wxMaxima

Contents

Il manuale utente di wxMaxima **2**

1 Introduzione a wxMaxima **4**
1.1 *Maxima* e wxMaxima 4
1.1.1 *Maxima* 4
1.1.2 wxMaxima 4
1.2 Workbook basics 5
1.2.1 The workbook approach 5
1.2.2 Celle 7
1.2.3 Horizontal and vertical cursors 7
1.2.4 Sending cells to Maxima 9
1.2.5 Command autocompletion 9
1.2.6 Side Panes 12
1.2.7 MathML output 12
1.2.8 Markdown support 12
1.2.9 Hotkeys 14
1.2.10 Raw TeX in the TeX export 14
1.3 File Formats 14
1.3.1 .mac 14
1.3.2 .wxm 16
1.3.3 .wxmx 16
1.4 Configuration options 16
1.4.1 Default animation framerate 16
1.4.2 Default plot size for new *maxima* sessions 16
1.4.3 Match parenthesis in text controls 16
1.4.4 Don't save the worksheet automatically 18
1.4.5 Where is the configuration saved? 18

2 Estensioni a *Maxima* **18**
2.1 Subscripted variables 18
2.2 User feedback in the status bar 19
2.3 Plotting 19
2.3.1 Embedding a plot into the work sheet 19
2.3.2 Making embedded plots bigger or smaller 19

2.3.3 Better quality plots 20
2.3.4 Opening embedded plots in interactive *gnuplot* windows . 20
2.3.5 Opening gnuplot's command console in `plot` windows . . 20
2.3.6 Embedding animations into the spreadsheet 20
2.3.7 Opening multiple plots in contemporaneous windows . . . 22
2.3.8 The "Plot using draw" side pane 23
2.4 Embedding graphics 25
2.5 Startup files 25
2.6 Special variables wx... 25
2.7 Pretty-printing 2D output 26
2.8 Bug reporting 27
2.9 Marking output being drawn in red 27

3 Risoluzione dei problemi 27
3.1 Cannot connect to *Maxima* 27
3.2 How to save data from a broken .wxmx file 28
3.3 I want some debug info to be displayed on the screen before my command has finished 28
3.4 Plotting only shows a closed empty envelope with an error message 29
3.5 Plotting an animation results in "error: undefined variable" . . . 29
3.6 I lost a cell contents and undo doesn't remember 29
3.7 *wxMaxima* starts up with the message "Maxima process terminated." 30
3.8 Maxima is forever calculating and not responding to input 30
3.9 My SBCL-based *Maxima* runs out of memory 30
3.10 Input sometimes is sluggish/ignoring keys on Ubuntu 30
3.11 *wxMaxima* halts when *Maxima* processes Greek characters or Umlauts 30
3.12 Plotting 32
3.12.1 Can I make *wxMaxima* output both image files and embedded plots at once? 32
3.12.2 Can I set the aspect ratio of a plot? 32

4 FAQ 33
4.1 Is there a way to make more text fit on a LaTeX page? 33
4.2 Is there a dark mode? 33
4.3 *wxMaxima* sometimes hangs for a several seconds once in the first minute 33

5 Argomenti della riga di comando 33

Il manuale utente di wxMaxima

wxMaxima è un'interfaccia utente grafica (GUI) per il sistema di algebra computazionale (CAS) *Maxima*. wxMaxima consente di utilizzare tutte le funzioni di *Maxima*, inoltre, fornisce comode procedure guidate per accedere alle funzionalità più comunemente utilizzate. Questo manuale descrive alcune delle funzionalità che rendono wxMaxima una delle GUI più popolari per *Maxima*.

Figure 1: logo wxMaxima

Prima di esaminare i contenuti della guida, è meglio evidenziare alcuni punti relativi alla sua consultazione. Immediatamente di seguito c'è un breve sommario. Facendo clic su uno degli elementi elencati si sposta il cursore nella parte superiore della sezione indicata del manuale. Facendo clic su Sommario si sposta il cursore su un sommario esteso. Questa tabella estesa può essere usata per arrivare direttamente a parti specifiche del manuale. Il collegamento Contenuti viene visualizzato lungo il testo a scopo di orientamento nella navigazione.

- Introduzione:

Basi di wxMaxima

- Estensioni:

I comandi che wxMaxima aggiunge a *Maxima*

- Risoluzione dei problemi:

Cosa fare se wxMaxima non si comporta come ci si aspetta

- FAQ:

Domande ricorrenti

- RigaDiComando:

Gli argomenti a riga di comando che wxMaxima supporta

1 Introduzione a wxMaxima

1.1 *Maxima* e wxMaxima

Nel dominio open source, i grandi sistemi sono normalmente suddivisi in progetti più piccoli che sono più facili da gestire per piccoli gruppi di sviluppatori. Ad esempio, un programma di masterizzazione di CD sarà costituito da uno strumento a riga di comando che effettivamente masterizza il CD e da un'interfaccia utente grafica che consente agli utenti di utilizzarlo senza dover conoscere tutte le opzioni della riga di comando, e di fatto senza utilizzare affatto la riga di comando. Un vantaggio di questo approccio è che il lavoro di sviluppo che è stato investito nel programma a riga di comando può essere condiviso da molti programmi: lo stesso programma a riga di comando del masterizzatore CD può essere utilizzato come plug-in "invia-al-CD" per un'applicazione di gestione file, per la funzione "masterizza su CD" di un lettore musicale e come masterizzatore di CD per uno strumento di backup su DVD. Un altro vantaggio è che suddividere una grande attività in parti più piccole consente agli sviluppatori di fornire diverse interfacce utente per lo stesso programma.

Un sistema di algebra computazionale (CAS) come *Maxima* si inserisce in questo quadro. Un CAS può fornire la logica dietro un'applicazione di calcolatrice di precisione arbitraria o può fare trasformazioni automatiche di formule sullo sfondo di un sistema più grande (ad esempio, [Sage] (https://www.sagemath.org/)). In alternativa, può essere utilizzato direttamente come sistema indipendente. *Maxima* è utilizzabile tramite riga di comando. Spesso, tuttavia, un'interfaccia come *wxMaxima* si rivela un modo più efficiente per accedere al software, in particolare per i nuovi utenti.

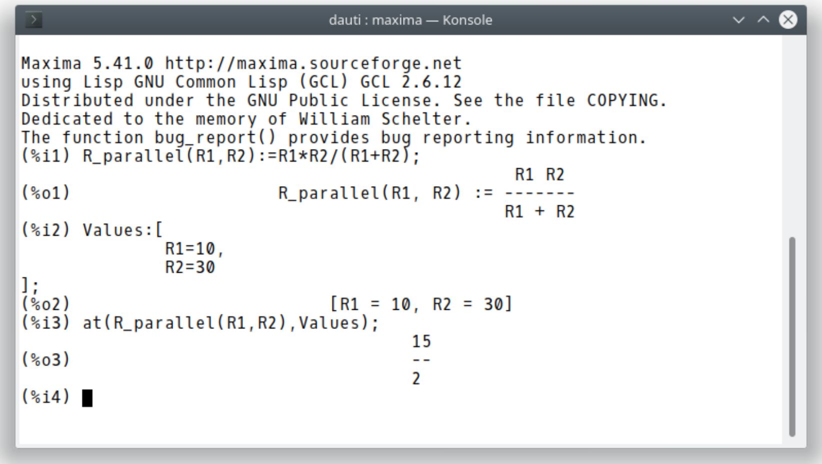
1.1.1 *Maxima*

Maxima è un sistema di algebra computazionale (CAS) completo. Un CAS è un programma in grado di risolvere problemi matematici riorganizzando le formule e trovando una formula che risolve il problema invece di emettere semplicemente il valore numerico del risultato. In altre parole, *Maxima* può fungere da calcolatrice che fornisce rappresentazioni numeriche di variabili e può anche fornire soluzioni analitiche. Inoltre, offre una gamma di metodi numerici di analisi per equazioni o sistemi di equazioni che non possono essere risolti analiticamente.

Extensive documentation for *Maxima* is available in the internet. Part of this documentation is also available in wxMaxima's help menu. Pressing the Help key (on most systems the F1 key) causes *wxMaxima*'s context-sensitive help feature to automatically jump to *Maxima*'s manual page for the command at the cursor.

1.1.2 wxMaxima

wxMaxima is a graphical user interface that provides the full functionality and flexibility of *Maxima*. wxMaxima offers users a graphical display and many



```
Maxima 5.41.0 http://maxima.sourceforge.net
using Lisp GNU Common Lisp (GCL) GCL 2.6.12
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
The function bug_report() provides bug reporting information.
(%i1) R_parallel(R1,R2):=R1*R2/(R1+R2);
                                                R1 R2
(%o1)                    R_parallel(R1, R2) := -------
                                               R1 + R2
(%i2) Values:[
              R1=10,
              R2=30
];
(%o2)                         [R1 = 10, R2 = 30]
(%i3) at(R_parallel(R1,R2),Values);
                                      15
(%o3)                                 --
                                      2
(%i4) 
```

Figure 2: schermata di Maxima, riga di comando

features that make working with *Maxima* easier. For example *wxMaxima* allows one to export any cell's contents (or, if that is needed, any part of a formula, as well) as text, as LaTeX or as MathML specification at a simple right-click. Indeed, an entire workbook can be exported, either as a HTML file or as a LaTeX file. Documentation for *wxMaxima*, including workbooks to illustrate aspects of its use, is online at the *wxMaxima* help site, as well as via the help menu.

