над масивами навіть у режимі прямих обчислень. Нею можна скористатися для виконання таких матричних операцій, як інвертування, транспонування, обчислення детермінанту і т. ін.

Таблична форма подання даних в *Scilab* дозволяє вирішувати різні задачі, пов'язані з інженерно-технічними обчисленнями (особливо там, де використовуються матриці і вектори) з дуже високою швидкістю. У свою чергу досить часто й у математиці доводиться працювати з даними, які подаються у вигляді таблиць. Наприклад, за допомогою табличного подання даних зручно розв'язувати системи лінійних рівнянь.

У середовище *Scilab* інтегрована також програма редагування блочних діаграм і симуляції Xcos, яка є аналогом програми Symulink з *MATLAB*. Незважаючи на те, що *Scilab* є клоном *MATLAB*, зрозуміло, що обидві системи мають різні програмні коди, а *Scilab* "вміє" конвертувати у свій формат документи з *MATLAB*.

Математична система *Scilab* дає змогу вирішувати низку завдань:

1. Розв'язування завдань лінійної алгебри.
2. Розв'язування нелінійних рівнянь і систем.
3. Розв'язування завдань оптимізації.

4. Диференціювання й інтегрування.

5. Розв'язування звичайних диференціальних рівнянь і систем.

6. Оброблення експериментальних даних (інтерполяція й апроксимація, метод найменших квадратів).

Програма належить до класу freeware, тобто безкоштовних, вона є *відкритою* системою, що дає змогу кожному бажаючому одержати доступ до кодів системи і внести до неї зміни, додавши нові функції, типи даних або просто настроїти систему "під себе". Підтримку системи забезпечує консорціум *Scilab*, до складу якого входять більше 20 учасників, у тому числі французькі компанії INRIA та ENPC, а також ESI Group та Scilab Enterprises.

У роботі розглядається українська локалізація версії 6.0.2. Ця версія містить у своєму складі додатки ATOMS (AutomaTic mOdules Management for Scilab) для керування зовнішніми модулями та Xcos для моделювання й емуляції гібридних динамічних систем.

Інтерфейс системи

Вікно програми структуроване у чотирьох вікнах-областях, які називають також *компонентами*:

1. *Перегляд файлів*. Область для вибору файлів на зразок провідника.

2. *Вікно команд*. Основна область, призначена для введення команд та одержання результатів. Ця область називається "консоль".

3. *Перегляд змінних*. Відображення усіх змінних, що використовуються у поточній сесії.

4. *Журнал команд*. Список команд, що були введені у поточній сесії.

Будь-яку область можна вилучити, натиснувши кнопку закриття у заголовку вікна-області. Але в будь-який момент початкове (повне) розташування вікон (компонування) можна відновити. Для цього потрібно:

1. Звернутися до настройок програми.

2. Відкрити групу “Загальні”, з якої вибрати пункт “Компонування”.

3. Натиснути кнопку **«Відновити початкове компонування»**.

Крім цього, вікно програми містить звичайні елементи Windows: заголовок, рядок меню, панель інструментів і рядок стану.

Рядок стану зокрема відображає номер рядка документа та позицію, де знаходиться курсор.

Вікно команд (консоль)

Основою інтерфейсу системи є *вікно команд*, яке називається *консоль*. Саме у ньому вводяться дані та виводяться результати. Інтерфейс вікна має текстовий вигляд. До тексту за замовчуванням застосоване певне форматування. Але його можна змінити. Для цього слід виконати команду **Змінні ▸ Налаштування** або натиснути на панелі стандартних інструментів кнопку налаштування  . Після цього з’явиться вікно вибору параметрів налаштування, в якому на вкладці “Шрифти” можна змінити тип, розмір і накреслення шрифту.

Для очищення вікна команд слід виконати команду ***clc***, натиснути функціональну клавішу <**F2**> або виконати команду **Змінні ▸ Спорожніти консоль**.

Вікно перегляду змінних

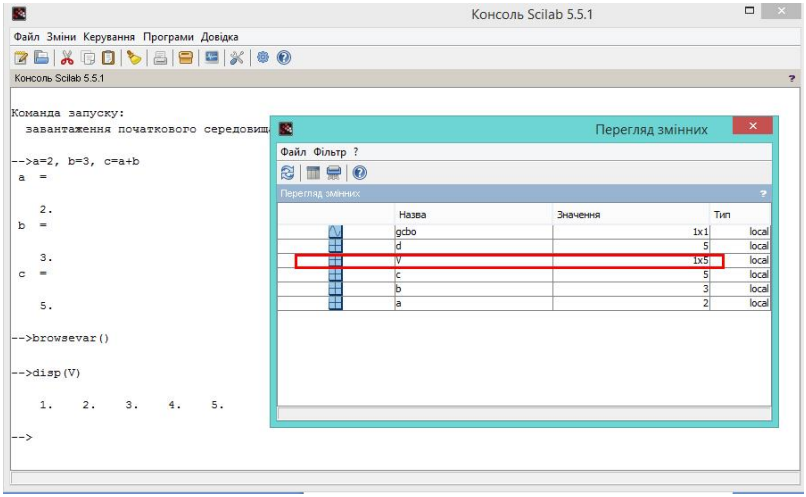
Усі змінні, що використовуються в поточній сесії, відображаються в окремому вікні, яке за замовчуванням є одним з елементів інтерфейсу системи. У вікні відображається перелік змінних і їх атрибути: розмірність змінної або її значення, тип даних і область видимості. Вікно перегляду змінних за замовчуванням відображає тільки змінні, визначені користувачем, і не показує системні змінні

(наприклад, змінна *%e*, у якій зберігається наближене значення числа *e*). За необхідності відображення у вікні перегляду системних змінних слід клацнути в області вікна перегляду змінних, що призведе до появи меню цього вікна. У ньому слід вибрати команду **Фільтр** і натиснути на пункті **Приховати змінні Scilab** і тим самим зняти прапорець з цього пункту.

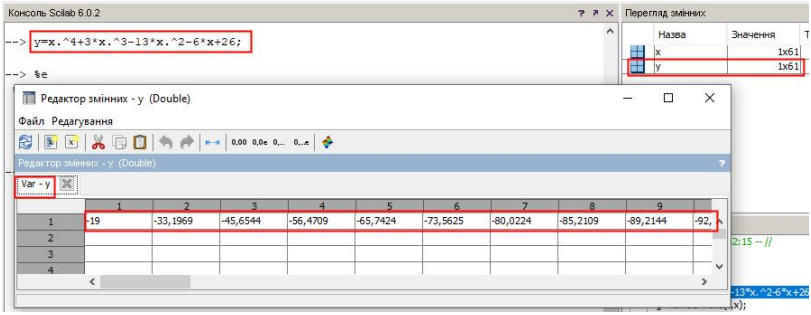
Так саме і відображення тих чи інших груп системних змінних регулюється командою **Фільтр**. Список змінних у вікні оновлюється автоматично, але оновлення можна ініціювати і вручну натиснувши у вікні перегляду змінних кнопку  **«Освіжити змінну»**.

 У вікні перегляду змінних не відображуються значення тих масивів, які містять достатньо багато елементів. Одержати їх можна функцією ***disp***, яка має такий синтаксис:

disp (ім'я змінної 1, ім'я змінної 2, …)



Переглянути значення будь-якої змінної можна ще простіше, для чого потрібно у вікні перегляду змінних двічі натиснути на рядку з її назвою. Це спричиніть появу вікна редактора змінних в якому можна не тільки переглянути значення змінної, але й змінити їх не використовуючи для цього відповідні команди.



Якщо область перегляду змінних була вилучена, то для відображення потрібно виконати команду **Програми ▸ Перегляд змінних** або ввести у рядку введення функцію ***browsevar***() і натиснути клавішу **<Enter>**.

Журнал команд

У журналі команд відображаються усі команди, що були введені останнім часом під час роботи з системою, у тому числі і команди з попередніх сесій. Вікно за замовчуванням є одним з компонентів (елементів інтерфейсу) системи.

Будь-яку команду з журналу можна виконати у вікні консолі, для чого потрібно просто двічі клацнути на рядку з її назвою.

Якщо ця область була вилучена, то для її відображення слід виконати команду **Програми ▸ Журнал команд**, після чого в окремому вікні з’явиться вікно з переліком команд.

Сесія

Сеанс роботи із системою називають *сесією* (*session*). ***Сесія*** – це поточний документ, що містить роботу користувача, тобто рядки введення даних, команд та функцій, виведення результатів, повідомлення про помилки тощо.  Визначення змінних і функцій, які розташовані в робочій області пам'яті, (але *не саму сесію*) можна записати на диск за командою **Файл ▸ Зберегти середовище…** Сам файл зберігається з розширенням SOD. Слід зазначити, що можливості збереження всього *тексту сесії* команда **Зберегти середовище** не дає. Це зроблено навмисно: сесія є результатом проб і помилок, її текст разом із правильними визначеннями може містити повідомлення про помилки, непотрібні виведення і т. ін. Необхідності зберігати все це, як правило, немає. Проте це не значить, що відсутня можливість записати раціональне зерно, створене під час роботи над розв'язком задачі. Слід просто скористатися редактором і відлагоджувачем, які дозволяють (після налагодження програми) одержати документ у коректній формі без синтаксичних і інших помилок. Завантаження з диска даних робочої області відбувається за командою **Файл ▸ Завантажити середовище…**. Якщо все ж таки є потреба повністю зберегти сесію, то це можна зробити за допомогою оператора ***diary***, що призначений для ведення *щоденника сесії*. Оператор має два формати, за допомогою яких починається і припиняється процес запису сесії:

- diary (filename) – запис на диск всі введені команди й отримані результати до файлу з ім'ям filename; ім'я файлу подається в апострофах і може включати шлях;
- diary(0) – припиняє запис у файл.

 Після закінчення роботи файл автоматично не зберігається.

За замовчуванням система після її завантаження шукає документи в системній папці документів користувача. Для зміни папки у поточній сесії слід виконати команду **Файл ▸ Змінити поточний каталог…**, вибрати потрібну папку і натиснути «**OK**». Якщо ж потрібно змінити папку за замовчуванням, то для цього слід звернутися до налаштувань програми, в групі "Загальні" встановити перемикач в положення "Використовувати типовий каталог" і задати шлях до нього. Відображення поточної папки для збереження документів здійснюється командою **Файл ▸ Показати поточний каталог…**.

Документ системи

Будь-які обчислення (часом досить складні) у системі можна виконати в режимі *прямих обчислень,* тобто без підготовки програми. Це перетворює *Scilab* у потужний калькулятор, здатний здійснювати не тільки звичайні для калькуляторів обчислення (наприклад, виконувати арифметичні операції й обчислювати елементарні функції), але й здійснювати операції з векторами й матрицями, комплексними числами, рядами й поліномами, розв'язувати системи лінійних рівнянь і т. ін.

Документ являє собою послідовність рядків введення у вигляді операторів, команд і функцій і результатів виведення. Нова інформація вводиться у рядку, на початку якого знаходиться символи "-->", які є ознакою запрошення введення інформації.

В одному сеансі роботи з системою можна працювати з тільки з одним документом (інші закриваються).

Для використання системи у режимі прямих обчислень потрібно знати, як саме у цьому разі вводиться інформація.

Правила введення інформації в документ

1. Призначення функціональних клавіш при виконанні

дій у рядку введення відповідає загальноприйнятим у текстових редакторах. Так, переміщення вліво або вправо досягається натисненням на клавіатурі стрілки вліво або вправо; на початок або кінець рядка – клавішами **<Home>** та **<End>**; **<Esc>** призначена для очищення рядка, **<Ins>** для ввімкнення або вимкнення режиму вставки і т. ін.

2. *Введення нової інформації* здійснюється відразу після символів "-->".

3. Будь-яке введення завершується натисканням **<Enter>**.

4. Для *обчислення* математичного виразу і *відображення* у документі результату за закінченням уведення виразу слід натиснути **<Enter>**.

```
2+3 <Enter>
 ans =
```

Результат обчислень виводиться в рядках виведення без символу "-->". При цьому результат обчислення автоматично присвоюється системній змінній *ans* (від англ. answer – відповідь):

5. Якщо *не потрібно виводити результат*, то введення інформації закінчується символом "**;**".

В одному рядку можна ввести кілька операторів та (або) команд, відокремлюючи їх символом "**,**":

```
-->2+3, 3/5
 ans =
      5.
 ans =
      0.6
```

У цьому прикладі система послідовно виконує дві операції: додавання та ділення. Оскільки друга операція не містила присвоювання, результат збережено у змінній *ans*.

6. Якщо виведення результату обчислення не потрібне, то його можна відключити, завершуючи оператор символом ";":

2+3; 3/5

7. Якщо вираз, що вводиться, є довгим і не поміщається у рядку, то наприкінці незавершеного рядка слід ввести дві крапки “..”.

8. Нову інформацію можна додавати використовуючи *стек* раніше введеної інформації. Доступ до стеку здійснюється з рядку введення клавішами управління курсором “нагору” й “униз”, які дозволяють гортати раніше введені команди знизу-вверх і зверху-вниз. Надалі їх можна повторно використовувати або створювати на їх основі інші.

9. Ще зручніше додавати інформацію використовуючи журнал команд. Для цього достатньо просто здійснити подвійне натискання на рядку з потрібним введенням з журналу.

10. *Редагувати* раніше введену інформацію не можна.

Сценарій

Сценарій – це послідовність команд, операторів і функцій, що підключаються до документа як одне ціле (на зразок підпрограми у мовах програмування) і виконуються. Для створення сценаріїв система має спеціальний засіб – *редактор сценаріїв*. Доступ до редактора здійснюється за командою **Програми ▸ SciNotes** або натисканням на панелі інструментів кнопки **«Запустити SciNotes»**  . Після цього з’являється вікно редактора, в якому і створюється сценарій. Призначення редактора, так саме, як і взагалі для редакторів для створення програм алгоритмічними мовами, потрійне. По-перше, він дозволяє виконувати усі типові дії редагування послідовності введення, тобто сценарію. По-друге, він здійснює синтаксичну перевірку команд та функцій. І, нарешті, він дозволяє завантажити сценарій на виконання у вікно консолі.

