

# wxMaxima

## Contents

Das Handbuch von wxMaxima	2
1 Einführung in wxMaxima	3
1.1 Maxima und wxMaxima	3
1.1.1 Maxima	4
1.1.2 wxMaxima	4
1.2 Grundlagen zum Arbeitsblatt	5
1.2.1 Der Arbeitsblatt-Ansatz	5
1.2.2 Zellen	6
1.2.3 Horizontale und vertikale Cursors	8
1.2.4 Eingabezellen an Maxima schicken	8
1.2.5 Auto-Vervollständigung	8
1.2.6 Seitenleisten	11
1.2.7 MathML-Ausgabe	11
1.2.8 Markdown-Unterstützung	11
1.2.9 Tastenkürzel	13
1.2.10 Direkte Eingabe von TeX-Befehlen	13
1.3 Dateiformate	13
1.3.1 .mac	13
1.3.2 .wxm	15
1.3.3 .wxmx	15
1.4 Auto-Vervollständigung	15
1.4.1 Die Standard-Bildwiederholrate bei Animationen	15
1.4.2 Die Standardgröße der Diagramme bei neuen Maxima Sitzungen	15
1.4.3 Automatisches Schließen von Klammern	17
1.4.4 Arbeitsblatt nicht automatisch speichern	17
1.4.5 Wo wird die Konfiguration gespeichert?	17
2 Erweiterungen für Maxima	17
2.1 Variablen mit tiefgestelltem Index	17
2.2 Meldungen in der Statusleiste	18
2.3 Diagramme	18
2.3.1 Der Einbau von Diagrammen in das Arbeitsblatt	18
2.3.2 Eingebettete Diagramme größer oder kleiner machen	18
2.3.3 Hochqualitativere Diagramme	19

2.3.4	Öffnen eingebetteter Diagramme in interaktiven gnuplot Fenstern . . .	19
2.3.5	Öffnen der Gnuplot Kommandozeile in plot Fenstern . . . . .	19
2.3.6	Einbetten von Animationen in das Arbeitsblatt . . . . .	19
2.3.7	Mehrere Diagramme gleichzeitig in Fenstern öffnen . . . . .	21
2.3.8	Die “Plotte mittels Draw”-Seitenleiste . . . . .	22
2.4	Einbetten von Bildern . . . . .	23
2.5	Startbefehle . . . . .	23
2.6	Spezielle Variablen, die wxMaxima definiert . . . . .	24
2.7	2D-Tabellen sauber ausgeben . . . . .	24
2.8	Fehler melden . . . . .	24
2.9	Rotes Markieren von Formelteilen . . . . .	26
3	Fehlersuche . . . . .	26
3.1	Keine Verbindung zu Maxima möglich . . . . .	26
3.2	Wie repariere ich kaputte .wxmx-Dateien? . . . . .	26
3.3	Ich will Statusmeldungen am Bildschirm ausgeben, während mein Befehl ausgeführt wird . . . . .	27
3.4	Statt eines Diagramms wird ein Briefumschlag mit einer Fehlermeldung dargestellt . . . . .	27
3.5	Animationen enden in “Error: Undefined Variable” . . . . .	27
3.6	Undo erinnert sich nicht an das, was ich brauche . . . . .	27
3.7	wxMaxima meldet gleich beim Hochfahren “Maxima hat sich beendet.” . . .	28
3.8	Maxima hört nicht auf zu rechnen und reagiert nicht auf Eingaben . . . . .	28
3.9	SBCL beschwert sich über einen Mangel an Speicher . . . . .	28
3.10	Ubuntu: Die Tastatur ist langsam oder ignoriert einzelne Tasten . . . . .	28
3.11	wxMaxima stoppt, wenn Maxima griechische Buchstaben oder Umlaute verarbeitet . . . . .	28
3.12	Diagramme . . . . .	30
3.12.1	Kann wxMaxima das eingebettete Diagramm gleich noch als Datei ausgeben? . . . . .	30
3.12.2	Kann ich das Seitenverhältnis des Plots angeben? . . . . .	30
4	Häufig gestellte Fragen . . . . .	31
4.1	Gibt es eine Möglichkeit mehr Text auf eine LaTeX-Seite zu schreiben? . . . .	31
4.2	Gibt es einen Dark Mode? . . . . .	31
4.3	wxMaxima hängt manchmal in der ersten Minute einmal für mehrere Sekunden . . . . .	31
5	Kommandozeilen-Optionen . . . . .	31

## Das Handbuch von wxMaxima

wxMaxima ist ein graphisches Benutzerinterface (GUI) für das Computer-Algebrasystem Maxima. wxMaxima erlaubt die Nutzung aller Funktionen dieses Programms und bietet Assistenten für dessen wichtigste Funktionen an. Dieses Handbuch beschreibt die Features, die wxMaxima zu einer der beliebtesten graphischen Benutzerumgebung für Max-

ima gemacht haben.



Figure 1: wxMaxima logo

Vor dem eigentlichen Inhalt ist es vielleicht angebracht, die Navigation zu erklären: Die nun folgende Liste enthält Hyperlinks auf die einzelnen Kapitel der Dokumentation. Ein Klick auf Inhaltsverzeichnis führt zu einem detaillierten Inhaltsverzeichnis. Dieser Link kommt öfter vor, um die Navigation zu erleichtern.

- Einführung:

Die Grundlagen

- Erweiterungen:

Die Kommandos, mittels deren wxMaxima Maxima erweitert

- Fehlersuche:

Was tun, wenn wxMaxima nicht wie erwartet funktioniert

- FAQ:

Häufig gestellte Fragen

- Befehlszeile:

Die Kommandozeilenargumente von wxMaxima

---

## 1 Einführung in wxMaxima

### 1.1 Maxima und wxMaxima

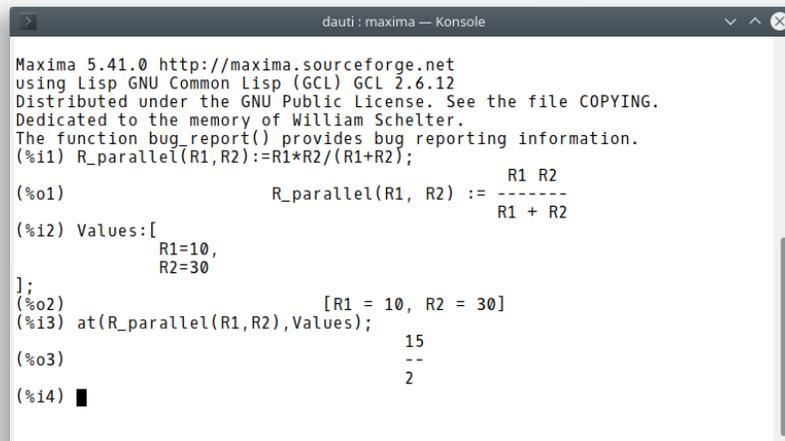
Im Open-Source-Bereich ist es üblich, große Systeme in kleine Projekte aufzuteilen, die jeweils klein genug sind, dass eine endliche Menge an Entwicklern ausreicht, um sie weiterzuentwickeln. Beispielsweise ist ein CD-Brennprogramm aus einem Kommandozeilen-tool zusammengesetzt, das die CDs brennt und eine graphische Benutzerumgebung, die

dieses Tool bedient. Ein Vorteil dieses Ansatzes ist es, daß dasselbe Kommandozeilenprogramm von vielen Stellen aus genutzt werden: Das Brennprogramm, ein "Sende an die CD"-Knopf und beispielsweise ein Backup-Tool. Ein weiterer Vorteil ist es, dass auf diese Weise für eine Funktionalität mehrere Benutzerinterfaces angeboten werden können.

