Wichtige Befehle von Mathematica als Ersatz für einen Taschenrechner

Stefan Englert

stefan.englert@gmx.net

26. März 2006

Motivation

Im Laufe meiner bisherigen Schulkariere habe ich schon viele Taschenrechner und Funktionsplotter verwendet. Vor einem halben Jahr bin ich dann auf `Mathematica` aufmerksam geworden und kann behaupten, dass es alle meine bisherigen Erwartungen übertrifft.

Mathematica bietet viele Vorteile, von denen ich einige kurz aufzählen will:

- Als eingeschriebener Student bekommt man dieses Programm meist kostenlos
- Mathematica rechnet symbolisch
- Es kommt selbst mit komplexen Zahlen, unendlichen Reihen und Potenzreihen klar

Kurz gesagt: In Mathematica ist nicht mehr die Frage entscheidend, ob man eine gewünschte Rechenoperation durchführen kann, sonder nur noch *wie*.

Dies hat aber leider zur Folge, dass die Bedienung von Mathematica auf den ersten Blick alles andere als leicht – wahrscheinlich sogar verwirrend ist. Man muss sich eine Vielzahl unterschiedlicher Befehle merken, um Mathematica bedienen zu können.

In diesem Punkt schafft meine Zusammenfassung hoffentlich Abhilfe, da sie aus meiner Sicht wichtige Befehle von Mathematica als graphischer Taschenrechner systematisch zusammenfasst.

Ich setze gewisse Grundkenntnisse über die Bedienung und Struktur von Mathematica voraus. Folglich ist dieses Dokument nicht als Einstieg in Mathematica gedacht.

Inhaltsverzeichnis

1. Einleitung und Glossar

Um die Erklärung der Kommandos von `Mathematica` zu erleichtern, werden zuerst einige Abkürzungen oder Verallgemeinerungen eingeführt.

Diese Einleitung soll gleichzeitig als eine Art Glossar dienen.

Bezeichnung	**Beschreibung**	**Beispiel**
Ausdruck	Eine beliebige Kombination von Kommandos, Parametern und Variablen	`Simplify[x+4]`
Bedingung	Bedingungen, die gewisse Variablen erfüllen	a > 0
data	Eine auf zweiter Ebene verschachtelte Mathematicaliste	`{{1,2},{2,3},{5,6}}`
Esc	Symbolisiert das Drücken der Escape Taste. Wird in Mathematica durch einen gepunkteten senkrechten Strich dargestellt (⁝)	Esc p Esc wird zu π
f	siehe Funktion	
Funktion	Eine von x abhängige mathematische Funktion	`y[x]`
gls	Ein Gleichungssystem oder mehrere Gleichungssysteme in einer Liste	y == x + 4
Kommando	Ein Mathematicabefehl	`Simplify`, `Plot`
liste	Eine im Normalfall unverschachtelte Mathematicaliste	`{1,2,x,x+y}`
mat	Eine Matrix (verschachtelte Liste)	
Parameter	Eine durch den Benutzer mit einer Zeichenkette belegte Variable	x=2 *oder* y=x+4
Typ	Die Art der Zahl	Integer, Real, Complex
Variable	Eine beliebige zuvor nicht definierte Zeichenkette	x *oder* y
Wert	Eine beliebige nicht näher bestimmte Zahl	4 *oder* 3,456
Zahl	Eine natürliche Zahl	2 *oder* 10

In den weiteren Erklärungen enthält die linken Spalte immer eine Beschreibung der Operation, die man durchführen möchte, und die rechten Spalte das dazu gehörige Mathematica-Kommando bzw. die dazu notwendige Zeichenfolge.

Dabei müssen *schräg grau* gedruckte Bereiche geeignet angepasst werden.

