

Einführung in Maxima

Robert Glöckner © 2006

Inhaltsverzeichnis

1. Einführung in Maxima.....	2
2. Einführung.....	3
2.1. Starten von Maxima.....	3
2.2. Kommandoeingabe.....	3
2.3. Zuweisungen.....	4
2.4. Beenden von Maxima.....	4
2.5. Hilfsfunktionen.....	5
2.6. Darstellung der Ergebnisse.....	6
3. Rechnen mit Maxima.....	9
3.1. Operatoren.....	9
3.2. Algebra.....	9
3.3. Lösen von Gleichungssystemen.....	10
3.4. Trigonometrische Funktionen.....	11
3.5. Komplexe Zahlen.....	13
3.6. Ableitungen, Grenzwerte, Integrale.....	14
3.7. Lösen von Differentialgleichungen.....	15
3.8. Matrizenrechnung.....	16
4. Programmieren in Maxima.....	19
5. Wichtige Maxima-Funktionen.....	21

\$Header: /home/cvs/lisp/maxima-einfuehrung.html,v 1.24 2006/01/17

23:43:45 robert Exp \$

1. Einführung in Maxima

Copyright (C) Robert Glöckner

email (entferne alle Ziffern): 2R4o5b66e7r8t9.G0l5oe55ck8ne5r@6w0eb4.d3e

This program-documentation is free software-documentation; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program-documentation is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program-documentation; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

To keep this document short, a link to the text of the [GPL-LICENCE](#).

Inoffizielle Übersetzung: Diese Programm-Dokumentation ist freie Software. Sie können es unter den Bedingungen der GNU General Public License, wie von der Free Software Foundation veröffentlicht, weitergeben und/oder modifizieren, entweder gemäß Version 2 der Lizenz oder (nach Ihrer Option) jeder späteren Version.

Die Veröffentlichung dieser Programm-Dokumentation erfolgt in der Hoffnung, daß es Ihnen von Nutzen sein wird, aber OHNE IRGEND EINE GARANTIE, sogar ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK. Details finden Sie in der GNU General Public License.

Sie sollten ein Exemplar der GNU General Public License zusammen mit dieser Programm-Dokumentation erhalten haben. Falls nicht, schreiben Sie an die Free Software Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110, USA.

Um das Dokument nicht aufzublähen, hier ein Link auf die [inoffizielle deutsche Übersetzung der GPL](#)

Ich möchte nur kurz betonen, dass ich selbst Maxima-Anfänger bin und der Text nur auf die Grundlagen der Maxima-Benutzung eingehen kann. Die Beispiele haben keinen tieferen Sinn. Sie dienen lediglich der Darstellung der Möglichkeiten von Maxima.

Für weitere Anregungen bin ich immer dankbar.

email (entferne alle Ziffern): 2R4o5b66e7r8t9.G0l5oe55ck8ne5r@6w0eb4.d3e

Verbesserungsvorschläge durch: Volker van Nek, Robert Figura.

2. Einführung

2.1. Starten von Maxima

Maxima ist ein in Lisp geschriebenes freies Computer-Algebra System ([homepage](#)). Es ist auf verschiedenen Betriebssystemen lauffähig. Es gibt mehrere Möglichkeiten das Programm zu verwenden:

- auf der Konsole (hierzu maxima, bzw. maxima.bat starten)
- eine rudimentäre grafische Oberfläche bietet xmaxima (mitgeliefert)
- eine grafische FormelAusgabe bietet wxmaxima
- für Leute die LaTeX benutzen ist texmax und emaxima interessant
- für Emacs-verrückte gibt es einen mitgelieferten maxima und emaxima Modus (Start im Emacs mit M-x maxima oder Öffnen einer .max Datei)

Startet man Maxima (auf der Konsole) so erhält man folgende Meldung:

```
Maxima 5.9.2 http://maxima.sourceforge.net
Using Lisp GNU Common Lisp (GCL) GCL 2.6.7 (aka GCL)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
This is a development version of Maxima. The function bug_report()
provides bug reporting information.
(%i1)
```

Es erscheint eine Meldung über die freie Lizenz, die Widmung an Prof. W. Schelter (ihm haben wir die freie Version von Maxima zu verdanken) und ein sog. Label (%i1). Jede Eingabe wird mit einer Marke (Label) gekennzeichnet. Marken, welche mit einem i beginnen kennzeichnen Benutzereingaben, o-Markierungen kennzeichnen Ausgaben des Programms. Der Benutzer sollte dies bei der Namensgebung eigener Variablen oder Funktionen berücksichtigen, um Verwechslungen zu vermeiden.

2.2. Kommandoeingabe

Kommandos werden entweder mit einem Semikolon ; oder einem \$ abgeschlossen. Es reicht nicht, Return oder Enter zu drücken, Maxima wartet auf eines der beiden Zeichen, vorher beginnt Maxima nicht mit der Auswertung der Eingabe. Ist das letzte Zeichen ein Semikolon, so wird das Ergebnis der Verarbeitung angezeigt, im Fall eines Dollarzeichens wird die Anzeige unterdrückt. Dies kann bei sehr langen Ergebnissen sinnvoll sein, um die Wartezeit zu reduzieren und die Übersicht zu wahren.