Ein Computer-Algebrasystem (CAS) wie Maxima ist hervorragend für diesen Ansatz geeignet: Es kann die Logik hinter einem Taschenrechner liefern, der mit beliebig langen Zahlen hantieren kann, Formeln für ein größeres System, z.B. Sage umstellen. Alternativ kann es direkt verwendet werden. Dies kann von der Kommandozeile aus geschehen, oder von wxMaxima aus, das eine komfortablere Bedienung unterstützt.

### 1.1.1 Maxima

Maxima ist ein komplettes Computer-Algebrasystem (CAS): Ein Programm, das die Formel, nicht nur die Zahl sucht, die ein mathematisches Problem löst. Auch wenn es darauf spezialisiert ist, mit Buchstaben zu rechnen, bietet es auch eine Menge an Funktionen, die Probleme lösen, für die nur numerische Lösungen existieren.



```
dauti: maxima — Konsole
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);
(%o1)          R_parallel(R1, R2) := 
$$\frac{R1 R2}{R1 + R2}$$

(%i2) Values:[
      R1=10,
      R2=30
];
(%o2)          [R1 = 10, R2 = 30]
(%i3) at(R_parallel(R1,R2),Values);
(%o3)          
$$\frac{15}{2}$$

(%i4) █
```

Figure 2: Maxima Screenshot, Kommandozeile

Umfangreiche Dokumentation für Maxima ist im Internet verfügbar. Teile davon sind auch im Hilfe-Menü von wxMaxima erhältlich. Ein Druck auf die Hilfe-Taste (auf den meisten Systemen ist dies die F1-Taste) springt automatisch zur Stelle im Maxima-Handbuch, wo das Kommando unter dem Cursor erklärt wird.

### 1.1.2 wxMaxima

wxMaxima ist ein graphisches Benutzer-Interface, das die komplette Funktionalität und Flexibilität von Maxima zu nutzen erlaubt. Zudem bietet es viele Funktionalitäten an,

die die Verwendung von Maxima einfacher gestalten. Beispielsweise kann es den Inhalt eines Arbeitsblattes, einer Zelle des Arbeitsblattes oder einen Teil einer Formel nach LaTeX, MathML oder in eine RTF-Formel für eine Textverarbeitung konvertieren. Dokumentation für wxMaxima kann im Internet gefunden werden, oder über das Hilfemenü.

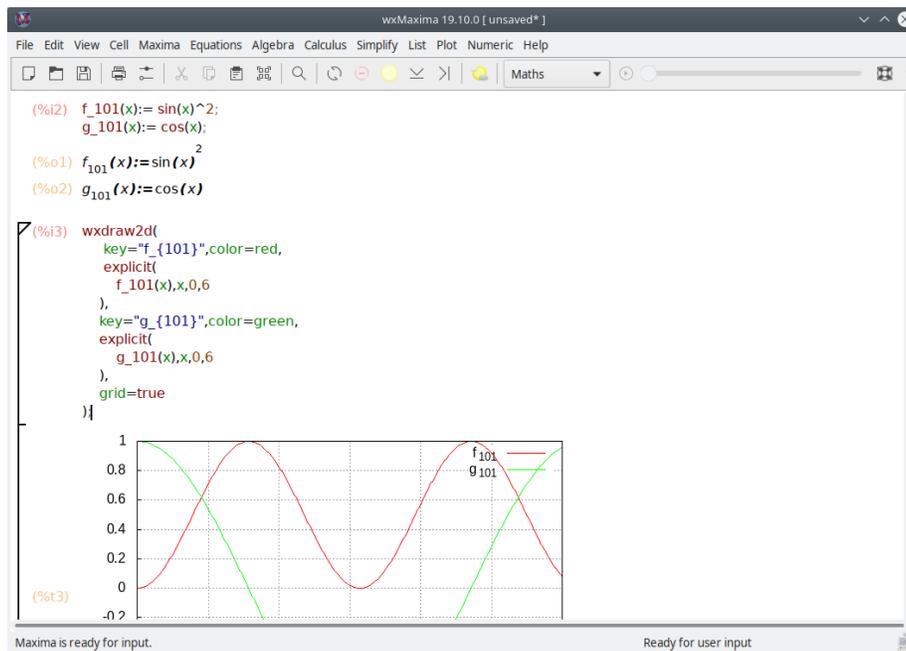


Figure 3: wxMaxima Fenster

wxMaxima lässt alle Berechnungen im Hintergrund durch das Kommandozeilen-Werkzeug Maxima durchführen.

## 1.2 Grundlagen zum Arbeitsblatt

wxMaxima ist größtenteils selbsterklärend. Diese Internetseite bietet einige Beispiele an, wovon das “10-Minuten-Tutorial” besonders zu empfehlen ist. Dieses Handbuch beschreibt einige zusätzliche Aspekte von wxMaxima.

### 1.2.1 Der Arbeitsblatt-Ansatz

Eine der Sachen, die neue Benutzer oft verwirrt, ist, dass das Arbeitsblatt von wxMaxima in Zellen aufgeteilt ist, die nur auf Befehl vom Benutzer an Maxima gesendet werden. Dieser Ansatz ist jedoch für die Fehlersuche praktisch. Auch hat der gegenüber der Alternative, nach jeder Änderung alle Zellen im das Arbeitsblatt neu auszuwerten, oft viel Zeit.

Wenn Text in wxMaxima eingegeben wird, erzeugt er automatisch eine neue Zelle des Arbeitsblattes. Wenn dies eine Code-Zelle ist, kann ihr Inhalt an Maxima gesendet werden und das Resultat dieser Aktion wird unter der Zelle angezeigt, wie unten abgebildet.

```
(%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)
```

Figure 4: Eingabe/Ausgabe Zelle

Wird eine Code-Zelle ausgewertet, weist ihr Maxima eine Marke zu (standardmäßig ist diese in roter Schrift gehalten und beginnt mit einem %i), über das sie später wieder referenziert werden kann. Für die Ausgabe wird eine Marke generiert, die standardmäßig mit %o beginnt und nicht angezeigt wird, außer der Benutzer hat der Formel einen sprechenden Namen gegeben, der stattdessen angezeigt werden kann.

Außer Code-Zellen kennt wxMaxima auch Testzellen und solche mit Bildern oder Überschriften. Jede Zelle hat ihren eigenen Speicher für das Rückgängigmachen von Aktionen, was sich oft als hilfreich erwiesen hat. Zudem besitzt, wie in fast allen Applikationen, das Arbeitsblatt einen eigenen Speicher für die Rückgängigmachen-Funktion.

Die nun folgende Abbildung zeigt verschiedene Zelltypen (Titelzellen, Untertitelzellen, Textzellen, Eingabe/Ausgabezellen und eine Grafikkelle).

### 1.2.2 Zellen

Das Arbeitsblatt ist in Zellen aufgeteilt, von denen jede mehr Zellen oder Elemente der folgenden Typen enthalten kann:

- Eine oder mehr Zeilen an Eingabe für Maxima
- Ein oder mehrere Bilder
- Ausgaben oder Fragen von Maxima
- Einen Textblock mit Dokumentation
- Eine Überschrift

Wenn Text eingegeben ist, erzeugt wxMaxima normalerweise gleich eine Code-Zelle. Andere Zelltypen können über das "Zellen"-Menü, die dort dokumentierten Tastenkombinationen oder über die Werkzeugleiste erzeugt werden. Wenn eine nicht-mathematische Zelle (Überschrift, Text, etc.) erzeugt wurde, wird alles was eingetippt wird, als Text interpretiert.

Zusätzlicher Text kann in einer mathematischen Zelle als Kommentar zwischen /\* und \*/ (wie z.B. in der Programmiersprache C) eingegeben werden: /\* Dieser Kommentar wird von Maxima nicht beachtet \*/.

