



TI-Nspire™

**TI-Nspire™ CAS
Referenzhandbuch**

Dieser Leitfaden ist gültig für die TI-Nspire™ Software-Version 4.2. Die aktuellste Version der Dokumentation finden Sie unter education.ti.com/guides.

Wichtige Informationen

Außer im Fall anderslautender Bestimmungen der Lizenz für das Programm gewährt Texas Instruments keine ausdrückliche oder implizite Garantie, inklusive aber nicht ausschließlich sämtlicher impliziter Garantien der Handelsfähigkeit und Eignung für einen bestimmten Zweck, bezüglich der Programme und der schriftlichen Dokumentation, und stellt dieses Material nur im „Ist-Zustand“ zur Verfügung. Unter keinen Umständen kann Texas Instruments für besondere, direkte, indirekte oder zufällige Schäden bzw. Folgeschäden haftbar gemacht werden, die durch Erwerb oder Benutzung dieses Materials verursacht werden, und die einzige und exklusive Haftung von Texas Instruments, ungeachtet der Form der Beanstandung, kann den in der Programmlizenz festgesetzten Betrag nicht überschreiten. Zudem haftet Texas Instruments nicht für Forderungen anderer Parteien jeglicher Art gegen die Anwendung dieses Materials.

Lizenz

Bitte lesen Sie die vollständige Lizenz im Verzeichnis
C:\Program Files\TI Education\<TI-Nspire™ Product Name>\license.

© 2006 - 2016 Texas Instruments Incorporated

Inhaltsverzeichnis

Wichtige Informationen	2
Inhaltsverzeichnis	3
Vorlagen für Ausdrücke	5
Alphabetische Auflistung	12
A	12
B	21
C	25
D	53
E	65
F	74
G	84
I	92
L	100
M	117
N	125
O	135
P	137
Q	146
R	150
S	165
T	192
U	208
V	209
W	210
X	213
Z	214
Sonderzeichen	223
Leere (ungültige) Elemente	252
Tastenkürzel zum Eingeben mathematischer Ausdrücke	255
Auswertungsreihenfolge in EOS™ (Equation Operating System)	257
Fehlercodes und -meldungen	259
Warncodes und -meldungen	268
Allgemeine Hinweise	270
Hinweise zu TI Produktservice und Garantieleistungen	270

Vorlagen für Ausdrücke

Vorlagen für Ausdrücke bieten Ihnen eine einfache Möglichkeit, mathematische Ausdrücke in der mathematischen Standardschreibweise einzugeben. Wenn Sie eine Vorlage eingeben, wird sie in der Eingabezeile mit kleinen Blöcken an den Positionen angezeigt, an denen Sie Elemente eingeben können. Der Cursor zeigt, welches Element eingegeben werden kann.

Verwenden Sie die Pfeiltasten oder drücken Sie **tab**, um den Cursor zur jeweiligen Position der Elemente zu bewegen, und geben Sie für jedes Element einen Wert oder Ausdruck ein. Drücken Sie **enter** oder **ctrl enter**, um den Ausdruck auszuwerten.

Vorlage Bruch

ctrl **÷** **Tasten**



Hinweis: Siehe auch **/ (Dividieren)**, Seite 225.

Beispiel:

$$\frac{12}{8 \cdot 2} = \frac{3}{4}$$

Vorlage Exponent

^ **Taste**



Hinweis: Geben Sie den ersten Wert ein, drücken Sie **^** und geben Sie dann den Exponenten ein. Um den Cursor auf die Grundlinie zurückzusetzen, drücken Sie die rechte Pfeiltaste (**▶**).

Hinweis: Siehe auch **^ (Potenz)**, Seite 226.

Beispiel:

$$2^3 = 8$$

Vorlage Quadratwurzel

ctrl **x²** **Tasten**



Hinweis: Siehe auch **√(Quadratwurzel)**, Seite 237.

Beispiel:

$$\sqrt{\{9, a, 4\}} = \{3, \sqrt{a}, 2\}$$

Vorlage n-te Wurzel

ctrl ⌘ Tasten



Hinweis: Siehe auch **root()**, Seite 161.

Beispiel:

$$\sqrt[3]{8}$$

$$\sqrt[3]{\{8, 27, b\}}$$

$$\left\{ \begin{array}{l} \frac{1}{2,3,b^3} \\ 2,3,b^3 \end{array} \right\}$$

Vorlage e Exponent

ex ⌘ Tasten



Potenz zur natürlichen Basis e

Hinweis: Siehe auch **e^()**, Seite 65.

Example:

$$e^1$$

$$e^{1.}$$

e

2.71828182846

Vorlage Logarithmus

ctrl ⌘ Taste



Berechnet den Logarithmus zu einer bestimmten Basis. Bei der Standardbasis 10 wird die Basis weggelassen.

Hinweis: Siehe auch **log()**, Seite 112.

Beispiel:

$$\log_4(2.)$$

0.5

Vorlage Stückweise (2 Teile)

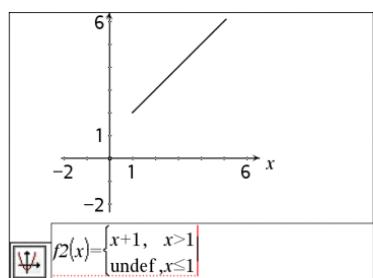
Katalog > 



Ermöglicht es, Ausdrücke und Bedingungen für eine stückweise definierte Funktion aus zwei-Stücken zu erstellen. Um ein Stück hinzuzufügen, klicken Sie in die Vorlage und wiederholen die Vorlage.

Hinweis: Siehe auch **piecewise()**, Seite 139.

Beispiel:



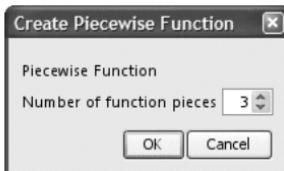
Vorlage Stückweise (n Teile)

Katalog >

Ermöglicht es, Ausdrücke und Bedingungen für eine stückweise definierte Funktion aus n -Teilen zu erstellen. Fragt nach n .

Beispiel:

Siehe Beispiel für die Vorlage
Stückweise (2 Teile).



Hinweis: Siehe auch **piecewise()**, Seite 139.

Vorlage System von 2 Gleichungen

Katalog >



Erzeugt ein System aus zwei Gleichungen. Um einem vorhandenen System eine Zeile hinzuzufügen, klicken Sie in die Vorlage und wiederholen die Vorlage.

Hinweis: Siehe auch **system()**, Seite 191.

Beispiel:

$$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=5 \end{cases}, x, y\right) \quad x=\frac{5}{2} \text{ and } y=-\frac{5}{2}$$

$$\text{solve}\left(\begin{cases} y=x^2-2 \\ x+2\cdot y=1 \end{cases}, x, y\right) \quad x=\frac{-3}{2} \text{ and } y=\frac{1}{4} \text{ or } x=1 \text{ and } y=-1$$

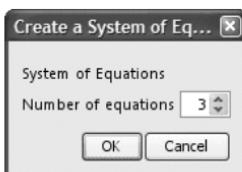
Vorlage System von n Gleichungen

Katalog >

Ermöglicht es, ein System aus N Gleichungen zu erzeugen. Fragt nach N .

Beispiel:

Siehe Beispiel für die Vorlage
Gleichungssystem (2
Gleichungen).



Hinweis: Siehe auch **system()**, Seite 191.

Vorlage Absolutwert

Katalog >



Hinweis: Siehe auch **abs()**, Seite 12.

Beispiel:

Vorlage Absolutwert

Katalog > 

$$\left| \begin{Bmatrix} 2, -3, 4, -4^3 \end{Bmatrix} \right| \quad \{ 2, 3, 4, 64 \}$$

Vorlage dd°mm'ss.ss"

Katalog > 

$$30^\circ 15' 10''$$

Beispiel:

$$\frac{30^\circ 15' 10''}{64800} \quad \frac{10891 \cdot \pi}{64800}$$

Ermöglicht es, Winkel im Format **dd°mm'ss.ss"** einzugeben, wobei **dd** für den Dezimalgrad, **mm** die Minuten und **ss.ss** die Sekunden steht.

Vorlage Matrix (2 x 2)

Katalog > 

$$\begin{bmatrix} \square & \square \\ \square & \square \end{bmatrix}$$

Beispiel:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot a \quad \begin{bmatrix} a & 2 \cdot a \\ 3 \cdot a & 4 \cdot a \end{bmatrix}$$

Erzeugt eine 2 x 2 Matrix.

Vorlage Matrix (1 x 2)

Katalog > 

$$\begin{bmatrix} \square & \square \end{bmatrix}$$

Beispiel:

$$\text{crossP}\left(\begin{bmatrix} 1 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 4 \end{bmatrix}\right) \quad \begin{bmatrix} 0 & 0 & -2 \end{bmatrix}$$

Vorlage Matrix (2 x 1)

Katalog > 

$$\begin{bmatrix} \square \\ \square \end{bmatrix}$$

Beispiel:

$$\begin{bmatrix} 5 \\ 8 \end{bmatrix} \cdot 0.01 \quad \begin{bmatrix} 0.05 \\ 0.08 \end{bmatrix}$$

Vorlage Matrix (m x n)

Katalog > 

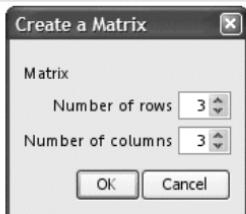
Die Vorlage wird angezeigt, nachdem Sie aufgefordert wurden, die Anzahl der Zeilen und Spalten anzugeben.

Beispiel:

$$\text{diag}\left(\begin{bmatrix} 4 & 2 & 6 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix} \right) \quad \begin{bmatrix} 4 & 2 & 9 \end{bmatrix}$$

Vorlage Matrix (m x n)

Katalog > 



Hinweis: Wenn Sie eine Matrix mit einer großen Zeilen- oder Spaltenanzahl erstellen, dauert es möglicherweise einen Augenblick, bis sie angezeigt wird.

Vorlage Summe (Σ)

Katalog > 

$$\sum_{\square=\square}^{\square} (\square)$$

Beispiel:

$$\sum_{n=3}^{7} (n) \quad 25$$

Hinweis: Siehe auch $\Sigma()$ (sumSeq), Seite 238.

Vorlage Produkt (Π)

Katalog > 

$$\prod_{\square=\square}^{\square} (\square)$$

Beispiel:

$$\prod_{n=1}^{5} \left(\frac{1}{n} \right) \quad \frac{1}{120}$$

Hinweis: Siehe auch $\Pi()$ (prodSeq), Seite 238.

Vorlage Erste Ableitung

Katalog > 

$$\frac{d}{d\square} (\square)$$

Beispiel:

Mit der Vorlage „Erste Ableitung“ können Sie auch die erste Ableitung an einem Punkt berechnen.

Vorlage Erste Ableitung

Katalog >

Hinweis: Siehe auch **d()** (Ableitung), Seite 234.

$$\frac{d}{dx}(x^3) \quad 3 \cdot x^2$$

$$\frac{d}{dx}(x^3)|_{x=3} \quad 27$$

Vorlage Zweite Ableitung

Katalog >

$$\frac{d^2}{dx^2}(\square)$$

Mit der Vorlage „Zweite Ableitung“ können Sie auch die zweite Ableitung an einem Punkt berechnen.

Hinweis: Siehe auch **d()** (Ableitung), Seite 234.

Beispiel:

$$\frac{d^2}{dx^2}(x^3) \quad 6 \cdot x$$

$$\frac{d^2}{dx^2}(x^3)|_{x=3} \quad 18$$

Vorlage n-te Ableitung

Katalog >

$$\frac{d^{\square}}{dx^{\square}}(\square)$$

Mit der Vorlage „n-te Ableitung“ können Sie die n-te Ableitung.

Hinweis: Siehe auch **d()** (Ableitung), Seite 234.

Beispiel:

$$\frac{d^3}{dx^3}(x^3) \quad 6$$

$$\frac{d^3}{dx^3}(x^3)|_{x=3}$$

Vorlage Bestimmtes Integral

Katalog >

$$\int_a^b \square \, dx$$

Hinweis: Siehe auch **f() integral()**, Seite 223.

Beispiel:

$$\int_a^b x^2 \, dx \quad \frac{b^3}{3} - \frac{a^3}{3}$$

Vorlage Unbestimmtes Integral

Katalog >

$$\int \square \, dx$$

Beispiel:

Vorlage Unbestimmtes Integral

Katalog > 

Hinweis: Siehe auch `ʃ()` `integral()`, Seite 223.

$$\int x^2 dx$$

$$\frac{x^3}{3}$$

Vorlage Limes

Katalog > 

$$\lim_{\square \rightarrow \square} (\square)$$

Beispiel:

$$\lim_{x \rightarrow 5} (2 \cdot x + 3)$$

13

Verwenden Sie – oder (–) für den linksseitigen Grenzwert. Verwenden Sie + für den rechtsseitigen Grenzwert.

Hinweis: Siehe auch `limit()`, Seite 102.

Alphabetische Auflistung

Elemente, deren Namen nicht alphabetisch sind (wie +, !, und >) finden Sie am Ende dieses Abschnitts (Seite 223). Wenn nicht anders angegeben, wurden sämtliche Beispiele im standardmäßigen Reset-Modus ausgeführt, wobei alle Variablen als nicht definiert angenommen wurden.

A

abs() (Absolutwert)

Katalog >

abs(AusdrI)⇒Ausdruck

$$\left| \left\{ \frac{\pi}{2}, \frac{-\pi}{3} \right\} \right| = \left\{ \frac{\pi}{2}, \frac{\pi}{3} \right\}$$

abs(Liste I)⇒Liste

$$|2-3 \cdot i| = \sqrt{13}$$

abs(Matrix I)⇒Matrix

$$|z| = |z|$$

Gibt den Absolutwert des Arguments zurück.

$$|x+y \cdot i| = \sqrt{x^2+y^2}$$

Hinweis: Siehe auch **Vorlage Absolutwert**, Seite 7.

Ist das Argument eine komplexe Zahl, wird der Betrag der Zahl zurückgegeben.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt.

amortTbl()

Katalog >

amortTbl([NPmt,N,I,PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [WertRunden]])⇒Matrix

amortTbl(12,60,10,5000,,12,12)

0	0.	0.	5000.
1	-41.67	-64.57	4935.43
2	-41.13	-65.11	4870.32
3	-40.59	-65.65	4804.67
4	-40.04	-66.2	4738.47
5	-39.49	-66.75	4671.72
6	-38.93	-67.31	4604.41
7	-38.37	-67.87	4536.54
8	-37.8	-68.44	4468.1
9	-37.23	-69.01	4399.09
10	-36.66	-69.58	4329.51
11	-36.08	-70.16	4259.35
12	-35.49	-70.75	4188.6

Amortisationsfunktion, die eine Matrix als Amortisationstabellen für eine Reihe von TVM-Argumenten zurückgibt.

NPmt ist die Anzahl der Zahlungen, die in der Tabelle enthalten sein müssen. Die Tabelle beginnt mit der ersten Zahlung.

N, I, PV, Pmt, FV, PpY, CpY und PmtAt werden in der TVM-Argumentetabelle (Seite 206) beschrieben.

- Wenn Sie Pmt nicht angeben, wird standardmäßig Pmt=tvmPmt (N,I,PV,FV,PpY,CpY,PmtAt) eingesetzt.

- Wenn Sie *FV* nicht angeben, wird standardmäßig *FV=0* eingesetzt.
- Die Standardwerte für *PpY*, *CpY* und *PmtAt* sind dieselben wie bei den TVM-Funktionen.

WertRunden (roundValue) legt die Anzahl der Dezimalstellen für das Runden fest.
Standard=2.

Die Spalten werden in der Ergebnismatrix in der folgenden Reihenfolge ausgegeben:
Zahlungsnummer, Zinsanteil,
Tilgungsanteil, Saldo.

Der in Zeile *n* angezeigte Saldo ist der Saldo nach Zahlung *n*.

Sie können die ausgegebene Matrix als Eingabe für die anderen Amortisationsfunktionen *ΣInt()* und *ΣPrn()*, Seite 239, und *bal()*, Seite 21, verwenden.

and (und)

Boolescher Ausdr1 **and** Boolescher Ausdr2 \Rightarrow Boolescher Ausdruck

$$\begin{array}{c} x \geq 3 \text{ and } x \geq 4 \\ \{x \geq 3, x \leq 0\} \text{ and } \{x \geq 4, x \leq -2\} \quad \{x \geq 4, x \leq -2\} \end{array}$$

Boolesche Liste1 **and** Boolesche Liste2
 \Rightarrow Boolesche Liste

Boolesche Matrix1 **and** Boolesche Matrix2 \Rightarrow Boolesche Matrix

Gibt „wahr“ oder „falsch“ oder eine vereinfachte Form des ursprünglichen Terms zurück.

Ganzzahl1 and Ganzzahl2 \Rightarrow Ganzzahl

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer **and**-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 32-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis dann 1, wenn beide Bits 1 sind; andernfalls ist das Ergebnis 0. Der zurückgegebene Wert stellt die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Im Hex-Modus:

0h7AC36 and 0h3D5F	0h2C16
--------------------	--------

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

0b100101 and 0b100	0b100
--------------------	-------

and (und)

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 10).

Geben Sie eine dezimale ganze Zahl ein, die für eine 32-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen.

angle() (Winkel)

angle(Ausdr1)⇒Ausdruck

Gibt den Winkel des Arguments zurück, wobei das Argument als komplexe Zahl interpretiert wird.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt.

Im Dec-Modus:

37 und 0b100

4

Hinweis: Eine binäre Eingabe kann bis zu 64 Stellen haben (das Präfix 0b wird nicht mitgezählt). Eine hexadezimale Eingabe kann bis zu 16 Stellen aufweisen.

Im Grad-Modus:

angle{0+2·i}

90

Im Neugrad-Modus:

angle{0+3·i}

100

Im Bogenmaß-Modus:

angle{1+i}

$\frac{\pi}{4}$

angle{z}

$-\pi \cdot (\text{sign}(z)-1)$

$\frac{2}{2}$

angle{x+i·y}

$\frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right)$

angle(Liste1)⇒Liste

angle(Matrix1)⇒Matrix

Gibt als Liste oder Matrix die Winkel der Elemente aus *Liste1* oder *Matrix1* zurück, wobei jedes Element als komplexe Zahl interpretiert wird, die einen zweidimensionalen kartesischen Koordinatenpunkt darstellt.

angle{ { 1+2·i, 3+0·i, 0-4·i } }

$\left\{ \frac{\pi}{2} - \tan^{-1}\left(\frac{1}{2}\right), 0, -\frac{\pi}{2} \right\}$

ANOVA *Liste1, Liste2[, Liste3,...,Liste20][,Flag]*

Führt eine einfache Varianzanalyse durch, um die Mittelwerte von zwei bis maximal 20 Grundgesamtheiten zu vergleichen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186)

Flag=0 für Daten, *Flag*=1 für Statistik

Ausgabeveriable	Beschreibung
stat.F	Wert der F Statistik
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Gruppen-Freiheitsgrade
stat.SS	Summe der Fehlerquadrate zwischen den Gruppen
stat.MS	Mittlere Quadrate der Gruppen
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittleres Quadrat für die Fehler
stat.sp	Verteilte Standardabweichung
stat.xbarlist	Mittelwerte der Eingabelisten
stat.CLowerList	95 % Konfidenzintervalle für den Mittelwert jeder Eingabeliste
stat.CUpperList	95 % Konfidenzintervalle für den Mittelwert jeder Eingabeliste

ANOVA2way (ANOVA 2fach)**ANOVA2way** *Liste1, Liste2[, Liste3,...,Liste10]
[,LevZei]*

Berechnet eine zweifache Varianzanalyse, um die Mittelwerte von zwei bis maximal 10 Grundgesamtheiten zu vergleichen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186)

LevZei=0 für Block

LevZei=2,3,...,*Len*-1, für Faktor zwei, wobei
Len=length(*Liste1*)=length(*Liste2*) = ... = length
(*Liste10*) und *Len* / *LevZei* ∈ {2,3,...}

Ausgaben: Block-Design

Ausgabevariable	Beschreibung
stat.F	F Statistik des Spaltenfaktors
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade des Spaltenfaktors
stat.SS	Summe der Fehlerquadrate des Spaltenfaktors
stat.MS	Mittlere Quadrate für Spaltenfaktor
stat.FBlock	F Statistik für Faktor
stat.PValBlock	Kleinste Wahrscheinlichkeit, bei der die Nullhypothese verworfen werden kann
stat.dfBlock	Freiheitsgrade für Faktor
stat.SSBlock	Summe der Fehlerquadrate für Faktor
stat.MSBlock	Mittlere Quadrate für Faktor
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittlere Quadrate für die Fehler
stat.s	Standardabweichung des Fehlers

Ausgaben des SPALTENFAKTORS

Ausgabevariable	Beschreibung
stat.Fcol	F Statistik des Spaltenfaktors
stat.PValCol	Wahrscheinlichkeitswert des Spaltenfaktors
stat.dfCol	Freiheitsgrade des Spaltenfaktors
stat.SSCol	Summe der Fehlerquadrate des Spaltenfaktors
stat.MSCol	Mittlere Quadrate für Spaltenfaktor

Ausgaben des ZEILENFAKTORS

Ausgabevariable	Beschreibung
stat.Frow	F Statistik des Zeilenfaktors
stat.PValRow	Wahrscheinlichkeitswert des Zeilenfaktors
stat.dfRow	Freiheitsgrade des Zeilenfaktors

Ausgabevariable	Beschreibung
stat.SSRow	Summe der Fehlerquadrate des Zeilenfaktors
stat.MSRow	Mittlere Quadrate für Zeilenfaktor

INTERAKTIONS-Ausgaben

Ausgabevariable	Beschreibung
stat.FInteract	F Statistik der Interaktion
stat.PValInteract	Wahrscheinlichkeitswert der Interaktion
stat.dfInteract	Freiheitsgrade der Interaktion
stat.SSInteract	Summe der Fehlerquadrate der Interaktion
stat.MSInteract	Mittlere Quadrate für Interaktion

FEHLER-Ausgaben

Ausgabevariable	Beschreibung
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittlere Quadrate für die Fehler
s	Standardabweichung des Fehlers

Ans (Antwort)

ctrl **(→)** **Taste**

Ans⇒Wert

Gibt das Ergebnis des zuletzt ausgewerteten Ausdrucks zurück.

56	56
56+4	60
60+4	64

approx() (Approximieren)

Katalog >

approx(Ausdr1)⇒Ausdruck

Gibt die Auswertung des Arguments ungeachtet der aktuellen Einstellung des Modus **Auto oder Näherung** als Dezimalwert zurück, sofern möglich.

Gleichwertig damit ist die Eingabe des Arguments und Drücken von **ctrl enter**.

approx($\frac{1}{3}$)	0.333333
approx($\left\{\frac{1}{3}, \frac{1}{9}\right\}$)	{0.333333, 0.111111}
approx({sin(π), cos(π)})	{0., -1.}
approx([√2 √3])	[1.41421 1.73205]
approx([1 1 / 3 9])	[0.333333 0.111111]

approx(Liste1)⇒Liste

approx({sin(π), cos(π)})	{0., -1.}
approx([√2 √3])	[1.41421 1.73205]

Gibt, sofern möglich, eine Liste oder *Matrix* zurück, in der jedes Element dezimal ausgewertet wurde.

►approxFraction()

Katalog >

Ausdr►approxFraction([Tol])⇒Ausdruck

Liste ►**approxFraction([Tol])⇒Liste**

Matrix ►**approxFraction([Tol])⇒Matrix**

Gibt die Eingabe als Bruch mit der Toleranz *Tol* zurück. Wird *tol* weggelassen, so wird die Toleranz 5.E-14 verwendet.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie @>**approxFraction (...)** eintippen.

$\frac{1}{2} + \frac{1}{3} + \tan(\pi)$	0.833333
0.8333333333333333	►approxFraction(5.E-14)
$\frac{5}{6}$	
{π, 1.5}	►approxFraction(5.E-14)
$\left\{ \frac{5419351}{1725033}, \frac{3}{2} \right\}$	

approxRational()

Katalog >

approxRational(Ausdr[, Tol])⇒Ausdruck

approxRational(Liste[, Tol])⇒Liste

approxRational(Matrix[, Tol])⇒Matrix

Gibt das Argument als Bruch mit der Toleranz *Tol* zurück. Wird *tol* weggelassen, so wird die Toleranz 5.E-14 verwendet.

approxRational(0.333, 5·10 ⁻⁵)	$\frac{333}{1000}$
approxRational({0.2, 0.33, 4.125}, 5.E-14)	$\left\{ \frac{1}{5}, \frac{33}{100}, \frac{33}{8} \right\}$

arccos()Siehe $\cos^{-1}()$, Seite 38**arccosh()**Siehe $\cosh^{-1}()$, Seite 39.**arccot()**Siehe $\cot^{-1}()$, Seite 40.**arccoth()**Siehe $\coth^{-1}()$, Seite 41.**arccsc()**Siehe $\csc^{-1}()$, Seite 43.**arccsch()**Siehe $\csch^{-1}()$, Seite 44.**arcLen() (Bogenlänge)**

Katalog >

arcLen(Ausdr1,Var,Start,Ende)
⇒AusdruckGibt die Bogenlänge von *Ausdr1* von *Start* bis *Ende* bezüglich der Variablen *Var* zurück.

Die Bogenlänge wird als Integral unter Annahme einer Definition im Modus Funktion berechnet.

arcLen(Liste1,Var,Start,Ende)⇒ListeGibt eine Liste der Bogenlängen für jedes Element von *Liste1* zwischen *Start* und *Ende* bezüglich der Variablen *Var* zurück.

$$\frac{\text{arcLen}(\cos(x),x,0,\pi)}{\text{arcLen}(f(x),x,a,b)} \quad 3.8202$$

$$\int_a^b \sqrt{\left(\frac{d}{dx}(f(x))\right)^2 + 1} dx$$

$$\text{arcLen}(\{\sin(x),\cos(x)\},x,0,\pi) \quad \{3.8202, 3.8202\}$$

arcsec()Siehe $\sec^{-1}()$, Seite 165.

arcsech()**Siehe $\text{sech}^{-1}()$, Seite 166.****arcsin()****Siehe $\sin^{-1}()$, Seite 176.****arcsinh()****Siehe $\sinh^{-1}()$, Seite 178.****arctan()****Siehe $\tan^{-1}()$, Seite 193.****arctanh()****Siehe $\tanh^{-1}()$, Seite 194.****augment() (Erweitern)****Katalog > ****augment(Liste1, Liste2)⇒Liste**

augment({1, -3, 2}, {5, 4}) {1, -3, 2, 5, 4}

Gibt eine neue Liste zurück, die durch Anfügen von *Liste2* ans Ende von *Liste1* erzeugt wurde.

augment(Matrix1, Matrix2)⇒Matrix

Gibt eine neue Matrix zurück, die durch Anfügen von *Matrix2* an *Matrix1* erzeugt wurde. Wenn das Zeichen "," verwendet wird, müssen die Matrizen gleiche Zeilendimensionen besitzen, und *Matrix2* wird spaltenweise an *Matrix1* angefügt. Verändert weder *Matrix1* noch *Matrix2*.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
$\begin{bmatrix} 5 \\ 6 \end{bmatrix} \rightarrow m2$	$\begin{bmatrix} 5 \\ 6 \end{bmatrix}$
augment(<i>m1, m2</i>)	$\begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{bmatrix}$

avgRC() (Durchschnittliche Änderungsrate)

Katalog >

avgRC(Ausdr1, Var [=Wert] [, Schritt])
 \Rightarrow Ausdruck

$$\text{avgRC}(f(x), x, h) = \frac{f(x+h) - f(x)}{h}$$

avgRC(Ausdr1, Var [=Wert] [, Liste1])
 \Rightarrow Liste

$$\text{avgRC}(\sin(x), x, h)|_{x=2} = \frac{\sin(h+2) - \sin(2)}{h}$$

avgRC(Liste1, Var [=Wert] [, Schritt])
 \Rightarrow Liste

$$\text{avgRC}(x^2 - x + 2, x) = 2 \cdot (x - 0.4995)$$

avgRC(Matrix1, Var [=Wert] [, Schritt])
 \Rightarrow Matrix

$$\text{avgRC}(x^2 - x + 2, x, 0.1) = 2 \cdot (x - 0.45)$$

$$\text{avgRC}(x^2 - x + 2, x, 3) = 2 \cdot (x + 1)$$

Gibt den rechtsseitigen Differenzenquotienten zurück (durchschnittliche Änderungsrate).

