

Львівський національний університет імені Івана Франка

Злобін Г.Г.

## СИСТЕМИ КОМП'ЮТЕРНОЇ МАТЕМАТИКИ В НАУКОВИХ ОБЧИСЛЕННЯХ

навчальний посібник

Львів-2013

УДК 004.7(075.8)  
ББК 32.973Я73  
З 14

Рекомендовано до використання у навчальному процесі кафедрою радіофізики  
протокол № ????? від ?? лютого 2012 р.

У посібнику розглянуті вільні системи комп'ютерної математики — Scilab і Maxima.  
Для студентів старших курсів університетів.

Текст посібника змакетовано і записано у формат pdf в офісному пакеті  
OpenOfficeOrg.3.2.1 Ukr в операційній системі Linux.

Рецензенти:

Права автора застережено.

Вступ до комп'ютерної математики.	5
Maple (6). Mathcad (6). Mathematica (7). MATLAB (7). MuPAD (8). Derive (9). Maxima (10). Axiom (11). Scilab (11). GNU Octave (12). Sage (12).	
1. Вільні системи комп'ютерної математики.	13
1.1. СКМ Scilab	13
1.1.1. Початок роботи з Scilab	13
1.1.2. Арифметичні вирази в Scilab	14
1.1.3. Змінні в Scilab	15
1.1.4. Вбудовані функції та функції користувача в Scilab	16
1.1.5. Чисельне диференціювання та інтегрування в Scilab	18
1.1.6. Масиви та матриці в Scilab. Розв'язання задач лінійної алгебри	19
1.1.7. Дії з поліномами в Scilab	23
1.1.8. Побудова графіків в Scilab	25
Функція plot (25). Функція plot2d (29). Побудова сходишкових графіків (33). Побудова графіків в полярній системі координат (33).	
1.1.9. Форматування графіків через меню графічного вікна	34
Форматування об'єкта Figure (Графічне вікно) (35). Форматування об'єкта Axes (Осі графіка) (36). Форматування об'єкта Polyline (Лінія графіка) (40).	
1.1.10. Побудова тривимірних графіків в Scilab	42
Функції plot3d і plot3d1 (42). Функції meshgrid, surf та mesh (43). Побудова параметричних кривих в тривимірному просторі (44).	
1.1.11. Розв'язання нелінійних рівнянь	45
1.1.12. Згладжування експериментальних даних	46
1.1.13. Інтерполяція табличних даних	47
1.1.14. Чисельне інтегрування диферівнянь	49
1.1.15. Програмування в Scilab	52
Функції вводу-виводу в Scilab ((53). Оператор присвоювання (54). Умовний оператор (54). Оператор альтернативного вибору (55). Цикли в Scilab (56). Робота з файлами в Scilab (56).	
1.1.16. Створення графічних додатків в Scilab	59
Динамічне створення інтерфейсних елементів графічного вікна (60). Командна кнопка (PushButton) (61). Текстові поля (Мітки) (62). Елемент "Радіокнопка" (63). Елемент "Вікно редагування" (65). Інтерфейсний елемент "Списки рядків символів" (66).	
2. Система комп'ютерної алгебри Maxima	68
2.1. Інтерфейси Maxima	68
2.2. Початок роботи у Maxima	69
2.3. Вирази та елементарні функції	70
2.4. Поліноми та алгебраїчні перетворення	75
2.5. Розв'язання рівнянь	76
2.6. Операції математичного аналізу	79
2.7. Побудова графіків і поверхонь	82
2.8. Матричні обчислення	85
2.9. Списки та масиви	89
2.10. Розв'язування диферівнянь в Maxima	92
2.11. Основи програмування в системі Maxima	98
2.12. Транслятор і компілятор в MAXIMA	101
Рекомендована література	102
Додаток 1. Функції Scilab для роботи з матрицями	109
Додаток 2. Перелік основних функцій системи Maxima	113

Додаток 3. Перелік основних пакетів розширення Maxima	119
Додаток 4. Список основних математичних констант, доступних в Maxima	120
Додаток 5. Список основних математичних функцій, доступних в Maxima	121

Системами комп'ютерної алгебри (СКА, в іноземній літературі – CAS – computer algebra system) називають системи, які працюють з обробкою математичних виразів, значно зменшуючи час, який потрібно затратити на виконання громіздких, але тривіальних перетворень та розрахунків.

Система комп'ютерної математики (СКМ, в іноземній літературі — CMS — computer mathematics system) — це більш узагальнене поняття, яке об'єднує в собі можливості комп'ютерної алгебри, числових методів, математичної статистики, візуалізації даних та багато інших складових.

За допомогою СКА користувач легко може розкривати дужки в довгих і громіздких виразах, робити різні маніпуляції з поліномами, розв'язувати рівняння і системи рівнянь (як звичайних, так і диференціальних), аналітично отримувати границі, похідні, невизначені/визначені інтеграли (всіх типів), різні перетворення функцій (Фур'є, Лапласа,  $z$ -перетворення). Деякі системи навіть відображають хід виведення результату (проміжні дії, наприклад *Axiom*). СКА також часто мають можливість виконувати деякі чисельні розрахунки, як розрахунки з машинною точністю, так і з довільною кількістю цифр, будувати графіки, діаграми, гістограми, векторні поля, 3D- графіки та об'єкти.

Зазвичай область застосування цих систем є дуже широкою – від вузькоспеціалізованих досліджень до загального ознайомлення у школах та вищих закладах освіти, тому більшість з цих систем мають досить зручний та інтуїтивний інтерфейс.

Існують спеціалізовані СКА (*GAP*, *Magnus*, *Singular* і ін.) та загального призначення (*Maple*, *Mathematica*, *Maxima*, *Axiom* і ін.). Слід також зауважити, що існують бібліотеки (*GiNaC*, *Math++*), які підтримують символні обчислення в існуючих мовах програмування (*C++*, *Java*, *Lisp* та ін.).

Перші розробки в області СКА почались на початку 60-х років 20 століття і пов'язані з двома областями науки — теоретичною фізикою та штучним інтелектом. Тоді виникли такі системи як *Reduce*, *muMATH*, *Derive*, *Macsyma*. Особливе місце серед цих розробок займає мова Аналітик, яка була реалізована апаратним чином на ЕОМ Мир-2 (1969 р.). Західні розробки СКА орієнтувались на програмну реалізацію аналітичних обчислень. На сьогоднішній день існує дуже велика кількість різних СКА/СКМ. Основні і найбільш популярні з них це:

- Maple;
- Mathematica;
- Maxima;
- MuPAD;
- Sage;
- Derive.

До основних задач СКА можна віднести:

- спрощення до найменшого можливого виразу або до стандартної форми, включно з автоматичним спрощенням;

- заміну символів чи чисел для заданих виразів;

- зміну форми виразу - розкриття дужок та степенів, частковий та повний розклад на множники, розклад на прості дроби, врахування обмежуючих умов, перетворення з тригонометричної у експоненціальну форму та навпаки, перетворення логарифмічних виразів тощо;

- диференціювання та інтегрування ;

- аналітичну глобальну оптимізацію;

- розв'язування лінійних та деяких типів нелінійних рівнянь та систем рівнянь на різних областях визначення;

- розв'язування деяких типів інтегральних та диференціальних рівнянь;

- визначення граничних значень виразів;

інтегральні перетворення;  
роботу з послідовностями;  
матричні операції;  
статистичне числення;  
доведення теорем.

## Maple

Maple — одна з найпотужніших і найбільш популярних СКА. Ця система була створена групою вчених, які займаються символьними обчисленнями (The Symbolic Group), організованою Кейтом Геддом (Keith Geddes) і Гастоном Гоне (Gaston Gonnet) в 1980 р. в університеті Waterloo, Канада. Спочатку вона була реалізована на великих ЕОМ і пройшла довгий шлях апробації, увібравши у своє ядро велику кількість бібліотек математичних функцій і правил їхніх перетворень, вироблених математикою і математиками за століття розвитку.

Системі Maple присвячені сотні книг. Досить повний список книг можна знайти на сайті розробника цієї системи - компанії Waterloo Maple Software ([www.maplesoft.com](http://www.maplesoft.com)). Також Maple має хорошу систему довідкової інформації, яка встановлюється разом з програмою. В цій документації є помічник системи для початківців, детальний опис всіх бібліотек та функцій, велика кількість прикладів.

СКА Maple належить до широко використовуваних систем завдяки великій кількості бібліотек, зручності інтерфейсу, та значним часом лідерства цієї системи на ринку.

Maple Software продає як студентську (99 у.о.), так і професійну (1995 у.о.) версії Maple.

## Mathcad

Mathcad — система комп'ютерної алгебри з класу систем автоматизованого проектування, орієнтована на підготовку інтерактивних документів з обчисленнями і візуальним супроводом, відрізняється легкістю використання і застосування для колективної роботи.

Mathcad був задуманий і спочатку написаний Алленом Раздовим з Массачусетського технологічного інституту (MIT), співзасновником компанії Mathsoft Inc., яка з 2006 р. є частиною корпорації PTC (Parametric Technology Corporation).

Mathcad має простий і інтуїтивний для використання інтерфейс користувача. Для введення формул і даних можна використовувати як клавіатуру, так і спеціальні панелі інструментів.

Деякі з математичних можливостей Mathcad (версії до 13.1 включно) засновані на підмножині системи комп'ютерної алгебри Maple (MKM, Maple Kernel Mathsoft). Остання версія — 14 — використовує символьне ядро MuPAD.

Робота здійснюється в межах робочого аркуша, на якому рівняння і вирази відображаються графічно, на противагу текстовому запису в мовах програмування. Під час створення документів-програм використовується принцип WYSIWYG (What You See Is What You Get — «що бачиш, те й отримуєш»).

Незважаючи на те, що ця програма здебільшого орієнтована на користувачів-непрограмістів, Mathcad також використовують в складніших проектах, для візуалізації результатів математичного моделювання шляхом використання поширених обчислень і традиційних мов програмування.

Mathcad доволі зручно використовувати для навчання, обчислень і інженерних розрахунків. Відкрита архітектура застосунку у поєднанні з підтримкою технологій .NET і XML дають змогу інтегрувати Mathcad практично в будь-які ІТ-структури і інженерні застосунки. Є можливість створення електронних книг (e-Book).

Основні можливості системи: числові розрахунки, аналітичні перетворення, лінійна алгебра, візуалізація даних.

Залежно від комплектації вартість ліцензії Mathcad 15.0 складає від 3240 грн. до 191600 грн.

## Mathematica

У 80-і роки можливостями символьної математики захопився після захисту докторської дисертації С. Вольфрам (Stephen Wolfram) зі США. Його інтереси були настільки серйозні, що він заснував фірму Wolfram Research, Inc., яка розпочала проект створення престижної математичної системи Mathematica. Версія Mathematica 1.0 з'явилася в 1988 р. Революційною розробкою фірми стала версія 2.0 системи Mathematica 2, що з'явилася в 1991 р. і успішно дожила до наших днів. Мета нового проекту була досить амбіційною - розробка потужного й універсального ядра системи, здатного працювати на різних обчислювальних платформах, створення багатофункціональної мови програмування, орієнтованої на математичну обробку даних, підготовка сучасного користувацького інтерфейсу й великого набору прикладних пакетів і розширень системи, потужної мови програмування математичних перетворень й обчислень. Система отримала властивості адаптації й навчання нових математичних законів і закономірностей.

Mathematica має сучасний користувацький інтерфейс, який дає змогу створювати документи у формі Notebooks («записних книжок»). Вони поєднують вихідні дані, опис алгоритмів розв'язування задач, програм і результатів в найрізноманітнішій формі (математичні формули, числа, вектори, матриці, графіки). Із самого початку велика увага приділялася графіці, у тому числі динамічній, а також можливостям мультимедіа - відтворенню динамічних зображень і синтезу звуків з підтримкою цифрового звукового каналу. Набір функцій графіки й опцій, що змінюють їхні дії, і директив досить повний. Графіка завжди була козирною картою систем Mathematica і забезпечувала їм лідерство серед систем комп'ютерної математики.

Основні можливості системи: аналітичні перетворення, числові розрахунки, теорія чисел, лінійна алгебра, графіка і звук, розробка програм.

Залежно від комплектації вартість ліцензії Mathematica складає від 1345 грн. до 15881 грн.

## MATLAB

СКМ MATLAB (від англ. «Matrix Laboratory») — це пакет прикладних програм для розв'язання задач технічних обчислень та однойменна мова програмування, яку можна використовувати у цьому пакеті. MATLAB працює у більшості сучасних операційних систем включно з Linux, Mac OS, Solaris та Microsoft Windows.

MATLAB як мова програмування був розроблений Клівом Моулером (англ. Cleve Moler) наприкінці 1970-х років, коли він був деканом факультету комп'ютерних наук в університеті Нью-Мексіко. MATLAB розроблявся з метою дати студентам факультету можливість використання програмних бібліотек Linpack и EISPACK без необхідності вивчення мови Фортран. Невдовзі нова мова програмування поширилась серед інших університетів і була з великим інтересом зустрінута вченими, які працювали в області прикладної математики. Ще й зараз в Інтернеті можна знайти версію 1982 року, написану мовою Фортран, яка поширюється з відкритим кодом. Інженер Джон Літл (англ. John N. (Jack) Little) познайомився з цією мовою під час візиту Кліва Моулера в Стенфордський університет у 1983 р. Зрозумівши, що нова мова програмування має великий комерційний потенціал, він об'єднався з Клівом Моулером і Стівом Бангертом (англ. Steve Bangert). Спільними зусиллями вони переписали MATLAB мовою C і заснували у 1984 р. компанію The MathWorks для наступного розвитку. Ці переписані мовою C бібліотеки довгий час

були відомі як JACSPAC. Спочатку MATLAB призначався для проектування систем керування (основна спеціальність Джона Літтла), але він швидко завоював популярність в інших наукових і інженерних областях. Він також широко використовувався в освіті, зокрема, для викладання лінійної алгебри і числових методів.

Мова MATLAB є високорівневою інтерпретованою мовою програмування, яка містить базовані на матрицях структури даних, широкий спектр функцій, інтегроване середовище розробки, об'єктно-орієнтовані можливості та інтерфейси до програм, що написані іншими мовами програмування. Програми, написані мовою MATLAB, є двох типів — функції і сценарії. Функції мають вхідні і вихідні аргументи, а також власний робочий простір для зберігання проміжних результатів обчислень і змінних. Сценарії ж використовують спільний робочий простір. Як сценарії, так і функції не компілюються в машинний код а зберігаються як текстові файли. Є також можливість зберігати так звані pre-parsed програми — функції і сценарії, перетворені до вигляду, який зручний для машинного виконання. Загалом такі програми виконуються швидше звичайних функцій і сценаріїв.

Основною особливістю мови MATLAB є її широкі можливості у роботі з матрицями, які розробники мови висловили лозунгом «думай векторно» (англ. Think vectorized).

MATLAB надає користувачу велику кількість (кілька сотень) функцій для аналізу даних, які охоплюють практично всі області математики, зокрема:

- матриці і лінійна алгебра;
- поліноми і інтерполяція;
- математична статистика і аналіз даних;
- обробка даних;
- диференціальні рівняння;
- розріджені матриці;
- цілочисельна арифметика;

MATLAB надає зручні засоби для розробки алгоритмів включно з використанням концепцій об'єктно-орієнтованого програмування.

Пакет MATLAB містить різні інтерфейси для отримання доступу до зовнішніх підпрограм, написаних іншими мовами програмування, даними, клієнтів і серверів, що спілкуються через технології Component Object Model або Dynamic Data Exchange, а також периферійних пристроїв, які взаємодіють безпосередньо з MATLAB. Більшість цих можливостей відомі під назвою MATLAB API.

Для MATLAB є можливість створювати спеціальні набори інструментів (англ. toolbox), які розширюють його функціональність. Ці набори інструментів можна використовувати в багатьох областях: цифрова обробка сигналів, зображень та даних, фінансові розрахунки, аналіз і синтез географічних карт, ввід і аналіз експериментальних даних, візуалізація даних, засоби розробки, бази даних, наукові і математичні пакети, аналітичні обчислення.

Залежно від комплектації вартість ліцензії MATLAB 15.0 складає від 29 448 грн. до 117 792 грн.

## MuPAD

СКМ MuPAD - спільний проект SciFace Software і дослідницької групи університету міста Падерборн (Німеччина). MuPAD з'явилася на ринку наукового ПЗ зовсім недавно - в 1997 р., коли багато знаменитих продуктів типу Maple або MathCad уже були визнаними торговими марками й мали довгу історію. Однак проект виявився цілком життєздатним, і новачок досить швидко став завойовувати власне місце під сонцем. Версію 4.0, сьогодні без усяких знижок можна поставити в перший ряд ведучих СКМ, представлених на світовому ринку. MuPAD містить:

- засоби для аналітичних перетворень;
- пакети програм для лінійної алгебри, диференціальних рівнянь, теорії чисел, статистики та функціонального програмування;



інтерактивну графічну систему, яка підтримує анімацію і прозорі області в 3D;  
 пакети програм числового аналізу;  
 мову програмування, яка підтримує об'єктно-орієнтоване та функціональне програмування.

Синтаксис MuPAD був змодельований на Паскалі, і був схожий на той, який використовується в алгебрі комп'ютерної системи Maple. Важлива відмінність між ними полягає в тому MuPAD забезпечує підтримку об'єктно-орієнтованого програмування. Це означає, що кожен об'єкт "несе із собою" методи, які дозволено використовувати на ньому.

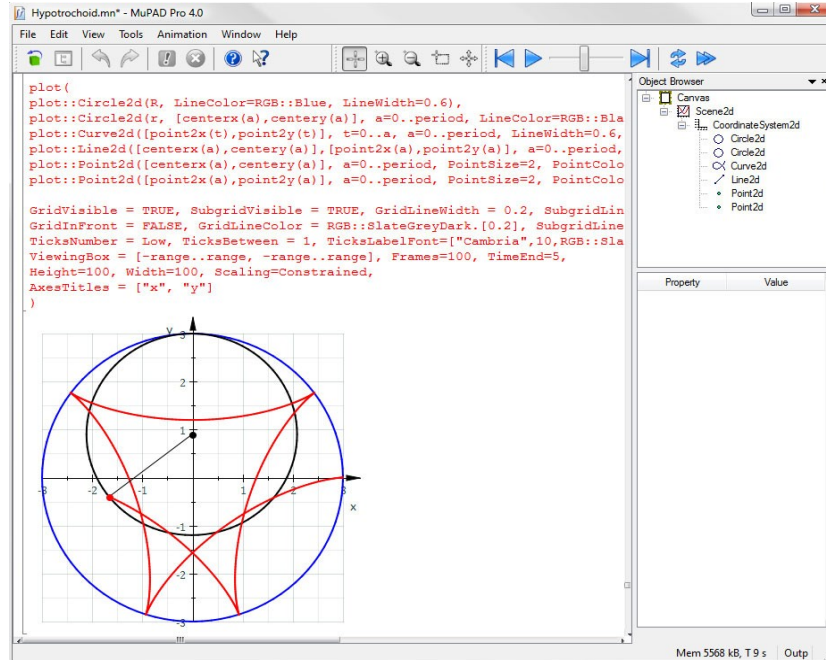


Рис. 1.

Зупинимося коротенько на внутрішній будові програми. Як і багато інших продуктів цього класу, MuPAD складається з ядра, яке виконує всі обчислення, і оболонки, що містить робочу область користувача, довідкову систему й пакети розширень, в яких зібрані функції, що належать до різних областей математики. Така двокомпонентна структура дає змогу економно витрачати системні ресурси, а також створювати на основі MuPAD розподілені обчислювальні системи. Підтримуються чотири найважливіші комп'ютерні платформи - Windows, Macintosh, Linux і Solaris, однак потрібно відзначити, що нові версії для різних ОС виходять не завжди синхронно.

MuPAD має досить традиційний для систем цього типу інтерфейс (рис. 1): ви вводите команди й формули, ядро повертає відповіді й результати - таким чином, основний документ нагадує робочий зошит, "Notebook" у термінах MuPAD. Однак крім комірок вводу/виводу він також може містити відформатований текст, графічні й OLE-об'єкти, файли можна зберігати не тільки у власному форматі (MNB), але і експортувати в HTML, RTF, DOC.

Залежно від комплектації вартість ліцензії MuPAD складає від 67 у.о. до 767 у.о.

## Derive

Derive – компактна СКА, яка випускається компанією Soft Warehouse, Inc./ Texas Instruments Inc. з 1988 року під девізом «A Mathematical Assistant for Your Personal Computer». Основні характерні особливості Derive:

- невеликий об'єм дистрибутиву;
- надійність і висока потужність розрахунків, достатня для розв'язування багатьох видів диференціальних рівнянь в аналітично;
- раціональність, від користувачів вимагається тільки заповнення полів вводу.

Derive прекрасно працює навіть на достатньо слабких ПЕОМ. Завдяки невеликому розміру Derive інтегровано навіть в деякі калькулятори (фірм Texas Instruments та Hewlett Packard).

## Maxima

Maxima - система комп'ютерної алгебри, реалізована мовою Lisp. Maxima має великий набір засобів для виконання аналітичних обчислень, чисельних обчислень і побудови графіків. За кількістю можливостей система близька до таких комерційних систем як Maple й Mathematica. У той же час вона має найвищий ступінь можливості перенесення на різні платформи. Це єдина з існуючих нині систем аналітичних обчислень, що може працювати на всіх сучасних операційних системах та ЕОМ, починаючи від самих потужних, закінчуючи КПК (кишенькові ПК).

Maxima (<http://maxima.sf.net>) походить від системи «Macysma», що розроблялася в МІТ (Massachusetts Institute of Technology) з 1968 р. по 1982 р. Варіант продукту (відомий як DOE Macysma) супроводжувався професором Вільямом Шелтером у Техаському університеті з 1982 р. до смерті в 2001 р. В 1998 р. Шелтер одержав від Департаменту енергії дозвіл опублікувати вихідний код DOE Macysma під ліцензією GPL, і в 2000 р. він створив проект «Maxima» на SourceForge ([www.sourceforge.net](http://www.sourceforge.net)) для підтримки й розвитку «DOE Macysma», перейменованого в Maxima. Таким чином, Macysma фактично стала родоначальником всього напрямку програм символічної математики. Ліцензування в 70-ті рр. програмних кодів Macysma призвело до створення інших систем комп'ютерної математики – Maple фірми Waterloo Maple Inc. та Mathematica фірми Wolfram Research. Спільність цих програмних продуктів проявляється як у схожому синтаксисі (табл. 1), так й у спільних алгоритмах.

	Maxima	Maple	Mathematica
границя	<code>limit(x-7,x,3)</code>	<code>limit(x-7,x=3)</code>	<code>Limit[x-7,x-&gt;3]</code>
розгортка виразу	<code>expand((a+b)^3)</code>	<code>expand((a+b)^3)</code>	<code>Expand[(a+b)^3]</code>
розклад на множники	<code>factor(%); ezgcd(num, denom);</code>	<code>factor(%); normal(%);</code>	<code>Factor[%]</code>
розв'язування рівнянь	<code>solve(a*x^2=4,x)</code>	<code>solve(a*x^2=4,x);</code>	<code>Solve[a x^2==4,x]</code>
3D-графіка	<code>plot3d(sin(x*y),[x,-2,2], [y,-1,1]);</code>	<code>plot3d(sin(x*y), x=- 2..2,y=-1..1);</code>	<code>Plot3D[Sin[x y],{x,- 2,2},{y,-1,1}]</code>
квадратний корінь	<code>sqrt(3)</code>	<code>sqrt(3)</code>	<code>Sqrt[3]</code>
визначення функції	<code>f(x):=x^2+1; або define(f(x),x^2+1)</code>	<code>f:=x-&gt;x^2+1; або f:=unapply(x^2+1,x)</code>	<code>f[x_]=x^2+1 або f=Function[x,x^2+1]</code>

Табл. 1. Команди Maxima, Maple, Mathematica (фрагмент)

Основні можливості Maxima:

операції з поліномами;

виконання обчислень з елементарними функціями, в тому числі з логарифмами, експонентами та тригонометричними функціями;

виконання обчислень із спеціальними функціями, в тому числі з еліптичними функціями та інтегралами;

обчислення границь і похідних;

аналітичне обчислення означених та неозначених інтегралів;

розв'язування інтегральних рівнянь;  
розв'язування алгебраїчних рівнянь і систем;  
виконання операцій над степеневими рядами та рядами Фур'є;  
виконання операцій над матрицями та списками, велика бібліотека функцій для розв'язання задач лінійної алгебри;  
операції з тензорами;  
обчислення з теорії чисел, теорії груп та абстрактної алгебри.  
Для розширення можливостей системи можна використовувати додаткові пакети (див. Додаток 3).

### Axiom

Axiom – СКА загального призначення, яка суттєво вирізняється з поміж інших СКА строгою типізацією даних, наявністю в ній великої кількості структур даних як простих і загальновідомих: стрічки, списки, вектори, множини, кортежі, словники, математичних структур даних: матриці, потоки, послідовності, многочлени, так і високо абстрактних таких як ідеали і алгебри Клейна. Ще однією характерною особливістю Axiom є компіляція і оптимізація вхідних даних перед їх виконанням, що значно пришвидшує час виконання команд.

Історія система Axiom починається з середини 70-х років минулого століття коли дослідники з IBM створили СКА Scratchpad для машин IBM 3081, і пізніше IBM 3090. В 2001 р. Axiom закінчив своє існування як комерційний проект, через рік Axiom був випущений під модифікованою BSD-ліцензією. 27 травня 2003 р. Axiom став безплатним проектом з відкритим вихідним кодом.

Axiom не має власної графічної оболонки, тому для роботи з ним треба або працювати з текстовим вікном, або використовувати зовнішні графічні оболонки, такі як TeXmacs. Також слід зауважити, що на відміну від інших СКА Axiom є досить складною і менш інтуїтивно зрозумілою, проте дуже потужною і ефективною.

### Scilab

Scilab — пакет наукових програм для чисельних обчислень, що надає потужне відкрите оточення для інженерних і наукових розрахунків. Scilab містить сотні математичних функцій з можливістю додавання нових, написаних різними мовами (C, C++, Fortran тощо). Так само є різноманітні структури даних (списки, поліноми, раціональні функції, лінійні системи), інтерпретатор і мова високого рівня.

Scilab був спроектований так, щоб бути відкритою системою, де користувачі можуть додавати свої типи даних і операції над цими даними шляхом перевантаження.

У системі доступно безліч інструментів:

- 2d і 3d-графіки, анімація;
- лінійна алгебра, розріджені матриці (sparse matrices);
- поліноміальні та раціональні функції;
- інтерполяція, апроксимація;
- чисельне інтегрування диференціальних рівнянь;
- Scicos: гібрид системи моделювання динамічних систем і симуляції;
- оптимізація;
- обробка сигналів;
- паралельна робота;
- статистика;
- інтерфейс до Fortran, Tcl/Tk, C, C++, Java, LabVIEW.

Scilab має схожу з MATLAB мову програмування, є утиліта, що дає змогу конвертувати документи Matlab → Scilab.

Scilab дає змогу працювати з елементарними і великим числом спеціальних функцій (Бесселя, Неймана, інтегральні функції), має могутні засоби для роботи з матрицями, поліномами, проводити чисельні обчислення (наприклад чисельне інтегрування) і розв'язання задач лінійної алгебри, оптимізації і симуляції, могутні статистичні функції, а також засіб для побудови і роботи з графіками. Для чисельних розрахунків використовуються бібліотеки Lapack, LINPACK, ODEPACK, Atlas та інші.

До складу пакету також входить інструмент Scicos для редагування блокових діаграм і симуляції (аналог simulink в пакеті — MATLAB). Є можливість спільної роботи Scilab з програмою LabVIEW.

## GNU Octave

GNU Octave — мова високого рівня, призначена для виконання математичних обчислень. Вона є зручним командним інтерфейсом для розв'язування лінійних і нелінійних математичних задач, а також проведення інших арифметичних експериментів, використовуючи мову, у більшості випадків сумісну з MATLAB. Крім того, Octave можна використовувати для пакетної обробки. Мова Octave оперує арифметикою дійсних і комплексних скалярів і матриць, має розширення для розв'язування лінійних алгебраїчних рівнянь, знаходження коренів систем нелінійних алгебраїчних рівнянь, роботи з поліномами, розв'язування різних диференціальних рівнянь, інтегрування систем диференціальних і алгебро-диференціальних рівнянь першого порядку, інтегрування функцій на скінченних і нескінченних інтервалах. Цей список можна легко розширити, використовуючи мову Octave (або використовуючи динамічно завантажувані модулі, створені мовами C, C++, Фортран тощо).

## Sage

Sage (анг. мудрець) - система комп'ютерної алгебри, яка покриває багато областей математики включно з алгеброю, комбінаторикою, обчислювальною математикою і матаналізом. Sage може виконуватися як локально, так і віддалено.

Перша версія Sage була випущена 24 лютого 2005 року як вільне програмне забезпечення з ліцензією GNU GPL. Первісною метою проекту було «створення відкритого програмного забезпечення альтернативного системам Magma, Maple, Mathematica, і MATLAB. Розробником Sage є Вільям Стейн - математик Університету Вашингтона.

Можливості Sage:

інтерфейс notebook для перегляду і повторного використання введених команд та отриманих результатів, включно з графіками і текстовими анотаціями, доступний з більшості сучасних Веб-переглядачів. Доступне захищене з'єднання через протокол HTTPS, коли конфіденційність має значення;

інтерфейс вводу на основі командного рядка з використанням мови Ipython;

підтримка паралельних обчислень з використанням як багатоядерних процесорів, так і багатопроцесорних систем і систем розподілених обчислень;

матаналіз, реалізований на основі систем Maxima і SymPy;

лінійна алгебра, реалізована на основі систем GSL, SciPy і NumPy;

бібліотеки елементарних та спеціальних математичних функцій;

плоскі і тривимірні графіки для функцій і даних;

засоби для роботи з матрицями і масивами даних з підтримкою розріджених масивів;

різні статистичні бібліотеки функцій, які використовують функціональність R і SciPy;

набір інструментів для додавання власного користувацького інтерфейсу до обчислень і додатків;

засоби для обробки зображень з використанням pylab і Python;

засоби візуалізації та аналізу теорії графів;

процедури для імпорту та експорту різних форматів даних: зображень, відео, аудіо, САПР, ГІС, документів і медичних форматів;

підтримка комплексних чисел, символьних обчислень і обчислень з довільною точністю;

підготовка науково-технічної документації з використанням редактора формул і можливістю вбудовування Sage в документацію формату LaTeX;

мережеві інструменти для з'єднання з базами даних SQL, підтримка мережевих протоколів, включно з HTTP, NNTP, IMAP, SSH, IRC, FTP;

програмні інтерфейси для роботи з системами Mathematica, Magma, і Maple.

## 1. Вільні системи комп'ютерної математики

### 1.1. СКМ Scilab

Scilab - це система комп'ютерної математики, яка призначена для виконання інженерних і наукових обчислень, а саме:

розв'язання нелінійних рівнянь і систем;

розв'язання задач лінійної алгебри;

розв'язання задач оптимізації;

диференціювання та інтегрування;

обробка експериментальних даних (інтерполяція та апроксимація, метод найменших квадратів);

розв'язання звичайних диференціальних рівнянь і систем.

Крім того, Scilab надає широкі можливості із створення і редагування різних видів графіків і поверхонь.

Незважаючи на те, що система Scilab 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 потужну власну мову програмування високого рівня, що говорить про можливість розв'язання нових завдань. Систему можна встановити в операційних системах:

Microsoft Windows;

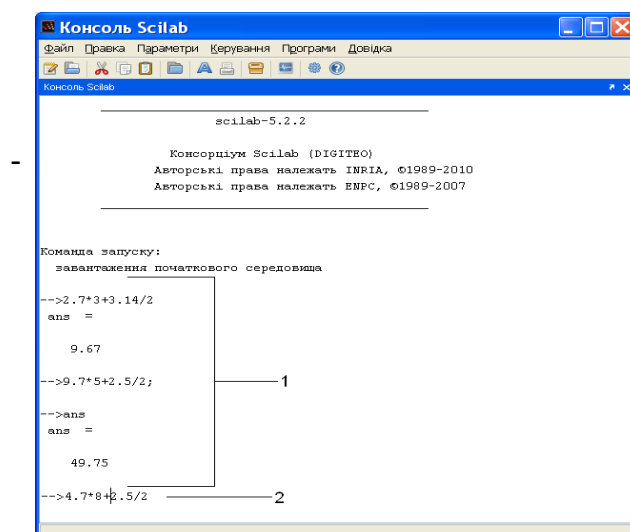
Linux;

Mac OS X.

Інсталяційні пакети можна завантажити з Веб-сторінки [www.scilab.org/](http://www.scilab.org/)

#### 1.1.1. Початок роботи з Scilab

Після запуску Scilab на екрані з'явиться основне вікно програми. 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 зі знаком запрошення зазвичай називають командним рядком. Введення команд в Scilab здійснюється з клавіатури. натискання клавіші **Enter** змушує систему виконати команду і вивести результат (рис. 1.1).

Рис. 1.1. Вікно Scilab

1- зона перегляду, 2- зона редагування

Очевидно, що всі виконувані команди не можуть одночасно знаходитися в полі зору користувача. Переглянути ту інформацію, яка покинула видиму частину вікна, можна, якщо скористатися стандартними засобами перегляду, а саме: лініями прокрутки або клавішами переміщення курсора **Page Up**, **Page Down**.

Клавіші  $\uparrow$  і  $\downarrow$  дають змогу переміщуватись у буфері вже введених команд. Так, якщо в порожньому командному рядку натиснути клавішу  $\uparrow$ , то з'явиться остання введена команда, повторне натискання викличе передостанню і так далі. Клавіша  $\downarrow$  виводить команди в зворотному порядку. Таким чином, можна сказати, що вся інформація в робочій області знаходиться або в зоні перегляду або в зоні редагування.

Слід наголосити, що в зоні перегляду не можна нічого виправити або ввести. Єдина допустима операція, крім перегляду, це виділення інформації за допомогою миші та копіювання її в буфер обміну, наприклад, для наступного копіювання в командний рядок.

Зона редагування - це фактично командний рядок. У ній діють елементарні методи редагування:

$\rightarrow$  - переміщення курсора вправо на один символ;

$\leftarrow$  - переміщення курсора вліво на один символ;

**Home** - переміщення курсора на початок рядка;

**End** - переміщення курсора в кінець рядка;

**Delete** - 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. За відсутності знака «;» результат дії команди відразу ж виводиться в робочу область

```
-->2.7*3+3.14/2
ans =
  9.67
-->2.7*3+3.14/2;
-->
```

Поточний документ, який відображає роботу користувача з системою Scilab з рядками введення, виведення та повідомлення про помилки, називають сесією. Значення всіх змінних, які обчислені у поточній сесії, зберігаються в спеціально зарезервованій області пам'яті (робочому просторі системи). За потреби визначення всіх змінних і функцій, які входять в поточну сесію, можна зберегти у файлі (саму сесію зберегти не можна).

### 1.1.2. Арифметичні вирази в Scilab

Для виконання найпростіших арифметичних операцій в Scilab застосовують наступні оператори: + додавання, - віднімання, \* множення, / ділення зліва направо, \ ділення справа наліво, ^ піднесення до степеня. Обчислити значення арифметичного виразу можна, якщо ввести його в командний рядок і натиснути клавішу Enter. У робочій області з'явиться результат, наприклад:

```
--> 2.35*(1.8-0.25)+1.34^2/3.12
ans =
  4.2180
```

Якщо вираз, який вводиться, є довшим, ніж рядок символів на дисплеї, то перед натисканням клавіші **Enter** слід набрати три або більше крапок (це означатиме продовження командного рядка), наприклад:

```
--> 1+2+3+4+5+6....
```

```

7+8+9+10+....
+11+12+13+14+15
ans =
    120

```

### 1.1.3. Змінні в Scilab

В робочій області **Scilab** можна визначати змінні для подальшого використання їх у виразах. Будь-яка змінна до використання у формулах і виразах повинна мати значення. Для цього використовують оператор присвоєння

ім'я\_змінної = вираз

Ім'я змінної не повинно 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 різних змінних. Вираз у правій частині оператора присвоєння може бути числом, арифметичним виразом, рядком символів (рядок символів має виділятися одинарними лапками). Розглянемо приклад:

```

--> a=2.3
a =
2.3000
--> b=-34.7
b =
-34.7000
--> //Присвоєння значень змінним x та y, обчислення значення змінної z
--> x=1;y=2; z=(x+y)-a/b
z =
3.0663
--> c+3/2
!--error 4
Невизначена змінна: c
--> //Повідомлення про помилку – змінна c не визначена
--> //-----
--> //Задання символічної змінної
--> c='a'
c =
a
--> // Задання рядкової змінної
--> h='факультет електроніки'
h = факультет електроніки

```

Після виконання оператора присвоєння значення змінної та її ім'я зберігаються в робочій області Scilab. Якщо потреба у зберіганні значення змінної відпала, видалити змінну з робочої області можна командою `clear ім'я_змінної`. Команда `clear` видаляє з робочої області усі змінні, які були визначені під час сесії Scilab. Розглянемо приклад:

```

--> // Присвоєння значень змінним x та y
--> x=-7; y=-9;
--> //Видалення змінної x
--> clear x
--> x
!--error 4
Невизначена змінна: x
--> //Повідомлення про помилку – змінна x не визначена
--> y
y =

```

```

-9
-->// Присвоєння значень змінним a та b
-->a=1;b=2;
-->//Видалення змінних a та b
-->clear;
-->a
!--error 4
Невизначена змінна: a
-->b
!--error 4
Невизначена змінна: b

```

Окрім змінних користувача в системі **Scilab** є наперед визначені системні змінні:

- ans — у цій змінній зберігається результат останньої операції;
- %i — уявна одиниця;
- %pi — число  $\pi = 3.141592653589793$ ;
- %e — число  $e = 2.7182818$ ;
- %inf — машинний символ нескінченності( $\infty$ );
- %NaN — невизначений результат ( $0/0, \infty/\infty$  тощо);
- %eps — умовний нуль %eps=2.220E-16.

#### 1.1.4. Вбудовані функції та функції користувача в **Scilab**

В арифметичних виразах можна використовувати елементарні функції, які вбудовані в систему **Scilab**. Повний перелік вбудованих функцій можна знайти в довідці до **Scilab**, тут розглянемо лише частину із них:

Елементар- на функція	Функція в Scilab	Елементар- на функція	Функція в Scilab
$\sin x$	sin(x)	$\arctg x$	atan(x)
$\cos x$	cos(x)	$e^x$	exp(x)
$tg x$	tan(x)	$\sqrt{x}$	sqrt(x)
$ctg x$	cotg(x)	$ x $	abs(x)
$\arcsin x$	asin(x)	$\log_{10}(x)$	log10(x)
$\arccos x$	acos(x)	$\log_2(x)$	log2(x)
$\ln(x)$	log(x)		

Окрім вбудованих елементарних функцій в **Scilab** можна використовувати функції користувача.

Є два способи створення функцій користувача:

за допомогою оператора **deff**;

за допомогою оператора **function**.