Maxima unterscheidet Groß- und Kleinschreibung. Alle eingebauten Funktionen und Konstanten sind kleingeschrieben (simp, solve, ode2, sin, cos, %e, %pi, inf etc). slmP oder SIMP werden von Maxima nicht den eingebauten Funktionen

zugeordnet. Benutzerfunktionen und -variablen können klein und/oder groß geschrieben werden.

Auf vorangegangene Ergebnisse und Ausdrücke kann mittels % zugriffen werden. % bezeichnet das letzte Ergebnis, %i13 die 13. Eingabe und %o27 das 27. Ergebnis,

' '%i42 wiederholt die Berechnung der 42. Eingabe, %th(2) ist das vorletzte Ergebnis.

2.3. Zuweisungen

Ausdrücke werden mit : einem Symbol zugewiesen. Funktionen werden mit := einem Symbol zugewiesen.

```
(%i151) value : 3;

(%o151)
3
(%i152) equation : a + 2 = b;

(%o152)
a + 2 = b
(%i153) function(x) := x + 3;

(%o153)
function(x) := x + 3
(%i154) function(3);

(%o154)
6
(%i155) function(b);

(%o155)
b + 3
```

Zuweisungen werden mit kill einzeln oder auch insgesamt gelöscht werden.

```
(%i150) kill(equation);

(%o150)
done
(%i151) (%i151) equation;

(%o151)
equation
(%i152) function(3);

(%o152)
6
(%i153) kill(all);

(%o0)
done
(%i1) (%i1) function(3);

(%o1)
function(3)
(%i2)
```

2.4. Beenden von Maxima

Zum Abbrechen eines Kommandos drückt man die Tastenkombination `Strg-C` oder `Strg-G`. Meldet sich der Debugger, so beendet man diesen durch Eingabe von `Q`.

Zum Beenden von Maxima gibt man `quit()`; ein (Bemerkung: unter xmaxima das Menue benutzen).

2.5. Hilfsfunktionen

Neben den Hilfsfunktionen der Benutzerumgebung enthält Maxima eigene Hilfsfunktionen. Mit `apropos` kann nach Befehlen bezüglich eines Stichwortes gesucht werden, mit `describe` können detaillierte Befehlsbeschreibungen angezeigt werden:

```
(%i3) apropos('plot);
(%o3) [plot, plot2d, plot2dopen, plot2d_ps, plot3d, plotheight, plotmode,
      plotting, plot_format, plot_options]
(%i4) describe("plot");
```

Manchmal fragt `describe` auch nach, welcher Teilbereich beschrieben werden soll (hier nur ein kleiner Ausschnitt der angezeigten Informationen):

```
0: (maxima.info)Plotting.
1: Definitions for Plotting.
2: OPENPLOT_CURVES :Definitions for Plotting.
3: PLOT2D :Definitions for Plotting.
4: PLOT2D_PS :Definitions for Plotting.
5: PLOT3D :Definitions for Plotting.
6: PLOT_OPTIONS :Definitions for Plotting.
7: SET_PLOT_OPTION :Definitions for Plotting.
Enter n, all, none, or multiple choices eg 1 3 : 5
Info from file /usr/share/info/maxima.info:PLOT3D
(expr,xrange,yrange,...,options,..)
-- Function: PLOT3D ([expr1,expr2,expr3],xrange,yrange,...,options,..)
   plot3d(2^(-u^2+v^2),[u,-5,5],[v,-7,7]);
   would plot z = 2^(-u^2+v^2) with u and v varying in [-5,5] and
   [-7,7] respectively, and with u on the x axis, and v on the y axis.

An example of the second pattern of arguments is
   plot3d([cos(x)*(3+y*cos(x/2)),sin(x)*(3+y*cos(x/2)),y*sin(x/2)],
   [x,-%pi,%pi],[y,-1,1],[grid,50,15])

which will plot a moebius band, parametrized by the 3 expressions
given as the first argument to plot3d. An additional optional
argument [grid,50,15] gives the grid number of rectangles in the x
direction and y direction.
....
```

Mit der Funktion `example` können Beispiele zu einigen Funktionen von Maxima angezeigt werden (hier gekürzt):

```
(%i3) example(integrate);

(%i4) test(f):=block([u],u:integrate(f,x),ratsimp(f-diff(u,x)))
(%o4) test(f) := block([u], u : integrate(f, x), ratsimp(f - diff(u, x)))
(%i5) test(sin(x))
(%o5) 0
(%i6) test(1/(x+1))
```

```
(%o6)                                0
(%i7) test(1/(x^2+1))
(%o7)                                0
(%i8) integrate(sin(x)^3,x)
                                     3
                                     cos (x)
(%o8)  ----- - cos(x)
                                     3
...

