# Wprowadzenie do programu Maxima

## O programie Maxima

Maxima jest programem komputerowym typu CAS (Computer Algebra System – system algebry komputerowej), wspomagającym wykonywanie obliczeń matematycznych, zarówno symbolicznych, jak i numerycznych. Zastosowanie rachunku symbolicznego pozwala na rozwiązywanie wielu problemów matematycznych w sposób dokładny. Na przykład licząc całkę oznaczoną o postaci:

$$\int_1^2 \cos x \, dx$$

po wpisaniu integrate(cos(x),x,1,2); otrzymamy rozwiązanie dokładne postaci sin(2)-sin(1).

Program ten wywodzi się z opracowanego w Massachusetts Institute of Technology pod koniec lat 60-tych na zlecenie Departamentu Energii USA programu Macsyma, a umożliwia on m.in.

- wykonywanie obliczeń numerycznych z dowolną dokładnością,
- upraszczanie wyrażeń algebraicznych i trygonometrycznych,
- symboliczne rozwiązywanie równań (w tym różniczkowych),
- symboliczne rozwiązywanie układów równań,
- różniczkowanie i całkowanie symboliczne,
- operacje na macierzach,
- rysowanie wykresów funkcji,
- wykonywanie obliczeń z zakresu rachunku prawdopodobieństwa,
- definiowanie własnych funkcji przez użytkownika,
- programowanie w Lispie,
- eksport otrzymanych wyników do formatów: HTML, LaTeX oraz formatów graficznych: PNG, PostScript.

## Podstawy

Po uruchomieniu Maximy otworzy nam się nowy plik, który zapisujemy (Plik/Zapisz jako) pod wybraną przez nas nazwą w odpowiednim folderze. Plik automatycznie dostaje rozszerzenie *.wxm.

Główne okno programu służy do wprowadzania poleceń, które kończymy średnikiem **;** bądź znakiem dolara **$** Zakończenie polecenia średnikiem powoduje, że Maxima oblicza wyrażenie i wyświetla wynik jego działania. Zakończenie polecenia symbolem dolara sprawia, że Maxima oblicza wyrażenie, przechowuje jego wynik, lecz nie wyświetla go. Program rozróżnia wielkość liter, więc f(x) nie jest równoznaczne F(x). W celu wykonania polecenia stosujemy kombinację klawiszy **Shift+Enter**. Przykładowo chcąc obliczyć wynik dodawania liczb 87 i 63 wprowadzamy 87+63; i zatwierdzamy polecenie kombinacją Shift+Enter:

```
(%i1) 87+63;
(%o1) 150
```

Należy zauważyć, że każde polecenie w Maximie jest numerowe, przy czym wiersze wejściowe są oznaczone: %i1, %i2, %i3 itd. (gdzie i jest skrótem od input), a wiersze wyjściowe są oznaczone: %o1, %o2, %o3 itd. (gdzie o jest skrótem od output). Maxima umożliwia odwołanie do ostatniego wyniku poprzez użycie symbolu % lub do jednego z poprzednich wyników poprzez wykonanie polecenia %k (k to numer wybranego wyniku). Na przykład:

```
(%i1) 87+63$
(%i2) %+10;
(%o2) 160
```

Jeżeli obliczenia przeprowadzamy na liczbach niewymiernych, to wyniki często otrzymujemy w formie symbolicznej:

```
(%i3) sqrt(2);
(%o3) √2
```

Jeżeli wynik obliczeń chcemy wyrazić w postaci dziesiętnej (przybliżonej), to możemy zastosować polecenie **numer**:

```
(%i4) sqrt(2),numer;
(%o4) 1.414213562373095
```

Komentarze w programie Maxima wprowadzamy, używając znaków /* oraz */

```
(%i5) /* Komentarz */ 1+2;
(%o5) 3
```

## Operatory matematyczne

Program akceptuje standardowe operatory arytmetyczne oraz operatory przypisania:

| | |
|---|---|
| dodawanie | + |
| odejmowanie | - |
| mnożenie | * |
| dzielenie | / |
| potęgowanie | ^ lub ** |
| mnożenie macierzy | . |
| silnia | ! |

