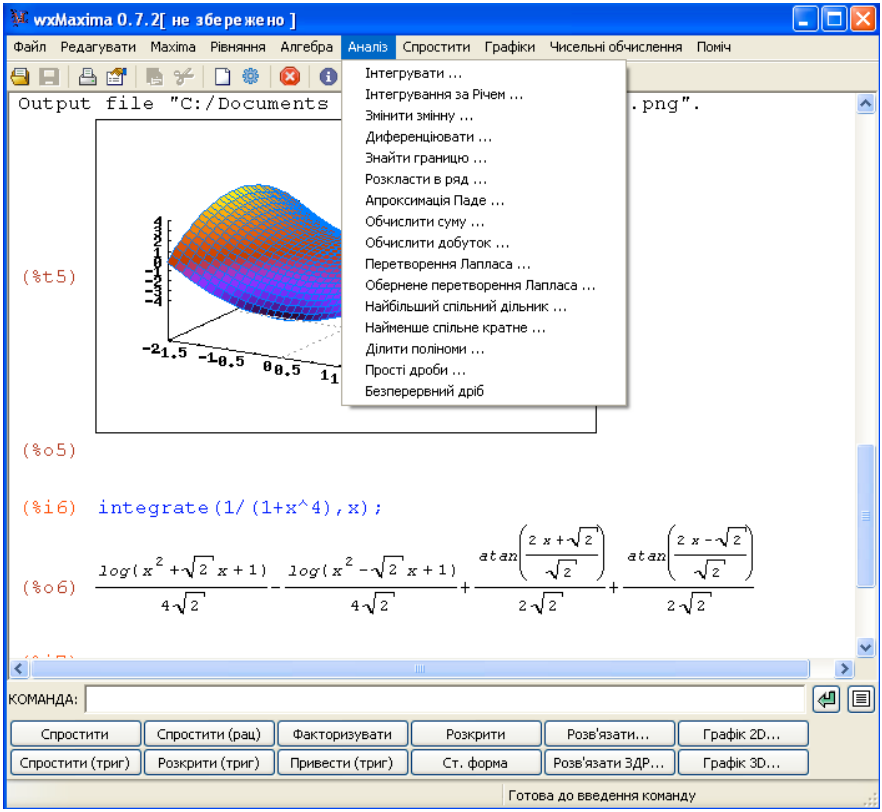
С.О. Семеріков

Maxima 5.13: довідник користувача



Київ, 2007

Міністерство освіти та науки України
Національний педагогічний університет
ім. М.П. Драгоманова
Кафедра інформатики

С.О. Семеріков

Maxima 5.13: довідник користувача

Київ, 2007

УДК 681.51.001

Семеріков С.О. Maxima 5.13: довідник користувача / За ред. академіка АПН України М.І. Жалдака. – Київ, 2007. – 48 с.

В довіднику наведено короткий огляд основних прийомів роботи у вільно поширюваній системі комп'ютерної математики Maxima (російська та українська версії), спрямованої на застосування у вітчизняній системі освіти.

Для широкого кола студентів ВНЗ різного профілю, педагогів, наукових та інженерних працівників.

Рецензенти:

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

Ю.В. Триус – д. пед. н., проф., проректор з навчально-методичної роботи Європейського університету (м. Київ)

ISBN 967-4182-25-3

Зміст

1. Maxima: вчора, сьогодні, завтра

Максима – це відома алгебраїчна система, розробка якої почалася в Масачусетському технологічному інституті (MTI) в 60-х роках минулого століття у рамках проекту MAC. Спочатку дослідження символьної й алгебраїчної обробки математичних виразів (symbolic and algebraic computing, SAC) було пов'язане зі штучним інтелектом, однак незабаром вона перетворилася на окрему самостійну область досліджень, що відноситься зараз більше до математики, ніж до штучного інтелекту.

Maxima – одна з програм для виконання математичних обчислень, символьних перетворень, а також для побудови різноманітних графіків. Складні обчислення оформляються у вигляді окремих процедур, що можуть потім використовуватися при рішенні інших задач. Система поширюється під ліцензією GPL і доступна як користувачам ОС Linux, так і користувачам Windows.

Maxima дає можливість фахівцям розв'язувати велику кількість достатньо складних задач, не вдаючись у тонкощі програмування. Завдяки цьому програма одержала широке поширення у фізиці, біології, економіці тощо.

Уміння проводити аналітичні розрахунки – одне з головних достоїнств цієї програми. Maxima «вміє» перетворювати і спрощувати алгебраїчні вирази, диференціювати й обчислювати визначені і невизначені інтеграли, обчислювати скінченні і нескінченні суми і добутки, розв'язувати алгебраїчні і диференціальні рівняння і системи, а також розкладати функції в ряди і знаходити границі. Крім того, Maxima має стандартні доповнення для аналітичних розрахунків.

Для задач, які неможливо розв'язати аналітично, Maxima має у своєму розпорядженні велику кількість ефективних алгоритмів для проведення чисельних розрахунків. Maxima дозволяє розв'язувати задачі оптимізації (лінійного програмування, знаходження екстремумів функцій), а також задачі математичної статистики. У Maxima реалізовано адаптивний контроль точності, заснований на виборі внутрішніх алгоритмів, що дозволяють її максимізувати.

Пакет має вбудовану довідкову систему з прикладами використання тих чи інших функцій.

Система настільки гнучка й універсальна, що може надати неоціненну допомогу в розв'язуванні математичних задач як школяреві, котрий осягає основи математики, так і майбутньому науковцеві, котрий використовує математичні методи для розв'язання різних

прикладних задач.

Історія розробки Maxima поділяється на три періоди: науково-дослідний проект у MTI, проект під керівництвом Вільяма Шелтера і поточний проект Maxima.

MACSYMA (Проект Mac's SYmbolic MAnipulation System) була розроблена групою Mathlab у лабораторії комп'ютерних наук MTI (спочатку відомої як Проект MAC) у 1969-1972 р. Ця робота була підтримана грантами NSG 1323 NASA, N00014-77-C-0641 Дослідницького агентства ВМС, ET-78-C-02-4687 Міністерства енергетики США і F49620-79-C-020 ВПС США. Macsyma була потім модифікована для використання під операційною системою UNIX (на комп'ютерах DEC VAX і робочих станціях Sun) Ричардом Фейтманом і його колегами з Каліфорнійського університету (Берклі); ця версія Macsyma відома як VAXIMA.

Ліцензування в 70-ті рр. програмних кодів Macsyma призвело до створення інших систем комп'ютерної математики – Maple фірми Waterloo Maple Inc. та Mathematica фірми Wolfram Research. Спільність цих програмних продуктів виражається як у схожому синтаксисі (табл. 1), так й у спільних алгоритмах.

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

	Maxima	**Maple**	**Mathematica**
границя	`limit(x-7,x,3);`	`limit(x-7,x=3);`	`Limit[x-7,x->3]`
розгортка виразу	`expand((a+b)^3);`	`expand((a+b)^3);`	`Expand[(a+b)^3]`
розклад на множники	`factor(%);` `ezgcd(num, denom);`	`factor(%);` `normal(%);`	`Factor[%]`
розв'язання рівнянь	`solve(a*x^2=4,x);`	`solve(a*x^2=4,x);`	`Solve[a x^2==4,x]`
3D-графіка	`plot3d(sin(x*y), [x,-2,2], [y,-1,1]);`	`plot3d(sin(x*y), x=-2..2, y=-1..1);`	`Plot3D[Sin[x y], {x,-2,2}, {y,-1,1}]`
квадратний корінь	`sqrt(3);`	`sqrt(3);`	`Sqrt[3]`

функції	`f(x):=x^2+1/2;`
	або
	`define(f(x),x^2+1/2);`
		`f:=x->x^2+1/2;`
		або
		`f:=unapply(x^2+1/2,x);`
			`f[x_]=x^2+1/2`
			або
			`f=Function[x,x^2+1/2]`

Таким чином, Macsyma фактично стала родоначальником всього

напрямку програм символьної математики.

“Академічність”, неінтуїтивний інтерфейс користувача Macsyma у 80-ті роки суттєво звузили сферу її використання, до того ж лоббіювання інтересів інших фірм, що виробляли подібні програмні продукти, призвели до фактичної зупинки роботи над нею.

Новий етап у розвитку Maxima настав у 1999 році, коли минув термін дії патенту, і права на Maxima повернулись до одного з її авторів – Вільяма Шелтера, який виконав повну переробку системи та залучив до її відкритої розробки провідних спеціалістів.

Вільям Шелтер розробляв і підтримував цю версію Maxima із самого початку проекту до своєї передчасної кончини в 2001 році. З листопада 2001 року проект Maxima підтримується роботою команди на чолі з Джеймсом Амундсоном і Ричардом Фейтманом.

Сьогодні проект продовжує активно розвиватися, і участь в ньому є кращою візитною карткою для математиків та програмістів з усього світу. Завдяки зусиллям інтернаціональної команди розробників Maxima набула ряд особливостей, що дозволяють використовувати її у вітчизняній системі освіті: система повністю відкрита, ліцензійно чиста і безкоштовна, незалежна від використовуваної операційної системи й апаратної платформи; невелика за розміром, невимоглива до апаратних ресурсів; надає користувачу широкий вибір інтерфейсів.

2. Інтерфейси Maxima

Система комп'ютерної математики Maxima адаптована до потреб різних категорій користувачів. Текстове ядро системи (рис. 1) дозволяє побудувати такі спеціалізовані інтерфейси за допомогою технології "програмних обгорток".