Ausdr1 kann eine benutzerdefinierte Funktion sein (siehe **Func**).

Wenn *Wert* angegeben ist, setzt er jede vorausgegangene Variablenzuweisung oder jede aktuelle „|“ Ersetzung für die Variable außer Kraft.

Schritt ist der Schrittwert. Wird *Schritt* nicht angegeben, wird als Vorgabewert 0,001 benutzt.

Beachten Sie, dass die ähnliche Funktion **centralDiff()** den zentralen Differenzenquotienten benutzt.

B

bal()

Katalog >

bal(NPmt,N,I,PV,[Pmt],[FV],[PpY], [CpY],[PmtAt],[WertRunden]) \Rightarrow Wert

$$\text{bal}(5, 6, 5.75, 5000, , 12, 12) = 833.11$$

bal(NPmt,AmortTabelle) \Rightarrow Wert

$$\text{tbl:}= \text{amortTbl}(6, 6, 5.75, 5000, , 12, 12)$$

Amortisationsfunktion, die den Saldo nach einer angegebenen Zahlung berechnet.

0	0.	0.	5000.
1	-23.35	-825.63	4174.37
2	-19.49	-829.49	3344.88
3	-15.62	-833.36	2511.52
4	-11.73	-837.25	1674.27
5	-7.82	-841.16	833.11
6	-3.89	-845.09	-11.98

N, I, PV, Pmt, FV, PpY, CpY und *PmtAt* werden in der TVM-Argumentetabelle (Seite 206) beschrieben.

$$\text{bal}(4, \text{tbl}) = 1674.27$$

NPmt bezeichnet die Zahlungsnummer, nach der die Daten berechnet werden

sollen.

N, I, PV, Pmt, FV, PpY, CpY und PmtAt
werden in der TVM-Argumentetabelle
(Seite 206) beschrieben.

- Wenn Sie *Pmt* nicht angeben, wird standardmäßig *Pmt=tvmPmt* $\{N,I,PV,FV,PpY,CpY,PmtAt\}$ eingesetzt.
- Wenn Sie *FV* nicht angeben, wird standardmäßig *FV=0* eingesetzt.
- Die Standardwerte für *PpY*, *CpY* und *PmtAt* sind dieselben wie bei den TVM-Funktionen.

WertRunden (roundValue) legt die Anzahl der Dezimalstellen für das Runden fest.
Standard=2.

bal(NPmt,AmortTabelle) berechnet den Saldo nach jeder Zahlungsnummer *NPmt* auf der Grundlage der Amortisationstabelle *AmortTabelle*. Das Argument *AmortTabelle (amortTable)* muss eine Matrix in der unter **amortTbl()**, Seite 12, beschriebenen Form sein.

Hinweis: Siehe auch **ΣInt()** und **ΣPrn()**, Seite 239.

►Base2

Ganzzahl1 ►Base2⇒*Ganzzahl*

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Base2 eintippen.

Konvertiert *Ganzzahl1* in eine Binärzahl.
Dual- oder Hexadezimalzahlen weisen stets das Präfix 0b bzw. 0h auf. Null (nicht Buchstabe O) und b oder h.

0b *binäre_Zahl*

0h *hexadezimale_Zahl*

Eine Dualzahl kann bis zu 64 Stellen haben, eine Hexadezimalzahl bis zu 16.

256►Base2	0b100000000
0h1F►Base2	0b11111

Ohne Präfix wird *GanzzahlI* als Dezimalzahl behandelt (Basis 10). Das Ergebnis wird unabhängig vom Basis-Modus binär angezeigt.

Negative Zahlen werden als Binärkomplement angezeigt. Beispiel:

-1 wird angezeigt als

0hFFFFFFFFFFFFFFF im Hex-Modus

0b111...111 (64 Einsen) im Binärmodus

- 2^{63} wird angezeigt als

0h8000000000000000 im Hex-Modus

0b100...000 (63 Nullen) im Binärmodus

Geben Sie eine dezimale ganze Zahl ein, die außerhalb des Bereichs einer 64-Bit-Dualform mit Vorzeichen liegt, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Die folgenden Beispiele verdeutlichen, wie diese Anpassung erfolgt:

2^{63} wird zu - 2^{63} und wird angezeigt als

0h8000000000000000 im Hex-Modus

0b100...000 (63 Nullen) im Binärmodus

2^{64} wird zu 0 und wird angezeigt als

0h0 im Hex-Modus

0b0 im Binärmodus

- $2^{63} - 1$ wird zu $2^{63} - 1$ und wird angezeigt als

0h7FFFFFFFFFFFFFFF im Hex-Modus

0b111...111 (64 1's) im Binärmodus

►Base10

Katalog >

Ganzzahl1 ►Base10⇒Ganzzahl

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Base10 eintippen.

Konvertiert *Ganzzahl1* in eine Dezimalzahl (Basis 10). Ein binärer oder hexadezimaler Eintrag muss stets das Präfix 0b bzw. 0h aufweisen.

0b *binäre_Zahl*

0h *hexadezimale_Zahl*

Null (nicht Buchstabe O) und b oder h.

Eine Dualzahl kann bis zu 64 Stellen haben, eine Hexadezimalzahl bis zu 16.

Ohne Präfix wird *Ganzzahl1* als Dezimalzahl behandelt. Das Ergebnis wird unabhängig vom Basis-Modus dezimal angezeigt.

0b10011 ►Base10

19

0h1F ►Base10

31

►Base16

Katalog >

Ganzzahl1 ►Base16⇒Ganzzahl

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Base16 eintippen.

Wandelt *Ganzzahl1* in eine Hexadezimalzahl um. Dual- oder Hexadezimalzahlen weisen stets das Präfix 0b bzw. 0h auf.

0b *binäre_Zahl*

0h *hexadezimale_Zahl*

Null (nicht Buchstabe O) und b oder h.

Eine Dualzahl kann bis zu 64 Stellen haben, eine Hexadezimalzahl bis zu 16.

Ohne Präfix wird *Ganzzahl1* als Dezimalzahl behandelt (Basis 10). Das Ergebnis wird unabhängig vom Basis-Modus hexadezimal angezeigt.

256 ►Base16

0h100

0b111100001111 ►Base16

0hF0F

Geben Sie eine dezimale ganze Zahl ein, die für eine 64-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter ►Base2, Seite 22.

binomCdf()

binomCdf(*n,p*)⇒*Zahl*

binomCdf(*n,p,untereGrenze,obereGrenze*)⇒*Zahl*, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

binomCdf(*n,p,obereGrenze*) für $P(0 \leq X \leq obereGrenze)$
⇒*Zahl*, wenn *obereGrenze* eine Zahl ist, *Liste*, wenn *obereGrenze* eine Liste ist

Berechnet die kumulative Wahrscheinlichkeit für die diskrete Binomialverteilung mit *n* Versuchen und der Wahrscheinlichkeit *p* für einen Erfolg in jedem Einzelversuch.

Für $P(X \leq obereGrenze)$ setzen Sie *untereGrenze*=0

binomPdf()

binomPdf(*n,p*)⇒*Zahl*

binomPdf(*n,p,XWert*)⇒*Zahl*, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeit an einem *XWert* für die diskrete Binomialverteilung mit *n* Versuchen und der Wahrscheinlichkeit *p* für den Erfolg in jedem Einzelversuch.

C**ceiling() (Obergrenze)**

ceiling(*AusdrI*)⇒*Ganzzahl*

`ceiling(.456)`

1.

Gibt die erste ganze Zahl zurück, die \geq dem Argument ist.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

Hinweis: Siehe auch **floor()**.

ceiling(Liste1)⇒Liste

ceiling(Matrix1)⇒Matrix

Für jedes Element einer Liste oder Matrix wird die kleinste ganze Zahl, die größer oder gleich dem Element ist, zurückgegeben.

$\text{ceiling}(\{-3.1, 1.2, 5\})$	{ -3, 1, 3 }
$\text{ceiling}\begin{pmatrix} 0 & -3.2 \cdot i \\ 1.3 & 4 \end{pmatrix}$	$\begin{pmatrix} 0 & -3 \cdot i \\ 2 & 4 \end{pmatrix}$

centralDiff()

centralDiff(Ausdr1,Var [=Wert],[,Schritt])
⇒Ausdruck

centralDiff(Ausdr1,Var [,Schritt])
| Var=Wert⇒Ausdruck

centralDiff(Ausdr1,Var [=Wert],[,Liste])
⇒Liste

centralDiff(Liste1,Var [=Wert],[,Schritt])
⇒Liste

centralDiff(Matrix1,Var [=Wert],[,Schritt])
⇒Matrix

Gibt die numerische Ableitung unter Verwendung des zentralen Differenzenquotienten zurück.

Wenn *Wert* angegeben ist, setzt er jede vorausgegangene Variablenzuweisung oder jede aktuelle „|“ Ersetzung für die Variable außer Kraft.

Schritt ist der Schrittewert. Wird *Schritt* nicht angegeben, wird als Vorgabewert 0,001 benutzt.

Wenn Sie *Liste1* oder *Matrix1* verwenden, wird die Operation über die Werte in der Liste oder die Matrixelemente abgebildet.

Hinweis: Siehe auch **und d()**.

$\text{centralDiff}(\cos(x), x, h)$	$\frac{-(\cos(x-h)-\cos(x+h))}{2 \cdot h}$
$\lim_{h \rightarrow 0} (\text{centralDiff}(\cos(x), x, h))$	$-\sin(x)$
$\text{centralDiff}(x^3, x, 0.01)$	$3 \cdot (x^2 + 0.000033)$
$\text{centralDiff}(\cos(x), x) _{x=\frac{\pi}{2}}$	$-1.$
$\text{centralDiff}(x^2, x, \{0.01, 0.1\})$	$\{2 \cdot x, 2 \cdot x\}$

cFactor(Ausdr1[,Var])⇒Ausdruck

cFactor(Liste1[,Var])⇒Liste

cFactor(Matrix1[,Var])⇒Matrix

cFactor(Ausdr1) gibt Ausdr1 nach allen seinen Variablen über einem gemeinsamen Nenner faktorisiert zurück.

Ausdr1 wird soweit wie möglich in lineare rationale Faktoren zerlegt, selbst wenn dies die Einführung neuer nicht-reeller Zahlen bedeutet. Diese Alternative ist angemessen, wenn Sie die Faktorisierung bezüglich mehr als einer Variablen vornehmen möchten.

cFactor(Ausdr1,Var) gibt Ausdr1 nach der Variablen Var faktorisiert zurück.

Ausdr1 wird soweit wie möglich in Faktoren zerlegt, die linear in Var sind, mit möglicherweise nicht-reellen Konstanten, selbst wenn irrationale Konstanten oder Unterausdrücke, die in anderen Variablen irrational sind, eingeführt werden.

Die Faktoren und ihre Terme werden mit Var als Hauptvariable sortiert. Gleichartige Potenzen von Var werden in jedem Faktor zusammengefasst. Beziehen Sie Var ein, wenn die Faktorisierung nur bezüglich dieser Variablen benötigt wird und Sie irrationale Ausdrücke in anderen Variablen akzeptieren möchten, um die Faktorisierung bezüglich Var so weit wie möglich vorzunehmen. Es kann sein, dass als Nebeneffekt in gewissem Umfang eine Faktorisierung nach anderen Variablen auftritt.

cFactor($a^3 \cdot x^2 + a \cdot x^2 + a^3 + a \cdot x$)	
$a \cdot (a^2 + 1) \cdot (x - i) \cdot (x + i)$	
cFactor($x^2 + \frac{4}{9}$)	$\frac{(3 \cdot x - 2 \cdot i) \cdot (3 \cdot x + 2 \cdot i)}{9}$
cFactor($x^2 + 3$)	$x^2 + 3$
cFactor($x^2 + a$)	$x^2 + a$

cFactor($a^3 \cdot x^2 + a \cdot x^2 + a^3 + a \cdot x$)	
$a \cdot (a^2 + 1) \cdot (x - i) \cdot (x + i)$	
cFactor($x^2 + 3, x$)	$(x + \sqrt{3} \cdot i) \cdot (x - \sqrt{3} \cdot i)$
cFactor($x^2 + a, x$)	$(x + \sqrt{a} \cdot i) \cdot (x - \sqrt{a} \cdot i)$

cFactor() (Komplexer Faktor)

Katalog >

Bei der Einstellung Auto für den Modus **Auto oder Näherung** ermöglicht die Einbeziehung von *Var* auch eine Näherung mit Gleitkommakoeffizienten in Fällen, wo irrationale Koeffizienten nicht explizit bezüglich der integrierten Funktionen ausgedrückt werden können. Selbst wenn es nur eine Variable gibt, kann das Einbeziehen von *Var* eine vollständigere Faktorisierung ermöglichen.

Hinweis: Siehe auch **factor()**.

$$\begin{aligned} \text{cFactor}\left(x^5+4 \cdot x^4+5 \cdot x^3-6 \cdot x-3\right) \\ x^5+4 \cdot x^4+5 \cdot x^3-6 \cdot x-3 \end{aligned}$$

$$\begin{aligned} \text{cFactor}\left(x^5+4 \cdot x^4+5 \cdot x^3-6 \cdot x-3, x\right) \\ (x-0.964673) \cdot (x+0.611649) \cdot (x+2.12543) \cdot (x \end{aligned}$$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

char() (Zeichenstring)

Katalog >

char(Ganzzahl) \Rightarrow Zeichen

char(38)	"&"
char(65)	"A"

Gibt ein Zeichenstring zurück, das das Zeichen mit der Nummer *Ganzzahl* aus dem Zeichensatz des Handhelds enthält. Der gültige Wertebereich für *Ganzzahl* ist 0–65535.

charPoly()

Katalog >

charPoly(Quadratmatrix, Var)
 \Rightarrow Polynomausdruck

$$m := \begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 0 \\ -2 & 2 & 5 \end{bmatrix} \quad \begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 0 \\ -2 & 2 & 5 \end{bmatrix}$$

charPoly(Quadratmatrix, Ausdr)
 \Rightarrow Polynomausdruck

$$\text{charPoly}(m, x) \quad -x^3+5 \cdot x^2+7 \cdot x-35$$

charPoly(Quadratmatrix1, Matrix2)
 \Rightarrow Polynomausdruck

$$\text{charPoly}\left(m, x^2+1\right) \quad -x^6+2 \cdot x^4+14 \cdot x^2-24$$

Gibt das charakteristische Polynom von *Quadratmatrix* zurück.
 Das charakteristische Polynom einer $n \times n$ Matrix *A*, gekennzeichnet durch $p_A(\lambda)$, ist das durch

$$\text{charPoly}(m, m) \quad 0$$

$$p_A(\lambda) = \det(\lambda \cdot I - A)$$

definierte Polynom, wobei *I* die $n \times n$ -Einheitsmatrix kennzeichnet.

Quadratmatrix1 und *Quadratmatrix2* müssen dieselbe Dimension haben.

χ^2 **2way BeobMatrix****chi22way BeobMatrix**

Berechnet eine χ^2 Testgröße auf Grundlage einer beobachteten Matrix *BeobMatrix*. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Informationen zu den Auswirkungen leerer Elemente in einer Matrix finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabeveriable	Beschreibung
stat. χ^2	Chi-Quadrat-Testgröße: sum(beobachtet - erwartet) ² /erwartet
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade der Chi-Quadrat-Testgröße
stat.ExpMat	Berechnete Kontingenztafel der erwarteten Häufigkeiten bei Annahme der Nullhypothese
stat.CompMat	Berechnete Matrix der Chi-Quadrat-Summanden in der Testgröße

χ^2 **Cdf(untereGrenze, obereGrenze, Freigrad) \Rightarrow Zahl,**
wenn *untereGrenze* und *obereGrenze* Zahlen sind,
Liste, wenn *untereGrenze* und *obereGrenze* Listen
sind

chi2Cdf(untereGrenze, obereGrenze, Freiheitsgrad)
 \Rightarrow Zahl, wenn *untereGrenze* und *obereGrenze*
Zahlen sind, *Liste*, wenn *untereGrenze* und
obereGrenze Listen sind

Berechnet die Verteilungswahrscheinlichkeit χ^2
zwischen *untereGrenze* und *obereGrenze* für die
angegebenen Freiheitsgrade *FreiGrad*.

Für $P(X \leq obereGrenze)$ setzen Sie *untereGrenze*=
0.

Informationen zu den Auswirkungen leerer Elemente
in einer Liste finden Sie unter "Leere (ungültige)
Elemente" (Seite 252).

χ^2 GOF BeobListe,expListe,FreiGrad

chi2GOF BeobListe,expListe,FreiGrad

Berechnet eine Testgröße, um zu überprüfen, ob die Stichprobendaten aus einer Grundgesamtheit stammen, die einer bestimmten Verteilung genügt.
`obsList` ist eine Liste von Zählern und muss Ganzzahlen enthalten. Eine Zusammenfassung der Ergebnisse wird in der Variablen `stat.results` gespeichert. (Seite 186)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

Ausgabeveriable	Beschreibung
stat. χ^2	Chi-Quadrat-Testgröße: sum((beobachtet - erwartet) ² /erwartet)
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade der Chi-Quadrat-Testgröße
stat.CompList	Liste der Chi-Quadrat-Summanden in der Testgröße

χ^2 Pdf(XWert,FreiGrad) \Rightarrow Zahl, wenn Xwert eine Zahl ist, Liste, wenn Xwert eine Liste ist

chi2Pdf(XWert,FreiGrad) \Rightarrow Zahl, wenn Xwert eine Zahl ist, Liste, wenn Xwert eine Liste ist

Berechnet die Wahrscheinlichkeitsdichtefunktion (Pdf) einer χ^2 -Verteilung an einem bestimmten XWert für die vorgegebenen Freiheitsgrade FreiGrad.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

ClearAZ (LöscheAZ)

Katalog >

ClearAZ

Löscht alle Variablen mit einem Zeichen im aktuellen Problembereich.

Wenn eine oder mehrere Variablen gesperrt sind, wird bei diesem Befehl eine Fehlermeldung angezeigt und es werden nur die nicht gesperrten Variablen gelöscht.
Siehe **unLock**, Seite 209

5→b	5
b	5
ClearAZ	Done
b	b

ClrErr (LöFehler)

Katalog >

ClrErr

Löscht den Fehlerstatus und setzt die Systemvariable *FehlerCode* (*errorCode*) auf Null.

Ein Beispiel für **ClrErr** finden Sie als Beispiel 2 im Abschnitt zum Befehl **Versuche (Try)**, Seite 202.

Das **Else** im Block **Try...Else...EndTry** muss **ClrErr** oder **PassErr** (**Ügebfehler**) verwenden. Wenn der Fehler verarbeitet oder ignoriert werden soll, verwenden Sie **ClrErr**. Wenn nicht bekannt ist, was mit dem Fehler zu tun ist, verwenden Sie **PassErr**, um ihn an den nächsten Error Handler zu übergeben. Wenn keine weiteren **Try...Else...EndTry** Error Handler unerledigt sind, wird das Fehlerdialogfeld als normal angezeigt.

Hinweis: Siehe auch **PassErr**, Seite 138, und **Try**, Seite 202.

Hinweis zum Eingeben des Beispiels: Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

colAugment() (Spaltenerweiterung)

Katalog >

colAugment(*Matrix1*, *Matrix2*)⇒*Matrix*

Gibt eine neue Matrix zurück, die durch Anfügen von *Matrix2* an *Matrix1* erzeugt wurde. Die Matrizen müssen gleiche Spaltendimensionen haben, und *Matrix2* wird zeilenweise an *Matrix1* angefügt. Verändert weder *Matrix1* noch *Matrix2*.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
$\begin{bmatrix} 5 & 6 \end{bmatrix} \rightarrow m2$	$\begin{bmatrix} 5 & 6 \end{bmatrix}$
colAugment(<i>m1,m2</i>)	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$

colDim() (Spaltendimension)**Katalog >** **colDim(*Matrix*)** \Rightarrow AusdruckGibt die Anzahl der Spalten von *Matrix* zurück.

$$\text{colDim}\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$$

3

Hinweis: Siehe auch **rowDim()**.**colNorm() (Spaltennorm)****Katalog >** **colNorm(*Matrix*)** \Rightarrow AusdruckGibt das Maximum der Summen der absoluten Elementwerte der Spalten von *Matrix* zurück.

$$\begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix} \rightarrow \text{mat}$$

$$\text{colNorm}(\text{mat})$$

Hinweis: Undefinierte Matrixelemente sind nicht zulässig. Siehe auch **rowNorm()**.**comDenom() (Gemeinsamer Nenner)****Katalog >** **comDenom(*Ausdr1[,Var]*)** \Rightarrow Ausdruck**comDenom(*Liste1[,Var]*)** \Rightarrow Liste**comDenom(*Matrix1[,Var]*)** \Rightarrow Matrix

$$\text{comDenom}\left(\frac{y^2+y}{(x+1)^2} + y^2+y\right)$$

$$\frac{x^2 \cdot y^2 + x^2 \cdot y + 2 \cdot x \cdot y^2 + 2 \cdot x \cdot y + 2 \cdot y^2 + 2 \cdot y}{x^2 + 2 \cdot x + 1}$$

comDenom(*Ausdr1*) gibt den gekürzten Quotienten aus einem vollständig entwickelten Zähler und einem vollständig entwickelten Nenner zurück.

comDenom(*Ausdr1,Var*) gibt einen gekürzten Quotienten von Zähler und Nenner zurück, der bezüglich *Var* entwickelt wurde. Die Terme und Faktoren werden mit *Var* als der Hauptvariablen sortiert. Gleichtartige Potenzen von *Var* werden zusammengefasst. Es kann sein, dass als Nebeneffekt eine Faktorisierung der zusammengefassten Koeffizienten auftritt. Verglichen mit dem Weglassen von *Var* spart dies häufig Zeit, Speicherplatz und Platz auf dem Bildschirm und macht den Ausdruck verständlicher. Außerdem werden anschließende Operationen an diesem Ergebnis schneller, und es wird weniger wahrscheinlich, dass der

$$\text{comDenom}\left(\frac{y^2+y}{(x+1)^2} + y^2+y, x\right)$$

$$\frac{x^2 \cdot y \cdot (y+1) + 2 \cdot x \cdot y \cdot (y+1) + 2 \cdot y \cdot (y+1)}{x^2 + 2 \cdot x + 1}$$

$$\text{comDenom}\left(\frac{y^2+y}{(x+1)^2} + y^2+y, y\right)$$

$$\frac{y^2 \cdot (x^2 + 2 \cdot x + 2) + y \cdot (x^2 + 2 \cdot x + 2)}{x^2 + 2 \cdot x + 1}$$

comDenom() (Gemeinsamer Nenner)

Katalog >

Speicherplatz ausgeht.

Wenn *Var* nicht in *Ausdr1* vorkommt, gibt **comDenom(Ausdr1,Var)** einen gekürzten Quotienten eines nicht entwickelten Zählers und eines nicht entwickelten Nenners zurück. Solche Ergebnisse sparen meist sogar noch mehr Zeit, Speicherplatz und Platz auf dem Bildschirm. Solche partiell faktorisierten Ergebnisse machen ebenfalls anschließende Operationen mit dem Ergebnis schneller und das Erschöpfen des Speicherplatzes weniger wahrscheinlich.

Sogar wenn kein Nenner vorhanden ist, ist die Funktion **comden** häufig ein gutes Mittel für das partielle Faktorisieren, wenn **factor()** zu langsam ist oder den Speicherplatz erschöpft.

Tipp: Geben Sie diese Funktionsdefinition **comden()** ein, und verwenden Sie sie regelmäßig als Alternative zu **comDenom()** und **factor()**.

Define *comden(exprn)=comDenom(exprn,abc)*
Done

$$\text{comden}\left(\frac{y^2+y}{(x+1)^2} + y^2 + y\right) \quad \frac{(x^2+2 \cdot x+2) \cdot y \cdot (y+1)}{(x+1)^2}$$

$$\text{comden}\left(\frac{1234 \cdot x^2 \cdot (y^3-y) + 2468 \cdot x \cdot (y^2-1)}{1234 \cdot x \cdot (x \cdot y+2) \cdot (y^2-1)}\right)$$

completeSquare ()

Katalog >

completeSquare(AusdrOdGl, Var)
⇒ Ausdruck oder Gleichung

$$\text{completeSquare}(x^2+2 \cdot x+3, x) \quad (x+1)^2+2$$

$$\text{completeSquare}(x^2+2 \cdot x=3, x) \quad (x+1)^2=4$$

completeSquare(AusdrOdGl, Var^Potenz)
⇒ Ausdruck oder Gleichung

$$\text{completeSquare}(x^6+2 \cdot x^3+3 \cdot x^3) \quad (x^3+1)^2+2$$

completeSquare(AusdrOdGl, Var1, Var2 [...])⇒ Ausdruck oder Gleichung

$$\text{completeSquare}(x^2+4 \cdot x+y^2+6 \cdot y+3=0, x, y) \quad (x+2)^2+(y+3)^2=10$$

completeSquare(AusdrOdGl, {Var1, Var2 [...]})⇒ Ausdruck oder Gleichung

$$\text{completeSquare}(3 \cdot x^2+2 \cdot y+7, y^2+4 \cdot x=3, \{x, y\}) \quad 3 \cdot \left(x+\frac{2}{3}\right)^2+7 \cdot \left(y+\frac{1}{7}\right)^2=\frac{94}{21}$$

Konvertiert einen quadratischen Polynomausdruck der Form $a \cdot x^2+b \cdot x+c$ in die Form $a \cdot (x-h)^2+k$

- oder -

Konvertiert eine quadratische Gleichung der Form $a \cdot x^2+b \cdot x+c=d$ in die Form $a \cdot (x-h)^2=k$

$$\text{completeSquare}(x^2+2 \cdot x \cdot y, x, y) \quad (x+y)^2-y^2$$

Das erste Argument muss ein quadratischer Ausdruck oder eine Gleichung im Standardformat bezüglich des zweiten

completeSquare ()

Katalog >

Arguments sein.

Das zweite Argument muss ein einzelner univariater Term bzw. ein einzelner univariater Term hoch einer rationalen Potenz sein, z. B. x , y^2 oder $z^{(1/3)}$.

Die dritte und vierte Syntax versuchen, das Quadrat mit Bezug auf $Var1, Var2 [...]$ zu vervollständigen.

conj() (Komplex Konjugierte)

Katalog >

conj(Ausdr1)⇒Ausdruck

$$\text{conj}(1+2\cdot i) \quad 1-2\cdot i$$

conj(Liste l)⇒Liste

$$\text{conj}\begin{bmatrix} 2 & 1-3\cdot i \\ -i & -7 \end{bmatrix} \quad \begin{bmatrix} 2 & 1+3\cdot i \\ i & -7 \end{bmatrix}$$

conj(Matrix l)⇒Matrix

$$\text{conj}(z) \quad z$$

Gibt das komplexe Konjugierte des Arguments zurück.

$$\text{conj}(x+i\cdot y) \quad x-y\cdot i$$

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt.

constructMat()

Katalog >

constructMat

(Ausdr,Var1,Var2,AnzZeilen,AnzSpalten) ⇒Matrix

$$\text{constructMat}\left(\frac{1}{i+j}, i, j, 3, 4\right) \quad \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 4 & 5 \\ 1 & 1 & 1 & 1 \\ 3 & 4 & 5 & 6 \\ 1 & 1 & 1 & 1 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

Gibt eine Matrix auf der Basis der Argumente zurück.