 Редактор сценаріїв є найбільш ефективним

засобом створення нових математичних алгоритмів. Під час формування сценарію одночасно можна перевіряти його працездатність, тобто завантажувати його на виконання у середовищі *Scilab*, скориставшись командою редактора з групи **Виконати**. Наприклад, за командою **Виконати ▸ файл з виведенням** здійснюється завантаження сценарію до вікна консолі, де він виконується, і користувач одержує результати його виконання.

Файл сценарію призначений для виконання його надалі у середовищі *Scilab*. Подальша робота з ним може відбуватися двома шляхами.

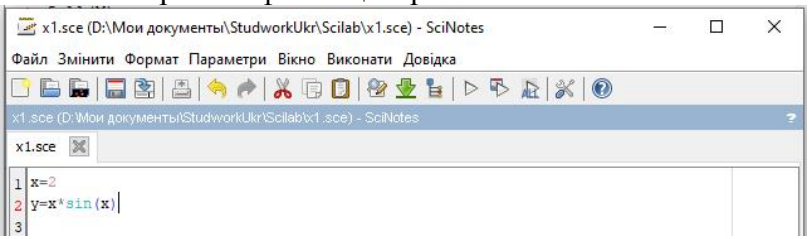
1. За командою **Файл ▸ Виконати** здійснюється розрахунки за сценарієм, при цьому виводяться тільки результати.

2. За командою **Файл ▸ Відкрити файл** відбувається завантаження сценарію в редактор сценарію *SciNotes*, звідки він надалі завантажується на виконання. За таким варіантом у документі відображається як результати, так і усе введення.

До редактора сценаріїв можна завантажити кілька файлів, кожний з яких буде відображатися в окремій вкладці.

Приклад.

1. Створюємо файл сценарію:



2. Запам’ятовуємо його командою **Файл ▸ Зберегти** в потрібному місті.

3. У середовищі *Scilab* виконуємо **Файл ▸ Відкрити** і у вікні відкриття файлів виберемо збережений файл.

Результат виконання:

```
x  =

   2.
y  =

   1.8185949

Execution done.
```

Довідкова система

Звернення до довідкової системи здійснюється командою **Довідка ▸ Довідка Scilab** або натисканням кнопки  на панелі стандартних інструментів. Отримати довідку можна також шляхом використання оператора ***help***. Наприклад, для отримання довідки відносно оператора ***exec*** слід ввести

```
help exec
```

Система має бібліотеку демонстраційних прикладів (необхідність їх встановлення на ПК користувача визначається під час інсталяції системи). Звернення до неї відбувається за командою **Довідка ▸ Демонстрації Scilab**. Демонстраційні приклади відкриваються в новому вікні, яке потрібно закривати по закінченню перегляду прикладу.

Вхідна мова системи

Алфавіт

Алфавіт визначає сукупність символів і слів, що використовуються для запису команд.

Алфавіт системи містить:

- малі і великі латинські літери;
- арабські цифри від 0 до 9;
- системні змінні;
- оператори;
- імена вбудованих функцій;

- спеціальні знаки.

Правила синтаксису мови системи

1. Усі імена команд і функцій записуються літерами латинського алфавіту.
2. Аргументи операторів і функцій записуються у круглих дужках.
3. Велика і маленька літери розрізняються.
4. У числах ціла частина від дробової відокремлюється крапкою.
5. Знаки арифметичних операцій у виразах потрібно обов'язково вказувати.
6. Порядок дій у математичних виразах відповідає загальноприйнятому порядку дій у математиці.

Система здійснює синтаксичний контроль введення і за наявності помилки видає про це повідомлення.

Текстові коментарі

Основним режимом роботи системи є, звичайно, робота з математичними об'єктами. Але система дозволяє вводити пояснення до документа (тобто коментарі), які роблять документ більш зрозумілим.

Для введення коментарів з будь-якої позиції рядка вводяться два символи "//", після чого вводиться безпосередньо сам коментар.

Об'єкти системи

Обчислення здійснюються за допомогою математичних виразів, які є головним об'єктом будь-якої математичної системи. Вираз визначає те, що повинно бути обчислено в чисельному вигляді. Математичні вирази складаються з чисел, констант, змінних, операторів, функцій і спеціальних знаків.

```
-->6+7
 ans  =

    13.

-->2.345*log(10)-1
 ans  =

    4.399562
```

Числа

За найпростішим варіантом у режимі прямих обчислень відбувається робота з числами. Число – найпростіший об'єкт системи, що представляє кількісні дані. Вони можуть бути дійсними: цілими, дробовими, з фіксованою точкою. Ціла частина відокремлюється від дробової частин крапкою. Знак "+" для додатних чисел не використовується. Проміжки між символами в числах не допускаються. Наприклад: 0, -3, 2.301.

Константи

Константа – це попередньо визначене числове або символьне значення, представлене унікальним ім'ям. Числа (наприклад, 1, -2 і 1.23) є числовими константами без імені.

Оператори

Оператори – це елементи мови, за допомогою яких створюються математичні вирази. До них, наприклад, належать знаки арифметичних операцій, обчислення сум, добутків, похідних, інтегралів і т. ін. Оператори використовуються разом з операндами. Наприклад, у виразі "5-1" знак "-" є оператором віднімання, а числа "5" і "1" – операндами. Після визначення операндів оператори стають блоками, *що виконуються*.

Арифметичні оператори

Оператори – це елементи мови, за допомогою яких створюються математичні вирази. Вони призначені для виконання арифметичних дій над числовими величинами і конструювання математичних виразів. Оператори використовуються разом з операндами. Наприклад, у виразі "5-1" знак "-" є оператором віднімання, а числа "5" і "1" – операндами.

 За замовчуванням результат обчислення містить вісім цифр:

```
-->3+7.123456789
 ans =
    10.123457
```

Але цю кількість можна змінити. Для цього використовується функція ***printf***, яка є емулятором однойменної функції мови *C*. Синтаксис функції:

printf(формат, ім'я змінної 1, ім'я змінної 2, …)

Оскільки функція емулює функцію мови *C*, аргумент "формат" описується у відповідності з правилами цієї мови. Наприклад:

```
-->printf("%1.10f",ans);
10.1234567890
```

Змінні

Зазвичай у документі доцільно користуватися не конкретними числовими значеннями, а змінними. ***Змінна*** – це пойменований об'єкт, значення якого в документі може змінюватися.  Імена констант, змінних і інших об'єктів називають ідентифікаторами.

У свою чергу і значення обчислення можна запам'ятати у змінній.

Правила надання імен ідентифікаторам:

1. вони можуть складатися з будь-яких латинських і грецьких літер, цифр;
2. великі і малі літери розрізняються;
3. починаються тільки з літери;
4. не можуть містити проміжків;
5. не можуть збігатися з іменами вбудованих чи визначених користувачем функцій;
6. не можуть містити символів кирилиці.

```
-->c=2; b=3
 b  =

    3.

-->c+b
 ans  =

    5.
```

 Для *надання* змінній значення застосовується символ “=”.

Змінні можуть бути локальними і глобальними. Локальні змінні використовуються і зберігаються тільки в поточній сесії. Для створення глобальної змінної її потрібно описати за допомогою оператора ***global***.  За однім оператором ***global*** можна описати кілька змінних, при цьому вони повинні відокремлюватися проміжком.

```
-->global x z
```

Одержання списку змінних, що використовуються в поточній сесії, здійснюється функцією ***who***. Застосування цієї функції з аргументом “global” виводить список глобальних змінних, а з аргументом “local” – список змінних, що використовуються в поточній сесії, а також системних змінних:

```
-->who("local")
 ans =
!a                              !
!                               !
!editvar                        !
…
!browsevar                      !
!                               !
!printf                         !
!                               !
!scicos_pal                     !
!                               !
!%scicos_menu                   !
…
```

Змінні розташовуються в робочій області, їх можна знищувати; а можна взагалі очистити всю цю область. Для цього використовується оператор ***clear***. При його використанні без аргументів знищуються всі змінні, а для знищення окремої або кількох змінної в якості операндів використовуються імена змінних, які відокремлюються одна від одної проміжком:

clear a [b c …]

Системні (зарезервовані) змінні

Системні змінні – це невелика група особливих об'єктів, які не можна віднести ні до класу констант, ні до класу змінних. Прикладом такої змінної є *ans*, яка зберігає результат останньої операції. У системі також є невелика група скалярних системних змінних, відмінністю яких від інших є наявність перед їх іменем символу "%".

Перелік основних системних змінних містить наступна таблиця.

Таблиця 1

Системні змінні

Уведення	Призначення
%i	Уявна одиниця ($\sqrt{-1}$).
%pi	Число π.
%e	Число e (основа натурального логарифма, 2,7182818).
%eps	Похибка операцій над числами із плаваючою крапкою ($2{,}22^{-16}$).
%inf	Системна нескінченність.
%t	Логічне “істинне”.
%f	Логічне “хибне”.

Наприклад, введення системної змінної %pi за замовчуванням еквівалентно введенню числа “3.1415927”. Системні змінні захищені і не можуть бути вилучені. Як правило їх не можна і *перевизначати.* Наприклад, спроба надати системної змінної %pi інше значення, викличе помилку:

```
-->%pi=3.14
!--error 13
redefining permanent variable.
```

Функції

Scilab має убудовані функції для розрахунку всіх математичних дій, а також багато спеціальних функцій.

Правила запису функцій:

1. Ім’я функції чутливе до регістру, їх назви записуються маленькими літерами;
2. Аргументи функції беруться до круглих дужок.
3. Функції можна вкладати одну до одної.

У наступній таблиці наведено поширені математичні функції.

Таблиця 2

Елементарні математичні функції

Математичні функції	
abs(x)	$\|x\|$
sqrt *(x)*	$\sqrt{x}$
Тригонометричні функції	
sin(x)	Синус
cos(x)	Косинус
tan(x)	Тангенс
ctg(x)	Котангенс
asin(x)	Арксинус
acos(x)	арккосинус
Експоненціальні	
exp(x)	експонента числа *x*
log(x)	натуральний логарифм числа *x*

Функції користувача

Користувач може самостійно створювати власні функції як для конкретного сеансу роботи, так і для постійного використання.

Функція користувача будується за певними правилами:

1. Вона починається з ключового слова “function”, після якого через проміжок записується аналітичний вираз функції. Функція може мати вхідні параметри, наприклад, function y=f(x).
2. Математичний вираз функції, наприклад, x*cos(x).
3. Закінчується опис функції користувача ключовим словом “endfunction”.

Складові функції користувача відокремлюються комою.

Отже, для наведеного прикладу функція користувача буде мати вигляд:

```
-->function y=f(x), y=x*cos(x), endfunction
```

Але такий варіант підходить у тому випадку, якщо "тіло" функції містить тільки один оператор. Якщо ж є потреба застосувати у функції кілька операторів, то вона формується за "лінійним" принципом, коли кожний рядок містить один оператор. Рядок введення можна закінчувати символом ";", але можна і взагалі нічого не ставити:

```
function difur=syst(t,y)
difur=zeros(2,1)
difur(1)=cos(y(1)*y(2))
difur(2)=sin(y(1)+y(2)*t)
endfunction
```

При створенні користувацьких функцій у редакторі сценаріїв для їх відокремлення від інших введень передбачено автоматичне виділення їх іншим кольором.

Виклик функції:

```
--> y=[1 2];
--> difur=syst(1,y)
 difur  =
  -0.4161468
   0.14112
```

Типи даних

У системі визначається шість типів числових цілих даних: 32, 16 і 8-бітові дані зі знаком і без знака.

Визначити тип змінної можна за допомогою функції ***type***(ім'я змінної), яка повертає ціле число, що визначає належність змінної до певного типу. Наприклад, число "1" – означає дійсну або комплексну матрицю, 2 – поліноміальну матрицю, 4 – логічний тип і т. ін.

```
--> M=[0.25 1.26; 7 8]
```

```
 M  =
   0.25   1.26
   7.     8.
--> type(M)
 ans  =
   1.
--> D = [%t %f; %T %F]
 D  =
  T F
  T F
 --> type(D)
 ans  =
   4.
```

За допомогою спеціальних команд можна перетворювати дані одного типу до іншого.

Наприклад, перетворення даних дійсного типу до цілого здійснюється за допомогою функції ***iconvert***(матриця дійсних елементів, тип перетворення).

Тип перетворення – код типу даних. Наприклад, якщо тип = 1, то це функція повертає цілі числа в діапазоні [-128, 127].

```
--> m=[1.23, 1.25; 2.45, 3.45]
 m  =
   1.23   1.25
   2.45   3.45
--> y=iconvert(m,1)
 y  =
  1  1
  2  3
```

Комплексні числа

Система працює і з комплексними числами $a + bi$. Вони записуються в алгебраїчній формі з уявною

одиницею, яка позначається %i у вигляді a+b*%i, де a і b – відповідно дійсна й уявна частини числа, наприклад:
(3+5*%i) + 2*%i.

Логічний тип

Система дозволяє використовувати змінні логічного типу. При цьому істина подається літералами %t або %T, а хибне - %f або %F.

З такими змінним можна проводити звичайні операції, що застосовуються над даними такого типу (кон'юнкція, диз'юнкція і т. ін.:

```
-> a=%t
 a  =
  T
--> b=%f
 b  =
  F

--> a&b
 ans  =
  F

--> a|b
 ans  =
  T
```

Створення масивів

Основним типом даних системи є масиві.

Створення векторів

Для створення вектора-рядка і надання йому значень слід ввести його ім'я, знак присвоювання і в квадратних дужках через проміжок або кому елементи вектора.

Наприклад, якщо слід створити вектор-рядок *V* зі значеннями елементів 1, 2, 3 і 4, то використовується запис V=[l 2 3 4] або V=[1,2,3,4], які за змістом є ідентичними:

```
-->V=[1 2 3 4]
V =
    1. 2. 3. 4.
```

При створенні вектора-стовпчика його елементи відокремлюються через “;”. Створити вектор-стовпчик можна також шляхом транспонування вектора-рядка. Символом транспонування є апостроф:

```
-->V2=V'
V2 =
    1.
    2.
    3.
    4.
```

Створити вектор можна і шляхом надання значення його довільному елементу.  Усі інші елементи за таким варіантом дорівнюватиме “0”.

