

Polecenia programu **wxMaxima** potrzebne do zajęć *Chemia kwantowa A* - *laboratorium komputerowe*

• Całkowanie

`integrate(wyrazenie, zmienna)`

Obliczanie całki nieoznaczonej po zmiennej x z funkcji postaci `wyrazenie`

`integrate(wyrazenie, zmienna, liczba1, liczba2)`

Obliczanie całki oznaczonej po zmiennej x z funkcji postaci `wyrazenie`

Przykłady:

`integrate(sin(x), x);` - oblicz $\int \sin x dx$

`integrate(x**2 + 5*x, x, 1, 2)` - oblicz $\int_1^2 (x^2 + 5x) dx$

`f(x) := x^2*exp(-x^2);` - zdefiniowanie funkcji $f(x)$.

`integrate(f(x), x, 0, inf)` - oblicz $\int_0^\infty f(x) dx = \int_0^\infty x^2 e^{-x^2} dx$

• Definicja funkcji

W celu zdefiniowania funkcji należy użyć znaku `:=`

Znak `*` oznacza mnożenie, znak `**` i znak `^` oznaczają podnoszenie do potęgi

Program wxMaxima rozpoznaje m.in. takie nazwy funkcji elementarnych, jak: `sin`, `cos`, `exp(a)` (inaczej `%e^a`), `acos` (arccos) `atan` (arctg) itp. oraz wiele funkcji specjalnych (więcej informacji można znaleźć w zakładce **Pomoc(Help)** programu.

Przykłady:

`f(x) := x^2*sin(x);` $f(x) = x^2 \sin(x)$

`chi(u,t) := t/u;` $\chi(t, u) = \frac{t}{u}$

`log10(x) := log(x)/log(10)` wprowadzenie funkcji logarytm dziesiętny z x , oznaczanej `log10(x)`. UWAGA! `log(x)` oznacza w wxMaxima $\ln x = \log_e(x)$, czyli logarytm naturalny z x . Jeśli potrzebna jest funkcja oznaczająca logarytm dziesiętny z x , to użytkownik musi ją zdefiniować.

Ponizej definicja funkcji: $\Psi(n, x) := \begin{cases} 0 & \text{dla } x < 0 \\ (\frac{1}{\sqrt{2}}) \cdot \sin(\frac{n\pi x}{4}) & \text{dla } 0 \leq x \leq 4 \\ 0 & \text{dla } x > 4 \end{cases}$

`f(n,x) := 1/sqrt(2)*sin(n*pi*x/4);`

`Psi(n,x) := if x<0 or x>4 then 0 else f(n,x);`

- **Metoda najmniejszych kwadratów**

`load(lsqares)`

Załadowanie pakietu pozwalającego na stosowanie metody najmniejszych kwadratów.

`M:matrix([0.00143,-4.51], [0.00132,-2.25], [0.00123,-0.24], [0.0011,3.0]);`

Wprowadzenie danych (np. pomiarów eksperymentalnych) w postaci listy par wartości (x_i, y_i) , dla $i=1, \dots, N$ (tu $N=4$), gdzie x_i -wartość zmiennej niezależnej, y_i -wartość zmiennej zależnej.

`lsquares_estimates(M, [x,y], y=A*x+B, [A,B]);`

Znalezienie takich wartości parametrów A i B , dla których $\sum_{i=1}^N (y_i - (Ax_i + B))^2$ ma najmniejszą wartość.

- **Nadawanie wartości**

W celu nadania wartości zmiennej, stałej lub zmiennym(stałym) z listy należy używać znaku :

Przykłady:

`a:5.25` nadaje a wartość 5.25

`u:t` nadaje u wartość t

`[b,c,d]:[2,5,8]` nadaje wartości: $b=2, c=5, d=8$

UWAGA!

