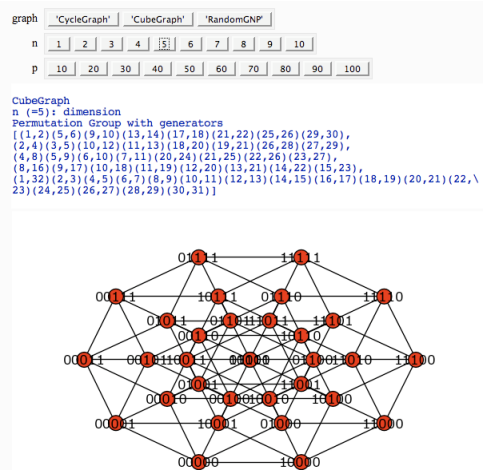
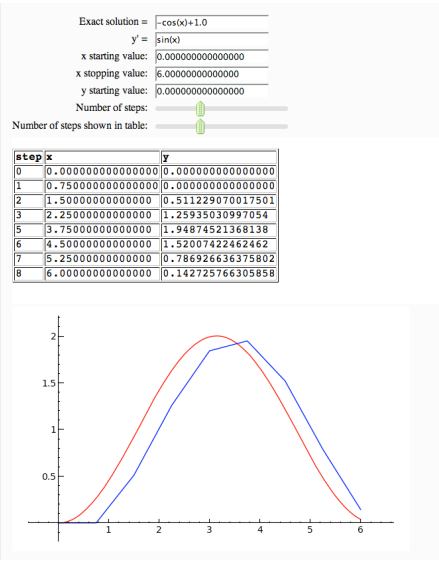
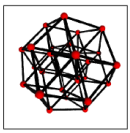
С.В. Шокалюк

Основи роботи в SAGE







Київ
2008

Міністерство освіти та науки України
Національний педагогічний університет
імені М.П. Драгоманова

С.В. Шокалюк

Основи роботи в SAGE

Київ
2008

УДК 373.167.1:002

Шокалюк С.В. Основи роботи в SAGE / За ред. академіка АПН України М.І. Жалдака. – К.: НПУ імені М.П. Драгоманова, 2008. – 64 с.

В посібнику для практичної підготовки наведено короткий огляд основних прийомів роботи у вільно поширюваній Web-системі комп'ютерної математики SAGE, спрямованої на підтримку алгебраїчних та геометричних досліджень.

Для широкого кола студентів ВНЗ різного профілю, педагогів, наукових та інженерних працівників, вчителів та учнів старших класів, які застосовують інформаційні технології математичного призначення.

Рецензенти:

В.М. Соловйов – д.ф.-м.н., професор, завідувач кафедри економічної кібернетики Черкаського національного університету ім. Богдана Хмельницького

І.О. Теплицький – к.пед.н., доцент кафедри інформатики та прикладної математики Криворізького державного педагогічного університету

Друкується згідно з рішенням ученої ради Криворізького державного педагогічного університету, протокол №10 від 8 травня 2008 р.

ISBN 966-8413-20-6

ЗМІСТ

ПЕРЕДМОВА

Однією з проблем, що постають в процесі навчання комп'ютерної математики, є вибір середовища для роботи. Як комерційні, так і вільно поширювані системи суттєво різняться за функціональністю (загального призначення, спеціалізовані), інтерфейсом (командного рядка, графічним), розміром (від кількасот кілобайт до кількох гігабайт), вбудованою мовою програмування тощо. Безальтернативне ознайомлення лише з однією системою комп'ютерної математики (навіть такої розвиненої, як Maxima) неминуче впливатиме на подальшу професійну діяльність, обмежуючи клас розв'язуваних задач можливостями конкретного програмного продукту.

Як в педагогічній, так і в інженерній та науково-дослідницькій роботі діє єдиний принцип: *вибір інструмента визначається задачею*, тому обійтися лише однією системою комп'ютерної математики (СКМ) не вдається, та й не потрібно.

Фактором, що утруднює вивчення та застосування різних СКМ, є синтаксичні відмінності у застосуванні одних й тих самих команд, що можуть змінюватися навіть в межах однієї СКМ (наприклад, при виході нової версії). Інша проблема – те, що досить часто в системах загального призначення не вистачає можливостей, що є в спеціалізованих системах, та навпаки (зауважимо, що частково ця проблема може бути розв'язана через застосування сучасних об'єктних технологій інтеграції програмного забезпечення, таких як COM та CORBA). В результаті для роботи буває необхідним цілий «зоопарк» СКМ, встановлених на одному комп'ютері, що в умовах загальноосвітньої школи та ВНЗ реалізувати практично неможливо через ліцензійні чи адміністративні перешкоди.

Проблема вибору СКМ та підтримки великої інсталяційної бази може бути розв'язана через застосування мережних технологій, коли користувач за допомогою спеціалізованого клієнтського програмного забезпечення (ПЗ) звертається до серверної частини СКМ, що виконує команди користувача та поповертає результат до клієнтського ПЗ. Таку можливість надають, зокрема, Matlab Web Server, webMathematica та wxMaxima. І, хоча далеко не всі СКМ мають вбудовані мережні можливості, для тих з них, що поряд з візуальним, підтримують командний інтерфейс, можливе створення мережної надбудови.

Логічним заключним кроком буде створення оболонки, що інтегрує в собі можливості різних СКМ за допомогою клієнт-серверних технологій, надаючи користувачу рівних доступ до різних СКМ за допомогою єдиної командної мови (а за необхідності – легко переключатися на мову будь-якої СКМ). Тоді на клієнтському комп'ютері не буде потреби у встановленні та налагодженні спільної роботи різних СКМ – достатньо

звернутися до математичного серверу засобами Web-браузера.

Web-СКМ – новий перспективний напрямок розвитку СКМ; перші представники таких систем з'явились лише на початку нашого століття. Інтеграція СКМ у єдине мережне середовище – найкраща ілюстрація сучасної концепції «комп'ютер – це мережа», що знаходить своє відображення у перенесенні прикладного ПЗ (навіть «робочих столів») у Web-середовища.

Для користувача це надає можливість мобільного доступу до програм та даних, для адміністратора комп'ютерного класу знімає проблему підтримки великої інсталяційної бази та ліцензування ПЗ, для викладачів – суттєво розширює спектр використовуваного ПЗ, а для учнів та студентів створює умови для дистанційного навчання.

Метою роботи автора посібника є створення єдиного діяльнісного середовища для навчання технології проведення математичних досліджень на основі об'єднання можливостей систем дистанційного навчання (СДН) та мережних СКМ. Якщо принципи роботи з СДН достатньо представлені у вітчизняній методичній літературі, то в тому, що стосується Web-СКМ, цей посібник є першою ластівкою, адже й описаний програмний продукт є першим представником Web-інтеграторів СКМ.

С.О. Семеріков,
серпень 2008 р.

ВСТУП

SAGE (*Software for Algebra and Geometry Experimentation* – програмне забезпечення для алгебраїчних та геометричних досліджень) – це безкоштовне вільно поширюване середовище математичних обчислень для виконання символьних, алгебраїчних та чисельних розрахунків, інтерфейс якого написаний потужною і досить популярною мовою програмування Python. SAGE об'єднав можливості популярних вільно поширюваних математичних програм та бібліотек, таких як PARI, GAP, GSL, Singular, MWRANK, NetworkX, Maxima, Sympy, GMP, Numpy, mathplotlib та багатьох інших засобами Python, Lisp, Fortran 95 та C/C++.

SAGE є серверним ПЗ, що базується на відомому Python-CMS Zope. Інтерактивність Web-клієнта забезпечується широким застосуванням технології AJAX, що є основою більшості продуктів Web 2.0, а адекватність відображення математичної інформації – браузерними математичними шрифтами (jsMath).

SAGE має власне символьне ядро, проте виступає переважно як інтегратор різних систем, надаючи їм єдиний Web-інтерфейс. Можливість виконання на Web-сторінках, генерованих SAGE, програм мовами Fortran, Python, Lisp, Java та ін., надає їм надвисокого рівня інтерактивності, порівняного з традиційними СКМ (Mathematica, MathCAD, Maple), без суттєвих вимог до апаратних ресурсів комп'ютера користувача (необхідні лише браузер та мережне з'єднання).

Проектом SAGE керує професор факультету математики Вашингтонського університету (м. Сіетл) Вільям Штейн. Кінцевою метою проекту є створення відкритого високоякісного програмного забезпечення як гідної альтернативи комерційним програмним засобам, таким як Maple, Mathematica, MuPAD чи Matlab.

Перша версія SAGE з'явилася у лютому 2006 року, друга версія датується жовтнем того ж року, останньою на сьогодні (вересень 2008 р.) є версія 3.1.2. За 3,5 роки розвитку SAGE об'єднав у собі більш ніж 100 компонентів (СКМ, обчислювальних бібліотек, пакетів візуалізації та ін.). Найновіша версія SAGE завжди доступна за адресою `http://www.sagemath.org/`.

Основними складовими SAGE є:

– інтерфейси до СКМ Magma, Maple, Mathematica, Matlab, MuPAD та ін.;

– якісні пакети для алгебри та обчислень (Maxima), швидких високоточних обчислень (GMP), алгебраїчної геометрії (Singular), лінійної алгебри (Linbox), графіки (Gnuplot), теорії груп (GAP), теорії чисел (PARI), оптимізації (GSL) та ін.

– мови програмування (Python, Lisp, Fortran 95, C/C++ та ін.).

SAGE має два інтерфейси – локальний інтерфейс командного рядка (рис. 1) та Web-інтерфейс (рис. 2).

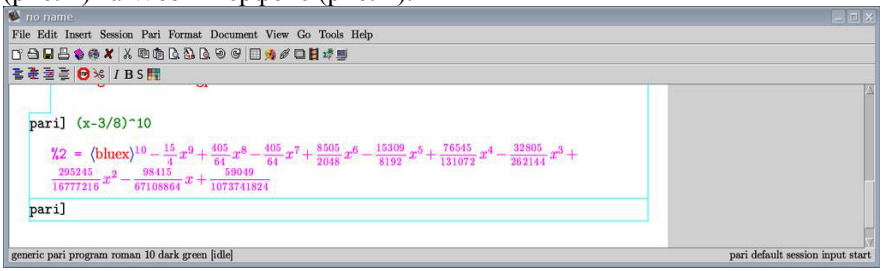


Рис. 1. Інтерфейс командного рядка SAGE

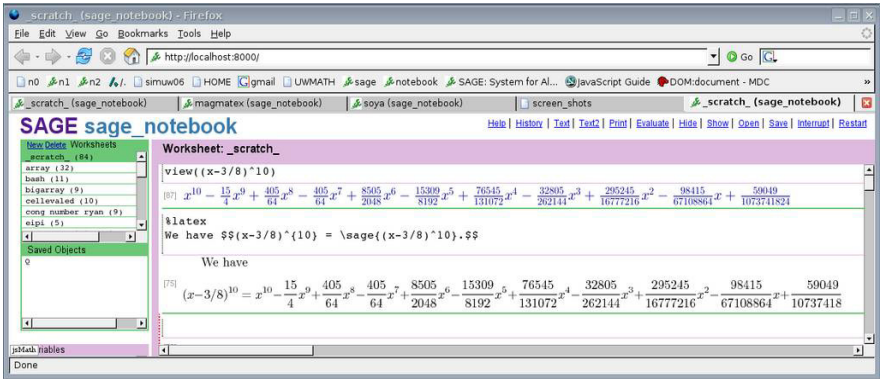


Рис. 2. Web-інтерфейс SAGE

Web-інтерфейс SAGE отримав назву блокнотного (*notebook*), тому що він являє собою комп'ютерну модель записника, який математики традиційно використовують для виконання математичних розрахунків. Подібний інтерфейс є традиційним для розвинених СКМ: достатньо порівняти представлені на рис. 3 блокнотні інтерфейси Maple 10 (зверху), Mathematica 5.2 (усередині) та SAGE 3 (знизу).

Наявність Web-інтерфейсу, безкоштовність та відкритість середовища SAGE – це основні, але не єдині переваги цього інтегратора. Слід вказати ще такі можливості SAGE:

– невимогливість до апаратної складової обчислювальної системи;

– індиферентність до використовуваного браузера та операційної системи;

– підтримка інтерфейсів комерційних систем комп'ютерної математики – Maple, Magma, Mathematica, Matlab та ін.;

– подання математичних виразів у природний спосіб не вимагає встановлення додаткового програмного забезпечення – достатньо дозавантажити математичні шрифти;

– публікація робочих листів (Worksheets) у мережі Internet;

– підтримка технології Wiki;

– потужний інструментарій для побудови статичних та динамічних графічних зображень (на площині та у просторі).

Для організації роботи у локальній мережі достатньо встановити SAGE на будь-якому комп'ютері.

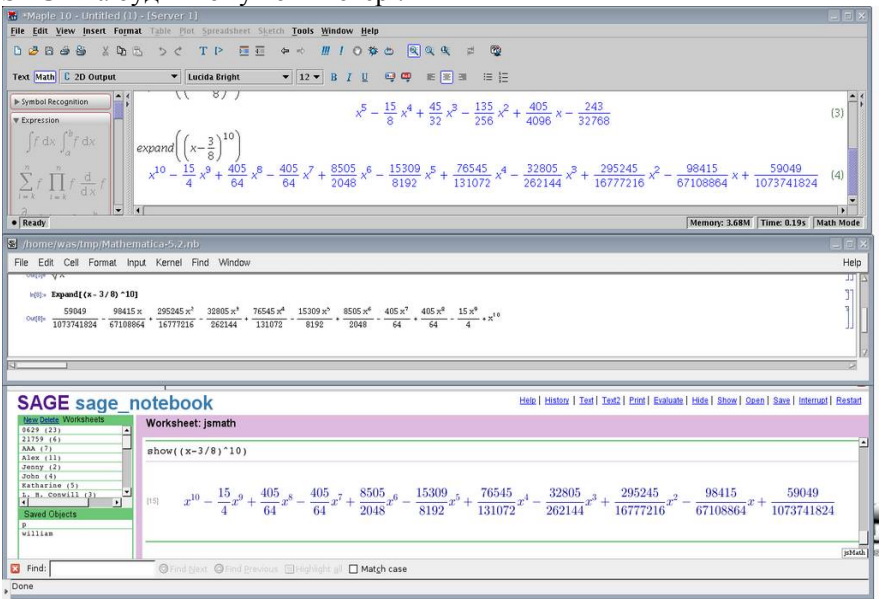


Рис. 3. Блокнотні інтерфейси Maple, Mathematica, SAGE

Враховуючи такі можливості SAGE, як розвинений Web-інтерфейс та підтримка технології Wiki, дана програма була обрана в якості інструментального засобу для дистанційного навчання розділу «Інформаційні технології навчального призначення: математика» курсу інформатики рядом шкіл м. Кривого Рогу. Починаючи з 2008 р., SAGE використовується у Криворізькому державному педагогічному університеті в процесі навчання теорії алгоритмів, моделювання, методів оптимізації, чисельних методів, теорії кодування, паралельних та розподілених обчислень, розпізнавання образів, вейвлет-аналізу та інших навчальних дисциплін.

Також SAGE може бути використаний в процесі навчання елементарної та вищої математики, у тому числі лінійної та вищої алгебри, геометрії, математичного аналізу, методів математичної фізики, теорії чисел, комбінаторики, теорії графів та багатьох інших розділів.

