5. Обрамлення Maxima

5.1 Класичні графічні інтерфейси Maxima

5.1.1 Графічний інтерфейс wxMaxima

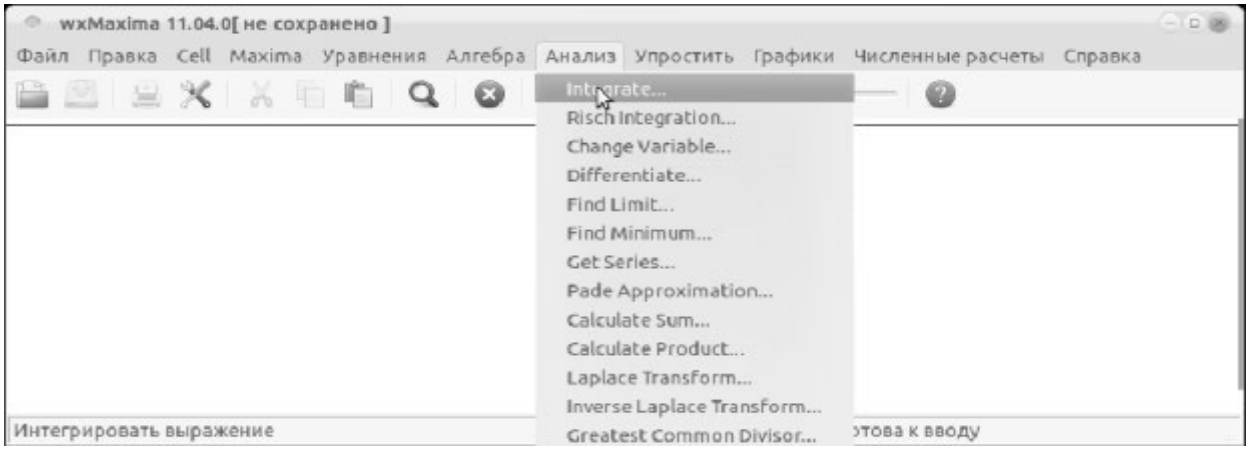
Для зручності роботи відразу звернемося до графічного інтерфейсу wxMaxima, тому що він є найбільш дружнім для користувачів-початківців системи.

Достоїнствами wxMaxima є:

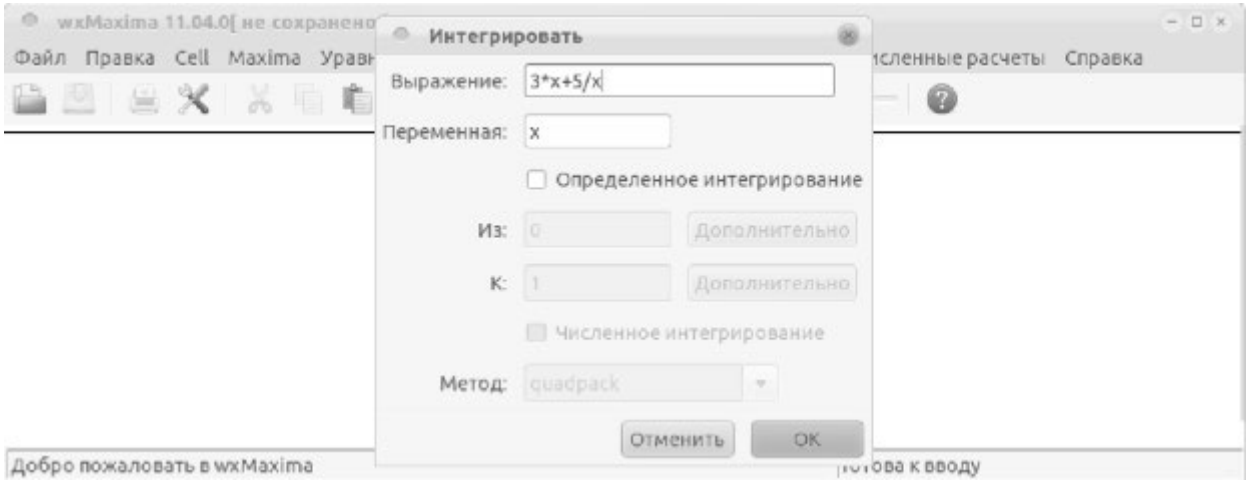
- можливість графічного виведення формул;
- спрощене введення функцій, що найбільш часто використовуються (через діалогові вікна);
- можливість включення графічних ілюстрацій безпосередньо до тексту робочої книги (при використанні формату wxMaxima)

5.1.1.1 Робоче вікно wxMaxima

Розглянемо робоче вікно програми (див. рис. 5.1 та рис. 5.2). Зверху вниз розташовуються: текстове меню програми – доступ до основних функцій та налаштувань програми. У текстовому меню wxMaxima знаходяться функції для вирішення великої кількості типових математичних завдань, розділені за групами: рівняння, алгебра, аналіз, спростити, графіки, чисельні обчислення. Введення команд через діалогові вікна полегшує роботу з програмою для новачків.



Мал. 5.1.Інтерфейс wxMaxima, вибір команди інтегрування.



Мал. 5.2.Інтерфейс wxMaxima, обчислення інтегралу.

Наприклад, пункт меню Аналіз/Інтегрувати дозволяє обчислити певний чи невизначений інтеграл. Після введення необхідних параметрів, у робочому вікні ми побачимо команду та результат обчислення:

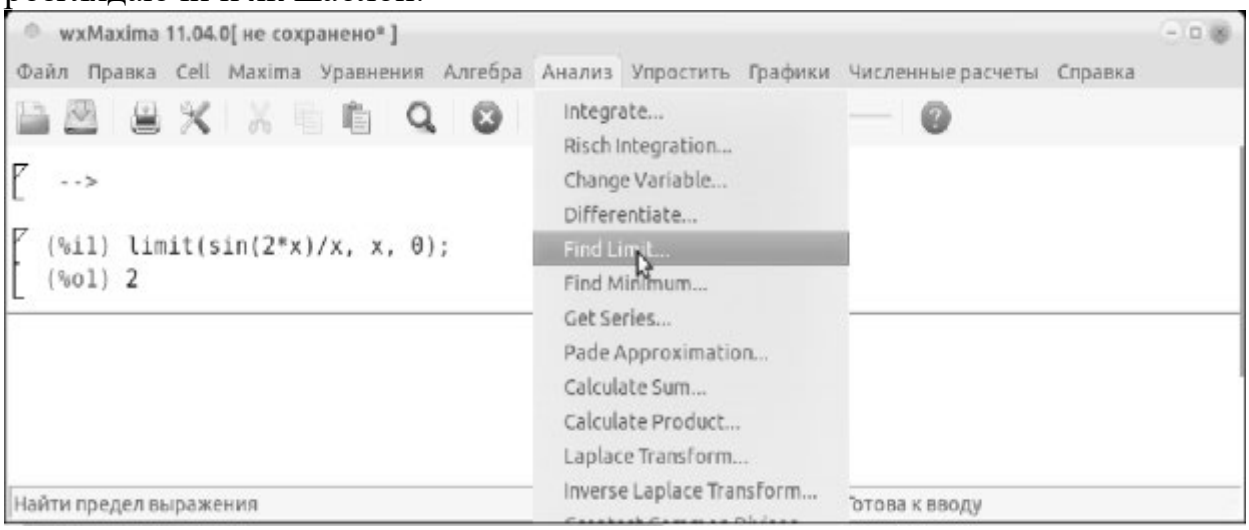
```
(%i1) integrate(3*x+5/x,x);
```

$$(\%o1)\ 5\log(x)+\frac{3x^2}{2}$$

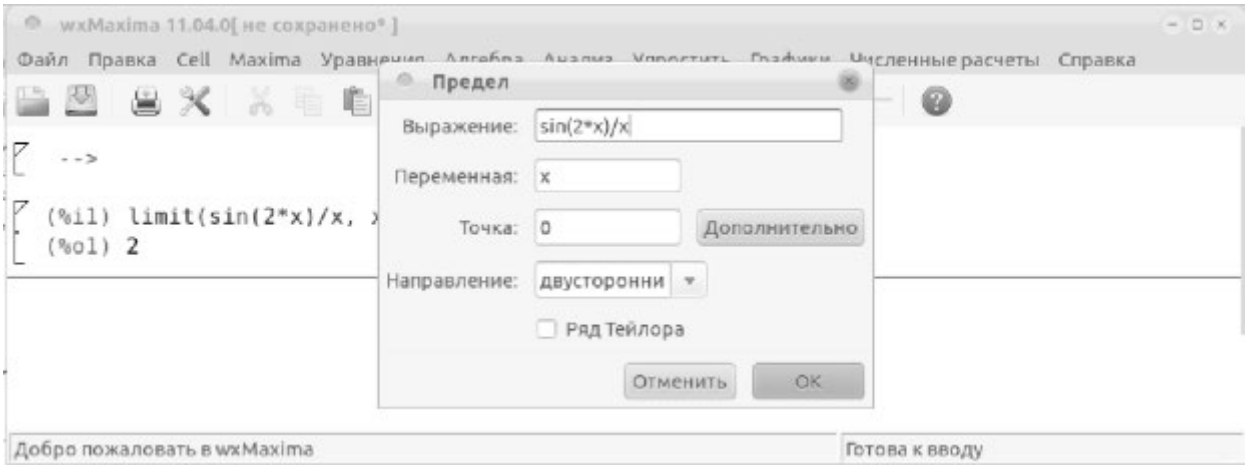
Приклад використання команд меню для обчислення межі

$$\lim_{x\to 0}\frac{sin(2x)}{x}$$

представлений на рис. 5.3 та рис. 5.4. Слід зазначити, що оболонка wxMaxima при виклику команди та відповідного діалогового вікна генерує текстову команду, що інтерпретується обчислювальним ядром Maxima. Переданий ядру Maxima рядок виводиться в командне вікно аналогічно команді, введеній вручну. Після генерації та першого виконання команди (або набору команд) можна доповнювати та редагувати автогенеровану команду, розглядаючи її як шаблон.



Мал. 5.3.Інтерфейс wxMaxima, вибір команди find limit.

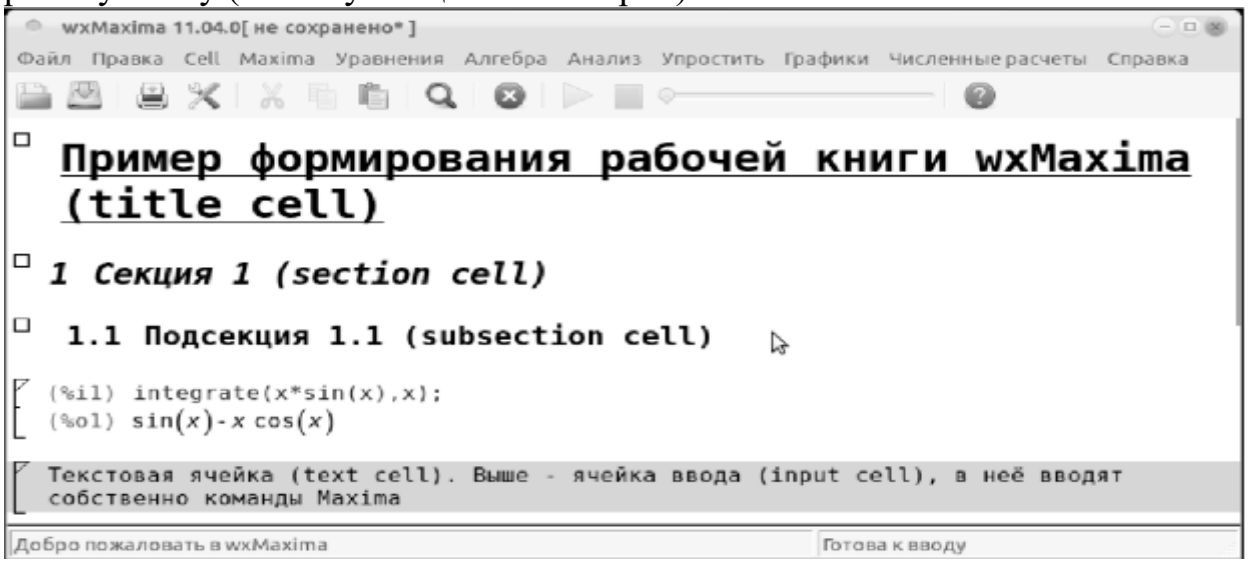


Мал. 5.4.Інтерфейс wxMaxima, вікно введення – обчислення межі.

Нижче розташовується графічне меню основних команд з піктограмами, що відповідають функціям, що найчастіше використовуються для роботи з файлами: відкрити / зберегти / друк даних, а також функціям правки - копіювати / видалити / вставити текст та інші.

Центральну частину робочого вікна wxMaxima займає командне вікно (псевдотермінал), у якому вводяться команди системи та виводяться результати.

В останніх версіях інтерфейсного пакета wxMaxima підтримується концепція осередків у робочій книзі. Осередок включає або набір команд Maxima, або результати їх виконання (в т. ч. графіки). Крім того, за аналогією з Maple та Mathematica wxMaxima підтримує текстові осередки (text cells) для пояснень та коментарів, а також осередки для заголовків та номерів секцій (title cells, section cells, subsection cells). Приклад книги Maxima з осередками зазначених типів представлено на рис. 5.5. Допускається вставка зображень у робочу книгу (також у спеціальні комірки).



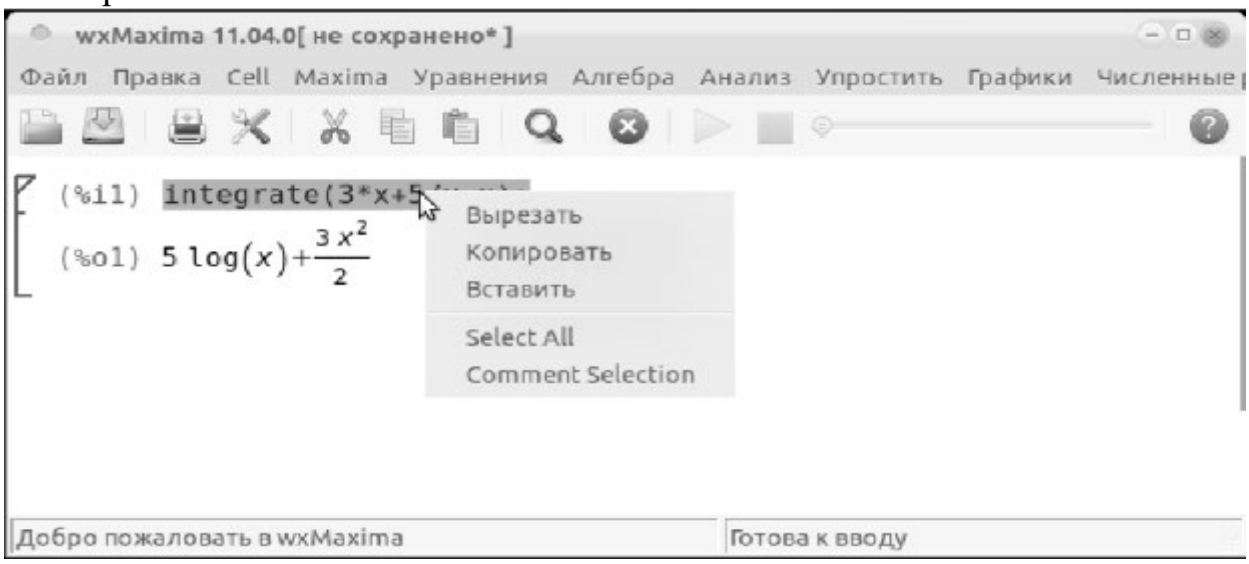
Мал. 5.5.Приклад вставки осередків різних типів у книгу wxMaxima.

При збереженні книги (у форматі wxm) у файл виводяться лише вхідні комірки (input). Тому при роботі зі збереженим документом не обов'язково інтерпретувати всі комірки, хоча це можливо команда Evaluate all cells з меню Cells).

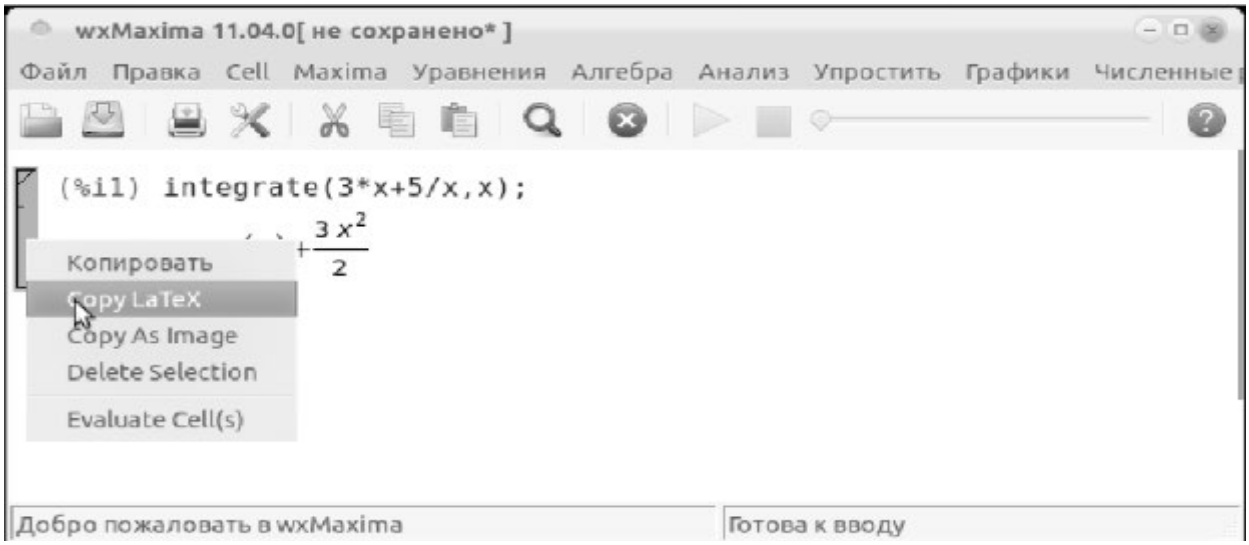
Робочу книгу Maxima можна експортувати у формати HTML або pdf latex.

Інтерпретація поточного осередку, де може бути кілька команд, здійснюється після натискання комбінації клавіш Ctrl+Enter, або командою меню Cells. Якщо необхідно запобігти висновку команди, слід явно завершити її символом $. Сучасні версії wxMaxima автоматично завершують введення, якщо це необхідно, символом ";".

При використанні інтерфейсу wxMaxima можна виділити у командному вікні необхідну формулу та викликавши контекстне меню правою кнопкою миші: скопіювати будь-яку формулу у текстовому вигляді, у форматі TEX або у вигляді графічного зображення для наступної вставки в будь-який документ. Приклад контекстних меню під час роботи з wxMaxima дивись на рис. 5.6 рис. 5.7 та рис. 5.8.



Мал. 5.6.Інтерфейс wxMaxima. Контекстне меню рядка введення.



Мал. 5.7.Інтерфейс wxMaxima. Контекстне меню комірки.

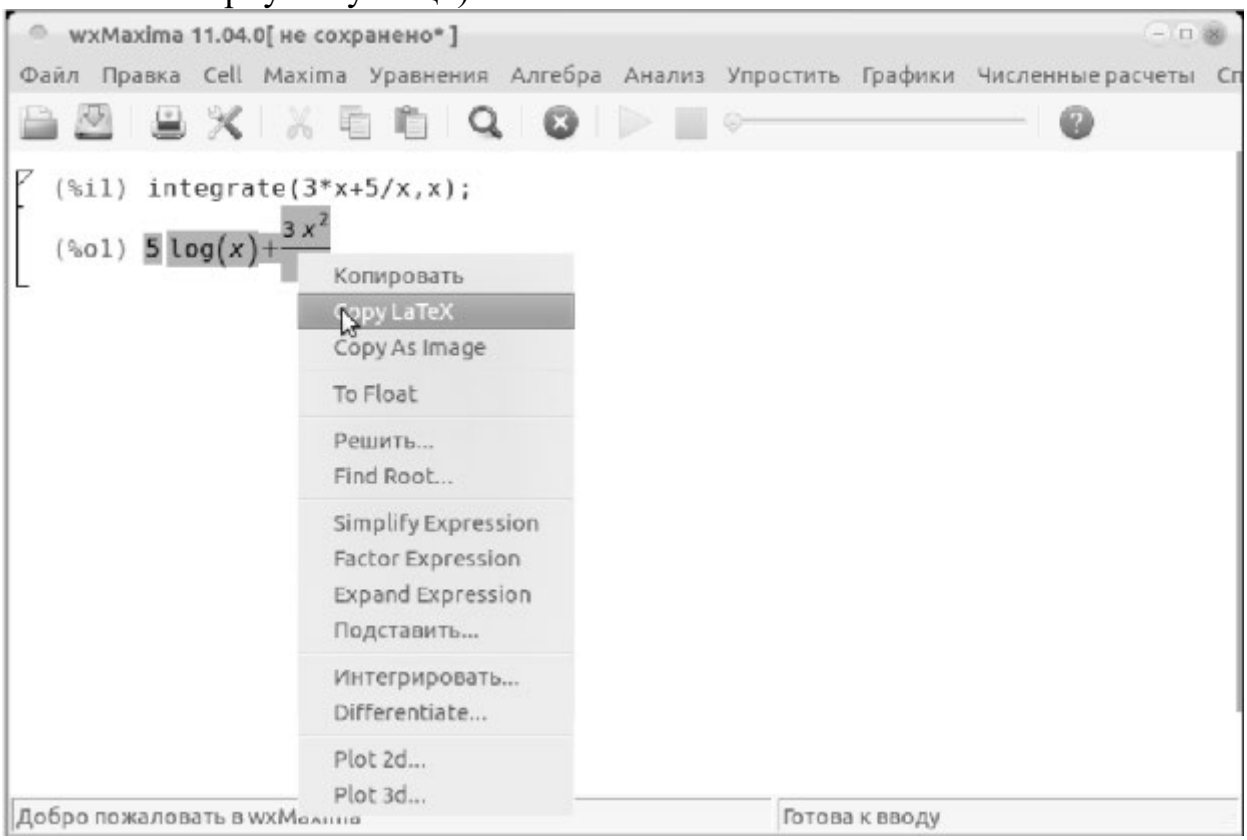
Також, у контекстному меню, при виборі результату обчислення, пропонуються ряд операцій із вибраним виразом (наприклад, спрощення, розкриття дужок, інтегрування, диференціювання та ін.).

За замовчуванням wxMaxima припускає, що команда, що вводиться за допомогою кнопки, застосовується до останнього висновку (тобто аргумент команди -%). Усі кнопки або пункти меню у верхній чи нижній частині робочого вікна відповідають тій чи іншій команді Maxima.

Крім того, wxMaxima надає зручний інтерфейс до документації системи Maxima.

Меню редагування⟶ налаштування забезпечує досить широкі можливості настроювання графічного інтерфейсу wxMaxima. Передбачено три групи параметрів:

- опції, що визначають окремі особливості виконання команд;
- опції виклику обчислювального ядра Maxima;
- опції, що визначають стиль графічного інтерфейсу (мова, шрифти, колірну гаму тощо).



Мал. 5.8.Інтерфейс wxMaxima. Контекстне меню рядка виводу.

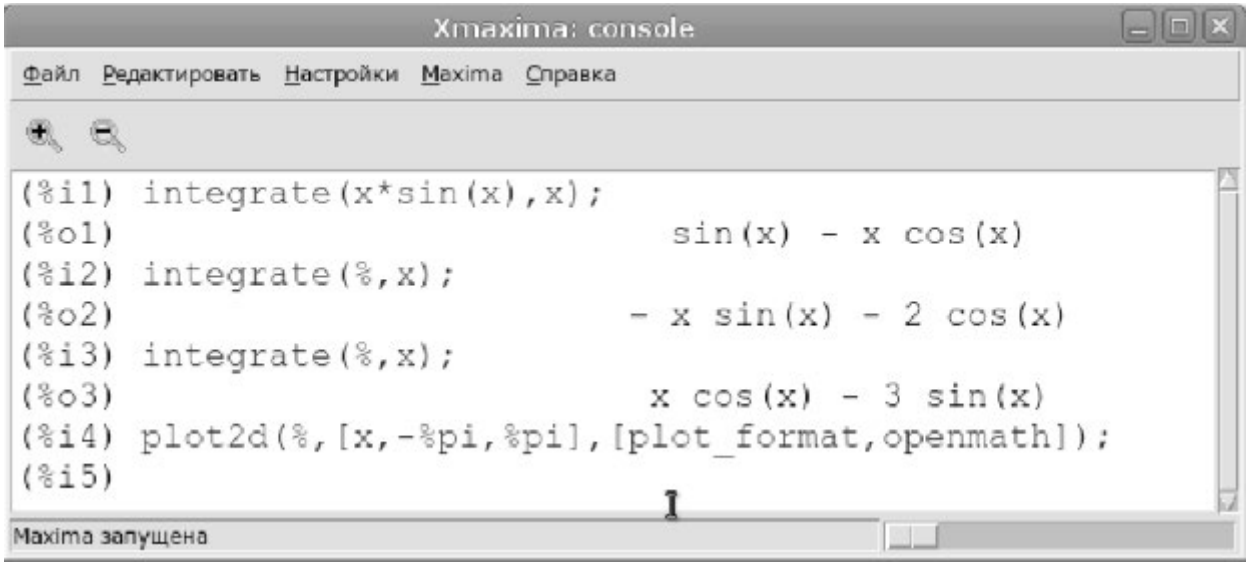
Управління процесом обчислень здійснюється командами пункту головного меню Maxima. Користувачеві надаються такі можливості:

- перервати обчислення, перезапустити Maxima, очистити пам'ять;
- переглянути вміст пам'яті (змінні, функції, визначення тощо);
- змінити формат перегляду результатів.