The screenshot shows the wxMaxima 19.10.0 interface. The main window contains the following content:

- Title cell:** **This is a title cell**
- Section cell:** **1 This is a section cell**
- Subsection cell:** **1.1 This is a subsection cell**
- Subsubsection cell:** **1.1.1 This is a subsubsection cell**
- Heading level 5 cell:** **1.1.1.1 This is a heading of level 5**
- Heading level 6 cell:** **1.1.1.1.1 This is a heading of level 6**

Below the headings, there is a text cell: "Now a Maxima input cell (and the computed output):"

The input cell contains the code: `(%i1) 'integrate(x^2,x)=integrate(x^2,x);`

The output cell contains the result: `(%o1)  $\int x^2 dx = \frac{x^3}{3}$`

Below the output, there is a text cell: "This is a text only cell. Now comes a pagebreak cell and then a image cell:"

A horizontal dotted line follows, then a figure caption: "Figure 1:./home/dauti/Software/wxmaxima/info/wxMaximaLogo.png"

An image cell contains the wxMaxima logo, which is a stylized 'M' inside a square frame.

At the bottom left, it says "Maxima is ready for input." and at the bottom right, "Ready for user input".

On the right side, there is a "Table of Contents" panel with the following entries:

- 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

Below the table of contents is an "insert" panel with a grid of options:

Text	Title
Section	Subsection
Subsubsection	Heading 5
Heading 6	Image
Pagebreak	

Figure 5: Ein Beispiel verschiedener Zelltypen

### 1.2.3 Horizontale und vertikale Cursors

Wenn in einer Textverarbeitung versucht wird, einen Satz auszuwählen, wird diese versuchen, Beginn und Ende der Auswahl so zu verschieben, dass ganze Wörter ausgewählt sind. wxMaxima wird aus diesem Grund, wenn mehr als eine Zelle ausgewählt wird, die Auswahl auf ganze Zellen ausdehnen.

Was ungewöhnlich erscheinen kann ist, dass wxMaxima alternativ einen vertikalen oder einen horizontalen Cursor darstellen kann:

- Der Cursor wird vertikal dargestellt wird, wenn er zwischen zwei Zellen zu liegen kommt.
- Innerhalb einer Zelle wird der Cursor vertikal dargestellt.

### 1.2.4 Eingabezellen an Maxima schicken

Die Befehle in einer Codezelle werden ausgeführt, sobald CTRL+ENTER, SHIFT+ENTER oder ENTER key im Nummernblock gedrückt wird. wxMaxima nimmt Befehle mit CTRL+ENTER oder SHIFT+ENTER entgegen, aber wxMaxima kann auch konfiguriert werden, dass Befehle nach ENTER ausgeführt werden..

### 1.2.5 Auto-Vervollständigung

wxMaxima versucht, automatisch die Namen von Befehlen oder Variablen zu vervollständigen, wenn der Menüpunkt (Vervollständige Befehl) angewählt wird, oder die Tastenkombination CTRL+SPACE gedrückt wird. Die automatische Vervollständigung erkennt oft den Kontext, in dem sie ausgeführt wird, und kann beispielsweise Dateinamen oder Einheiten für ezUnits vorschlagen.

Außer der Vervollständigung des Namens einer Datei, einer Einheit, eines Kommandos oder eines Variablennamens kann für viele Befehle eine Liste der erwarteten Argumente angezeigt werden. Hierfür muss einfach SHIFT+CTRL+SPACE gedrückt werden, oder der entsprechende Menüeintrag gewählt (Zelle/Zeige Parameter).

#### 1.2.5.1 Griechische Zeichen

Computer speichern Zeichen meist als 8-Bit-Werte, was maximal 256 unterschiedliche Typen von Zeichen erlaubt. Die meisten Sprachen nutzen inklusive Steuerzeichen, Ziffern und ein paar Zeichen, aus denen Graphiken zusammengesetzt werden können, weniger als dies.

Für die Mehrzahl der Länder ist aus diesem Grund eine Codepage mit 256 Zeichen definiert worden, die allerdings in den meisten Ländern beispielsweise keine griechischen Buchstaben beinhaltet, die in der Mathematik oft verwendet werden. Um diese Begrenzung zu umgehen, wurde Unicode entwickelt: Eine Weise, Zeichen auszudrücken, die englischen Text wie gewohnt arbeiten lässt, aber viel mehr als 256 Zeichen ermöglichen.

Wenn Maxima mit einem Lisp-Compiler generiert wurde, der Unicode unterstützt oder sich nicht darum kümmert, auf welche Weise Zeichen codiert werden, unterstützt es Unicode.

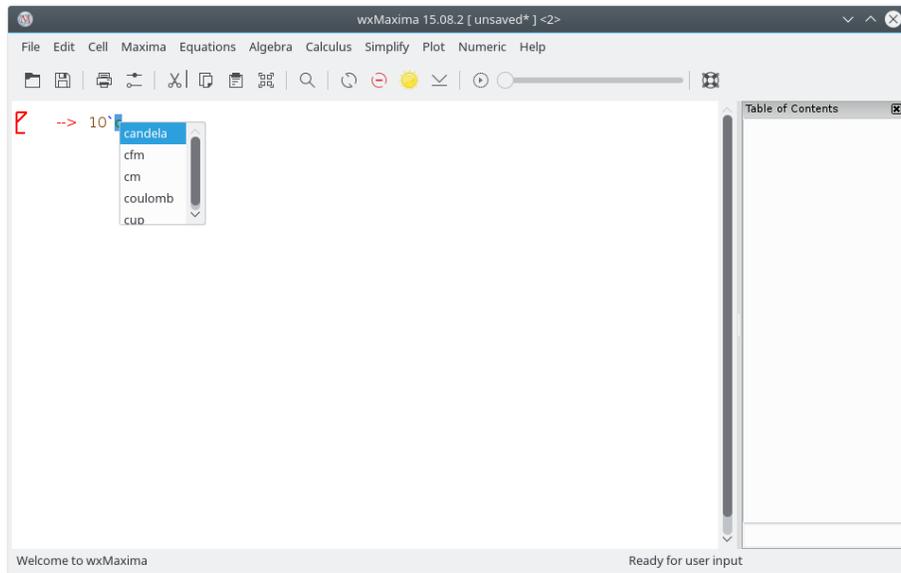


Figure 6: ezUnits

Da meist eine dieser Bedingungen gegeben ist, bietet wxMaxima eine Methode an, griechische Zeichen mit der Tastatur einzugeben:

- Ein griechischer Buchstabe kann eingegeben werden, indem erst auf die ESC-Taste gedrückt wird, und dann sein Name einzugeben begonnen wird.
- Ebenso kann er eingegeben werden, indem ESC gedrückt wird, ein Buchstabe (bzw. zwei für den griechischen Buchstaben omicron) und dann erneut ESC. In diesem Fall werden die folgenden Zeichen unterstützt:

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

key	Greek letter	key	Greek letter	key	Greek letter
H	Eta	Om	Omicron	Y	Psi
T	Theta	P	Pi	O	Omega

Derselbe Mechanismus erlaubt es auch, einige andere mathematische Symbole einzugeben:

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

keys to enter	mathematical symbol
qed	end of proof
nabla	a nabla operator
sum	sum sign
prod	product sign
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

Wenn das gewünschte Zeichen auf diese Weise nicht erreichbar ist, können Zeichen auch als ESC (Nummer des Zeichens (hexadezimal)) ESC eingegeben werden.

ESC 61 ESC erzeugt daher ein  $\alpha$ .

Viele der Unicode-Symbole (mit der Ausnahme der logischen Verknüpfungen) haben keine direkte Entsprechung in Maxima und werden daher als normale Zeichen interpretiert. Falls Maxima eine Lisp-Version verwendet, die Unicode nicht unterstützt, kann das eine Fehlermeldung verursachen.