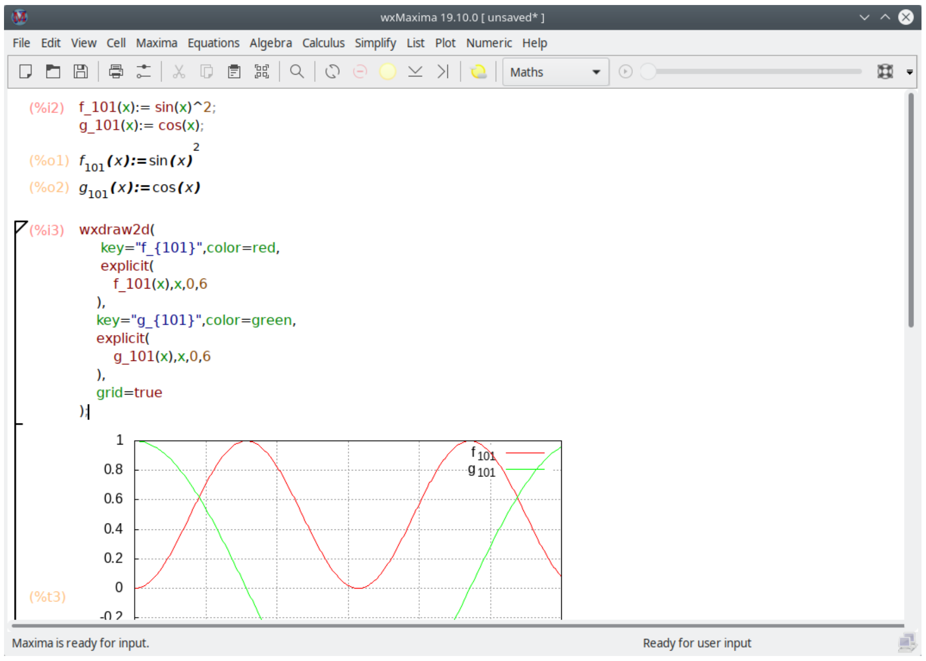
The calculations that are entered in *wxMaxima* are performed by the *Maxima* command-line tool in the background.

1.2 Workbook basics

Much of *wxMaxima* is self-explaining, but some details require attention. This site contains a number of workbooks that address various aspects of *wxMaxima*. Working through some of these (particularly the "10 minute *(wx)Maxima* tutorial") will increase one's familiarity with both the content of *Maxima* and the use of *wxMaxima* to interact with *Maxima*. This manual concentrates on describing aspects of *wxMaxima* that are not likely to be self-evident and that might not be covered in the online material.

1.2.1 The workbook approach

One of the very few things that are not standard in *wxMaxima* is that it organizes the data for *Maxima* into cells that are evaluated (which means: sent to *Maxima*) only when the user requests this. When a cell is evaluated, all commands in


wxMaxima 19.10.0 [unsaved*]
File Edit View Cell Maxima Equations Algebra Calculus Simplify List Plot Numeric Help
Maths
1
0.8
0.6
0.4
0.2
0
-0.2
f_{101}
g_{101}

```
(%i2) f_101(x):= sin(x)^2;
      g_101(x):= cos(x);
```

(%o1) $f_{101}(x) := \sin(x)^2$

(%o2) $g_{101}(x) := \cos(x)$

```
(%i3) wxdraw2d(
        key="f_{101}",color=red,
        explicit(
          f_101(x),x,0,6
        ),
        key="g_{101}",color=green,
        explicit(
          g_101(x),x,0,6
        ),
        grid=true
      );
(%t3)
```

```
Maxima is ready for input.                Ready for user input
```

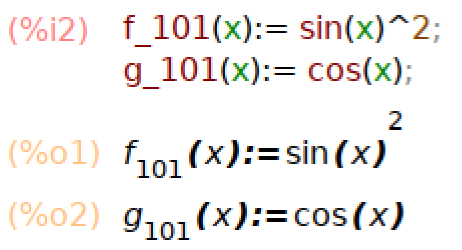
Figure 3: wxMaxima window

that cell, and only that cell, are evaluated as a batch. (The preceding statement is not quite accurate: One can select a set of adjacent cells and evaluate them together. Also, one can instruct *Maxima* to evaluate all cells in a workbook in one pass.) *wxMaxima*'s approach to submitting commands for execution might feel unfamiliar at the first sight. It does, however, drastically ease work with big documents (where the user does not want every change to automatically trigger a full re-evaluation of the whole document). Also, this approach is very handy for debugging.

If text is typed into *wxMaxima* it automatically creates a new worksheet cell. The type of this cell can be selected in the toolbar. If a code cell is created the cell can be sent to *Maxima*, which causes the result of the calculation to be displayed below the code. A pair of such commands is shown below.

On evaluation of an input cell's contents the input cell *Maxima* assigns a label to the input (by default shown in red and recognizable by the `%i`) by which it can be referenced later in the *wxMaxima* session. The output that *Maxima* generates also gets a label that begins with `%o` and by default is hidden, except if the user assigns the output a name. In this case by default the user-defined label is displayed. The `%o`-style label *Maxima* auto-generates will also be accessible, though.

Besides the input cells *wxMaxima* allows for text cells for documentation, image



```
(%i2) f_101(x):= sin(x)^2;
      g_101(x):= cos(x);
```

$$(\%o1)\quad f_{101}(x):=\sin(x)^2$$

$$(\%o2)\quad g_{101}(x):=\cos(x)$$

Figure 4: Input/output cell

cells, title cells, chapter cells and section cells. Every cell has its own undo buffer so debugging by changing the values of several cells and then gradually reverting the unneeded changes is rather easy. Furthermore the worksheet itself has a global undo buffer that can undo cell edits, adds and deletes.

The figure below shows different cell types (title cells, section cells, subsection cells, text cells, input/output cells and image cells).

1.2.2 Celle

The worksheet is organized in cells. Each cell can contain other cells or the following types of content:

- one or more lines of *Maxima* input
- una o più immagini
- output of, or a question from, *Maxima*
- a text block that can for example be used for documentation
- un titolo, sezione o subsezione.

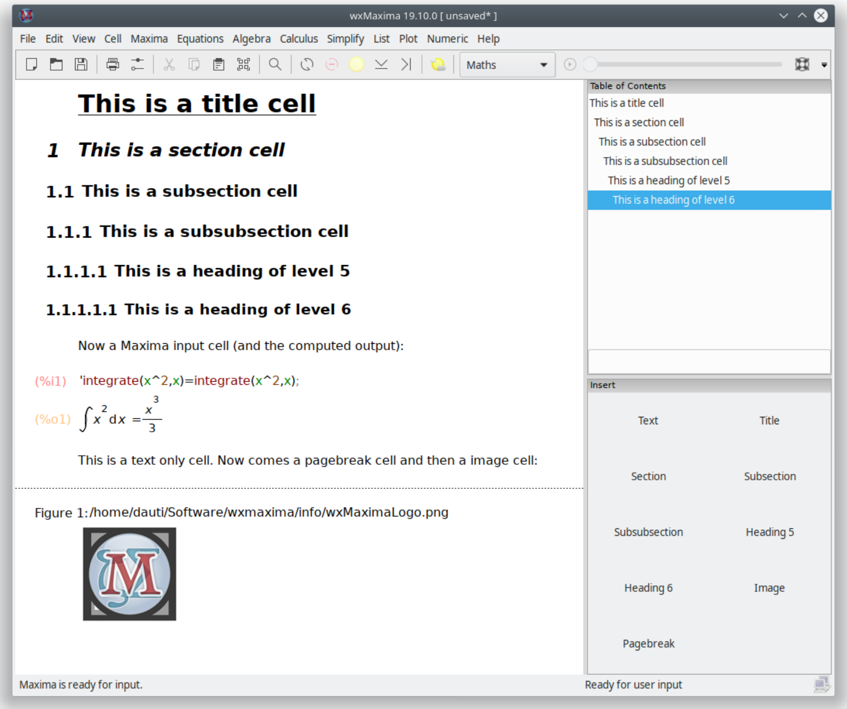
The default behavior of *wxMaxima* when text is entered is to automatically create a math cell. Cells of other types can be created using the Cell menu, using the hot keys shown in the menu or using the drop-down list in the toolbar. Once the non-math cell is created, whatever is typed into the file is interpreted as text.

Additional comment text can be entered into a math cell if bracketed as follows: `/*This comment will not be sent to Maxima for evaluation*/`.

1.2.3 Horizontal and vertical cursors

If the user tries to select a complete sentence a word processor will try to extend the selection to automatically begin and end with a word boundary. Likewise *wxMaxima* if more than one cell is selected will extend the selection to whole cells.

