HÖHERE LEHRANSTALT FÜR ELEKTROTECHNIK
INFORMATIONSTECHNIK, ENERGIETECHNIK & INDUSTRIELLE ELEKTRONIK
A-3101 St. Pölten, Waldstrasse 3, Tel: +43(0)2742-75051-300, Fax -230
Homepage: http://et.htlstp.ac.at E-Mail: et@htlstp.ac.at

Grafiken mit MAXIMA

(ab Version 5.23)

Wilhelm Haager
HTL St. Pölten, Abteilung Elektrotechnik
wilhelm.haager@htlstp.ac.at

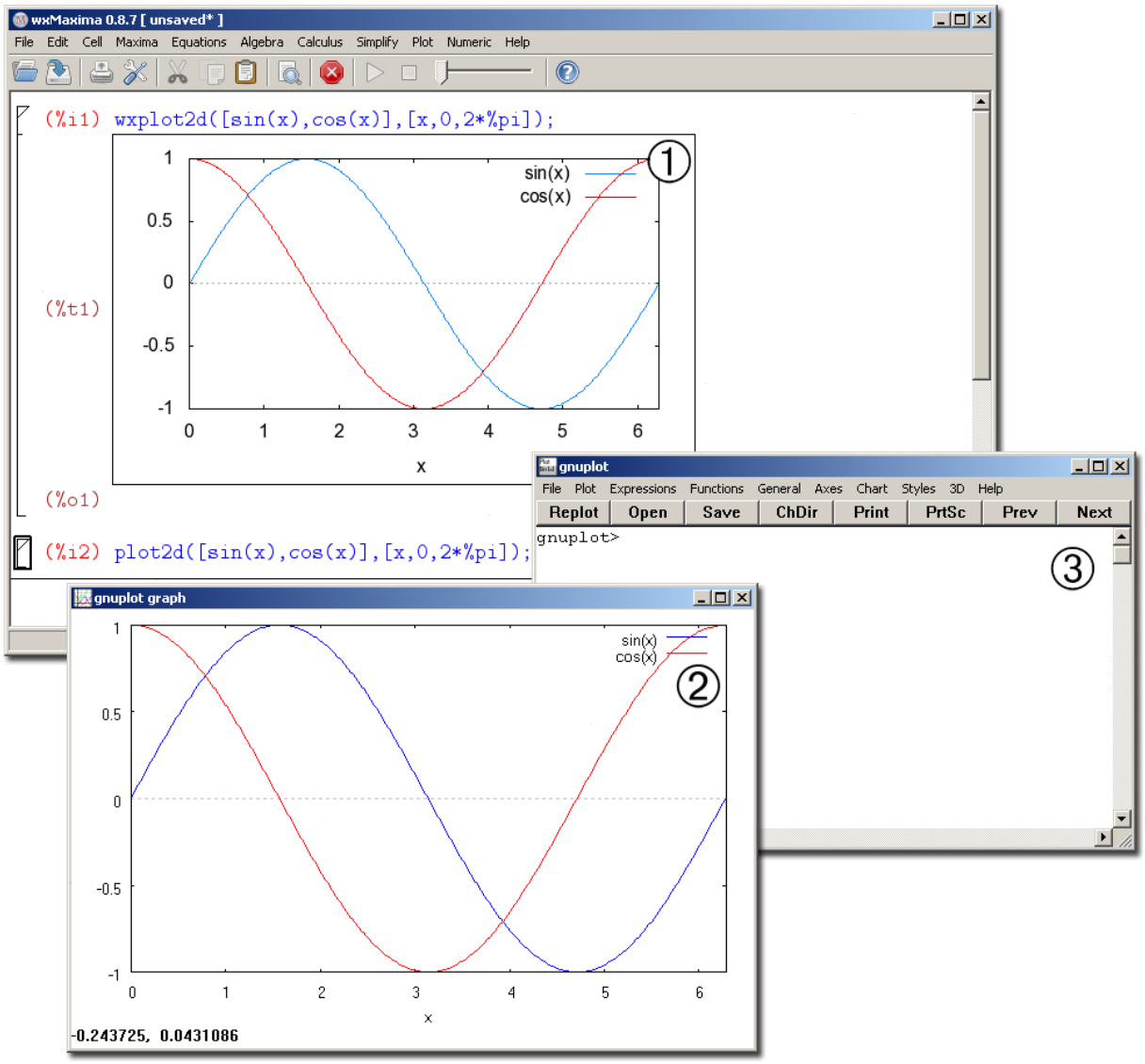
26. November 2011

Inhaltsverzeichnis

1 Grundlagen

Zu Darstellung von Grafiken verwendet Maxima das Programm *Gnuplot* [2], das bei der Erstellung der Grafik aufgerufen wird. Dabei gibt es zwei verschiedene Methoden des Aufrufes:

1. Bei Aufruf der Standard-Plotroutinen `plot2d`, `plot3d`, `draw2d`, `draw3d`, etc. wird ein Gnuplot-Fenster geöffnet, in das die Grafik gezeichnet wird. Erst nach Schließen dieses Fensters kann mit Maxima weitergearbeitet werden. Die Größe dieses Fensters und somit die Größe



① . . . Grafik im wxMaxima-Arbeitsfenster
② . . . Grafik im Gnuplot-Ausgabefenster
③ . . . Gnuplot-Konsole

der Grafik kann mit der Maus verändert werden. Bei 2d-Plots können mit der Maus Koordinaten gemessen werden, bei 3d-Plots kann mit der Maus die Grafik in beliebige Richtung gedreht werden.

Ein Rechtsklick auf den Fenstertitel öffnet ein Kontextmenü, mit dem die Grafik ausgedruckt und die Gnuplot-Konsole geöffnet werden kann. Mit der Gnuplot-Konsole könnte Gnuplot als eigenständiges Plotprogramm – auch unabhängig von Maxima – verwendet werden.

2. Durch Voranstellen der Buchstaben „`wx`“ an die Namen der Plotroutinen (`wxplot2d`, `wxplot3d`, `wxdraw2d`, `wxdraw3d`, `...`) werden die Grafiken von Gnuplot im Hintergrund im PNG-Format in Bildschirmauflösung erstellt und direkt in das wxMaxima-Arbeitsfenster gezeichnet. Diese Methode zur Erzeugung von Grafiken ist bequem und vorteilhaft für interaktives Arbeiten, da die Grafiken während der gesamten Maxima-Sitzung im Arbeitsfenster erhalten bleiben. Durch Anklicken mit der Maus kann die Grafik in die Zwischenablage kopiert oder als File abgespeichert werden, auf Grund ihrer geringen Auflösung ist aber eine Weiterverwendung nicht sinnvoll.

Zwei verschiedene Gnuplot-Interfaces stehen zur Verfügung, die Standardfunktionen von Maxima mit dem Wortstamm `plot` in den Funktionsnamen sowie die Funktionen des Zusatzpaketes *Draw* mit dem Wortstamm `draw` in den Funktionsnamen.

Die Plotroutinen des Zusatzpaketes *Draw* sind zwar etwas komplizierter in der Anwendung, bieten aber wesentlich mehr Möglichkeiten, die Grafiken mit Hilfe von Optionen an die eigenen Wünsche anzupassen. Außerdem ist es bei ihnen möglich, Ausgabeformat (eps, png, jpg, etc.) und Ausgabeziel (den Filenamen) in Gnuplot festzulegen *nachdem* die Grafik erstellt wurde, was mit den Standardfunktionen des Gnuplot-Interfaces *Plot* offenbar nicht möglich ist.

2 Gnuplot

Gnuplot ist ein befehlszeilenorientiertes Plotprogramm. Beim Aufruf durch Maxima ist die Eingabe von Befehlen in der Gnuplot-Konsole im Allgemeinen nicht erforderlich, kann aber, insbesondere bei Verwendung des Gnuplot-Interfaces *Draw*, sehr hilfreich sein. Daher werden im Folgenden die Grundprinzipien von Gnuplot und die (wenigen) wichtigsten Befehle erläutert.

2.1 Befehle

`plot`*f(x) opts*	Plotten der Funktion *f(x)* mit den Optionen *opts*; grundlegender Befehl zur Erstellung von Grafiken, wird aber im Zusammenhang mit Maxima nicht benötigt.
`replot`	Erneutes Zeichnen einer Grafik, gegebenenfalls mit geänderten Einstellungen.
`set terminal` *term opts*	Festlegen des Ausgabeformates, gegebenenfalls mit Optionen `opts`.
`set output "`*filename*`"`	Festlegen des Ausgabezieles (Filename mit der passenden Erweiterung).
`set size` *xscale,yscale*	Skalierung des Diagrammes im Verhältnis zur Größe der gesamten Grafik.
`set size ratio` *n*	Festlegen des Ansichtsverhältnisses Höhe/Breite des Diagrammes.
`show` *item*	Anzeigen der aktuellen Werte für die Einstellungen für *item*.
`cd "`*directory*`"`	Wechsel des Arbeitsverzeichnisses für die Grafikausgabe
`pwd`	Anzeigen des Arbeitsverzeichnisses
`load "`*filename*`"`	Laden von Gnuplot-Befehlen aus einem File *filename*.
`set/unset key`	Ein-/Ausschalten der Legende
`set/unset hidden3d`	Ein-/Ausschalten der Anzeige verdeckter Linien bei 3d-Plots
`set/unset grid`	Ein-/Ausschalten eines Koordinatengitters
`set cntrparam` *spez*	Steuerung der Isolinien in einem Contour-Plot
`set view` *rotx rotz scale scaley*	Festlegen der Ansicht von 3d-Plots
`quit`	Beenden von Gnuplot; wird im Zusammenhang mit Maxima nicht benötigt.

Wichtige Gnuplot-Befehle

Gnuplot-Befehle werden zeilenweise in der Gnuplot-Konsole eingegeben, bei Eingabe mehrerer Befehle in einer Zeile sind diese durch einen Strichpunkt zu trennen. Befehle und Parameter sind durch Leerzeichen zu trennen, einzelne Koordinaten durch Beistriche. Koordinaten*bereiche* sind in der Form $[x_1 : x_2]$ anzugeben.

Der grundlegende Plotbefehl `plot` wird bei Erstellung der Grafik von Maxima automatisch aufgerufen, eine Eingabe des Befehls durch den Benutzer in der Gnuplot-Konsole ist nicht sinnvoll.

Bei Verwendung des Gnuplot-Interfaces *Draw* kann eine bereits von Maxima erstellte Grafik mit dem Befehl `replot` nochmals gezeichnet werden, gegebenenfalls mit geänderten Parametern, die mit dem Befehl `set` eingestellt und mit dem Befehl `show` angezeigt werden können.

Mit `set terminal` (abgekürzt `set term`) wird das Fileformat für die erstellte Grafik festgelegt. Eine Vielzahl von Formaten ist möglich [2], die wichtigsten sind im Abschnitt 2.2 zusammengefasst.

Mit `set output` wird das Ausgabeziel, der Name des Grafikfiles, festgelegt. Wird nicht der vollständige Pfadname als Parameter angegeben, sondern nur der Filename, so wird das File im augenblicklichen Arbeitsverzeichnis (gesetzt mit `cd`, angezeigt mit `pwd`) abgespeichert. Mit dem Filenamen `stdout` wird die Grafik (unabhängig vom eingestellten Gnuplot-Terminal) im Gnuplot-Ausgabefenster ausgegeben.

Der Befehl `set output` immer sollte *nach* einem eventuellen Befehl `set terminal` erfolgen.