```

2.6. Darstellung der Ergebnisse

Die Darstellung der Ergebnisse von Maxima, ist im Wesentlichen von der verwendeten Oberfläche abhängig. Während die Ausgabe auf der Konsole und im einfachen Emacs-Modus auf die Darstellung von ASCII Zeichen begrenzt ist, zeigen der erweiterte Emacs-Modus, Imaxima, TexMacs und WxMaxima die Ergebnisse in grafischer Form an. D. h. es werden entsprechende Symbole für Pi, Integral, Summe usw. verwendet. Allgemein zeichnen sich die Ausgaben von Maxima durch exakte (rationale) Arithmetik aus:

```
(%i38) 1/11 + 9/11;
(%o38)
          10
          --
          11

```

Irrationale Zahlen werden in ihrer symbolischen Form beibehalten (mit % wurde auf das Ergebnis der letzten Berechnung zugegriffen):

```
(%i39) (sqrt(3) - 1)^4;
(%o39)
          4
          (sqrt(3) - 1)
(%i40) expand(%);
(%o40)
          28 - 16 sqrt(3)

```

Mit `ev(Ausdruck, numer);` oder kurz: `Ausdruck, numer;` oder `float(Ausdruck)` kann eine Dezimaldarstellung erzwungen werden (beachten Sie hier die Referenz auf das vorangegangene Ergebnis Nr. 40 via %o40):

```
(%i41) %o40, numer;
(%o41)
          0.28718707889796
(%i5) float(%e);
(%o5)
          2.718281828459045

```

Die Voreinstellung der Genauigkeit bei Fließkommazahlen beträgt 16 Stellen, wobei die letzte Stelle unsicher ist. Die Genauigkeit kann beliebig eingestellt werden, wenn der Zahlentyp `bfloat` verwendet wird. Die Anzahl der angezeigten Stellen wird mit `fpprec` gesteuert. Man kann dazu `fpprec` nur für die Auswertung einer Zeile setzen, wie dies in Zeile `%i46` geschieht, oder für alle folgenden Berechnungen setzen, wie dies in Zeile `%i48` geschieht:

```
(%i45) bfloat(%o40);

(%o45)                2.871870788979631B-1
(%i46) bfloat(%o40), fpprec=100;

(%o46) 2.871870788979633035608585359060421289151159390339099511070883287690717#
294591994067016610118840227891B-1
(%i47) (%i47) fpprec;

(%o47)                16
(%i48) fpprec : 20;

(%o48)                20
(%i49) bfloat(%o40);

(%o49)                2.8718707889796330358B-1
(%i50) (%i50) bfloat(%o40), fpprec=100;

(%o50) 2.871870788979633035608585359060421289151159390339099511070883287690717#
294591994067016610118840227891B-1
```

Die Eingabe von bestimmten Konstanten (`e`, `i`, `pi`, ...) erfolgt mit vorangestelltem `%` (`%e`, `%i`, `%pi`...).

Die Darstellung von bestimmten Konstanten (`e`, `i`, `pi`, ...), Operatoren (Summe, Integral, Ableitungen, ...) und anderen Symbolen (Klammern, Brüche, ...) ist abhängig von der gewählten Oberfläche. Im Textmodus, der von der Konsole, dem einfachen Emacs-Modus und der mitgelieferten `xmaxima` Oberfläche geboten wird, werden Konstanten mit einem `%` vorangestellt (`%e`, `%i`, `%pi`, ...), Operatoren werden in ASCII-Grafik dargestellt, Klammern werden nicht in der Größe expandiert und Brüche mit Hilfe von `-` dargestellt:

```
(%i51) sqrt(-3);

(%o51)                sqrt(3) %i
(%i52) exp(5 * a);

(%o52)                5 a
                    %e

(%i54) (%i54) integrate( f(x) , x, 0, inf);

(%o54)                inf
                    /
                    [
                    I    f(x) dx
                    ]
                    /
```

0

Die Darstellung im erweiterten Emacs-Modus, in IMaxima und in TexMax ist grafisch, d. h. für π , e , Integrale, Summen, ... werden entsprechende Symbole verwendet.

Maxima kann natürlich auch Funktionen plotten. Die Funktion

`plot2d([Funktionsliste], [X-Var, Min, Max], [Y-Var, Min, Max]);`
kann eine Gruppe von Funktionen plotten, hierzu gibt man die Liste von Funktionen in eckigen Klammern und durch Kommas getrennt als ersten Parameter an, es folgt eine Liste, welche die abhängige Variable und den Plotbereich (x-Achse) angibt. Es gibt noch viele andere Plotmöglichkeiten (`apropos('plot)`, `describe(plot)`)

```
(%i55) plot2d([sin(x), cos(x)], [x, 0, 5]);
```

```
(%i58) apropos('plot);
```

```
(%o58) [plot, plot2d, plot2dopen, plot2d_ps, plot3d, plotheight, plotmode,
        plotting, plot_format, plot_options]
```

```
(%i59) (%i59) describe(plot);
```

```
0: (maxima.info)Plotting.
```

```
1: Definitions for Plotting.
```

```
2: openplot_curves :Definitions for Plotting.
```

```
3: plot2d :Definitions for Plotting.
```

```
4: plot2d_ps :Definitions for Plotting.
```

```
5: plot3d :Definitions for Plotting.
```

```
6: plot_options :Definitions for Plotting.
```

```
7: set_plot_option :Definitions for Plotting.
```

```
Enter space-separated numbers, `all' or `none': Enter space-separated numbers,
`all' or `none': none
```


3. Rechnen mit Maxima

3.1. Operatoren

Die üblichen arithmetischen Operatoren stehen zur Verfügung:

- + Addition,
- - Subtraktion,
- * skalare Multiplikation,
- . Slalarmultiplikation von Vektoren und Matrix-Multiplikation,
- / Division,
- ** oder ^ Potenzfunktion,
- sqrt() Wurzelfunktion,
- exp() Exponentialfunktion,
- log() natürliche Logarithmusfunktion
- ...