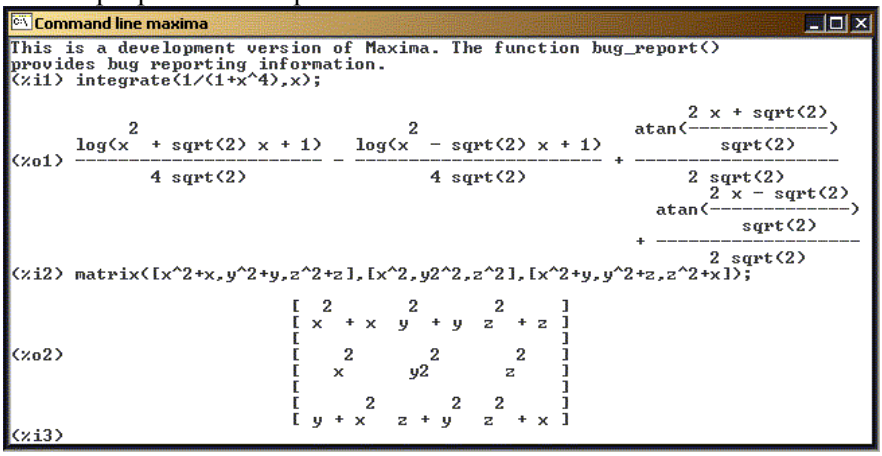


Рис. 1

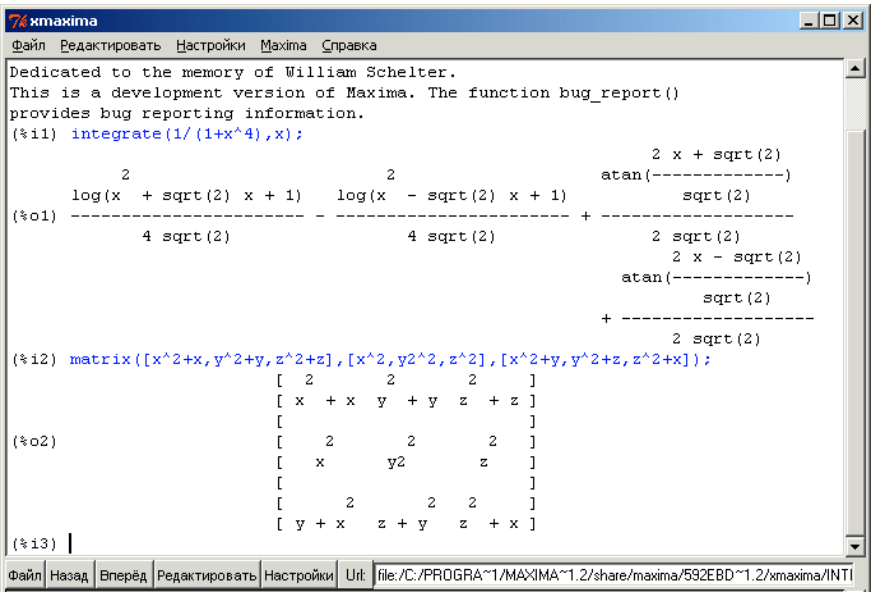


Рис. 2

Графічний інтерфейс xmaxima (рис. 2) додає до ядра СКМ можливості побудови графіків та виклику довідкової системи. Сьогодні цей інтерфейс нами повністю локалізований і рекомендується до застосування на комп'ютерних системах з мінімальними ресурсами.

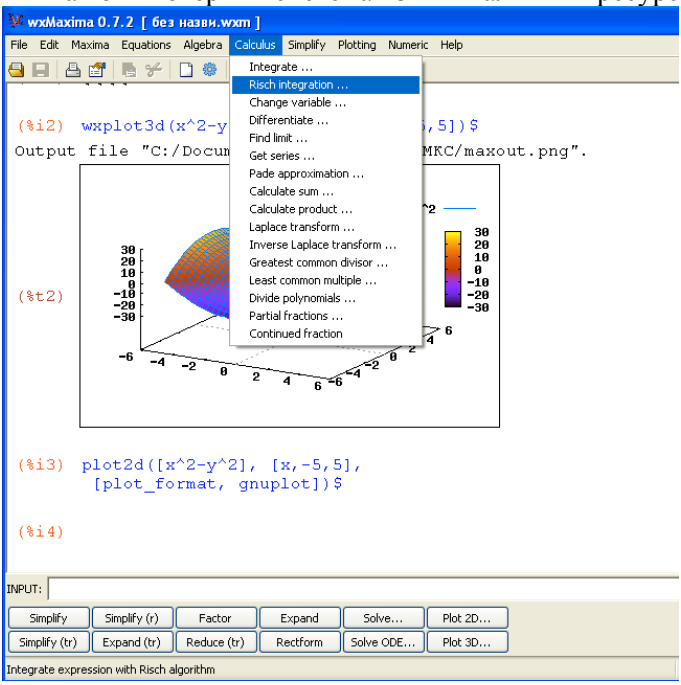


Рис. 3

Найбільш придатним для початківців виявився інтерфейс wxMaxima (рис. 3), що має розвинені можливості конструювання вхідних виразів та інтерактивну допомогу. Інтерфейс TeXmacs відповідає концепції WYSIWYW (What You See Is What You Want) та спрямований на використання науковцями, надаючи їм уніфіковане середовище для створення структурованих документів з різними типами об'єктів (текстових, графічних, математичних, інтерактивних тощо). Для відображення результатів використовується система TeX (рис. 4).

Інтерфейс Symaxx/2 (рис. 5) орієнтований на подання задачі у вигляді набору взаємопов'язаних текстових, графічних і обчислювальних об'єктів та спрямований на застосування студентами інженерних спеціальностей.

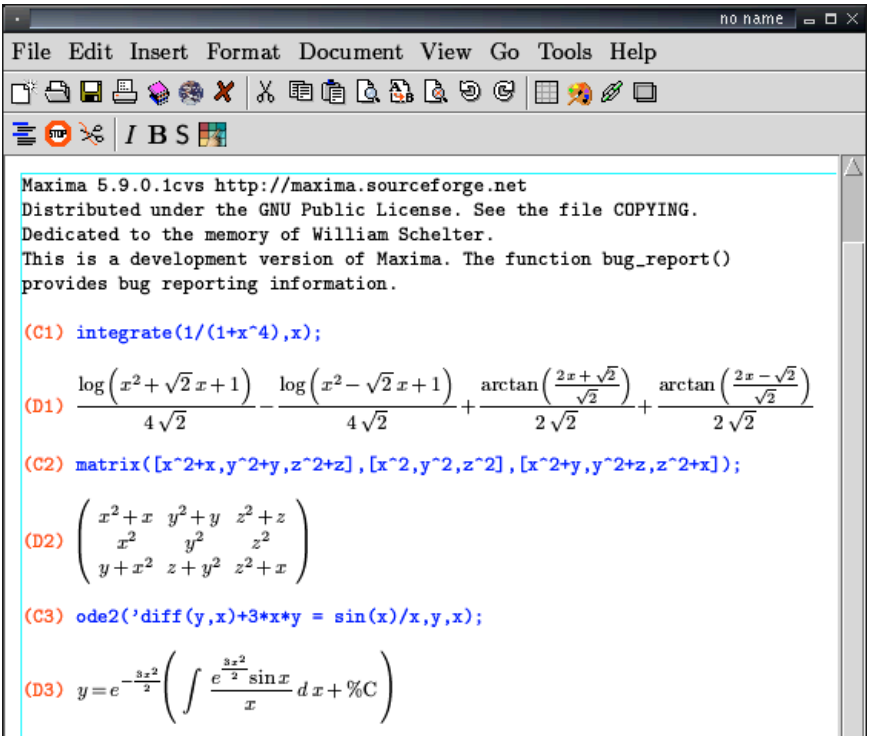

no name
File Edit Insert Format Document View Go Tools Help
Maxima 5.9.0.1cvs http://maxima.sourceforge.net
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
This is a development version of Maxima. The function bug_report()
provides bug reporting information.
(C1) integrate(1/(1+x^4),x);
(D1) $\frac{\log\left(x^2+\sqrt{2}\,x+1\right)}{4\sqrt{2}}-\frac{\log\left(x^2-\sqrt{2}\,x+1\right)}{4\sqrt{2}}+\frac{\arctan\left(\frac{2x+\sqrt{2}}{\sqrt{2}}\right)}{2\sqrt{2}}+\frac{\arctan\left(\frac{2x-\sqrt{2}}{\sqrt{2}}\right)}{2\sqrt{2}}$
(C2) matrix([x^2+x,y^2+y,z^2+z],[x^2,y^2,z^2],[x^2+y,y^2+z,z^2+x]);
(D2) $\begin{pmatrix} x^2+x & y^2+y & z^2+z \\ x^2 & y^2 & z^2 \\ y+x^2 & z+y^2 & z^2+x \end{pmatrix}$
(C3) ode2('diff(y,x)+3*x*y = sin(x)/x,y,x);
(D3) $y=e^{-\frac{3x^2}{2}}\left(\int \frac{e^{\frac{3x^2}{2}}\sin x}{x}\,dx+\%C\right)$

Рис. 4

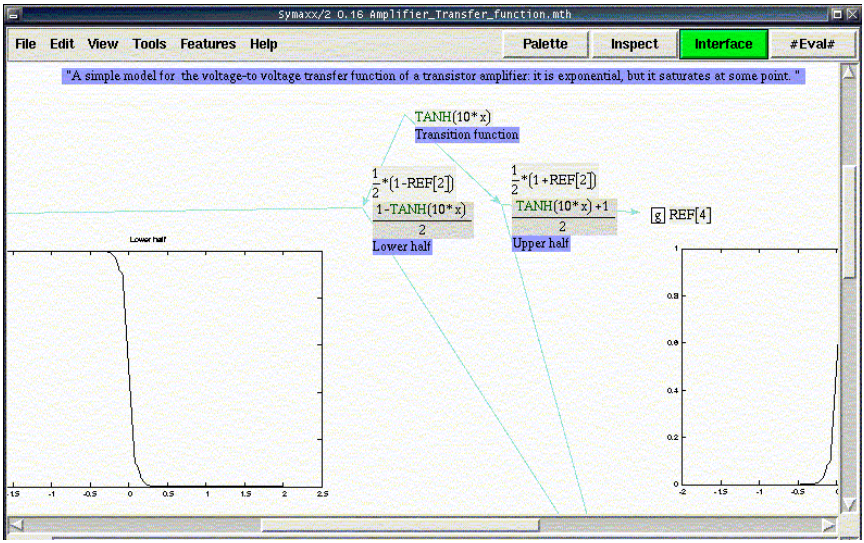

Symaxx/2 0.16 Amplifier_Transfer_function.mth
File Edit View Tools Features Help
Palette Inspect Interface #Eval#
"A simple model for the voltage-to voltage transfer function of a transistor amplifier: it is exponential, but it saturates at some point. "
TANH(10*x)
Transition function
1/2*(1-REF[2])
1-TANH(10*x)
2
Lower half
1/2*(1+REF[2])
TANH(10*x)+1
2
Upper half
REF[4]
Lower half

Рис. 5

Головним недоліком існуючих інтерфейсів Maxima була відсутність їхніх локалізованих версій, що є необхідною умовою популяризації даної системи серед вітчизняних користувачів. Адже для того, щоб вільно працювати у розвиненому середовищі комп'ютерної математики, необхідно володіти не тільки англійською, а й її «математичним діалектом».

Починаючи з моменту входження у групу розробників Maxima в 2002 році, нами ведеться цілеспрямована робота з її локалізації. Результатами цієї роботи є створення російського варіанту інтерфейсу xmaxima (2004 р.), російського (2006 р.) та українського (2007 р.) варіантів інтерфейсу wxMaxima (рис. 6).