Ще один варіант створення вектора пов’язаний з використанням оператора ”:”. У цьому разі вектор створюється у вигляді числової послідовності, елементи якої змінюються з певним кроком від якогось початкового значення до якогось кінцевого. Синтаксис оператора має вигляд:

ім’я масиву=початкове значення: крок зміни: кінцеве значення

Наприклад:

```
--> V3=1:2:5
V3 =
    1. 3. 5.
```

Якщо опустити аргумент “крок зміни”, він дорівнюватиме “1”, тобто вектор сформується як послідовність цілих чисел.

Створити вектор можна також з матриці, про які мови піде нижче. Наведений приклад демонструє перетворення матриці у вектор-стовпчик:

```
--> M=[1 3; 6 8]
 M  =
   1.   3.
   6.   8.

--> V=M(:)
 V  =
   1.
   6.
   3.
   8.
```

Для звернення до *n*-го елемента вектору V використовується запис у вигляді ім'я вектора(номер елемента), наприклад:

```
k=V(1)+25
```

Створення матриць

Створити матрицю можна так само як вектор, тобто ввести її ім'я, знак присвоювання, і в квадратних дужках – перелік її елементів.  Для відокремлення елементів рядків використовується проміжок або кома, а для відокремлення одного рядка від іншого – крапка з комою. Наприклад, для створення матриці *M* розмірності 2×2 з елементами 1,2,3,4, у рядку введення слід ввести:

```
M = [1,2;3,4].
```

Створити вектор або матрицю можна також шляхом поєднання кількох векторів (дія *конкатенації*). Зрозуміло, що при створенні матриці обов'язковою є вимога однакової розмірності векторів.

Приклад.

```
-->A=[1 2];
-->B=[3 4];
-->C=[5 6];
-->V=[A B C]
V =
    1. 2. 3. 4. 5. 6.
-->M1=[A;B;C]
M1 =
    1. 2.
    3. 4.
    5. 6.
```

Для звернення до конкретного елемента матриці M використовується запис M(j,i), де M – ім'я матриці, j – номер рядка та i – номер стовпця:

```
 -->M1(3,2)
 ans =
    6.
```

Як було зазначено раніше, масив може бути створений у вигляді числової послідовності, *елементи якої змінюються з певним кроком від деякого початкового значення до деякого кінцевого* за допомогою оператора ":". Взагалі цей оператор відіграє в системі важливу роль. Наприклад, він дає змогу одержати доступ до окремих блоків матриці – рядків, стовпчиків. Наступні приклади демонструють формування з матриці M векторів V1 і V2, перший з яких є другим рядком матриці M, а другий – першим стовпчиком матриці M.

```
-->M=[1 2 3; 4 5 6]
 M =
    1. 2. 3.
    4. 5. 6.
-->V1=M(2,:)
```

```
 V1 =
      4. 5. 6.
-->V2=M(:,1)
 V2 =
      1.
      4.
```

Оператор ":" взагалі дає змогу виділити з матриці блок елементів, створивши таким чином нову матрицю. Наступний приклад демонструє створення нової матриці M1 з 1 і 2 рядків та 2 і 3 стовпчиків матриці M.

```
-->M1=M(1:2,2:3)
 M1 =
      2. 3.
      5. 6.
```

Застосування комбінації з двох квадратних дужок "[]" дає змогу вилучати з масивів окремі елементи або їх блоки. Наприклад:

```
-->M=[1 2 3; 4 5 6];
-->V1=M(2,:)
 V1 =
      4. 5. 6.
-->M(2,:)=[]              // Вилучення 2-го рядка матриці M
 M =
      1. 2. 3.
-->V1(2)=[]               // Вилучення 2-го елемента вектора V1
 V1 =
      4. 6.
```

Scilab має кілька функцій, за допомогою яких можна створити спеціальні матриці або перетворювати їх.

Таблиця 3

Функції для створення спеціальних матриць

Функція	Призначення
1	**2**
matrix (M1 [,m,n])	Перетворення матриці *M* в матрицю з іншім розміром
cat(i,M1,M2)	Поєднання матриць *M1* і *M2*. При цьому, якщо значення аргументу *i* дорівнює “1”, то об’єднування здійснюється по рядках, а якщо “2”, то – по стовпчиках. Дія функції тотожна дії оператора конкатенації M=[M1 M2].
eye(m,n)	Створення одиничної матриці (всі її елементи дорівнюють нулю, крім елементів головної діагоналі, значення яких дорівнює “1”).
ones(m,n)	Створення матриці, всі елементи якої дорівнюють “1”. Для створення вектора одному з аргументів надати значення “1”.
trial (M [,k])	Створення нижньої трикутної матриці, починаючи з k-ї діагоналі. За відсутності аргументу k або якщо k=0 матриця формується починаючи з головної діагоналі.
zeros(m,n)	Створення матриці, всі елементи якої дорівнюють “0”.
sparse([i1 j1;i2 j2;…;in jn],[n1,n2,…,nn])	Створення розрідженої матриці. Аргументами функції є індекси не нульових елементів (i1 j1) і їх значення (n1).
full(M)	Відображення розрідженої матриці

Продовження таблиці 3

1	2
rand(m,n, «тип розподілу»)	Створення матриці, всі елементи якої формуються генератором випадкових чисел.
grand(m,n,)	Створення матриці, всі елементи якої формуються генератором випадкових чисел з можливістю вибору розподілу.
diag(V)	Створення діагональної матриці, елементами головної діагоналі якої є значення вектора *V*; всі інші її елементи дорівнюють “0”.
sort(M)	Впорядкування вектору. Якщо аргумент є матрицею, то сортування здійснюється по стовпцям.

Для усіх функцій аргументи *m,n* – це розмірність матриці.

Приклади перетворення матриці в інші за допомогою функції matrix

```
-> M=[1 2; 0 3; 4 8]
 M  =
   1.  2.
   0.  3.
   4.  8.

--> matrix(M, 2, 3)
 ans  =
   1.  4.  3.
   0.  2.  8.

--> V=matrix(M, 1, 6)
```

```
 V  =
   1.   0.   4.   2.   3.   8.

--> V1=matrix(M, 6, 1)
 V1  =
   1.
   0.
   4.
   2.
   3.
   8.
```

Наведені приклади у тому числі демонструють як можна перетворити матрицю на вектор.

Створення розрідженої матриці

```
--> M=sparse([1 2; 2 3; 3 4], [3 6 9])
 M  =
(  3,  4) sparse matrix

(  1,  2)      3.
(  2,  3)      6.
(  3,  4)      9.

--> full(M)
 ans  =
   0.   3.   0.   0.
   0.   0.   6.   0.
   0.   0.   0.   9.
```

Приклади генерації випадкових чисел

Неперервний рівномірний випадковий розподіл на інтервалі [0,1) (1 да інтервалу не включається)

```
--> grand(4,4,"def")
 ans  =
  0.8147237   0.1269868   0.6323592   0.2784982
```

```
  0.135477    0.9688678   0.3081671   0.188382
  0.9057919   0.9133759   0.0975404   0.5468815
  0.8350086   0.221034    0.5472206   0.9928813
```

Неперервний рівномірний випадковий розподіл на інтервалі [low,high) (high да інтервалу не включається)

```
--> grand(4,4,"uin",0,10)
 ans  =
  2.  6.  6.  1.
  3.  5.  3.  6.
  2.  7.  1.  6.
  7.  8.  7.  0.
```

Нормальний розподіл. Параметри функції: (m, n, “nor”, Av, Std), де Av – середнє значення, Std – середнє квадратичне відхилення

```
--> grand(4,4,"nor",0,1)
 ans  =
  0.4388861  -1.0682781   1.0433      -0.540479
 -1.0288046   0.0388806  -0.4565521  -0.827319
  0.1979328   0.1788246  -1.2085547   1.5340618
  0.3833912  -0.7169367   1.5310142   2.6169952
```

Операції над матрицями

У системі для роботи з масивами виконуються різноманітні операції. Основні такі операції зведені у наступній таблиці.

Таблиця 4

Операції з матрицями

Операція		**Примітка**
1		**2**
+	Додавання	Масиви мають бути однакової розмірності.
-	Віднімання	

Продовження таблиці 4

*	Множення масивів	Масиви мають бути однакової розмірності. При цьому кількість рядків у першому масиві повинна дорівнювати кількості стовпчиків у другому, а кількість стовпчиків у першому – кількості рядків у другому. Отже, у разі виконання множення для векторів, один із них повинен бути вектором-стовпчиком, а другий – вектором-рядком.
*	Множення на скаляр	
^	Піднесення до степеня	Операція еквівалента множенню матриці саму на себе. Якщо при цьому показник степені менше “1”, то матриця множиться на матрицю, *обернену до себе*.
‘	Транспонування	
\	Ліве ділення	$(A \setminus B) \Rightarrow (A^{-1} \cdot B)$. Операція може бути використана для розв’язання рівняння $A \cdot X = B$, де X – вектор з невідомими значеннями.
/	Праве ділення	$(B / A) \Rightarrow (B \cdot A^{-1})$. Операція може бути використана для розв’язання рівняння $X \cdot A = B$, де X – вектор з невідомими значеннями.
.*	Поелементне множення матриць	Масиви мають бути однакової розмірності.

Продовження таблиці 4

.^	Поелементне піднесення до ступеня	
.\	Поелементне ділення матриць справа наліво	Масиви мають бути однакової розмірності.
./	Поелементне ділення матриць зліва направо	Масиви мають бути однакової розмірності.
.^	Поелементне піднесення до степеня	Масиви мають бути однакової розмірності.

Приклади операцій.

```
-->V=[1; 2]; V1=[3 4];
-->V*V1
 ans =
      3. 4.
      6. 8.
-->A=[1 2; 3 4]; B=[3 4; 5 6];
-->X=A\B                // Розв'язування матричного рівняння AX=B
 X =
      - 1. - 2.
      2. 3.
-->X=B/A                // Розв'язування матричного рівняння XA=B
 X =
      0. 1.
      - 1. 2.
-->X*A-B                // Перевірка
 ans =
      0. 0.
      0. 0.
```

В усіх розглянутих прикладах елементами матриць є числа, але система дає змогу створювати і працювати з матрицями, елементами яких є рядки символів. Наприклад, з такими матрицями можна додавати, транспонувати.

Визначити, які елементи більше заданого числа

```
--> a=[4,2,5,1]
 a  =
   4.   2.   5.   1.
--> k=[3]
 k  =
   3.
--> disp(a>3)
 T F T F
--> disp(a>k)
 T F T F
```

Списки

Список містить елементи, які є довільними об'єктами системи. Для створення списку використовується команда ***list***

```
--> l=list(12,["ab", "cd"])
 l  =
      l(1)
   12.
      l(2)
!ab  cd  !
```

Для роботи зі списками використовують спеціальні операції: вставка на певне місце, додавання на початок і в кінець списку, вилучення, конкатенація і т. ін.

Крім «звичайних» списків бувають також типізовані і орієнтовані на матрицю списки.

Побудова графіків

Побудова 2-D графіка

Для побудови двовимірного графіка використовується функція ***plot2d***, яка за найпростішим варіантом має синтаксис:

plot2d([x,]y)

, де x – вектор точок (координат), y є вектором значень функції від параметра x. Зрозуміло, що вектори повинні бути однакової розмірності. За відсутності першого аргументу координати осі абсцис (x) визначаються автоматично.

Функцію можна визначити різними шляхами:

1. Описати її перед застосуванням ***plot2d***.

```
x=[0:0.01:2*%pi]'; y=sin(exp(x)); plot2d(y)
```

2. Описати безпосередньо в ***plot2d***.

```
x=[0:0.01:2*%pi]'; plot2d(x, sin(exp(x)))
```

 Зверніть увагу, що x перетворюється у вектор-стовпчик.

На жаль, система спроможна побудувати не усі графіки. Наприклад, авторові так й не вдалося побудувати графік $y = x \cdot \sin(\frac{1}{x})$.

Разом із функцією ***plot2d*** для побудови двовимірних графіків можна використовувати функцію ***plot***, яка є повним аналогом функції ***plot2d*** і призначена для сумісності із системою *MATLAB*.

Додаткові аргументи функції дозволяють визначити стиль графіка, встановити межі графіка і т. ін.

plot2d ([x],y,[, додаткові аргументи для графіків])

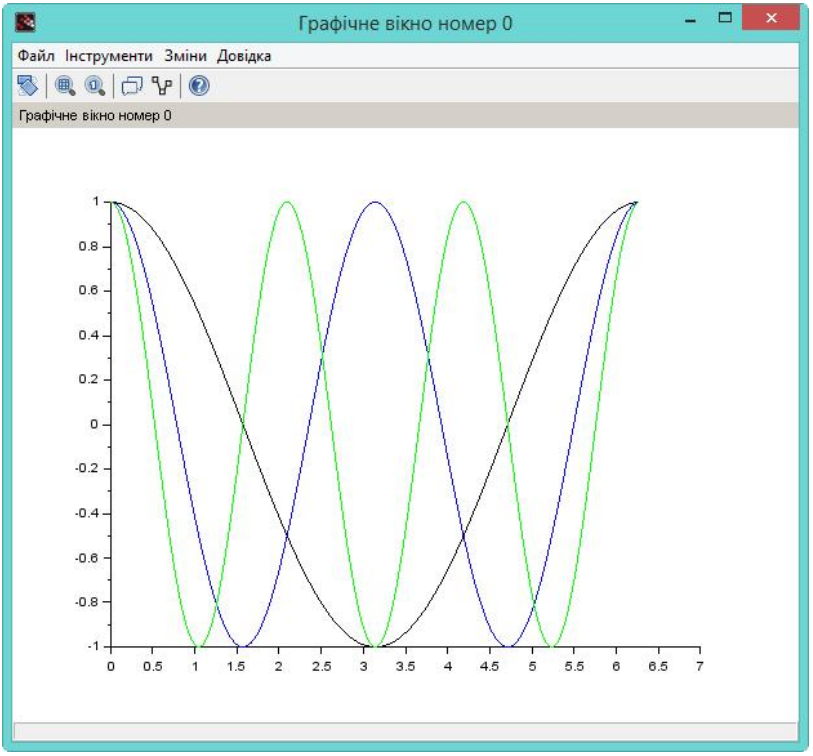
Наприклад визначення кольору лінії здійснюється так:

```
x=[0:0.1:2*%pi]';
plot2d(x, sin(x), style=color("green"))
```

З прикладу зрозуміло, що значення кольору задається англійською назвою аргументу *color*.

