Maxima Referenz

Erste Schritte

`;`
: Befehlsende (mit Echo)

`$`
: Befehlsende (ohne Echo)

`%`
: Platzhalter für das letztes Resultat

`%i`*n*
: Platzhalter für den Input mit der Nummer *n*

`%o`*n*
: Platzhalter für den Output mit der Nummer *n*

`Help/Maxima Manual`
: Hilfefunktion (nur in xMaxima)

`Edit/Save Console to File`
: xMaxima-Session speichern

`Control-G`
: laufende Rechnung beenden

`load(`*Dateiname*`)`
: Die Datei *Dateiname* laden

`File/Restart`
: Maxima zurücksetzen

Arithmetik

`+`
: Addition

`-`
: Subtraktion

`*`
: Multiplikation

`/`
: Division

`^ oder **`
: Potenzieren

`sqrt(`*Radikand*`)`
: Quadratwurzel

`abs(`*`Ausdruck`*`)`
: Absolutbetrag von *`Ausdruck`*

`float(`*`Ausdruck`*`)`
: *`Ausdruck`* als Fliesskommazahl darstellen

`fpprec : `*`n`*
: Floating point precision = *`n`* Stellen

`bfloat(`*`Ausdruck`*`)`
: *`Ausdruck`* als Bigfloat-Zahl darstellen

`sin(`*`x`*`)`
: Sinus des Winkels *`x`* (im Bogenmass)

`cos(`*`x`*`)`
: Cosinus des Winkels *`x`* (im Bogenmass)

`tan(`*`x`*`)`
: Tangens des Winkels *`x`* (im Bogenmass)

`asin(`*`x`*`)`
: Arkussinus von *`x`* (Ergebnis im Bogenmass)

`acos(x)`
: Arkuscosinus von *`x`* (Ergebnis im Bogenmass)

`atan(x)`
: Arkustangens von *`x`* (Ergebnis im Bogenmass)

`log(`*`x`*`)`
: Natürlicher Logarithums (Logarithmus zur Basis e)

Zahlentheorie

`n``!`
: Fakultät von *`n`*

`binomial(`*`n`*`, `*`k`*`)`
: Binomialkoeffizient $\binom{n}{k}$

`factor(`*`n`*`)`
: Primfaktorzerlegung von *`n`*

`gcd(`*`a`*`, `*`b`*`)`
: Grösster gemeinsamer Teiler von *`a`* und *`b`*

`lcm(`*`a`*`, `*`b`*`)`
: Kleinstes gemeinsames Vielfaches von *`a`* und *`b`*.
Hinweis: `lcm` ist erst nach der Eingabe von `load(functs);` verfügbar.

`mod(`*`Divisor`*`,`*`Dividend`*`)`
: Divisionsrest

`primep(n)`
: ist *n* eine Primzahl? (wahr/falsch)

`prev_prime(n)`
: grösste Primzahl kleiner als *n*

`next_prime(n)`
: kleinste Primzahl grösser als *n*

Algebra

`ratsimp(Ausdruck)`
: versucht, den Quotienten *Ausdruck* zu vereinfachen

`factor(Ausdruck)`
: versucht, *Ausdruck* in Faktoren zu zerlegen

`expand(Ausdruck)`
: ausmultiplizieren der Summen in *Ausdruck*

Variablen und Konstanten

Variable `:` *Ausdruck*
: Ordnet *Variable* den Wert von *Ausdruck* zu

`values`
: Variable, die alle vom Benutzer aktuell definierten Variablen enthält.

`kill(`*Var1*`,` *Var2*`,...)`
: löscht die Bindungen der Variablen *Var1*, *Var2*, ...

`kill(all)`
: löscht die Bindungen aller definierten Variablen

`%pi`
: Symbol für die Kreiszahl π

`%e`
: Symbol für die eulersche Zahl e

`%i`
: Symbol für die imaginäre Einheit i

`%inf`
: Symbol für ∞

`subst([`*Var1=Wert1,Var2=Wert2,...*`],` *Ausdruck*`)`
: Ersetzt die Variablen *Var1*, *Var2*, ... in *Ausdruck* *einmalig* durch die Werte *Wert1*, *Wert2*, ...