3.2. Algebra

Niemand ist vor Leichtsinnsfehlern bei der Umformung, Transformation usw. von algebraischen Ausdrücken gefeit. Hier eignet sich Maxima hervorragend bei der Unterstützung analytischer Berechnungen.

Hier ein einfaches Beispiel für die Behandlung von Polynomen. Zunächst wird via `expand` Expandiert, anschließend eine Ersetzung vorgenommen, danach mittels `ratsimp` ein gemeinsamer Nenner gesucht und anschließend via `factor` Faktorisiert:

```
(%i72) (5*a + 3*a*b )^3;
```

```
(%o72) (3 a b + 5 a)3
```

```
(%i73) expand(%);
```

```
(%o73) 27 a3 b3 + 135 a3 b2 + 225 a3 b + 125 a3
```

```
(%i74) %, a=1/x;
```

```
(%o74) 
$$\frac{27 b^3}{x^3} + \frac{135 b^2}{x^3} + \frac{225 b}{x^3} + \frac{125}{x^3}$$

```

```
(%i75) (%i75) ratsimp(%);
```

```
(%o75) 27 b3 + 135 b2 + 225 b + 125
```

```
(%o75) -----
              3
             x

(%i76) factor(%);

              3
          (3 b + 5)
          -----
              3
             x
```

3.3. Lösen von Gleichungssystemen

Maxima ist in der Lage, exakte Lösungen auch von nichtlinearen algebraischen Gleichungssystemen zu berechnen. Im folgenden Beispiel werden 3 Gleichungen nach drei Unbekannten aufgelöst:

```
(%i77) eq1: a + b + c = 6;

(%o77)          c + b + a = 6
(%i78) eq2: a * b + c = 5;

(%o78)          c + a b = 5
(%i79) eq3: a + b * c = 7;

(%o79)          b c + a = 7

(%i93) s: solve([eq1, eq2, eq3], [a, b, c]);

(%o93)          [[a = 1, b = 3, c = 2], [a = 1, b = 2, c = 3]]
```

Die Lösung wird in Form einer Liste einer Liste in eckigen Klammern dargestellt. Im Folgenden wird gezeigt, wie auf die Elemente einer Liste zugegriffen werden kann und wie man die Lösungen in andere Gleichungen einsetzen kann.

```
(%i94) (%i94) s[1];

(%o94)          [a = 1, b = 3, c = 2]

(%i95) s[2];

(%o95)          [a = 1, b = 2, c = 3]

(%i98) eq4: a * a + 2 * b * b + c * c;

(%o98)          2      2      2
          c  + 2 b  + a
(%i99) (%i99) eq4, s[1];

(%o99)          23
(%i100) eq4, s[2];
```

```
(%o100)                                     18
(%i101)
```

3.4. Trigonometrische Funktionen

Es stehen u.a. tan, sin, cos, tanh, sinh, cosh und deren Umkehrfunktionen zur Verfügung.

```
(%i102) example(trig);

(%i103) tan(%pi/6)+sin(%pi/12)
(%o103)

$$\sin\left(\frac{\pi}{12}\right) + \frac{1}{\sqrt{3}}$$

(%i104) ev(% ,numer)
(%o104) 0.83616931429214658
(%i105) sin(1)
(%o105)

$$\sin(1)$$

(%i106) ev(sin(1),numer)
(%o106) 0.8414709848078965
(%i107) beta(1/2,2/5)
(%o107)

$$\beta\left(-, -\right)$$

(%i108) ev(% ,numer)
(%o108) 3.6790939804058804
(%i109) diff(atanh(sqrt(x)),x)
(%o109)

$$\frac{1}{2(1-x)\sqrt{x}}$$

(%i110) fpprec:25
(%i111) sin(5.0B-1)
(%o111) 4.794255386042030002732879B-1
(%i112) cos(x)^2-sin(x)^2
(%o112)

$$\cos^2(x) - \sin^2(x)$$

(%i113) ev(% ,x:%pi/3)
(%o113)

$$-\frac{1}{2}$$

(%i114) diff(%th(2),x)
(%o114)

$$-4 \cos(x) \sin(x)$$

(%i115) integrate(%th(3),x)
(%o115)

$$\frac{\sin(2x)}{2} + x^2 - \frac{\sin(2x)}{2}$$

(%i116) expand(% )
(%o116)

$$\frac{\sin(2x)}{2}$$

```

Trigonometrische Ausdrücke lassen sich in Maxima leicht manipulieren. Die Funktion `trigexpand` benutzt die Summe-der-Winkel-Funktion, um Argumente innerhalb jeder trigonometrischen Funktion so stark wie möglich zu vereinfachen.

```
(%o116)          sin(2 x)
                -----
                 2
(%i117) trigexpand(%)
(%o117)          cos(x) sin(x)
```