Przykład:

```
(%i6) (2+2*3!)/(2^2);
(%o6) 3.5
```

## Stałe matematyczne

Maxima posiada zdefiniowane podstawowe stałe matematyczne:

| | |
|---|---|
| %e | stała Eulera e = 2,71828... |
| %pi | liczba π = 3,14159... |
| %i | jednostka urojona i =$\sqrt{-1}$ |
| inf | nieskończoność ∞ |
| minf | minus nieskończoność −∞ |

Przykład:

```
(%i7) %e;
(%o7) e
(%i8) float(%e);
(%o8) 2.718281828459045
```

## Podstawowe funkcje matematyczne

Podstawowe funkcje stosowane w matematyce jakie można znaleźć w Maximie:

| | |
|---|---|
| abs(x) | wartość bezwzględna liczby x |
| entier(x) | całość liczby x |
| sqrt(x) | pierwiastek kwadratowy liczby x |
| log(x) | logarytm naturalny liczby x |
| exp(x) | exponent liczby x |
| sin(x) | sinus liczby x |
| asin(x) | arcus sinus liczby x |

Przykład:

$$10 \cdot \sin(\pi/2)$$

```
(%i9) 10*sin(%pi/2);
(%o9) 10
```

## Definiowanie zmiennych i funkcji

Wyrażenia, tak jak liczby, mogą być przechowywane w zmiennych. Nazwa zmiennej może składać się ze znaków: A do Z, a do z, 0 do 9, % oraz _. Przykłady poprawnych nazw: a, tmp1, s_1. Wartość do zmiennej przypisuje się za pomocą operatora przypisania :. Operator ten przypisuje wartość po prawej strony wyrażenia do zmiennej po lewej stronie. Przykładem może być przypisanie wartości 10 do zmiennej o nazwie x:

```
(%i10)  x:10;
(x)     10
```

Od tej pory nazwą x możemy operować w dalszych obliczeniach tak jak stałą 10. Maxima umożliwia także stworzenie własnych funkcji za pomocą operatora :=. Po nazwie funkcji piszemy zawsze nawiasy (), które zawierają listę argumentów (zmiennych) oddzielonych od siebie przecinkami. Jako przykład utworzymy funkcję realizującą logarytm dziesiętny argumentu x, po czym ją wywołamy sprawdzając poprawność obliczeń:

```
(%i11)  log10(x) := log(x)/log(10)$
(%i12)  log10(10);
(%o12)  1
```

Program umożliwia sterowanie wykonywalnością funkcji poprzez operator apostrofu ‘. Funkcja lub zmienna, którą poprzedza nie jest rozwijana/wykonywana:

```
(%i13)  a:1$
(%i14)  10*’a;
(%o14)  10a
```

Czasami przydatne jest usunięcie zmiennych lub funkcji bez zamykania programu. Dokonujemy tego za pomocą polecenia kill(). Wszystkie zmienne możemy wyczyścić z pamięci za pomocą polecenia kill(all).

## Liczby zespolone

W zakresie działań na liczbach zespolonych przydatnymi funkcjami w Maximie są:

| | |
|---|---|
| cabs(z) | oblicza moduł liczby zespolonej z |
| realpart(z) | podaje część rzeczywistą liczby zespolonej z |
| imagpart(z) | podaje część urojoną liczby zespolonej z |
| conjugate(z) | podaje sprzężenie liczby zespolonej z |
| rectform(z) | podaje liczbę zespoloną z w postaci a + ib, |
| polarform(z) | podaje liczbę zespoloną z w postaci $\|z\|e^{i\theta}$, gdzie $\theta \in \arg z$ |

Przykład:

```
(%i15)  z:1-%i*2$
(%i16)  cabs(z);
(%o16)  √5
(%i17)  polarform(z);
(%o17)  √5%e^(-%i atan(2))
```