2. Bedienung

Aktuellen Ausdruck auswerten bzw. berechnen (Auswertungstaste)	`Shift + Enter` `Enter` (Num-Block)	oder
Neue Zeile	`Enter`	
Vorherige Zeile wieder hineinkopieren	`Shift + L`	
Dezimalkomma	.	(Punkt)
Ausgabe unterdrücken, Verschiedene Befehle voneinander trennen	;	

3. Zahlen, Buchstaben, Symbole

Imaginäre Einheit	`I`	großes i
Eulersche Zahl	`E`	großes e
Pi (π)	`Pi` *oder* Esc p Esc	
Grad	`Degree` *oder* °	z.B. 180°
Goldener Schnitt	`GoldenRatio`	
Unendlich (∞)	Esc inf Esc	
Ableitung/Integration	Esc pd Esc *bzw.* Esc dd Esc	∂ *bzw.* d
Ungleich	!=	
Kreuzprodukt	Esc cross Esc	$\times$
Vereinigung	Esc un Esc	$\cup$
Durchschnitt	Esc inter Esc	$\cap$

4. Eingaben

Normale Ebene	Strg + Leertaste
Bruch	Strg + Shift + /
Hochzahlen	Strg + 6 *oder* Strg + ^
Wurzel	Strg + 2
Zwischen Eingabeebenen wechseln (*z.B. bei Wurzel, Integral, Summe*)	Strg + 5 *oder* Strg + %
Matrizen	
Neue Spalte	Strg + ,
Neue Zeile	Strg + Enter

5. Formatierungen

Generell kann man alle Mathematica-Kommandos anstelle von *Kommando*`[...]` auch in der Form ... `//`*Kommando* nutzen. Bei den folgenden Kommandos, die hauptsächlich der Formatierung dienen, bietet sich dies aber besonders an.

Numerische Lösung	//N
In Form einer Matrix	//MatrixForm
Schönere Form (*runde statt eckige Klammern*)	//TraditionalForm
Spaltendarstellung	//ColumForm
Lange Ergebnisse kurz darstellen	//Short
Wie es in TEX dargestellt werden würde	//TeXForm

Des Weiteren ist in dieser Form sinnvoll: `//Simplify` oder `//Factor` (*siehe später*).

6. Erläuterungen zu Kommandos

Sehr zu empfehlen ist das in Mathematica eingebaute Handbuch. Es enthält sehr viele Informationen zu den einzelnen Kommandos meist auch mit Beispielen. Man erreicht es entweder über *Help/Find Selected Funktions...* oder durch Drücken der `F1`-Taste.

Hilfe zu einem Kommando Wert/Ausdruck einer Variablen	`?`*`Kommando/Variable`*
Detailliertere Hilfe	`??`*`Kommando`*
Erweiterungen zu einem Kommando	`Options[`*`Kommando`*`]`

7. Kommandos

7.1. Allgemein

Sofortige Zuweisung	*`Ausdruck`* `=` *`Ausdruck/Wert`*	
Verzögerte Zuweisung (bei jeder Verwendung wieder neu)	*`Ausdruck`* `:=` *`Ausdruck/Wert`*	
Variable kurzzeitig durch einen Wert ersetzen (Substitution)	*`AusdruckmitVariable`* `/.` *`Substitutionsregel (z.B. x→3)`*	
Letztes Ergebnis	%	
Vorletztes Ergebnis	%%	usw.
Auf Teil eines Ergebnisses oder einer Liste zugreifen	*`Liste`*`[[`*`Nr.`*`]]`	negativ = Zählung von Hinten
Inhalt eines Parameters löschen	`Clear[`*`Parameter`*`]`	(*vgl.* ***`Remove`***)
Parameter aus Kernel löschen	`Remove[`*`Parameter`*`]`	(*vgl.* ***`Clear`***)
Alle Parameter aus Kernel löschen	`Remove["Global`*"]`	