### 1.2.6 Seitenleisten

Die meisten Maxima-Befehle und eine Liste der zuletzt verwendeten Befehle werden in Seitenleisten angeboten, die über das "Ansicht"-Menü aktiviert und an beliebige Stellen in oder außerhalb des wxMaxima-Fensters verschoben werden können. Ebenso kann eine Seitenleiste mit griechischen Buchstaben geöffnet werden, die eine alternative Methode anbietet, griechische Buchstaben einzugeben.

### 1.2.7 MathML-Ausgabe

Einige Textverarbeitungen verstehen entweder MathML (Der Formeleditor von LibreOffice 5.1 kann zumindest entsprechende Gleichungen importieren). Andere unterstützen Mathematik im RTF-Format. wxMaxima bietet daher einige entsprechende Optionen im Rechtsklick-Menü an.

### 1.2.8 Markdown-Unterstützung

wxMaxima unterstützt einige Markdown-Kommandos, die nicht mit Konstrukten kollidieren, die in mathematischen Formeln vorkommen. Eines dieser Elemente sind Aufzählungen:

Normaler Text

- \* Ein eingerückter Aufzählungspunkt
- \* Ein zweiter Aufzählungspunkt



```
* Eine Aufzählung in der Aufzählung
* Ein zweiter Aufzählungspunkt der Aufzählung in der Aufzählung
* Ein Punkt in der äußeren Aufzählung
Normaler text
```

wxMaxima erkennt Text, der mit einem “>” beginnt, als ein Zitat:

```
Normaler Text
> Zitat Zitat Zitat
> Zitat Zitat Zitat
> Zitat Zitat Zitat
> Zitat Zitat Zitat
Normaler Text
```

Die TeX- und HTML-Ausgabe von wxMaxima erkennen auch => und ersetzen es durch das entsprechende Unicode-Symbol:

```
cogito => sum.
```

Andere Symbole, die vom HTML- und TeX-Export erkannt werden, sind <= und >=, <=>, einfache Pfeile (<->, -> und <-) und +/-. Die TeX-Ausgabe erkennt ebenfalls << und >>.

### 1.2.9 Tastenkürzel

Die meisten Tastenkürzel entstammen den Menüs, was bedeutet, dass sie mit diesen übersetzt werden können, falls die Tastatur der aktuellen Sprache dies erforderlich macht. Nicht dort dokumentiert ist:

- CTRL+SHIFT+DELETE löscht eine komplette Zelle.
- CTRL+TAB or CTRL+SHIFT+TAB startet das automatische Vervollständigen.
- SHIFT+SPACE fügt ein nicht-umbrechbares Leerzeichen ein.

### 1.2.10 Direkte Eingabe von TeX-Befehlen

Wenn eine Textzelle mit “TeX:” beginnt, wird der Rest ihres Inhalts bei der Konvertierung des Dokuments nach TeX unverändert ausgegeben.

## 1.3 Dateiformate

Das Arbeitsblatt kann auf verschiedene Weisen gespeichert werden:

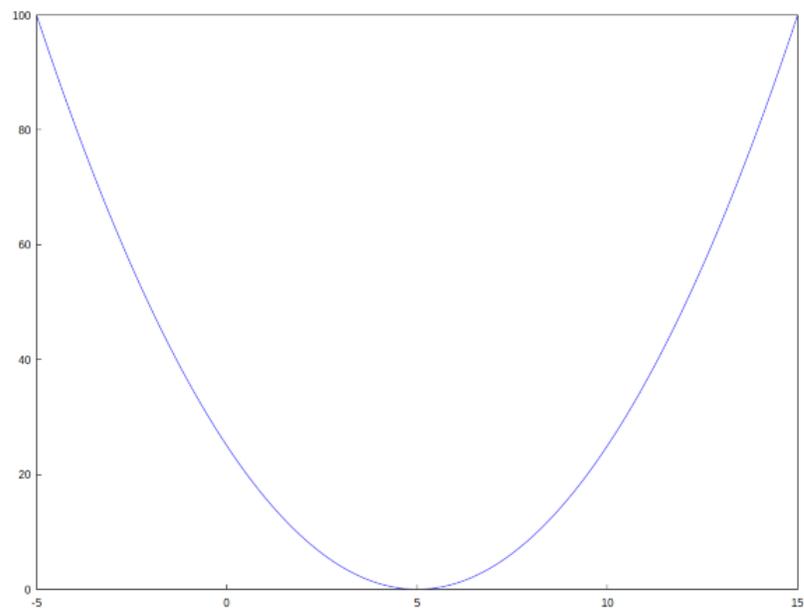
### 1.3.1 .mac

.mac-Dateien sind Textdateien mit Maxima-Befehlen. Sie können über Maxima’s `batch()` oder `load()`-Befehl oder in wxMaxima über “Datei/Batch File laden” gelesen werden.

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()

Achtung: Obwohl die Datei `Quadratic.mac` eine übliche Maxima-extension hat (`.mac`), kann sie nur durch `wxMaxima` verarbeitet werden, der Befehl `wxdraw2d()` ist eine `wxMaxima`-Erweiterung für Maxima.

`.mac`-Dateien eignen sich dafür, eine Bibliothek von Maxima-Befehlen aufzubauen. Ausreichend Information über die Struktur eines `wxMaxima`-Arbeitsblatts, dass sie es erlauben, dieses wieder zu rekonstruieren, enthalten sie aber nicht.

### 1.3.2 `.wxm`

`.wxm`-Dateien sollten das Arbeitsblatt außer Maxima's Ausgabe enthalten. Maxima >5.38 kann `.wxm`-Dateien über den `load()`-Befehl lesen. Das `.wxm`-Format ist allerdings oft weniger kompatibel mit älteren Versionen von `wxMaxima`, als das `.wxmx`-Format.

### 1.3.3 `.wxmx`

Dieses XML basiertes Dateiformat enthält das gesamte Arbeitsblatt, inklusive Eigenschaften wie den Zoom-Faktor oder die Watchlist. Es ist das empfohlene Dateiformat für `wxMaxima`-Arbeitsblätter.

## 1.4 Auto-Vervollständigung

Einige Variablen, über die Maxima konfiguriert wird, können auf zwei Arten eingestellt werden:

- Die können global im Konfigurationsdialog eingestellt werden.
- Ebenso können sie von Maxima aus überschrieben werden.

### 1.4.1 Die Standard-Bildwiederholrate bei Animationen

Die Bildwiederholgeschwindigkeit für Animationen wird in der Variable `wxanimate_framerate` gespeichert. `wxMaxima` setzt sie auf den Wert aus dem Konfigurationsdialog, wenn ein neues Maxima gestartet wird.

### 1.4.2 Die Standardgröße der Diagramme bei neuen Maxima Sitzungen

`wxplot_size` definiert, wie groß die nachfolgenden in das Arbeitsblatt eingebauten Diagramme sind.

Diese Variable kann notfalls auch nur für das aktuelle Diagramm gesetzt werden:

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

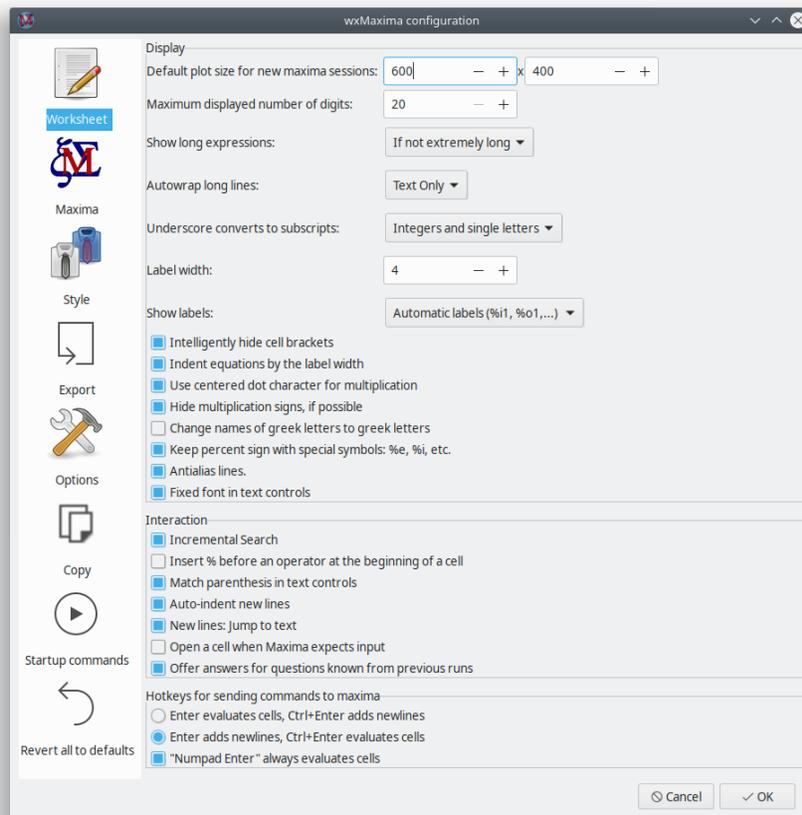


Figure 9: wxMaxima Konfiguration 1

### 1.4.3 Automatisches Schließen von Klammern

Diese Option schaltet zwei Funktionen ein:

- Wenn eine öffnende Klammer eingegeben wird, fügt wxMaxima automatisch eine schließende Klammer ein.
- Markierter Text wird, wenn man eine geöffnete Klammer einzugeben versucht, automatisch in Klammern gesetzt.

### 1.4.4 Arbeitsblatt nicht automatisch speichern

Wenn diese Option gewählt ist, wird das Arbeitsblatt nur überschrieben, wenn der Benutzer es speichert. Ein aktuelles Backup wird in diesem Fall im temp-Verzeichnis abgelegt.

Ist diese Option nicht angewählt, arbeitet wxMaxima wie eine moderne Handy-Applikation:

- Dateien werden beim Schließen automatisch gespeichert
- und sie werden alle n Minuten zwischengespeichert.

### 1.4.5 Wo wird die Konfiguration gespeichert?

Auf Linux/Unix-Rechern wird die Konfiguration im Home-Verzeichnis in der Datei `.wxMaxima` gespeichert, oder (für wxWidgets > 3.1.1) in `.config/wxMaxima.conf` (XDG-Standard). Die wxWidgets-Version kann über `wxbuild_info()` oder über das Hilfe->Über-Menü abgerufen werden. wxWidgets ist eine Bibliothek für graphische Benutzeroberflächen, die auf verschiedensten Betriebssystemen läuft und für das wx im Namen von wxMaxima verantwortlich ist. (Da der Dateiname mit einem Punkt beginnt, ist `.wxMaxima` oder `.config` eine versteckte Datei).

Unter Windows wird die Konfiguration in der Registry unter `HKEY_CURRENT_USER\Software\wxMaxima` gespeichert.

---

## 2 Erweiterungen für Maxima

wxMaxima's Hauptaufgabe ist es, die Ein- und Ausgaben von Maxima graphisch darzustellen. An einigen Stellen fügt es jedoch Funktionalitäten zu Maxima hinzu:

### 2.1 Variablen mit tiefgestelltem Index

Wenn `wxsubscripts` auf `true` gesetzt ist, werden Variablennamen im Format `x_y` mit einem tiefgestellten `y` dargestellt, wenn

- `y` ein einzelner Buchstabe ist
- oder `y` eine ganze Zahl ist

Wenn nicht, kann jede Variable als mit tiefgestelltem `y` ausgestattet deklariert werden mittels `wxdeclare_subscript(variablenname);` oder `wxdeclare_subscript([variablenname1,variablenname2,...]);` Rückgängig gemacht werden kann dies mittels `wxdeclare_subscript(variablenname,false);`

## 2.2 Meldungen in der Statusleiste

Kommandos, die lange arbeiten, können Fortschrittsinformationen in die Statuszeile ausgeben. Diese Informationen überschreiben deren Inhalt. `wxstatusbar()` kann sogar in Bibliotheken verwendet werden, bei denen nicht bekannt ist, ob sie mit `wxMaxima` oder in einem Maxima ohne Frontend verwendet werden: Wenn `wxMaxima` diese Funktion nicht definiert hat, wird sie von Maxima einfach als undefinierte Funktion betrachtet und nicht weiter bearbeitet.

```
for i:1 thru 10 do (  
  /* Gib dem User Bescheid, wie weit wir schon sind */  
  wxstatusbar(concat("Pass ",i)),  
  /* (sleep n) ist eine Lisp-Funktion, die mit einem */  
  /* "?" Zeichen vorher verwendet werden kann. */  
  /* Sie verzögert die Programmausführung für n Sekunden */  
  /* (in diesem Beispiel: für 3 Sekunden). */  
  ?sleep(3)  
)$
```

## 2.3 Diagramme

Diagramme haben per definitionem mit einer graphischen Benutzerumgebung zu tun, weswegen an dieser Stelle Erweiterungen von Maxima zu erwarten sind.

### 2.3.1 Der Einbau von Diagrammen in das Arbeitsblatt

Maxima weist normalerweise `gnuplot` an, Diagramme in eigenen Fenstern darzustellen. Wenn dem entsprechenden `draw` oder `plot`-Kommando ein "wx" vorangestellt wird, wird das Bild stattdessen in das Arbeitsblatt eingebaut: `wxplot2d` arbeitet wie `plot2d`, aber im Arbeitsblatt, `wxplot3d` entspricht `plot3d`, `wxdraw` und `wxhistogram` funktionieren wie `draw` und `histogram`, aber betten das Resultat ein.

### 2.3.2 Eingebettete Diagramme größer oder kleiner machen

Wie bereits beschrieben kann die Größe von Diagrammen mittels der Variable `wxplot_size` definiert werden. Ihr Inhalt kann jederzeit gelesen oder geändert werden:

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

Wenn die Größe nur des aktuellen Diagramms geändert werden soll, erlaubt Maxima dies für die aktuelle Zelle zu machen. In diesem Beispiel wird `wxplot_size = [wert1, wert2]` an den `wxdraw2d( )` Befehl angehängt, es ist nicht Teil des `wxdraw2d` Befehls.

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

### 2.3.3 Hochqualitative Diagramme

Gnuplot bietet keine portable Möglichkeit an, festzustellen, ob es Cairo unterstützt, eine Bibliothek, die Graphik mit Antialiasing und den unterschiedlichsten Linienmustern zu zeichnen erlaubt. Wurde Gnuplot mit diesen Möglichkeiten kompiliert, schaltet die entsprechende Option im Konfigurationsmenü oder ein `wxplot_pngcairo:true` die Unterstützung hierfür ein. Wurde Gnuplot ohne Cairo-Unterstützung kompiliert, führt `wxplot_pngcairo:true` jedoch zu Fehlermeldungen anstelle von Plots.

### 2.3.4 Öffnen eingebetteter Diagramme in interaktiven gnuplot Fenstern

Wenn ein Diagramm mittels einem `wxdraw`-ähnlichen Befehl erstellt wurde (`wxplot2d` und `wxplot3d` werden hier nicht unterstützt) und die Gnuplot-Datei nicht allzu lang ist, bietet `wxMaxima` einen Rechts-Klick-Menüpunkt an, der das Diagramm in einem interaktiven Gnuplot-Fenster öffnet.

### 2.3.5 Öffnen der Gnuplot Kommandozeile in plot Fenstern

Wenn unter Windows die Variable `gnuplot_command` auf "wgnuplot" geändert wird, erlaubt `gnuplot`, eine Kommandokonsole zu öffnen, stiehlt aber jedes Mal, wenn ein Diagramm gezeichnet wird, den Tastaturfokus für kurze Zeit (die Zeichen, die währenddessen eingegeben werden, verschwinden).