Ausdr ist ein Ausdruck in Variablen Var1 und Var2. Die Elemente in der resultierenden Matrix ergeben sich durch Berechnung von Ausdr für jeden inkrementierten Wert von Var1 und Var2.

Var1 wird automatisch von 1 bis AnzZeilen inkrementiert. In jeder Zeile wird Var2 inkrementiert von 1 bis AnzSpalten.

CopyVar

Katalog >

CopyVar *Var1, Var2*

CopyVar *Var1., Var2.*

CopyVar *Var1, Var2* kopiert den Wert der Variablen *Var1* auf die Variable *Var2* und erstellt ggf. *Var2*. Variable *Var1* muss einen Wert haben.

Wenn *Var1* der Name einer vorhandenen benutzerdefinierten Funktion ist, wird die Definition dieser Funktion nach Funktion *Var2* kopiert. Funktion *Var1* muss definiert sein.

Var1 muss die Benennungsregeln für Variablen erfüllen oder muss ein indirekter Ausdruck sein, der sich zu einem Variablenamen vereinfachen lässt, der den Regeln entspricht.

CopyVar *Var1., Var2.* kopiert alle Mitglieder der *Var1.*-Variablengruppe auf die *Var2.*-Gruppe und erstellt ggf. *Var2..*

Var1. muss der Name einer bestehenden Variablengruppe sein, wie die Statistikergebnisse *stat.* *nn* oder Variablen, die mit der Funktion **LibShortcut()** erstellt wurden. Wenn *Var2.* schon vorhanden ist, ersetzt dieser Befehl alle Mitglieder, die zu beiden Gruppen gehören, und fügt die Mitglieder hinzu, die noch nicht vorhanden sind. Wenn einer oder mehrere Teile von *Var2.* gesperrt ist/sind, wird kein Teil von *Var2.* geändert.

Define $a(x)=\frac{1}{x}$	Done
Define $b(x)=x^2$	Done
CopyVar <i>a,c: c(4)</i>	$\frac{1}{4}$
CopyVar <i>b,c: c(4)</i>	16

<i>aa.a:=45</i>	45
<i>aa.b:=6.78</i>	6.78
CopyVar <i>aa.,bb.</i>	Done
getVarInfo()	
<i>aa.a</i> "NUM" "□" 0	
<i>aa.b</i> "NUM" "□" 0,	
<i>bb.a</i> "NUM" "□" 0	
<i>bb.b</i> "NUM" "□" 0	

corrMat() (Korrelationsmatrix)

Katalog >

corrMat(*Liste1, Liste2[, ..., Liste20]*)

Berechnet die Korrelationsmatrix für die erweiterte Matrix [*Liste1* *Liste2* ... *Liste20*].

►cos

Katalog >

Ausdr **►cos**

Hinweis: Sie können diesen Operator über

die Tastatur Ihres Computers eingeben, indem Sie @>cos eintippen.

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

$$1 - (\cos(x))^2$$

Drückt Ausdr durch Kosinus aus. Dies ist ein Anzeigeumwandlungsoperator. Er kann nur am Ende der Eingabezeile verwendet werden.

►cos reduziert alle Potenzen von

$$\sin(\dots) \text{ modulo } 1 - \cos(\dots)^2,$$

so dass alle verbleibenden Potenzen von cos (...) Exponenten im Bereich (0, 2) haben. Deshalb enthält das Ergebnis dann und nur dann kein sin(...), wenn sin(...) im gegebenen Ausdruck nur bei geraden Potenzen auftritt.

Hinweis: Dieser Umrechnungsoperator wird im Winkelmodus Grad oder Neugrad (Gon) nicht unterstützt. Bevor Sie ihn verwenden, müssen Sie sicherstellen, dass der Winkelmodus auf Radian eingestellt ist und Ausdr keine expliziten Verweise auf Winkel in Grad oder Neugrad enthält.

cos() (Kosinus)

 Taste

cos(Ausdr1)⇒Ausdruck

Im Grad-Modus:

$\cos\left(\frac{\pi}{4}\right)$	$\frac{\sqrt{2}}{2}$
$\cos(45)$	$\frac{\sqrt{2}}{2}$
$\cos(\{0,60,90\})$	$\left\{1, \frac{1}{2}, 0\right\}$

cos(Liste1)⇒Liste

cos(Ausdr1) gibt den Kosinus des Arguments als Ausdruck zurück.

cos(Liste1) gibt in Form einer Liste für jedes Element in Liste1 den Kosinus zurück.

Hinweis: Der als Argument angegebene Winkel wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, G oder r benutzen, um den Winkelmodus vorübergehend aufzuheben.

Im Neugrad-Modus:

$\cos(\{0,50,100\})$	$\left\{1, \frac{\sqrt{2}}{2}, 0\right\}$
----------------------	---

Im Bogenmaß-Modus:

$\cos\left(\frac{\pi}{4}\right)$	$\frac{\sqrt{2}}{2}$
$\cos(45^\circ)$	$\frac{\sqrt{2}}{2}$

cos(Quadratmatrix 1)⇒Quadratmatrix

Gibt den Matrix-Kosinus von *Quadratmatrix 1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Kosinus jedes einzelnen Elements.

Wenn eine skalare Funktion $f(A)$ auf *Quadratmatrix 1* (A) angewendet wird, erfolgt die Berechnung des Ergebnisses durch den Algorithmus:

Berechnung der Eigenwerte (λ_i) und Eigenvektoren (V_i) von A .

Quadratmatrix 1 muss diagonalisierbar sein. Sie darf auch keine symbolischen Variablen ohne zugewiesene Werte enthalten.

Bildung der Matrizen:

$$B = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \lambda_n \end{bmatrix} \text{ and } X = [V_1, V_2, \dots, V_n]$$

Dann ist $A = X B X^{-1}$ und $f(A) = X f(B) X^{-1}$.
Beispiel: $\cos(A) = X \cos(B) X^{-1}$, wobei:

$$\cos(B) =$$

$$\begin{bmatrix} \cos(\lambda_1) & 0 & \dots & 0 \\ 0 & \cos(\lambda_2) & \dots & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & \cos(\lambda_n) \end{bmatrix}$$

Alle Berechnungen werden unter Verwendung von Fließkomma-Operationen ausgeführt.

Im Bogenmaß-Modus:

$\cos\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.212493 & 0.205064 & 0.121389 \\ 0.160871 & 0.259042 & 0.037126 \\ 0.248079 & -0.090153 & 0.218972 \end{bmatrix}$
--	---

cos⁻¹() (Arkuskosinus)

trig Taste

cos⁻¹(Ausdr1)⇒Ausdruck

Im Grad-Modus:

cos⁻¹(Liste1)⇒Liste

cos⁻¹(1)

0

cos⁻¹(Ausdr1) gibt den Winkel, dessen Kosinus *Ausdr1* ist, als Ausdruck zurück.

cos⁻¹(Liste1) gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Kosinus zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie arccos (...) eintippen.

cos⁻¹(Quadratmatrix1)⇒Quadratmatrix

Gibt den inversen Matrix-Kosinus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Kosinus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt cos().

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Neugrad-Modus:

cos⁻¹(0)

100

Im Bogenmaß-Modus:

cos⁻¹({0,0,2,0,5})

{ $\frac{\pi}{2}$, 1.36944, 1.0472}

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

cos⁻¹ $\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$

[1.73485+0.064606·i -1.49086+2.10514
-0.725533+1.51594·i 0.623491+0.77836·i
-2.08316+2.63205·i 1.79018-1.27182·i]

Um das ganze Ergebnis zu sehen, drücken Sie ▲ und verwenden dann ▲ und ▶, um den Cursor zu bewegen.

cosh() (Cosinus hyperbolicus)

Katalog >

cosh(Ausdr1)⇒Ausdruck

Im Grad-Modus:

cosh(Liste1)⇒Liste

cosh $\left(\left(\frac{\pi}{4}\right)^r\right)$

cosh(45)

cosh(Ausdr1) gibt den Cosinus hyperbolicus des Arguments als Ausdruck zurück.

cosh(Liste1) gibt in Form einer Liste für jedes Element aus *Liste1* den Cosinus hyperbolicus zurück.

cosh(Quadratmatrix1)⇒Quadratmatrix

Im Bogenmaß-Modus:

cosh() (Cosinus hyperbolicus)

Katalog >

Gibt den Matrix-Cosinus hyperbolicus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Cosinus hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

$$\cosh \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} 421.255 & 253.909 & 216.905 \\ 327.635 & 255.301 & 202.958 \\ 226.297 & 216.623 & 167.628 \end{bmatrix}$$

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

cosh⁻¹() (Arkuskosinus hyperbolicus)

Katalog >

cosh⁻¹(Ausdr1)⇒Ausdruck

$$\cosh^{-1}(1) = 0$$

cosh⁻¹(Liste1)⇒Liste

$$\cosh^{-1}(\{1,2,1,3\}) = \{0,1.37286,\cosh^{-1}(3)\}$$

cosh⁻¹(Ausdr1) gibt den inversen Cosinus hyperbolicus des Arguments als Ausdruck zurück.

cosh⁻¹(Liste1) gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Cosinus hyperbolicus zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arccosh(...)** eintippen.

cosh⁻¹(Quadratmatrix1)⇒Quadratmatrix

Gibt den inversen Matrix-Cosinus hyperbolicus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Cosinus hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\cosh^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} 2.52503+1.73485 \cdot i & -0.009241-1.4908i \\ 0.486969-0.725533 \cdot i & 1.66262+0.62349i \\ -0.322354-2.08316 \cdot i & 1.26707+1.79018i \end{bmatrix}$$

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

cot() (Kotangens)

Taste

cot(Ausdr1)⇒Ausdruck

Im Grad-Modus:

cot() (Kotangens)

trig Taste

cot(Liste1) \Rightarrow Liste

Gibt den Kotangens von *Ausdr1* oder eine Liste der Kotangens aller Elemente in *Liste1* zurück.

Hinweis: Der als Argument angegebene Winkel wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, G oder r benutzen, um den Winkelmodus vorübergehend aufzuheben.

cot(45)

1

Im Neugrad-Modus:

cot(50)

1

Im Bogenmaß-Modus:

$$\cot(\{1, 2, 1, 3\}) \quad \left\{ \frac{1}{\tan(1)}, -0.584848, \frac{1}{\tan(3)} \right\}$$

cot⁻¹⁽⁾ (Arkuskotangens)

trig Taste

cot⁻¹(Ausdr1) \Rightarrow Ausdruck

Im Grad-Modus:

cot⁻¹(1)

45

cot⁻¹(Liste1) \Rightarrow Liste

Gibt entweder den Winkel, dessen Kotangens *Ausdr1* ist, oder eine Liste der inversen Kotangens aller Elemente in *Liste1* zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie arccot (...) eintippen.

cot⁻¹(1)

50

Im Neugrad-Modus:

cot⁻¹(1)

π/4

Im Bogenmaß-Modus:

cot⁻¹(1)

π/4

coth() (Kotangens hyperbolicus)

Katalog >

coth(Ausdr1) \Rightarrow Ausdruck

coth(1.2)

1.19954

coth(Liste1) \Rightarrow Liste

coth(\{1, 3, 2\})

$$\left\{ \frac{1}{\tanh(1)}, 1.00333 \right\}$$

Gibt den hyperbolischen Kotangens von *Ausdr1* oder eine Liste der hyperbolischen Kotangens aller Elemente in *Liste1* zurück.

coth⁻¹⁽⁾ (Arkuskotangens hyperbolicus)

Katalog >

coth^{-1(Ausdr1)⇒Ausdruck}

coth^{-1(Liste1)⇒Liste}

Gibt den inversen hyperbolischen Kotangens von *Ausdr1* oder eine Liste der inversen hyperbolischen Kotangens aller Elemente in *Liste1* zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arccoth (...) eintippen.**

coth ^{-1(3.5)}	0.293893
coth ^{-1({-2,2,1,6})}	$\left\{ \frac{-\ln(3)}{2}, 0.518046, \frac{\ln\left(\frac{7}{5}\right)}{2} \right\}$

count() (zähle)

Katalog >

count(Wert1oderListe1 [,Wert2oderListe2 [...]])⇒Wert

Gibt die kumulierte Anzahl aller Elemente in den Argumenten zurück, deren Auswertungsergebnisse numerische Werte sind.

Jedes Argument kann ein Ausdruck, ein Wert, eine Liste oder eine Matrix sein. Sie können Datenarten mischen und Argumente unterschiedlicher Dimensionen verwenden.

Für eine Liste, eine Matrix oder einen Zellenbereich wird jedes Element daraufhin ausgewertet, ob es in die Zählung eingeschlossen werden soll.

Innerhalb der Lists & Spreadsheet Applikation können Sie anstelle eines beliebigen Arguments auch einen Zellenbereich verwenden.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

count(2,4,6)	3
count({2,4,6})	3
count(2,{4,6},{8 10} 12 14)	7
count(1/2,3+4·i,undef,"hello",x+5.,sign(0))	2

Im letzten Beispiel werden nur $1/2$ und $3+4\cdot i$ gezählt. Die übrigen Argumente ergeben unter der Annahme, dass x nicht definiert ist, keine numerischen Werte.

countIf()

Katalog >

countIf(Liste,Kriterien)⇒Wert

Gibt die kumulierte Anzahl aller Elemente in der *Liste* zurück, die die festgelegten *Kriterien* erfüllen.

countIf({1,3,"abc",undef,3,1},3)	2
Zählt die Anzahl der Elemente, die 3 entsprechen.	

Kriterien können sein:

- Ein Wert, ein Ausdruck oder eine Zeichenfolge. So zählt zum Beispiel 3 nur Elemente in der *Liste*, die vereinfacht den Wert 3 ergeben.
- Ein Boolescher Ausdruck, der das Sonderzeichen ? als Platzhalter für jedes Element verwendet. Beispielsweise zählt ?<5 nur die Elemente in der *Liste*, die kleiner als 5 sind.

Innerhalb der Lists & Spreadsheet Applikation können Sie anstelle der *Liste* auch einen Zellenbereich verwenden.

Leere (ungültige) Elemente in der Liste werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

Hinweis: Siehe auch **sumIf()**, Seite 190, und **frequency()**, Seite 82.

countIf({ "abc", "def", "abc", 3 }, "def") 1

Zählt die Anzahl der Elemente, die "def." entsprechen

countIf({ $x^{-2}, x^{-1}, 1, x, x^2$ }, x) 1

Zählt die Anzahl der Elemente, die x entsprechen; dieses Beispiel nimmt an, dass die Variable x nicht definiert ist.

countIf({ 1, 3, 5, 7, 9 }, ?<5) 2

Zählt 1 und 3.

countIf({ 1, 3, 5, 7, 9 }, 2 < ? < 8) 3

Zählt 3, 5 und 7.

countIf({ 1, 3, 5, 7, 9 }, ? < 4 or ? > 6) 4

Zählt 1, 3, 7 und 9.

cPolyRoots()

cPolyRoots(Poly,Var)⇒Liste

polyRoots(y^3+1, y) { -1 }

cPolyRoots(KoeffListe)⇒Liste

cPolyRoots(y^3+1)

$$\left\{ -1, \frac{1}{2} - \frac{\sqrt{3}}{2}i, \frac{1}{2} + \frac{\sqrt{3}}{2}i \right\}$$

Die erste Syntax **cPolyRoots(Poly,Var)** gibt eine Liste mit komplexen Wurzeln des Polynoms Poly bezüglich der Variablen Var zurück.

polyRoots($x^2+2\cdot x+1, x$) { -1, -1 }

Poly muss dabei ein Polynom in einer Variablen sein.

cPolyRoots({ 1, 2, 1 }) { -1, -1 }

Die zweite Syntax **cPolyRoots(KoeffListe)** liefert eine Liste mit komplexen Wurzeln für die Koeffizienten in KoeffListe.

Hinweis: Siehe auch **polyRoots()**, Seite 143.

crossP() (Kreuzprodukt)

crossP(Liste1, Liste2)⇒Liste

Gibt das Kreuzprodukt von *Liste1* und *Liste2* als Liste zurück.

Liste1 und *Liste2* müssen die gleiche Dimension besitzen, die entweder 2 oder 3 sein muss.

crossP(Vektor1, Vektor2)⇒Vektor

Gibt einen Zeilen- oder Spaltenvektor zurück (je nach den Argumenten), der das Kreuzprodukt von *Vektor1* und *Vektor2* ist.

Entweder müssen *Vektor1* und *Vektor2* beide Zeilenvektoren oder beide Spaltenvektoren sein. Beide Vektoren müssen die gleiche Dimension besitzen, die entweder 2 oder 3 sein muss.

crossP({*a1,b1*},{*a2,b2*})

{0,0,*a1·b2-a2·b1*}

crossP({0.1,2.2,-5},{1,-0.5,0})

{-2.5,-5.,-2.25}

crossP([1 2 3],[4 5 6]) [-3 6 -3]

crossP([1 2],[3 4]) [0 0 -2]

csc() (Kosekans)

csc(Ausdr1)⇒Ausdruck

Im Grad-Modus:

csc(45)

$\sqrt{2}$

csc(Liste1)⇒Liste

Gibt den Kosekans von *Ausdr1* oder eine Liste der Konsekans aller Elemente in *Liste1* zurück.

Im Neugrad-Modus:

csc(50)

$\sqrt{2}$

Im Bogenmaß-Modus:

$\csc\left\{1, \frac{\pi}{2}, \frac{\pi}{3}\right\}$ $\left\{\frac{1}{\sin(1)}, 1, \frac{2\sqrt{3}}{3}\right\}$

csc⁻¹() (Inverser Kosekans)

csc⁻¹(Ausdr1) ⇒ Ausdruck

Im Grad-Modus:

csc⁻¹() (Inverser Kosekans)

trig Taste

csc⁻¹(ListeI) ⇒ Liste

Gibt entweder den Winkel, dessen Kosekans *AusdrI* entspricht, oder eine Liste der inversen Kosekans aller Elemente in *ListeI* zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arccsc (...)** eintippen.

csc⁻¹(1)

90

Im Neugrad-Modus:

csc⁻¹(1)

100

Im Bogenmaß-Modus:

csc⁻¹({1,4,6})

$$\left\{ \frac{\pi}{2}, \sin^{-1}\left(\frac{1}{4}\right), \sin^{-1}\left(\frac{1}{6}\right) \right\}$$

csch() (Kosekans hyperbolicus)

Katalog >

csch(AusdrI) ⇒ Ausdruck

csch(ListeI) ⇒ Liste

Gibt den hyperbolischen Kosekans von *AusdrI* oder eine Liste der hyperbolischen Kosekans aller Elemente in *ListeI* zurück.

csch(3)

$$\frac{1}{\sinh(3)}$$

csch({1,2,1,4})

$$\left\{ \frac{1}{\sinh(1)}, 0.248641, \frac{1}{\sinh(4)} \right\}$$

csch⁻¹() (Inverser Kosekans hyperbolicus)

Katalog >

csch⁻¹(AusdrI) ⇒ Ausdruck

csch⁻¹(ListeI) ⇒ Liste

Gibt den inversen hyperbolischen Kosekans von *AusdrI* oder eine Liste der inversen hyperbolischen Kosekans aller Elemente in *ListeI* zurück.

csch⁻¹(1)

$$\sinh^{-1}(1)$$

csch⁻¹({1,2,1,3})

$$\left\{ \sinh^{-1}(1), 0.459815, \sinh^{-1}\left(\frac{1}{3}\right) \right\}$$

cSolve() (Komplexe Lösung)

Katalog >

cSolve(Gleichung, Var)⇒Boolescher Ausdruck

cSolve(Gleichung, Var=Schätzwert)
⇒Boolescher Ausdruck

cSolve(Ungleichung, Var)⇒Boolescher Ausdruck

cSolve($x^3 = -1, x$)
$x = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ or } x = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ or } x = -1$
solve($x^3 = -1, x$)

Gibt mögliche komplexe Lösungen einer Gleichung oder Ungleichung für *Var* zurück. Das Ziel ist, Kandidaten für alle reellen und nicht-reellen Lösungen zu erhalten. Selbst wenn *Gleichung* reel ist, erlaubt **cSolve()** nicht-reelle Lösungen im reellen Modus.

Obwohl alle undefinierten Variablen, die mit einem Unterstrich (_) enden, so verarbeitet werden, als wären sie reell, kann **cSolve()** Polynomgleichungen für komplexe Lösungen lösen.

cSolve() setzt den Bereich während der Berechnung zeitweise auf komplex, auch wenn der aktuelle Bereich reell ist. Im Komplexen benutzen Bruchexponenten mit ungeradem Nenner den Hauptzweig und sind nicht reell. Demzufolge sind Lösungen mit **solve()** für Gleichungen, die solche Bruchexponenten besitzen, nicht unbedingt eine Teilmenge der mit **cSolve()** erzielten Lösungen.

cSolve() beginnt mit exakten symbolischen Verfahren. Außer im Modus **Exakt** benutzt **cSolve()** bei Bedarf auch die iterative nähерungsweise polynomische Faktorisierung.

Hinweis: Siehe auch **cZeros()**, **solve()** und **zeros()**.

Hinweis: Enthält *Gleichung* Funktionen wie beispielsweise **abs()**, **angle()**, **conj()**, **real()** oder **imag()**, ist sie also kein Polynom, sollten Sie einen Unterstrich () drücken) hinter *Var* setzen. Standardmäßig

cSolve($x^3 = -1, x$)
$\left\{ \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i, \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i, -1 \right\}$

Im Modus Angezeigte Ziffern auf Fix 2:

exact(cSolve($x^5 + 4 \cdot x^4 + 5 \cdot x^3 - 6 \cdot x - 3 = 0, x$))
$x \cdot (x^4 + 4 \cdot x^3 + 5 \cdot x^2 - 6) = 3$
cSolve(Ans, x)
$x = -1.11 + 1.07 \cdot i \text{ or } x = -1.11 - 1.07 \cdot i \text{ or } x = -2.1$

Um das ganze Ergebnis zu sehen, drücken Sie und verwenden dann und , um den Cursor zu bewegen.

wird eine Variable als reeller Wert behandelt.

Bei Verwendung von *var_* wird die Variable als komplex behandelt.

cSolve($\text{conj}(z_)=1+i,z_$)

$z_ = 1-i$

Sie sollten *var_* auch für alle anderen Variablen in *Gleichung* verwenden, die nicht-reelle Werte haben könnten. Andernfalls erhalten Sie möglicherweise unerwartete Ergebnisse.

**cSolve(Glch1 and Glch2 [and...],
VarOderSchätzwert1,
VarOderSchätzwert2 [, ...])**
⇒Boolescher Ausdruck

**cSolve(Gleichungssystem,
VarOderSchätzwert1,
VarOderSchätzwert2 [, ...])**
⇒Boolescher Ausdruck

Gibt mögliche komplexe Lösungen eines algebraischen Gleichungssystems zurück, in dem jede *VarOderSchätzwert* eine Variable darstellt, nach der Sie die Gleichungen auflösen möchten.

Sie haben die Option, eine Ausgangsschätzung für eine Variable anzugeben. *VarOderSchätzwert* muss immer die folgende Form haben:

Variable

– oder –

Variable = reelle oder nicht-reelle Zahl

Beispiel: x ist gültig und $x=3+i$ ebenfalls.

Wenn alle Gleichungen Polynome sind und Sie KEINE Anfangsschätzwerte angeben, dann verwendet **cSolve()** das lexikalischeGröbner/Buchbergersche Eliminationsverfahren beim Versuch, **alle** komplexen Lösungen zu bestimmen.

Hinweis: In folgenden Beispielen wird ein Unterstrich (**ctrl** **U** drücken) verwendet, damit die Variablen als komplex behandelt werden.

cSolve() (Komplexe Lösung)

Katalog >

Komplexe Lösungen können, wie aus nebenstehendem Beispiel hervorgeht, sowohl reelle als auch nicht-reelle Lösungen enthalten.

$$\text{cSolve}\left(u_{_} \cdot v_{_} - u_{_} = v_{_} \text{ and } v_{_}^2 = -u_{_}, \{u_{_}, v_{_}\}\right)$$
$$u_{_} = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ and } v_{_} = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ or } u_{_} = \frac{1}{2} \downarrow$$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangleleft und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

Gleichungssysteme, die aus Polynomen bestehen, können zusätzliche Variablen ohne Wert aufweisen, die aber für numerische Werte stehen, welche später eingesetzt werden können.

Sie können auch Lösungsvariablen angeben, die in der Gleichung nicht erscheinen. Diese Lösungen verdeutlichen, dass Lösungsfamilien willkürliche Konstanten der Form ck enthalten können, wobei k ein ganzzahliger Index im Bereich 1 bis 255 ist.

Bei Gleichungssystemen aus Polynomen kann die Berechnungsduauer oder Speicherbelastung stark von der Reihenfolge abhängen, in welcher Sie die Lösungsvariablen angeben. Übersteigt Ihre erste Wahl die Speicherkapazität oder Ihre Geduld, versuchen Sie, die Variablen in der Gleichung und/oder *VarOderSchätzwert*-Liste umzuordnen.

Wenn Sie keine Schätzwerte angeben und eine Gleichung in einer Variablen nicht-polynomisch ist, aber alle Gleichungen in allen Lösungsvariablen linear sind, so verwendet **cSolve()** das Gaußsche Eliminationsverfahren beim Versuch, alle Lösungen zu bestimmen.

Wenn ein System weder in all seinen Variablen polynomial noch in seinen Lösungsvariablen linear ist, dann bestimmt **cSolve()** mindestens eine Lösung anhand eines iterativen näherungsweisen Verfahrens. Hierzu muss die Anzahl der Lösungsvariablen gleich der Gleichungsanzahl sein, und alle anderen Variablen in den Gleichungen müssen zu

$$\text{cSolve}\left(u_{_} \cdot v_{_} - u_{_} = c \cdot v_{_} \text{ and } v_{_}^2 = -u_{_}, \{u_{_}, v_{_}\}\right)$$
$$u_{_} = \frac{-(\sqrt{1-4 \cdot c_{_}}+1)^2}{4} \text{ and } v_{_} = \frac{\sqrt{1-4 \cdot c_{_}}+1}{2} \text{ or } u_{_} \rightarrow$$

$$\text{cSolve}\left(u_{_} \cdot v_{_} - u_{_} = v_{_} \text{ and } v_{_}^2 = -u_{_}, \{u_{_}, v_{_}, w_{_}\}\right)$$
$$u_{_} = \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ and } v_{_} = \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \text{ and } w_{_} = c8 \text{ or } u_{_} \rightarrow$$

$$\text{cSolve}\left(u_{_} + v_{_} = e^{w_{_}} \text{ and } u_{_} - v_{_} = i, \{u_{_}, v_{_}\}\right)$$
$$u_{_} = \frac{e^{w_{_}} + i}{2} \text{ and } v_{_} = \frac{e^{w_{_}} - i}{2}$$

$$\text{cSolve}\left(e^{z_{_}} = w_{_} \text{ and } w_{_} = z_{_}^2, \{w_{_}, z_{_}\}\right)$$
$$w_{_} = 0.494866 \text{ and } z_{_} = -0.703467$$

Zahlen vereinfachbar sein.

Zur Bestimmung einer nicht-reellen Lösung ist häufig ein nicht-reeller Schätzwert erforderlich. Für Konvergenz sollte ein Schätzwert ziemlich nahe bei einer Lösung liegen.

cSolve($e^{z_} = w_$ and $w_ = z_^2$, { $w_$, $z_$ }, 1+i)
 $w_ = 0.149606 + 4.8919 \cdot i$ and $z_ = 1.58805 + 1 \cdot i$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangleleft und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

CubicReg (Kubische Regression)

CubicReg X , Y , [$Häuf$] [, *Kategorie*, *Mit*]]

Berechnet die kubische polynomiale Regression $y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$ auf Listen X und Y mit der Häufigkeit $Häuf$. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden Datenpunkt X und Y an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden X und Y Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot x^3 + b \cdot x^2 + c \cdot x + d$
stat.a, stat.b, stat.c, stat.d	Regressionskoeffizienten

Ausgabeveriable	Beschreibung
stat.R ²	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X List</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häufigkeit</i> , <i>Kategorieliste</i> und <i>Mit Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y List</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häufigkeit</i> , <i>Kategorieliste</i> und <i>Mit Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

cumulativeSum() (kumulierteSumme)

Katalog > 

cumulativeSum(Liste1)⇒Liste

cumulativeSum({1,2,3,4}) {1,3,6,10}

Gibt eine Liste der kumulierten Summen der Elemente aus *Liste1* zurück, wobei bei Element 1 begonnen wird.

cumulativeSum(Matrix1)⇒Matrix

Gibt eine Matrix der kumulierten Summen der Elemente aus *Matrix1* zurück. Jedes Element ist die kumulierte Summe der Spalte von oben nach unten.