7.2. Rechnen

Grundrechenarten	Normale Eingabe (`+ - * /`) Dies funktioniert auch bei Listen / Matrizen, dann jedoch Komponentenweise. `*` kann weggelassen werden
Numerische Ausgabe	`N[`*`Variable`*`,`*`Stellenzahl`*`]`
Vereinfachen	`Simplify[`*`Ausdruck`*`]`
Vereinfachen (rechenintensiver)	`FullSimplify[`*`Ausdruck`*`]`
Ausmultiplizieren	`Expand[`*`Ausdruck`*`]`
Kürzen	`Cancel[`*`Ausdruck`*`]`
Faktorisieren	`Factor[`*`Ausdruck`*`]`
Auf einen Nenner bringen	`Together[`*`Ausdruck`*`]`
Partialbruchzerlegung	`Apart[`*`Ausdruck`*`]`
Trigonometrische Funktionen in Exponentialfunktionen umwandeln und zurück	`TrigToExp[`*`Ausdruck`*`]` `ExpToTrig[`*`Ausdruck`*`]`

7.3. Komplexes Rechnen

Betrag	`Abs[`*`Variable`*`]`
Phase	`Arg[`*`Variable`*`]`
Realteil	`Re[`*`Variable`*`]`
Imaginärteil	`Im[`*`Variable`*`]`
Konjugiert Komplexe	`Conjugate[`*`Variable`*`]`

7.4. Gleichungssysteme

Gleichungssystem	`gls = Ausdruck == Ausdruck`
Komponentenweise Gleichungssysteme aus Listen (z.B. Koordinaten) erstellen	`Thread[Liste1==Liste2]`
Gleichungssytem mit einer Variablen lösen	`Solve[gls,Variable]`
Gleichungssytem mit mehreren Variablen lösen	`Solve[{gls1,gls2},{Variable1,Variable2}]`
Gleichungssytem/e numerisch lösen *(Siehe auch Kapitel A auf Seite 17)*	`NSolve[gls/e,Variable]`

Lösen rekursiv gestellter Gleichungssysteme *Ab Version 6 ist kein Package mehr nötig.*	`Needs["DiscreteMath`RSolve`"]`
	`RSolve[{f[n]==f[n-1],`
	`f[1]==1},f[n],n]`

Ungleichungen lösen
 	
Ab Version 6:
	`Needs["Algebra`InequalitySolve`"]`
	`InequalitySolve[Gleichungssytem/e,Variable]`
	`Reduce[Gleichungssytem/e,Variable,Reals]`

Gleichungssystem nach einer Variablen auflösen	`Eliminate[Gleichungssytem/e,Variable]`

7.5. Differentialgleichungen

Differentialgleichungen symbolisch lösen	`DSolve[Glssysteme,Funktion,Variable]`
	`DSolve[y'[x]+y[x]==1,y[x],x]`
	`DSolve[y'[x]+y[x]==1,y,x]` (letzes liefert eine Purefuktion)

Differentialgleichungen numerisch lösen (*genügend Anfangsbedingungen durch zusätzliche Gleichungssysteme eingeben, zur Eindeutigen Bestimmung*)	`NDSolve[Glssysteme,Funktion,{Variable,von,bis}]` `NDSolve[y'[x]+y[x]==1,y[x],{x,0,10}]`

7.6. Funktionen

Funktionsdefinition *(Für mögliche Bedingungen siehe Kapitel B auf Seite 18)*	`f = Ausdruck` bzw. `f[x_] = Ausdruck mit Variable`
Minimum einer Funtkion finden	`FindMinimum[f, {Variable,Startwert}]`
Maximum einer Funtkion finden	`FindMaximum[f, {Variable,Startwert}]`
Wertetablelle / Liste erzeugen	`Table[{Ausdruck1,Ausdruck2}, {Variable,von,bis,Schrittgröße}]` oder `Table[Ausdruck,{Variable,AnzahlSchrittemit1}]`
Funktion, die für positive Zahlen = 1 und negative Zahlen = 0 ist	`UnitStep[...]`