What isn't standard is that *wxMaxima* provides drag-and-drop flexibility by


wxMaxima 19.10.0 [unsaved*]
File Edit View Cell Maxima Equations Algebra Calculus Simplify List Plot Numeric Help
Maths
Table of Contents
This is a title cell
This is a section cell
This is a subsection cell
This is a subsubsection cell
This is a heading of level 5
This is a heading of level 6
Insert
Text
Title
Section
Subsection
Subsubsection
Heading 5
Heading 6
Image
Pagebreak
Maxima is ready for input.
Ready for user input

This is a title cell

1 This is a section cell

1.1 This is a subsection cell

1.1.1 This is a subsubsection cell

1.1.1.1 This is a heading of level 5

1.1.1.1.1 This is a heading of level 6

Now a Maxima input cell (and the computed output):

```
(%i1) 'integrate(x^2,x)=integrate(x^2,x);
```

(%o1) $\int x^2 dx = \frac{x^3}{3}$

This is a text only cell. Now comes a pagebreak cell and then a image cell:

Figure 1:/home/dauti/Software/wxmaxima/info/wxMaximaLogo.png

Figure 5: Esempio di celle wxMaxima diverse

defining two types of cursors. *wxMaxima* will switch between them automatically when needed:

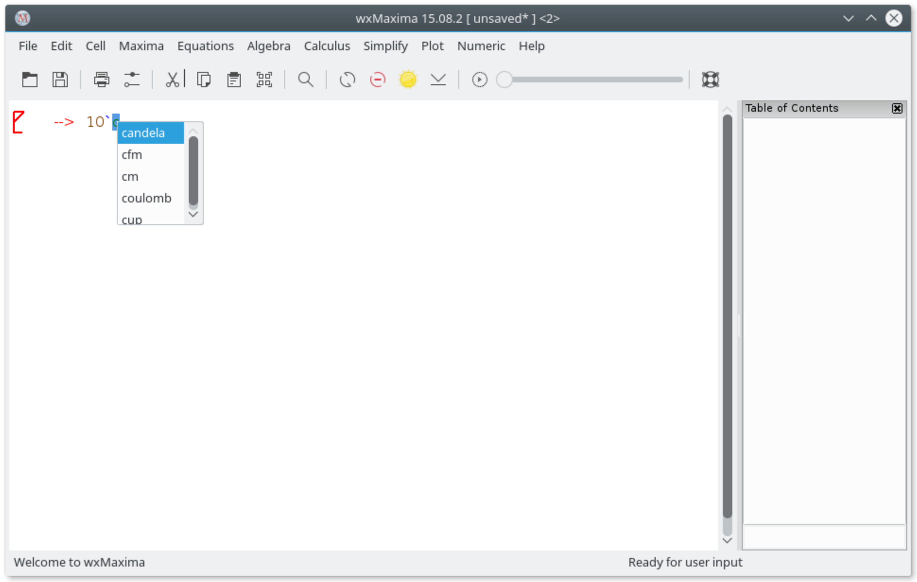
- The cursor is drawn horizontally if it is moved in the space between two cells or by clicking there.
- A vertical cursor that works inside a cell. This cursor is activated by moving the cursor inside a cell using the mouse pointer or the cursor keys and works much like the cursor in a text editor.

1.2.4 Sending cells to Maxima

The command in a code cell are executed once CTRL+ENTER, SHIFT+ENTER or the ENTER key on the keypad is pressed. The *wxMaxima* default is to enter commands when either CTRL+ENTER or SHIFT+ENTER is entered, but *wxMaxima* can be configured to execute commands in response to ENTER.

1.2.5 Command autocompletion

wxMaxima contains an autocompletion feature that is triggered via the menu (Cell/Complete Word) or alternatively by pressing the key combination CTRL+SPACE. The autocompletion is context-sensitive. For example if activated within an unit specification for ezUnits it will offer a list of applicable units.


wxMaxima 15.08.2 [unsaved*] <2>
File Edit Cell Maxima Equations Algebra Calculus Simplify Plot Numeric Help
Table of Contents
Welcome to wxMaxima
Ready for user input

```
--> 10`
candela
cfm
cm
coulomb
cup
```

Figure 6: ezUnits

Besides completing a file name, a unit name or the current command's or variable's name the autocompletion is able to show a template for most of the commands indicating the type (and meaning) of the parameters this program expects. To activate this feature press SHIFT+CTRL+SPACE or select the respective menu item (Cell/Show Template).

1.2.5.1 Greek characters

Computers traditionally stored characters in 8-bit values. This allows for a maximum of 256 different characters. All letters, numbers, and control symbols (end of transmission, end of string, lines and edges for drawing rectangles for menus *etc.*) of nearly any given language can fit within that limit.

For most countries the codepage of 256 characters that has been chosen does not include things like Greek letters, though, that are frequently used in mathematics. To overcome this type of limitation Unicode has been invented: An encoding that makes English text work like normal, but to use much more than 256 characters.

Maxima allows Unicode, if it was compiled using a Lisp compiler that either supports Unicode or that doesn't care about the font encoding. As at least one of this pair of conditions is likely to be true. *wxMaxima* provides a method of entering Greek characters using the keyboard:

- A Greek letter can be entered by pressing the ESC key and then starting to type the Greek character's name.
- Alternatively it can be entered by pressing ESC, one letter (or two for the Greek letter omicron) and ESC again. In this case the following letters are supported:

key	Greek letter	key	Greek letter	key	Greek letter
a	alpha	i	iota	r	rho
b	beta	k	kappa	s	sigma
g	gamma	l	lambda	t	tau
d	delta	m	mu	u	upsilon
e	epsilon	n	nu	f	phi
z	zeta	x	xi	c	chi
h	eta	om	omicron	y	psi
q	theta	p	pi	o	omega
A	Alpha	I	Iota	R	Rho
B	Beta	K	Kappa	S	Sigma
G	Gamma	L	Lambda	T	Tau
D	Delta	M	Mu	U	Upsilon
E	Epsilon	N	Nu	P	Phi
Z	Zeta	X	Xi	C	Chi
H	Eta	Om	Omicron	Y	Psi
T	Theta	P	Pi	O	Omega

The same mechanism also allows to enter some miscellaneous mathematical symbols:

keys to enter	mathematical symbol
hbar	Planck's constant: a h with a horizontal bar above it
Hbar	a H with a horizontal bar above it
2	squared
3	to the power of three
/2	1/2
partial	partial sign (the d of dx/dt)
integral	integral sign
sq	square root
ii	imaginary
ee	element
in	in
impl implies	implies
inf	infinity
empty	empty
TB	big triangle right
tb	small triangle right
and	and
or	or
xor	xor
nand	nand
nor	nor
equiv	equivalent to
not	not
union	union
inter	intersection
subseteq	subset or equal
subset	subset
notsubseteq	not subset or equal
notsubset	not subset
approx	approximately
propto	proportional to
neq != /= or #	not equal to
+/- or pm	a plus/minus sign
<= or leq	equal or less than
>= or geq	equal or greater than
<< or ll	much less than
>> or gg	much greater than
qed	end of proof
nabla	a nabla operator
sum	sum sign
prod	product sign

keys to enter	mathematical symbol
exists	there exists sign
nexists	there is no sign
parallel	a parallel sign
perp	a perpendicular sign
leadsto	a leads to sign
->	a right arrow
–>	a long right arrow

If a special symbol isn't in the list it is possible to input arbitrary Unicode characters by pressing ESC [number of the character (hexadecimal)] ESC.

ESC 61 ESC therefore results in an `a`.

Please note that most of these symbols (notable exceptions are the logic symbols) do not have a special meaning in *Maxima* and therefore will be interpreted as ordinary characters. If *Maxima* is compiled using a Lisp that doesn't support dealing with Unicode characters they might cause an error message instead.

1.2.6 Side Panes

Shortcuts to the most important *Maxima* commands or things like a table of contents, windows with debug messages or a history of the last issued commands can be accessed using the side panes. They can be enabled using the "View" menu. They all can be moved to other locations inside or outside the *wxMaxima* window. Other useful panes is the one that allows to input Greek letters using the mouse.