З метою формування первинних навичок проведення математичних експериментів у новому програмному середовищі було створено дista-

нційний курс «SAGE: легкий старт» (http://kdpu.edu.ua/moodle). До змістовної частини курсу включені питання з виконання основних операцій з робочими листами, способів отримання довідкової інформації, а також завдання, які ілюструють основні можливості SAGE з виконання елементарних математичних розрахунків (чисельних та символьних), операцій з виразами, розв'язання рівнянь та їх систем, а також приклади побудови графічних зображень. Добір завдань був виконаний з урахуванням змісту лише шкільного курсу математики, адже дистанційний курс розрахований на будь-якого користувача, який розпочинає роботу в SAGE.

Даний посібник для практичної підготовки є не просто друкованим варіантом матеріалів дистанційного курсу: його можна використовувати як для самостійного опанування SAGE, так і в якості довідника. У додатках до посібника наведені розвинені приклади застосування SAGE для побудови інтерактивних моделей та розв'язання задач теорії кодування. В останньому додатку показано, як можна розширити можливості SAGE та інтегрувати у нього нове ПЗ.

В якості продовження даної роботи планується створення серії посібників, орієнтованих на застосування SAGE в якості середовища для навчальних досліджень.

1. ПОЧАТОК РОБОТИ У SAGE

Після виконання всіх операцій з підготовки SAGE до роботи (див. додаток А) та його запуску за виділеною IP-адресою у вікні Web-браузера буде відкрита домашня сторінка користувача SAGE (рис. 1.1).

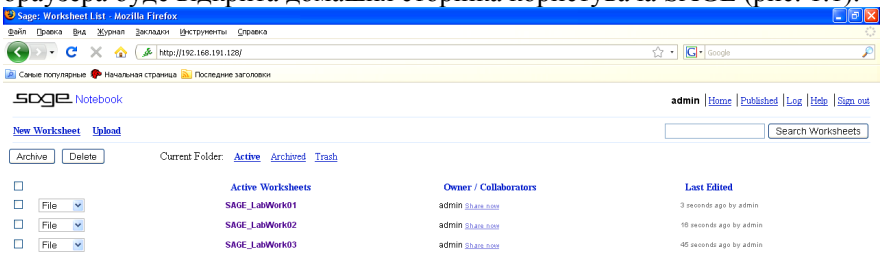


Рис. 1.1. Домашня сторінка користувача SAGE (Worksheet List)

На домашній сторінці представлений персональний список робочих листів. Робочі листи (*Worksheets*) – це основні елементи SAGE, оскільки математичні розрахунки виконуються саме на робочих листах.

Всі робочі листи блокноту розподілені по папках за категоріями:

– активні листи (листи, необхідні користувачу для роботи найближчим часом);

– архівні листи;

– листи, підготовлені до вилучення.

На початку сеансу роботи з блокнотом SAGE поточною папкою (*Current Folder*) є папка з активними листами – *Active*. Додати до списку перегляду архівні листи можна, обравши посилання *Archived*, а перейти у режим перегляду листів, підготовлених до вилучення – *Trash*.

Активні листи можна перемістити до папки-архіву (кнопка *Archive*) або до папки-кошика (кнопка *Delete*). Листи з архіву можуть бути перенесені до папки *Active* (кнопка *Unarchive*) або вилучені. У кошику SAGE з робочими листами можуть бути виконані дії, аналогічні до тих, які передбачені у Кошику операційної системи Windows – остаточного знищення або відновлення вилучених листів (*Undelete*). Перш ніж дати команду перенесення листа (чи листів) між папками, їх треба відмітити пташкою.

Над робочими листами, як над файлами, можна виконати операції перейменування (*Rename*), редагування (*Edit*), копіювання (*Copy Worksheet*), публікації в мережі Інтернет (*Publish*), перегляду хід роботи з листом (*Revisions*). Окрім того, можна визначити перелік користувачів, які можуть спільно працювати з певним робочим листом (*Collaborate*). Список зазначених дій представлений на домашній сторінці користувача у вигляді випадаючого списку *File* ліворуч від заголовку кожного робочо-

го листа.

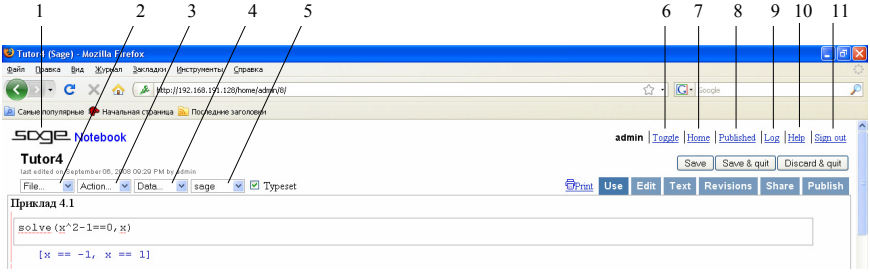
Вибрати робочий лист для подальшої роботи користувач може одним із зазначених способів:

1) створити новий робочий лист, вибравши посилання *New worksheets*;

2) завантажити робочий лист (файл з розширенням *sws*) з довільного носія або з Інтернету за вказаною URL-адресою, вибравши посилання *Upload*;

3) відкрити існуючий робочий лист, вибравши його зі списку. У разі завеликої кількості листів передбачена можливість здійснення автоматичного пошуку необхідного робочого листа (*Search Worksheets*), аналогічно до пошуку звичайної Web-сторінки в межах сайту.

Web-сторінка з робочим листом та його основними елементами представлені на рис. 1.2.



1 – банер-посилання на сайт `www.sagemath.org`
2 – список *File* допустимих операцій над листами блокноту
3 – список *Action* допустимих дій над даними листа
4 – список *Data* для роботи з файлами даних
5 – список пакетів, інтерфейси яких підтримує SAGE
6 – *Toggle*– приховати/показати панель управління
7 – *Home* – перейти до персонального списку листів
8 – *Published* – показати список опублікованих листів
9 – *Log* – показати історію останніх дій
10 – *Help* – викликати довідкову систему
11 – *Sign out* – завершити сеанс роботи з блокнотом

Рис. 1.2. Вікно робочого листа

З листом можна виконати такі операції (2): друкувати (*Print*), перейменувати (*Rename worksheet*), копіювати (*Copy worksheet*) та вилучити (*Delete worksheet*). Окрім того, у вікні робочого листа можна дати команду створення нового робочого листа (*New worksheet*) та завантаження робочого листа з довільного носія чи мережі Інтернет (*Upload worksheet from a file*). Команда *Download to a file* служить для збереження робочого листа у вигляді файлу з розширенням *sws*. Збереження файлу виконується у каталог за замовчанням, визначений в параметрах браузера.

З даними робочого листа можна працювати у режимах *Use*, *Edit* та *Text*.

Режим *Use* – основний режим роботи – режим перегляду та виконання математичних розрахунків.

Режим *Edit* – режим редагування. У даному режимі користувач має можливість додати до змісту робочого листа тексти завдань та виконати різні маніпуляції з блоками математичних розрахунків (вилучати, переміщувати та ін.). *Зауваження*: завершуючи роботу в цьому режимі, обов'язково збережіть зміни (кнопка *Save Changes*), інакше вони будуть відкинуті без попередження.

Режим *Text* служить для відображення даних робочого листа (завдань, команд та результатів) у текстовому (нерозміченому) форматі.

Вибір *Revisions* дозволяє переглянути історію роботи над листом.

За допомогою *Share* визначається перелік користувачів, які можуть спільно працювати з певним робочим листом.

Publish відкриє вікно для підготовки робочого листа до публікації у мережі.

В основному режимі роботи користувач має можливість ввести у прямокутну комірку команду чи програмний код для виконання математичних розрахунків та переглянути результат обчислень.

Автодоповнення вводу в SAGE суттєво спрощує процес введення команд: увівши перші літери команди і натиснувши клавішу табуляції, користувач отримує список команд, що починаються з введеної послідовності символів.

Дати команду «розпочати обчислення» за введеною командою чи програмним кодом у комірці можна, натиснувши комбінацію клавіш `<Shift> + <Enter>` або обравши посилання `evaluate` під командним рядком. Під коміркою з'явиться зелена вертикальна риска, яка вказує на те, що на сервері SAGE виконуються обчислення. По завершенню обчислень риска зникає і під коміркою з'являється результат обчислення або повідомлення про допущену помилку.

Результати обчислень можуть бути подані у текстовому форматі (за замовчуванням) або у природній математичній формі. Для відображення математичних виразів у природній формі слід встановити математичні шрифти викликом *jsMath Control Panel* у відповідному вікні повідомлення (рис. 1.3).

Рис. 1.3. Вікно попередження jsMath

Найчастіше помилки виникають через неправильний синтаксис команди. Для виправлення таких помилок доречним є звернення до довідкової системи SAGE. Контекстну довідку можна отримати, ввівши `?` після імені команди та `<Shift> + <Enter>`.

Введення `??` після назви функції дозволяє переглянути її програмний код.

Після виведеного результату на робочому листі автоматично з'являється нова комірка для введення команди. Додати нову комірку у довільному місці робочого листа можна вибором горизонтальної синьої лінії, яка з'являється у момент нависання курсору миші над будь-якою коміркою. Вилучення зайвої порожньої комірки виконується клавішею `<Backspace>`.

За замовчування робочий лист налаштований на роботу з командами SAGE. Вибрати іншу СКМ можна у списку (5) або командою виду `%назва_СКМ`. Наприклад: `%maxima`, `%maple` і т.д.

Основні дії над робочим листом представлені у списку *Action...* (3):

Interrupt – перервати процес обчислень

Restart worksheet – перезапустити сеанс роботи з робочим листом

Quit worksheet – завершити сеанс роботи з робочим листом

Evaluate All – виконати програмні коди у всіх комірках

Hide All Output – приховати всі результати виконання команд

Show All Output – показати всі результати виконання команд

One Cell Mode – залишити доступною для роботи лише одну першу комірку листа

Multi Cell Mode – зробити доступними для роботи всі комірки листа

Для завантаження чи створення файлу даних передбачена відповідна команда *Upload or create Data File* списку *Data…* (4).

Завершити сеанс роботи з листом можна одним із передбачених способів, а саме:

– зберегти зміни – кнопка *Save*;

– зберегти зміни та закрити вікно з робочим листом – кнопка *Save & quit*;

– проігнорувати зміни та закрити вікно з робочим листом – кнопка *Discard & quit*.

2. ОПЕРАЦІЇ З ВИРАЗАМИ

Вирази SAGE – це комбінації чисел, змінних, арифметичних операторів (“+”, “–”, “*”, “/”, “%”, “^”, “**”), операторів порівняння (“==”, “<>”, “!=”, “<”, “>”, “<=”, “>=”), логічних операторів (“not”, “and”, “or”), дужок (“(”, “)”) і функцій (табл. 2.1.).

Таблиця 2.1.

Основні математичні функції

Команда	*Функція*	*Опис дії функції*
`abs(x)`	$\|x\|$	модуль числа (абсолютне значення)
`sqrt(x)`	$\sqrt{x}$	квадратний корінь
`exp(x)`	e^x	експонента
`ln(x)`	$\ln x$	натуральний логарифм
`log(x,b)`	$\log_b x$	логарифм за основою b
`factorial(n)`	$n!$	факторіал числа n ($n!=1\cdot2\cdot3\cdot\ldots\cdot n$)
`sin(x)`	$\sin x$	синус
`cos(x)`	$\cos x$	косинус
`tan(x)`	$\mathrm{tg}\, x$	тангенс
`cot(x)`	$\mathrm{ctg}\, x$	котангенс
`asin(x)`	$\arcsin x$	арксинус
`acos(x)`	$\arccos x$	арккосинус
`atan(x)`	$\mathrm{arctg}\, x$	арктангенс
`acot(x)`	$\mathrm{arcctg}\, x$	арккотангенс
`sinh(x)`	$\mathrm{sh}\, x$	синус гіперболічний
`cosh(x)`	$\mathrm{ch}\, x$	косинус гіперболічний
`tanh(x)`	$\mathrm{th}\, x$	тангенс гіперболічний
`coth(x)`	$\mathrm{cth}\, x$	котангенс гіперболічний

Таблиця 2.2.

Приклади запису математичних виразів

Вираз	*Команда*
$\sqrt{5x-3+x^2}$	`sqrt(5*x-3+x^2)`
$\sin x-\frac{\sqrt{3}}{2}x$	`sin(x)-sqrt(3)/2*x`
$\frac{3}{2\sqrt{x+4}}+\frac{4}{x-4}$	`3/(2*sqrt(x+4))+4/(x-4)`
$3x\ln x$	`3*x*log(x)` або `3*x*ln(x)`
$(1-\mathrm{e}^{3x})^2$	`(1-exp(3*x))^2` або `(1-exp(3*x))**2`

Вираз	*Команда*
$x^3\log_3 x$	`x^3*log(x,3)`
$3e^x - 5\cdot 8^x + 1$	`3*exp(x)-5*8^x+1`

Над виразами в середовищі SAGE можна виконати операції спрощення, розкриття дужок та розкладання на множники.

Операцію спрощення (без розкриття дужок) виконує функція `simplify`.

Приклад 2.1. Спростити вираз: $3x^2 + 5x + 17x - x^2$.

```
sage: simplify(3*x^2+5*x +17*x-x^2)
2*x^2 + 22*x
```

Інколи, досить зручно для роботи визначити вираз як певну змінну. (Цей крок цілком виправданий, якщо над виразом треба виконати кілька операцій.) Подальша робота зі змінною типу «вираз» може бути виконана двома способами:

1) застосувати змінну як параметр відповідної функції (див. приклад 2.2*а*);

2) для змінної, як для об'єкту, виконати функцію-метод (див. приклад 2.3*б*).

Приклад 2.2. Спростити вираз: $(x - 1)(x - 1)(2x - 3)$.

а)

```
sage: f=(x-1)*(x-1)*(2*x-3); simplify(f)
(x - 1)^2*(2*x - 3)
```

б)

```
sage: f=(x-1)*(x-1)*(2*x-3); f.simplify()
(x - 1)^2*(2*x - 3)
```

Для розкриття дужок використовується функція `expand`.

Приклад 2.3. Розкрити дужки у виразі: $(x - 1)(x^2 - 1)$.

```
sage: expand((x-1)*(x^2-1))
x^3-x^2-x+1
```

Приклад 2.4. Розкрити дужки у виразі: $3x(x - 6) - (2x^2 - 14)$.

```
sage: a=3*x*(x-6)-(2*x^2-14); a
3*x*(x-6)-(2*x^2-14)
sage: expand(a)
x^2-18*x+14
```

Приклад 2.5. Розкрити дужки у виразі: $(x - 1)(x^2 - 2x + 2)$.

```
sage: b=(x-1)*(x^2-2*x+2); b
(x-1)*(x^2-2*x+2)
sage: b.expand()
x^3-3*x^2+4*x-2
```

Операцію розкладання на множники виконує функція `factor`.

Приклад 2.6. Розкласти на множники: $x^{12} - 1$.

```
sage: factor(x^12-1)
(x-1)*(x+1)*(x^2-x+1)*(x^2+1)*(x^2+x+1)*(x^4-x^2+1)
```

Якщо у виразі присутня більше ніж одна змінна, SAGE вимагає оголошення всіх їх символьними змінними – `var('x,y')` або `(x,y)=var('x,y')`.

Приклад 2.7. Розкласти на множники: $a^2 - ab - 4a + 4b$.

```
sage: var('a,b')
(a,b)
sage: factor(a^2-a*b-4*a+4*b)
(a-4)*(a-b)
```

Приклад 2.8. Розкласти на множники: $ax + ay - az + nx + ny - nz$.

```
sage: var('x,y,z,a,n')
(x,y,z,a,n)
sage: exp=a*x+a*y-a*z+n*x+n*y-n*z
-n*z-a*z+n*y+a*y+n*x+a*x
sage: factor(exp) або exp.factor()
(n+a)*(-z+y+x)
```

Для обчислення значення виразу при певних значеннях змінних треба даний вираз подати як відповідну функцію.