Ein leeres (ungültiges) Element in *Liste1* oder *Matrix1* erzeugt ein ungültiges Element in der resultierenden Liste oder Matrix. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1 \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

cumulativeSum(*m1*)

$$\begin{bmatrix} 1 & 2 \\ 4 & 6 \\ 9 & 12 \end{bmatrix}$$

Cycle (Zyklus)

Katalog > 

Cycle (Zyklus)

Funktionslisting, das die ganzen Zahlen von 1 bis 100 summiert und dabei 50 überspringt.

Übergibt die Programmsteuerung sofort an die nächste Wiederholung der aktuellen Schleife (**For**, **While** oder **Loop**).

Cycle ist außerhalb dieser drei Schleifenstrukturen (**For**, **While** oder **Loop**) nicht zulässig.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und

Cycle (Zyklus)

Katalog >

Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

```
Define g()=Func
  Local temp,i
  0→temp
  For i,1,100,1
    If i=50
      Cycle
      temp+i→temp
    EndFor
  Return temp
EndFunc
```

Done

g()

5000

►Cylind (Zylindervektor)

Katalog >

Vektor ►Cylind

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>**Cylind** eintippen.

Zeigt den Zeilen- oder Spaltenvektor in Zylinderkoordinaten [r,∠θ, z] an.

Vektor muss genau drei Elemente besitzen. Er kann entweder ein Zeilen- oder Spaltenvektor sein.

[2 2 3] ►Cylind

$\left[2\cdot\sqrt{2} \quad \angle\frac{\pi}{4} \quad 3\right]$

cZeros() (Komplexe Nullstellen)

Katalog >

cZeros(*Ausdr*, *Var*)⇒Liste

Gibt eine Liste möglicher reeller und nicht-reeller Werte für *Var* zurück, die *Ausdr*=0 ergeben. cZeros() tut dies durch Berechnung von

exp(list(cSolve(*Ausdr*=0, *Var*), *Var*). Ansonsten ist cZeros() ähnlich wie zeros().

Hinweis: Siehe auch cSolve(), solve() und zeros().

Hinweis: Ist *Ausdr* nicht-polynomial mit Funktionen wie beispielsweise abs(), angle(), conj(), real() oder imag(), sollten Sie einen Unterstrich (**ctrl** **underline** drücken) hinter *Var* setzen. Standardmäßig wird eine

Im Modus Angezeigte Ziffern auf Fix 3:

cZeros($x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3, x$)

{ -1.1138+1.07314·i, -1.1138-1.07314·i, -2, }

Um das ganze Ergebnis zu sehen, drücken Sie ▲ und verwenden dann ▲ und ▶, um den Cursor zu bewegen.

cZeros(conj(z_-)-1-i, z_-)

{ 1-i }

Variable als reeller Wert behandelt. Bei Verwendung von *var_* wird die Variable als komplex behandelt.

Sie sollten *var_* auch für alle anderen Variablen in *Ausdr* verwenden, die nicht-reelle Werte haben könnten. Andernfalls erhalten Sie möglicherweise unerwartete Ergebnisse.

```
cZeros({Ausdr1,Ausdr2[, ... ]},  
{  
VarOderSchätzwert1  
,VarOderSchätzwert2[, ... ]})⇒Matrix
```

Gibt mögliche Positionen zurück, in welchen die Ausdrücke gleichzeitig Null sind. Jeder *VarOderSchätzwert* steht für eine Unbekannte, deren Wert Sie suchen.

Sie haben die Option, eine Ausgangsschätzung für eine Variable anzugeben. *VarOderSchätzwert* muss immer die folgende Form haben:

Variable

– oder –

Variable = reelle oder nicht-reelle Zahl

Beispiel: x ist gültig und $x=3+i$ ebenfalls.

Wenn alle Ausdrücke Polynome sind und Sie KEINE Anfangsschätzwerte angeben, dann verwendet **cZeros()** das lexikalische Gröbner/Buchbergersche Eliminationsverfahren beim Versuch, alle komplexen Nullstellen zu bestimmen.

Komplexe Nullstellen können, wie aus nebenstehendem Beispiel hervorgeht, sowohl reelle als auch nicht-reelle Nullstellen enthalten.

Jede Zeile der sich ergebenden Matrix stellt eine alternative Nullstelle dar, wobei die Komponenten in derselben Reihenfolge wie in der *VarOderSchätzwert*-Liste angeordnet sind. Um eine Zeile zu erhalten ist die Matrix nach [Zeile] zu indizieren.

Hinweis: In folgenden Beispielen wird ein Unterstrich _ (**ctrl** **l** drücken) verwendet, damit die Variablen als komplex behandelt werden.

$$\text{cZeros}\left(\left\{u_{_} \cdot v_{_} - u_{_} - v_{_}, v_{_}^2 + u_{_}\right\}, \left\{u_{_}, v_{_}\right\}\right)$$

$$\begin{bmatrix} 0 & 0 \\ \frac{1-\sqrt{3}}{2} \cdot i & \frac{1+\sqrt{3}}{2} \cdot i \\ \frac{1+\sqrt{3}}{2} \cdot i & \frac{1-\sqrt{3}}{2} \cdot i \end{bmatrix}$$

Zeile 2 extrahieren:

$$\text{Ans}[2] \quad \left[\frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i \quad \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i \right]$$

Gleichungssysteme, die aus Polynomen bestehen, können zusätzliche Variablen haben, die zwar ohne Werte sind, aber gegebene numerische Werte darstellen, die später eingesetzt werden können.

Sie können auch unbekannte Variablen angeben, die nicht in den Ausdrücken erscheinen. Diese Nullstellen verdeutlichen, dass Nullstellenfamilien willkürliche Konstanten der Form c_k enthalten können, wobei k ein ganzzahliger Index im Bereich 1 bis 255 ist.

Bei polynomialem Gleichungssystemen kann die Berechnungsdauer oder Speicherbelastung stark von der Reihenfolge abhängen, in der Sie die Unbekannten angeben. Übersteigt Ihre erste Wahl die Speicherkapazität oder Ihre Geduld, versuchen Sie, die Variablen in den Ausdrücken und/oder der *VarOderSchätzwert*-Liste umzuordnen.

Wenn Sie keine Schätzwerte angeben und ein Ausdruck in einer Variablen nicht-polynomial ist, aber alle Ausdrücke in allen Unbekannten linear sind, so verwendet **czeros()** das Gaußsche Eliminationsverfahren beim Versuch, alle Nullstellen zu bestimmen.

Wenn ein System weder in all seinen Variablen polynomial noch in seinen Unbekannten linear ist, dann bestimmt **czeros()** mindestens eine Nullstelle anhand eines iterativen Näherungsverfahrens. Hierzu muss die Anzahl der Unbekannten gleich der Ausdruckanzahl sein, und alle anderen Variablen in den Ausdrücken müssen zu Zahlen vereinfachbar sein.

$$\text{cZeros}\left(\left\{ u_{_} \cdot v_{_} - u_{_} - c_{_} \cdot v_{_}, v_{_}^2 + u_{_} \right\}, \{u_{_}, v_{_}\} \right)$$

$$\begin{bmatrix} 0 & 0 \\ \frac{-(\sqrt{1-4 \cdot c_{_}}-1)^2}{4} & \frac{-(\sqrt{1-4 \cdot c_{_}}-1)}{2} \\ \frac{-(\sqrt{1-4 \cdot c_{_}}+1)^2}{4} & \frac{\sqrt{1-4 \cdot c_{_}}+1}{2} \end{bmatrix}$$

$$\text{cZeros}\left(\left\{ u_{_} \cdot v_{_} - u_{_} - v_{_}, v_{_}^2 + u_{_} \right\}, \{u_{_}, v_{_}, w_{_}\} \right)$$

$$\begin{bmatrix} 0 & 0 & c4 \\ \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i & c4 \\ \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i & \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i & c4 \end{bmatrix}$$

$$\text{cZeros}\left(\left\{ u_{_} + v_{_} - e^{w_{_}}, u_{_} - v_{_} - i \right\}, \{u_{_}, v_{_}\} \right)$$

$$\begin{bmatrix} e^{w_{_}} + i & e^{w_{_}} - i \\ 2 & 2 \end{bmatrix}$$

$$\text{cZeros}\left(\left\{ e^{z_{_} - w_{_}}, w_{_} - z_{_}^2 \right\}, \{w_{_}, z_{_}\} \right)$$

$$\begin{bmatrix} 0.494866 & -0.703467 \end{bmatrix}$$

cZeros() (Komplexe Nullstellen)

Katalog >

Zur Bestimmung einer nicht-reellen Nullstelle ist häufig ein nicht-reeller Schätzwert erforderlich. Für Konvergenz muss ein Schätzwert ziemlich nahe bei der Nullstelle liegen.

D

dbd()

Katalog >

dbd(Datum1,Datum2)⇒Wert

Zählt die tatsächlichen Tage und gibt die Anzahl der Tage zwischen *Datum1* und *Datum2* zurück.

Datum1 und *Datum2* können Zahlen oder Zahlenlisten innerhalb des Datumsbereichs des Standardkalenders sein. Wenn sowohl *Datum1* als auch *Datum2* Listen sind, müssen sie dieselbe Länge haben.

Datum1 und *Datum2* müssen innerhalb der Jahre 1950 und 2049 liegen.

Sie können Datumseingaben in zwei Formaten vornehmen. Die Datumsformate unterscheiden sich in der Anordnung der Dezimalstellen.

MM.TTJJ (üblicherweise in den USA verwendetes Format)

TTMM.JJ (üblicherweise in Europa verwendetes Format)

$$\text{cZeros}\left(\left\{ e^{z-w_-, w_- - z_-^2} \right\}, \left\{ w_-, z_- = 1+i \right\} \right)$$
$$[0.149606 + 4.8919 \cdot i \quad 1.58805 + 1.54022 \cdot i]$$

dbd(12.3103,1.0104) 1

dbd(1.0107,6.0107) 151

dbd(3112.03,101.04) 1

dbd(101.07,106.07) 151

►DD (Dezimalwinkel)

Katalog >

Zahl ►DD⇒Wert

Im Grad-Modus:

Liste1 ►DD⇒Liste

$(1.5^\circ) \blacktriangleright \text{DD}$

1.5°

Matrix1 ►DD⇒Matrix

$(45^\circ 22' 14.3") \blacktriangleright \text{DD}$

45.3706°

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>DD eintippen.

$\left\{ 45^\circ 22' 14.3", 60^\circ 0' 0" \right\} \blacktriangleright \text{DD}$

$\{ 45.3706^\circ, 60^\circ \}$

Gibt das Dezimaläquivalent des Arguments

Im Neugrad-Modus:

►DD (Dezimalwinkel)

Katalog >

zurück. Das Argument ist eine Zahl, eine Liste oder eine Matrix, die gemäß der Moduseinstellung als Neugrad, Bogenmaß oder Grad interpretiert wird.

1 ►DD

$\frac{9}{10}$

Im Bogenmaß-Modus:

(1.5) ►DD

85.9437°

►Decimal (Dezimal)

Katalog >

Ausdr1 ►Decimal⇒Ausdruck

$\frac{1}{3}$ ►Decimal

0.333333

Liste1 ►Decimal⇒Ausdruck

Matrix1 ►Decimal⇒Ausdruck

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Decimal eintippen.

Zeigt das Argument in Dezimalform an. Dieser Operator kann nur am Ende der Eingabezeile verwendet werden.

Definie

Katalog >

Define Var = Expression

Define Function(Param1, Param2, ...) = Expression

Definiert die Variable *Var* oder die benutzerdefinierte Funktion *Function*.

Parameter wie z.B. *Param1* enthalten Platzhalter zur Übergabe von Argumenten an die Funktion. Beim Aufrufen benutzerdefinierter Funktionen müssen Sie Argumente angeben (z.B. Werte oder Variablen), die zu den Parametern passen. Beim Aufruf wertet die Funktion *Ausdruck (Expression)* unter Verwendung der übergebenen Parameter aus.

Var und *Funktion (Function)* dürfen nicht der Name einer Systemvariablen oder einer integrierten Funktion / eines integrierten

Define g(x,y)=2·x-3·y	Done
g(1,2)	-4
1→a: 2→b: g(a,b)	-4
Define h(x)=when(x<2,2·x-3,-2·x+3)	Done
h(-3)	-9
h(4)	-5

Befehls sein.

Hinweis: Diese Form von **Definiere (Define)** ist gleichwertig mit der Ausführung folgenden Ausdrucks: *expression* → *Function(Param1,Param2)*.

Define Function(Param1, Param2, ...) =
Func
Block
EndFunc

Define <i>g(x,y)</i> =Func	<i>Done</i>
If <i>x>y</i> Then	
Return <i>x</i>	
Else	
Return <i>y</i>	
EndIf	
EndFunc	

Define Program(Param1, Param2, ...) =
Prgm
Block
EndPrgm

<i>g(3,-7)</i>	<i>3</i>
----------------	----------

In dieser Form kann die benutzerdefinierte Funktion bzw. das benutzerdefinierte Programm einen Block mit mehreren Anweisungen ausführen.

Block kann eine einzelne Anweisung oder eine Serie von Anweisungen in separaten Zeilen sein. *Block* kann auch Ausdrücke und Anweisungen enthalten (wie **If**, **Then**, **Else** und **For**).

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Hinweis: Siehe auch **Definiere LibPriv (Define LibPriv)**, Seite 55, und **Definiere LibPub (Define LibPub)**, Seite 56.

Define <i>g(x,y)</i> =Prgm	<i>Done</i>
If <i>x>y</i> Then	
Disp <i>x</i> , " greater than ", <i>y</i>	
Else	
Disp <i>x</i> , " not greater than ", <i>y</i>	
EndIf	
EndPrgm	

<i>g(3,-7)</i>	<i>3 greater than -7</i>
----------------	--------------------------

	<i>Done</i>
--	-------------

Definiere LibPriv (Define LibPriv)

Define LibPriv Var = Expression

Define LibPriv Function(Param1, Param2, ...) =
Expression

Define LibPriv Function(Param1, Param2, ...) = **Func**
Block

EndFunc**Define LibPriv Program(Param1, Param2, ...)** =**Prgm****Block****EndPrgm**

Funktioniert wie **Define**, definiert jedoch eine Variable, eine Funktion oder ein Programm für eine private Bibliothek. Private Funktionen und Programme werden im Katalog nicht angezeigt.

Hinweis: Siehe auch **Definiere (Define)**, Seite 54, und **Definiere LibPub (Define LibPub)**, Seite 56.

Define LibPub Var = Expression**Define LibPub Function(Param1, Param2, ...)** = **Expression****Define LibPub Function(Param1, Param2, ...)** = **Func****Block****EndFunc****Define LibPub Program(Param1, Param2, ...)** = **Prgm****Block****EndPrgm**

Funktioniert wie **Definiere (Define)**, definiert jedoch eine Variable, eine Funktion oder ein Programm für eine öffentliche Bibliothek. Öffentliche Funktionen und Programme werden im Katalog angezeigt, nachdem die Bibliothek gespeichert und aktualisiert wurde.

Hinweis: Siehe auch **Definiere (Define)**, Seite 54, und **Definiere LibPriv (Define LibPriv)**, Seite 55.

DelVar**Katalog >** **DelVar** *Var1[, Var2] [, Var3] ...***DelVar** *Var.*

Löscht die angegebene Variable oder Variablengruppe im Speicher.

Wenn eine oder mehrere Variablen gesperrt sind, wird bei diesem Befehl eine Fehlermeldung angezeigt und es werden nur die nicht gesperrten Variablen gelöscht. Siehe **unlock**, Seite 209.

DelVar *Var.* löscht alle Mitglieder der Variablengruppe *Var.* (wie die Statistikergebnisse *stat.nn* oder Variablen, die mit der Funktion **LibShortcut()** erstellt wurden). Der Punkt (.) in dieser Form des Befehls **DelVar** begrenzt ihn auf das Löschen einer Variablengruppe; die einfache Variable *Var* ist nicht davon betroffen.

$2 \rightarrow a$	2
$(a+2)^2$	16
DelVar <i>a</i>	<i>Done</i>
$(a+2)^2$	$(a+2)^2$

delVoid()**Katalog >** **delVoid(Liste1)⇒Liste**

Gibt eine Liste mit dem Inhalt von *Liste1* aus, wobei alle leeren (ungültigen) Elemente entfernt sind.

Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

<i>aa.a:=45</i>	45
<i>aa.b:=5.67</i>	5.67
<i>aa.c:=78.9</i>	78.9
getVarInfo()	$\begin{bmatrix} aa.a & "NUM" & "[45]" \\ aa.b & "NUM" & "[5.67]" \\ aa.c & "NUM" & "[78.9]" \end{bmatrix}$
DelVar <i>aa.</i>	<i>Done</i>
getVarInfo()	"NONE"

derivative()**Siehe *d()*, Seite 234.**

deSolve(*ODE1.Oder2.Ordnung, Var, abhängigeVar*) \Rightarrow eine allgemeine Lösung

Ergibt eine Gleichung, die explizit oder implizit eine allgemeine Lösung für die gewöhnliche Differentialgleichung erster oder zweiter Ordnung (ODE) angibt. In der ODE:

- Verwenden Sie einen Ableitungsstrich (drücken Sie) , um die erste Ableitung der abhängigen Variablen gegenüber der unabhängigen Variablen zu kennzeichnen.
- Kennzeichnen Sie die entsprechende zweite Ableitung mit zwei Strichen.

Das Zeichen ' wird nur für Ableitungen innerhalb von deSolve() verwendet.
Verwenden Sie für andere Fälle d().

Die allgemeine Lösung einer Gleichung erster Ordnung enthält eine willkürliche Konstante der Form ck, wobei k ein ganzzahliger Index im Bereich 1 bis 255 ist.
Die Lösung einer Gleichung zweiter Ordnung enthält zwei derartige Konstanten.

Wenden Sie solve() auf eine implizite Lösung an, wenn Sie versuchen möchten, diese in eine oder mehrere äquivalente explizite Lösungen zu konvertieren.

Beachten Sie beim Vergleich Ihrer Ergebnisse mit Lehrbuch- oder Handbuchlösungen bitte, dass die willkürlichen Konstanten in den verschiedenen Verfahren an unterschiedlichen Stellen in der Rechnung eingeführt werden, was zu unterschiedlichen allgemeinen Lösungen führen kann.

deSolve($y''+2 \cdot y' + y = x^2, x, y$)

$$y = (c3 \cdot x + c4) \cdot e^{-x} + x^2 - 4 \cdot x + 6$$

right(*Ans*) \rightarrow temp $(c3 \cdot x + c4) \cdot e^{-x} + x^2 - 4 \cdot x + 6$

$$\frac{d^2}{dx^2}\{\text{temp}\} + 2 \cdot \frac{d}{dx}\{\text{temp}\} + \text{temp} - x^2 = 0$$

DeiVar *temp*

Done

deSolve($y' = (\cos(y))^2, x, x, y$) $\tan(y) = \frac{x^2}{2} + c4$

solve(*Ans,y*) $y = \tan^{-1}\left(\frac{x^2 + 2 \cdot c4}{2}\right) + n3 \cdot \pi$

Ans|c4=c-1 and n3=0 $y = \tan^{-1}\left(\frac{x^2 + 2 \cdot (c-1)}{2}\right)$

deSolve() (Lösung)

Katalog >

deSolve

(ODE1.Ordnung and Anfangsbedingung, Var, abhängigeVar) ⇒ eine spezielle Lösung

Ergibt eine spezielle Lösung, die *ODE1.Ordnung* und *Anfangsbedingung* erfüllt. Dies ist in der Regel einfacher, als eine allgemeine Lösung zu bestimmen, Anfangswerte einzusetzen, nach der willkürlichen Konstanten aufzulösen und dann diesen Wert in die allgemeine Lösung einzusetzen.

Anfangsbedingung ist eine Gleichung der Form

abhängigeVar (unabhängigerAnfangswert) = abhängigerAnfangswert

Der *unabhängigeAnfangswert* und *abhängigeAnfangswert* können Variablen wie beispielsweise *x0* und *y0* ohne gespeicherte Werte sein. Die implizite Differentiation kann bei der Prüfung impliziter Lösungen behilflich sein.

deSolve

(ODE2.Ordnung and

Anfangsbedingung1

andAnfangsbedingung2, Var, abhängigeVar) ⇒ eine spezielle Lösung

Ergibt eine spezielle Lösung, die *ODE2.Ordnung* erfüllt und in einem Punkt einen bestimmten Wert der abhängigen Variablen und deren erster Ableitung aufweist.

Verwenden Sie für *Anfangsbedingung1* die Form

abhängigeVar (unabhängigerAnfangswert) = abhängigerAnfangswert

Verwenden Sie für *Anfangsbedingung2* die Form

$\sin(y) = (y \cdot e^x + \cos(y)) \cdot y' \rightarrow ode$	
$\sin(y) = (e^x \cdot y + \cos(y)) \cdot y'$	
deSolve(<i>ode</i> and $y(0)=0, x, y) \rightarrow soln$	
$\frac{(2 \cdot \sin(y) + y^2)}{2} = (e^x - 1) \cdot e^{-x} \cdot \sin(y)$	
$soln x=0 \text{ and } y=0$	true
<i>ode y'=impDif(soln,x,y)</i>	true
DelVar <i>ode,soln</i>	Done

$y'' = y^{\frac{3}{2}}$ and $y(0)=0$ and $y'(0)=0, t, y$	
$\frac{3}{2} \cdot y^{\frac{4}{3}} = t$	
solve(<i>Ans,y</i>)	
$y = \frac{2^{\frac{2}{3}} \cdot (3 \cdot t)^{\frac{3}{4}}}{4}$ and $t \geq 0$	

deSolve() (Lösung)

Katalog >

abhängigeVar (*unabhängigerAnfangswert*)
= *anfänglicher1.Ableitungswert*

deSolve

(

ODE2.Ordnung

andRandbedingung1 **andRandbedingung2**,
Var, *abhängigeVar*) \Rightarrow eine spezielle
Lösung

Ergibt eine spezielle Lösung, die
ODE2.Ordnung erfüllt und in zwei
verschiedenen Punkten angegebene Werte
aufweist.

$$\text{deSolve}\left(w'' = \frac{2 \cdot w'}{x} + \left(9 + \frac{2}{x^2}\right) \cdot w = x \cdot e^x \text{ and } w\left(\frac{\pi}{6}\right) = 0 \text{ and } w\left(\frac{\pi}{3}\right) = 0, x, w\right)$$

$$w = \frac{x \cdot e^x}{(\ln(e))^2 + 9} + \frac{e^{\frac{3}{2}} \cdot x \cdot \cos(3 \cdot x)}{(\ln(e))^2 + 9} - \frac{e^{\frac{6}{2}} \cdot x \cdot \sin(3 \cdot x)}{(\ln(e))^2 + 9}$$

deSolve($y'' = x$ and $y(0) = 1$ and $y'(2) = 3, x, y$)

$$y = \frac{x^3}{6} + x + 1$$

deSolve($y'' = 2 \cdot y'$ and $y(3) = 1$ and $y'(4) = 2, x, y$)

$$y = e^{2 \cdot x - 8} - e^{-2 + 1}$$

det() (Matrixdeterminante)

Katalog >

det(*Quadratmatrix*[, *Toleranz*])
 \Rightarrow Ausdruck

Gibt die Determinante von *Quadratmatrix* zurück.

Jedes Matrixelement wird wahlweise als 0 behandelt, wenn sein Absolutwert kleiner als *Toleranz* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Toleranz* ignoriert.

- Wenn Sie **ctrl enter** verwenden oder den Modus **Autom. oder Näherung** auf 'Approximiert' einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Toleranz* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:

$$5E-14 \cdot \max(\dim(\text{Quadratmatrix})) \cdot$$

$$\det\begin{bmatrix} a & b \\ c & d \end{bmatrix} = a \cdot d - b \cdot c$$

$$\det\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = -2$$

$$\det\left(\text{identity}(3) - x \cdot \begin{bmatrix} 1 & -2 & 3 \\ -2 & 4 & 1 \\ -6 & -2 & 7 \end{bmatrix}\right) = -(98 \cdot x^3 - 55 \cdot x^2 + 12 \cdot x - 1)$$

$$\begin{bmatrix} 1.E20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow \text{mat1} \quad \begin{bmatrix} 1.E20 & 1 \\ 0 & 1 \end{bmatrix}$$

$$\det(\text{mat1}) = 0$$

$$\det(\text{mat1}, 1) = 1.E20$$

rowNorm(Quadratmatrix)**diag() (Matrixdiagonale)****diag(Liste)⇒Matrix**

diag([2 4 6])

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

diag(Zeilensmatrix)⇒Matrix**diag(Spaltenmatrix)⇒Matrix**

Gibt eine Matrix mit den Werten der Argumentliste oder der Matrix in der Hauptdiagonalen zurück.

diag(Quadratmatrix)⇒Zeilensmatrix

Gibt eine Zeilensmatrix zurück, die die Elemente der Hauptdiagonalen von *Quadratmatrix* enthält.

Quadratmatrix muss eine quadratische Matrix sein.

$$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{bmatrix}$$

diag(Ans)

$$\begin{bmatrix} 4 & 6 & 8 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \\ 4 & 2 & 9 \end{bmatrix}$$

dim() (Dimension)**dim(Liste)⇒Ganzzahl**

dim({0,1,2})

3

Gibt die Dimension von *Liste* zurück.

dim(Matrix)⇒Liste

Gibt die Dimensionen von Matrix als Liste mit zwei Elementen zurück {Zeilen, Spalten}.

$$\dim \begin{pmatrix} 1 & -1 \\ 2 & -2 \\ 3 & 5 \end{pmatrix}$$

{3,2}

dim(String)⇒Ganzzahl

dim("Hello")

5

Gibt die Anzahl der in der Zeichenkette *String* enthaltenen Zeichen zurück.

dim("Hello "&"there")

11

Disp AusdruckOderString1 [,
AusdruckOderString2] ...

Zeigt die Argumente im *Calculator* Protokoll an. Die Argumente werden hintereinander angezeigt, dabei werden Leerzeichen zur Trennung verwendet.

Dies ist vor allem bei Programmen und Funktionen nützlich, um die Anzeige von Zwischenberechnungen zu gewährleisten.

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define *chars*(*start,end*)=Prgm

```
For i,start,end
Disp i, " ",char(i)
EndFor
EndPrgm
```

Done

chars(240,243)

240 ø

241 ñ

242 ö

243 ö

Done

►DMS (GMS)

Ausdr ►DMS

Liste ►DMS

Matrix ►DMS

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>►DMS eintippen.

Interpretiert den Parameter als Winkel und zeigt die entsprechenden GMS-Werte (engl. DMS) an (GGGGGG°MM'SS.ss"). Siehe °, ', " (Seite 243) zur Erläuterung des DMS-Formats (Grad, Minuten, Sekunden).

Hinweis: ►DMS wandelt Bogenmaß in Grad um, wenn es im Bogenmaß-Modus benutzt wird. Folgt auf die Eingabe das Grad-Symbol °, wird keine Umwandlung vorgenommen. Sie können ►DMS nur am Ende einer Eingabezeile benutzen.