За допомогою функції в одному графічному вікні можна побудувати графіки кількох функцій. У цьому випадку перелік функцій подається у вигляді списку, тобто у квадратних дужках, а самі аналітичні вирази функцій відокремлюються проміжком:

```
plot2d(x,[cos(x) cos(2*x) cos(3*x)])
```



Кожний графік відображається іншим кольором.

Побудова графіків у полярній системі координат

Для побудови графіка у полярних координатах застосовується функція ***polarplot***. Синтаксис функції:
polarplot(діапазон значень кута, функція, за якої будується полярній графік[, додаткові аргументи для графіків])

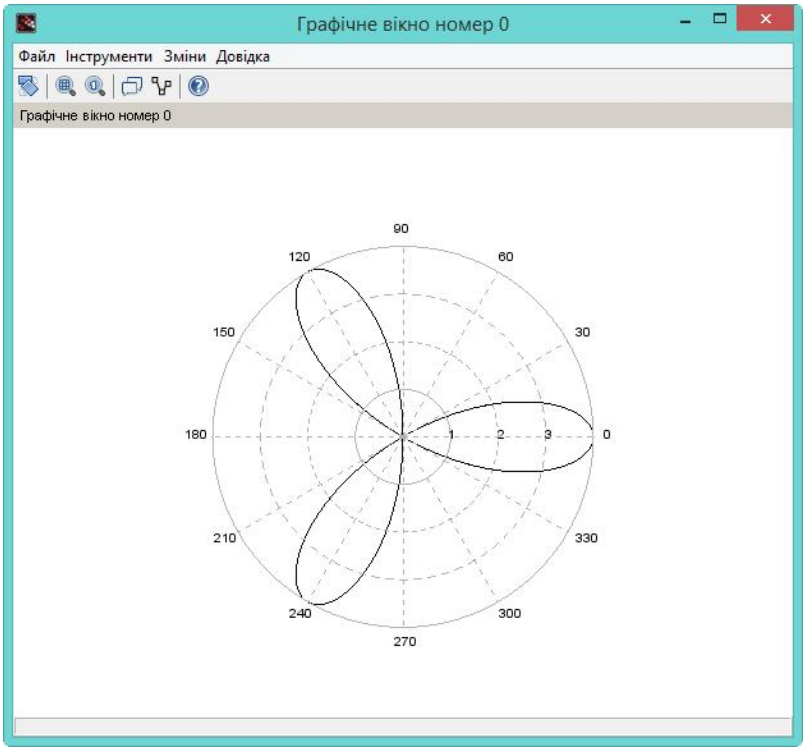
Зрозуміло, що і перший, і другий аргументи є обов'язковими.

Приклад. Побудувати у полярних координатах графік функції $4\cos(4\varphi)$, де ϕ змінюється на інтервалі [0; 2π] з кроком "0,01". Розв'язок матиме вигляд:

```
--> fi=[0:0.01:2*%pi]
--> polarplot(fi, 4*cos(3*fi))
```

Або:

```
--> fi=[0:0.01:2*%pi]
--> f=4*cos(3*fi)
--> polarplot(fi, f)
```



Так саме як і для графіків у декартовій системі координат, при побудові графіка у полярній системі для нього можна задати колір лінії:

```
-->fi=[0:0.01:2*%pi];
-->polarplot(fi,f, style=color("red"))
```

Так саме як і для графіків у декартовій системі координат, в одному графічному вікні можна побудувати кілька графіків, але синтаксис при цьому інший:

polarplot(функція_1, функція_2,…)

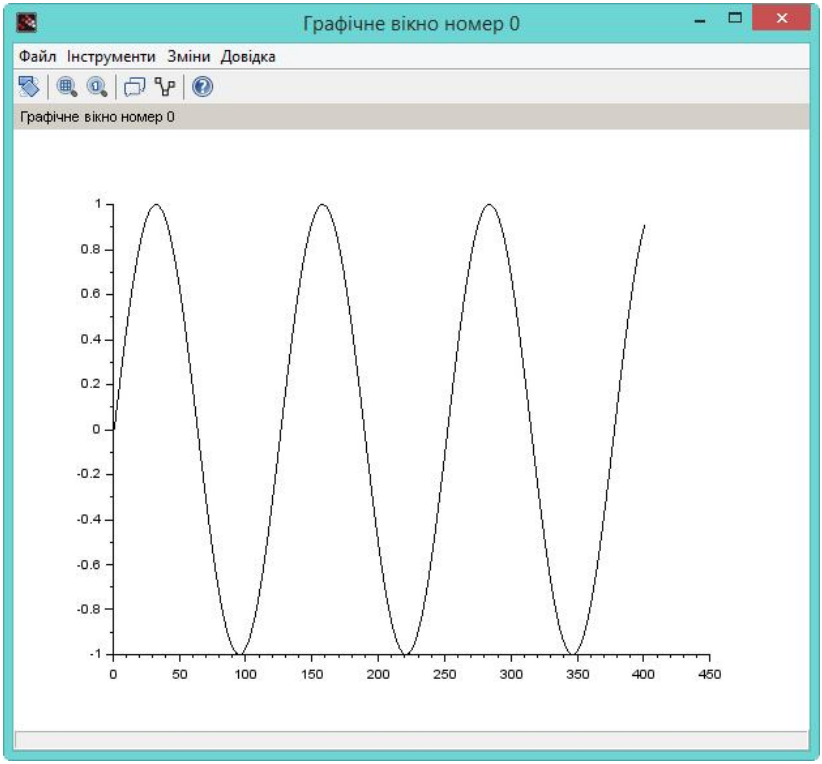
Як бачимо, при побудові графіків у полярній системі координат аргументами є перелік функцій, які відокремлюються комами.

Приклад:

polarplot(4*sin(fi), cos(8*fi))

Графічне вікно

Графік відображається у спеціальному графічному вікні, ім'я якого складається з тексту "Графічне вікно номер" і порядкового номера вікна. Наприклад, для першого вікна ім'я буде "*Графічне вікно номер 0*". Це вікно стає активним і саме в ньому будуються усі інші графіки.



Отже, якщо залишати це вікно відкритим, то це дозволяє будувати графіки кількох функцій в одному графічному вікні.

Для виведення графіка в іншому графічному вікні слід створити нове графічне вікно. Для цього в активному графічному вікні слід виконати команду **Файл ▸ Створити рисунок**. Після цього вже це вікно стає активним.

Але такий варіант є не дуже зручним і його можна використовувати під час роботи "в ручному режимі" з документом. Значно ефективнішим варіантом створення нового графічного вікна є застосування функції ***scf***, яка має такий синтаксис:

scf([номер_графічного_вікна])[;]

Аргумент "номер_графічного_вікна" є не обов'язковим. За його відсутністю система надасть йому значення (порядковий номер) першого вільного вікна. Функція повертає достатньо велику кількість параметрів вікна, тому за відсутності одержання якогось параметру доцільно закінчувати функцію символом ";".

Такий варіант створення графічного вікна має ще одне відчутне достоїнство: він дозволяє відкрити і не закривати в сесії кілька графічних вікон.

Scilab має також і функцію ***clf*** для повного очищення графічного вікна, що має синтаксис, аналогічний із функцією ***scf***.

clf([номер_графічного_вікна])

Головне меню вікна містить три пункти:

1. **Файл**. Серед команд цього пункту відмітимо такі:

- **Експортувати**. Дозволяє зберегти графік у кількох графічних форматах, у тому числі таких поширених як GIF, BMP, а також PDF.
- **Копіювати до буфера**. Дозволяє скопіювати графік до буферу обміну.

2. **Інструменти**.

- **Збільшити**. Збільшення розміру ділянки графіка. Після вибору цього інструмента з'являється рамка, за допомогою якої обирається область для збільшення.
- **Обертання на площині/у просторі**. Для цього потрібно виконати команду **Інструменти ▸ Обертання на площині/у просторі** або на панелі інструментів натиснути кнопку цього інструмента  , після чого встановити курсор на графік, натиснути праву кнопку миші і, не відпускаючи її, виконати обертання графіка.

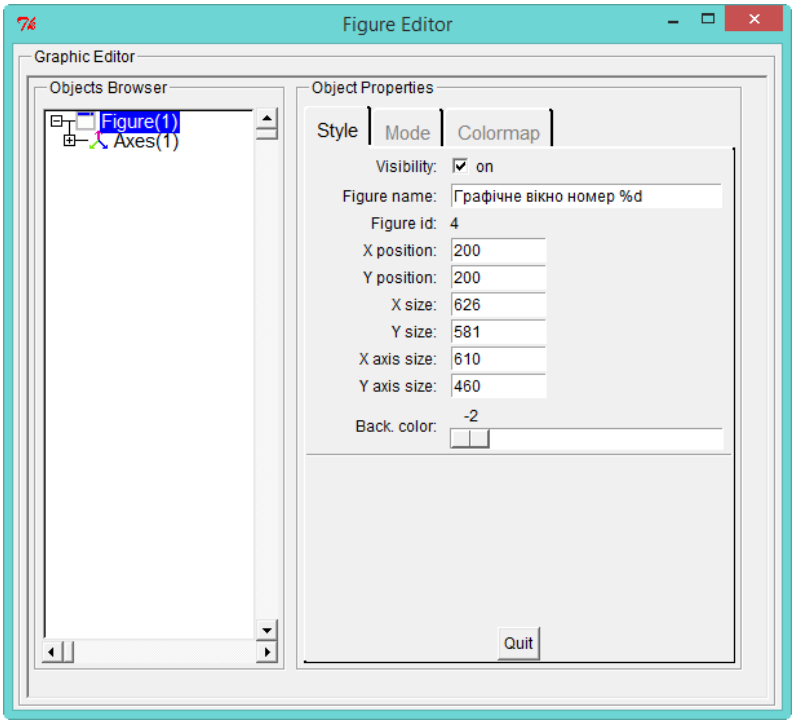
3. **Зміни**. Містить кілька дій, основними з яких є редагування загальних властивостей графіка або властивостей осей графіка.

Під рядком команд розташована панель стандартних інструментів:

1. Обертання графіку.
2. Збільшення ділянки.
3. Повернення до початкових розмірів.
4. Підбір за вмістом.
5. Увімкнути або вимкнути режим підписів до даних.
6. Зміна режиму внесення змін до даних кривої.
7. Допомога.

Графічний редактор (GED)

Звернутися до графічного редактора можна за командою **Зміни ▸ Властивості рисунка**. Після цього з'являється вікно "*Fugure Editor*" (редактор графіка), у якому можна змінити параметри графіка, що знаходиться у графічному вікні.



Вікно поділено на дві частини: ліва містить перелік об’єктів (“Objects Browser”), а права (“Object Properties”) – відображає властивості об’єкта після виділення його в лівій частині.

Інформація у лівій частині вікна структурована таким чином:

1. *Figure*. Загальні властивості графіка. Наприклад, тут змінюються координати осей, колір фону.

2. *Axes*. Властивості осей графіка, при цьому для кожної осі відводиться своя вкладка. Користувач може змінити місце розташування масштабної шкали, вивести допоміжну сітку, змінити її колір і т. ін.

3. *Compound*. Загальні властивості всіх графіків. Таких можливостей не багато: практично тут можна тільки

тимчасово приховати самі графіки, знявши прапорець біля дії “Visibility: on”.

4. *Polyline*. Властивості окремого графіка. Тут можна змінити колір лінії, її форму, товщину та багато іншого.

Інший варіант звернення до графічного редактора полягає у застосуванні команди **Зміни ▸ Властивості осей**. Знов-таки це призводить до відображення вікна “*Axes Editor*” (редактор осей), але при цьому відразу відкривається доступ до властивостей осей графіка.

Побудова 3-D графіка

Для побудови тривимірного графіка використовується функція ***plot3d***, яка за найпростішим варіантом має синтаксис:

plot3d(x’,y,z)

, де *x* і *y* – вектори однакової розмірності *n*, z – матриця розмірністю *n***n*, елементи якої розраховуються за значеннями *x* і *y*.

Така само, як і двовимірний графік, побудований 3-D графік можна обертати.

Зміна параметрів графіка із середовища Scilab

За замовчуванням графік будується з конкретними параметрами, до яких належать шрифти, колір, розміри і т. ін. Для зміни цих параметрів після побудови графіка використовується функція ***xset***. Синтаксис функції:

xset(ім’я параметра[, арг_1, арг_2 арг_3, арг_4, арг_5])

де арг_1–арг_5 – додаткові аргументи, що конкретизують 1-й параметр; їх кількість і значення залежить від значення першого аргументу. Аргумент “ім’я параметра” може набувати досить багато значень, наприклад:

- default – відновлення параметрів за замовчуванням;
- “foreground”, color – встановлення основного кольору;
- “background”, color – встановлення кольору фона;

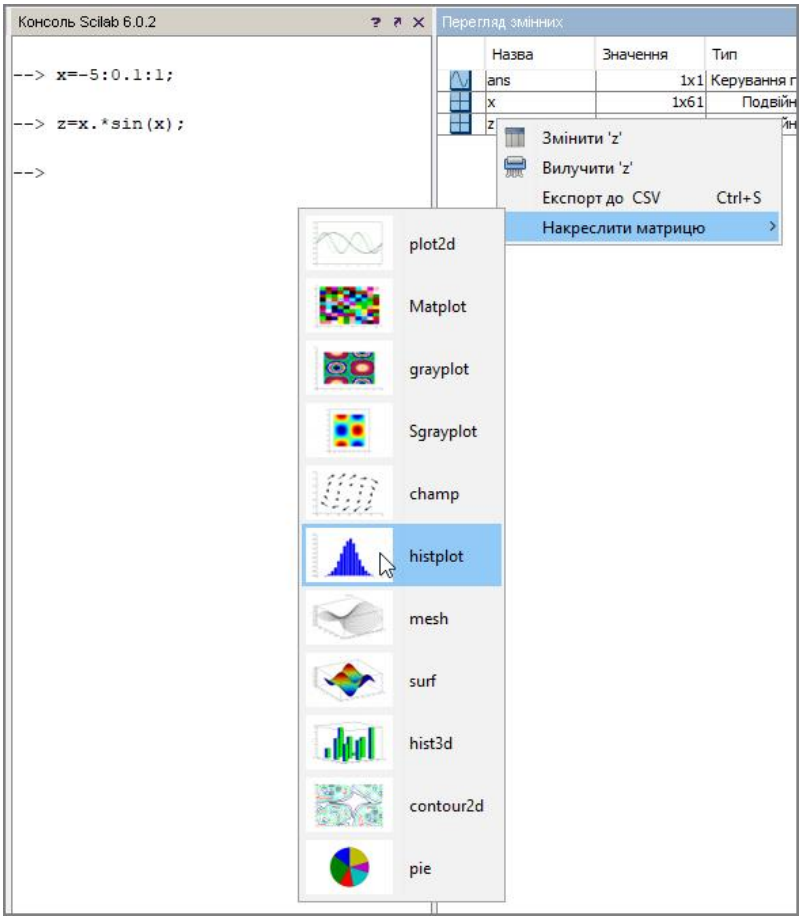
- “color”, value – встановлення основного кольору;
- “font”, fontid, fontsize – встановлення шрифту;
- “font size”, fontsize – встановлення розміру шрифту;
- “line style”, value – встановлення стиля шрифту і т. ін.