7.7. Lineare Optimierung

Minimum linearer Gleichungssysteme finden	`Minimize[Zielfkt,Bedingungen,Variablen]` (*Bei mehr als zwei Elementen Listen verwenden*)
Maximum linearer Gleichungssysteme finden	`Maximize[Zielfkt,Bedingungen,Variablen]` (*Bei mehr als zwei Elementen Listen verwenden*)

7.8. Regressions- und Interpolationsfunktionen

Interpolation (alle Punkte sind enthalten, Purefunktion) *(Siehe auch Kapitel A auf Seite 16)*	`Interpolation[data]`
Interpolation (alle Punkte sind enthalten, Polynom hohen Grades)	`InterpolationPoynomial[data]`
Regressionsfunktion (auch 3D)	`Fit[data,ListemöglicherFunktionen,Variable]`

7.9. Zeichnen 2D

Zeichnen in 2D (explizit) *(Siehe auch Kapitel A auf Seite 17)*	`Plot[`*f*`,{`*Variable*`,`*von*`,`*bis*`}]`

Zeichnen in 2D (implizit)
(Siehe auch Kapitel A auf Seite 17)
Ab Version 6 können Gleichungen in `ContourPlot` *verwendet werden:*
	`Needs["Graphics`ImplicitPlot`"]`
	`ImplicitPlot[`*f*`,{`*Variable*`,`*von*`,`*bis*`}]`
	`ContourPlot[`*gls*`,{`*Variable*`,`*von*`,`*bis*`},`
	`{`*Variable*`,`*von*`,`*bis*`}]`

Zeichnen in 2D (parameter) *(Siehe auch Kapitel A auf Seite 17)*	`ParametrixPlot[{`*x[Variable],y[Variable]*`},` `{`*Variable*`,`*von*`,`*bis*`}]`

Flächen zwischen den Graphen einfärben
(Siehe auch Kapitel A auf Seite 17)
Ab Version 6 gibt es die Option:
	`Needs["Graphics`FilledPlot`"]`
	`FilledPlot[{`*f1,f2*`},{`*Variable*`,`*von*`,`*bis*`}]`
	
	`Filling→Axis` *für* `Plot`

Eine Wertetabelle / Liste zeichnen *(Siehe auch Kapitel A auf Seite 16)*	`ListPlot[`*Liste*`]`

7.10. Zeichnen 3D

Zeichnen in 3D (explizit) *(Siehe auch Kapitel A auf Seite 17)*	`Plot3D[`*f*`, {`*Variable1*`,`*von*`,`*bis*`},` `{`*Variable2*`,`*von*`,`*bis*`}]`
Zeichnen in 3D (parameter) *(Siehe auch Kapitel A auf Seite 17)*	`ParametricPlot3D[{`*x[Variable]*`,`*y[Variable]*`,` *z[Variable]*`},{`*Variable*`,`*von*`,`*bis*`}]`

7.11. Vektorrechnung

Vektoren	*vektor*`={`*koord1*`,`*koord2*`,`*koord3*`}`	
Skalarprodukt zweier Vektoren	*vektor1*`.`*vektor2*	(Punkt)
Kreuzprodukt *(Siehe auch Kapitel 3 auf Seite 4)*	*vektor1* × *vektor2*	(Cross)

7.12. Verktoranalysis

Alle Kommandos dieses Abschnitts benötigen das Package *VectorAnalysis*, deshalb muss dies zuvor mit `Needs["Calculus`VectorAnalysis`"]` oder ab Version 6 mit `Needs["VectorAnalysis`"]`eingebunden werden.