Po wykonaniu polecenia:

`g:w*m` nadanie g wartości $w \cdot m$

a następnie nadaniu w i m konkretnych wartości liczbowych, na przykład:

`w:8`

`m:7`

program obliczy wartość g posługując się wartościami $w=8$ i $m=7$ dopiero po wydaniu polecenia:

`''g`

- **Obliczanie wartości wyrażenia dla danej wartości zmiennej**

`at(wyrażenie,zmienna=liczba)`

oblicz wartość wyrażenia `wyrażenie` dla wartości `liczba` zmiennej `zmienna`

Przykład:

`wyr: diff(x**5 + 5*x*exp(-x),x,2)` przyporządkuj etykietę `wyr` drugą pochodną funkcji $x^5 + 5xe^{-x}$

`at(wyr,x=0)` oblicz wartość `wyr` dla $x=0$ (czyli drugiej pochodnej funkcji $x^5 + 5xe^{-x}$ dla $x=0$)

• Podstawianie

`subst(wyrazenie1,wyrazenie2,wyrazenie3)`

Podstaw wyrażenie `wyrazenie1` zamiast wyrażenia `wyrazenie2` w wyrażeniu `wyrazenie3`. (UWAGA! `wyrazenie2` musi być pełnym "podwyrażeniem" wyrażenia `wyrazenie3`)

Przykłady:

`subst (a,x+y,(x+y)/(x-y))`

Podstaw a zamiast $x + y$ w wyrażeniu $\frac{x+y}{x-y}$ - efekt wyrażenie $\frac{a}{x-y}$

Program nie wykona polecenia: `subst (a,x+y,(x+y+1)/(x-y))`, ponieważ $x + y$ nie jest pełnym podwyrażeniem $\frac{x+y+1}{x-y}$ (pełnym podwyrażeniem jest $x + y + 1$)

• Postać dziesiętna

Zdarza się, że Maxima wyświetla wynik obliczeń w postaci ułamka zwykłego, a potrzebna jest postać dziesiętna tego wyniku. Aby uzyskać postać dziesiętną liczby przypisanej etykietce `wynik` należy użyć polecenia:

`float(wynik)`

Przykłady:

`wyn1:197/8` nadaj etykietce `wyn1` wartości $\frac{197}{8}$

`float(wyn1)` wyświetl `wyn1` w postaci dziesiętnej

`float(%pi/3)` wyświetl przybliżoną wartość $\frac{\pi}{3}$ w postaci dziesiętnej

• Przyjmowanie założeń

`assume(wyrazenie)`

Przyjmij, że spełnione jest `wyrazenie`, przy czym: `wyrazenie` może zawierać wyłącznie następujące operatory relacji: `<`, `<=`, `equal`, `notequal`, `>=`, `>`.

Przykład

`assume(a>0);` Zakładamy, że wartości a są większe od zera.

• Przypisywanie matematycznych właściwości zmiennym i funkcjom

Zmiennym i funkcjom można przypisać następujące właściwości matematyczne:

`integer`, `noninteger`, `even`, `odd`, `rational`, `irrational`, `real`,
`imaginary`, `complex`, `analytic`, `increasing`, `decreasing`, `oddfun`,
`evenfun`, `posfun`, `commutative`, `lassociative`, `rassociative`,
`symmetric`, `antisymmetric`

Przykłady:

`declare(m,integer)` m jest całkowite

`declare([k,l],even)` k i l są parzyste

`notequal(t,u)` $t \neq u$

• Rozwiązywanie równań

`eq1: x^2*y = 2*a/u` postać równania (1)

`solve(eq1,u)` - wyznacz u z równania (1)

• Rozwiązywanie równań różniczkowych

`ode2(wyrazenie,y,x);`

rozwiąż równanie różniczkowe zwyczajne I lub II rzędu o postaci `wyrazenie`, w którym y oznacza zmienną zależną, a x zmienną niezależną

Przykłady:

Znak `'` przed `diff` zapobiega wykonaniu operacji różniczkowania.