У першому способі використовують запис

**deff(рядок1,рядок2)**

тут **рядок1** містить інформацію про вхідні і вихідні параметрах та ім'я функції, **рядок2** містить оператор присвоєння значення вихідному параметру функції. Розглянемо приклад задання функції двох змінних

$$f(x, y) = \sqrt{\left| \cos\left(\frac{x}{y}\right) \right|} + e^y$$



Для цього прикладу **рядок1** містить '**c=fun(x,y)**', **x,y** є вхідними параметрами функції, **c** — вихідний параметр, **fun** - ім'я функції, **рядок2** містить оператор присвоєння значення функції '**c=sqrt(abs(cos(y/x)))+exp(y)**'. Таким чином задання функції користувача для цього прикладу виглядатиме так:

```
deff('c=fun(x,y)','c=sqrt(abs(cos(y/x)))+exp(y)');
```

Перевіримо роботу створеної функції користувача

```
-->deff('c=fun(x,y)','c=sqrt(abs(cos(y/x)))+exp(y)');
```

```
-->a=-4.5; b=8.09; g=fun(a,b)
```

```
g =
```

```
3262.162
```

Розглянемо задання функції за допомогою оператора **function**:

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

```
оператори
```

```
endfunction
```

тут **ff** — ім'я функції, **x1,x2,...,xm** — вхідні параметри, **y1,y2,...,yn** — вихідні параметри функції. Всі імена змінних всередині функції, а також вхідні і вихідні параметри є локальними (тобто визначені лише всередині функції). Створимо функцію для пошуку коренів рівняння

$$ax^3 + bx^2 + cx + d = 0.$$

Після ділення на **a** отримаємо

$$x^3 + rx^2 + sx + t = 0,$$

тут  $r=b/a$ ,  $s=c/a$ ,  $t=d/a$ . Заміною  $x=y-r/3$  отримане рівняння зводиться до вигляду:

$$y^3 + py + q = 0,$$

тут  $p=(3s-r^2)/3$ ,  $q=2r^3/27-rs/3+t$ . Розв'язки цього рівняння можна обчислити за формулою Кардано

$$y_1 = u + v, y_2 = \frac{-(u+v)}{2} + \frac{(u-v)}{2} i \sqrt{3}, y_3 = \frac{-(u+v)}{2} - \frac{(u-v)}{2} i \sqrt{3},$$

$$u = \sqrt[3]{\frac{-q}{2} + \sqrt{D}}, v = \sqrt[3]{\frac{-q}{2} - \sqrt{D}},$$

$$D = \left(\frac{p}{3}\right)^3 + \left(\frac{q}{2}\right)^3$$

```
function [x1,x2,x3]=cub(a,b,c,d)
```

```
r=b/a;
```

```
s=c/a;
```

```
t=d/a;
```

```
p=(3*s-r^2)/3;
```

```
q=2*r^3/27-r*s/3+t;
```

```
D=(p/3)^3+(q/2)^2;
```

```
u=(-q/2+sqrt(D))^(1/3);
```

```
v=(-q/2-sqrt(D))^(1/3);
```

```
y1=u+v;
```

```
y2=-(u+v)/2+(u-v)/2*i*sqrt(3);
```

```
y3=-(u+v)/2-(u-v)/2*i*sqrt(3);
```

```
x1=y1-r/3;
```

```
x2=y2-r/3;
```

```
x3=y3-r/3;
```

```
endfunction
```

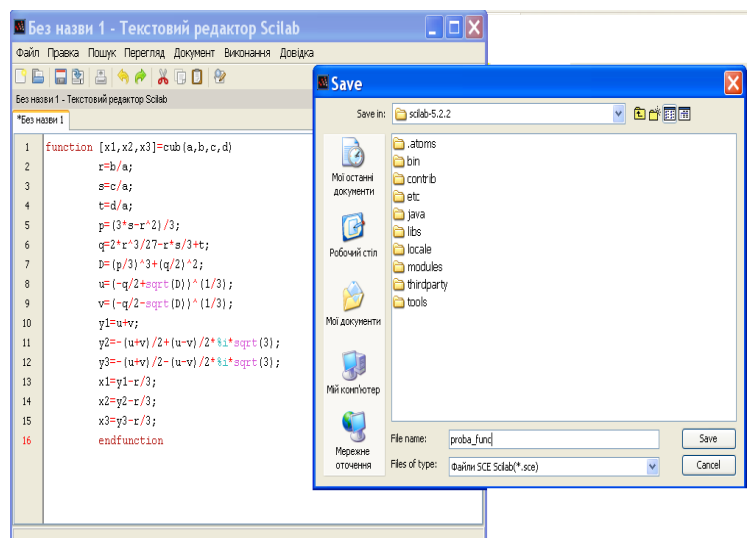


Рис. 1.2. Текстовий редактор Scilab

Складену функцію можна набрати в текстовому редакторі **SciPad** (Програми-Редактор) і зберегти у файлі з розширенням **.sce**. Для того, щоб під час роботи сесії **Scilab** мати можливість звернутись до цієї функції, її потрібно завантажити за допомогою команди **exec(ім'я\_файлу)** або командою головного меню **Файл – Виконати**.

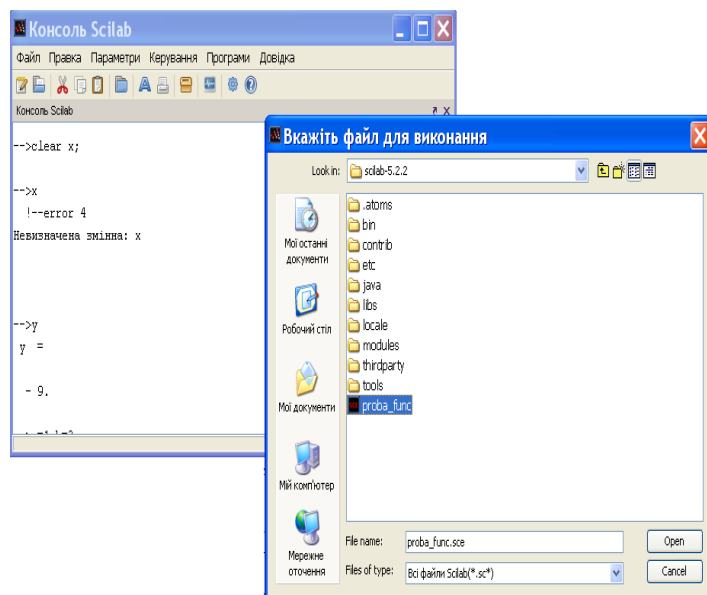


Рис. 1.3. Завантаження файлу з текстом функції для виконання  
а — в ОС Microsoft Windows, б — в ОС Linux

Після виконання функції до неї можна звернутись, наприклад:

```
-->[x1,x2,x3]=cub(1,2,3,4)
x3 =

- 2.2522851 - 0.3473662i
x2 =

- 0.1746854 - 0.8521365i
x1 =

0.4269705 + 1.1995027i
```

### 1.1.5. Чисельне диференціювання та інтегрування в Scilab

Обчислити визначений інтеграл в **Scilab** можна за допомогою функції **intg(a, b, f)**, де **a** і **b** — нижня і верхня межі інтегрування, **f** — ім'я підінтегральної функції.

Розглянемо використання функції **intg** на прикладі обчислення постійної складової ряду Фур'є для пилкоподібного сигналу:

$$a_0 = \frac{2}{T} \int_0^T kt \, dt$$

де  $T=0.001$  с.,  $k=5000$  В/с.

Для обчислення інтегралу в **Scilab** спочатку задамо підінтегральну функцію **s(t)**

```
--> function y=f(t),y=5e3*t,endfunction;
```

Після цього можна обчислити інтеграл  
 -->T=0.001;  
 -->[I,er]=intg(0,T,f)  
 er =

2.776D-17  
 I =

0.0025  
 -->a0=2/T\*I  
 a0 =

5.

Для перевірки отриманого результату обчислимо  $a_0$  аналітично:

$$a_0 = \frac{2}{T} \frac{kT^2}{2} = 5$$

Окрім функції **intg(a, b, f)** для обчислення визначених інтегралів в Scilab можна використовувати функції:

**inttrap([x],y)** — обчислення інтегралу методом трапецій;

**integrate(fun, x, a, b, [er1 [,er2]])** — обчислення інтегралу з використанням квадратурних формул Ньютона-Котеса;

**intsplin([x],s)** — інтегрування результатів експерименту з використанням сплайн-інтерполяції.

Для ознайомлення із синтаксисом цих функцій можна скористатись довідковою системою **Scilab**

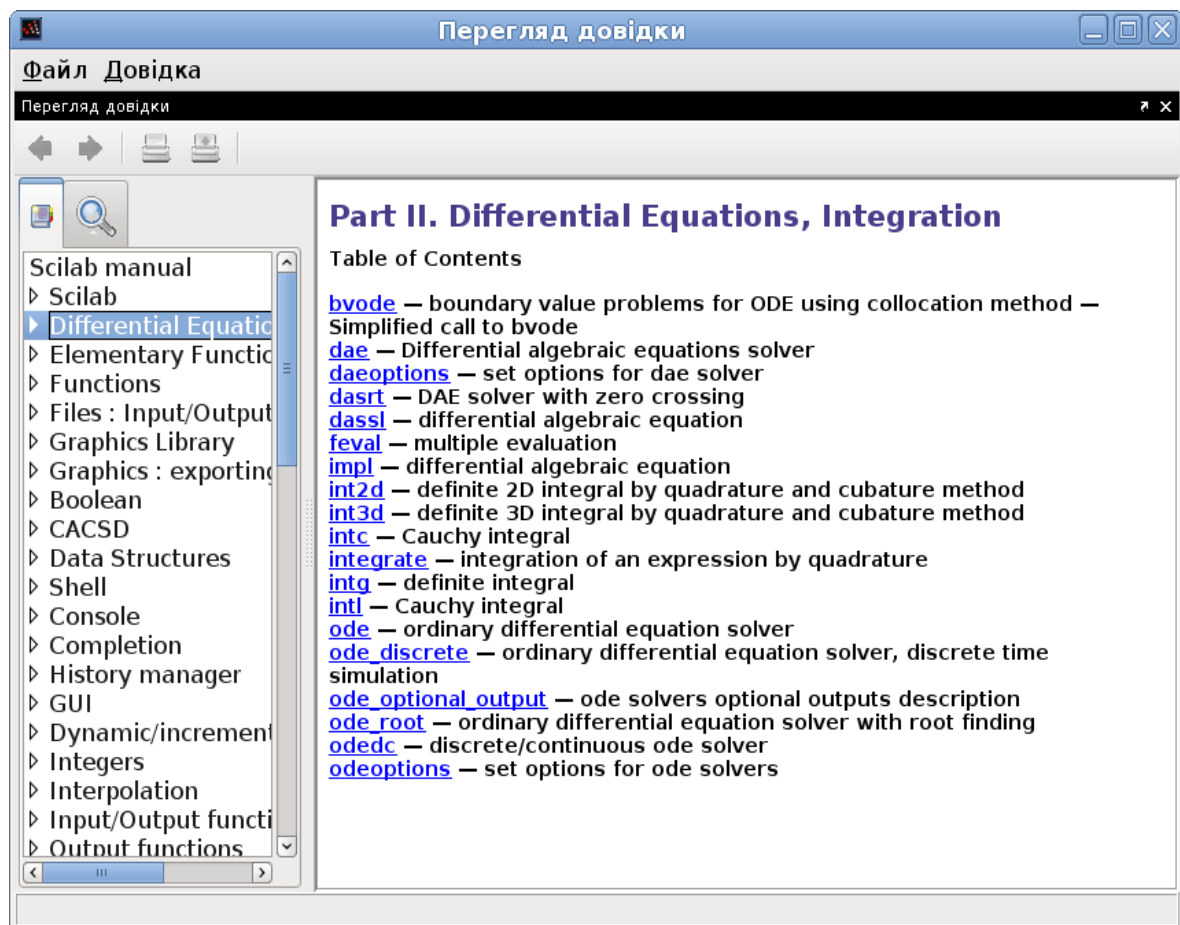


Рис. 1.4. Вбудована довідка **Scilab**

В **Scilab** можна обчислити похідну функції в заданій точці. Диференціювання здійснюється за допомогою команди **numdiff (f, x0)**, де **f** — диференційована функція від змінної **x**, **x0** — координата точки в якій необхідно обчислити похідну. Розглянемо приклад застосування команди **numdiff**. Обчислимо похідну функції **f(x)** в точці **x0=4**

$$f(x) = \frac{1}{\sqrt{x^2 - 5x + 8}}$$

Аналітичний розв'язок:

$$f'(x) = \frac{-(2x-5)}{(x^2-5x+8)2\sqrt{x^2-5x+8}} = \frac{-(2*4-5)}{(16-5*4+8)2\sqrt{16-5*4+8}} = \frac{-3}{16}$$

Розв'язання задачі в **Scilab**:

```
-->function y=f(x), y=1/(sqrt(x^2-5*x+8)) endfunction;
-->numdiff(f,4)
ans =

-0.1875000
--> -3/16
ans =

-0.1875
```

#### 1.1.6. Масиви та матриці в Scilab. Розв'язання задач лінійної алгебри

Для роботи з множиною даних зручно використовувати масиви. Наприклад, можна створити масив для зберігання числових або символьних даних. У цьому випадку замість створення змінної для зберігання кожного значення можна створити один масив, де кожному елементу буде присвоєно порядковий номер.

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

Змінну, яка є списком даних, називають одновимірним масивом, або вектором. Для доступу до даних, що зберігаються в певному елементі масиву, необхідно вказати ім'я масиву та порядковий номер цього елемента (індекс масиву).

Окрім одновимірних масивів в **Scilab** можна використовувати двовимірні масиви (матриці). Для доступу до даних, що зберігаються в такому масиві, потрібно 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 **Scilab** дорівнює одиниці. Індекси можуть бути лише цілими додатніми числами.

Задати одновимірний масив в **Scilab** можна наступним чином:

**name = Xn: dX: Xk**

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

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

**name = Xn: Xk**

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

```
--> Xn=-3.5;dX=1.5;Xk=4.5;
--> X=Xn:dX:Xk
X =
-3.5000 -2.0000 -0.5000 1.0000 2.5000 4.0000
--> Y=sin(X/2)
Y =
-0.9840 -0.8415 -0.2474 0.4794 0.9490 0.9093
--> A=0:5
A =
0 1 2 3 4 5
--> 0:5
ans =
0 1 2 3 4 5
```

Ще один метод визначення векторів і матриць полягає в поелементному їх заданні. Так, для визначення вектора-рядка потрібно ввести ім'я масиву, а потім після знака присвоєння, в квадратних дужках через пробіл або кому перерахувати елементи масиву:

**name=[x1 x2 ... xn] або name=[x1, x2, ..., xn]**

Наступний приклад ілюструє задання векторів-рядків V і W:

```
--> V=[1 2 3 4 5]
V =
1 2 3 4 5
--> W=[1.1,2.3,-0.1,5.88]
W =
1.1000 2.3000 -0.1000 5.8800
```

Для задання вектор-стовпця як розділовий знак використовують “.”

```
--> X=[1;2;3]
X =
1
2
3
```

Крім того, матриці і вектори можна формувати, складаючи їх із раніше заданих матриць і векторів:

```
--> v1=[1 2 3]; v2=[4 5 6]; v3=[7 8 9];
--> //Горизонтальне склеювання векторів-рядків
--> V=[v1 v2 v3]
V = 1 2 3 4 5 6 7 8 9
--> //Вертикальне склеювання векторів-рядків
--> V=[v1; v2; v3]
V =
1 2 3
4 5 6
7 8 9
--> //Горизонтальне склеювання матриць
--> M=[V V V]
M =
1 2 3 1 2 3 1 2 3
```

```
4 5 6 4 5 6 4 5 6
```

```
7 8 9 7 8 9 7 8 9
```

```
--> //Вертикальне склеювання матриць
```

```
--> M=[V;V]
```

```
M =
```

```
1 2 3
```

```
4 5 6
```

```
7 8 9
```

```
1 2 3
```

```
4 5 6
```

```
7 8 9
```

Важливу роль у разі роботи з матрицями грає знак двокрапки «:». Вказуючи його замість індексу при зверненні до масиву, можна отримувати доступ до груп його елементів. Наприклад:

```
--> A=[5 7 6 5; 7 10 8 7;6 8 10 9;5 7 9 10]
```

```
A =
```

```
5. 7. 6. 5.
```

```
7. 10. 8. 7.
```

```
6. 8. 10. 9.
```

```
5. 7. 9. 10.
```

```
--> //Виокремимо з матриці A другий стовпчик
```

```
--> A(:,2)
```

```
ans =
```

```
7.
```

```
10.
```

```
8.
```

```
7.
```

```
--> // Виокремимо з матриці A третій рядок
```

```
--> A(3,:)
```

```
ans =
```

```
6. 8. 10. 9.
```

```
--> // Виокремимо з матриці A підматрицю M
```

```
--> M=A(3:4,2:3)
```

```
M =
```

```
8. 10.
```

```
7. 9.
```

```
--> //Видалимо з матриці A другий стовпчик
```

```
--> A(:,2)=[]
```

```
A =
```

```
5. 6. 5.
```

```
7. 8. 7.
```

```
6. 10. 9.
```

```
5. 9. 10.
```

```
--> // Видалимо з матриці A третій рядок
```

```
--> A(3,:)=[]
```

```
A =
```

```
5. 6. 5.
```

```
7. 8. 7.
```

```
5. 9. 10.
```

```
--> //Зформувати вектор-стовпець із матриці M
```

```
--> v=M(:)
```

```
v =
```

8.  
7.  
10.  
9.

--> //Викопіювати із вектора v другий, третій і четвертий елементи

--> b=v(2:4)

b =

7.  
10.  
9.

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

Операція	Призначення
+	додавання
-	віднімання
'	транспонування
*	матричне множення
^	піднесення до степеня
\	ліве ділення ( $A \setminus B \Rightarrow (A^{-1} B)$ )
/	праве ділення ( $B/A \Rightarrow (B \cdot A^{-1})$ )
.*	поелементе множення
.^	поелементе піднесення до степеня
.\	поелементе ліве ділення
./	поелементе праве ділення

Розглянемо приклад:

-->A=[1 2 0;-1 3 1;4 -2 5];

-->B=[-1 0 1;2 1 1;3 -1 -1];

--> //Обчислити  $(A'+B)^2 - 2A(0.5B'-A)$

-->(A'+B)^2-2\*A\*(1/2\*B'-A)

ans =

10. 8. 24.  
11. 20. 35.  
63. - 30. 68.

--> //Розв'язати матричні рівняння  $A \cdot X=B$  и  $X \cdot A=B$ .

-->A=[3 2;4 3];

-->B=[-1 7;3 5];

--> //розв'язок матричного рівняння  $AX=B$ :

-->X=A \ B

X =

- 9. 11.  
13. - 13.

--> //Перевірка

-->A\*X-B

ans =

0. 0.  
0. 0.

--> // розв'язок матричного рівняння  $XA=B$ :

```
-->X=B/A
X =
- 31. 23.
- 11. 9.
```

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:

```
--> x=[0.1 -2.2 3.14 0 -1];
--> sin(x)
ans =
0.0998334 - 0.8084964 0.0015927 0. - 0.8414710
```

Для роботи з матрицями і векторами в Scilab є великий набір спеціальних функцій, які описані в додатку 1. Розглянемо приклади застосування матричних функцій до розв'язання систем лінійних рівнянь.

Приклад 1. Використання оберненої матриці

Якщо система лінійних рівнянь задана матричним рівнянням  $Ax=b$ , то знайти її розв'язок можна за формулою  $x=A^{-1}b$

```
--> A=[2 1 -5 1;1 -3 0 -6;0 2 -1 2;1 4 -7 6];b=[8;9;-5;0];
--> x=inv(A)*b
x =
3.
- 4.
- 1.
1.
```

Приклад 2. Розв'язання системи лінійних рівнянь методом Гауса

//Матриця і вектор вільних членів системи:

```
A=[2 -1 1;3 2 -5;1 3 -2]; b=[0;1;4];
```

//Приведення розширеної матриці до трикутного вигляду:

```
C=trref([A b]);
```

//Визначення розмірності розширеної матриці:

```
[n,m]=size(C); //m- номер останнього стовпця матриці C
```

//Копіювання останнього стовпця матриці C:

```
x=C(:,m) //x - розв'язок системи
```

```
--> x =
0.4642857
1.6785714
0.75
```

### 1.1.7. Дії з поліномами в Scilab

Система **Scilab** має певні засоби для роботи з поліномами. Нагадаємо, що поліном  $n$ -го степеня від  $x$  має дві рівноцінні форми задання:

$$a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3 + \dots + a_n * x^n = \prod_{i=1}^n (x - x_i)$$

тут  $a_0, a_1, a_2, a_3 \dots a_n$  — коефіцієнти полінома,  $x_1, x_2, x_3 \dots x_n$  — нулі полінома (значення змінної  $x$ , у яких значення полінома рівне нулю).

Задати поліном в можна за допомогою функції

**poly(список, 'незалежна\_змінна', 'с або r')**



у списку задають коефіцієнти полінома або його корені, другим аргументом функції є ім'я незалежної змінної, третім — параметр, який визначає спосіб задання полінома: **c** — для задання полінома використовують значення його коефіцієнтів, **r** — для задання полінома використовують значення його нулів. Розглянемо два приклади задання полінома.

Приклад 1. Задання полінома через його коефіцієнти

Задамо поліном п'ятого порядку з коефіцієнтами  $a_0 = 8, a_1 = 0, a_2 = 0, a_3 = 1, a_4 = 7, a_5 = 17$

```
-->poly([8,0,0,1,7,17],'x','c')
```

ans =

$$8 + x^3 + 7x^4 + 17x^5$$

Приклад 2. Задання полінома через його нулі

Задамо поліном другого порядку з нулями 7 і 2

```
-->poly([7,2],'m','r')
```

ans=

$$14 - 9m + m^2$$

До поліномів можна застосовувати такі операції — додавання і віднімання, множення, ділення, піднесення до степеня. Наступний приклад ілюструє виконання дій над поліномами.

Приклад 3. Дії із поліномами

```
-->a=poly([8,0,0,1,7,17],'x','c') //Задання полінома від x коефіцієнтами
```

a =

$$8 + x^3 + 7x^4 + 17x^5$$

```
-->b=poly([7,2],'x','r') //Задання полінома від x коренями полінома
```

b =

$$14 - 9x + x^2$$

```
-->c=a+b
```

c =

$$22 - 9x^2 + x^3 + x^4 + 7x^5 + 17x^6$$

```
-->r=c-a
```

r =

$$14 - 9x + x^2$$

```
-->d=a*b
```

d =

$$112 - 72x^2 + 8x^3 + 14x^4 + 89x^5 + 176x^6 - 146x^7 + 17x^8$$

```
-->s=d/a
```

s =

$$14 - 9x + x^2$$

```
-->v=b^2
```

v =

$$196 - 252x^2 + 109x^3 - 18x^4 + x^5$$

Для знаходження коренів рівняння

$$a_0 + a_1 * x + a_2 * x^2 + a_3 * x^3 + \dots a_n * x^n = 0$$

використовують функцію **roots(a)**, тут **a** — вектор із коефіцієнтами полінома. Результатом роботи функції є вектор коренів рівняння. Розглянемо приклад:

```
-->V=[-1 0 8 -8 2];
-->p=poly(V,'x','c')
p =
      2      3      4
- 1 + 8x - 8x + 2x
-->V=roots(p)'
V =
  0.4588039 - 0.3065630i  1.5411961  2.306563
-->poly(V,'x','r')
ans =
      2      3      4
- 0.5 + 4x - 4x + x
```

### 1.1.8. Побудова графіків в Scilab

**Scilab** містить певний набір функцій для графічного подання інформації. Для побудови графіків функції однієї змінної використовують функції **plot** і **polarplot**. Перша будує графіки в декартових координатах, друга - в полярних координатах.

#### Функція plot

Звертання до функції **plot** виглядає наступним чином:

**plot(x, y),**

тут **x** - вектор значень незалежної змінної, **y** - вектор значень функції **f**. Побудуємо графік функції  $y = \sin^2(2 \cdot x)$ ,  $x \in [-4; 4]$

```
-->x=-4:0.1:4;
-->y=(sin(2*x))^2;
-->plot(x,y)
```

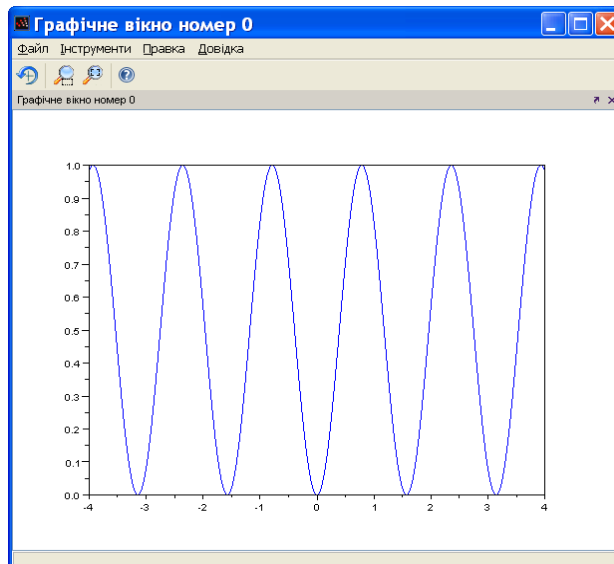


Рис. 1.5.

Для побудови декількох графіків в одному вікні використовують таке звертання до функції **plot**

**plot(x1, y1, x2, y2, x3, y3),**

тут **x1** - список значень аргумента **x** для функції  $f_1 = y_1(x)$ , **x2** - значень аргумента **x** для функції  $f_2 = y_2(x)$ , і т.д. Якщо графіки будуються в одному проміжку значень аргумента, то в якості **x1**, **x2**, **x3** може виступати один і той же вектор. Побудуємо графіки функцій:

```

 $f_1 = \sin(x), f_2 = \cos(x), f_3 = \sqrt{|x|}, x \in [-4; 4]$ 
-->x=-2*%pi:0.01:2*%pi;
-->f1=sin(x);
-->f2=cos(x);
-->f3=sqrt(abs(x));
-->plot(x,f1,x,f2,x,f3)

```

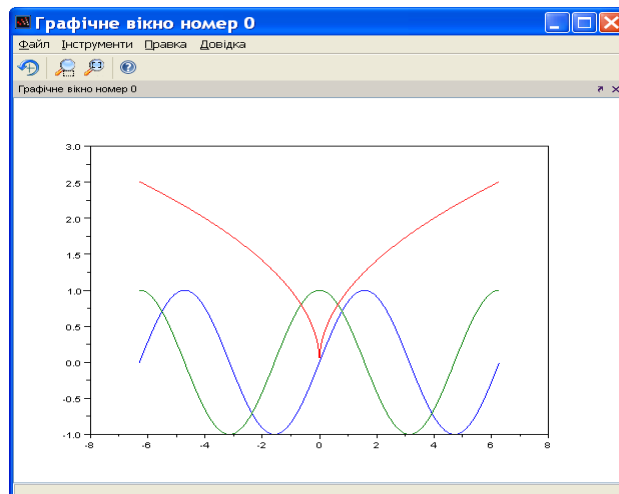


Рис. 6.

Починаючи з Scilab 4 можна виводити декілька графіків в одному вікні, не суміщаючи їх в одних координатних осях. Наприклад, якщо графічне вікно повинно містити чотири самостійні графіки, все вікно розбивається на 4 області, а потім у кожен 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])**

тут **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 підінтервалів по осях X і Y;

**grid** - логічна змінна, яка визначає наявність (% t) або відсутність координатної сітки (% f). Цей параметр слід вказувати для обох осей, наприклад, [% t,% t];

**bound** - логічна змінна, яка у разі значення true дає змогу ігнорувати параметри tics (2) і tics (4).

**title** - заголовок, який буде виводитись над графічною областю;

**x-leg, y-leg** - підписи 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].

Після визначення області в ній можна вивести графік функції за допомогою команди plot. Для прикладу побудуємо в одному вікні графіки функцій  $y = \sin(2x)$ ,  $z = \cos(3x)$ ,  $u = \cos(\sin(2x))$ ,  $v = \sin(\cos(3x))$  в одному графічному вікні, але кожен графік матиме свою систему координат.

```

-->x=[-10:0.01:10];
-->y=sin(2*x); z=cos(3*x); u=cos(sin(2*x)); v=sin(cos(3*x));
-->rect=[min(x),-1,max(x),1];
-->tics=[2,11,10,5];
-->plotframe(rect,tics,['%t,%t'],["Function y=sin(2x)",...
"X","Y"], [0,0,0.5,0.5]);
-->plot(x,y);
-->plotframe(rect,tics,['%f,%f'],["Function y=cos(3x)",...
"X","Y"], [0.5,0,0.5,0.5]);
-->plot(x,z);
-->plotframe(rect,tics,['%f,%f'],["Function y=cos(sin(2x))",...
"X","Y"], [0,0.5,0.5,0.5]);
-->plot(x,u);
-->plotframe(rect,tics,['%t,%t'],["Function y=sin(cos(3x))",...

```

```
"X","Y"], [0.5,0.5,0.5,0.5]);
-->plot(x,v);
```

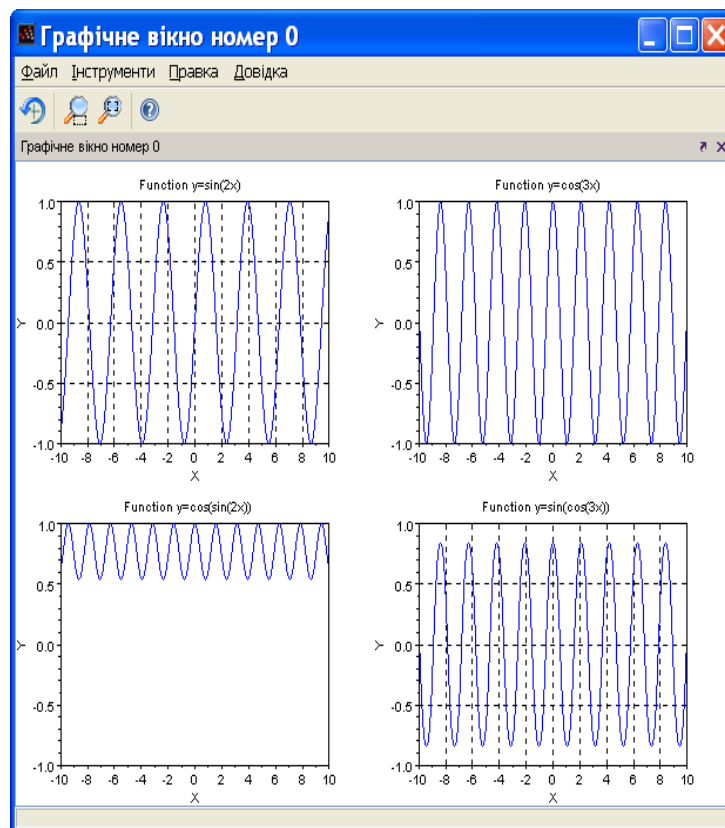


Рис. 1.7.

Для відображення декількох графіків в одному вікні можна також використати функцію **subplot (m, n, p)** або **subplot (mnp)**. Виконання функції призводить до того, що графічне вікно розбивається на **m** вікон по вертикалі і **n** вікон по горизонталі, поточним вікном стає вікно з номером **p** (результат роботи функції **plot(x,y)** буде виведено у це вікно). Для прикладу побудуємо графіки функцій  $y = \sin(x)$ ,  $z = \cos(x)$ ,  $u = \cos(\sin(x))$ ,  $v = \sin(\cos(x))$ ,  $w = \exp(\sin(x))$  і  $r = \exp(\cos(x))$  в одному графічному вікні, кожен у своїй системі координат, використовуючи команду **subplot**:

```
-->x=[-10:0.01:10];
-->y=sin(x); z=cos(x);
-->u=cos(sin(x)); v=sin(cos(x));
-->w=exp(sin(x)); r=exp(cos(x));
-->subplot(3,2,1);
-->plot(x,y);
-->subplot(3,2,2);
-->plot(x,z);
-->subplot(3,2,3);
-->plot(x,u);
-->subplot(3,2,4);
-->plot(x,v);
-->subplot(3,2,5);
-->plot(x,w);
-->subplot(3,2,6);
-->plot(x,r);
```

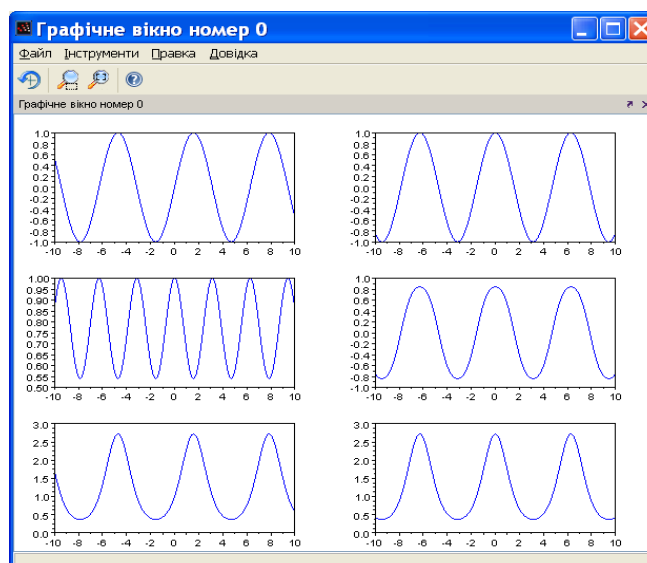


Рис. 1.8.

Вигляд графіка можна змінювати, додавши при зверненні до функції **plot** крім основних аргументів ще один аргумент — рядок, що містить від одного до трьох символів, які визначатимуть колір лінії, тип маркера і тип лінії графіка. У цьому разі звертання до функції **plot** буде таким:

### **plot (x, y, string)**

Рядок **string** виглядає так: 'параметр1параметр2параметр3', параметр1 визначає колір лінії графіка, параметр2 — тип точок графіка, параметр3 — тип лінії

параметр1	параметр2		параметр3
колір	.	крапка	- суцільна
y жовтий	o	коло	: пунктирна
m рожевий	+	знак "плюс"	-. штрихпунктирна
c голубий	x	хрестик	- штрихова
r червоний	*	зірочка	
g зелений	s	квадрат	
b синій	d	ромб	
w білий	v	трикутник вершиною вниз	
	<	трикутник вершиною вліво	
	>	трикутник вершиною вправо	
	p	п'ятикутна зірка	
	h	шестикутна зірка	

Якщо потрібно побудувати кілька графіків, то для кожного графіка потрібно вказати по три аргументи — **x, y, string**. Для *n* графіків — **plot (x1, y1, string1, x2, y2, string2, x3, y3, string3 ... xn, yn, stringn)**. Для прикладу побудуємо три графіки:

$x \in [0; 3]$  :

$f_1(x) = x^2$  — пунктирна лінія жовтого кольору;

$f_2(x) = 1 + x$  — штрихова лінія голубого кольору;

$f_3(x) = e^x$  — суцільна лінія

чорного кольору.

```
-->x=0:0.1:3;
```

```
-->f1=x^2;
```

```
-->f2=x+1;
```

```
-->f3=%e^x;
```

```
-->plot(x,f1,'y:',x,f2,'c--',x,f3,'k-')
```

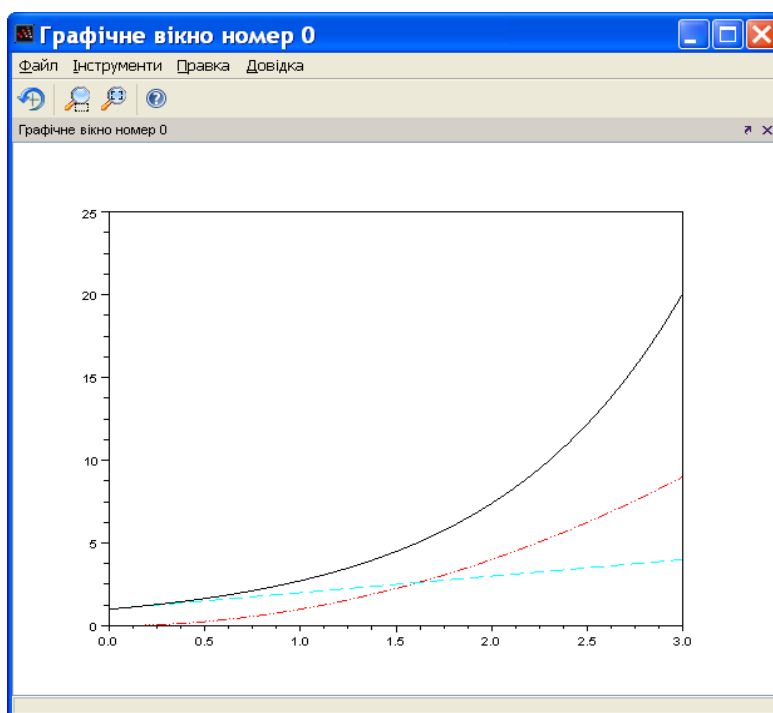


Рис. 1.9.

Для побудови координатної сітки на графіку можна скористатися функцією **xgrid(rgb)**. Кожен з параметрів **r**, **g** і **b** може набувати значення — 0 або 1. Параметр **r** визначає наявність (**r** = 1) або відсутність (**r** = 0) червоної складової в кольорі лінії, **g** - зеленої і **b** - синьої. Різні комбінації значень цих параметрів дають змогу отримувати різні кольори. Наприклад, внаслідок звертання до функції **xgrid(100)** в графічній області буде відображена сітка червоного кольору. Значення параметрів слід вводити підряд без розділювачів.

Для виводу назв графіка і осей використовують функцію

**xtitle(name, xname, yname),**

тут **name** — назва графіка, **xname** — назва осі X, **yname** — назва осі Y.

Для опису графіків використовують функцію

**legend(opys1,opys2,...,opysn,place,frame)**

тут **opys1** — опис першого графіка, **opys2** — опис другого графіка, ..., **opysn** — опис n-ного графіка. Параметр **place** визначає місце виведення опису (1 — правий верхній кут графіка, 2 — лівий верхній кут графіка, 3 — лівий нижній кут графіка, 4 — правий нижній кут графіка, 5 — місце виведення опису задається користувачем після побудови графіка. Параметр **frame=%t** викликає побудову прямокутної рамки довкола опису (**frame=%f** — рамка не будується).

### Функція plot2d

Окрім функції **plot** для побудови двовимірних графіків можна використовувати функцію **plot2d**:

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

тут **logflag** — рядок з двох символів, кожен з яких визначає тип осей (**n** — нормальна вісь, **l** — логарифмічна вісь), за замовчуванням — «**nn**»; **x** - масив абсцис; **y** — масив ординат (якщо потрібно побудувати кілька графіків, то кожен стовпець яких містить масив ординат окремого графіка). Якщо масив **x** є вектором, то функції **y1**, **y2**, ..., **yn** мають залежати від однієї і тієї ж змінної **x**. Коли ж **x** та **y** є матрицями одного розміру, то кожен стовпець матриці **y** відображається щодо відповідного стовпця матриці **x**;

**keyi = valuei** - послідовність значень властивостей графіка **key1 = value1**, **key2 = value2**, ..., **keyn = valuen**, визначають його зовнішній вигляд. Параметр **keyi=valuei** може набирати таких значень:

**style** — визначає масив числових значень кольорів графіка. Кількість елементів масиву має співпадати з кількістю зображуваних графіків. Можна скористатися функцією **color**, яка за назвою (**color("ім'я\_кольору")**) або коду **rgb** (**color(r, g, b)**) кольору формує потрібний id (код) кольору;

**rect** — значення параметра **keyi = valuei** функції **plot2d** — це вектор [**xmin**, **ymin**, **xmax**, **ymax**], який визначає розмір вікна навколо графіка. Тут **xmin**, **ymin** - положення верхньої лівої вершини; **xmax** — ширина вікна; **ymax** — висота вікна;

**axesflag** — значення параметра **keyi = valuei** функції **plot2d** — визначає наявність рамки навколо графіка. Необхідно виділити наступні базові значення цього параметра: 0 — немає рамки; 1 — зображення рамки, вісь OY зліва (за замовчуванням); 3 — зображення рамки, вісь OY праворуч; 5 — зображення осей, що проходять через точку (0,0);

параметр **nax** задає число основних і проміжних поділок. Якщо параметр **axesflag** = 1 (за замовчуванням), то **nax** є масивом з чотирьох значень: [**nx**, **Nx**, **ny**, **Ny**]. Тут **Nx** (**Ny**) — число основних поділок с підписами під віссю X (Y), **nx** (**ny**) — число проміжних поділок;

**leg** — значення параметра **keyi = valuei** функції **plot2d** — рядок, який визначає легенди для кожного графіка: "**leg1 @ leg2 @ leg3...legn**", тут **leg1** - легенда першого графіка, ..., **legn** — легенда n-го графіка. Розглянемо кілька прикладів використання функції **plot2d**.