Приклад 2.9. Обчислити значення виразу $\sqrt[3]{x} + \frac{1}{8}\sin(10x)$ при $x = 2$.

```
sage: f(x)=x^(1/3)+1/8*sin(10*x); f(x)
sin(10*x)/8+x^(1/3)
sage: f(2)
sin(20)/8+2^(1/3)
```

Отримати числове значення виразу дозволяє функція `RR`:

```
sage: RR(f(2))
1.37403920623583
```

Приклад 2.10. Обчислити значення виразу $a^2 - ab - 4a + 4b$ при $b = 2$.

```
sage: g(a,b)=(a^2-a*b-4*a+4*b);g(a,b)
-a*b+4*b+a^2-4*a
sage: g(a,2)
a^2-6*a+8
```

3. ПОБУДОВА ГРАФІЧНИХ ЗОБРАЖЕНЬ

Засобами mathplotlib в SAGE можна побудувати:

– графічні примітиви (точка, стрілка, лінія, коло, круг, заповнений (зафарбований) многокутник, сфера);

– графіки функціональних залежностей, заданих аналітично, параметрично та у полярних координатах;

– правильні многогранники;

– поверхні, задані аналітично та параметрично,

а також додавати підписи до графічних об'єктів.

3.1. Побудова графічних примітивів на площині

Графічний примітив *точка* на площині визначають функції `point` або `points`. Зазначені функції не просто задають, а й будують примітив, використовуючи параметри системи координат за замовчуванням.

Приклад 3.1. Побудувати точку на площині.

```
sage: point((-1,2))
```

або

```
sage: show(point((-1,2)))
```

або

```
sage: a=point((-1,2)); a
```

або

```
sage: a=point((-1,2)); show(a)
```

або

```
sage: a=point((-1,2)); a.show()
```

Зазначені функції можуть бути використані і для побудови множини точок, заданих відповідним чином.

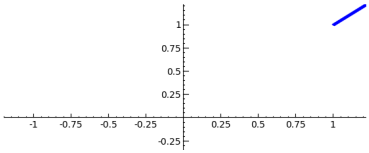
Приклад 3.2. Побудувати дві червоні точки збільшеного розміру.

```
sage: point(((-1,1),(1,-1)),pointsize=30,rgbcolor='red')
```

Для побудови примітиву *стрілка* за даними координатами кінців використовується функція `arrow`.

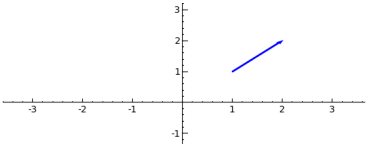
Приклад 3.3. Побудувати стрілку за даними координатами.

```
sage: arrow((1,1),(2,2))
```



Якщо уважно подивитися на результат виконання побудови, то можна помітити, що на рисунку стрілка недобудована. Для адекватного відображення слід розширити область видимості системи координат:

```
sage: show(arrow((1,1),(2,2)),xmin=-3,xmax=3,ymin=-3,ymax=3)
```



Приклад 3.4. Побудувати лінію від точки (–1,5) до точки (2,–3).

```
sage: line(((-1,5), (2,-3)))
```

Приклад 3.5. Побудувати ламану у вигляді пунктирної лінії з маркерами по точках (–2,2), (–1,–3), (0.5,0), (2,6) та (7,–4).

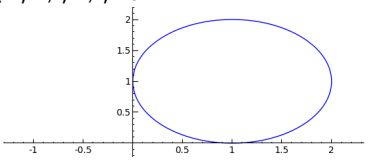
```
sage: line( ((-2,2),(-1,-3),(0.5,0),(2,6),(7,-4)),\
      linestyle='--', marker='^')
```

Обернений слеш (\) використовується для перенесення команди на наступний рядок.

Для побудови примітива *коло* за координатами центра та радіусом використовується функція `circle`.

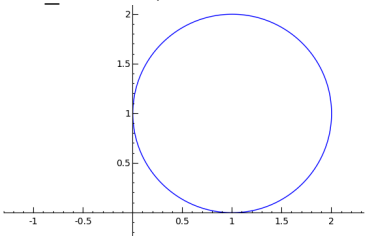
Приклад 3.6. Побудувати коло одиничного радіусу з центром у точці (1,1).

```
sage: c1=circle((1,1),1); c1
```



Побудоване коло більше нагадує еліпс. Це пов'язане з тим, що формат зображення системи координат за замовчуванням не відповідає співвідношенню 1:1. Формат зображення можна змінити за допомогою параметра aspect_ratio функції show, встановивши його значення рівним 1:

```
sage: show(c1,aspect_ratio=1)
```



Функція `circle` також може бути використана для побудови круга, змінивши значення параметра `fill` за замовчуванням з `false` на `true`.

Приклад 3.7. Побудувати круг одиничного радіусу з центром у точці (0,0).

```
sage: c2= circle((0,0),fill=true,rgbcolor=(0.1,0.2,0.3))
sage: c2.show(aspect_ratio=1)
```

Інший спосіб побудови круга пропонує функція `disk`.

Приклад 3.8. Побудувати круг одиничного радіусу з центром у точці (0,0).

```
sage: show(disk((0,0),1,(0,2*pi)),aspect_ratio=1);
```

Використовуючи функцію `disk`, також можна побудувати частину круга.

Приклад 3.9. Побудувати сектор круга з центром у точці (–1,–1) довільного радіуса. Початкова дуга сектора відповідає куту $\pi/4$, а кінцева дуга – $3\pi/4$.

```
sage: show(disk((-1,-1),4,(pi/4,3*pi/4)),aspect_ratio=1)
```

Графічний примітив *многокутник* будує функція `polygon`.

Приклад 3.10. Побудувати трикутник, якщо вершини задані координатами (1,2), (5,6), (5,0).

```
sage: polygon([[1,2], [5,6], [5,0]], rgbcolor=(1,0,1))
```

Приклад 3.11. Побудувати правильний шестикутник.

```
sage: L=[[cos(pi*i/3),sin(pi*i/3)] for i in range(6)];
sage: show(polygon(L,rgbcolor=(1,0,1)),aspect_ratio=1)
```

3.2. Побудова графіків функціональних залежностей

Для побудови графіків функціональних залежностей SAGE має такі функції:

`plot` – для функціональних залежностей, заданих аналітично у декартових координатах;

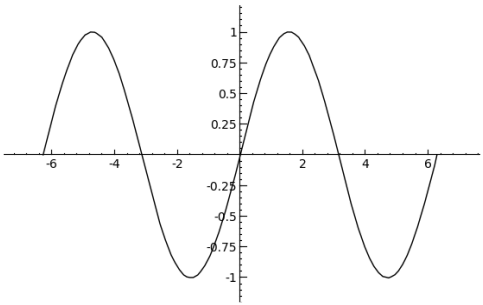
`parametric_plot` – для функціональних залежностей, заданих параметрично у декартових координатах;

`polar_plot` – для функціональних залежностей, заданих у полярних координатах.

Продемонструємо на прикладах можливості застосування даних функцій для побудови різноманітних графіків.

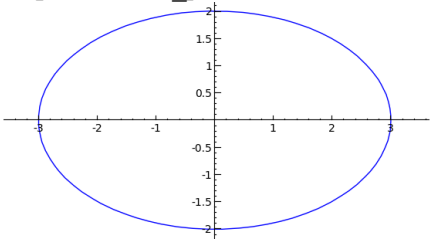
Приклад 3.12. Побудувати синусоїду на двох періодах.

```
sage: plot(sin(x),(-2*pi,2*pi),rgbcolor='black')
```



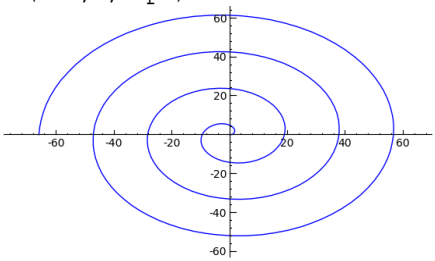
Приклад 3.13. Побудувати еліпс, заданий параметрично.

```
sage: t=var('t');parametric_plot((3*sin(t),2*cos(t)),0,2*pi)
```



Приклад 3.14. Побудувати спіраль Архімеда.

```
sage: polar_plot(3*x,0,7*pi)
```



Окрім того, SAGE має ряд специфічних функцій для графічного відображення числових даних та побудови графіків:

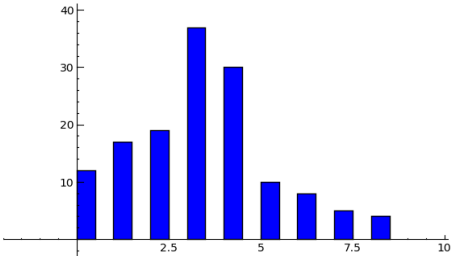
`bar_chart` – для побудови стовпчикової діаграми;

`contour_plot` – для побудови контурних ліній функції від двох змінних;

`plot_vector_field` – для побудови векторного поля для двох функції від двох змінних та інші.

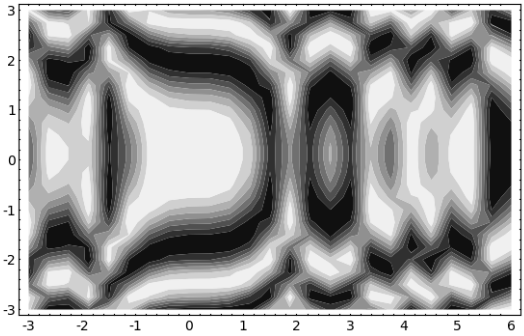
Приклад 3.15. Побудувати стовпчикову діаграму для вибірки 12, 17, 19, 37, 30, 10, 8, 5, 4.

```
sage: bar_chart([12, 17, 19, 37, 30, 10, 8, 5, 4])
```



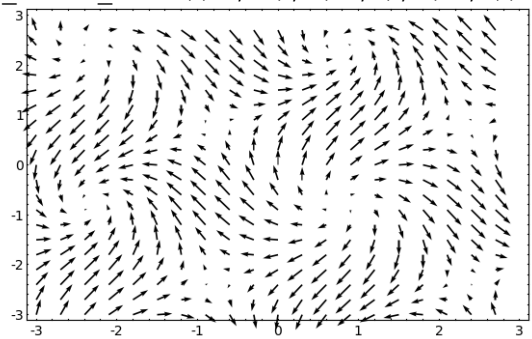
Приклад 3.16. Побудувати контурні лінії функції $f(x,y) = \cos(x^3+y^2)$.

```
sage: f(x,y)=cos(x^3+y^2)
sage: contour_plot(f,(-3,6),(-3,3))
```



Приклад 3.17. Побудувати векторне поле для функцій f_1=sin($x + y$) та f_2 = cos($x – y$).

```
sage: f1(x,y)=sin(x+y)
sage: f2(x,y)=cos(x-y)
sage: plot_vector_field((f1,f2), (-3,3), (-3,3))
```



3.3. Різноманітні побудови у просторі

Точку або множину точок у просторі можна визначити за допомогою функції `point3d`.

Приклад 3.18. Побудувати точку у просторі.

```
sage: point3d((-1,2,2),size=10)
```

Приклад 3.19. Побудувати дві зелені точки у просторі, надавши їм оптимального розміру.

```
sage: point3d(((-1,1,2),(1,-1,2)),size=5,rgbcolor='green')
```

Для побудови ламаної у просторі слід використати функцію `line3d`, яка має певні відмінності у застосуванні параметрів.

Приклад 3.20. Побудувати ламану у просторі по точках (1,2,3), (1,0,-2), (3,1,4), (2,1,-2).

```
sage: line3d([(1,2,3), (1,0,-2), (3,1,4), (2,1,-2)],\
      color='green',radius=0.02)
```

Виконати побудову сфери та правильних многогранників можна за

допомогою функцій sphere, tetrahedron, cube, octahedron, dodecahedron, icosahedron.

Для побудови у просторі поверхонь, заданих аналітично або параметрично, служать функції plot3d та parametric_plot3d відповідно.

Приклад 3.21. Побудувати гіперболоїд.

```
sage: u,v=var('u,v')
sage: plot3d(u^2-v^2, (u,-1,1), (v,-1,1),\
      plot_points=[50,50])
```

Приклад 3.22. Побудувати параболоїд.

```
sage: u,v=var('u,v')
sage: parametric_plot3d([u*cos(v),u*sin(v),u^2],\
      (u,0,1),(v,0,2*pi+0.4))
```

Приклад 3.23. Побудувати конус.

```
sage: u,v=var('u,v')
sage: parametric_plot3d([u*cos(v),u*sin(v),u],\
      (u,-1,0), (v, 0, 2*pi+0.5))
```

Приклад 3.24. Побудувати «серце».

```
sage: u,v=var('u,v')
sage: f_x = cos(u)*(4*sqrt(1-v^2)*sin(abs(u))^abs(u))
sage: f_y = sin(u) *(4*sqrt(1-v^2)*sin(abs(u))^abs(u))
sage: f_z = v
sage: parametric_plot3d([f_x,f_y,f_z],(u,-pi,pi), (v,-1,1),\
      frame=False, color='red')
```

Примітки:

1. Для побудови комбінації об'єктів достатньо поставити між ними знак «+»:

```
sage: plot(sin(x),(-2*pi,2*pi),rgbcolor='black')+\
          plot(cos(x),(-2*pi,2*pi),rgbcolor=hue(0.3))
```

2. Додати підписи до графічних зображень можна як у площині, так і у просторі, використовуючи функцію text.

3. Докладніше про можливі параметри графічних функцій можна дізнатися із контекстної довідки:

```
sage: arrow?
```

4. Параметри графічних примітивів за замовчуванням можна отримати командою назва_примітиву.options. Наприклад:

```
sage: circle.options
{'alpha':1, 'thickness':1, 'rgbcolor':(0,0,1), 'fill':False}
```

5. Для збереження двовимірних побудов у файлі звертайтесь до контекстного меню зображення.

6. Анімувати зображення (організувати в ньому циклічну зміну побудов) можна за допомогою функції animate.

4. РОЗВ'ЯЗАННЯ РІВНЯНЬ ТА СИСТЕМ РІВНЯНЬ

Розв'язати алгебраїчне рівняння або систему таких рівнянь у SAGE можна використовуючи функцію `solve`.

Приклад 4.1. Розв'язати рівняння $x^2 - 1 = 0$.

```
sage: solve(x^2-1==0,x)
```

або

```
sage: solve(x^2-1,x)
```

або

```
sage: (x^2-1).solve(x)
[x == -1, x == 1]
```

Приклад 4.2. Розв'язати рівняння $x^3 + 2x^2 - 4x - 5 = 0$.

```
sage: solve(x^3+2*x^2-4*x-5,x)
[x == (-sqrt(21) - 1)/2, x == (sqrt(21) - 1)/2, x == -1]
```

Приклад 4.3. Розв'язати систему рівнянь

$$\begin{cases} x^3 + y^3 = 19 \\ x^2 y + x y^2 = -6 \end{cases}$$

```
sage: x,y=var('x,y')
sage: solve([x^3+y^3==19,x^2*y+x*y^2==-6],x,y)
[[x == -2, y == 3], [x == 3, y == -2]]
```

Застосовуючи функцію `solve` до рівнянь, які не можуть бути розв'язані алгебраїчно, на виході отримаємо введене рівняння, як демонструє наступний приклад:

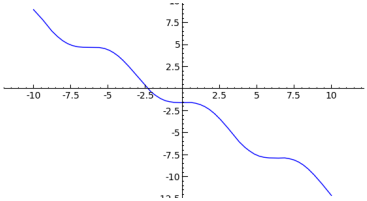
Приклад 4.4. Розв'язати рівняння $\sin x - x - \frac{\pi}{2} = 0$.

```
sage: eq=sin(x)-x-pi/2
sage: solve(eq==0)
[x == (2*sin(x) - pi)/2]
```

Якщо корінь рівняння не може бути визначений алгебраїчно, можна знайти його наближене значення графічно, побудувавши відповідний графік (чи графіки) за допомогою функції `plot` – відділити корінь, а потім чисельно, за допомогою функції `find_root`, уточнити корінь.