Der Befehl `set size` legt nicht die Größe der Grafik selbst fest, sondern die Diagrammgröße innerhalb der Grafik im Verhältnis zur Standardgröße. Bei Werten größer als 1 werden Teile des Diagrammes abgeschnitten. Mit `set size ratio` wird das Ansichtsverhältnis Höhe/Breite des Diagrammes festgelegt, ein negativer Wert bewirkt nicht ein Verhältnis der Achsen*längen*, sondern der Achsen*skalierungen*. So bewirkt beispielsweise `set size ratio -1`, dass x- und y-Achse gleich skaliert werden (und somit beispielsweise ein Kreis unverzerrt ein Kreis bleibt).

Mit dem Befehl `set cntrparam` können die Anzahl und die Werte der Isolinien bei Contour-Plots auf unterschiedlichste Weise angegeben werden:

`set cntrparam levels` *n*	...	Automatische Berechnung von *n* Isolinien
`set cntrparam discrete` *z1,z2,...*	...	Angabe der Werte der Isolinien
`set cntrparam incremental` *z1,dz,z2*	...	Isolinien von *z1* bis *z2* im Abstand von *dz*

Der Befehl `set view` *rotx rotz scale scalez* legt die Ansicht von 3d-Plots fest. Dabei ist *rotx* der Drehwinkel um die x-Achse (von der senkrechten Ansicht weggerechnet), *rotz* der Drehwinkel um die z-Achse, *scale* ein globaler Skalierungsfaktor und *scalez* ein zusätzlicher Skalierungsfaktor für die z-Achse.

Mit dem Befehl `load` kann eine Batchverarbeitung von Gnuplot-Befehlen, die in einem (editierbaren) File abgespeichert sind, durchgeführt werden.

2.2 Gnuplot Terminals

Die meisten Gnuplot-Terminals besitzen zusätzliche Optionen, mit denen die Eigenschaften der Grafik in Bezug auf Auflösung, Bildgröße, Farbtiefe, Schriftart und Schriftgröße eingestellt werden können. Anzahl und Format dieser Optionen sind aber leider nicht einheitlich für alle Terminals gleich, im Einzelfall ist das Gnuplot-Manual [2] zu konsultieren.

Pixel-Grafikformate:	
`dumb`	ASCII-Art Grafik im Retrolook
`gif`	GIF-Grafik (geeignet für Webseiten), 8 Bit Farbtiefe
`png`	Portable Network Graphics, als Ersatz für GIF entwickelt, 24 Bit Farbtiefe
`jpeg`	JPG-Grafik
`latex`	LaTeX-Code mit der *Picture*-Umgebung
`windows`	zur direkten Bildschirmanzeige im Gnuplot-Ausgabefenster
Vektor-Grafikformate:	
`aifm`	Adobe-Illustrator Format (Dateierweiterung `.ai`)
`dxf`	Zeichnungs-Austauschformat von AutoCAD
`eepic`	LaTeX-Code in der erweiterten *Picture*-Umgebung (mit dem Paket `eepic`)
`hpgl`	Hewlett-Packard Graphics Language für Stiftplotter und CNC-Fräsmaschinen
`postscript`	Postscript-Grafik (Dateierweiterung `.ps`)
`postscript eps`	Encapsulated Postscript-Grafik (Dateierweiterung `.eps`)
`svg`	Scalable Vector Graphics

Wichtige Gnuplot-Terminals

2.3 Initialisierung

`gnuplot.ini`	Initialisierungsfile für die Erstellung von Grafiken
`wgnuplot.ini`	Initialisierungsfile für das Erscheinungsbild von Gnuplot
`GNUPLOT`	Umgebungsvariable, enthält den Pfad für die Initialisierungsfiles.

Gnuplot-Initialisierung

Gewünschte Standardeinstellungen für die Erstellung von Grafiken, beispielsweise das Gnuplot-Arbeitsverzeichnis, können in einem File mit dem Namen `gnuplot.ini` festgelegt werden. Dieses File kann beliebige Gnuplot-Befehle enthalten, die beim Starten von Gnuplot ausgeführt werden.

Standardeinstellungen für das Erscheinungsbild von Gnuplot selbst, beispielsweise Textgröße und Fenstergröße, sind im File `wgnuplot.ini` festgelegt. Dieses File wird aber vom Benutzer nicht direkt erstellt, sondern über das Kontext-Menü der Gnuplot-Konsole.

Die Verzeichnisse, in denen sich die Initialisierungsfiles `gnuplot.ini` und `wgnuplot.ini` befinden, sind in der Umgebungsvariable `GNUPLOT` festzulegen.

3 Grafikinterface Plot

Plot ist das Standard-Gnuplot-Interface von Maxima, relativ bequem in der Anwendung, aber nicht sehr flexibel in den Gestaltungsmöglichkeiten für Grafiken; außerdem sind die Möglichkeiten zur Erstellung von 3d-Grafiken sehr eingeschränkt. Leider ist es bei Verwendung dieses Interfaces *nicht* möglich, nachträgliche Änderungen in bezug auf graphische Ausgestaltung, Ausgabeformat und Ausgabeziel in der Gnuplot-Konsole vorzunehmen.

3.1 Plotbefehle

`plot2d(`*f(x)*`,`*xrange*`,`*opts*`)`	Plotten der Funktion *f(x)*, mit den Optionen *opts* im Bereich *xrange*; der Bereich *xrange muss* hier angegeben werden.
`plot2d([discrete,`*xwerte*`,`*ywerte*`],`*opts*`)`	Plotten der Punkte mit der Angabe der x-Werte und y-Werte in zwei getrennten Listen *xwerte* und *ywerte*; die Angabe eines Bereiches für die x-Werte ist hier optional.
`plot2d([discrete,`*werte*`],`*opts*`)`	Andere Möglichkeit zum Plotten von Punkten; die Punkte sind hier in einer doppelt geschachtelten Liste `[[`*x1,y1*`],[`*x2,y2*`],...]` angegeben.
`plot2d([parametric,`*x(t)*`,`*y(t)*`,`*trange*`],`*opts*`)`	Plotten einer Kurve in Parameterdarstellung mit dem Parameter *t* im Parameterbereich *trange*.
`plot3d(`*f(x,y)*`,`*xrange*`,`*yrange*`,`*opts*`)`	Plotten der Funktion *f(x,y)*, mit den Optionen *opts* im Bereich *xrange* und *yrange*.
`plot3d([`*x(u,v)*`,`*y(u,v)*`,`*z(u,v)*`],`*urange*`,`*vrange*`,`*opts*`)`	Plotten eines dreidimensionalen Objektes in Parameterdarstellung mit den Parametern *u* und *v* in den jeweiligen Bereichen *urange* und *vrange*.
`contour_plot(`*f(x,y)*`,`*xrange*`,`*yrange*`,`*opts*`)`	Plotten der Isolinien einer Funktion *f(x,y)* mit den Optionen *opts* im Bereich *xrange* und *yrange*.
[*x,x1,x2*]	Angabe eines Bereiches für die Variable *x* mit der Untergrenze *x1* und der Obergrenze *x2*.

Plotbefehle des Gnuplot-Interfaces *Plot*

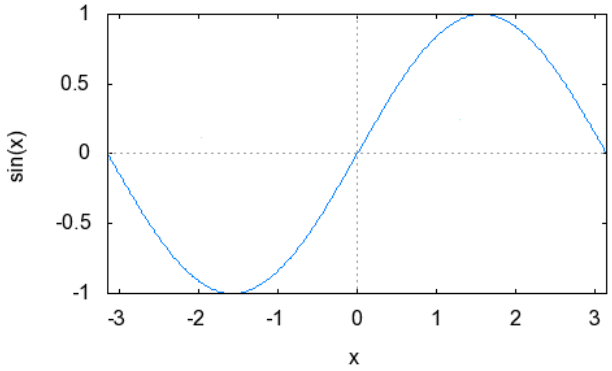
Der Befehl `(wx)plot2d` stellt einen Kurvenverlauf oder einen durch Punkte vorgegebenen Linienzug in einem zweidimensionalen kartesischen Koordinatensystem dar. Dafür gibt es drei Möglichkeiten, die beliebig in einem einzigen Diagramm kombiniert dargestellt werden können:

1. Eine Funktion $y = f(x)$; der Darstellungsbereich auf der x-Achse muss in der Form *[x,x1,x2]* angegeben werden, die Angabe eines Bereiches auf der y-Achse ist optional.
2. Eine Kurve in Parameterdarstellung $x(t), y(t)$ in Abhängigkeit eines (beliebig wählbaren) Parameters t. Hat dieser Parameter tatsächlich den Namen „t“, so kann die Angabe eines Bereiches entfallen, er wird dann entsprechend dem in den Plot-Optionen eingestellten Wert (für alle parametrischen Kurven gleich) vorgegeben (siehe Abschnitt 3.2).
3. Einzelne Punkte, die gegebenenfalls (bei entsprechenden Optionen) durch einen Linienzug verbunden werden. Für die Angabe der Punkte gibt es zwei Möglichkeiten: entweder als zwei Listen mit jeweils den x-Werten und den y-Werten, oder aus einer einzigen (zweifach geschachtelten) Liste mit den einzelnen Punkten, die jeweils aus einer Liste mit dem x-Wert und dem y-Wert bestehen.

Plotten einer Funktion in einem vorgegebenen Intervall; der Bereich für die y-Achse wird automatisch ermittelt.

```
(%i1) wxplot2d(sin(x),[x,-%pi,%pi])$
```

(%t1)



Vorgabe von x-Werten in einer Liste:

```
(%i2) xwerte:[0,3,6,4,6,3,0,2,0];
(%o2) [0,3,6,4,6,3,0,2,0]
```

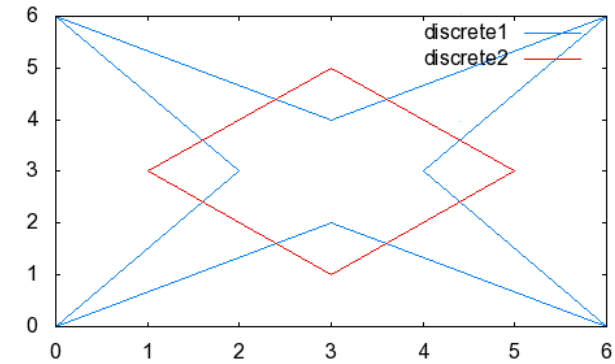
Vorgabe von y-Werten in einer Liste:

```
(%i3) ywerte:[0,2,0,3,6,4,6,3,0];
(%o3) [0,2,0,3,6,4,6,3,0]
```

Zwei punktweise Plots in einem Diagramm mit beiden Methoden der Punktevorgabe. Standardmäßig werden die Punkte durch Liniensegmente verbunden.

```
(%i4) wxplot2d([[discrete,xwerte,ywerte],
[discrete,[[3,1],[5,3],[3,5],[1,3],[3,1]]]])$
```

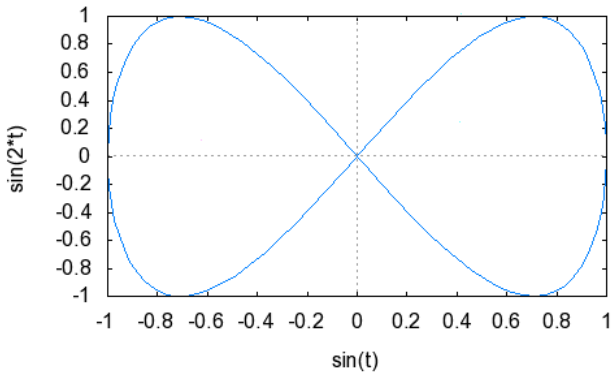
(%t4)



Plotten einer Kurve in Parameterdarstellung; um eine glatte Kurve zu erhalten, muss die Anzahl der Stützstellen mit der Option `nticks` erhöht werden.

```
(%i5)  wxplot2d([parametric,sin(t),sin(2*t),
       [t,0,2*%pi]],[nticks,100])$
```

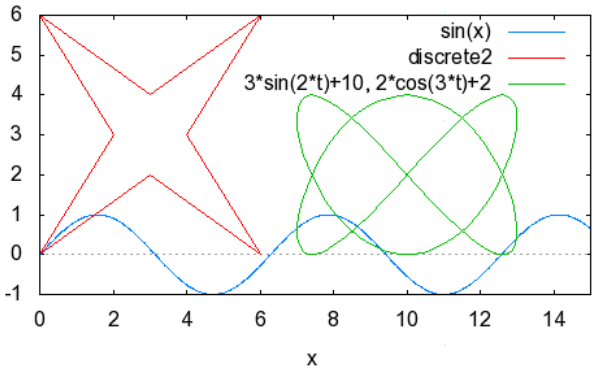
(%t5)



Kombination von drei unterschiedlichen 2d-Plots in einem einzigen Diagramm; die drei Kurven sind in einer Liste anzugeben.

```
(%i6)  wxplot2d([sin(x),[discrete,xwerte,ywerte],
       [parametric,10+3*sin(2*t),2+2*cos(3*t),
       [t,0,2*%pi]]],[x,0,15],[nticks,100])$
```

(%t6)



In ein einziges Diagramm können beliebig viele Kurvenverläufe, zu einer Liste zusammengefasst, gezeichnet werden.