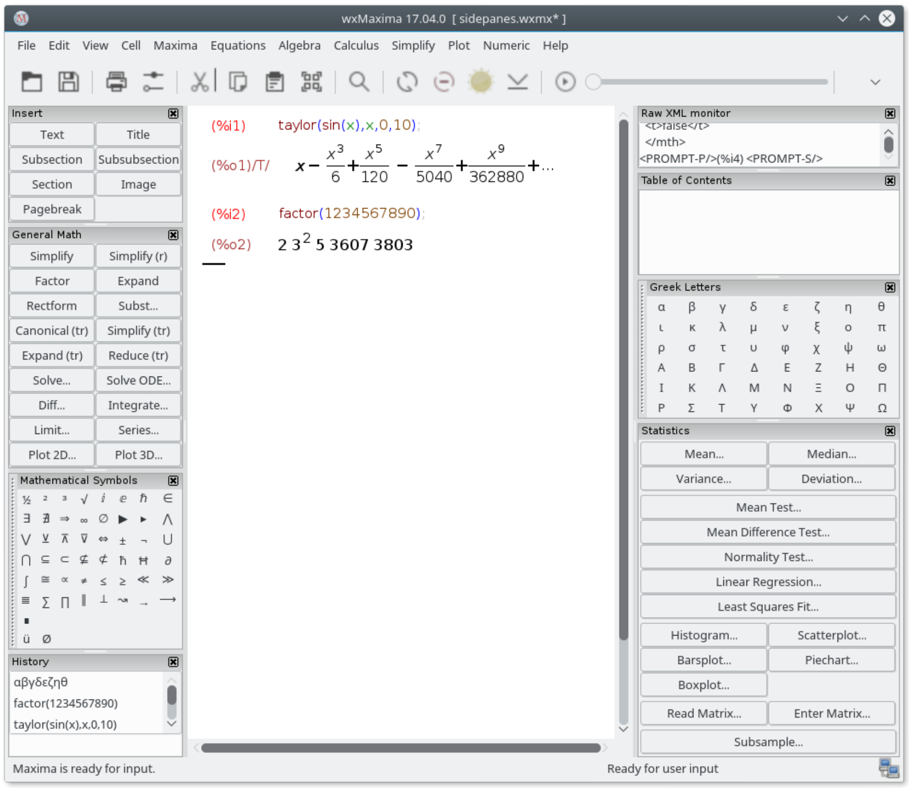
1.2.7 MathML output

Several word processors and similar programs either recognize MathML input and automatically insert it as an editable 2D equation - or (like LibreOffice 5.1) have an equation editor that offers an "import MathML from clipboard" feature. Others support RTF maths. *wxMaxima* therefore offers several entries in the right-click menu.

1.2.8 Markdown support

wxMaxima offers a set of standard markdown conventions that don't collide with mathematical notation. One of this elements is bullet lists.

```
Ordinary text
 * One item, indentation level 1
 * Another item at indentation level 1
   * An item at a second indentation level
   * A second item at the second indentation level
```


wxMaxima 17.04.0 [sidepanes.wxmx*]
File Edit View Cell Maxima Equations Algebra Calculus Simplify Plot Numeric Help
Insert
Text Title Subsection Subsubsection Section Image Pagebreak
General Math
Simplify Simplify (r) Factor Expand Rectform Subst... Canonical (tr) Simplify (tr) Expand (tr) Reduce (tr) Solve... Solve ODE... Diff... Integrate... Limit... Series... Plot 2D... Plot 3D...
Mathematical Symbols
History
αβγδεζηθ
factor(1234567890)
taylor(sin(x),x,0,10)
Raw XML monitor
Table of Contents
Greek Letters
α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω Α Β Γ Δ Ε Ζ Η Θ Ι Κ Λ Μ Ν Ξ Ο Π Ρ Σ Τ Υ Φ Χ Ψ Ω
Statistics
Mean... Median... Variance... Deviation... Mean Test... Mean Difference Test... Normality Test... Linear Regression... Least Squares Fit... Histogram... Scatterplot... Barsplot... Piechart... Boxplot... Read Matrix... Enter Matrix... Subsample...
Maxima is ready for input.
Ready for user input

```
(%i1) taylor(sin(x),x,0,10);
```

$$(\%o1)/T/ \quad x-\frac{x^3}{6}+\frac{x^5}{120}-\frac{x^7}{5040}+\frac{x^9}{362880}+\ldots$$

```
(%i2) factor(1234567890);
```

$$(\%o2) \quad 2\;3^2\;5\;3607\;3803$$

```
<t>false</t>
</mth>
<PROMPT-P/>(%i4) <PROMPT-S/>
```

Figure 7: Example of different side panes

```
 * A third item at the first indentation level
Ordinary text
```

wxMaxima will recognize text starting with `>` chars as block quotes:

```
Ordinary text
> quote quote quote quote
> quote quote quote quote
> quote quote quote quote
Ordinary text
```

wxMaxima's TeX and HTML output will also recognize `=>` and replace it by the corresponding Unicode sign:

```
cogito => sum.
```

Other symbols the HTML and TeX export will recognize are `<=` and `>=` for comparisons, a double-pointed double arrow (`<=>`), single- headed arrows (`<->`, `->` and `<-`) and `+/-` as the respective sign. For TeX output also `<<` and `>>` are recognized.

1.2.9 Hotkeys

Most hotkeys can be found in the text of the respective menus. Since they are actually taken from the menu text and thus can be customized by the translations of *wxMaxima* to match the needs of users of the local keyboard, we do not document them here. A few hotkeys or hotkey aliases, though, are not documented in the menus:

- CTRL+SHIFT+DELETE deletes a complete cell.
- CTRL+TAB or CTRL+SHIFT+TAB triggers the auto-completion mechanism.
- SHIFT+SPACE inserts a non-breaking space.

1.2.10 Raw TeX in the TeX export

If a text cell begins with `TeX:` the TeX export contains the literal text that follows the `TeX:` marker. Using this feature allows the entry of TeX markup within the *wxMaxima* workbook.

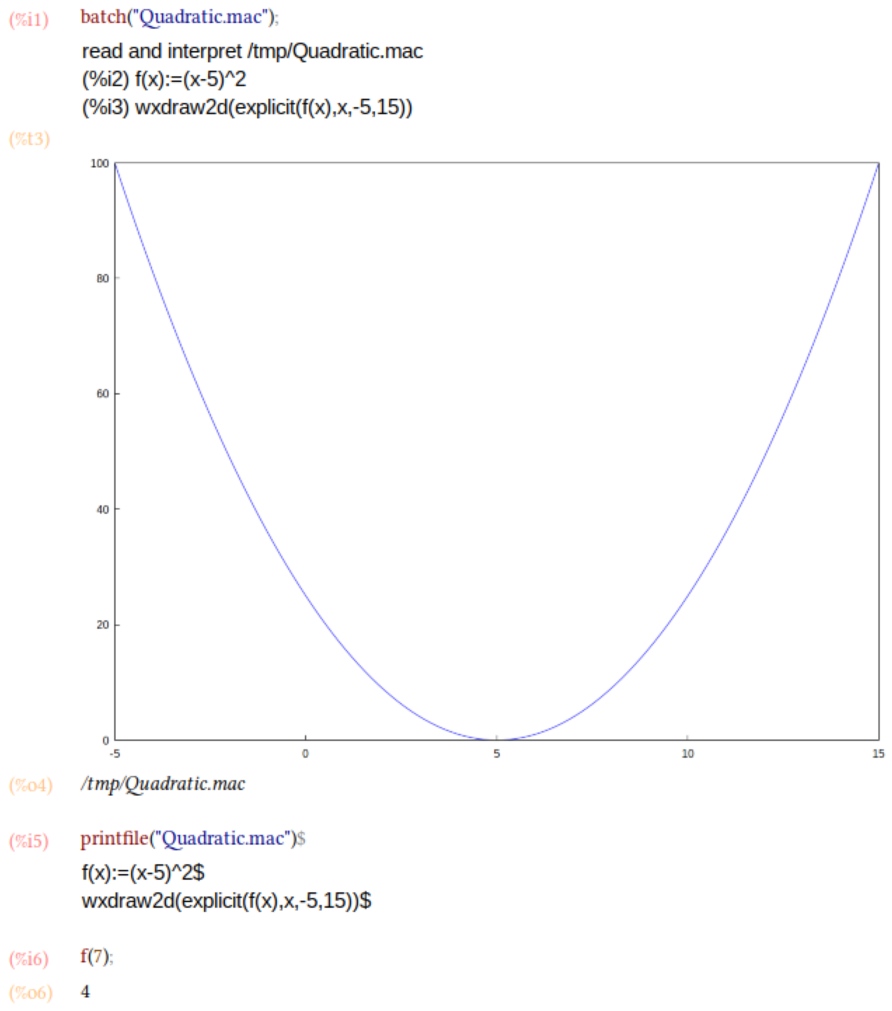
1.3 File Formats

The material that is developed in a *wxMaxima* session can be stored for later use in any of three ways:

1.3.1 .mac

`.mac` files are ordinary text files that contain *Maxima* commands. They can be read using *Maxima*'s `batch()` or `load()` command or *wxMaxima*'s File/Batch File menu entry.

One Example is shown below. `Quadratic.mac` defines a function and afterwards generates a plot with `wxdraw2d()`. Afterwards the contents of the file `Quadratic.mac` are printed and new defined function `f()` is evaluated.



```
(%i1)   batch("Quadratic.mac");
        read and interpret /tmp/Quadratic.mac
        (%i2) f(x):=(x-5)^2
        (%i3) wxdraw2d(explicit(f(x),x,-5,15))
(%t3)
(%o4)   /tmp/Quadratic.mac

(%i5)   printfile("Quadratic.mac")$
        f(x):=(x-5)^2$
        wxdraw2d(explicit(f(x),x,-5,15))$

(%i6)   f(7);
(%o6)   4
```

Figure 8: Loading a `.mac` file with `batch()`

Attention: Although the file `Quadratic.mac` has an usual *Maxima* extension (`.mac`), it can only be read by *wxMaxima*, since the command `wxdraw2d()` is a wxMaxima-extension to *Maxima*.