Abfrage des Koordinatensystems	`CoordinateSystem`
Abfrage der Koordinatendarstellung	`Coordinates[]`
Abfrage der möglichen Werte für die einzelnen Koordinaten	`CoordinateRanges[]`
Neues Koordinatensystem einstellen (*z.B.: Cartesian/Cylindrial/Sperical*)	`SetCoordinates[`*`Koordinatensystem`*`]`
Transformation in kartesische Koordinaten	`CoordinatesToCartesian[{`*`r`*`,`ϕ`,`*`z`*`}]`
Transformation von kartesischen Koordinaten	`CoordinatesFromCartesian[{`*`a`*`,`*`b`*`,`*`c`*`}]`
Kreuzprodukt im eingestellten Koordinatensystem	`CrossProduct[`*`v`*`,`*`w`*`]`
Skalarprodukt im eingestellten Koordinatensystem	`DotProduct[`*`v`*`,`*`w`*`]`
Spatprodukt im eingestellten Koordinatensystem	`ScalarTripleProdukt[`*`u`*`,`*`v`*`,`*`w`*`]`
Berechnung des Gradienten einer skalaren Funktion	`Grad[`*`f[x,y,z]`*`]`
Wendet den Laplace-Operator auf eine skalare oder vektorielle Funktion an	`Laplacian[`*`f[x,y,z]`*`]` *oder* `Laplacian[ {`*`f1[k1,k2,k3]`*`,`*`f2[k1,k2,k3]`*`,`*`f3[k1,k2,k3]`*`}]`
Berechnet die Divergenz einer vektoriellen Funktion	`Div[{`*`f1[k1,k2,k3]`*`,`*`f2[k1,k2,k3]`*`,`*`f3[k1,k2,k3]`*`}]`
Berechnet die Rotation einer vektoriellen Funktion	`Curl[{`*`f1[k1,k2,k3]`*`,`*`f2[k1,k2,k3]`*`,`*`f3[k1,k2,k3]`*`}]`

7.13. Matrizen

Matrixdefinition	`mat = Marix` Input|Create Table/Matrix/Palletes	
Zugriff auf einzelne Elemente	`mat[[2,3]]`	2. Zeile 3. Spalte
Transponierte Matrix	`Transpose[mat]`	
Determinante	`Det[mat]`	
Matrixmultiplikation	`mat1.mat2`	(Punkt)
N-te Matrixpotenz	`MatrixPower[mat,n]`	
Eigenwerte	`Eigenvalues[mat]`	
Eigenvektoren	`Eigenvector[mat]`	
Charakteristisches Polynom	`CharacteristicPolynomial[mat,x]`	

7.14. Gleichungssysteme mit Matrizen

Lineare Gleichungssysteme lösen (*findet nur eine Lösung*)	`LinearSolve[mat,vektor]`
Homogenes Gleichungssystem lösen (*findet nur eine Lösung*)	`NullSpace[mat]`

7.15. Grenzwert

Grenzwert *(Bei keiner Angabe der Richtung (Direction) rechtsseitiger Grenzwert)*	`Limit[Ausdruck,Variable → Grenze,Direction → Zahl]`

Hinweis: Die Berechnung des Grenzwertes lässt sich durch `Needs["Calculus`Limit`"]` verbessern. *Dies ist ab Version 6 nicht mehr möglich noch ist es nötig.*

7.16. Summen und Produkte

Summen (*auch unendliche*)	Direkte Eingabe oder `Sum[f[x],{x,von,bis,Schrittweite}]`
Numerische Summen (*auch unendliche*)	`NSum[f[x],{x,von,bis,Schrittweite}]`
Produkte (*auch unendliche*)	Direkte Eingabe oder `Product[f[x],{x,von,bis,Schrittweite}]`
Numerische Produkte (*auch unendliche*)	`NProduct[f[x],{x,von,bis,Schrittweite}]`

7.17. Differentiation und Integration

Partielle Differentiation *(Siehe auch Kapitel A auf Seite 16)*	`D[Ausdruck,nachVariable]` oder $\partial_{Variable}$ *Ausdruck* oder `f'[x]`
Totales Differential *(Siehe auch Kapitel A auf Seite 16)*	`Dt[Ausdruck,Variable]`
Integration *(Siehe auch Kapitel A auf Seite 16)*	Direkte Eingabe oder `Integrate[Ausdruck,{Variable,von,bis}]`
Numerische Integration	`NIntegrate[Ausdruck,{Variable,von,bis}]`