5.1.2 Графічний інтерфейс xMaxima

Інтерфейс xMaxima практично є специфічним видом веб-браузера, т.к. Цей інтерфейс передбачає обмін даними з обчислювальним ядром Maxima через сокет. Інтерфейс відрізняється простотою (точніше мінімалізмом). В останніх версіях xMaxima при старті відкриваються одночасно вікно браузера системи допомоги та консоль команд.

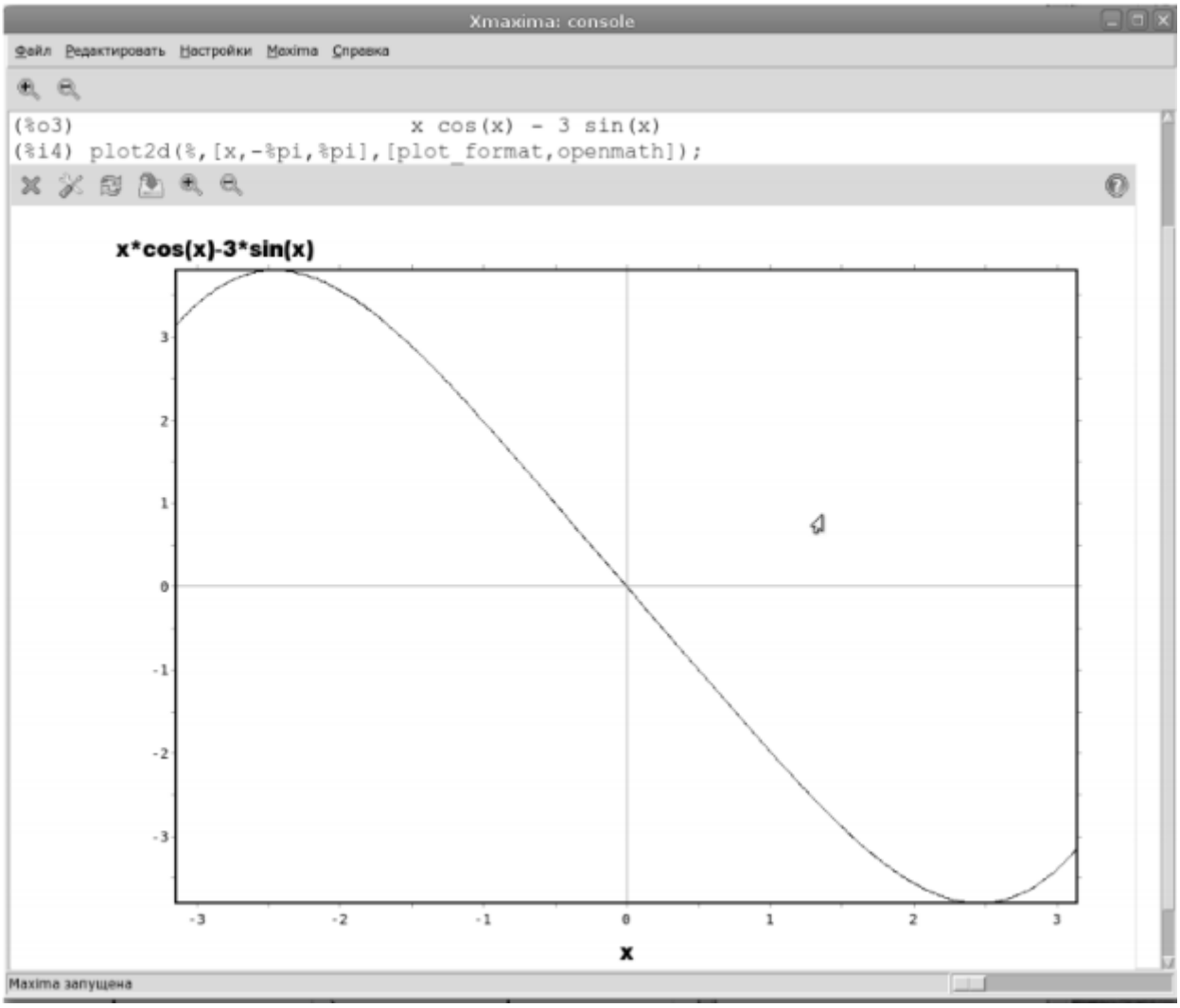
Передбачається, що користувач володіє командами Maxima та макромовою програмування. Загальний вигляд командного вікна xMaxima представлено на рис. 5.9. Пункти меню File, Edit, Options дозволяють керувати сесією Maxima, зберігати та запускати batch-файли. У робочу книгу xMaxima можна вбудовувати графіки у форматі openmath (залежно від встановлення опції plot window). Приклад робочого вікна xMaxima із простими графіками представлений на рис. 5.10. Графік у робочій книзі можна крутити, редагувати, охороняти у файл. Як і wxMaxima, інтерфейс xMaxima надає доступ до html-файлу допомоги пакетом Maxima.



Мал. 5.9.Загальний вигляд робочого вікна xMaxima

5.1.3 Використання редактора TeXmacs як інтерфейс Maxima

Широкі можливості роботи Maxima та інших математичних пакетах надає редактор TeXmacs. Розробник позиціонує його як LaTeX-Редактор, проте це не зовсім так. TeXmacs використовує власний внутрішній формат, але дозволяє експортувати документи до LaTeX (при цьому отриманий TeX-файл дуже схожий на результат експорту до .tex документа OpenOffice). TexMacs добре локалізований та повністю підтримує російську мову, а також усі можливості стандартного текстового процесора

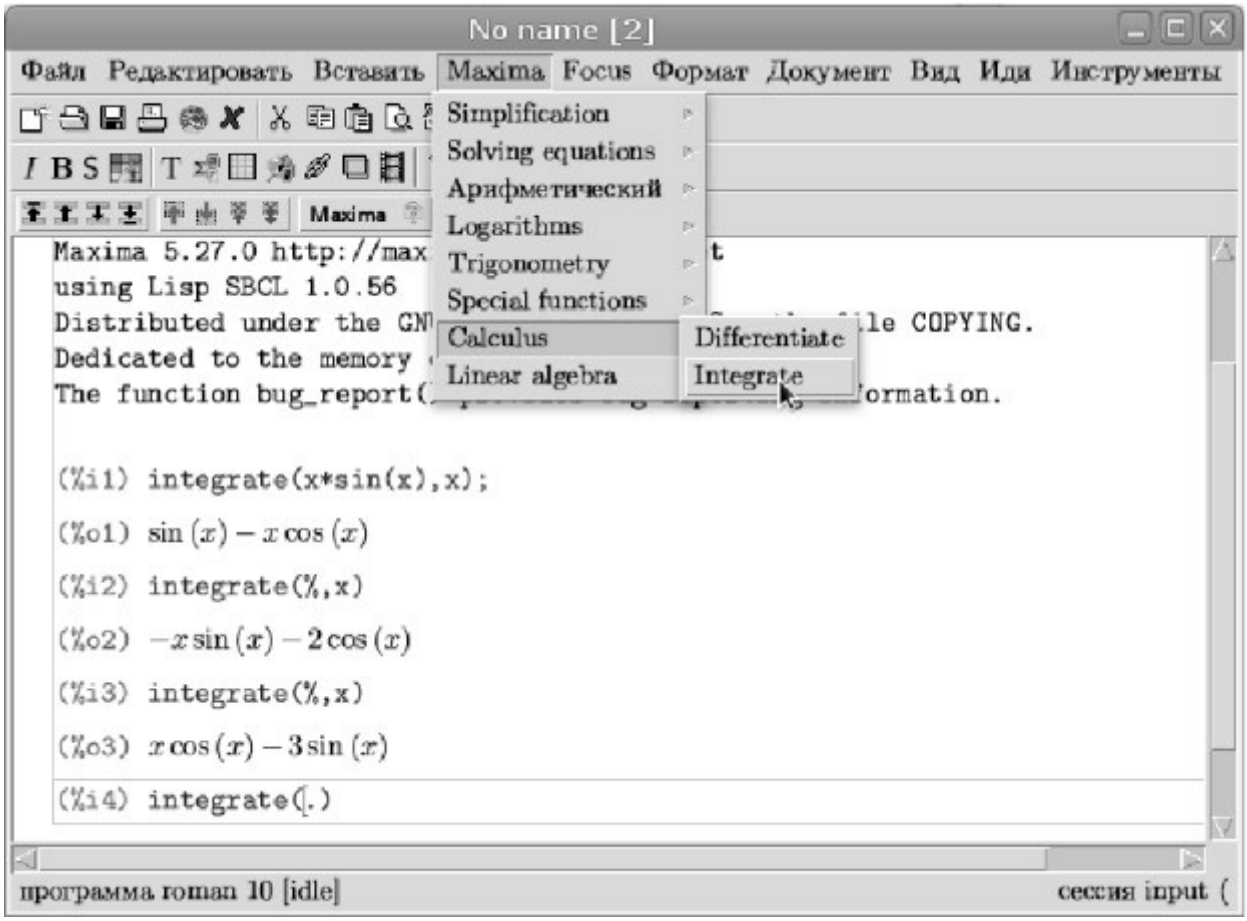


Мал. 5.10.Вбудований графік у робочій книзі xMaxima

У TexMacs реалізовано підхід до структури документа, багато в чому ідентичний LaTeX, а також можливості введення та редагування складних математичних формул. Недоліком редактора є невдалий вибір способу локалізації, що ускладнює відкриття документів TeXmacs за допомогою інших редакторів (OpenOffice та ін.).

Важливою особливістю TeXmacs є можливість вбудовувати текст сесії роботи з різними математичними пакетами (в т.ч. і Maxima). Загальний вигляд робочого вікна TeXmacs представлено на рис. 5.11. Послідовність вставки сесії Maxima у текст документа показано на рис. 5.12.

Можливість вбудовувати в текст документа графічні ілюстрації, а також можливість розщеплювати сесію для введення пояснень та коментарів робить TeXmacs дуже привабливим засобом для роботи з Maxima. У сучасних версіях TeXmacs при запуску сесії Maxima в головному меню з'являється пункт Maxima, в якому передбачено меню з переліком основних команд Maxima. Недоліками TeXmacs є відсутність русифікації під час роботи в Maxima-режимі, а також проблеми на деяких дистрибутивах із запуском сесії Maxima.



Мал. 5.11.Робоче вікно TeXmacs із запущеною сесією Maxima

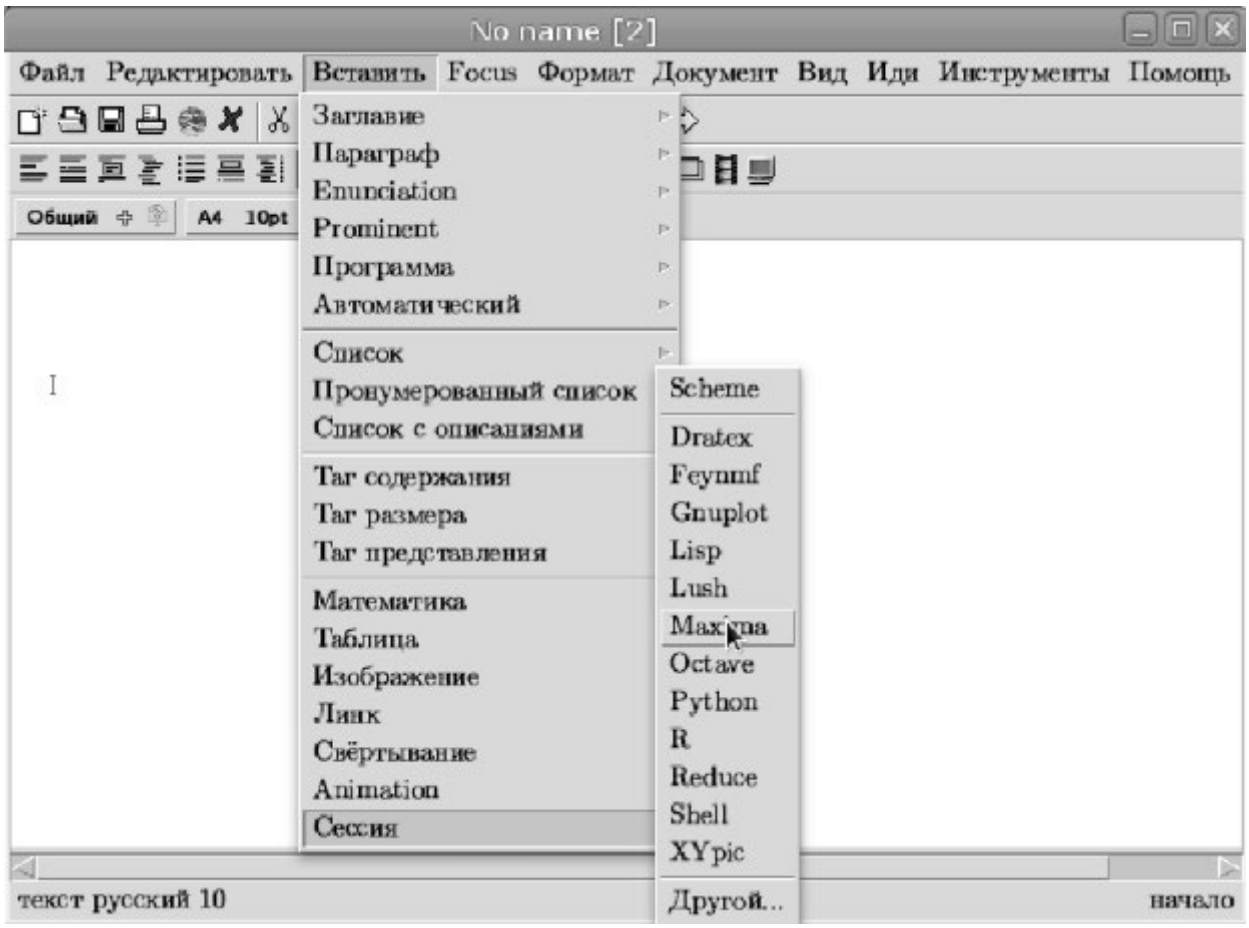
Для вирішення проблем із запуском Maxima-сесії з TeXmacs можливим рішенням є редагування файлу /usr/lib/texmacs/TeXmacs/bin/maxima_detect, в якому треба посилання на #!/bin/sh замінити посиланням на #!/bin/bash у самому початку файлу.

Остаточну версію TeXmacs-документів доцільно представляти у PDF-форматі (цей редактор забезпечує прямий експорт у PDF). При збереженні документів у форматі TeXmacs та їх подальшому редагуванні можливе редагування полів введення сесії Maxima з перерахунком результатів.

5.1.4 Робота з Maxima з Emacs

Універсальний редактор Emacs також може використовуватися як front-end Maxima. Для цього передбачено кілька режимів: maxima-mode, EMaxima та iMaxima.

Основний режим роботи з Maxima в Emacs – maxima-mode. Цей режим запускається клавіатурною комбінацією Mx-maxima-mode (зазвичай натисканням alt-M-alt-x і після появи підказки набір maxima). Цей режим дещо аскетичний (схожий xMaxima), але досить зручний. Загальний вигляд робочого вікна даного режиму представлений на рис. 5.13. На цьому ж малюнку видно меню навігації по поточній сесії, що дозволяє показувати необхідну ділянку сесії, зберігати частину результатів у протокол, повторювати введення команд, що вже використовувалися в даній сесії, і т.п.

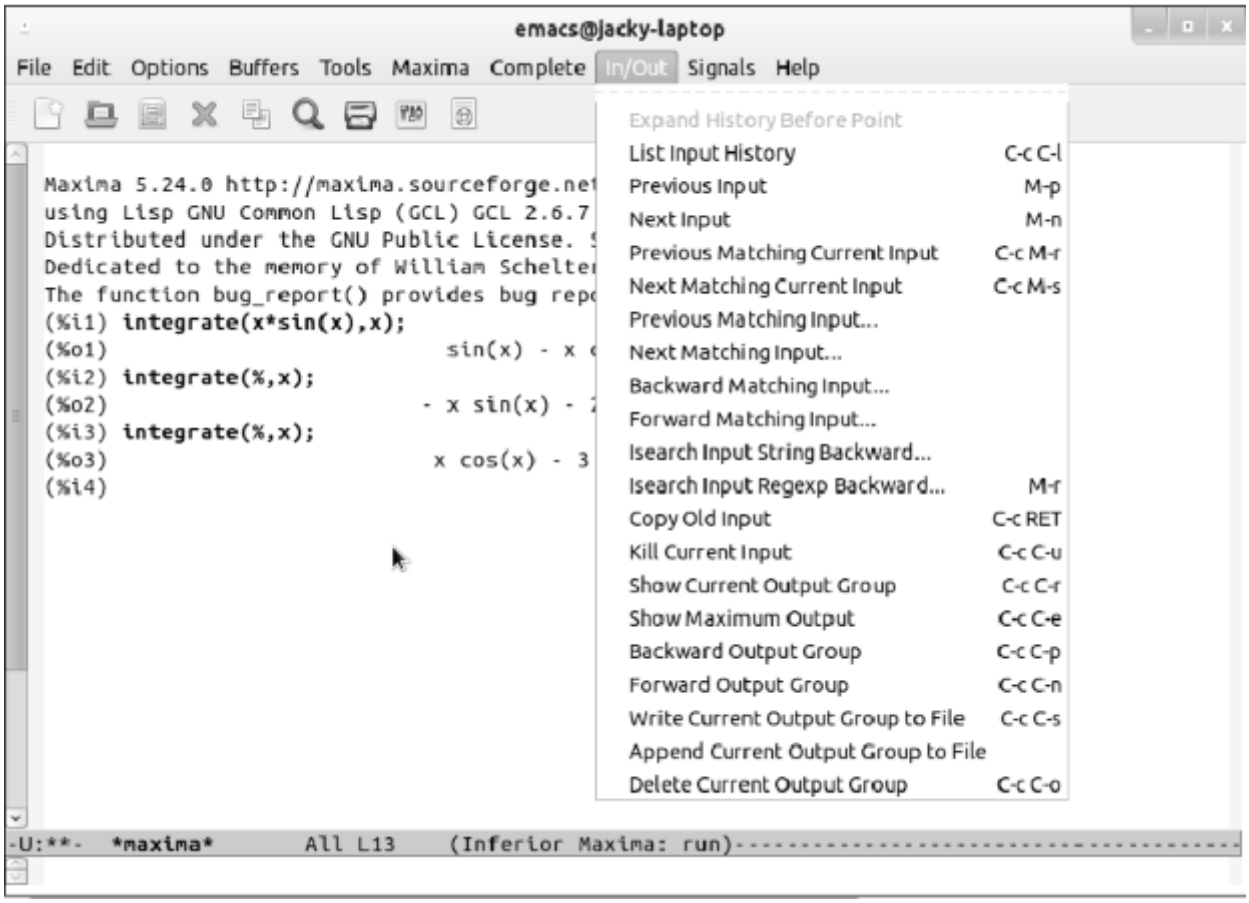


Мал. 5.12.Запуск сесії Maxima у поточному документі TeXmacs

Графіки в робочу книгу, відкриту Emacs, не вбудовуються. Збереження копії малюнка має виконуватися засобами gnuplot чи openmath.

Інтерфейс EMaxima – скоріше не самостійний режим, а надбудова над режимом LaTeX, яка, напевно, сподобається тим, хто використовує Emacs для редагування LaTeX-документів. На відміну від режиму Maxima, який призначений для звичайного ізольованого запуску повноцінної Maxima-сесії, тут йдеться про можливість вставляти окремі команди Maxima і, природно, результати їх обчислень прямо в редагований LaTeX документ. Запуск режиму здійснюється командою EMaxima-mode (Mx emaxima).

У найпростішому випадку з використанням EMaxima можна створити комірку Maxima комбінацією Cc Co ("open cell"), ввести в ній будь-яку команду або набір команд Maxima у текстовій нотації та отримати результат обчислення цієї команди або у звичайному текстовому вигляді натисканням Cc Cu c, або LaTeX-виді за допомогою Cc Cu C (т. Е. Ctrl-c Ctrl-u Shift-c). Тут "u c" походить від "update cell"; а суміжні команди, що генерують висновок у простій текстовій формі та у формі LaTeX, завжди прив'язані в EMaxima до однакових малих і великих букв відповідно. Приклад роботи з EMaxima представлено на рис. 5.14 де показані результати створення комірки з Maxima-кодом і результати доповнення комірки (команди можна вибирати з меню EMaxima у верхній частині робочого вікна).

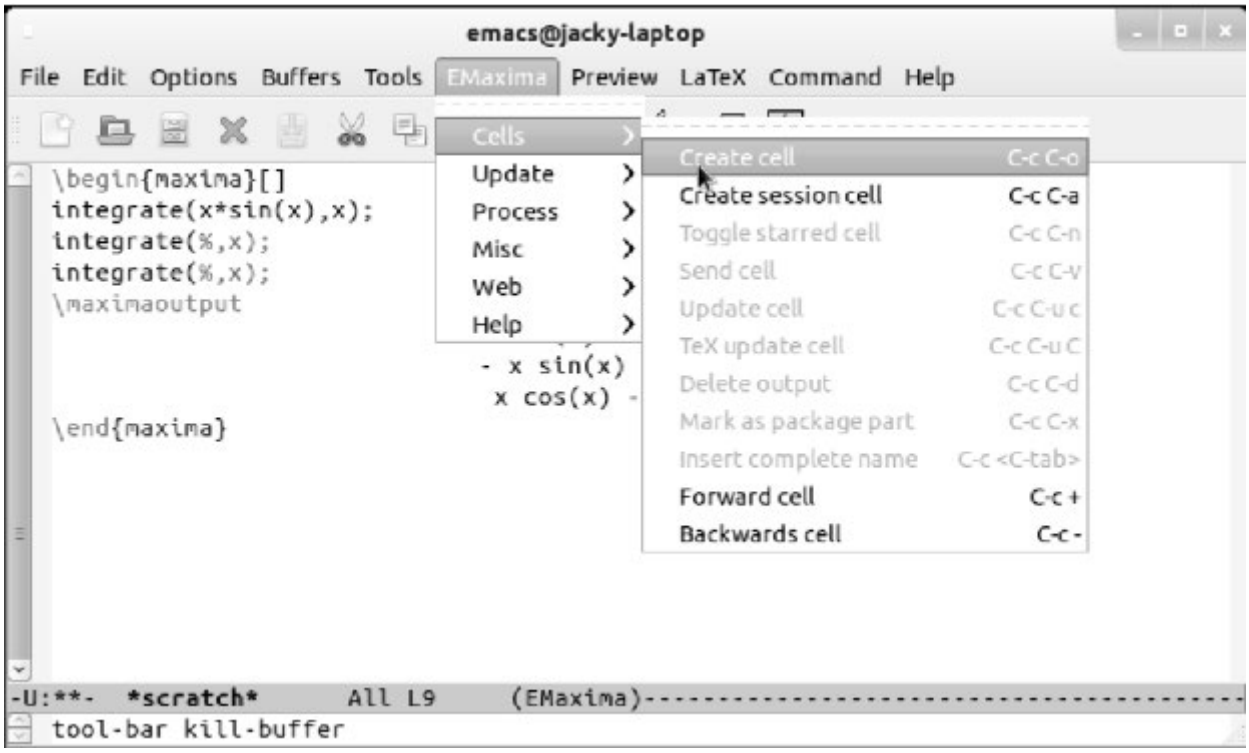


Мал. 5.13.Робоче вікно Emacs із запущеною сесією Maxima

Використовувати інтерфейс EMaxima зручно при створенні об'ємних документів LaTeX математичного характеру, що передбачає включення результатів символьних обчислень.

Останній Emacs-інтерфейс до Maxima - iMaxima - відрізняється від інших самостійним (а не за допомогою LaTeX-документ, як у EMaxima) графічним поданням математичних формул. Власне, саме для цього він і створений, і його відмінність від Maxima-mode полягає саме у можливості графічного відображення TeX-коду, що генерується Maxima.