З повним переліком аргументів можна ознайомиться у довідковій системі. Конкретні значення того чи іншого аргументу можна також визначити з довідкової системи. Наприклад, значення аргументу “color” можуть бути: 0 або 1 – чорний колір, 2 – синій, 3 – зелений, 4 – яскраво блакитний, 5 – червоний і т. ін. Наприклад, встановлення зеленого кольору для основного кольору відбувається за допомогою такого введення:

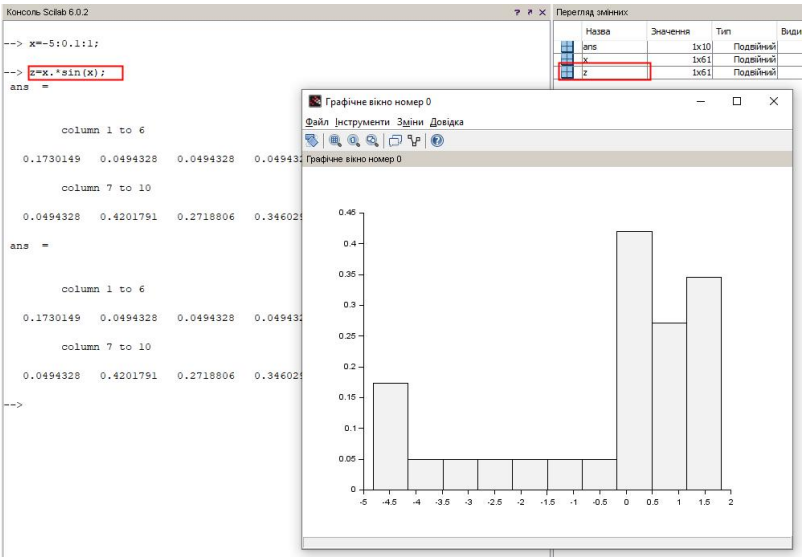
```
color=3;
xset(“foreground”, color)
```

Побудова графіку з вікна перегляду змінних

Система взагалі дозволяє користувачеві дуже просто побудувати кілька поширених типів графіку для змінної. Для цього у вікні перегляду змінних слід викликати контекстне меню на рядку з іменем змінної, після чого з’явиться меню з переліком видів графіків, з якого вибираємо потрібний.



Наприклад, вибираємо гістограму (histplot):



Розв'язування завдань лінійної алгебри

Функції знаходження числових характеристик матриці

Для визначення кількості рядків і стовпчиків матриці *M* використовують функцію ***size***(M), аргументом якої є ім'я масиву:

```
-->M=[1 2; 3 4; 5 6]; size(M)
 ans =
    3. 2.
```

Якщо потрібно визначити тільки кількість рядків або тільки кількість стовпчиків матриці, то синтаксис функції модифікується: додається другий аргумент, який має значення "1" або "r", якщо слід визначити кількість рядків, "2" або "c" – якщо слід визначити стовпчиків. Наприклад:

```
-->M=[1 2; 3 4; 5 6]; size(M,1)
 ans =
    3.
```

Загальну кількість елементів матриці або довжину вектора обчислює функція ***length***(M), аргументом якої є ім'я матриці або вектора:

```
-->M=[1 2; 3 4; 5 6]; length(M)
 ans =
      6.
-->V=[1 2 3 4]; length(V)
 ans =
      4.
```

Для визначення максимального за значенням елемента матриці або вектора використовується функція ***max***(A), аргументом якої є ім'я матриці або вектора. Якщо потрібно визначити максимальне значення для кожного з рядків або стовпчиків матриці, то синтаксис функції модифікується: додається другий аргумент, який має значення "r", якщо слід визначити максимальне значення для кожного рядка, або "c" – якщо для кожного стовпчика:

```
-->M=[1 2; 3 4; 5 6];
-->max(M)
 ans =
      6.
-->max(M,'c')
 ans =
      2.
      4.
      6.
```

Аналогічно застосовується функція ***min***(A), яка призначена для визначення мінімального за значенням елемента матриці або вектора.

Для обчислення визначника (детермінанту) квадратної матриці використовується функція ***det***(M), аргументом якої є ім'я матриці:

```
-->M=[1 2;3 4]
```

```
 M =
      1. 2.
      3. 4.
-->det(M)
 ans =
        – 2.
```

Для обчислення рангу матриці використовується функція ***rank***(M), аргументом якої є ім'я матриці:

```
-->M=[1 2; 3 4];
-->rank(M)
ans =
      2.
```

Функції, що реалізують чисельні алгоритми розв'язування задач лінійної алгебри

Обчислення матриці, оберненої до *M*

Нагадаємо, що *оберненою* по відношенню до матриці *M* називається така матриця, яка при її множенні на матрицю *M* дає одиничну матрицю.

Для обчислення оберненої матриці використовується функція ***inv***(M), аргументом якої є ім'я матриці, для якої знаходиться обернена матриця.

Приклад.

```
-->M=[1 2; 3 4]
 M =
      1. 2.
      3. 4.
-->A=inv(M)
 A =
      - 2. 1.
      1.5 - 0.5
-->M*A
```

```
 ans =
      1. 0.
      4.441D-16 1.
-->A*M
 ans =
      1. 0.
      1.110D-16 1.
```

Як бачимо, одержана обернена матриця A достатньо близька до одиничної, але все ж таки повністю не є одиничною, що є наслідком похибки чисельних обчислень.

Розв'язування системи лінійних алгебраїчних рівнянь типу $ax = b$

Система дає змогу розв'язувати системи лінійних алгебраїчних рівнянь типу $ax = b$. Для значень a формується матриця коефіцієнтів при невідомих, кожний рядок якої містить коефіцієнти одного рівняння, а для значень b формується вектор-стовпчик з вільних коефіцієнтів. Безпосередньо для розв'язку застосовують функцію ***linsolve***, яка має такий синтаксис:

linsolve(матриця коефіцієнтів при невідомих, вектор-стовпчик з вільних коефіцієнтів)

Функція повертає значення невідомих системи.

Приклад розв'язування системи лінійних рівнянь $\begin{cases} x1 + 2x2 - 7 = 0 \\ x1 + x2 - 6 = 0 \end{cases}$ має такий вигляд:

```
-->A=[1 2;1 1];b=[-7;-6];
-->x=linsolve(A,b)
 x =
      5.
      1.
```

Отже, шукані значення $x1$=5, $x2$=1.

Якщо система немає розв'язку, то про це видається

повідомлення “WARNING:Conflicting linear constraints!” (Конфліктуючі умови для лінійних рівнянь). Наприклад, така ситуація виникне при спробі розв’язування системи лінійних рівнянь $\begin{cases} x1 + x2 + 1 = 0 \\ x1 + x2 = 0 \end{cases}$:

```
-->A=[1 1;1 1]; b=[1;0];
-->x=linsolve(A,b)
WARNING:Conflicting linear constraints!
 x = []
```

Система лінійних алгебраїчних рівнянь може мати множину рішень, але функція все одне повертає тільки одне значення. Нижче наведений приклад такої ситуації при розв’язуванні системи лінійних рівнянь $\begin{cases} 3x1 - x2 - 1 = 0 \\ 9x1 - 3x2 - 3 = 0 \end{cases}$.

```
-->A=[3 -1;9 -3]; b=[-1;-3];
-->x=linsolve(A,b)
 x =
      0.3
      - 0.1
-->A*x+b
 ans =
1.0D-15 *
      - 0.3330669
      - 0.8881784
```

Систему лінійних алгебраїчних рівнянь можна розв’язати також використовуючи відомі математичні методи.

Наведемо приклад такого розв’язку за допомогою правила Крамера. Воно формулюється так. Якщо визначник $\Delta = \det(M)$ матриці системи з n рівнянь з n невідомими не дорівнює нулю, то система має єдине рішення, що визначається за формулою Крамера $x_i = \frac{\Delta_i}{\Delta}$, де Δ_i –

визначник матриці M, який був отриманий з цієї матриці шляхом заміни і-го стовпчика стовпчиком вільних членів системи рівнянь

Приклад

```
                    // M – матриця коефіцієнтів
-> M =

  3. -4.  2.
  2.  1.  4.
  5. -2. -1.
                    // SK – вектор вільних членів
--> SK =
  5.
  9.
  3.
```

// M1 – матриця коефіцієнтів із заміною першого стовпчика на вектор вільних членів

```
M1=M;M1(:,1)=SK

--> M1 =
  5. -4.  2.
  9.  1.  4.
  3. -2. -1.
```

// M2 – матриця коефіцієнтів із заміною другого стовпчика на вектор вільних членів

```
M2=M;M2(:,2)=SK

--> M2 =
  3.  5.  2.
  2.  9.  4.
  5.  3. -1.
```

// M3 – матриця коефіцієнтів із заміною третього стовпчика на вектор вільних членів

```
M3=M;M3(:,3)=SK
--> M3  =
  3.  -4.   5.
  2.   1.   9.
  5.  -2.   3.
                    // D – визначник матриці коефіцієнтів
--> D=det(M) =
 -85.
-->d  =
                    // Визначники матриць M1, M2, M3
d(1)=det(M1); d(2)=det(M2); d(3)=det(M3);
-91.
 -31.
 -138.
                    // Значення невідомих
--> x  = d/D
  1.0705882
  0.3647059
  1.6235294
                    // Перевірка
-> P = M*x =
  5.
  9.
  3.
```

Робота з поліномами

Нагадаємо, що *поліномом* або *алгебраїчним рівнянням* називається будь-яке рівняння виду $a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n = 0$.

Для створення полінома використовується функція ***poly***.

Синтаксис функції:

[p]=**poly**(a, “x”, [“flag”])

, де p – це ім’я полінома (його можна і не задавати).

Аргументи функції: a – матриця або дійсне число, x – символьна змінна, “flag” – символьна змінна, яка визначає спосіб завдання полінома і може набувати значення “roots” (скорочення “r”) або “coeff” (c). За замовчуванням – “roots”. Якщо “flag” має значення “r”, то поліном створюється з параметрами a_i для відповідних змінних x_i. Якщо “flag” має значення “c”, то значення параметрів a_i сприймаються як корені, для яких потрібно розрахувати коефіцієнти полінома.

Функція повертає вектор коефіцієнтів полінома.

Наступний приклад демонструє використання функції ***poly*** для створення поліномів p1, який має корінь що дорівнює “2” і p2 з коефіцієнтом “2”:

```
-->p1=poly(2,"x","r")
 p1 =
      - 2 + x
-->p2=poly(2,"x","c")
 p2 =
          2
```

У наступних прикладах створюються поліноми з відповідними коефіцієнтами для кубічного рівняння.

```
-->poly([1 2 3 4],"x","c")
 ans =
```

$$1+2x+3x^2+4x^3$$

```
-->poly([1 0 0 4],"x","c")
 ans =
```

$$1+4x^3$$

З поліномами можна виконувати дії множення, ділення (створювати з них дроби), додавати та віднімати.

Розв'язування рівняння з одним невідомим

Scilab може розв'язувати алгебраїчне рівняння з *одним* невідомим. Наприклад, потрібно знайти корені для рівняння $x^2 = 1$. Якщо – як у цьому прикладі – рівняння не представлено у вигляді поліному, то його попередньо слід перетворити в поліном: $x^2 - 1 = 0$. Після цього застосовується функція ***roots***, єдиним аргументом якої є ім'я полінома. Функція повертає знайдені корені полінома.

Приклад.

```
-->p=poly([-1 0 1],'x','c')
 p =
      -1+x^2
-->R=roots(p)
 R =
      1.
      - 1.
```

Якщо рівняння не має розв'язку на множині дійсних чисел ($x^2 + 1 = 0$), то *Scilab* шукає рішення серед комплексних чисел:

```
-->p=poly([1 0 1],'x','c')
 p =
      1+x^2
-->R=roots(p)
 R =
      i
      - i
```

Операції математичного аналізу

Обчислення сум елементів матриці

Для обчислення суми значень елементів матриці призначена функція ***sum***, яка за найпростішим варіантом має синтаксис:

sum(ім'я масива)

де ім'я масиву – вектор або матриця.

Якщо слід обчислити окремо суму значень кожного стовпчика матриці, то другий аргумент функції є числом "1", а якщо для рядка – "2".

Обчислення добутків елементів матриці

Для обчислення добутків використовується функція ***prod***, яка за найпростішим варіантом має синтаксис:

prod(ім'я масива)

де ім'я масиву – вектор або матриця.

Якщо слід обчислити окремо добуток значень кожного стовпчика, то другий аргумент функції є числом "1", а якщо для рядка – "2".

Інтеграли

Для знаходження визначених інтегралів в системі використовується функція ***intg***. У найпростішому вигляді функція має такий синтаксис:

intg(нижня межа інтегрування, верхня межа інтегрування, f),

де *f* – це підінтегральний вираз у вигляді текстового рядка або функції користувача.