### 2.3.6 Einbetten von Animationen in das Arbeitsblatt

Es ist meist schwer, aus 3D-Diagrammen quantitative Aussagen zu entnehmen. Eine Alternative ist es, den 3. Parameter auf das Mauselement zu legen. Das Kommando `with_slider_draw` ist eine Version von `wxdraw2d`, die mehrere Plots erstellt und es erlaubt, mittels eines Schiebers zwischen diesen hin- und herzuschalten. `wxMaxima` kann die Animation auch als animierte `.gif`-Datei exportieren.

Die ersten beiden Argumente von `with_slider_draw` sind der Name der Variable des zu variierenden Parameters und die Liste der Werte, die dieser annehmen soll. Darauf folgen die ganz normalen Argumente, die `wxdraw2d` akzeptiert:

```
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
);

```

Für 3D-Diagramme ist dieselbe Funktionalität als `with_slider_draw3d` verfügbar, die auch rotierende 3D-Diagramme erstellen kann:

```

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

```

Wenn es nur um die generelle Form der Kurve geht, reicht es oft, das Bild ein wenig zu bewegen, so, dass es von der Intuition erfasst werden kann:

```

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

```

Wer plot draw vorzieht, dem steht ein zweiter Satz an Funktionen zur Verfügung:

- `with_slider` und
- `wxanimate`.

Die Standard-Bildwiderholrate für Animationen kann im Konfigurations-Dialog von wx-Maxima eingestellt werden. Um die Geschwindigkeit einer einzelnen Animation zu ändern, kann die Variable `wxanimate_framerate` geändert werden:

```

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

```

Die Animations-Funktionen haben eine Eigentümlichkeit, die daher rührt, dass sie make-list verwenden: Der Parameter, der sich von Bild zu Bild ändert, wird nur eingesetzt, wenn die Variable direkt sichtbar ist. Das folgende Beispiel scheitert daher:

```
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)
)$
```

Wenn Maxima explizit gebeten wird, den Wert einzusetzen, funktioniert der Befehl stattdessen:

```
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 Mehrere Diagramme gleichzeitig in Fenstern öffnen

Strenggenommen kein Feature von wxMaxima. Aber Maxima erlaubt auf vielen Systemen das folgende Beispiel von Mario Rodriguez auszuführen:

```
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));
```

Mehrere Diagramme im gleichen Fenster plotten ist auch möglich:

```
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 Die "Plotte mittels Draw"-Seitenleiste

Die "Plotte mittels Draw"-Seitenleiste enthält einen Codegenerator, der es erlaubt, einen Teil der Flexibilität des draw-Pakets von Maxima zu nutzen.

#### 2.3.8.1 2D

Generiert einen draw()-Befehl, der mittels der anderen Knöpfe der Seitenleiste mit einer 2D-Szene gefüllt werden kann.

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

Wie "2D", aber für dreidimensionale Diagramme.

#### 2.3.8.3 Ausdruck

Fügt den aktuellen draw()-Befehl den Plot einer Kurve wie  $\sin(x)$ ,  $x*\sin(x)$  oder  $x^2+2*x-4$  hinzu. Besteht noch kein draw()-Befehl, wird automatisch eine 2D-Szene erzeugt.

#### 2.3.8.4 Impliziter Plot

Markiert alle Punkte, an denen eine Bedingung wie  $y=\sin(x)$ ,  $y*\sin(x)=3$  oder  $x^2+y^2=4$  erfüllt ist und zeichnet diese Kurve in das aktuelle Diagramm ein. Gibt es kein aktuelles Diagramm, wird ein 2D-Diagramm erzeugt.

#### 2.3.8.5 Parametrische Plots

Bewegt eine Variable von einem Start- zu einem Endwert und verwendet getrennte Ausdrücke wie  $t*\sin(t)$  und  $t*\cos(t)$ , um die x-, die y- (und in 3D-Diagrammen auch die z-) Koordinaten zu generieren.

#### 2.3.8.6 Punkte

Zeichnet eine Reihe von Punkten, die optional miteinander verbunden werden. Die Koordinaten der Punkte können aus einer Liste von Listen, einem 2-dimensionalen Array oder einer Liste oder einem Array pro Axe entnommen werden.

#### 2.3.8.7 Diagrammtitel

Bestimmt den Titel des Diagramms.

#### 2.3.8.8 Achsen

Die Einstellungen für die Achsen.

#### 2.3.8.9 Höhenlinien

(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 Name der Kurve

Fügt einen Eintrag zur Legende hinzu, der für die nächsten Objekte gilt. Ein leerer Name bedeutet, dass die nun folgenden Objekte keinen eigenen Eintrag erhalten.

#### 2.3.8.11 Linienfarbe

Setzt die Linienfarbe für die die nun folgenden Plots des aktuellen `draw`-Befehls.

#### 2.3.8.12 Füllfarbe

Setzt die Füllfarbe für die nun folgenden Objekte des aktuellen `draw`-Kommandos.

#### 2.3.8.13 Gitter

Ein Assistent, der die Gitterlinien einzustellen hilft.

#### 2.3.8.14 Genauigkeit

Erlaubt das Wählen zwischen Geschwindigkeit und Genauigkeit bei der Erstellung der folgenden Kurven.

### 2.4 Einbetten von Bildern

Das `.wxmx`-Dateiformat erlaubt es, Bilder per Drag-And-Drop einzubetten. Es ist aber auch möglich, Maxima anzuweisen, Bilder einzubauen:

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

### 2.5 Startbefehle

Der Konfigurationsdialog von wxMaxima bietet an, zwei Dateien mit Maxima-Befehlen zu bearbeiten:

- Eine Datei mit Befehlen, die Maxima beim Hochfahren ausführt: `maxima-init.mac`
- Eine Datei mit zusätzlichen Kommandos, die ausgeführt werden, wenn Maxima von wxMaxima gestartet wird: `wxmaxima-init.mac`

Diese Dateien liegen im Benutzerverzeichnis von Maxima, normalerweise im Ordner `.maxima` im Home-Verzeichnis. Der genaue Ort kann über das Kommando `maxima_userdir` ermittelt werden.

## 2.6 Spezielle Variablen, die wxMaxima definiert

- `wxsubscripts` informiert Maxima, welche Variablen mit Unterstrich (z.B. `R_150`) in Variablen mit Subscript konvertieren soll. Details werden unter `wxdeclare_subscript` beschrieben.
- `wxfilename`: Diese Variable sagt, welchen Dateinamen das Arbeitsblatt laut wxMaxima besitzt.
- `wxplot_pngcairo` sagt Maxima, ob es versuchen soll, `gnuplot` zu instruieren, `pngcairo` zu nutzen, um hochqualitative Bilder zu nutzen.
- `wxplot_size` legt die Größe eingebetteter Diagramme fest.
- `wxchangedir`: Normalerweise setzt wxMaxima Maxima's Arbeitsverzeichnis auf dasjenige, in dem sich das Arbeitsblatt befindet. Dies ist praktisch, wenn Maxima auf Dateien zugreifen soll (z.B. via `read_matrix`). Unter Windows geht dieses Feature manchmal schief und kann daher auf `false` gesetzt werden.
- `wxanimate_framerate`: Die Geschwindigkeit, mit der Animationen abgespielt werden sollen.
- `wxanimate_autoplay`: Sollen Animationen automatisch abgespielt werden?

## 2.7 2D-Tabellen sauber ausgeben

Die Funktion `table_form()` konvertiert 2D-Listen in eine lesbarere Tabellenform als Maxima selbst. Die Eingabe ist eine Liste von einer oder mehreren Listen. Wie der "print" Befehl wird die Ausgabe auch gemacht, wenn der Befehl durch ein Dollarzeichen abgeschlossen wurde. Das abschließende `done` kann durch Verwendung eines `$` anstelle eines `;` unterdrückt werden.