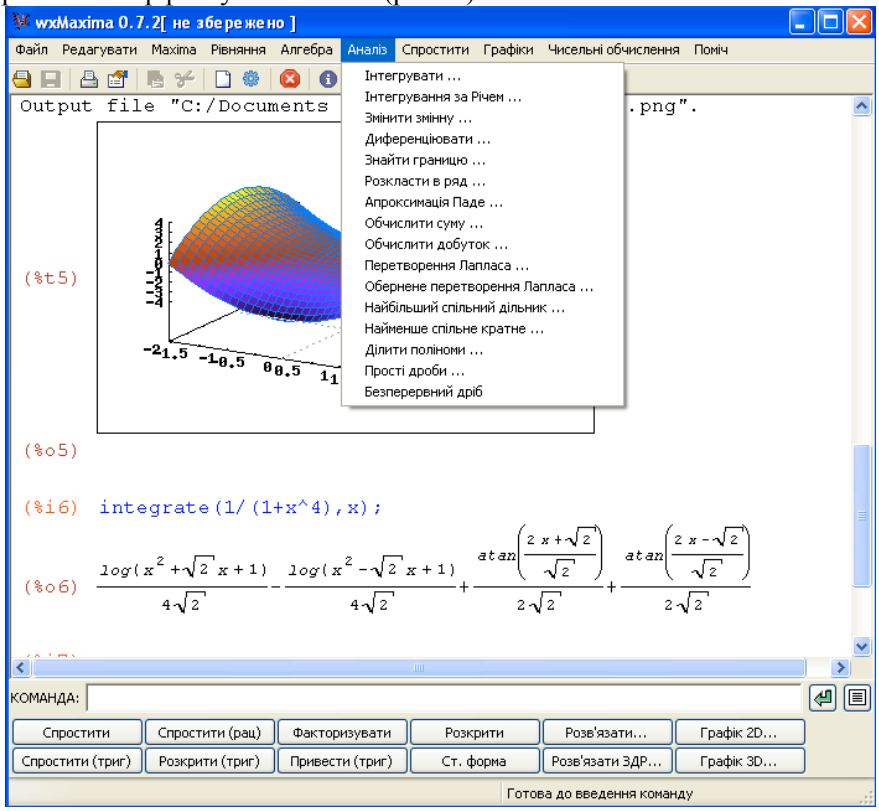


Рис. 6

Починаючи з серпня 2007 р., результати виконаної роботи входять у стабільну версію системи, що доступна для завантаження за адресою: `http://downloads.sourceforge.net/maxima/maxima-5.13.0.exe`

3. Початок роботи у Maxima

При старті виводиться інформація про систему та мітка `(C1)` або `(%i1)`, в залежності від поточної конфігурації. Кожна введена команда та її вивід позначаються міткою з метою повторного використання. Символ `C` (від command) або `%i` (від input) використовуються для позначення команд, введених користувачем, а `D` (від display) або `%o` (від output) – при відображенні результатів обчислень.

Для ініціалізації процесу обчислень необхідно ввести команду, символ "`;`" (крапка з комою) та натиснути Enter. Якщо вивід результату на екран не потрібен, то замість крапки з комою використовується символ "`$`". Звернутися до результату останньої команди можна за допомогою символу "`%`". Для повтору раніше введеної команди, скажімо, `(C2)`, досить ввести два апострофа та мітку потрібної команди: `''C2`.

Система Maxima звертає увагу на регістр введених символів в іменах вбудованих констант та функцій: запис `sin(x)` *не* еквівалентний запису `SIN(x)`.

Для стандартних математичних констант застосовуються наступні позначення: `%e` для основи натуральних логарифмів, `%i` для уявної одиниці та `%pi` для числа π.

Надання значення будь-якій змінній виконується за допомогою знака "`:`" (двокрапка), а символ "`=`" (дорівнює) застосовується при визначенні рівнянь чи підстановок.

```
(C1) x:2;
(D1)                          2
(C2) y:3;
(D2)                          3
(C3) x + y;
(D3)                          5
```

Функція `kill` анулює значення змінних. Параметр `all` цієї функції видаляє всі змінні, включаючи мітки.

```
(C8) kill(x);
(D8)                         DONE
(C9) x + y;
(D9)                        x + 3
(C10) kill(all);
(D0)                         DONE
(C1) x + y;
(D1)                        y + x
```

Maxima підтримує точні обчислення з великими числами, розмір яких обмежений лише ресурсами комп'ютера.

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

```
(C1) 144^25;
(D1) 910043815000214977332758527534256632492715260325658624
(C2) %^(1/25);
(D2) 144
```

Знак "^" або "**" – оператор піднесення до степеня.

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

4. ВИРАЗИ ТА ТРИГОНОМЕТРИЧНІ ФУНКЦІЇ

Складні вирази складаються з частин, що можуть інтерпретуватися по-різному. Робота з частинами виразів нагадує роботу зі списками. Так, для виділення будь-якої заданої частини виразу використовується функція `part`.

```
(C1) f:a+b*x^2+c*x^3;    //надаємо f значення функції
(D1) c·x^3+b·x^2+a       //результат
(C2) part(f,2);          //виділяємо другу частину виразу
(D2) b·x^2               //результат виконання
(C3) kill(all);
```

Наступні функції повертають окремі частини виразу:

`first(f)` – повертає перший елемент `f`;

`last(f)` – повертає останній елемент `f`;

`rest(f)` – повертає `f` з вилученим першим елементом;

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

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

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

а) комбінування числових підвиразів:

```
(C1) factor(6*x*2);
(D1)                    12x;
```

б) приведення подібних множників у добутках:

```
(C2) factor(x^3*y*x^5);
(D2)                    x^8·y
```

в) приведення подібних членів суми:

```
(C3) factor(x+12+4*x);
(D3)                    5x+12;
```

г) скорочення на найбільший поліноміальний дільник:

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

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

д) розкладання поліномів і пониження ступеня виразів:

```
(C5) factor((x+1)^2-x^2);
(D5)                                  2x+1
```

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

```
(C6) factor((2*x)/(x^2-1)-1/(x+1));
(D6)                                  1/(x-1)
(C7) kill(all);
```

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

```
(C1) expand((x-a)*(x-b)*(x-c));
(D1)                        x^3-c*x^2-b*x^2-a*x^2+b*c*x+a*c*x-a*b*c
(C2) kill(all);
```

Команда `ev` дозволить одержати чисельне значення виразу. Її перший аргумент є виразом, що обчислюється, а другий – опція `numer`. Нагадаємо, що символ `%` означає результат попереднього обчислення.

```
(C1) 1/100+1/1001;
(D1)                                  201/10100
(C2) (1+sqrt(2))^5;
(D2)                                  (sqrt(2)+1)^5
(C3) expand(%);
(D3)                                  29*sqrt(2)+41
(C4) ev(29*sqrt(2) + 41, numer);
(D4)                          82.01219330881976
```

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

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

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

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

Запис `mBn` є скорочена форма виразу $m*10^n$.

Кількість значущих цифр у поданні числа визначається спеціальною змінною `fpprec`. Збільшення її значення приводить до зрос-

тання точності результату, наприклад,

```
(C7) fpprec;
(D7)                                  16
(C8) fpprec:100;
(D8)                                  100
(C9) ''c5;
(D9) 8.201219330881976076576049236862485 25#
03077530516721861648463104778270702443 4954#
835068385111442261515 5B1
```

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

Система Maxima може працювати з числами довільної довжини і точності:

```
(C10) 100!;
(D10)  933262154439441526816992388562667004#
907159682643816214685929638952175999932299 1#
560894146397615651828625369792082722375825 1#
185210916864000000000000000000000000
```

Наприклад, обчислимо число π з точністю 200 знаків після коми:

```
(C11) %pi, numer;
(D11)                    3.141592653589793
(C12) fpprec:200;
(D12)                    200
(C13) bfloat(%PI);
(D13) 3.1415926535897932384626433832795028 84#
197169399375105820974944592307816406286208 99#
862803482534211706798214808651328230664709 38#
446095505822317253594081284811174502841027 01#
938521105559644622948954930381 9B0
```

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

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

```
(C16) block([fpprec:100], bfloat(%pi));
(D16) 3.1415926535897932384626433832795028#
841971693993751058209749445923078164062#
862089986280348253421170680B0
(C17) ' 'c15;
(D17)                          3.141592653589793B0
```

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

```
(C18) block([fpprec:100],sin(bfloat(%pi)));
(D18) - 2.5705791980297237116895690647137814#
273847841138560124733757044937800020983036 8#
3173940359193381364537 7B-101
(C19) block([fpprec:100],sin(%pi)) ;
(D19)                          0
(C20) sin(%pi/6)^2 +cos(%pi/6)^2;
(D20)                          1
(C21) sec(%PI/3);
(D21)                          2
(C22) asin(1/2);
                               %PI
(D22)                          ---
                               6
```

На жаль, Maxima не може в символьному виді обчислити значення обернених тригонометричних функцій у деяких точках:

```
(C23) asin(sqrt(3)/2);
```

$$(D23)\qquad ASIN\left(\frac{SQRT(3)}{2}\right)$$

```
(C24) %, expand;
```

$$(D24)\qquad ASIN\left(\frac{SQRT(3)}{2}\right)$$

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

```
(C15) sin(u+v)*cos(u)^3;
```

$$(D15)\qquad \cos^3(u)\sin(v+u)$$

```
(C16) trigexpand(%);
```

$$(D16)\qquad \cos^3(u)(\cos(u)\sin(v)+\sin(u)\cos(v))$$

Функція `trigreduce` приведе тригонометричний вираз до суми

елементів, кожний з який містить єдиний sin чи cos:

```
(C17) trigreduce(%);
```

$$\text{(D17)} \qquad \frac{\sin(v+4u)+\sin(v-u)}{8}+\frac{3\sin(v+2u)+3\sin(v)}{8}$$

Для обчислення натурального логарифма використовується функція log:

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

$$\text{(D27)} \qquad \log(a^5b^6)$$

№	Функція	Дія
1	part(exp)	Виділення будь-якої заданої частини виразу
2	first(f)	Повертає перший елемент виразу
3	last(f)	Повертає останній елемент виразу
4	rest(f)	Повертає вираз з вилученим першим елементом
5	factor(exp)	Спрощує вираз
6	expand(exp)	Розкриття дужок
7	ev(exp)	Дозволяє одержати чисельне значення виразу
8	bfloat(exp)	Виведення числа в експонентній формі
9	fpprec	Кількість значущих цифр у поданні числа
10	block	Вивід зі збільшеною точністю для окремих команд
11	sin()	Синус
12	cos()	Косинус
13	tan()	Тангенс
14	cot()	Котангенс
15	asin()	Арксинус
16	acos()	Арккосинус
17	atan()	Арктангенс
18	acot()	Арккотангенс
19	sec()	Секанс
20	csc()	Косеканс
21	trigexpand (exp)	Використовує формули перетворення сум двох кутів для спрощення виразу
22	trigreduce (exp)	Перетворює тригонометричний вираз на суму елементів, кожен з яких містить єдиний sin чи cos
23	log (exp)	Обчислення натурального логарифма

5. Поліноми та алгебраїчні перетворення

Поліномом називають вираз, що складається з декількох частин одного виду: $P(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + ... + a_n x^n$.