(%i1) `'diff(v,t)=a` Podajemy postać równania I rzędu

(%o1) $\frac{d}{dt}v = a$

(%i2) `ode2(%,v,t);`

(%o2) $v = a * t + \%c$

(%i3) `'diff(s,u,2)=3;` Podajemy postać równania II rzędu

(%o3) $\frac{d^2}{du^2}s = 3$

(%i4) `ode2(%,s,u);`

(%o4) $s = \frac{3u^2}{2} + \%k2u + \%k1$

$\%c, \%k1$ i $\%k2$ to stałe różniczkowania

• Różniczkowanie

`diff(sin(x),x)` - oblicz pochodną $\sin(x)$ po x

`f(x,u):= x*u^2 + exp(a*u)` definicja funkcji zmiennych x i u

`diff(f(x,u),u,2)` - oblicz drugą pochodną funkcji $f(x,u)$ po u

• Specjalne symbole

`%pi` oznacza wartość liczby π

`%e` oznacza wartość liczby e (podstawy logarytmów naturalnych)

`%i` oznacza liczbę $i = \sqrt{-1}$

• Upraszczenie wyrażeń

Następujące polecenia mogą być pomocne przy uzyskaniu prostszej formy skomplikowanych wyrażeń:

`expand(wyrazenie);`

powoduje rozwinięcie (wymnożenie) iloczynów sum, rozdzielenie liczników ułamów, które są sumami, na odpowiednie składniki itp. w wyrażeniu `wyrazenie`.

`ratsimp(wyrazenie);`

upraszcza `wyrazenie` i wszystkie występujące w nim podwyrażenia włącznie z argumentami funkcji.

Przykłady:

`expand((x-1)^5);` da w wyniku: $x^5 - 5x^4 + 10x^3 - 10x^2 + 5x - 1$

`expand((x-1)^2-x^2-1)/(x-1);` da w wyniku: $\frac{2x^2}{x-1} - \frac{2x}{x-1}$

`ratsimp(%);`

spowoduje uproszczenie poprzedniego wyniku do $2x$

`sin(x/(x^2+x)) = exp((log+1)^2 - log(x)^2);`

wprowadzenie wyrażenia: $\sin\left(\frac{x}{x^2+x}\right) = e^{(\log(x)+1)^2 - \log^2(x)}$

`ratsimp(%);`

upraszcza poprzednie wyrażenie do $\sin \frac{1}{x+1} = ex^2$

• Usunięcie wcześniejszego przyporządkowania

`kill(etykieta)`

powoduje usunięcie wcześniejszego przyporządkowania dla `etykieta`.

Jest to przydatne, jeśli, na przykład, nadaliśmy poprzednio zmiennej `x` jakąś wartość, a teraz chcemy znowu używać `x` jako zmiennej.

`kill(x)` usunięcie wartości przyporządkowanej zmiennej `x`

`kill(all)` usunięcie wszystkich wcześniejszych przyporządkowań

• Wartości i wektory własne macierzy

Polecenie `eigenvalues(M)` (synonim `eivals(M)`) służy do znajdowania wartości własnych, a `eigenvectors(M)` (synonim: `eivects(M)`) do znajdowania zarówno wartości własnych jak i wektorów własnych uprzednio zdefiniowanej macierzy M , przy czym elementy macierzy mogą mieć zarówno wartości liczbowe jak i symboliczne.

Polecenia:

```
load(linearalgebra);
```

```
eigens_by_jacobi(M);
```

służą do znajdowania wartości własnych i wektorów własnych uprzednio zdefiniowanej SYMETRYCZNEJ macierzy M , której elementami MUSZĄ być LICZBY. Pierwsze z poleceń powoduje załadowanie odpowiedniego pakietu.