Im Grad-Modus:

{45.371}►DMS 45°22'15.6"

{ { 45.371,60 } }►DMS { 45°22'15.6",60° }

domain()**domain(Ausdr1, Var) \Rightarrow Ausdruck**Gibt den Definitionsbereich von *Ausdr1* in Bezug auf *Var* zurück.**domain()** kann verwendet werden, um Definitionsbereiche von Funktionen zu erkunden. Es ist auf reelle und endliche Bereiche beschränkt.

Diese Funktionalität ist aufgrund von Schwächen von Computer-Algebra-Vereinfachungs- und Lösungsalgorithmen eingeschränkt.

Bestimmte Funktionen können nicht als Argumente für **domain()** verwendet werden, unabhängig davon, ob sie explizit oder innerhalb von benutzerdefinierten Variablen und Funktionen auftreten. In dem folgenden Beispiel kann der Ausdruck nicht vereinfacht werden weil $\int()$ eine nicht zulässige Funktion ist.

$$\text{domain} \left(\begin{pmatrix} x \\ \frac{1}{t} \ dt, x \\ 1 \end{pmatrix} \right) \cdot \text{domain} \left(\begin{pmatrix} x \\ \frac{1}{t} \ dt, x \\ 1 \end{pmatrix} \right)$$

dominanterTerm (), dominantTerm()**dominantTerm(Expr1, Var [, Point])**
 \Rightarrow expression**dominantTerm(Expr1, Var [, Point]) |**
Var>Point \Leftrightarrow expression**dominantTerm(Expr1, Var [, Point]) |**
Var<Point \Rightarrow expressionGibt den dominanten Term einer Potenzreihendarstellung von *Expr1* entwickelt um *Point* zurück. Der dominante Term ist derjenige, dessen Betrag nahe *Var* = *Point* am schnellsten anwächst. Die resultierende Potenz von (*Var* - *Point*) kann einen negativen und/oder

domain(x^2, x)	$-\infty < x < \infty$
domain($\frac{x+1}{x^2+2 \cdot x}, x$)	$x \neq -2 \text{ and } x \neq 0$
domain($(\sqrt{x})^2, x$)	$0 \leq x < \infty$
domain($\frac{1}{x+y}, y$)	$y \neq -x$

dominantTerm($\tan(\sin(x)) - \sin(\tan(x)), x$) $\frac{x^7}{30}$ dominantTerm($\frac{1 - \cos(x-1)}{(x-1)^3}, x, 1$) $\frac{1}{2 \cdot (x-1)}$ dominantTerm($x^{-2} \cdot \tan\left(x^{\frac{1}{3}}\right), x$) $\frac{1}{x^3}$ dominantTerm($\ln(x^2-1) \cdot x^{-2}, x$) $\frac{\ln(x \cdot \ln(x))}{x^2}$

dominanterTerm (), dominantTerm()

Bruchexponenten haben. Der Koeffizient dieser Potenz kann Logarithmen von (*Var* - *Point*) und andere Funktionen von *Var* enthalten, die von allen Potenzen von (*Var* - *Point*) dominiert werden, die dasselbe Exponentenzeichen haben.

Point ist vorgegeben als 0. *Point* kann ∞ oder $-\infty$ sein; in diesen Fällen ist der dominante Term eher derjenige mit dem größten Exponenten von *Var* als der mit dem kleinsten Exponenten von *Var*.

dominantTerm(...) gibt "**dominantTerm(...)**" zurück, wenn es keine Darstellung bestimmen kann wie für wesentliche Singularitäten wie z.B. $\sin(1/z)$ bei $z=0$, $e^{-1/z}$ bei $z=0$ oder e^z bei $z = \infty$ oder $-\infty$.

Wenn die Folge oder eine ihrer Ableitungen eine Sprungstelle bei *Point* hat, enthält das Ergebnis wahrscheinlich Unterausdrücke der Form $\text{sign}(\dots)$ oder $\text{abs}(\dots)$ für eine reelle Expansionsvariable oder $(-1)^{\text{floor}(\dots\text{angle}(\dots))}$ für eine komplexe

Expansionsvariable, die mit “_” endet.

Wenn Sie beabsichtigen, den dominanten Term nur für Werte auf einer Seite von *Point* zu verwenden, hängen Sie an

dominantTerm(...) je nach Bedarf “| *Var* > *Point*”, “| *Var* < *Point*”, “| “*Var* \geq *Point*” oder “*Var* \leq *Point*” an, um ein einfacheres Ergebnis zu erhalten.

dominantTerm() wird über Listen und Matrizen mit erstem Argument verteilt.

dominantTerm() können Sie verwenden, wenn Sie den einfachsten möglichen Ausdruck wissen möchten, der asymptotisch zu einem anderen Ausdruck wie *Var* \rightarrow *Point* ist. **dominantTerm()** ist ebenfalls hilfreich, wenn nicht klar ersichtlich ist, welchen Grad der erste Term einer Folge haben wird, der nicht Null ist und Sie nicht iterativ interaktiv oder mit einer Programmschleife schätzen möchten.

Hinweis: Siehe auch **series()**, Seite 168.

$\text{dominantTerm}\left(e^{\frac{-1}{z}}, z\right)$ $\text{dominantTerm}\left(e^{\frac{-1}{z}}, z, 0\right)$ $\text{dominantTerm}\left(\left(1 + \frac{1}{n}\right)^n, n, \infty\right)$ $\text{dominantTerm}\left(\tan^{-1}\left(\frac{1}{x}\right), x, 0\right) \quad \frac{\pi \cdot \text{sign}(x)}{2}$ $\text{dominantTerm}\left(\tan^{-1}\left(\frac{1}{x}\right), x\right) x > 0 \quad \frac{\pi}{2}$	e e e e e
--	--

dotP() (Skalarprodukt)**Katalog >** **dotP(Liste1, Liste2)⇒Ausdruck**dotP({*a,b,c*},{*d,e,f*}) $a \cdot d + b \cdot e + c \cdot f$

Gibt das Skalarprodukt zweier Listen zurück.

dotP({1,2},{5,6}) 17

dotP(Vektor1, Vektor2)⇒AusdruckdotP([*a b c*],[*d e f*]) $a \cdot d + b \cdot e + c \cdot f$

Gibt das Skalarprodukt zweier Vektoren zurück.

dotP([1 2 3],[4 5 6]) 32

Es müssen beide Zeilenvektoren oder beide Spaltenvektoren sein.

E**e^()** **Taste****e^(*Ausdr1*)⇒Ausdruck**e¹ e Gibt **e** hoch *Ausdr1* zurück.e¹. 2.71828**Hinweis:** Siehe auch Vorlage **e Exponent**, Seite 6.e^{3²} e^9 **Hinweis:** Das Drücken von **ex** zum Anzeigen von **e^()** ist nicht das gleiche wie das Drücken von **[E]** auf der Tastatur.Sie können eine komplexe Zahl in der polaren Form $r e^{i\theta}$ eingeben. Verwenden Sie diese aber nur im Winkelmodus Bogenmaß, da die Form im Grad- oder Neugrad-Modus einen Bereichsfehler verursacht.**e^(*Liste1*)⇒Liste**e^{1,1,0,5} {**e**,2.71828,1.64872}Gibt **e** hoch jedes Element der *Liste1* zurück.**e^(*Quadratmatrix1*)⇒Quadratmatrix**Ergebnis den Matrix-Exponenten von *Quadratmatrix1*. Dies ist nicht gleichbedeutend mit der Berechnung von **e** hoch jedes Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.
$$\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \begin{bmatrix} 782.209 & 559.617 & 456.509 \\ 680.546 & 488.795 & 396.521 \\ 524.929 & 371.222 & 307.879 \end{bmatrix}$$
Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

eff()**Katalog >** **eff(Nominalzinssatz, CpY)⇒Wert**

eff(5.75,12)

5.90398

Finanzfunktion, die den Nominalzinssatz *Nominalzinssatz* in einen jährlichen Effektivsatz konvertiert, wobei *CpY* als die Anzahl der Verzinsungsperioden pro Jahr gegeben ist.

Nominalzinssatz muss eine reelle Zahl sein und *CpY* muss eine reelle Zahl > 0 sein.

Hinweis: Siehe auch **nom()**, Seite 129.

eigVc() (Eigenvektor)**Katalog >** **eigVc(Quadratmatrix)⇒Matrix**

Im Komplex-Formatmodus "kartesisch":

Ergibt eine Matrix, welche die Eigenvektoren für eine reelle oder komplexe *Quadratmatrix* enthält, wobei jede Spalte des Ergebnisses zu einem Eigenwert gehört. Beachten Sie, dass ein Eigenvektor nicht eindeutig ist; er kann durch einen konstanten Faktor skaliert werden. Die Eigenvektoren sind normiert, d. h. wenn $V = [x_1, x_2, \dots, x_n]$, dann:

$$x_1^2 + x_2^2 + \dots + x_n^2 = 1$$

Quadratmatrix wird zunächst mit Ähnlichkeitstransformationen bearbeitet, bis die Zeilen- und Spaltennormen so nahe wie möglich bei demselben Wert liegen. Die *Quadratmatrix* wird dann auf die obere Hessenberg-Form reduziert, und die Eigenvektoren werden mit einer Schur-Faktorisierung berechnet.

$$\begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix} \rightarrow m1 \quad \begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix}$$

eigVc(*m1*)

-0.800906	0.767947	(
0.484029	0.573804+0.052258·i	0.5738·
0.352512	0.262687+0.096286·i	0.2626

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

eigVl() (Eigenwert)**Katalog >** **eigVl(Quadratmatrix)⇒Liste**

Im Komplex-Formatmodus "kartesisch":

Ergibt eine Liste von Eigenwerten einer reellen oder komplexen *Quadratmatrix*.

Quadratmatrix wird zunächst mit Ähnlichkeitstransformationen bearbeitet, bis die Zeilen- und Spaltennormen so nahe wie möglich bei demselben Wert liegen.

$$\begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix} \rightarrow m1 \quad \begin{bmatrix} -1 & 2 & 5 \\ 3 & -6 & 9 \\ 2 & -5 & 7 \end{bmatrix}$$

eigVl(*m1*)

{ -4.40941, 2.20471+0.763006·i, 2.20471-0·
--

eigVI() (Eigenwert)

Katalog >

Die *Quadratmatrix* wird dann auf die obere Hessenberg-Form reduziert, und die Eigenwerte werden aus der oberen Hessenberg-Matrix berechnet.

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

Else

Siehe If, Seite 92.

Elseif

Katalog >

```
If Boolescher Ausdr1 Then  
    Block1  
Elseif Boolescher Ausdr2 Then  
    Block2  
    :  
Elseif Boolescher AusdrN Then  
    BlockN  
EndIf  
    :
```

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

```
Define g(x)=Func  
    If x≤-5 Then  
        Return 5  
    ElseIf x>-5 and x<0 Then  
        Return -x  
    ElseIf x≥0 and x≠10 Then  
        Return x  
    ElseIf x=10 Then  
        Return 3  
    EndIf  
EndFunc
```

Done

EndFor

Siehe For, Seite 80.

EndFunc

Siehe Func, Seite 84.

EndIf

Siehe If, Seite 92.

EndLoop

Siehe Loop, Seite 116.

euler ()**Katalog >**

euler(Ausdr, Var, abhVar, {Var0, VarMax}, abhVar0, VarSchritt [, eulerSchritt]) \Rightarrow Matrix

euler(AusdrSystem, Var, ListeAbhVar, {Var0, VarMax}, ListeAbhVar0, VarSchritt [, eulerSchritt]) \Rightarrow Matrix

euler(AusdrListe, Var, ListeAbhVar, {Var0, VarMax}, ListeAbhVar0, VarSchritt [, eulerSchritt]) \Rightarrow Matrix

Verwendet die Euler-Methode zum Lösen des Systems

$$\frac{d \text{ depVar}}{d \text{ Var}} = \text{Expr}(\text{Var}, \text{depVar})$$

mit $\text{abhVar}(\text{Var0})=\text{abhVar0}$ auf dem Intervall $[\text{Var0}, \text{VarMax}]$. Gibt eine Matrix zurück, deren erste Zeile die Ausgabewerte von Var definiert und deren zweite Zeile den Wert der ersten Lösungskomponente an den entsprechenden Var -Werten definiert usw.

Ausdr ist die rechte Seite, die die gewöhnliche Differentialgleichung (ODE) definiert.

AusdrSystem ist das System rechter Seiten, welche das ODE-System definieren (entspricht der Ordnung abhängiger Variablen in *ListeAbhVar*).

AusdrListe ist eine Liste rechter Seiten,

Differentialgleichung:

$$y' = 0.001 \cdot y \cdot (100 - y) \text{ und } y(0) = 10$$

$$\begin{aligned} \text{euler}\left(0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1\right) \\ \begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9 & 11.8712 & 12.9174 & 14.042 \end{bmatrix} \end{aligned}$$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

Vergleichen Sie das vorstehende Ergebnis mit der exakten CAS-Lösung, die Sie erhalten, wenn Sie deSolve() und seqGen() verwenden:

$$\begin{aligned} \text{deSolve}(y' = 0.001 \cdot y \cdot (100 - y) \text{ and } y(0) = 10, t, y) \\ y = \frac{100 \cdot (1.10517)^t}{(1.10517)^t + 9}. \end{aligned}$$

$$\begin{aligned} \text{seqGen}\left(\frac{100 \cdot (1.10517)^t}{(1.10517)^t + 9}, t, y, \{0, 100\}\right) \\ \{10., 10.9367, 11.9494, 13.0423, 14.2189\} \end{aligned}$$

Gleichungssystem:

$$\begin{cases} y_1' = y_1 + 0.1 \cdot y_1 \cdot y_2 \\ y_2' = 3 \cdot y_2 - y_1 \cdot y_2 \end{cases}$$

$$\text{mit } y_1(0) = 2 \text{ und } y_2(0) = 5$$

euler ()

Katalog >

welche das ODE-System definieren (entspricht der Ordnung abhängiger Variablen in *ListeAbhVar*).

Var ist die unabhängige Variable.

ListeAbhVar ist eine Liste abhängiger Variablen.

{*Var0*, *VarMax*} ist eine Liste mit zwei Elementen, die die Funktion anweist, von *Var0* zu *VarMax* zu integrieren.

ListeAbhVar0 ist eine Liste von Anfangswerten für abhängige Variablen.

VarSchritt ist eine Zahl ungleich Null, sodass **sign**(*VarSchritt*) = **sign**(*VarMax*-*Var0*) und Lösungen an *Var0+i·VarSchritt* für alle i=0,1,2,... zurückgegeben werden, sodass *Var0+i·VarSchritt* in [*var0*,*VarMax*] ist (möglicherweise gibt es keinen Lösungswert an *VarMax*).

eulerSchritt ist eine positive ganze Zahl (standardmäßig 1), welche die Anzahl der Euler-Schritte zwischen Ausgabewerten bestimmt. Die tatsächliche von der Euler-Methode verwendete Schrittgröße ist *VarSchritt/eulerSchritt*.

$$\begin{aligned} \text{euler}\left(\left\{\begin{array}{l} y_1' + 0.1 \cdot y_1 \cdot y_2 \\ 3 \cdot y_2' - y_1 \cdot y_2 \end{array}\right., t, \{y_1, y_2\}, \{0.5\}, \{2.5\}, 1\right) \\ \left[\begin{array}{cccccc} 0. & 1. & 2. & 3. & 4. & 5. \\ 2. & 1. & 1. & 3. & 27. & 243. \\ 5. & 10. & 30. & 90. & 90. & -2070. \end{array}\right] \end{aligned}$$

exact() (Exakt)

Katalog >

exact(*Ausdr1* [, *Toleranz*]) \Rightarrow *Ausdruck*

$$\frac{1}{4}$$

exact(*Liste1* [, *Toleranz*]) \Rightarrow *Liste*

$$\frac{333333}{1000000}$$

exact(*Matrix1* [, *Toleranz*]) \Rightarrow *Matrix*

$$\frac{1}{3}$$

Benutzt den Rechenmodus 'Exakt' und gibt nach Möglichkeit die rationale Zahl zurück, die dem Argument äquivalent ist.

$$\frac{7x}{2} + y$$

Toleranz legt die Toleranz für die Umwandlung fest, wobei die Vorgabe 0 (null) ist.

$$\left\{ \frac{1}{5}, \frac{33}{100}, \frac{33}{8} \right\}$$

Exit (Abbruch)

Beendet den aktuellen **For**, **While**, oder **Loop** Block.

Exit ist außerhalb dieser drei Schleifenstrukturen (**For**, **While** oder **Loop**) nicht zulässig.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Funktionslisting:

```
Define g()=Func           Done
    Local temp,i
    0→temp
    For i,1,100,1
        temp+i→temp
    If temp>20 Then
        Exit
    EndIf
    EndFor
EndFunc
```

g()

21

►exp**Ausdr ►exp**

Drückt *Ausdr* durch die natürliche Exponentialfunktion e aus. Dies ist ein Anzeigeumwandlungsoperator. Er kann nur am Ende der Eingabezeile verwendet werden.

$\frac{d}{dx} \left(e^x + e^{-x} \right)$	2·sinh(x)
2·sinh(x)►exp	$e^x - e^{-x}$

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>exp eintippen.

exp() (e hoch x)

Taste

exp(Ausdr1)⇒Ausdruck

e^1	e
$e^{1.}$	2.71828
e^{3^2}	e^9

Gibt e hoch *Ausdr1* zurück.

Hinweis: Siehe auch Vorlage **e** Exponent, Seite 6.

Sie können eine komplexe Zahl in der polaren Form $r\text{ei}\theta$ eingeben. Verwenden Sie diese aber nur im Winkelmodus Bogenmaß, da die Form im Grad- oder Neugrad-Modus einen Bereichsfehler verursacht.

exp(Liste1)⇒Liste

$e^{\{1,1,0.5\}}$	{ e,2.71828,1.64872 }
-------------------	-----------------------

Gibt e hoch jedes Element der *Liste1* zurück.

exp() (e hoch x)

Taste

exp(Quadratmatrix1)⇒Quadratmatrix

Ergibt den Matrix-Exponenten von *Quadratmatrix1*. Dies ist nicht gleichbedeutend mit der Berechnung von e hoch jedes Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$e \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$	$\begin{bmatrix} 782.209 & 559.617 & 456.509 \\ 680.546 & 488.795 & 396.521 \\ 524.929 & 371.222 & 307.879 \end{bmatrix}$
--	---

exp@list() (Ausdruck in Liste)

Katalog >

exp@list(Ausdr,Var)⇒Liste

Untersucht *Ausdr* auf Gleichungen, die durch das Wort "or" getrennt sind und gibt eine Liste der rechten Seiten der Gleichungen in der Form *Var=Ausdr* zurück. Dies erlaubt Ihnen auf einfache Weise das Extrahieren mancher Lösungswerte, die in den Ergebnissen der Funktionen **solve()**, **cSolve()**, **fMin()** und **fMax()** enthalten sind.

Hinweis: **exp@list()** ist für die Funktionen **zeros** und **cZeros()** unnötig, da diese direkt eine Liste von Lösungswerten zurückgeben.

Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **exp@>list(...)** eintippen.

$$\text{solve}(x^2-x-2=0,x) \quad x=-1 \text{ or } x=2$$

$$\text{exp@list}(\text{solve}(x^2-x-2=0,x),x) \quad \{-1,2\}$$

expand() (Entwickle)

Katalog >

expand(Ausdr1 [, Var])⇒Ausdruck

$$\text{expand}((x+y+1)^2) \quad x^2+2 \cdot x \cdot y+2 \cdot x+y^2+2 \cdot y+1$$

expand(Liste1 [,Var])⇒Liste

$$\text{expand}\left(\frac{x^2-x+y^2-y}{x^2 \cdot y^2-x^2 \cdot y-x \cdot y^2+x \cdot y}\right) \quad \frac{1}{x-1}-\frac{1}{x}+\frac{1}{y-1}-\frac{1}{y}$$

expand(Matrix1 [,Var])⇒Matrix

expand(Ausdr1) gibt *Ausdr1* bezüglich sämtlicher Variablen entwickelt zurück. Die Entwicklung ist eine Polynomentwicklung für Polynome und eine

Partialbruchentwicklung für rationale Ausdrücke.

expand() versucht *Ausdr1* in eine Summe und/oder eine Differenz einfacher Ausdrücke umzuformen. Dagegen versucht **factor()** *Ausdr1* in ein Produkt und/oder einen Quotienten einfacher Faktoren umzuformen.

expand(Ausdr1,Var) entwickelt *Ausdr1* bezüglich *Var*. Gleichartige Potenzen von *Var* werden zusammengefasst. Die Terme und Faktoren werden mit *Var* als der Hauptvariablen sortiert. Es kann sein, dass als Nebeneffekt in gewissem Umfang eine Faktorisierung oder Entwicklung der zusammengefassten Koeffizienten auftritt. Verglichen mit dem Weglassen von *Var* spart dies häufig Zeit, Speicherplatz und Platz auf dem Bildschirm und macht den Ausdruck verständlicher.

Selbst wenn es nur eine Variable gibt, kann das Einbeziehen von *Var* eine vollständigere Faktorisierung des Nenners, die für die Partialbruchentwicklung benutzt wird, ermöglichen.

Tipp: Für rationale Ausdrücke ist **propFrac()** eine schnellere, aber weniger weitgehende Alternative zu **expand()**.

Hinweis: Siehe auch **comDenom()** zu einem Quotienten aus einem entwickelten Zähler und entwickeltem Nenner.

expand(Ausdr1,[Var]) vereinfacht auch Logarithmen und Bruchpotenzen ungeachtet von *Var*. Für weitere Zerlegungen von Logarithmen und Bruchpotenzen können Einschränkungen notwendig werden, um sicherzustellen, dass manche Faktoren nicht negativ sind.

expand(Ausdr1, [Var]) vereinfacht auch Absolutwerte, **sign()** und Exponenten ungeachtet von *Var*.

Hinweis: Siehe auch **tExpand()** zur

$$\begin{aligned} \text{expand}\left((x+y+1)^2, y\right) &= y^2 + 2 \cdot y \cdot (x+1) + (x+1)^2 \\ \text{expand}\left((x+y+1)^2, x\right) &= x^2 + 2 \cdot x \cdot (y+1) + (y+1)^2 \\ \text{expand}\left(\frac{x^2 - x + y^2 - y}{x^2 \cdot y^2 - x^2 \cdot y - x \cdot y^2 + x \cdot y}, y\right) &= \\ &\quad \frac{1}{y-1} - \frac{1}{y} + \frac{1}{x \cdot (x-1)} \\ \text{expand}(Ans, x) &= \frac{1}{x-1} - \frac{1}{x} + \frac{1}{x \cdot (y-1)} \end{aligned}$$

$$\begin{aligned} \text{expand}\left(\frac{x^3 + x^2 - 2}{x^2 - 2}\right) &= \frac{2 \cdot x}{x^2 - 2} + x + 1 \\ \text{expand}(Ans, x) &= \frac{1}{x - \sqrt{2}} + \frac{1}{x + \sqrt{2}} + x + 1 \end{aligned}$$

$$\begin{aligned} \ln(2 \cdot x \cdot y) + \sqrt{2 \cdot x \cdot y} &= \ln(2 \cdot x \cdot y) + \sqrt{2 \cdot x \cdot y} \\ \text{expand}(Ans) &= \ln(x \cdot y) + \sqrt{2 \cdot \sqrt{x \cdot y} + \ln(2)} \\ \text{expand}(Ans), y \geq 0 &= \ln(x) + \sqrt{2 \cdot \sqrt{x} \cdot \sqrt{y} + \ln(y)} + \ln(2) \\ \text{sign}(x \cdot y) + |x \cdot y| + e^{2 \cdot x + y} &= \text{sign}(x \cdot y) + |x \cdot y| + e^{2 \cdot x + y} \\ \text{expand}(Ans) &= e^{2 \cdot x + y} + \text{sign}(x \cdot y) + |x \cdot y| \\ \text{sign}(x) \cdot \text{sign}(y) + |x| \cdot |y| + (e^x)^2 \cdot e^y &= \end{aligned}$$

expand() (Entwickle)

Katalog >

trigonometrischen Entwicklung von
Winkelsummen und -produkten.

expr() (String in Ausdruck)

Katalog >

expr(String)⇒Ausdruck

Gibt die in *String* enthaltene Zeichenkette
als Ausdruck zurück und führt diesen sofort
aus.

expr("1+2+x^2+x")	$x^2 + x + 3$
expr("expand((1+x)^2)")	$x^2 + 2 \cdot x + 1$
"Define cube(x)=x^3" → funcstr	"Define cube(x)=x^3"
expr(funcstr)	Done
cube(2)	8

ExpReg (Exponentielle Regression)

Katalog >

ExpReg X, Y [, Häuf] [, Kategorie, Mit]

Berechnet die exponentielle Regression $y = a \cdot b^x$ auf
Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine
Zusammenfassung der Ergebnisse wird in der
Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension
besitzen.

X und *Y* sind Listen von unabhängigen und
abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten.
Jedes Element in *Häuf* gibt die Häufigkeit für jeden
entsprechenden Datenpunkt *X* und *Y* an. Der
Standardwert ist 1. Alle Elemente müssen
Ganzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die
entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren
Kategoriecodes. Nur solche Datenelemente, deren
Kategoriecode in dieser Liste enthalten ist, sind in der
Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente
in einer Liste finden Sie unter "Leere (ungültige)
Elemente" (Seite 252).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot (b)^x$
stat.a, stat.b	Regressionskoeffizienten
stat.r ²	Koeffizient der linearen Bestimmtheit für transformierte Daten
stat.r	Korrelationskoeffizient für transformierte Daten ($x, \ln(y)$)
stat.Resid	Mit dem exponentiellen Modell verknüpfte Residuen
stat.ResidTrans	Residuum für die lineare Anpassung der transformierten Daten.
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X List</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häufigkeit</i> , <i>Kategorieliste</i> und <i>Mit Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y List</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häufigkeit</i> , <i>Kategorieliste</i> und <i>Mit Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

F

factor() (Faktorisiere)

Katalog >

factor(Ausdr1[, Var])⇒Ausdruck

$$\text{factor}(a^3 \cdot x^2 - a \cdot x^2 - a^3 + a) \\ a \cdot (a-1) \cdot (a+1) \cdot (x-1) \cdot (x+1)$$

factor(Liste1[, Var])⇒Liste

$$\text{factor}(x^2 + 1) \\ x^2 + 1$$

factor(Matrix1[, Var])⇒Matrix

$$\text{factor}(x^2 - 4) \\ (x-2) \cdot (x+2)$$

factor(Ausdr1) gibt Ausdr1 nach allen seinen Variablen bezüglich eines gemeinsamen Nenners faktorisiert zurück.

$$\text{factor}(x^2 - 3) \\ x^2 - 3$$

$$\text{factor}(x^2 - a) \\ x^2 - a$$

Ausdr1 wird soweit wie möglich in lineare rationale Faktoren aufgelöst, selbst wenn dies die Einführung neuer nicht-reeller Unterausdrücke bedeutet. Diese Alternative ist angemessen, wenn Sie die Faktorisierung bezüglich mehr als einer Variablen vornehmen möchten.

factor(Ausdr1,Var) gibt Ausdr1 nach der Variablen Var faktorisiert zurück.

$$\text{factor}(a^3 \cdot x^2 - a \cdot x^2 - a^3 + a, x) \\ a \cdot (a^2 - 1) \cdot (x-1) \cdot (x+1)$$

Ausdr1 wird soweit wie möglich in reelle Faktoren aufgelöst, die linear in Var sind, selbst wenn dadurch irrationale Konstanten oder Unterausdrücke, die in anderen Variablen irrational sind, eingeführt

$$\text{factor}(x^2 - 3, x) \\ (x + \sqrt{3}) \cdot (x - \sqrt{3})$$

$$\text{factor}(x^2 - a, x) \\ (x + \sqrt{a}) \cdot (x - \sqrt{a})$$

werden.