Цей режим можна налаштувати таким чином, щоб усередині нього запускався режим Maxima (тобто Maxima-Emacs), і користуватися всіма командами останнього та їх клавіатурними прив'язками. Тобто. фактично режим iMaxima у такому варіанті можна розглядати як графічний інтерфейс над Maxima-Emacs; саме це може додати додаткову привабливість останньому. На відміну від усіх розглянутих вище інтерфейсів, iMaxima - сторонній проект, який розробляється окремо. Для його встановлення необхідно додатково встановити пакет breqn, який відповідає за перенесення рядків у математичних формулах у форматі LaTeX. Інструкцію з встановлення самої iMaxima та breqn можна знайти на сайті проекту.



Мал. 5.14.Робоче вікно Emacs із запущеною сесією EMaxima

5.2 Робота з Maxima у KDE: інтерфейс Cantor

Інтерфейскористувача Cantor складається з трьох частин:

- Панель вкладок, за допомогою якої можна перемикатися між документами;
- Панель довідки, де буде показано опис команди, якщо запровадити "? команда";
- Панель поточного документа з меню команд нагадує інтерфейс wxMaxima.

Пакет Cantor розрахований на робочий стіл KDE, тому графічний інтерфейс написаний за допомогою бібліотек Qt.

5.2.1 Документ Cantor

У Cantor ви працюєте з документом. У ньому можна вводити вирази, проводити обчислення та бачити результати. Аналогічно wxMaxima або Emacs, документ Cantor включає комірки, що містять команди поточного пакета та результати виконання. Поряд із командами та результатами, можна вводити і коментарі.

Набір доступних у виразах команд залежить від системи комп'ютерної алгебри, що використовується, тому корисно знати синтаксис конкретної системи. Якщо ви знаєте назву команди, можна переглянути її опис, ввівши "? команда". Щоб переглянути приклади документів Cantor, виберіть пункт меню Файл⟶ Завантажити приклади...та завантажте документи, опубліковані іншими користувачами.

5.2.2 Налаштування

У меню Налаштування можна настроїти зовнішній вигляд поточного документа. Параметр Показувати результати за допомогою LaTeX впливає те, у вигляді будуть показані результати обчислень. Якщо його включено, результат буде оброблений системою LaTeX створення візуально зрозумілих формул. Наприклад, 3*x^2*sqrt(2)*x+2/3 перетвориться на $3x^2 \cdot \sqrt{2} \cdot x + \frac{2}{3}$

Підсвічування синтаксису підвищує читання коду, виділяючи кольором ключові слова та парні дужки. Якщо увімкнути параметр Автодоповнення, Cantor буде показувати можливі продовження команди, що вводиться, при натисканні клавіші Tab. Якщо існує лише одне продовження команди, при натисканні клавіші Tab назва команди буде автоматично введена повністю.

Параметр "Показувати номери рядків" дозволяє додати нумерацію введених виразів. Нумерацію можна використовувати для встановлення попередніх результатів у новий вираз.

5.2.3 Інші можливості Cantor

Документи Cantor зберігаються у вбудованому форматі cws. Кінцевий документ користувача можна зберігати у форматі LaTeX При роботі з Maxima як backend графічні документи можна вбудовувати в документ, за допомогою контекстного меню вони зберігаються у форматі eps.

Опцію Вбудовувати графік у файл можна вимкнути, при цьому графіки відтворюватимуться програмою gnuplot в окремому вікні. За наявності pdf-псевдопринтера можна роздрукувати документ Cantor у форматі pdf (у т.ч. включені до тексту графіки та коментарі).

5.3 Інтегроване середовище Sage

Sage(англ. "Мудрець") - система комп'ютерної алгебри, що покриває багато областей математики, включаючи алгебру, комбінаторику, обчислювальну математику та матаналіз. Першу версію Sage було випущено 24 лютого 2005 року у вигляді вільного програмного забезпечення з ліцензією GNU GPL. Початковою метою проекту було "створення відкритого програмного забезпечення альтернативного системам Magma, Maple, Mathematica, та MATLAB". Розробником Sage є Вільям Стейн – математик Університету Вашингтона (Офіційний сайт: http://sagemath.org).

Численні можливості Sage включають:

- Інтерфейс notebook для перегляду та повторного використання введених команд та отриманих результатів, включаючи графіки та текстові анотації, доступний з більшості сучасних веб-браузерів. Доступно захищене з'єднання через протокол HTTPS, коли конфіденційність має значення. Також Sage може виконуватися як локально, і віддалено.

- Інтерфейс введення на основі командного рядка з використанням мультипарадигменної мови IPython.
- Підтримка паралельних обчислень з використанням багатоядерних процесорів, так і багатопроцесорних систем і систем розподілених обчислень.
- Внутрішня інфраструктура на python, що підтримує взаємодію з математичними пакетами на python: SymPy, SciPy та NumPy.
- Різні статистичні бібліотеки функцій, що використовують функціональність R та SciPy.
- Можливість побудови плоских та тривимірних графіків для функцій та даних.
- Засоби роботи з матрицями та масивами даних із підтримкою розріджених масивів.
- Набір інструментів для додавання власного інтерфейсу до обчислень і додатків.
- Мережеві інструменти для з'єднання з базами даних SQL, підтримка мережевих протоколів, включаючи HTTP, NNTP, IMAP, SSH, IRC, FTP.

Sage— сам собою потужний засіб завдяки численним об'єктно-орієнтованим можливостям і великому обсягу можливостей, реалізованому на python для вирішення різноманітних завдань. Однак слід враховувати, що основна ідея Sage - інтеграція різних математичних пакетів, як відкритих, так і пропрієтарних. Поряд із функцією інтеграції, Sage включає досить розвинені власні можливості - численні функції та структури даних. Перетворення результатів, отриманих, наприклад, Maxima, до структур Sage може виявитися досить складним завданням.

5.4 Побудова графічних ілюстрацій: пакет draw

Maxima має кілька альтернативних бібліотек для відображення графіків функцій, наборів точок, тривимірних тіл, градієнтів і т.д. За замовчуванням використовується бібліотека Plot, але для вирішення деяких завдань може бути зручніше бібліотека Draw.

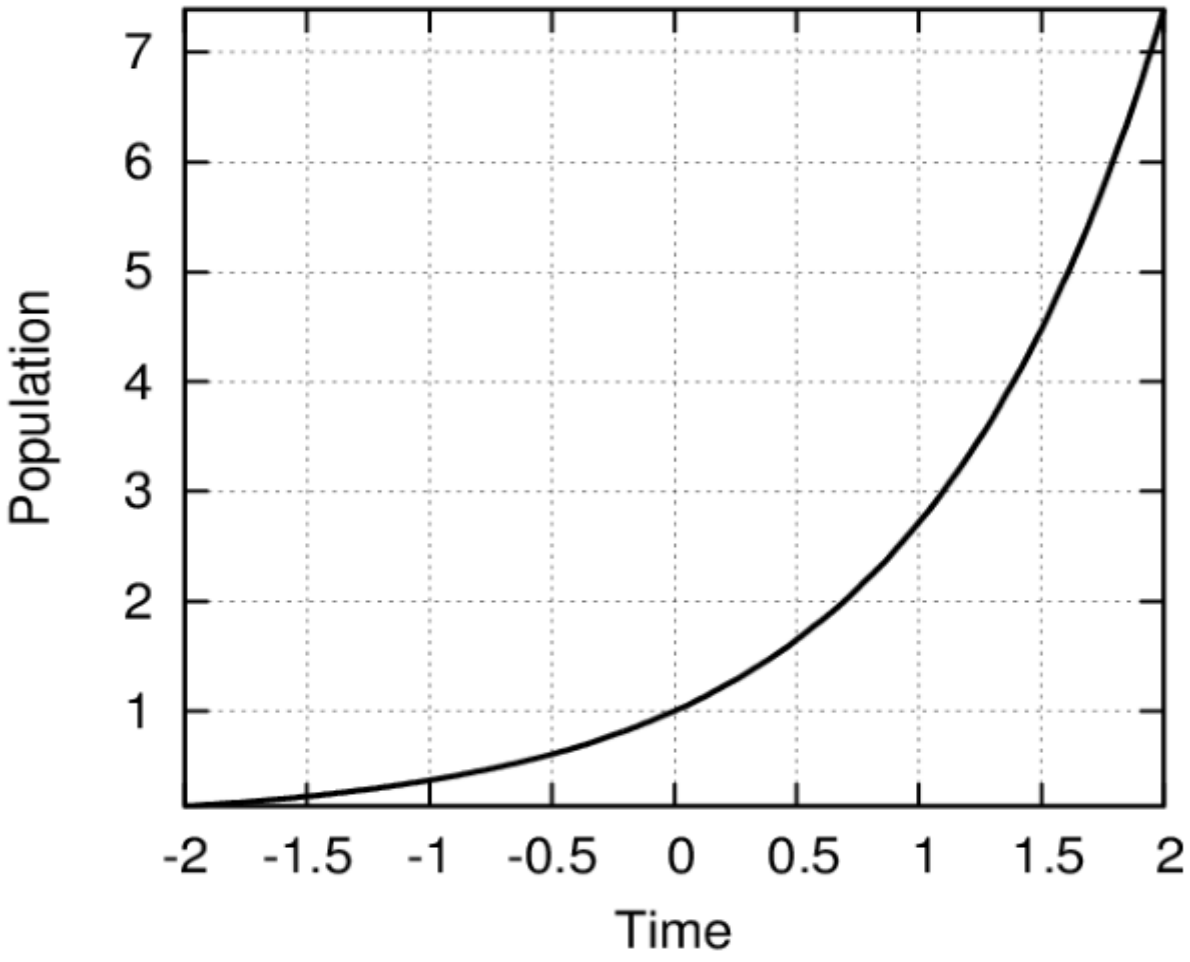
Варіанти використання команди plot2d розглянуті вище, тож нижче ілюструються можливості draw. Бібліотека draw побудована на інтерфейсі Maxima-gnuplot. Бібліотека включає три основні функції, доступні на рівні Maxima: draw2d, draw3d, draw. Перед використанням draw необхідно завантажити командою load(draw).

Розглянемо нескладний приклад. На графіці (рис. 5.15) показано криву $y = exp(x)$. Графік побудований з використанням функції $draw2d$. Функції, задані явно, вказуються командою $explicit$. Для кожної функції вказується ім'я змінної та межі зміни абсциси. Межі ординати вибираються автоматично. Команда побудови графіка:

```
(% i4)     draw2d(grid = true, xlabel = "Time",
```

```
    ylabel = "Population", explicit(exp(u),u,-2,2))$
```

На графіці показано сітка (grid=true), і навіть мітки осей (xlabel і ylabel).



Мал. 5.15.Графік побудований за допомогою функції draw2d

Висновокграфіка на друк організується за допомогою вказівки типу терміналу. Можливі варіанти – screen (екран за замовчуванням), png, jpg, eps, eps_color, gif, animated_gif, wxt, aquaterm.

Командадля виведення графіка на друк має вигляд (зазначений термінал eps - encapsulated postscript):

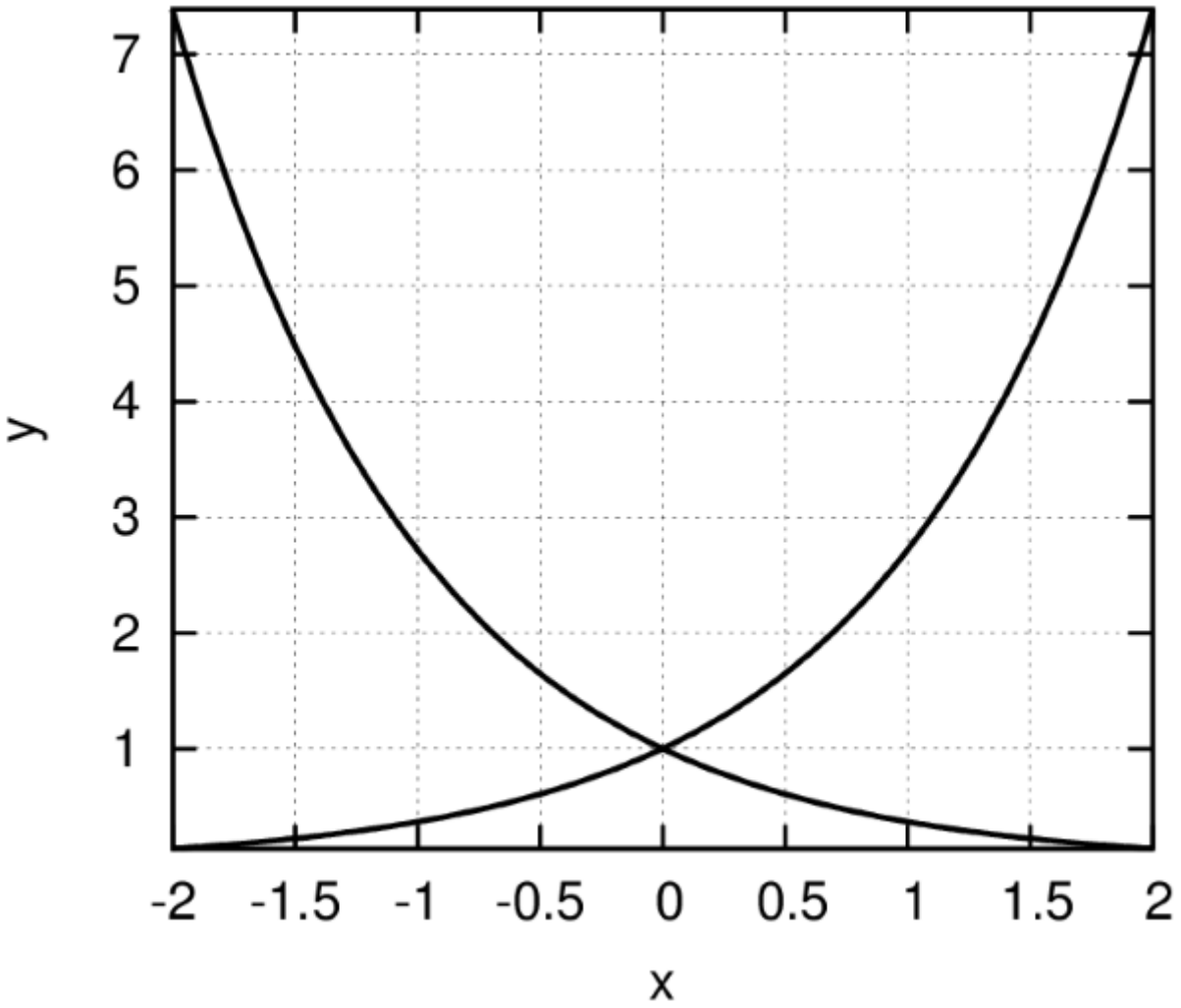
```
(%i6) draw2d(terminal = eps, grid = true, xlabel =
"Time",
      ylabel = "Population", explicit(exp(u),u,-2,2))$
```

зазамовчування малюнок зберігається у файл maxima_out.eps; Вказівка імені файлу для виведення здійснюється командою file_name = "ім'я файлу".

Побудуємо аналогічний графік (рис. 5.16), але з виведенням кривих $y = \exp(x)$і $y = \exp(-x)$в одних осях із збереженням графіка у файл draw_2.eps. Необхідна команда:

```
(%i7) draw2d(terminal=eps, file_name="draw_2",grid=true,
      xlabel = "x", ylabel = "y", explicit (exp (u), u, -
2,2),
```

```
explicit(exp(-u),u,-2,2))$
```



Мал. 5.16.Графік двох функцій (використана функція draw2d

За допомогою пакета draw можна будувати й графіки функцій, що були задані неявно. У цьому випадку функція задається командою implicit, наприклад (результат побудови – на рис. 5.17):

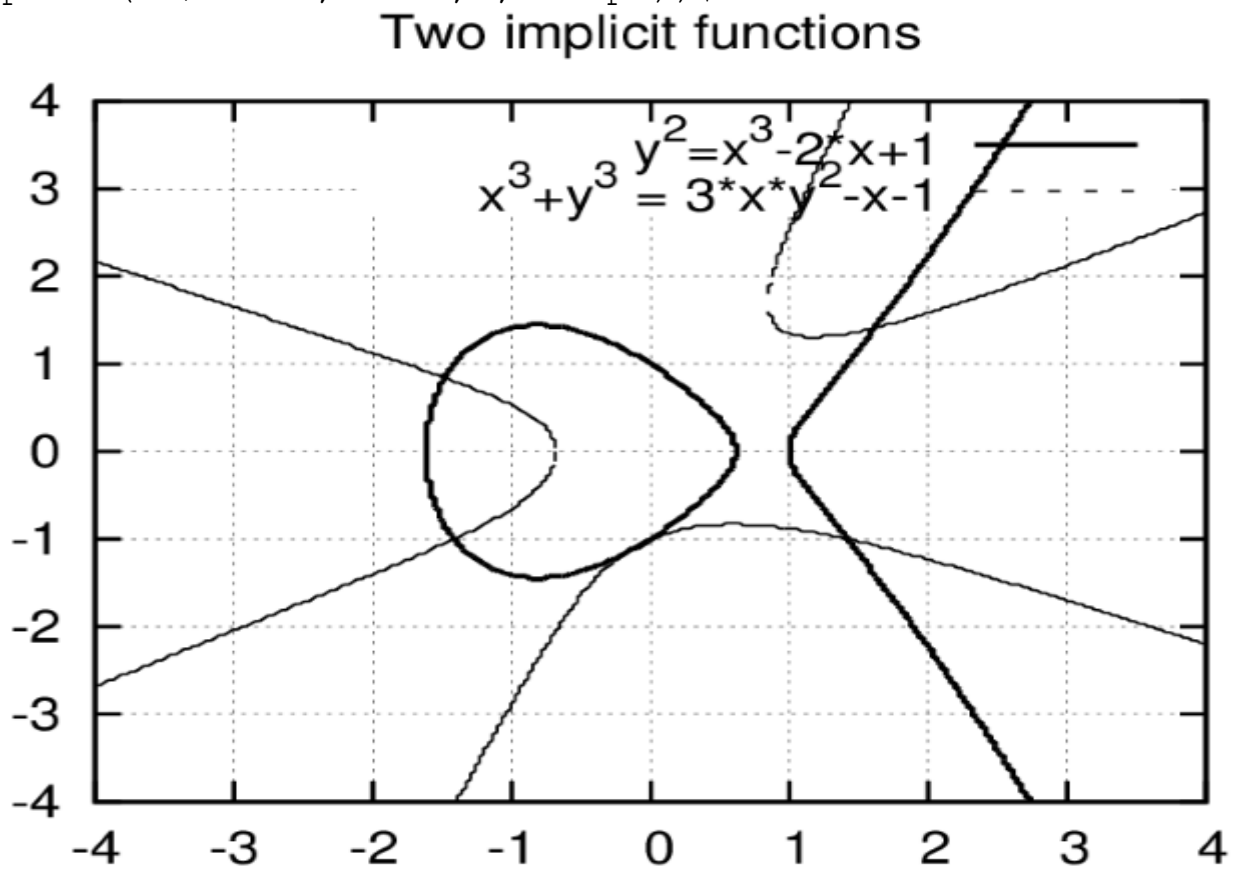
```
(%i1) load(draw)$
(%i2) draw2d(grid = true, title = "Two implicit
functions",
      line_type = solid, key = "y^2=x^3-2*x+1",
      implicit(y^2=x^3-2*x+1, x, -4,4, y, -4,4),
      line_type = dots, key = "x^3+y^3 = 3*x*y^2-x-1",
      implicit(x^3+y^3 = 3*x*y^2-x-1, x,-4,4, y,-4,4))$
```

На графіку добре видно, що криві проведені різними лініями (одна суцільна, інша точкова). Для вказівки типу лінії використана опція line_type = тип лінії (можливі значення - solid і dots).

Крім графіків неявних функцій, за допомогою draw можуть бути побудовані графіки параметричних функцій або функцій, заданих у полярних координатах. У цих випадках замість команд explicit або implicit використовуються команди

parametric і polar відповідно. Приклад графіка функції полярних координатах — на рис. 5.18. Відповідна команда:

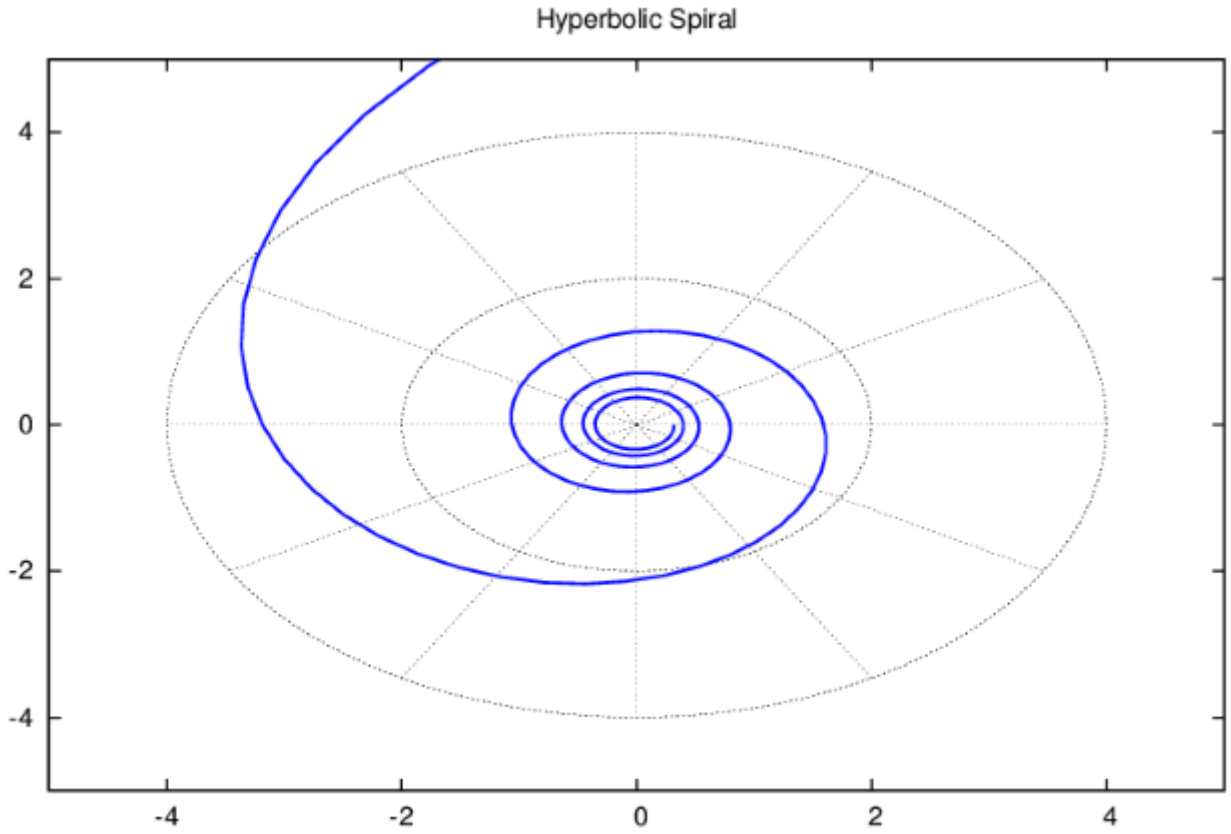
```
draw2d(user_preamble = "set grid polar", nticks = 200,
xrange = [-5,5], yrange = [-5,5], color = blue,
line_width = 3,
title = "Hyperbolic Spiral",
polar(10/theta,theta,1,10*%pi))$
```



Мал. 5.17.Графік двох функцій, заданих неявно

В останньому прикладі наведено параметри побудови графіка: інтервали зміни x і y, рівні xrange та yrange, товщина лінії line_width та її колір color. Крім того, важливим параметром є user_preamble. Ця опція вказує команди gnuplot, визначені користувачем та виконуються перед побудовою даного графіка.

*Для використання міток осей та заголовків російською мовою необхідно в user_preamble або у спеціальному файлі .gnuplot (цей файл містить команди gnuplot, що виконуються при старті програми) вказати російське кодування командою set encoding koi8r або українське кодування set encoding koi8u.*Крім того, часто необхідно вказати і шрифт для виведення заголовка або міток.