Die Möglichkeiten zum Erstellen von 3d-Grafiken mit dem Gnuplot-Interface *Plot* sind sehr beschränkt, nur der Vollständigkeit halber sind sie hier aufgeführt. Empfehlenswert für 3d-Grafiken ist jedenfalls die Verwendung des Gnuplot-Interfaces *Draw*!

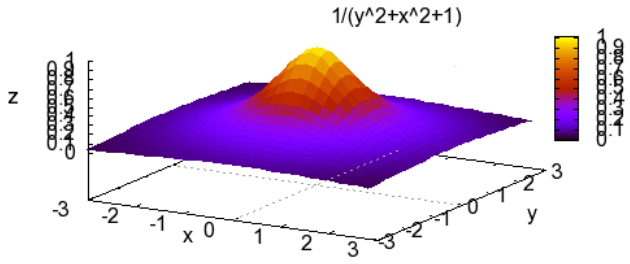
Für 3d-Grafiken gibt es zwei Möglichkeiten, wobei in ein Diagramm aber nur ein einziges Grafikobjekt gezeichnet werden kann:

1. Eine Funktion $z = f(x, y)$; die Darstellungsbereiche auf der x-Achse und y-Achse sind anzugeben, ein z-Bereich kann aber nicht (!) angegeben werden.
2. Ein Objekt in Parameterdarstellung $x(u, v), y(u, v), z(u, v)$ in Abhängigkeit zweier (beliebig wählbarer) Parameter u und v. Die Bereiche für u und v sind anzugeben, die Bereiche für x, y und z können nicht angegeben werden.

3d-Plot einer Funktion in zwei Variablen; die Bereichsangaben für x und y sind optional, ihre Defaultwerte sind in der Liste `plot_options` gespeichert.

```
(%i7)  wxplot3d(1/(1+x^2+y^2),[x,-3,3],[y,-3,3])$
```

(%t7)



Parameterdarstellung einer Fläche im dreidimensionalen Raum

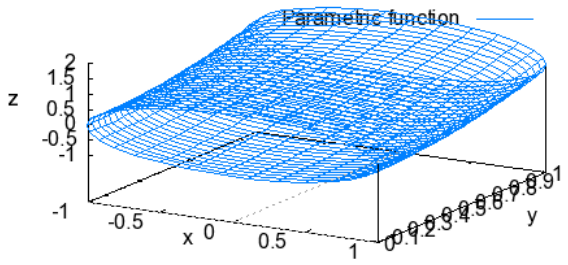
```
(%i8)  [fx:cos(x),fy:y,fz:sin(x)+y^2];
```

(%o8) $[\cos(x), y, y^2+\sin(x)]$

3d-Plot der Fläche in Parameterdarstellung

```
(%i9)  wxplot3d([fx,fy,fz],[x,0,2*%pi],[y,0,1],
       [palette,false])$
```

(%t9)

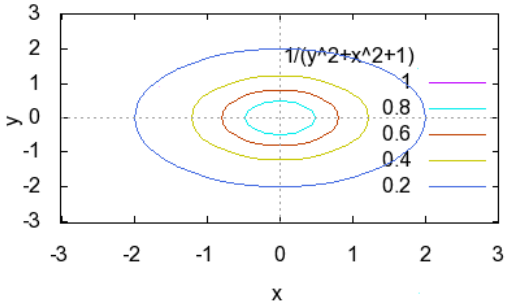


`contour_plot` zeichnet Isolinien einer Funktion in zwei Variablen innerhalb des Bereiche `xrange` und `yrange`, Anzahl und Werte der Isolinien können über die Option `gnuplot_preamble` festgelegt werden (Abschnitte 2.1 und 3.2).

Isolinien einer Funktion in zwei Variablen. Sinnvoll wäre hier eine explizite Vorgabe der Funktionswerte für die Linien, sowie das Ausschalten der Legende mit entsprechenden Plot-Optionen.

```
(%i10) wxcontour_plot(1/(1+x^2+y^2),
       [x,-3,3],[y,-3,3])$
```

(%t10)



3.2 Optionen

Mit zusätzlichen optionalen Parametern der Plotfunktionen kann die Gestalt der Grafiken in Hinblick auf Farben, Linientypen, Diagrammgrößen, Beschriftung, Ausgabeformate etc. den Bedürfnissen angepasst werden. Die Optionen bestehen aus Listen (meist mit zwei Elementen); das erste Element ist immer der Optionsname, die weiteren Elemente sind die zugehörigen Werte. Die Plot-Optionen können in jedem Plot-Befehl als zusätzliche Parameter angegeben werden, sie können mit dem Befehl `set_plot_option` aber als Defaultwerte auch global festgelegt werden und gelten dann für alle nachfolgenden Plotbefehle. Mit dem Befehl `plot_options` werden die eingestellten Defaultwerte aller Plot-Optionen angezeigt.

Befehle:	
`plot_options`	Ausgabe aller Plot-Optionen
`set_plot_option([`*name*`,`*v*`])`	Setzen der Plot-Option *name* auf den Wert *v*
Wichtige Optionen:	
`[y,` *ymin*`,` *ymax*`]`	Skalierung der y-Achse
`[x,` *xmin*`,` *xmax*`]`	Skalierung der x-Achse, Pflichtparameter bei 2d-Plots, Option bei 3d-Plots
`[nticks,`*n*`]`	Anzahl der Stützstellen zur Kurvenberechnung (Default:10)
`[adapt_depth,` *n*`]`	Maximale Zahl der Aufteilung der Kurvenabschnitte zwischen zwei Stützstellen (Default:10)
`[gnuplot_preamble, "`*text*`"]`	Gnuplot-Vorspann, enthält beliebige Gnuplot-Befehle, die *vor* dem Plot ausgeführt werden.
`[xlabel,"`*text*`"]`	Beschriftung der x-Achse bei 2d-Plots
`[ylabel,"`*text*`"]`	Beschriftung der y-Achse bei 2d-Plots
`[logx]/[logy]`	Logarithmische Skalierung der x-/y-Achse bei 2d-Plots
`[legend,"`*text1*`","`*text2*`",...]`	Legenden für die einzelnen Kurven in einem 2d-Plot
`[style,`*style1*`,`*style2*`,...]`	Plotstile für die einzelnen Kurven in einem 2d-Plot in der Form `[`*name*`,`*w*`,`*c*`]`.
`[gnuplot_term,` *terminal*`]`	Ausgabe*format* (Gnuplot-Terminal); ignoriert bei den wx-Routinen
`[gnuplot_out_file, "`*filename*`"]`	Ausgabe*ziel*; ignoriert bei den wx-Routinen
`[grid,`*nx*`,`*ny*`]`	Anzahl der Gitterlinien bei 3d-Plots in x- und y-Richtung
`[gnuplot_pm3d, true/false]`	Kontrolliert färbiges Ausfüllen der Flächen bei 3d-Plots

Plot-Optionen für das Gnuplot-Interface *Plot*

Die Optionen `x` und `y` legen die angezeigten Koordinatenbereiche fest, bei 2d-Plots ist `x` kein optionaler Parameter, sondern ein Pflichtparameter.

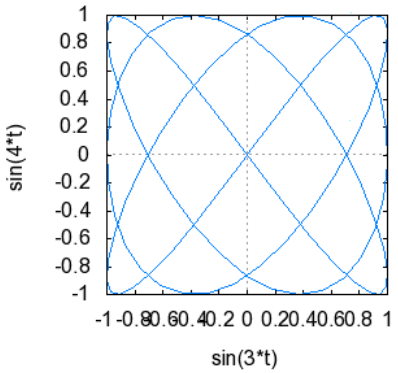
`nticks` und `adapt_depth` legen die Anzahl und maximale Teilung von Stützstellen für die Kurvenberechnung fest. Eine explizite Angabe ist nur dann notwendig, wenn die dargestellten Kurven nicht hinreichend „glatt“ erscheinen.

Die Option `gnuplot_preamble` kann als Zeichenkette beliebige Gnuplot-Befehle, durch Strichpunkte getrennt, enthalten, die vor dem eigentlichen Plotbefehl ausgeführt werden. Damit können Einstellungen an Gnuplot weitergereicht werden, für die es in Maxima keine eigenen Plot-Optionen gibt (etwa für die Steuerung der Isolinien bei Contour-Plots oder für die Einstellung des Ansichtsverhältnisses).

Lissajous-Figur; mit der Option *gnuplot_preamble* werden gleiche Skalierungen für beide Achsen erzwungen, mit `nticks` wird die Punktezahl entsprechend erhöht.

```
(%i11) wxplot2d([parametric,sin(3*t),sin(4*t),
       [t,0,2*%pi]],[nticks,100],
       [gnuplot_preamble,"set size ratio -1"])$
```

(%t11)



Die Option `style` legt Kurventyp, Linienstärke und Linienfarbe (und gegebenenfalls den Punktetyp) fest. Die Linienstärke ist bei Pixelgrafiken in Pixel angegeben, bei Vektorgrafiken in Vielfachen von 0.25pt (etwa 0.088mm). Für den Kurventyp sind folgende Werte möglich:

- `lines` ... Linienzug
- `points` ... Punkte (mit zusätzlicher Angabe des Punktetyps als Ganzzahl)
- `linespoints` ... Linienzug und Punkte
- `impulses` ... Balken (Angaben für Linienbreite und Farbe werden ignoriert)

Für die Farben gelten folgende Ganzzahlwerte:

1 ... blau 2 ... rot
3 ... magenta 4 ... gelb
5 ... braun 6 ... grün
7 ... cyan

Liste mit x-Werten

```
(%i12) lx:[1,2,3,4,5,6,7,8,9];
(%o12) [1,2,3,4,5,6,7,8,9]
```

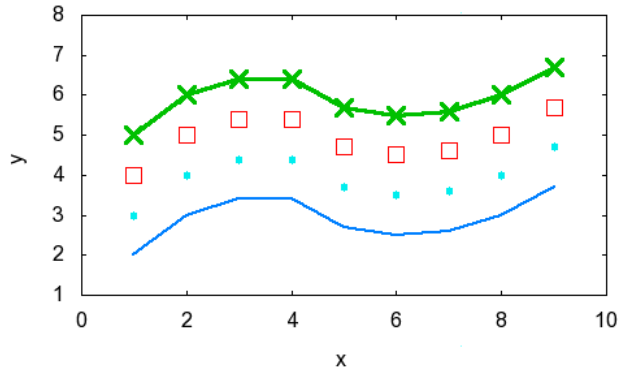
Liste mit y-Werten

```
(%i13) ly:[1,2,2.4,2.4,1.7,1.5,1.6,2,2.7];
(%o13) [1,2,2.4,2.4,1.7,1.5,1.6,2,2.7]
```