## Wektory i macierze

Wektorem (listą) jest wszystko co zostało ujęte w nawiasy kwadratowe **[]**. Pierwszy element wektora ma indeks 1. Funkcja **matrix** tworzy macierz składającą się z n wierszy. Każdy wiersz jest równoliczną listą. Przykład:

```
(%i18)  matrix([a1,b1],[a2,b2]);
(%o18)  [ a1  b1 ]
        [ a2  b2 ]
```

Wybrane funkcje operujące na macierzach:

| | |
|---|---|
| determinant(M) | wyznacznik macierzy M, |
| adjoint(M) | macierz dołączona do M, |

| | |
|---|---|
| invert(M) | macierz odwrotna do M, |
| charpoly(M,x) | wielomian charakterystyczny det(M – xI), |
| rank(M) | rząd macierzy M, |
| transpose(M) | macierz transponowana, |
| mat_trace(M) | ślad macierzy, |
| col(A,k) | zwraca k-tą kolumnę macierzy A, |
| row(A,k) | zwraca k-ty wiersz macierzy |

## Równania i układy równań

Funkcja solve() stara się rozwiązać symbolicznie dowolne równanie lub układ równań. Pierwszym argumentem jest wyrażenie zawierające niewiadomą var. Może być ono dane w trzech postaciach:

- f(x)=g(x)
- f(x)=0
- f(x)

Dwie ostatnie postaci są równoważne. Gdy równanie zawiera tylko jedną zmienną, można opuścić drugi argument. Wynik zwrócony jest w postaci listy. Jako przykład niech posłuży rozwiązanie równania kwadratowego o postaci $x^2-5x+6=0$

```
(%i19)  solve(x^2-5*x+6=0,x);
(%o19)  [x=3,x=2]
```

Gdy chcemy znaleźć rozwiązanie układu równań:

$$\begin{cases} xy-x=0 \\ x^2+2xy-y=2 \end{cases}$$

```
(%i20)  r1:x*y-x=0$
        r2:x^2+2*x*y-y-2=0$
        solve([r1,r2],[x,y]);
(%o20)  [[x=0,y=-2],[x=1,y=1],[x=-3,y=1]]
```

Podstawowe polecenia programu do rozwiązywania równań:

| | |
|---|---|
| solve(eqn, x) | rozwiązuje równanie eqn względem zmiennej x |
| solve([eqn1, eqn2],[x,y]) | rozwiązuje układ równań eqn1 i eqn2 względem zmiennych x i y |
| lhs(eqn) | odczytuje lewą stronę równania eqn |
| rhs(eqn) | odczytuje prawą stronę równania eqn |
| realroots(poly) | znajduje pierwiastki rzeczywiste wielomianu poly |
| allroots(poly) | znajduje numeryczne przybliżenie pierwiastków wielomianu poly |
| find root(eqn, x, x0, x1) | znajduje przybliżone rozwiązanie równania eqn na przedziale $[x_0,x_1]$ |

## Granice, pochodne i całki

Polecenie limit(expr, x, val, dir) znajduje granicę $\lim_{x\to val}$ expr(x). Opcjonalny argument dir określa czy szukamy granicy z prawej (plus) czy z lewej (minus) strony:

```
(%i21)  limit(sin(x)/x,x,0);
(%o21)  1
```

Polecenia diff() używamy do znajdowania pochodnych. Występuję ono w kilku wariantach:

- diff(expr) -zwraca różniczkę zupełną, czyli sumę po pochodnych po wszystkich zmiennych

```
(%i22)  diff(f(x)*f(y));
(%o22)  f(x)(d/dy f(y))del(y)+(d/dx f(x))f(y)del(x)
```

gdzie del(x) oznacza $\frac{\partial}{\partial x}$

- diff(expr, x, n) -wywołanie diff(f(x), x, n) odpowiada: $\frac{\partial^n}{\partial x^n} f(x)$
- diff(expr, x1, n1,....,xn, nm) -odpowiada zapisowi: diff(...(diff(expr,xn,nm)...),x1,n1) przy czyn expr musi jawnie zależeć od wszystkich zmiennych x1,x2,....xn:

```
(%i23)  diff(expr(x,y,z),x,1,y,2,z,3);
(%o23)  d^6/(dx dy^2 dz^3) expr(x,y,z)
```