Мал. 5.18.Графік функції у полярних координатах

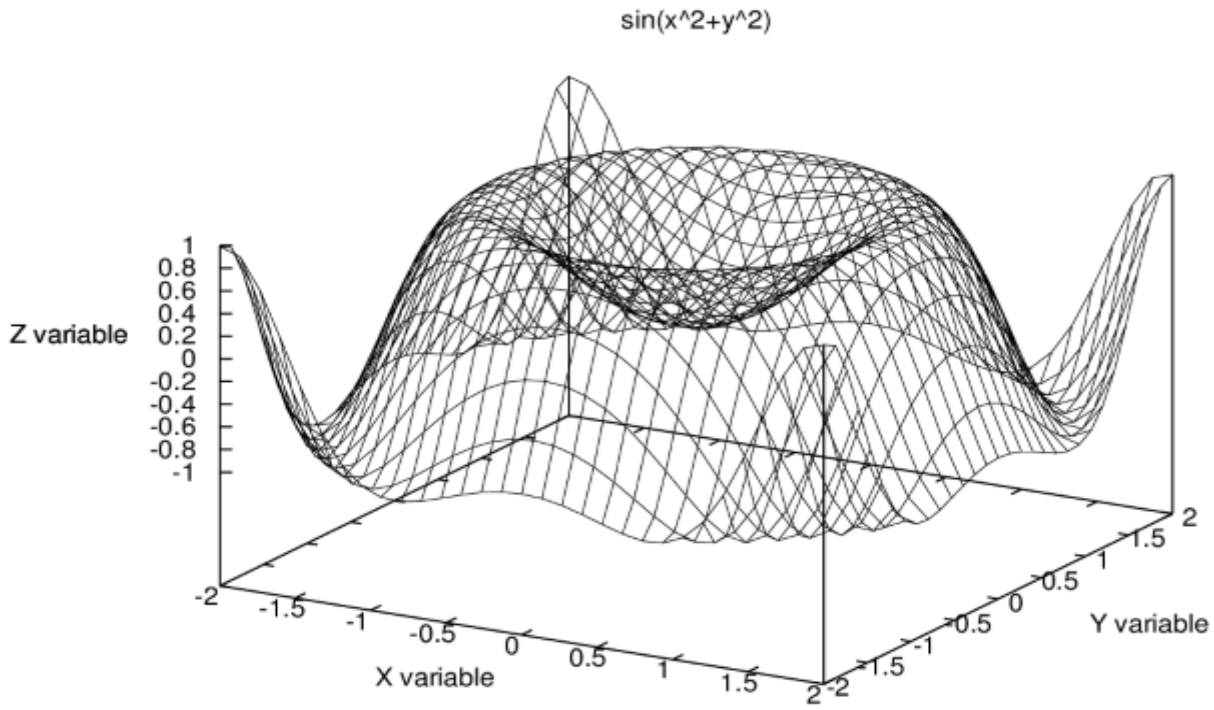
Функція $draw3d$ дозволяє будувати тривимірні графіки. Приклад:

```
(% i8)     draw3d(zlabel = "Z variable", ylabel = "Y
variable",
     explicit(sin(x^2+y^2),x,-2,2,y,-2,2), xlabel="X
variable")$
```

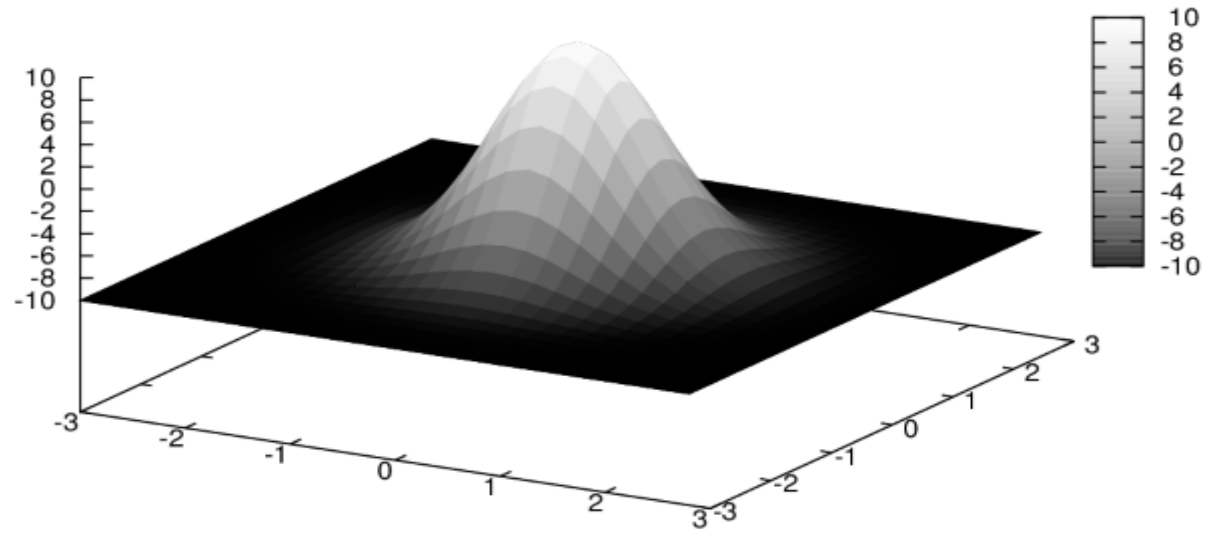
Міткаосі z вказується командою zlabel=ім'я. Виведення графіка на друк аналогічне зазначеному вище. Приклад (із зазначенням, окрім міток осей, та назви графіка командою title=ім'я) наведено на рис. 5.19.

Очевидно, що за допомогою функції$draw3d$, Можна будувати і пофарбовані поверхні (або напівтонові). Для цього як аргумент функції $draw3d$ вказується опції$enhanced3d$ (Вказує на побудову тривимірної пофарбованої поверхню) і$palette$ (palette = color - кольорова поверхня, palette = gray - відтінки сірого). Приклад поверхні, пофарбованої відтінками сірого – на рис. 5.20. Необхідна команда:

```
(%i12)     draw3d(terminal=eps,surface_hide=true,
enhanced3d=true,
     palette=gray, explicit(20*exp(-x^2-y^2)-10,x,-
3,3,y,-3,3))$
```



Мал. 5.19.Поверхня побудована за допомогою функції draw3d



Мал. 5.20.Пофарбована поверхня (використана функція draw3d).

Пакет draw дозволяє і будувати кілька графіків на одному малюнку, а також надає ряд інших корисних можливостей, але для їх використання необхідно ознайомитися з документацією, що постачається з пакетом Maxima.

6. Моделювання з Maxima

6.1 Загальні питання моделювання

На сьогоднішній день існує велика література з різних аспектів моделювання систем, природних, технічних та економічних об'єктів.

Один із широко застосовуваних універсальних програмних засобів для вирішення задач моделювання – пакет Matlab та його розширення для візуальної побудови блокових моделей – пакет Simulink. Підхід, пов'язаний із побудовою блокових моделей, розвивався протягом багатьох років, і має велику теоретичну базу (див., наприклад, відомий підручник з моделювання систем [23]).

Пакет Maxima, що розглядається в даній книзі, не розрахований на побудову блокових моделей технічних систем або систем масового обслуговування, але може бути дуже корисний для побудови та аналізу аналітичних або емпіричних ідентифікованих моделей.

Побудова, класифікація та ідентифікація математичних моделей - широка сфера наукової та практичної діяльності, широко освітлена в літературі різного призначення.

Визначимо модель як зображення істотних сторін реальної системи (або системи, що конструюється), в зручній формі відображає інформацію про неї. Моделі бувають концептуальні, фізичні або математичні (інші назви: феноменологічні, емпіричні та аналітичні) залежно від того, яка сторона явища в даному випадку найбільш істотна, від методів, які можна використовувати при побудові моделі, кількості та якості наявної інформації (див. [24]).

На думку П. Ейкхоффа, при побудові моделі інформація має бути представлена у зручній формі. Це суттєво, оскільки модель має створити передумови для наступних рішень. Якщо модель дуже складна, її корисність стає сумнівною. Відносна простота є основною характеристикою моделі. Модель є спрощеним відображенням дійсності. У багатьох випадках, для того, щоб модель була корисною, її складність повинна перебувати у певному співвідношенні зі складністю описуваного об'єкта (приклад: біологічні системи).

За способом подання інформації про досліджуваний об'єкт і спосіб побудови моделі поділяються на такі групи:

- словесні або вербальні моделі (описи об'єкта моделювання природною мовою);
- фізичні моделі, що передбачають уявлення основних властивостей об'єкта моделювання будь-яким матеріальним об'єктом (моделлю, макетом тощо);
- формальні моделі, що є описом об'єкта моделювання формальною мовою, до цієї групи належать математичні моделі.

У свою чергу, математичні моделі поділяються на (див. [25]) графічні, табличні, алгоритмічні, аналітичні.

Засобами Maxima можна будувати досить широке коло різних математичних моделей.

Процедуру побудови моделі у багатьох джерелах називають ідентифікацією, у своїй даний термін часто належить до побудови аналітичних математичних моделей динамічних об'єктів.

Зазвичай ідентифікація – багатоетапна процедура. Основні її етапи такі:

1. Структурна ідентифікація полягає у визначенні структури математичної моделі на підставі теоретичних міркувань
2. Параметрична ідентифікація включає проведення ідентифікуючого експерименту і визначення оцінок параметрів моделі за експериментальними даними.
3. Перевірка адекватності – перевірка якості моделі у сенсі обраного критерію близькості виходів моделі та об'єкта.

6.1.1 Аналітичні моделі

Аналітичні моделі є відображенням взаємозв'язків між змінними об'єкта у вигляді математичної формули або групи таких формул. Моделювання засноване на двох основних ознаках:

- за принципом практичної обмеженості кількості фундаментальних законів природи;
- на принципі подібності, що означає, що явища різної фізичної природи можуть описуватись однаковими математичними залежностями.

Для аналітичного моделювання характерно те, що процеси функціонування елементів системи записуються у вигляді деяких функціональних співвідношень (алгебраїчних, інтегро диференціальних, звичайно різницевих тощо) або логічних умов. Аналітична модель може бути досліджена такими методами (див. [26]):

- аналітичним, коли прагнуть отримати у загальному вигляді явні залежності для шуканих характеристик;
- чисельним, коли, не вміючи вирішувати рівнянь у загальному вигляді, прагнуть отримати числові результати за конкретних початкових даних;
- якісним, коли, не маючи рішення у явному вигляді, можна знайти деякі властивості рішення (наприклад, оцінити стійкість рішення).

Найбільш повне дослідження процесу функціонування системи можна провести, якщо відомі явні залежності, що пов'язують шукані характеристики з початковими умовами, параметрами та змінними системи. Проте такі залежності вдається отримати лише порівняно простих систем. При ускладненні систем дослідження їх аналітичним методом наштовхується на значні труднощі, які часто бувають непереборними. Тому, бажаючи використовувати аналітичний метод, у разі йдуть істотне спрощення початкової моделі, щоб мати можливість вивчити хоча б загальні властивості системи. Таке дослідження на спрощеній моделі аналітичним методом допомагає

отримати орієнтовні результати визначення більш точних оцінок іншими методами.

Можливості суто теоретичної побудови математичної моделі зменшуються зі зростанням складності та новизни досліджуваного об'єкта. Втім, досвід показує, що нерідко навіть для широко використовуваних на практиці і, здавалося б добре вивчених об'єктів і процесів, суто аналітичним шляхом побудувати задовільну модель не вдається і це спонукає дослідника формування моделі переважно на експериментальній основі, тобто. у класі емпіричних (ідентифікованих) моделей.

За рівнем відповідності моделі реальному об'єкту моделі можна поділити:

- - на заможні - що спираються на закони, що характеризують об'єкт моделювання в галузі їх застосування;
- - апроксимації - побудовані на основі наближених або емпіричних формул, що характеризують об'єкт (їх, на відміну від перших, називають неспроможними - див [26]).

6.1.2 Ідентифіковані моделі

В основі всіх нині дуже численних методів ідентифікації або досвідченого ототожнення моделі з об'єктом-оригіналом лежить ідея уявного експерименту з "чорним ящиком" (М. Вінер). У граничному (теоретичному) випадку "чорний ящик" є якоюсь системою, про структуру і внутрішні властивості якої невідомо зовсім нічого. Натомість входи, тобто. зовнішні фактори, що впливають на цей об'єкт, і виходи, що є реакцією на вхідні впливи, доступні для спостережень (вимірювань) протягом необмеженого часу. Завдання полягає в тому, щоб за даними про входи і виходи виявити внутрішні властивості об'єкта або, іншими словами, побудувати модель (див. [27]).

Модель чорної скриньки є початковим етапом вивчення складних систем.

Дослідження об'єкта моделювання припускає застосування двох стратегій:

1. Здійснюється активний експеримент. На вхід подаються спеціальні сформовані тестові сигнали, характер та послідовність яких визначено заздалегідь розробленим планом. Перевага: за рахунок оптимального планування експерименту необхідна інформація про властивості та характеристики об'єкта виходить за мінімального обсягу первинних експериментальних даних і відповідно за мінімальної трудомісткості дослідних робіт. Але ціна за це досить висока: об'єкт виводиться із його природного стану (або режиму функціонування), що не завжди можливо.
2. Здійснюється пасивний експеримент. Об'єкт функціонує у своєму природному режимі, але при цьому організуються систематичні виміри та реєстрація значень його вхідних та вихідних змінних. Інформацію отримують таку ж, але необхідний обсяг даних зазвичай більше, ніж у першому випадку.

Насправді при побудові ідентифікованих (емпіричних) моделей часто доцільна змішана стратегія експерименту. За можливості вільного маніпулювання параметрами об'єкта моделювання проводиться активний

експеримент. Його результати доповнюють даними пасивного експерименту, що охоплює решту значних змінних. "Чорна скринька" - теоретично граничний випадок. Насправді є обсяг вихідної інформації. Насправді доводиться мати справу з " сірим " (частково прозорим) ящиком.

Побудова моделі зводиться до наступних етапів (див. [24]):

1. вибір структури моделі із фізичних міркувань;
2. припасування параметрів до наявних даних (оцінювання);
3. перевірка та підтвердження моделі (діагностична перевірка);
4. використання моделі та її призначення.

Виходячи з переліку наукових напрямів та різноманітності додатків, по цих етапах не можна дати будь-яких загальних рекомендацій. Структура моделі вибирається на основі вихідної (апріорної) інформації про систему та переслідуваних цілях На практиці відшукання відповідної моделі може бути досить важким завданням навіть для вузької прикладної області.

Розрізняють три основні класи постановки задачі ідентифікації об'єкта:

Для складних і слабо вивчених об'єктів системного характеру достовірні вихідні дані про внутрішні властивості та структурні особливості зазвичай відсутні.

Тому завдання ідентифікації включає як визначення внутрішньої структури об'єкта, так і визначення залежностей, що зв'язують входи і виходи (узагальненого оператора).

На початковій стадії моделювання будуються емпіричні моделі, що ідентифікуються (на основі статистичної обробки експериментальних результатів).

1. Другий клас завдань ідентифікації характеризується тим, що є апріорні дані про структуру об'єкта, що моделюється, в принципі є. Однак не визначено вклад компонентів об'єкта в його результуючі характеристики. Завдання цього класу, пов'язані з уточненням структури та оцінювання параметрів, часто зустрічаються на практиці та характерні для об'єктів та процесів середньої складності, зокрема технологічних.
2. Третій клас завдань пов'язаний з відносно простими і добре вивченими об'єктами, структура яких відома точно і йдеться лише про те, щоб за експериментальними даними оцінити значення всіх або деяких параметрів, що входять до досліджуваної структури (параметрична ідентифікація). Очевидно, що моделі даного класу тісно стуляються з аналітичними моделями, що вимагають експериментального довизначення, і чіткої межі між ними не існує. Це наймасовіший клас завдань.

Незалежно від характеру розв'язуваної на основі ідентифікації задачі, побудова моделі базується на результатах вимірювань відповідних змінних величин.

Реальні властивості переважної більшості складних об'єктів, а також неминучі випадкові похибки вимірювань, що лежать в основі ідентифікації, надають останній статистичний характер, що спричиняє необхідність

отримання великих обсягів первинних експериментальних даних з їх подальшою обробкою. Тому практично побудова моделей шляхом ідентифікації неминуче пов'язані з використанням комп'ютерів, як із отриманні первинних даних (автоматизація експерименту), так їх обробки та використання.

У Maxima включено досить багато коштів на вирішення завдань моделювання, параметричної ідентифікації, дослідження моделей.

6.2 Статистичні методи аналізу даних

6.2.1 Введення-виведення матричних даних

Для читання та запису матричних або потокових даних у складі Maxima передбачено пакет numericalio.

Функції пакета розраховані на введення/виведення даних, кожне поле яких передбачається атомом (у сенсі Lisp), тобто. цілих чисел, чисел із плаваючою точкою, рядків або символів. Атоми сприймаються numericalio так само, як при інтерактивному введенні консолі або виконанні batch-файлу. Можливе використання різних символів-сепараторів для поділу полів даних (параметр $separator_flag$).

Основні функції пакету numericalio:

- $read_matrix(file_name)$

  інші форми виклику
  $read_matrix(file_name, separator_flag), read_matrix(S), read_matrix(S, separator_flag)$
  Функція $read_matrix$ зчитує матрицю із файлу. Тут $file_name$ - Ім'я файлу, з якого зчитуються дані, S - Ім'я потоку. Якщо не вказано $separator_flag$ дані передбачаються розділеними пробілами. Функція повертає обліковий об'єкт.

- $read_list(file_name)$

  інші форми -
  $read_list(file_name, separator_flag), read_list(S), read_list(S, separator_flag)$
  — зчитує список із файлу або потоку.

- $write_data(X, file_name)$

  інші форми -
  $write_data(object, file_name, separator_flag), write_data(X, S), write_data(object, S, separator_flag)$
  - Здійснює виведення об'єкта $object$ (Списку, матриці, масиву Lisp або Maxima та ін) у файл file_name (або об'єкта X в потік S). Матриці виводяться по

стовпцях та рядках з використанням пробілу або іншого символу розділювача (див. $separator_flag$).

Поряд із зазначеними простими функціями, використовуються більш специфічні:

$read_lisp_array, read_maxima_array, read_hashed_array, read_nested_list$

призначені для зчитування масивів у форматі Lisp або Maxima, особливості застосування яких не розглядаються у цій книзі.

6.2.2 Функції Maxima для розрахунку описової статистики

Система Maxima містить низку функцій виконання статистичних розрахунків (описової статистики), об'єднані в пакет descriptive. Функції, що входять до складу descriptive, дозволяють виконати розрахунок дисперсії, середньоквадратичного відхилення, медіани, моди тощо. Назви функцій та короткий опис виконуваних ними дій наведено у табл. 6.1.

Таблиця 6.1. Функції пакету descriptive

Функція	Дія, що виконуються	Синтаксис виклику та примітки
$mean$	Обчислення середнього	$mean(list)$ або $mean(matrix)$
$geometric_mean$	Обчислення середнього геометричного	$geometric_mean(list)$ або $geometric_mean(matrix)$
$harmonic_mean$	Обчислення середнього гармонійного	$harmonic_mean(list)$ або $harmonic_mean(matrix)$
cor	Обчислює кореляційну матрицю	$cor(matrix)$ або $cor(matrix, logical_value)$ $logical_value$ дорівнює $true$ або $false$ (При розрахунку за коваріаційною матрицею)
$cov, cov1$	Обчислює коварійну матрицю	$cov1(matrix), cov(matrix)$
$median$	Обчислює медіану	$median(list), median(matrix)$
$std, std1$	Обчислює середньоквадратичне відхилення (корінь	аналогічно var

Функція	Дія, що виконуються	Синтаксис виклику та примітки
	квадратний зvar або$var1$)	
$var, var1$	Обчислює дисперсію випадкової величини	$var1(list), var(list)$ $var1(matrix), var(matrix),$
$central_moment$	Обчислює центральний момент порядку k	$central_moment(list, k),$ $central_moment(matrix, k)$
$noncentral_mom$	Вираховує момент порядку k	$noncentral_moment(list, k),$ $noncentral_moment(matrix, k)$
$skewness$	Обчислення асиметрії	$skewness(list), skewness(matr$
$kurtosis$	Обчислення ексцесу	$kurtosis(list), kurtosis(matrix$
$quantile$	Обчисленняp - квантиля	quantile(list, p), quantile(matrix, p)
$maxi, mini$	Вибір найбільшого та найменшого значення у вибірці відповідно	$maxi(list), maxi(matrix),$ $mini(list), mini(matrix)$
$mean_deviation,$	Сума абсолютних відхилень від середнього або медіани відповідно	Аналогічно$mean, median$
$range$	Розмах варіації вибірки	$range(list), range(matrix)$
$list_correlations$	Повертає список, що включає дві матриці - матрицю, зворотну коваріаційну, і матрицю	$list_correlations(matrix),$ $list_correlations(matrix,$ $logical_value)$де$logical_value$ дорівнює $true$ або$false$ (При розрахунку за

Функція	Дія, що виконуються	Синтаксис виклику та примітки
	приватних коефіцієнтів кореляції	коварійною матрицею)
$subsample$	Аналог функції submatrix	
$global_variances$	Повертає список, що містить різні види дисперсії	$global_variances(matrix)$

Побудова графічних ілюстрацій провадиться за допомогою функцій $scatterplot$ (безпосередня візуалізація даних), $histogram$ (будує гістограму), $barsplot$ (також будує гістограму, але за дискретними або нечисловими даними), $boxplot$ (графік Бокса-Віскера), $piechart$ (Кругова діаграма). Синтакісіс виклику та параметри функцій багато в чому аналогічні компонентам пакету draw (див. приклади нижче).

Основні загальні опції (успадковані від draw):

- terminal- пристрій, на який виводиться діаграма (можливі значення – eps і png, за замовчуванням діаграма виводиться на екран, інші варіанти);
- file_name- Ім'я файлу, в який виводити гістограму (розширення встановлюється за опцією terminal)
- title- Основний заготовок;
- xlabel, ylabel- Назви (мітки) осей,

Розглянемо приклад використання функцій пакету descriptive для статистичної обробки масивів даних. Дані беремо з файлів, що входять до складу пакету descriptive (файли biomed.data, wind.data та ін.). Перед початком роботи завантажуємо необхідні пакети descriptive та numericalio. За допомогою функції $read_matrix$ зчитується матриця, що містить 100 рядків та 5 стовпців.

```
(%i1)    load(descriptive)$ load(numericalio)$
     s:read_matrix (file_search ("wind.data"))$
     length(s);
(%o4)    100
(% i5)   mean(s); / * Розраховуємо середнє значення. При
обробці
         матриці отримуємо списоксередніх по стовпцях. *
/
```

$$(\%o5)$$

$$[9.948499999999999, 10.1607, 10.8685, 15.7166, 14.8441]$$

```
(%i6)    median(s);
```

```
(%o6)        [10.06, 9.855, 10.73, 15.48, 14.105]
(%i7)     var(s);
(%o7) [17.22190675000001, 14.98773651000001, 15.47572875,
32.17651044000001, 24.42307619000001]
(% i8)    std(s);
(%o8) [4.149928523480858, 3.871399812729242, 3.933920277534866,
5.672434260526957, 4.941970881136392]
(% i9)    mini(s);
(%o9)          [0.58, 0.5, 2.67, 5.25, 5.17]
(%i10)    maxi(s);
(%o10)         [20.25, 21.46, 20.04, 29.63, 27.63]
(%i11)    mini(%); /* При обробці списку та пошуку в ньому
мінімального
          елемента отримуємо одне значення! */
(%o11) 20.04
```

Для побудови діаграм розкиду (xy-діаграм) призначена функція $scatterplot$. Синтаксис виклику:

$scatterplot(list)$
$scatterplot(list, option_1, option_2, \dots)$
$scatterplot(matrix)$
$scatterplot(matrix, option_1, option_2, \dots)$

Дані для функції $scatterplot$ можуть представлятися вектором (списком) або матрицею. Одномірні масиви розглядаються як тимчасові ряди з рівновіддаленими точками.

Основні опції, специфічні для цієї функції:

- point_size- Розмір точки на графіці (ціле позитивне число);
- point_type- вид точки (відсутність точок - none (-1), dot (0), plus (1), multiply (2), asterisk (3), square (4), filled_square (5), circle (6), filled_circle (7), up_triangle (8), filled_up_triangle (9), down_triangle (10), filled_down_triangle (11), diamant (12), filled_diamant (13));
- color- колір точки (той самий набір кольорів, що у пакеті draw);
- grid- Наявність сітки на графіку ($true/false$).

Для побудови гістограм використовується функція $histogram$ (синтаксис виклику аналогічний $scatterplot$ та основні опції ідентичні опціям $scatterplot$). Розглянемо додаткові опції, специфічні для $histogram$:

- nclasses(за замовчуванням 10) - число класів гістограми, або список із зазначенням меж класів та їх число, або лише межі;
- frequency- Вказує масштаб шкали ординат, можливі значення: абсолютний, відносний і відсотковий (default, absolute, persent);

- htics (default, auto)- Формат проміжних поділів на гістограмі, можливі значення - auto, endpoints, intervals, або список міток;

При побудові гістограм доступні також локальні і глобальні опції пакету draw.

Для графічного представлення описової статистики служить діаграма Бокса-Уіскера ("ящик-з-усами"), яка є зручним способом наочно представити статистичні дані п'ятьма параметрами: найменше та найбільше значення вибірки, нижній, середній та верхній квартілі. На даній діаграмі можуть бути показані викиди (якщо вони є).

