Maxima – eine kurze Bedienungsanleitung

Maxima ist ein Mathematik-Paket, das für alle wichtigen Betriebssysteme gratis zur Verfügung steht. Sein Kern ist seit Jahren erprobt und bewährt (entspricht 'Derive'), deshalb wird es auch an Universitäten gerne eingesetzt. Das Programm selbst ist rein textbasiert, deswegen enthalten alle Pakete eine zusätzliche grafische Oberfläche zur einfachen Benutzung. Die zugrundeliegende Programmiersprache ist Lisp, man kann eigene Erweiterungen programmieren.
Die integrierte Hilfe ist gut organisiert, viele Arbeitsschritte können im Menü ausgewählt werden, was für Anfänger recht praktisch ist.

Ich verwende hier **wxMaxima** für Windows, Linux und OSX. Die Beispiele zeigen nur einen kleinen Teil der Programmfähigkeiten, Maxima deckt aber den gesamten Schulmathematik und Universitäts-Mathestoff ab.

Ein- und Ausgabe:
einzelne Eingaben werden mit einem **Strichpunkt** beendet. Man kann jederzeit mit <RETURN> eine neue Zeile beginnen, wenn dies optisch gewünscht ist.
Die Tastenkombination **<SHIFT><RETURN>** übergibt die Eingabe an Maxima und zeigt das Ergebnis an.
Eingaben sind editierbar und können neuerlich ausgeführt werden.
Wird die Eingabe mit einem **Dollarzeichen** $ beendet, erzeugt Maxima keinen Output, das ist z.B. bei Variablen- oder Funktionsdefinitionen sinnvoll, wo die Ausgabezeile nichts neues enthält.
Ein **Beistrich** erlaubt die Angabe zusätzlicher Werte.
Die Eingabezeilen (input) werden automatisch durchnummeriert in der Form (%i1), (%i2),...
Die Ausgaben weden entsprechend als (%o1), (%o2) gekennzeichnet.

Das aktuelle Ergebnis kann mittles **%** weiterverwendet werden, vorangegangene Terme durch Angabe der Nummer wie etwa %o2. Besser ist es natürlich, Ausdrücke mit einem guten Namen zu belegen und diesen weiterzuverwenden.

Spezialsymbole:
Maxima verwendet keine Spezialzeichen, sondern ausschließlich den Standardzeichensatz.
Sonderzeichen werden durch ein vorangestelltes % gekennzeichnet

%e	Euler-sche Zahl
%pi	Pi
%phi	goldener Schnitt
%i	imaginäre Einheit der komplexen Zahlen
inf	positiv unendlich
minf	negativ unendlich
infinity	komplex Unendlich

Konventionen:
- Terme werden durch einen Doppelpunkt erklärt. z.B. a : 7 oder wert : 2
- Malzeichen müssen IMMER geschrieben werden, da sonst keine Bezeichner mit mehr als einem Buchstaben möglich wären. z.B. 3*x^2 + 4*x – 2
- Funktionen werden mittels := erklärt. z.B. f(x) := sin(x)
- das normale Gleichheitszeichen = wird für die Gleichsetzung von Ausdrücken benutzt.
- Listen erscheinen in eckiger Klammer. Man kann auf Einzelkomponenten zugreifen [a,b]: [2,4];
- Die zwei Seiten (Terme) einer Gleichung heißen lhs() und rhs() (left – und right hand side)

Maxima als Taschenrechner

(%i1)	12+7; <SHIFT><RETURN>	
(%o1)	19	
(%i2)	6^66;	große Zahlen
(%o2)	228025031986703799742184233008522791795627262581145 6	
(%i3)	6^666;	
(%o3)	177309806357755465270499427328[459 digits]407237966516507939097024659456	

(%i1)	12/7;	
(%o1)	$\frac{12}{7}$	Maxima rechnet mit Brüchen
(%i2)	%^4;	
(%o2)	$\frac{20736}{2401}$	Berechnungsergebnis exakt als Bruch
(%i3)	float(%);	
(%o3)	8.636401499375261	als Dezimalzahl
(%i4)	1.0/3	Rechnung mit Dezimalzahl
(%o4)	0.33333333333333	bleibt Dezimalzahl, CPU-Genauigkeit max. 16 Stellen