Функція ***intg*** у відповідь повертає значення визначеного інтегралу:

```
-->intg(1,5,f)
ans  =
    - 5.8927325
```

Результат функції можна записати у змінну:

```
-->k=intg(1,5,f)
```

Диференціали

Для знаходження диференціалів у певній точці використовується функція ***numderivative***, яка за найпростішим варіантом має такий синтаксис:

numderivative(f, coord)

де f – це є ім'я функції користувача, що диференціюється, а coord – координата точки, для якої потрібно обчислити похідну. Координати точки можуть бути задані ідентифікатором (як скаляром, так і вектором) або безпосередньо числом.

Наприклад:

```
-->function y=f(x), y=x^2+1/2, endfunction
-->numderivative(f,1)
ans  =
    2.
```

Scilab чисельна система, але вона дозволяє виконати деякі символьні обчислення. Наприклад система має функцію ***devart***, що обчислює в аналітичному вигляді похідну полінома.

```
--> p1=poly([3 -5 1], "y", "c")
 p1  =
        2
  3 -5y +y

--> derivat(p1)
 ans  =
  -5 +2y
```

Розв'язування диференційних рівнянь

Для розв'язування диференційних рівнянь або систем диференційних рівнянь застосовується функція ***ode***, яка за

найпростішім варіантом має синтаксис:

ode(y_0,x_0,C,f)

де y_0 – початкова умова: дійсне число для одного диференційного рівняння для або вектор для системи диференційних рівнянь,

x_0 – початкове значення інтервалу інтегрування: дійсне число для одного диференційного рівняння для або вектор для системи диференційних рівнянь,

C – координати осі x: початкова_координата:крок:кінцева_координата (використовується зокрема під час побудови графіка функції),

f – функція користувача – права частина рівняння або системи рівнянь.

Результатом роботи функції є множина одержаних чисельних значень розв'язку.

Приклад. Знайти загальний розв'язок звичайного диференційного рівняння першого порядку $y' - 10x = 0$ на інтервалі [2,10] з початковою умовою $y_0 = -1$.

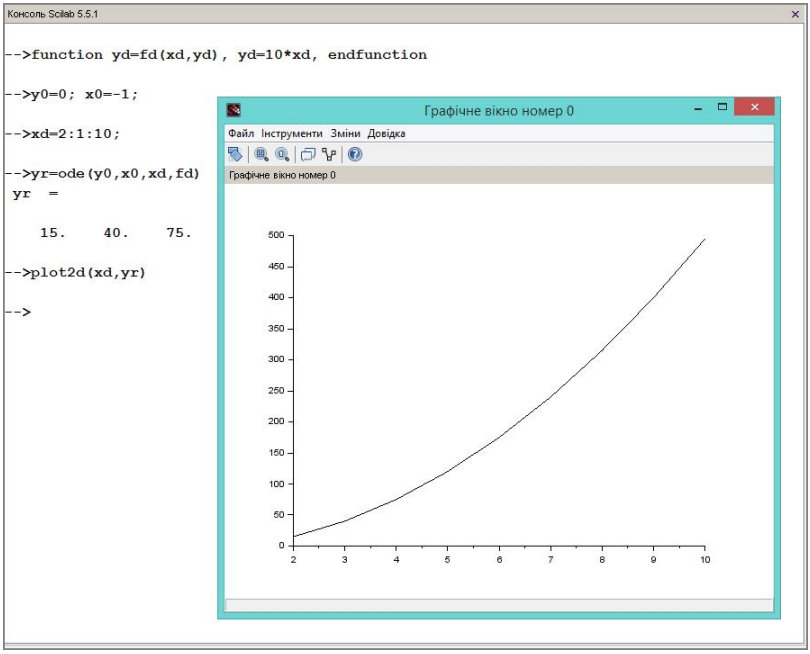
Розв'язування буде складатися з такої послідовності:

```
-->function yd=f(x,y), yd=10*x, endfunction;

-->y0=0; x0=-1;

-->x=2:1:10;

-->y=ode(y0,x0,x,f)
```

У першому рядку формується користувацька функція, на яку у функції ***ode*** здійснюється посилання. Оскільки вона має являти собою праву частину диференційного рівняння, то початкове рівняння перетворюється у вигляд $y' = 10x$.

У другому рядку задається початкова умова $y_0 = -1$, а у третьому – інтервал зміни незалежної змінної [2,10].

Одержимо графічний розв'язок диференційного рівняння. Для цього застосуємо функцію ***plot2d***(x,y).



Як бачимо, координатна сітка для осі x як раз і відбиває інтервал її зміни [2,10].

Слід мати на увазі, що початкове значення x_0 має бути більшим за початкове значення координати осі x. У протилежному випадку система генерує помилку на зразок наведеної нижче:

```
-->function yd=fd(xd,yd), yd=10*xd, endfunction

-->y0=0; x0=3;

-->xd=2:1:10;

-->yr=ode(y0,x0,xd,fd)
intdy--  t (=r1) illegal
      where r1 is :   0.3000000000000D+01
      t n est pas entre tcur - hu (= r1) et tcur (=r2)
      where r1 is :   0.2648986971070D+01   and r2 :   0.1594894424187D+01
lsoda--  problems due to intdy. itask=i1,tout=r1
      where i1 is :          1
      where r1 is :   0.3000000000000D+01
                    !--error 9999
illegal input
```

Як вище було зазначено, функція ***ode*** може бути використана для розв'язування систем диференційних рівнянь.

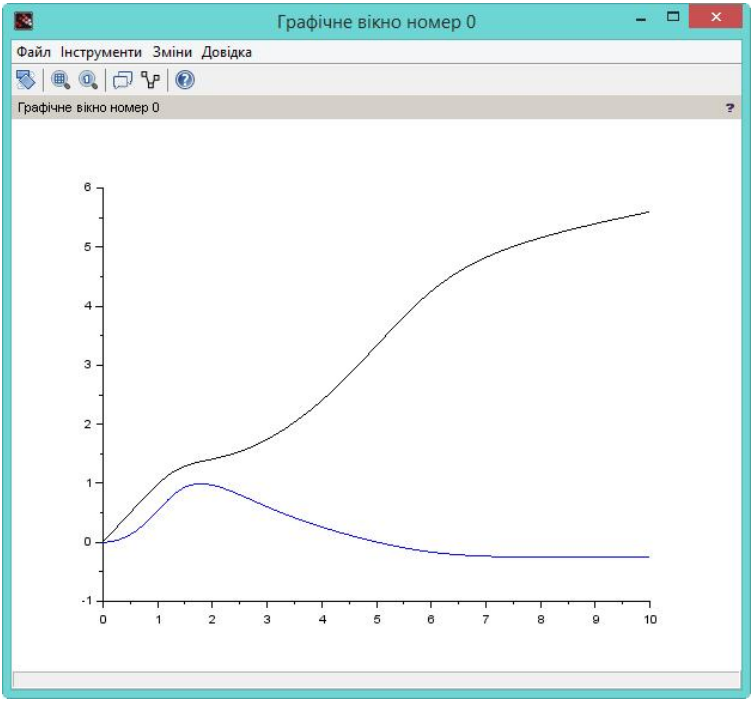
Приклад. Розв'язати систему диференційних рівнянь $\begin{cases} x' = \cos(y \cdot x) \\ y' = \sin(x + ty) \end{cases}$ на інтервалі $[0,10]$ з початковими умовами x_0 і y_0.

Розв'язування буде мати такий вигляд:

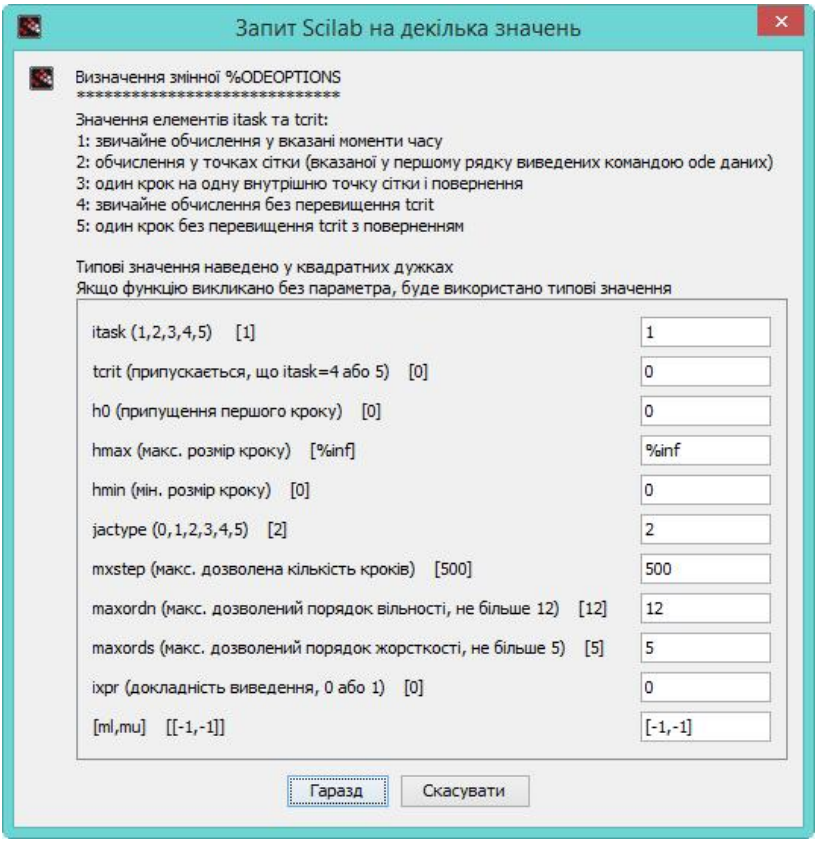
```
-->function difur=syst(t,y)
-->difur=zeros(2,1);
-->difur(1)=cos(y(1)*y(2));
-->difur(2)=sin(y(1)+y(2)*t);
-->endfunction
-->x0=[0;0]; t0=0;
-->t=0:0.1:10;
-->y=ode(x0,t0,t,syst)
```

Як бачимо з наведеного фрагменту, єдиною відмінністю під час розв'язування систем диференційних рівнянь порівняно до розв'язування окремих диференційних рівнянь є створення двох векторів: правих частин рівнянь (difur) і початкових умов (x0).

Графічне розв’язування диференційного рівняння за допомогою функції ***plot2d***(x,y) має такий вигляд.



Scilab дозволяє налаштувати “під себе” параметри обчислення для розв’язування диференційних рівнянь. Для цього він має спеціальний засіб, звернення до якого здійснюється за функцією ***odeoptions()***. Назва функції вводиться саме маленькими літерами. Після звернення до функції з’явиться діалогове вікно, у якому користувач і задає параметри обчислення для розв’язування диференційних рівнянь. Інформація у верхній частині вікна надає користувачу інформацію відносно призначення і значень за замовчуванням, що використовує система під час розв’язування.



Вище було розглянуто застосування функції ***ode*** у найпростішому вигляді. Але вона має досить велику кількість аргументів, що відчутно збільшують можливості користувача під час розв'язування диференційних рівнянь.

Наприклад, автоматично у процесі розв'язування *Scilab* здійснює вибір між методом прогнозування і корекції (nonstiff predictor-corrector) для нежорсткої і методом BDF (Backward Differentiation Formula) Адамса-Мултона для жорстких задач. Але система дозволяє користувачеві і самостійно його визначати. Це здійснюється шляхом застосування у функції спеціального аргументу, що

записується першим у списку аргументів і подається у символьному вигляді у подвійних лапках, наприклад “adams”.

Система пропонує кілька методів, зокрема:

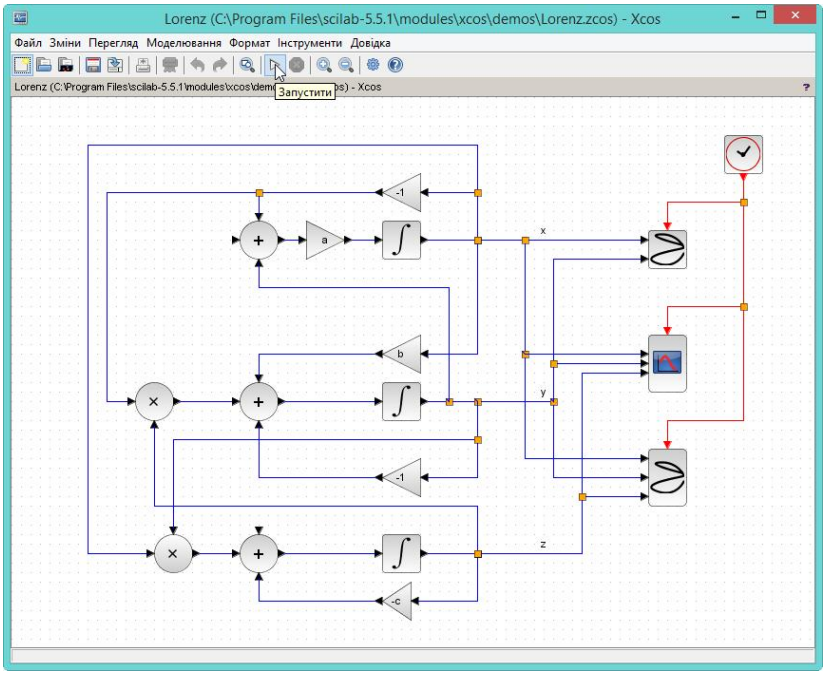
- *rk4* – адаптивний метод Рунге-Кута 4-го порядку;
- *rkf* – побудовано на методі Рунге-Кута-Фельберга (Fehlberg's Runge-Kutta) 4-5-го порядку точності із автоматичним оцінюванням похибки;
- *fix* – метод Рунге-Кута з фіксованим кроком.