You can be use `.mac` files for writing your own library of macros. But since they don't contain enough structural information they cannot be read back as a *wxMaxima* session.

1.3.2 .wxm

.wxm files contain the worksheet except *Maxima*'s output. On Maxima versions >5.38 they can be read using *Maxima*'s `load()` function just as .mac files can be. With this plain-text format it sometimes is unavoidable that worksheets that use new features are not downwards-compatible with older versions of *wxMaxima*.

1.3.3 .wxmx

This XML-based file format saves the complete worksheet including things like the zoom factor and the watchlist. It is the preferred file format.

1.4 Configuration options

For some common configuration variables *wxMaxima* offers two ways of configuring:

- The configuration dialog box below lets you change their default values for the current and subsequent sessions.
- Also, the values for most configuration variables can be changed for the current session only by overwriting their values from the worksheet, as shown below.

1.4.1 Default animation framerate

The animation framerate that is used for new animations is kept in the variable `wxanimate_framerate`. The initial value this variable will contain in a new worksheet can be changed using the configuration dialogue.

1.4.2 Default plot size for new *maxima* sessions

After the next start plots embedded into the worksheet will be created with this size if the value of `wxplot_size` isn't changed by *maxima*.

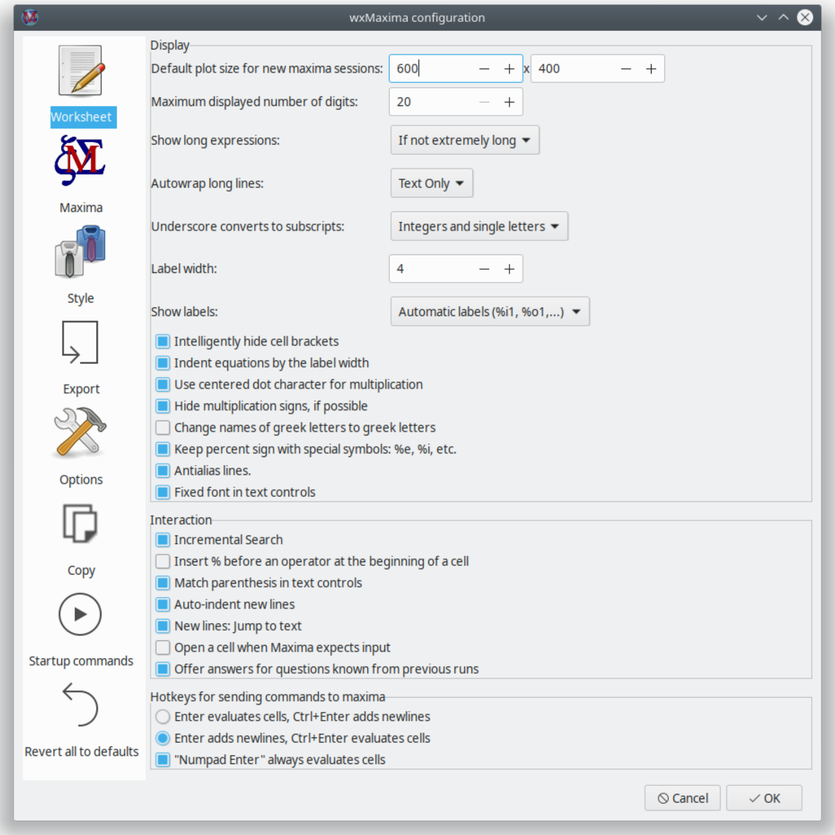
In order to set the plot size of a single graph only use the following notation can be used that sets a variable's value for one command only:

```
wxdraw2d(
   explicit(
       x^2,
       x,-5,5
   )
), wxplot_size=[480,480]$
```

1.4.3 Match parenthesis in text controls

This option enables two things:

- If an opening parenthesis, bracket or double quote is entered *wxMaxima* will insert a closing one after it.



Display

Default plot size for new maxima sessions: 600 x 400

Maximum displayed number of digits: 20

Show long expressions: If not extremely long

Autowrap long lines: Text Only

Underscore converts to subscripts: Integers and single letters

Label width: 4

Show labels: Automatic labels (%i1, %o1,...)

- [✓] Intelligently hide cell brackets
- [✓] Indent equations by the label width
- [✓] Use centered dot character for multiplication
- [✓] Hide multiplication signs, if possible
- [ ] Change names of greek letters to greek letters
- [✓] Keep percent sign with special symbols: %e, %i, etc.
- [✓] Antialias lines.
- [✓] Fixed font in text controls

Interaction

- [✓] Incremental Search
- [ ] Insert % before an operator at the beginning of a cell
- [✓] Match parenthesis in text controls
- [✓] Auto-indent new lines
- [✓] New lines: Jump to text
- [ ] Open a cell when Maxima expects input
- [✓] Offer answers for questions known from previous runs

Hotkeys for sending commands to maxima

- ○ Enter evaluates cells, Ctrl+Enter adds newlines
- ● Enter adds newlines, Ctrl+Enter evaluates cells
- [✓] "Numpad Enter" always evaluates cells

Figure 9: wxMaxima configuration 1

- If text is selected if any of these keys is pressed the selected text will be put between the matched signs.

1.4.4 Don't save the worksheet automatically

If this option is set the file the worksheet is in is overwritten only on request of the user. In case of a crash/power loss/… a recent backup copy is still made available in the temp directory, though.

If this option isn't set *wxMaxima* behaves more like a modern cellphone app:

- Files are saved automatically on exit
- And the file will automatically be saved every 3 minutes.

1.4.5 Where is the configuration saved?

If you are using Unix/Linux, the configuration information will be saved in a file `.wxMaxima` in your home directory (if you are using wxWidgets < 3.1.1), or `.config/wxMaxima.conf` ((XDG-Standard) if wxWidgets >= 3.1.1 is used). You can retrieve the wxWidgets version from the command `wxbuild_info();` or by using the menu option Help->About. wxWidgets is the cross-platform GUI library, which is the base for *wxMaxima* (therefore the `wx` in the name). (Since the filename starts with a dot, `.wxMaxima` or `.config` will be hidden).

If you are using Windows, the configuration will be stored in the registry. You will find the entries for *wxMaxima* at the following position in the registry: `HKEY_CURRENT_USER\Software\wxMaxima`

---

2 Estensioni a *Maxima*

wxMaxima is primarily a graphical user interface for *Maxima*. As such, its main purpose is to pass along commands to *Maxima* and to report the results of executing those commands. In some cases, however, *wxMaxima* adds functionality to *Maxima*. *wxMaxima*'s ability to generate reports by exporting a workbook's contents to HTML and LaTeX files has been mentioned. This section considers some ways that *wxMaxima* enhances the inclusion of graphics into a session.

2.1 Subscripted variables

if `wxsubscripts` is set to true variable names of the format `x_y` are displayed using a subscript if

- `y` is a single letter
- `y` is an integer

If the variable name doesn't match these requirements it can still be declared as "to be subscripted" using the command `wxdeclare_subscript(variable_name);`

or `wxdeclare_subscript([variable_name1,variable_name2,...]);` Declaring a variable as subscripted can be reverted using the following command: `wxdeclare_subscript(variable_name,false);`

2.2 User feedback in the status bar

Long-running commands can provide user-feedback in the status bar. This user feedback is replaced by any new feedback that is placed there (allowing to use it as a progress indicator) and is deleted as soon as the current command sent to *Maxima* is finished. It is safe to use `wxstatusbar()` even in libraries that might be used with plain *Maxima* (as opposed to *wxMaxima*): If *wxMaxima* isn't present the `wxstatusbar()` command will just be left unevaluated.

```
for i:1 thru 10 do (
    /* Tell the user how far we got */
    wxstatusbar(concat("Pass ",i)),
    /* (sleep n) is a Lisp function, which can be used */
    /* with the character "?" before. It delays the */
    /* program execution (here: for 3 seconds) */
    ?sleep(3)
)$
```

2.3 Plotting

Plotting (having fundamentally to do with graphics) is a place where a graphical user interface will have to provide some extensions to the original program.

2.3.1 Embedding a plot into the work sheet

Maxima normally instructs the external program *gnuplot* to open a separate window for every diagram it creates. Since many times it is convenient to embed graphs into the work sheet instead *wxMaxima* provides its own set of plot functions that don't differ from the corresponding *maxima* functions save in their name: They are all prefixed by a "wx". For example `wxplot2d` corresponds to `plot2d`, `wxplot3d` corresponds to `plot3d`, `wxdraw` corresponds to `draw` and `wxhistogram` corresponds to `histogram`.