(%i1)	bruch : 1/7;	
(%o1)	1/7	
(%i2)	bruch, float;	
(%o2)	0.14285714285714	als normale Dezimalzahl
(%i3)	fpprec : 100;	Rechengenauigkeit für float
(%o3)	100	
(%i4)	b = bfloat(1/7);	big float
(%o4)	1.4285714285714285714285714285[44 digits]42857142857142857142857142 9b-1	1.4... x 10^{-1} big float
(%i5)	fpprintprec : 10;	mit 10 Stellen gut gerundet anzeigen
(%o5)	10	
(%i6)	b, numer;	im Standardformat anzeigen
(%o6)	0.14285714	

Variable und Funktionen

(%i1)	a:3;	eine Variable
(%o1)	3	
(%i2)	f(x) := (a*x-2)^5;	eine Funktion

(%o2)	f(x):=(a*x-2)^5	(Echo der Eingabe)
(%i3)	f(2);	Funktionswert berechnen
(%o3)	1024	
(%i4)	f(x);	aktuelle Funktionsgleichung
(%o4)	$(3x-2)^5$	
(%i5)	kill(a);	Variable a jetzt wieder ohne Wert
(%o5)	done	
(%i6)	f(3);	Funktion wieder mit Formvariable
(%o6)	$(3a-2)^5$	

Termumformungen

(%i1)	a: (2*x-4)^3; b: (x-2)^2;	zwei Terme
(%o1)	$(2x-4)^3$	
(%o2)	$(x-2)^2$	
(%i3)	expand(a);	a ausrechnen
(%o3)	$8*x^3-48*x^2+96*x-64$	
(%i4)	factor(b-4);	einen Term faktorisieren
(%o4)	(x-4)x	
(%i5)	a/b;	
(%o5)	$\frac{(2x-4)^3}{(x-2)^2}$	korrektes Ergebnis (an Stelle 2 nicht definiert, daher ungekürzt)
(%i6)	ratsimp(%);	'rational simplify'
(%o6)	8x-16	der Bruch vollständig gekürzt

Funktionen

(%i1)	sin(%pi/2) + cos(%pi/3);	
(%o1)	$\frac{3}{2}$	
(%i2)	log(inf);	natürlicher Logarithmus
(%o2)	inf	
(%i3)	log2(x) := log(x)/log(2); float(log2(8));	
(%o3)	3.0	

Berechnungen

(%i1)	10!;	Faktorielle
(%o1)	3628800	
(%i2)	factor(%);	(Prim-)Faktorzerlegung
(%o2)	$2^8*3^4*5^2*7$	
(%i3)	%i^3;	komplexe Zahlen

(%o3)	-%i	
(%i4)	binomial(10,3);	Binomialkoeffizient 10 über 3
(%o4)	120	

Gleichungen

(%i1)	solve(x^2-4=0,x);	reell
(%o1)	[x=-2,x=2]	
(%i2)	solve(x^2+4=0,x);	komplex
(%o2)	[x=-2*%i,x=2*%i]	
(%i3)	solve(x^4-x^2-4=0,x);	höheren Grades
(%o3)	... zwei reelle und 2 komplexe Lösungen exakt ...	
(%i4)	linsolve([2*b+a=8,2*a-b=1],[a,b]);	lineares Gleichungssystem
(%o4)	[a=2,b=3]	

Folgen und Reihen, Grenzwerte

(%i1)	load("functs");	Zusatzfunktionen laden
(%i2)	arithmetic(10,2,3);	arithmetic(a,d,n) <a,a+d,a+2d,...> arithm. Folge
(%o2)	14	
(%i3)	arithsum(10,2,3);	arithmetische Reihe, die Summe
(%o3)	36	
(%i4)	geometric(10,2,3);	geometric(a,c,n) <a,ac,ac^2,..> geometrische Folge
(%o4)	40	
(%i5)	geosum(10,2,3);	geometrische Reihe, die Summe
(%o5)	70	
(%i6)	limit((2-4*x)/(x+3),x,inf);	Grenzwert für x gegen unendlich
(%o6)	-4	