Приклад 4.5. Розв'язати рівняння $\sin x - x - \frac{\pi}{2} = 0$.

```
sage: plot(eq,-10,10)
```

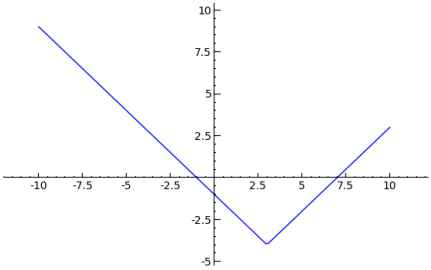


```
sage: find_root(eq,-5,0)
-2.309881460010057
```

Якщо рівняння має кілька коренів (графік відповідної функції перетинає вісь O*x* більше ніж в одній точці або графіки мають кілька точок перетину), функція `find_root` має бути застосована до кожного кореня окремо, з вказівкою відрізку, на якому був відділений корінь.

Приклад 4.6. Розв'язати рівняння $|x - 3| - 3 = 0$

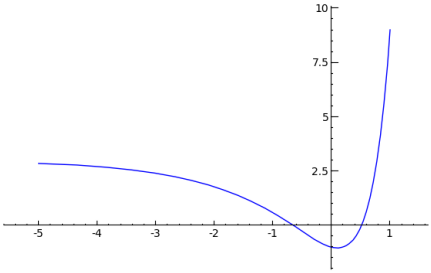
```
sage: solve(abs(x-3)-4,x)
[abs(x - 3) == 4]
sage: plot(abs(x-3)-4,-10,10)
```




```
sage: find_root(abs(x-3)-4,-2.5,0)
-1.0
```

Приклад 4.7. Розв'язати рівняння $2^{4x} - 5 \cdot 2^x + 3 = 0$.

```
sage: f=2^(4*x)-5*2^x+3
sage: plot(f,-5,1)
```




```
sage: print 'x1=',find_root(f,-1,0)
sage: print 'x2=', find_root(f,0,1)
x1= -0.662266966896
x2= 0.511202342796
```

5. ЛІНІЙНА АЛГЕБРА

Основними об'єктами лінійної алгебри є вектори та матриці. Продемонструємо на прикладах способи задання векторів та матриць в SAGE, а також виконання основних операцій над ними.

Приклад 5.1. Задати вектори $a(-1,5,3)$ та $b(2,-2,0)$.

```
sage: a=vector([-1,5,3]);a
(-1, 5, 3)
sage: b=vector([2,-2,0]);b
(2, -2, 0)
```

Приклад 5.2. Над векторами a і b виконати такі операції:

а) знайти суму векторів a і b;

б) визначити вектор c за правилом $3a-2b$;

в) обчислити скалярний добуток векторів a і b;

г) обчислити векторний добуток векторів a і b;

д) обчислити довжини векторів a і b.

Розв'язання

а)

```
sage: a+b
(1, 3, 3)
```

б)

```
sage: c=3*a-2*b; c
(-7, 19, 9)
```

в)

```
sage: a*b
-12
```

г)

```
sage: a.cross_product(b)
(6, 6, -8)
```

д)

```
sage: abs(a)
sqrt(35)
sage: abs(b)
2*sqrt(2)
```

Розмірність вектора поповертає функція `len`.

Приклад 5.3. Задати матрицю $M = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$.

```
sage: M=matrix([[1,2],[3,4],[5,6]]);M
[1 2]
[3 4]
[5 6]
```

Приклад 5.4. Задати матриці:
а) одиничну, розмірності 4;
б) нульову, розмірності 3×5;
в) діагональну, діагональними елементами якої є числа –7 і 3;
г) матрицю довільної розмірності, заповнену випадковими цілими числами.

Розв'язання

а)
```
sage: IM=identity_matrix(4);IM
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```
б)
```
sage: ZM=zero_matrix(3,5);ZM
[0 0 0 0 0]
[0 0 0 0 0]
[0 0 0 0 0]
```
в)
```
sage: DM=diagonal_matrix([-7,3]);DM
[-7  0]
[ 0  3]
```
г)
```
sage: RM=random_matrix(ZZ,5,3);RM
[ 1 -3  7]
[ 3 -2  1]
[-1  2 -1]
[-1  1 -1]
[ 4 10 -1]
```
Приклад 5.5. Виконати множення матриці:
а) на скаляр;
б) на вектор *а*;
в) на вектор з координатами (–3;4);
г) на одиничну матрицю;
д) на діагональну матрицю.

Розв'язання

а)
```
sage: 3*RM
[ 3 -9 21]
[ 9 -6  3]
[-3  6 -3]
[-3  3 -3]
[12 30 -3]
```

б)
```
sage: M*a
Traceback (most recent call last):
...
TypeError: unsupported operand parent(s) for '*': 'Full
MatrixSpace of 3 by 2 dense matrices over Integer Ring' and
'Ambient free module of rank 3 over the principal ideal
domain Integer Ring'
```

В даному випадку система видає повідомлення про неможливість виконання операції множення матриці на вектор через невідповідність розмірностей матриці та вектора.

в)
```
sage: M*vector([-3,4])
(5, 7, 9)
```

г)
```
sage: M*IM
Traceback (most recent call last):
...
TypeError: unsupported operand parent(s) for '*': 'Full
MatrixSpace of 3 by 2 dense matrices over Integer Ring' and
'Full MatrixSpace of 4 by 4 dense matrices over Integer
Ring'
```

В даному випадку система повідомляє про неможливість виконання операції множення матриць через невідповідність їх розмірностей.

д)
```
sage: M*DM
[ -7   6]
[-21  12]
[-35  18]
```

Приклад 5.6. Задати довільну квадратну матрицю над кільцем цілих чисел та обчислити її детермінант.
```
sage: A=random_matrix(ZZ,4,4);A
[   1   97   -1 -131]
[   1   -1    2  -28]
[  -2    1  -13   -2]
[  -7   -1   -6    1]
sage: det(A)
-224895
```

Приклад 5.7. Задати довільну матрицю над кільцем раціональних чисел та транспонувати її.
```
sage: B=random_matrix(QQ,2,4);B
[  0   2   0   0]
```

```
[  0 1/2   1   2]
sage: B.transpose()
[  0   0]
[  2 1/2]
[  0   1]
[  0   2]
```

Приклад 5.8. Для матриць $\begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & -3 & -1 & 1 \\ 2 & -5 & 1 & 7 \end{pmatrix}$ та $\begin{pmatrix} 1 & -1 & 2 & 3 \\ 0 & 1 & 3 & 4 \\ -2 & 5 & 1 & -4 \\ 0 & 0 & 2 & -1 \end{pmatrix}$ знайти обернені матриці, якщо це можливо.

Розв'язання

```
sage: C=matrix([[1,1,1,1],[2,2,2,2],[3,-3,-1,1],\
      [2,-5,1,7]]);C
[ 1  1  1  1]
[ 2  2  2  2]
[ 3 -3 -1  1]
[ 2 -5  1  7]
sage: C^-1
Traceback (most recent call last):
...
ZeroDivisionError: input matrix must be nonsingular
```

В даному випадку система повідомляє про помилку виконання операції пошуку матриці, оберненої до даної, оскільки її детермінант дорівнює 0.

```
sage: D=matrix([[1,-1,2,3],[0,1,3,4],[-2,5,1,-4],\
      [0,0,2,-1]]);D
[ 1 -1  2  3]
[ 0  1  3  4]
[-2  5  1 -4]
[ 0  0  2 -1]
sage: D.inverse()
[ 31/12  -11/8  19/24 -11/12]
[ 11/12   -3/8  11/24  -7/12]
[ -1/12    1/8  -1/24   5/12]
[  -1/6    1/4  -1/12   -1/6]
```

6. ПОЧАТКИ АНАЛІЗУ

SAGE має потужний інструментарій для виконання основних операцій математичного аналізу, а саме:

`limit` – обчислення границі послідовності чи функції;

`sum` – обчислення скінченої та нескінченної суми послідовності;

`prod` – обчислення скінченого та нескінченого добутку послідовності;

`diff (derivative)` – визначення похідної функції;

`integral (integrate)` – визначення первісної та обчислення визначеного інтегралу;

`taylor` – розклад функції у ряд Тейлора.

Застосування даних функцій до розв'язування конкретних задач продемонструємо на прикладах.

Приклад 6.1. Обчислити границю послідовності $\left\{1+\frac{1}{2^k}\right\}$.

```
sage: k=var('k'); limit(1+1/2^k,k=10)
1025/1024
sage: RR(_)
1.00097656250000
```

Знаком _ позначається результат виконання останньої команди.

Приклад 6.2. Обчислити границю функції $f(x)=\left(1+\frac{1}{x}\right)^x$, коли $x \to \infty$, $x \to 5$ та $x \to 1{,}2$.

```
sage: f=(1+1/x)^x
sage: f.limit(x=oo)
e
sage: f.limit(x=5)
7776/3125
sage: RR(_)
2.48832000000000
sage: f.limit(x=1.2)
2.069615754672029
```

Приклад 6.3. Обчислити ліву та праву границі функції $y=\frac{\sqrt{7+x}-1}{x^2-4}$ у точці $x=2$.

```
sage: limit((sqrt(7+x)-1)/(x^2-4),x=2,dir='plus')
+Infinity
sage: limit((sqrt(7+x)-1)/(x^2-4),x=2,dir='minus')
-Infinity
```

Приклад 6.4. Обчислити суму $\sum_{k=1}^{50} \frac{1}{k(k+1)}$.

```
sage: sum([1/(k*(k+1)) for k in [1..50]])
50/51
```

Приклад 6.5. Обчислити добуток $\prod_{k=3}^{10} \frac{k^3-4}{k^2-1}$.

```
sage: prod([(k^3-4)/(k^2-1) for k in [3..10]])
463185169513/254016
sage: RR(prod([(k^3-4)/(k^2-1) for k in [3..10]]))
1.82344879658368e6
```

Приклад 6.6. Обчислити похідну функції $y = 2e^x \ln^2(x+1)$.

```
sage: diff(2*exp(x)*(ln(x+1))^2)
2*e^x*log(x + 1)^2 + 4*e^x*log(x + 1)/(x + 1)
```

Приклад 6.7. Обчислити першу та другу похідні для функції $y = \cos^2(x^2)$.

```
sage: diff(cos(x^2)**2)
-4*x*cos(x^2)*sin(x^2)
```

Перший спосіб обчислення другої похідної:

```
sage: diff(diff(cos(x^2)**2))
8*x^2*sin(x^2)^2 - 4*cos(x^2)*sin(x^2) - 8*x^2*cos(x^2)^2
```

Другий спосіб обчислення другої похідної:

```
sage: diff(cos(x^2)**2,2)
8*x^2*sin(x^2)^2 - 4*cos(x^2)*sin(x^2) - 8*x^2*cos(x^2)^2
```

Приклад 6.8. Обчислити значення похідної функції $f(x) = -\frac{1}{3}x^3 + \frac{3}{2}x^2 - 2x + 1$ в точці $x = 3$.

```
sage: df(x)=diff(-1/3*x^3+3/2*x^2-2*x+1); df(x)
-x^2 + 3*x - 2
sage: df(3)
-2
```

Приклад 6.9. Обчислити частинні похідні функції $f(x,y) = \frac{\sin x}{\cos 2y}$.

```
sage: diff(sin(x)/cos(2*y),x)
cos(x)/cos(2*y)
sage: diff(sin(x)/cos(2*y),y)
2*sin(x)*sin(2*y)/cos(2*y)^2
```

Приклад 6.10. Знайти первісну для функції $y = \frac{\arccos^2(7x)}{\sqrt{1-49x^2}}$.

```
sage: integral((acos(7*x)^2)/sqrt(1-49*x^2))
-arccos(7*x)^3/21
```

Приклад 6.11. Обчислити визначений інтеграл $\int_0^{\pi/2} \sin x \cos^2 x dx$.

```
sage: integral(sin(x)*cos(x)^2,x,0,pi/2)
1/3
```

Приклад 6.12. Обчислити подвійний інтеграл $\int_{-1}^{0} \int_{-y/2}^{-y} (2xy - x^2) dxdy$.

```
sage: integral(integral(2*x*y-x^2,x,-y/2,-y),y,-1,0)
-25/96
```

Приклад 6.13. Розкласти в околі точки $x = 1$ в ряд Тейлора функцію $y = e^{-x^2}$ та вивести перші 5 членів ряду.

```
sage: taylor(exp(-x^2),x,1,5)
e^-1 - 2*e^-1*(x - 1) + e^-1*(x - 1)^2 + 2*(x - 1)^3/(3*e) -
5*(x - 1)^4/(6*e) + (x - 1)^5/(15*e)
```

Приклад 6.14. Розкласти в ряд Тейлора функцію $y = \sin(\sin x)$.

```
sage: taylor(sin(sin(x)),x,0,7)
x - x^3/3 + x^5/10 - 8*x^7/315
```

7. ДИФЕРЕНЦІЙНІ РІВНЯННЯ

Для розв'язування звичайних диференційних рівнянь та їх систем аналітичним або чисельним методом SAGE має кілька функцій: `desolve`, `desolve_laplace`, `desolve_system` та ін.

Функція `desolve` виконує пошук загального розв'язку звичайного диференційного рівняння першого та другого порядків.

Функція `desolve_laplace` розв'язує задачу Коші для звичайного диференційного рівняння, використовуючи перетворення Лапласа.

Функція `desolve_system` розв'язує задачу Коші для системи довільної кількості звичайних диференційних рівнянь першого порядку.

Приклад 7.1. Розв'язати диференційне рівняння першого порядку:

$$y' - \frac{y}{x} = 0 .$$

```
sage: x = var('x')
sage: y = function('y',x) # визначення y як функції від x
# визначення de як безіменної (лямбда) функції, що містить
# ліву частину рівняння
sage: de = lambda(p): diff(y,x) - y/x
# знаходження кореня функції de як загального розв'язку
# диференційного рівняння
sage: desolve(de(y(x)),[y,x])
'%c*x'
```

Приклад 7.2. Розв'язати лінійне диференційне рівняння першого порядку

$$y' - \frac{y}{x} - x = 0 .$$

```
sage: x = var('x')
sage: y = function('y',x)
sage: de = lambda(p): diff(y,x) -y/x -x
sage: desolve(de(y(x)),[y,x])
'x*(x+%c)'
```

Приклад 7.3. Розв'язати диференційне рівняння другого порядку:

$$y'' - 1 + 2x = 0 .$$

```
sage: x = var('x')
sage: y = function('y',x)
sage: de = lambda(p): diff(y,x,2)-1+2*x
sage: desolve(de(y(x)),[y,x])
'-(2*x^3-3*x^2)/6+%k2*x+%k1'
```

Приклад 7.4. Розв'язати лінійне однорідне диференційне рівняння другого порядку зі сталими коефіцієнтами:

$$y'' - 5y' + 6y = 0\,.$$

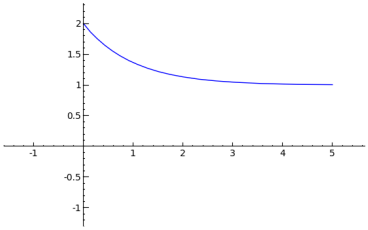
```
sage: x = var('x')
sage: y = function('y',x)
sage: de = lambda(p): diff(y,x,2)-5*diff(y,x)+6*y
sage: desolve(de(y(x)),[y,x])
'%k1*%e^(3*x)+%k2*%e^(2*x)'
```

Приклад 7.5. Знайти розв'язок задачі Коші для диференційного рівняння першого порядку:

$$y' + y = 1, y(0) = 2$$

та проілюструвати знайдений розв'язок.