Die Faktoren und ihre Terme werden mit *Var* als Hauptvariable sortiert. Gleichartige Potenzen von *Var* werden in jedem Faktor zusammengefasst. Beziehen Sie *Var* ein, wenn die Faktorisierung nur bezüglich dieser Variablen benötigt wird und Sie irrationale Ausdrücke in anderen Variablen akzeptieren möchten, um die Faktorisierung bezüglich *Var* so weit wie möglich vorzunehmen. Es kann sein, dass als Nebeneffekt in gewissem Umfang eine Faktorisierung nach anderen Variablen auftritt.

Bei der Einstellung Auto für den Modus **Auto oder Näherung** ermöglicht die Einbeziehung von *Var* auch eine Näherung mit Gleitkommakoeffizienten in Fällen, wo irrationale Koeffizienten nicht explizit bezüglich der integrierten Funktionen ausgedrückt werden können. Selbst wenn es nur eine Variable gibt, kann das Einbeziehen von *Var* eine vollständigere Faktorisierung ermöglichen.

Hinweis: Siehe auch **comDenom()** zu einer schnellen partiellen Faktorisierung, wenn **factor()** zu langsam ist oder den Speicherplatz erschöpft.

Hinweis: Siehe auch **cFactor()** zur kompletten Faktorisierung bis zu komplexen Koeffizienten, um lineare Faktoren zu erhalten.

factor(RationaleZahl) ergibt die rationale Zahl in Primfaktoren zerlegt. Bei zusammengesetzten Zahlen nimmt die Berechnungsdauer exponentiell mit der Anzahl an Stellen im zweitgrößten Faktor zu. Das Faktorisieren einer 30-stelligen ganzen Zahl kann beispielsweise länger als einen Tag dauern und das Faktorisieren einer 100-stelligen Zahl mehr als ein Jahrhundert.

So halten Sie eine Berechnung manuell an:

factor($x^5+4 \cdot x^4+5 \cdot x^3-6 \cdot x-3$)	$x^5+4 \cdot x^4+5 \cdot x^3-6 \cdot x-3$
---	---

factor($x^5+4 \cdot x^4+5 \cdot x^3-6 \cdot x-3, x$)	$(x-0.964673) \cdot (x+0.611649) \cdot (x+2.12543) \cdot (x^2+3.125 \cdot x+3.125)$
--	---

factor(152417172689)	123457 · 1234577
isPrime(152417172689)	false

- **Handheld:** Halten Sie die Taste **[on]** gedrückt und drücken Sie mehrmals **[enter]**.
- **Windows®:** Halten Sie die Taste **F12** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **Macintosh®:** Halten Sie die Taste **F5** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **iPad®:** Die App zeigt eine Eingabeaufforderung an. Sie können weiter warten oder abbrechen.

Möchten Sie hingegen lediglich feststellen, ob es sich bei einer Zahl um eine Primzahl handelt, verwenden Sie **isPrime()**. Dieser Vorgang ist wesentlich schneller, insbesondere dann, wenn *RationaleZahl* keine Primzahl ist und der zweitgrößte Faktor mehr als fünf Stellen aufweist.

Fcdf
 (
UntGrenze,
ObGrenze,
FreiGradZähler,
FreiGradNenner)
 ⇒ Zahl, wenn *UntGrenze* und *ObGrenze* Zahlen sind, Liste, wenn *UntGrenze* und *ObGrenze* Listen sind

FCdf
 (
UntGrenze,
ObGrenze,
FreiGradZähler,
FreiGradNenner)
 ⇒ Zahl, wenn *UntGrenze* und *ObGrenze* Zahlen sind, Liste, wenn *UntGrenze* und *ObGrenze* Listen sind

Berechnet die **F** Verteilungswahrscheinlichkeit zwischen *UntereGrenze* und *ObereGrenze* für die angegebenen *FreiGradZähler* (Freiheitsgrade) und *FreiGradNenner*.

Für $P(X \leq ObereGrenze)$, *UntGrenze* =0 setzen.

Fill (Füllen)

Katalog >

Fill Ausdr, MatrixVar⇒Matrix

Ersetzt jedes Element in der Variablen *MatrixVar* durch *Ausdr*.

MatrixVar muss bereits vorhanden sein.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow amatrix$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
Fill 1.01,amatrix	Done
amatrix	$\begin{bmatrix} 1.01 & 1.01 \\ 1.01 & 1.01 \end{bmatrix}$

Fill Ausdr, ListeVar⇒Liste

Ersetzt jedes Element in der Variablen *ListeVar* durch *Ausdr*.

ListeVar muss bereits vorhanden sein.

$\{1,2,3,4,5\} \rightarrow alist$	$\{1,2,3,4,5\}$
Fill 1.01,alist	Done
alist	$\{1.01,1.01,1.01,1.01,1.01\}$

FiveNumSummary

Katalog >

FiveNumSummary X,[Häuf],[Kategorie,Mit]

Bietet eine gekürzte Version der Statistik mit 1 Variablen auf Liste *X*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

X stellt eine Liste mit den Daten dar.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*-Wert an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Ein leeres (ungültiges) Element in einer der Listen *X*, *Freq* oder *Kategorie* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

Ausgabeverable	Beschreibung
stat.MinX	Minimum der x-Werte
stat.Q ₁ X	1. Quartil von x
stat.MedianX	Median von x

AusgabevARIABLE	Beschreibung
stat.Q ₃ X	3. Quartil von x
stat.MaxX	Maximum der x-Werte

floor() (Untergrenze)

Katalog >

floor(Ausdr) \Rightarrow Ganzzahl

$$\text{floor}(-2.14)$$

-3.

Gibt die größte ganze Zahl zurück, die \leq dem Argument ist. Diese Funktion ist identisch mit **int()**.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

floor(Liste I) \Rightarrow Liste

floor(Matrix I) \Rightarrow Matrix

Für jedes Element einer Liste oder Matrix wird die größte ganze Zahl, die kleiner oder gleich dem Element ist, zurückgegeben.

$$\text{floor}\left(\begin{Bmatrix} \frac{3}{2}, 0, -5.3 \end{Bmatrix}\right) = \{1, 0, -6\}$$

$$\text{floor}\left(\begin{bmatrix} 1.2 & 3.4 \\ 2.5 & 4.8 \end{bmatrix}\right) = \begin{bmatrix} 1, & 3 \\ 2, & 4. \end{bmatrix}$$

Hinweis: Siehe auch **ceiling()** und **int()**.

fMax() (Funktionsmaximum)

Katalog >

fMax(Ausdr, Var) \Rightarrow Boolescher Ausdruck

$$\text{fMax}(1-(x-a)^2-(x-b)^2, x) \quad x = \frac{a+b}{2}$$

fMax(Ausdr, Var, UntereGrenze)

$$\text{fMax}(0.5 \cdot x^3 - x - 2, x) \quad x = \infty$$

fMax(Ausdr, Var, UntereGrenze, ObereGrenze)

fMax(Ausdr, Var) | UntereGrenze \leq Var \leq ObereGrenze

Gibt einen Booleschen Ausdruck zurück, der mögliche Werte von *Var* angibt, welche *Ausdr* maximieren oder seine kleinste obere Grenze angeben.

Sie können den womit-Operator („|“) zur Beschränkung des Lösungsintervalls und/oder zur Angabe anderer Einschränkungen verwenden.

$$\text{fMax}(0.5 \cdot x^3 - x - 2, x) | x \leq 1 \quad x = -0.816497$$

Ist der Modus **Auto oder Näherung** auf Approximiert eingestellt, sucht **fMax()** iterativ nach einem annähernden lokalen

Maximum. Dies ist oft schneller, insbesondere, wenn Sie den Operator “|” benutzen, um die Suche auf ein relativ kleines Intervall zu beschränken, das genau ein lokales Maximum enthält.

Hinweis: Siehe auch **fMin()** und **max()**.

fMin() (Funktionsminimum)

fMin(Ausdr, Var)⇒Boolescher Ausdruck

fMin(Ausdr, Var, UntereGrenze)

fMin(Ausdr, Var, UntereGrenze, ObereGrenze)

**fMin(Ausdr, Var) | UntereGrenze≤Var
≤ObereGrenze**

Gibt einen Booleschen Ausdruck zurück, der mögliche Werte von *Var* angibt, welche *Ausdr* minimieren oder seine kleinste untere Grenze angeben.

Sie können den womit-Operator („|“) zur Beschränkung des Lösungsintervalls und/oder zur Angabe anderer Einschränkungen verwenden.

Ist der Modus **Auto oder Näherung** auf Approximiert eingestellt, sucht **fMin()** iterativ nach einem annähernden lokalen Minimum. Dies ist oft schneller, insbesondere, wenn Sie den Operator “|” benutzen, um die Suche auf ein relativ kleines Intervall zu beschränken, das genau ein lokales Minimum enthält.

Hinweis: Siehe auch **fMax()** und **min()**.

$$\text{fMin}\left(1-(x-a)^2-(x-b)^2, x\right)$$

$x = -\infty \text{ or } x = \infty$

$$\text{fMin}\left(0.5 \cdot x^3 - x - 2, x\right) | x \geq 1$$

$x = 1$.

For

Katalog >

For *Var*, *Von*, *Bis* [, *Schritt*]
Block
EndFor

Führt die in *Block* befindlichen Anweisungen für jeden Wert von *Var* zwischen *Von* und *Bis* aus, wobei der Wert bei jedem Durchlauf um *Schritt* inkrementiert wird.

Var darf keine Systemvariable sein.

Schritt kann positiv oder negativ sein. Der Standardwert ist 1.

Block kann eine einzelne Anweisung oder eine Serie von Anweisungen sein, die durch ":" getrennt sind.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

format() (Format)

Katalog > 

format(*Ausdr*[, *FormatString*]) \Rightarrow *String*

Gibt *Ausdr* als Zeichenkette im Format der Formatvorlage zurück.

Ausdr muss zu einer Zahl vereinfachbar sein.

FormatString ist eine Zeichenkette und muss diese Form besitzen: "F[n]", "S[n]", "E[n]", "G[n][c]", wobei [] optionale Teile bedeutet.

F[n]: Festes Format. n ist die Anzahl der angezeigten Nachkommastellen (nach dem Dezimalpunkt).

S[n]: Wissenschaftliches Format. n ist die Anzahl der angezeigten Nachkommastellen (nach dem Dezimalpunkt).

E[n]: Technisches Format. n ist die Anzahl der Stellen, die auf die erste signifikante

```

Define g()=Func
    Local tempsum,step,i
    0→tempsum
    1→step
    For i,1,100,step
        tempsum+i→tempsum
    EndFor
EndFunc

```

5050

format() (Format)

Katalog >

Ziffer folgen. Der Exponent wird auf ein Vielfaches von 3 gesetzt, und der Dezimalpunkt wird um null, eine oder zwei Stellen nach rechts verschoben.

G[n][c]: Wie Festes Format, unterteilt jedoch auch die Stellen links des Dezimaltrennzeichens in Dreiergruppen. c ist das Gruppentrennzeichen und ist auf "Komma" voreingestellt. Wenn c auf "Punkt" gesetzt wird, wird das Dezimaltrennzeichen zum Komma.

[Rc]: Jeder der vorstehenden Formateinstellungen kann als Suffix das Flag Rc nachgestellt werden, wobei c ein einzelnes Zeichen ist, das den Dezimalpunkt ersetzt.

fPart() (Funktionsteil)

Katalog >

fPart(AusdrI)⇒Ausdruck

fPart(-1.234) -0.234

fPart(Liste I)⇒Liste

fPart({1, -2, 3, 7.003}) {0, -0.3, 0.003}

fPart(Matrix I)⇒Matrix

Gibt den Bruchanteil des Arguments zurück.

Bei einer Liste bzw. Matrix werden die Bruchanteile aller Elemente zurückgegeben.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

FPdf()

Katalog >

FPdf(XWert,FreiGradZähler,FreiGradNenner)
⇒Zahl, wenn XWert eine Zahl ist, Liste, wenn XWert eine Liste ist

FPdf(XWert,FreiGradZähler,FreiGradNenner)
⇒Zahl, wenn XWert eine Zahl ist, Liste, wenn XWert eine Liste ist

Berechnet die F Verteilungswahrscheinlichkeit bei XWert für die angegebenen FreiGradZähler (Freiheitsgrade) und FreiGradNenner.

freqTable►list()

Katalog > 

freqTable►list(Liste1,HäufGanzzahlListe)
⇒Liste

Gibt eine Liste zurück, die die Elemente von *Liste1* erweitert gemäß den Häufigkeiten in *HäufGanzzahlListe* enthält. Diese Funktion kann zum Erstellen einer Häufigkeitstabelle für die Applikation 'Data & Statistics' verwendet werden.

freqTable►list({1,2,3,4},{1,4,3,1})
{1,2,2,2,2,3,3,3,4}

freqTable►list({1,2,3,4},{1,4,0,1})
{1,2,2,2,2,4}

Liste1 kann eine beliebige gültige Liste sein.

HäufGanzzahlListe muss die gleiche Dimension wie *Liste1* haben und darf nur nicht-negative Ganzzahlelemente enthalten. Jedes Element gibt an, wie oft das entsprechende *Liste1*-Element in der Ergebnisliste wiederholt wird. Der Wert 0 schließt das entsprechende *Liste1*-Element aus.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **freqTable@>list(...)** eintippen

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

frequency() (Häufigkeit)

Katalog > 

frequency(Liste1,binsListe)⇒Liste

Gibt eine Liste zurück, die die Zähler der Elemente in *Liste1* enthält. Die Zähler basieren auf Bereichen (*bins*), die Sie in *binsListe* definieren.

datalist:{1,2,e,3,π,4,5,6,"hello",7}
{1,2,2.71828,3,3.14159,4,5,6,"hello",7}

frequency(datalist,{2.5,4.5}) {2,4,3}

Wenn *binsListe* {b(1), b(2), ..., b(n)} ist, sind die festgelegten Bereiche {?≤b(1), b(1)<?≤b(2),...,b(n-1)<?≤b(n), b(n)>?}. Die Ergebnisliste enthält ein Element mehr als die *binsListe*.

Jedes Element des Ergebnisses entspricht der Anzahl der Elemente aus *Liste1*, die im Bereich dieser bins liegen. Ausgedrückt in Form der **countif()** Funktion ist das Ergebnis { countif(Liste, ?≤b(1)), countif(Liste, b(1)<?

Erklärung des Ergebnisses:

2 Elemente aus *Datenliste (Datalist)* sind ≤2.5

4 Elemente aus *Datenliste* sind >2.5 und ≤4.5

3 Elemente aus *Datenliste* sind >4.5

Das Element "Hallo" ist eine Zeichenfolge und kann nicht in einem der definierten bins

frequency() (Häufigkeit)

Katalog > 

$\leq b(2)), \dots, \text{countIf}(\text{Liste}, b(n-1) < ? \leq b(n)),$ platziert werden.
 $\text{countIf}(\text{Liste}, b(n) > ?)\}$.

Elemente von *Liste1*, die nicht "in einem bin platziert" werden können, werden ignoriert. Leere (ungültige) Elemente werden ebenfalls ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

Innerhalb der Lists & Spreadsheet Applikation können Sie für beide Argumente Zellenbereiche verwenden.

Hinweis: Siehe auch **countIf()**, Seite 41.

FTest_2Samp (Zwei-Stichproben F-Test)

Katalog > 

FTest_2Samp *Liste1, Liste2[, Häufigkeit1 [, Häufigkeit2[, Hypoth]]]*

FTest_2Samp *Liste1, Liste2[, Häufigkeit1 [, Häufigkeit2[, Hypoth]]]*

(Datenlisteneingabe)

FTest_2Samp *sx1,n1,sx2,n2[, Hypoth]*

FTest_2Samp *sx1,n1,sx2,n2[, Hypoth]*

(Zusammenfassende statistische Eingabe)

Führt einen F -Test mit zwei Stichproben durch. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Für $H_a : \sigma_1 > \sigma_2$ setzen Sie *Hypoth*>0

Für $H_a : \sigma_1 \neq \sigma_2$ (Standard) setzen Sie *Hypoth* =0

Für $H_a : \sigma_1 < \sigma_2$ setzen Sie *Hypoth*<0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabevariable	Beschreibung
Statistik.F	Berechnete F Statistik für die Datenfolge
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann

Ausgabeveriable	Beschreibung
stat.dfNumer	Freiheitsgrade des Zählers = n1-1
stat.dfDenom	Freiheitsgrade des Nenners = n2-1
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.x1_bar	Stichprobenmittelwerte der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.x2_bar	
stat.n1, stat.n2	Stichprobenumfang

Func

Katalog >

Func

Block

EndFunc

Vorlage zur Erstellung einer benutzerdefinierten Funktion.

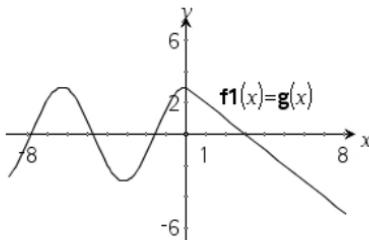
Block kann eine einzelne Anweisung, eine Reihe von durch das Zeichen „;“ voneinander getrennten Anweisungen oder eine Reihe von Anweisungen in separaten Zeilen sein. Die Funktion kann die Anweisung **Zurückgeben (Return)** verwenden, um ein bestimmtes Ergebnis zurückzugeben.

Hinweis zum Eingeben des Beispieles:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Definieren Sie eine stückweise definierte Funktion:

```
Define g(x)=Func
If x<0 Then
  Return 3·cos(x)
Else
  Return 3-x
EndIf
EndFunc
```

Ergebnis der graphischen Darstellung g(x)



G

gcd() (Größter gemeinsamer Teiler)

Katalog >

gcd(Zahl1, Zahl2)⇒Ausdruck

gcd(18,33) 3

Gibt den größten gemeinsamen Teiler der beiden Argumente zurück. Der **gcd** zweier Brüche ist der **gcd** ihrer Zähler dividiert durch das kleinste gemeinsame Vielfache (**lcm**) ihrer Nenner.

gcd() (Größter gemeinsamer Teiler)**Katalog >**

In den Modi Auto oder Approximiert ist der gcd von Fließkommabrüchen 1,0.

gcd(Liste1, Liste2)⇒Liste

$$\text{gcd}(\{12, 14, 16\}, \{9, 7, 5\})$$

{3, 7, 1}

Gibt die größten gemeinsamen Teiler der einander entsprechenden Elemente von *Liste1* und *Liste2* zurück.

gcd(Matrix1, Matrix2)⇒Matrix

$$\text{gcd}\left(\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}, \begin{bmatrix} 4 & 8 \\ 12 & 16 \end{bmatrix}\right)$$

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

Gibt die größten gemeinsamen Teiler der einander entsprechenden Elemente von *Matrix1* und *Matrix2* zurück.

geomCdf()**Katalog >**

geomCdf(*p*,*untereGrenze*,*obereGrenze*)⇒Zahl,
wenn *untereGrenze* und *obereGrenze* Zahlen sind,
Liste, wenn *untereGrenze* und *obereGrenze* Listen
sind

geomCdf(*p*,*obereGrenze*) für $P(1 \leq X \leq \text{obereGrenze})$
⇒Zahl, wenn *obereGrenze* eine Zahl ist, *Liste*,
wenn *obereGrenze* eine Liste ist

Berechnet die kumulative geometrische Wahrscheinlichkeit von *UntereGrenze* bis *ObereGrenze* mit der angegebenen Erfolgswahrscheinlichkeit *p*.

Für $P(X \leq \text{obereGrenze})$ setzen Sie *untereGrenze* = 1.

geomPdf()**Katalog >**

geomPdf(*p*,*XWert*)⇒Zahl, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeit an einem *XWert*, die Anzahl der Einzelversuche, bis der erste Erfolg eingetreten ist, für die diskrete geometrische Verteilung mit der vorgegebenen Erfolgswahrscheinlichkeit *p*.

Get**Katalog >** **Get [EingabeString,] *var*[, *statusVar*]**

Beispiel: Dies liest den Status einer grünen LED.

Get [*EingabeString*,] *Fkt(arg1, ...argn)*
[, *statusVar*]

Programmierbefehl: Fordert einen Wert von einem Gerät an, das an den USB-Anschluss angeschlossen ist, und weist den Wert der Variablen *Var* zu.

Das angeschlossene USB-Gerät muss für die Kommunikation mit TI-Nspire™ Handhelds oder der Software konfiguriert sein.

Implizite Vereinfachung findet statt. Zum Beispiel wird eine empfangene Zeichenfolge „123“ als numerischer Wert interpretiert. Um die Zeichenfolge beizubehalten, verwenden Sie **GetStr** statt **Get**.

Mit dem optionalen Argument *EingabeString* können Sie eine Buchstabenzeichenfolge oder eine Stringvariable an das angeschlossene Gerät senden und die Daten mithilfe eines einzelnen Befehls erhalten.

Wenn Sie das optionale Argument *statusVar* einbeziehen, wird ihm ein Wert auf Basis des Erfolgs der Operation zugewiesen. Ein Wert von null bedeutet, dass keine Daten empfangen wurden.

In der zweiten Synthax ermöglicht das Argument *Fkt()* es einem Programm, die empfangene Zeichenfolge als Funktionsdefinition zu speichern. Diese Syntax verhält sich so, als hätte das Programm den folgenden Befehl ausgeführt:

Definiere *Fkt(arg1, ...argn) = empfanger String*

Anschließend kann das Programm die so definierte Funktion *Fkt()* nutzen.

Hinweis: Sie können den Befehl **Get** in benutzerdefinierten Programmen, aber nicht in Funktionen verwenden.

Hinweis: Siehe auch **GetStr**, page 89 und **Send**, page 166.

Send "READ GREEN 1"

Get green1

Äquivalenter einzelner Befehl:

Get "READ GREEN 1", green1

getDenom() (Nenner holen)

Katalog >

getDenom(AusdrI)⇒Ausdruck

Transformiert das Argument in einen Ausdruck mit gekürztem gemeinsamem Nenner und gibt dann den Nenner zurück.

getDenom $\left(\frac{x+2}{y-3} \right)$	$y-3$
getDenom $\left(\frac{2}{7} \right)$	7
getDenom $\left(\frac{1 + \frac{y^2+y}{x}}{y^2} \right)$	$x \cdot y$

getLangInfo()

Katalog >

getLangInfo()⇒Zeichenkette

getLangInfo()	"en"
---------------	------

Gibt eine Zeichenkette zurück, die der Abkürzung der gegenwärtig aktiven Sprache entspricht. Sie können den Befehl zum Beispiel in einem Programm oder einer Funktion zum Bestimmen der aktuellen Sprache verwenden.

Englisch = "en"

Dänisch = "da"

Deutsch = "de"

Finnisch = "fi"

Französisch = "fr"

Italienisch = "it"

Holländisch = "nl"

Holländisch (Belgien) = "nl_BE"

Norwegisch = "no"

Portugiesisch = "pt"

Spanisch = "es"

Schwedisch = "sv"

getLockInfo()

Katalog >

getLockInfo(Var)⇒Wert

Gibt den aktuellen Gesperrt/Entsperrt-Status der Variablen *Var* aus.

Wert =0: *Var* ist nicht gesperrt oder ist nicht vorhanden.

Wert =1: *Var* ist gesperrt und kann nicht geändert oder gelöscht werden.

Siehe **Lock**, Seite 112, und **unlock**, Seite 209.

a:=65	65
Lock a	Done
getLockInfo(a)	1
a:=75	"Error: Variable is locked."
DelVar a	"Error: Variable is locked."
Unlock a	Done
a:=75	75
DelVar a	Done

getMode()

Katalog >

getMode(ModusNameGanzzahl)⇒Wert

getMode(0)⇒Liste

getMode(ModusNameGanzzahl) gibt einen Wert zurück, der die aktuelle Einstellung des Modus *ModusNameGanzzahl* darstellt.

getMode(0) gibt eine Liste mit Zahlenpaaren zurück. Jedes Paar enthält eine Modus-Ganzzahl und eine Einstellungs-Ganzzahl.

Eine Auflistung der Modi und ihrer Einstellungen finden Sie in der nachstehenden Tabelle.

Wenn Sie die Einstellungen mit **getMode(0)** → *var* speichern, können Sie **setMode(var)** in einer Funktion oder in einem Programm verwenden, um die Einstellungen nur innerhalb der Ausführung dieser Funktion bzw. dieses Programms vorübergehend wiederherzustellen. Siehe **setMode()**, Seite 170.

getMode(0)	{1,7,2,1,3,1,4,1,5,1,6,1,7,1,8,1}
getMode(1)	7
getMode(8)	1

Modus Name	Modus Ganzzahl	Einstellen von Ganzzahlen
Angezeigte Ziffern	1	1=Fließ, 2=Fließ 1, 3=Fließ 2, 4=Fließ 3, 5=Fließ 4, 6=Fließ 5, 7=Fließ 6, 8=Fließ 7, 9=Fließ 8, 10=Fließ 9, 11=Fließ 10, 12=Fließ 11, 13=Fließ 12, 14=Fix 0, 15=Fix 1, 16=Fix 2, 17=Fix 3, 18=Fix 4, 19=Fix 5, 20=Fix 6, 21=Fix 7, 22=Fix 8, 23=Fix 9, 24=Fix 10, 25=Fix 11, 26=Fix 12

Modus Name	Modus Ganzzahl	Einstellen von Ganzzahlen
Winkel	2	1 =Bogenmaß, 2 =Grad, 3 =Neugrad
Exponentialformat	3	1 =Normal, 2 =Wissenschaftlich, 3 =Technisch
Reell oder komplex	4	1 =Reell, 2 =Kartesisch, 3 =Polar
Auto oder Approx.	5	1 =Auto, 2 =Approximiert, 3 =Exakt
Vektorformat	6	1 =Kartesisch, 2 =Zylindrisch, 3 =Sphärisch
Basis	7	1 =Dezimal, 2 =Hex, 3 =Binär
Einheitensystem	8	1 =SI, 2 =Eng/US

getNum() (Zähler holen)

Katalog >

getNum(Ausdr1)⇒Ausdruck

Transformiert das Argument in einen Ausdruck mit gekürztem gemeinsamem Nenner und gibt dann den Zähler zurück.

$$\begin{aligned} \text{getNum}\left(\frac{x+2}{y-3}\right) && x+2 \\ \text{getNum}\left(\frac{2}{7}\right) && 2 \\ \text{getNum}\left(\frac{1}{x} + \frac{1}{y}\right) && x+y \end{aligned}$$

GetStr

Katalog >

GetStr [EingabeString,] Var[, statusVar]

Zum Beispiel siehe **Get**.

**GetStr [EingabeString,] Fkt(arg1, ...argn
[, statusVar]**

Programmierbefehl: Verhält sich genauso wie der Befehl **Get**, der empfangene Wert wird aber immer als String interpretiert. Der Befehl **Get** interpretiert die Antwort hingegen als Ausdruck, es sei denn, sie ist in Anführungszeichen ("") gesetzt.

Hinweis: Siehe auch **Get**, page 85 und **Send**, page 166.

getType()

Katalog >

getType(var)⇒*String*

Gibt eine Zeichenkette zurück, die den Datentyp einer Variablen *var* anzeigt.

Wenn *var* nicht definiert ist, wird die Zeichenkette „NONE“ zurückgegeben.

{1,2,3}→temp	{1,2,3}
getType(temp)	"LIST"
3·i→temp	3·i
getType(temp)	"EXPR"
DelVar temp	Done
getType(temp)	"NONE"

getVarInfo()

Katalog >

getVarInfo()⇒*Matrix* oder *String*

getVarInfo(BiblioNameString)⇒*Matrix* oder *String*

getVarInfo() gibt eine Informationsmatrix (Name, Typ, Erreichbarkeit einer Variablen in der Bibliothek und Gesperrt/Entsperrt-Status) für alle Variablen und Bibliotheksobjekte zurück, die im aktuellen Problem definiert sind.

Wenn keine Variablen definiert sind, gibt **getVarInfo()** die Zeichenfolge "KEINE" (NONE) zurück.

getVarInfo(BiblioNameString) gibt eine Matrix zurück, die Informationen zu allen Bibliotheksobjekten enthält, die in der Bibliothek *BiblioNameString* definiert sind. *BiblioNameString* muss eine Zeichenfolge (in Anführungszeichen eingeschlossener Text) oder eine Zeichenfolgenvariable sein.