Приклад 1. Задання кольорів графіка

```
x=[-2*%pi:0.1:2*%pi];  
y=[sin(x);cos(x)];  
plot2d(x,y',style=[color("red"),color(0,176,0)]);
```

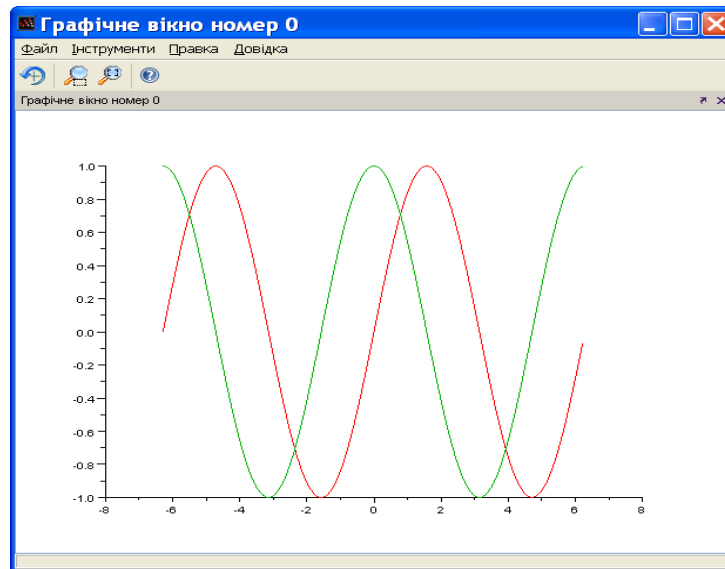


Рис. 1.10.

Приклад 2. Задання розмірів вікна навколо графіка за допомогою параметра **rect**

```
x=[-2*%pi:0.1:2*%pi];  
y=[sin(x);cos(x)];  
plot2d(x,y',style=[color("red"),color(0,176,0)], rect=[-8,-2,8,2]);
```

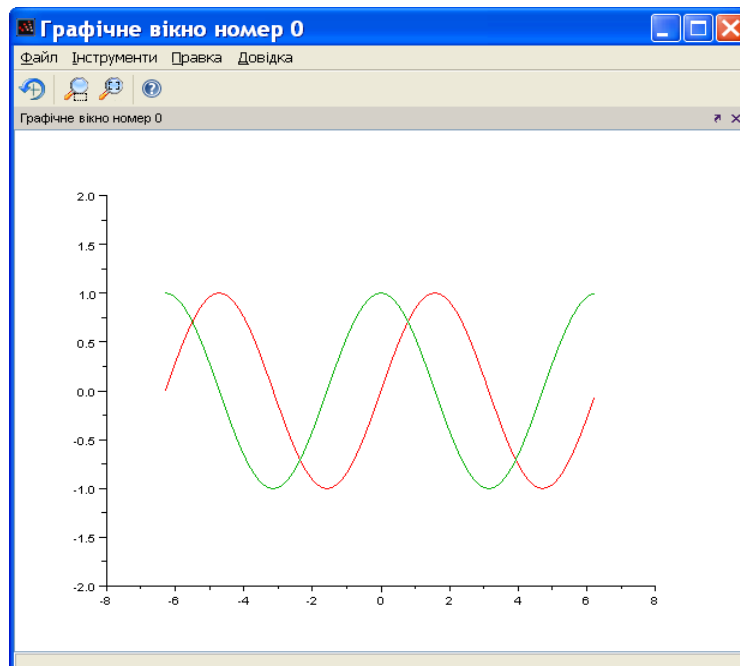


Рис. 1.11.

Приклад 3. Побудова рамки навколо графіка та визначення розміру координатних осей з параметром **axesflag**

```
-->x=[-2*%pi:0.1:2*%pi];  
-->y=[sin(x); cos(x)];  
-->subplot(2,2,1)  
-->plot2d(x,y',style=[color("red"), color("blue")], axesflag=0);
```

```
-->subplot(2,2,2)
-->plot2d(x,y',style=[color("red"), color("blue")], axesflag=1);
-->subplot(2,2,3)
-->plot2d(x,y',style=[color("red"), color("blue")], axesflag=3);
-->subplot(2,2,4)
-->plot2d(x,y',style=[color("red"), color("blue")], axesflag=5);
```

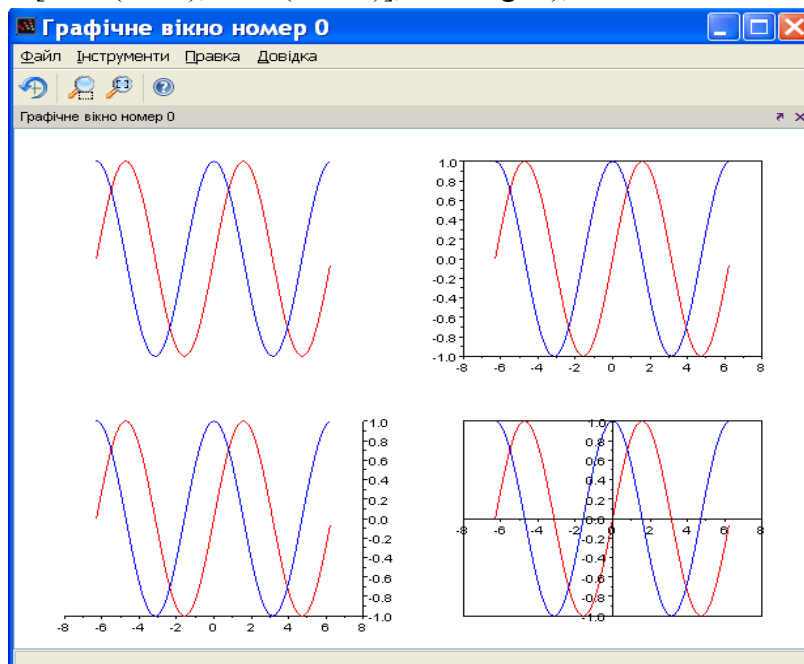


Рис. 1.12.

Приклад 4. Нанесення на координатні осі графіка основних і проміжних поділок за допомогою параметра **nax**

```
-->x=[-8:0.1:8];
-->y=[sin(x); cos(x)];
-->plot2d(x,y',style=[color("red"),color("blue")],axesflag=1,nax=[4,9,3,6]);
```

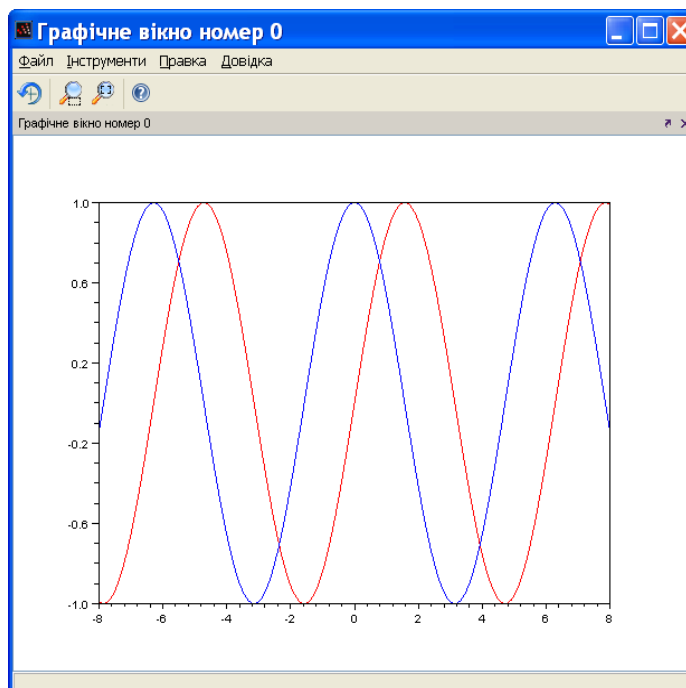


Рис. 1.13.



Приклад 5. Виведення легенди графіка за допомогою параметра **leg**

```
-->x=[-2*%pi:0.1:2*%pi];
```

```
-->y=[sin(x); cos(x)];
```

```
-->plot2d(x,y',axesflag=5, leg="sin(x)@cos(x)");
```

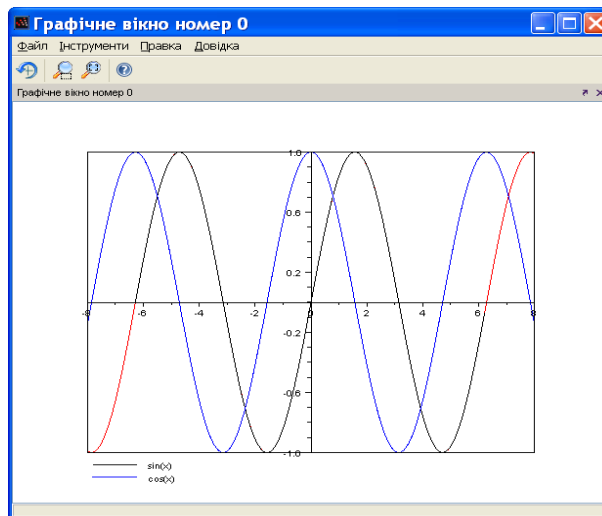


Рис. 1.14.

Слід зазначити, що зовсім не обов'язково використовувати повну форму запису функції **plot2d** з усіма її параметрами. У найпростішому випадку до неї можна звернутися коротко, як і до функції **plot**.

Функцію **plot2d** можна використовувати для побудови точкових графіків. У цьому випадку звертання до функції має вигляд:

**plot2d (x, y, d)**

тут **d** — від'ємне число, яке визначає тип маркера:

Число	Маркер
-0	крапка
-1	+
-2	хрестик
-3	плюс, вписаний в коло
-4	зафарбований ромб
-5	незафарбований ромб
-6	трикутник вершиною вгору
-7	трикутник вершиною вниз
-8	плюс, вписаний в ромб
-9	коло
-10	*
-11	квадрат
-12	трикутник вершиною вправо
-13	трикутник вершиною вліво
-14	п'ятикутна зірка

Розглянемо приклад побудови точкового графіка:

```
-->x=[-2*%pi:0.25:2*%pi];
```

```
-->y=sin(x);
```

```
-->plot2d(x,y,-8);
```

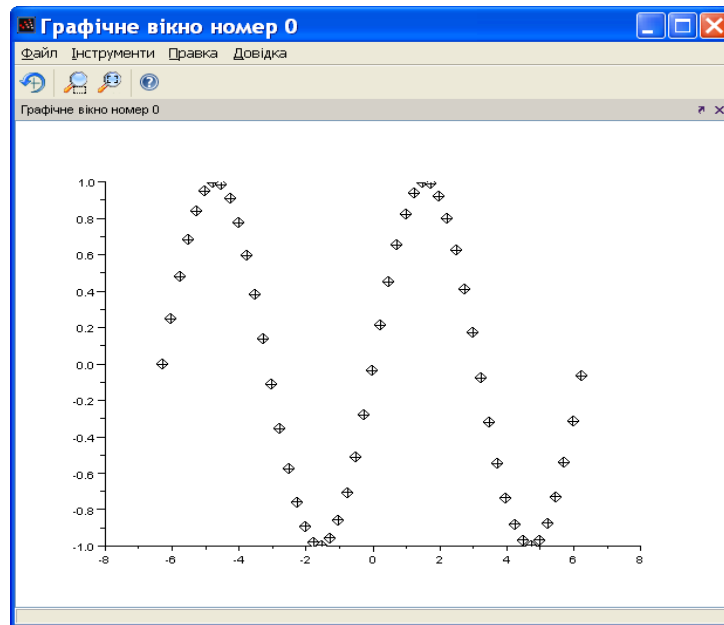


Рис. 1.15.

### Побудова сходинкових графіків

Для побудови сходинкових графіків в Scilab використовують функцію **plot2d2 (x, y)**. За синтаксисом вона повністю співпадає з функцією **plot2d**. Головною відмінністю **plot2d2** є те, що X та Y можуть бути незалежними один від одного функціями, важливо лише, щоб масиви X і Y були розбиті на однакову кількість інтервалів.

```
-->x=[1947 1958 1970 1980 1999 2006];
```

```
-->y=[2.003 3.1 3.6 4.7 5.2 5.4];
```

```
-->plot2d2(x,y);
```

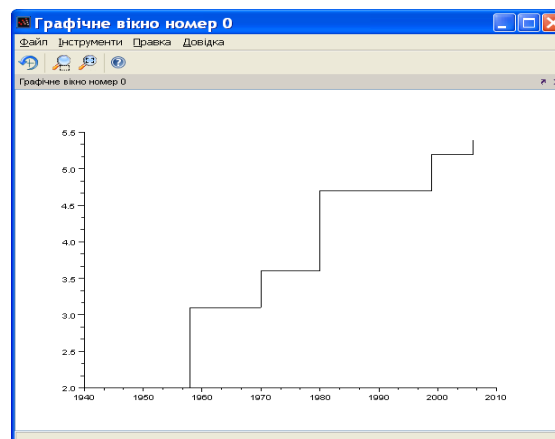


Рис. 1.16.

### Побудова графіків в полярній системі координат

В **Scilab** для формування графіка в полярній системі координат необхідно сформувати масиви значень полярного кута і полярного радіуса, а потім звернутися до функції **polarplot**:  
**polarplot (fi, ro, [key1 = value1, key2 = value2,..., keyn = valuen]),**

тут **fi** — полярний кут; **ro** — полярний радіус; **keyi = valuei** — послідовність значень властивостей графіка (див. опис параметра **keyi = valuei** для функції **plot2d**). Розглянемо приклад побудови графіків функцій в полярній системі координат:

```
-->fi=0:0.01:2*%pi;
-->ro=3*cos(5*fi);ro1=3*cos(3*fi);
-->polarplot(fi,ro,style=color("red"));
-->polarplot(fi,ro1,style=color("blue"));
```

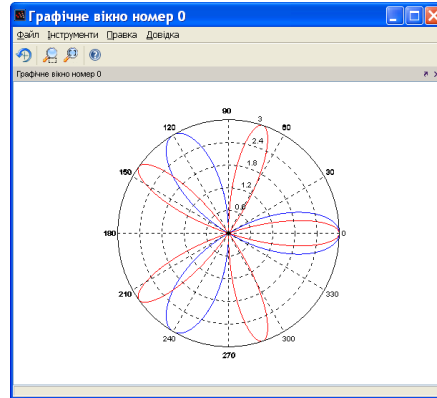


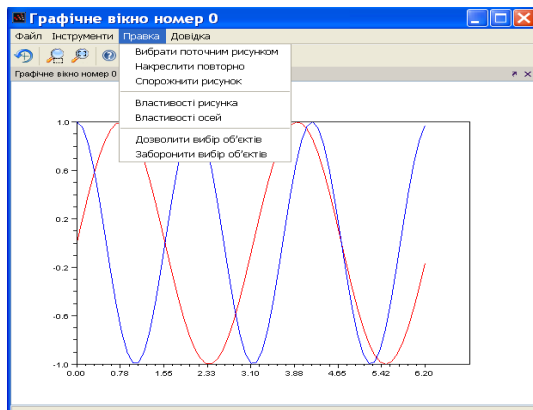
Рис. 1.17.

#### 1.1.9. Форматування графіків через меню графічного вікна

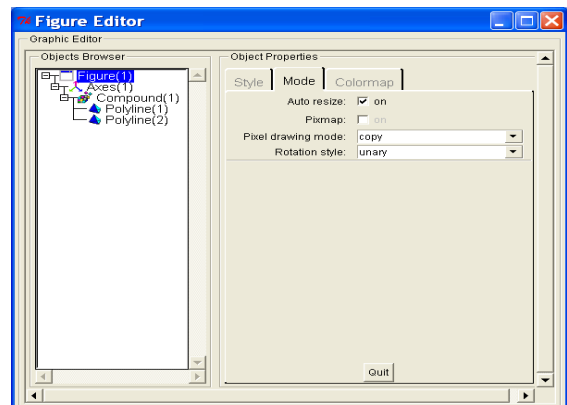
В **Scilab** зовнішній вигляд графіка можна змінювати, використовуючи можливості графічного вікна, в якому він відображається. Можливості форматування ми розглянемо на прикладі побудови графіків функцій  $y_1 = \sin(2x)$  та  $y_2 = \sin(3x)$  на інтервалі  $[0; 2\pi]$  з кроком 0,1. Сформуємо масив **x** і скористаємося функцією **plot2d**

```
-->x=[0:0.1:2*%pi];
-->y=[sin(2*x); cos(3*x)];
-->plot2d(x,y',style=[color("red"),color("blue")],axesflag=1,nax=[4,9,3,6]);
```

Перехід до режиму форматування здійснюють командою **Правка — Властивості рисунка** меню графічного вікна, яка відкриває вікно форматування отриманого графіка **Figure Editor**. Ліва частина вікна — **Object Browser** (рис.1.18б) є полем перегляду об'єктів, доступних для форматування. Спочатку в полі **Object Browser** завжди відображаються два об'єкти: **Figure** (Графічне вікно) і його дочірній об'єкт **Axes** (Осі). Значок «плюс» біля об'єкта вказує на те, що він містить об'єкти більш низького порядку. Якщо клацнути по значку «плюс» в об'єкта **Axes** (Осі), з'явиться об'єкт — **Compound** (1) (Група) також із позначкою «плюс». Об'єкт **Compound** (1) містить побудовані в одних координатних осях графіки функцій  $y_1 = \sin(2x)$  і  $y_2 = \cos(3x)$  — відповідно **Polyline** (1) і **Polyline** (2).



а



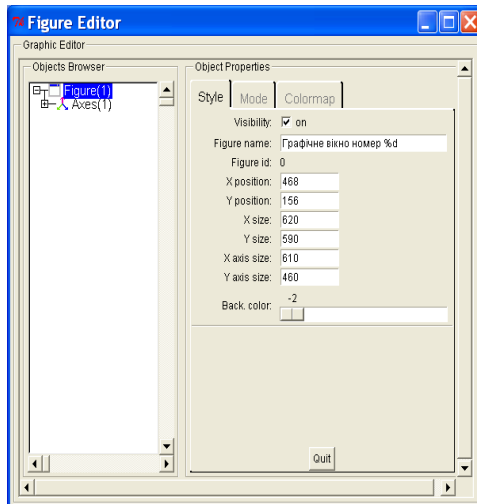
б

Рис. 1.18. а — вікно відображення графіка, б — вікно редактора **Figure Editor**

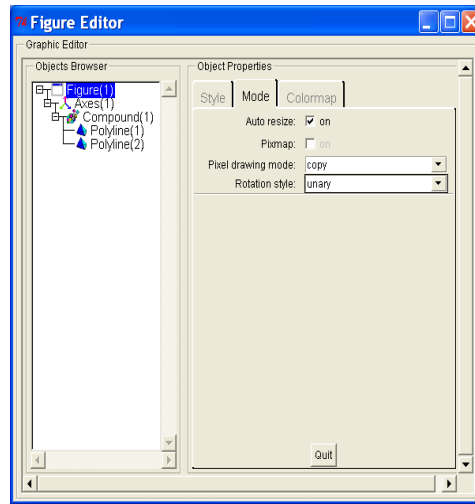
Об'єкт **Figure** (Графічне вікно) — це графічне вікно і власне графік, який відображається у графічному вікні. Розглянемо методи зміни властивостей графічного вікна.

### Форматування об'єкта **Figure** (Графічне вікно)

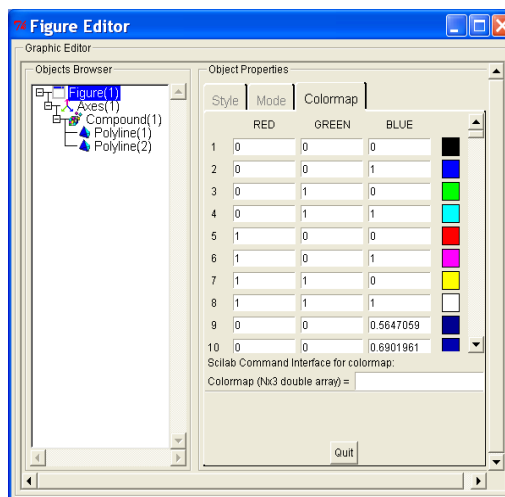
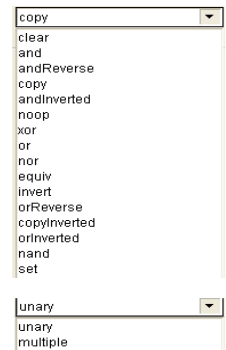
Клацання “мишкою” по об'єкту **Figure(1)** (Графічне вікно) робить його активним, а в правій області вікна — **Object Properties** з'являються властивості активного об'єкта, які можна змінити шляхом зміни параметрів у закладках **Style**, **Mode**, **Color**.



а



б



в

Рис. 1.19. Закладки вікна **Object Properties** а — **Style**, б — **Mode**, в — **Color**

На рис. 1.19.а зображена закладка **Style** вікна форматування **Figure Editor** для об'єкта **Figure(1)**. Тут можна змінити значення наступних властивостей:

**Visibility** (відображення графіка) — перемикач, що приймає значення «on» і «off». За замовчуванням встановлено стан «on» — графік виводиться на екран;

**Figure name** (ім'я графіка) — це послідовність символів, які виводяться в рядку заголовка графічного вікна. За замовчуванням графічному вікну присвоюється ім'я **Scilab Graphic (% d)**, де % d — це порядковий номер графіка (Figure id). Для першого графічного вікна Figure id дорівнює 0, для другого — 1, для третього — 2 і т. д. Однак ви можете ввести будь-яке ім'я.

**X position**, **Y position** — ці поля визначають положення графічного вікна на дисплеї у пікселях по горизонталі й вертикалі відповідно. Точка з координатами [0; 0] — це верхній

лівий кут екрану. **X size**, **Y size** — це відповідно ширина і висота графічного вікна у пікселях;

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

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

**Auto resize** — властивість, яка дає змогу змінювати розмір графіка. Коли цей режим увімкнений («on» — за замовчуванням), ми можемо змінювати розмір графічного вікна і графіка, перетягуючи межі вікна “мишкою”. У вимкненому стані («off») графік буде зберігати свої розміри;

**Pixmap** — режим растрового зображення. У вимкненому стані («off» за замовчуванням) зображення формується безпосередньо на екрані. У ввімкненому стані («on») графік створюється як растрове зображення і скеровується в графічне вікно командою **show\_pixmap ()**. Слід зауважити, що режим **Pixmap** використовують з метою згладжування переходів між кадрами під час створення анімованих графіків;

**Pixel drawing mode** — властивість, яка визначає спосіб формування зображення на екрані. За замовчуванням встановлений режим «сору». У цьому випадку точно виконується необхідна операція побудови графіка. Однак часто необхідно нанести зображення на вже існуюче, при цьому колір новозбудованого 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** — цю властивість можна застосувати лише до тривимірних графіків. Режим «unary» (за замовчуванням) призначений для обертання лише виділених графіків, у ввімненому стані «multiple» обертаються всі тривимірні графіки.

Для прикладу змінимо заголовок графіка ( **Figure name** — встановимо заголовок **red: y1=sin(2x) blue: y2=cos(3x)**) та колір фону ( **Back. Color** — встановимо колір фону із номером 17). На рис. 1.20 зображена копія зміненого вікна графіка

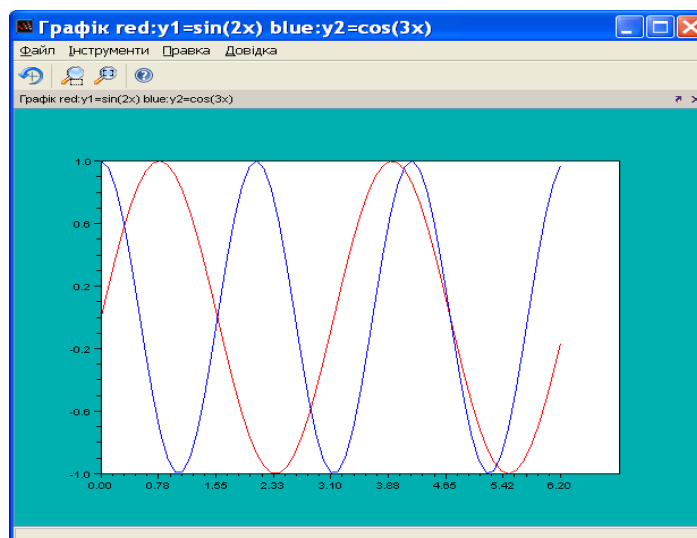


Рис. 1.20.

### Форматування об'єкта **Axes** (Осі графіка)

Для зміни властивостей об'єкта **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. Дані **X**, **Y** і **Z** ідентичні, з тією лише різницею, що дають встановлювати бажаний зовнішній вигляд відповідно для осей **X**, **Y** і **Z**. Тому ми розглянемо лише закладку **X** (див. рис. 1.21).

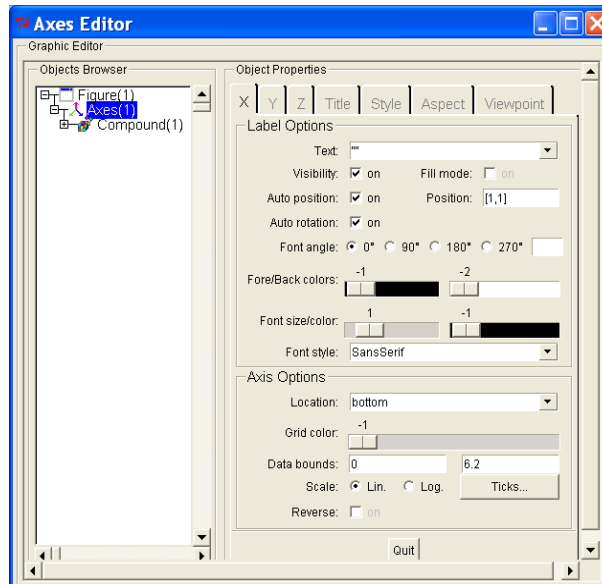


Рис. 1.21.

На закладці **X** всі властивості розділені на дві області: **Label Options** (Властивості підписів осей) і **Axis Options** (Властивості осей). В області **Label Options** можна встановити:

**Label** — власне підпис осі: будь-яка послідовність символів;

**Visibility** — видимість: перемикач, який перебуває в стані «on» або «off». За замовчуванням осі графіка виводяться на екран (положення «on»);

**Fill mode** — режим заливки: перемикач, який перебуває в стані «on» або «off» («off» за замовчуванням). Для того, щоб визначити колір фону навколо підпису осі, необхідно встановити стан «on»;

**Auto position** — автоматичне визначення положення підпису осі графіка. За замовчуванням встановлено стан «on»: підпис виводиться вниз, по центру осі. Проте положення підпису можна визначити і самостійно, для цього в поле **Position** задаються координати вектором  $[x, y]$ . У цьому разі перемикач **Auto position** автоматично перейде в стан «off»;

**Auto Rotation** — режим автоматичного обертання підписи осі. За замовчуванням цей режим вимкнений (стан перемикача «off»);

**Font angle** — кут повороту підпису осі. Можна встановити одне з запропонованих значень: 0, 90, 180 і 270 градусів, а також будь-який довільний кут повороту підпису в останньому полі ;

**Fore / Back colors** — колір символів і колір фону підпису осі, які встановлюються за допомогою повзунка, кожному положенню якого відповідає певний колір. Усього доступно 35 кольорів;

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

**Font style** — стиль накреслення символів підписи осі. За замовчуванням встановлений стиль **Helvetica** (рубаний).

В області **Axis Options** можна змінити:

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

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

**Data bounds** — обмеження даних. Для кожної осі можна зменшити діапазон вихідних даних, за якими формується графік, зробивши його більш детальним;

**Scale** — масштаб осі графіка. Існує два автоматичні режими: **lin** (лінійний) і **log** (логарифмічний). Натискання кнопки **Ticks** (Засічки) призводить до появи вікна модифікації поділу осі **Edit Axes Ticks**;

**Sub ticks** — проміжні засічки. У цьому полі потрібно ввести число засічок, які будуть виводитися між основними поділами осі. Слід зазначити, що проміжні засічки не підписуються. У вікні **Edit Axes Ticks** формується таблиця основних засічок (без засічок **Sub ticks**). Перший стовпець **Locations** задає положення засічки, а другий **Labels** — підпис засічки. Для зручності редагування таблиці вікно забезпечено кнопками **Insert**, **Delete**, **Apply**, **Quit**. Кнопка **Insert** дає змогу вставити у вікно готову таблицю засічок (або її фрагмент) за допомогою буфера обміну. Вставка здійснюється з позиції активної клітинки. Кнопка **Delete** видаляє не лише активну клітинку, але і весь рядок, якому вона належить. Кнопка **Apply** підтверджує зміни, а **Quit** служить для виходу з вікна **Edit Axes Ticks**;

Остання опція на закладці **X** — це перемикач **Reverse**. Якщо встановити його в стан «**on**», то графік дзеркально відобразиться щодо осі **Y**. Якщо ж увімкнути цей режим на закладці **Y**, то графік буде дзеркально відображений відносно осі **X**.

Закладка **Title** вікна форматування осей **Axes Editor** призначена для зміни властивостей назви графіка. Вона містить лише одну область — **Label Options**, ідентичну області **Label Options** закладок **X**, **Y** і **Z**.

Проілюструємо можливості зміни властивостей координатних осей графіка за допомогою закладок **X**, **Y**, **Title** вікна форматування осей **Axes Editor**. Переформатуємо графік із попереднього прикладу (рис. 20) таким чином:

виведемо підписи для осі **X** «Вісь X» і для осі **Y** «Вісь Y»;

встановимо для підписів стиль шрифту **Monospaced** і розмір 3;

для обох осей повзунок **Grid Color** встановимо в положення 1;

визначимо для осі **X** розміщення **middle**, а на закладці **Y** увімкнемо режим

**Reverse**;

Внаслідок проведених маніпуляцій отримаємо таке графічне вікно (рис. 1.22)

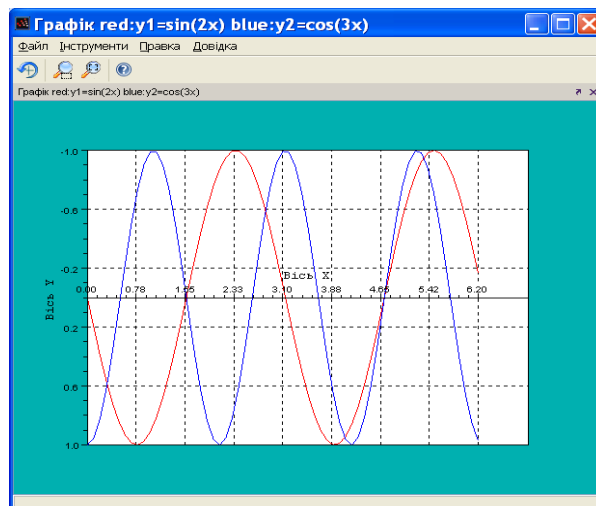


Рис. 1.22.

Наступна закладка — **Style** вікна форматування осей графіка **Axes Editor** надає можливість змінювати наступні властивості лінії осі і підписів засічок:

**Visibility** ( відображення) — перемикач, який перебуває в стані «**on**» (за замовчуванням) або «**off**». В стані «**off**» графік взагалі не виводиться у вікно;

**Font style** — стиль накреслення символів підписів зарубок на осі. За замовчуванням встановлений стиль **Helvetica**;

**Font color** — повзунок, кожне положення якого визначає колір символів підписів засічок. За замовчуванням встановлений в положенні -1 — чорний колір;

**Font size** — розмір символів підписів засічок на осі, можливі значення від 0 до 6. За замовчуванням для шрифту встановлений розмір 1;

**Fore. color** — повзунок, кожне положення якого визначає колір власне координатної осі. За замовчуванням встановлений в положенні -1 — чорний колір;

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

**Thickness** — товщина лінії координатної осі, зумовлена повзунком з положеннями від 1 до 30. За замовчуванням для товщини лінії встановлено значення 1;

**Line style** — стиль накреслення лінії. Є шість режимів: **solid** — суцільна лінія, інші режими — варіації пунктирної лінії;

Закладка **Aspect** вікна форматування **Axes Editor** дає змогу змінювати наступні властивості:

**Auto clear** — якщо перемикач встановлений в положення «**on**», графічне вікно буде автоматично очищатися щоразу перед побудовою нового графіка. Якщо ж цей режим вимкнений (за замовчуванням), графіки будуть накладатися в одних координатних осях відповідно до режиму **Auto scale**;

**Auto scale** — режим поновлення меж координатних осей графіка. В положенні перемикача «**on**» (за замовчуванням) новий графік змінить межі попереднього графіка, щоб сформуватися на всьому заданому інтервалі, але в тому ж масштабі, що і попередній графік. У вимкненому режимі **Auto scale** новий графік буде побудований в межах осей попереднього графіка і, можливо, буде відображати лише частину заданого інтервалу.

Перемикач **Boxed** на закладці **Aspect** вікна форматування **Axes Editor** визначає, обмежувати графік прямокутною рамкою (положення «**on**» за замовчуванням) чи виводити лише координатні осі (положення «**off**»).

**Isoview** — цю властивість використовують для того, щоб встановити однаковий масштаб для всіх осей графіка. За умовчанням встановлено стан перемикача «**off**».

**Tight limits** — якщо цей режим увімкнений, осі графіка змінюються таким чином, щоб точно відповідати значенню властивості **Data bounds** закладок **X**, **Y** і **Z**. При значенні «**off**» (за замовчуванням) осі можуть збільшити вихідний інтервал, щоб було простіше вибрати масштаб осі і нанести на неї засічки.

**Cube scaling** — ця опція застосовна лише до тривимірних графіків. У стані перемикача «**on**» вихідні дані обмежуються так, щоб поверхня помістилася в куб розміром 1. Це дає змогу наочніше зобразити 3D-графік у тих випадках, коли масштаб координатних осей надто відрізняється від однієї осі до іншої. За замовчуванням встановлено стан перемикача «**off**».

**Clip state** — режим кадрування (обрізки) графіка. Можливий один із наступних станів перемикача: «**off**» означає, що створюване зображення не кадрується; «**clipgrf**» (за замовчуванням) — від створюваного зображення обрізається область, що знаходиться поза межами осей; «**on**» — від створюваного зображення обрізається область, що знаходиться поза межами, заданих властивістю **Clip box**.

**Clip box** — прямокутна область, яка буде відображатися після обрізки зображення. Спочатку в полях **X** і **Y** задають координати верхньої лівої точки прямокутника (**upper-left point coordinates**), потім ширина і висота - поля **W** і **H**.

**Margins** — ця властивість встановлює відстань від межі графічного вікна до області графіка: **Left** (лівий край), **Right** (правий), **Top** (верхній), **Bottom** (нижній). Значення має перебувати в інтервалі [0: 1]. За замовчуванням кожному полю присвоєно значення 0.125.

**Axes bounds** — ця властивість задає частину графіка, яка буде виводитися в координатних осях. **Left** і **Up** визначають положення верхнього лівого кута, **Width** і **Height** — ширину і висоту фрагмента графіка. Значення має знаходитися в інтервалі [0: 1]. За замовчуванням відображуваний фрагмент задається матрицею  $\begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$ .





**Mark mode** — режим, що дозволяє будувати точкові графіки (положення перемикача «on»). За замовчуванням цю властивість вимкнено.

**Mark style** — стиль маркера (можливі такі значення: **dot** — крапка, **plus** — знак «плюс», **cross** — хрестик, **star** — плюс, вписаний в коло. **filled diamond** — зафарбований ромб, **diamond** — ромб, **triangle up** - трикутник вершиною вгору, **triangle down** — трикутник вершиною вниз, **diamond plus** — плюс, вписаний в ромб, **circle** — коло, **asterisk** — зірочка, **square** — квадрат, **triangle right** — трикутник вершиною вправо, **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.

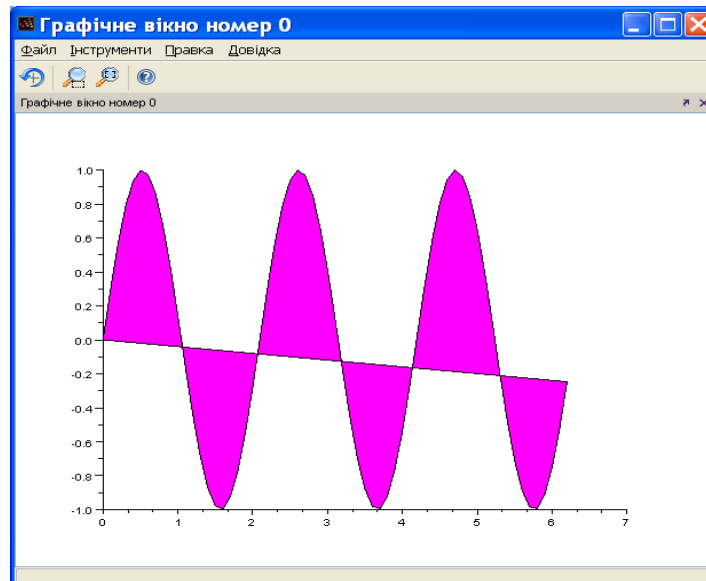


Рис. 1.24. Графік функції  $y = \sin(3x)$  зі значенням **Foreground 1**, **Background 6**, увімкненим перемикачем **Fill mode** і режимом **Closed**

На рис. 1.25 зображений точковий графік функції  $y = \sin(3x)$  з типом маркера **filled diamond**, розмір маркера 1, значення кольору заливки маркера **Mark foreground 5**.

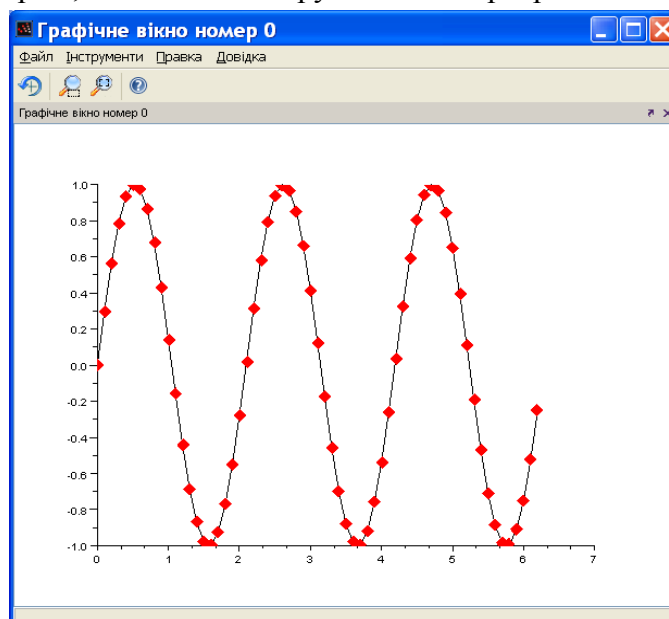


Рис. 1.25.

Закладка **Data** вікна форматування **Polyline Editor** дає змогу уточнити область даних, за якими будується графіка. У випадковому списку **Data field** вказується поточний діапазон (два масиви типу double, в кожному з них 63 значення - [63x2 double array]). Якщо ж у цьому списку можна вибрати рядок **Edit data**, то можна відредагувати таблицю вихідних даних.

Закладка **Clipping** (Обрізка) вікна форматування **Polyline Editor** дає змогу встановити межі прямокутної області — **Clip box** (Кадр), яка залишиться видимою після обрізки зображення.

Нагадаємо, що в полях **X** і **Y** слід вказати  $x$ ,  $y$  координати верхнього лівого кута кадру, а в полях **W**, **H** — його ширину і висоту. Режим **Clip state** може набувати одного із трьох станів: «**off**» — означає, що створюваний графік некадрований; «**clipgrf**» (за замовчуванням) — від створюваної графіки обрізається область, яка знаходиться поза межами осей; «**on**» — від створюваної графіки обрізається область, яка знаходиться поза межами, заданих властивістю **Clip box**.

Більш докладно дії з **Figure Editor** описані в [1] та довідці **Scilab** (graphic objects editor).

### 1.1.10. Побудова тривимірних графіків в **Scilab**

Окрім засобів для побудови двовимірних графіків **Scilab** має кілька функцій для побудови тривимірних графіків. Загалом процес побудови графіка функції  $z(x, y)$  можна розділити на три етапи:

1. Створення в області побудови графіка прямокутної сітки шляхом проведення відрізків ліній  $x_i$  та  $y_j$ , які паралельні координатним осям  $OX$ ,  $OY$

$$x_i = x_0 + i h, h = \frac{x_n - x_0}{n}, i = 1, 2, \dots, n$$

$$y_j = y_0 + j d, d = \frac{y_n - y_0}{k}, j = 1, 2, \dots, k$$

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

3. Звертання до функції побудови тривимірних графіків.

#### Функції **plot3d** і **plot3d1**

Функції **plot3d** і **plot3d1** дають змогу побудувати поверхню за масивом значень функції  $z(x, y)$ , отриманим на прямокутній сітці в площині значень  $x, y$ . Функція **plot3d** будує поверхню і заливає її одним кольором, а **plot3d1** — поверхню, кожна комірка якої має колір, який залежить від значення функції у відповідному вузлі сітки.