Die Funktion `trigreduce`, konvertiert einen Ausdruck in eine Form, welche eine Summe von Einzeltermen, bestehend aus jeweils einer sin- oder cos- Funktion.

```
(%o117)          cos(x) sin(x)
```

```
(%i118) trigreduce(%)
```

```
(%o118)          sin(2 x)
                -----
                 2
```

```
(%i119) sech(x)^2*tanh(x)/coth(x)^2+cosh(x)^2*sech(x)^2*tanh(x)/coth(x)^2
        +sech(x)^2*sinh(x)*tanh(x)/coth(x)^2
```

```
(%o119)          2          2          2          2          2
        sech (x) sinh(x) tanh(x)  cosh (x) sech (x) tanh(x)  sech (x) tanh(x)
        ----- + ----- + -----
                2          2          2
                coth (x)      coth (x)      coth (x)
```

(`sech()` ist eine hyperbolische Sekantenfunktion.) `trigsimp` ist eine Vereinfachungsroutine, welche verschiedene trigonometrische Funktionen in sin und cos Equivalente umwandelt.

```
(%i120) trigsimp(%)
```

```
(%o120)          5          4          3
        sinh (x) + sinh (x) + 2 sinh (x)
        -----
                 5
                cosh (x)
```

`exponentialize()` transformiert trigonometrische Funktionen in ihre komplexen Exponentialfunktionen.

```
(%i121) ev(sin(x),exponentialize)
```

```
(%o121)          %i x      - %i x
        %i (%e      - %e      )
        -----
                 2
```

`taylor(Funktion, Variable, Entwicklungspunkt, Grad);` transformiert trigonometrische Funktionen in ihre komplexen Exponentialfunktionen.

```
(%i122) taylor(sin(x)/x,x,0,4)
```

```
(%o122)/T/
          2      4
         x      x
1 - --- + --- + . . .
   6      120
```

```
(%i123) ev(cos(x)^2-sin(x)^2,sin(x)^2 = 1-cos(x)^2)
```

```
(%o123)
          2
2 cos (x) - 1
```

```
(%o123) done
```

3.5. Komplexe Zahlen

Die Funktionen `realpart` und `imagpart` geben den Real- bzw. Imaginärteil eines komplexen Ausdruckes zurück.

```
(%i144) z: a + b * %i;
```

```
(%o144) %i b + a
```

```
(%i145) z^2;
```

```
(%o145)
          2
(%i b + a)
```

```
(%i146) exp(z);
```

```
(%o146)
          %i b + a
%e
```

`trigrat()` transformiert (komplexe) Exponentialfunktionen in entsprechende `sin()` und `cos()` Funktionen um.

```
(%i148) trigrat(exp(z));
```

```
(%o148)
          a          a
%i %e sin(b) + %e cos(b)
```

Komplexe Zahlen lassen sich mit `imagpart()` und `realpart()` in die entsprechenden Real- und Imaginärteile aufspalten:

```
(%i149) imagpart(%);
```

```
(%o149)
          a
%e sin(b)
```

```
(%i150) realpart(%th(2));
```

3.6. Ableitungen, Grenzwerte, Integrale

Mit Maxima lassen sich u. a. Ableitungen, Integrale, Taylorentwicklungen, Grenzwerte, exakte Lösungen gewöhnlicher Differentialgleichungen berechnen.

Zunächst definieren wir ein Symbol f als Funktion von x auf 2 Arten. Beachten Sie die Unterschiede bei der Auswertung der Ableitung.

```
(%i129) f : x^3;
```

```
(%o129)          3
              x
```

```
(%i130) diff(f,x);
```

```
(%o130)          2
              3 x
```

```
(%i131) kill(f);
```

```
(%o131)          done
```

```
(%i132) f(x) := x^3;
```

```
(%o132)          3
              f(x) := x
```

```
(%i133) diff(f,x);
```

```
(%o133)          0
```

```
(%i134) diff(f(x),x);
```

```
(%o134)          2
              3 x
```

Ein Beispiel für eine Taylorreihenentwicklung und eine Grenzwertberechnung:

```
(%i140) f(x) := sin(x) / x;
```

```
(%o140)          sin(x)
              f(x) := -----
                      x
```

```
(%i141) taylor(f(x),x,0,5);
```

```
(%o141)/T/          2      4
                  x      x
              1 - -- + --- + . . .
                  6      120
```

```
(%i142) limit(f(x),x,0);
```

```
(%o142)          1
```

Integrale lassen sich, sofern möglich, bestimmt und unbestimmt berechnen:

```
(%i109) integrate(%e^x/(2+%e^x),x)
```

```
(%o109)          x
              log(%e  + 2)
```

```
(%i116) integrate(x^(5/4)/(1+x)^(5/2),x,0,inf)
```

```
(%o116)          9  1
                beta(-, -)
                4  4
```

3.7. Lösen von Differentialgleichungen

Ableitungen, bzw. Differentiale können so eingegeben werden, dass sie ausgewertet oder nicht ausgewertet werden. Das Apostroph wirkt als Maskierung und verhindert die Auswertung durch Maxima:

```
(%i9) 'diff (y, x);
(%o9)          dy
              --
              dx
(%i10) diff (y, x);
(%o10)          0
```

Zum Lösen von gewöhnlichen Differentialgleichungen stehen folgende Funktionen zur Verfügung: `ode2`, `ic1`, `ic2`, `bc1`, `bc2`.

`ic1`, `ic2` sind auf Anfangswertaufgaben 1. bzw. 2. Ordnung spezialisiert.