7.18. Reihenentwicklung

Taylor-Reihenentwicklung	`Series[Ausdruck,{Variable, OrtderEntwicklung,Genauigkeit}]`
In Normalform umwandeln (*Abbrechendes O verschwinden lassen*)	`Normal[Reihenentwicklung]`

7.19. Fourier-Transformation

Fourier-Transformation für diskrete Daten	`Fourier[data]`
Umkehrtransformation zu Fourier	`InverseFourier[data]`
Fourier-Transformation von Funktionen	`FourierTransform[f[t],t,w]`
Rücktransformation	`InverseFourierTransform[f[w],w,t]`

7.20. Listen

Liste, die einen Ausdruck x-mal enthält	`Table[Ausdruck,{Anzahl}]`
Liste, die einen Ausdruck mit verschiedenen Werten für eine Variable enthält	`Table[Ausdruck,{Variable,von,bis}]`oder `Table[Ausdruck,{Variable,von,bis,Schrittweite}]`
Listen vereinigen (*Doppelte Elemente zweimal*)	`Join[lst1,lst2]`
Listen vereinigen (*Doppelte Elemente einmal*)	`lst1 ∪ lst2`
Durchschnitt zweier Listen	`lst1 ∩ lst2`
Ein Element hinzufügen (*Am Ende der Liste*)	`Append[lst,Element]`
Ein Element entfernen	`Delete[lst,ElementNummer]`
Listenklammern entfernen	`Flatten[lst,Ebene]` (*Ohne Ebenenangabe alle Klammen bis auf eine entfernen*)
Liste Aufteilen	`Partition[lst,GrößederTeile]`
Elemente einer Liste in ein Kommando (welches normal `[ ]` besitzt) einsetzen	`Map[Kommandoohne[],lst]` oder `Kommandoohne[]/@lst`

7.21. Zahlen

Teiler einer Zahl	`Divisors[`*`Zahl`*`]`
Primfaktorzerlegung	`FactorInteger[`*`Zahl`*`]` (*Ausgabe als Liste zusammen mit der Anzahl*)
Größter gemeinsamer Teiler	`GCD[`*`Zahl1`*`,`*`Zahl2`*`,...]`
Kleinstes gemeinsames Vielfaches	`LCM[`*`Zahl1`*`,`*`Zahl2`*`,...]`

7.22. Neue Notation festlegen

In Mathematica ist es auch möglich eigene Notationen/Symbole an Stelle der sonst üblichen Mathematicanotation zu verwenden. Dies hat den Vorteil, dass man sich Schreibarbeit erspart oder dass man die Übersichtlichkeit erhöht, da Mathematica diese Notation/Symbole auch bei den Ausgaben verwendet.
Für die folgenden Befehle benötigt man `Needs["Utilities`Notation`"]` oder ab Version 6 `Needs["Notation`"]`, das zusätzlich auch die *Notation Palette* öffnet.

Neue Notation festlegen	`Notation[`*`Kommando[x_]`* $\Leftrightarrow$ *`Kommando[x_]`*`]`
Es ist nicht leider möglich diesen Befehl direkt einzugeben. Die Eingabe erfolgt entweder über die Notationspalette oder Esc *notation* Esc	
Beispiel:	`Notation[`$\lvert x\_\rvert$ $\Leftrightarrow$ `Abs[x_]]`
Festgelegte Notation entfernen	`RemoveNotation[... ⇔ ...]`

7.23. Sonstiges

Aufzählung 1,2,3,4,5, ...,n (Liste)	`Range[`*`n`*`]`	`{1,2,3,...,n}`
Modulo-Rechnung	`Mod[`*`Zahl`*`,`*`Divident`*`]`	
Runden auf nächste ganze Zahl	`Round[`*`Zahl`*`]`	
Zufallszahl (*Typ*: Integer, Real, Complex)	`Random[`*`Typ`*`,`*`Zahlenbereich`*`]`	Auto=Real,1

A. Erweiterungen zu Kommandos

Die meisten Kommandos in Mathematica können durch optionale Erweiterungen angepasst werden. Diese *Erweiterungen* werden dabei nach einem zusätzlichen Komma `,` vor der abschließenden Klammer `]` eingefügt.