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

Das folgende Beispiel zeigt das Zusammensetzen der Listen für solch eine Tabelle.

Da Matrizen effektiv Listen von Listen sind, können auch sie in Tabellen verwandelt werden.

## 2.8 Fehler melden

wxMaxima bietet einige Funktionen für das Melden von Fehlern an:

- `wxbuild_info()` sammelt Informationen über die derzeit ausgeführte Version von wxMaxima

```
(%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]

	<i>1st value</i>	<i>2nd value</i>	<i>3rd value</i>
(titleList)	1	2	3
	1	4	9

Figure 10: Ein drittes Tabellen-Beispiel

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

(M) 
$$\begin{bmatrix} 1st\ value & 2nd\ value & 3rd\ value \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{bmatrix}$$

	<i>1st value</i>	<i>2nd value</i>	<i>3rd value</i>
(%t17)	1	2	3
	1	4	9

Figure 11: Ein anderes table\_form Beispiel

- `wxbug_report()` sagt, wo und wie Fehler gemeldet werden können

## 2.9 Rotes Markieren von Formelteilen

Der `box()`-Befehl stellt sein Argument in wxMaxima rot dar.

---

# 3 Fehlersuche

## 3.1 Keine Verbindung zu Maxima möglich

Maxima und wxMaxima kommunizieren über ein lokales Netzwerk miteinander. Dies kann durch eine übereifrige Firewall unterbunden werden, die auch Verbindungen innerhalb des Rechners verbietet. Manchmal ist es schwer, herauszufinden, welches Programm erlaubt sein soll, da Maxima manchmal unter Namen wie `sbcl`, `gcl`, `ccl`, `lisp.exe` oder ähnlichen firmiert.

Auf Unix/Linux-Rechnern kann ein weiterer Grund sein, dass das "loopback"-Netzwerkgerät nicht korrekt eingestellt ist.

## 3.2 Wie repariere ich kaputte .wxmx-Dateien?

Die meisten modernen XML-basierten Formate sind von ihrem Inhalt her einfache .zip-Dateien. wxMaxima schaltet bei ihnen die Kompression nicht ein, weswegen ihr Inhalt in einem normalen Texteditor lesbar ist.

Wenn die Zip-Signatur am Ende der Datei intakt ist, kann man die Datei nach `<irgendwas>.zip` umbenennen, entpacken, reparieren, neu packen und wieder nach `<irgendwas>.wxmx` umbenennen. Dies ist auch eine praktische Möglichkeit, Word oder Powerpoint die originalen Bilddateien zu entlocken. Ist dies nicht der Fall, ist dies nicht das Ende der Welt: Oft gibt es eine funktionierende `wxmx~`-Datei.

Und auch wenn diese Datei nicht existiert: Eine `.wxmx`-Datei ist ein Containerformat, und der XML-Anteil wird unkomprimiert gespeichert. Es ist möglich, die `.wxmx`-Datei in eine `.txt`-Datei umzubenennen und mit einem Texteditor den XML-Anteil der Datei (beginnt mit `<?xml version="1.0" encoding="UTF-8" ?>` und endet mit `</wxMaximaDocument>` zu sichern. Vor und nach diesem Text wird unlesbarer (binärer) Inhalt im Texteditor sichtbar sein).

Wird eine Textdatei mit diesem Text (z.B. indem er mit Copy+Paste in eine neue Datei eingefügt wird) als `.xml`-Datei gespeichert, weiß wxMaxima, wie man den Text-Teil des Dokuments rekonstruiert.

### 3.3 Ich will Statusmeldungen am Bildschirm ausgeben, während mein Befehl ausgeführt wird

Normalerweise gibt wxMaxima erst etwas aus, wenn die komplette Ausgabe steht. Das disp-Kommando wird hingegen sofort ausgeführt:

```
for i:1 thru 10 do (  
  disp(i),  
  /* (sleep n) ist eine Lisp-Funktion, die mit einem */  
  /* "?" Zeichen vorher verwendet werden kann. */  
  /* Sie verzögert die Programmausführung für n Sekunden */  
  /* (in diesem Beispiel: für 3 Sekunden). */  
  ?sleep(3)  
)$
```

### 3.4 Statt eines Diagramms wird ein Briefumschlag mit einer Fehlermeldung dargestellt

wxMaxima konnte die Datei, die Maxima gnuplot instruiert hat, zu generieren, nicht lesen.

Mögliche Gründe für diesen Fehler sind:

- `implicit_plot` wurde verwendet, ohne vorher via `load()` geladen zu werden.
- Maxima hat versucht, Gnuplot zu etwas zu bewegen, was die installierte Version von Gnuplot nicht versteht. In diesem Fall stehen die betreffenden Befehle in dem Verzeichnis, auf das `maxima_userdir` zeigt in einer Datei mit der Endung `.gnuplot`. Der Inhalt dieser Datei ist oft hilfreich, um den Fehler zu suchen.
- Gnuplot wurde über wxMaxima's Konfigurationsdialog instruiert, `pngcairo` zu verwenden, unterstützt diese Ausgabefunktionen aber nicht. Dies ist auf dem Mac ein typisches Problem.
- Gnuplot hat keine gültige `.png`-Datei ausgegeben.

### 3.5 Animationen enden in "Error: Undefined Variable"

Der Wert für den Parameter, der bei der Animation variiert werden soll, war für Maxima nicht sichtbar. Ein explizites `subst()` behebt dieses Problem, wie in einem Beispiel weiter oben gezeigt wird.

### 3.6 Undo erinnert sich nicht an das, was ich brauche

Es gibt zwei Rückgängigmach- Funktionen, die beide die wichtige Information enthalten können:

- Ein Rückgängigmach-Speicher für die aktuell ausgewählte Zelle und einer für das gesamte Arbeitsblatt, der aktiv ist, wenn keine Zelle ausgewählt ist.
- Wenn die Ausgabe einer Marke (z.B. `%o20`) zugewiesen wurde, kann diese Marke eingegeben werden und der betreffende Text erscheint.

- Wenn nicht: Keine Panik: Im “Ansicht”-Menü kann eine Seitenleiste eingeschaltet werden, die die letzten Befehle anzeigt.
- Wenn nichts anderes funktioniert, bietet Maxima die Möglichkeit an, alle bisherigen Befehle nochmals auszuführen:

```
playback();
```

### 3.7 wxMaxima meldet gleich beim Hochfahren “Maxima hat sich beendet.”

Ein möglicher Grund ist, dass Maxima nicht dort gefunden werden kann, wo dies in wxMaxima’s Konfigurationsdialog angegeben ist. Korrektur des dort angegebenen Pfades zum Programm löst dieses Problem.

### 3.8 Maxima hört nicht auf zu rechnen und reagiert nicht auf Eingaben

Theoretisch kann es passieren, daß wxMaxima nicht erkennt, dass Maxima mit der Berechnung fertig ist und daher nie neue Befehle an Maxima sendet. In diesem Fall kann der Befehl ‘Trigger Evaluation’ die Synchronisation wieder herstellen.

### 3.9 SBCL beschwert sich über einen Mangel an Speicher

Der Lisp Compiler SBCL reserviert gleich beim Hochfahren allen Speicher, den es im Betrieb nutzen will. Wird mehr Speicher benötigt, muss ihm über den Kommandozeilenparameter `--dynamic-space-size` gesagt werden, wie viel Speicher benötigt wird. SBCL kann nur in etwa die Hälfte des freien Speichers des Systems verwenden. Auf 32-Bit-Rechnern ist dies meist unter 999 Megabytes. Auf 64-Bit-Systemen können mehr als 1280 MB verwendet werden, was notwendig ist um das Lapack-Paket zu compilieren.

Kommandozeilenparameter für Maxima (und daher für SBCL) können in wxMaxima’s Konfigurationsdialog eingegeben werden.

### 3.10 Ubuntu: Die Tastatur ist langsam oder ignoriert einzelne Tasten

Das Installieren von `ibus-gtk` behebt dieses Problem meist. Auf (<https://bugs.launchpad.net/ubuntu/+source/wxwidgets3.0/+bug/1421558>) findet man genauere Angaben dazu.

### 3.11 wxMaxima stoppt, wenn Maxima griechische Buchstaben oder Umlaute verarbeitet

Wenn Maxima mittels SBCL compiliert wurde, können die folgenden Befehle zur `.sbclrc` hinzugefügt werden:

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

Wo diese Datei abgelegt werden muss, ist systemabhängig. Aber ein mit SBCL compiliertes Maxima kann durch den folgenden Befehl verleitet werden, den Ort zu nennen:

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

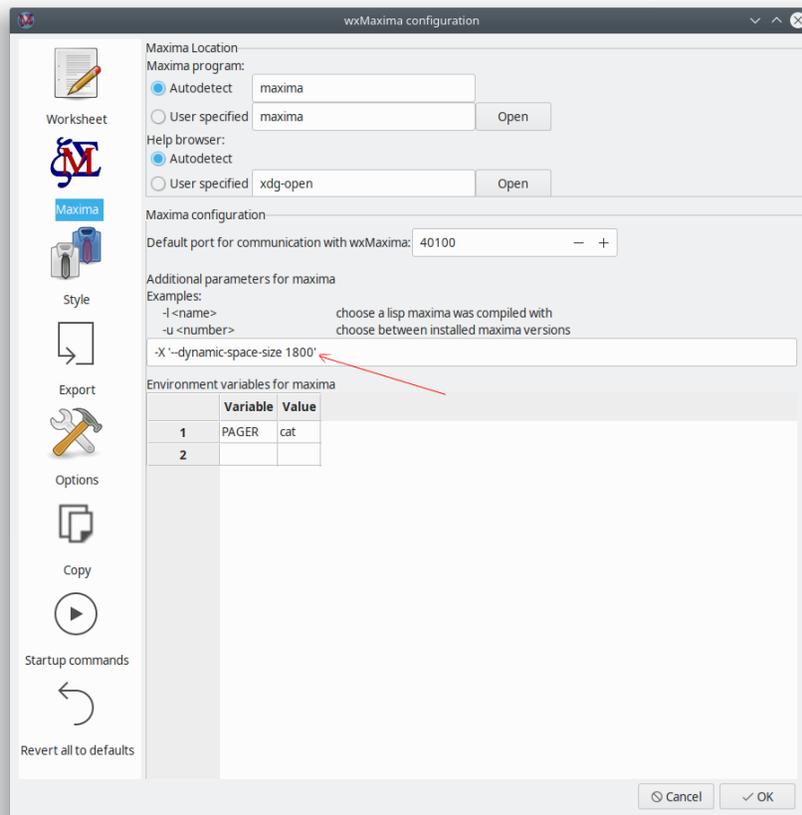


Figure 12: SBCL Speicherkonfiguration

## 3.12 Diagramme

### 3.12.1 Kann wxMaxima das eingebettete Diagramm gleich noch als Datei ausgeben?

Das Arbeitsblatt enthält png-Dateien. wxMaxima erlaubt dem User anzugeben, wo sie generiert werden sollen:

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

Wenn ein unterschiedliches Format benutzt wird, ist es einfacher, Bilder zu generieren und dann in das Arbeitsblatt zu importieren:

```
load("draw");  
pngdraw(name,[contents]):=  
(  
    draw(  
        append(  
            [  
                terminal=pngcairo,  
                dimensions=wxplot_size,  
                file_name=name  
            ],  
            contents  
        )  
    ),  
    show_image(sprintf(false,"~a.png",name))  
);  
pngdraw2d(name,[contents]):=  
    pngdraw(name,gr2d(contents));  
  
pngdraw2d("Test",  
    explicit(sin(x),x,1,10)  
);
```

### 3.12.2 Kann ich das Seitenverhältnis des Plots angeben?

Nicht direkt bei Maxima. Aber es gibt Gnuplot-Kommandos dafür:

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

---

## 4 Häufig gestellte Fragen

### 4.1 Gibt es eine Möglichkeit mehr Text auf eine LaTeX-Seite zu schreiben?

Ja. Verwenden Sie das LaTeX Paket “geometry” um die Größe der Ränder anzugeben.

Ja, gibt es. Geben Sie einfach die folgenden Zeilen im LaTeX-Vorspann (z.B. indem sie das entsprechende Feld im Konfigurationsdialog (“Exportieren”->“Zusätzliche Zeilen für die LaTeX preamble”) angeben.

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

### 4.2 Gibt es einen Dark Mode?

Wenn wxWidgets neu genug ist, sollte wxMaxima sich automatisch im Dark Mode befinden, wenn der Rest des Betriebssystems es ist. Das Arbeitsblatt selbst besitzt standardmäßig einen hellen Hintergrund. Dies kann aber über die Konfiguration geändert werden. Als Alternative erlaubt Ansicht/Invertiere die Helligkeit des Arbeitsblattes, das Arbeitsblatt schnell zwischen hell und dunkel umzustellen.

### 4.3 wxMaxima hängt manchmal in der ersten Minute einmal für mehrere Sekunden

wxMaxima lagert große Aufgaben, wie das Interpretieren des >1000-Seiten-Handbuchs von Maxima an Hintergrundtasks aus. Solange die Ergebnisse dieser Tasks nicht benötigt werden, kann während dieser Zeit ganz normal weitergearbeitet werden. Wird aber eine Aktion ausgeführt, für die die Ergebnisse eines Tasks benötigt werden, muss wxMaxima warten, bis dieser beendet ist.

---

## 5 Kommandozeilen-Optionen

Die meisten Betriebssysteme bieten unkompliziertere Arten, um ein GUI-Programm zu starten, als die Kommandozeile. wxMaxima bietet trotzdem mehrere Kommandozeilenoptionen an.

- `-v` oder `--version`: Gibt die Versionsnummer aus
- `-h` oder `--help`: Gibt einen kurzen Hilfetext aus
- `-o` oder `--open=<str>`: Öffnet den Dateinamen, der als Argument dieser Kommandozeilenoption angegeben wurde.
- `-e` oder `--eval`: Evaluiere die Datei nach dem Öffnen.
- `-b` oder `--batch`: Nach dem Öffnen werden alle Code-Zellen der Datei(en) evaluiert und die Datei danach geschlossen. Dies ist beispielsweise praktisch, wenn Maxima Dateien generiert. Wenn Maxima Fehler detektiert, wird dieser Vorgang abgebrochen, eventuelle Fragen an den User (Mathematik ist an einigen Stellen ein interaktiver Prozess) werden wie üblich an den User weitergereicht.
- `--logtostdout`: Gebe alle all Debug-Nachrichten auch auf die Konsole aus.

- `--pipe`: Gebe die Nachrichten von Maxima auch auf die Konsole aus.
- `--exit-on-error`: Schließe wxMaxima, wenn Maxima einen Fehler detektiert.
- `-f` oder `--ini=<str>`: Speichere die Konfiguration in der Datei `<str>`
- `-u`, `--use-version=<str>`: Verwende Maxima-Version `<str>`
- `-l`, `--lisp=<str>`: Verwende ein Maxima, das mit dem Lisp Compiler `<str>` kompiliert wurde.
- `-X`, `--extra-args=<str>`: Erlaubt, Maxima zusätzliche Argumente mitzugeben
- `-m` oder `--maxima=<str>`: Sagt, wo die ausführbare Datei von maxima liegt.
- `--enableipc`: Lässt Maxima wxMaxima über Interprozesskommunikation ansteuern. Diese Option vorsichtig verwenden.

Manche Betriebssysteme verwenden bei Kommandozeilenparametern einen Schrägstrich statt eines Minuszeichens.