

Quick Start

Using **wxMaxima**

in *Introduction to Quantum Chemistry* – computer laboratory

- **Assigning values**

Sign : is used to assign a value to a variable or a constant (also to a list of variables or constants). **Sign ;** ends a command line (not obligatory).

Examples:

a:5.25; assigns 5.25 to *a*

u:t; assigns *t* to *u*

[b,c,d]:[2,5,8]; assigns values: $b=2,c=5,d=8$

ATTENTION! When a command assigning result of an arithmetic operation to a variable is performed:

g:w*m; assigns $w \cdot m$ to *g*

and in subsequent steps numerical values are assigned to operands, e.g.

w:8; m:7;

the numerical result of the operation will be calculated (here as $8 \cdot 7$), only after command: `''g;` is used.

- **Assumption**

`assume(expression)`

Assume that `expression` in parenthesis is true. The only relation operators allowed to appear in `expression` are: `<`, `<=`, `equal`, `notequal`, `>=`, `>`.

Example

`assume(a>0);` The values of *a* are greater than zero.

- **Declaration of mathematical properties**

Maxima recognizes certain mathematical properties of functions and variables. These so-called features are:

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

Examples:

`declare(m,integer)` *m* is integer

`declare([k,l],even)` *k i l* are even

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

- **Dependence on variables**

`depends(fuction, variable)`

`depends(fuction, [variable list])`

`depends([function list], [variable list])`

Dependence of an unknown function on some variables is declared. The dependence is recognized only by `diff` command.

Example

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

Function f depends on r , θ and φ

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

Each of the variables r , θ and φ depends on x , y , z .

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

returns a general formula for $\frac{df}{dx}$:

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

- **Differentiation**

`diff(sin(x), x)` - returns the derivative of $\sin(x)$ with respect to x

`f(x,u) := x*u^2 + exp(a*u)` function $f(x, u)$ is defined

`diff(f(x,u), u, 2)` returns the second derivative of $f(x, u)$ with respect to u

- **Eigenvalues and eigenvectors of a matrix**

`eigenvalues(M)` (also shortened to `eivals(M)`) returns the eigenvalues of a previously defined matrix M

`eigenvectors(M)` (also shortened to `eivects(M)`) returns both eigenvalues and eigenvectors of a previously defined matrix M (elements of M can be both numbers and symbols)

Commands:

```
load(linearalgebra); (loads the appropriate package)
```

```
eigens_by_jacobi(M);
```

return eigenvalues and eigenvectors of a previously defined symmetric matrix M . The elements of M must be numbers.

Examples

1. Calculate eigenvalues of a matrix A

$$A = \begin{bmatrix} aa & bb \\ bb & aa \end{bmatrix}$$

Definition of A :

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

Calculation of eigenvalues of A : `eigenvalues(A)`

returns the line

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

containing *[[list of eigenvalues of A],[list of multiplicities of the eigenvalues of A]]*

2. Calculate both eigenvalues and eigenvectors of a matrix B

$$B = \begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 1 \\ 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 1 \\ 1 & 0 & 0 & 0 & 1 & 4 \end{bmatrix}$$

Definition of B :

```
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]);
```

Eigenvectors and eigenvalues of B :

```
eigenvectors(B);
```

returns the line:

```
[[[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]]
```

containing *[[[eigenvalues of B],[list of multiplicities of the eigenvalues of B]], 6 eigenvectors of B]*

• Find roots

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

finds a root of the function f over a closed interval $[a, b]$.

Values of the function in points a and b , i.e. $f(a)$ and $f(b)$, should have opposite signs.

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

finds a root of the expression `expr` over the closed interval $[a, b]$ (values of x , for which `expr=0`). The expression `expr` can be an equation, in which case `find_root` seeks for a root of `lhs(expr)-rhs(expr)=0`.

• Function and arithmetic operators

The symbols $+$ $*$ $/$ and $^$ represent addition, multiplication, division, and exponentiation, respectively

Sign := is used to define a function.

wxMaxima recognizes (among other) such elementary function names as: **sin(a)**, **cos(a)**, **exp(a)** (same as **%e^a**), **acos** (arccos) **atan** (arctg) as well as many special functions (for details see wxMaxima manual - choose **Help**)

Examples:

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

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

$$\text{log10}(x) := \log(x)/\log(10) \quad \log_{10}(x) = \ln(x)/\ln(10)$$

ATTENTION! in wxMaxima **log(x)** means $\ln x = \log_e(x)$. If $\log_{10}(x)$ is needed, it has to be defined (here will be denoted as **log10(x)**).

$$\text{The function: } \Psi(n, x) := \begin{cases} 0 & \text{dla } x < 0 \\ (\frac{1}{\sqrt{2}}) \cdot \sin(\frac{n\pi x}{4}) & \text{dla } 0 \leq x \leq 4 \\ 0 & \text{dla } x > 4 \end{cases} \quad \text{is defined below:}$$

$$f(n, x) := 1/\text{sqrt}(2) * \sin(n * \%pi * x / 4);$$

$$\text{Psi}(n, x) := \text{if } x < 0 \text{ or } x > 4 \text{ then } 0 \text{ else } f(n, x);$$

• Integration

integrate(expression, variable)

Computes (symbolically) an indefinite integral of **expression** with respect to x

integrate(expression, variable, a, b)

Computes (symbolically) a definite integral of **expression** with respect to x , with limits of integration a and b .