Plotten von Punkten mit verschiedenen Plotstilen: als Linienzug, als Punkte (in verschiedenen Größen und Formen), sowie als Kombination von beiden.

```
(%i14) wxplot2d(makelist([discrete,lx,c+ly],c,1,4),
[x,0,10],[y,1,8],[legend,""],
[style,[lines,2,1],[points,2,6,1],
[points,4,2,7],[linespoints,3,4,3]])$
```

(%t14)



(Etwas trickreiche) Erzeugung einer Liste von Punkt-Plots mit jeweils einem einzigen Punkt

```
(%i15) plotlist:apply(append,makelist(makelist
([discrete,[[nx,ny]]],nx,1,13),ny,1,7))$
```

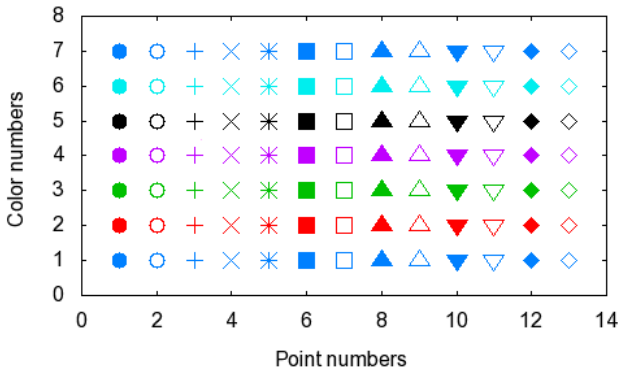
(Etwas trickreiche) Erzeugung einer Liste von Plotstilen mit allen Farben und allen Punktformen

```
(%i16) stylelist:cons(style,apply(append,makelist(
makelist([points,4,ny,nx],nx,1,13),ny,1,7)))$
```

Plotten aller möglichen Farben und Punktformen. Mit Hilfe entsprechender Optionen wird die Legende unterdrückt und eine Beschriftung der Achsen veranlasst.

```
(%i17) wxplot2d(plotlist,stylelist,[x,0,14],[y,0,8],
[legend,""],[xlabel,"Point numbers"],
[ylabel,"Color numbers"])$
```

(%t17)



`gnuplot_term` legt das Ausgabe*format* fest, wobei neben den Werten `default` und `ps` (für encapsulated postscript) sämtliche Gnuplot-Terminals (Abschnitt 2.2) möglich sind. Die Option `gnuplot_out_file` legt das Ausgabe*ziel* fest, bei fehlender Pfadangabe wird aber (unverständlichwerweise) *nicht* das Gnuplot-Arbeitsverzeichnis verwendet, sondern das Verzeichnis entsprechend der Betriebssystem-Umgebungsvariable `HOMEPATH`. Die Optionen `gnuplot_term` und `gnuplot_out_file` werden von den wx-Plotroutinen ignoriert.

Für 3d-Plots gibt es kaum direkte Einstellungsmöglichkeiten mittels Plot-Optionen. Mit Hilfe der Option `gnuplot_preamble` können Einstellungen aber auf Gnuplot-Ebene vorgenommen werden:

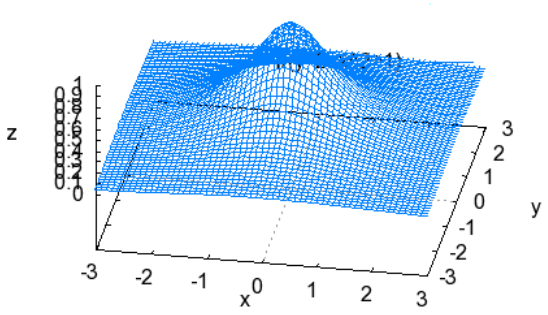
Erzeugung eines Gnuplot-Vorspannes für 3d-Plots zwecks Anzeige verdeckter Linien und zum Festlegen einer bestimmten Ansicht

```
(%i18) p:"unset hidden3d;set view 20,10,1,3";
(%o18) unset hidden3d;set view 20,10,1,3
```

3d-Plot mit einem Gnuplot-Vorspann, geänderter Anzahl an Gitterlinien und ausgeschalteter Farbpalette

```
(%i19) wxplot3d(1/(1+x^2+y^2),[x,-3,3],[y,-3,3],
       [grid,60,60],[gnuplot_preamble, p],
       [palette,false])$
```

(%t19)



4 Grafikinterface Draw

Das Gnuplot-Interface *Draw* von *Mario Rodríguez Riotorto* [3] stellt einen weiteren Satz von Maxima-Grafikroutinen zur Verfügung, die sich in den Parametern, insbesondere in der Stuktur der Optionen, von den Standardroutinen unterscheiden. Die Anwendung dieser Routinen, alle mit dem Wortstamm `draw` in den Befehlsnamen, ist zwar etwas komplizierter als die der Standardroutinen, sie sind aber wesentlich flexibler in den Gestaltungsmöglichkeiten für Grafiken. Außerdem ist es mit ihnen möglich, nachträgliche Änderungen in bezug auf Ausgabeformat und Ausgabeziel in der Gnuplot-Konsole vorzunehmen.

Vor ihrer Verwendung muss das Paket *Draw* geladen werden.

4.1 Plotbefehle

`draw(`*scene1*`,`*scene2*`,...,`*opts*`,...)`	Erstellen einer Grafik als Zusammenstellung von *Szenen* und globalen Optionen *opts*.
`gr2d(`*opts*`,`*graphic_object*`,...)`	Erstellen einer 2d-*Szene* als Zusammenstellung von beliebigen 2d-Grafikobjekten mit den Optionen *opts*.
`gr3d(`*opts*`,`*graphic_object*`,...)`	Erstellen einer 3d-*Szene* als Zusammenstellung von beliebigen 3d-Grafikobjekten mit den Optionen *opts*.
`draw2d(`*opts*`,`*graphic_object*`,...)`	Erstellen einer 2d-Grafik als Zusammenstellung von beliebigen 2d-Grafikobjekten mit den Optionen *opts*.
`draw3d(`*opts*`,`*graphic_object*`,...)`	Erstellen einer 3d-Grafik als Zusammenstellung von beliebigen 3d-Grafikobjekten mit den Optionen *opts*.
Optionen:	
name1=value1`,`*name2=value2*`,...`	

Plotbefehle des Gnuplot-Interfaces *Draw*

Eine Grafik besteht aus einzelnen (2d- und 3d-) *Szenen*, die mit dem grundlegenden Plotbefehl `draw` in einer rechteckigen Anordnung ausgedruckt werden; jede Szene entspricht dabei einem einzelnen Diagramm.

Eine Szene wird mit den Befehlen `gr2d` (für den zweidimensionalen Fall) und `gr3d` (für den dreidimensionalen Fall) aus einzelnen *Grafikobjekten* und gegebenenfalls zusätzlichen *Optionen* zusammengesetzt.

In den meisten Fällen enthält eine Grafik nur ein einziges Diagramm (d. h. nur eine einzige Szene). In diesem Fall bewirken die Plotbefehle `draw2d` und `draw3d` die Zusammenstellung der Grafikobjekte zu einer Szene und gleichzeitig das Erstellen der Grafik aus dieser Szene. Die Befehle

`draw2d(...)` und `draw3d(...)`

sind also gleichbedeutend mit

`draw(gr2d(...))` und `draw(gr3d(...))`.

Die Parameterliste enthält beliebige Grafikobjekte und Optionen. Die zu einem Grafikobjekt gehörenden Optionen sind diesen *voranzustellen*. Die Position *globaler* Parameter (zum Beispiel zum Festlegen von Ausgabeformat und Ausgabeziel) ist beliebig.

Optionen haben die Form von Gleichungen mit der linken Seite als Name und der rechten Seite als Wert, wobei der Wert auch eine Liste (zum Beispiel bei Bereichsangaben) sein kann.

Bei komplexeren Grafiken mit vielen Objekten kann die Parameterliste der Plotbefehle sehr lang und unübersichtlich werden. Es ist daher guter Stil, entweder

- die Eingabe diesem Falle durch Zeilenvorschübe und Einrückungen entsprechend zu strukturieren, oder (noch besser)
- die Grafikobjekte nicht direkt in der Parameterliste der Plotbefehle zu formulieren, sondern ihnen in eigenständigen Anweisungen Variablennamen zuzuweisen und diese Variablen in die Parameterliste einzusetzen.

4.2 2d-Grafikobjekte

`explicit`, `parametric` und `implicit` erzeugen Grafikobjekte aus mathematischen Ausdrücken in kartesischen Koordinaten. `polar` erzeugt ein Grafikobjekt einer Funktion in Polarkoordinaten. Da die darzustellenden Bereiche fixer Bestandteil der Grafikobjekte sind, müssen sie im Gegensatz zum Gnuplot-Interface *Plot* nicht im Stil von Optionen angegeben werden.

Zusätzlich steht eine Vielzahl an geometrischen Formen zur Verfügung, mit denen beliebige Grafiken erstellt werden können. `points` dient prinzipiell zur Darstellung diskreter Punkte; diese Punkte können durch Setzen entsprechender Optionen durch Geradenstücke verbunden werden. Somit lassen sich mit dem Grafikopjekt `points` auch beliebige Linienzüge und angenäherte Kurven darstellen (siehe Abschnitt 4.4).

`explicit(`*f(x)*`,`*x*`,`*x1*`,`*x2*`)`	Funktion *f(x)* zwischen den Werten *x1* und *x2*
`parametric(`*x(t)*`,`*y(t)*`,`*t*`,`*t1*`,`*t2*`)`	Kurve *x(t),y(t)* in Parameterdarstellung mit dem Parameter *t* zwischen den Werten *t1* und *t2*
`implicit(`*equation*`,`*x*`,`*x1*`,`*x2*`,`*y*`,`*y1*`,`*y2*`)`	Implizite Kurve, durch die Gleichung *equation* angegeben, in den Variablen *x* und *y* innerhalb der Bereiche *x1*...*x2* und *y1*...*y2*
`polar(`*r(φ)*`,`*φ*`,`*φ1*`,`*φ2*`)`	Funktion in Polarkoordinaten; Radius *r* in Abhängigkeit des Winkels *φ* in Grad innerhalb des Bereiches *φ1*...*φ2*
`points(`*xwerte*`,`*ywerte*`)`	Punkte; *xwerte* und *ywerte* sind Listen mit den x- bzw. y-Werten
`points(`*p1*`,`*p2*`,...)`	Punkte; jeder Punkt *pi* ist eine Liste aus den Koordinatenwerten: [*px,py*].
`polygon(`*xwerte*`,`*ywerte*`)`	Polygon mit Angabe der Eckpunkte in den Listen *xwerte* und *ywerte*.
`polygon(`*p1*`,`*p2*`,...)`	Polygon; jeder Eckpunkt *pi* ist eine Liste aus den Koordinatenwerten: [*px,py*].
`rectangle(`*p1*`,`*p2*`)`	Rechteck mit den diametralen Eckpunkten *p1* und *p2* in der Form [*px,py*].
`ellipse(`*x0*`,`*y0*`,`*a*`,`*b*`,`*w1*`,`*w2*`)`	Ellipse (oder Kreis) mit dem Mittelpunkt [*x0,y0*], den Halbachsen *a* und *b*, sowie dem Anfangs- und Endwinkel *w1* bzw. *w2*.
`label(["`*text*`",`*x*`,`*y*`],...)`	Schreibt ein Label *text* an die Position [*x,y*]; Bündigkeit und Orientierung können über Optionen festgelegt werden.
`vector([`*x*`,`*y*`],[`*dx*`,`*dy*`])`	Vektor mit dem Ursprung [*x,y*] und den Koordinaten [*dx,dy*]
`image(`*m*`,`*x*`,`*y*`,`*nx*`,`*ny*`)`	Bildobjekt mit der Matrix *m* aus *nx*×*nx* Bildpunkten und den Koordinaten [*x,y*] des linken unteren Bildpunktes