Przykłady

1. Tylko wartości własne:

```
A:matrix([aa,bb],[bb,aa])
```

 zdefiniowanie macierzy $A = \begin{vmatrix} aa & bb \\ bb & aa \end{vmatrix}$

Polecenie:

```
eigenvalues(A)
```

spowoduje wyświetlenie linii:

```
[[aa-bb,bb+aa],[1,1]]
```

zawierającej [[wartości własne A],[krotności wartości własnych A]]

2. Wartości i wektory własne

```
B:matrix([4,1,0,0,0,1],[1,4,1,0,0,0],[0,1,4,1,0,0],[0,0,1,4,1,0],[0,0,0,1,4,1],[1,0,0,0,1,4]);
```

Zdefiniowano macierz: $B = \begin{vmatrix} 4 & 1 & 0 & 0 & 0 & 1 \\ 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 0 & 1 & 4 \end{vmatrix}$

Polecenie:

```
eigenvectors(B);
```

spowoduje wyświetlenie linii:

```
[[[6,2,3,5],[1,1,2,2]],[1,1,1,1,1,1],[1,-1,1,-1,1,-1],[1,0,-1,1,0,-1],[0,1,-1,0,1,-1],[1,0,-1,-1,0,1],[0,1,1,0,-1,-1]]
```

zawierającej [[[wartości własne B],[krotności wartości własnych B]], 6 wektorów własnych B]

• Wykresy

```
plot2d(wyrazenie, [x,liczba1,liczba2], [y,liczba3,liczba4])
```

Narysuj wykres funkcji zmiennej x postaci *wyrazenie*, dla wartości x zawartych między *liczba1* a *liczba2*, przy czym (*opcjonalne*) wartości funkcji y zawarte są między *liczba3* a *liczba4*. Na wykresie mogą nie być widoczne osie $x=0$ i $y=0$.

```
set_plot_option([gnuplot_preamble," set zeroaxis"]);
```

Po wykonaniu powyższej instrukcji na następnych wykresach będą widoczne osie $x=0$ i $y=0$.

```
contour_plot(wyrazenie, [x,liczba1,liczba2], [y,liczba3,liczba4], opcje)
```

Narysuj kontury funkcji *wyrazenie*, to znaczy zbiory punktów o współrzędnych (x,y) , dla których *wyrazenie* (będące funkcją x i y) ma stałą wartość, dla zakresu wartości x od *liczba1* do *liczba2* i zakresu wartości y od *liczba3* do *liczba4*.

Przykłady:

1. Proste wykresy

```
plot2d(sin(x), [x,-%pi,%pi], [y,-2,2])
```

Narysuj wykres $\sin(x)$ dla $-\pi < x < \pi$, przy czym na osi y ma być skala od -2 do 2.

```
plot2d(sin(x), [x,-2*%pi,2*%pi])
```

Narysuj wykres $\sin(x)$ dla $-2\pi < x < 2\pi$.

```
f(x) := sin(x)-cos(x)
```

```
plot2d(f(x), [x,-2*%pi,2*%pi])
```

Rysowanie wcześniej zdefiniowanej funkcji $f(x) = \sin(x) - \cos(x)$

2. Wykres kilku funkcji z legendą i opisem osi

```
plot2d([sin(t),cos(t),f(t)], [t,-2*%pi,2*%pi], [y,-3,3],
```

```
[legend," sin(t)"," cos(t)"," sin(t)-cos(t)],
```

```
[xlabel,"t"], [ylabel,"Wartosci funkcji"])
```

Rysowanie kilku funkcji ($\sin(t)$, $\cos(t)$ oraz funkcji zdefiniowanej w przykładzie 1 jako $f(x)$) dla $-2\pi < t < 2\pi$, przy czym na osi wartości funkcji (y) ma być skala od -3 do 3, oś argumentów ma być oznaczona t , a oś rzędnych *Wartosci funkcji*.

3. Wartości dyskretne i funkcja na wykresie

```
xy: [[26.0,953.1],[26.5,935.1],[27,917.8],[28.0,885.0]]
plot2d([[discrete,xy], 10*8.314*298/V], [V,25.0,29.0],
[style, [points,5,2,6], [lines,1,1]],
[legend," experiment","theory"],
[xlabel,"volume [l]"], [ylabel,"pressure[hPa]"]])
```

Naniesienie na wykresie danych eksperymentalnych (podanych wcześniej jako pary xy) i zależności teoretycznej. Dane eksperymentalne (`[discrete,xy]`) jako punkty, a zależność teoretyczna jako linia (odpowiednio (`points` i (`lines` w liście `[style]`). Cyfry określają wielkość, kształt i kolor punktów oraz grubość i kolor linii. Oś argumentów oznaczona `volume [l]`, oś rzędnych oznaczona `pressure[hPa]`