Для побудови діаграм Бокса-Уіскера використовується функція$boxplot$. Синтаксис виклику:$boxplot(data)$або$boxplot(data, option_1, option_2, ...)$.

Параметр$data$- Список або матриця з кількома стовпцями. Опції функції $boxplot$ ідентичні опціям$scatterplot$.

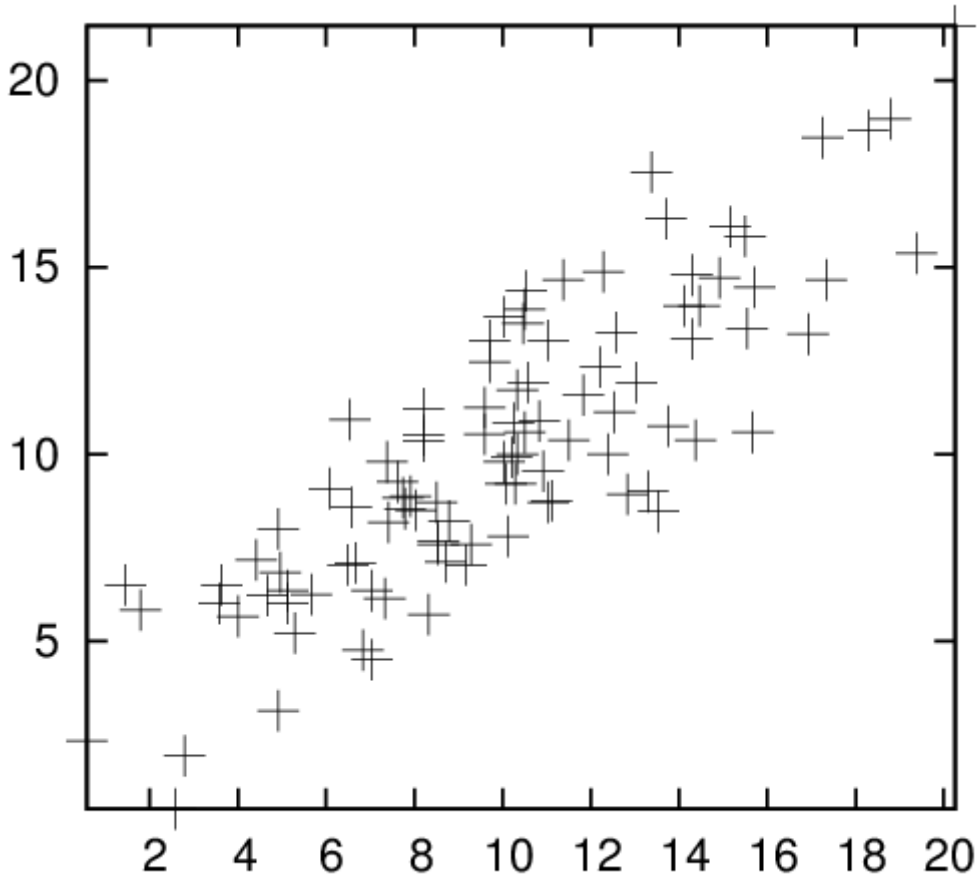
Стовпчасті діаграми (зазвичай частотні) будуються для даних, розбитих на категорії, за допомогою функції$barsplot$. Ці діаграми дозволяють графічно відобразити різницю між даними категорій.

Досить поширеним способом графічного зображення структури статистичних сукупностей є секторна діаграма, оскільки ідея цілого наочно виражається колом, що представляє всю сукупність. Відносна величина кожного значення зображується як сектора кола, площа якого відповідає вкладу цього значення суму значень. Цей вид графіків будується в Maxima функцією $piechart$.

Розглянемо приклади використання графічних утиліт пакета descriptive.

Для подальшого використання зчитуємо дані із файлу wind.data (це тестовий файл у складі пакета descriptive, містить матрицю 100x5).

```
(%i1)    load(descriptive)$ load(numericalio)$
    s:read_matrix (file_search("wind.data"))$
(% i4)   x:makelist(s[k][1],k,1,length(s))$
(% i5)   y:makelist(s[k][2],k,1,length(s))$
(%i6)    m:makelist([x[k],y[k]],k,1,100)$
(%i7)    xy:apply('matrix,m)$
```



Мал. 6.1.Точковий графік

Будуємо графік (точковий) залежності y від x (Див. рис.6.1). Результати зберігаються у файлі maxima_out.eps (ім'я файлу – за замовчуванням, він створюється в домашньому каталозі користувача). Необхідні команди:

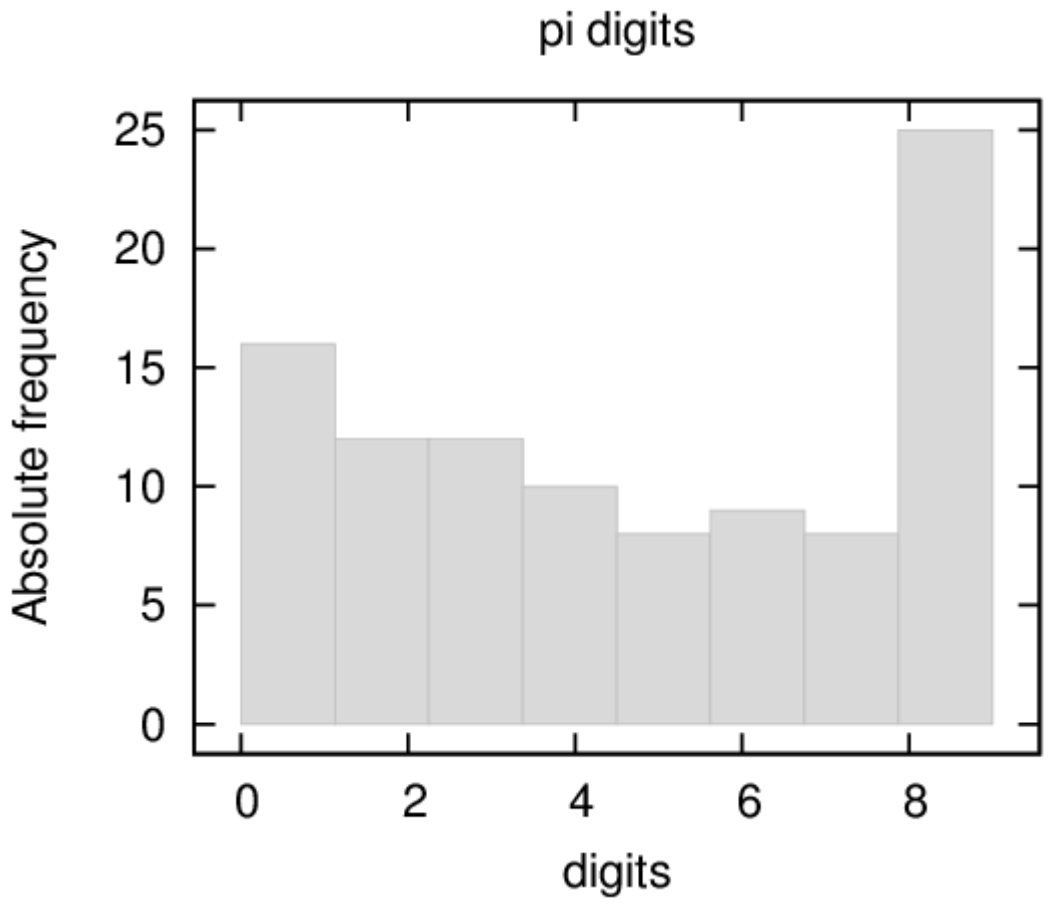
```
(% i8)   scatterplot(xy,terminal=eps);
```

Зчитуючи дані з файлу pidigits.data, будуємо гістограму частотного розподілу десяткових знаків числа π (див. рис.6.2). Результати зберігаються у файлі histogram.eps (ім'я файлу задається опцією file_name = "histogram", він створюється в домашньому каталозі користувача). Необхідні команди:

```
load (descriptive)$ s1 : read_list (file_search
("pidigits.data"))$
histogram (s1, nclasses=8, title="pi digits",
xlabel="digits",
      ylabel="Absolute frequency", fill_color=grey,
fill_density=0.6,
      terminal=eps, file_name="histogram")$
```

Наступний приклад - графік Бокса-Уіскера з анотаціями по осях (див. рис.6.3). Необхідні команди (результат зберігається у файлі boxwisker.eps, англомовні найменування замінені російськими перекладами):

```
(%i10)   boxplot(s,title="Test plot", xlabel="Seasons",
    terminal=eps,file_name="boxwisker")$
```



Мал. 6.2.Гістограма

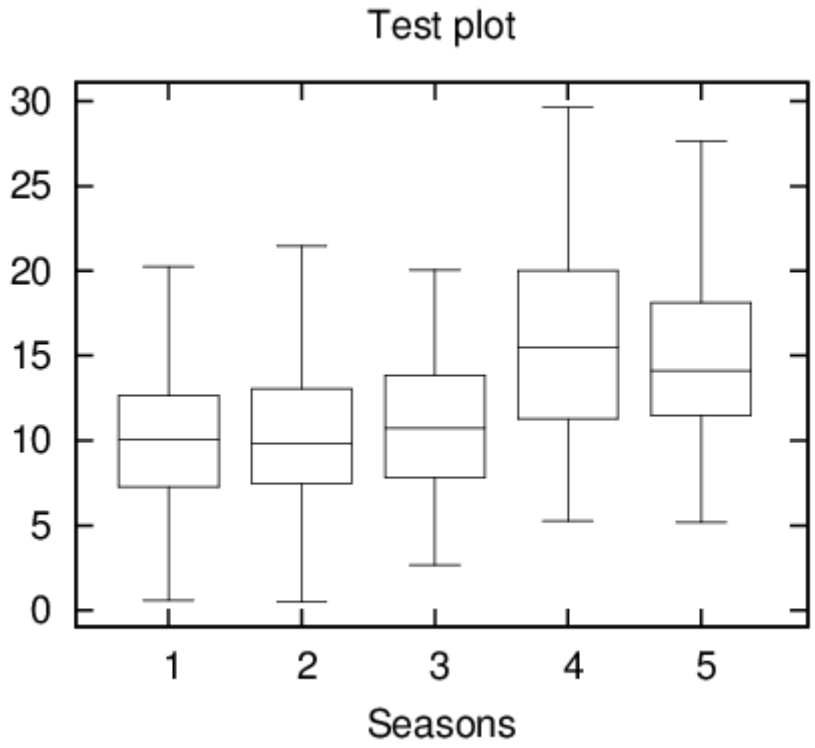
Приклад побудови стовпчастої діаграми з використанням функції$barsplot$ - На рис.6.4. Графік побудований командою (результат зберігається у файлі barsplot.eps):

```
load (descriptive)$
l1:makelist(random(8),k,1,50)$
l2:makelist(random(8),k,1,100)$
barsplot(l1,l2, box_width=1/2,
fill_density=3/4,sample_keys=["A","B"],
     bars_colors=[grey10,grey50], terminal=eps,
file_name="barsplot")$
```

Основні опції команди$barsplot$:

- box_width- Відносна ширина прямокутників (за замовчуванням 3/4, величина в межах [0, 1]);
- grouping- індикатор, що показує, як видаються множинні зразки (можливі значення clustered і stacked);
- groups_gap- натуральне число, що представляє розрив між двома сусідніми групами (відносна величина за замовчуванням 1);
- bars_colors- Список кольорів для множинних зразків (за замовчуванням [ ]);

- start_at- Вказує початок графіка по осі x (за промовчанням 0).



Мал. 6.3.Графік Бокса-Уіскера

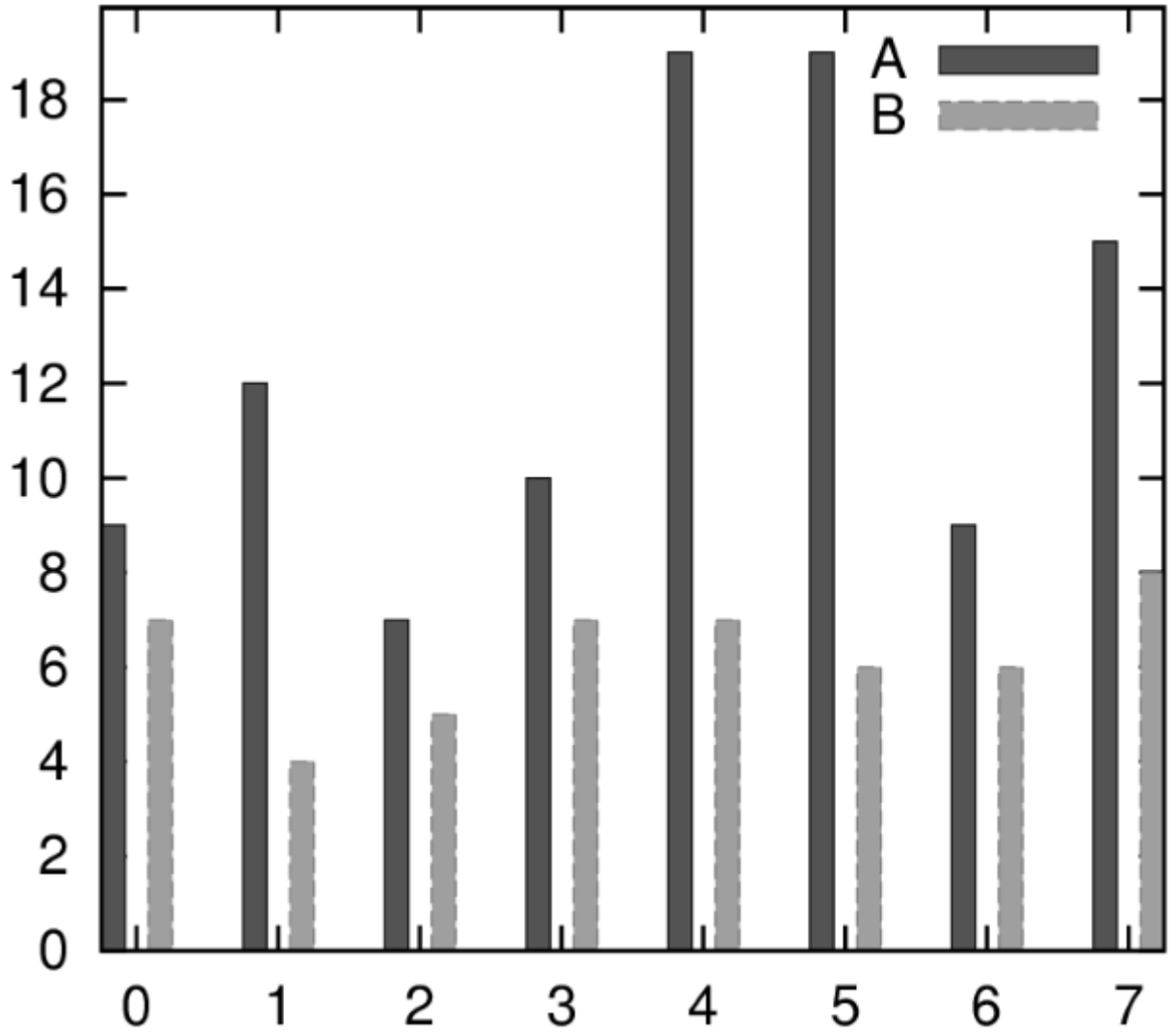
З функцією $barsplot$ можна використовувати опції пакету draw.

Для побудови кругових (секторних) діаграм використовується функція $piechart$.

Приклад використання $piechart$:

```
load (descriptive)$
s1 : read_list (file_search ("pidigits.data"))$
piechart(s1, xrange=[-1.1, 1.3], yrange=[-1.1, 1.1],
    title="Digit frequencies in pi")$
```

Кольори секторів та радіус діаграми описуються опціями sector_colors та pie_radius.



Мал. 6.4.Гістограма розподілу у групах

На жаль, базова програма виведення графіки Maxima – gnuplot – написана дуже давно, і сприймає кириличні символи тільки в кодуваннях KOI8-R або KOI8-U (в останніх версіях – і utf8). Можливим рішенням (прийнятим для побудови графіків тут) є створення файлу .gnuplot, що містить такі команди: set encoding koi8r або set encoding utf8.

Однак і в цьому випадку може виникнути потреба в редагуванні файлу maxout.gnuplot, що містить команди для створення останнього графіка.

6.2.3 Перевірка статистичних гіпотез

Для перевірки статистичних гіпотез Maxima включений пакет stats. Він дозволяє, зокрема, проводити зіставлення середніх чи дисперсій двох вибірок. Передбачено та перевірку нормальності розподілу, а також низку інших стандартних тестів. для використання stats пакет необхідно завантажити командою load("stats"); необхідні пакети descriptive та distrib завантажуються автоматично.

Функції пакету stats повертають дані типу $inference_result$. Об'єкти цього містять необхідні результати для аналізу статистичних розподілів і перевірки гіпотез.

Функція $test_mean$ дозволяє оцінити середнє значення та довірчий інтервал за вибіркою. Синтаксис виклику:

test_mean(x) або test_mean ($x, option_1, option_2, \ldots$).

Функція $test_mean$ використовує перевірку за критерієм Стьюдента. Аргумент x — список або одновимірна матриця з вибіркою, що тестується. Можливе також використання центральної граничної теореми (опція $asymptotic$). Опції $test_mean$:

- 'mean, за умовчанням 0, очікуване середнє значення;
- 'alternative, за умовчанням 'twosided, вид гіпотези, що перевіряється (можливі значення 'twosided, 'greater і 'less);
- 'dev, за умовчанням 'unknown, величина середньоквадратичного відхилення, якщо воно відоме ('unknown або позитивний вираз);
- 'conflevelза умовчанням 95/100, рівень значущості для довірчого інтервалу (величина в межах від 0 до 1);
- 'asymptotic, за умовчанням $false$, вказує який критерій використовувати (t -Критерій або центральну граничну теорему).

Результати, які повертає функція:

- 'mean_estimate- Середнє за вибіркою;
- 'conf_level- Рівень значущості, обраний користувачем;
- 'conf_interval- Оцінка довірчого інтервалу;
- 'method- Використана процедура;
- 'hypotheses- Статистичні гіпотези, що перевіряються (нульова H_0 та альтернативна H_1);
- 'statistic- Число ступенів свободи для перевірки нульової гіпотези;
- 'distribution- Оцінка розподілу розподілу вибірки;
- 'p_value- ймовірність помилкового вибору гіпотези H_1, якщо виконується H_0.

Приклади використання $test_mean$:

Виконується t -Тест з невідомою дисперсією. Нульова гіпотеза H_0: середнє дорівнює 50 проти альтернативної гіпотези. H_1: середнє менше 50; відповідно до результатів розрахунку, величина ймовірності p занадто велика, щоб відкинути H_0.

```
(%i1) load("stats")$
(%i2) data: [78,64,35,45,45,75,43,74,42,42]$
(%i3) test_mean(data,'conflevel=0.9,'alternative='less,
'mean=50);
```

$$(\%o3)\ \left(\begin{array}{c} MEAN\ TEST \\ mean\_estimate = 54.3 \\ conf\_level = 0.9 \\ conf\_interval = [-\infty, 61.513142735027 14] \\ method = Exact\ t-test.\ Unknown\ variance. \\ hypotheses = H0:\ mean = 50,\ H1:\ mean < 50 \\ statistic = .8244705235071678 \\ distribution = [student\_t, 9] \\ p\_value = .7845100411786887 \end{array}\right)$$

Наступний тест – перевірка гіпотези H_0(Середнє дорівнює 50) проти альтернативної гіпотези H_1середнє за вибіркою відмінно від 50. Відповідно до величини $p << 1$нульова гіпотеза. Цей тест застосовується для великих вибірок.

```
(%i1)    load("stats")$
(%i2)    test_mean([36,118,52,87,35,256,56,178,
57,57,89,34,25,98,35,
                    98,41,45,198,54,79,63,35,45,44,
75,42,75,45,45, 45,51,123,54,151],
'asymptotic=true,'mean=50);
```

$$(\%o2)\ \left(\begin{array}{c} MEAN\ TEST \\ mean\_estimate = 74.88571428571429 \\ conf\_level = 0.95 \\ conf\_interval = [57.72848600856193, 92.04294256286664] \\ method = Large\ sample\ z-test.\ Unknown\ variance. \\ hypotheses = H0:\ mean = 50,\ H1:\ mean \# 50 \\ statistic = 2.842831192874313 \\ distribution = [normal, 0, 1] \\ p\_value = .0044714746520022 61 \end{array}\right)$$

Функція $test_means_difference$ дозволяє перевірити, чи належать вибірки x_1і x_2до однієї генеральної сукупності.

Синтаксис виклику: $test_means_difference(x_1, x_2)$ або t $est_means_difference(x_1, x_2, option_1, option_2, ...)$.

Ця функція виконує t-тест для порівняння середніх за вибірками x_1і x_2(x_1, x_2- Списки або одновимірні матриці). Порівняння вибірок може проводитися на підставі центральної граничної теореми (для великих вибірок). Опції функції $test_means_difference$ такі ж, як і для $test_mean$, крім оцінок середньоквадратичних відхилень вибірок (якщо вони відомі) Список опцій:

- 'alternative, за умовчанням 'twosided, вид гіпотези, що перевіряється (можливі значення 'twosided, 'greater і 'less);
- 'dev1, 'dev2, за умовчанням 'unknown, величини середньоквадратичних відхилень для вибірок x_1 і x_2, якщо вони відомі ('unknown або позитивний вираз);
- 'conflevelза умовчанням 95/100, рівень значущості для довірчого інтервалу (величина в межах від 0 до 1);
- 'asymptotic, за умовчанням $false$, вказує який критерій використовувати (t -Критерій або центральну граничну теорему).

Виведення результатів $test_means_difference$ не відрізняється від висновку результатів $test_mean$.

Приклади використання $test_means_difference$: Для двох малих вибірок перевіряється гіпотеза H_0 від рівності середніх проти альтернативної гіпотези H_1: Відмінність математичних очікувань статистично значуще, тобто. вибірки належать до різних генеральних сукупностей.

```
(%i1)    load("stats")$
(%i2)    x: [20.4,62.5,61.3,44.2,11.1,23.7]$
(%i3)    y: [1.2,6.9,38.7,20.4,17.2]$
(% i4)
test_means_difference(x,y,'alternative='greater);
```

$$(\%o4)\ \left(\begin{array}{c} DIFFERENCE\ OF\ MEANS\ TEST \\ diff\_estimate = 20.31999999999999 \\ conf\_level = 0.95 \\ conf\_interval = [-.04597417812881588, \infty] \\ method = Exact\ t-test.\ Welch\ approx. \\ hypotheses = H0:\ mean1 = mean2,\ H1:\ mean1 > mean2 \\ statistic = 1.838004300728477 \\ distribution = [student\_t, 8.62758740184604] \\ p\_value = .05032746527991905 \end{array}\right)$$

Оцінка довірчого інтервалу для дисперсії вибірки проводиться за допомогою функції $test_variance$.

Синтаксис виклику:

$test_variance(x)$ або $test_variance(x, option_1, option_2, ...)$

Ця функція використовує тест χ^2. Передбачається, що розподіл вибірки x нормальне. Опції функції $test_variance$:

- 'mean, за умовчанням 'unknown, оцінка математичного очікування (середнє за вибіркою), якщо воно відоме;
- 'alternative, за умовчанням 'twosided, вид гіпотези, що перевіряється (можливі значення 'twosided, 'greater і 'less);

- 'variance, за умовчанням 1, це оцінка дисперсії вибірки для порівняння з фактичною дисперсією;
- 'conflevel, за замовчуванням 95/100, рівень значущості довірчого інтервалу (величина в межах від 0 до 1).

Основний результат, що повертається функцією - оцінка дисперсії вибірки $var_estimate$ та довірчий інтервал для неї.

Приклад: Перевірка, чи дисперсія вибірки відрізняється з невідомим математичним очікуванням від значення 200.

```
(%i1) load("stats")$
(%i2) x: [203,229,215,220,223,233,208,228,209]$
(%i3)
      test_variance(x,'alternative='greater,'variance=200
);
```

$$(\%o3)\ \left(\begin{array}{c} VARIANCE\ TEST \\ var\_estimate = 110.75 \\ conf\_level = 0.95 \\ conf\_interval = [57.13433376937479, \infty] \\ method = Variance\ Chi - square\ test.\ Unknown\ mean. \\ hypotheses = H0:\ var\ =\ 200,\ H1:\ var\ >\ 200 \\ statistic = 4.430000000000001 \\ distribution = [chi2, 8] \\ p\_value = .8163948512777688 \end{array}\right)$$

Порівняння дисперсій двох вибірок проводиться за допомогою функції $test_variance$ Синтаксис виклику $test_variance_ratio(x_1, x_2)$ або $test_variance_ratio(x_1, x_2, option_1, option_2, ...)$.

Ця функція призначена для зіставлення дисперсій двох вибірок з нормальним розподілом за критерієм Фішера (F-Тест). Аргументиx_1іx_2- Списки або одномірні матриці, що містять незалежні вибірки.