Gleichungen und Gleichungssysteme

Ausdruck1 = *Ausdruck2*
: Gleichungsobjekt

solve(*Gleichung*,*Variable*)
: löst die algebraische *Gleichung* nach *Variable* auf

allroots(*Gleichung*)
: Näherungslösungen der algebraischen *Gleichung*

solve([*Glg1*,*Glg2*,...],[*Var1*,*Var2*,...])
: löst das System aus *Glg1*, *Glg2*, ... nach *Var1*, *Var2*, ... auf

%*rn*
: Platzhalter für eine freie Variable bei der Lösung von Gleichungssystemen

Funktionen

Funktionsname(*Var1*,*Var2*,...) := *Ausdruck*
: Definiert die Funktion *Funktionsname* mit den formalen Parametern *Var1*, *Var2*, ...

Funktionsname(*Arg1*,*Arg2*,...)
: wertet die Funktion mit den aktuellen Parametern (Argumenten) *Arg1*, *Arg2*, ... aus

functions
: Variable, die alle vom Benutzer aktuell definierten Funktionen enthält.

Graphen (2d)

plot2d(*Ausdr*, *x_Bereich*, ..., *Optionen*, ...)
plot2d([*Ausdr1*, *Ausdr2*, ...], ..., *Optionen*, ...)
plot2d([*Ausdr1*, *Ausdr2*, ...], *x_Bereich*, ..., *Optionen*, ...)
: Zeichnet den Graphen von einem oder mehreren Ausdrücken als Funktion in einer Variablen. *Ausdr*, *Ausdr1*, *Ausdr2*, ... sind entweder Ausdrücke, Funktionen oder eine Liste der Form

  - [discrete, [*x1*, ..., *xn*], [*y1*, ..., *yn*]]
  - [discrete, [[*x1*, *y1*], ..., [*xn*, *yn*]]]
  - [parametric, *x_Ausdr*, *y_Ausdr*, *t_Bereich*]

Graphik-Optionen

plot_options;
: Zeigt die Graphik-Optionen in der Form [*Grafik-Option*, *Wert(e)*] an.

[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 (Grundeinstellung ist 29)

`[xlabel, "text"]`
Beschriftung der x-Achse bei 2d-Plots

`[ylabel, "text"]`
Beschriftung der y-Achse bei 2d-Plots

`[legend, "text1", "text2",...]`
Legenden für die einzelnen Kurven in einem 2d-Plot

`[style, stil1, stil2, ...]`
Plotstile für die einzelnen Plots in einem 2d-Plot in der Form, wobei jeder *stil* entweder aus dem Schlüsselwort `lines` (Voreinstellung) oder `points` besteht. *stil* kann aber auch eine Liste `[typ, w, c]`, wobei `typ` wieder entweder für `lines` oder `points` steht, `w` die Breite der Line bzw. des Punktes in Pixeln und `c` eine Farbnummer ist. Wird die Farb-Nummer weggelassen, werden die Farben automatisch gewählt.

`[color, farbe1, farbe2, ...]`
Farben für die einzelnen Plots in einem 2d-Plot. Es stehen die Werte `blue`, `red`, `green`, `magenta`, `black`, und `cyan`, zur Auswahl.

`[gnuplot_term, terminal]`
Ausgabeformat Hier eine Auswahl: `png`, `jpg`, `gif`, ...

`[gnuplot_out_file, "Dateiname"]`
Ausgabeziel. Beispiel: `"U:/Eigene Dateien/grafik1.png"`

Verschiedenes

`powerset(M)`
Liefert alle Teilmengen der Menge `M`.

`listify(M)`
Wandelt die Menge `M` in eine Liste mit den entsprechenden Elementen um.

`makelist(Ausdruck,Var,Start,Ende)`
erzeugt eine Liste, wobei die Variable `Var` im `Ausdruck` in Einerschritten von `Start` bis `Ende` „läuft“

`lreduce(f, liste)`
Wendet die Funktion `f` (mit zwei Argumenten und einem Wert) sukzessive auf die Elemente der Liste `liste` an.

`display2d`
Variable zur Beinflussung der Output-Darstellung
`true`: Ausgabe erfolgt mehrzeilig und zentriert (Voreinstellung)
`false`: einzeilige, linksbündige Ausgabe

`powerdisp`
: Variable zur Beinflussung der Darstellung von Polynomen
`true`: Summanden werden nach aufsteigenden Exponenten sortiert.
`false`: Summanden werden nach absteigenden Exponenten sortiert (Voreinstellung).

`lhs(`*`gleichung`*`)`
: Liefert den Ausdruck links des Gleichheitszeichens zurück.

`rhs(`*`gleichung`*`)`
: Liefert den Ausdruck rechts des Gleichheitszeichens zurück.

`liste``[`*`k`*`]`
: Liefert das *`k`*-te Element der Liste *`liste`* zurück.