2.3.2 Making embedded plots bigger or smaller

As noted above, the configure dialog provides a way to change the default size plots are created with which sets the starting value of `wxplot_size`. The plotting routines of *wxMaxima* respect this variable that specifies the size of a plot in pixels. It can always be queried or used to set the size of the following plots:

```
wxplot_size:[1200,800]$
wxdraw2d(
    explicit(
        sin(x),
        x,1,10
    )
)$
```

If the size of only one plot is to be changed *Maxima* provides a canonical way to change an attribute only for the current cell. In this usage the specification `wxplot_size = [value1, value2]` is appended to the `wxdraw2d( )` command, and is not part of the `wxdraw2d` command.

```
wxdraw2d(
    explicit(
        sin(x),
        x,1,10
    )
),wxplot_size=[1600,800]$
```

2.3.3 Better quality plots

Gnuplot doesn't seem to provide a portable way of determining whether it supports the high-quality bitmap output that the cairo library provides. On systems where *gnuplot* is compiled to use this library the pngcairo option from the configuration menu (that can be overridden by the variable `wxplot_pngcairo`) enables support for antialiasing and additional line styles. If `wxplot_pngcairo` is set without *gnuplot* supporting this the result will be error messages instead of graphics.

2.3.4 Opening embedded plots in interactive *gnuplot* windows

If a plot was generated using the `wxdraw`-type commands (`wxplot2d` and `wxplot3d` isn't supported by this feature) and the file size of the underlying *gnuplot* project isn't way too high *wxMaxima* offers a right-click menu that allows to open the plot in an interactive *gnuplot* window.

2.3.5 Opening gnuplot's command console in `plot` windows

On MS Windows, if in *Maxima*'s variable `gnuplot_command` "gnuplot" is replaced by "wgnuplot", *gnuplot* offers the possibility to open a console window, where *gnuplot* commands can be entered into. Unfortunately, enabling this feature causes *gnuplot* to "steal" the keyboard focus for a short time every time a plot is prepared.

2.3.6 Embedding animations into the spreadsheet

3D diagrams tend to make it hard to read quantitative data. A viable alternative might be to assign the 3rd parameter to the mouse wheel. The `with_slider_draw` command is a version of `wxdraw2d` that does prepare multiple plots and allows to switch between them by moving the slider on top of the screen. *wxMaxima* allows to export this animation as an animated gif.

The first two arguments for `with_slider_draw` are the name of the variable that is stepped between the plots and a list of the values of these variable. The arguments that follow are the ordinary arguments for `wxdraw2d`:

```
with_slider_draw(
    f,[1,2,3,4,5,6,7,10],
    title=concat("f=",f,"Hz"),
    explicit(
        sin(2*%pi*f*x),
        x,0,1
    ),grid=true
);
```

The same functionality for 3D plots is accessible as `with_slider_draw3d`, which allows for rotating 3d plots:

```
wxanimate_autoplay:true;
wxanimate_framerate:20;
with_slider_draw3d(
     ,makelist(i,i,1,360,3),
    title=sconcat(" =", ),
    surface_hide=true,
    contour=both,
    view=[60, ],
    explicit(
        sin(x)*sin(y),
        x,- , ,
        y,- ,
    )
)$
```

If the general shape of the plot is what matters it might suffice to move the plot just a little bit in order to make its 3D nature available to the intuition:

```
wxanimate_autoplay:true;
wxanimate_framerate:20;
with_slider_draw3d(
    t,makelist(i,i,0,2* ,.05* ),
    title=sconcat(" =", ),
    surface_hide=true,
    contour=both,
    view=[60,30+5*sin(t)],
    explicit(
        sin(x)*y^2,
        x,-2* ,2* ,
        y,-2* ,2*
    )
)$
```

For those more familiar with `plot` than with `draw` there is a second set of functions:

- `with_slider` and
- `wxanimate`.

Normally the animations are played back or exported with the frame rate chosen in the configuration of *wxMaxima*. To set the speed an individual animation is played back the variable `wxanimate_framerate` can be used:

```
wxanimate(a, 10,
    sin(a*x), [x,-5,5]), wxanimate_framerate=6$
```

The animation functions use *Maxima*'s `makelist` command and therefore shares the pitfall that the slider variable's value is substituted into the expression only if the variable is directly visible in the expression. Therefore the following example will fail:

```
f:sin(a*x);
with_slider_draw(
    a,makelist(i/2,i,1,10),
    title=concat("a=",float(a)),
    grid=true,
    explicit(f,x,0,10)
)$
```

If *Maxima* is explicitly asked to substitute the slider's value plotting works fine instead:

```
f:sin(a*x);
with_slider_draw(
    b,makelist(i/2,i,1,10),
    title=concat("a=",float(b)),
    grid=true,
    explicit(
        subst(a=b,f),
        x,0,10
    )
)$
```

2.3.7 Opening multiple plots in contemporaneous windows

While not being a provided by *wxMaxima* this feature of *Maxima* (on setups that support it) sometimes comes in handily. The following example comes from a post from Mario Rodriguez to the *Maxima* mailing list:

```
load(draw);

/* Parabola in window #1 */
draw2d(terminal=[wxt,1],explicit(x^2,x,-1,1));

/* Parabola in window #2 */
draw2d(terminal=[wxt,2],explicit(x^2,x,-1,1));

/* Paraboloid in window #3 */
draw3d(terminal=[wxt,3],explicit(x^2+y^2,x,-1,1,y,-1,1));
```

Plotting multiple plots in the same window is possible, too:

```
wxdraw(
    gr2d(
        key="sin (x)",grid=[2,2],
        explicit(sin(x),x,0,2*%pi)),
    gr2d(
    key="cos (x)",grid=[2,2],
    explicit(cos(x),x,0,2*%pi))
 );
```

2.3.8 The “Plot using draw” side pane

The “Plot using draw” sidebar hides a simple code generator that allows to generate scenes that make use of some of the flexibility of the *draw* package *maxima* comes with.

2.3.8.1 2D

Generates the skeleton of a `draw()` command that draws a 2D scene. This scene later has to be filled with commands that generate the scene’s contents, for example by using the buttons in the rows below the “2D” button.

One helpful feature of the 2D button is that it allows to setup the scene as an animation in which a variable (by default it is t) has a different value in each frame: Often a moving 2D plot allows easier interpretation than the same data in a non-moving 3D one.

2.3.8.2 3D

Generates the skeleton of a `draw()` command that draws a 3D scene. If neither a 2D or a 3D scene are set up all of the other buttons set up a 2D scene that contains the command the button generates.

2.3.8.3 Expression

Appends a standard plot of an expression like `sin(x)`, `x*sin(x)` or `x^2+2*x-4` to the `draw()` command the cursor currently is in. If there is no draw command a 2D scene with the plot is generated. Each scene can be filled with any number of plots.

2.3.8.4 Implicit plot

Tries to find all points an expression like `y=sin(x)`, `y*sin(x)=3` or `x^2+y^2=4` is true at and plots the resulting curve in the `draw()` command the cursor currently is in. If there is no draw command a 2D scene with the plot is generated.

2.3.8.5 Parametric plot

Steps a variable from a lower limit to an upper limit and uses two expressions like `t*sin(t)` and `t*cos(t)` for generating the x, y (and in 3D plots also z) coordinates of a curve that is put into the current draw command.

2.3.8.6 Points

Draws many points that can optionally be joined. The coordinates of the points are taken from a list of lists, a 2D array or one list or array for each axis.

2.3.8.7 Diagram title

Draws a title on the upper end of the diagram,

2.3.8.8 Axis

Sets up the axis.

2.3.8.9 Contour

(Only for 3D plots): Adds contour lines similar to the ones one can find in a map of a mountain to the plot commands that follow in the current `draw()` command and/or to the ground plane of the diagram. Alternatively this wizard allows skipping drawing the curves entirely only showing the contour plot.

2.3.8.10 Plot name

Adds a legend entry showing the next plot's name to the legend of the diagram. An empty name disables generating legend entries for the following plots.

2.3.8.11 Line colour

Sets the line colour for the following plots the current draw command contains.

2.3.8.12 Fill colour

Sets the fill colour for the following plots the current draw command contains.

2.3.8.13 Grid

Pops up a wizard that allows to set up grid lines.

2.3.8.14 Accuracy

Allows to select an adequate point in the speed vs. accuracy tradeoff that is part of any plot program.

2.4 Embedding graphics

If the `.wxmx` file format is being used embedding files in a *wxMaxima* project can be done as easily as per drag-and-drop. But sometimes (for example if an image's contents might change later on in a session) it is better to tell the file to load the image on evaluation:

```
show_image("man.png");
```

2.5 Startup files

The config dialogue of *wxMaxima* offers to edit two files with commands that are executed on startup:

- A file that contains commands that are executed on starting up *Maxima*: `maxima-init.mac`
- one file of additional commands that are executed if *wxMaxima* is starting *Maxima*: `wxmaxima-init.mac`

These files are in the Maxima user directory (usually `maxima` in Windows, `.maxima` otherwise) in the user's home directory / user profile directory. The location can be found out with the command: `maxima_userdir;`

2.6 Special variables wx...

- `wxsubscripts` tells *Maxima* if it should convert variable names that contain an underscore (`R_150` or the like) into subscripted variables. See `wxdeclare_subscript` for details which variable names are automatically converted.
- `wxfilename`: This variable contains the name of the file currently opened in *wxMaxima*.
- `wxplot_pngcairo` tells whether *wxMaxima* tries to use *gnuplot*'s pngcairo terminal that provides more line styles and a better overall graphics quality.
- `wxplot_size` defines the resolution of embedded plots.
- `wxchangedir`: On most operating systems *wxMaxima* automatically sets *Maxima*'s working directory to the directory of the current file. This allows file I/O (e.g. by `read_matrix`) to work without specifying the whole path to the file that has to be read or written. On Windows this feature

sometimes causes error messages and therefore can be set to `false` from the config dialogue.