Над поліномами можна виконувати наступні арифметичні операції: додавання, віднімання, множення та ділення.

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

(D1) $x^2 - 1$

```
(C2) p2:x-1;
```

(D2) $x - 1$

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

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

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

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

```
(C5) p1/p2;
```

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

```
(C6) p1-p2;
```

(D6) $x^2 - x$

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

```
(C7) divide(p1*p2, p1);
(D7)                          [x - 1, 0]
```

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

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

(D8) $x - 1$

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

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

Функція `gfactor` здійснює розклад полінома, використовуючи комплексні числа

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

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

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

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

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

Функція `ratsimp` виносить за дужки найбільший спільний дільник:

```
(C12) ratsimp(%);
```

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

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

```
(C13) sqrt(x^2);
(D13)                          ABS(x)
(C14) assume(x<0);
(D14)                          [x < 0]
(C15) sqrt(x^2);
(D15)                           - x
```

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

```
(C16) forget(x<0);
(D16)                          [x < 0]
(C17) sqrt(x^2);
(D17)                          abs(x)
```

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

```
(C18) z1:-3+%i*4;
(D18)                          4 %i - 3
(C19) z2:4-2*%i;
(D19)                          4 - 2 %i
(C20) z1*z2;
(D20)                          (4 - 2 %i) (4 %i - 3)
(C21) expand(%);
(D21)                          22 %i - 4
(C22) realpart(''c20);
(D22)                          - 4
(C23) imagpart(''c20);
(D23)                          22
```

Функція `denom` виділяє знаменник, а `num` – чисельник виразу:

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

(D24) $$x-1$$

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

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

№	Функція	Дія
1	divide	Добування частки і залишку від ділення одного полінома на інший
2	gcd	Визначає найбільший спільний дільник поліномів
3	factor	Здійснює розклад полінома на множники (для чисел – канонічний розклад)
4	gfactor	Здійснює розклад полінома, використовуючи комплексні числа
5	ratsimp	Виносить за дужки найбільший спільний дільник
6	realpart	Повертає дійсну частину комплексного виразу
7	imagpart	Повертає уявну частину комплексного виразу
8	assume	Враховує додаткові умови, що задаються нерівностями
9	forget	Знімає всі обмеження, накладені за допомогою assume
10	denom	Виділяє знаменник виразу
11	num	Виділяє чисельник виразу

6. Розв'язання рівнянь

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

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

При розв'язанні тригонометричних рівнянь видається тільки одне з нескінченної множини можливих рішень:

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

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

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

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

Якщо рівняння не має рішень на множині дійсних чисел, то Maxima шукає рішення серед комплексних чисел:

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

Якщо ми хочемо одержати рішення рівняння в чисельному виді, після функції `solve` варто використовувати опцію `numer` або `float`.

```
(C8) b:x^5+8*x^4+31*x^3+80*x^2+94*x=-20;
```

(D8) $$x^5+8x^4+31x^3+80x^2+94x=-20;$$

```
(C10) solve(b,x),numer;
(D10)        [x = - sqrt(3) - 2, x = sqrt(3) - 2, x = - 2,
              x = - 3 %i - 1, x = 3 %i - 1]
(C11) solve(b,x),float;
RAT replaced 3.464101615137754 by 8733//2521 =
3.464101547005157
RAT replaced 3.464101615137754 by 8733//2521 =
3.464101547005157
(D11)  [x = - 3.732050773502579, x = - 0.26794922649742,
```

```
                    x = - 2, x = - 3 %i - 1, x = 3 %i - 1]
```

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

```
(C12) allroots((1+2*X)^3 = 13.5*(1+X^5));
(D12)  [X = 0.82967499021294, X = - 1.015755543828121,
        X = 0.96596251521964 %i - 0.40695972319241,
        X = - 0.96596251521964 %i - 0.40695972319241,
        X = 1.0]
```

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

```
(C13) realroots(-1-x+x^5,5.0e-6)
(D13)                          x = 612003/524288
(C14) ev(%[1],float)
(D14)                          X = 1.167303085327148
(C15) ev(-1-X+X^5,%)
(D16)                          - 7.396496210176906E-6
```

Перевірку рішення потрібно виконувати у такий спосіб:

```
(C17) result:solve (x^2=1, x);
(D17)                          [x=-1,x=1]
(C18) x^2=1,result;
(D18)                          1=1
```

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

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

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

Наприклад, знайдемо наближене рішення рівняння $e^x=x^2$. Побудуємо графіки функцій e^x і x^2, виконавши команду `plot2d([%e^x, x^2], [x, -1, 1])`.

Ми одержимо зображення графіків функцій e^x і x^2 на одному кресленні для значень аргументу x, що змінюється на відрізку [–1, 1].

Дивлячись на графіки, бачимо, що за рішення буде узята точка перетину цих графіків $x \approx -0.7$ (рис. 7).

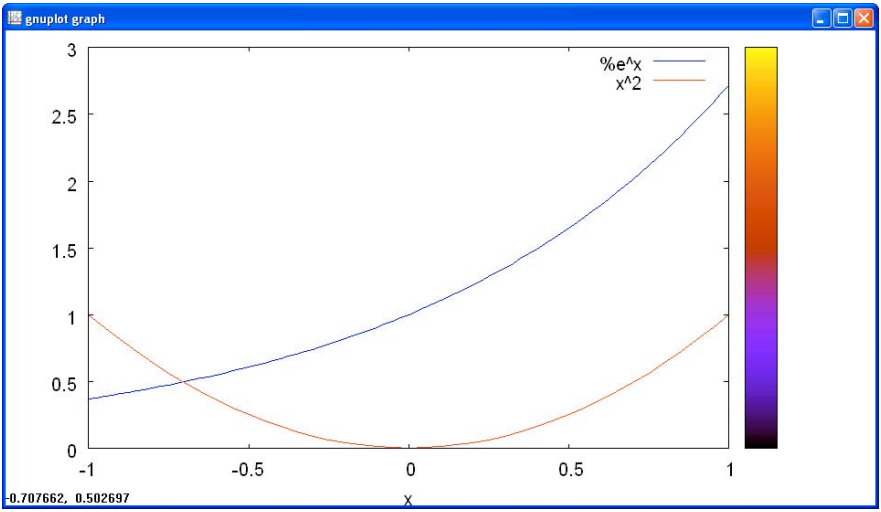


Рис. 7

№	*Функція*	*Дія*
1	`solve`	Розв’язує рівняння та системи алгебраїчних рівнянь
2	`allroots`	Обчислює всі дійсні та комплексні корені поліноміальних рівнянь
3	`realroots`	Шукає всі дійсні корені (комплексні не беруться до увагу) поліноміального виразу
4	`lhs(exp)`	Повертає ліву частину виразу
5	`rhs(exp)`	Повертає праву частину виразу
6	`plot2d`	Побудова графіків

7. Операції математичного аналізу

Maxima може обчислювати похідні й інтеграли, обчислювати межі і знаходити точні рішення звичайних диференціальних рівнянь.

До числа найбільш часто використовуваних математичних операцій належить обчислення похідних функцій як в аналітичному, так і в символьному вигляді. Для добування похідної використовується функція `diff`, першим аргументом якої є функція, другим – змінна, по якій виконується диференціювання, і третім (необов'язковим) – порядок похідної:

```
(C1) f:(x-2*sqrt(x))/x^2;
```

(D1) $$\frac{x-2\sqrt{x}}{x^2}$$

```
(C2) diff(f, x);
```

(D2) $$\frac{1-\frac{1}{\sqrt{x}}}{x^2}-\frac{2(x-2\sqrt{x})}{x^3}$$

```
(C3) expand(' 'c2);
```

(D3) $$\frac{3}{x^{\frac{5}{2}}}-\frac{1}{x^2}$$

```
(C4) g:x^6;
```

(D4) x^6

```
(C5) diff(g, x, 1);
```

(D5) $6x^5$

```
(C6) diff(g, x, 4);
```

(D6) $360x^2$

При обчисленні кратних похідних по декількох змінних після указання функції перелічуються змінні диференціювання з указівкою відповідних кратностей, наприклад

```
(C7) diff(x^6*y^3, x, 4, y, 2);
```

(D7) $2160\,x^2\,y^2$

Одна з найважливіших операцій – обчислення первісних і визначених інтегралів у символьному виді. Первісна – це функція $F(x)$, що задовольняє рівнянню

$$\int f(x)dx = F(x) + C,$$

де C – стала інтегрування. Обчислення визначеного інтеграла з межами – верхньою b і нижньою a – виконується по формулі

$$\int_a^b f(x)dx = F(b) - F(a)$$

Визначений інтеграл може бути представлений як аналітичним, так і чисельним значенням. Для обчислення значень визначених інтегралів розроблений ряд наближених методів – від простих (прямокутників і трапецій) до складних, що автоматично адаптуються до характеру зміни підінтегральної функції $f(x)$.

Для інтегрування в системі Maxima використовуються наступні функції:

`integrate(f,x)` – повертає первісну (невизначений інтеграл) підінтегральної функції `f` по змінній `x`.