Do obliczania zarówno całek oznaczonych jak i nieoznaczonych używamy polecenia integrate():

- integrate(expr,x) -oblicza całkę z nieoznaczoną expr względem zmiennej x = R f(x)dx

```
(%i24)  integrate(f(x),x);
(%o24)  ∫ f(x) dx
```

- integrate(expr, x, a, b) -oblicza całkę oznaczoną:

```
(%i25)  integrate(f(x),x,a,b);
                 b
(%o25)           ⌠
                 ⌡ f(x) dx
                 a
```

Istnieje wiele funkcji umożliwiających całkowanie numeryczne. Przykładem może być funkcja romberg() z pakietu o tej samej nazwie - bardzo prosty i wydajny algorytm (metoda Romberga). Wymaga aby całka była właściwa. Umożliwia numeryczne liczenie całek wielokrotnych:

```
(%i26)  load(romberg)$
        f(x):=exp(-2*x^2)$
        romberg(f(x),x,0,%pi);
(%o26)  0.6266570689954041
```

## Równania różniczkowe

W zakresie równań różniczkowych Maxima potrafi między innymi:

- rozwiązywać równania różniczkowe zwyczajne rzędu pierwszego:
  F(x, y(x), y'(x))=0
- rozwiązywać równania różniczkowe zwyczajne rzędu drugiego:
  F(x, y(x), y'(x), y''(x)) = 0
- rozwiązywać układy dwóch lub więcej równań różniczkowych rzędu pierwszego:
  $\frac{dy}{dx} = f(x,y,z), \quad \frac{dz}{dx} = g(x,y,z)$

Oto lista przydatnych funkcji:

| | |
|---|---|
| ode2(równanie, y, x) | podaje rozwiązanie ogólne równania różniczkowego rzędu pierwszego lub drugiego |
| ic1(rozwiązanie ogólne, x=$x_0$, y=$y_0$) | podaje rozwiązanie szczególne równania różniczkowego rzędu pierwszego spełniające warunek y($x_0$) = $y_0$ |
| ic2(rozwiązanie ogólne, x=$x_0$, y=$y_0$, diff(y,x)=$y_1$) | podaje rozwiązanie szczególne równania różniczkowego rzędu drugiego spełniające warunki: y($x_0$) = $y_0$, y'($x_0$) = $y_1$ |
| bc2(rozwiązanie ogólne, x=$x_1$, y=$y_1$, x=$x_2$, y=$y_2$) | podaje rozwiązanie szczególne równania różniczkowego rzędu drugiego spełniające warunki: y($x_1$) = $y_1$, y($x_2$) = $y_2$, |
| desolve([równanie1, równanie2], [y(x), z(x)]) | podaje rozwiązanie ogólne układu dwóch równań różniczkowych |

W celu uzyskania rozwiązania szczególnego, spełniającego warunki y(x0) = y0, z(x0) = z0, należy najpierw wpisać:

**atvalue(y(x),x=$x_0$,$y_0$)**

oraz atvalue(z(x),x=x0,z0), a dopiero później skorzystać z funkcji desolve. Jako przykład równanie różniczkowe:

$$\frac{dy}{dx} - y = 2xe^x$$

```
(%i27)  r:'diff(y,x)-y=2*x*exp(x);
(r)     d/dx y-y=2 x %e^x
(%i28)  ode2(r, y, x);
(%o28)  y=(x^2+%c)%e^x
```

Jeżeli Maxima potrafi rozwiązać równanie różniczkowe funkcja method podaje nam metodę, którą program wykorzystał do rozwiązania tego równania:

```
(%i29)  method;
(%o29)  linear
```