```
sage: x = var('x')
sage: y = function('y',x)
sage: de = lambda(p): diff(y,x)+y-1
sage: sol=desolve_laplace(de(y(x)),["x","y"],[0,2]);sol
'%e^-x+1'
sage: plot(exp(-x)+1,0,5)
```



Приклад 7.6. Знайти розв'язок задачі Коші для диференційного рівняння другого порядку:

$$y'' + y = 0, y(0) = 0, y'(0) = 1.$$

```
sage: x = var('x')
sage: y = function('y', x)
sage: de=lambda(p): diff(y,x,x)+y
sage: desolve_laplace(de(y(x)),["x","y"],[0,0,1])
'sin(x)'
```

Приклад 7.7. Розв'язати задачу Коші для системи диференційних рівнянь

$$\begin{cases} x_t' + y = 1 \\ y_t' - x = -1 \end{cases},$$

якщо $x(0) = 1, y(0) = -1$.

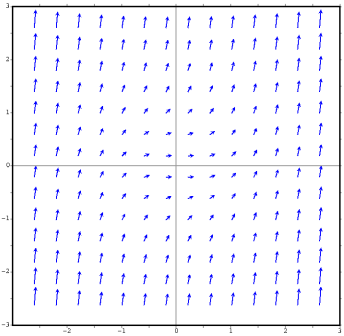
```
sage: t = var('t')
sage: x = function('x', t)
sage: y = function('y', t)
sage: de1 = lambda z: diff(x,t) + y - 1
sage: de2 = lambda z: diff(y,t) - x + 1
```

```
sage: des = [de1([x(t),y(t)]),de2([x(t),y(t)])]
sage: vars = ["t","x","y"]
sage: desolve_system(des,vars)
['(1-y(0))*sin(t)+(x(0)-1)*cos(t)+1', '(x(0)-
1)*sin(t)+(y(0)-1)*cos(t)+1']
sage: desolve_system(des,vars,[0,1,-1])
['2*sin(t)+1', '1-2*cos(t)']
```

Оскільки поточна версія системи SAGE не має спеціальної функції для ілюстрації розв'язку диференційного рівняння, будемо використовувати для побудови характеристичного векторного поля для даного диференційного рівняння спеціальні функції пакету Maxima.

Приклад 7.8. Побудувати характеристичне векторне поле для диференційного рівняння $y' = x^2 + y^2$.

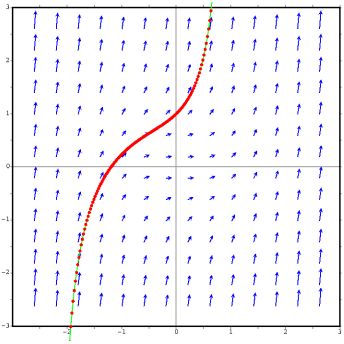
```
sage: maxima.eval('load("plotdf")')
sage: maxima.eval('plotdf(x^2+y^2,[x,-3,3],[y,-3,3])')
```



Приклад 7.9. Проілюструвати розв'язок задачі Коші для диференційного рівняння

$$y' = x^2 + y^2, \text{ якщо } y(0) = 1.$$

```
sage: maxima.eval('load("plotdf")')
sage: maxima.eval('plotdf(x^2+y^2,[trajectory_at,0,1],/
      [x,-3,3],[y,-3,3])')
```



8. ЕЛЕМЕНТИ КОМБІНАТОРИКИ

Даний розділ присвячений огляду функцій системи SAGE для розв'язання основних задач комбінаторики та математичної статистики.

Набір всіх перестановок з елементів заданої множини поповертає функція `permutations`, кількість перестановок – `number_of_permutations`.

Приклад 8.1. Визначити набір всіх перестановок з елементів множини [2,5,7,9] та підрахувати їх кількість.

```
sage: permutations([2,5,7,9])
[[2, 5, 7, 9], [2, 5, 9, 7], [2, 7, 5, 9], [2, 7, 9, 5],
[2, 9, 5, 7], [2, 9, 7, 5], [5, 2, 7, 9], [5, 2, 9, 7],
[5, 7, 2, 9], [5, 7, 9, 2], [5, 9, 2, 7], [5, 9, 7, 2],
[7, 2, 5, 9], [7, 2, 9, 5], [7, 5, 2, 9], [7, 5, 9, 2],
[7, 9, 2, 5], [7, 9, 5, 2], [9, 2, 5, 7], [9, 2, 7, 5],
[9, 5, 2, 7], [9, 5, 7, 2], [9, 7, 2, 5], [9, 7, 5, 2]]
sage: number_of_permutations([2,5,7,9])
24
```

Приклад 8.2. Визначити набір всіх перестановок з елементів множини ['f', 'g', 's'] та підрахувати їх кількість.

```
sage: mset=['f','g','s']
sage: permutations(mset)
[['f', 'g', 's'], ['f', 's', 'g'], ['g', 'f', 's'],
['g', 's', 'f'], ['s', 'f', 'g'], ['s', 'g', 'f']]
sage: number_of_permutations(mset)
6
```

Набір розміщень (перестановок з урахуванням порядку) без повторень визначає функція `arrangements`. Кількість наборів розміщень поповертає функція `number_of_arrangements`.

Приклад 8.3. Визначити набір розміщень без повторень, взявши по 2 елементи з множини [2,5,7,9] та підрахувати їх кількість.

```
sage: mset=[2,5,7,9]
sage: arrangements(mset,2)
[[2, 5], [2, 7], [2, 9], [5, 2], [5, 7], [5, 9], [7, 2],
[7, 5], [7, 9], [9, 2], [9, 5], [9, 7]]
sage: number_of_arrangements(mset,2)
12
```

Набір розміщень з повтореннями поповертає функція `permutations_iterator`. Спеціальної функції для підрахунку кількості розміщень з повтореннями немає.

Приклад 8.4. Визначити набір розміщень з повтореннями, взявши по 3 елементи з множини [2,5,7,9].

```
sage: X=permutations_iterator(mset,3)
```

```
sage: [x for x in X]
[[2, 5, 7], [2, 5, 9], [2, 7, 5], [2, 7, 9], [2, 9, 5],
[2, 9, 7], [5, 2, 7], [5, 2, 9], [5, 7, 2], [5, 7, 9],
[5, 9, 2], [5, 9, 7], [7, 2, 5], [7, 2, 9], [7, 5, 2],
[7, 5, 9], [7, 9, 2], [7, 9, 5], [9, 2, 5], [9, 2, 7],
[9, 5, 2], [9, 5, 7], [9, 7, 2], [9, 7, 5]]
```

Набір комбінацій (перестановок без урахування порядку) без повторень повертає функція `combinations`. Визначити кількість таких комбінацій повертає функція `number_of_combinations`.

Приклад 8.5. Визначити набір комбінацій без повторень, взявши по 2 елементи з множини [2,5,7,9] та підрахувати їх кількість.

```
sage: combinations(mset,2)
[[2, 5], [2, 7], [2, 9], [5, 7], [5, 9], [7, 9]]
sage: number_of_combinations(mset,2)
6
```

Набір комбінацій (сполук) з повтореннями повертає функція `combinations_iterator`. Спеціальної функції для підрахунку кількості комбінацій з повтореннями немає.

Приклад 8.6. Визначити набір комбінацій з повтореннями, взявши по 3 елементи з множини [2,5,7,9].

```
sage: X=combinations_iterator(mset,3)
sage: [x for x in X]
[[2, 5, 7], [2, 5, 9], [2, 7, 9], [5, 7, 9]]
```

Набір кортежів повертає функція `tuples`. Кількість кортежів визначає `number_of_tuples`.

Приклад 8.7. Визначити набір кортежів, взявши по 2 елементи з множини [2,5,7,9] та підрахувати їх кількість.

```
sage: tuples(mset,2)
[[2, 2], [5, 2], [7, 2], [9, 2], [2, 5], [5, 5], [7, 5],
[9, 5], [2, 7], [5, 7], [7, 7], [9, 7], [2, 9], [5, 9],
[7, 9], [9, 9]]
sage: number_of_tuples(mset,2)
16
```

Функція `fibonacci(n)` повертає *n*-й елемент послідовності чисел Фібоначі.

Приклад 8.8. Визначити 27-й елемент послідовності чисел Фібоначі.

```
sage: fibonacci(27)
196418
```

Приклад 8.9. Показати перші 27 елементів послідовності чисел Фібоначі.

```
sage: [fibonacci(i) for i in range(28)]
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377,
610, 987, 1597, 2584, 4181, 6765, 10946, 17711, 28657,
46368, 75025, 121393, 196418]
```

Функція `euler_number(n)` повертає *n*-й елемент послідовності чисел Ейлера.

Приклад 8.10. Визначити 8-й елемент послідовності чисел Ейлера.

```
sage: euler_number(8)
1385
```

Приклад 8.11. Показати перші 8 елементів послідовності чисел Ейлера.

```
sage: [euler_number(i) for i in range(9)]
[1, 0, -1, 0, 5, 0, -61, 0, 1385]
```

9. ЕЛЕМЕНТИ ПРОГРАМУВАННЯ

Оскільки у попередніх розділах при наведенні прикладів застосування окремих функцій мали місце елементи програмування, даний розділ присвячений огляду основних можливостей програмування мовою Python у середовищі SAGE. Зазначимо, що за рахунок написання програмних кодів суттєво розширюється не лише коло задач, які можна розв'язати у даному середовищі, а й саме середовище (див. додаток Д).

Програма, написана будь-якою мовою програмування, насамперед оперує *змінними*.

При написанні програмних кодів у середовищі SAGE оголошення змінних виконується за допомогою ключового слова `var`:

```
x,y = var('x y').
```

Для надання змінним певного значення використовується символ `=`:

```
a = 2, b = 2/5, c = 3.7, d=[1,4,5], s = "Hello, world!".
```

Наведений приклад демонструє надання змінним значень типу ціле число, раціональне число, дійсне число, множина та рядок відповідно.

Вивести значення змінної на екран можна одним із способів: або просто зазначити її ім'я, або застосувати оператор `print`:

```
print s.
```

Для виведення значень кількох змінних у стовпчик (без використання функції `print`) їх слід розділити крапкою з комою:

```
a;b;c;d;s.
```

Для виведення значень кількох змінних у рядок із застосуванням функції `print` змінні слід розділити комою:

```
print a,b,c,d,s.
```

Для роз'яснення складного тексту програми до нього включають коментарі. Для виділення коментарів довжиною не більше одного рядка служить символ #:

```
x =2 #Set the variable x equal to 2.
```

Виділити текст коментарів у кілька рядків можна послідовністю символів `"""`:

```
"""
Приклад багаторядкового коментаря.
Наступний код записує число 3 у змінну x, а потім друкує
значення змінної x.
"""
x =3
print x
```

Як і всі мови програмування SAGE забезпечує можливість прийняття рішення (виконання розгалуження) і для цього використовує оператор `if`. Синтаксис застосування оператору `if` у неповній формі такий:

```
if <логічний вираз>:
   <твердження 1>
   <твердження 2>
...
```

На відміну від інших мов, форматний відступ після двокрапки є обов'язковим: відступи використовуються для запису групових інструкцій – блоків, причому всередині одного блоку відступ має бути однаковим.

Логічний вираз може бути простим, що містить лише допустимі оператори порівняння (==,<>,!=,<,<=,>,>=):

```
x =6
if x >5:
   print x
   print "Greater"
```

або складеним із простих за допомогою логічних операцій and, or та not:

```
a=7
b=9
if a>5 and b<10:
    print "These expressions are both true."
if a<5 or b<10:
   print "At least one of these expressions is true."
print not a>5.
```

Синтаксис повної форми оператора if:

```
if <логічний вираз>:
   <твердження 1>
else:
   <твердження 2>
```

Якщо розгалуження передбачає більше двох можливих варіантів, оператор if застосовують у такому форматі:

```
if <умова1>:
   <твердження 1>
elif <умова2>:
   <твердження 2>
elif <умова3>:
   <твердження 3>
...
else:
   <твердження N>
```

Циклічні повторення у програмі SAGE забезпечують конструкції while та for.

Синтаксис організації циклу з передумовою while такий:

```
while <логічний вираз>:
```

```
   <твердження 1>
   <твердження 2>
...
```

Приклад 9.1. Вивести елементи списку, використовуючи цикл з передумовою while.

```
sage: x=[50,51,52,53,54,55,56,57,58,59]
sage: i=0
sage: while i<=9:
          print x[i],
          i=i+1
50 51 52 53 54 55 56 57 58 59
```

Більш зручним для роботи з елементами списку є цикл for:

```
for <змінна> in <список>:
   <твердження 1>
   <твердження 2>
   ...
```

Оператор in виконує автоматичний пошук елемента у списку (перевіряє належність елемента списку). Якщо елемент знайдений у списку, оператор in повертає True, в противному разі – False:

```
sage: print 53 in [50,51,52,53,54,55,56,57,58,59]
sage: print 75 in [50,51,52,53,54,55,56,57,58,59]
True
False
```

Для перевірки неналежності елемента списку використовується оператор not in:

```
sage: print 53 not in [50,51,52,53,54,55,56,57,58,59]
sage: print 75 not in [50,51,52,53,54,55,56,57,58,59]
False
True
```

Приклад 9.2. Вивести елементи списку, використовуючи цикл for.

```
sage: for x in [50,51,52,53,54,55,56,57,58,59]:
          print x
50
51
52
...
```

Впорядкований цілочисельний список можна отримати за допомогою функції range:

```
sage: range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
sage: range(3,10)
[3, 4, 5, 6, 7, 8, 9]
sage: range(3,10,2)
```

```
[3, 5, 7, 9]
sage: range(10,3,-1)
[10, 9, 8, 7, 6, 5, 4]
sage: for i in range(11):
          print i,
0 1 2 3 4 5 6 7 8 9 10
```

В модулі numarray визначена функція arange, параметрами якої можуть бути й дійсні числа:

```
# from numarray import *
sage: arange(0,1,0.1)
array([0., 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])
```

Для багатократного використання програмного коду його оформлюють у функцію за таким правилом:

```
def <ім'я_функції>(arg1,arg2,...argN):
   <твердження 1>
   <твердження 2>
   ...
```

Твердження def – заголовок функції – містить ім'я функції, яке складається з латинських літер і не допускає наявності пропусків, та параметри функції, зазначені у круглих дужках.

Приклад 9.3. Визначити функцію з іменем addnums, аргументами якої будуть два числа. Функція повинна виконувати додавання чисел та повертати їх суму.

```
sage: def addnums(num1,num2):
          answer = num1 + num2
          return answer
#Виконаємо виклик функції для додавання чисел 4 і 5:
sage: addnums(4,5)
9
```

Визначення рекурсивних функцій не має жодних особливостей.

Приклад 9.4. Створіть аналог функції fibonacci.