`integrate(f,x,min,max)` – повертає значення визначеного інтеграла з межами від `min` до `max`.

```
(C8) f:x^2/(4*x^6+1);
```

(D8) $$\frac{x^2}{4x^6+1}$$

```
(C9) integrate(f, x);
```

(D9) $$(D9)\quad \frac{\operatorname{atan}(2x^3)}{6}$$

У випадку неоднозначної відповіді Maxima може задавати додаткові питання, як у наступному прикладі:

```
(C10) integrate(x^n,x);
Is  n + 1  zero or nonzero?
nonzero;
```

(D10) $$\frac{x^{n+1}}{n+1}$$

```
(C11) integrate(x^n,x);
Is  n + 1  zero or nonzero?
zero;
(D11)                          log(x)
```

Можна використовувати функцію `assume` для задання додаткових умов (не забувайте потім видалити накладені обмеження):

```
(C12) assume(notequal(n,-1));
(D12)                       [not equal(n, - 1)]
(C13) integrate(x^n,x);
```

(D13) $$\frac{x^{n+1}}{n+1}$$

```
(C14) forget(notequal(n,-1));
```

```
(D14)                    [not equal(n, - 1)]
(C15) integrate(x^n,x);
Is  n + 1  zero or nonzero?
zero;
(D15)                    log(x)
```

Для знаходження визначеного інтегралу варто вказати додаткові аргументи – межі інтегрування:

```
(C16) integrate(x^2, x, 0, 6);
(D16)                    72
(C17) integrate(sin(x), x, 0, %pi);
(D17)                    2
```

Maxima здатна обчислювати навіть кратні інтеграли з фіксованими і змінними верхньою чи нижньою межами. Кратний, наприклад подвійний, інтеграл з фіксованими межами має вид:

$$\int_a^b \int_c^d f(x,y)dxdy$$

Наступний приклад при двох форматах уведення виконує обчислення подвійного невизначеного інтеграла подвійним застосуванням функції `integrate`:

```
(C18) integrate(integrate(x^3+y^3,x), y);
```

(D18) $$\frac{x^4 y}{4}+\frac{x y^4}{4}$$

Maxima допускає задання і нескінченних межі інтегрування. Для позначення нескінченності використовується змінна `inf`:

```
(C19) integrate(1/x^2, x, 1, inf);
(D19)                    1
(C20) integrate(1/(1+x^2), x, -inf, inf);
(D20)                    %pi
(C21) integrate(1/x, x, 0, inf);
Integral is divergent  -- an error.
Quitting. To debug this try DEBUGMODE(true);)
```

В останньому прикладі система повідомила про неможливість обчислення інтеграла, тому що він розходиться (*is divergent*).

При обчисленні досить складних інтегралів відповідь не завжди буде представлена у найбільш простому вигляді. У наступному прикладі Maxima не може в символьному виді одержати відповідь, рівну $\pi/4$:

```
(C22) g:1/sqrt(2-x^2);
```

(D22) $$\frac{1}{\sqrt{2-x^2}}$$

```
(C23) integrate(g,x, 0,1);
```

(D23) $$\operatorname{asin}\left(\frac{\sqrt{2}}{2}\right)$$

Існують особливі випадки обчислення інтегралів, коли при обчисленні потрібно враховувати деякі умови.

```
(C38) integrate(1/(sqrt(1-x^n)),x,0,2);
Is  n  an integer?
Потрібно вибрати один із трьох варіантів відповіді:
Acceptable answers are Yes, Y, No, N, Unknown, Uk
Y;
```

Далі дати ще одну відповідь про те, чи є *n* позитивним, негативної або рівним 06

```
Is  n  positive, negative, or zero?
positive;
```

У результаті одержимо одне з можливих рішень:

(D38) $$\int_0^2 \frac{1}{\sqrt{1-x^n}}\,dx$$

Серед операцій математичного аналізу насамперед треба відзначити знаходження сум:

$$\sum_{i=\text{imin}}^{\text{imax}} f_i \ .$$

У цих операціях індекс *i* приймає цілі значення від мінімального (початкового) *imin* до максимального (кінцевого) *imax* із кроком, рівним +1. Суми і добутки легко обчислюються математичними системами, такі обчислення просто описуються на всіх мовах програмування. Однак важливою рисою системи символьної математики є обчислення сум і добутків в аналітичному виді (якщо це можливо) при великій кількості членів – аж до нескінченності.

Для обчислення скінчених і нескінченних сум варто записати суму в символьному виді, після чого спростити отриманий вираз:

```
(C24) sum(1/n^2,n,1,inf);
```

(D24) $$\sum_{n=1}^{\infty} \frac{1}{n^2}$$

```
(C25) %,simpsum;
(D25)                          %pi^2/6
```

Операція обчислення добутків $\prod_{i=i\,\text{min}}^{i\,\text{max}} f_i$ представлена функцією `product`.

Наприклад, обчислимо добуток $\prod_{1}^{5} x+i^2$:

```
(C63) product(x+i^2,i,1,5);
(D63)          (x + 1) (x + 4) (x + 9) (x + 16) (x + 25)
(C63) expand(%);
```

(D63) $x^5+55x^4+1023x^3+7645x^2+21076x+14400$

Багато функцій при наближенні аргументу до деякого значення чи до деякої області значень прагнуть до визначеної *межі*. Так, функція $sin(x)/x$ при x, що прагне до нуля ($x\to 0$), відповідає межа 1 у вигляді усуваної невизначеності 0/0.

Межею деяких функцій може бути нескінченність, тоді як багато функцій прагнуть до скінченної межі при аргументі x, що прагне до нескінченності. Система Maxima не тільки чисельно знаходить межі функцій, заданих аналітично, але і дозволяє знайти межі у вигляді математичного виразу. Для обчислення меж використовується функція `limit`.

```
(C28) limit(1/x,x,inf);
(D28)                          0
```

Для обчислення однобічних меж використовується додатковий параметр, що приймає значення `plus` для обчислення межі праворуч і `minus` – ліворуч.

Наприклад, дослідимо на неперервність функцію arctg(1/(x–4)). Ця функція не визначена в точці x=4. Обчислимо межі праворуч і ліворуч:

```
(C28) limit(atan(1/(x-4)), x, 4, plus);
```

(D28) $\frac{\%pi}{2}$

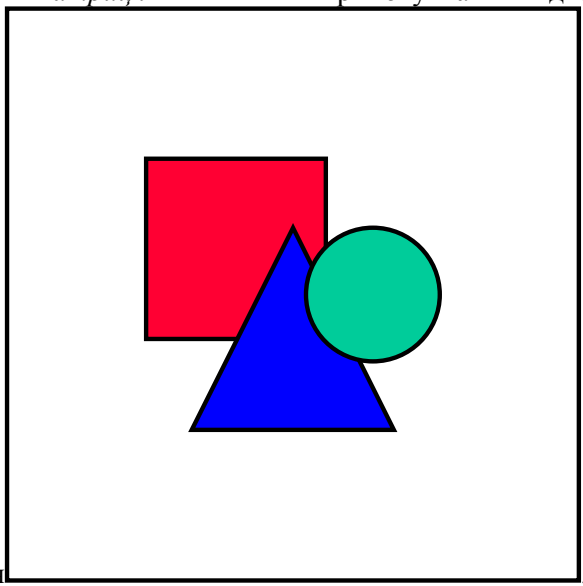
```
(C29) limit(atan(1/(x-4)), x, 4, minus);
```

(D29) $\frac{\%pi}{2}$

Як бачимо, точка x=4 є точкою розриву I роду для даної функції, тому що існують межі ліворуч і праворуч, рівні $-\pi/2$ і $\pi/2$ відповідно.

№	*Функція*	*Дія*
1	`diff`	Знаходить похідну функції
2	`integrate`	Знаходить інтеграл
3	`sum`	Обчислення скінчених і нескінченних сум
4	`product`	Обчислює добутків
5	`limit`	Обчислення меж

8. Матричні обчислення

Матриця – прямокутна двовимірна таблиця  , що містить *m* рядків і *n* стовпців елементів, кожен з яких може бути представлений числом, сталою, змінною, символьним чи математичним виразом.

Квадратна матриця – матриця, у якої число рядків рядків *m* дорівнює числу стовпців *n*.

Сингулярна (вироджена) матриця – квадратна матриця, у якої детермінант (визначник) дорівнює 0. Така матриця звичайно не спрощується при символьних обчисленнях.

Одинична матриця – це квадратна матриця, у якої діагональні елементи рівні 1, а інші елементи рівні 0.

Транспонована матриця – квадратна матриця, у якої стовпці і рядки міняються місцями.

Обернена матриця – це матриця M^{-1}, що, будучи помножена на вихідну квадратну матрицю M, дає одиничну матрицю.

Східчаста форма матриці відповідає умовам, коли перший ненульовий елемент у кожнім рядку є 1 і перший ненульовий елемент кожного рядка з'являється праворуч від першого елемента в попередній рядку, тобто всі елементи нижче першого ненульового у рядку – нулі.

Ранг матриці – найбільший з порядків відмінних від нуля мінорів

квадратної матриці.

Слід матриці – сума діагональних елементів квадратної матриці.

Визначник матриці – це поліном від елементів квадратної матриці, кожен член якого є добутком *n* елементів, узятих по одному з кожного рядка і кожного стовпця зі знаком добутку, заданим парністю перестановок:

$$\det A = \sum a_{1j}(-1)^{j+1} M_1^{<j>},$$

де $M_1^{<j>}$ – визначник матриці порядку *n*–1, отриманої з матриці A викреслюванням першого рядка і *j*-го стовпця.

Maxima дозволяє легко маніпулювати матрицями. У наступному прикладі задаються дві матриці, що потім складаються (+) і перемножуються (.):

```
(C1) A:matrix([1,2],[3,4]);
```

(D1)
$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
(C2) B:matrix([1,1],[1,1]);
```

(D2)
$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

```
(C3) A + B;
```

(D3)
$$\begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix}$$

```
(C4) A . B;
```

(D4)
$$\begin{bmatrix} 3 & 3 \\ 7 & 7 \end{bmatrix}$$

Функція `determinant` обчислює визначник матриці.

```
(C5) determinant(A);
(D5)                          - 2

(C6) determinant(matrix([a,b],[c,d]));
(D6)                          a d - b c
```

Транспонування матриці здійснюється функцією `transpose`.

```
(C7) transpose(A);
```

(D7)
$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$$

Для одержанню оберненої матриці використовується операція ^^-1 чи функція `invert`.

```
(C8) A^^-1;
```

(D8)
$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

```
(C9) invert(A);
```

(D9)
$$\begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$$

Як відомо, кожен елемент b_{ij} оберненої матриці $B=A^{-1}$ отримується діленням алгебраїчного доповнення A_{ij} відповідного елемента вихідної матриці на її визначник |A|. Для того, щоб винести 1/|A| як співмножник, застосовується функція `detout`.

```
(C10) invert(A), detout;
```

(D10)
$$-\frac{\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}}{2}$$

Переконаємося в правильності отриманого результату, помноживши A на обернену до неї матрицю:

```
(C11) A . d9;
```

(D11)
$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

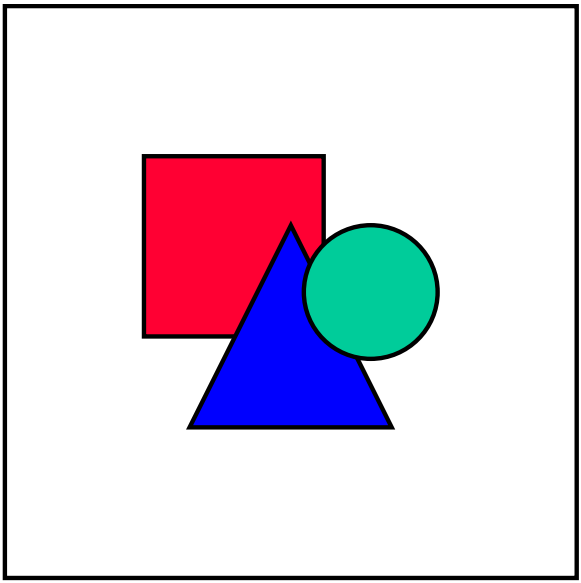
Будьте уважні: у результаті виконання операції ^-1 отримається матриця, кожен елемент якої буде обернений до елементу вихідної, а не обернена матриця.