Beispiel: `Plot[f[x],{x,0,2},PlotRange→All]`

D

Variable als nicht konstant festlegen	`NonConstants→{`*`Variable`*`}`

Dt

Variable als konstant festlegen	`Constants→{`*`Variable`*`}`

Integrate

Zusatzinformationen definieren	`Assumptions→{`*`z.B. a>0`*`}`

Interpolation

Grad der Polynome zur Interpolation	`InterpolationOrder→`*`1-9`*
Ableitungen an den jeweiligen Punkten mit definieren	*`data=`*`{{x1,{y1,y1'},{x2,{y2,y2'},...}`

ListPlot

Punktdicke	`PlotStyle→PlotSize[`*`0.02`*`]`
Punkte verbinden	`PlotJoined→True`

NSolve

Genauigkeit	`WorkingPrecision→`*`Genauigkeit`*

. . . Plot

Bei Plot-Kommandos kann es sinnvoll sein, dass der Ausdruck ausgewertet wird bevor er an Plot weitergegeben wird; z.B. wenn er weitere Mathematicakommandos enthält. Dies wird erreicht durch `Plot[Evaluate[`*`Ausdruck`*`],{`*`Variable`*`,`*`von`*`,`*`bis`*`}]`.

Achsen ein/ausblenden	`Axes→True/False`	True
Achsen beschriften	`AxesLabel→{`*`x-Achse`*`,`*`y-Achse`*`}`	
Rahmen ein/ausblenden	`Frame→True/False`	False
Achsenverhältnis (gleich = 1)	`AspectRatio→`*`Zahl`*	Auto
Wertebereich der y-Achse	`PlotRange→{`*`Min`*`,`*`Max`*`}` `All` für gesamter Wertebereich	Auto
Farbe	`PlotStyle→`*`Farbe`*	0,0,0
Direkt	Input|Color Selector	
RGB-Farben	`RGBColor[`*`0`*`,`*`0`*`,`*`0`*`]`	0-255
Graustufen	`GreyLevel[`*`n`*`]`	0-1
Liniendicke	`PlotStyle→Thickness[`*`0.01`*`]`	
Detailgrad	`PlotPoints→`*`Genauigkeit`*	
Ursprung	`AxesOrigin→{`*`Koord1`*`,`*`Koord2`*`}`	{0,0}
Zwischenlinien anzeigen	`GridLines→Automatic`	None

. . . Plot3D

Für die Verwendung von `Evaluate[...]` siehe unter **. . . Plot** (Seite 17).

Detailgrad	`PlotPoints→`*`Genauigkeit`*
Ansicht	Input|3D Viewpoint Selector
Achsen beschriften	`AxesLabel→{`*`x-Achse`*`,`*`y-Achse`*`,`*`z-Achse`*`}`
Keine Oberflächenfarbe	`Shading→False`
Zwischenlinien anzeigen	`FaceGrids→All`
Box um Graph entfernen	`Boxed→False`

Des Weiteren funktionieren die meisten **. . . Plot**-Erweiterungen in analoger Weise.

B. Bedingungen bei Funktionen

Bei Funktionsdefinitionen ist es möglich Bedingungen an die verwendeten Variablen zu stellen. Diese *Bedingungen* werden dabei nach einem zusätzlichen Slash und Strichpunkt `/;` noch in der Zeile der Funktionsdefinition eingefügt. Dabei ist darauf zu achten, dass man die verzögerte Zuweisung mittels `:=` verwendet.