4. Rysowanie linii odpowiadających stałym wartościom

Podane poniżej polecenia powodują narysowanie linii poziomych (tu: dla wartości 1, 2, 3 i 4 na osi rzędnych) (przydatne np. do rysowania poziomów energetycznych)

```
g(n,t):=n
```

Definicja funkcji zależnej pozornie od pomocniczej "fałszywej" zmiennej t

```
my_preamble:" set xzeroaxis;set xtics(''0, ''2)";
```

Likwidacja oznaczeń osi odpowiadającej pomocniczej zmiennej.

```
plot2d([g(1,t),g(2,t),g(3,t),g(4,t)], [t,0,2], [xlabel," "],
[ylabel," n"], [legend," n1", " n2"," n3"," n4"],
[ggnplot_preamble, my_preamble]);
```

Narysowanie poziomych linii (odpowiadająca fałszywej zmiennej oś odciętych - bez oznaczenia).

5. Rysowanie konturów

```
contour_plot(x^2+y^2, [x,-2,2], [y,-1,1])
```

Wykreśl zbiory punktów na płaszczyźnie, dla których $x^2 + y^2$ ma stałą wartość, czyli okręgi o początku w punkcie $(0,0)$. Na wykresach stosunek skali na osi poziomej do skali na osi pionowej wynosi zwykle 2:1, żeby zatem uniknąć deformacji (zobaczyć okrąg, a nie elipsę) należy przyjąć takie zakresy zmiennych x i y .

W niektórych wersjach programu stosunek skali na osi poziomej do skali na osi pionowej wykresu wynosi 4/3. Należy wówczas odpowiednio dostosować zakresy zmiennych x i y .

```
contour_plot(x^2+y^2, [x,-4/3,4/3], [y,-1,1])
```

• Zależność funkcji od zmiennych

```
depends(funkcja, zmienna)
```

```
depends(funkcja,[lista zmiennych])
```


`depends([lista funkcji],[lista zmiennych])`

Polecenie `depends` pozwala na zadeklarowanie zależności funkcji, której postaci nie podajemy, od zmiennej lub zmiennych. Tak zadeklarowana zależność jest rozpoznawana wyłącznie przez polecenie `diff` (różniczkowanie). Przydaje się to np. do otrzymywania wzorów na pochodne funkcji złożonych.

Przykład

```
%i1 depends(f,[r,theta,phi])
```

Funkcja f zależy od zmiennych r , θ (czyt. teta) i φ (czyt. fi)

```
%i2 depends([r,theta,phi],[x,y,z])
```

Każda ze zmiennych r , θ i φ zależy od zmiennych x , y , z .

Polecenie :

```
%i3 diff(f,x);
```

spowoduje wyświetlenie ogólnego wzoru na $\frac{df}{dx}$:

$$\frac{df}{d\theta} \frac{d\theta}{dx} + \frac{df}{dr} \frac{dr}{dx} + \frac{df}{d\varphi} \frac{d\varphi}{dx} \quad (1)$$

- **Znajdowanie miejsca zerowego funkcji lub wyrażenia**

```
find_root(f,a,b)
```

znajduje miejsce zerowe funkcji f w przedziale $\langle a, b \rangle$.

Wartości funkcji f dla a i b , czyli $f(a)$ i $f(b)$ MUSZĄ mieć RÓŻNE ZNAKI, inaczej zostanie wyświetlony komunikat o popełnieniu błędu.

```
find_root(expr,x,a,b)
```

znajduje wartości x , zawarte w przedziale $\langle a, b \rangle$, dla których wyrażenie `expr` jest równe zero. Wyrażenie `expr` może być także równaniem. Poszukiwane są wówczas wartości x , dla których $\text{LewaStrona}(\text{expr}) - \text{PrawaStrona}(\text{expr}) = 0$.