Звертання до цих функцій таке:

**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 значень змінної  $x$ ; **y** — вектор значень змінної  $y$ ; **z** — матриця значень функції; **theta**, **alpha** — числа, які задають в сферичних координатах кут огляду побудованої поверхні; **leg** — підписи координатних осей графіка (символи, які розділені знаком @);

**flag** — масив, із трьох цілих чисел [**mode**, **type**, **box**], параметр **mode** — задає колір поверхні та наявність/відсутність сітки на ній, **type** — задає масштаб графіка, **box** — задає рамку довкола відображуваної поверхні; **ebox** — вектор [**xmin**, **xmax**, **ymin**, **ymax**, **zmin**, **zmax**], який задає межі області, в яку буде виведена поверхня (цей параметр можна використовувати лише для **type=1**); **keyn=valuen** — послідовність значень **key1=value1**, **key2=value2**, ..., **keyn=valuen**, які задають товщину лінії, її колір, колір фону графічного вікна, наявність маркера тощо (див. опис функції **plot2d**). Як приклад розглянемо побудову поверхні (рис. 1.26)

$$z = 5y^2 - x^2, x \in [-2; 2], y \in [-3; 3]$$

на сітці з кроком 0.1.

```
-->x=[-2:0.1:2];
-->y=[-3:0.1:3];
-->for i=1:length(x) // цикл по i
-->for j=1:length(y) // цикл по j
-->z(i,j)=5*y(j)^2-x(i)^2; // заповнення матриці відліків функції
-->end
-->end
-->plot3d1(x',y',z,-125,51); // відображення поверхні
```

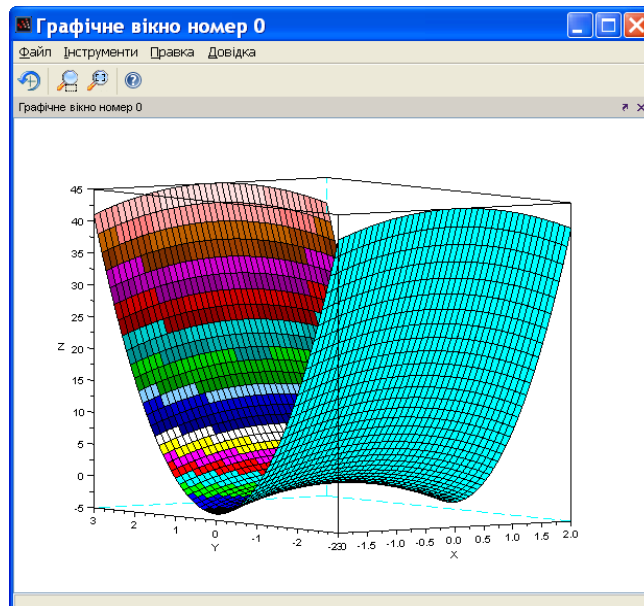


Рис. 1.26.

У цьому прикладі для отримання матриці відліків функції використано вкладені цикли по **i** (для доступу до елементів вектора **x**) та **j** (для доступу до елементів вектора **y**).

Для створення прямокутної сітки і заповнення матриці відліків функції можна використати функцію **eval3dp** (див. довідку системи **Scilab**).

### Функції **meshgrid**, **surf** та **mesh**

Для побудови прямокутної сітки можна використати функцію **meshgrid**

**[x y]=meshgrid(-x<sub>поч</sub>:крок:x<sub>кінець</sub>, -y<sub>поч</sub>:крок:y<sub>кінець</sub>)**

тут **x,y** — вектори значень змінних **x** та **y**. Після створення прямокутної сітки для заповнення матриці відліків функції використовують оператор присвоєння

**z=вираз x. від та y.;**

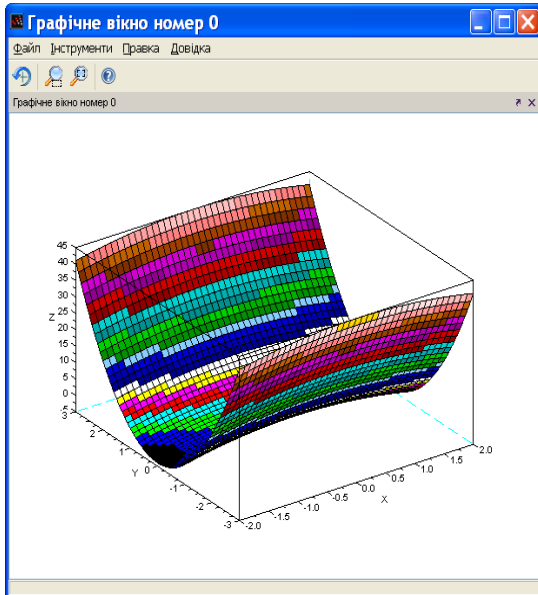
Для побудови поверхні використовують функцію **surf(x,y,z)** або **mesh(x,y,z)**. Як приклад розглянемо побудову поверхні  $z = 5y^2 - x^2$  спочатку за допомогою **surf** (див. Рис. 1.27а) а потім **mesh** (див. Рис. 1.27б).

Приклад 1.

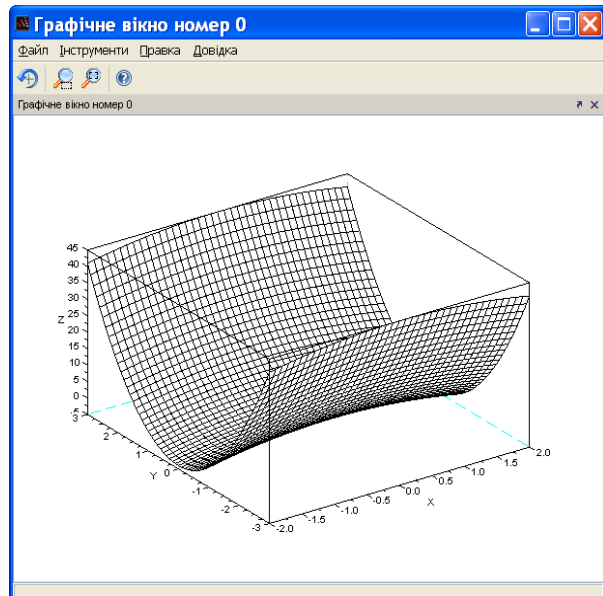
```
[x y]=meshgrid(-2:0.1:2,-3:0.1:3); // побудова прямокутної сітки
z=5*y.^2-x.^2; // заповнення матриці відліків функції
surf(x,y,z) // побудова поверхні
```

Приклад 2.

```
[x y]=meshgrid(-2:0.1:2,-3:0.1:3); // побудова прямокутної сітки
z=5*y.^2-x.^2; // заповнення матриці відліків функції
mesh(x,y,z) // побудова поверхні
```



а



б

Рис. 1.27. а — для побудови поверхні використана функція **surf(x,y,z)**, б — для побудови поверхні використана функція **mesh(x,y,z)**

### Побудова параметричних кривих в тривимірному просторі

Для побудови параметричних кривих в **Scilab** використовують функцію **param3d(x, y, z, [theta, alpha, leg, flag, ebox])**.

тут **x, y, z** — відліки координат точок параметричної кривої; **theta, alpha** — кут спостереження в сферичних координатах; **leg** — підписи координатних осей графіка (символи, відокремлювані знаком @, наприклад, 'X @ Y @ Z'); **flag** — масив, який містить три цілочисельні параметри: [**mode, type, box**]: **mode** — встановлює спосіб і місце нанесення ліній рівня, **type** — задає масштаб графіка, **box** — задає наявність/відсутність рамки навколо відображуваного графіка.

Розглянемо приклад побудови параметричної кривої  $x = t \cdot \sin(t)$ ;  $y = t \cdot \cos(t)$ ;  $z = t \cdot |t|$  з кутами спостереження 45 і 60 градусів.

```
t=-50*%pi:0.1:50*%pi;
x=t.*sin(t);
y=t.*cos(t);
z=t.*abs(t)/(50*%pi);
param3d(x,y,z,45,60);
```

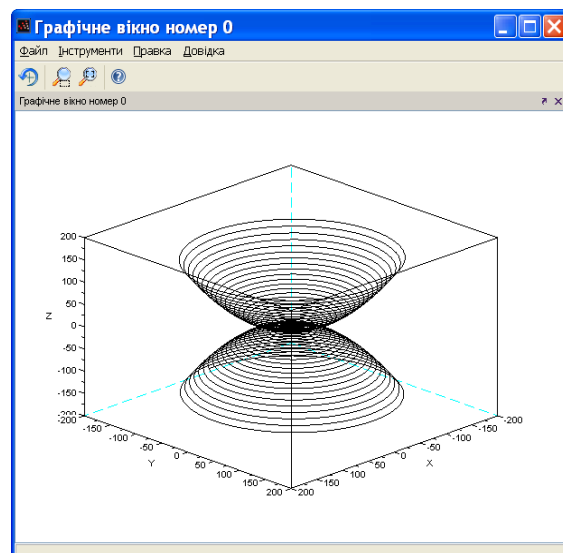


Рис. 1.28.

Окрім розглянутих функцій для побудови тривимірних графіків можна використати функції **plot3d2**, **plot3d3**, **param3d1**, **contour**, **contourf**, **hist3d**, які описані в [1] та довідковій системі **Scilab**.

### 1.1.11. Розв'язання нелінійних рівнянь

Рівняння  $f(x) = 0$ , в якому невідоме є аргументом трансцендентних функцій, називають трансцендентним рівнянням. До трансцендентних рівнянь належать показникові, логарифмічні і тригонометричні. У загальному випадку аналітичний розв'язок рівняння  $f(x) = 0$  можна знайти лише для вузького класу функцій. Найчастіше доводиться розв'язувати це рівняння чисельними методами. Для цього спочатку відокремлюють корені рівняння, тобто знаходять досить малі проміжки, в яких міститься лише один корінь. Визначити їх можна шляхом побудови графіка функції  $f(x)$ . Після цього знаходять корені із заданою точністю за допомогою функції **fsolve(x0,f)**, тут **x0** — початкове наближення, **f** — функція, яка описує ліву частину рівняння  $f(x) = 0$ . Розглянемо приклад:

```
0.2e^x - 2(x - 1)^2 = 0
-->deff('[y]=f(x)','y=exp(x)/5-2*(x-1)^2');
-->x=0:0.1:6;
-->y=f(x); // вибір нульового наближення
-->plot(x,y); // з графіка функції f(x)
-->x(1)=fsolve(0,f); x(2)=fsolve(2,f); x(3)=fsolve(5,f);
-->x
x =
    0.5778406
    1.7638701
    5.1476865
```

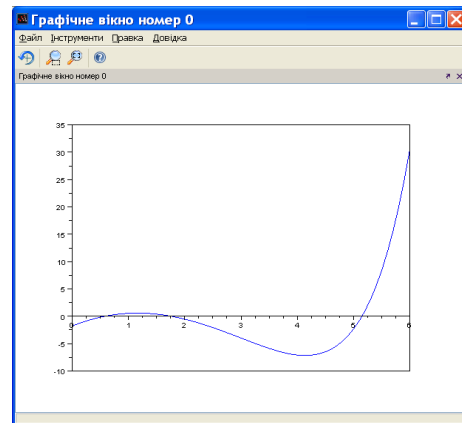


Рис. 1.29.

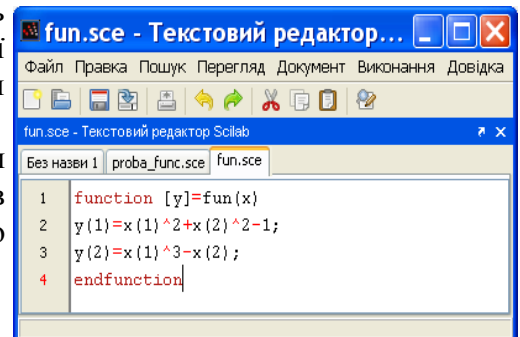
Функцію також використовують для розв'язання систем нелінійних рівнянь. Розглянемо два приклади.

Приклад 1.  $x^2 + y^2 = 1$ ;  $x^3 - y = 0$

Ця система рівнянь має два корені, які лежать на перетині кола одиничного радіуса і кубічної параболи, тому як нульове наближення можна вибрати вектори  $[0.5 \ 0.5]$  та  $[-0.5 \ -0.5]$

Користуючись вбудованим текстовим редактором створимо функцію і збережемо її в підкаталозі **scilab-5.2.2** з іменем **fun.sce**. Завантажимо створену функцію через меню **Scilab** (Файл — Виконати) і після цього виконаємо команди

```
-->fsolve([0.5 0.5],fun)
ans =
    0.8260314    0.5636242
-->p=fun([0.8260314 0.5636242])// перевірка отриманого розв'язку
p =
    0.0000001
    4.884D-08
-->fsolve([-0.5 -0.5],fun)
ans =
    - 0.8260314    - 0.5636242
-->p=fun([-0.8260314 -0.5636242])// перевірка отриманого розв'язку
p =
    0.0000001
    - 4.884D-08
```



### 1.1.12. Згладжування експериментальних даних

Через присутність в результатах експерименту шумів вимірювання (завади, похибки дискретизації) для визначення числових параметрів теоретичних моделей використовують згладжування (апроксимацію) експериментальних даних методом найменших квадратів. Ідея згладжування методом найменших квадратів полягає в тому, що числові параметри теоретичної моделі підбирають таким чином, щоб сума квадратів відхилень експериментальних точок від теоретичної кривої була найменшою

$$S = \sum_{i=1}^n (y_i - f(x_i, a_0, a_1, \dots, a_k))^2 \rightarrow \min$$

тут  $y_i(x_i)$  — результати експерименту,  $f(x, a_0, a_1, \dots, a_k)$  — теоретична модель,  $a_0, a_1, \dots, a_k$  — числові параметри теоретичної моделі. Залежно від вигляду теоретичної моделі згладжування методом найменших квадратів зводиться до Розв'язання системи лінійних або нелінійних рівнянь відносно коефіцієнтів  $a_0, a_1, \dots, a_k$ .

Для проведення згладжування результатів експерименту можна використати функцію  
**[a,S]=datafit(F,z,c)**

тут **F** — функція, параметри якої необхідно підібрати; **z** — матриця результатів експерименту; **c** — вектор початкових наближень; **a** — вектор коефіцієнтів; **S** — сума квадратів відхилень виміряних значень від розрахункових.

Розглянемо задачу про згладжування експериментальної залежності споживаної асинхронним двигуном потужності з кола змінного струму апроксимуючим виразом

$$P = a_1 + a_2 U + a_3 U^2 + a_4 U^3$$

<b>U (В)</b>	132	140	150	162	170	180	190	200	211	220	232	240	251
<b>P (Вт)</b>	330	350	385	425	450	485	540	600	660	730	920	1020	1350

```
//Функція, яка обчислює різницю між експериментальними
//і теоретичними значеннями.
function [zr]=G(c,z)
zr=z(2)-c(1)-c(2)*z(1)-c(3)*z(1)^2-c(4)*z(1)^3
endfunction
//Результати експерименту
-->x=[1.32 1.40 1.50 1.62 1.70 1.80 1.90 2.00,2.11,2.20,2.32,2.40,2.51];
-->y=[3.30 3.50 3.85 4.25 4.50 4.85 5.40 6.00 6.60 7.30 9.20 10.20 13.50];
//Формування матриці з результатами експерименту
-->z=[x;y];
//Вектор початкових наближень апроксимуючого виразу
-->c=[0;0;0;0];
//Розв'язання задачі
[a,err]=datafit(G,z,c)
-->[a,err]=datafit(G,z,c)
err =
    0.5287901
a =
    - 51.576664
    95.594671
    - 55.695312
    11.111453
//Вивід на графік результатів експерименту
plot2d(x,y,-4);
```

```
//Побудова графіка апроксимуючої функції
t=1.32:0.01:2.51;
Ptc=a(1)+a(2)*t+a(3)*t^2+a(4)*t^3;
plot2d(t,Ptc);
```

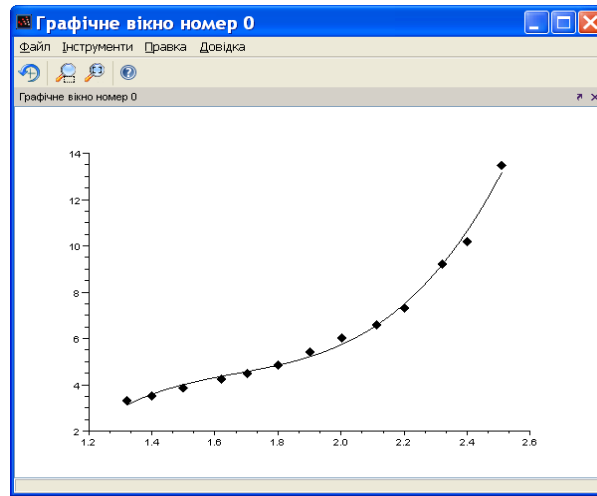


Рис. 1.30.

### 1.1.13. Інтерполяція табличних даних

В деяких випадках функціональні залежності задають таблицями. Коли значення аргумента функції співпадає з одним із вузлів таблиці, то значення функції береться із відповідного аргументу вузла таблиці. Коли ж значення аргумента функції є в проміжку між двома вузлами таблиці, то для обчислення значення функції використовують інтерполяцію (кусково-лінійна, сплайн-інтерполяція, інтерполяційний поліном Лагранжа тощо).

В Scilab для побудови лінійної інтерполяції використовують функцію

$$y = \text{interp}(z, x)$$

тут  $z$  - матриця вихідних даних;  $x$  - вектор абсцис;  $y$  - вектор значень лінійного виразу в точках  $x$ .

Розглянемо задачу кусково-лінійної інтерполяції характеристики терморпарі платиновордій-платина. Для розв'язання задачі використаємо другий рядок градуювальної таблиці терморпарі платиновордій-платина (тип S). У цій задачі потрібно за значенням термоелектрорушійної сили визначити різницю температур між “холодним” і “гарячим” спаєм. (<http://www.temperatures.ru/sprav/sprav1.php?page=6>)

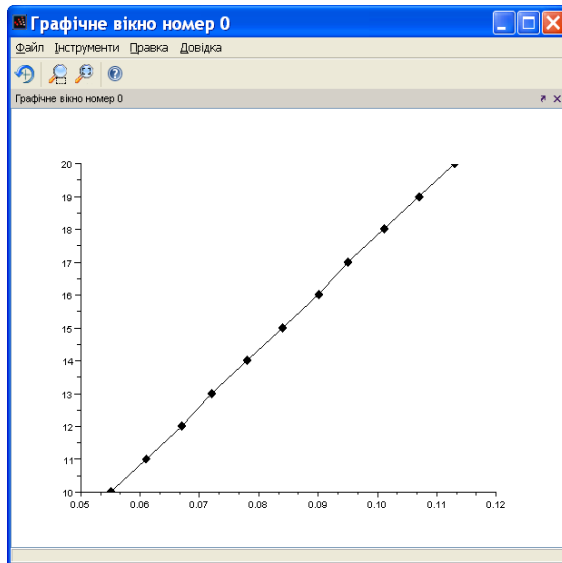
°C	0	1	2	3	4	5	6	7	8	9	10
ТермоЕРС, мВ											
0	0.000	0.005	0.011	0.016	0.022	0.027	0.033	0.038	0.044	0.050	0.055
10	0.055	0.061	0.067	0.072	0.078	0.084	0.090	0.095	0.101	0.107	0.113
20	0.113	0.119	0.125	0.131	0.137	0.143	0.149	0.155	0.161	0.167	0.173
30	0.173	0.179	0.185	0.191	0.197	0.204	0.210	0.216	0.222	0.229	0.235
40	0.235	0.241	0.248	0.254	0.260	0.267	0.273	0.280	0.286	0.292	0.299
50	0.299	0.305	0.312	0.319	0.325	0.332	0.338	0.345	0.352	0.358	0.365
60	0.365	0.372	0.378	0.385	0.392	0.399	0.405	0.412	0.419	0.426	0.433
70	0.433	0.440	0.446	0.453	0.460	0.467	0.474	0.481	0.488	0.495	0.502
80	0.502	0.509	0.516	0.523	0.530	0.538	0.545	0.552	0.559	0.566	0.573
90	0.573	0.580	0.588	0.595	0.602	0.609	0.617	0.624	0.631	0.639	0.646

Приклад:

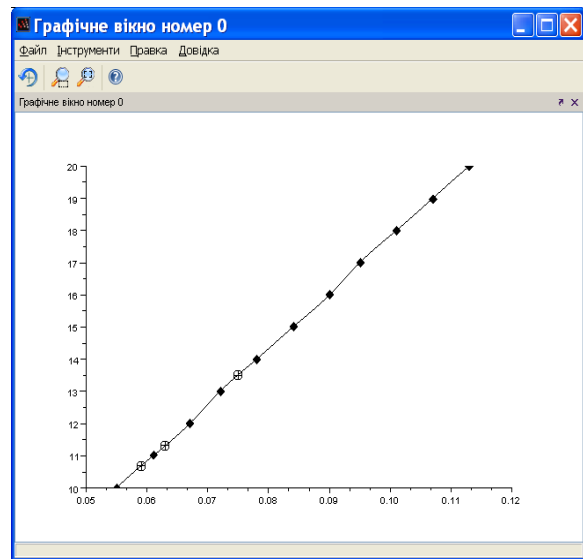
```
-->v=[0.055 0.061 0.067 0.072 0.078 0.084 0.090 0.095 0.101 0.107 0.113];
```



```
-->t=[10 11 12 13 14 15 16 17 18 19 20];
-->plot2d(v,t,-4);
-->z=[v;t];
-->u=0.055:0.002:0.113;
-->ptd=interpLn(z,u);
-->plot2d(u,ptd);
```



а



б

Рис. 1.31. а — результати кусково-лінійної інтерполяції, б — результати інтерполяції кубічним сплайном

Побудова кубічного сплайна в **Scilab** складається з двох етапів: спочатку обчислюють коефіцієнти сплайна за допомогою функції **d = splin (x, y)**, а потім обчислюються значення інтерполяційного полінома для значень незалежної змінної, вписаних у вектор **t**

$$z = \text{interp} (t, x, y, d)$$

Функція **d = splin (x, y)** має такі параметри: **x** — строго зростаючий вектор, який має щонайменше дві компоненти; **y** — вектор того ж формату, що й **x**; **d** — вектор із коефіцієнтами кубічного сплайна.

```
-->t=[10 11 12 13 14 15 16 17 18 19 20];
-->v=[0.055 0.061 0.067 0.072 0.078 0.084 0.090 0.095 0.101 0.107 0.113 ];
-->plot2d(v,t,-4);//Графік табличних даних
-->koeff=splin(v,t);
-->X=[0.059 0.063 0.075];
-->Y=interp(X,v,t,koeff) //Значення функції в заданих точках
Y =
    10.688088    11.311912    13.519491
plot2d(X,Y,-3); //Нанесення точок на графік
//Обчислення значень кубічного сплайна в діапазоні термоЕРС від 0.055мВ до 0.113мВ
u=0.055:0.001:0.113;
-->ptd=interp(u,v,t,koeff);
-->plot2d(u,ptd);
```

#### 1.1.14. Чисельне інтегрування дифрівнянь

Для чисельного інтегрування дифрівнянь в **Scilab** можна використати функцію

$$[Y, w, iw] = \text{ode} ([\text{type}], y0, t0, t [, rtol [, atol]], f [, jac] [, w, iw])$$

для якої обов'язковими входними параметрами є: **y0** — вектор початкових умов; **t0** — початкова точка інтервалу інтегрування; **t** — координати вузлів сітки, в яких відбувається пошук розв'язку; **f** — зовнішня функція, яка задає праву частину дифрівняння або системи дифрівнянь; **y** — вектор розв'язків дифрівняння.

Таким чином, для того щоб розв'язати звичайне дифрівняння

$$\frac{dy}{dt} = f(y, t)$$

потрібно викликати функцію **y = ode (y0, t0, t, f)**.

Розглянемо необов'язкові параметри функції **ode**:

**type** — параметр, за допомогою якого можна вибрати числовий метод інтегрування або тип розв'язуваної задачі, вказавши один із рядків:

**adams** — застосовують для розв'язування дифрівнянь або систем дифрівнянь методом прогнозу-корекції Адамса;

**stiff** — вказують для інтегрування жорстких дифрівнянь;

**rk** — використовують для інтегрування дифрівнянь або дифрівнянь систем методом Рунге-Кутти четвертого порядку;

**rkf** — вказують у разі вибору п'ятиетапного методу Рунге-Кутти четвертого порядку;

**fix** — метод Рунге-Кутти з фіксованим кроком;

**rtol**, **atol** — відносна і абсолютна похибки обчислень, вектор, розмірність якого співпадає з розмірністю вектора **y**, за замовчуванням **rtol** = 0.00001, **atol** = 0.0000001, у разі використання параметрів **rkf** і **fix** — **Rtol** = 0.001, **atol** = 0.0001;

**jac** — матриця, яка є якобіаном правої частини жорсткої системи дифрівнянь, задають матрицю зовнішньою функцією **J = jak (t, y)**;

**w**, **iw** — вектори, призначені для збереження інформації про параметри інтегрування, які застосовують для того, щоб наступні обчислення виконувалися з тими ж параметрами. Розглянемо кілька прикладів застосування функції **ode**.

Приклад 1. Вільні коливання фізичного маятника

Малі вільні коливання фізичного маятника описуються дифрівнянням

$$\frac{d^2 x}{dt^2} + 0.9 \frac{dx}{dt} + x = 0$$

Перетворимо дифрівняння другого порядку до системи дифрівнянь першого порядку

$$\frac{dx}{dt} = y;$$

$$\frac{dy}{dt} = -0.9y - x$$

Слід наголосити, що методи чисельного інтегрування дифрівнянь дають змогу отримати лише часткові розв'язки. Задамо початкові умови

$$x(0) = 0; y(0) = 5.$$

Складемо сценарій чисельного інтегрування рівняння вільних коливань фізичного маятника. Для задання системи дифрівнянь першого порядку використаємо функцію **syst(t,x)**

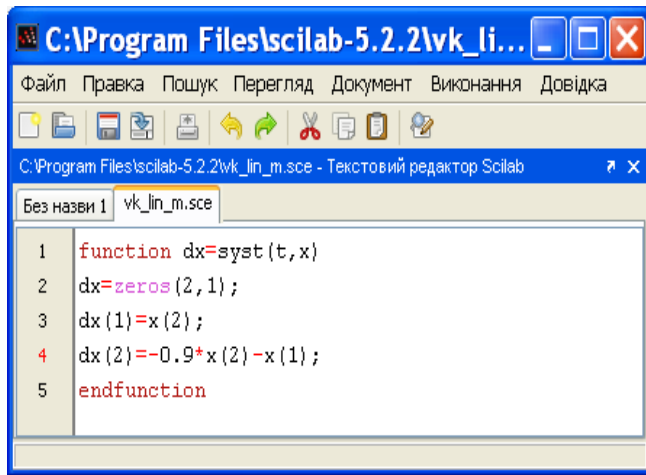


Рис. 1.32. Створення функції  $\text{syst}(t,x)$  за допомогою вбудованого текстового редактора

```

//Інтегрування системи дифрівнянь, заданої функцією syst(t,x)
-->x0=[0;5];//Задання початкових умов
-->t0=0;
-->t=0:0.01:10;//Задання проміжка інтегрування
-->x=ode(x0,t0,t,syst);
//Побудова графіків розв'язку дифрівняння
-->plot(t,x)

```

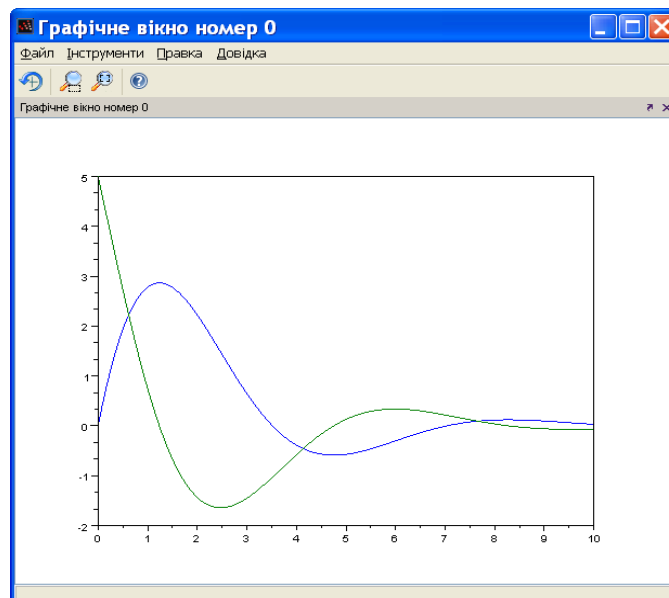


Рис. 1.33. Графіки розв'язку дифрівняння: синій -  $x(t)$  , зелений -  $\frac{dx(t)}{dt}$

## Приклад 2. Вимушені коливання лінійної коливної системи

Розглянемо вимушені коливання лінійної коливної системи з одним ступенем вільності, які описуються дифрівнянням

$$\frac{d^2 x}{dt^2} + 2 \frac{dx}{dt} + x = \sin(5t)$$

Перетворимо дифрівняння другого порядку до системи дифрівнянь першого порядку

$$\frac{dx}{dt} = y;$$

$$\frac{dy}{dt} = -2y - x + \sin(5t)$$

Оскільки нас цікавить періодичний розв'язок, то початкові умови можна задати будь-які, наприклад  $x(0)=0.5$ ;  $y(0)=0$ . Складемо сценарій чисельного інтегрування рівняння вільних коливань фізичного маятника. Для задання системи диференціальних рівнянь першого порядку використаємо функцію `syst(t,y)`

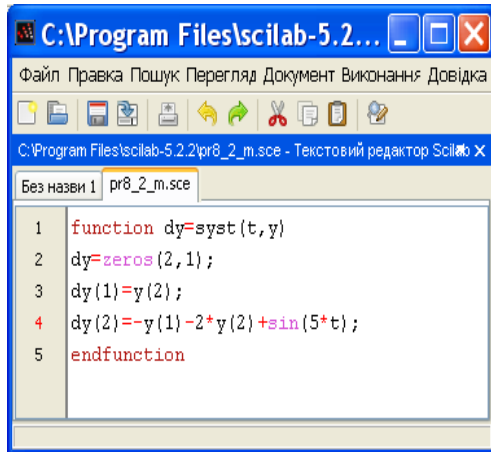


Рис. 1.33. Функція `syst(t,y)`

```

//Інтегрування системи диференціальних рівнянь, заданої функцією syst(t,y)
-->x0=[0.5;0]; //Задання початкових умов
-->t0=0;
-->t=0:0.05:25; //Задання проміжку інтегрування
-->y=ode(x0,t0,t,syst);
//Побудова графіків розв'язку диференціальних рівнянь
-->plot(t,y)

```

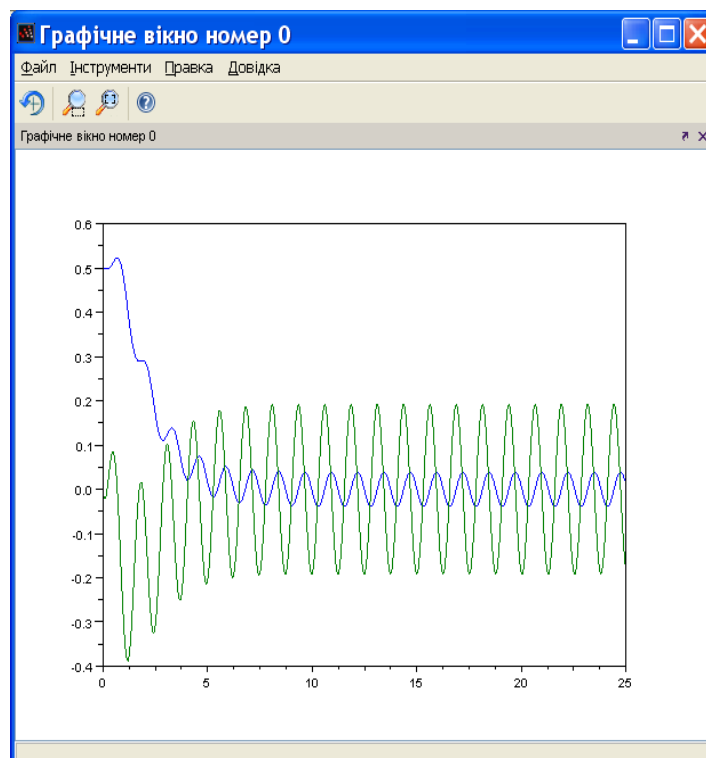


Рис. 1.34. Вимушені коливання лінійної коливної системи:

синій -  $x(t)$  , зелений -  $\frac{dx(t)}{dt}$

### Приклад 3. Система з автоколиваннями (генератор Ван-дер-Поля)

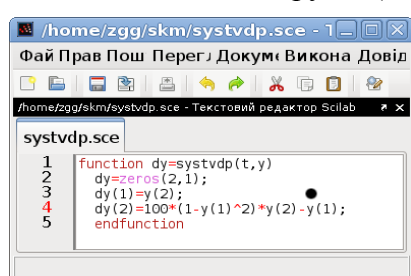
$$\frac{d^2 x}{dt^2} - \epsilon(1-x^2) \frac{dx}{dt} + x = 0, \epsilon > 0$$

Перетворимо диференціальне рівняння другого порядку до системи диференціальних рівнянь першого порядку

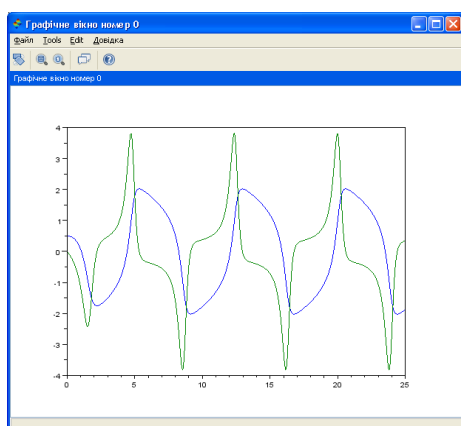
$$\frac{dx}{dt} = y;$$

$$\frac{dy}{dt} = \epsilon(1-x^2)y - x$$

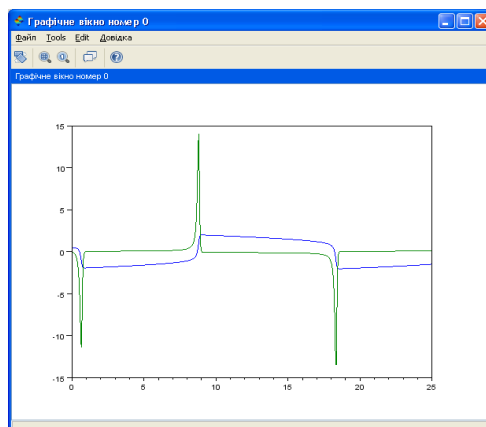
Оскільки нас цікавить періодичний розв'язок, то початкові умови можна задати будь-які, наприклад  $x(0)=0.5; y(0)=0$ . Складемо сценарій чисельного інтегрування рівняння коливань генератора Ван-дер-Поля. Для задання системи диференціальних рівнянь першого порядку використаємо функцію `systvdp(t,y)`. Випадок  $\epsilon=10$  відповідає системі з чергуванням швидких і повільних рухів (жорстка система)



```
//Інтегрування системи диференціальних рівнянь, заданої
// функцією systvdp(t,y)
-->x0=[0.5;0]; //Задання початкових умов
-->t0=0;
-->t=0:0.05:25; //Задання проміжка інтегрування
-->y=ode(x0,t0,t,systvdp);
//Побудова графіків розв'язку диференціальних рівнянь
-->plot(t,y)
```



а



б

Рис. 1.35. Автоколивання генератора Ван-дер-Поля  
а -  $\epsilon=2$ , б -  $\epsilon=10$  (так звана “жорстка” система)

#### 1.1.15. Програмування в Scilab

В **Scilab** вбудована потужна мова програмування з підтримкою об'єктів. У цьому розділі розглянемо можливості структурного програмування, наступний розділ присвячений візуальному програмуванню в середовищі **Scilab**.

Як вже зазначалося раніше, 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) необхідно:

1. Викликати команду Editor з меню (див. 4. Вбудовані функції та функції користувача в Scilab).
2. У вікні редактора **Scipad** набрати текст програми.
3. Зберегти текст програми за допомогою команди **File - Save** у файлі з розширенням `sce`, наприклад, `file.sce`.

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

Програмний режим досить зручний, оскільки він дає змогу зберегти розроблений алгоритм у файлі і повторювати його для інших вихідних даних в інших сесіях. Крім звернень до функцій і операторів присвоювання, у програмних файлах можуть використовуватися оператори мови програмування **Scilab** (мову програмування **Scilab** будемо називати **sci-** мовою). Розглянемо основні оператори **sci-**мови.

### Функції вводу-виводу в Scilab

Для організації найпростішого вводу даних в Scilab можна скористатися функціями

```
x = input ('title');  
або  
x = x_dialog ('title', 'rjadok');
```

Функція **input** виводить в командному рядку **Scilab** підказку **title** і чекає, поки користувач введе значення, яке передається змінній **x**. Функція **x\_dialog** виводить на екран діалогове вікно з ім'ям **title**, після чого користувач може натиснути ОК, і тоді **rjadok** буде передано змінній **x**, або ввести нове значення замість **rjadok**, і це значення буде передано змінній **x**.

На рис. 1.36 зображене діалогове вікно, яке формується рядком **x = x\_dialog ('Input X','5 ')**.

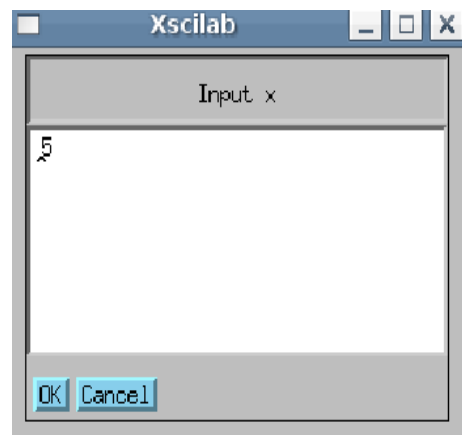


Рис. 1.36. Вікно вводу

Функція **input** перетворює введені значення до числового типу даних, а функція **x\_dialog** повертає рядкове значення. Тому для вводу числових значень потрібно використати перетворення рядкового значення в число функцію **evstr**:

```
x = evstr (x_dialog ('title', 'rjadok'));
```

Для виводу в текстовому режимі можна використовувати функцію **disp**:

```
disp (b)
```

тут **b** - ім'я змінної або виділений лапками текст.

## Оператор присвоювання

Оператор присвоєння виглядає так:

**a = b**

тут **a** - ім'я змінної або елементу масиву, **b** - значення або вираз. Внаслідок виконання оператора присвоювання змінній **a** присвоюється значення виразу **b**.

## Умовний оператор

Одним з основних операторів, які реалізують розгалуження в більшості мов програмування, є умовний оператор **if**. Є звичайна і розширена форми оператора **if** в Scilab. Звичайний **if** має вигляд:

```
if умова
оператори1
else
оператори2
end
```

тут умова - логічний вираз, оператори1, оператори2 — оператори мови **Scilab** або вбудовані функції. Оператор **if** працює за таким алгоритмом: якщо умова істинна, то виконуються оператори1, якщо хибна - оператори2.

В **Scilab** для запису складених логічних виразів можна використовувати логічні операції:

- логічне і - &, and;
- логічне або - |, or ;
- логічне заперечення — not;

і операції відношення:

- менше <;
- більше >;
- рівно == ;
- нерівно ~=, <>;
- менше або дорівнює <=;
- більше або дорівнює >=.

За потреби перевіряти багато умов можна скористатися розширеною формою

оператора **if**:

```
if умова1
оператори1
elseif умова2
оператори2
elseif умова 3
оператори3
...
elseif умова n
операторин
else
оператори
end
```

У цьому випадку оператор **if** працює так:

якщо умова1 істинна, то виконуються оператори1, інакше перевіряється умова2, якщо вона істинна, то виконуються оператори2, інакше перевіряється умова3 і т. д. Якщо ні одна з умов у гілках else і elseif не виконується, то виконуються оператори у гілці else.