Опції функції$test_variance_ratio$:

- 'mean1, 'mean2, за умовчанням 'unknown, оцінки математичних очікувань вибірокx_1іx_2якщо вони відомі;
- 'alternative, за умовчанням 'twosided, вид гіпотези, що перевіряється (можливі значення 'twosided, 'greater і 'less);
- 'conflevel, за замовчуванням 95/100, рівень значущості довірчого інтервалу (величина в межах від 0 до 1).

Основний результат, що повертається функцією$test_variance_ratio$- Відношення дисперсій вибірок$ratio_estimate$.

Приклад: перевіряється гіпотеза про рівність дисперсій двох вибірок проти альтернативної гіпотезою у тому, що дисперсія першої більше, ніж дисперсія другий.

```
(%i1) load("stats")$
(%i2) x: [20.4,62.5,61.3,44.2,11.1,23.7]$
(%i3) y: [1.2,6.9,38.7,20.4,17.2]$
(% i4)
      test_variance_ratio(x,y,'alternative='greater);
```

$$(\%o4)\ \left(\begin{array}{c} VARIANCE\ RATIO\ TEST \\ ratio\_estimate = 2.316933391522034 \\ conf\_level = 0.95 \\ conf\_interval = [.3703504689507263, \infty] \\ method = Variance\ ratio\ F-test.\ Unknown\ means. \\ hypotheses = H0:\ var1\ =\ var2,\ H1:\ var1\ >\ var2 \\ statistic = 2.316933391522034 \\ distribution = [f,5,4] \\ p\_value = .2179269692254463 \end{array}\right)$$

За відсутності уявлень про розподіл вибірки може використовуватися непараметричний тест порівняння середніх. Оцінка медіани безперервної вибірки проводиться за допомогою функції$test_sign$. Синтаксис виклику

$test_sign(x)$ або

$test_sign(x, option_1, option_2, ...)$.

Функція$test_sign$допускає дві опції:$alternative$(аналогічно$test_mean$) та$median$(за замовчуванням 0 або оцінка значення медіани для перевірки статистичної значущості).

Результати, які повертає функція:

- 'med_estimate:медіана вибірки;
- 'method:використана процедура;
- 'hypotheses:статистичні гіпотези, що перевіряються (нульоваH_0та альтернативнаH_1);
- 'statistic:число ступенів свободи для перевірки нульової гіпотези;
- 'distribution:оцінка розподілу розподілу вибірки;
- 'p_value:ймовірність помилкового вибору гіпотезиH_1, якщо виконуєтьсяH_0.

Приклад: перевірка гіпотезиH_0про рівність медіани вибірки 6 проти альтернативної гіпотезиH_1: медіана більше 6

```
(%i1)    load("stats")$
(%i2)    x: [2,0.1,7,1.8,4,2.3,5.6,7.4,5.1,6.1,6]$
(%i3)    test_sign(x,'median=6,'alternative='greater);
```

$$(\%o3)\begin{pmatrix} SIGN\ TEST \\ med\_estimate = 5.1 \\ method = Non\ parametric\ sign\ test. \\ hypotheses = H0:\ median\ =\ 6,\ H1:\ median\ >\ 6 \\ statistic = 7 \\ distribution = [binomial, 10, 0.5] \\ p\_value = .054687499999999 \end{pmatrix}$$

Аналогічна функція - $test_signed_rank(x)$

(або із зазначенням опцій $test_signed_rank(x, option_1, option_2, ...)$),

яка використовує тест правила знаків Вілкоксона для оцінки гіпотези про медіану безперервної вибірки. Опції та результати функції $test_signed_rank$ такі ж, як і для функції $test_sign$.

Приклад: перевірка гіпотези H_0: медіана дорівнює 15 проти альтернативної гіпотези. H_1: медіана більше 15

```
(%i1) load("stats")$
(%i2) x: [17.1,15.9,13.7,13.4,15.5,17.6]$
(%i3) test_signed_rank(x,median=15,alternative=greater);
```

$$(\%o3)\begin{pmatrix} SIGNED\ RANK\ TEST \\ med\_estimate = 15.7 \\ method = Exacttest \\ hypotheses = H0:\ med\ =\ 15,\ H1:\ med\ >\ 15 \\ statistic = 14 \\ distribution = [signed\_rank, 6] \\ p\_value = 0.28125 \end{pmatrix}$$

Непараметричне порівняння медіан двох вибірок реалізовано в одній функції $test_rank_sum$. У цій функції використовується тест Вілкоксона-Манна-Уітні. U-критерій Манна-Уітні - непараметричний метод перевірки гіпотез, що часто використовується як альтернатива t-тесту Стьюдента. Зазвичай цей тест використовується для порівняння медіан двох розподілів. x_1 і x_2, які не є нормальними (відсутність нормальності не дозволяє застосувати t-Тест).

Синтаксис виклику:

$test_rank_sum(x_1, x_2)$ або $test_rank_sum(x_1, x_2, option_1)$.

Функція допускає лише одну опцію: $alternative$ (аналогічно $test_means_difference$).

Результати, які повертає функція:

- 'method:використана процедура;

- 'hypotheses:статистичні гіпотези, що перевіряються (нульова H_0 та альтернативна H_1);
- 'statistic:число ступенів свободи для перевірки нульової гіпотези;
- 'distribution:оцінка розподілу розподілу вибірки;
- 'p_value:ймовірність помилкового вибору гіпотези H_1, якщо виконується H_0.
-

Приклад: перевірка, чи однакові медіани вибірок x_1 і x_2.

```
(%i1) load("stats")$
(%i2) x:[12,15,17,38,42,10,23,35,28]$
(%i3) y:[21,18,25,14,52,65,40,43]$
(% i4)      test_rank_sum(x, y);
```

$$(\%o4)\ \left(\begin{array}{c} RANK\ SUM\ TEST \\ method = Exact\ test \\ hypotheses = H0:\ med1 = med2,\ H1:\ med1\ \#\ med2 \\ statistic = 22 \\ distribution = [rank\_sum, 9, 8] \\ p\_value = .1995886466474702 \end{array}\right)$$

Для вибірок більшого обсягу розподіл вибірок є приблизно нормальним. Порівнюємо гіпотези H_0:

медіана 1 = медіана 2 та H_1: медіана 1 < медіана 2.

```
(%i1) load("stats")$
(%i2) x: [39,42,35,13,10,23,15,20,17,27]$
(%i3) y:
[20,52,66,19,41,32,44,25,14,39,43,35,19,56,27,15]$
(% i4)      test_rank_sum(x,y,'alternative='less);
```

$$(\%o4)\ \left(\begin{array}{c} RANK\ SUM\ TEST \\ method = Asymptotic\ test.\ Ties \\ hypotheses = H0:\ med1 = med2,\ H1:\ med1 < med2 \\ statistic = 48.5 \\ distribution = [normal, 79.5, 18.95419580097078] \\ p\_value = .050969856665984 41 \end{array}\right)$$

Перевірка нормальності розподілу здійснюється функцією $test_normality(x)$. У цій функції реалізовано тест Шапіро-Вілка. Вибірка x (Список або одномірна матриця) повинна бути розміром не менше 2, але не більше 5000 елементів (інакше видається повідомлення про помилку). Функція повертає два значення: statistic – величина W-статистики та величина ймовірності p (якщо p більше прийнятого рівня значимості, нульова гіпотеза про нормальність розподілу вибірки x не відкидається). Статистика W характеризує близькість вибіркового розподілу до нормального (чим ближче W до 1, тим менша ймовірність помилково прийняти гіпотезу про нормальність розподілу).

Приклад: перевірка гіпотези про нормальний розподіл генеральної сукупності за заданою вибіркою.

```
(%i1) load("stats")$
(%i2) x:[12,15,17,38,42,10,23,35,28]$
(%i3) test_normality(x);
```

$$(\%o3)\ \begin{pmatrix} SHAPIRO\ -\ WILK\ TEST \\ statistic = .9251055695162439 \\ p\_value = .4361763918860427 \end{pmatrix}$$

6.2.4 Розрахунок коефіцієнтів лінійної регресії

Коефіцієнти та оцінка статистичної значущості для найпростішої лінійної регресії можуть оцінюватися за допомогою функції $simple_linear_regression$ із пакету stats. Функція обчислює коефіцієнти та параметри лінійної регресії $y = a_0 + a_1 \cdot x$ (Тобто тільки найпростішою).

Синтаксис виклику:

$simple_linear_regression(x)$ або

$simple_linear_regression(x, option_1)$.

Опції функції $simple_linear_regression : conflevel$ (Рівень значимості, зазвичай 0.95, див. вище) і $regressor$ (за замовчуванням x, ім'я незалежної змінної). Ця функція виводить велику кількість статистичних параметрів:

- 'model:отримане рівняння регресії;
- 'means:середня;
- 'Variances:дисперсії обох змінних;
- 'correlation:коефіцієнт кореляції;
- 'adc:коефіцієнт детермінації;
- 'a_estimation:оцінка параметра a ;
- 'a_conf_int:довірчий інтервал для a ;
- $'b_estimation$:оцінка параметра b;
- 'b_conf_int:довірчий інтервал для b ;
- 'hypotheses:нульова та альтернативна гіпотеза щодо параметра b ;
- 'statistic:статистичні характеристики вибірки, використані для перевірки нульової гіпотези;
- 'distribution:розподіл вибірки;
- 'p_value:величина ймовірності для перевірки гіпотези про статистичну значущість b ;
- 'v_estimation:оцінка залишкової дисперсії;
- 'v_conf_int:довірчий інтервал для залишкової дисперсії
- 'cond_mean_conf_int:довірчий інтервал для середнього;
- 'new_pred_conf_int:довірчий інтервал для нового передбачення;

- Residuals:список, що містить залишки.

За замовчуванням на консоль виводяться лише параметри 1, 4, 14, 9, 10, 11, 12 та 13 у цьому списку. Інші параметри приховані, але доступ до них забезпечується за допомогою функцій $items_inference$ або $take_inference$.

Задаємо вихідними даними

```
(% i9)     s:[[125,140.7], [130,155.1], [135,160.3],
                          [140,167.2], [145,169.8]]$
```

Обчислюємо коефіцієнти та інші параметри регресії

```
(%i10)     z:simple_linear_regression(s,conflevel=0.99);
```

(%o10)

$$\left(\begin{array}{c} SIMPLE\ LINEAR\ REGRESSION \\ model = 1.405999999999986\ x - 31.18999999999804 \\ correlation = 0.96116852552552 \\ v\_estimation = 13.57966666666665 \\ b\_conf\_int = [0.044696336625253, 2.767303663374718] \\ hypotheses = H0 : b = 0,\ H1 : b\#0 \\ statistic = 6.032686683658114 \\ distribution = [student\_t, 3] \\ p\_value = 0.0038059549413203 \end{array}\right)$$

```
(%i11)     z:simple_linear_regression(s,conflevel=0.95);
```

(%o11)

$$\left(\begin{array}{c} SIMPLE\ LINEAR\ REGRESSION \\ model = 1.405999999999986\ x - 31.18999999999804 \\ correlation = 0.96116852552552 \\ v\_estimation = 13.57966666666665 \\ b\_conf\_int = [0.66428743645021, 2.147712563549759] \\ hypotheses = H0 : b = 0,\ H1 : b\#0 \\ statistic = 6.032686683658114 \\ distribution = [student\_t, 3] \\ p\_value = 0.0038059549413203 \end{array}\right)$$

Деякі додаткові параметри:

```
(% i5)     take_inference(model,z), x=133;
```

(%o5) 155.808

```
(%i6) take_inference(means,z);
```

(%o6) [135.0, 158.62]

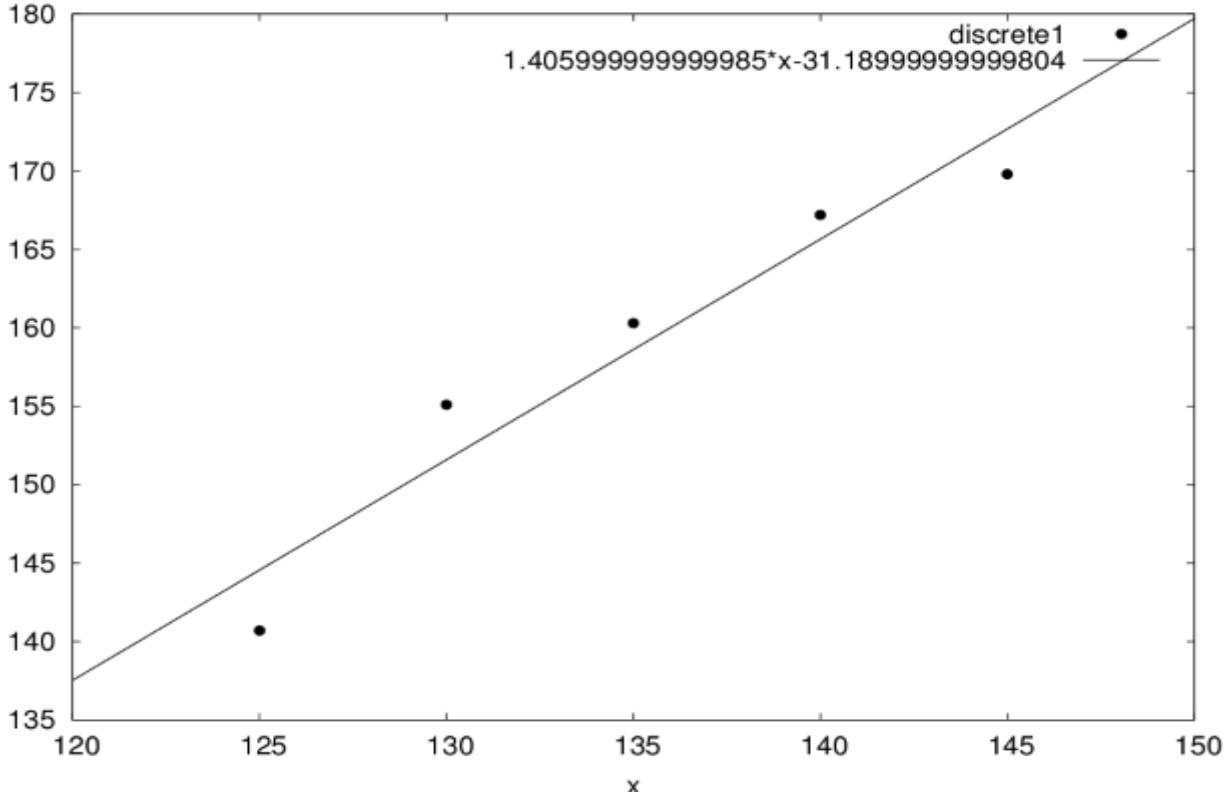
```
(%i7) take_inference(new_pred_conf_int,z), x=133;
```

(%o7) [132.0728595995113, 179.5431404004887]

Графічна ілюстрація побудованої лінійної залежності див. на рис.6.5. Використана команда:

```
(%i11)     plot2d([[discrete,s],
take_inference(model,z)],
```

```
     [x, 120, 150], [style,[points],[lines]],
[gnuplot_term,ps],
     [gnuplot_out_file, "regress.eps"])$
```



Мал. 6.5.Проста лінійна регресія

6.2.5 Використання методу найменших квадратів

Пакет Maxima включає потужний модуль для лінійного та нелінійного оцінювання параметрів різних моделей з використанням методу найменших квадратів – пакет lsqares.

Основна функція пакету lsqares – це функція$lsquares_estimates$.

Синтаксис виклику:

$lsquares_estimates(D, x, e, a)$ або

$lsquares_estimates(D, x, e, a, initial = L, tol = t)$

Функція призначена для оцінки параметрів, що найкраще відповідають рівняннюe у зміннихx іa з набору даних D, які визначаються методом методом найменших квадратів. Функція$lsquares_estimates$ спочатку намагається знайти точне рішення, і якщо це не вдається, шукає приблизне рішення. Значення, що повертається — список виду$[a = ..., b = ..., c = ...]$. Елементи списку забезпечують мінімум середньоквадратичної помилки. ДаніD мають бути матрицею. Кожен ряд - один запис або один випадок, кожен стовпець відповідає значенням певної змінної.

Список зміннихx дає назву для кожного стовпцяD (навіть для стовпців, які не входять до аналізу). Список параметрів містить назви параметрів, котрим перебувають оцінки. Рівнянняe є виразом або рівнянням у зміннихx іa ; якщо e записано не у формі рівняння, його розглядають як рівняння$e = 0$. Якщо

деяке точне рішення може бути знайдено за допомогою $solve$, дані D можуть містити й нечислові значення.

Додаткові аргументи $lsquares_estimates$ визначені як рівняння та передаються "дослівно" функції $lbfgs$, яка використовується, щоб знайти оцінки чисельним методом, коли точний результат не знайдено. Однак, якщо жодного точного рішення не знайдено, у кожного елемента D має бути числове значення, у тому числі константи (такі як $\%pi$ і $\%e$) чи числові літерали (цілі числа, раціональні, з плаваючою точкою, і з плаваючою точкою підвищеної точності). Чисельні розрахунки виконуються зі звичайною арифметикою з точкою, що плаває, таким чином всі інші види чисел перетворюються до значень з плаваючою точкою. Для роботи з $lsquares_estimates$ необхідно завантажити цю функцію командою load($lsquares$).

Приклад (точне рішення):

```
(%i1) load (lsquares)$
(%i2) M:matrix([1,1,1],[3/2,1,2],[9/4,2,1],
                                        [3,2,2],[2,2,1]);
```

$$(\%o2)\quad \begin{pmatrix} 1 & 1 & 1 \\ \frac{3}{2} & 1 & 2 \\ \frac{9}{4} & 2 & 1 \\ 3 & 2 & 2 \\ 2 & 2 & 1 \end{pmatrix}$$

```
(%i3) lsquares_estimates(M,[z,x,y],(z+D)^2=A*x+B*y+C,
[A,B,C,D]);
```

$$(\%o3)\ [[A=-\frac{59}{16}, B=-\frac{27}{16}, C=\frac{10921}{1024}, D=-\frac{107}{32}]]$$

Інший приклад (точне рішення відсутня, знаходиться наближене):

```
(%i1) load (lsquares)$
      M: matrix ([1, 1], [2, 7/4], [3, 11/4], [4, 13/4]);
```

$$(\%o2)\quad \begin{pmatrix} 1 & 1 \\ 2 & \frac{7}{4} \\ 3 & \frac{11}{4} \\ 4 & \frac{13}{4} \end{pmatrix}$$

```
(%i3) lsquares_estimates(M,[x,y],y=a*x^b+c,[a,b,c],
initial=[3,3,3], iprint=[-1,0] );
```

$$(\%o3)\ [[a=1.375751433061394, b=.7148891534417651,$$
$$c=-.40209089100629 51]]$$

Для розрахунку нев'язок для рівняння e при підстановці в нього даних, що містяться в матриці D, можна використовувати функцію $lsquares_residuals(D,x,e,a)$ (Сенс параметрів той же, що і для функції $lsquares_estimates$).

Приклад використання функції $lsquares_estimates$ і $lsquares_residuals$ (Ті ж дані, що використані для розрахунку параметрів простої лінійної регресії):

```
(%i1) load (lsquares)$
(%i2) s:[[125,140.7],[130,155.1],[135,160.3],
                                    [140,167.2], [145,169.8]];
```

$$(\%o2)\ [[125, 140.7], [130, 155.1], [135, 160.3], [140, 167.2], [145, 169.8]]$$

```
(%i3) D: apply (matrix, s);
```

$$(\%o3)\ \begin{pmatrix} 125 & 140.7 \\ 130 & 155.1 \\ 135 & 160.3 \\ 140 & 167.2 \\ 145 & 169.8 \end{pmatrix}$$

```
(% i4)      a: lsquares_estimates(D,[y,x],y = A+B*x,
[A,B]);
```

$$(\%o4)\qquad [[A = \frac{8231525}{267474}, B = \frac{87875}{133737}]]$$

```
(% i5)      float(%);
```

$$(\%o5)$$

$$[[A = 30.77504729431646, B = 0.65707321085414]]$$

```
(%i6) lsquares_residuals (D, [y, x], y = A + B * x, first
(a));
```

$$(\%o6)\ [1.774751938506171, -2.687102297793416, -1.103882994234965, -0.63768814912851, 2.653921502650718]$$

Інші функції, що входять до складу пакету lsquares, за синтаксисом використання та ідеї реалізації аналогічні наведеним (див. документацію розробника).

6.3 Моделювання динамічних систем

Багато моделей, засновані на нелінійних диференціальних рівняннях, демонструють дивовижні властивості, причому рішення більшості їх можна отримати лише чисельно.

Моделі, що ґрунтуються на завданнях Коші для ОДУ, часто називають динамічними системами, підкреслюючи, що, як правило, вони містять похідні за часом t та описують динаміку деяких параметрів. Проблеми, пов'язані з динамічними системами, насправді дуже різноманітні і часто не зводяться до простого інтегрування ОДУ.

6.3.1 Моделювання системи хімічних реакцій

Розглянемо інший приклад. Досліджуємо систему із трьох диференціальних рівнянь, що описують модель хімічної кінетики:

$$A \to B$$
$$B \to C$$

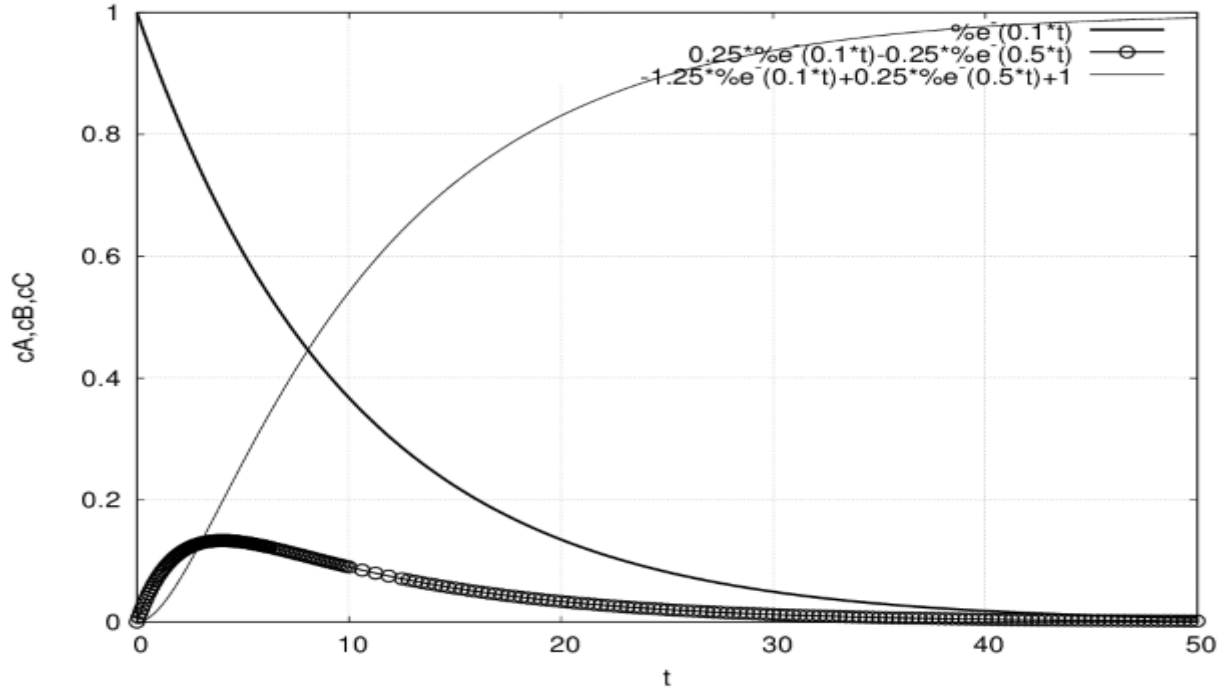
Система відповідних диференціальних рівнянь

$$\frac{dc_A}{dt} = -k_1 c_A$$
$$\frac{dc_B}{dt} = k_1 c_A - k_2 c_B$$
$$\frac{dc_C}{dt} = k_2 c_B$$

Початкові умови:

$$c_A = 1, c_B = 0, c_C = 0$$

Результати рішення наведено на рис.6.6.