```
(C12) A^-1;
```

(D12)
$$\begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{4} \end{bmatrix}$$

Використання матриць дозволяє легко розв'язувати системи лінійних рівнянь з декількома перемінними. Нехай A – матриця коефіцієнтів системи, X – матриця невідомих, B – матриця вільних членів системи. Тоді матриця X знаходиться за формулою $X=A^{-1}.B$, де операція "." означає матричне множення.

Наприклад, розв'яжемо наступну систему рівнянь матричним способом.



Спочатку заповнимо відповідні матриці, а потім одержимо матрицю результатів:

```
(C14) A:matrix([1, 2, 1], [2, 1, 1], [1, 3, 1]);
```

$$\text{(D14)}\quad \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 1 \\ 1 & 3 & 1 \end{bmatrix}$$

```
(C15) B:matrix([0, 1, 0]);
(D15)                    [ 0  1  0 ]
(C16) (A^^-1).B;
```

$$\text{(D16)}\quad \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

Функція `ematrix(m,n,x,i,j)` створює матрицю, у якій всі елементи дорівнюють нулю, крім елемента на позиції i, j, який має значення x.

```
(C17) ematrix(3,3,2,1,1);
```

$$\text{(D17)}\quad \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Функція `diagmatrix n,x)` утворить діагональну матрицю розмі-

ром *n* і діагональним елементом, рівним *x*.

```
(C18) diagmatrix (3, 1) ;
```

$$\text{(D19)} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Функція `ident(n)` формує діагональну матрицю.

```
(C20) ident(3);
```

$$\text{(D20)} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Функція `minor(m,i,j)` формує матрицю, в який видаляє рядок *i* і стовпець *j*.

```
(C21) A:matrix([1,2],[3,4]);
```

$$\text{(D21)} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
(C22) minor(A,1,1);
(D22)                [ 4 ]
```

Функція `row(m,i)` повертає з матриці *m* *i*-ий рядок.

```
(C23) A:matrix([1,0,3],[3,4,3],[5,6,7]);
```

$$\text{(D23)} \quad \begin{bmatrix} 1 & 0 & 3 \\ 3 & 4 & 3 \\ 5 & 6 & 7 \end{bmatrix}$$

```
(C24) row(A,3);
(D24)                  [ 5  6  7 ]
```

Функція `zeromatrix (m,n)` формує нульову матрицю розміром mxn.

```
(C25) zeromatrix (3,3) ;
```

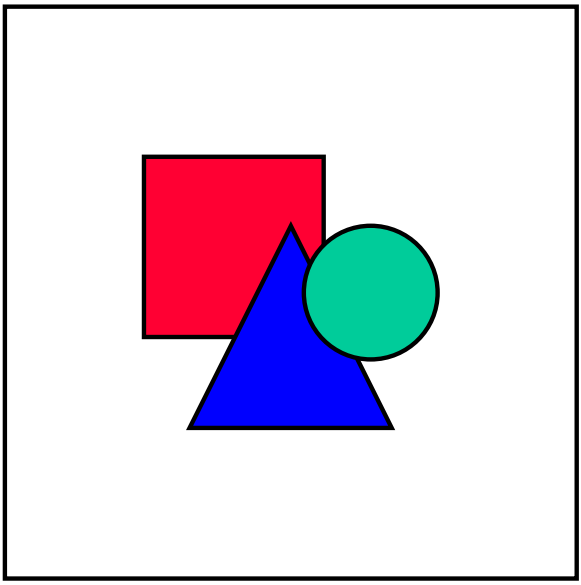
$$\text{(D25)} \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Функція `col(m,j)` повертає *j*-ий стовпець матриці *m*.

```
(C26) col(A,1) ;
```

(D26) $\begin{bmatrix} 1 \\ 3 \\ 5 \end{bmatrix}$

№	Функція	Дія
1.	`determinant`	Обчислює визначник матриці
2.	`transpose`	Транспонування матриці
3.	`invert` чи `^^-1`	Обчислення оберненої матриці
4.	`detout`	Для того щоб винести 1/|A| як співмножник
5.	`ematrix(m,n,x,i,j)`	Створює матрицю, у якій всі елементи дорівнюють нулю, крім елемента який коштує на позиції i,j і має значення x.
6.	`diagmatrix(n,x)`	Утворить діагональну матрицю розміром n і діагональним елементом рівним x.
7.	`ident(n)`	Формує діагональну матрицю
8.	`minor(m,i,j)`	Формує матрицю в який видаляє i рядок і j стовпець
9.	`row(m,i)`	Повертає з матриці m і-ю рядок
10.	`zeromatrix(m,n)`	Формує нульову матрицю розміром mxn.



9. Списки та масиви

Списки мають велике значення при обробці масивів даних і є основою для створення векторів і матриць.

Часто математичні чи інші об'єкти містять множину даних, що бажано об'єднувати під загальним ім'ям. Наприклад, під об'єктом з ім'ям M можна мати на увазі квадратну матрицю розміром 10x10 із загальним числом елементів, рівним 100. Людину з ім'ям Victor, наприклад, можна характеризувати цілим списком різних даних – символьними прізвищем, ім'ям і по батькові, цілим роком народження, дійсними ростом, вагою тощо.

Для об'єднання даних можуть використовуватися *списки* (list). Maxima має великі можливості роботи з об'єктами-списками, що містять не тільки однотипні, а й різнотипні елементи. Зокрема, елементами списків можуть бути числа, константи, змінні, вирази і навіть самі списки. Списки використовуються для конструювання інших типів даних – масивів, матриць і векторів.

Мовою даної системи список – це сукупність довільних даних у квадратних дужках, наприклад:

```
(C1) victor:[petrov,victor,victorovich,1.68,100];
(D1)                [petrov, victor, victorovich, 1.68, 100]
```

Списки характеризуються *розміром*, що являє собою добуток числа елементів по кожнім напрямку (*розмірності*). Наприклад, одновимірний список є *вектором* і характеризується числом елементів по одному напрямку. При цьому вектор може бути рядком чи вектором-стовпцем. Двовимірний список являє собою *матрицю*.

Для генерації списків з елементами, що є дійсними і цілими числами, чи навіть цілими виразами, використовується функція `makelist`, що створює список.

Наступний приклад генерує список `f` з 15 елементів, заповнений випадковими цілими числами в діапазоні від 0 до 10.

```
(C2) f:makelist(random(10),c,1,15);
(D2)          [9, 0, 1, 0, 3, 2, 3, 9, 5, 1, 2, 0, 9, 3, 6]
```

Приклад генерації списку з дійсними числами:

```
(C3) f:makelist(random(10)/3.1,c,1,15);
(D3)  [1.290322580645161, 0.64516129032258,
2.258064516129032, 0.96774193548387, 2.903225806451613,
2.580645161290323, 1.935483870967742, 0.0,
2.903225806451613, 2.580645161290323, 0.64516129032258,
0.64516129032258, 0.96774193548387, 0.64516129032258,
0.96774193548387]
```

Приклад генерації списку, що складає з 10 букв a.

```
(C4)f:makelist(a,c,1,10);
(D4)                          [a, a, a, a, a, a, a, a, a, a]
```

Списки відносяться до даних складної структури. Тому при роботі з ними виникає необхідність контролю за структурою, інакше застосування списків може привести до грубих помилок, як явних, що супроводжуються видачею повідомлення про помилку, так і неявних.

listp(exp) – повертає true, якщо exp є списком, у противному випадку повертає false

```
(C5) listp(f);
(D6)                          true
(C6) a:3;
(D6)                          3;
(C7) listp(a);
(D7)                          false
```

length(exp) – повертає число елементів одновимірного списку exp чи число розмірностей у випадку багатовимірного списку.

```
(C8) length(f);
(D8)                          15
```

member(exp,list) – виводить true, якщо змінна exp входить у список list, у противному випадку false.

```
(C9) member(petrov,victor);
(D9)                          true
(C10) member(petrov1,victor);
(D10)                         false
```

Списки можна представити у вигляді особливої структури даних – стека. Стек – це структура даних, що нагадує стопку тарілок у шафі. При цьому тарілки відіграють роль даних. Чергову тарілку можна покласти тільки зверху (на *вершину* стека). На *дні* стека лежить перша поміщена в нього тарілка. Стек підпорядковується наступному правилу: останнє введене значення вилучається першим, а перше введене значення вилучається останнім. Стек відноситься до структур даних динамічного типу, його розміри міняються в процесі обчислень. Стік може бути порожнім, якщо з нього вилучити всі дані.

Maxima представляє наступні можливості роботи зі стеками:

cons(exp,list) – додає в початок списку list елемент exp:

```
(C11) cons(35,victor(petrov,victor,victorovich,1.68,100));
(D11)        [35, petrov, victor, victorovich, 1.68, 100]
```

endcons(exp,list) – додає в кінець списку list елемент exp:

```
(C12) endcons(medik,victor);
```

```
(D12)  [petrov, victor, victorovich, 1.68, 100, medik]
```

`last(list)` – повертає останній елемент списку `list`:

```
(C13) last(victor);
(D13)                    100
```

`first(list)` – повертає перший елемент списку `list`:

```
(C14) first(victor);
(D14)                    petrov
```

`rest(exp,n)` – видаляє зі списку `n` елементів:

```
(C15) rest(victor,1);     //видаляє один елемент списку
(D15)                     [victor, victorovich, 1.68, 100]
(C16) rest(victor,5);     // видаляє всі елементи списку
(D16)                     []
```

Тривіальна процедура спілкування зі стеком (додати/вилучити дані) обмежує можливості стекових операцій. З життєвого досвіду ми знаємо, що, виявивши наполегливість, можна вставити тарілку й у середину стопки. Maxima надає ряд розширених можливостей для роботи зі списками, що виходять за рамки звичайних стекових операцій.