Як приклад використання оператора розглянемо обчислення значення сигналу  $s(t)$ , заданого наступним алгоритмом:

1. Якщо  $(t > 0) \& (t < zT)$  то  $s(t) = A * t / zT$ ;
2. Якщо  $(t > zT) \& (t < cT)$  то  $s(t) = A - A * (t - zT) / (c - z) * T$ ;
3. Якщо  $(t > cT) \& (t < T)$  то  $s(t) = B$ ;

//Задання параметрів сигналу

A=input('A=');

B=input('B=');

z=input('z=');

c=input('c=');

t=input('t=');

if (t>0)&(t<zT)

s = A\*t/zT

elseif (t>zT)&(t<cT)

s = A - A\*(t-zT)/(c-z)\*T

elseif (t>cT)&(t<T)

s = B

end

disp('s(t)=',s);

end

### Оператор альтернативного вибору

Ще одним способом організації розгалужень є оператор альтернативного вибору **select** :

select параметр

case значення1 then оператори1

case значення2 then оператори2

...

else оператори

end

Оператор **select** працює таким чином:

якщо значення параметра рівне значенню1, то виконуються оператори1, інакше, якщо параметр дорівнює значенню2, то виконуються оператори2. В іншому випадку, якщо значення параметра співпадає з значенням3, то виконуються оператори3 і т. д. Якщо значення параметра не співпадає ні з одним зі значень у групах case, то виконуються оператори, які йдуть після службового слова **else**.

Для прикладу розглянемо завдання визначення дня тижня за датою за умови, що в місяці 31 день і 1-е число - понеділок.

Розв'язання завдання:

Якщо залишок від ділення заданого числа D на сім буде дорівнює одиниці, то це понеділок, двійці - вівторок, трійці - середа і так далі. Обчислити залишок від ділення числа x на k можна за формулою  $x - \text{int}(x / k) * k$ . Отже для побудови алгоритму необхідно використовувати сім умовних операторів.



```

D=input('Введіть дату від 1 до 31');
//Обчислення залишку від ділення D на 7 та порівняння його з числами від 0 до 6.
select D-int(D/7)*7
case 1 then disp('Monday');
case 2 then disp('Tuesday');
case 3 then disp('Wednesday');
case 4 then disp('Thursday');
case 5 then disp('Friday');
case 6 then disp('Saturday');
else
disp('Sunday');
end
-->exec('G:\Lecture Scilab EG\2\l2.sci');disp('exec done');
Enter a number from 1 to 31-->19
Friday

```

### Цикли в Scilab

Для організації циклу з невідомою кількістю повторень використовують оператор `while` умова оператори `end` тут умова - логічний вираз; оператори будуть виконуватися циклічно, поки логічний вираз (умова) істинний.

Для організації циклу з відомою кількістю повторень використовують оператор `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 дорівнює 1, то `hx` можна опустити, і в цьому випадку оператор `for` буде таким.

```

for x = xn: xk
оператори
end

```

Виконання циклу починається з присвоєння параметру стартового значення (`x = xn`). Потім виконується перевірка, чи не перевищує `x` кінцеве значення (`x > xk`). Якщо `x > xk`, то цикл вважається завершеним, і управління передається наступному за тілом циклу оператору. Якщо ж `x < xk`, то виконуються оператори в циклі (тіло циклу). Далі параметр циклу збільшує своє значення на `hx` (`x = x + hx`). Після чого знову проводиться перевірка значення параметра циклу, і алгоритм повторюється.

### Робота з файлами в Scilab

Функції роботи з файлами в `sci`-мові аналогічні відповідним функціям у мові `C`. Для відкриття файлу в `sci`-мові призначена функція `moren`, яка має вигляд:

```

[fd, err] = moren (ім'я_файлу, mode)
mode - режим роботи з файлом:

```

```

'R' - текстовий файл відкривається в режимі читання;
'Rb' - двійковий файл відкривається в режимі читання;
'W' - відкривається 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 буде використовуватися для додавання даних в кінець файлу; якщо файлу немає, він буде створений;

'Ab' - відкривається двійковий файл, який буде використовуватися для додавання даних в кінець файлу; якщо файлу немає, він буде створений;

'R +' - відкривається текстовий файл, який буде використовуватися в режимі читання і запису;

'Rb +' - відкривається двійковий файл, який буде використовуватися 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 +' - відкривається текстовий файл, який буде використовуватися для додавання даних у кінець файлу і читання даних, якщо файлу немає, він буде створений;

'Ab +' - відкривається двійковий файл, який буде використовуватися для додавання даних у кінець файлу і читання даних, якщо файлу немає, він буде створений.

Функція **mopen** повертає ідентифікатор відкритого файлу **fd** і код помилки **err**. Ідентифікатор файлу - ім'я (код), за яким описані нижче функції будуть звертатися до реального файлу на диску. У змінній **err** повертається значення 0 у разі вдалого відкриття файлу. Якщо **err**  $\neq$  0, то це означає, що файл відкрити не вдалося.

Для запису даних у відкритий файл використовують функцію `mfprintf(f, s1, s2)` тут *f* - ідентифікатор файлу (за значенням повертається функцією `mopen`), *s1* — керуючий рядок, *s2* - список виведених змінних. В керуючому рядку для кожної змінної із списку виводу потрібно вказати специфікації перетворення (опції в квадратних дужках є необов'язковими)

`%[керуючий символ][ширина][.точність][модифікатор]тип`

керуючий символ	Призначення
-	вирівнювання числа вліво. Права сторона доповнюється прогалинами (за замовчуванням виконується вирівнювання вправо)
+	перед числом виводиться знак «+» або «-»
прогалина	перед додатнім числом виводиться прогалина, перед від'ємним «-»
#	перед числом виводиться код системи числення (0 - вісімкова система числення, 0x (0X) - шістнадцяткова)

ширина	Призначення
n	задає кількість символів у полі виводу. Якщо заданої кількості символів недостатньо, то поле виводу розширюється до мінімально необхідного. Якщо ж ширина поля виводу більша, ніж потрібно, то незаповнені позиції заповнюються прогалинами
0n	те ж, що й n, але незаповнені позиції заповнюються нулями

.точність	Призначення
n	для типів e, E, f задає кількість знаків після десяткової крапки. Якщо опція .точність не задана, то використовується значення за замовчуванням

модифікатор	Призначення
c	ввід - символний тип CHAR, вивід - один байт
d,i	десятькове число із знаком
o	вісімкове число (int unsigned)
u	десятькове число без знаку
x, X	шістнадцятькове число (int unsigned) без знаку ( x - використовуються символи 0, 1, 2,...9, a,b,... f, X - використовуються символи 0, 1, 2,...9, A,B,... F)
f	значення зі знаком виду [-] dddd.dddd
e	значення зі знаком виду [-] d.dddde [+ -] ddd
E	значення зі знаком виду [-] d.ddddE [+ -] ddd
g	значення зі знаком типу E або F залежно від значення виразу і точності
G	значення зі знаком типу E або F в залежності від значення виразу і точності
s	рядок символів
h	для d, i, o, u, x, X коротке ціле
l	для d, i, o, u, x, X довге ціле

В керуючому рядку можна використовувати спеціальні символи:

спецсимвол	Призначення
b	зсув позиції виводу вліво
\n	перехід на новий рядок
\r	перехід в початок рядка
\t	горизонтальна табуляція
\'	символ «'»
\"	символ «"»
\?	символ «?»

Для закриття файлу в scі-мові призначена функція

**fclose(f)**

тут **f** - ідентифікатор файлу. За допомогою функції **fclose ('all')** можна закрити відразу всі відкриті файли, окрім стандартних системних файлів.

Розглянемо приклад запису в текстовий файл.

//У текстовий файл abc.txt вписані розміри матриці N і M

//та сама матриця A(N,M)

//N - кількість рядків матриці

N=3;

//M- кількість стовпців матриці

M=4;

A=[2 4 6 7; 6 3 2 1; 11 12 34 10];

//Відкриваємо порожній файл abc.txt в режимі запису

f=fopen('abc.txt','w');

//Записуємо в файл abc.txt N і M, розділені символом табуляції

```

mfprintf(f,'%d\t%d\n',N,M);
for i=1:N
for j=1:M
// Записуємо в файл abc.txt елемент матриці A(i,j)
mfprintf(f,'%g\t',A(i,j));
end
//Після запису в файл рядка матриці записуємо символ <<кінець рядка>>
mfprintf(f,'\n');
end
mclose(f);

```

На рис. 1.37 зображено вікно текстового редактора із вмістом файлу abc.txt

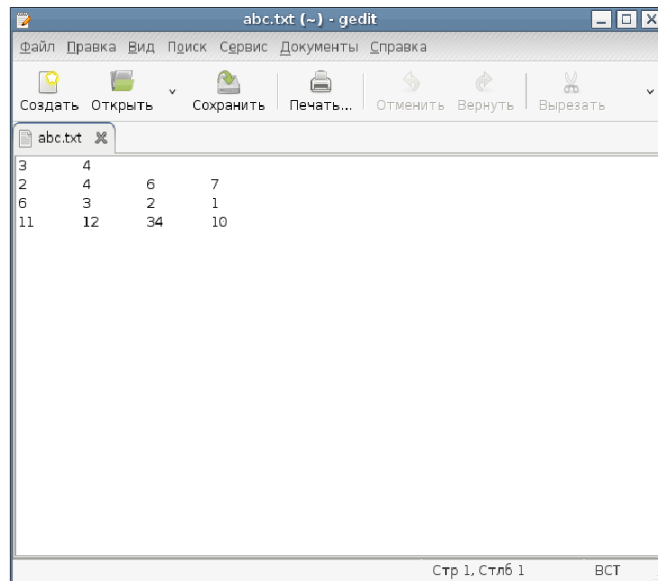


Рис. 1.37

Для читання даних із текстового файлу використовують функцію **змінна=mfscanf(f,'специфікація перетворення');** тут **f** - ідентифікатор файлу (за значенням повертається функцією **mopen**)

Розглянемо приклад читання даних із текстового файлу:

```

f=mopen('abc.txt','r');
N=mfscanf(f,'%d');
M=mfscanf(f,'%d');
for i=1:N
for j=1:M
A(i,j)=mfscanf(f,'%g');
end
end
mclose(f);

```

#### 1.1.16. Створення графічних додатків в Scilab

В **Scilab** окрім сценаріїв командного рядка можна створювати сценарії з графічним інтерфейсом. Основним графічним об'єктом в середовищі **Scilab** є вікно. Для створення порожнього графічного вікна служить функція **figure**

**f = figure ();**

Внаслідок виконання цієї команди буде створено графічне вікно з ім'ям **objfigure1**. За замовчуванням перше вікно отримує ім'я **objfigure1**, друге — **objfigure2** і т. д. Вказівник на графічне вікно записується в змінну **f**. Для задання властивостей вікна використовують функцію

**set(f,'властивість','значення')**

До властивостей вікна належать:

**position** — використовують для задання розташування і розміру вікна шляхом задання списку **[x y dx dy]**, де **x, y** — екранні координати верхнього лівого кута вікна, **dx** — розмір вікна по горизонталі (ширина вікна), **dy** — розмір вікна по вертикалі (висота вікна);

**figure\_name** — використовують для задання заголовку вікна.

Розглянемо приклад задання властивостей вікна:

**set(f,'position',[20,40,600,450])** — вікно має ширину 600 крапок і висоту 450 рядків, лівий кут вікна розташований у 40-му рядку та 20-тій крапці растру дисплея;

**set(f,'figure\_name','FIRST WINDOW')** — вікно матиме заголовок FIRST WINDOW.

Окрім задання властивостей вікна за допомогою функції **set** їх можна задати під час створення вікна

**f=figure('position',[20,40,600,450],'figure\_name','FIRST WINDOW');**



На рис. 1.38 зображена копія екрану дисплея з вікном із заданими властивостями

Рис. 1.38

Функція **close(f)** закриває вікно, а функція **delete(f)** вікно видаляє.

#### Динамічне створення інтерфейсних елементів графічного вікна

В **Scilab** використовують динамічний спосіб створення інтерфейсних компонентів. Він полягає в тому, що на стадії виконання програми можуть створюватися (і видалятися) ті чи інші елементи керування (кнопки, мітки, прапорці тощо) а їх властивостям присвоюються відповідні значення.

Для створення будь-якого інтерфейсного компонента із заданими властивостями використовують функцію **uicontrol**, яка повертає вказівник на створений елемент:

**C = uicontrol (F, 'Style', 'тип\_елемента', 'властивість\_1', значення\_1, 'властивість\_2', значення\_2,..., 'властивість\_k', значення\_k);**

тут **C** - вказівник на створюваний елемент; **F** - вказівник на об'єкт, всередині якого буде створюватися елемент (найчастіше цим об'єктом буде вікно); перший аргумент функції **uicontrol** не є обов'язковим, і якщо він відсутній, то батьком (власником) створюваного елемента є поточний графічний об'єкт — поточне графічне вікно;

**'Style'** - службовий рядок **Style**, який задає стиль створюваного елемента (символьне ім'я);

**'тип\_елемента'** - визначає, до якого класу належить створюваний елемент (**PushButton**, **Radiobutton**, **Edit**, **StaticText**, **Slider**, **Panel**, **Button Group**, **Listbox** або інші), цю властивість потрібно задавати для кожного елемента;

**'властивість\_k'**, **значення\_k** — визначають властивості та значення окремих елементів, вони будуть описані нижче конкретно для кожного компонента.

В існуючого інтерфейсного об'єкта можна змінити ті чи інші властивості за допомогою функції:

**set (C, 'властивість\_1', значення\_1, 'властивість\_2', значення\_2, ..., 'властивість\_k', значення\_k)**

тут **C** - вказівник на динамічний елемент, параметри якого потрібно змінити. **C** може бути і вектором динамічних елементів, в цьому випадку функція **set** буде задавати значення властивостей для всіх об'єктів **C (i)**;

**'властивість\_k'**, **значення\_k** - змінні параметри та їх значення.

Отримати встановлені значення параметрів елементів можна за допомогою функції:

**get (C, 'властивість')**

тут **C** - вказівник на динамічний інтерфейсний елемент, для якого потрібно визначити значення параметра;

**'властивість'** - ім'я параметра, значення якого потрібно визначити.

#### Командна кнопка (PushButton)

Командну кнопку типу **PushButton** створюють за допомогою функції **uicontrol**, в якій параметру **'Style'** надається значення **'pushbutton'**. За замовчуванням на ній немає напису, кнопка має сірий колір і розташовується в лівому нижньому кутку вікна. Напис на кнопці можна встановити за допомогою властивості **String**

```
//створюємо графічне вікно
d=figure();
// створюємо кнопку
dbt=uicontrol(d,'Style','pushbutton');
//задаємо напис YES на кнопці
set(dbt,'String','YES');
```

Модифікуємо програму створення кнопки, в якій задамо:

місце розташування і заголовок вікна;

напис на кнопці;

місце розташування кнопки.

```
//створюємо графічне вікно
f=figure();
//задаємо місце розташування вікна
set(f,'position',[0,0,250,100])
// задаємо заголовок вікна
set(f,'figure_name','Пуск');
//створюємо кнопку (style - pushbutton)
// місце розташування кнопки задається параметром position
```

```
Button=uicontrol('style','pushbutton','string','Пуск','position',[50,50,100,20]);
```

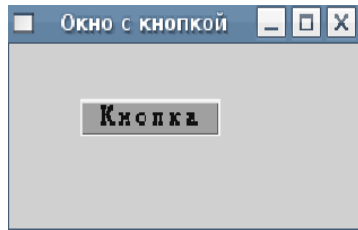


Рис. 1.39. Вікно з кнопкою (замінити)

“Клацання мишкою” по кнопці генерує подію **CallBack**, яка вказується як параметр функції **uicontrol**. Значенням параметра **CallBack** є рядок з ім'ям функції, яка викликається під час “клацання мишкою” по кнопці. У цьому випадку звертання до функції **uicontrol** є таким

```
Button = uicontrol ('style', 'pushbutton', 'string', 'Button', 'CallBack', 'Function');
```

тут **Function** - ім'я функції, яка виконується після настання події **CallBack**.

Для прикладу напишемо програму побудови графіка функції  $y = \sin(x)$ . Для побудови графіка потрібно у вікні “клацнути мишкою” по кнопці “Пуск”. Після “клацання мишкою” по кнопці “Пуск” викликається обробник події - функція **gr\_sin**, яка в окремому вікні буде графік функції  $y = \sin(x)$ .

```
f=figure();
set(f,'position',[0,0,250,100])
set(f,'figure_name','График');
//Створюємо кнопку, “клацання” якою викликає функцію gr_sin.
Button=uicontrol('style','pushbutton','string','Button',...
'position',[50,50,100,20],'CallBack','gr_sin');
function y=gr_sin()
x=-5:0.2:5;
y=sin(x);
plot(x,y);
xgrid();
endfunction
```

### Текстові поля (Мітки)

Наступним найбільш часто використовуваним елементом є текстове поле для відображення символічної інформації. Для визначення текстового поля значення параметра **'Style'** у функції **uicontrol** повинно мати значення **'text'**. Елемент призначений для виведення символічного рядка (або декількох рядків). Виведений текст - значення параметра **'String'** може бути змінений тільки з програми.

Розглянемо приклад створення текстового поля (мітки) за допомогою функції **uicontrol**

```
f=figure();
uicontrol('Style','text','Position',[10,130,150,20],'String',...
'Mitka');
```



Рис. 1.40. Вікно з текстовим полем (замінити)

Однією з основних властивостей елемента “текстове поле” є горизонтальне вирівнювання тексту, яке визначається властивістю **HorizontalAlignment**. Це властивість може набувати одне з наступних значень:

**left** - вирівнювання тексту по лівому краю;

**center** - вирівнювання тексту по центру (значення за замовчуванням);

**right** - вирівнювання по правому краю.

Для прикладу розглянемо вікно, яке містить чотири текстові поля із різними значеннями властивості **HorizontalAlignment**.

```
hFig=figure();
set(hFig,'Position',[50,50,300,200]);
hSt1=uicontrol('Style','text','Position',[30,30,150,20],...
'String','Mitka 1');
set(hSt1,'BackgroundColor',[1 1 1]);
set(hSt1,'HorizontalAlignment','left');
hSt2=uicontrol('Style','text','Position',[30,60,150,20],...
'HorizontalAlignment','center','BackgroundColor',[1 1 1],...
'String','Mitka 2');
hSt3=uicontrol('Style','text','Position',[30,90,150,20],...
'HorizontalAlignment','right','BackgroundColor',[1 1 1],...
'String','Mitka 3');
hSt4=uicontrol('Style','text','Position',[30,120,150,20],...
'BackgroundColor',[1 1 1],'String','Mitka 4');
```

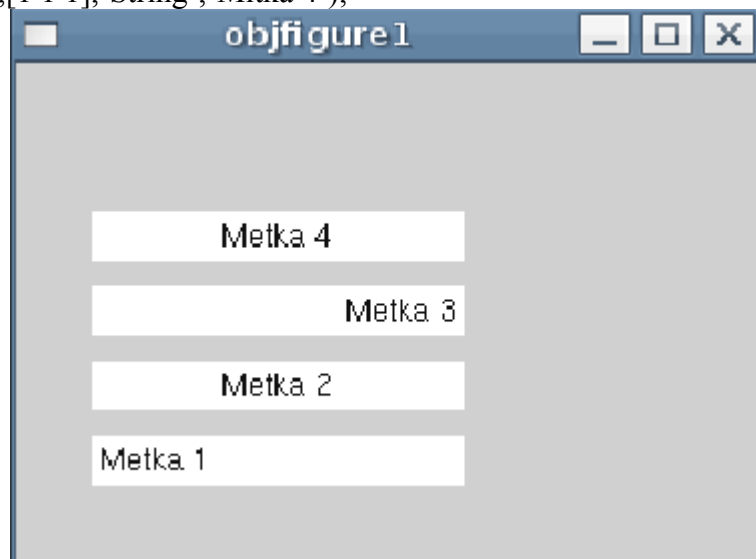


Рис. 1.41. Вікно з кількома текстовими полями (замінити)

#### Елемент “Радіокнопка”

Розглянемо ще один елемент - перемикач (**radiobutton** -Радіокнопка), який дає змогу перемикатися між двома станами або вимикати одну із властивостей. Створити перемикач (**radiobutton**) можна за допомогою функції **uicontrol**.

```
hFig=figure();
Rb=uicontrol('Style','radiobutton','String','name','value',1,...
'Position',[25,150,70,30]);
```

Під час створення перемикача потрібно задати його стан (параметр '**value**'), перемикач може бути увімкнений (значення '**value**' дорівнює 1) або вимкнений (значення



'value' дорівнює 0). Задати значення властивості 'value' можна також і за допомогою функції **set**

**set (Rb, 'value', 0)**

Отримати значення властивості 'value' можна за допомогою функції **get**.

Ввімкнення/вимкнення перемикача генерує подію 'CallBack', яка запускає на виконання задану функцію, наприклад:

**r1=icontrol('Style','radiobutton','String','sin(x)','value', 0,'CallBack','F1');**

тут у разі увімкнення перемикача буде згенерована подія 'CallBack', яка запустить на виконання функцію F1.

Розглянемо приклад програми з двома перемикачами, які дають змогу задавати функції  $\cos(x)$  — перший перемикач та  $\sin(x)$  — другий перемикач, для яких “натискання мишкою” кнопки “Plot” викликає побудову заданої функції (або заданих функцій). “Натискання мишкою” кнопки “Закрити” викликає закриття вікна.

//Створюємо графічне вікно.

hFig=figure('Position',[50,50,200,200]);

//Створення перемикачів

hRb1=icontrol('Style','radiobutton','String','sin(x)',...  
'value',1, 'Position',[25,100,60,20]);

hRb2=icontrol('Style','radiobutton','String','cos(x)',...  
'value',1, 'Position',[25,140,60,20]);

// Створюємо кнопку Plot, яка відповідно до натиснутих радіокнопок

//будує графіки вибраних користувачем функцій

Button=icontrol('style','pushbutton','string','Plot',...  
'position',[20,50,80,20],'CallBack','Radio');

// Створюємо кнопку Close, яка за допомогою обробника Final закриває вікно.

Button1=icontrol('style','pushbutton','string','Close',...  
'position',[20,25,80,20],'CallBack','Final');

//Функція Radio реагує на клацання радіокнопкою  
function Radio()

newaxes;

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

if get(hRb1,'value')==1 // Якщо увімкнена перша кнопка,

y=sin(x);

plot(x,y,'-r'); //то побудова графіка sin(x)

xgrid();

end;

if get(hRb2,'value')==1 //Якщо увімкнена друга кнопка,

y=cos(x);

plot(x,y,'-b'); //то побудова графіка cos(x)

xgrid(); //Нанесення масштабної сітки на графік

end;

endfunction

//Функція, яка реагує на “натискання” кнопки Close і закриває вікно.

function Final()

close(hFig);

endfunction



Рис. 1.42. Вікно програми з двома перемикачами

### Елемент “Вікно редагування”

Інтерфейсний елемент “Вікно редагування” ( властивість **'Style'** має значення **'edit'**) можна використовувати для вводу і виводу символної інформації. Текст, який набирається в вікні редагування, можна коригувати. Підтримуються операції з буфером обміну. Натискання клавіші **Enter** після вводу тексту призводить до генерації події **CallBack**.

Рядок вводу визначається параметром **'String'**, який відповідає тексту у вікні редагування. Введений текст можна притиснути до лівого або правого краю вікна вводу заданням відповідного значення властивості **HorizontalAlignment** (за аналогією з елементом «Мітка»). Якщо введений текст містить числове значення, яке потрібне для роботи програми, то значення властивості **'String'** можна перетворити в числовий формат за допомогою функції **eval** (або **evstr**).

Для прикладу розглянемо програму розв'язання квадратного рівняння (переробити)

```
f=figure(); //Створення графічного об'єкта.
```

```
//Задаємо розмір вікна.
```

```
set(f,'position',[0,0,700,300])
```

```
//Задаємо заголовок вікна
```

```
set(f,'figure_name','РІВНЯННЯ');
```

```
//Створення текстових полів для позначення полів вводу коефіцієнтів.
```

```
//Підпис A=.
```

```
lab_a=uicontrol(f,'style','text','string','A=','position',...  
[50, 250, 100, 20]);
```

```
//Підпис B=.
```

```
lab_b=uicontrol(f,'style','text','string','B=','position',...  
[150, 250, 100, 20]);
```

```
//Підпис C=.
```

```
lab_c=uicontrol(f,'style','text','string','C=','position',...  
[250, 250, 100, 20]);
```

```
//Поле редагування для вводу коефіцієнта a.
```

```
edit_a=uicontrol(f,'style','edit','string','1','position',...  
[50, 230, 100, 20]);
```

```
// Поле редагування для вводу коефіцієнта b.
```

```
edit_b=uicontrol(f,'style','edit','string','2','position',...  
[150, 230, 100, 20]);
```

```
// Поле редагування для вводу коефіцієнта c.
```

```
edit_c=uicontrol(f,'style','edit','string','1','position',...  
[250, 230, 100, 20]);
```

```
//Текстове поле для виводу розв'язків.
```

```
textresult=uicontrol(f,'style','text','string','', 'position',...  
[5, 80, 650, 20]);
```

```
BtSolve=uicontrol('style','pushbutton','string','Розв'язати',...  
'CallBack','Solve','position',[50,50,120,20]);
```

```
BtClose=uicontrol('style','pushbutton','string','Закрити',...  
'CallBack','_Close','position',[300,50,120,20]);
```

```
//Функція для розв'язування рівняння.
```

```
function Solve()
```

```
//Зчитування коефіцієнтів з текстових полів і
```

```

//перетворення їх в числові значення.
a=eval(get(edit_a,'string'));
b=eval(get(edit_b,'string'));
c=eval(get(edit_c,'string'));
d=b*b-4*a*c;
x1=(-b+sqrt(d))/2/a;
x2=(-b-sqrt(d))/2/a;
set(textresult,'string',sprintf
("Корені рівняння\t x1=%1.2f\t x2=%1.2f",x1,x2));
endfunction
// Функція закриття вікна.
function _Close()
close(f)
endfunction

```

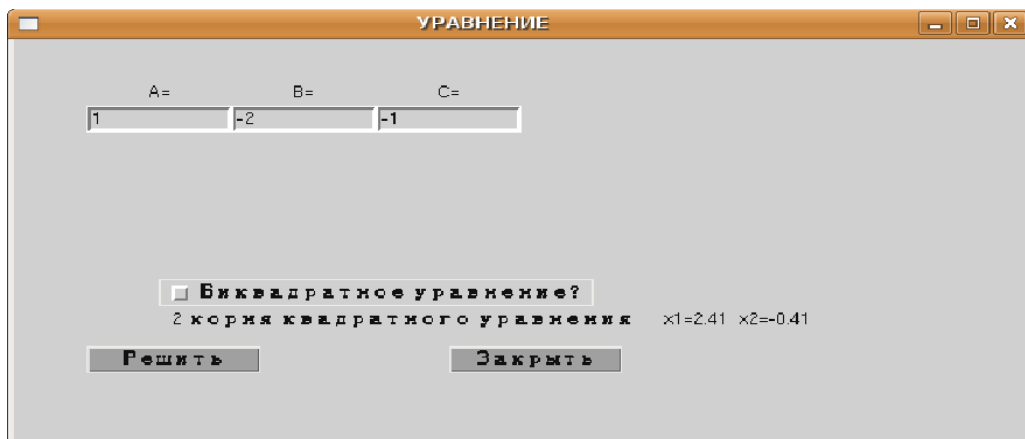


Рис. 1.43. Вікно програми розв'язання квадратного рівняння (замінити)

### Інтерфейсний елемент “Списки рядків символів”

Інтерфейсний елемент “Список рядків символів” у найпростішому випадку можна розглядати як вікно з масивом рядків у ньому. Якщо довжина списку перевищує висоту вікна, то для переміщення у списку може використовуватися вертикальна смуга прокручування, яка генерується автоматично.

Створити список рядків символів можна за допомогою функції **uicontrol** надаючи параметру **'Style'** значення **'listbox'**. Розглянемо це на простому прикладі

Переробити на програму для побудови графіків  $\sin$ ,  $\cos$ ,  $\exp(-x)/\cos(x)$

```

//Створення графічного вікна.
f=figure();
// Створення listbox
h=uicontrol(f,'style','listbox','position',[10 10 150 160]);
//Заповнення списку.
set(h,'string','строка 1|строка 2|строка 3');
set(h,'value',[1 3]);
//Виділення item 1 і 3 в списку.

```



Рис. 1.44. Список з виділеними елементами

Список дає змогу користувачу вибрати один або декілька рядків і залежно від вибору виконати ту чи іншу дію.

Вибір рядка здійснюють клацанням лівої кнопки миші в той момент, коли курсор вказує на обраний рядок. Одночасно з підсвічуванням рядка її номер заноситься у властивість 'value' і генерується подія 'CallBack'. Рядки в списку нумеруються від 1. Для вибору розрізнених рядків потрібно натиснути клавішу Ctrl і “клацати мишею” по виділених рядках. У цьому разі кожен виділений рядок підсвічується, а його номер запам'ятовується у векторі 'value'.

Для вибору групи сусідніх рядків можна натиснути і утримувати клавішу Shift, а потім “клацнути мишкою” по першому та останньому рядку групи. Всі проміжні рядки теж будуть виділені і всі їхні номери запам'ятовуються у векторі 'value'.

За допомогою розглянутих у цьому параграфі елементів і вбудованих функцій Scilab можна створювати візуальні програми для розв'язання інженерних і математичних задач будь-якої складності.

## 2. Система комп'ютерної алгебри Maxima

### 2.1. Інтерфейси Maxima

Система комп'ютерної математики Maxima працює як з командним рядком (рис. 2.1), так і з графічним інтерфейсом (оболонки xMaxima, Symaxx/2, wxMaxima).

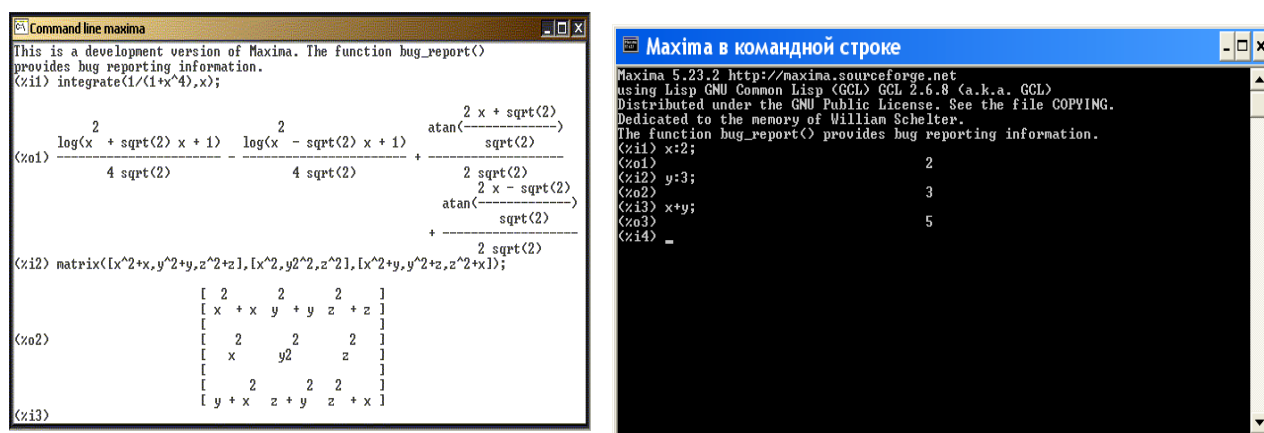


Рис. 2.1. Система Maxima у режимі командного рядка

Найбільш придатним для початківців виявився інтерфейс wxMaxima (рис. 2.2), який має розвинені можливості конструювання вхідних виразів та інтерактивну допомогу.

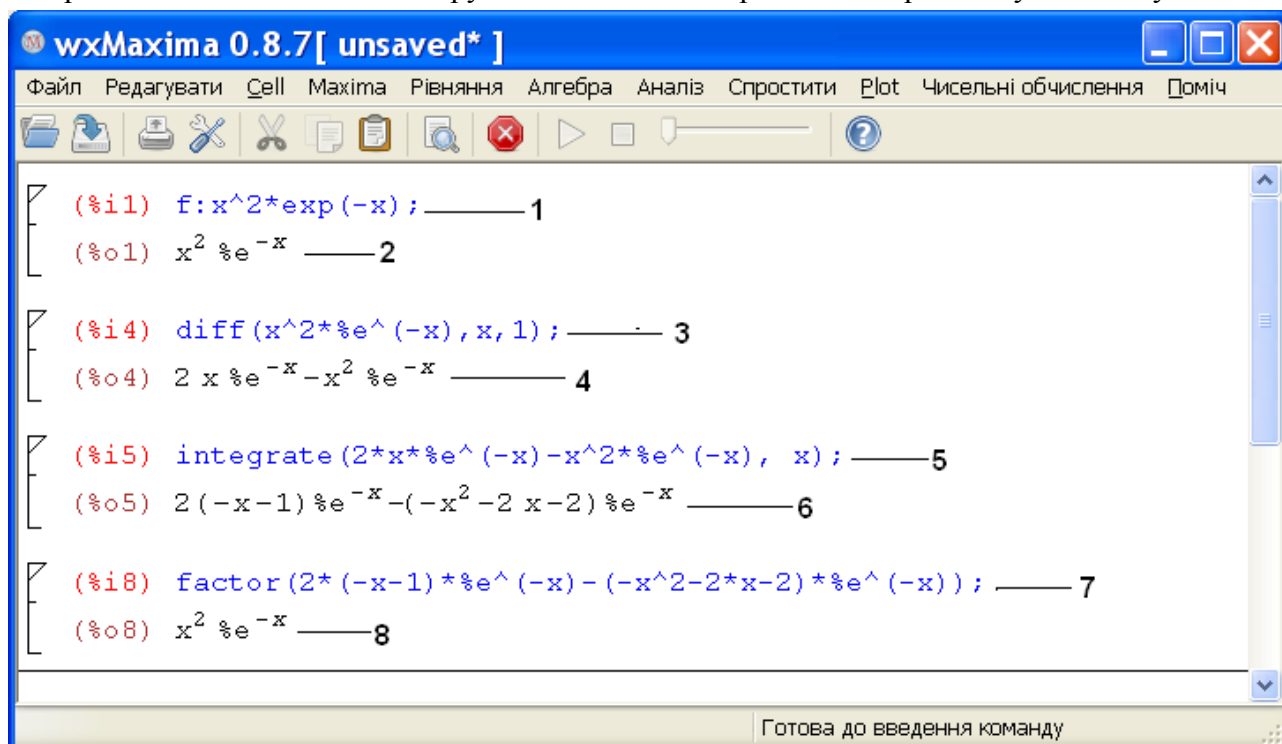


Рис. 2.2.

1 — команда присвоєння значення змінній f, 2 — вивід значення змінної f, 3 — команда диференціювання змінної f, 4 — результат диференціювання, 5 — команда інтегрування виразу  $2xe^{-x}-x^2e^{-x}$ , 6 — результат інтегрування, 7 — команда спрощення виразу  $2(-x-1)e^{-x}-(x^2-2x-2)e^{-x}$ , 8 — результат спрощення виразу  $2(-x-1)e^{-x}-(x^2-2x-2)e^{-x}$

В інтерфейсі wxMaxima ввід команди завершується знаком ;. Для виконання команди потрібно натиснути клавішу Enter (обробка і виконання команди без виводу результату виконання на дисплей) або Ctrl+Enter (обробка і виконання команди та вивід результату виконання на дисплей). Для виконання операцій з результатами виконання

команд можна використовувати контекстне меню — потрібно “виділити мишкою” вираз і клацнути правою кlawішею мишки. Далі у контекстному меню слід вибрати потрібну вам операцію. На рис. 2.4 зображені вікна wxMaxima з командами диференціювання та інтегрування, які були введені через контекстне меню

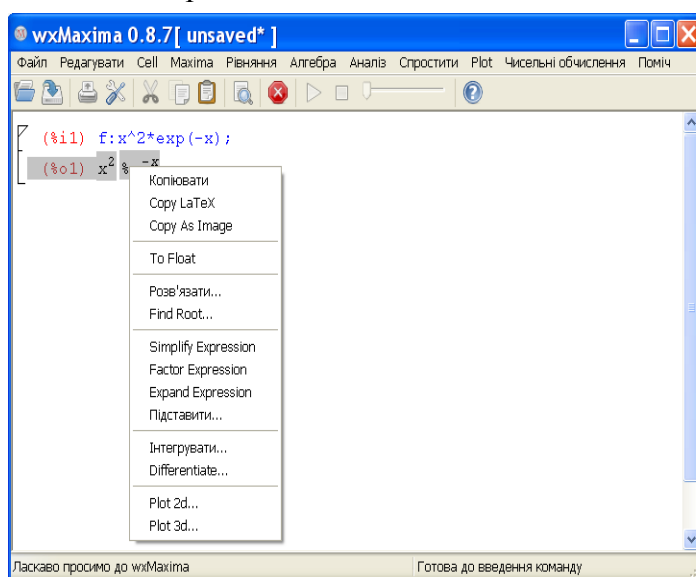


Рис. 2.3. Контекстне меню wxMaxima

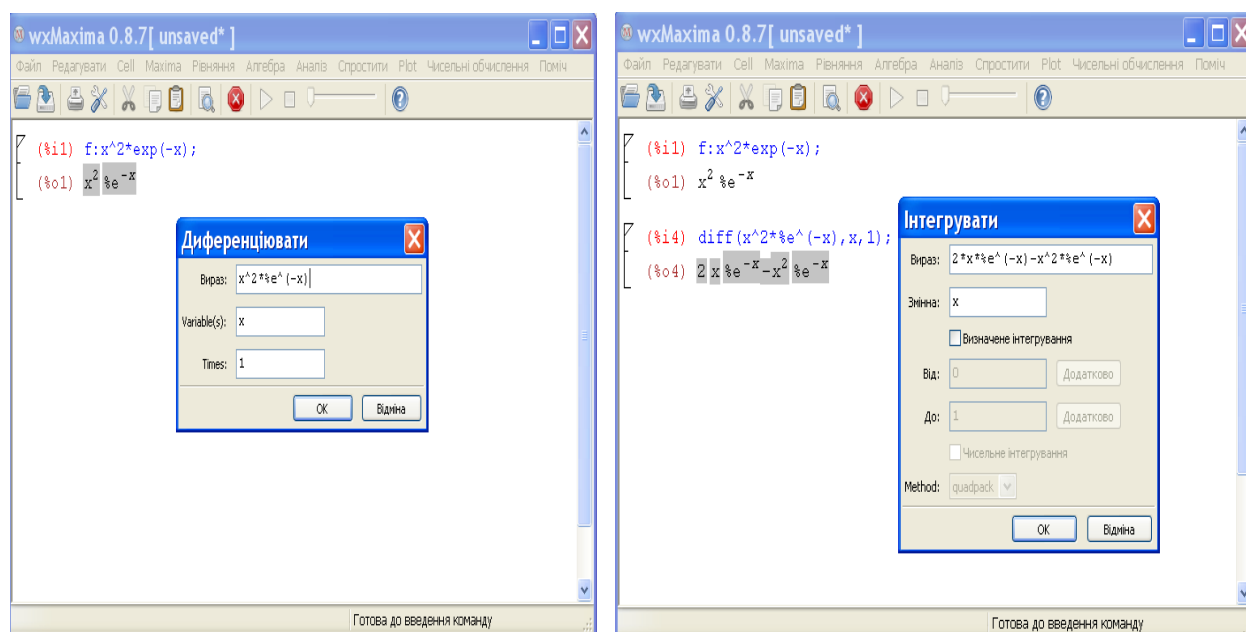


Рис. 2.4.

## 2.2. Початок роботи у Maxima

Після старту Maxima виводиться інформація про систему та мітка (C1) або (%i1) залежно від поточної конфігурації. Кожна введена команда та її вивід позначаються міткою з метою повторного використання. Символ C (від command) або %i (від input) використовуються для позначення команд, введених користувачем, а D (від display) або %o (від output) – у разі відображення результатів обчислень.

Для ініціалізації процесу обчислень необхідно ввести команду, символ “;” (крапка з комою) та натиснути Enter. Якщо вивід результату на екран не потрібен, то замість крапки з комою використовується символ “\$”. Звернутися до результату останньої команди можна за

допомогою символу “%”. Для повтору раніше введеної команди, скажімо, (C2), досить ввести два апострофа та мітку потрібної команди: ‘ ‘C2.

Система Махіма звертає увагу на регістр введених символів в іменах вбудованих констант та функцій: запис  $\sin(x)$  не еквівалентний запису  $\text{SIN}(x)$ .

Для стандартних математичних констант застосовують наступні позначення: %e - основа натуральних логарифмів, %i - уявна одиниця, %pi - число  $\pi$ , %phi - золоте січення, %gamma - постійна Ейлера, inf — нескінченність (в додатньому напрямку), minf - нескінченність (у від’ємному напрямку), infinite - нескінченність (на комплексній площині).

Надання значення будь-якій змінній виконується за допомогою знака “:” (двокрапка), а символ “=” (дорівнює) застосовують для визначення рівнянь та підстановок.

(C1) x:2;

(D1) 2

(C2) y:3;

(D2) 3

(C3) x + y;

(D3) 5

Функція kill анулює значення змінних. Параметр all цієї функції видаляє всі змінні, включно з мітками.

(C8) kill(x);

(D8) DONE

(C9) x + y;

(D9) x + 3

(C10) kill(all);

(D0) DONE

(C1) x + y;

(D1) y + x

Махіма підтримує точні обчислення з великими числами, розмір яких обмежений лише ресурсами ЕОМ.

Наприклад, обчислимо  $144^{25}$ , після чого візьмемо корінь 25-го степеня з результату:

(C1) 144^25;

(D1) 910043815000214977332758527534256632492715260325658624

(C2) %^(1/25);

(D2) 144

Знак “^” або “\*^” – це оператор піднесення до степеня.

Якщо є змінна з певним значенням і ми бажаємо застосувати сам цей символ, а не його значення, ми можемо заблокувати його обчислення за допомогою знака “’” (апостроф). Обернена операція виконується функцією ev.

### 2.3. Вирази та елементарні функції

Для запису виразів в Махіма використовують числа, імена змінних, математичних функцій, дужки і знаки арифметичних операцій (+, -, \*, /, \*\* або ^ - тут знак \*\* або ^ означає піднесення до степеня). Вирази в Махіма можуть перебувати в двох формах:

недосконалий (англ. noun form) – необчислений вираз;

досконалий (англ. verb form) – обчислений вираз.

Під час уведення виразу він перебуває в недосконалій формі, після натискання клавіші Enter (у wxMaxima Alt+Enter) відбувається обчислення виразу.

Робота з частинами виразів нагадує роботу зі списками. Так, для виділення будь-якої заданої частини виразу використовується функція part.

(C1) f:a+b\*x^2+c\*x^3; //надаємо f значення функції

(D1)  $c \cdot x^3 + b \cdot x^2 + a$  //результат

```
(C2) part(f,2);           //виділяємо другу частину виразу
(D2)  $b \cdot x^2$            //результат виконання
(C3) kill(all);
```

Наступні функції повертають окремі частини виразу:

first(f) – повертає перший елемент f;

last(f) – повертає останній елемент f;

rest(f) – повертає f з вилученим першим елементом.

Спрощення математичних виразів – одна з найважливіших задач символічної математики. Часто складний математичний вираз, що лякає новачків своїм грізним видом, тотожно дорівнює одиниці або зводиться до простого виразу після ряду перетворень.

Для спрощення виразів використовується функція factor(exp). Вона виконує послідовність алгебраїчних перетворень над виразом exp і повертає найпростішу зі знайдених форм.

Функція factor(exp) працює з будь-якими математичними виразами: поліномами, раціональними виразами, розширеними раціональними виразами (зі змінними у дробових степенях).

а) комбінування числових підвиразів:

```
(C1) factor(6*x^2);
```

```
(D1) 12x;
```

б) приведення подібних множників у добутках:

```
(C2) factor(x^3*y*x^5);
```

```
(D2)  $x^8 y$ 
```

в) приведення подібних членів суми:

```
(C3) factor(x+12+4*x);
```

```
(D3) 5x+12;
```

г) скорочення на найбільший поліноміальний дільник:

```
(C4) factor((x^2-2*y*x+y^2)/(x^2-y^2));
```

```
(D4)  $\frac{x-y}{x+y}$ 
```

д) розкладання поліномів і пониження степеню виразів:

```
(C5) factor((x+1)^2-x^2);
```

```
(D5) 2x+1
```

е) приведення загальних знаменників до виразів зі зниженим степенем з виключенням чи скороченням змінних:

```
(C6) factor((2*x)/(x^2-1)-1/(x+1));
```

```
(D6)  $\frac{1}{x-1}$ 
```

```
(C7) kill(all);
```

Для розкриття дужок використовується функція expand – ще одна типова операція комп'ютерної алгебри. За змістом вона протилежна спрощенню виразів. Часто компактна форма подання виразів обумовлена визначеними операціями з їхнього спрощення.