Differenzieren

(%i1)	diff(sin(x),x);	erste Ableitung
(%o1)	cos(x);	
(%i2)	diff(y^5,y,4);	vierte Ableitung
(%o2)	120y	
(%i3)	diff(a^2*b^4*c,b);	partielle Ableitung
(%o3)	$4a^2b^3c$	

Integrieren

(%i1)	integrate(1/t,t);	Stammfunktion
(%o1)	log(t)	
(%i2)	integrate(16*x^3-2*x, x,2,5);	Bestimmtes Integral
(%o2)	2415	

Statistik – Wahrscheinllichkeitsfunktionen
pdf ... probability density function, cdf ... cumulative density function

(%i1)	load(descriptive);	
(%i2)	s : [2,3.5,1,2,5,4.2,3.8];	ein Datensatz, Messwerte
(%o2)	[2,3.5,1,2,5,4.2,3.8]	
(%i3)	mean(s);	arithmetisches Mittel
(%o3)	3.071428571	
(%i4)	median(s);	der Median von s
(%o4)	3.5	
(%i5)	std(s);	Standardabweichung (Nenner n)
(%o5)	1.325264702	
(%i6)	histogram(s);	erzeugt ein einfaches Histogramm
(%i7)	piechart(s);	Tortendiagramm
(%i8)	boxplot(s);	Diagramm mit Quartilen

(%i1)	load(distrib);	
(%i2)	pdf_binomial(1,3,0.25);	Binomialverteilung (DICHTE), x=1, n=3, p=0.25 1 mal Erfolg bei 3 Versuchen mit Erfolgswahrscheinlichkeit p
(%o2)	0.421875	
(%i3)	cdf_normal(0.45,0,1), float;	Normalverteilung Φ(z) z = 0.45, m=0,s=1
(%o3)	0.67364478	
(%i4)	lotto(n) := binomial(6,n) *binomial(39,6-n) /binomial(45,6);	hypergeometrische Verteilung (DICHTE): Lottowahrscheinlichkeit für n Treffer 6 aus 45.
(%i5)	pdf_hypergeometric(4,6,39,6);	k=4 Treffer, r=6 gute, s=39 schlechte, n=6 Werte gezogen
(%o5)	741/543004	gekürztes Ergebnis, 100*float(%) für Antwort in Prozent

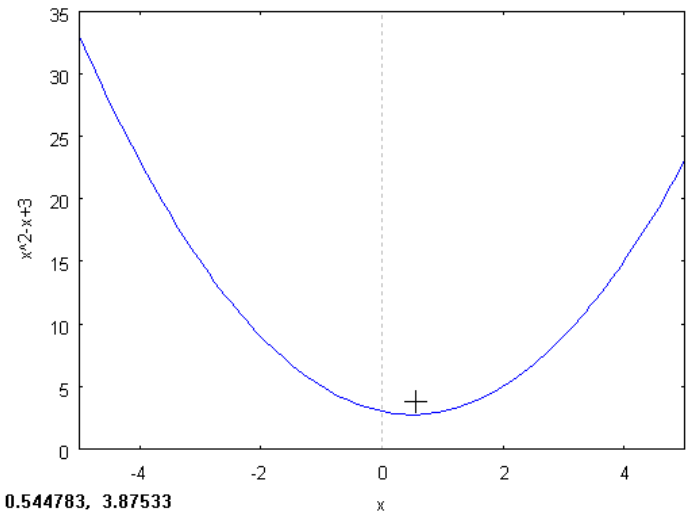
allgemein:

pdf_ ergibt die probability density function, die Wahrscheinlichkeit für genau k Treffer

cdf_ ergibt die cumulative density function, die Wahrscheinlichkeit für höchstens k Treffer (0,1,2, bis k mal aufaddiert)