```
sage: def fib(n):
        if n<1:
          return "Функція не визначена"
        elif n==1 or n==2:
          return 1
        else:
          return fib(n-1)+fib(n-2)

sage: fib(27)
196418
```

Написання програмних кодів засобами SAGE та оформлення їх у функції незамінні при розробці інтерактивних демонстрацій (додаток В).

ЛІТЕРАТУРА

1. Семеріков С.О. Maxima 5.13: довідник користувача / За ред. академіка АПН України М.І. Жалдака. – К.: 2007. – 48 с.
2. Granville, W.A. Differential Calculus and SAGE. – 2008. – X+275 p.
3. Joyner, D. Constructions in Sage: A Sage cookbook. – 2008. – VI+98 p.
4. Joyner, D., Stein, W. Sage Installation Guide. – 2008. – 15 p.
5. Joyner, D. Introductory Differential Equations using SAGE. – 2007. – VIII+216 p.
6. Kohel, D.R. Cryptography. – 2007. – II +136 p.
7. Maxima 5.16.3 Manual.
8. Stein, W. Sage Reference Manual. – 2008. – XII+3460 p.
9. Stein, W. Sage Tutorial: www.sagemath.org. – CreateSpace, 2008. – 100 p.
10. Stein, W., Joyner, D. Sage Programming Guide. – 2008. – 86 p.

ДОДАТКИ

А. Підготовка SAGE до роботи

А.1. Інсталяція

Для розгортання SAGE на сервері класу або локальному комп'ютері необхідно перш за все встановити Vmware Player (`http://www.vmware/download/player`). На рис. А.1 показано його розташування на DVD-додатку.

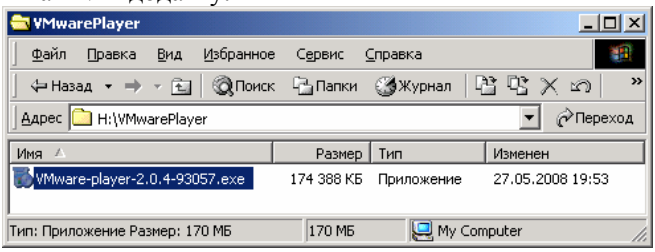


Рис. А.1

Розпаковка інсталяційного архіву відбувається на диск, що містить каталог для тимчасових файлів Temp (зазвичай це диск C), тому на ньому має бути близько 350 Мб вільного місця – у протилежному випадку інсталятор повідомить про його нестачу (рис. А.2).

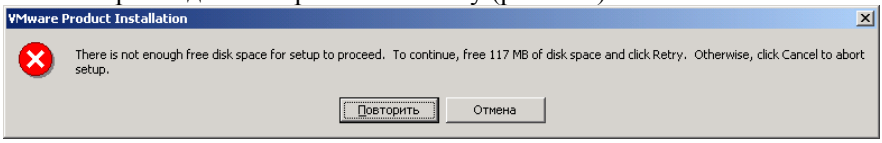


Рис. А.2

Для початку інсталяції необхідно натиснути Next >, обрати каталог для встановлення Vmware Player (рис. А.3), способи запуску програми (рис. А.4) та натиснути Install. Обираючи каталог для розміщення програми, впевніться, що на диску, де він розташований, вільно близько 270 Мб, інакше інсталятор запропонує обрати інший диск (рис. А.5).

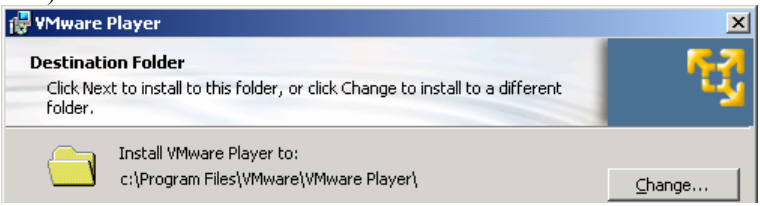


Рис. А.3

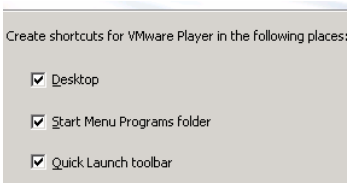


Рис. А.4

Рис. А.5

Після завершення інсталяції необхідно натиснути на кнопку Finish та погодитись із необхідністю перезавантаження операційної системи (рис. А.6).

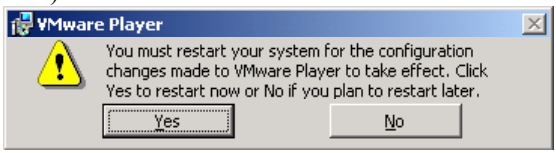


Рис. А.6

Останній крок – копіювання з DVD-додатку на будь-який диск, що містить 3–4 Гб вільного місця, каталогу sage-vmware-3.x.y, де x.y – номер підверсії (`http://sagemath.org/bin/microsoft_windows`); по завершенню копіювання необхідно зняти з каталогу та усіх файлів в ньому атрибут «Лише читання» (рис. А.7).

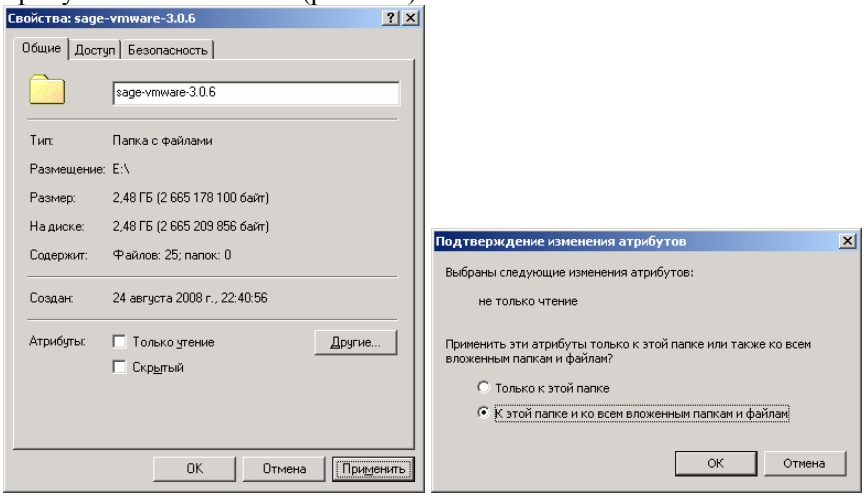


Рис. А.7

А.2. Запуск сервера

При першому запуску Vmware Player пропонується погодитись із ліцензійною угодою на безоплатне використання даного програмного

забезпечення необмежений час (рис. А.8).

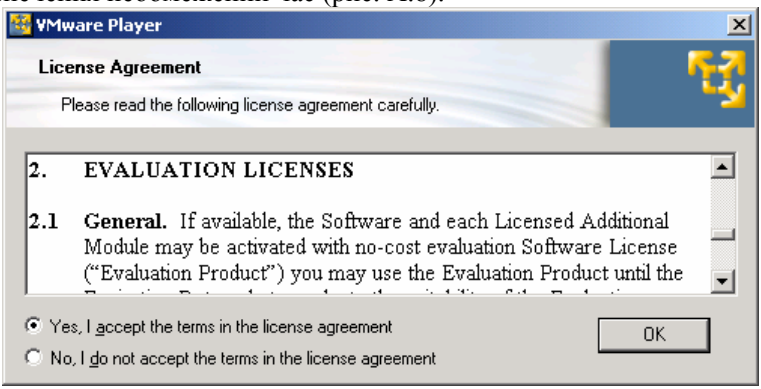


Рис. А.8

В головному вікні програми за допомогою пункту «Open» в скопійованому каталозі виберіть файл sage_vmx.vmx (рис. А.9).

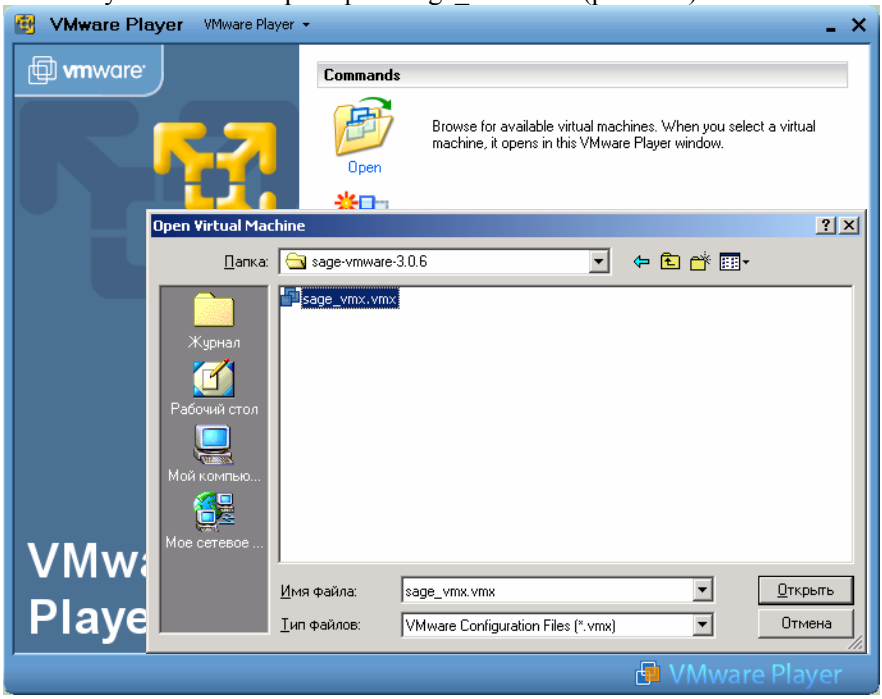


Рис. А.9

Після завантаження Linux у вікні Vmware Player необхідно перейти до нього натисканням Ctrl-G, у відповідь на запрошення «sage login:»

для запуску Web-СКМ ввести «notebook» та натиснути Enter і Ctrl-Alt (рис. А.10).

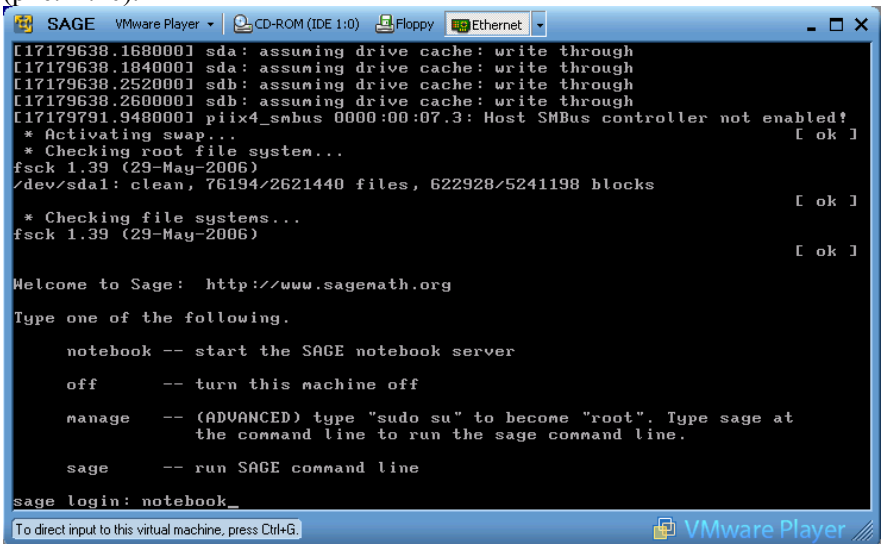


Рис. А.10

А.3. З'єднання з сервером

Запуск блокнотного серверу супроводжується виведенням вказівки стосовно IP-адреси, звернення до якої дасть доступ до Web-інтерфейсу SAGE. Цю адресу можна застосовувати як на тому ж комп'ютері, де працює сервер SAGE, так і в мережі комп'ютерного класу. На рис. А.11 показано приклад застосування виділеної IP-адреси для доступу до серверу SAGE з Web-браузера Firefox.

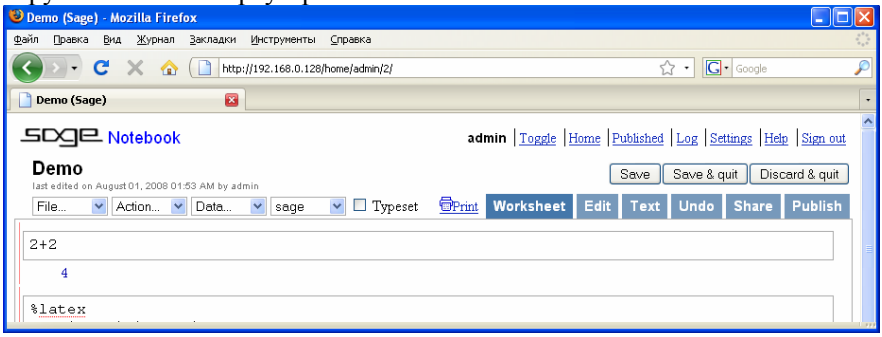


Рис. А.11

А.4. Зупинка серверу

Завершення роботу серверу виконується за необхідності одним з

двох способів:

1) вибором пункту Exit меню Vmware Player: блокнотний режим серверу при цьому зберігається, тому при наступному завантаженні файлу sage_vmx.vmx у відповідь на запрошення «sage login:» вводити «notebook» непотрібно;

2) подвійним натисканням Ctrl-C у вікні Vmware Player та введенням «off» у відповідь на запрошення «sage login:»: блокнотний режим серверу при цьому не зберігається.

Б. SAGE: «швидка допомога»

Команда SAGE	Інтерпретація
Базові команди	
<Shift>+<Enter>	обчислити вираз
com<tab>	видати список команд, які починаються з *com*
command?	надати контекстну довідку з команди *command*
command??	вивести програмний код команди *command*
_	звернутися до результату попередньої команди
%*package_name*	перейти до пакету з іменем *package_name*
Основні константи та функції	
π=pi, e=e,i=i, ∞=oo	
sin, cos, tan, sec, csc, cot, sinh, cosh, tanh, sech, csch, coth log, ln, exp, abs, sqrt, factorial, floor, ceil	
var('x_1 x_2 … x_n') або var('x_1,…,x_n')	означити $x_1,\ldots,x_n$ як символьні змінні
f(x)=x^2 f=lambda x: x^2 def f(x): return x^2	визначити функцію на прикладі $f(x)=x^2$
Операції над виразами	
factor(*exp*)	розкласти на множники
expand(*exp*)	розкрити дужки
simplify(*exp*)	звести подібні без розкриття дужок
RR(exp)	отримати числове значення виразу
Побудова графічних зображень	
point((x_1,y_1),…,(x_n,y_n)),*options*)	побудувати множину точок на площині

Команда SAGE	Інтерпретація
arrow((x$_1$,y$_1$),(x$_2$,y$_2$),*options*)	побудувати стрілку (вектор)
line([(x$_1$,y$_1$),…,(x$_n$,y$_n$)],*options*)	побудувати лінію на площині по точках, заданих у вигляді списку
polygon([(x$_1$,y$_1$),…,(x$_n$,y$_n$)],*options*)	побудувати зафарбований многокутник
circle((x,y),r,*options*)	побудувати коло
disk((x,y),r,(angle1,angle2),*options*)	побудувати сектор круга
plot(f(x),x$_{min}$,x$_{max}$,*options*) parametric_plot((f(t),g(t)), t$_{min}$,t$_{max}$,*options*) polar_plot(f(t),t$_{min}$,t$_{max}$,*options*)	побудувати графік функціональної залежності, заданої аналітично, параметрично та у полярних координатах
bar_chart([list_of_numeical_data])	побудувати стовпчикову діаграму
contour_plot(f,(xmin,xmax),(ymin,ymax),/ *options*)	побудувати контурні лінії для функції від двох змінних
plot_vector_field((f,g),(xmin,xmax),(ymin,ymax))	побудувати векторне полу для двох функцій від двох змінних
circle((1,1),1)+ line([(0,0),(2,2)])	задати комбінацію графіків
animate([об'єкт$_1$, … , об'єкт$_n$], *options*).show(delay=20)	із заданим інтервалом циклічно показати перераховані об'єкти
point3d((x$_1$,y$_1$,z$_1$),…,(x$_n$,y$_n$,z$_n$)),*options*)	побудувати множину точок у просторі
line3d([(x$_1$,y$_1$,z$_1$),…,(x$_n$,y$_n$,z$_n$)], *options*)	побудувати ламану у просторі
sphere((x,y,z),r,*options*)	побудувати сферу
tetrahedron((x,y,z),size,*options*)	побудувати тетраедр
cube((x,y,z),size,*options*)	побудувати куб
octahedron((x,y,z),size,*options*)	побудувати октаедр
dodecahedron((x,y,z),size,*options*)	побудувати додекаедр
icosahedron((x,y,z),size,*options*)	побудувати ікосаедр
plot3d(f(x,y), [x$_b$,x$_e$],[y$_b$,y$_e$],*options*) parametric_plot3d((f(t),g(t),h(t)), [t$_b$,t$_e$],*options*) list_plot3d([(x$_1$,y$_1$),…,(x$_n$,y$_n$)], *options*)	побудувати поверхню, задану функцією двох змінних, параметрично та переліком точок

Команда SAGE	Інтерпретація
Розв'язання рівнянь	
`f(x)==g(x)`	задати рівняння
`solve(f(x)==g(x),x)`	розв'язати рівняння аналітично
`solve([f(x,y)==0,g(x,y)==0],x,y)`	розв'язати систему рівнянь аналітично
`find_root(f(x),a,b)` або `find_root(f(x)==0,a,b)`	знайти наближено нулі функції або корені рівняння
Лінійна алгебра	
`vector([1,2])`	задати вектор
`matrix([[1,2,3],[4,5,6]])`	задати матрицю
`identity_matrix(n)`	задати одиничну матрицю розмірності *n*
`zero_matrix(n,m)`	задати нульову матрицю розмірності *nxm*
`diagonal_matrix([`a_{11}`,`a_{22}`,…,`a_{nn}`])`	задати діагональну матрицю з діагональними елементами $a_{11},a_{22},\ldots,a_{nn}$
`random_matrix(Ring,nrow,ncol)`	задати матрицю випадкових чисел з кільця *Ring* (ZZ, QQ, RR, CC)
`det(A)`	обчислити визначник (детермінант) матриці A
`inverse(A)` або `A^-1`	знайти матрицю, обернену до матриці A
`A.transpose()`	виконати транспонування матриці
Початки аналізу	

`limit(f(x),x=a)`
`limit(f(x),x=a,dir='minus'`
`limit(f(x),x=a,dir='plus')`
	обчислити границю
	обчислити ліву границю
	обчислити праву границю

`diff(f(x),x)` `derivative(f(x,y),x)`	обчислити похідну
`integral(f(x),x)` `integrate(f(x),x,a,b)`	обчислити інтеграл
`sum([f(i) for i in [k..n]])`	обчислити суму елементів ряду
`prod([f(i) for i in [k..n]])`	обчислити добуток елементів ряду
`taylor(f(x),x,`x_0`,n)`	виконати розкладання у ряд Тейлора

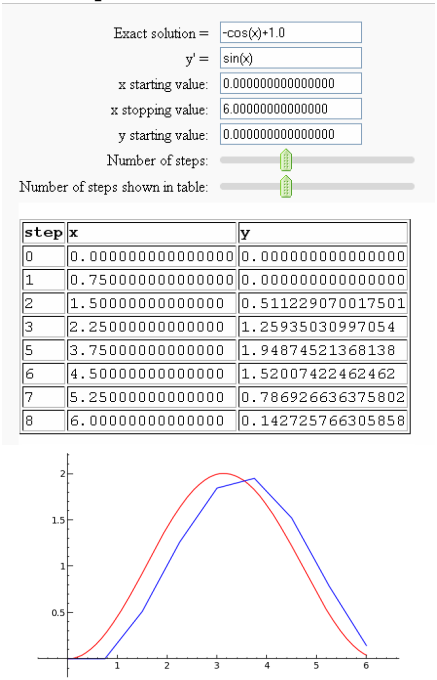
Команда SAGE	Інтерпретація
Диференційні рівняння	
`desolve(de,vars)`	знайти загальний розв'язок звичайного диференційного рівняння *de* відносно змінних *vars*
`desolve_laplace(de,vars,ics)`	знайти розв'язок задачі Коші для звичайного диференційного рівняння *de* за початкових умов *ics*, використовуючи перетворення Лапласа
`desolve_system(de,vars,ics)`	знайти розв'язок задачі Коші для системи звичайних диференційних рівнянь
Елементи комбінаторики	
`permutations(set)`	визначити множину перестановок
`arrangements(set,k)`	визначити множину розміщень без повторень
`permutations_iterator(set,k)`	визначити множину розміщень з повтореннями
`combinations(set,k)`	визначити множину комбінацій (сполук) без повторень
`combinations_iterator(set,k)`	визначити множину комбінацій (сполук) з повтореннями
`tuples(set,k)`	визначити множину кортежів
`number_of_permutations(set)`	обчислити кількість перестановок
`number_of_arrangements(set,k)`	обчислити кількість розміщень
`number_of_combinations(set,k)`	обчислити кількість комбінацій
`number_of_tuples(set,k)`	обчислити кількість кортежів
`fibonacci(n)`	визначити *n*-ий елемент послідовності чисел Фібоначі
`euler_number(n)`	визначити *n*-ий елемент послідовності чисел Ейлера

B. Інтерактивні демонстрації

B.1. Метод Ейлера для розв'язання диференційних рівнянь