Wenn die Bibliothek *BiblioNameString* nicht existiert, wird ein Fehler angezeigt.

getVarInfo()	"NONE"
Define x=5	Done
Lock x	Done
Define LibPriv.y={1,2,3}	Done
Define LibPub z(x)=3·x ² -x	Done
getVarInfo()	$\begin{bmatrix} x & \text{"NUM"} & \{\} & 1 \\ y & \text{"LIST"} & \text{"LibPriv"} & 0 \\ z & \text{"FUNC"} & \text{"LibPub"} & 0 \end{bmatrix}$
getVarInfo(tmp3)	"Error: Argument must be a string"
getVarInfo("tmp3")	[volyl2 "NONE" "LibPub" 0]

getVarInfo()

Katalog >

Beachten Sie das Beispiel links, in dem das Ergebnis von **getVarInfo()** der Variablen *vs* zugewiesen wird. Beim Versuch, Zeile 2 oder Zeile 3 von *vs* anzuzeigen, wird der Fehler *„Liste oder Matrix ungültig“* zurückgegeben, weil mindestens eines der Elemente in diesen Zeilen (Variable *b* zum Beispiel) eine Matrix ergibt.

Dieser Fehler kann auch auftreten, wenn *Ans* zum Neuberechnen eines **getVarInfo()**-Ergebnisses verwendet wird.

Das System liefert den obigen Fehler, weil die aktuelle Version der Software keine verallgemeinerte Matrixstruktur unterstützt, bei der ein Element einer Matrix eine Matrix oder Liste sein kann.

<i>a:=1</i>	1
<i>b:=[1 2]</i>	[1 2]
<i>c:=[1 3 7]</i>	[1 3 7]
<i>vs:=getVarInfo()</i>	$\begin{bmatrix} a & \text{"NUM"} & "[]" & 0 \\ b & \text{"MAT"} & "[]" & 0 \\ c & \text{"MAT"} & "[]" & 0 \end{bmatrix}$
<i>vs[1]</i>	[1 "NUM" "[]" 0]
<i>vs[1,1]</i>	1
<i>vs[2]</i>	"Error: Invalid list or matrix"
<i>vs[2,1]</i>	[1 2]

Goto (Gehe zu)

Katalog >

Goto MarkeName

Setzt die Programmausführung bei der Marke *MarkeName* fort.

MarkeName muss im selben Programm mit der Anweisung **Lbl** definiert worden sein.

Hinweis zum Eingeben des Beispieles:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define *g()*=Func

Done

Local *temp,i*
0 → *temp*
1 → *i*
Lbl *top*
temp+i → *temp*
If *i*<10 Then
i+1 → *i*
Goto *top*
EndIf
Return *temp*
EndFunc

g()

55

►Grad (Neugrad)

Katalog >

Ausdr1 ►Grad⇒*Ausdruck*

Im Grad-Modus:

Wandelt *Ausdr1* ins Winkelmaß Neugrad um.

(1.5)►Grad

(1.66667)^g

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Grad eintippen.

Im Bogenmaß-Modus:

(1.5)►Grad

(95.493)^g

identity() (Einheitsmatrix)**Katalog >** **identity(Ganzzahl)⇒Matrix**Gibt die Einheitsmatrix mit der Dimension *Ganzzahl* zurück.*Ganzzahl* muss eine positive ganze Zahl sein.

identity(4)

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

If**Katalog >** **If Boolescher Ausdr Anweisung**

```
Define g(x)=Func
      If x<0 Then
          Return x2
      EndIf
  EndFunc
```

*Done***If Boolescher Ausdr Then Block EndIf**Wenn *Boolescher Ausdr* wahr ergibt, wird die Einzelanweisung *Anweisung* oder der Anweisungsblock *Block* ausgeführt und danach mit *EndIf* fortgefahrene.Wenn *Boolescher Ausdr* falsch ergibt, wird das Programm fortgesetzt, ohne dass die Einzelanweisung bzw. der Anweisungsblock ausgeführt werden.*Block* kann eine einzelne Anweisung oder eine Serie von Anweisungen sein, die durch ":" getrennt sind.**Hinweis zum Eingeben des Beispiels:**

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

g(-2)

4

If**If Boolescher Ausdr Then***Block1***Else***Block2***EndIf**

Wenn Boolescher Ausdr wahr ergibt, wird *Block1* ausgeführt und dann *Block2* übersprungen.

Wenn Boolescher Ausdr falsch ergibt, wird *Block1* übersprungen, aber *Block2* ausgeführt.

Block1 und *Block2* können einzelne Anweisungen sein.

If Boolescher Ausdr1 Then*Block1***Elseif Boolescher Ausdr2 Then***Block2*

:

Elseif Boolescher AusdrN Then*BlockN***EndIf**

Gestattet Programmverzweigungen. Wenn Boolescher Ausdr1 wahr ergibt, wird Block1 ausgeführt. Wenn Boolescher Ausdr1 falsch ergibt, wird Boolescher Ausdr2 ausgewertet usw.

Define $g(x) = \text{Func}$

Done

```
If x < 0 Then
    Return -x
Else
    Return x
EndIf
EndFunc
```

 $g(12)$

12

 $g(-12)$

12

Define $g(x) = \text{Func}$ If $x < -5$ Then

Return 5

Elseif $x > -5$ and $x < 0$ Then

Return -x

Elseif $x \geq 0$ and $x \neq 10$ Then

Return x

Elseif $x = 10$ Then

Return 3

EndIf

EndFunc

Done

 $g(-4)$

4

 $g(10)$

3

ifFn()

ifFn(BoolescherAusdruck, Wert_wenn_wahr [, Wert_wenn_falsch [, Wert_wenn_unbekannt]]) \Rightarrow Ausdruck, Liste oder Matrix

Wertet den Booleschen Ausdruck BoolescherAusdruck (oder jedes einzelne Element von BoolescherAusdruck) aus und erstellt ein Ergebnis auf der Grundlage folgender Regeln:

- BoolescherAusdruck kann einen Einzelwert, eine Liste oder eine Matrix testen.

ifFn($\{1, 2, 3\} < 2.5, \{5, 6, 7\}, \{8, 9, 10\}$)

{5, 6, 10}

Testwert von **1** ist kleiner als 2.5, somit wird das entsprechende

Wert_wenn_wahr-Element von **5** in die Ergebnisliste kopiert.

Testwert von **2** ist kleiner als 2.5, somit wird das entsprechende

Wert_wenn_wahr-Element von **6** in die

- Wenn ein Element von *BoolescherAusdruck* als wahr bewertet wird, wird das entsprechende Element aus *Wert_wenn_wahr* zurückgegeben.
- Wenn ein Element von *BoolescherAusdruck* als falsch bewertet wird, wird das entsprechende Element aus *Wert_wenn_falsch* zurückgegeben. Wenn Sie *Wert_wenn_falsch* weglassen, wird *Undef* zurückgegeben.
- Wenn ein Element von *BoolescherAusdruck* weder wahr noch falsch ist, wird das entsprechende Element aus *Wert_wenn_unbekannt* zurückgegeben. Wenn Sie *Wert_wenn_unbekannt* weglassen, wird *Undef* zurückgegeben.
- Wenn das zweite, dritte oder vierte Argument der Funktion **ifFn()** ein einzelnen Ausdruck ist, wird der Boolesche Test für jede Position in *BoolescherAusdruck* durchgeführt.

Hinweis: Wenn die vereinfachte Anweisung *BoolescherAusdruck* eine Liste oder Matrix einbezieht, müssen alle anderen Listen- oder Matrixanweisungen dieselben(n) Dimension(en) haben, und auch das Ergebnis wird dieselben(n) Dimension(en) haben.

Ergebnisliste kopiert.

Testwert von **3** ist nicht kleiner als **2.5**, somit wird das entsprechende *Wert_wenn_falsch*-Element von **10** in die Ergebnisliste kopiert.

ifFn({{1,2,3}<2.5,4},{8,9,10}) {4,4,10}

Wert_wenn_wahr ist ein einzelner Wert und entspricht einer beliebigen ausgewählten Position.

ifFn({{1,2,3}<2.5,{5,6,7}}) {5,6,undef}

Wert_wenn_falsch ist nicht spezifiziert. *Undef* wird verwendet.

ifFn({{2,"a"}<2.5,{6,7},{9,10}),"err") {6,"err"}

Ein aus *Wert_wenn_wahr* ausgewähltes Element. Ein aus *Wert_wenn_unbekannt* ausgewähltes Element.

imag() (Imaginärteil)

imag(Ausdr1) \Rightarrow Ausdruck

Gibt den Imaginärteil des Arguments zurück.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt. Siehe auch **real()**, Seite 153

imag(Liste1) \Rightarrow Liste

Gibt eine Liste der Imaginärteile der Elemente zurück.

imag(1+2·i)

2

imag(z)

0

imag(x+i·y)

y

imag({{-3,4-i,i}})

{0,-1,1}

imag() (Imaginärteil)**Katalog >** **imag(Matrix I)⇒Matrix**

Gibt eine Matrix der Imaginärteile der Elemente zurück.

$$\text{imag} \begin{bmatrix} a & b \\ i \cdot c & i \cdot d \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 \\ c & d \end{bmatrix}$$

impDif() (Implizite Ableitung)**Katalog >** **impDif(Gleichung, Var, abhängigeVar [,Ord])⇒Ausdruck**

$$\text{impDif}(x^2+y^2=100, x, y)$$

$$\frac{x}{y}$$

wobei der Vorgabewert für die Ordnung *Ord* 1 ist.

Berechnet die implizite Ableitung für Gleichungen, in denen eine Variable implizit durch eine andere definiert ist.

Umleitung**Siehe #(,), Seite 241.****inString() (In String)****Katalog >** **inString(Quellstring, Teilstring[, Start])**
⇒Ganzzahl

$$\text{inString}("Hello there", "the")$$

$$7$$

$$\text{inString}("ABCEFG", "D")$$

$$0$$

Gibt die Position des Zeichens von *Quellstring* zurück, an der das erste Vorkommen von *Teilstring* beginnt.Start legt fest (sofern angegeben), an welcher Zeichenposition innerhalb von *Quellstring* die Suche beginnt. Vorgabe = 1 (das erste Zeichen von *Quellstring*).Enthält *Quellstring* die Zeichenkette *Teilstring* nicht oder ist *Start* > Länge von *Quellstring*, wird Null zurückgegeben.**int() (Ganze Zahl)****Katalog >** **int(Ausdr)⇒Ganzzahl**

$$\text{int}(-2.5)$$

$$-3$$

int(Liste l)⇒Liste

$$\text{int}([-1.234 \ 0 \ 0.37])$$

$$[-2. \ 0 \ 0.]$$

int(Matrix I)⇒Matrix

int() (Ganze Zahl)

Katalog >

Gibt die größte ganze Zahl zurück, die kleiner oder gleich dem Argument ist. Diese Funktion ist identisch mit **floor()**.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

Für eine Liste oder Matrix wird für jedes Element die größte ganze Zahl zurückgegeben, die kleiner oder gleich dem Element ist.

intDiv() (Ganzzahl teilen)

Katalog >

intDiv(Zahl1, Zahl2)⇒Ganzzahl

intDiv(-7,2) -3

intDiv(Liste1, Liste2)⇒Liste

intDiv(4,5) 0

intDiv(Matrix1, Matrix2)⇒Matrix

intDiv({12,-14,-16},{5,4,-3}) {2,-3,5}

Gibt den mit Vorzeichen versehenen ganzzahligen Teil von $(Zahl1 \div Zahl2)$ zurück.

Für eine Liste oder Matrix wird für jedes Elementpaar der mit Vorzeichen versehene ganzzahlige Teil von $(\text{Argument 1} \div \text{Argument 2})$ zurückgegeben.

integral

Siehe **J()**, Seite 223.

interpolate()

Katalog >

interpolate(xWert, xListe, yListe, yStrListe)⇒Liste

Differentialgleichung:

Diese Funktion tut folgendes:

$y' = -3 \cdot y + 6 \cdot t + 5$ und $y(0) = 5$

Bei gegebenen $xListe$, $yListe=f(xListe)$ und $yStrListe=f'(xListe)$ für eine unbekannte Funktion f wird eine kubische Interpolierende zur Approximierung der Funktion f bei $xWert$ verwendet. Es wird angenommen, dass $xListe$ eine Liste

$rK := rk23(-3 \cdot y + 6 \cdot t + 5, t, y, \{0, 10\}, 5, 1)$

0.	1.	2.	3.	4.
5.	3.19499	5.00394	6.99957	9.00593
10				

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangleleft und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

interpolate()

Katalog >

monoton steigender oder fallender Zahlen ist; jedoch kann diese Funktion auch einen Wert zurückgeben, wenn dies nicht der Fall ist. Diese Funktion geht *xListe* durch und sucht nach einem Intervall $[xListe[i], xListe[i+1]]$, das *xWert* enthält. Wenn sie ein solches Intervall findet, gibt sie einen interpolierten Wert für $f(xWert)$ zurück; anderenfalls gibt sie **undef** zurück.

xListe, *yListe* und *yStrListe* müssen die gleiche Dimension ≥ 2 besitzen und Ausdrücke enthalten, die zu Zahlen vereinfachbar sind.

xWert kann eine nicht definierte Variable, eine Zahl oder eine Zahlenliste sein.

Verwenden Sie die Funktion `interpolate()`, um die Funktionswerte für die Liste *xWert* zu berechnen:

```
xvaluelist:=seq(i,i,0,10,0.5)
{0,0.5,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,5.5,6.,6.5,}
xlist:=matlist(rk[1])
{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.}
ylist:=matlist(rk[2])
{5.,3.19499,5.00394,6.99957,9.00593,10.997,}
yprimelist:=-3*y+6*t+5|y=ylist and t=xlist
{-10.,-1.41503,1.98819,2.00129,1.98221,2.006,}
interpolate(xvaluelist,xlist,ylist,yprimelist)
{5.,2.67062,3.19499,4.02782,5.00394,6.0001,}
```

invχ²()

Katalog >

invχ²(Fläche,FreiGrad)

invChi2(Fläche,FreiGrad)

Berechnet die inverse kumulative χ^2 (Chi-Quadrat) Wahrscheinlichkeitsfunktion, die durch Freiheitsgrade *FreiGrad* für eine bestimmte *Fläche* unter der Kurve festgelegt ist.

invF()

Katalog >

invF(Fläche,FreiGradZähler,FreiGradNenner)

invF(Fläche,FreiGradZähler,FreiGradNenner)

Berechnet die inverse kumulative F Verteilungsfunktion, die durch *FreiGradZähler* und *FreiGradNenner* für eine bestimmte *Fläche* unter der Kurve festgelegt ist.

invNorm()

Katalog >

invNorm(Fläche[,μ,σ])

Berechnet die inverse kumulative Normalverteilungsfunktion für eine bestimmte *Fläche* unter der Normalverteilungskurve, die durch

invNorm()

Katalog >

μ und σ festgelegt ist.

invt()

Katalog >

invt(Fläche,FreiGrad)

Berechnet die inverse kumulative Student-t-Wahrscheinlichkeitsfunktion, die durch Freiheitsgrade, *FreiGrad*, für eine bestimmte Fläche unter der Kurve festgelegt ist.

iPart() (Ganzzahliger Teil)

Katalog >

iPart(Zahl)⇒Ganzzahl

iPart(-1.234) -1.

iPart(Liste I)⇒Liste

iPart($\left\{ \frac{3}{2}, -2.3, 7.003 \right\}$) {1, 2, 7.}

iPart(Matrix I)⇒Matrix

Gibt den ganzzahligen Teil des Arguments zurück.

Für eine Liste oder Matrix wird der ganzzahlige Teil jedes Elements zurückgegeben.

Das Argument kann eine reelle oder eine komplexe Zahl sein.

irr()

Katalog >

irr(CF0,CFListe [,CFFreq])⇒Wert

list1:= {6000,-8000,2000,-3000} {6000, -8000, 2000, -3000}

Finanzfunktion, die den internen Zinsfluss einer Investition berechnet.

list2:= {2,2,2,1} {2,2,2,1}

CF0 ist der Anfangs-Cash-Flow zum Zeitpunkt 0; dies muss eine reelle Zahl sein.

irr(5000,list1,list2) -4.64484

CFListe ist eine Liste von Cash-Flow-Beträgen nach dem Anfangs-Cash-Flow *CF0*.

CFFreq ist eine optionale Liste, in der jedes Element die Häufigkeit des Auftretens für einen gruppierten (fortlaufenden) Cash-Flow-Betrag angibt, der das entsprechende Element von *CFListe* ist. Der Standardwert ist 1; wenn Sie Werte eingeben, müssen

diese positive Ganzzahlen < 10.000 sein.

Hinweis: Siehe auch **mirr()**, Seite 121.

isPrime() (Primzahltest)

isPrime(Zahl)⇒Boolescher konstanter Ausdruck

Gibt "wahr" oder "falsch" zurück, um anzuseigen, ob es sich bei *Zahl* um eine ganze Zahl ≥ 2 handelt, die nur durch sich selbst oder 1 ganzzahlig teilbar ist.

Übersteigt *Zahl* ca. 306 Stellen und hat sie keine Faktoren ≤ 1021 , dann zeigt **isPrime** (*Zahl*) eine Fehlermeldung an.

Möchten Sie lediglich feststellen, ob es sich bei *Zahl* um eine Primzahl handelt, verwenden Sie **isPrime()** anstelle von **factor()**. Dieser Vorgang ist wesentlich schneller, insbesondere dann, wenn *Zahl* keine Primzahl ist und ihr zweitgrößter Faktor ca. fünf Stellen übersteigt.

Hinweis zum Eingeben des Beispieles:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

isPrime(5)	true
isPrime(6)	false

Funktion zum Auffinden der nächsten Primzahl nach einer angegebenen Zahl:

```
Define nextprim(n)=Func      Done
                           Loop
                           n+1→n
                           If isPrime(n)
                           Return n
                           EndLoop
                           EndFunc
```

nextprim(7)	11
-------------	----

isVoid()

isVoid(Var)⇒Boolescher konstanter Ausdruck

isVoid(Ausdr)⇒Boolescher konstanter Ausdruck

isVoid(Liste)⇒Liste Boolescher konstanter Ausdrücke

Gibt wahr oder falsch zurück, um anzuseigen, ob das Argument ein ungültiger Datentyp ist.

Weitere Informationen zu ungültigen

a:=_	-
isVoid(a)	true
isVoid({1,_,3})	{ false,true,false }

Elementen finden Sie (Seite 252).

L

Lbl (Marke)**Lbl MarkeName**

Definiert in einer Funktion eine Marke mit dem Namen *MarkeName*.

Mit der Anweisung **Goto MarkeName** können Sie die Ausführung an der Anweisung fortsetzen, die unmittelbar auf die Marke folgt.

Für *MarkeName* gelten die gleichen Benennungsregeln wie für einen Variablenamen.

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define $g() = \text{Func}$

Done

Local temp, i $0 \rightarrow \text{temp}$ $1 \rightarrow i$ Lbl *top* $\text{temp} + i \rightarrow \text{temp}$ If $i < 10$ Then $i + 1 \rightarrow i$ Goto *top*

EndIf

Return *temp*

EndFunc

 $g()$

55

lcm() (Kleinste gemeinsames Vielfaches)**lcm(Zahl1, Zahl2)⇒Ausdruck** $\text{lcm}(6,9)$

18

lcm(Liste1, Liste2)⇒Liste $\text{lcm}\left(\left\{\frac{1}{3}, -14, 16\right\}, \left\{\frac{2}{15}, 7, 5\right\}\right) = \left\{\frac{2}{3}, 14, 80\right\}$ **lcm(Matrix1, Matrix2)⇒Matrix**

Gibt das kleinste gemeinsame Vielfache der beiden Argumente zurück. Das **lcm** zweier Brüche ist das **lcm** ihrer Zähler dividiert durch den größten gemeinsamen Teiler (**gcd**) ihrer Nenner. Das **lcm** von Dezimalbruchzahlen ist ihr Produkt.

Für zwei Listen oder Matrizen wird das kleinste gemeinsame Vielfache der entsprechenden Elemente zurückgegeben.

left() (Links)**left(Quellstring[, Anz])** \Rightarrow String

left("Hello",2)

"He"

Gibt *Anz* Zeichen zurück, die links in der Zeichenkette *Quellstring* enthalten sind.

Wenn Sie *Anz* weglassen, wird der gesamte *Quellstring* zurückgegeben.

left(Liste1[, Anz]) \Rightarrow Liste

left({1,3,-2,4},3)

{1,3,-2}

Gibt *Anz* Elemente zurück, die links in *Liste1* enthalten sind.

Wenn Sie *Anz* weglassen, wird die gesamte *Liste1* zurückgegeben.

left(Vergleich) \Rightarrow Ausdruckleft($x < 3$) x

Gibt die linke Seite einer Gleichung oder Ungleichung zurück.

libShortcut()**libShortcut(BiblioNameString,
VerknNameString)****[, BiblioPrivMerker]** \Rightarrow Liste von Variablen

Erstellt eine Variablengruppe im aktuellen Problem, die Verweise auf alle Objekte im angegebenen Bibliotheksdokument *BiblioNameString* enthält. Fügt außerdem die Gruppenmitglieder dem Variablenmenü hinzu. Sie können dann auf jedes Objekt mit *VerknNameString* verweisen.

Setzen Sie *BiblioPrivMerker=0*, um private Bibliotheksobjekte auszuschließen (Standard)

Setzen Sie *BiblioPrivMerker=1*, um private Bibliotheksobjekte einzubeziehen

Informationen zum Kopieren einer Variablengruppe finden Sie unter **CopyVar** (Seite 35).

Informationen zum Löschen einer Variablengruppe finden Sie unter **DelVar** (Seite 57).

Dieses Beispiel setzt ein richtig gespeichertes und aktualisiertes Bibliotheksdokument namens **linalg2** voraus, das als *clearmat*, *gauss1* und *gauss2* definierte Objekte enthält.

getVarInfo("linalg2")

$$\begin{bmatrix} clearmat & "FUNC" & "LibPub" \\ gauss1 & "PRGM" & "LibPriv" \\ gauss2 & "FUNC" & "LibPub" \end{bmatrix}$$

libShortcut("linalg2", "la")

{la.clearmat, la.gauss2}

libShortcut("linalg2", "la", 1)

{la.clearmat, la.gauss1, la.gauss2}

limit() oder lim() (Limes)

Katalog >

limit(Ausdr1, Var, Stelle [,Richtung])
⇒ Ausdruck

limit(Liste1, Var, Stelle [,Richtung])
⇒ Liste

limit(Matrix1, Var, Stelle [,Richtung])
⇒ Matrix

Gibt den angeforderten Grenzwert zurück.

Hinweis: Siehe auch **Vorlage Limes**, Seite 11.

Richtung: negativ=von links, positiv=von rechts, ansonsten=beide. (Wird keine Angabe gemacht, gilt für **Richtung** die Vorgabe beide.)

Grenzen bei positiv ∞ und negativ ∞ werden stets zu einseitigen Grenzen von der endlichen Seite aus umgewandelt.

Je nach den Umständen gibt **limit()** sich selbst oder undef zurück, wenn kein eindeutiger Grenzwert ermittelt werden kann. Das heißt nicht unbedingt, dass es keinen eindeutigen Grenzwert gibt. undef bedeutet lediglich, dass das Ergebnis entweder eine unbekannte Zahl endlicher oder unendlicher Größenordnung ist, oder es ist die Gesamtmenge dieser Zahlen.

limit() arbeitet mit Verfahren wie der Regel von L'Hospital; es gibt daher eindeutige Grenzwerte, die es nicht ermitteln kann. Wenn *Ausdr1* über *Var* hinaus weitere undefinierte Variablen enthält, müssen Sie möglicherweise Einschränkungen dafür verwenden, um ein brauchbareres Ergebnis zu erhalten.

Grenzwerte können sehr anfällig für Rundungsfehler sein. Vermeiden Sie nach Möglichkeit die Einstellung Approximiert für den Modus **Auto oder Näherung** sowie Näherungszahlen beim Berechnen von Grenzwerten. Andernfalls kann es sein, dass Grenzen, die Null oder unendlich sein

$$\lim_{x \rightarrow 5} (2 \cdot x + 3)$$

13

$$\lim_{x \rightarrow 0^+} \left(\frac{1}{x} \right)$$

∞

$$\lim_{x \rightarrow 0} \left(\frac{\sin(x)}{x} \right)$$

1

$$\lim_{h \rightarrow 0} \left(\frac{\sin(x+h) - \sin(x)}{h} \right)$$

$\cos(x)$

$$\lim_{n \rightarrow \infty} \left(\left(1 + \frac{1}{n} \right)^n \right)$$

e

$$\lim_{x \rightarrow \infty} (a^x)$$

undef

$$\lim_{x \rightarrow \infty} (a^x) | a > 1$$

∞

$$\lim_{x \rightarrow \infty} (a^x) | a > 0 \text{ and } a < 1$$

0

müssten, dies nicht sind und umgekehrt
endliche Grenzwerte ungleich Null nicht
erkannt werden.

LinRegBx

LinRegBx $X, Y[, Häuf][, Kategorie, Mit]$

Berechnet die lineare Regression $y = a + b \cdot x$ auf Listen X und Y mit der Häufigkeit $Häuf$. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden Datenpunkt X und Y an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden X und Y Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.RegEqn	Regressionsgleichung: $a + b \cdot x$
stat.a, stat.b	Regressionskoeffizienten
stat.r ²	Bestimmungskoeffizient
stat.r	Korrelationskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten X -Liste, die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorie</i> und <i>Mit</i> -Kategorien verwendet wurde

Ausgabeveriable	Beschreibung
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

LinRegMx

Katalog > 

LinRegMx *X,Y[,Häuf][,Kategorie,Mit]*

Berechnet die lineare Regression $y = m \cdot x + b$ auf Liste *X* und *Y* mit der Häufigkeit *Häuf*. Eine

Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden Datenpunkt *X* und *Y* an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $m \cdot x + b$
stat.m, stat.b	Regressionskoeffizienten
stat.r ²	Bestimmungskoeffizient
stat.r	Korrelationskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den

Ausgabevariable	Beschreibung
	Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

LinRegtIntervals (Lineare Regressions-t-Intervalle)

Katalog > 

LinRegtIntervals *X,Y[,F[0[,KStufe]]]*

Für Steigung. Berechnet ein Konfidenzintervall des Niveaus K für die Steigung.

LinRegtIntervals *X,Y[,F[1,XWert[,KStufe]]]*

Für Antwort. Berechnet einen vorhergesagten y-Wert, ein Niveau-K-Vorhersageintervall für eine einzelne Beobachtung und ein Niveau-K-Konfidenzintervall für die mittlere Antwort.

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

F ist eine optionale Liste von Frequenzwerten. Jedes Element in *F* gibt die Häufigkeit für jeden entsprechenden *X* und *Y* Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzahlen ≥ 0 sein.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a+b \cdot x$
stat.a, stat.b	Regressionskoeffizienten
stat.df	Freiheitsgrade
stat.r ²	Bestimmungskoeffizient

AusgabevARIABLE	Beschreibung
stat.r	Korrelationskoeffizient
stat.Resid	Residuen von der Regression

Nur für Steigung

AusgabevARIABLE	Beschreibung
[stat.CLower, stat.CUpper]	Konfidenzintervall für die Steigung
stat.ME	Konfidenzintervall-Fehlertoleranz
stat.SESlope	Standardfehler der Steigung
stat.s	Standardfehler an der Linie

Nur für Antwort

AusgabevARIABLE	Beschreibung
[stat.CLower, stat.CUpper]	Konfidenzintervall für die mittlere Antwort
stat.ME	Konfidenzintervall-Fehlertoleranz
stat.SE	Standardfehler der mittleren Antwort
[stat.LowerPred, stat.UpperPred]	Vorhersageintervall für eine einzelne Beobachtung
stat.MEPred	Vorhersageintervall-Fehlertoleranz
stat.SEPred	Standardfehler für Vorhersage
stat. \hat{y}	a + b · XWert

LinRegtTest (t-Test bei linearer Regression)

Katalog > 

LinRegtTest X,Y[,Häuf[,Hypoth]]

Berechnet eine lineare Regression auf den X - und Y -Listen und einen t -Test auf dem Wert der Steigung β und den Korrelationskoeffizienten ρ für die Gleichung $y = \alpha + \beta x$. Er berechnet die Null-Hypothese $H_0: \beta = 0$ (gleichwertig, $\rho = 0$) in Bezug auf eine von drei alternativen Hypothesen.