## Rozwijanie, rozkładanie oraz upraszczanie wyrażeń

Mimo istnienia pewnych algorytmów dla standardowych procedur, upraszczanie wyrażeń (algebraicznych, trygonometrycznych, logarytmicznych i in.) jest nadal jednymi z trudniejszych zadań dla systemów algebry komputerowej. Trudność leży w wyborze, kiedy i którą procedurę wybrać. Przykładowo, niektóre matematyczne sytuacje wymagają rozkładu na czynniki, inne natomiast wymagają, byśmy najpierw rozwinęli dane wyrażenie, a następnie upraszczali. Systemy algebry komputerowej, do których zalicza się Maxima, nie mają możliwości określenia, którą z dróg wybrać najpierw. Dlatego małą ilość uproszczeń da się wykonać automatycznie – do wielu wyrażeń można zastosować więcej niż jedno polecenie upraszczające.

| | |
|---|---|
| ratsimp(expr) | upraszcza wyrażenie |
| fullratsimp(expr) | wielokrotnie upraszcza wyrażenie |
| rootscontract(expr) | zamienia iloczyn pierwiastków na pierwiastek iloczynu |
| radcan(expr) | upraszcza wyrażenia logarytmiczne, wykładnicze i pierwiastkowe |
| logcontract(expr) | składa wiele funkcji logarytmicznych w pojedyncze logarytmy |
| expand(expr) | rozwija wyrażenie |
| ratexpand(expr) | rozwija wyrażenie (polecenie na ogół stosowane do wielomianów) |
| factor(expr) | rozkłada wyrażenie na czynniki |
| trigsimp(expr) | upraszcza wyrażenia trygonometryczne (wykorzystuje związek $\sin^2 x + \cos^2 x = 1$) |
| trigexpand(expr) | rozkłada funkcje trygonometryczne sumy oraz iloczynu kątów (może wymagać wielokrotnego użycia tego polecenia) |
| trigreduce(expr) | składa sumy oraz potęgi sinusów i cosinusów w sinusy i cosinusy wielokrotności ich argumentów |
| trigrat(expr) | upraszcza (do postaci kanonicznej) wyrażenie trygonometryczne |
| facsum(expr,$c_0$,...,$c_n$); | rozwija wyrażenie do postaci, w której czynniki $c_0$ do $c_n$ są niezależnie od siebie; uprzednio należy zmienić wartość: facsum_combine na false |

Przykłady:

```
(%i30)  (x^2-1)/(x-1);
(%o30)  (x^2-1)/(x-1)
(%i31)  ratsimp(x^2-1)/(x-1);
(%o31)  x+1
```

```
(%i32)  '((x-1)/(sqrt(x)-1)) = radcan((x-1)/(sqrt(x)-1));
(%o32)  (x-1)/(sqrt(x)-1) = sqrt(x)+1
(%i33)  expand((a+1)^2);
(%o33)  a^2 + 2a + 1

(%i34)  factor(a^2-5*a+6);
(%o34)  (a - 3) (a - 2)

(%i35)  '((cos(x)+sin(x))^2) = trigsimp((cos(x)+sin(x))^2);
(%o35)  (sin x + cos x)^2 = 2 cos x sin x + 1

(%i36)  trigreduce(sin(x)^3*cos(x));
(%o36)  (2 sin(2 x) - sin(4 x))/8
```

## Pakiety

Tak jak w wielu innych środowiskach obliczeniowych, Maxima dostarcza zbioru podstawowych poleceń oraz duży wybór pakietów dodatkowych. Polecenie:

**load(pakiet);**

ładuje zawartość pliku pakiet. Na przykład pakiet solve rat ineq wczytujemy za pomocą polecenia:

```
(%i39)  load(solve_rat_ineq)$
```

## Wykresy

Rysowanie wykresów funkcji jednej bądź dwu zmiennych jest ważnym zadaniem do oceniania zachowania się funkcji. Maxima używa zewnętrznych pakietów do rysowania wykresów. Istnieje możliwość wyboru między dwoma takimi pakietami: gnuplot (domyślny) oraz xmaxima (dawniej znana jako openmath).

Standardowe polecenia do rysowania wykresów w Maximie to plot2d, plot3d. Do wyświetlania obrazów w głównym oknie (inline) stosuje się funkcję wxplot2d oraz wxplot3d.