`bc1`, `bc2` sind auf Randwertaufgaben 1. bzw. 2. Ordnung spezialisiert.

```
(%i23) dgl1: -'diff(y,x) * sin(x) + y * cos(x) = 1;
```

```
(%o23)          cos(x) y - sin(x)  $\frac{dy}{dx}$  = 1
```

```
(%i24) ode2( dgl1, y, x);
```

```
(%o24)          y = sin(x)  $\left(\frac{1}{\tan(x)} + \%c\right)$ 
```

```
(%i25) (%i25) trigsimp(%);
```

```
(%o25)          y = %c sin(x) + cos(x)
```

Anfangswertaufgabe: Harmonische Schwingungen z.B. eines Pendels werden durch folgende Differentialgleichung beschrieben und mittels `ode2` und `ic2` gelöst:

```
(%i38) dgl2: 'diff(y,x,2) + y = 0;
```

```
(%o38)           $\frac{d^2 y}{dx^2} + y = 0$ 
```

```
(%i39) ode2(dgl2, y, x);
```

```
(%o39)          y = %k1 sin(x) + %k2 cos(x)
```

```
(%i40) ic2(% , x=0, y=y0, 'diff(y,x)=0);
```

```
(%o40) y = cos(x) y0
```

Randwertaufgabe: Bei gleichmäßiger Belastung, lässt sich die Biegelinie eines ruhenden und auf 2 Säulen liegenden Balkens unter bestimmten Umständen durch folgende Differentialgleichung beschreiben und mittels `ode2` und `bc2` lösen:

```
(%i41) dgl3: 'diff(y,x,2) = x - x^2;
```

```
(%o41)      2
            d y      2
            --- = x - x
            2
            dx
```

```
(%i42) ode2(dgl3, y, x);
```

```
(%o42)      4      3
            x  - 2 x
            y = - ---- + %k2 x + %k1
                12
```

```
(%i43) bc2(%, x=0, y=0, x=1, y=0);
```

```
(%o43)      4      3      x
            x  - 2 x  - x
            y = - ---- - ---
                12      12
```

```
(%i45) expand(%);
```

```
(%o45)      4      3      x
            x  - x  - x
            y = - --- + --- - ---
                12      6      12
```

3.8. Matrizenrechnung

Mit Maxima lassen sich allgemeine Matrizenoperationen durchführen.

```
(%i79) m:matrix([a,0],[b,1])
```

```
(%o79)      [ a  0 ]
            [   ]
            [ b  1 ]
```

```
(%i80) m^2
```

```
(%o80)      [  2   ]
            [ a  0 ]
            [   ]
            [  2   ]
            [ b  1 ]
```

```
(%i81) m . m
```

```
(%o81)      [  2   ]
            [ a  0 ]
            [   ]
            [ a b + b 1 ]
```

```
(%i82) m[1,1]*m
```


3. Rechnen mit Maxima

```

(%o82)          [ 2      ]
                [ a      0 ]
                [          ]
                [ a b   a ]

(%i83) 1-%th(2)+%
(%o83)          [ 1      1 ]
                [          ]
                [ 1 - b  a ]

(%i84) m^(-1)
(%o84)          [ 1      ]
                [ -      0 ]
                [ a      ]
                [          ]
                [ b      ]
                [ - - 1 ]
                [ a      ]

(%i85) [x,y] . m
(%o85)          [ b y + a x  y ]

(%i86) matrix([a,b,c],[d,e,f],[g,h,i])
(%o86)          [ a  b  c ]
                [          ]
                [ d  e  f ]
                [          ]
                [ g  h  i ]

(%i87) %^2
(%o87)          [          2          ]
                [ c g + b d + a      c h + b e + a b  c i + b f + a c ]
                [          ]
                [          2          ]
                [ f g + d e + a d  f h + e      + b d  f i + e f + c d ]
                [          ]
                [          2          ]
                [ g i + d h + a g  h i + e h + b g  i      + f h + c g ]
(%o87)          done

```

Außerdem lassen sich u. a. die Determinante, die Inverse, die Eigenwerte und Eigenvektoren einer Matrix berechnen. Die Matrix darf dabei auch symbolische Ausdrücke enthalten.

`eigenvalues(m)` ergibt als Ergebnis eine Liste, bestehend aus 2 Unterlisten. Die erste Unterliste enthält die Eigenwerte, die 2. Unterliste die entsprechenden Multipliktoren.

```

(%i60) m : matrix( [1, 0, 0], [0, 2, 0], [0, 0, 3]);
(%o60)          [ 1  0  0 ]
                [          ]
                [ 0  2  0 ]
                [          ]
                [ 0  0  3 ]

(%i61) eigenvalues(m);
(%o61)          [[1, 2, 3], [1, 1, 1]]

```

3. Rechnen mit Maxima

Die Funktion `eigenvectors` berechnet Eigenwerte, deren Multiplikatoren, sowie die Eigenvektoren der gegebenen Matrix. Die Ergebnisse werden in Listen, bzw. Unterlisten zusammengefasst. Es gibt verschiedene Möglichkeiten die Auswertung zu beeinflussen `nondiagonalizable`, `hermitianmatrix`, `knoweigvals`, diese werden mit `describe(eigenvectors)`; beschrieben.

```
(%i62) eigenvectors(m);
```

```
(%o62)      [[1, 2, 3], [1, 1, 1]], [1, 0, 0], [0, 1, 0], [0, 0, 1]]
```

```
(%i71) part( %, 2);
```

```
(%o71)      [1, 0, 0]
```

Weiterhin gibt es Funktionen zur Transponierung (`transpose`), Berechnung der Determinante (`determinant`), Berechnung des charakteristischen Polynomes `charpoly(Matrix, Variable)`; , Berechnung der Inversen (`invert`). Das Schlüsselwort `detout` faktorisiert dabei die Determinante aus der Inversen.