Ist die Bedingung nicht erfüllt, wird der Ausdruck unausgewertet zurückgegeben.

Beispiel: `f[x_] := Sqrt[x] /; x > 0`

Als Bedingungen eignet sich jedes Mathematica-Kommando, das als Ergebnis ***`true`*** oder ***`false`*** liefert. Siehe dazu das folgende Kapitel.

C. True/False Kommandos

Viele Kommandos in Mathematica liefern als Ergebnis entweder ***`true`*** oder ***`false`***. Im Folgenden will ich eine kleine Auswahl besonders wichtiger Abfragen geben.

Direkte Eingabe (Variable muss vorher mit einem Wert belegt worden sein)	z.B. *x>0*
Prüfe auf Zahl	`NumberQ[`*Variable*`]`
Prüfe auf ganze Zahl	`IntegerQ[`*Variable*`]`
Prüfe auf eine gerade Zahl	`EvenQ[`*Variable*`]`
Prüfe auf eine ungerade Zahl	`OddQ[`*Variable*`]`
Prüfe auf eine Primzahl	`PrimeQ[`*Variable*`]`
Prüfe auf eine Liste	`ListQ[`*Variable*`]`
Prüfe auf eine Matrix	`MatrixQ[`*Variable*`]`

Viele weitere derartiger Abfragen lassen sich über `?*Q` ermitteln.

Diese und andere „True/False Kommandos“ (*TFKommandos*) lassen sich auch verknüpfen bzw. kombinieren:

Nicht-Verknüpfung	`Not[`*TFKommando*`]`
Und-Verknüpfung	*TFKommando* `&&` *TFKommando*
Oder-Verknüpfung	*TFKommando* `||` *TFKommando*
Ausschließende-Oder-Verknüpfung	`Xor[`*TFKommando*`,`*TFKommando*`]`

D. Literaturverzeichnis

- Michael Kofler, Mathematica - Einführung, Anwendung, Referenz
- Mathematica Kurs des Universitätsrechenzentrum Heidelberg
  http://www.urz.uni-heidelberg.de/Ausbildung/Kurse/MathemBlk/PHY01/wkurs.pdf

Tipps zum Umgang mit Mathematica

Abschließend möchte ich noch einige Tipps zum Umgang mit Mathematica geben. Sie sollen dabei helfen gewisse Fehlermeldungen von Mathematica zu verstehen, bzw. zu verhindern.

Tipp 1 Alle Mathematica-internen-Symbole und Befehle fangen mit Großbuchstaben an. Selbst vergebe Namen beispielsweise für Funktionen oder Variablen sollten also mit Kleinbuchstaben beginnen

Tipp 2 Verwendet man ein Kommando, das ein zusätliches Packages erfordert, noch bevor man dieses Package geladen hat, so funktioniert es auch danach nicht, da das Kommando als Variable im Kernel abgespeichert wurde und erst wieder etwa mittels `Remove["Global`*"]` entfernt werden muss.

Tipp 3 Hat man keine zusätzlichen Packages geladen, so ist es schneller einfach den Kernel zu beenden (z.B. durch schließen des entsprechenden Eintrags aus der Taskleiste), als umständlich durch `Remove["Global`*"]` alle Parameter aus selbigen zu entfernen.

Tipp 4 Besonders die umständlich einzutippenden `Needs[...]`-Befehle lassen sich komfortabel per Copy & Paste aus diesem PDF[1] übernehmen

Tipp 5 Verwenden Sie längere aussagekräftige Zeichenketten für ihre selbst definierten Variablen und vermeiden Sie zusammengesetzte Symbole wie $\vec{a}$, $\bar{a}$, a_1 oder a_b, da diese in Mathematica anders abgespeichert werden und deshalb zu Komplikationen führen können. Belegen Sie eine Variable – vorzugsweise `x` – niemals mit einem Wert.

[1] Download unter: http://www.stefan.englert.de.vu