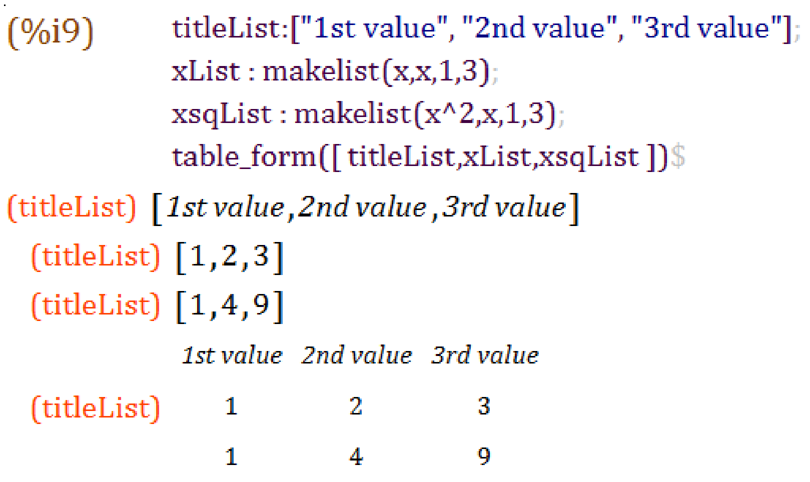
- `wxanimate_framerate`: The number of frames per second the following animations have to be played back with.
- `wxanimate_autoplay`: Automatically play animations by default?

2.7 Pretty-printing 2D output

The function `table_form()` displays a 2D list in a form that is more readable than the output from *Maxima*'s default output routine. The input is a list of one or more lists. Like the "print" command, this command displays output even when ended with a dollar sign. Ending the command with a semicolon results in the same table along with a "done" statement.

```
table_form(
    [
        [1,2],
        [3,4]
    ]
)$
```

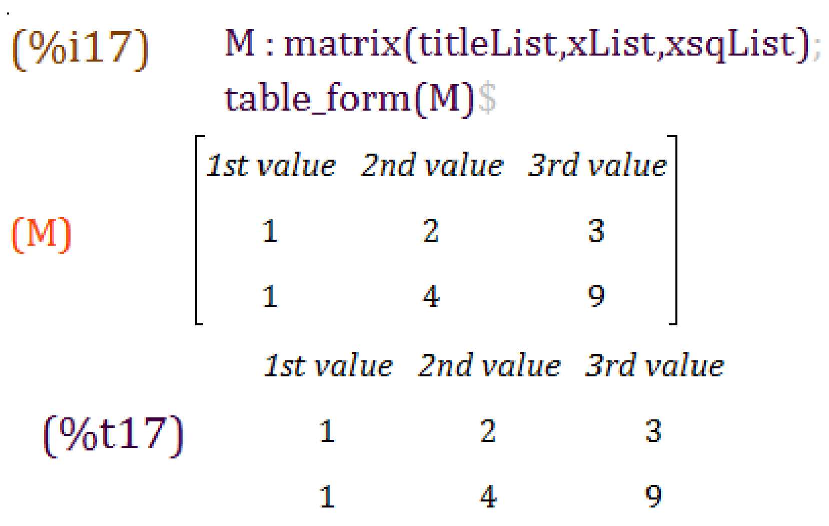
As the next example shows, the lists that are assembled by the `table_form` command can be created before the command is executed.



```
(%i9)   titleList:["1st value", "2nd value", "3rd value"];
        xList : makelist(x,x,1,3);
        xsqList : makelist(x^2,x,1,3);
        table_form([ titleList,xList,xsqList ])$
(titleList) [1st value,2nd value,3rd value]
  (titleList) [1,2,3]
  (titleList) [1,4,9]
              1st value  2nd value  3rd value
  (titleList)     1          2          3
                  1          4          9
```

Figure 10: A third table example

Also, because a matrix is a list of lists, matrices can be converted to tables in a similar fashion.



```
(%i17)  M : matrix(titleList,xList,xsqList);
        table_form(M)$

        [ 1st value   2nd value   3rd value ]
(M)     [     1           2           3     ]
        [     1           4           9     ]

            1st value   2nd value   3rd value
(%t17)          1           2           3
                1           4           9
```

Figure 11: Another table_form example

2.8 Bug reporting

wxMaxima provides a few functions that gather bug reporting information about the current system:

- `wxbuild_info()` gathers information about the currently running version of *wxMaxima*
- `wxbug_report()` tells how and where to file bugs

2.9 Marking output being drawn in red

Maxima's `box()` command causes *wxMaxima* to print its argument with a red foreground.

---

3 Risoluzione dei problemi

3.1 Cannot connect to *Maxima*

Since *Maxima* (the program that does the actual mathematics) and *wxMaxima* (providing the easy-to-use user interface) are separate programs that communicate by the means of a local network connection. Therefore the most probable cause is that this connection is somehow not working. For example a firewall could be set up in a way that it doesn't just prevent unauthorized connections from the internet (and perhaps to intercept some connections to the internet, too), but it also to blocks inter-process-communication inside the same computer. Note that since *Maxima* is being run by a Lisp processor the process communication that is blocked from does not necessarily have to be named “maxima”. Common names of the program that opens the network connection would be sbcl, gcl, ccl, lisp.exe or similar names.

On Un*x computers another possible reason would be that the loopback network that provides network connections between two programs in the same computer isn’t properly configured.

3.2 How to save data from a broken .wxmx file

Internally most modern XML-based formats are ordinary zip-files. *wxMaxima* doesn’t turn on compression, so the contents of .wxmx files can be viewed in any text editor.

If the zip signature at the end of the file is still intact after renaming a broken .wxmx file to .zip most operating systems will provide a way to extract any portion of information that is stored inside it. This can be done when there is the need of recovering the original image files from a text processor document. If the zip signature isn’t intact that does not need to be the end of the world: If *wxMaxima* during saving detected that something went wrong there will be a `wxmx~` file whose contents might help.

And even if there isn’t such a file: The `.wxmx` file is a container format and the XML portion is stored uncompressed. It it is possible to rename the `.wxmx` file to a `.txt` file and to use a text editor to recover the XML portion of the file’s contents (it starts with `<?xml version="1.0" encoding="UTF-8"?>` and ends with `</wxMaximaDocument>`. Before and after that text you will see some unreadable binary contents in the text editor).

If a text file containing only this contents (e.g. copy and paste this text into a new file) is saved as a file ending in `.xml`, *wxMaxima* will know how to recover the text of the document from it.

3.3 I want some debug info to be displayed on the screen before my command has finished

Normally *wxMaxima* waits for the whole 2D formula to be transferred before it begins to typeset. This saves time for making many attempts to typeset a only partially completed equation. There is a `disp` command, though, that will provide debug output immediately and without waiting for the current *Maxima* command to finish:

```
for i:1 thru 10 do (
   disp(i),
   /* (sleep n) is a Lisp function, which can be used */
   /* with the character "?" before. It delays the */
   /* program execution (here: for 3 seconds) */
   ?sleep(3)
)$
```

3.4 Plotting only shows a closed empty envelope with an error message

This means that *wxMaxima* could not read the file *Maxima* that was supposed to instruct *gnuplot* to create.

Possible reasons for this error are:

- The plotting command is part of a third-party package like `implicit_plot` but this package was not loaded by *Maxima*'s `load()` command before trying to plot.
- *Maxima* tried to do something the currently installed version of *gnuplot* isn't able to understand. In this case, a file ending in `.gnuplot` located in the directory, which *Maxima*'s variable `maxima_userdir` is pointing, contains the instructions from *Maxima* to *gnuplot*. Most of the time, this file's contents therefore are helpful when debugging the problem.
- Gnuplot was instructed to use the pngcairo library that provides antialiasing and additional line styles, but it was not compiled to support this possibility. Solution: Uncheck the "Use the cairo terminal for plot" checkbox in the configuration dialog and don't set `wxplot_pngcairo` to true from *Maxima*.
- Gnuplot didn't output a valid `.png` file.

3.5 Plotting an animation results in "error: undefined variable"

The value of the slider variable by default is only substituted into the expression that is to be plotted if it is visible there. Using a `subst` command that substitutes the slider variable into the equation to plot resolves this problem. At the end of section Embedding animations into the spreadsheet you can see an example.