Grafik

```
(%i1)   plot2d(x^2-x+3,[x,-5,5]);
```



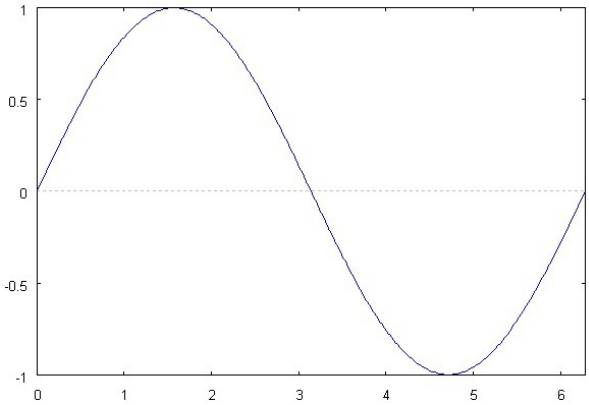
Plots erscheinen in einem eigenen Fenster.
Die Grafiken passen sich in Windows auch im Nachhinein der Fenstergröße an.

Stellst Du dem Plotbefehl die Buchstaben wx voran, so wird die Grafik in das Dokument eingebettet!

wxplot2d(x^2-x+3,[x,-5,5]);

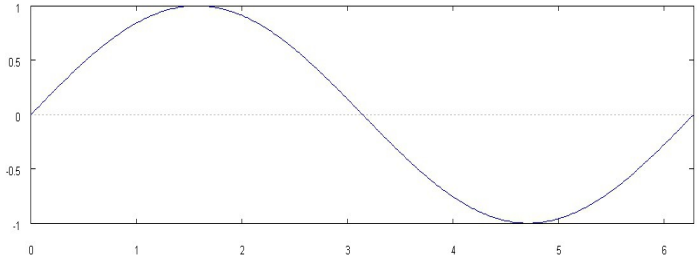
Maxima skaliert Grafiken automatisch so, dass ein für die Ansicht vernünftiges Seitenverhältnis entsteht.

```
plot2d(sin(x), [x,0,2*%pi]);
```



Dabei werden waagrechte und senkrechte Achse meist nicht mit gleich großen Einheiten gezeigt. Möchte man, dass die Achsen gleich skaliert sind, fügt man eine Anweisung an die plot-Routine an, um diese Automatik abzuschalten (Verhältnis -1):

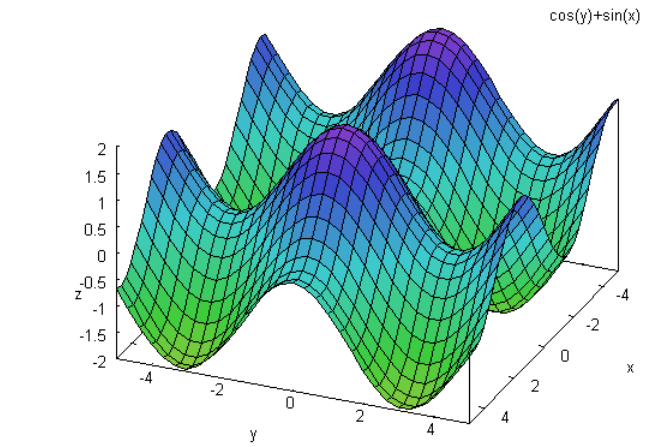
```
plot2d(sin(x),[x,0,2*%pi],[gnuplot_preamble,"set size ratio -1"]);
```



ein 3D-Graph:

```
(%i1)   plot3d(sin(x)+cos(y),[x,-5,5],[y,-5,5]);
```

(%o1)



Die 3D-Grafik kannst Du in Windows anclicken und drehen (wenn sie nicht eingebettet ist).

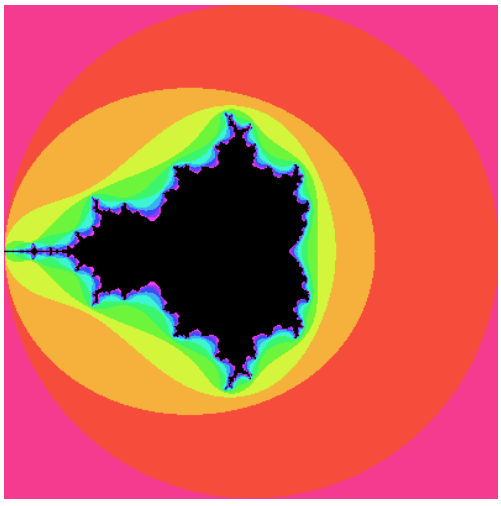
Maxima beherrscht auch das Lösen von Differentialgleichungen, Reihenentwicklungen, Statistik und Testverfahren,... und sogar die fraktale Geometrie!