Мал. 6.6.Кінетика хімічних реакцій

```
(%i1) eq1:'diff(ca(t),t)=-k1*ca(t);
      eq2:'diff(cb(t),t)=k1*ca(t)-k2*cb(t);
      eq3:'diff(cc(t), t) = k2 * cb (t);
```

$$(\%o1)\ \frac{d}{dt}\,\mathrm{ca}\,(t) = -k1\,\mathrm{ca}\,(t)$$
$$(\%o2)\ \frac{d}{dt}\,\mathrm{cb}\,(t) = k1\,\mathrm{ca}\,(t) - k2\,\mathrm{cb}\,(t)$$
$$(\%o3)\ \frac{d}{dt}\,\mathrm{cc}\,(t) = k2\,\mathrm{cb}\,(t)$$

```
(% i4)      atvalue(ca(t),t=0,1);
```

$$(\%o4)\ 1$$

```
(% i5)      atvalue(cb(t),t=0,0);
      atvalue(cc(t),t=0,0);
```

$$(\%o5)\ 0$$
$$(\%o6)\ 0$$

```
(%i7) sol:desolve([eq1,eq2,eq3],[ca(t),cb(t),cc(t)]);
```

$$(\%o7)\ [\mathrm{ca}\,(t) = e^{-k1\,t}, \mathrm{cb}\,(t) = \frac{k1\,e^{-k1\,t}}{k2-k1} - \frac{k1\,e^{-k2\,t}}{k2-k1},$$
$$\mathrm{cc}\,(t) = \frac{k1\,e^{-k2\,t}}{k2-k1} - \frac{k2\,e^{-k1\,t}}{k2-k1} + 1]$$

```
(% i8)      ratsimp(sol);
```

$$(\%o8)\ [\mathrm{ca}\,(t) = e^{-k1\,t}, \mathrm{cb}\,(t) = \frac{\left(k1\,e^{k2\,t} - k1\,e^{k1\,t}\right) e^{-k2\,t-k1\,t}}{k2-k1},$$
$$\mathrm{cc}\,(t) = \frac{\left(\left((k2-k1)\,e^{k1\,t} - k2\right) e^{k2\,t} + k1\,e^{k1\,t}\right) e^{-k2\,t-k1\,t}}{k2-k1}]$$

```
(% i9)      k1: 0.1; k2:0.5;ev((sol));
```

$$(\%o9)\ 0.1$$

$$(\%o10)\ 0.5$$

$$(\%o11)\ [\mathrm{ca}\,(t) = e^{-0.1\,t}, \mathrm{cb}\,(t) = 0.25\,e^{-0.1\,t} - 0.25\,e^{-0.5\,t}, \mathrm{cc}\,(t) =$$
$$-1.25\,e^{-0.1\,t} + 0.25\,e^{-0.5\,t} + 1]$$

```
(%i12)      plot2d([%e^(-0.1*t),0.25*%e^(-0.1*t)-
                                          0.25*%e^(-0.5*t),
-1.25*%e^(-0.1*t)+0.25*%e^(-0.5*t)+1],[t,0,50],
      [gnuplot_preamble, "set grid"],
      [gnuplot_term,"png size 500,500"],
      [gnuplot_out_file,"chem.png"]);
```

Дещо складніше завдання — моделювання кінетики паралельно-послідовних реакцій, що протікають за схемою:

$$A \to B$$
$$B + C \to A$$
$$B \to C$$

Залежно від констант швидкості хімічних реакцій, ця система може бути досить жорсткою.

6.3.1.1 Приклад командного файлу для вирішення жорсткої системи

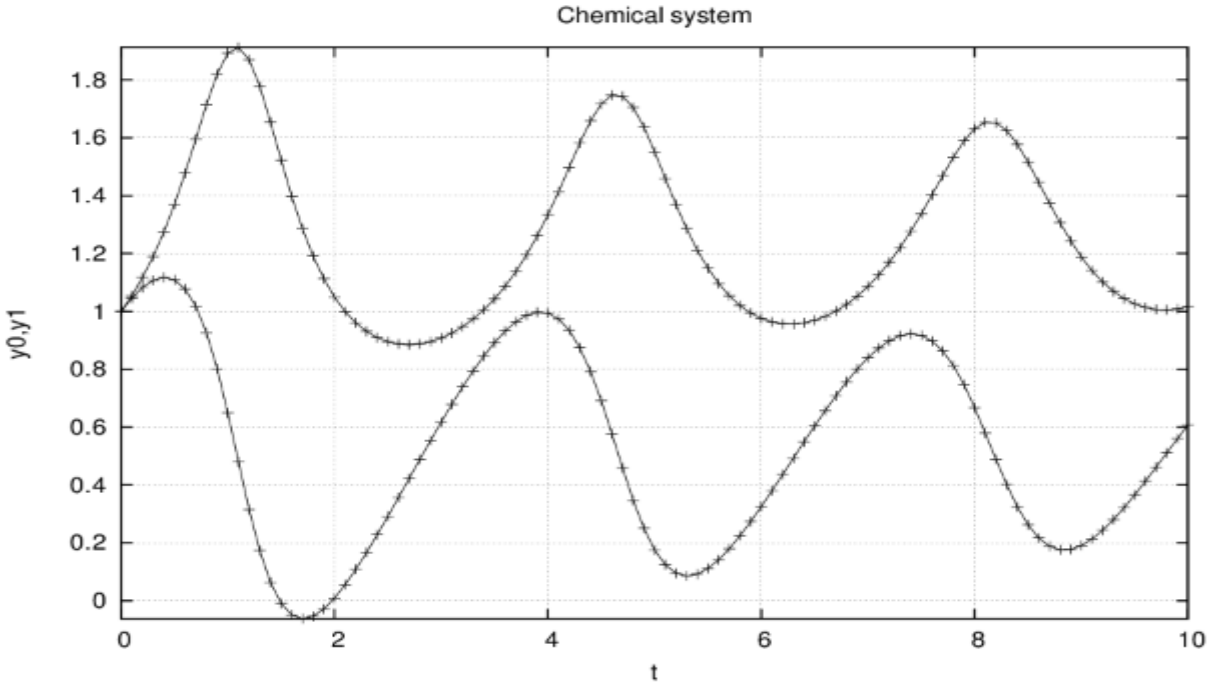
ОДУ в Maxima (дана система нелінійна, тому використовуємо метод Рунге-Кутта, проте розрахунки ускладнюються жорсткістю системи):

```
load("dynamics");
load("draw");
k1: 0.1; k2:100; k3:10;
eq1:-k1*ca+k3*cb*cc;
eq2:k1*ca-k3*cb*cc-k2*cb;
eq3:k2*cb;
t_range: [t,0,100,0.01];
sol: rk([eq1,eq2,eq3],[ca,cb,cc],[1,0,0],t_range)$
len:length(sol);
t:makelist(sol[k][1],k,1,len)$
ca:makelist(sol[k][2],k,1,len)$
cb:makelist(sol[k][3],k,1,len)$
cc:makelist(sol[k][4],k,1,len)$
draw2d(title="Chemical system",xlabel="ca",ylabel="cb",
```

```
grid=true,points_joined=true,points(t,ca),
points(t,cb),points(t,cc),terminal=eps);
```

Ця система досить важко вирішується за допомогою функціїrk. Збільшення констант до$k2 = 1000$і$k3 = 100$робить завдання практично нерозв'язним засобами dynamics пакета.

Найпростішим класичним прикладом існування автоколивань у системі хімічних реакцій є тримолекулярна модель "Брюсселятор", запропонована у Брюсселі Пригожиним та Лефевром (1965). Основною метою щодо цієї моделі було встановлення якісних типів поведінки, сумісних з фундаментальними законами хімічної та біологічної кінетики. У цьому сенсі брюселятор відіграє роль базової моделі, таку як гармонійний осцилятор у фізиці, або моделі Вольтерра в динаміці популяцій.



Мал. 6.7.Зміна концентрацій при моделюванні автоколивальної хімічної реакції (брюсселятора)

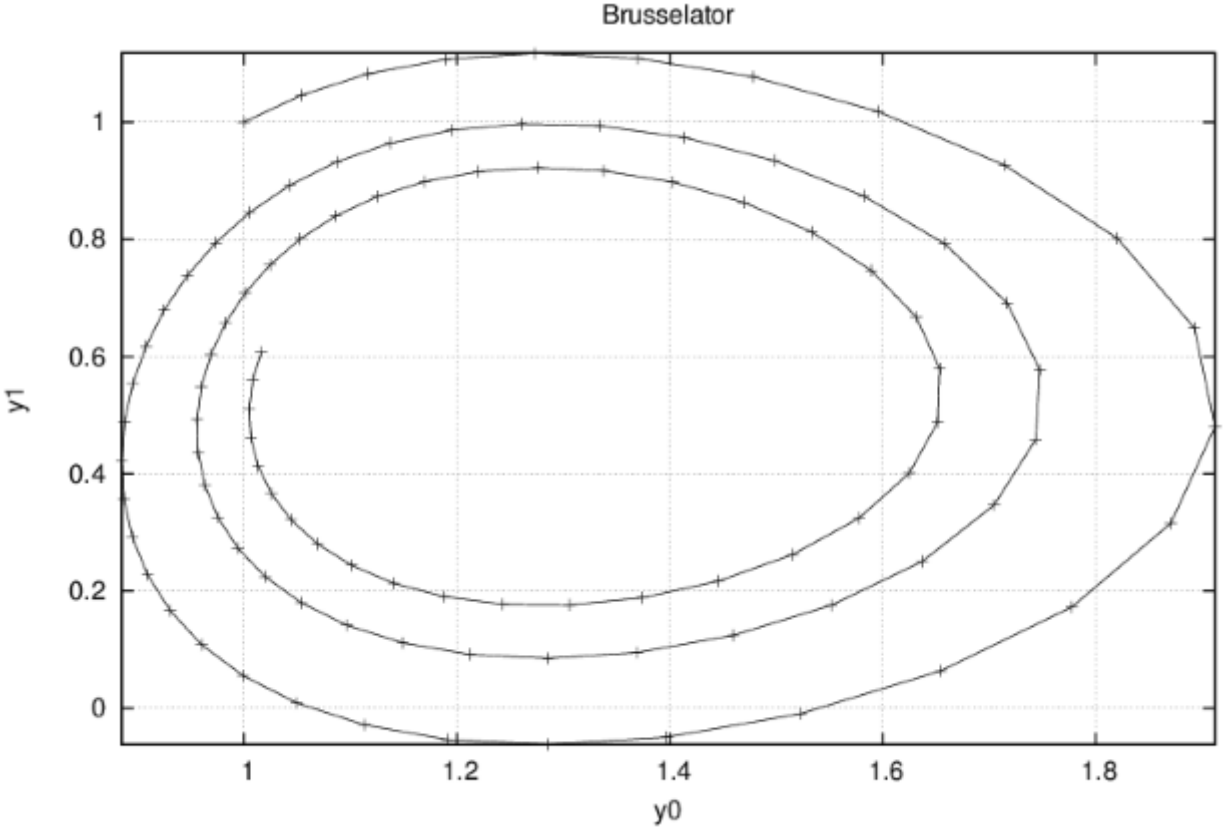
Тут брюселятор сприймається як приклад автоколивальної системи.

Опис моделі брюселятора Maxima наведено в наступному командному файлі:

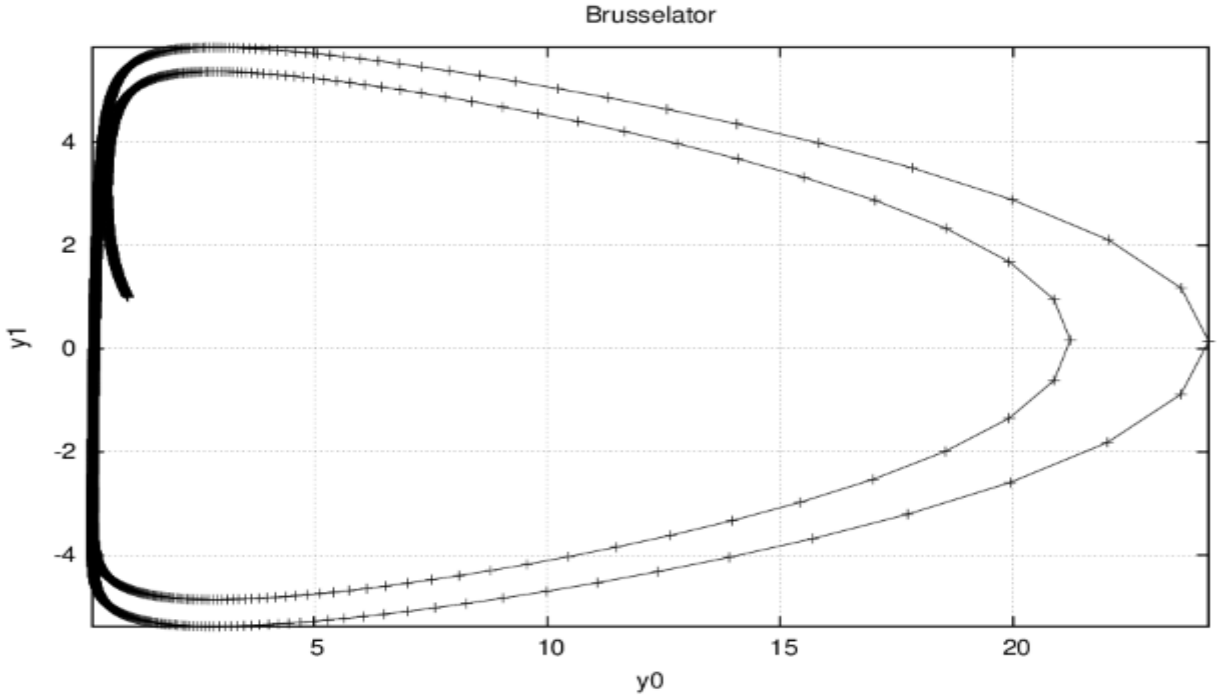
```
load("dynamics");
load("draw");
B: 0.5;
eq1:-(B+1)*y0+y0^2*y1+1;
eq2:B*y0-y0^2+1;
t_range: [t,0,10,0.1];
sol: rk([eq1,eq2],[y0,y1],[1,1],t_range)$
len:length(sol);
t:makelist(sol[k][1],k,1,len)$
y0:makelist(sol[k][2],k,1,len)$
y1:makelist(sol[k][3],k,1,len)$
```

```
draw2d(title="Brusselator",xlabel="t",ylabel="y0,y1",
grid=true,points_joined=true,
points(t,y0),points(t,y1),terminal=eps);
```

Графічна ілюстрація (автоколивальний режим у системі) — на рис.67., а фазові портрети — на рис.6.8 та рис.6.9.



Мал. 6.8.Фазовий портрет для брюселятора (=0.5)



Мал. 6.9.Фазовий портрет для брюселятора (=2.5)

При проведенні розрахунків слід звернути увагу на жорсткість системи ОДУ, що описує брюсселятор, зокрема при $B = 2.5$ для побудови наведеної ілюстрації необхідно було зменшити крок часу до 0.002. При запуску командного файлу, що містить команди завантаження пакетів, рекомендується перезапустити Maxima.

6.3.2 Фазові портрети динамічних систем

Для вивчення динамічних систем центральним моментом є аналіз фазових портретів, тобто рішень, що виходять при виборі різних початкових умов.

Рішення ОДУ часто зручніше зображати не у вигляді графіка $y0(t), y1(t), \cdots$, а фазовому просторі, з кожної з осей якого відкладаються значення кожної зі знайдених функцій. За такої побудови графіка аргумент t буде присутня на ньому лише параметрично.

Як правило, розв'язання задач Коші для ОДУ та їх систем — завдання добре розроблене і з обчислювальної точки зору досить просте. Насправді частіше зустрічаються інші, складніші завдання, зокрема, дослідження поведінки динамічної системи залежно від початкових умов. У цьому найчастіше буває необхідним вивчити лише асимптотическое рішення ОДУ, тобто. $y(t \to \infty)$, Звані атрактором. Дуже наочно можна візуалізувати таку інформацію на фазовій площині, багато в чому завдяки тому, що існує лише кілька типів атракторів, і для них можна побудувати чітку класифікацію.

З одного боку, кожне рішення виходитиме з точки, координати якої є початковими умовами, але, виявляється, для більшості ОДУ цілі сімейства траєкторій будуть закінчуватися в тих самих атракторах (стаціонарних точках або граничних циклах). Безліч рішень, обчислене для різних початкових умов, утворює фазовий портрет динамічної системи. З обчислювальної точки зору завдання дослідження фазового портрета часто зводиться до звичайного сканування сімейств рішень ОДУ за різних початкових умов.

Подальше ускладнення завдань аналізу фазових портретів пов'язані з їх залежністю від параметрів, які входять у систему ОДУ. Зокрема, при плавній зміні параметра моделі може змінюватися розташування атракторів на фазовій площині, можуть виникати нові атрактори і припиняти своє існування старі. У першому випадку, за відсутності особливостей, відбуватиметься просте переміщення атракторів по фазовій площині (без зміни їх типів та кількості), а в другому — фазовий портрет динамічної системи докорінно перебудовуватиметься. Критичне поєднання параметрів, у яких фазовий портрет системи якісно змінюється, називається теорії динамічних систем точкою біфуркації.

Розглянемо кілька найвідоміших класичних прикладів динамічних систем, маючи на увазі. Це моделі динаміки популяцій (Лотка-Вольтерри), генератора автоколивань (Ван дер Поля), турбулентної конвекції (Лоренця) та хімічної реакції з дифузією (Пригожина). Для вивчення динамічних систем розроблено

спеціальну теорію, центральним моментом якої є аналіз фазових портретів, тобто рішень, що виходять при виборі всіляких початкових умов.

6.3.3 Модель динаміки популяцій

Модель взаємодії "хижак-жертва" незалежно запропонували у 1925-1927 роках. Лотка та Вольтерра. Два диференціальні рівняння моделюють тимчасову динаміку чисельності двох біологічних популяцій жертв x та хижаків y. Передбачається, що жертви розмножуються з постійною швидкістю, які чисельність убуває внаслідок поїдання хижаками. Хижаки розмножуються зі швидкістю, пропорційною кількості їжі, і вмирають природним чином.

Модель була створена для біологічних систем, але з певними коректурами застосовна до конкуренції фірм, будівництва фінансових пірамід, зростання народонаселення, екологічної проблематики та ін.

Ця модель Вольтерра-Лотка з логістичною поправкою описується системою рівнянь

$$\frac{d}{dt} y(t) = y(t)\,(a - b\,z(t)) - \alpha\, y(t)^2$$
$$\frac{d}{dt} z(t) = (-c + d\,y(t))\; z(t) - \alpha\, z(t)^2$$

з умовами заданої чисельності "жертв" та "хижаків" у початковий момент $t = 0$.

Вирішуючи це завдання при різних значеннях, отримуємо різні фазові портрети (звичайний коливальний процес і поступова загибель популяцій). Результати наведено на рис.6.10, рис.6.11 та рис.6.12.

```
(%i1) a:4$ b:2.5$ c:2$ d:1$alpha = 0 $
      eq1: 'diff (y (t), t) = (ab * z (t)) * y (t) -alpha
* y (t) ^ 2;
      eq2: 'diff (z (t), t) = (-c+d*y(t))*z(t)-
alpha*y(t)^2;
      atvalue(y(t),t=0,3); atvalue(z(t),t=0,1);
```

$$(\%o6)\ \frac{d}{dt}\,\mathrm{y}(t) = \mathrm{y}(t)\,(4 - 2.5\,\mathrm{z}(t)) - \alpha\,\mathrm{y}(t)^2$$
$$(\%o7)\ \frac{d}{dt}\,\mathrm{z}(t) = (\mathrm{y}(t) - 2)\;\mathrm{z}(t) - \alpha\,\mathrm{y}(t)^2$$
$$(\%o8)\ 3$$
$$(\%o9)\ 1$$

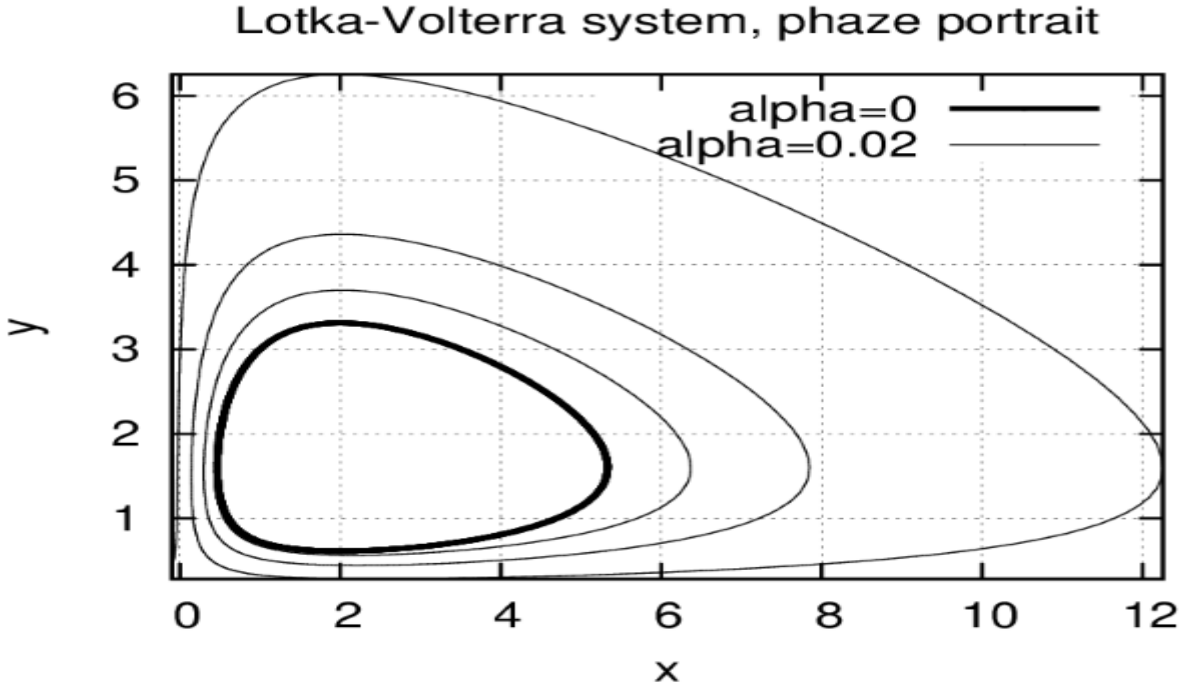
```
(%i10)     desolve([eq1,eq2],[y(t),z(t)]);
rat: replaced -2.5 by -5/2 = -2.5
rat: replaced -2.5 by -5/2 = -2.5
rat: replaced -2.5 by -5/2 = -2.5
rat: replaced 2.5 by 5/2 = 2.5
```

$$(\%o10)$$
$$[\mathrm{y}(t) = \mathrm{ilt}\left(-\frac{5\,\mathrm{laplace}(\mathrm{y}(t)\,\mathrm{z}(t),t,g2176)+2\,\alpha\,\mathrm{laplace}\left(\mathrm{y}(t)^2,t,g2176\right)-6}{2\,g2176-8}, g2176, t\right),$$
$$\mathrm{z}(t) = \mathrm{ilt}\left(\frac{\mathrm{laplace}(\mathrm{y}(t)\,\mathrm{z}(t),t,g2176)-\alpha\,\mathrm{laplace}\left(\mathrm{y}(t)^2,t,g2176\right)+1}{g2176+2}, g2176, t\right)]$$

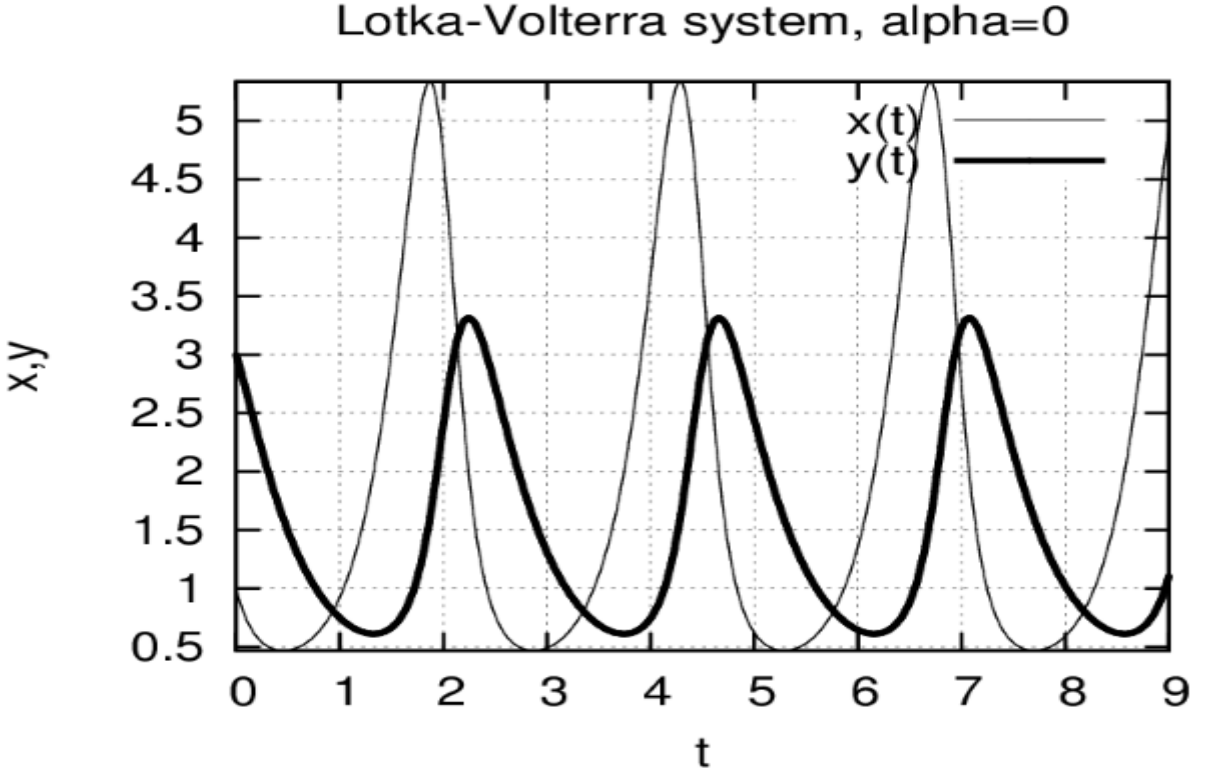
Очевидна проблема — нерозв'язність цієї системи у вигляді методом перетворення Лапласа, т.к. вона нелінійна.