2d-Grafikobjekte

Laden des Paketes *Draw*

```
(%i20) load(draw);
(%o20) C:/Programme/Maxima-5.23.2/share/maxima/...
```

Definition einer Funktion $y=f(x)$ als Grafikobjekt

```
(%i21) g1:explicit(2*sin(x),x,-%pi,%pi);
(%o21) explicit(2 sin(x), x, −π, π)
```

Kurve in Parameterdarstellung als Grafikobjekt

```
(%i22) g2:parametric(2*sin(phi),2*cos(phi),phi,0,2*%pi);
(%o22) parametric(2 sin(φ), 2 cos(φ), φ, 0, 2 π)
```

Implizite Funktion als Grafikobjekt

```
(%i23) g3:implicit(x^2-y^2=1,x,-4,4,y,-4,4);
```

(%o23) implicit $\left(x^2-y^2=1,x,-4,4,y,-4,4\right)$

Funktion $r=f(\varphi)$ in Polarkoordinaten als Grafikobjekt

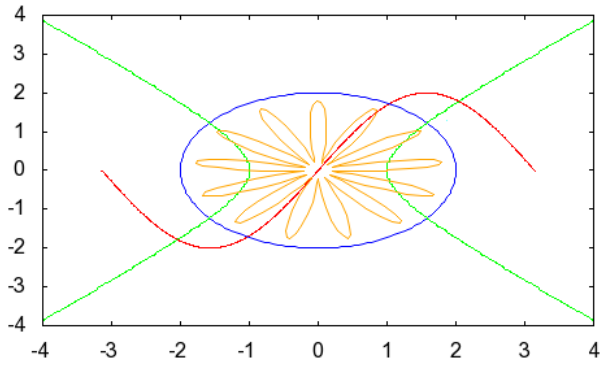
```
(%i24) g4:polar(1+0.8*sin(13*t),t,0,2*%pi);
```

(%o24) polar $(0.8\sin(13\,t)+1,t,0,2\,\pi)$

Plotten aller Grafikobjekte in ein einziges Diagramm; um glatte Kurven zu erhalten und dem Diagramm etwas „Farbe“ zu verleihen, wurden entsprechende Optionen angegeben.

```
(%i25) wxdraw2d(nticks=200,color=red,g1,color=blue,g2,
               color=green,g3,color=orange,g4)$
```

(%t25)



Erzeugung eines Polygons als Grafikobjekt aus jeweils einer Liste von x-Werten und y-Werten

```
(%i26) poly:polygon(xwerte+2,ywerte+2);
```

(%o26) polygon $([2,5,8,6,8,5,2,4,2],[2,4,2,5,8,6,8,5,2])$

Erzeugung eines Grafikobjekts „Punkte“ aus jeweils einer Liste von x-Werten und y-Werten

```
(%i27) punkte:points([1,3,5,7,9,9,9,9,9,7,5,3,1,1,1,1],
       [1,1,1,1,1,3,5,7,9,9,9,9,9,7,5,3]);
```

(%o27) points $([1,3,5,7,9,9,9,9,9,7,5,3,1,1,1,1],$
$[1,1,1,1,1,3,5,7,9,9,9,9,9,7,5,3])$

Festlegung eines Rechtecks mit zwei gegenüberliegenden Punkten

```
(%i28) rechteck:rectangle([1,-2],[6,-7]);
```

(%o28) rectangle $([1,-2],[6,-7])$

Ellipse als Grafikobjekt

```
(%i29) ell:ellipse(6,-6,3,2,0,360);
```

(%o29) ellipse $(6,-6,3,2,0,360)$

Balkendiagramm als Grafikobjekt

```
(%i30) balken:bars([-7,2,1],[-5,5,1],[-3,7,1],[-1,6,1]);
```

(%o30) bars $([-7,2,1],[-5,5,1],[-3,7,1],[-1,6,1])$

Erzeugung dreier Vektoren als Grafikobjekte

```
(%i31) [v1,v2,v3]:[vector([-8,-8],[6,0]),
       vector([-8,-8],[6,6]),vector([-2,-8],[0,6])];
```

(%o31) $[\text{vector}([-8,-8],[6,0]),\text{vector}([-8,-8],[6,6]),$
$\text{vector}([-2,-8],[0,6])]$

Diagrammbeschriftung als Grafikobjekt „Label“; alle Beschriftungen bilden hier ein einziges Grafikobjekt.

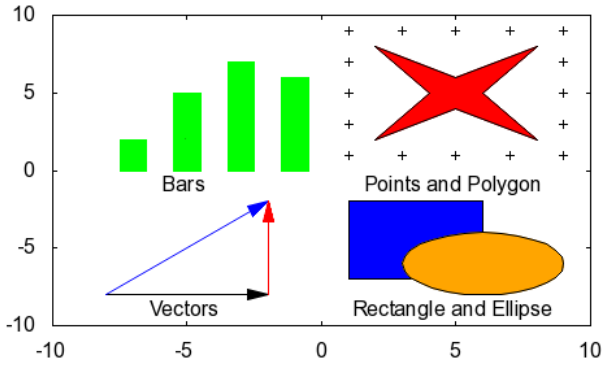
```
(%i32) text:label(["Bars",-5,-1],
                  ["Points and Polygon",5,-1],
                  ["Vectors",-5,-9],
                  ["Rectangle and Ellipse",5,-9]);
```

(%o32) label$([\texttt{Bars},-5,-1],[\texttt{Points and Polygon},5,-1],$
$[\texttt{Vectors},-5,-9],[\texttt{Rectangle and Ellipse},5,-9])$

Plotten aller Grafikobjekte in ein einziges Diagramm; zum Setzen von Farben und der Gestalt der Vektorspitzen wurden entsprechende Optionen angegeben.

```
(%i33) wxdraw2d(xrange=[-10,10],yrange=[-10,10],
                punkte,poly,
                fill_color=blue,rechteck,
                fill_color=orange,ell,
                fill_color=green,balken,
                head_length=0.8,head_angle=15,v1,
                color=blue,v2,color=red,v3,
                color=black,text)$
```

(%t33)



`image` erstellt ein Bildobjekt bestehend aus quadratischen farbigen Flächen, die den einzelnen Bildpunkten entsprechen. Die Farbwerte können in der Matrix *m* auf zwei Arten angegeben werden:

- Sind die Matrixelemente Listen aus drei Zahlen, so werden diese Zahlen als Rot-, Grün- und Blauwerte der entsprechenden Bildpunkte interpretiert. Dabei entsprechen diese Zahlen aber nicht *absoluten* Farbwerten, sondern *relativen* Farbwerten bezogen auf die größte Zahl, die der vollen Farbsättigung entspricht!
- Sind die Matrixelemente einzelne Zahlenwerte, so werden diese entsprechend der Farbpalette interpretiert, die mit der Option `palette` festgelegt wird.

Erzeugung einer Matrix mit Farbwerten entsprechend der Standard-Farbpalette

```
(%i34) m1:matrix([0,50,100],[50,100,150],[100,150,200]);
```

(%o34)
$$\begin{pmatrix} 0 & 50 & 100 \\ 50 & 100 & 150 \\ 100 & 150 & 200 \end{pmatrix}$$

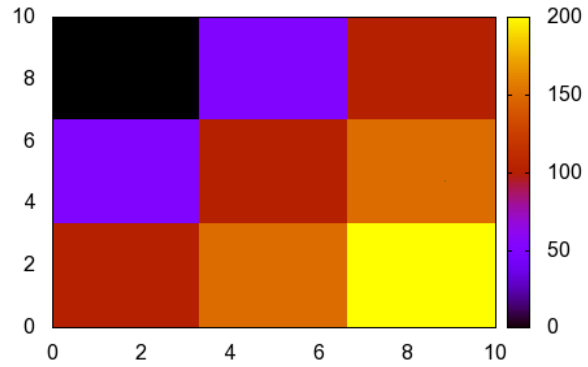
Erzeugung eines Bildobjektes

```
(%i35) im1:image(m1,0,0,10,10)$
```

Darstellung des Bildobjektes

```
(%i36) wxdraw2d(im1)$
```

(%t36)



Mit dem Zusatzpaket *Picture*, das automatisch mit dem Paket *Draw* geladen wird, erhält Maxima sehr beschränkte Möglichkeiten der Bildbearbeitung: Bildobjekte können aus XPM-Files geladen werden, Farbkanäle können aus Bildern extrahiert werden, Bilder können aus Farbkanälen zusammengesetzt werden.

4.3 3d-Grafikobjekte

`explicit(`*f(x,y)*`,`*x*`,`*x1*`,`*x2*`,`*y*`,`*y1*`,`*y2*`)`	Funktion *f(x,y)* in den Bereichen *x1*...*x2* und *y1*...*y2*
`implicit(`*equation*`,`*x*`,`*x1*`,`*x2*`,`*y*`,`*y1*`,`*y2*`,`*z*`,`*z1*`,`*z2*`)`	Implizite Kurve, durch die Gleichung *equation* angegeben, in den Variablen *x*, *y* und *z* innerhalb des Bereiches *x1*...*x2*, *y1*...*y2*, *z1*...*z2*
`parametric(`*x(t)*`,`*y(t)*`,`*z(t)*`,`*t*`,`*t1*`,`*t2*`)`	Kurve *x(t),y(t),z(t)* in Parameterdarstellung mit dem Parameter *t* zwischen den Werten *t1* und *t2*
`parametric_surface(`*x(u,v)*`,`*y(u,v)*`,`*z(u,v)*`,`*u*`,`*u1*`,`*u2*`,`*v*`,`*v1*`,`*v2*`)`	Parametrische Fläche *x(u,v),y(u,v),z(u,v)* mit den Parametern *u* und *v* in den Bereichen *u1*...*u2* und *v1*...*v2*
`cylindrical(`*r(z,φ)*`,`*z*`,`*z1*`,`*z2*`,`*φ*`,`*φ1*`,`*φ2*`)`	Fläche in Zylinderkoordinaten; Radius *r(z,φ)* in Abhängigkeit der z-Koordinate *z* innerhalb des Bereiches *z1*...*z2* und des Azimutwinkels *φ* innerhalb des Bereiches *φ1*...*φ2*
`spherical(`*r(φ,ϑ)*`,`*φ*`,`*φ1*`,`*φ2*`,`*ϑ*`,`*ϑ1*`,`*ϑ2*`)`	Fläche in Kugelkoordinaten; Radius *r(φ,ϑ)* in Abhängigkeit des Polarwinkels *ϑ* innerhalb des Bereiches *ϑ1*...*ϑ2* und des Azimutwinkels *φ* in Grad innerhalb des Bereiches *φ1*...*φ2*
`points(`*xwerte*`,`*ywerte*`,`*zwerte*`)`	Punkte; *xwerte*, *ywerte* und *zwerte* sind Listen mit den x-, y- bzw. z-Werten.
`points(`*p1*`,`*p2*`,...)`	Punkte; jeder Punkt *pi* ist eine Liste aus den Koordinatenwerten: [*px,py,pz*].
`label(["`*text*`",`*x*`,`*y*`,`*z*`],...)`	Schreibt ein Label *text* an die Position [*x,y,z*]; Bündigkeit, Schriftart etc. können über Optionen festgelegt werden.
`vector([`*x*`,`*y*`,`*z*`],[`*dx*`,`*dy*`,`*dz*`])`	Vektor mit dem Ursprung [*x,y,z*] und den Koordinaten [*dx,dy,dz*]

3d-Grafikobjekte

`explicit` und `implicit` erzeugen Grafikobjekte durch mathematische Funktionen in kartesi-

schen Koordinaten als Flächen im dreidimensionalen Raum. `parametric` erzeugt eine Kurve aus der Parameterdarstellung einer Funktion mit einem Parameter t, `parametric_surface` erzeugt eine Fläche aus der Parameterdarstellung einer Funktion mit zwei Parametern u und v. `cylindrical` und `spherical` erstellen eine Fläche aus einer Funktion in Zylinderkoordinaten bzw. Kugelkoordinaten.