```
(C1) expand((x-a)*(x-b)*(x-c));
```

```
(D1)  $x^3 - cx^2 - bx^2 - ax^2 + bcx + acx - abc$ 
```

```
(C2) kill(all);
```

Команда ev дає змогу отримати числове значення виразу. Її перший аргумент є виразом, що обчислюється, а другий – опція numer. Нагадаємо, що символ % означає результат попереднього обчислення.

```
(C1) 1/100+1/1001;
```

```
(D1)  $\frac{201}{10100}$ 
```

```
(C2) (1+sqrt(2))^5;
```



```
(D2)  $(\sqrt{2} + 1)^5$ 
(C3) expand(%);
(D3)  $29\sqrt{2} + 41$ 
(C4) ev(29*sqrt(2) + 41, numer);
(D4) 82.01219330881976
```

Допускається більш зручна форма функції ev, яка вимагає вказання тільки її аргументів:

```
(C5) 29*sqrt(2) + 41, numer;
(D5) 82.01219330881976
```

За замовчуванням результат містить 16 значущих цифр. Для виведення числа в експонентній формі використовується функція bfloat:

```
(C6) bfloat(d3);
(D6) 8.201219330881976B1
```

Запис mBn є скороченою формою виразу  $m \cdot 10^n$ .

Кількість значущих цифр у поданні числа визначається спеціальною змінною fpprec. Збільшення її значення приводить до зростання точності результату, наприклад,

```
(C7) fpprec;
(D7) 16
(C8) fpprec:100;
(D8) 100
(C9) 'c5;
(D9) 8.20121933088197607657604923686248525#
030775305167218616484631047782707024434954#
8350683851114422615155B1
```

Символ # наприкінці виведеного рядка означає, що число не вмістилося на одному рядку і його частина, що залишилася, переноситься на наступний. В останньому прикладі ми використовували повторення раніше введеної команди ('c5).

Система Maxima може працювати з числами довільної довжини і точності:

```
(C10) 100!;
(D10) 933262154439441526816992388562667004#
9071596826438162146859296389521759999322991#
5608941463976156518286253697920827223758251#
185210916864000000000000000000000000000000
```

Наприклад, обчислимо число  $\pi$  з точністю 200 знаків після коми:

```
(C11) %pi, numer;
(D11) 3.141592653589793
(C12) fpprec:200;
(D12) 200
(C13) bfloat(%PI);
(D13) 3.141592653589793238462643383279502884#
19716939937510582097494459230781640628620899#
86280348253421170679821480865132823066470938#
44609550582231725359408128481117450284102701#
9385211055596446229489549303819B0
```

Якщо потрібно вивести зі збільшеною точністю тільки результати декількох команд, то варто використовувати функцію block. Першим аргументом цієї функції є список локальних змінних, тобто змінних, значення яких будуть відновлені після завершення виконання команд блоку. При вказанні таких змінних можливе присвоєння їм тимчасових значень. Після списку локальних змінних через кому вказується послідовність команд.

```
(C14) fpprec:16;
(D14) 16
(C15) bfloat(%PI);
(D15) 3.141592653589793B0
```

```
(C16) block([fpprec:100], bfloat(%pi));
(D16) 3.14159265358979323846264338327950#
2884197169399375105820974944592307816406#
286208998628034825342117068B0
(C17) ' 'c15;
(D17) 3.141592653589793B0
```

В Махіма доступні прямі і зворотні тригонометричні функції: sin (синус), cos (косинус), tan (тангенс), cot (котангенс), asin (арксинус), acos (арккосинус), atan (арктангенс), acot (арккотангенс). Крім них, є менш відомі секанс (sec) і косеканс (csc).

```
(C18) block([fpprec:100], sin(bfloat(%pi)));
(D18) - 2.570579198029723711689569064713781#
2738478411385601247337570449378000209830368#
31739403591933813645377B-101
(C19) block([fpprec:100], sin(%pi)) ;
(D19) 0
(C20) sin(%pi/6)^2 +cos(%pi/6)^2;
(D20) 1
(C21) sec(%PI/3);
(D21) 2
(C22) asin(1/2);
(D22)  $\frac{\%PI}{6}$ 
```

На жаль, Махіма не може в символічному вигляді обчислити значення обернених тригонометричних функцій у деяких точках:

```
(C23) asin(sqrt(3)/2);
(D23)  $ASIN\left(\frac{\sqrt{3}}{2}\right)$ 
(C24) %, expand;
(D24)  $ASIN\left(\frac{\sqrt{3}}{2}\right)$ 
```

Махіма легко оперує тригонометричними виразами. Так, функція trigexpand використовує формули перетворення сум двох кутів для подання уведеного виразу в якомога більш простому виді:

```
(C15) sin(u+v)*cos(u)^3;
(D15)  $\cos^3(u) \sin(v+u)$ 
(C16) trigexpand(%);
(D16)  $\cos^3(u)(\cos(u)\sin(v)+\sin(u)\cos(v))$ 
```

Функція trigreduce приведе тригонометричний вираз до суми елементів, кожний з яких містить єдиний sin чи cos:

```
(C17) trigreduce(%);
(D17)  $\frac{\sin(v+4u)+\sin(v-u)}{8} + \frac{3\sin(v+2u)+3\sin(v)}{8}$ 
```

Для обчислення натурального логарифма використовують функцію log:

```
(C25) log(%E^2);
(D25) 2
(C26) 5*log(a)+6*log(b);
(D26) 6 log(b) + 5 log(a)
(C27) logcontract(%);
(D27)  $\log(a^5b^6)$ 
```

### Функції для роботи з виразами

№	Функція	Призначення
1	part ( <b>вираз</b> )	Виділення будь-якої заданої частини виразу
2	first ( <b>вираз</b> )	Повертає перший елемент виразу
3	last ( <b>вираз</b> )	Повертає останній елемент виразу
4	rest ( <b>вираз</b> )	Повертає вираз без першого елемента
5	factor ( <b>вираз</b> )	Спрощує вираз
6	gfactor ( <b>вираз</b> )	Спрощує комплексний вираз
7	expand ( <b>вираз</b> )	Розкриття дужок
8	combine ( <b>вираз</b> )	Об'єднує доданки з однаковими знаменниками
9	xthru ( <b>вираз</b> )	Приводить вираз до спільного знаменника
10	ev ( <b>вираз</b> )	Виконує обчислення виразу
11	bfloat ( <b>вираз</b> )	Виведення числа в експонентній формі
12	fpprec	Кількість значущих цифр у поданні числа
13	block	Вивід зі збільшеною точністю для окремих команд
14	trigexpand ( <b>вираз</b> )	Використовує формули перетворення сум двох кутів для спрощення виразу
15	trigreduce ( <b>вираз</b> )	Перетворює тригонометричний вираз на суму елементів, кожен з яких містить єдиний sin чи cos
16	rat ( <b>вираз</b> )	Приводить вираз до канонічного вигляду
17	logcontract ( <b>вираз</b> )	Перетворює суму логарифмів на логарифм добутку

### Елементарні функції

№	Функція	Призначення
1	sin()	Синус
2	cos()	Косинус
3	tan()	Тангенс
4	cot()	Котангенс
5	sec()	Секанс
6	csc()	Косеканс
7	asin()	Арксинус
8	acos()	Арккосинус
9	atan()	Арктангенс
10	acot()	Арккотангенс
11	asec()	Арксеканс
12	acsc()	Арккосеканс
13	log(exp)	Обчислення натурального логарифма
14	sinh()	Гіперболічний синус
15	cosh()	Гіперболічний косинус

16	<code>tanh()</code>	Гіперболічний тангенс
14	<code>asinh()</code>	Зворотній гіперболічний синус
15	<code>acosh()</code>	Зворотній гіперболічний косинус
16	<code>atanh()</code>	Зворотній гіперболічний тангенс
17	<code>abs()</code>	Абсолютне значення
18	<code>floor()</code>	Округлення до найменшого цілого
19	<code>ceiling()</code>	Округлення до найбільшого цілого
20	<code>fix()</code>	Ціла частина
21	<code>float()</code>	Перетворення до формату з плаваючою комою

## 2.4. Поліноми та алгебраїчні перетворення

*Поліномом* називають вираз, який складається з декількох частин одного виду:

$$P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n.$$

Над поліномами можна виконувати наступні арифметичні операції:

додавання;

віднімання;

множення;

ділення.

(C1) `p1:x^2-1;`

(D1)  $x^2 - 1$

(C2) `p2:x-1;`

(D2)  $x - 1$

(C3) `expand(p1*p2);`

(D3)  $x^3 - x^2 - x + 1$

(C4) `expand((p1+p2));`

(D4)  $x^2 + x - 2$

(C5) `p1/p2;`

(D5)  $\frac{x^2 - 1}{x - 1}$

(C6) `p1-p2;`

(D6)  $x^2 - x$

Функцію `divide` можна використовувати для добування частки і залишку від ділення одного полінома на інший:

(C7) `divide(p1*p2, p1);`

(D7)  $[x - 1, 0]$

Функція `gcd` визначає найбільший спільний дільник поліномів, а `factor` здійснює розклад полінома на множники:

(C8) `gcd(x^3-1, x^2-1, x-1);`

(D8)  $x - 1$

(C9) `factor(x^8-1);`

(D9)  $(x - 1)(x + 1)(x^2 + 1)(x^4 + 1)$

Функція `gfactor` здійснює розклад полінома, використовуючи комплексні числа

(C10) `gfactor(x^8-1);`

(D10)  $(x - 1)(x + 1)(x - \%I)(x + \%I)(x^2 - \%I)(x^2 + \%I)$

Підстановка будь-якого виразу замість змінної здійснюється за допомогою операції `=`. Наприклад, замінимо усі входження `x` у виразі на `5/z`:

(C11)  $x^4 + 3x^3 - 2x$ ,  $x=5/z$ ;

(D11) 
$$-\frac{10}{z} + \frac{375}{z^3} + \frac{625}{z^4}$$

Функція ratsimp виносить за дужки найбільший спільний дільник:

(C12) ratsimp(%);

(D12) 
$$-\frac{10z^3 - 375z - 625}{z^4}$$

Використовуючи функцію assume (*to assume* – допускати), можна при обчисленнях враховувати додаткові умови, що задаються нерівностями:

(C13) sqrt(x^2);

(D13) ABS(x)

(C14) assume(x<0);

(D14) [x < 0]

(C15) sqrt(x^2);

(D15) - x

Функція forget (*to forget* – забувати) знімає всі обмеження, накладені за допомогою assume:

(C16) forget(x<0);

(D16) [x < 0]

(C17) sqrt(x^2);

(D17) abs(x)

Функції realpart і imagpart повертають дійсну і уявну частину комплексного виразу:

(C18) z1:-3+%i\*4;

(D18) 4 %i - 3

(C19) z2:4-2\*%i;

(D19) 4 - 2 %i

(C20) z1\*z2;

(D20) (4 - 2 %i) (4 %i - 3)

(C21) expand(%);

(D21) 22 %i - 4

(C22) realpart('c20);

(D22) - 4

(C23) imagpart('c20);

(D23) 22

Функція denom виділяє знаменник, а num – чисельник виразу:

(C24) denom((x^2-x-1)/(x-1));

(D24) x-1

(C25) num((x^2-x-1)/(x-1));

(D25)  $x^2 - x - 1$

№	Функція	Призначення
1	divide	Добування частки і залишку від ділення одного полінома на інший
2	gcd	Визначає найбільший спільний дільник поліномів
3	factor	Здійснює розклад полінома на множники (для чисел – канонічний розклад)
4	gfactor	Здійснює розклад полінома, використовуючи комплексні числа
5	ratsimp	Виносить за дужки найбільший спільний дільник
6	realpart	Повертає дійсну частину комплексного виразу

7	imagpart	Повертає уявну частину комплексного виразу
8	assume	Враховує додаткові умови, що задаються нерівностями
9	forget	Знімає всі обмеження, накладені за допомогою assume
10	denom	Виділяє знаменник виразу
11	num	Виділяє чисельник виразу

## 2.5. Розв'язання рівнянь

За допомогою функції `solve` Maxima можна розв'язувати рівняння і системи алгебраїчних рівнянь. Рівна нулевій права частина рівняння може бути опущена:

```
(C1) solve (x^2=1, x);
(D1) [x = - 1, x = 1]
(C2) solve (x^2-1,x);
(D2) [x = - 1, x = 1]
(C3) solve (log(x+3)=1, x);
(D3) [x = %E - 3]
```

Під час розв'язання тригонометричних рівнянь видається тільки одне з нескінченної множини можливих розв'язків:

```
(C4) solve(sin(x)-1, x);
SOLVE is using arc-trig functions to get
a solution. Some solutions will be lost.
```

```
(D4) x =  $\frac{\%PI}{2}$ 
```

У наступному прикладі функція `solve` використана для розв'язання системи з трьох рівнянь із трьома невідомими:

```
(C5) s:[x+y+z=3, x+2*y-z=2, x+y*z+z*x=3];
(D5) [z + y + x = 3, - z + 2 y + x = 2, y z + x z + x = 3]
(C6) solve(s, [x,y,z]);
(D6) [[x = 1, y = 1, z = 1],
      [x = 7, y = - 3, z = - 1]]
```

Якщо рівняння не має розв'язків на множині дійсних чисел, то Maxima шукає розв'язки серед комплексних чисел:

```
(C7) solve(x^2+1,x);
(D7) [x = - %i, x = %i]
```

Якщо ми хочемо одержати розв'язки рівняння в числовому вигляді, то після функції `solve` варто використовувати опцію `numer` або `float`.

```
(C8) b:x^5+8*x^4+31*x^3+80*x^2+94*x=-20;
(D8) x^5+8x^4+31x^3+80x^2+94x=-20;
(C10) solve(b,x),numer;
(D10) [x = - sqrt(3) - 2, x = sqrt(3) - 2, x = - 2,
      x = - 3 %i - 1, x = 3 %i - 1]
```

```
(C11) solve(b,x),float;
```

```
RAT replaced 3.464101615137754 by 8733//2521 = 3.464101547005157
```

```
RAT replaced 3.464101615137754 by 8733//2521 = 3.464101547005157
```

```
(D11) [x = - 3.732050773502579, x = - 0.26794922649742,
      x = - 2, x = - 3 %i - 1, x = 3 %i - 1]
```

Функція `allroots` обчислює всі дійсні і комплексні корені поліноміальних рівнянь.

```
(C12) allroots((1+2*X)^3 = 13.5*(1+X^5));
(D12) [X = 0.82967499021294, X = - 1.015755543828121,
      X = 0.96596251521964 %i - 0.40695972319241,
```

```
X = - 0.96596251521964 %i - 0.40695972319241,
X = 1.0]
```

Функція `realroots` шукає всі дійсні корені (комплексні не беруться до уваги) поліноміального виразу.

```
(C13) realroots(-1-x+x^5,5.0e-6)
```

```
(D13) x = 612003
      524288
```

```
(C14) ev(%[1],float)
```

```
(D14) X = 1.167303085327148
```

```
(C15) ev(-1-X+X^5,%)
```

```
(D16) - 7.396496210176906E-6
```

Перевірку розв'язку потрібно виконувати у такий спосіб:

```
(C17) result:solve (x^2=1, x);
```

```
(D17) [x=-1,x=1]
```

```
(C18) x^2=1,result;
```

```
(D18) 1=1
```

Функція `lhs(exp)` повертає ліву частину `exp`.

Функція `rhs(exp)` повертає праву частину `exp`.

У тих випадках, коли неможливо розв'язати задане рівняння аналітично, можна наближено обчислити значення кореня у такий спосіб. Спочатку за допомогою функції `plot2d` будуються графіки (див. далі) лівої і правої частин рівняння, а далі за графіком знаходиться наближення.

Наприклад, знайдемо наближений розв'язок рівняння  $e^x - x^2 = 0$ . Побудуємо графіки функцій  $e^x$  і  $x^2$ , виконавши команду `plot2d([%e^x, x^2], [x, -1, 1])`. Ми одержимо зображення графіків функцій  $e^x$  і  $x^2$  на одному кресленні для значень аргументу  $x$ , що змінюється на відрізок  $[-1, 1]$ .

Точка перетину цих графіків  $x \approx -0.7$  (рис. 6) задає наближене значення кореня рівняння  $e^x - x^2 = 0$ .

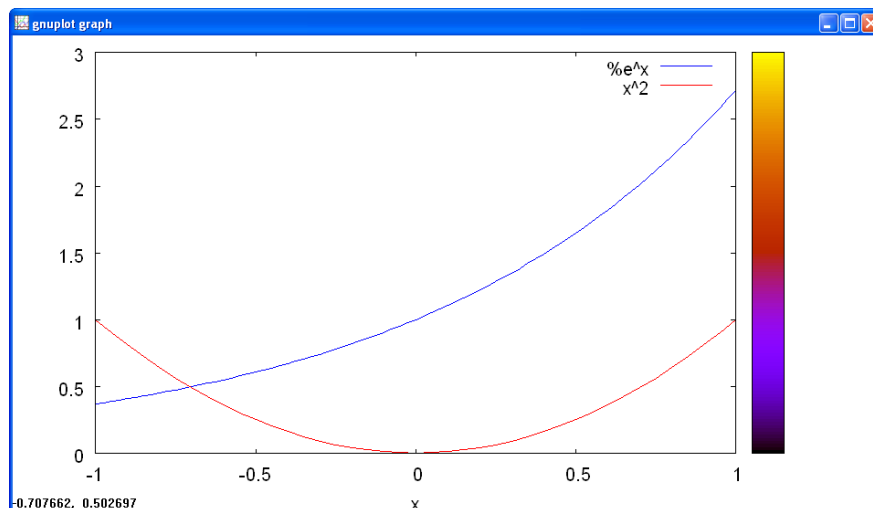


Рис. 2.5.

№	Функція	Призначення
1	<code>solve</code>	Розв'язує рівняння та системи алгебраїчних рівнянь
2	<code>allroots</code>	Обчислює всі дійсні та комплексні корені поліноміальних рівнянь
3	<code>realroots</code>	Шукає всі дійсні корені (комплексні не беруться до уваги) поліноміального виразу
4	<code>lhs(exp)</code>	Повертає ліву частину виразу

5	rhs (exp)	Повертає праву частину виразу
---	-----------	-------------------------------

## 2.6. Операції математичного аналізу

В Maxima можна обчислювати похідні й інтеграли, обчислювати межі і знаходити точні розв'язки звичайних диференціальних рівнянь.

До числа найбільш часто використовуваних математичних операцій належить обчислення похідних функцій в аналітичному вигляді. Для добування похідної використовується функція `diff`, першим аргументом якої є функція, другим – змінна, за якою виконується диференціювання, і третім (необов'язковим) – порядок похідної:

(C1) `f: (x-2*sqrt(x))/x^2;`

(D1) 
$$\frac{x-2\sqrt{x}}{x^2}$$

(C2) `diff(f, x);`

(D2) 
$$\frac{1-\frac{1}{\sqrt{x}}}{x^2} - \frac{2(x-2\sqrt{x})}{x^3}$$

(C3) `expand(' 'c2);`

(D3) 
$$\frac{3}{x^{\frac{5}{2}}} - \frac{1}{x^2}$$

(C4) `g:x^6;`

(D4) 
$$x^6$$

(C5) `diff(g, x, 1);`

(D5) 
$$6x^5$$

(C6) `diff(g, x, 4);`

(D6) 
$$360x^2$$

Під час обчислення кратних похідних по декількох змінних після задання функції перераховуються змінні диференціювання із заданням відповідних кратностей, наприклад

(C7) `diff(x^6*y^3, x, 4, y, 2);`

(D7) 
$$2160 x^2 y^2$$

Одна з найважливіших операцій – аналітичне обчислення первісних і визначених інтегралів. Первісна – це функція  $F(x)$ , яка задовольняє рівняння

$$\int f(x)dx = F(x) + C,$$

де  $C$  – стала інтегрування. Обчислення визначеного інтеграла з межами – верхньою  $b$  і нижньою  $a$  – виконується за формулою

$$\int_a^b f(x)dx = F(b) - F(a)$$

Визначений інтеграл може бути представлений як аналітичним, так і числовим значенням. Для обчислення значень визначених інтегралів розроблений ряд наближених методів – від простих (прямокутників і трапецій) до складних, які автоматично адаптуються до характеру зміни підінтегральної функції  $f(x)$ .

Для інтегрування в системі Maxima використовують наступні функції:

`integrate(f,x)` – повертає первісну (невизначений інтеграл) підінтегральної функції  $f$  за змінною  $x$ .

`integrate(f,x,min,max)` – повертає значення визначеного інтеграла з межами від  $\min$  до  $\max$ .

(C8) `f:x^2/(4*x^6+1);`



$$(D8) \quad \frac{x^2}{4x^6 + 1}$$

(C9) integrate(f, x);

$$(D9) \quad \frac{\text{atan}(2x^3)}{6}$$

У випадку неоднозначної відповіді Maxima може задавати додаткові питання, як у наступному прикладі:

(C10) integrate(x^n, x);

Is n + 1 zero or nonzero?  
nonzero;

$$(D10) \quad \frac{x^{n+1}}{n+1}$$

(C11) integrate(x^n, x);

Is n + 1 zero or nonzero?  
zero;

$$(D11) \quad \log(x)$$

Можна використовувати функцію assume для задання додаткових умов (не забувайте потім видалити накладені обмеження):

(C12) assume(notequal(n, -1));

(D12) [not equal(n, - 1)]

(C13) integrate(x^n, x);

$$(D13) \quad \frac{x^{n+1}}{n+1}$$

(C14) forget(notequal(n, -1));

(D14) [not equal(n, - 1)]

(C15) integrate(x^n, x);

Is n + 1 zero or nonzero?  
zero;

$$(D15) \quad \log(x)$$

Для знаходження визначеного інтегралу варто вказати додаткові аргументи – межі інтегрування:

(C16) integrate(x^2, x, 0, 6);

(D16) 72

(C17) integrate(sin(x), x, 0, %pi);

(D17) 2

В Maxima можна обчислювати навіть кратні інтеграли з фіксованими і змінними верхньою чи нижньою межами. Кратний, наприклад подвійний, інтеграл з фіксованими межами має вид:

$$\int_a^b \int_c^d f(x, y) dx dy$$

Наступний приклад з подвійним звертанням до функції integrate виконує обчислення подвійного невизначеного інтеграла:

(C18) integrate(integrate(x^3+y^3, x), y);

$$(D18) \quad \frac{x^4 y}{4} + \frac{xy^4}{4}$$

В Maxima можна задавати і нескінченні межі інтегрування. Для позначення нескінченності використовується змінна inf:

(C19) integrate(1/x^2, x, 1, inf);

(D19) 1

(C20) integrate(1/(1+x^2), x, -inf, inf);

(D20) %pi

```
(C21) integrate(1/x, x, 0, inf);
Integral is divergent -- an error.
Quitting. To debug this try DEBUGMODE(true);)
```

В останньому прикладі система повідомила про неможливість обчислення інтеграла, тому що він розходиться (*is divergent*).

Під час обчислення досить складних інтегралів відповідь не завжди буде представлена у найбільш простому вигляді. У наступному прикладі Maxima не може в символьному виді одержати відповідь, рівну  $\pi/4$ :

```
(C22) g:1/sqrt(2-x^2);
```

$$(D22) \quad \frac{1}{\sqrt{2-x^2}}$$

```
(C23) integrate(g, x, 0, 1);
```

$$(D23) \quad \operatorname{asin}\left(\frac{\sqrt{2}}{2}\right)$$

Існують особливі випадки обчислення інтегралів, коли при обчисленні потрібно враховувати деякі умови.

```
(C38) integrate(1/(sqrt(1-x^n)), x, 0, 2);
```

Is n an integer?

Потрібно вибрати один із трьох варіантів відповіді:

Acceptable answers are Yes, Y, No, N, Unknown, Uk

Y;

Далі дати ще одну відповідь про те, чи є  $n$  позитивним, негативної або рівним 0

Is n positive, negative, or zero?

positive;

У результаті одержимо один із можливих розв'язків:

$$(D38) \quad \int_0^2 \frac{1}{\sqrt{1-x^n}} dx$$

Серед операцій математичного аналізу насамперед треба відзначити обчислення сум:

$$\sum_{i=\min}^{\max} f_i.$$

У цих операціях індекс  $i$  приймає цілі значення від мінімального (початкового)  $\min$  до максимального (кінцевого)  $\max$  із кроком, рівним  $+1$ . Суми і добутки легко обчислюються математичними системами, такі обчислення просто описуються усіма мовами програмування. Однак важливою рисою системи символьної математики є обчислення сум і добутків в аналітичному вигляді (якщо це можливо) при великій кількості членів – аж до нескінченності.

Для обчислення скінченних і нескінченних сум варто записати суму в символьному вигляді, після чого спростити отриманий вираз:

```
(C24) sum(1/n^2, n, 1, inf);
```

$$(D24) \quad \sum_{n=1}^{\infty} \frac{1}{n^2}$$

```
(C25) %,simpsum;
```

$$(D25) \quad \pi^2/6$$

Операція обчислення добутків  $\prod_{i=\min}^{\max} f_i$  представлена функцією `product`.

Наприклад, обчислимо добуток  $\prod_1^5 x+i^2$ :

```
(C63) product(x+i^2, i, 1, 5);
```

$$(D63) \quad (x+1)(x+4)(x+9)(x+16)(x+25)$$

