Maple Kurzreferenz

Sommersemester 2016

Tastaturkürzel

`STRG + T`	Textmodus
`STRG + M`	Mathematischer Eingabemodus
`STRG + J`	Zeile danach einfügen
`STRG + K`	Zeile davor einfügen

Grundlegendes

`;`	Abschluss einer Eingabe zur Ausführung
`:`	Abschluss einer Eingabe zur Ausführung ohne Ausgabe
`%, %%, %%%`	referenziert den letzten, vorletzten und drittletzten Befehl
`#`	Einleitung eines Kommentars
`?`	Aktivierung der Hilfe (z.B. ?evalf)
`:=`	Operator für die Zuweisung von Objekten an Variablen
`restart`	setzt Maple mit allen Variablen zurück
`with(package)`	Laden eines Paketes package mit Zusatzprogrammen
`about(x)`	was ist über x bekannt

Exakte Arithmetik und Gleitkommaarithmetik

`+,-,*,/`	Grundrechenarten
`^,**`	Potenz
`!`	Fakultät
`sqrt(x),root(x,n)`	Quadratwurzel aus x, n-te Wurzel aus x
`exp(x)`	Exponentialfunktion von x
`log(x),ln(x)`	Natürlicher Logarithmus von x
`sin(x),cos(x),tan(x),cot(x)`	trigonometrische Funktionen von x
`abs(x)`	Betrag einer Zahl x
`evalf(expr)`	numerische Auswertung eines Ausdrucks expr
`evalf(expr,n)`	numerische Auswertung von expr auf n Stellen

Substitution, Umformung und Vereinfachung (?simplify, ?combine)

`subs(x=a,expr)`	substituiert im Ausdruck expr die Varable x durch a (Zahl oder Variable)
`eval(expr,x=a)`	evaluiert Ausdruck expr an der Stelle x=a
`simplify(expr)`	allgemeine, unspezifische, unkontrollierte Vereinfachung
`factor(expr)`	Faktorisierung eines Ausdrucks
`combine(expr)`	Zusammenfassen von Termen des Ausdrucks expr
`expand(expr)`	multipliziert die in expr enthaltenen Faktoren aus

Vektoren und Matrizen (?LinearAlgebra, ?Matrix, , ?Vector)

`with(LinearAlgebra)`	fügt das LinearAlgebra Modul hinzu
`y:=Vector([0,1,1])`	definiere Vektor y
`y:=<0,1,1>`	alternative Definition des Vektors y
`A:=Matrix([[1,1,3],[1,1,1],[1,2,1]])`	definiere Matrix A
`A:=<<1,1,1>|<1,1,2>|<3,1,1>>`	alternative Definition der Matrix A (spaltenweise)
`x:=LinearSolve(A,y)`	löse lineares Gleichungssystem Ax=y
`transpose(A), inverse(A), det(A)`	transponierte und inverse Matrix, Determinante von A

Differentiation und Integration (?diff, ?Jacobian)

`with(VectorCalculus,Jacobian)`	fügt das Analysis Modul hinzu
`diff(expr,x)`	Ableitung des algebraischen Ausdrucks expr nach x
`diff~(vec,x)`	Ableitung des Vektors von algebraischen Ausdrücken vec nach x
`Jacobian(vec,x)`	Jakobimatrix des Vektors vec nach Variablenliste
`convert(vec,list)`	konvertiert Vektor vec in Variablenliste
`int(expr,x)`	unbestimmte Integration
`sum(expr,x)`	unbestimmte Summation
`int(expr,x=a..b)`	bestimmte Integration
`sum(expr,x=a..b)`	bestimmte Summation
`limit(expr,x=a)`	Grenzwerte (Limes der Funktion f(x) für x gegen a)

Differentialgleichungen (?dsolve, ?odeplot)

`dgl:=diff(x(t),t)=-5*x(t);`	Definiere Differentialgleichung
`ab:=x(0)=1;`	Definiere Anfangsbedingungen
`sol:=dsolve({dgl, ab},type=numeric);`	Löse Differentialgleichung numerisch
`odeplot(sol,t=0..10);`	Stelle Lösung grafisch dar

Programmierung, Kontrollstrukturen (?if, ?for)

Wird benötigt zur Berechnung der Christoffelsymbole zur Bestimmung der Coriolismatrix

```
k:= 7;
if k = 3 then
  print("k gleich 3");
else
  print("k gleich 3");
fi;
```

IF - Abfrage

```
for i from 6 by 2 to 10 do
  print(i);
od;
```

FOR - Schleife

Grafikausgabe (?plot, ?plot3d)

`with(plots)`	fügt das plots Modul hinzu
`plot(expr(x),x=a..b)`	graphische Darstellung der Funktion expr im Intervall [a,b] (2D-Plot)
`plot([expr1(x),expr2(x)],x=a..b)`	gleichzeitige graphische Darstellung der beiden Funktion expr1 und expr2 im Intervall [a,b]
`plot3d(expr(x,y),x=a..b,y=c..d)`	grafische Darstellung der Funktion expr(x,y) über dem Rechteck [a..b,c..d] (3D-Plot)
`f := plot(expr(x),x=a..b)`	speichert entsprechende Darstellung in Plot-Struktur f1
`display(f1, f2)`	gleichzeitige Darstellung mehrerer Plot-Strukturen