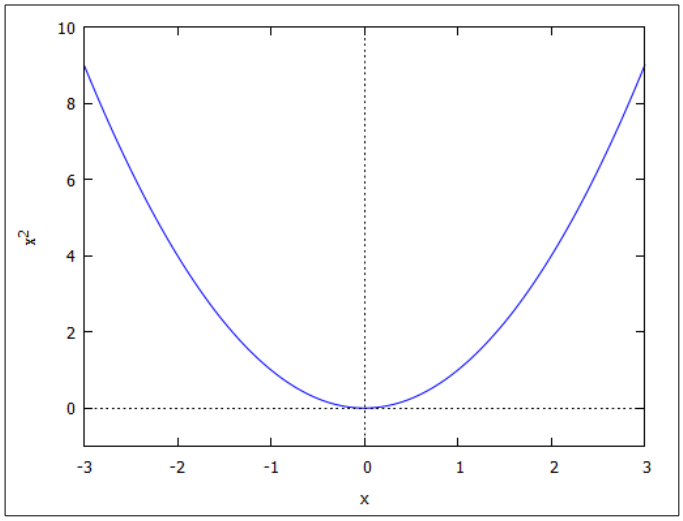
Polecenie :

**plot2d(f(x), [x,x min,x max], opt1, opt2, ...)**

jest jednym z najczęściej stosowanych poleceń do wykreślania funkcji jednej zmiennej postaci y = f(x). Za jego pomocą rysujemy wykres funkcji f w przedziale [x min,x max] przy dodatkowych opcjach opt1, opt2, ... Dodatkowe opcje są podawane w formie listy, tzn. w nawiasach kwadratowych.

Przykładowo, jeśli chcemy ograniczyć zasięg wyświetlanych wartości funkcji f, to używamy opcji postaci [y,y min,y max]. Przy wykorzystaniu tej opcji wykres zawsze będzie się znajdował wewnątrz podanego zasięgu, niezależnie od wartości osiąganych w wykresie. Jeżeli poziomy zasięg nie jest sprecyzowany, będzie on ustawiony jako minimum i maksimum drugiej współrzędnej punktów wykresu. Przykładowo:

```
(%i40)  plot2d(x^2,[x,-3,3],[y,-1,10]);
```



Funkcje służące do rysowania wykresów w Maximie:

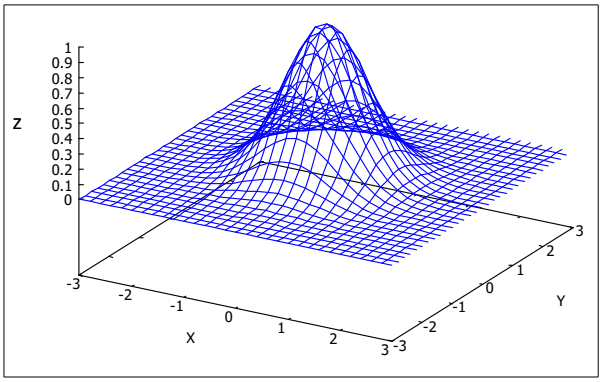
| | |
|---|---|
| plot2d([f(x)], [x,a,b]) | rysuje wykres funkcji f w przedziale [a, b] |
| plot2d([f1(x), . . . , fn(x)], [x,a,b]) | rysuje wykresy funkcji f1, . . . , fn w przedziale [a, b] na jednym rysunku |
| plot3d([f(x,y)], [x,a,b], [y,c,d]) | rysuje wykres funkcji f dla (x, y) ∈ [a, b] × [c, d] |
| plot3d([x(t,s), y(t,s), z(t,s)], [t,a,b], [s,c,d]) | rysuje wykres powierzchni zadanej parametrycznie: (x(t, s), y(t, s), z(t, s)), t ∈ [a, b], s ∈ [c, d] |
| wxplot2d() oraz wxplot3d() | wyświetlanie obrazów w głównym oknie (inline) |

W celu sformatowania wykresu na własny użytek należy do powyższych funkcji wprowadzić dodatkowe polecenia. Oto niektóre z nich:

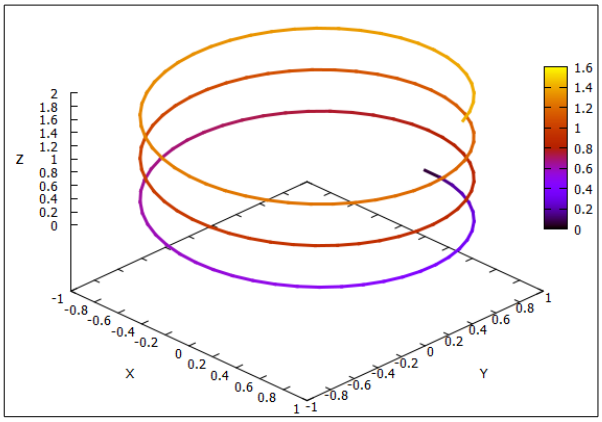
- **[gnuplot preamble, "set zeroaxis"]** – dodaje osie układu współrzędnych,
- **[gnuplot preamble, "set nokey"]** – usuwa legendę,
- **[gnuplot curve titles, ["title 'tytuł'"]]** – zmienia legendę,
- **[gnuplot curve styles, ["with lines n"]]** – zmienia styl linii, n ∈ {1, . . . , 15},
- **[gnuplot term, ps], [gnuplot out file, "plot.eps"]** – zapisuje wykres do pliku graficznego z rozszerzeniem .eps8 pod nazwą plot.eps,
- **[gnuplot term, png], [gnuplot out file, "plot.png"]** – zapisuje wykres do pliku graficznego z rozszerzeniem .png pod nazwą plot.png.

Przykłady:

```
(%i41)  draw3d(explicit(exp(-x^2-y^2),x,-3,3,y,-3,3))$
```




```
(%i42)  draw3d(enhanced3d=sqrt(t),line_width=3,nticks=150,
        parametric(sin(10*t),cos(10*t),t,t,0,2),terminal=wxt)$
```




## Pozostałe

Po więcej informacji na temat interfejsu (API) Maximy oraz przykładów jej obsługi zapraszam na stronę:

maxima.sourceforge.net

# Zadania do wykonania

Skrypty zadań proszę zapisywać w osobnych komórkach w ramach jednego pliku. Każdą komórkę proszę opatrzyć komentarzem z numerem zadania.

1. Sprowadzić do najprostszej postaci wyrażenie:

$$(a^{-1}-b^{-1})\frac{\sqrt{a}+\sqrt{b}}{\sqrt{b}-\sqrt{a}}-\frac{2}{\sqrt{ab}}$$

2. Rozwiązać w zbiorze liczb rzeczywistych równanie:

$$x^5-2x^4-4x^3+4x^2-5x+6=0$$

3. Znaleźć wszystkie pierwiastki wielomianu:

$$W(x)=x^5+x^2+1$$

4. Sprowadzić do postaci iloczynowej wielomian:

$$W(x)=2x^5+3x^4-x^3+x^2-3x-2$$

5. Rozwiązać układ równań:

$$\begin{cases} 2x^2+xy-4=0 \\ 2y^2+xy-10=0 \end{cases}$$

6. Wyznaczyć granicę:

$$\lim_{x\to 2}\frac{1}{x-2}$$

7. Obliczyć całkę oznaczoną:

$$\int_0^4 \sin(x)^2\,dx$$

8. Narysować wykres krzywej:

$$f(t)=\left(\cos(3t)\cos(\frac{t}{2}),\ \cos(5t)\sin(\frac{t}{2})\right)\quad \text{dla}\quad t\in[0,\ 2\pi]$$

9. Narysować elipsoidę daną równaniami:

$$x(t)=\cos a\cos b,\ y(t)=\sin a\cos b,\ z(t)=\sin b,\ a\in[0,2\pi],\ b\in\left[\frac{-\pi}{2},\frac{\pi}{2}\right]$$

10. Rozwiązać równanie różniczkowe:

$$y''-2y'+2y=e^{2x}+x$$