Hier die **Mandelbrotmenge** mit Standardwerten:

```
(%i1)   load("dynamics");
```

```
(%i2)   mandelbrot([filename,"firstTest"]);
```

(%o2)

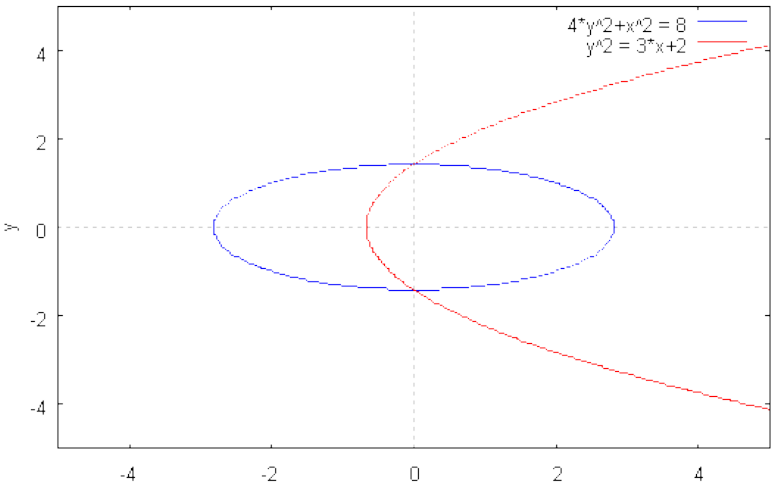


Implizit definierte Funktionen (wo also nicht f(x)=... oder f(y)=... explizit angebbar ist) lassen sich ebenfalls darstellen, wenn man zuvor die passende Bibliothek lädt.

```
(%i1)  load(implicit_plot);
```

```
(%i2)  implicit_plot([x^2+4*y^2=8,y^2=3*x+2],[x,-5,5],[y,-5,5]);
```

(%o2)



Diese Plots sehen oft etwas 'unrund' aus, da sie nur genähert ausgewertet werden (man könnte die Auflösung aber verbessern). Den Verlauf und die Lage der Kurven und eventuelle Schnittpunkte kann man trotzdem sehr gut sehen.

Ergänzung:

Definitionsbereiche: manche Rechnungen lassen sich nur dann ausführen, wenn zusätzliche Informationen zu den Variablen vorliegen. Ein Beispiel: Was ist die Stammfunktion von x hoch n?

```
(%i1)  integrate(x^n,x);
```

```
(%o1)  "Is  "n+1"  zero or nonzero?"
```

Da hat Maxima recht!

```
(%i2)  assume(n>0);
       integrate(x^n,x);
```

(%o2)

$$\frac{x^{n+1}}{n+1}$$

Beim Rechnen mit Formvariablen ist es wichtig, dass diesen nicht bereits ein Wert zugewiesen wurde. Um eine **zuvor erfolgte Wertzuweisung zu entfernen**, gibt es die Funktion *kill*.
Durch *kill(d)* ist das Symbol d wieder 'ungebunden'. *kill(all)* löscht alle Bindungen. Alternativ kann man im Menü Maxima / Speicher löschen auswählen. Die Funktion *reset()* vernichtet auch alle Zuordnungen von globalen Variablen.

Neue Funktionen aus bereits bekannten aufbauen: *define()* z.B Ableitungsfunktionen

```
f(x) := x^3-6*x^2+1;
df(x):= define(diff(f(x),x));        oder  ''(df(x),diff(f(x),x))
ddf(x):= define(diff(f(x),x,2));     oder  ''(ddf(x),diff(f(x),x,2))
```

[mit df(x) := diff(...) könnten wir später keine Funktionswerte bestimmen, da mit df(2) zuerst 2 eingesetzt, dann erst differenziert würde. Nach x? Nein, nach 2. Und das geht nicht]