4. Programmieren in Maxima

Bis jetzt haben Sie gesehen, wie man Maxima im interaktiven Modus wie einen Taschenrechner benutzt. Für Berechnungen, welche wiederholt Kommandosequenzen durchlaufen müssen, sind Programme geeigneter.

Programme werden gewöhnlich in einem Texteditor (Emacs) geschrieben und dann mittels `batch` in Maxima geladen.

Ein kleines Statistik-Münzwurfprogramm. Mit `block([Lokale_Variablen], Kommando1, Kommando2...)` wird ein Block von Kommandos definiert. Die lokalen Variablen können auch mit Startwerten versehen werden, wie im Folgenden gezeigt wird:

```
(%i13) muenze_werfen(n) := block( [muenze, statistik:[0,0] ], /* Kommentare wie
in C/C++ */
    print("Ich werde die Münze ", n, "mal werfen. 1 = Zahl, 2 = Kopf"),
    for i: 1 thru n do ( /* eine for schleife, für genauere informationen
siehe DO */
        muenze : random(2) + 1, /* 1 + Zufallsgenerator von 0 bis 1 */
        statistik[muenze] : statistik[muenze] + 1 ), /* Array-Indizes beginnen
mit EINS! */
    print("Zahl wurde ", statistik[1], "mal geworfen"),
    print("Kopf wurde ", statistik[2], "mal geworfen"),
    n );
```

```
(%o13) muenze_werfen(n) := block([muenze, statistik : [0, 0]],
print("Ich werde die Münze ", n, "mal werfen. 1 = Zahl, 2 = Kopf"),
for i thru n do (muenze : random(2) + 1,
statistik
: statistik
+ 1),
muenze
muenze

print("Zahl wurde ", statistik , "mal geworfen"),
1

print("Kopf wurde ", statistik , "mal geworfen"), n)
2

(%i14) muenze_werfen(1000);
```

```
Ich werde die Münze 1000 mal werfen. 1 = Zahl, 2 = Kopf
Zahl wurde 485 mal geworfen
Kopf wurde 515 mal geworfen
(%o14) 1000
```

Dieselben Ausgaben in die Datei `Daten.txt` umlenken:

```
muenze_werfen(n) := block( [muenze, statistik:[0,0] ], /* Kommentare wie in
C/C++ */
    with_stdout( "Daten.txt",
        print("Ich werde die Münze ", n, "mal werfen. 1 = Zahl, 2 = Kopf"),
        for i: 1 thru n do ( /* eine for schleife, für genauere informationen siehe
DO */
            muenze : random(2) + 1, /* 1 + Zufallsgenerator von 0 bis 1 */
            statistik[muenze] : statistik[muenze] + 1 ), /* Array-Indizes beginnen mit
```

4. Programmieren in Maxima

```
EINS! */
  print("Zahl wurde ", statistik[1], "mal geworfen"),
  print("Kopf wurde ", statistik[2], "mal geworfen")
),
n
);
```

5. Wichtige Maxima-Funktionen

Die Referenz befindet sich unter `doc/html/maxima_toc.html` (im Installationspfad von Maxima). In Maxima können Sie Hilfe durch die Kommandos `describe`, `apropos` und `example` erhalten. `apropos(Stichwort)` gibt eine Liste von evtl. geeigneten Kommandos. `describe(Befehl)` beschreibt die einzelnen Kommandos genauer (evtl. müssen Sie eine Auswahl treffen, welcher Aspekt genauer beschrieben werden soll) und `example(Befehl)` gibt Beispiele für den jeweiligen Befehl aus (soweit vorhanden).

`allroots(a)` Findet alle (allgemein komplexen) Wurzeln einer Polynomialgleichung.

`append(a,b)` Fügt Liste `b` an `a` an.

`apropos(a)`; Liefert zu einem Stichwort mögliche Befehle/Funktionen.

`batch(a)` Lädt und startet Programm/File `a`.

`bc1, bc2` (DGL, $x=x_0$, $y=y_0$, $x=x_1$, $y=y_1$) Lösung einer Randwertaufgabe einer DGL nach Behandlung mit `ode2`.

`charpoly(Matrix, Variable)` Berechnet das charakteristische Polynom einer Matrix bzgl. der gegebenen Variable.

`coeff(a,b,c)` Koeffizienten von `b` der Potenz `C` in Ausdruck `a`.

`concat(a,b)` Generiert ein Symbol `ab`.

`cons(a,b)` Fügt `a` in Liste `b` als erstes Element ein.

`demoivre(a)` Transformiert alle komplexen Exponentialterme in trigonometrische.

`denom(a)` Nenner von `a`.

`depends(a,b)` Erklärt `a` als Funktion von `b` (nützlich für Differentialgleichungen).