```
(C63) expand(%);
```

(D63)  $x^5 + 55x^4 + 1023x^3 + 7645x^2 + 21076x + 14400$

Багато функцій під час наближення аргументу до деякого значення чи до деякої області значень прямують до визначеної *границі*. Так, функції  $\sin(x)/x$  при  $x$ , що прямує до нуля ( $x \rightarrow 0$ ), відповідає границя 1 у вигляді усуваної невизначеності 0/0.

Границею деяких функцій може бути нескінченність, тоді як багато функцій прямують до скінченної границі при аргументі  $x$ , який прямує до нескінченності. Система Maxima не тільки чисельно знаходить границі функцій, заданих аналітично, але і дає змогу знайти границі у вигляді математичного виразу. Для обчислення границь використовується функція `limit`.

(C28) `limit(1/x, x, inf);`

(D28) 0

Для обчислення односторонніх границь використовується додатковий параметр, який приймає значення `plus` для обчислення межі праворуч і `minus` – ліворуч.

Наприклад, дослідимо на неперервність функцію  $\arctg(1/(x-4))$ . Ця функція не визначена в точці  $x=4$ . Обчислимо границі справа і зліва:

(C28) `limit(atan(1/(x-4)), x, 4, plus);`

(D28)  $\frac{\%pi}{2}$

(C29) `limit(atan(1/(x-4)), x, 4, minus);`

(D29)  $\frac{\%pi}{2}$

Як бачимо, точка  $x=4$  є точкою розриву I роду для цієї функції, тому що існують границі зліва і справа, рівні  $-\pi/2$  і  $\pi/2$  відповідно.

№	Функція	Призначення
1	<code>diff</code>	Обчислення похідної від функції
2	<code>integrate</code>	Обчислення інтегралу
3	<code>sum</code>	Обчислення скінчених і нескінчених сум
4	<code>product</code>	Обчислення добутку
5	<code>limit</code>	Обчислення границь

## 2.7. Побудова графіків і поверхонь

Для побудови графіків у wxMaxima можна використовувати функції;

`plot2d` – побудова графіків на площині (функція із Maxima);

`plot3d` – побудова тривимірних графіків (функція із Maxima);

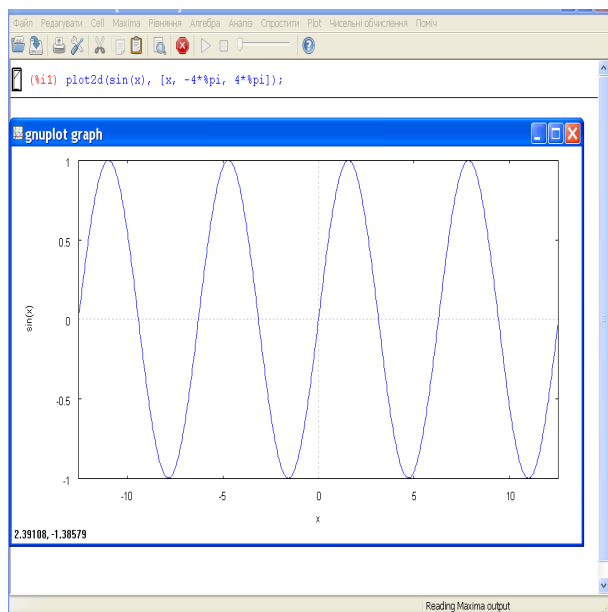
`wxplot2d` – побудова графіків на площині (функція із wxMaxima);

`wxplot3d` – побудова тривимірних графіків (функція із wxMaxima).

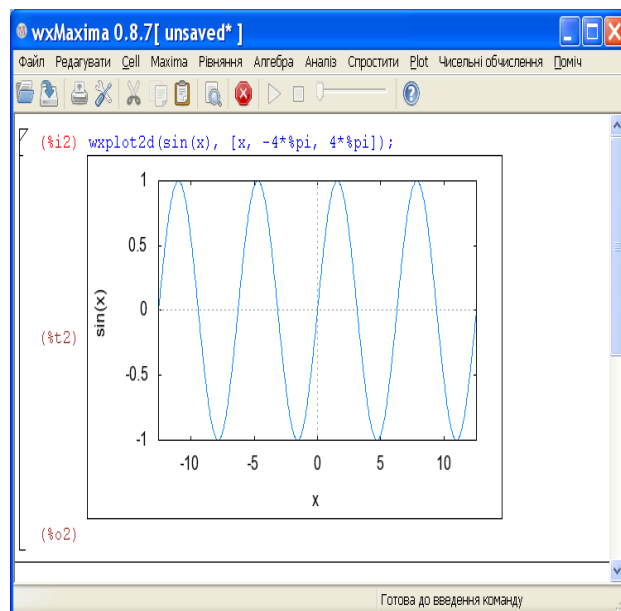
Функція `plot2d(f(x), [x, a, b])` або `plot2d(f(x), [x, a, b], [y, c, d])` в окремому вікні будує графік залежності  $y=f(x)$  на відрізку  $[a, b]$ . Параметр  $[y, c, d]$  задає розмір вікна графіка по осі  $y$ . Якщо цей параметр не задано, то масштаб побудови графіка вибирається автоматично. Функція `wxplot2d` будує графік у вікні wxMaxima. На рис. 2.6 зображений графік функції  $y=\sin x$  на відрізку  $[-4\pi, 4\pi]$ . За допомогою опції `parametric` можна будувати графіки функцій, заданих неявно. На рис. 2.7 зображений графік неявно заданої функції за допомогою команди

`plot2d ([parametric, cos(t), sin(t), [t, -%pi, %pi], [nticks, 80]], [x, -4/3, 4/3])`

Опція `[nticks, 80]` задає кількість точок на кривій, опція `[x, -4/3, 4/3]` задає розмір вікна графіка по горизонтальній осі.



а



б

Рис. 2.6. а – побудова графіка за допомогою `plot2d`, б – побудова графіка за допомогою `wxplot2d`

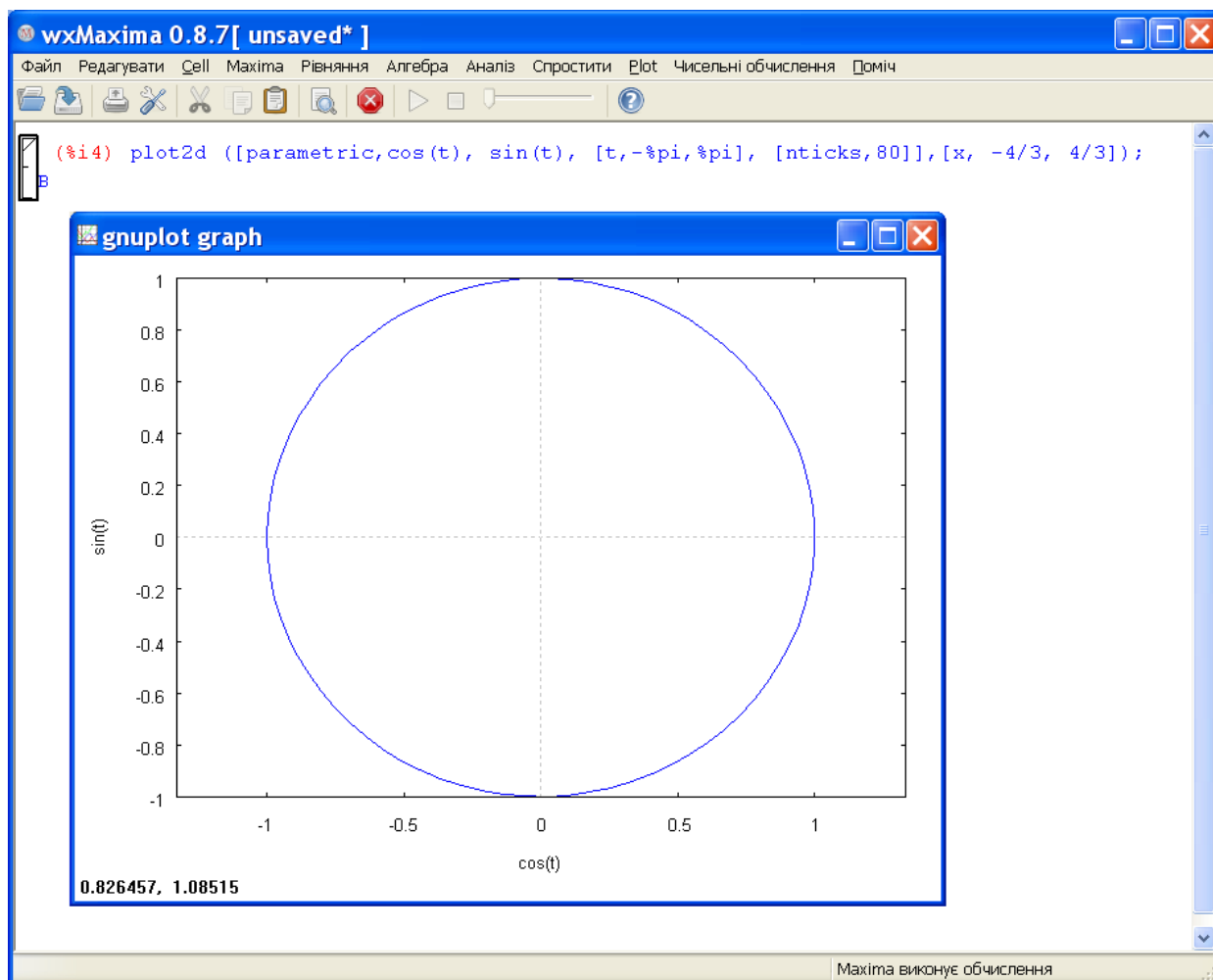


Рис. 2.7. Побудова графіка параметрично заданої функції за допомогою `plot2d`

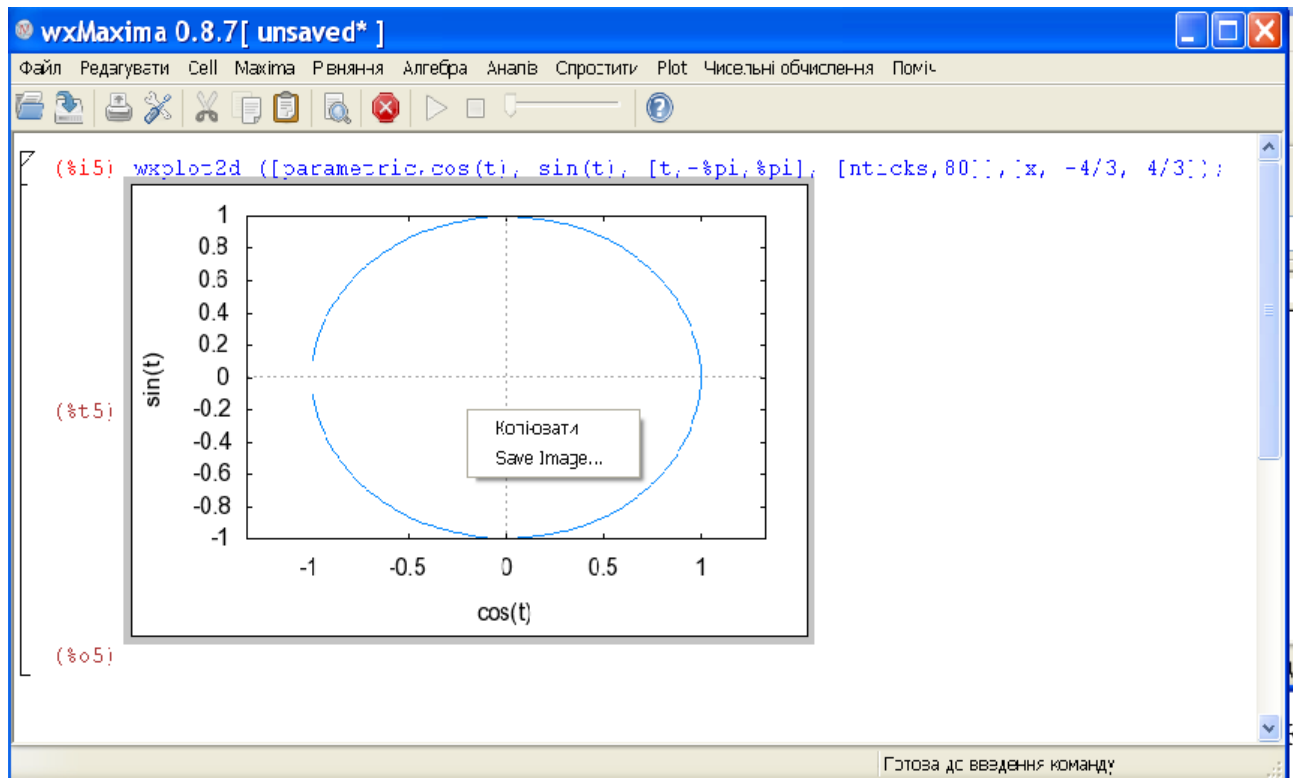


Рис. 2.8. Побудова графіка параметрично заданої функції за допомогою wxplot2d

Графіки, побудовані функцією wxplot2d за допомогою контекстного меню, можна скопіювати в буфер обміну або ж зберегти у графічному файлі з розширенням .png (див. рис. 2.8).

Для побудови поверхонь можна використовувати функції plot3d та wxplot3d. На рис. 2.9 зображена поверхня, побудована за допомогою команди

`plot3d (2^(-u^2 + v^2), [u, -3, 3], [v, -2, 2]);`

а на рис. 2.10 ця ж поверхня, побудована командою

`wxplot3d (2^(-u^2 + v^2), [u, -3, 3], [v, -2, 2]);`

Більш докладно побудова графіків і поверхонь описана в [6].

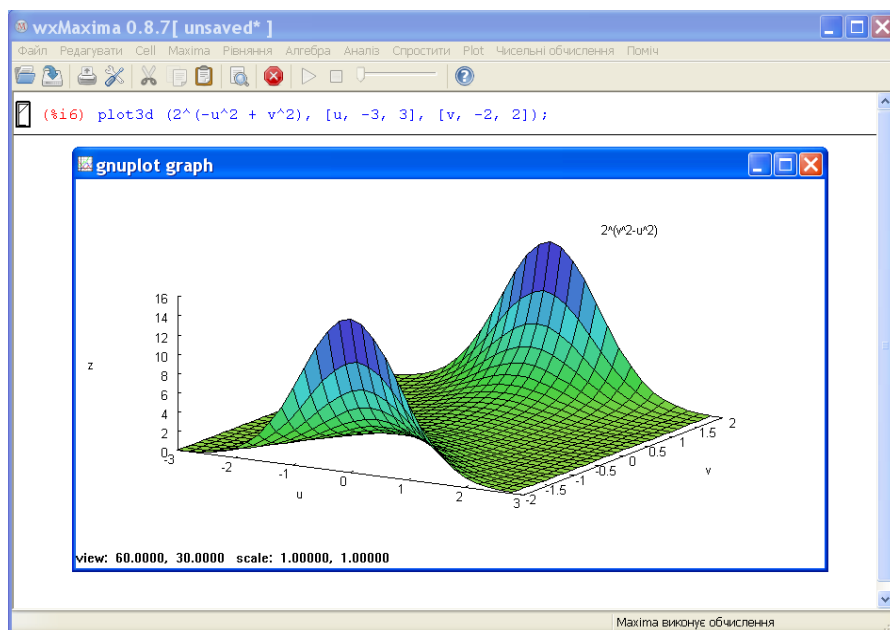


Рис. 2.9.

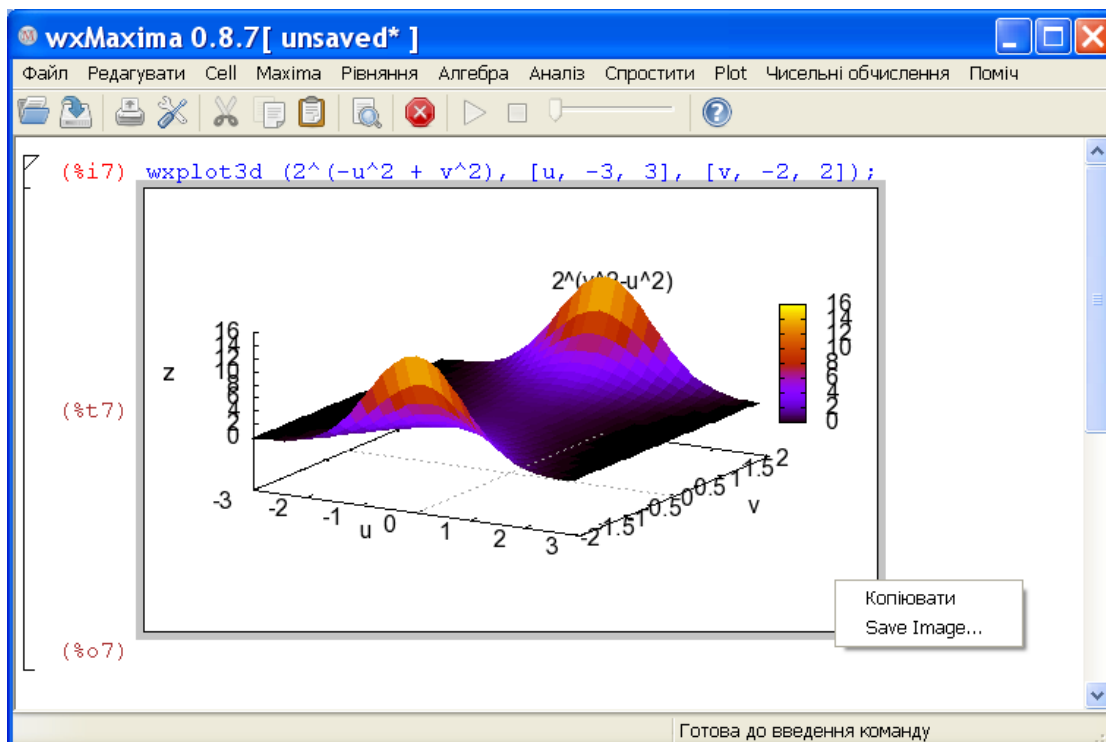


Рис. 2.10.

## 2.8. Матричні обчислення

*Матриця* – прямокутна двовимірна таблиця, яка містить  $m$  рядків і  $n$  стовпців елементів, кожен з яких може бути представлений числом, сталою, змінною, символьним чи математичним виразом.

*Квадратна матриця* – матриця, у якій число рядків  $m$  дорівнює числу стовпців  $n$ .

*Сингулярна (вироджена) матриця* – квадратна матриця, у якій детермінант (визначник) дорівнює 0. Така матриця звичайно не спрощується при символьних обчисленнях.

*Одинична матриця* – це квадратна матриця, у якій діагональні елементи рівні 1, а інші елементи рівні 0.

*Транспонована матриця* – квадратна матриця, у якій стовпці і рядки міняються місцями.

*Обернена матриця* – це матриця  $M^{-1}$ , що, будучи помножена на вихідну квадратну матрицю  $M$ , дає одиничну матрицю.

*Східчаста* форма матриці відповідає умовам, коли перший ненульовий елемент у кожному рядку є 1 і перший ненульовий елемент кожного рядка з'являється праворуч від першого елемента в попередньому рядку, тобто всі елементи нижче першого ненульового у рядку – нулі.

*Ранг* матриці – найбільший з порядків відмінних від нуля мінорів квадратної матриці.

*Слід* матриці – сума діагональних елементів квадратної матриці.

*Визначник* матриці – це поліном від елементів квадратної матриці, кожен член якого є добутком  $n$  елементів, узятих по одному з кожного рядка і кожного стовпця зі знаком добутку, заданим парністю перестановок:

$$\det A = \sum a_{1j} (-1)^{j+1} M_1^{<j>},$$

де  $M_1^{<j>}$  – визначник матриці порядку  $n-1$ , отриманої з матриці  $A$  викреслюванням першого рядка і  $j$ -го стовпця.

В Maxima реалізовані матричні операції. У наступному прикладі задаються дві матриці, що потім складаються (+) і перемножуються (.):

(C1) `A:matrix([1,2],[3,4]);`

(D1) 
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

(C2) `B:matrix([1,1],[1,1]);`

(D2) 
$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

(C3) `A + B;`

(D3) 
$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$$

(C4) `A . B;`

(D4) 
$$\begin{bmatrix} 3 & 3 \\ 7 & 7 \end{bmatrix}$$

Функція `determinant` обчислює визначник матриці.

(C5) `determinant(A);`

(D5) 
$$-2$$

(C6) `determinant(matrix([a,b],[c,d]));`

(D6) 
$$a d - b c$$

Транспонування матриці здійснюється функцією `transpose`.

(C7) `transpose(A);`

(D7) 
$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

Для одержання оберненої матриці використовують операцію  $^{-1}$  або функцію `invert`.

(C8) `A-1;`

(D8) 
$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

(C9) `invert(A);`

(D9) 
$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

Як відомо, кожен елемент  $b_{ij}$  оберненої матриці  $B=A^{-1}$  отримується діленням алгебраїчного доповнення  $A_{ij}$  відповідного елемента вихідної матриці на її визначник  $|A|$ . Для того, щоб винести  $1/|A|$  як співмножник, застосовується функція `detout`.

(C10) `invert(A), detout;`

(D10) 
$$-\frac{\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}}{2}$$

Переконаємося в правильності отриманого результату, помноживши  $A$  на обернену до неї матрицю:

(C11) `A . d9;`

(D11) 
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Будьте уважні: внаслідок виконання операції  $^{-1}$  отримається матриця, кожен елемент якої буде обернений до елемента вихідної, а не обернена матриця.

(C12)  $A^{-1}$ ;

(D12) 
$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} \end{bmatrix}$$

Використання матриць дає змогу розв'язувати системи лінійних рівнянь. Нехай  $A$  – матриця коефіцієнтів системи,  $X$  – матриця невідомих,  $B$  – матриця вільних членів системи. Тоді матриця  $X$  знаходиться за формулою  $X=A^{-1} \cdot B$ , де операція “ $\cdot$ ” означає матричне множення.

Наприклад, розв'яжемо наступну систему рівнянь матричним способом.

$$AX=B, A=\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 3 & 1 \end{bmatrix}, B=\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Спочатку заповнимо відповідні матриці, а потім одержимо матрицю результатів:

(C14) `A:matrix([1, 2, 1], [2, 1, 1], [1, 3, 1]);`

(D14) 
$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 3 & 1 \end{bmatrix}$$

(C15) `B:matrix([0, 1, 0]);`

(D15) 
$$\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

(C16) `(A-1).B;`

(D16) 
$$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Функція `ematrix(m,n,x,i,j)` створює матрицю, у якій всі елементи дорівнюють нулю, крім елемента на позиції  $i,j$ , який має значення  $x$ .

(C17) `ematrix(3,3,2,1,1);`

(D17) 
$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Функція `diagmatrix(n,x)` утворить діагональну матрицю розміром  $n \times n$  і діагональним елементом, рівним  $x$ .

(C18) `diagmatrix(3, 1);`

(D19) 
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Функція `ident(n)` формує діагональну матрицю.

(C20) `ident(3);`

(D20) 
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Функція `minor(m,i,j)` формує матрицю, в якій видаляє рядок  $i$  і стовпець  $j$ .

(C21) `A:matrix([1,2], [3,4]);`



$$(D21) \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

(C22) `minor(A,1,1);`

$$(D22) \quad [ \ 4 \ ]$$

Функція `row(m,i)` повертає з матриці  $m$   $i$ -ий рядок.

(C23) `A:matrix([1,0,3],[3,4,3],[5,6,7]);`

$$(D23) \quad \begin{bmatrix} 1 & 0 & 3 \\ 3 & 4 & 3 \\ 5 & 6 & 7 \end{bmatrix}$$

(C24) `row(A,3);`

$$(D24) \quad [ \ 5 \ 6 \ 7 \ ]$$

Функція `zeromatrix(m,n)` формує нульову матрицю розміром  $m \times n$ .

(C25) `zeromatrix(3,3);`

$$(D25) \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Функція `col(m,j)` повертає  $j$ -ий стовпець матриці  $m$ .

(C26) `col(A,1);`

$$(D26) \quad \begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$$

Функція `echelon(M)` здійснює перетворення матриці методом виключення Гауса до трикутної форми з нормуванням діагональних елементів до 1 (аналогічний результат дає функція `triangularize(M)`, але без нормування діагональних елементів):

(%i1) `a:matrix([1,2,3],[4,5,x],[6,7,y]);`

$$(\%o1) \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & x \\ 6 & 7 & y \end{bmatrix}$$

(%i2) `b:echelon(a);`

$$(\%o2) \quad \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & -\frac{x-12}{3} \\ 0 & 0 & 1 \end{bmatrix}$$

(%i3) `c:triangularize(a);`

$$(\%o3) \quad \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & x-12 \\ 0 & 0 & -3y+5x-6 \end{bmatrix}$$

Для обчислення рангу матриці (ранг матриці рівний розміру найбільшого невикористаного мінора матриці) використовують функцію `rank`, наприклад:

(%i4) `b:matrix([1,1],[2,2],[3,3],[4,5]);`

$$(\%o4) \quad \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \\ 4 & 5 \end{bmatrix}$$

```
(%i5) rank(b);
(%o5) 2
```

Лінійні дії над векторами в системі Maxima задаються у вигляді  $a+b$ ,  $a-b$ ,  $\alpha*v$ , де  $v$  – деяка константа.

$a*b*c*...$  – повертає векторний добуток векторів. `innerproduct(a,b,c,...)` або  $a.b.c...$  – повертає скалярний добуток векторів.

```
(%i24) a:[1,2,3];
(%o24) [1, 2, 3]
```

```
(%i25) b:[4,5,6];
(%o25) [4, 5, 6]
```

```
(%i29) a*b;
(%o29) [4, 10, 18]
```

```
(%i30) a.b;
(%o30) 32
```

Рис. 2.11. Векторний і скалярний добуток векторів

Результат скалярного добутку векторів визначається за формулою:  
 $a.b = a_1*b_1 + a_2*b_2 + a_3*b_3$

Результат векторного добутку векторів визначається за формулою:  
 $a*b = [a_1*b_2 - a_2*b_1, a_2*b_3 - a_3*b_2, a_3*b_1 - a_1*b_3]$

Змішаний добуток векторів  $a, b, c$  можна задавати за допомогою

`matrix:`

```
(C28) v:matrix(a,b,c);
```

```
(D28) 
$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```

`determinant(v)` – повертає визначник матриці  $v$ , рядками якої є координати векторів  $a, b, c$  відповідно.

№	Функція	Призначення
1	<code>determinant</code>	Обчислення визначника матриці
2	<code>transpose</code>	Транспонування матриці
3	<code>invert</code> або $^{-1}$	Обчислення оберненої матриці
4	<code>detout</code>	Винести $1/ A $ як співмножник
5	<code>ematrix(m,n,x,i,j)</code>	Створення нульової матриці з елементом рівним $x$ на позиції $i,j$ .
6	<code>diagmatrix(n,x)</code>	Створення діагональної матриці розміром $n \times n$ із значенням $x$ на діагоналі.
7	<code>ident(n)</code>	Створення одиничної діагональної матриці
8	<code>minor(m,i,j)</code>	Видалення із заданої матриці $i$ -того рядка та $j$ -того стовпця (створення мінора)
9	<code>row(m,i)</code>	Повертає з матриці $m$ $i$ -тий рядок
10	<code>zeromatrix(m,n)</code>	Створення нульової матриці розміром $m \times n$ .
11	<code>echelon(M)</code>	Приведення матриці до трикутної форми методом виключення Гауса з нормуванням діагональних елементів

12	triangularize(M)	Приведення матриці до трикутної форми методом виключення Гауса без нормування діагональних елементів
13	rank(M)	Обчислення рангу матриці
14	a*b*c	Обчислення векторного добутку векторів
15	innerproduct(a,b,c,...) або a.b.c...	Обчислення скалярного добутку векторів

## 2.9. Списки та масиви

Списки мають велике значення для обробки масивів даних і є основою для створення векторів і матриць.

Часто математичні чи інші об'єкти містять множину даних, яку бажано об'єднувати під загальним ім'ям. Наприклад, під об'єктом з ім'ям М можна мати на увазі квадратну матрицю розміром 10x10 із загальним числом елементів, рівним 100. Людину з ім'ям Victor, наприклад, можна характеризувати цілим списком різних даних – символьними прізвищем, ім'ям і по батькові, роком народження (цілим числом), ростом, вагою (дійсними числами) тощо.

Для об'єднання даних можуть використовуватися *списки* (list). Maxima має великі можливості роботи з об'єктами-списками, які містять не тільки однотипні, а й різнотипні елементи. Зокрема, елементами списків можуть бути числа, константи, змінні, вирази і навіть самі списки. Списки використовують для конструювання інших типів даних – масивів, матриць і векторів.

Мовою системи Maxima список – це сукупність довільних даних у квадратних дужках, наприклад:

```
(C1) victor:[petrov,victor,victorovich,1.68,100];
```

```
(D1) [petrov, victor, victorovich, 1.68, 100]
```

Елементом списку може бути інший список

```
(%i2) list4:[1,2,[3,4],[5,6,7]];
```

```
(%o2) [1,2,[3,4],[5,6,7]]
```

Посилання на елемент списку здійснюється за номером

```
%i3) list4[1];
```

```
(%o3) 1
```

```
%i4) list4[3];
```

```
(%o4) [3,4]
```

Списки характеризуються *розміром*, який є добутком числа елементів за кожним напрямком (*розмірністю*). Наприклад, одновимірний список є *вектором* і характеризується числом елементів за одним напрямком. Вектор може бути рядком або вектором-стовпцем. Двовимірний список є *матрицею*.

Для генерації списків з елементами, які є дійсними і цілими числами, чи навіть цілими виразами, використовується функція makelist, яка створює список.

Наступний приклад генерує список f із 15 елементів, заповнений випадковими цілими числами в діапазоні від 0 до 10.

```
(C2) f:makelist(random(10),c,1,15);
```

```
(D2) [9, 0, 1, 0, 3, 2, 3, 9, 5, 1, 2, 0, 9, 3, 6]
```

Приклад генерації списку з дійсними числами:

```
(C3) f:makelist(random(10)/3.1,c,1,15);
```

```
(D3) [1.290322580645161, 0.64516129032258,
2.258064516129032, 0.96774193548387, 2.903225806451613,
2.580645161290323, 1.935483870967742, 0.0,
2.903225806451613, 2.580645161290323, 0.64516129032258,
```

0.64516129032258, 0.96774193548387, 0.64516129032258, 0.96774193548387]

Приклад генерації списку, який складається з десяти букв а.

(C4) f: makelist(a, c, 1, 10);

(D4) [a, a, a, a, a, a, a, a, a, a]

Списки відносяться до даних складної структури. Тому під час роботи з ними виникає необхідність контролю за структурою, інакше застосування списків може привести до грубих помилок, як явних, що супроводжуються видачею повідомлення про помилку, так і неявних.

listp(exp) – повертає true, якщо exp є списком, інакше повертає false

(C5) listp(f);

(D6) true

(C6) a: 3;

(D6) 3;

(C7) listp(a);

(D7) false

length(exp) – повертає число елементів одновимірному списку exp або число розмірностей у випадку багатовимірному списку.

(C8) length(f);

(D8) 15

member(exp, list) – виводить true, якщо змінна exp входить у список list, false – якщо змінна exp не входить у список list.

(C9) member(petrov, victor);

(D9) true

(C10) member(petrov1, victor);

(D10) false

Списки можна зобразити особливою структурою даних – стеком. Стек – це структура даних, яка нагадує стос тарілок у шафі (тарілки відіграють роль даних). Чергову тарілку можна покласти тільки згори (на *вершину* стеку). На *дні* стеку лежить перша поміщена в нього тарілка. Стек підпорядковується наступному правилу: останнє введенне значення вилучається першим, а перше введенне значення вилучається останнім. Стек відноситься до структур даних динамічного типу, його розміри змінюються в процесі обчислень. Стек може бути порожнім, якщо з нього вилучити всі дані.

Maxima надає наступні можливості роботи зі стеками:

cons(exp, list) – додає в початок списку list елемент exp:

(C11) cons(35, victor(petrov, victor, victorovich, 1.68, 100));

(D11) [35, petrov, victor, victorovich, 1.68, 100]

endcons(exp, list) – додає в кінець списку list елемент exp:

(C12) endcons(medik, victor);

(D12) [petrov, victor, victorovich, 1.68, 100, medik]

last(list) – повертає останній елемент списку list:

(C13) last(victor);

(D13) 100

first(list) – повертає перший елемент списку list:

(C14) first(victor);

(D14) petrov

rest(exp, n) – видаляє зі списку n елементів:

(C15) rest(victor, 1); // видаляє один елемент списку

(D15) [victor, victorovich, 1.68, 100]

(C16) rest(victor, 5); // видаляє всі елементи списку

(D16) []

Тривіальна процедура додати/вилучити дані обмежує можливості стекових операцій. З життєвого досвіду ми знаємо, що, виявивши наполегливість, можна вставити тарілку і в

середину стоса. Maxima надає ряд розширених можливостей для роботи зі списками, які виходять за рамки звичайних стекових операцій.

Так, наприклад, для розширення списку шляхом внесення в нього нових елементів використовуються наступні функції:

`append(list1, list2, list3, ...)` – поєднує списки в зазначеному порядку:

(C17) `append(f, victor);`

(D17) `[0.32258064516129, 2.580645161290323, 2.903225806451613, 0.64516129032258, 2.903225806451613, 2.258064516129032, 1.612903225806452, 1.935483870967742, 1.612903225806452, 0.0, 1.935483870967742, 2.903225806451613, 0.0, 0.32258064516129, 0.32258064516129, petrov, victor, victorovich, 1.68, 100]`

`delete(exp1, exp2)` – видаляє елемент `exp1` зі списку `exp2`:

(C18) `delete(petrov, victor);`

(D18) `[victor, victorovich, 1.68, 100]`

Крім додавання в список нових даних, є можливість зміни порядку елементів у списку.

`reverse(list)` – повертає список зі зворотним порядком розташування елементів:

(C19) `reverse(victor);`

(D19) `[100, 1.68, victorovich, victor, petrov]`

`sort(list)` – сортує елементи списку `list` у канонічному порядку:

(C20) `sort(victor);`

(D20) `[1.68, 100, petrov, victor, victorovich]`

(C21) `sort(f);`

(D21) `[0.0, 0.0, 0.32258064516129, 0.32258064516129, 0.32258064516129, 0.64516129032258, 1.612903225806452, 1.612903225806452, 1.935483870967742, 1.935483870967742, 2.258064516129032, 2.580645161290323, 2.903225806451613, 2.903225806451613, 2.903225806451613]`

Сукупність даних утворює *масив* (array). Масиви можуть бути одновимірними (один список), двовимірними і багатовимірними (два чи більш списків). Одновимірні масиви називають *векторами*, двовимірні – *матрицями*. У загальному випадку масив характеризується *розмірністю* (числом вимірів) і *розміром* – числом елементів за усіма вимірами.

Масив створюють оператором `array(name, exp1, exp2, exp3...)`

(C29) `d:array[1,2,3,4,5,6,7,8,9,0,1,2];`

(D29) `array1,2,3,4,5,6,7,8,9,0,1,2`

№	Функція	Призначення
1	<code>makelist</code>	Створює список
2	<code>listp(exp)</code>	Повертає true, якщо <code>exp</code> є списком, інакше повертає false
3	<code>length(exp)</code>	Повертає число елементів одновимірного списку <code>exp</code> чи число розмірностей у випадку багатовимірного списку
4	<code>member(exp, list)</code>	Виводить true, якщо змінна <code>exp</code> входить у список <code>list</code> , false - якщо змінна <code>exp</code> не входить у список <code>list</code>
5	<code>cons(exp, list)</code>	Додає в початок списку <code>list</code> елемент <code>exp</code>

6	<code>endcons(exp, list)</code>	Додає в кінець списку <code>list</code> елемент <code>exp</code>
7	<code>last(list)</code>	Повертає останній елемент списку <code>list</code>
8	<code>first(list)</code>	Повертає перший елемент списку <code>list</code>
9	<code>rest(exp, n)</code>	Видаляє зі списку <code>n</code> елементів
10	<code>append(li1, li2, li3, ...)</code>	Об'єднує списки в зазначеному порядку
11	<code>delete(exp1, exp2)</code>	Видаляє елемент <code>exp1</code> зі списку <code>exp2</code>
12	<code>reverse(list)</code>	Повертає список зі зворотним порядком розташування елементів
13	<code>sort(list)</code>	Сортує елементи списку <code>list</code> у канонічному порядку
14	<code>array(name, exp1, exp2, exp3 ...)</code>	Оголошення масиву

## 2.10. Розв'язування дифрівнянь в Maxima

*Дифрівнянням* називають рівняння, до складу яких входять похідні функції  $y(x)$ , що задають розв'язок рівняння. Дифрівняння можуть бути задані в різній формі, наприклад, у загальновідомій формі Коші:

$$y'(x) = f(x, y)$$

Найвищий порядок похідної або диференціала, що входять у дифрівняння, називають *порядком* цього дифрівняння.

Наприклад,  $ydx - xdy = 0$  – дифрівняння першого порядку,  $2xy'' - 8y^4y' = 16x^6$  – дифрівняння другого порядку,  $y''' - 6y'' + 5y' = 10x^8e^x$  – дифрівняння третього порядку.

Дифрівняння і системи дифрівнянь можуть бути лінійними і нелінійними. Для лінійних рівнянь звичайно існують аналітичні розв'язки. Нелінійні дифрівняння зазвичай аналітичних розв'язків не мають, але можуть розв'язуватися наближеними чисельними методами.

Дифрівняння широко використовують в практиці математичних обчислень. Вони є основою розв'язання задач моделювання – особливо в динаміці.

В Maxima передбачені засоби розв'язання задачі Коші для систем звичайних диференціальних рівнянь, заданих як в явній формі

$$dx/dt = F(t, x),$$

так і в неявній

$$Mdx/dt = F(t, x),$$

де  $M$ -матриця,  $x$  - вектор,  $F(t, x)$  - вектор-функція.

Для розв'язання звичайних дифрівнянь першого і другого порядку можна використовувати функцію `ode2`:

`ode2 (дифрівняння, змінна, незалежна_змінна)`

Нагадаємо, що розв'язки однорідних рівнянь виду  $x'' + a * x' + b * x = 0$  шукають за результатами розв'язання характеристичного рівняння  $r^2 + ar + b = 0$ . Можливі такі варіанти комбінацій його коренів  $r_1, r_2$ :

$r_1, r_2$  - дійсні і різні. Розв'язок і його першу похідну записують у формі  $x = k_1 * e^{r_1 * t} + k_2 * e^{r_2 * t}$ ,  $x' = k_1 * r_1 * e^{r_1 * t} + k_2 * r_2 * e^{r_2 * t}$ .

Корені дійсні і однакові  $r_1 = r_2$ . Розв'язок і його першу похідну записують у формі  $x = (k_1 + k_2 * t) * e^{r_1 * t}$ ,  $x' = k_2 * e^{r_1 * t} + (k_1 + k_2 * t) * r_1 * e^{r_1 * t}$ .

Корені  $r_1, r_2$  - комплексні. Якщо  $r_1 = \alpha + \beta i$ ,  $r_2 = \alpha - \beta i$ , то розв'язок і його першу похідну записують у формі  $x = e^{\alpha t} (k_1 * \cos(\beta t) + k_2 * \sin(\beta t))$ ,  $x' = \alpha e^{\alpha t} (k_1 * \cos(\beta t) + k_2 * \sin(\beta t)) + e^{\alpha t} (-k_1 * \sin(\beta t) + \beta * k_2 * \cos(\beta t))$ .

Для визначення числових значень коефіцієнтів  $k_1$ ,  $k_2$  використовують початкові умови  $x(0)$ ,  $x'(0)$ :

$r_1, r_2$  - дійсні і різні

$$k_1 + k_2 = x(0);$$

$$k_1 r_1 + k_2 r_2 = x'(0).$$

Корені дійсні і однакові  $r_1 = r_2$

$$k_1 = x(0);$$

$$k_2 + k_1 r_1 = x'(0).$$

Корені  $r_1, r_2$  — комплексні

$$k_1 = x(0);$$

$$\alpha k_1 + \beta k_2 = x'(0).$$

Розв'язавши одну із цих систем лінійних рівнянь відносно  $k_1$  та  $k_2$ , отримаємо частковий розв'язок диференціального другого порядку. Алгоритм знаходження розв'язку диференціального  $x'' + a x' + b x = 0$  можна записати так:

1. для заданого диференціального побудувати характеристичне рівняння  $r^2 + ar + b = 0$ ;
2. знайти корені характеристичного рівняння;
3. на підставі початкових умов знайти числові значення коефіцієнтів  $k_1$ ,  $k_2$  з відповідної системи лінійних рівнянь;
4. отримані значення коренів  $r_1, r_2$  та коефіцієнтів  $k_1$ ,  $k_2$  підставити у відповідний вираз для  $x(t)$ .

Розглянемо два приклади.

Приклад 1.

$$x'' + 4x' + x = 0, x(0)=5, x'(0)=0$$

Характеристичне рівняння  $r^2 + 4r + 1 = 0$  має корені дійсні корені

$$r_1 = -2 - \sqrt{3}$$

$$r_2 = -2 + \sqrt{3}$$

Загальний розв'язок рівняння такий:

$$x = k_1 e^{-(2+\sqrt{3})t} + k_2 e^{-(2-\sqrt{3})t}$$

Врахування початкових умов дає систему із двох лінійних рівнянь

$$k_1 + k_2 = x(0)$$

$$k_1 r_1 + k_2 r_2 = x'(0)$$

За правилом Крамера її розв'язки виглядають так:

$$k_1 = \frac{\begin{vmatrix} x(0) & 1 \\ x'(0) & r_2 \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ r_1 & r_2 \end{vmatrix}} \quad \text{або} \quad k_1 = \frac{x(0)r_2 - x'(0)}{r_2 - r_1},$$

$$k_2 = \frac{\begin{vmatrix} 1 & x(0) \\ r_1 & x'(0) \end{vmatrix}}{\begin{vmatrix} 1 & 1 \\ r_1 & r_2 \end{vmatrix}} \quad \text{або} \quad k_2 = \frac{x'(0) - x(0)r_1}{r_2 - r_1}.$$

Остаточно

$$k_1 = \frac{5(-2 + \sqrt{3})}{2\sqrt{3}}, \quad k_2 = \frac{-5(-2 - \sqrt{3})}{2\sqrt{3}}.$$

Частковий розв'язок диференціального виглядає так

$$x(t) = \frac{5(-2 + \sqrt{3})}{2\sqrt{3}} \exp((-2 - \sqrt{3})t) + \frac{-5(-2 - \sqrt{3})}{2\sqrt{3}} \exp((-2 + \sqrt{3})t),$$

Порівнюємо отриманий розв'язок з результатами обчислень у wxMaxima.

Функція ode2 (dr1, x, t); знаходить загальний розв'язок однорідного рівняння. Функція ic2(%t=0,x=5,diff(x,t)=0); за заданими початковими умовами визначає коефіцієнти k1, k2. Легко бачити, що розв'язок, обчислений нами, співпадає з розв'язком, отриманим у wxMaxima.

```
(%i1) dr1: 'diff (x, t, 2) +4*' diff (x, t)+x = 0;
(%o1)  $\frac{d^2}{dt^2}x + 4\left(\frac{d}{dt}x\right) + x = 0$ 

(%i2) ode2 (dr1, x, t);
(%o2)  $x = \%k1 e^{\frac{(2\sqrt{3}-4)t}{2}} + \%k2 e^{\frac{(-2\sqrt{3}-4)t}{2}}$ 

(%i3) ic2(%t=0,x=5,diff(x,t)=0);
(%o3)  $x = \frac{(10\sqrt{3}+15)}{6} e^{\frac{(2\sqrt{3}-4)t}{2}} - \frac{(10\sqrt{3}-15)}{6} e^{\frac{(-2\sqrt{3}-4)t}{2}}$ 
```

Приклад 2.

$$x'' + 0.4 * x' + x = 0, x(0)=5, x'(0)=0$$

Характеристичне рівняння  $r^2 + 0.4r + 1 = 0$  має комплексні корені

$$r1 = -0.2 - \sqrt{(0.04-1)} \quad \text{або} \quad r1 = -0.2 + i\sqrt{(0.96)}$$

$$r2 = -0.2 + \sqrt{(0.04-1)} \quad \text{або} \quad r2 = -0.2 - i\sqrt{(0.96)}$$

Врахування початкових умов дає систему із двох лінійних рівнянь

$$k1 = 5$$

$$-0.2*k1 + \sqrt{0.96}*k2 = 0 \quad \text{або} \quad k2 = 1/\sqrt{0.96}$$

Частковий розв'язок дифрівняння виглядає так

$$x = e^{-0.2t} (5*\cos(t*\sqrt{0.96}) + 1/\sqrt{0.96}*\sin(t*\sqrt{0.96}))$$

Порівнюємо отриманий розв'язок з результатами обчислень у wxMaxima.

Функція ode2 (dr1, x, t); знаходить загальний розв'язок однорідного рівняння. Функція ic2(%t=0,x=5,diff(x,t)=0); за заданими початковими умовами визначає коефіцієнти k1, k2. Легко бачити, що розв'язок, обчислений нами, співпадає з розв'язком, отриманим у wxMaxima ( $\frac{2*\sqrt{(6)}}{5} = \sqrt{(0.96)}$ )

```
(%i5) dr2: 'diff (x, t, 2) +0.4*' diff (x, t)+x = 0;
(%o5)  $\frac{d^2}{dt^2}x + 0.4\left(\frac{d}{dt}x\right) + x = 0$ 

(%i6) ode2 (dr2, x, t);
rat: replaced 0.4 by 2/5 = 0.4
(%o6)  $x = \%e^{-\frac{t}{5}} \left( \%k1 \sin\left(\frac{2\sqrt{6}t}{5}\right) + \%k2 \cos\left(\frac{2\sqrt{6}t}{5}\right) \right)$ 

(%i7) ic2(%t=0,x=5,diff(x,t)=0);
(%o7)  $x = \%e^{-\frac{t}{5}} \left( \frac{5 \sin\left(\frac{2\sqrt{6}t}{5}\right)}{2\sqrt{6}} + 5 \cos\left(\frac{2\sqrt{6}t}{5}\right) \right)$ 
```

Загальний розв'язок неоднорідного рівняння з постійними коефіцієнтами записують як суму загального розв'язку відповідного однорідного рівняння і часткового розв'язку неоднорідного.

Розглянемо приклади:

Приклад 1. Вимушені коливання маятника під дією гармонійної вимушуючої сили

$$x'' + 4x' + x = \sin(t)$$

Частковим розв'язком цього рівняння є  $x(t) = -0.25 \cos(t)$

```
(%i12) dr1: 'diff (x, t, 2) +4*' diff (x, t)+x = sin(t);
(%o12)  $\frac{d^2}{dt^2}x + 4\left(\frac{d}{dt}x\right) + x = \sin(t)$ 

(%i13) ode2 (dr1, x, t);
(%o13)  $x = -\frac{\cos(t)}{4} + \%k1 e^{\frac{(2\sqrt{3}-4)t}{2}} + \%k2 e^{\frac{(-2\sqrt{3}-4)t}{2}}$ 
```



Приклад 2. Вимушені коливання маятника під дією вимушуючої сили з косинусною і синусною складовими

$$x'' + 0.4x' + x = a \cos(wt) + b \sin(wt)$$

Перша частина отриманого виразу є загальним розв'язком однорідного рівняння, друга — частковим розв'язком неоднорідного рівняння

```
(%i17) dr2: 'diff (x, t, 2) + 0.4* diff (x, t) + x = a*cos(w*t) + b* sin(w*t);
(%o17)  $\frac{d^2}{dt^2}x + 0.4 \left(\frac{d}{dt}x\right) + x = b \sin(t w) + a \cos(t w)$ 

(%i18) ode2(dr2,x,t);
rat: replaced 0.4 by 2/5 = 0.4
(%o18)  $x = e^{-\frac{t}{5}} \left( \%k1 \sin\left(\frac{2\sqrt{6}t}{5}\right) + \%k2 \cos\left(\frac{2\sqrt{6}t}{5}\right) \right) - \frac{(25bw^2 - 10aw - 25b) \sin(tw) + (25aw^2 + 10bw - 25a) \cos(tw)}{25w^4 - 46w^2 + 25}$ 
```

Для пошуку розв'язків лінійних диференціальних рівнянь високих порядків (вище другого) можна використовувати перетворення Лапласа [6].

Наступний приклад показує застосування перетворення Лапласа для розв'язання диференціального рівняння  $x'''(t) + x(t) = 0$ :

(C1) ode: 'diff(x(t), t, 3) + x(t) = 0;

(D1)  $\frac{d^3}{dt^3} x(t) + x(t) = 0$

(C2) atvalue(x(t), t=0, 1); // задаємо  $x(0) = 1$

(D2) 1

(C3) atvalue('diff(x(t), t), t=0, 3); // задаємо  $x'(0) = 3$

(D3) 3

(C4) atvalue('diff(x(t), t, 2), t=0, 8); // задаємо  $x''(0) = 8$

(D4) 8

(C5) lap\_ode: laplace(ode, t, s); // пряме перетворення Лапласа

(D5)  $s^3 \text{laplace}(x(t), t, s) + \text{laplace}(x(t), t, s) - s^2 - 3s - 8 = 0$

// розв'язуємо рівняння як алгебраїчне

(C6) sol: solve(%, 'laplace(x(t), t, s));

(D6)  $\text{laplace}(x(t), t, s) = \frac{s^2 + 3s + 8}{s^3 + 1}$

// обернене перетворення Лапласа

(C7) map(lambda([eq], ilt(eq, s, t)), sol);

(D7) 
$$[x(t) = e^{t/2} \left( \frac{11 \sin\left(\frac{\sqrt{3}t}{2}\right)}{\sqrt{3}} - \cos\left(\frac{\sqrt{3}t}{2}\right) \right) + 2e^{-t}]$$

(C8) plot2d([%e^(t/2)\*((11\*sin(sqrt(3)\*t)/2)/sqrt(3)-cos((sqrt(3)\*t)/2))+2\*%e^(-t)], [t, -5, 5])\$

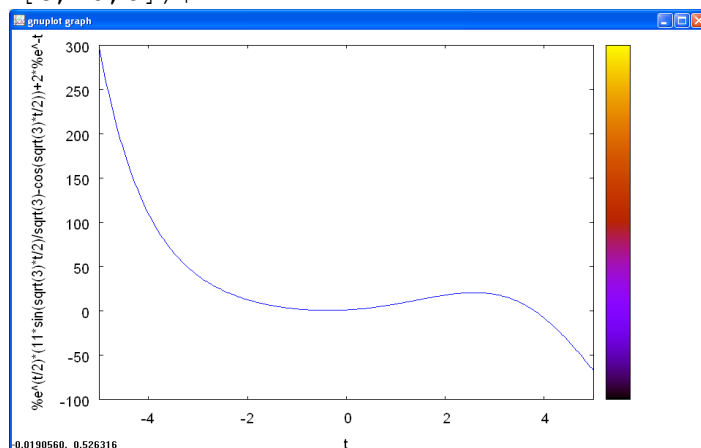


Рис. 2.8.