Examples:

$$\text{integrate}(\sin(x), x); \quad \text{- computes } \int \sin x dx$$

$$\text{integrate}(x**2+5*x, x, 1, 2) \quad \text{- computes } \int_1^2 (x^2 + 5x) dx$$

$$f(x) := x^2 * \exp(-x)^2; \quad \text{- function } f(x) \text{ is defined.}$$

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

- **Least squares method**

```
load(lsquares)
```

loads a package containing necessary commands.

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

Data (e.g. experimental results) are given as a pair list: (x_i, y_i) for $i=1, \dots, N$ (here $N=4$), where x_i is a value of the independent variable and y_i a value of the dependent variable

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

Finds the values of parameters A i B , for which $\sum_{i=1}^N (y_i - (Ax_i + B))^2$ has the least value.

- **Ordinary differential equations**

```
ode2(expression,y,x);
```

`ode2` solves an ordinary differential equation (ODE) of first or second order. An ODE is given by `expression`, the second argument is the dependent variable `y`, and the third argument is the independent variable `x`. When successful, it returns either an explicit or implicit solution for the dependent variable. `%c` is used to represent the integration constant in the case of first-order equations, and `%k1` and `%k2` the constants for second-order equations.

Examples:

`%in` denotes the n -th input line, `%on` denotes the n -th output line.

Sign `'` before `diff` prevents performing of differentiation.

(%i1) `'diff(v,t)=a` the form of ODE of first order is given

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

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

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

(%i3) `'diff(s,u,2)=3;` the form of ODE of second order is given

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

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

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

`%c, %k1` i `%k2` are the integration constants

• Plotting

```
plot2d(expression, [x,number1,number2], [y,number3,number4])
```

returns a plot of a function `expression` of the independent variable x , for x values between `number1` and `number2`, (*optional*) function values y displayed are between `number3` and `number4`.

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

The axes $x=0$ and $y=0$ will appear on plots displayed after this command has been executed (if the x and y axes were previously not visible on plots)

```
contour_plot(expression, [x,number1,number2], [y,number3,number4], options)
```

Plots contours of a function `expression`, i.e. sets of points (x,y) , for which `expression` (a function of x and y) has a constant value, for x values ranging from `number1` to `number2` and y values ranging from `number3` to `number4`.

Examples:

1. Simple plots

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

Plots $\sin(x)$ for $-\pi < x < \pi$, with y scale from -2 to 2.

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

Plots $\sin(x)$ for $-2\pi < x < 2\pi$.

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

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

Plots previously defined function $f(x) = \sin(x) - \cos(x)$

2. Plots of several functions with a legend and axis labels

```
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, "Function values"])
```

Plots three functions $\sin(t)$, $\cos(t)$ and $f(x)$ from Example 1) for $-2\pi < t < 2\pi$, with y -axis scale from -3 to 3, abscissa (x -axis) labeled `t`, and ordinate (y -axis) labeled `Function values`.

3. Discrete values and a function on a plot

```
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]''])
```

Experimental data (defined earlier as pairs `xy`) and theoretical dependence are displayed on a plot. Experimental data `[discrete,xy]` are shown as points and theoretical dependence as a line (`points` and `lines` in a list `[style]`, respectively). The numbers denote specific size, shape and colour of points and lines. Abscissa labeled `volume[l]`, ordinate labeled `pressure[hPa]`

4. Plotting of lines corresponding to constant values

The following commands return a plot of horizontal lines for values 1, 2, 3 and 4.

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

function is defined as dependent on "false" auxiliary variable `t`

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

Tics are removed from a horizontal axis corresponding to "false" auxiliary variable.

```
plot2d([g(1,t),g(2,t),g(3,t),g(4,t)], [t,0,2], [xlabel, '''],
[ylabel, ''n''], [legend, ''n1'', ''n2'', ''n3'', ''n4''],
[gnuplot_preamble, my_preamble]);
```

5. Contour plots

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

Plot sets of points, for which $x^2 + y^2$ is constant i.e. circles centered in a (0,0) point. Often (y -axis scale):(x -axis scale) is equal to 2:1, so in order to avoid a deformation of the plot the appropriate variable ranges should be used.

If y -axis and x -axis scales are related in a different way (4:3 is a common case) then variable ranges should be adjusted.

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

• Simplification

The following commands can be used to simplify the form of complicated expressions:

`expand(expression);`

expands products of sums in expression `expression`.

`ratsimp(expression);`

simplifies `expression` and all subexpressions including function arguments

Examples:

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

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

`ratsimp(%);`

returns the previous result simplified to $2x$

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

the expression: $\sin\left(\frac{x}{x^2+x}\right) = e^{(\log(x)+1)^2 - \log^2(x)}$ is defined

`ratsimp(%);`

simplifies previous expression to $\sin\frac{1}{x+1} = ex^2$

• Solving equations

`eq1: x^2*y = 2*a/u` the form of equation `eq1` is given

`solve(eq1,u)` - solve (symbolically) `eq1` to find u

• Special symbols

`%pi` denotes the value of π

`%e` denotes the value of the number e

`%i` denotes $i = \sqrt{-1}$

• Substitution

`subst(expression1,expression2,expression3)`

Substitutes the expression `expression1` instead of `expression2` in `expression3`.

The expression `expression2` has to be a complete "subexpression" of `expression3`.

Examples:

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

a is substituted instead of $x + y$ in $\frac{x+y}{x-y}$ and the result is expression $\frac{a}{x-y}$

However, the command: `subst (a,x+y,(x+y+1)/(x-y))`, will not be executed, because $x + y$ is not a complete subexpression of $\frac{x+y+1}{x-y}$ ($x + y + 1$ is such a complete subexpression)