`desolve(a,b)` Versucht ein lineares System `a` von gew. DGLs nach unbekanntem `b` mittels Laplace-Transformation zu lösen.

`describe(a)` Beschreibt einen Befehl oder eine Funktion näher. Evtl. wird nachgefragt, welcher Aspekt eines Befehls oder einer Befehlsgruppe näher beschrieben werden soll.

`determinant(a)` Determinante

`diff(a,b1,c1,b2,c2,...,bn,cn)` Gemischte partielle Ableitung von `a` nach `bi` der Stufe `ci`.

`eigenvalues(a)` Berechnet die Eigenwerte und ihre Multiplikatoren.

`eigenvectors(a)` Berechnet Eigenvektoren, Eigenwerte und Multiplikatoren.

`entermatrix(a,b)` Matriceingabe

`ev(a,b1,b2,...,bn)` Berechnet Ausdruck `a` unter Annahmen `bi` (Gleichungen, Zuweisungen, Schlüsselwörter (numer - Zahlenwerte, detout - Matrixinverse ohne Determinante, diff - alle Ableitungen werden ausgeführt). NUR bei direkter Eingabe kann `ev` weggelassen werden.

`example(a)` Zeigt Beispiele für die Verwendung eines Befehls oder einer Funktion an. Nicht für jede Funktion sind Beispiele vorhanden.

`expand(a)` Algebraische Expansion (Distribution).

`exponentialize(a)` Transformiert trigonometrische Funktionen in ihre komplexen Exponentialfunktionen.

5. Wichtige Maxima-Funktionen

`factor(a)` Faktorisiert a .

`freeof(a,b)` Ergibt wahr, wenn b nicht a enthält.

`grind(a)` Darstellung einer Variable oder Funktion in einer kompakten Form.

`ic1, ic2 (dgl, x=x0, y=y0, dy0/dx = y1)` Lösung einer Anfangswertaufgabe einer DGL (nach Behandlung mit `ode2`).

`ident(a)` Einheitsmatrix $a \times a$.

`imagpart(a)` Imaginärteil von a .

`integrate(a,b)` Berechnungsversuch des unbestimmten Integrals a nach b .

`integrate(a,b,c,d)` Berechnung des Integrals a nach b in den Grenzen $b=c$ und $b=d$.

`invert(a)` Inverse der Matrix a .

`kill(a)` Vernichtet Variable/Symbol a .

`limit(a,b,c)` Grenzwertbestimmung des Ausdrucks a für b gegen c .

`lhs(a)` Linke Seite eines Ausdrucks.

`loadfile(a)` Lädt eine Datei a und führt sie aus.

`makelist(a,b,c,d)` Generiert eine Liste von $a(b)$ mit $b=c$ bis $b=d$.

`map(a,b)` Wendet a auf b an.

`matrix(a1,a2,...,an)` Generiert eine Matrix aus Zeilenvektoren.

`num(a)` Zähler von a .

`ode2(a,b,c)` Löst gewöhnliche Differentialgleichungen 1. und 2. Ordnung a für b als Funktion von c .

`part(a,b1,...,bn)` Extrahiert aus a die Teile b_i .

`playback(a)` Zeigt die a letzten Labels an, wird a weggelassen, so werden alle Zeilen zurückgespielt.

`print(a1, a2, a3, ...)` Zeigt die Auswertung der Ausdrücke an.

`ratsimp(a)` Vereinfacht a und gibt einen Quotienten zweier Polynome zurück.

`realpart(a)` Realteil von a

`rhs(a)` Rechte Seite einer Gleichung a .

`save(a,b1,..., bn)` Generiert eine Datei a (im Standardverzeichnis), welche Variablen, Funktionen oder Arrays b_i enthält. So generierte Dateien lassen sich mit `loadfile` zurückspielen. Wenn $b1$ `all` ist, wird alles bis auf die Labels gespeichert.

`solve(a,b)` Algebraischer Lösungsversuch für ein Gleichungssystem oder eine Gleichung a für eine Variable oder eine Liste von Variablen b . Gleichungen können $=0$ abkürzen.

`string(a)` Konvertiert a in Maximas lineare Notation.

`stringout(a,b1,...bn)` Generiert eine Datei a im Standardverzeichnis, bestehend aus Symbolen b_i . Die Datei ist im Textformat und nicht dazu geeignet von Maxima geladen zu werden. Die Ausdrücke können aber genutzt werden, um sie in Fortran, Basic oder C-Programmen zu verwenden.

`subst(a,b,c)` Ersetzt a für b in c .

`taylor(a,b,c,d)` Taylorreihenentwicklung von a nach b in Punkt c bis zur Ordnung d .

`transpose(a)` Transponiert Matrix a .

`trigexpand(a)` Eine Vereinfachungsroutine, welche trigonometrische Winkelsummen nutzt, um einen Ausdruck zu vereinfachen.

`trigreduce(a)` Eine Vereinfachungsroutine für trigonometrische Produkte und

Potenzen.

`trigsimp(a)` Eine Vereinfachungsroutine, welche verschiedene trigonometrische Funktionen in sin und cos Equivalente umwandelt.

`with_stdout(Datei, Ausdrücke);` Leitet die Ausgabe der Ausdrücke in die angegebene Datei um.