Так, наприклад, для розширення списку шляхом включення в нього нових елементів використовуються наступні функції:

`append(list1,list2,list3,…)` – поєднує списки в зазначеному порядку:

```
(C17) append(f,victor);
(D17)              [0.32258064516129, 2.580645161290323,
2.903225806451613, 0.64516129032258, 2.903225806451613,
2.258064516129032, 1.612903225806452, 1.935483870967742,
1.612903225806452, 0.0, 1.935483870967742,
2.903225806451613, 0.0, 0.32258064516129, 0.32258064516129,
petrov, victor, victorovich, 1.68, 100]
```

`delete(exp1,exp2)` – видаляє елемент `exp1` зі списку `exp2`:

```
(C18) delete(petrov,victor);
(D18)                     [victor, victorovich, 1.68, 100]
```

Крім додавання в список нових даних, є можливість зміни порядку елементів у списку.

`reverse(list)` – повертає список зі зворотним порядком розташування елементів:

```
(C19) reverse(victor);
(D19)              [100, 1.68, victorovich, victor, petrov]
```

`sort(list)` – сортує елементи списку `list` у канонічному порядку:

```
(C20) sort(victor);
```

```
(D20)                [1.68, 100, petrov, victor, victorovich]

(C21) sort(f);
(D21)        [0.0, 0.0, 0.32258064516129, 0.32258064516129,
0.32258064516129, 0.64516129032258, 1.612903225806452,
1.612903225806452, 1.935483870967742, 1.935483870967742,
2.258064516129032, 2.580645161290323, 2.903225806451613,
2.903225806451613, 2.903225806451613]
```

Лінійна алгебра – один з фундаментальних розділів математики. Він багато в чому сприяв розвитку методів обчислень. Засоби лінійної алгебри (перетворення матриць, розв'язання систем лінійних рівнянь тощо) широко використовуються при рішенні задач механіки, електро- і радіотехніки й інших галузей науки і техніки.

Наступна група функцій системи Maxima дозволяє здійснювати над векторами основні операції лінійної алгебри.

Лінійні дії над векторами в системі Maxima задаються у вигляді $a+b$, $a-b$, $\alpha*v$, де v – деяка константа.

`a*b*c*…` – повертає векторний добуток векторів:

```
(C23) a:[1,2,3];
(D23)                        [1, 2, 3]
(C24) b:[4,5,6];
(D24)                        [4, 5, 6]
(C25) c:[7,8,9];
(D25)                        [7, 8, 9]
(C26) a*b*c;
(D26)                        [28, 80, 162]
```

`innerproduct(a,b,c,…)` або `a.b.c…` – повертає скалярний добуток векторів:

```
(C27) a.b.c;
(D27)                        [122, 244, 366]
```

Мішаний добуток векторів a, b, c можна задавати за допомогою `matrix`:

```
(C28) v:matrix(a,b,c);
```

$$\text{(D28)} \qquad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

`determinant(v)` – повертає визначник матриці `v`, рядками якої є координати векторів a, b, c відповідно.

Сукупність даних утворює *масив* (array). Масиви можуть бути одновимірними (один список), двовимірними і багатовимірними (два

чи більш списків). Одновимірні масиви називають *векторами*, двовимірні – *матрицями*. У загальному випадку масив характеризується *розмірністю* (числом вимірів) і *розміром* – числом елементів по всім вимірам.

Масив створюється оператором `array(name,exp1,exp2,exp3…)`

```
(C29)d:array[1,2,3,4,5,6,7,8,9,0,1,2];
(D29)                          array1,2,3,4,5,6,7,8,9,0,1,2
```

№	*Функція*	*Дія*
1	`makelist`	Створює список
2	`listp(exp)`	Повертає `true`, якщо `exp` є списком, у противному випадку повертає `false`
3	`length(exp)`	Повертає число елементів одномірного списку `exp` чи число розмірностей у випадку багатовимірного списку
4	`member(exp,list)`	Виводить `true`, якщо змінна `exp` входить у список `list`, у противному випадку `false`
5	`cons(exp,list)`	Додає в початок списку `list` елемент `exp`
6	`endcons(exp,list)`	Додає в кінець списку `list` елемент `exp`
7	`last(list)`	Повертає останній елемент списку `list`
8	`first(list)`	Повертає перший елемент списку `list`
9	`rest(exp,n)`	Видаляє зі списку `n` елементів
10	`append(li1,li2,li3,…)`	Поєднує списки в зазначеному порядку
11	`delete(exp1,exp2)`	Видаляє елемент `exp1` зі списку `exp2`
12	`reverse(list)`	Повертає список зі зворотним порядком розташування елементів
13	`sort(list)`	Сортує елементи списку `list` у канонічному порядку
14	`a*b*c`	Повертає векторний добуток векторів
15	`innerproduct(a,b,c,…)` або `a.b.c…`	Повертає скалярний добуток векторів
16	`array(name,exp1,exp2, exp3…)`	Оголошення масиву

10. Диференціальні рівняння

Одна із поширених математичних задач – розкладання заданої аналітичної функції в степеневий ряд Тейлора відносно деякої вузлової точки з абсцисою x_0. Такий ряд нерідко простіше самої функції (у тому сенсі, що він не вимагає обчислення навіть елементарних функцій і обчислюється за допомогою лише арифметичних операцій) і дає однакове подання для функцій, що розкладаються, у вигляді звичайних степеневих поліномів.

Більшість достатньо гладких функцій, що не мають розривів в області розкладання, досить точно відтворюються рядом Тейлора. Як правило, такі розкладання достатні прості в околицях вузлової точки.

Maxima здатна знаходити розкладання функцій у ряд Тейлора. Наприклад, одержимо поліном Тейлора порядку 4 для функції `f(x)=ln x` у точці `x=1`:

```
(C26) g:log(x);
(D26)                          log(x)
(C27) taylor(g,x,1,4);
(D27)                  x-1-(x-1)^2/2+(x-1)^3/3-(x-1)^4/4+...
```

Диференціальними прийнято називати рівняння, до складу яких входять похідні функції *y(x)*, що представляє рішення рівняння. Диференціальні рівняння можуть бути представлені в різній формі, наприклад, у загальновідомій формі Коші:

$$y'(x) = f(x, y)$$

Найвищий порядок похідної чи диференціала, що входять у диференціальне рівняння, називають *порядком* цього диференціального рівняння.

Наприклад, $ydx - xdy = 0$ – диференціальне рівняння першого порядку, $2xy'' - 8y^4y' = 16x^6$ – диференціальне рівняння другого порядку, $y''' - 6y'' + 5y' = 10x^8e^x$ – диференціальне рівняння третього порядку.

Диференціальні рівняння і системи диференціальних рівнянь можуть бути лінійними і нелінійними. Для лінійних рівнянь звичайно існують рішення в аналітичному виді. Нелінійні диференціальні рівняння в загальному випадку аналітичних рішень не мають, але можуть розв'язуватися наближеними чисельними методами.

Диференціальні рівняння широко використовуються в практиці

математичних обчислень. Вони є основою розв'язання задач моделювання – особливо в динаміці.

Для розв'язання диференціальних рівнянь у символьному виді використовується функція `desolve`.

Наприклад, потрібно вирішити диференціальне рівняння наступного виду: $y'' - y' + 2y = (5x - 14)e^{2x}$.

```
(C1) s:'diff(y(x),x,2)-'diff(y(x),x)+2*y=(5*x-14)*%e^(2*x);
```

(D1) $$2y + \frac{d^2}{dx^2}y(x) - \frac{d}{dx}y(x) = (5x-14)\%e^{2x}$$

```
(C2) desolve(s,y(x));
```

(D2) $$y(x) = \frac{8y - 4\left(\frac{d}{dx}y(x)\Big|_{x=0}\right) + 4y(0) - 33}{4} + 2xy + \%e^{2x}\left(-2y + \frac{d}{dx}y(x)\Big|_{x=0} + 19\right) + \frac{5x\%e^{2x}}{2} - \frac{43\%e^{2x}}{4}$$

Розв'язок рівняння одержуємо з використанням початкових умов. Виконавши перетворення і позначивши константи через C1 і C2, одержимо остаточний результат: $y = C_1 e^x + C_2 e^{2x} + \left(\frac{5}{2}x^2 - 19x\right)e^{2x}$

Наступний приклад показує застосування перетворення Лапласа для розв'язання диференціального рівняння $x'''(t)+x(t)=0$:

```
(C1) ode: 'diff(x(t), t, 3) + x(t) = 0;
```

(D1) $$\frac{d^3}{dt^3}x(t) + x(t) = 0$$

```
(C2) atvalue(x(t),t=0,1);  //задаємо x(0)=1
(D2)                       1
(C3) atvalue('diff(x(t),t),t=0,3);   //задаємо x'(0)=3
(D3)                       3
(C4) atvalue('diff(x(t),t,2),t=0,8);   //задаємо x''(0)=8
(D4)                       8
(C5) lap_ode:laplace(ode,t,s); //пряме перетворення Лапласа
(D5)   s^3*laplace(x(t),t,s)+laplace(x(t),t,s)-s^2-3*s-8=0
//розв'язуємо рівняння як алгебраїчне
(C6) sol: solve(%, 'laplace(x(t), t, s));
```

(D6) $$[\text{laplace}(x(t),t,s) = \frac{s^2+3s+8}{s^3+1}$$

```
//обернене перетворення Лапласа
(C7) map( lambda( [eq], ilt(eq, s, t)), sol);
```

(D7)

$$[x(t)=\%e^{t/2}\left(\frac{11\sin\left(\frac{\sqrt{3}t}{2}\right)}{\sqrt{3}}-\cos\left(\frac{\sqrt{3}t}{2}\right)\right)+2\%e^{-t}\ ]$$

```
(C8) plot2d([%e^(t/2)*((11*sin((sqrt(3)*t)/2))/sqrt(3)-
cos((sqrt(3)*t)/2))+2*%e^(-t)],    [t,-5,5])$
```

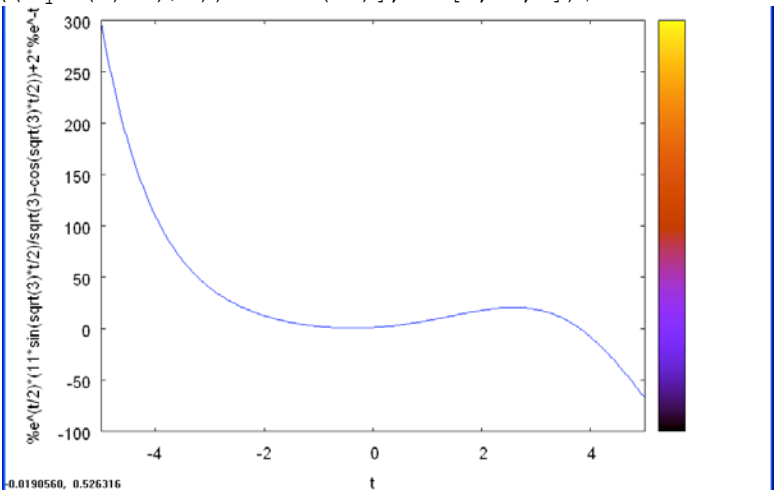


Рис. 8

Для побудови зображень тривимірних об’єктів використовується функція `plot3d`. Наприклад, зобразимо лист Мебіуса (рис. 9):

```
(C2) plot3d([cos(x)*(3+y*cos(x/2)), sin(x)*(3+y*cos(x/2)),
y*sin(x/2)], [x,-%pi,%pi],[y,-1,1],['grid,40,15]);
```