Використовуємо чисельний метод Рунге-Кутта з dynamics пакета.

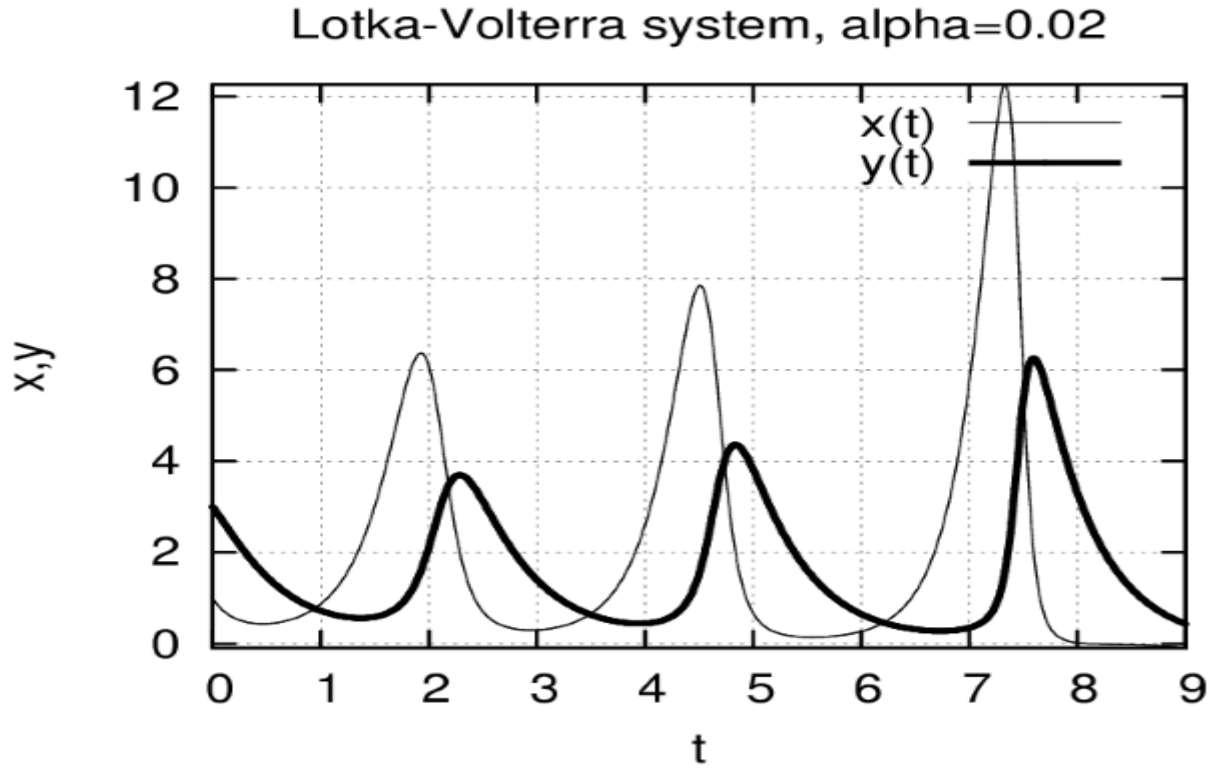
Результати рішення для значень$\alpha = 0$і$\alpha = 0.02$представлені на рис.6.11 та рис.6.12.



Мал. 6.10.Фазовий портрет для системи Лотка-Вольтерра



Мал. 6.11.Рішення системи Лотка-Вольтерра в залежності від часу (a = 0)



Мал. 6.12.Рішення системи Лотка-Вольтерра в залежності від часу (a = 0, 1)

Розглянемо командний файл для завдання моделювання системи Лотка-Вольтерра в Maxima:

```
a:4; b:2.5; c:2; d:1; alpha1:0;
ode1:(ab*x)*y-alpha1*x^2$ ode2:(-c+d*y)*x-alpha1*y^2$
alpha2:0.02;
ode3:(ab*x)*y-alpha2*x^2$ ode4:(-c+d*y)*x-alpha2*y^2$
load("dynamics");
t1:[]$ xg1:[]$ yg1:[]$ t2:[]$ xg2:[]$ yg2:[]$
l1:rk([ode1,ode2],[y,x],[1,3],[t,0,9,0.01])$
l2:rk([ode3,ode4],[y,x],[1,3],[t,0,9,0.01])$
for i:1 thru length(l1) do(t1:append(t1,[l1[i][1]])),
     xg1:append(xg1,[l1[i][2]]),
yg1:append(yg1,[l1[i][3]]));
for i:1 thru length(l2) do(t2:append(t2,[l2[i][1]])),
      xg2:append(xg2,[l2[i][2]]),
yg2:append(yg2,[l2[i][3]]));
load("draw");
draw2d(terminal='eps, file_name="lotka1",
grid=true,xlabel="x",
     ylabel = "y", title="Lotka-Volterra system, phaze
portrait",
     key = "alpha = 0", points_joined = true, point_type
= none,
     line_width = 4,color = black, points(xg1,yg1),
```

```
      points_joined = true, color = black, point_type =
none,
      line_width = 1,key="alpha=0.02", points(xg2,yg2))$
       draw2d(terminal='eps, file_name="lotka2",
grid=true,xlabel="t",
      ylabel = "x,y", title="Lotka-Volterra system,
alpha=0",
      key = "x (t)", points_joined = true, line_width =
1,
      color = black,point_type = none, points(t1,xg1),
      points_joined = true, line_width = 4, point_type =
none,
      color = black, key = "y(t)", points(t1,yg1))$
draw2d(terminal='eps, file_name="lotka3",
grid=true,xlabel="t",
      ylabel = "x,y", title="Lotka-Volterra system,
alpha=0.02",
      key = "x(t)", points_joined = true, point_type =
none,
      line_width = 1, color = black, points(t2, xg2),
      points_joined = true, line_width = 4, point_type =
none,
      color = black, key = "y(t)", points(t2,yg2))$
```

Диференціальні рівняння формуються символьними виразами, що визначають праві частини. Порядок проходження виразів для розрахунку правих частин ОДУ в першому списку функціїrk має відповідати один одному. Слід звернути увагу, що результат виконання функціїrk - Дворівневий список (кожен елемент списків$l1$і$l2$— також список, що містить значення незалежної змінної, і відповідні значення функцій, що шукаються), тому оп перетворюється на вектори, використовувані для побудови графічних ілюстрацій.

6.3.4 Рух твердого тіла

Розглянемо приклад побудови тривимірного фазового портрета. Знаходимо розв'язання задачі Ейлера вільного руху твердого тіла:

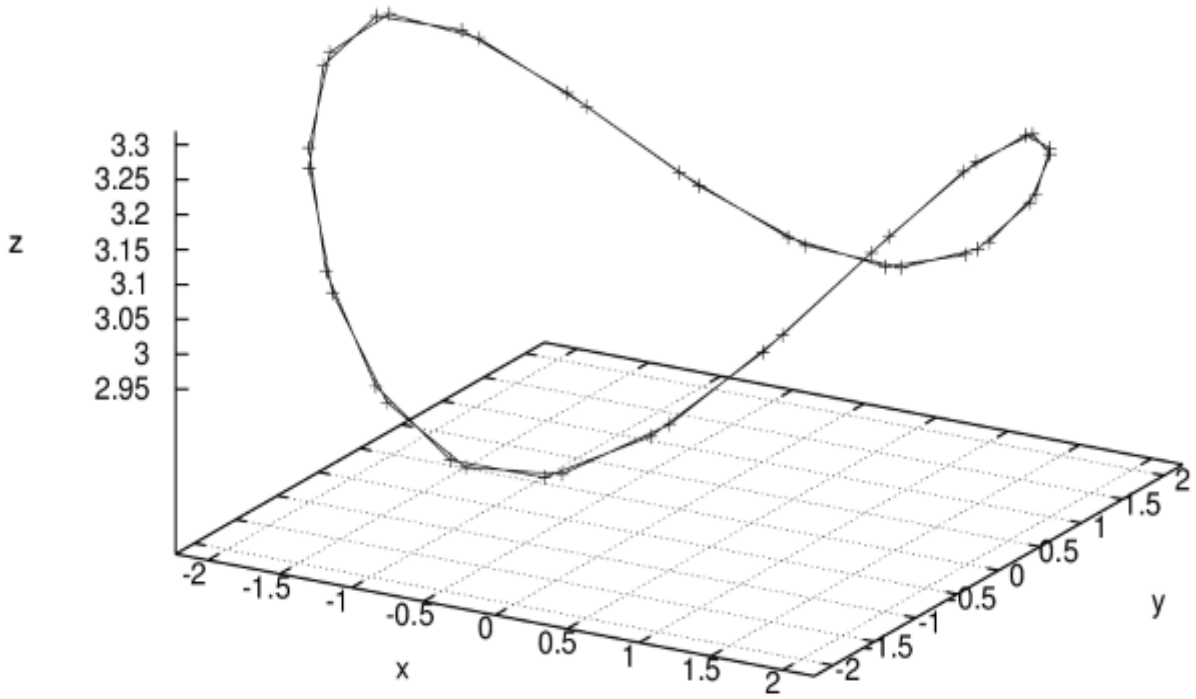
$$\frac{dx_1}{dt} = x_2 x_3,$$
$$\frac{dx_2}{dt} = -x_1 x_3,$$
$$\frac{dx_3}{dt} = -0.51 x_1 x_2.$$

```
(%i1)    eq1:'diff(x1(t)t) = x2 (t) * x3 (t);
    eq2:'diff(x2(t)t)=-x1(t)*x3(t);
    eq3:'diff(x3(t)t)=-0.51*x1(t)*x2(t);
```

$$(\%o1)\ \frac{d}{dt}x1\,(t) = x2\,(t)\ x3\,(t)$$
$$(\%o2)\ \frac{d}{dt}x2\,(t) = -x1\,(t)\ x3\,(t)$$
$$(\%o3)\ \frac{d}{dt}x3\,(t) = -0.51\,x1\,(t)\ x2\,(t)$$

```
(% i4)   load("dynamics")$ l: rk([y*z, -x*z,0.51*x*y],
    [x, y, z], [1,2,3], [t, 0,4,0.1]) $
```

Фазовий портрет даної динамічної системи (тривимірна крива) представлений на рис.6.13.



Мал. 6.13.Фазовий портрет тривимірної динамічної системи

6.3.5 Атрактор Лоренца

Одна з найзнаменитіших динамічних систем запропонована в1963 р. Лоренцем як спрощена модель конвективних турбулентних рухів рідини в посудині тороїдальної форми, що нагрівається. Система складається з трьох ОДУ і має три параметри моделі. Задаємо праві частини рівнянь моделі Лоренца:

```
(%i2) eq: [s * (yx), x * (rz) -y, x * y - b * z];
```

$$(\%o2)\quad [s\,(y-x), x\,(r-z)-y, x\,y-b\,z]$$

Задаємо тимчасові параметри рішення

```
(%i3) t_range: [t,0,50,0.05];
```

$$(\%o3)\quad [t, 0, 50, 0.05]$$

Задаємо параметри системи

```
(% i4)      s: 10.0; r: 28.0; b: 2.6667;
```

$$(\%o6)\quad 10.028.02.6667$$

Задаємо початкові значенняx, y, z

```
(%i7) init: [1.0,0,0];
```

$(\%o7)\quad [1.0, 0, 0]$

Виконуємо рішення

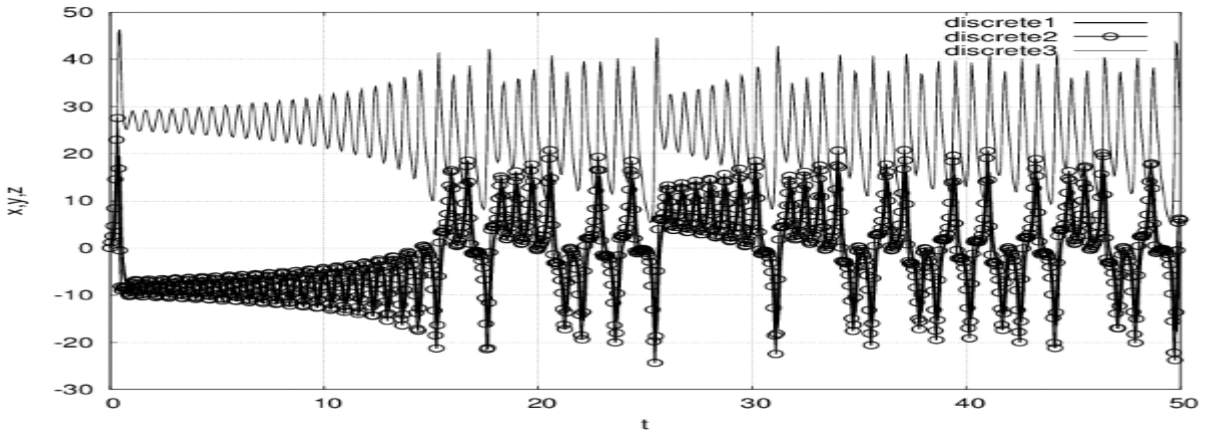
```
(% i8)      sol: rk(eq, [x, y, z], init, t_range) $
(% i9)      len:length(sol);
```

$(\%o9)\quad 1001$

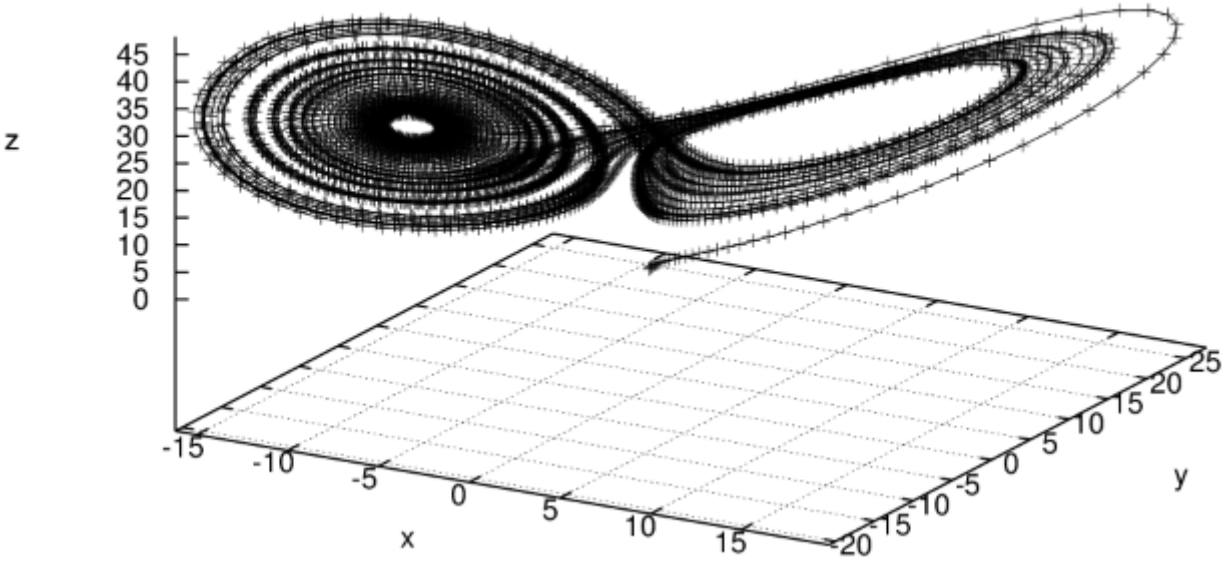
Виділяємо компоненти рішення та будуємо графічні ілюстрації:

```
(%i10)      t:makelist(sol[k][1],k,1,len)$
      x:makelist(sol[k][2],k,1,len)$
      y:makelist(sol[k][3],k,1,len)$
      z:makelist(sol[k][4],k,1,len)$
      plot2d([discrete,t,x])$ plot2d([discrete,t,y])$
```

Результати вирішення (хаотичні коливання x, y, z) представлений на рис.6.14 та рис.6.15 (фазовий портрет системи). На малюнках об'єднані в одних осях криві $x(t), y(t), z(t)$.



Мал. 6.14.Приклад формування динамічного хаосу (атрактор Лоренца)



Мал. 6.15.Тривимірний фазовий портрет (атрактор Лоренца)

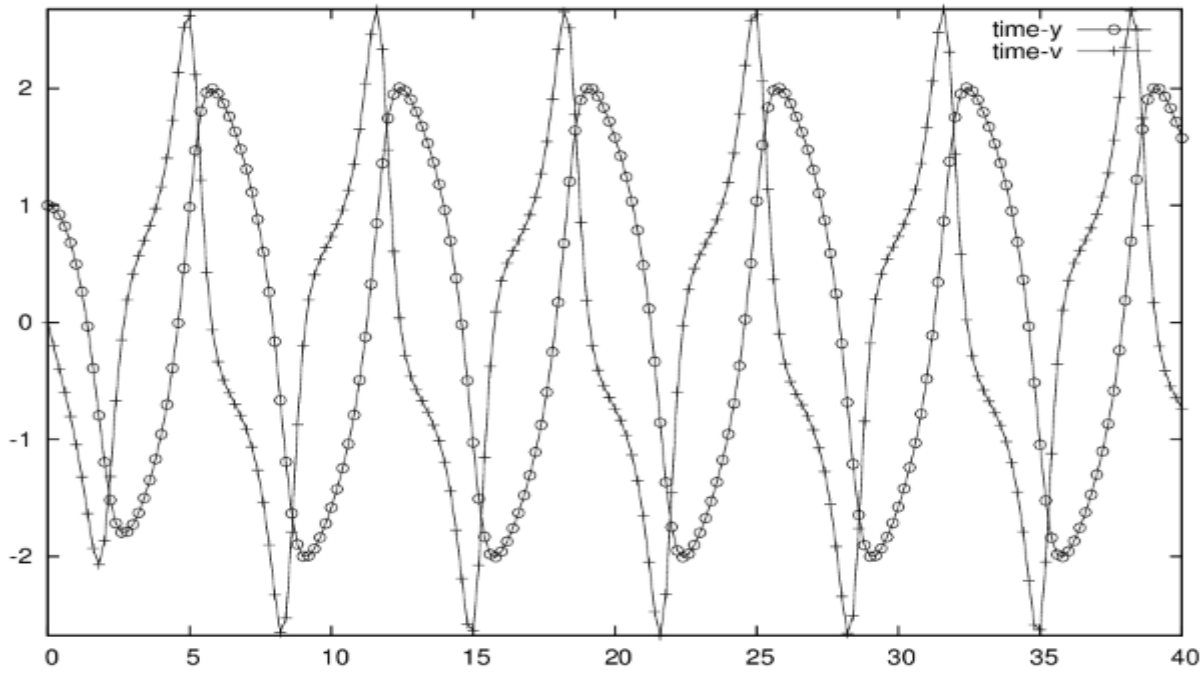
Рішенням системи Лоренца при певному поєднанні параметрів є дивний атрактор (або атрактор Лоренца) — безліч траєкторій, що притягує, на фазовому просторі, яке на вигляд ідентично випадковому процесу. У певному сенсі атрактор Лоренца є стохастичними автоколиваннями, що підтримуються в динамічній системі за рахунок зовнішнього джерела.

Рішення як дивного атрактора з'являється лише за деяких поєднаннях параметрів. Перебудова типу фазового портрета відбувається у сфері проміжних значеннях параметра r. Критичне поєднання параметрів, у яких фазовий портрет системи якісно змінюється, називається теорії динамічних систем точкою біфуркації. Фізичний сенс біфуркації в моделі Лоренца, згідно з сучасними уявленнями, описує перехід ламінарного руху рідини до турбулентного.

6.3.6 Модель автоколивальної системи: рівняння Ван дер Поля

Розглянемо рішення рівняння Ван дер Поля, що описує електричні коливання в замкнутому контурі, що складається з послідовно з'єднаних конденсатора, індуктивності, нелінійного опору і елементів, що забезпечують підкачування енергії ззовні. Невідома функція часу $y(t)$ має сенс електричного струму, а параметрі μ закладено кількісні співвідношення між складовими електричного ланцюга, у тому числі і нелінійною компонентою опору:

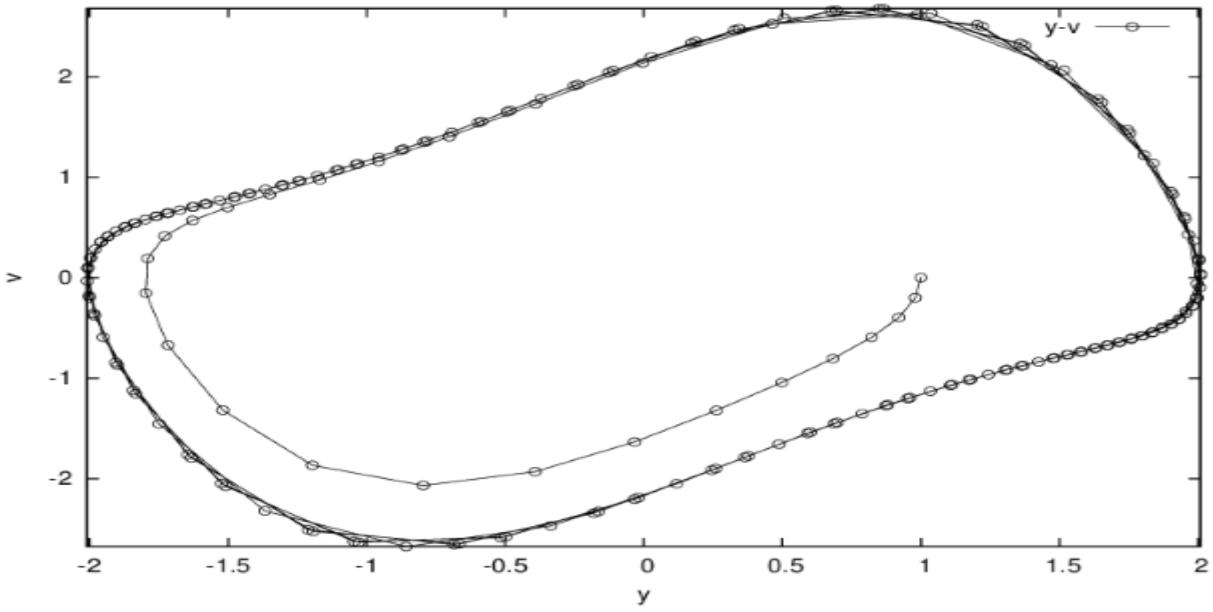
$$\frac{d^2y(t)}{dx^2} - \mu(1 - y(t)^2)\frac{dy(t)}{dt} + y(t) = 0$$



Мал. 6.16.Рішення рівняння Ван дер Поля

Рішенням рівняння Ван дер Поля є коливання, вид яких для $\mu = 1$ показаний на рис.6.16. Вони називаються автоколиваннями і відрізняються від розглянутих раніше (наприклад, чисельності популяцій у моделі Вольтерра) тим, що їх характеристики (амплітуда, частота, спектр) не залежать від

початкових умов, а визначаються виключно властивостями самої динамічної системи. Через деякий час розрахунків після виходу з початкової точки рішення виходить на той самий цикл коливань, званий граничним циклом. Атрактор типу граничного циклу є замкнутою кривою на фазовій площині. До нього асимптотично притягуються всі навколишні траєкторії, що виходять із різних початкових точок, як зсередини (рис.6.17), так і зовні граничного циклу.



Мал. 6.17.Фазовий портрет рівняння Ван дер Поля

Використаний командний файл Maxima (для побудови графічної ілюстрації використано пакет draw):

```
load("dynamics")$ load("draw")$
mu:1$ s:rk ([v,mu*(1-y^2)*vy],[y,v],[1,0],[t,0,40,0.2])$
time:makelist(s[k][1],k,1,length(s))$
y:makelist(s[k][2],k,1,length(s))$
v:makelist(s[k][3],k,1,length(s))$
draw2d(points_joined = true, point_type=6, key= "yv",
     xlabel="y",ylabel="v",points(y,v), terminal=eps)$
```