3.6 I lost a cell contents and undo doesn't remember

There are separate undo functions for cell operations and for changes inside of cells so chances are low that this ever happens. If it does there are several methods to recover data:

- *wxMaxima* actually has two undo features: The global undo buffer that is active if no cell is selected and a per-cell undo buffer that is active if the cursor is inside a cell. It is worth trying to use both undo options in order to see if an old value can still be accessed.
- If you still have a way to find out what label *Maxima* has assigned to the cell just type in the cell's label and its contents will reappear.

- If you don't: Don't panic. In the "View" menu there is a way to show a history pane that shows all *Maxima* commands that have been issued recently.
- If nothing else helps *Maxima* contains a replay feature:

```
playback();
```

3.7 *wxMaxima* starts up with the message "Maxima process terminated."

One possible reason is that *Maxima* cannot be found in the location that is set in the "Maxima" tab of *wxMaxima*'s configuration dialog and therefore won't run at all. Setting the path to a working *Maxima* binary should fix this problem.

3.8 Maxima is forever calculating and not responding to input

It is theoretically possible that *wxMaxima* doesn't realize that *Maxima* has finished calculating and therefore never gets informed it can send new data to *Maxima*. If this is the case "Trigger evaluation" might resynchronize the two programs.

3.9 My SBCL-based *Maxima* runs out of memory

The Lisp compiler SBCL by default comes with a memory limit that allows it to run even on low-end computers. When compiling a big software package like Lapack or dealing with extremely big lists or equations this limit might be too low. In order to extend the limits SBCL can be provided with the command line parameter `--dynamic-space-size` that tells SBCL how many megabytes it should reserve. A 32bit Windows-SBCL can reserve up to 999 Megabytes. A 64-bit SBCL version running on Windows can be instructed to use more than the about 1280 Megabytes compiling Lapack needs.

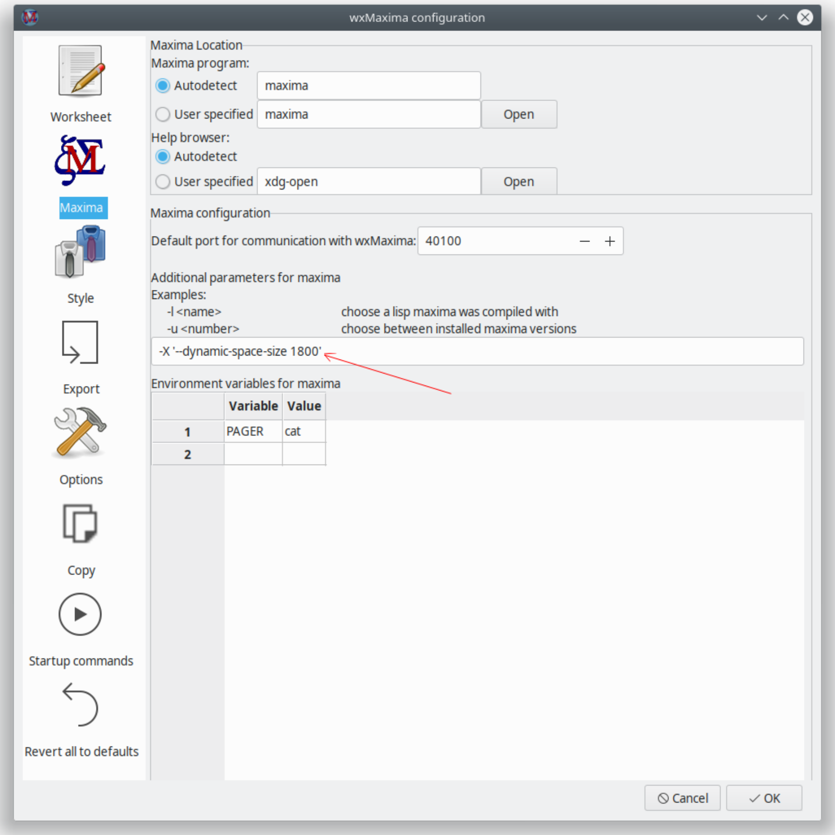
One way to provide *Maxima* (and thus SBCL) with command line parameters is the "Additional parameters for Maxima" field of *wxMaxima*'s configuration dialogue.

3.10 Input sometimes is sluggish/ignoring keys on Ubuntu

Installing the package `ibus-gtk` should resolve this issue. See (https://bugs.launchpad.net/ubuntu/+source/wxwidgets3.0/+bug/1421558) for details.

3.11 *wxMaxima* halts when *Maxima* processes Greek characters or Umlauts

If your *Maxima* is based on SBCL the following lines have to be added to your `.sbclrc`:



Maxima Location

Maxima program:

- [✓ Autodetect] maxima
- User specified maxima

Help browser:

- [✓ Autodetect]
- User specified xdg-open

Maxima configuration

Default port for communication with wxMaxima: 40100

Additional parameters for maxima

Examples:

-l <name>	choose a lisp maxima was compiled with
-u <number>	choose between installed maxima versions

```
-X '--dynamic-space-size 1800'
```

Environment variables for maxima

	Variable	Value
1	PAGER	cat
2		

Figure 12: sbcl memory

```
(setf sb-impl::*default-external-format* :utf-8)
```

The folder this file has to be placed in is system- and installation-specific. But any SBCL-based *Maxima* that already has evaluated a cell in the current session will happily tell where it can be found after getting the following command:

```
:lisp (sb-impl::userinit-pathname)
```

3.12 Plotting

3.12.1 Can I make *wxMaxima* output both image files and embedded plots at once?

The worksheet embeds .png files. *wxMaxima* allows the user to specify where they should be generated:

```
wxdraw2d(
    file_name="test",
    explicit(sin(x),x,1,10)
);
```

If a different format is to be used it is easier to generate the images and then to import them into the worksheet again:

```
load("draw");
pngdraw(name,[contents]):=
(
    draw(
        append(
            [
                terminal=pngcairo,
                dimensions=wxplot_size,
                file_name=name
            ],
            contents
        )
    ),
    show_image(printf(false,"~a.png",name))
);
pngdraw2d(name,[contents]):=
    pngdraw(name,gr2d(contents));

pngdraw2d("Test",
        explicit(sin(x),x,1,10)
);
```

3.12.2 Can I set the aspect ratio of a plot?

Not directly using *Maxima*. But there are gnuplot commands for it:

```
wxdraw2d(
    proportional_axis=xy,
    explicit(sin(x),x,1,10)
),wxplot_size=[1000,1000];
```

---

4 FAQ

4.1 Is there a way to make more text fit on a LaTeX page?

Yes. Use the LaTeX package "geometry" to specify the size of the borders.

You can add the following line to the LaTeX preamble (for example by using the respective field in the config dialogue ("Export"->"Additional lines for the TeX preamble"), to set borders of 1cm):

```
\usepackage[left=1cm,right=1cm,top=1cm,bottom=1cm]{geometry}
```

4.2 Is there a dark mode?

If wxWidgets is new enough *wxMaxima* will automatically be in dark mode if the rest of the operating system is. The worksheet itself is by default equipped with a bright background. But it can be configured otherwise. Alternatively there is a `View/Invert worksheet brightness` menu entry that allows to quickly convert the worksheet from dark to bright and vice versa.

4.3 *wxMaxima* sometimes hangs for a several seconds once in the first minute

wxMaxima delegates some big tasks like parsing *Maxima*'s >1000-page-manual to background tasks, which normally goes totally unnoticed. In the moment the result of such a task is needed, though, it is possible that *wxMaxima* needs to wait a couple of seconds before it can continue its work.

---

5 Argomenti della riga di comando

Most operating systems provide less complicated ways of starting programs than the command line so this possibility is only rarely used. *wxMaxima* still provides some command line switches, though.

- `-v` or `--version`: Output the version information
- `-h` or `--help`: Output a short help text
- `-o` or `--open=<str>`: Open the filename given as argument to this command-line switch

- `-e` or `--eval`: Evaluate the file after opening it.
- `-b` or `--batch`: If the command-line opens a file all cells in this file are evaluated and the file is saved afterwards. This is for example useful if the session described in the file makes *Maxima* generate output files. Batch-processing will be stopped if *wxMaxima* detects that *Maxima* has output an error and will pause if *Maxima* has a question: Mathematics is somewhat interactive by nature so a completely interaction-free batch processing cannot always be guaranteed.
- `--logtostdout`: Log all "debug messages" sidebar messages to stderr, too.
- `--pipe`: Pipe messages from Maxima to stdout.
- `--exit-on-error`: Close the program on any maxima error.
- `-f` or `--ini=<str>`: Use the init file that was given as argument to this command-line switch
- `-u`, `--use-version=<str>`: Use maxima version `<str>`.
- `-l`, `--lisp=<str>`: Use a Maxima compiled with Lisp compiler `<str>`.
- `-X`, `--extra-args=<str>`: Allows to specify extra Maxima arguments
- `-m` or `--maxima=<str>`: allows to specify the location of the *maxima* binary
- `--enableipc`: Lets Maxima control wxMaxima via interprocess communications. Use this option with care.

Instead of a minus some operating systems might use a dash in front of the command-line switches.