XCOS – моделювання динамічних систем

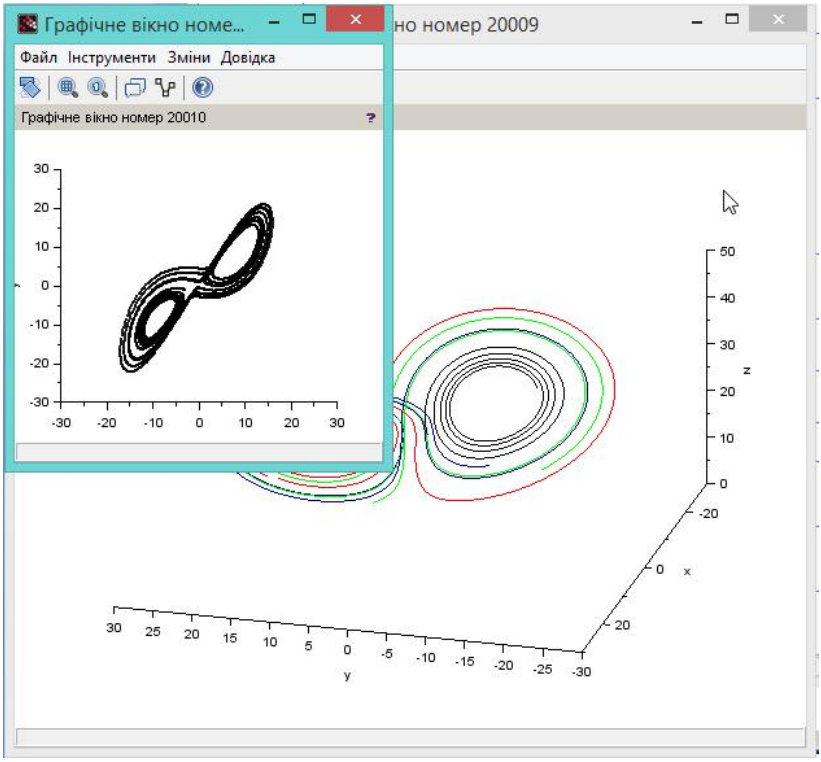
У складі системи є модуль, за допомогою якого можна моделювати динамічні моделі. Доступ до нього здійснюється за командою **Програми ▸ XCOS**, після чого з’являється вікно нового документа “*Без назви*” модуля XCOS і вікно, що містить перелік палітр інструментів моделювання.

Побудова моделі здійснюється у вікні “*Без назви*” шляхом перетягування потрібного інструмента з вікна потрібної палітри інструментів.

Scilab має бібліотеку прикладів побудови моделей у різних галузях, що дає можливість користувачам з’ясувати принципи побудови моделей. Доступ до прикладів здійснюється за командою **Довідка ▸ Демонстрації Scilab**, а потім вибором у вікні “*Демонстрації*” пункту “XCOS”. Приклади моделей згруповані за їх функціональним призначенням, наприклад, “Електричні системи”, “Системи керування” і т. ін. Для перегляду роботи моделі слід двічі натиснути на її назві зі списку. Це призведе до появи вікна з моделлю.



Для запуску моделі на виконання у цьому вікні слід виконати команду **Моделювання ▸ Запустити** або натиснути кнопку **«Запустити»**  на панелі інструментів. Це призведе до появи графічного вікна, в якому і буде відображатися процес моделювання.


Графічне вікно номе...
Файл Інструменти Зміни Довідка
Графічне вікно номер 20010
но номер 20009
30
20
10
0
-10
-20
-30
-30 -20 -10 0 10 20 30
50
40
30
20
10
0
-20
0
x
z
20
30 25 20 15 10 5 0 -5 -10 -15 -20 -25 -30
y

Лабораторні роботи

Лабораторна робота № 1

Тема. Введення даних. Математичні операції

Мета. Набування навичок роботи зі системою в режимі прямих обчислень.

Вказівки до виконання:

1. Для зміни локалізації (мови інтерфейсу) програми слід звернутися до настройок програми, у групі “Загальні” вибрати мову із списку “Типова мова”, а потім натиснути кнопку **«ОК»**. Систему після цього потрібно перезавантажити.

2. За замовчуванням система після її завантаження шукає документи системи в системній папці Documents and Setting/Ім’я користувача. Для зміни папки слід виконати команду **Файл ▸ Змінити поточний каталог**, знайти потрібну папку і натиснути **«ОК»**. Визначити папку, де саме зберігаються документи, можна за командою **Файл ▸ Показати поточний каталог**.

3. Для збереження лабораторної роботи з метою подальшої демонстрації її роботи є збереження тільки введення в документ, тобто сценарію (файла з розширення SCE). Це можна зробити різними шляхами, наприклад скопіювати виконану роботу до текстового редактора або редактора SciNotes і вилучити усю зайву інформацію, в тому числі символи “-->”, залишивши тільки введення на зразок такого:

```
// Моя перша програма
%pi*100+179
a=1; b=2;
c=a+b
```

4. Визначення змінних і функцій, які розташовані в робочій області пам’яті, можна записати на диск, виконавши команду **Файл ▸ Зберегти середовище…** з

розширенням BIN. Завантаження з диска цих даних здійснюється за командою **Файл ▸ Завантажити середовище…**

Завдання

1. Змініть шрифт інтерфейсу системи на “Courier”.
2. Змініть папку для збереження інформації на Вашу папку.
3. Встановіть режим запису тексту сесії в текстовий файл з довільним ім’ям.
4. Введіть у документ коментар, наприклад, “Моя перша програма у Scilab”.
5. Припиніть запис тексту сесії.
6. Відкрийте створений текстовий файл і переконайтеся, що він містить введену інформацію.
7. Відновіть режим запису тексту сесії в текстовий файл.
8. Обчисліть вирази:

- π*100+179.
- $c = a + b$ при $a = 2, b = 3$
- R = 157,43 : 23,456.

9. При виведенні змінної R установіть кількість позицій дробової частини “12”.
10. Виконайте такі дії з комплексними числами:

- сформуйте комплексні числа x=6+2i і y=7-2i і виконайте з ними операції додавання, віднімання, ділення і множення;
- добудьте корінь $\sqrt{x}$;
- обчисліть тангенс *y*.

11. Виведіть список системних змінних і змінних, що використовуються в поточній сесії.
12. Запам’ятайте змінні і їх значення з робочої області пам’яті поточної сесії.
13. Припиніть запис тексту сесії.

14. Закінчіть роботу з Scilab.
15. Завантажте збережені дані попередньої сесії.
16. Обчисліть вираз $l = c + a + b$, використовуючи значення змінних результатів збереженої сесії.
17. Знищить змінну *a*.
18. Спробуйте обчислити вираз $l = c + a + b$. Що відбудеться?
19. Знищить усі змінні.
20. Відкрийте вікно перегляду змінних і переконайтеся у відсутності створених Вами змінних.
21. Обчисліть вираз $c = a + b$ при $a = 2, b = 3$.
22. Переконайтеся у наявності створених Вами для обчислення виразу змінних.
23. Створіть сценарій, у якому передбачте виконання дій для обчислення виразів з пункту 8.
24. Збережіть сценарій.
25. Закінчіть роботу з Scilab.
26. Відкрийте Scilab і завантажте сценарій у новій сесії.
27. Виконайте сценарій (виведенням).
28. Виведіть значення змінних, що містить сценарій.
29. Створіть сценарій повного виконання лабораторної роботи. Переконайтеся у його працездатності.

 *Контроль знань та навичок*

Після виконання лабораторної роботи студент повинен ***знати***:

1. Для яких розрахунків призначена система?
2. Аналогом якої системи є Scilab?
3. Як змінити папку для збереження інформації?
4. Для чого змінюється папка для збереження інформації?
5. Що таке “сесія”?
6. Що таке “сценарій”?
7. Як створити коментар?

8. Яке призначення редактора сценаріїв?

9. Як створити сценарій?

10. Як встановити, припинити режим запису тексту сесії?

11. Як запам'ятати визначення змінних і функцій, які розташовані в робочій області пам'яті?

12. Як завантажити визначення змінних і функцій, які розташовані в робочій області пам'яті?

13. Що слід зробити для того, щоб результат виконання математичного виразу не було відображено на екрані?

14. Скільки значущих цифр система виводить на екрані за замовчуванням?

15. Які принципи запису комплексних чисел.

16. Що таке "системна змінна"?

17. Які є основні системні змінні системи і як ввести їх в документ?

18. Як одержати список змінних, що використовуються в поточній сесії?

19. Як можна знищити окрему змінну, всі змінні?

20. Як у вікні консолі відкрити вікно перегляду змінних?

21. Як запам'ятати результати сесії?

22. Що містять результати сесії?

23. Як можна визначити значення змінної?

Після виконання лабораторної роботи студент повинен ***уміти***:

1. Створити коментар.

2. Змінити папку для збереження інформації.

3. Встановити, припинити режим запису тексту сесії.

4. Запам'ятати змінні та їх значення з робочої області пам'яті.

5. Завантажити збережені змінні і їх значення з робочої області пам'яті.

6. Виконати математичні операції над дійсними і комплексними числами.

7. Змінити значність при виведенні результатів.

8. Одержати список змінних, що використовуються в поточній сесії.

9. Знищити окрему змінну, всі змінні.

10. Відкрити вікно перегляду змінних.

11. Визначити значення змінної.

12. Запам'ятати результати сесії.

13. Використовувати системні змінні.

14. Виконати операції з комплексними числами.

15. Сформувати файл-сценарій.

16. Завантажити файл сценарію у вікно консолі.

Лабораторна робота № 2

Тема. Створення векторів і матриць (масивів). Основні операції над масивами

Мета. Навчитися створювати вектори і матриці, виконувати з ними операції.

Вказівки до виконання:

1. Виконання лабораторної роботи здійснюється за допомогою редактора сценаріїв.

Завдання

1. Створіть вектори-рядки з 5 елементів кожний:

- *V1* – шляхом надання елементам вектора значень 1, 2, 3, 4, 5;
- *V2* – з вектора *V1* шляхом додавання до кожного його елемента числа "5";
- *V3* – з початковим значенням "індивідуальний номер" і кроком "2,5".

2. Введіть змінну *x* і надайте їй значення "індивідуальний номер".

3. Виконайте операцію множення вектора *V1* на *x*.

4. Виконайте операцію додавання векторів *V1* і *V2*.

5. Створіть вектор *V4* шляхом транспонування вектора *V1*.

6. Виконайте операцію множення векторів *V1* і *V4*.

7. Виконайте такі дії з елементами вектора *V1*:

- $V_1 - V_3$;
- $V_2 \cdot V_4$;
- V_3 / V_1;
- $\sqrt{V_2}$;
- V_3^2.

17. Створіть дві матриці *M1* і *M2* з двох рядків і п'яти стовпчиків кожна. Значення матриць можуть бути будь-якими, окрім нульових.

18. Здійсніть з цими матрицями наступні дії:

- поелементне множення;
- поелементне ділення справа наліво;
- поелементне ділення зліва направо.

19. Створіть матрицю *M3* шляхом транспонування матриці *M1*.

20. Створіть матрицю *M4* з векторів *V1* і *V2*.

21. Виконайте операцію множення вектора *M1* на скаляр *x*.

22. Виконайте множення матриць *M1* і *M3*.

23. Виконайте додавання матриць *M1* і *M2*.

24. Виконайте дії з елементами матриць:

- $M1_{1,1} + M2_{2,1}$;
- $M2_{1,1} \cdot M3_{1,2}$;
- $\sqrt{M3_{1,2}}$;
- $M1_{2,1}^2$.

25. З другого рядка матриці *M1* створіть вектор-рядок *V5*.

26. Створіть матриці *M5* і *M51* шляхом поєднання матриць *M1* і *M2* по рядках і по стовпчиках.

27. За допомогою відповідної функції створіть одиничну матрицю *M6* розмірністю 5*5.

28. За допомогою відповідної функції створіть матрицю *M7* розмірністю 5*5, всі елементи якої дорівнюють “0”.

29. За допомогою відповідної функції створіть матрицю *M8* розмірністю 5*5, всі елементи якої дорівнюють “1”.

30. Створіть вектор-рядок *V5* з п’яти елементів, елементи якого сформуйте генератором випадкових чисел.

31. Створіть діагональну матрицю *M8*, елементами головної діагоналі якої є значення вектора *V1*.

32. Запам’ятайте змінні та їх значення з робочої області пам’яті поточної сесії.

 *Контроль знань та навичок*

Після виконання лабораторної роботи студент повинен ***знати***:

1. Що таке “транспонування”?
2. Які є варіанти створення вектора?
3. Які є варіанти створення матриці?
4. Як створити з матриці вектор-рядок, вектор-стовпчик?
5. Як виконати з матрицями операції поелементного множення, поелементного ділення справа наліво та зліва направо.
6. Що являє собою операція поелементного множення?
7. Що являє собою операція поелементного ділення?
8. За допомогою якої функції створюються одиничні матриці?

9. За допомогою якої функції здійснюється поєднання матриць?

10. За допомогою якої функції створюється матриця, всі елементи якої дорівнюють “0”, “1”?

11. Як створити вектор або матрицю, елементи якої повинні бути сформовані генератором випадкових чисел?

12. Як створити діагональну матрицю, елементами головної діагоналі якої є значення вектора *V*, а всі інші її елементи дорівнюють “0”?

13. Як створити вектор у вигляді послідовності цілих чисел з довільним кроком зміни?

Після виконання лабораторної роботи студент повинен ***уміти***:

1. Створити вектор шляхом надання значення кожному елементу окремо.
2. Створити вектор-послідовність за формулою.
3. Виконати з векторами операції множення, додавання, транспонування.
4. Виконати над окремими елементами вектора арифметичні операції.
5. Виконати з матрицями операції поелементного множення, поелементного ділення справа наліво та зліва направо.
6. Створити матрицю шляхом поєднання матриць.
7. Створити матрицю шляхом надання значення кожному елементу окремо.
8. Створити матрицю з кількох векторів.
9. Створити з матриці вектор-рядок, вектор-стовпчик.
10. Створити одиничну матрицю.
11. Створити матрицю, всі елементи якої дорівнюють “0”, “1”.
12. Створити вектор або матрицю елементами якої є випадкові числа.

13. Створити діагональну матрицю, елементами головної діагоналі якої є значення вектора *V*, а всі інші її елементи дорівнюють “0”.

14. Створити вектор у вигляді послідовності цілих чисел.

Лабораторна робота № 3

Тема. Побудова графіків

Мета. Навчитися будувати і форматувати графіки в середовищі системи.