```
def tab_list(y, headers = None):
    '''
    Форматуємо список у html-таблицю
    '''
    s = '<table border = 1>'
    if headers:
        for q in headers:
            s = s + '<th>' + str(q) + '</th>'
    for x in y:
        s = s + '<tr>'
        for q in x:
            s = s + '<td>' + str(q) + '</td>'
        s = s + '</tr>'
    s = s + '</table>'
    return s
var('x y')
@interact
def euler_method(y_exact_in = input_box('-cos(x)+1.0', type = \
str, label = 'Exact solution = '), y_prime_in = \
input_box('sin(x)', type = str, label = "y' = "), start = \
input_box(0.0, label = 'x starting value: '), stop = \
input_box(6.0, label = 'x stopping value: '), startval = \
input_box(0.0, label = 'y starting value: '), nsteps = \
slider([2^m for m in range(0,10)], default = 10, label = \
'Number of steps: '), show_steps = slider([2^m for m in \
range(0,10)], default = 8, label = \
'Number of steps shown in table: ')):
    y_exact = lambda x: eval(y_exact_in)
    y_prime = lambda x,y: eval(y_prime_in)
    stepsize = float((stop-start)/nsteps)
    steps_shown = max(nsteps,show_steps)
    sol = [startval]
    xvals = [start]
    for step in range(nsteps):
        sol.append(sol[-1]+stepsize*y_prime(xvals[-1], sol[-1]))
        xvals.append(xvals[-1] + stepsize)
    sol_max=max(sol+[find_maximum_on_interval(y_exact,start, \
stop)[0]])
    sol_min=min(sol+[find_minimum_on_interval(y_exact,start, \
stop)[0]])
    show(plot(y_exact(x),start,stop,rgbcolor=(1,0,0))+ \
```

```
line([[xvals[index],sol[index]] for index in \
range(len(sol))]),xmin=start, xmax = stop, ymax = sol_max, \
ymin = sol_min)
    if nsteps < steps_shown:
        table_range = range(len(sol))
    else:
        table_range = range(0,floor(steps_shown/2)) + \
range(len(sol)-floor(steps_shown/2),len(sol))
    html(tab_list([[i,xvals[i],sol[i]] for i in table_range], \
headers = ['step','x','y']))
```

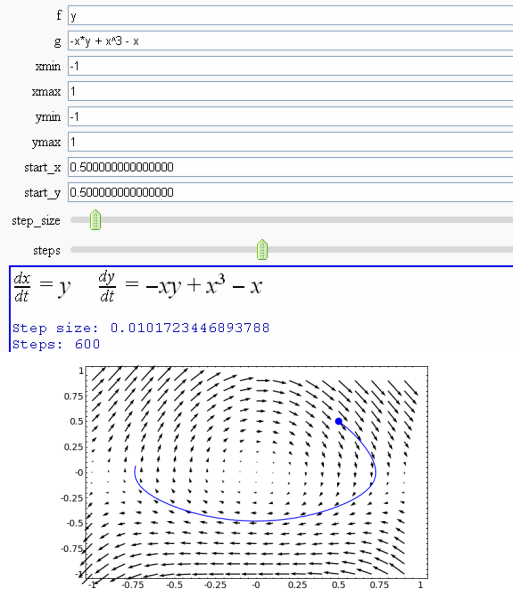


step	x	y
0	0.000000000000000	0.000000000000000
1	0.750000000000000	0.000000000000000
2	1.50000000000000	0.511229070017501
3	2.25000000000000	1.25935030997054
5	3.75000000000000	1.94874521368138
6	4.50000000000000	1.52007422462462
7	5.25000000000000	0.786926636375802
8	6.00000000000000	0.142725766305858

B.2. Векторне поле і метод Ейлера

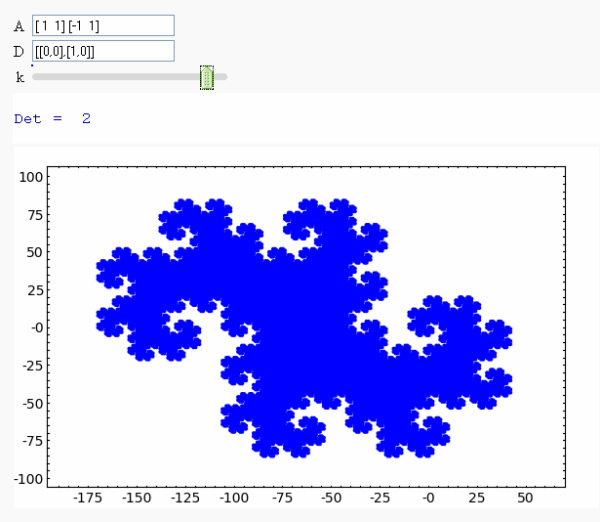
```
x,y = var('x,y')
@interact
def _(f = input_box(default=y), g=input_box(default=-x*y+x^3-x),
      xmin=input_box(default=-1), xmax=input_box(default=1),
      ymin=input_box(default=-1), ymax=input_box(default=1),
      start_x=input_box(default=0.5),
      start_y=input_box(default=0.5),
      step_size=(0.01,(0.001, 0.2)), steps=(600,(0, 1400)) ):
    old_f = f
    f = f.function(x,y)
```

```
    old_g = g
    g = g.function(x,y)
    steps = int(steps)
    points = [ (start_x, start_y) ]
    for i in range(steps):
        xx, yy = points[-1]
        try:
            points.append( (xx+step_size*f(xx,yy), \
yy+step_size*g(xx,yy)) )
        except (ValueError, ArithmeticError, TypeError):
            break

    starting_point = point(points[0], pointsize=50)
    solution = line(points)
    vector_field = plot_vector_field( (f,g), (x,xmin,xmax), \
(y,ymin,ymax) )

    result = vector_field + starting_point + solution

    html(r"<h2>$ \frac{dx}{dt} = %s$  $ \frac{dy}{dt} = \
%s$</h2>"%(latex(old_f),latex(old_g)))
    print "Step size: %s"%step_size
    print "Steps: %s"%steps
    result.show(xmin=xmin,xmax=xmax,ymin=ymin,ymax=ymax)
```



B.3. Фрактал

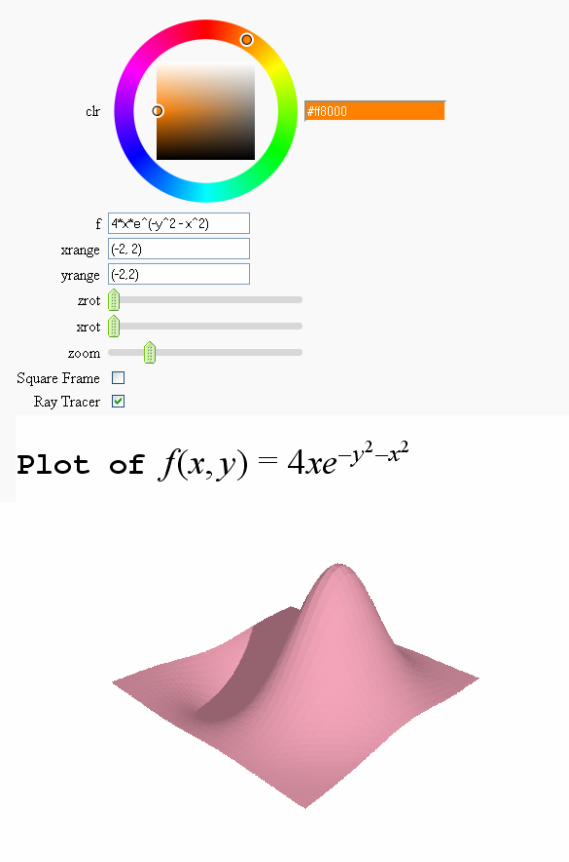
```
A = matrix([[1,1],[-1,1]])
D = [vector([0,0]), vector([1,0])]

@interact
def f(A = matrix([[1,1],[-1,1]]), D = '[[0,0],[1,0]]', \
k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(-n)*d[n] for n in range(k))
            ans.append(s)
        return ans
    G = points([v.list() for v in Dn(k)])
    show(G, frame=True, axes=False)
```



B.4. Інтерактивна побудова у просторі

```
var('x,y')
@interact
def example(clr=Color('orange'), f=4*x*exp(-x^2-y^2),
xrange='(-2, 2)', yrange='(-2,2)', zrot=(0,pi),
xrot=(0,pi), zoom=(1,(1/2,3)),
square_aspect=('Square Frame', False),
tachyon=('Ray Tracer', True)):
    xmin, xmax = sage_eval(xrange)
```

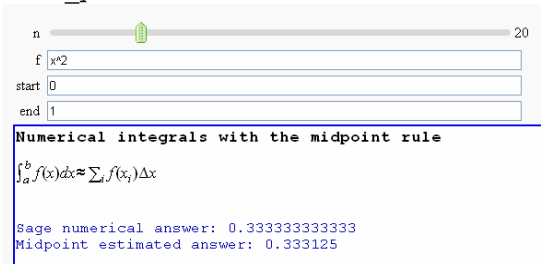
```
ymin, ymax = sage_eval(yrange)
P = plot3d(f, (x, xmin, xmax), (y, ymin, ymax), color=clr)
html('<h1>Plot of $f(x,y) = %s$</h1>'%latex(f))
aspect_ratio = [1,1,1] if square_aspect else [1,1,1/2]
show(P.rotate((0,0,1), -zrot).rotate((1,0,0),xrot),
     viewer='tachyon' if tachyon else 'jmol',
     figsize=6, zoom=zoom, frame=False,
     frame_aspect_ratio=aspect_ratio)
```

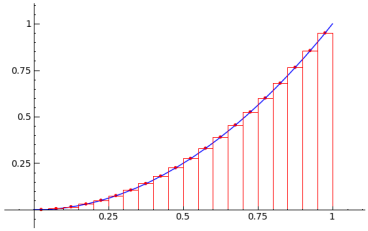


B.5. Обчислення інтегралу за правилом середніх прямокутників