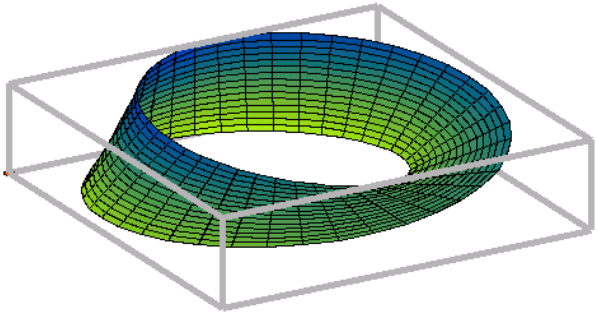


Рис. 9

```
(C3) plot3d(cos(x*y),[x,-3,3],[y,-3,3]);
```

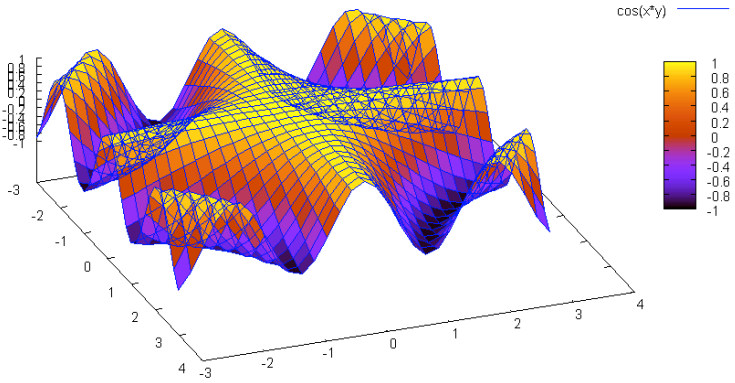


Рис. 10

№	*Функція*	*Дія*
1	`taylor(exp)`	Розкладає функцію в ряд Тейлора
2	`desolve([exp1,exp2…], [var1,var2…])`	Вирішує диференціальні рівняння і системи
3	`plot3d(exp)`	Будує тривимірні графіки

11. Основи програмування

Багато задач у системі Maxima зважуються з використанням лінійних алгоритмів і програм. Вони можуть бути представлені ланцюжком виразів, виконуваних послідовно від початку до кінця.

Однак у більшості випадків серйозні обчислення базуються на використанні циклічних і розгалужених алгоритмів і програм. При цьому, і залежності від проміжних чи вихідних даних, обчислення можуть йти по різних гілках програми, циклічно повторюватися і т.д. Для реалізації розгалужених програм мова програмування повинна містити керуючі структури, тобто спеціальні конструкції мови, що реалізують у програмах розгалуження. Вони використовуються у різних методах програмування, у тому числі процедурному і функціональному програмуванні.

До найважливішим керуючим структурам у мовах програмування відносяться цикли. З їхньою допомогою здійснюється циклічне виконання деякого виразу *expr* задане число раз. Це число нерідко визначається значенням деякої керуючою змінної, що змінюється або з кроком +1, або від початкового значення *imin* до кінцевого *imax* із кроком *di*. Цикли можуть вкладеними один в одного.

Приклад циклу, у якому змінна *a* виводиться на екран, приймає початкове значення –3, змінюється з кроком –7 і виконується до тих пір, поки змінна менше 26. Тут функція `ldisplay` виводить на екран не тільки значення змінної, але й саму змінну, а також надає значення цієї змінної мітці (E1, E2 і т.д.)

```
(C1)   for a:-3 thru 26 step 7 do ldisplay(a)$
(E1)          a = -3
...
(E5)          a = 25
```

Тепер, якщо потрібно, наприклад, звернутися до значення змінної *a*, отриманої на 3 кроці, потрібно набрати E3.

```
(C5) E3;
(D5) A=11
```

Використовуючи функцію `display`, ми одержимо наступний результат (виведемо на екран значення змінної та її ім'я):

```
(C6)   for a:-3 thru 26 step 7 do display(a)$
          a = -3
...
          a = 25
```

Функція `print` видасть на екран тільки значення змінних:

```
(C7) for a:-3 thru 26 step 7 do print(a)$
```

```
- 3
...
25
```

Змінній `step` можна надавати як цілі, так і дробові значення. Наприклад:

```
(C8) for a:-3 thru 0 step 0.25 do print(a)$
- 3
...
- 0.25
0.0
```

Наступний цикл обчислює значення змінної S+I і працює до тих пір, поки змінна I буде менше чи рівною 10.

```
(C9)   S:0$
(C10)   for I:1 while I<=10 do S:S+I;
(D11)          done
(C12)   S;
(D12)          55
```

Наступний приклад показує приклад обчислення S доти, поки I строго менше 10:

```
(C13) for I:1 unless I>10 do S:S+I;
(D13)                                    done
(C14) S;
(D14)                                     55
```

Наступний приклад показує використання вкладених циклів:

```
(C15) POLY:0$
(C16) for I:1 thru 5 do
        for J:I step -1 thru 1 do
           POLY:POLY+I*X^J$
(C17) POLY;
```

(D17) $5x^5+9x^4+12x^3+14x^2+15x$

Цикли дають можливість знайти значення функції на визначеному відрізку з кроком, як цілим, так і дробовим:

```
(C18) for y:-1 thru 1 step 0.5 do( t:y*y, print("t=",t));
t= 1
...
t= 0.25
t= 1.0
(D18)                                    done
```

Як у більшості мов програмування, умовні вирази задаються за допомогою оператора `if`. У системі Maxima також представлений цей оператор.

`if` (умова) `then` вираз1 `else` вираз2.

Наприклад:

```
(C19) x:0;
(D19)                                    0
(C20) if(x=0) then x:x+1 else x:x-1;
(D20)                                    1
```

Умовний оператор можна використовувати усередині циклів, наприклад:

```
(C21) GUESS:-3.0$
(C22) for I:1 thru 10 do (GUESS:SUBST(GUESS,X,.5*(X+10/X)),
        if abs(GUESS^2-10)<.00005 then return(GUESS));
(D22)                  - 3.1622807
```

Цикли можна використовувати, якщо потрібно обчислити і вивести на екран у виді таблиці значення функції `f` на інтервалі від $x_{\text{поч}}$ до $x_{\text{кін}}$ із кроком `d`.

Наприклад, нехай дана функція $f=\begin{cases} ax^2+b, x>0, b\neq 0 \\ \dfrac{x-a}{x-c}, x>0, b=0 \\ \dfrac{x}{c} \end{cases}$. Для її обчислення необхідно задати таку послідовність команд:

```
(C23) x:-2;
(D23)        -2;
(C24) a:2;
(D24)        2;
(C25) b:3;
(D25)        3;
(C26) c:5;
(D26)        5;
(C27) for i:0 while i<=5 step 0.5 do
(x:x+i,
if (x<0 and b!=0) then (f:a*x*x+b,print(f)) else
if (x>0 and b=0) then (f:(x-a)/(x-c),print(f)) else (f:x/c,
print(f)));
20.6
...
25.1
26.1
(D27)        DONE
```

Функція `SUBST(a,b,c)` – вираз `b` позначаємо `a` і підставляємо в `c`. Наприклад,

```
(C23) SUBST(A,X+Y,X+(X+Y)**2+Y);
```
(D23) $Y + X + A^2$

Функція `return(expr)` перериває виконання з висновком значення вираження `expr`.

Досі ми використовували систему Maxima в інтерактивному режимі, подібно калькулятору. Якщо часто доводиться виконувати визначену послідовність обчислень, то краще оформити її у виді програми, що викликається у випадку потреби. Нижче приводиться невелика програма для знаходження критичних точок функції `f(x)`. Користувачу пропонується увести функцію `f`, після чого обчислюється похідна уведеної функції і за допомогою функції `solve` розв'язується рівняння `f(x) = 0`. Програма записується в текстовий файл і потім завантажується в систему Maxima за допомогою функції `batch`. Наведемо текст програми:

```
critpts():=(
   print("Програма знаходження критичних точок"),
/* Запит на уведення функції     */
   print("Уведіть функцію f(x):"),
   f:read(),
/* Друкування уведеної функції (для контролю) */
   print("f = ",f),
/* У змінну eq заносимо значення похідної */
   eq:diff(f,x),
/* Розв'язуємо рівняння */
   solve(eq, x)
)$
```

Програма складається з єдиної функції (без аргументів), що називається `critpts`. Команди відокремлюються комами. Приклад виконання програми:

```
(C1) batch("c:\\work\\critpoints.mac");
batching #p C/work/critpoints.max
(C2) critpts() := (PRINT("Програма #
знаходження критичних точок"),
PRINT("Уведіть функцію f(x):"),
f : READ(),
PRINT("f = ", f),
eq : DIFF(f, x),
SOLVE(eq, x))
(C3) critpts() ;
Програма знаходження критичних точок
Уведіть функцію f(x):
(x+2)/(x^2+1);
```

$$f = \frac{x+2}{x^2+1}$$

```
(D3) [x = - sqrt(5) - 2, x = sqrt(5) - 2]
```

№	*Функція*	*Дія*
1.	`ldisplay`	виводить на екран не тільки значення змінної, а й саму змінну, а також надає значення цієї змінної мітці (E1, E2 і т.д.)
2.	`display`	виводить на екран значення змінної та її ім'я
3.	`for`	покроковий цикл
4.	`while`	умовний цикл
5.	`step`	крок
6.	`print`	виведення на екран
7.	`if`	умовний оператор
8.	`subst(a,b,c)`	вираз `b` позначаємо `a` і підставляємо в `c`
9.	`return(expr)`	перериває виконання з виводом значення виразу `expr`
10.	`batch`	відкриття файлу

Семеріков Сергій Олексійович

Maxima 5.13:

довідник користувача

Підп. до друку 15.09.2007
Папір офсетний №1
Ум. друк. арк. 5,4

Формат 80×84 1/16
Зам. №1-1509
Тираж 100 прим.

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

E-mail: cc@optima.com.ua