Вказівки до виконання:

1. Виконання лабораторної роботи здійснюється за допомогою редактора сценаріїв.
2. Пункти 1, 7, 8 виконуються за варіантами.
3. Кожний графік будується в окремому вікні. Нумерацію вікон починайте з “0”.
4. Враховуючи велику розмірність векторів, виведення результатів не здійснюйте.
5. При побудові функцій для виконання дій над елементами матриць застосовуйте операції поелементного множення або ділення матриць зліва направо.
6. У вікні графічного редактора об’єкт “Figure” надає доступ до загальних властивостей графіка, “Axes” – властивостей осей графіка, “Compound” – загальних властивостей усіх графіків, “Polyline” – до властивостей окремого графіка.
7. Під час побудови графіка в полярній системі координат для завдання кольору осі використовуйте додатковий аргумент color, значення англійською мовою якого і визначає колір лінії, наприклад color(“red”).
8. За закінченням редагування у вікні графічного редактора натисніть кнопку **«Quit»**.

Завдання

1. Побудуйте графіки наступних функцій. Для першого графіка визначити функцію графіка перед застосуванням ***plot2d***, а для другого – опишіть її безпосередньо в ***plot2d***. Кожний графік повинен мати унікальний колір і будуватися у власному вікні, тобто усі графічні вікна не закриваються.

- $y(x) = \frac{a^3}{(a^2+x^2)}$; $f(x) = x \cdot (1 - \sin(x))$
- $y(x) = \frac{x^3}{1000}$; $f(x) = x \cdot (2 - \cos(2x))$

де a – константа, значення якої відповідає Вашому індивідуальному номеру, x змінюється від 0 до 20 з кроком 0,05.

2. Здійсніть повне очищення відкритого графічного вікна, а потім відновіть у ньому побудовані у попередньому пункті графіки.

3. Перегляньте панель інструментів графічного вікна та визначте, які інструменти вона містить.

4. Застосуйте відповідний інструмент для збільшення ділянки графіка та повернення до попереднього розміру.

5. Створіть вектор z як послідовність зі ста чисел за такими даними:

- значення першого елемента дорівнює "індивідуальному номеру";
- крок приросту "0,1".

6. За допомогою тільки однієї функції ***plot2d*** побудуйте в одному графічному вікні графіки функцій *sin(z)* і *cos(z)*.

7. Побудуйте в полярній системі координат в одному графічному вікні:

- три графіка функції $4\cos(3\varphi)$, де ϕ змінюється на інтервалі $[0; 2\pi]$:

7.1. з кроком "0,5", колір лінії – чорний;

7.2. з кроком "0,25", колір лінії – зелений;

7.3. з кроком "0,01", колір лінії – червоний;

- три графіка функції $4\sin(3\varphi)$, де ϕ змінюється на інтервалі [0; 2π]:

7.1. з кроком “0,5”, колір лінії – чорний;

7.2. з кроком “0,25”, колір лінії – зелений;

7.3. з кроком “0,01”, колір лінії – червоний.

Поясніть відмінність між графіками.

8. Створіть вектор *x2*, що змінюється на інтервалі $[-4 \cdot \pi; 4 \cdot \pi]$ з кроком 0,01. Побудуйте графік параметричної функції:

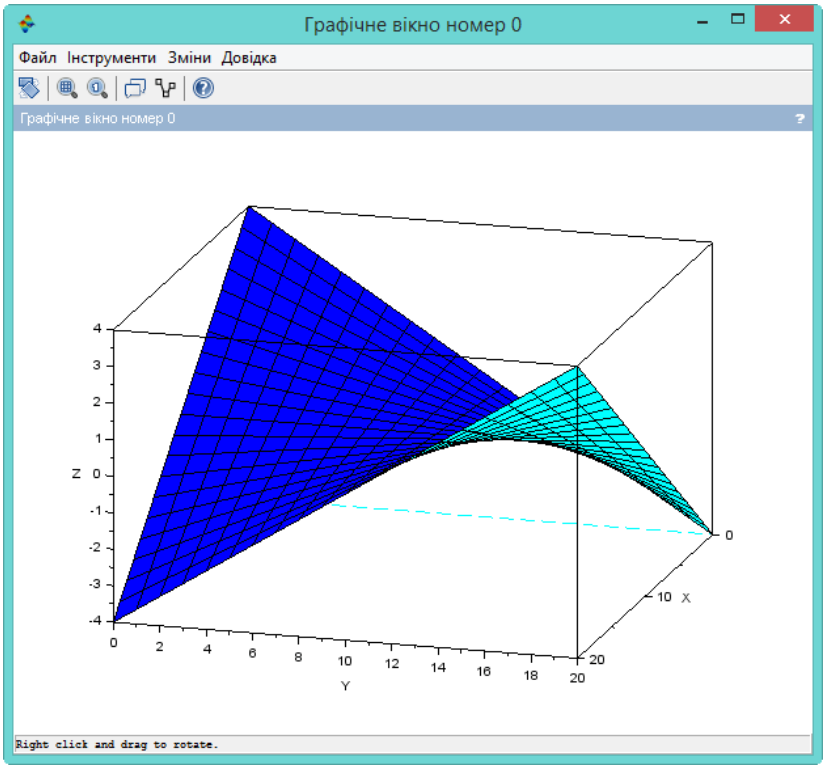
- $f(x2) = \frac{x2}{\cos(x2) \cdot \pi}$
- $f(x2) = x2 \cdot \pi \cdot \sin(2\pi)$

9. За допомогою графічного редактора виконайте для останнього графіка такі дії:

- змініть основний колір лінії графіка на червоний (властивість “foreground”);
- збільшить його розмір на всю ширину документа (властивість “X size”);
- встановіть лінії масштабної сітки для осі *x* (властивість “Grid color”);
- створіть загальний підпис до графіка (Axes, вкладка “Title”, поле “Text”).  Підпис вводиться у подвійних лапках.
- Збережіть графік у форматі JPEG.

10. Створіть вектор *x1*, що змінюється на інтервалі [0; 20] з кроком 1. Побудуйте матрицю ординат 3D-поверхні за формулою $M = \left\{\frac{(x1-10)}{5} \cdot \frac{(x1-10)}{5}\right\}$.

11. За допомогою інструмента обертання надайте графіку наступний вигляд:



12. Створіть для графіка підпис: “Приклад графіка функції в тривимірній декартовій системі координат”.

13. Збережіть графік у форматах PDF, GIF і BMP.

14. Побудуйте для матриці всі види 3D-графіків.

Контроль знань та навичок

Після виконання лабораторної роботи студент повинен ***знати***:

1. Що таке “2-D графік”?

2. Що потрібно зробити для того, щоб результат виконання математичного виразу не був відображений на екрані?

3. Яка функція призначена для побудови 2-D графіків? Які аргументи має ця функція?

4. Яка функція призначена для створення нового графічного вікна?

5. Яка функція здійснює повне очищення відкритого графічного вікна?

6. Які є варіанти створення нового графічного вікна?

7. Яка структура графічного вікна?

8. Яка функція призначена для побудови 3-D графіків? Які аргументи вона має?

9. Яка функція призначена для побудови графіка у полярній системі координат? Які аргументи вона має?

10. Які інструменти містить панель інструментів графічного вікна?

11. Як змінити основний колір лінії графіка?

12. Як збільшити ділянку графіка та повернутися до попереднього розміру?

13. Як встановити лінії масштабної сітки для осі?

14. Як створити для графіка загальний підпис?

15. У яких популярних форматах можна зберегти графік?

Після виконання лабораторної роботи студент повинен ***уміти:***

1. Побудувати 2-D і 3-D графік.

2. Побудувати графік у полярній системі координат.

3. Застосувати відповідні функції для створення нового графічного вікна та повного його очищення.

4. Розташувати в одній площині кілька графіків.

5. Розташувати кожний графік в окремому вікні, не закриваючи інші вікна.

6. Працювати з редактором графіка, в том числі змінити розмір графічного вікна, колір фону і лінії, відобразити сітку, створити загальний підпис до графіка.

7. Зберегти графік у різних графічних форматах.

8. Побудувати полярний графік.

9. Збільшити ділянку графіка та повернутися до попереднього розміру.

Лабораторна робота № 4

Тема. Математичний аналіз. Розв'язок рівнянь

Мета. Навчитися застосовувати оператори математичного аналізу Scilab. Набуття умінь та навичок розв'язку рівнянь.

1. Виконання лабораторної роботи здійснюється за допомогою редактора сценаріїв.

2. Пункти 7-11 виконуються за варіантами.

3. Під час виконання сценарію назви користувацьких функцій запам'ятовуються при іншому виконанні, зустрічаючи те саме ім'я система буде попереджувати про це повідомленням "Попередження: перевизначення функції…". Для запобігання цієї ситуації додайте перед рядком з назвою користувацької функції (або на початку сценарію) рядок із системною функцією funcprot(0), яка за таким варіантом відключає виведення попереджень.

Завдання

1. Створіть матрицю *M1* розмірністю 3*3 шляхом надання її елементам довільних значень, відмінних від нульових.

2. За допомогою відповідних функцій знайдіть для матриці *M1* такі її числові характеристики:

- кількість рядків, стовпчиків;
- кількість елементів;
- максимальний, мінімальний за значенням елемент;
- максимальний за значенням елемент 1-го рядка;
- максимальний за значенням елемент 2-го стовпчика;
- детермінант;
- ранг.

3. Створіть вектор-рядок *V1* з п'яти довільних елементів, значення яких відмінні від нуля.

4. За допомогою відповідних функцій знайдіть для нього такі числові характеристики:

- довжину;
- максимальний, мінімальний за значенням елемент.

5. Підрахуйте суму:

- елементів матриці *M1*;
- рядків матриці *M1*.
- елементів вектора *V1*.

6. Підрахуйте добуток:

- елементів матриці *M1*;
- стовпчиків матриці *M1*.
- елементів вектора *V1*.

7. Обчислить визначений інтеграл:

- $\int_0^4 (x^2 + 3x + 2) dx$;
- $\int_0^\pi \sqrt{(5 - c^3)}\, dc$.

8. Обчислить похідну для наступних функцій в точках 10, 1, 0 і 0,2:

- $f(t) = (a0 + a2 \cdot t^3) \cdot exp^{(a1 \cdot t)}$
- $f(t) = (a0 + a1 \cdot t^2) \cdot exp^{(-a2 \cdot t)}$

, де константа *a0* дорівнює *індивідуальному номеру*, *a1= a0+5, a2= a0+3*.

9. Розв'яжіть систему лінійних алгебраїчних рівнянь:

- $\begin{cases}2x-3y=0\\7x-5y=0\end{cases}$ $\begin{cases}3x+3y=3\\4x+4y=3\end{cases}$
- $\begin{cases}100x-2y=96\\-3x+57y=111\end{cases}$ $\begin{cases}7x-7y=-2\\6x-6y=-2\end{cases}$

10. Розв'яжіть диференційне рівняння першого порядку на інтервалі $[2,10]$ з початковою умовою $y_0=-1$:

- $y'+3{,}5x=0$
- $y'-\sin(x)=0$

11. Розв'яжіть систему диференційних рівнянь на інтервалі $[0,\pi]$ з початковими умовами $x(0)=3$ і $y(0)=0$.

- $\begin{cases}x'=x-1\\y'=x+2yt-3\end{cases}$
- $\begin{cases}x'=-2x+4y\\y'=-x+3yt\end{cases}$

Контроль знань та навичок

Після виконання лабораторної роботи студент повинен ***знати***:

1. Як визначити кількість рядків, стовпчиків, елементів матриці?
2. Як визначити максимальний або мінімальний елемент вектора або матриці?
3. Як визначити максимальний або мінімальний елемент конкретного рядка або стовпчика матриці?
4. За допомогою якої функції обчислюється детермінант матриці?
5. За допомогою якої функції обчислюється ранг матриці?
6. Як обчислити довжину вектора?
7. Як розв'язати систему лінійних алгебраїчних рівнянь виду $Ax=b$?
8. Як підрахувати суму всіх елементів матриці, окремого рядка або стовпчика?
9. Як підрахувати добуток всіх елементів матриці, окремого рядка або стовпчика?

10. Як обчислити визначений інтеграл?

11. Як обчислити похідну для певної точки?

12. За допомогою якої функції розв'язують диференційне рівняння першого порядку? Які аргументи вона має? Які є особливості використання цих аргументів?

13. Як розв'язати систему лінійних алгебраїчних рівнянь?

14. Як розв'язати систему диференційних рівнянь?

Після виконання лабораторної роботи студент повинен ***уміти***:

1. Визначити кількість рядків, стовпчиків, елементів матриці.

2. Визначити максимальний або мінімальний елемент вектора або матриці.

3. Визначити максимальний або мінімальний елемент конкретного рядка або стовпчика матриці.

4. Обчислити детермінант матриці.

5. Обчислити ранг матриці.

6. Обчислити довжину вектора.

7. Розв'язати систему лінійних алгебраїчних рівнянь виду $Ax = b$.

8. Обчислити суму всіх елементів матриці, для окремого рядка або стовпчика.

9. Обчислити добуток всіх елементів матриці, для окремого рядка або стовпчика.

10. Підрахувати суму або добуток значень елементів вектора.

11. Обчислити визначений інтеграл.

12. Обчислити похідну для певної точки.

13. Розв'язати систему лінійних алгебраїчних рівнянь.

14. Розв'язати диференційне рівняння першого порядку.

15. Розв'язати систему диференційних рівнянь.

Література

1. Фетісов В. С. Математична система Scilab: навч. посіб. Ніжин: Видавництво НДУ ім. М. Гоголя, 2011. 45 с.

ДЛЯ НОТАТОК

Навчальне видання

В. С. Фетісов

МАТЕМАТИЧНА СИСТЕМА SCILAB

Навчально-методичний посібник

2-ге видання,
перероблене і доповнене

Технічний редактор – І. П. Борис
Верстка, макетування – О. В. Борщ

Друкується за авторським редагуванням.

Підписано до друку 25.11.22 р. Формат 60x84/16 Папір офсетний
Гарнітура Times Обл.-вид. арк. 1,91 Електронне вид.
Замовлення № Ум. друк. арк. 4,88

Ніжинський державний університет
імені Миколи Гоголя.
м. Ніжин, вул. Воздвиженська, 3[А]
(04631) 7–19–72
E-mail: vidavn_ndu@ukr.net
www.ndu.edu.ua

Свідоцтво суб'єкта видавничої справи
ДК № 2137 від 29.03.05 р.