У більшості практичних задач диференціальні рівняння є нелінійними і аналітично вдається розв'язати лише окремі випадки. Одним із методів знаходження розв'язків нелінійних диференціальних рівнянь є використання методів числового інтегрування. Для цього в склад системи Maxima входить пакет dynamics, який потрібно завантажувати перед використанням. В пакеті dynamics є функція rk, яка реалізує метод Рунге-Кутти

$rk([eq], [vars], [init], [t\_range]),$

тут *eq* - список правих частин рівнянь, *vars* - список залежних змінних, *init* - список початкових умов, *t\_range* - список [t, t0, tend, ht], що містить символічне позначення незалежної змінної (t), її початкове значення (t0), кінцеве значення tend, крок інтегрування (ht).

Розглянемо приклад

$$x'' + 0.4 * x' + \sin(x) = 0, x(0)=5, x'(0)=0$$

Для використання функції rk приведемо диференціальне рівняння другого порядку до системи диференціальних рівнянь першого порядку

$$\begin{aligned} x' &= y; \\ y' &= -0.4 * y - \sin(x) \\ x(0) &= 1, y(0) = 0 \end{aligned}$$

За допомогою функції знайдемо розв'язок диференціального рівняння на проміжку [0; 2] з кроком 0.01

```
--> load("/usr/share/maxima/5.21.1/share/dynamics/dynamics.mac")$

(%i26) sol: rk([y, - 0.4 * y - sin(x)], [x, y], [1, 0], [t, 0, 2, 0.01]);
(%o26) [[0, 1, 0], [0.01, 0.99995798268217, -0.0083978272200781], [0.02, 0.9998321567134, -
0.016761677351751], [0.03, 0.99962286344484, -0.02509123481453], [0.04, 0.99933044736954, -
0.033386186785295], [0.05, 0.99895525609565, -0.041646223031947], [0.06, 0.99849764032141,
0.021010388838972, -0.65094283165375], [1.98, -0.027505658360074, -0.64810214416102], [
1.99, -0.033972254002606, -0.64520814117833], [2.0, -0.040409645178545, -0.64226133704423]
]

(%i19) dovg:length(sol);
(%o19) 201

(%i20) tg:makelist(sol[k][1], k, 1, dovg)$

(%i21) xg:makelist(sol[k][2], k, 1, dovg)$

(%i22) yg:makelist(sol[k][3], k, 1, dovg)$

(%i27) plot2d([[discrete, tg, xg], [discrete, tg, yg]]);
(%o27)

(%i29) plot2d([discrete, xg, yg]);
(%o29)
```

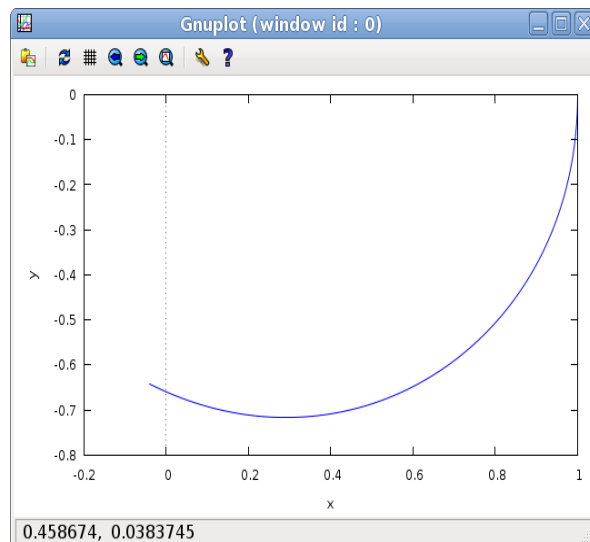
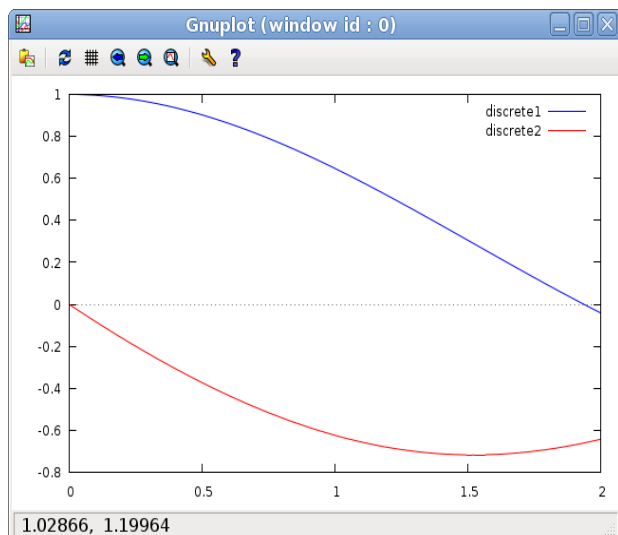


Рис. 2.9. Залежності  $x(t)$ ,  $y(t)$ ,  $y(x)$

## 2.12. Основи програмування в системі Maxima

Багато задач у системі Maxima розв'язують з використанням лінійних алгоритмів і програм. Вони можуть бути подані ланцюжком виразів, виконуваних послідовно від початку до кінця.

Однак у більшості випадків серйозні обчислення базуються на використанні циклічних і розгалужених алгоритмів і програм. При цьому залежно від проміжних чи вихідних даних, обчислення можуть йти по різних гілках програми, циклічно повторюватися тощо. Для реалізації розгалужених програм мова програмування повинна містити керуючі структури, тобто спеціальні конструкції мови, що реалізують у програмах розгалуження. Вони використовуються у різних методах програмування, у тому числі процедурному і функціональному програмуванні.

До найважливіших керуючих структур у мовах програмування відносять цикли. З їх допомогою здійснюється циклічне виконання деякого виразу *expr* задане число раз. Це число часто визначається значенням деякої керуючої змінної, яка змінюється або з кроком  $+1$ , або від початкового значення *imin* до кінцевого *imax* із кроком *di*. Цикли можуть вкладені один в один.

Розглянемо приклад циклу, у якому змінна *a* виводиться на екран, приймає початкове значення  $-3$ , змінюється з кроком  $-7$  і виконується доти, поки змінна менше 26. Тут функція `ldisplay` виводить на екран не тільки значення змінної, але й саму змінну, а також надає значення цієї змінної мітці (E1, E2 тощо)

```
(C1) for a:-3 thru 26 step 7 do ldisplay(a)$
(E1)      a = -3
...
(E5)      a = 25
```

Тепер, якщо потрібно, наприклад, звернутися до значення змінної *a*, отриманої на 3 кроці, потрібно набрати E3.

```
(C5) E3;
(D5) A=11
```

Використовуючи функцію `display`, ми одержимо наступний результат (виведемо на екран значення змінної та її ім'я):

```
(C6) for a:-3 thru 26 step 7 do display(a)$
      a = -3
...
```

a = 25

Функція print видасть на екран тільки значення змінних:

```
(C7) for a:-3 thru 26 step 7 do print(a)$  
- 3  
...  
25
```

Змінний step можна надавати як цілі, так і дробові значення. Наприклад:

```
(C8) for a:-3 thru 0 step 0.25 do print(a)$  
- 3  
...  
- 0.25  
0.0
```

Наступний цикл обчислює значення змінної S+I і працює до тих пір, поки змінна I буде менше чи рівною 10.

```
(C9) S:0$  
(C10) for I:1 while I<=10 do S:S+I;  
(D11) done  
(C12) S;  
(D12) 55
```

Наступний приклад показує приклад обчислення S доти, поки I строго менше 10:

```
(C13) for I:1 unless I>10 do S:S+I;  
(D13) done  
(C14) S;  
(D14) 55
```

Наступний приклад показує використання вкладених циклів:

```
(C15) POLY:0$  
(C16) for I:1 thru 5 do  
      for J:I step -1 thru 1 do  
        POLY:POLY+I*X^J$  
(C17) POLY;  
(D17)  $5x^5 + 9x^4 + 12x^3 + 14x^2 + 15x$ 
```

Цикли дають змогу обчислити значення функції на заданому відрізку з кроком, як цілим, так і дробовим:

```
(C18) for y:-1 thru 1 step 0.5 do( t:y*y, print("t=",t));  
t= 1  
...  
t= 0.25  
t= 1.0  
(D18) done
```

Як і у більшості мов програмування, умовні вирази задають за допомогою оператора if. У системі Maxima цей оператор має вигляд:

if (умова) then вираз1 else вираз2.

Наприклад:

```
(C19) x:0;  
(D19) 0  
(C20) if(x=0) then x:x+1 else x:x-1;  
(D20) 1
```

Умовний оператор можна використовувати усередині циклів, наприклад:

```
(C21) GUESS:-3.0$  
(C22) for I:1 thru 10 do (GUESS:SUBST(GUESS,X,.5*(X+10/X)),  
      if abs(GUESS^2-10)<.00005 then return(GUESS));
```

(D22) - 3.1622807

Цикли можна використовувати для виводу на екран таблиці значень функції  $f$  на інтервалі від  $x_{\text{поч}}$  до  $x_{\text{кін}}$  із кроком  $d$ .

Напишемо програму обчислення 20 значень сигналу, заданого на рис. 2.9, для  $A=7$ ,  $B=5$ ,  $a=0.4$ ,  $b=0.5$ ,  $T=1$ ;

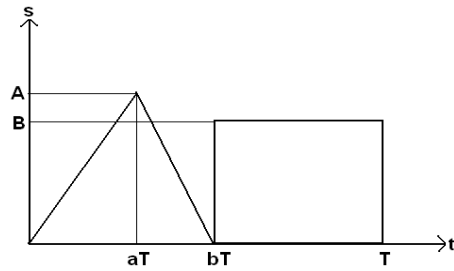


Рис. 2.9.

Для обчислення значень сигналу потрібно задати таку послідовність команд:

```
(C1) T:1;
(D1)      1;
(C2) a:0.4;
(D2)      0.4;
(C3) b:0.5;
(D3)      0.5;
(C4) A:7;
(D4)      7;
(C5) B:5;
(D4)      5;
(C27) for t:0 while t<T step 0.05 do
(if (t<a*T) then (s:A*t/(aT),print(s)) else
if (t>a*T and t<b*T) then (s:A*(t-a*T)/((b-a)*T)),print(3)) else (f:B,
print(s)));
```

Функція  $\text{SUBST}(a, b, c)$  – вираз  $b$  позначаємо  $a$  і підставляємо в  $c$ . Наприклад,

(C23)  $\text{SUBST}(A, X+Y, X+(X+Y)**2+Y)$  ;

(D23)  $Y + X + A^2$

Функція  $\text{return}(\text{expr})$  перериває виконання з передачею значення виразу  $\text{expr}$ .

Досі ми використовували систему Maxima в інтерактивному режимі подібно до калькулятора. Якщо часто доводиться виконувати визначену послідовність обчислень, то краще оформити її як програму, що викликається за потреби. Нижче приводиться невелика програма для знаходження критичних точок функції  $f(x)$ . Користувачу пропонується увести функцію  $f$ , після чого обчислюється похідна уведеної функції і за допомогою функції  $\text{solve}$  розв'язується рівняння  $f'(x) = 0$ . Програма записується в текстовий файл і потім завантажується в систему Maxima за допомогою функції  $\text{batch}$ . Наведемо текст програми:

```
critpts := (
  print("Програма знаходження критичних точок"),
  /* Запит на введення функції */
  print("Уведіть функцію f(x):"),
  f:read(),
  /* Друкування уведеної функції (для контролю) */
  print("f = ", f),
  /* У змінну eq заносимо значення похідної */
```

```

eq:diff(f,x),
/* Розв'язуємо рівняння */
solve(eq, x)
)$

```

Програма складається з єдиної функції (без аргументів), що називається critpts. Команди відокремлюються комами. Приклад виконання програми:

```

(C1) batch("c:\\work\\critpoints.mac");
batching #p C/work/critpoints.max
(C2) critpts() := (PRINT("Програма #
знаходження критичних точок"),
PRINT("Уведіть функцію f(x):"),
f : READ(),
PRINT("f = ", f),
eq : DIFF(f, x),
SOLVE(eq, x))
(C3) critpts() ;
Програма знаходження критичних точок
Уведіть функцію f(x):
(x+2)/(x^2+1);

$$f = \frac{x+2}{x^2+1}$$

(D3) [x = - sqrt(5) - 2, x = sqrt(5) - 2]

```

№	Функція	Призначення
1	ldisplay	виводить на екран не тільки значення змінної, а й саму змінну, а також надає значення цієї змінної мітці (E1, E2 і т.д.)
2	display	виводить на екран значення змінної та її ім'я
3	for	цикл з відомою кількістю повторень
4	while	цикл з невідомою кількістю повторень
5	step	крок
6	print	виведення на екран
7	if	умовний оператор
8	subst(a,b,c)	вираз b позначаємо як a і підставляємо в c
9	return(expr)	перериває виконання з виводом значення виразу expr
10	batch	завантаження програмного файлу

### 2.13. Транслятор і компілятор в Maxima

Застосування трансляції або компіляції функцій користувача дає змогу прискорити процес виконання цих функцій. Функція **translate** трансліює функцію Maxima'и у програму мовою LISP. Після цього функція зазвичай починає виконуватися швидше. Функція **compile** спочатку трансліює функцію Maxima'и у програму мовою LISP, а потім компілює цю функцію LISP'а у двійкові коди і завантажує їх в пам'ять.

Рекомендована література

1. Семеріков С.О. Maxima 5.13: довідник користувача / За ред. академіка АПН України М.І. Жалдака. – Київ, 2007. – 48 с.
2. Стахин Н.А. Основы работы с системой аналитических (символьных) вычислений Maxima. (ПО для решения задач аналитических (символьных) вычислений): Учебное пособие. – Москва: 2008. — 86 с.
3. В.А. Ильина, П.К. Силаев. Система аналитических вычислений Maxima для физиков-теоретиков. М.:МГУ им. . М.В. Ломоносова, 2007. - 113 с., <http://tex.bog.msu.ru/numtask/max07.ps>
4. Компьютерная математика с Maxima: Руководство для школьников и студентов /Е.А. Чичкарёв -М.:ALTLinux, 2009. - 233с.:ил. - (Библиотека ALTLinux).

Додаткова література

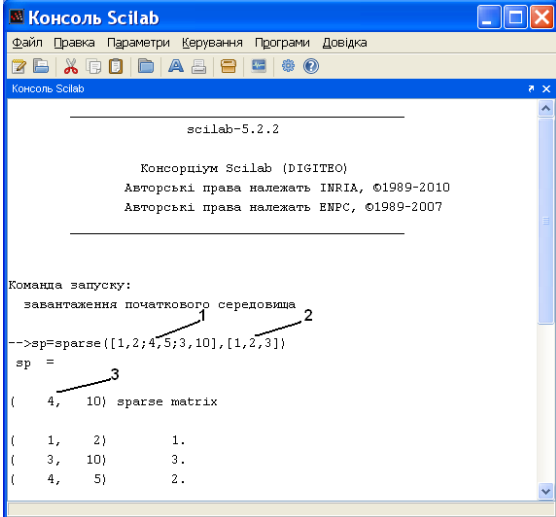
5. Д.М. Климов, В.М. Руденко Методы компьютерной алгебры в задачах механики. -М.: Наука, 1989. -215 с. Теоретическая механика. Вывод и анализ уравнений движения на ЭВМ: Учебн. Пособие для вузов. В 2 ч. Ч.1 / В.Г. Веретенников, И.И. Карпов, А.П. Маркеев и др.; Под ред. В.Г. Веретенникова -М.: Высш. шк., 1990. -174 с.
6. Компьютерная алгебра : Символьные и алгебраические вычисления: Пер. с англ./ Под ред. Б. Бухбергера, Дж. Коллинза, Р. Лооса. -М.: Мир, 1986. -392 с.
7. Г.Б. Ефимов, М.В. Грошева Об истории использования отечественных систем символьных преобразований в механических приложениях. Математичні машини і системи, 2008, № 1
8. Попов Б.О., Монцібович Б.Р. Розв'язування задач на машинах для інженерних розрахунків. К.: Наук. Думка, 1978. - 346 с.

## Додатки

## Додаток 1. Функції Scilab для роботи з матрицями

Функція	Призначення
<b>matrix(A [,n,m])</b> Приклад: <pre>--&gt;a=[1 2 3;4 5 6] a =   1.  2.  3.   4.  5.  6. --&gt;matrix(a,1,6) ans =   1.  4.  2.  5.  3.  6. --&gt;matrix(a,3,2) ans =   1.  5.   4.  3.   2.  6.</pre>	перетворює матрицю A в матрицю іншого розміру
<b>ones(m,n)</b>	створює матрицю одиниць з m рядків і n стовпців
<b>zeros(m,n)</b>	створює матрицю нулів з m рядків і n стовпців
<b>eye(m,n)</b> Приклад: <pre>--&gt;eye(3,3) ans =   1.  0.  0.   0.  1.  0.   0.  0.  1. --&gt;eye(2,3) ans =   1.  0.  0.   0.  1.  0. --&gt;m=3; n=4; --&gt;eye(m,n) ans =   1.  0.  0.  0.   0.  1.  0.  0.   0.  0.  1.  0.</pre>	формує одиничну матрицю з m рядків і n стовпців
<b>rand (n1, n2, ... nn [, p])</b> Приклад: <pre>--&gt;rand(2,2) ans =   0.2113249  0.0002211   0.7560439  0.3303271 --&gt;R=rand(2,2,2) R = (:,,1)   0.6653811  0.8497452   0.6283918  0.6857310 (:,,2)   0.8782165  0.5608486   0.0683740  0.6623569 --&gt;rand() ans =   0.7263507</pre>	формує багатовимірну матрицю випадкових чисел. Необов'язковий параметр p - це символічна змінна, за допомогою якої можна задати тип розподілу випадкової величини ('uniform' - рівномірний, 'normal' - гаусівський); rand (m, n) - формує матрицю m на n випадкових чисел; rand(M) - формує матрицю випадкових чисел, розмір якої співпадає з розміром матриці M; результат функції rand () - випадкове число
<b>sparse ([i1 j1; i2 j2;...; in jn], [n1, n2,..., nn ])</b> Приклад:	формує розріджену матрицю. Для створення матриці такого типу необхідно вказати індекси її ненульових елементів - [i1 j1, i2 j2,..., in jn], і їх



 <p>1 — індекси ненульових елементів, 2 — значення ненульових елементів, 3 — розмірність матриці</p>	<p>значення - <math>[n_1, n_2, \dots, n_n]</math>. Індеси одного елемента відокремлюються один від одного або пробілом, або комою, а пари індесів - відповідно крапкою з комою, значення елементів розділяються комами. При спробі переглянути матрицю подібного типу користувачеві буде надано повідомлення про її розмірності, а також значення ненульових елементів і їх місце розташування в матриці;</p>
<p><b>full (M)</b>  Приклад:  --&gt;sp=sparse([1,2;4,5;3,10],[1,2,3]);  --&gt;A=full(sp)  A =  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.</p>	<p>вивід розрідженої матриці M таблицею</p>
<p><b>hypermat (D [, V])</b>  Приклад:  --&gt;M=hypermat([2 3 2 2],1:24)  M =  (:,1,1)  1. 3. 5.  2. 4. 6.  (:,2,1)  7. 9. 11.  8. 10. 12.  (:,1,2)  13. 15. 17.  14. 16. 18.  (:,2,2)  19. 21. 23.  20. 22. 24.</p>	<p>створення багатовимірної матриці з розмірністю, заданою вектором D і значеннями елементів, що зберігаються у векторі V (використання параметра V необов'язкове)</p>
<p><b>diag (V [, k])</b></p>	<p>формує квадратну матрицю з елементами V на головній або на k-тій діагоналі; функція diag (A [, k]), формує вектор-стовпець, який містить елементи головної або k-тої діагоналі матриці A (k=0 – головна діагональ, k&gt;0 – діагональ над головною, k&lt;0 – діагональ під головною).</p>
<p><b>cat (n, A, B, [C, ...])</b></p>	<p>об'єднує матриці A і B або всі вхідні матриці (n = 1 – по рядках, n = 2 – по стовпцях).</p>
<p><b>tril (A [, k])</b></p>	<p>формує з матриці A нижню трикутну матрицю, починаючи з головної або з k-тої діагоналі.</p>
<p><b>triu (A [, k])</b></p>	<p>формує з матриці A верхню трикутну матрицю,</p>

	починаючи з головної або з k-ї діагоналі;
sort (X)	виконує упорядкування масиву X; якщо X - матриця, то сортування виконується по стовпцях
size (V [, fl])	визначає 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;
length (X)	визначає кількість елементів масиву X; якщо X — вектор, то його довжину; якщо X — матриця, то обчислює загальне число її елементів;
sum (X [, fl]) Приклад: -->M=[1 2 3;4 5 6;7 8 9]; -->Y=sum(M) //Сума елементів матриці Y = 45. -->S1=sum(M,1) //Сума елементів матриці по стовпцях S1 = 12 15 18 -->S2=sum(M,2) // Сума елементів матриці по рядках S2 = 6 15 24 --> V=[-1 0 3 -2 1 -1 1]; --> sum(V) //Сума елементів вектора ans = 1	обчислює суму елементів масиву X, має необов'язковий параметр fl. Якщо параметр fl відсутній, то функція sum (X) обчислює суму елементів масиву. Якщо fl = 'r' або fl = 1, то функція формує рядок з сумами елементів стовпців матриці X. Якщо fl = 'z' або fl = 2, то результатом роботи функції буде вектор-стовпець з сумами елементів рядків матриці X.
prod (X [, fl]) Приклад: -->M=[1 2 3;4 5 6;7 8 9]; -->prod(M) ans = 362880. -->p1=prod(M,1) p1 = 28 80 162 -->p2=prod(M,2) p2 = 6 120 504 --> V=[1,2,3]; --> prod(V) //Добуток елементів вектора ans = 6	обчислює добуток елементів масиву X, працює аналогічно функції sum;
max (M [, fl]) Приклад: -->M=[5 0 3;2 7 1;0 4 9]; -->max(M) ans = 9. -->max(M,'r') ans = 5. 7. 9. -->max(M,'c') ans = 5.	обчислює найбільший елемент в масиві M, має необов'язковий параметр fl. Якщо параметр fl відсутній, то функція max (M) визначає максимальний елемент масиву M; якщо fl = 'r', то функція формує рядок максимальних елементів стовпців матриці M; якщо fl = 'z', то результатом роботи функції буде вектор-стовпець, кожен елемент якого дорівнює максимальному елементу відповідних рядків матриці M. Функція [x, nom] = max (M [, fl]) поверне значення

7. 9. -->[x,nom]=max(M) nom = 3. 3. x = 9.	максимального елемента x і його індекси у масиві.
min (M [, fl]) Приклад: -->A=[5 10 3 2 7 1 25 4 0]; -->[x,nom]=min(A) nom = 7. x = 25.	обчислює найменший елемент у масиві M, працює аналогічно функції max.
mean (M [, fl]) Приклад: -->M=[5 0 3;2 7 1;0 4 9]; -->mean(M) ans = 3.4444444 -->mean(M,1) ans = 2.3333333 -->mean(M,2) ans = 2.6666667 3.3333333 4.3333333	обчислює середнє значення масиву M; якщо M двовимірний масив, то mean (M, 1) або mean (M, 'r') обчислює середнє значення рядків матриці M, а mean (M, 2) або mean (M, 'c') - середнє значення стовпців.
median (M [, fl]) Приклад: -->M=[5 0 3;2 7 1;0 4 9]; -->median(M) ans = 3. -->median(M,1) ans = 2. 4. -->median(M,2) ans = 3. 2. 4.	обчислює медіану (значення, яке ділить масив на дві частини) масиву M, працює аналогічно до функції mean.
det (M)	обчислює визначник квадратної матриці M.
rank (M [, tol])	обчислення рангу матриці M з точністю tol. (ранг матриці - максимальне число лінійно незалежних рядків)
norm (M [, fl])	обчислення норми квадратної матриці M; тип норми визначається необов'язковою змінною fl, за замовчуванням fl = 2. Функції norm (M) і norm (M, 2) еквівалентні і обчислюють другу норму матриці M2. Перша норма визначається функцією norm (M, 1). Функції norm (M, 'inf') і norm (M, 'fro') обчислюють відповідно нескінченну та евклідову 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 <code>norm (V, 'inf')</code> дорівнює модулю максимального елемента вектора по модулю. (Друга норма матриці - її найбільша сингулярне значення. Перша норма матриці - найбільша сума за стовпцями. Нескінченна норма - найбільша сума за рядками. Евклідова норма - корінь з суми квадратів всіх елементів матриці. Модуль вектора - корінь квадратний із суми квадратів його елементів.)
<code>cond (M)</code>	обчислює число обумовленості матриці M за другою нормою. (Число обумовленості дорівнює добутку норми вихідної матриці на норму оберненої.)
<code>spec (M)</code>	обчислює власні значення і власні вектори квадратної матриці M.
<code>inv(A)</code>	обчислює обернену матрицю
<code>linsolve (A, b)</code> Приклад: <code>--&gt;A=[1 2;1 1];b=[-7;-6];</code> <code>--&gt;x=linsolve(A,b)</code> x = 5. 1.	розв'язує систему лінійних алгебраїчних рівнянь $A \cdot x - b = 0$
<code>rref (A)</code>	приводить матрицю A до трикутної форми за методом виключення Гауса;
<code>lu (A)</code> Приклад: <code>--&gt;A=[2 -1 5;3 2 -5;1 1 -2]</code> A = 2. - 1. 5. 3. 2. - 5. 1. 1. - 2. <code>--&gt;[L,U]=lu(A)</code> U = 3. 2. - 5. 0. - 2.3333333 8.3333333 ! 0. 0. 0.8571429 ! L = 0.6666667 1. 0. 1. 0. 0. 0.3333333 - 0.1428571 1. <code>--&gt;LU=L*U</code> LU = 2. - 1. 5. 3. 2. - 5. 1. 1. - 2.	виконує LU-розклад матриці A

## Додаток 2. Перелік основних функцій системи Maxima

Функція або змінна	Призначення
<code>addcol</code>	Функція додає стовпець до матриці
<code>addrow</code>	Функція додає рядок до матриці

algsys	Функція знаходить розв’язок поліноміальної системи рівнянь (включно з одним рівнянням відносно одного невідомого)
allroots	Функція знаходить всі корені поліноміального рівняння
antidiff	Функція виконує інтегрування виразів з довільними функціями (перед пергим використанням потрібно завантажити функцію)
append	Функція склеює два списки
arrayinfo	Функція друкує інформацію про масив — його вид, кількість індексів, розмір
arrays	Змінна містить список імен уже визначених масивів
array	Функція визначає масив із заданим ім'ям, певною кількістю індексів і заданим розміром
assume	Функція виводить інформацію про змінну в базу даних
atom	Функція повертає “true” якщо аргумент не має структури, тобто складових частин
atvalue	Функція дає змогу задати значення функції та її похідних для деяких значень аргументів
augmented_lagrangian_method	Функція здійснює мінімізацію ФНП з обмеженнями
batch	Запуск на виконання файлу із сценарієм. Оператори виконуються до кінця сценарію (або до синтаксичної помилки чи некоректної операції)
bc2	Функція дає змогу врахувати крайові умови в розв’язках диференціальних рівнянь другого порядку
cabs	Функція обчислює модуль комплексного числа
carg	Функція обчислює аргумент (фазу) комплексного числа
cfdisrep	Функція перетворює список (зазвичай результат виконання функції cf) у ланцюговий дріб
cf	Функція створює ланцюговий дріб, апроксимуючий заданий вираз. Вираз повинен складатися з цілих чисел, квадратних коренів цілих чисел і знаків арифметичних операцій.
cfdirep	Функція перетворює список у ланцюговий дріб
changevar	Виконує заміну змінних в інтегралі
changevar	Функція обчислює характеристичний поліном матриці
closefile	Закриття файлу
COI	Функція виділяє заданий стовпець матриці
combine	Функція об’єднує складові з однаковими знаменниками
compile	Функція спочатку трансліює функцію Maxima на мову LISP, а потім компілює цю функцію LISP'a до двійкових кодів і завантажує їх в пам'ять
conjugate	Обчислює комплексно-спряжений вираз
cons	Додає елемент в початок списку

contrib_ode	Знаходить розв'язок дифрівняння
copylist	Створює копію списку
create_list	Створює список
copymatrix	Створює копію матриці
cspline	Виконує сплайн-інтерполяцію
define	Перетворює вираз у функцію
demoivre	Функція замінює всі експоненти з уявними показниками на відповідні тригонометричні функції
denom	Виділяє знаменник дробу
depends	Функція декларує залежність змінної від однієї або декількох інших змінних
desolve	Розв'язування дифрівнянь із застосуванням перетворення Лапласа
determinant	Функція обчислює детермінант матриці
diff	Обчислення похідної
display2d	Змінна вмикає або вимикає "двовимірне" відображення дробів, степенів тощо. Спочатку встановлено значення "true"
display	Функція виводить значення своїх аргументів разом з їх іменем, кожне в окремому рядку
displ	Функція виводить значення своїх аргументів, кожне в окремому рядку
divide	Обчислює частку і залишок від ділення двох многочленів
draw2d	Будує двовимірні графіки
draw3d	Будує тривимірні графіки
echelon	Перетворює матрицю до верхньої трикутної
eigenvalues	Функція аналітично обчислює власні значення матриці
eigenvectors	Функція аналітично обчислює власні значення і власні вектори матриці
eliminate	Функція виключає із системи рівнянь зазначені змінні.
endcons	Додає елемент в кінець списку
ev	Основна функція опрацювання виразів
expand	Розкриває дужки
exponentialize	приводить комплексний вираз до експоненціальної форми
express	Перетворює диференційні оператори у вирази
factor	Перетворює вираз у добуток співмножників
factorsum	Функція факторизує окремі доданки у виразі
fillarray	Функція заповнює одноіндексні масиви третього виду елементами списку
find_root	Знаходить корінь рівняння методом поділу відрізка навпіл
first	Виділяє перший елемент списку

float	Перетворює числові значення до машинного вигляду
fourier	Обчислює коефіцієнти ряду Фур'є
foursimp	Спрощує вирази для коефіцієнтів ряду Фур'є
fullratsimp	Функція викликає функцію "ratsimp" доти, поки вираз не перестане змінюватися
genmatrix	Функція повертає матрицю заданої розмірності, складену з елементів індексного масиву
gfactorsum	Перетворює комплексний вираз у добуток співмножників
gfactor	Перетворює комплексний вираз у добуток співмножників
gradef	Обчислює похідну функції від її аргументів
gramschmidt	Обчислює ортонормовану систему векторів
ic1	Функція враховує початкові умови у розв'язку дифференціального першого порядку
ic2	Функція враховує початкові умови у розв'язку дифференціального першого порядку
ident	Генерує одиничну матрицю заданої розмірності
ilt	Обчислює зворотнє перетворення Лапласа
imagpart	Виділяє уявну частину комплексного виразу
integrate	Функція виконує інтегрування заданого виразу по вказаній змінній (невизначена константа не додається). Можна також вказати межі інтегрування - в цьому випадку обчислюється визначений інтеграл
invert	Обчислює обернену матрицю
join	Компонує списки
kill	Видаляє інформацію про об'єкт/ об'єкти
lagrange	Виконує інтерполяцію за Лагранжем
lambda	Створює лямбда-вираз
laplace	Виконує пряме перетворення Лапласа
last	Виділяє останній елемент списку
lbfgs	Мінімізує ФНП
ldisplay	Функція виводить значення своїх аргументів разом з їх іменами і позначками
ldisp	Функція виводить значення своїх аргументів разом з їх позначками
length	Виводить довжину списку
lhs	Виділяє ліву частину рівняння
limit	Обчислює границі
linearinterpol	Лінійна інтерполяція
linsolve	Розв'язує системи лінійних і поліноміальних рівнянь
listarray	Виводить вміст масивів першого і другого видів
load	Завантажує файл (макрос Maxima, програму на Lisp або бінарний

	файл)
logcontract	Приводить логарифми у виразі до компактного вигляду
make_array	Створює масиви третього виду
makelist	Створює списки
map	Функція застосовує задану функцію до кожного елементу списку
matrix	Функція генерує матрицю, задану поелементно
matrixmap	Заповнює матрицю значеннями заданої функції
mattrace	Обчислює слід матриці
max	Визначає максимальне значення
min	Визначає мінімальне значення
minor	Обчислює мінори матриці
mnewton	Функція знаходить корені системи рівнянь методом Ньютона. Для використання функції необхідно спочатку завантажити пакет "mnewton"
multthru	Домножує складові виразу на заданий множник (дужки не розкриваються)
newton	Обчислює корінь рівняння методом Ньютона
nroots	Виводить кількість дійсних коренів поліноміального рівняння
num	Виділяє чисельник дробу
ode2	Розв'язує дифференціальні рівняння першого і другого порядків
odelin	Знаходить фундаментальний розв'язок лінійного дифференціального рівняння першого і другого порядків
pade	Функція апроксимує відрізок ряду Тейлора дрібно-раціональною функцією
part	Виділяє заданий елемент списку
plog	Обчислює основне значення комплексного логарифму
plot2d, wxplot2d	Будує двовимірні графіки
plot3d, wxplot3d	Будує тривимірні графіки
polarform	Приводить комплексне значення до тригонометричного вигляду
polyfactor	Змінна визначає форму виводу функції "allroots"
powerseries	Будує розклад у ряд за степенями
print	Виводить значення аргументів в один рядок
product	Виконує перемноження
properties	Виводить властивості змінної
radcan	спрощує вирази з вкладеними степенями та логарифмами
ratepsilon	Змінна задає точність перетворення дійсного числа в раціональне
ratexpand	Розкриває дужки у виразі і приводить його до канонічного вигляду
ratfac	Змінна вмикає або вимикає часткову факторизацію виразів при зведенні їх до стандартної форми. Спочатку встановлено значення



	"false"
ratsimpexpons	Змінна керує спрощенням показників степеню у виразах
ratsimp	Функція приводить всі частини (включно з аргументами функцій) виразу, яке не є дрібно-раціональною функцією, до канонічного вигляду, виконуючи спрощення, які не робить функція "rat"
ratsubst	Функція виконує синтаксичну підстановку для раціональних виразів
ratvars	Функція дає змогу змінити алфавітний порядок "чергування" змінних, прийнятий за замовчуванням
rat	Функція приводить вираз до канонічного виду і позначає його позначкою $\frac{\text{num}}{\text{den}}$ . Вона спрощує будь-який вираз, розглядаючи його як дробно-раціональну функцію, тобто працює з арифметичними операціями і піднесенням до цілого степеня
realpart	Виділяє дійсну частину комплексного виразу
read	Ввід даних
read_matrix, read_list	Ввід масиву даних
realroots	Функція знаходить дійсні корені поліноміального рівняння з дійсними коефіцієнтами
rectform	Функція перетворює комплексний вираз в алгебраїчну форму
remarray	Видаляє масив
remove	Видаляє властивості змінної
residue	Функція визначає нулі комплексного виразу на комплексній площині
rest	Видаляє залишок після видалення першого елемента списку
reverse	Функція змінює порядок елементів у списку на зворотній
rhs	Виділяє праву частину рівняння
romberg	Функція обчислює визначений інтеграл за алгоритмом Ромберга
rk	Функція реалізує метод Рунге-Кутта чисельного інтегрування дифрівнянь
row	Виділяє заданий рядок матриці
save	Зберігає робочу область у файл
solve	Знаходить розв'язки системи рівнянь
sort	Впорядковує елементи списку
sublist	Функція складає список з тих елементів заданого списку, для яких вказана логічна функція повертає значення "True"
submatrix	Виділяє з матриці підматрицю
subst	Виконує синтаксичну підстановку
sum	Обчислює задану суму
taylor	Виконує розклад функції в ряд Тейлора
tlimit	Знаходить границю з використанням ряду Тейлора

totalfourier	Будує ряд Фур'є
translate	Транслює функцію Maxima на мову LISP
transpose	Транспонує матрицю
trigexpand	Функція розкладає всі тригонометричні функції від сум в суми добутоків тригонометричних функцій
trigreduce	Функція перетворює всі добутки тригонометричних функцій в тригонометричні функції від сум
trigsimp	Застосовує до виразу правила $\sin^2(x)$ $\cos^2(x)$
trirat	Функція намагається спростити вираз з тригонометричними функціями
uniteigenvectors	Знаходить нормовані до одиниці власні вектори матриці
writefile	Записує дані у файл
write_matrix, write_list	Виводить масив чисел
xthru	Функція приводить вираз до спільного знаменника, не розкриваючи дужок і не факторизуючи доданки
zeromatrix	Генерує нульову матрицю
“	Дві одиночні лапки "а викликають додаткове обчислення в момент обробки а
‘	Одиночна лапка ' а блокує додаткове обчислення в момент обробки а

### Додаток 3. Перелік основних пакетів розширення Maxima

Пакет	Призначення
augmented_lagrangian	Мінімізація функції декількох змінних з обмеженнями методом невизначених множників Лагранжа
bode	Побудова діаграм Бode
contrib_ode	Додаткові функції для аналітичного розв'язання звичайних диференціальних рівнянь
descriptive	оцінка параметрів розподілу густини ймовірності за вибіркою
diag	Пакет для операцій з деякими видами діагональних матриць
distrib	Пакет для розрахунку різних розподілів ймовірності і їх параметрів (нормальний розподіл, розподіл Стюдента тощо)
draw	Інтерфейс Maxima-Gnuplot, який призначений для підготовки ілюстрацій поліграфічної якості
Dynamics	Різні функції, що відносяться до моделювання динамічних систем та фракталів
f90	Експорт коду Maxima в код на Фортран-90
graphs	Пакет для роботи з графами
grobner	Функції для роботи з базисом Гребнера (Groebner)
Impdiff	Обчислення похідних від функцій кількох змінних

implicit_plot	Побудова графіків неявно заданих функцій
interpol	Пакет з функціями інтерполяції
lapack	Функції пакета LAPACK для розв'язання задач лінійної алгебри
Lbfgs	Пакет мінімізації функцій кількох змінних квазіньютонівським методом (L-BFGS)
lindstedt	Пакет, розрахований на інтерпретацію деяких типів початкових умов для диференціальних рівнянь коливань систем
lsquares	Функції для оцінки параметрів різних залежностей методом найменших квадратів
makeOrders	Функція для операцій з поліномами
mnewton	Метод Ньютона для розв'язування систем нелінійних рівнянь
numericalio	Читання і запис файлів (переважно з матричними числовими даними)
opsubst	Пакет з функцією opsubst, що виконує заміну у виразах (за можливостями мало відрізняється від subst)
orthopoly	Пакет, що містить функції для операцій з ортогональними поліномами (Лежандра, Чебишева і ін)
plotdf	Пакет для побудови поля напрямків для автономних коливань систем з одним ступенем вільності (цікавий, але досить вузькоспеціалізований пакет)
romberg	Пакет з функціями для числового інтегрування
simplex	Пакет, призначений для розв'язання задач лінійного програмування
solve_rec	Пакет, що містить функції для спрощення рекурентних виразів
stats	Пакет статистичних функцій
stirling	Обчислення гамма-функції
stringproc	Пакет з функціями для обробки рядків символів
unit	Пакет з функціями для операцій з одиницями вимірювань
zeilberger	Функції для гіпергеометричного сумування

#### Додаток 4. Список основних математичних констант, доступних в Maxima

Позначення	Константа
%e	Натуральна одиниця
%i	Уявна одиниця
inf	$+\infty$
minf	$-\infty$
infinite	Нескінченність (на комплексній площині)
% phi	Золотий переріз ( $\phi$ )
% pi	Стала $\pi$
%gamma	Стала Ейлера

false, true	Логічні величини
-------------	------------------

Додаток 5. Список основних математичних функцій, доступних в Maxima

Позначення	Елементарна функція	Позначення	Елементарна функція
sqrt	Квадратний корінь	exp	Експонента
sin	Синус	log	Натуральний логарифм
cos	Косинус	sinh	Гіперболічний синус
tan	Тангенс	cosh	Гіперболічний косинус
cot	Котангенс	tanh	Гіперболічний тангенс
sec	Секанс	asinh	Гіперболічний арксинус
csc	Косеканс	acosh	Гіперболічний арккосинус
asin	Арксинус	atanh	Гіперболічний арктангенс
acos	Арккосинус	floor	Округлення до цілого з нестачею
atan	Арктангенс	ceiling	Округлення до цілого з надлишком
acot	Арккотангенс	fix	Ціла частина
asec	Арксеканс	float	Перетворення до формату з плаваючою крапкою
acsc	Арккосеканс	abs	Модуль