```
var('x')
@interact
def midpoint(n = slider(1,100,1,4), f = input_box(default = \
"x^2", type = str), start = input_box(default = "0", type = \
str), end = input_box(default = "1", type = str)):
    a = N(start)
```

```
    b = N(end)
    func = sage_eval(f, locals={'x':x})
    dx = (b-a)/n
    midxs = [q*dx+dx/2 + a for q in range(n)]
    midys = [func(x_val) for x_val in midxs]
    rects = Graphics()
    for q in range(n):
        xm = midxs[q]
        ym = midys[q]
        rects = rects + line([[xm-dx/2,0],[xm-dx/2,ym], \
[xm+dx/2,ym], [xm+dx/2,0]], rgbcolor=(1,0,0)) + point((xm,ym), \
rgbcolor = (1,0,0))
    min_y = find_minimum_on_interval(func,a,b)[0]
    max_y = find_maximum_on_interval(func,a,b)[0]
    html('<h3>Numerical integrals with the midpoint rule</h3>')
    html('$\int_{a}^{b}{f(x) dx} {\\approx} \sum_i{f(x_i) \Delta
x}$')
    print "\n\nSage numerical answer: " + \
str(integral_numerical(func,a,b,max_points = 200)[0])
    print "Midpoint estimated answer: " + \
str(RDF(dx*sum([midys[q] for q in range(n)])))
    show(plot(func,a,b) + rects, xmin = a, xmax = b, ymin = \
min_y, ymax = max_y)
```





Г. Основи теорії кодування

SAGE має інтерфейси для підключення як широко поширених пакетів загального призначення, так і для спеціалізованих систем. Одна з них

– GAP, ПЗ для алгебраїчних досліджень – включає в себе пакет GUAVA, що реалізує в собі основні об'єкти та методи теорії кодування. Застосування цього пакету дозволяє частково розв'язати проблему практичної підтримки курсу «Теорія кодування» у випадку відсутності лабораторних занять.

Розглянемо загальні функції теорії кодування, що надає SAGE.

1. Клас `LinearCode` та функція `LinearCodeFromVectorspace`.

```
sage: MS = MatrixSpace(GF(2),4,7)
sage: G = MS([[1,1,1,0,0,0,0], [1,0,0,1,1,0,0],
              [0,1,0,1,0,1,0], [1,1,0,1,0,0,1]])
sage: C = LinearCode(G) sage: C Linear code of length 7, dimension 4 over Finite Field of size 2
sage: C.base_ring()
Finite Field of size
sage: C.dimension()
4
sage: C.length()
7
sage: C.minimum_distance()
3
sage: C.spectrum()
[1, 0, 0, 7, 7, 0, 0, 1]
sage: C.weight_distribution()
[1, 0, 0, 7, 7, 0, 0, 1]
```

Наведемо приклад застосування для створення власної кодової функції, що повертає ерміттів гексакод [6, 3, 4] типу IV над GF(4):

```
def Hexacode():
  F = GF(4,"z")
  z = F.gen()
  MS = MatrixSpace(F, 3, 6)
  G = MS([[1, 0, 0, 1, z, z ],\
          [0, 1, 0, z, 1, z ],\
          [0, 0, 1, z, z, 1 ]])
  return LinearCode(G)
```

2. `spectrum` (ваговий розподіл), `minimum_distance`, `characteristic_function` та кілька реалізацій дзета-функції Івана Дуурсма (`zeta_polynomial`, `zeta_function`, `chinen_polynomial`):

```
sage: C = HammingCode(3,GF(2))
sage: C.zeta_polynomial()
2/5*T^2 + 2/5*T + 1/5
sage: C = best_known_linear_code(6,3,GF(2))
sage: C.minimum_distance()
3
```

```
sage: C.zeta_polynomial()
2/5*T^2 + 2/5*T + 1/
```

3. gen_mat, check_mat, decode, dual_code, extended_code, binomial_moment для класу лінійних кодів:

```
sage: C = HammingCode(3,GF(2))
sage: C.binomial_moment(4)
35
sage: C = HammingCode(3,GF(2))
sage: MS = MatrixSpace(GF(2),1,7)
sage: F = GF(2); a = F.gen()
sage: v1 = [a,a,F(0),a,a,F(0),a]
sage: C.decode(v1)
(1, 0, 0, 1, 1, 0, 1)
```

4. Предикати is_self_orthogonal, is_permutation_automorphism, "==", is_self_dual.

5. Функції перестановки: standard_form, automorphism_group_binary_code, is_permutation_automorphism, module_composition_factors:

```
sage: C = HammingCode(3,GF(2))
sage: G = C.automorphism_group_binary_code(); G
Permutation Group with generators [(2,3)(5,7), (2,5)(3,7),
                        (2,3,7,5)(4,6), (2,4)(6,7), (1,2)(3,4)]
sage: G.order()
168
sage: C = HammingCode(3,GF(2))
sage: C.gen_mat()
[1 0 0 1 0 1 0]
[0 1 0 1 0 1 1]
[0 0 1 1 0 0 1]
[0 0 0 0 1 1 1]
sage: C.redundancy_matrix()
[1 1 0]
[1 1 1]
[1 0 1]
[0 1 1]
sage: C.standard_form()[0].gen_mat()
[1 0 0 0 1 1 0]
[0 1 0 0 1 1 1]
[0 0 1 0 1 0 1]
[0 0 0 1 0 1 1]
sage: MS = MatrixSpace(GF(2),4,8)
sage: G = MS([[1,0,0,0,1,1,1,0],[0,1,1,1,0,0,0,0],
              [0,0,0,0,0,0,0,1],[0,0,0,0,0,1,0,0]])
```

```
sage: C = LinearCode(G)
sage: gp = C.automorphism_group_binary_code()
sage: C.module_composition_factors(gp)
[ rec(
  field := GF(2),
  isMTXModule := true,
  dimension := 1,
  generators := [ [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ]],
                [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ]],
  smashMeataxe := rec(
      algebraElement :=
     [ [ [ 5, 3 ], [ 5, 3 ] ], [ Z(2)^0, Z(2)^0, 0*Z(2), Z(2)^0,
                 0*Z(2), Z(2)^0, Z(2)^0, Z(2)^0 ] ],
      algebraElementMatrix := [ [ 0*Z(2) ] ],
      characteristicPolynomial := x_1,
      charpolFactors := x_1,
      nullspaceVector := [ Z(2)^0 ],
      ndimFlag := 1 ),
  IsIrreducible := true ), rec(
  field := GF(2),
  isMTXModule := true,
  dimension := 1,
  generators := [ [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ]],
                [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ]],
  smashMeataxe := rec(
      algebraElement := [ [ [ 5, 2 ], [ 1, 2 ] ], [ 0*Z(2),
0*Z(2), 0*Z(2), 0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0, 0*Z(2) ] ],
      algebraElementMatrix := [ [ 0*Z(2) ] ],
      characteristicPolynomial := x_1,
      charpolFactors := x_1,
      nullspaceVector := [ Z(2)^0 ],
      ndimFlag := 1 ),
  IsIrreducible := true ), rec(
  field := GF(2),
  isMTXModule := true,
  dimension := 1,
  generators := [ [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ]],
                [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ]],
  smashMeataxe := rec(
      algebraElement := [ [ [ 4, 2 ], [ 7, 4 ] ], [ 0*Z(2),
Z(2)^0, Z(2)^0, 0*Z(2), Z(2)^0, Z(2)^0, Z(2)^0, Z(2)^0 ] ],
      algebraElementMatrix := [ [ 0*Z(2) ] ],
      characteristicPolynomial := x_1,
      charpolFactors := x_1,
```

```
      nullspaceVector := [ Z(2)^0 ],
  ndimFlag := 1 ),
  IsIrreducible := true ), rec(
  field := GF(2),
  isMTXModule := true,
  dimension := 1,
  generators := [ [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ]],
                [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ], [ [ Z(2)^0 ] ]],
  smashMeataxe := rec(
      algebraElement := [ [ [ 4, 6 ], [ 1, 6 ] ], [ 0*Z(2),
Z(2)^0, Z(2)^0, 0*Z(2), Z(2)^0, 0*Z(2), Z(2)^0, Z(2)^0 ] ],
      algebraElementMatrix := [ [ Z(2)^0 ] ],
      characteristicPolynomial := x_1+Z(2)^0,
      charpolFactors := x_1+Z(2)^0,
      nullspaceVector := [ Z(2)^0 ],
  ndimFlag := 1 ),
  IsIrreducible := true ) ]
```

Основні класи кодів, що надає SAGE: БЧХ (`BCHCode`), Хеммінга (`HammingCode`), Уолша (`WalshCode`), Ріда-Соломона (`ReedSolomonCode`), Голея (`BinaryGolayCode`, `ExtendedBinaryGolayCode`, `ExtendedTernaryGolayCode`, `TernaryGolayCode`), торичні (`ToricCode`), дуадічні (`DuadicCodeEvenPair`, `DuadicCodeOddPair`), квадратичні лишкові (`QuadraticResidueCodeEven[Odd]Pair`), циклічні (`CyclicCodeFromGeneratingPolynomial`, `CyclicCodeFromCheckPolynomial`), лінійні (`LinearCodeFromCheckMatrix`, `RandomLinearCode` та ін.).

Д. Розширення можливостей SAGE

Відкритий характер системи дає можливість додавання до неї нових функцій, типів та класів, створювати нові бібліотеки та інтегрувати у неї нові програми як: 1) сценарії SAGE; 2) сценарії на Python з використанням бібліотеки SAGE; 3) програми на C/C++, інтегрованими засобами Cython; 4) код Cython; 5) програма мовою СКМ (наприклад, сценарій Maxima); 6) будь-яка комбінація з пп. 1–5.

Найпростіший спосіб додання нової функції до SAGE – вставка коду Cython. Якщо розпочати введення командою `%cython`, то при виконанні вводу він: а) зберігається у файлі; б) транслюється у мову C; в) компілюється у динамічну бібліотеку (.so), що завантажується надалі при повторному виклику уведеного.

Застосування Cython дозволяє за однакового з Python коду (додається лише %cython) досягти на обчислювальних операціях більш ніж 200-кратного прискорення. Наприклад, код для швидкого піднесення до степеня двійки

```
{{{
%cython
def is2pow(unsigned int n):
  while n != 0 and n%2 == 0:
    n = n >> 1
  return n == 1
time [n for n in range(10^5) if is2pow(n)]
}}}
```
дає результат
```
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192,
16384, 32768, 65536]
CPU time: 0.05 s, Wall time: 0.06 s
```
Перетворення його на код Python (видаленням `%cython`) показує вже
```
CPU time: 13.04 s, Wall time: 15.08 s
```
Інтеграція існуючого ПЗ до SAGE виконується за допомогою псевдотермінального інтерфейсу: єдиною вимогою до інтегрованого ПЗ є можливість його роботи із стандартними потоками введення/виведення. Взаємодія між інтегрованою програмою та SAGE виконується неблокованими (асинхронними) каналами введення/виведення за допомогою методів `_send`, `_so_far` та `_get`.

Для інтеграції власного математичного ПЗ у SAGE ми пропонуємо наступний шаблон інтерфейсу модельної програми `mysystem`:
```
"""
[[Наводиться опис ПЗ, до якого створюється інтерфейс]]
EXAMPLES:
[[Приклади застосування нового інтерфейсу у SAGE]]
AUTHORS:
    Ваші прізвище та ініціали
"""
from expect import Expect, ExpectElement, ExpectFunction, Func-
tionElement
from sage.misc.misc import verbose
class MySystem(Expect):
    """
    [[Наводиться допомога з mysystem]]
    """
    def __init__(self,
                 maxread=100000, script_subdirectory=None,
                 logfile=None, server=None, server_tmpdir=None):
        Expect.__init__(self,
                 name = 'mysystem', #ім'я системи
                 # Шаблон регулярного виразу для програмного
                 # вводу (чим складніше, тим краще)
```

```
                prompt = '>> ',
                # Команда запуску інтегрованої програми
                command = "mysystem",
                maxread = maxread, server=server,
                server_tmpdir=server_tmpdir,
                script_subdirectory=script_subdirectory,
                # Якщо цей прапор=true, то при натисканні
                # Ctrl-C перевантажується весь інтерфейс
                 restart_on_ctrlc = False,
                # Якщо цей прапор=true, друкує повідомлення
                # перед початком виконання команди
                verbose_start = False, logfile=logfile,
                # Максимальний розмір введення, при
                # перевищенні якого вивід іде у файл
                eval_using_file_cutoff=1024)
        self.__seq = 0
    def _repr_(self):
        return 'Інтерпретатор MySystem'
    def __reduce__(self):
        return reduce_load_mysystem, tuple([])
    def __getattr__(self, attrname):
        if attrname[:1] == "_":
            raise AttributeError
        return MySystemFunction(self, attrname)
    # тут вкажіть команду завершення роботи mysystem
    def _quit_string(self):
        raise NotImplementedError
    # тут вкажіть команду mysystem для читання з файлу filename
    def _read_in_file_command(self, filename):
        raise NotImplementedError
    # має повернути список з всіх та ідентифікаторів mysystem
    def trait_names(self):
        raise NotImplementedError
    # тут реалізуйте читання з файлу filename у mysystem
    def read(self, filename):
        raise NotImplementedError
    def kill(self, var): # знищення змінної із заданим ім'ям
        pass
    def console(self): # виконує консольну команду (див. нижче)
        pass
    def version(self): # отримує версію системи (див. нижче)
        pass
    def _object_class(self):
        return MySystemElement
```

```
    # визначає символ для позначення істини у mysystem
    def _true_symbol(self):
        raise NotImplementedError
    # визначає символ для позначення хиби у mysystem
    def _false_symbol(self):
        raise NotImplementedError
    # визначає символ для позначення еквівалентності у mysystem
    def _equality_symbol(self):
        raise NotImplementedError
    def help(self, command): # допомога по заданій команді
        raise NotImplementedError
class MySystemElement(ExpectElement):
    # Опишіть тут елементи інтегрованої системи
    def trait_names(self):
        return self.parent().trait_names()
class MySystemFunctionElement(FunctionElement):
    def _sage_doc_(self):
        M = self._obj.parent()
        return M.help(self._name)
class MySystemFunction(ExpectFunction):
    def _sage_doc_(self):
        M = self._parent
        return M.help(self._name)
def is_MySystemElement(x):
    return isinstance(x, MySystemElement)
mysystem = MySystem()# Створення об'єкту
def reduce_load_MySystem():
    return mysystem
import os
def mysystem_console(): # Виконує процес
    os.system('mysystem')
def mysystem_version():
    """
    Приклад:          sage: mysystem.version()
    """
    raise NotImplementedError
```

Більш простим варіантом інтеграції є випадок, коли система завантажується лише для виконання команди, поданої на її вхід з командного рядка чи файлу.

Шокалюк Світлана Вікторівна

Основи роботи в SAGE

Підп. до друку 25.09.2008
Папір офсетний №1
Ум. друк. арк. 3,72

Формат 80×84 1/16
Зам. №2-2509
Тираж 100 прим.

Жовтнева районна друкарня
50014, м. Кривий Ріг, вул. Електрична, 5
Тел. (0564) 407-29-02

E-mail: cc@optima.com.ua