Im Gegensatz zum Gnuplot-Interface *Plot* kann ein einziges Diagramm beliebig viele 3d-Grafikobjekte enthalten.

Erzeugung einer Kugel mit Hilfe einer impliziten Funktion

```
(%i37) gimp:implicit(1=x**2+y**2+z**2,
           x,-1,1,y,-1,1,z,-1,1);
```

(%o37) implicit $\left(1 = z^2 + y^2 + x^2, x, -1, 1, y, -1, 1, z, -1, 1\right)$

Zweidimensionale Funktion

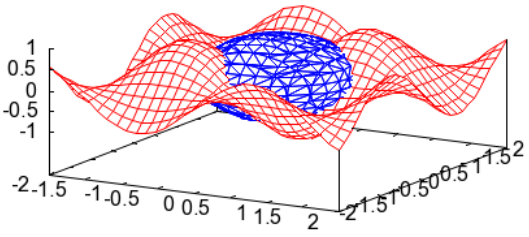
```
(%i38) gexp:explicit(sin(2*x)*sin(2*y),x,-2,2,y,-2,2);
```

(%o38) explicit $\left(\sin\left(2\,x\right)\,\sin\left(2\,y\right), x, -2, 2, y, -2, 2\right)$

Darstellung der Kugel und der zweidimensionalen Funktion in einem Diagramm. Zum Verdecken von unsichtbaren Gitterlinien und Umrissen wurde die Option `surface_hide` gesetzt.

```
(%i39) wxdraw3d(surface_hide=true,
           color=red,gexp,color=blue,gimp)$
```

(%t39)



Erzeugung einer toroidalen Spirale als parametrische Kurve im dreidimensionalen Raum mit vier Windungen um einen Torus.

```
(%i40) spiral:parametric((2-0.5*cos(t))*sin(t/4),
           (2-0.5*cos(t))*cos(t/4),
           0.5*sin(t),t,0,8*%pi);
```

(%o40) parametric $\left(\sin\left(\frac{t}{4}\right)\left(2-0.5\cos\left(t\right)\right),\right.$
$\left.\cos\left(\frac{t}{4}\right)\left(2-0.5\cos\left(t\right)\right),\right.$
$\left.0.5\sin\left(t\right), t, 0, 8\,\pi\right)$

Erzeugung eines Torus als 3d-Fläche in Parameterdarstellung.

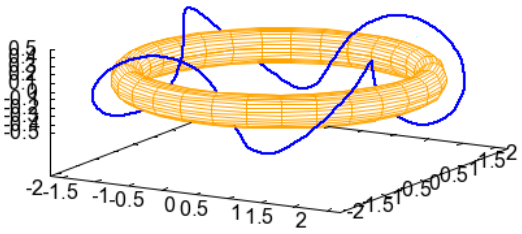
```
(%i41) torus:parametric_surface(
           (2-0.2*cos(phi))*sin(theta),
           (2-0.2*cos(phi))*cos(theta),
           0.2*sin(phi),phi,0,2*%pi,theta,0,2*%pi);
```

(%o41) parametric_surface $\left(\left(2-0.2\cos\left(\varphi\right)\right)\,\sin\left(\vartheta\right),\right.$
$\left.\left(2-0.2\cos\left(\varphi\right)\right)\,\cos\left(\vartheta\right), 0.2\sin\left(\varphi\right),\right.$
$\left.\varphi, 0, 2\,\pi, \vartheta, 0, 2\,\pi\right)$

Gemeinsame Darstellung von Torus und Spirale

```
(%i42) wxdraw3d(nticks=200,surface_hide=true,
                color=orange,torus,
                line_width=2,color=blue,spiral)$
```

(%t42)



Erzeugung eines Kegels mit einer Funktion in Zylinderkoordinaten

```
(%i43) cone:cylindrical(
          (z-15)*0.05,z,-15,15,phi,0,2*%pi);
```

(%o43) cylindrical $\left(0.05\,(z-15), z, -15, 15, \varphi, 0, 2\,\pi\right)$

Erzeugung einer schneckenhausartigen Fläche mit einer Funktion in Kugelkoordinaten

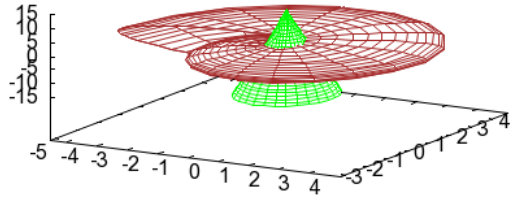
```
(%i44) snail:spherical(4+0.5*phi,phi,
                       -2*%pi,%pi,tht,0,%pi);
```

(%o44) spherical $\left(0.5\,\varphi+4, \varphi, -2\,\pi, \pi, tht, 0, \pi\right)$

Gemeinsame Darstellung von Kegel und Schneckenhaus

```
(%i45) wxdraw3d(surface_hide=true,color=green,cone,
                color=brown,snail)$
```

(%t45)



Erzeugung von 3d-Punkten in der xy-Ebene, in einem Kreis angeordnet

```
(%i46) pts:points(makelist(
       [sin(t*%pi/10),cos(t*%pi/10),0],t,1,20))$
```

Erzeugung von vier Vektoren, in Form einer Windrose angeordnet

```
(%i47) [v1,v2,v3,v4]:[vector([0,0,1],[0.7,0,0]),
       vector([0,0,1],[0,0.7,0]),vector([0,0,1],
       [-0.7,0,0]),vector([0,0,1],[0,-0.7,0])];
```

(%o47) [vector$([0,0,1],[0.7,0,0])$, vector$([0,0,1],[0,0.7,0])$, vector$([0,0,1],[-0.7,0,0])$, vector$([0,0,1],[0,-0.7,0])$]

Texte im dreidimensionalen Raum, der Text ist nicht auf *Achsen*, sondern auf die *Ansicht* ausgerichtet.

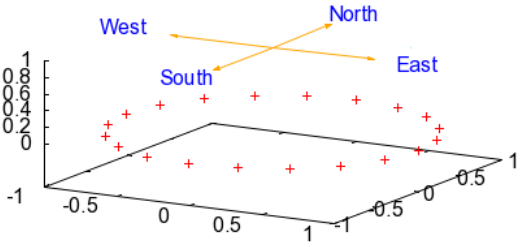
```
(%i48) text:label(["North",0,1,1],["East",1,0,1],
                  ["South",0,-1,1],["West",-1,0,1]);
```

(%o48) label$([North,0,1,1],[East,1,0,1],[South,0,-1,1],[West,-1,0,1])$

3d-Grafik mit Punkten, Vektoren und Texten

```
(%i49) wxdraw3d(color=red,pts,color=orange,
                v1,v2,v3,v4,color=blue,text)$
```

(%t49)



4.4 Allgemeine Optionen

`set_draw_defaults(`*opts*`,...)`	Setzen von Defaultwerten für Optionen
`terminal=`*term*	Ausgabeformat für die Grafik; mögliche Werte: `screen` (default), `png`, `jpg`, `eps`, `eps_color`, `pdf`.
`file_name="`*file*`"`	Ausgabeziel für die Grafik; default: `maxima_out`
`user_preamble="`*text*`"`	Gnuplot-Vorspann, enthält beliebige Gnuplot-Befehle, die *vor* dem Plot ausgeführt werden
`dimensions=[`*width*`,`*height*`]`	Abmessungen der Grafik: in Bildpunkten bei Pixelgrafiken, in 1/10 mm bei Vektorgrafiken
`columns=`*n*	Anzahl der Spalten bei mehreren Szenen in einer Grafik
`color=`*colorname*	Zeichenfarbe für Linien
`background_color=`*name*	Hintergrundfarbe für das Diagramm
`fill_color=`*name*	Füllfarbe für Rechtecke, Polygone und Kreise
`x(yz)range=[`*min*`,`*max*`]`	Darstellungsbereich auf der x(yz)-Achse
`logx(yz)=true/false`	Logarithmische Skalierung der x(yz)-Axchse
`grid=true/false`	Zeichnen von Gitterlinien in der xy-Ebene
`x(yz)tics=true/false`	Bestimmt, ob Skalenpukte auf der x(yz)-Achse automatisch gesetzt werden sollen
`x(yz)tics_rotate=true/false`	Bestimmt, ob die Beschriftung der Skalenpukte um 90 Grad gedreht werden soll
`title="`*text*`"`	Diagrammtitel
`key="`*text*`"`	Angabe eines Funktionsnamens in der Legende (default: Leerstring)
`x(yz)label="`*text*`"`	Beschriftung der x(yz)-Achse
`x(yz)axis=true/false`	Bestimmt, ob eine x(yz)-Achse gezeichnet werden soll
`x(yz)axis_width=`*width*	Linienbreite für die entsprechende Achse
`x(yz)axis_color=`*color*	Farbe für die entsprechende Achse
`x(yz)axis_type=solid/dots`	Linientyp für die entsprechende Achse: durchgezogen (`solid`) oder punktiert (`dots`), default: `dots`
`line_width=`*width*	Linienbreite
`line_type=solid/dots`	Linientyp (default: `solid`)
`point_size=`*size*	Größe von Punkten bei Punkt-Plots
`point_type=`*n*	Punkt-Typ, mögliche Werte -1,0,1,2,... 13.
`points_joined=true/false`	Angabe, ob Punkte durch Linienzüge verbunden werden (default: `false`)
`nticks=`*n*	Anzahl der primären Kurvenpunkte für den adaptiven Plotalgorithmus (default: 30)
`adapt_depth=`*n*	maximale Anzahl der Splittings für den adaptiven Plotalgorithmus (default: 10)

Allgemeine Plot-Optionen für das Gnuplot-Interface *Draw*

Mit `set_draw_defaults` können Defaultwerte für beliebige Optionen gesetzt werden, das Rücksetzen aller Defaultwerte erfolgt mit einer leeren Parameterliste: `set_draw_defaults()`.

Optionen können in *Listen* zusammengefasst werden, diese Listen können – auch verschachtelt – an Stelle einzelner Optionen in die Plotroutinen eingesetzt werden.

`terminal` und `file_name` geben Ausgabeformat und Ausgabeziel für die Grafik an. Die möglichen Ausgabeformate entsprechen aber *nicht* den in Abschnitt 2.2 angegebenen Gnuplot-Terminals. Der angegebene Filename (default: `maxima_out`) darf keine Erweiterung enthalten, die passende Erweiterung (`.eps`, `.png`, `.jpg`) wird automatisch hinzugefügt. Wird der Pfadname nicht angegeben, so wird das File im Gnuplot-Arbeitsverzeichnis abgespeichert. Bei den wx-Plotroutinen ist eine Angabe von `terminal` oder `file_name` nicht sinnvoll und führt zu einer Fehlermeldung.

Die Option `user_preamble` kann als Zeichenkette beliebige Gnuplot-Befehle, durch Strichpunkte getrennt, enthalten, die vor dem eigentlichen Plotbefehl ausgeführt werden. Damit können Einstellungen an Gnuplot weitergereicht werden, für die es in Maxima keine eigenen Plot-Optionen gibt.

Mit der Option `dimensions` wird die Größe der Grafik festgelegt. Die Angaben für die Breite und Höhe gelten nicht für das Diagramm, sondern für die gesamte Grafik einschließlich Achsenbeschriftung und Rand (Default: `[600,500]`).

- Bei Pixelgrafiken (`png`, `jpg`) werden die Abmessungen in Pixel angegeben; dies gilt auch für die wx-Plotroutinen (Default: `[500,300]`).
- Bei Vektorgrafiken (`eps`, `jpg`) werden die Abmessungen in 1/10 mm angegeben.

Die Optionen zum Festlegen einer Gnuplot-Präambel, der Bildgröße, der Farbe und Anzahl der primären Kurvenpunkte werden als Defaultwerte für alle nachfolgenden Grafiken festgelegt.

```
(%i50) set_draw_defaults(
           user_preamble="set size ratio -1",
           dimensions=[200,200],color=red,
           nticks=200)$
```

Optionen zum Unterdrücken von Skalenpunkten in einer Liste

```
(%i51) notics:[xtics=false,ytics=false];
(%o51) [xtics = false, ytics = false]
```

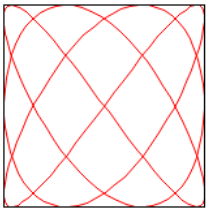
Erzeugung einer Lissajous-Figur als Kurve in Parameterdarstellung.

```
(%i52) lissa:parametric(sin(3*t),sin(4*t),t,0,2*%pi);
(%o52) parametric(sin(3 t), sin(4 t), t, 0, 2 π)
```

Plotten der Lissajous-Figur mit den neuen Defaultwerten und einer Liste von Optionen

```
(%i53) wxdraw2d(lissa,notics)$
```

(%t53)



Zurücksetzen aller Defaultwerte

```
(%i54) set_draw_defaults();
(%o54) []
```

Die Optionen `color` und `fill_color` legen Farben für Linien und Umrisse bzw. für ausgefüllte Flächen fest; die Farben können in hexadezimaler Schreibweise (`#rrggbb`) oder mit Namen entsprechend nachfolgender Tabelle angegeben werden.

`xrange`, `yrange` und `zrange` sind die Darstellungsbereiche in den entsprechenden Koordinaten in der Form `[`*min*`,`*max*`]`. Ihr Defaultwert ist `false`, in diesem Fall wird der Darstellungsbereich automatisch gewählt. Mit `logx`, `logy` und `logz` können die entsprechenden Achsen logarithmisch skaliert werden.

`grid`, `xtics`, `ytics` und `ztics` steuern die Darstellung von Gitterlinien und Skalenpunkten auf den Koordinatenachsen. Bei `grid=true` werden Gitterlinien an den Skalenpunkten gezeichnet. Der Wert von `xtics`, `ytics` und `zticks` kann `false` (keine Skalenpunkte), `true` (automatisches Ermitteln von Skalenpunkten) oder eine *Menge* von Werten sein, an denen Skalenpunkte (und ggf. Gitterlinien) gezeichnet werden:

`x(yz)tics={`*w1*`,`*w2*`,...}`

Jeder Wert kann auch eine *Liste* aus zwei Elementen sein: einer Zeichenkette, die dargestellt wird und dem Koordinatenwert, an dem die Zeichenkette dargestellt wird:

`x(yz)tics={["`*text1*`",`*w1*`],["`*text2*`",`*w2*`],...}`

Die Skalenbeschriftung der x(yz)-Achse kann mit `x(yz)tics_rotate=true` um 90 Grad gedreht werden.

white	light-green	light-pink
black	dark-green	dark-pink
gray0	spring-green	coral
gray10	forest-green	light-coral
gray20	sea-green	orange-red
gray30	blue	salmon
gray40	light-blue	light-salmon
gray50	dark-blue	dark-salmon
gray60	midnight-blue	aquamarine
gray70	navy	khaki
gray80	medium-blue	dark-khaki
gray90	royalblue	goldenrod
gray100	skyblue	light-goldenrod
gray	cyan	dark-goldenrod
light-gray	light-cyan	gold
dark-gray	dark-cyan	beige
red	magenta	brown
light-red	light-magenta	orange
dark-red	dark-magenta	dark-orange
yellow	turquoise	violet
light-yellow	light-turquoise	dark-violet
dark-yellow	dark-turquoise	plum
green	pink	purple

Farbnamen für das Gnuplot-Interface *Draw*

`title`, `xlabel`, `ylabel`, `zlabel` und `key` steuern die Diagrammbeschriftung.

Erstellen einer *Menge* aus Werten für Skalenpunkte auf der y-Achse und horizontale Gitterlinien.

```
(%i55) yt:setify(create_list(signum(n)
         *sqrt(abs(n)),n,[-4,-3,-1,0,1,3,4])/2);
```

(%o55) $-1, -\frac{1}{2}, 0, \frac{1}{2}, 1, -\frac{\sqrt{3}}{2}, \frac{\sqrt{3}}{2}$

Werte für Skalenpunkte auf der x-Achse und vertikale Gitterlinien als Listen, jeweils bestehend aus dem auszugebenden Text und der x-Koordinate.

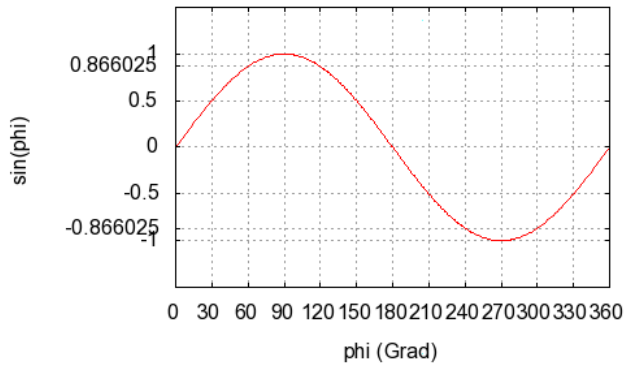
```
(%i56) xt:setify(makelist(
           [string(30*n),30*n*%pi/180],n,0,12));
```

(%o56) $\{[0,0], [120, \frac{2\pi}{3}], [150, \frac{5\pi}{6}], [180, \pi], [210, \frac{7\pi}{6}], [240, \frac{4\pi}{3}], [270, \frac{3\pi}{2}], [30, \frac{\pi}{6}], [300, \frac{5\pi}{3}], [330, \frac{11\pi}{6}], [360, 2\pi], [60, \frac{\pi}{3}], [90, \frac{\pi}{2}]\}$

Sinusfunktion mit Gitterlinien und Achsenbeschriftungen

```
(%i57) wxdraw2d(yrange=[-1.5,1.5],xtics=xt,
       ytics=yt,grid=true,color=red,
       explicit(sin(x),x,0,2*%pi),
       xlabel="phi (Grad)",ylabel="sin(phi)")$
```

(%t57)



Bei Punkt-Plots (mit dem Grafikobjekt `points`) kann der Punkt-Typ mit der Option `point_type` als Ganzzahl zwischen -1 und 16 oder als Name entsprechend folgender Tabelle eingegeben werden:

-1 $none	4 square	9 filled_up_triangle
0 dot	5 filled_square	10 down_triangle
1 plus	6 circle	11 filled_down_triangle
2 multiply	7 filled_circle	12 diamant
3 asterisk	8 up_triangle	13 filled_diamant

Werte für die Option `point_type`

Erzeugung von Punkten entlang einer Sinuskurve als Grafikobjekt

```
(%i58) sine:points(float(map(lambda([u],
           [u,sin(u)]),makelist(u,u,-6,6)/2)));
```

(%o58) points([[−3.0, −0.141], [−2.5, −0.598], [−2.0, −0.909], [−1.5, −0.997], [−1.0, −0.841], [−0.5, −0.479], [0.0, 0.0], [0.5, 0.479], [1.0, 0.841], [1.5, 0.997], [2.0, 0.909], [2.5, 0.598], [3.0, 0.141]])

Erzeugung von Punkten auf einer Geraden als Grafikobjekt

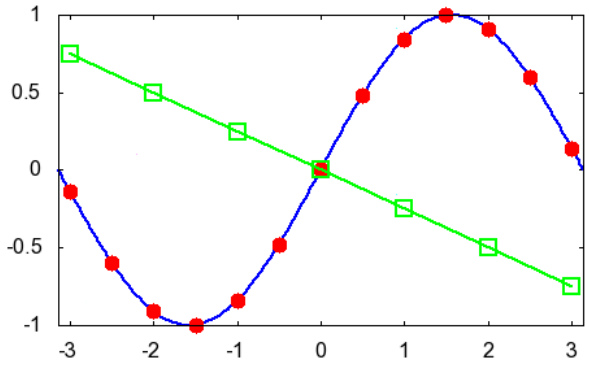
```
(%i59) line:points(float(makelist([k,-k/4],k,-3,3)));
```

(%o59) points([[−3.0, 0.75], [−2.0, 0.5], [−1.0, 0.25], [0.0, 0.0], [1.0, −0.25], [2.0, −0.5], [3.0, −0.75]])

Sinuskurve und zwei Punkt-Plots: Punkte entlang der Sinuslinie, sowie durch Liniensegmente verbundene Punkte auf einer Geraden.

```
(%i60) wxdraw2d(color=blue,line_width=2,
         explicit(sin(x),x,-%pi,%pi),
         color=red,point_size=2,point_type=7,
         sine,points_joined=true,point_type=4,
         color=green,line)$
```

(%t60)



4.5 Spezielle Optionen für Labels und Vektoren

`label_alignment=`*value*	Ausrichtung des Labels; mögliche Werte: `center` (default), `left`, `right`
`label_orientation=`*value*	Orientierung des Labels; mögliche Werte: `horizontal` (default), `vertical`
`head_length=`*len*	Länge der Pfeilspitzen von Vektoren in Einheiten der x-Achse (default: 2)
`head_angle=`*winkel*	Winkel der Pfeilspitzen von Vektoren (default: 45 Grad)
`head_type=`*typ*	Typ der Pfeilspitze von Vektoren; mögliche Werte: `filled` (default), `empty`, `nofilled`
`head_both=true/false`	Angabe, ob Vektoren Pfeilspitzen an beiden Enden haben

Plot-Optionen für Labels und Vektoren

Erzeugung dreier Zeichenketten

```
(%i61) [l1,l2,l3]:["left aligned label",
       "centered label","rigt aligned label"]$
```

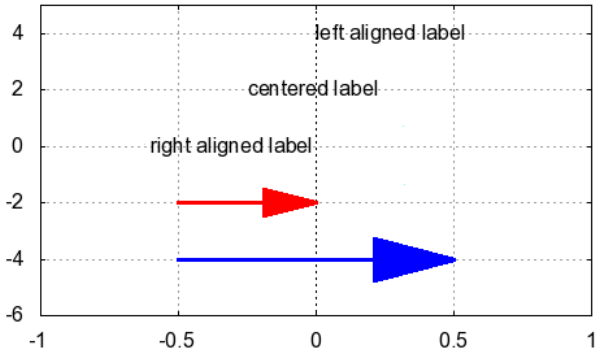
Zuweisen von langen Optionsnamen zu kurzen Variablennamen bringt zukünftige Schreiberleichterung.

```
(%i62) [la,hs,ha,hl]:[label_alignment,
       head_size,head_angle,head_length]$
```

Vektoren und unterschiedlich ausgerichtete Texte

```
(%i63) wxdraw2d(xrange=[-1,1],yrange=[-6,5],
yaxis=true,line_width=3,grid=true,la=left,
label([l1,0,4]),la=center,label([l2,0,2]),
la=right,label([l3,0,0]),ha=15,hl=0.2,
color=red,vector([-0.5,-2],[0.5,0]),
hl=0.3,color=blue,vector([-0.5,-4],[1,0]))$
```

(%t63)



4.6 Optionen für 2d-Grafiken

`axis_bottom=true/false`	Koodinatenachse auf der Diagramm-Unterseite (default: `true`)
`axis_top=true/false`	Koodinatenachse auf der Diagramm-Oberseite (default: `true`)
`axis_left=true/false`	Koodinatenachse auf der linken Diagrammseite (default: `true`)
`axis_right=true/false`	Koodinatenachse auf der rechten Diagrammseite (default: `true`)
`filled_func=true/false`	Bestimmt, ob der Bereich zwischen der Diagramm-Unterkante und der geplotteten Funktion färbig ausgefüllt werden soll.
`filled_func=`f	Färbiges Ausfüllen des Bereiches zwischen der zu plottenden Funktion und der Funktion f.
`transparent=true/false`	Bestimmt, ob Polygone (entsprechend der Option `fill_color`) ausgefüllt werden sollen.
`border=true/false`	Bestimmt, ob der Umriss von Polygonen gezeichnet werden soll.

Plot-Optionen für 2d-Grafiken

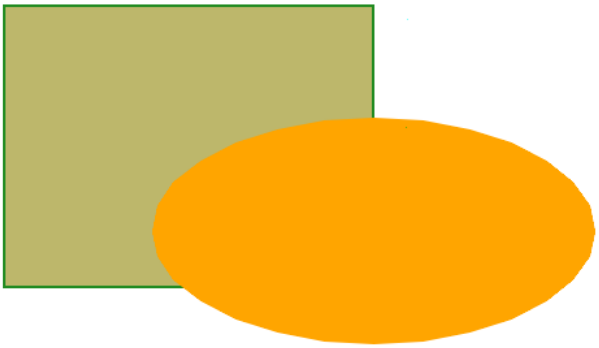
Liste zum Unterdrücken der Koordinatenachsen auf den Diagrammseiten und von Skalenmarkierungen.

```
(%i64) noframe:[axis_left=false,axis_right=false,
       axis_top=false,axis_bottom=false,
       xtics=false,ytics=false]$
```

Zeichnen verschiedener Formen mit unterschiedlichen Optionen

```
(%i65) wxdraw2d(noframe,line_width=2,
         fill_color=dark-khaki,
         color=forest-green,rechteck,
         border=false,fill_color=orange,ell)$
```

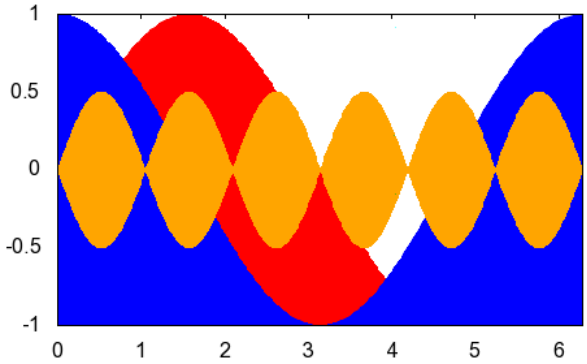
(%t65)



Färbiges Ausfüllen von Funktionen

```
(%i66) wxdraw2d(line_width=3,filled_func=true,
color=black,explicit(sin(x),x,0,2*%pi),
fill_color=blue,explicit(cos(x),x,0,2*%pi),
filled_func=0.5*sin(3*x),fill_color=orange,
explicit( -0.5*sin(3*x),x,0,2*%pi))$
```

(%t66)



4.7 Optionen für 3d-Grafiken

`view=[`φ`,`ϑ`]`	Festlegen des Ansichtswinkels: φ... um die x-Achse, ϑ... um die z-Achse
`axis_3d=true/false`	Anzeige aller Achsen in 3d-Grafiken (default: `true`)
`xu_grid=`*n*	Anzahl der Koordinatenwerte für Gitterlinien entlang der x-Achse
`yv_grid=`*n*	Anzahl der Koordinatenwerte für Gitterlinien entlang der y-Achse
`surface_hide=true/false`	Steuerung der Sichtbarkeit verdeckter Teile von 3d-Flächen
`enhanced3d=true/false`	Bestimmt das Einfärben von 3d-Flächen
`palette=[`*r*`,`*g*`,`*b*`]`	Nummern der Farbpaletten für die Rot-, Grün- und Blaukomponenten für Bildobjekte und 3d-Oberflächen
`colorbox=true/false`	Angabe, ob bei Diagrammen mit Farbpalette eine Farbskala gezeichnet werden soll (default: `true`)
`contour=`*value*	Steuerung der Darstellung von Isolinien (Höhenlinien) von 3d-Flächen; mögliche Werte: `none` (default), `base`, `surface`, `both`, `map`.
`contour_levels=`*n*	Festlegung der Isolinien auf *t* gleichverteilte Werte
`contour_levels=[`*x1*`,`*dx*`,`*x2*`]`	Festlegung von Isolinien im Abstand *dx* mit der Untergrenze *x1* und der Obergrenze *x2*
`contour_levels={õ\ti {x1}õ,õ\ti {x2}õ,õ\ldots õ}`	Isolinien werden an den Werten *x1*, *x2*, ... gezeichnet.
`ip_grid=[`*nx*`,`*ny*`]`	Anzahl der primären Gitterpunkte in x- bzw. y-Richtung bei `implicit`-Plots für den adaptiven Plotalgorithmus (default: `[50,50]`)
`ip_grid_in=[`*nx*`,`*ny*`]`	Anzahl der sekundären Gitterpunkte bei `implicit`-Plots (default: `[5,5]`)

Plot-Optionen für 3d-Grafiken

Die Option `view=[`φ`,`ϑ`]` legt die Ansicht auf die 3d-Grafik mit zwei Listenelementen fest:

- φ ist der Drehwinkel um die x-Achse aus der Waagrechten heraus; $\varphi = 0$ ergibt eine waagrechte Blickrichtung, $\varphi = 90$ eine Draufsicht.
- ϑ ist der Drehwinkel um die z-Achse. $\vartheta = 0$ ergibt eine Vorderansicht, $\vartheta = 90$ eine Seitenansicht.

`axis_3d` legt die Darstellung aller Achsen von 3d-Grafiken fest. Um die Achsen zur Gänze auszuschalten, müssen zusätzlich zu `axis_3d=false` die Achsenmarkierungen mit `x(xz)tics=false` unterdrückt werden.

`xu_grid` und `yv_grid` legen Anzahl der *Koordinatenwerte* für Gitterlinien entlang der x- bzw. y-Achse an. Die Anzahl der *Gitterlinien* in der jeweiligen Richtung ist um 1 höher als die in diesen Optionen angegebenen Werte. Die Gitterpunkte werden durch Geradenstücke verbunden, die Oberfläche wird daher umso genauer nachgebildet, je größer diese Werte sind.

Definition einer Funktion in zwei Variablen

```
(%i67) f:2/(x^2+y^2+1)+1/((x-5)^2+y^2+1);
```

$$(\%o67)\quad \frac{2}{y^2+x^2+1}+\frac{1}{y^2+(x-5)^2+1}$$

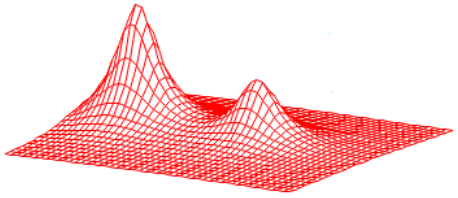
Erzeugung eines Graphikobjekts

```
(%i68) g:explicit(f,x,-2,10,y,-6,6)$
```

3D-Darstellung der Funktion

```
(%i69) wxdraw3d(xtics=false,ytics=false,
           ztics=false,axis_3d=false,xu_grid=50,
           color=red,surface_hide=true,g)$
```

(%t69)



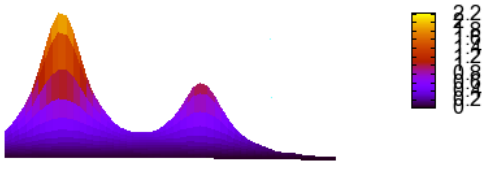
Wird die Option `enhanced3d` auf `true` gesetzt, so werden die Oberflächen mit einem Farbverlauf, der immer von der z-Koordinate abhängt, eingefärbt. Der Farbverlauf kann mit der Option `palette` eingestellt werden, wobei die Farbfunktionen, getrennt für den Rot-, Grün- und Blauanteil, mit einer Nummer von 0 bis 26 ausgewählt werden. Die Farbfunktionen sind in [1] angeführt.

Mit der Option `colorbox` kann die Darstellung einer Farbskala für den Farbverlauf gesteuert werden.

Funktion mit eingefärbter Oberfläche. Die gesamte Farbpalette, dargestellt in der nebenstehenden Farbskala, füllt unabhängig von den absoluten Koordinaten immer den gesamten Darstellungsbereich der z-Achse aus.

```
(%i70) wxdraw3d(xtics=false,ytics=false,
           ztics=false,axis_3d=false,xu_grid=50,
           enhanced3d=true,surface_hide=true,
           view=[90,0],g)$
```

(%t70)



Die Option `contours` steuert die Darstellung von Höhenschichtlinien, dabei sind folgende Werte möglich:

- `none`: Isolinien werden nicht gezeichnet (default).
- `base`: Isolinien werden auf die xy-Ebene projiziert.
- `surface`: Isolinien werden auf der Oberfläche gezeichnet.

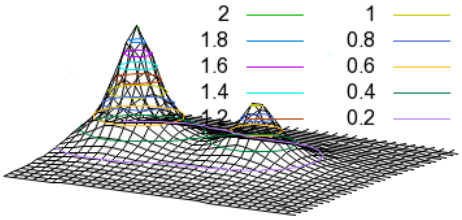
- `both`: Isolinien werden sowohl in der xy-Ebene als auch auf der Oberfläche.
- `map`: Ansicht von oben (entspricht `rot_vertical=0`), das Oberflächengitter wird nicht dargestellt.

Mit der Option `contour_levels` können die Anzahl und Werte der Isolinien festgelegt werden.

Darstellung von Isolinien auf der Oberfläche

```
(%i71) wxdraw3d(xtics=false,ytics=false,
           ztics=false,axis_3d=false,
           contour_levels=10,contour=surface,g)$
```

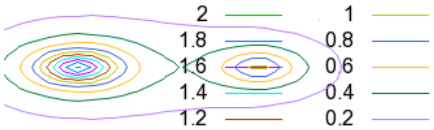
(%t71)



Darstellung der Funktion als Isolinien-Diagramm

```
(%i72) wxdraw3d(xtics=false,ytics=false,
           ztics=false,axis_3d=false,
           contour_levels=10,contour=map,g)$
```

(%t72)



Literaturverzeichnis

[1] Maxima Development Team: *Maxima Reference Manual V.5.23.* 2011.

[2] Philipp K. Janert: *Gnuplot in Action, Understanding Data with Graphs.* Manning Publications 2009.

[3] Mario Rodríguez Riotorto: *A Maxima-Gnuplot Interface.* `http://www.telefonica.net/web2/biomates/maxima/gpdraw`.