Alle Listen müssen die gleiche Dimension besitzen.

X und Y sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden X- und Y-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Hypoth ist ein optionaler Wert, der eine von drei alternativen Hypothesen angibt, in Bezug auf die die Nullhypothese ($H_0: \beta = p = 0$) untersucht wird.

Für $H_a: \beta > 0$ und $p > 0$ (Standard) setzen Sie *Hypoth*=0

Für $H_a: \beta < 0$ und $p < 0$ setzen Sie *Hypoth*<0

Für $H_a: \beta > 0$ und $p > 0$ setzen Sie *Hypoth*>0

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a + b \cdot x$
stat.t	t-Statistik für Signifikanztest
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade
stat.a, stat.b	Regressionskoeffizienten
stat.s	Standardfehler an der Linie
stat.SESlope	Standardfehler der Steigung
stat.r ²	Bestimmungskoeffizient
stat.r	Korrelationskoeffizient
stat.Resid	Residuen von der Regression

linSolve()

Katalog >

linSolve(*SystemLinearerGl*, *Var1*, *Var2*, ...) \Rightarrow Liste

$$\text{linSolve}\left(\begin{cases} 2x+4y=3 \\ 5x-3y=7 \end{cases}, \{x,y\}\right) \quad \left\{ \frac{37}{26}, \frac{1}{26} \right\}$$

linSolve(*LineareGl1 and LineareGl2 and ...*, *Var1*, *Var2*, ...) \Rightarrow Liste

$$\text{linSolve}\left(\begin{cases} 2x=3 \\ 5x-3y=7 \end{cases}, \{x,y\}\right) \quad \left\{ \frac{3}{2}, \frac{1}{6} \right\}$$

linSolve(*{LineareGl1, LineareGl2, ...}*, *Var1*, *Var2*, ...) \Rightarrow Liste

$$\text{linSolve}\left(\begin{cases} apple+4\cdot pear=23 \\ 5\cdot apple-pear=17 \end{cases}, \{apple,pear\}\right) \quad \left\{ \frac{13}{3}, \frac{14}{3} \right\}$$

linSolve(*SystemLinearerGl*, {*Var1*, *Var2*, ...}) \Rightarrow Liste

$$\text{linSolve}\left(\begin{cases} apple+4\cdot \frac{pear}{3}=14 \\ -apple+pear=6 \end{cases}, \{apple,pear\}\right) \quad \left\{ \frac{36}{13}, \frac{114}{13} \right\}$$

linSolve(*LineareGl1 and LineareGl2 and ...*, {*Var1*, *Var2*, ...}) \Rightarrow Liste

linSolve(*{LineareGl1, LineareGl2, ...}*, {*Var1*, *Var2*, ...}) \Rightarrow Liste

Liefert eine Liste mit Lösungen für die Variablen *Var1*, *Var2*, ...

Das erste Argument muss ein System linearer Gleichungen bzw. eine einzelne lineare Gleichung ergeben. Andernfalls tritt ein Argumentfehler auf.

Die Auswertung von **linSolve**(*x=1 and x=2,x*) führt beispielsweise zu dem Ergebnis "Argumentfehler".

Δlist() (Listendifferenz)

Katalog >

Δlist(*Liste1*) \Rightarrow Liste

$$\Delta\text{List}(\{20,30,45,70\}) \quad \{10,15,25\}$$

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **deltaList**(...) eintippen.

Ergibt eine Liste mit den Differenzen der aufeinander folgenden Elemente in *Liste1*. Jedes Element in *Liste1* wird vom folgenden Element in *Liste1* subtrahiert. Die Ergebnisliste enthält stets ein Element weniger als die ursprüngliche *Liste1*.

list►mat() (Liste in Matrix)

Katalog >

list►mat(*Liste [, ElementeProZeile]*)
⇒Matrix

Gibt eine Matrix zurück, die Zeile für Zeile mit den Elementen aus *Liste* aufgefüllt wurde.

list►mat({1,2,3})	[1 2 3]
list►mat({1,2,3,4,5},2)	[1 2 3 4 5 0]

ElementeProZeile gibt (sofern angegeben) die Anzahl der Elemente pro Zeile an.
Vorgabe ist die Anzahl der Elemente in *Liste* (eine Zeile).

Wenn *Liste* die resultierende Matrix nicht vollständig auffüllt, werden Nullen hinzugefügt.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **list@>mat(...)** eintippen.

►ln (Natürlicher Logarithmus)

Katalog >

Ausdr ►ln⇒Ausdruck

Führt dazu, dass der eingegebene *Ausdr* in einen Ausdruck umgewandelt wird, der nur natürliche Logarithmen (ln) enthält.

$$\left(\log_{10}(x) \right) \blacktriangleright \ln \frac{\ln(x)}{\ln(10)}$$

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>ln eintippen.

ln() (Natürlicher Logarithmus)

ctrl ex Tasten

ln(Ausdr)⇒Ausdruck

$$\ln(2.) \quad 0.693147$$

ln(Liste)⇒Liste

Gibt den natürlichen Logarithmus des Arguments zurück.

Bei Komplex-Formatmodus reell:

Gibt für eine Liste die natürlichen Logarithmen der einzelnen Elemente zurück.

$$\ln(\{-3,1,2,5\}) \quad \text{"Error: Non-real calculation"}$$

Bei Komplex-Formatmodus kartesisch:

$$\ln(\{-3,1,2,5\}) \quad \{ \ln(3)+\pi \cdot i, 0.182322, \ln(5) \}$$

ln() (Natürlicher Logarithmus)

ctrl ex Tasten

ln(Quadratmatrix I)⇒Quadratmatrix

Ergibt den natürlichen Matrix-Logarithmus von *Quadratmatrix I*. Dies ist nicht gleichbedeutend mit der Berechnung des natürlichen Logarithmus jedes einzelnen Elements. Näheres zum Berechnungsverfahren finden Sie im Abschnitt **cos()**.

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\ln \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 1.83145 + 1.73485 \cdot i & 0.009193 - 1.49086 \\ 0.448761 - 0.725533 \cdot i & 1.06491 + 0.623491 \\ -0.266891 - 2.08316 \cdot i & 1.12436 + 1.79018 \end{pmatrix}$$

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

LnReg

Katalog >

LnReg X, Y[, Häuf] [, Kategorie, Mit]

Berechnet die logarithmische Regression $y = a + b \cdot \ln(x)$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a + b \cdot \ln(x)$

Ausgabevariable	Beschreibung
stat.a, stat.b	Regressionskoeffizienten
stat.r ²	Koeffizient der linearen Bestimmtheit für transformierte Daten
stat.r	Korrelationskoeffizient für transformierte Daten ($\ln(x)$, y)
stat.Resid	Mit dem logarithmischen Modell verknüpfte Residuen
stat.ResidTrans	Residuen für die lineare Anpassung transformierter Daten
stat.XReg	Liste der Datenpunkte in der modifizierten X-Liste, die in der Regression mit den Beschränkungen für Häuf, Kategorieliste und Mit-Kategorien verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten Y-Liste, die schließlich in der Regression mit den Beschränkungen für Häuf, Kategorieliste und Mit-Kategorien verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für stat.XReg und stat.YReg

Local (Lokale Variable)

Katalog > 

Local Var1[, Var2] [, Var3] ...

Deklariert die angegebenen Variablen *Variable* als lokale Variablen. Diese Variablen existieren nur während der Auswertung einer Funktion und werden gelöscht, wenn die Funktion beendet wird.

Hinweis: Lokale Variablen sparen Speicherplatz, da sie nur temporär existieren. Außerdem stören sie keine vorhandenen globalen Variablenwerte. Lokale Variablen müssen für For-Schleifen und für das temporäre Speichern von Werten in mehrzeiligen Funktionen verwendet werden, da Änderungen globaler Variablen in einer Funktion unzulässig sind.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define rollcount()=Func

```

Local i
1 → i
Loop
If randInt(1,6)=randInt(1,6)
Goto end
i+1 → i
EndLoop
Lbl end
Return i
EndFunc
```

Done

rollcount()	16
rollcount()	3

Lock

Katalog >

LockVar1 [, Var2] [, Var3] ...

LockVar.

Sperrt die angegebenen Variablen bzw. die Variablengruppe. Gesperrte Variablen können nicht geändert oder gelöscht werden.

Die Systemvariable *Ans* können Sie nicht sperren oder entsperren, ebenso können Sie die Systemvariablengruppen *stat.* oder *tvm.* nicht sperren.

Hinweis: Der Befehl **Sperren (Lock)** löscht den Rückgängig/Wiederholen-Verlauf, wenn er für nicht gesperrte Variablen verwendet wird.

Siehe **unLock**, Seite 209, und **getLockInfo()**, Seite 88.

a:=65	65
Lock a	Done
getLockInfo(a)	1
a:=75	"Error: Variable is locked."
DelVar a	"Error: Variable is locked."
Unlock a	Done
a:=75	75
DelVar a	Done

log() (Logarithmus)

Tasten

log(Ausdr1[,Ausdr2])⇒Ausdruck

log(Liste1[,Ausdr2])⇒Liste

Gibt für den Logarithmus des Arguments zur Basis *Ausdr2* zurück.

$\log_{10}(2.)$	0.30103
$\log_4(2.)$	0.5
$\log_3(10) - \log_3(5)$	$\log_3(2)$

Hinweis: Siehe auch **Vorlage Logarithmus**, Seite 6.

Gibt bei einer Liste den Logarithmus der Elemente zur Basis *Ausdr2* zurück.

Wenn *Ausdr2* weggelassen wird, wird 10 als Basis verwendet.

Bei Komplex-Formatmodus reell:

$\log_{10}(\{-3,1,2,5\})$ Error: Non-real result

log(Quadratmatrix I[,Ausdr2])
⇒Quadratmatrix

Bei Komplex-Formatmodus kartesisch:

$\log_{10}(\{-3,1,2,5\})$
 $\left\{ \log_{10}(3) + 1.36438 \cdot i, 0.079181, \log_{10}(5) \right\}$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

log() (Logarithmus)

ctrl 10^x Tasten

Gibt den Matrix-Logarithmus von *Quadratmatrix1* zur Basis *Ausdr2* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Logarithmus jedes Elements zur Basis *Ausdr2*. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Wenn das Basisargument weggelassen wird, wird 10 als Basis verwendet.

$$\log_{10} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 0.795387 + 0.753438 \cdot i & 0.003993 - 0.6474 \cdot i \\ 0.194895 - 0.315095 \cdot i & 0.462485 + 0.2707 \cdot i \\ -0.115909 - 0.904706 \cdot i & 0.488304 + 0.7774 \cdot i \end{pmatrix}$$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangleleft und verwenden dann \blacktriangleright und \blacktriangleright , um den Cursor zu bewegen.

►logbase

Katalog >

Ausdr1 ►logbase(*Ausdr2*) \Rightarrow *Ausdruck*

Führt dazu, dass der eingegebene Ausdruck zu einem Ausdruck mit der Basis *Ausdr2* vereinfacht wird.

$$\log_3(10) - \log_5(5) \Rightarrow \log_5\left(\frac{10}{3}\right)$$

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>**logbase** (...) eintippen.

Logistic

Katalog >

Logistic *X, Y[, Häuf] [, Kategorie, Mit]*

Berechnet die logistische Regression $y = \frac{c}{1+a \cdot e^{-bx}}$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die

entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $c/(1+a \cdot e^{-bx})$
stat.a, stat.b, stat.c	Regressionskoeffizienten
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

LogisticD

LogisticD *X*, *Y* [, *[Iterationen]*, *[Häuf]* [, *Kategorie*, *Mit*]]

Berechnet die logistische Regression $y = (c/(1+a \cdot e^{-bx})-d)$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf* unter Verwendung einer bestimmten Anzahl von *Iterationen*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Iterationen ist ein optionaler Wert, der angibt, wie viele Lösungsversuche maximal stattfinden. Bei Auslassung wird 64 verwendet. Größere Werte führen in der Regel zu höherer Genauigkeit, aber

auch zu längeren Ausführungszeiten, und umgekehrt.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $c/(1+a \cdot e^{-bx})+d)$
stat.a, stat.b, stat.c, stat.d	Regressionskoeffizienten
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

Loop*Block***EndLoop**

Führt die in *Block* enthaltenen Anweisungen wiederholt aus. Beachten Sie, dass dies eine Endlosschleife ist. Beenden Sie sie, indem Sie die Anweisung **Goto** oder **Exit** in *Block* ausführen.

Block ist eine Folge von Anweisungen, die durch das Zeichen ":" voneinander getrennt sind.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define <i>rollcount()</i> =Func	Done
Local <i>i</i>	
1→ <i>i</i>	
Loop	
If randInt(1,6)=randInt(1,6)	
Goto <i>end</i>	
<i>i</i> +1→ <i>i</i>	
EndLoop	
Lbl <i>end</i>	
Return <i>i</i>	
EndFunc	
	16
<i>rollcount()</i>	3

LU (Untere/obere Matrixzerlegung)

LU *Matrix, lMatrix, uMatrix, pMatrix [,Tol]*

Berechnet die Doolittle LU-Zerlegung (LR-Zerlegung) einer reellen oder komplexen Matrix. Die untere (bzw. linke) Dreiecksmatrix ist in *lMatrix* gespeichert, die obere (bzw. rechte) Dreiecksmatrix in *uMatrix* und die Permutationsmatrix (in welcher der bei der Berechnung vorgenommene Zeilentausch dokumentiert ist) in *pMatrix*.

$$lMatrix \cdot uMatrix = pMatrix \cdot Matrix$$

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

- Wenn Sie **ctrl** **enter** verwenden oder den

$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$	Done
LU <i>m1,lower,upper,perm</i>		
<i>lower</i>	$\begin{bmatrix} 1 & 0 & 0 \\ 5 & 1 & 0 \\ 6 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix}$	
<i>upper</i>	$\begin{bmatrix} 6 & 12 & 18 \\ 0 & 4 & 16 \\ 0 & 0 & 1 \end{bmatrix}$	
<i>perm</i>	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	

LU (Untere/obere Matrixzerlegung)

Katalog >

Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.

- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
 $5E^{-14} \cdot \max(\dim(Matrix)) \cdot \text{rowNorm}(Matrix)$

Der **LU**-Faktorisierungsalgorithmus verwendet partielle Pivotisierung mit Zeilentausch.

$\begin{bmatrix} m & n \\ o & p \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} m & n \\ o & p \end{bmatrix}$
LU <i>m1,lower,upper,perm</i>	Done
lower	$\begin{bmatrix} 1 & 0 \\ m & 1 \\ o & \end{bmatrix}$
upper	$\begin{bmatrix} o & p \\ 0 & n - \frac{m \cdot p}{o} \end{bmatrix}$
perm	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

M

mat►list() (Matrix in Liste)

Katalog >

mat►list(*Matrix*)⇒*Liste*

Gibt eine Liste zurück, die mit den Elementen aus *Matrix* gefüllt wurde. Die Elemente werden Zeile für Zeile aus *Matrix* kopiert.

mat►list([1 2 3])	{1,2,3}
$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$
mat►list(<i>m1</i>)	{1,2,3,4,5,6}

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **mat@>list(...)** eintippen.

max() (Maximum)

Katalog >

max(*Ausdr1, Ausdr2*)⇒*Ausdruck*

max(*Liste1, Liste2*)⇒*Liste*

max(*Matrix1, Matrix2*)⇒*Matrix*

Gibt das Maximum der beiden Argumente zurück. Wenn die Argumente zwei Listen oder Matrizen sind, wird eine Liste bzw. Matrix zurückgegeben, die den Maximalwert für jedes entsprechende Elementpaar enthält.

max(2.3,1.4)	2.3
max({1,2},{-4,3})	{1,3}

max(*Liste*)⇒*Ausdruck*

max({0,1,-7,1.3,0.5})	1.3
-----------------------	-----

Gibt das größte Element von *Liste* zurück.

max() (Maximum)

Katalog >

max(Matrix1)⇒Matrix

Gibt einen Zeilenvektor zurück, der das größte Element jeder Spalte von *Matrix1* enthält.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

Hinweis: Siehe auch **fMax()** und **min()**.

$$\text{max}\left(\begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{bmatrix}\right) \quad [1 \ 0 \ 7]$$

mean() (Mittelwert)

Katalog >

mean(Liste[, Häufigkeitsliste])
⇒Ausdruck

Gibt den Mittelwert der Elemente in *Liste* zurück.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

mean(Matrix1[, Häufigkeitsmatrix])
⇒Matrix

Ergibt einen Zeilenvektor aus den Mittelwerten aller Spalten in *Matrix1*.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

$$\begin{array}{ll} \text{mean}(\{0.2,0.1,-0.3,0.4\}) & 0.26 \\ \text{mean}(\{1,2,3\},\{3,2,1\}) & \frac{5}{3} \end{array}$$

Im Vektorformat kartesisch:

$$\begin{array}{ll} \text{mean}\left(\begin{bmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{bmatrix}\right) & [-0.133333 \ 0.833333] \\ \hline \text{mean}\left(\begin{bmatrix} \frac{1}{5} & 0 \\ -1 & 3 \\ \frac{2}{5} & -\frac{1}{2} \\ 5 & 2 \end{bmatrix}\right) & \left[\frac{-2}{15} \ \frac{5}{6}\right] \\ \hline \text{mean}\left(\begin{bmatrix} 1 & 2 & 5 & 3 \\ 3 & 4 & 4 & 1 \\ 5 & 6 & 6 & 2 \end{bmatrix}\right) & \left[\frac{47}{15} \ \frac{11}{3}\right] \end{array}$$

median() (Median)

Katalog >

median(Liste[, freqList])⇒Ausdruck

$$\text{median}(\{0.2,0.1,-0.3,0.4\}) \quad 0.2$$

Gibt den Medianwert der Elemente in *Liste* zurück.

Jedes *freqList*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

median() (Median)

Katalog >

median(*Matrix1[, freqMatrix]*) \Rightarrow Matrix

Gibt einen Zeilenvektor zurück, der die Medianwerte der einzelnen Spalten von *Matrix1* enthält.

$$\text{median} \begin{pmatrix} 0.2 & 0 \\ 1 & -0.3 \\ 0.4 & -0.5 \end{pmatrix} \quad [0.4 \quad -0.3]$$

Jedes *freqMatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

Hinweise:

- Alle Elemente der Liste bzw. der Matrix müssen zu Zahlen vereinfachbar sein.
- Leere (ungültige) Elemente in der Liste oder Matrix werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

MedMed

Katalog >

MedMed *X, Y [, Häuf] [, Kategorie, Mit]*

Berechnet die Median-Median-Linie $y = (m \cdot x + b)$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabeveriable	Beschreibung
stat.RegEqn	Median-Median-Linien-Gleichung: $m \cdot x + b$
stat.m, stat.b	Modellkoeffizienten
stat.Resid	Residuen von der Median-Median-Linie
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

mid() (Teil-String)

Katalog >

mid(Quellstring, Start[, Anzahl]) \Rightarrow String

Gibt *Anzahl* Zeichen aus der Zeichenkette *Quellstring* ab dem Zeichen mit der Nummer *Start* zurück.

Wird *Anzahl* weggelassen oder ist sie größer als die Länge von *Quellstring*, werden alle Zeichen von *Quellstring* ab dem Zeichen mit der Nummer *Start* zurückgegeben.

Anzahl muss ≥ 0 sein. Bei *Anzahl* = 0 wird eine leere Zeichenkette zurückgegeben.

mid(Quellliste, Start [, Anzahl]) \Rightarrow Liste

Gibt *Anzahl* Elemente aus *Quellliste* ab dem Element mit der Nummer *Start* zurück.

Wird *Anzahl* weggelassen oder ist sie größer als die Dimension von *Quellliste*, werden alle Elemente von *Quellliste* ab dem Element mit der Nummer *Start* zurückgegeben.

Anzahl muss ≥ 0 sein. Bei *Anzahl* = 0 wird eine leere Liste zurückgegeben.

mid(QuellstringListe, Start[, Anzahl])
 \Rightarrow Liste

Gibt *Anzahl* Strings aus der Stringliste

mid("Hello there",2)	"ello there"
mid("Hello there",7,3)	"the"
mid("Hello there",1,5)	"Hello"
mid("Hello there",1,0)	"[]"

mid({9,8,7,6},3)	{7,6}
mid({9,8,7,6},2,2)	{8,7}
mid({9,8,7,6},1,2)	{9,8}
mid({9,8,7,6},1,0)	{}

mid({ "A", "B", "C", "D" },2,2)	{"B", "C"}
---------------------------------	------------

mid() (Teil-String)

Katalog >

QuellstringListe ab dem Element mit der Nummer Start zurück.

min() (Minimum)

Katalog >

min(Ausdr1, Ausdr2)⇒Ausdruck

$\min(2,3,1,4)$ 1.4

min(Liste1, Liste2)⇒Liste

$\min(\{1,2\}, \{-4,3\})$ $\{-4,2\}$

min(Matrix1, Matrix2)⇒Matrix

Gibt das Minimum der beiden Argumente zurück. Wenn die Argumente zwei Listen oder Matrizen sind, wird eine Liste bzw. Matrix zurückgegeben, die den Minimalwert für jedes entsprechende Elementpaar enthält.

min(Liste)⇒Ausdruck

$\min(\{0,1,-7,1,3,0,5\})$ -7

Gibt das kleinste Element von *Liste* zurück.

min(Matrix1)⇒Matrix

$\min\begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & 0.3 \end{bmatrix}$ $\begin{bmatrix} -4 & -3 & 0.3 \end{bmatrix}$

Gibt einen Zeilenvektor zurück, der das kleinste Element jeder Spalte von *Matrix1* enthält.

Hinweis: Siehe auch **fMin()** und **max()**.

mirr()

Katalog >

mirr

(

Finanzierungsrate

$list1 := \{6000, -8000, 2000, -3000\}$

$\{6000, -8000, 2000, -3000\}$

,*Reinvestitionsrate*,*CF0*,*CFListe*

$list2 := \{2, 2, 2, 1\}$

$\{2, 2, 2, 1\}$

[*CFFreq*])

$\text{mirr}(4.65, 12, 5000, list1, list2)$ 13.41608607

Finanzfunktion, die den modifizierten internen Zinsfluss einer Investition zurückgibt.

Finanzierungsrate ist der Zinssatz, den Sie für die Cash-Flow-Beträge zahlen.

Reinvestitionsrate ist der Zinssatz, zu dem die Cash-Flows reinvestiert werden.

CF0 ist der Anfangs-Cash-Flow zum Zeitpunkt 0; dies muss eine reelle Zahl sein.

CFListe ist eine Liste von Cash-Flow-Beträgen nach dem Anfangs-Cash-Flow *CF0*.

CFFreq ist eine optionale Liste, in der jedes Element die Häufigkeit des Auftretens für einen gruppierten (fortlaufenden) Cash-Flow-Betrag angibt, der das entsprechende Element von *CFListe* ist. Der Standardwert ist 1; wenn Sie Werte eingeben, müssen diese positive Ganzahlen < 10.000 sein.

Hinweis: Siehe auch **irr()**, Seite 98.

mod() (Modulo)

mod(Ausdr1, Ausdr2)⇒Ausdruck

mod(7,0)	7
----------	---

mod(Liste1, Liste2)⇒Liste

mod(7,3)	1
----------	---

mod(Matrix1, Matrix2)⇒Matrix

mod(-7,3)	-2
-----------	----

Gibt das erste Argument modulo das zweite Argument gemäß der folgenden Identitäten zurück:

mod(-7,-3)	-1
------------	----

mod({12,-14,16},{9,7,-5})	{3,0,-4}
---------------------------	----------

$$\text{mod}(x,0) = x$$

$$\text{mod}(x,y) = x - y \text{ floor}(x/y)$$

Ist das zweite Argument ungleich Null, ist das Ergebnis in diesem Argument periodisch. Das Ergebnis ist entweder Null oder besitzt das gleiche Vorzeichen wie das zweite Argument.

Sind die Argumente zwei Listen bzw. zwei Matrizen, wird eine Liste bzw. Matrix zurückgegeben, die den Modulus jedes Elementpaares enthält.

Hinweis: Siehe auch **remain()**, Seite 156

mRow() (Matrixzeilenoperation)

mRow(Ausdr, Matrix1, Index)⇒Matrix

$\text{mRow}\left(\frac{-1}{3}, \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, 2\right)$	$\begin{bmatrix} 1 & 2 \\ -1 & 4 \\ 3 & 3 \end{bmatrix}$
---	--

Gibt eine Kopie von *Matrix1* zurück, in der jedes Element der Zeile *Index* von *Matrix1* mit *Ausdr* multipliziert ist.

mRowAdd() (Matrixzeilenaddition)**Katalog >**

mRowAdd(*Ausdr, Matrix1, Index1, Index2*)
 \Rightarrow Matrix

Gibt eine Kopie von *Matrix1* zurück, wobei jedes Element in Zeile *Index2* von *Matrix1* ersetzt wird durch:

$$\text{Ausdr} \times \text{Zeile } \text{Index1} + \text{Zeile } \text{Index2}$$

$\text{mRowAdd}\left(-3, \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, 1, 2\right)$	$\begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix}$
$\text{mRowAdd}\left(n, \begin{bmatrix} a & b \\ c & d \end{bmatrix}, 1, 2\right)$	$\begin{bmatrix} a & b \\ a \cdot n + c & b \cdot n + d \end{bmatrix}$

MultReg**Katalog >**

MultReg *Y, X1[,X2[,X3,...[,X10]]]*

Berechnet die lineare Mehrfachregression der Liste *Y* für die Listen *X1, X2, ..., X10*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+\dots$
stat.b0, stat.b1, ...	Regressionskoeffizienten
stat.R ²	Multiples Bestimmtheitsmaß
stat.yList	\hat{y} List = $b_0+b_1 \cdot x_1+\dots$
stat.Resid	Residuen von der Regression

MultRegIntervals**Katalog >**

MultRegIntervals *Y, X1[,X2[,X3,...[,X10]]],XWertListe[,KNiveau]*

Berechnet einen vorhergesagten y-Wert, ein Niveau-K-Vorhersageintervall für eine einzelne Beobachtung und ein Niveau-K-Konfidenzintervall für die mittlere Antwort.

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

Ausgabeverable	Beschreibung
stat.RegEqn	Regressionsgleichung: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+\dots$
stat. \hat{y}	Eine Punktschätzung: $\hat{y} = b_0 + b_1 \cdot x_1 + \dots$ für $XWertListe$
stat.dfError	Fehler-Freiheitsgrade
stat.CLower, stat.CUpper	Konfidenzintervall für eine mittlere Antwort
stat.ME	Konfidenzintervall-Fehlertoleranz
stat.SE	Standardfehler der mittleren Antwort
stat.LowerPred,	Vorhersageintervall für eine einzelne Beobachtung
stat.UpperrPred	
stat.MEPred	Vorhersageintervall-Fehlertoleranz
stat.SEPred	Standardfehler für Vorhersage
stat.bList	Liste der Regressionskoeffizienten, $\{b_0, b_1, b_2, \dots\}$
stat.Resid	Residuen von der Regression

MultRegTests $Y, X1[,X2[,X3,\dots[,X10]]]$

Der lineare Mehrfachregressionstest berechnet eine lineare Mehrfachregression für die gegebenen Daten sowie die globale F -Teststatistik und t -Teststatistik für die Koeffizienten.

Eine Zusammenfassung der Ergebnisse wird in der Variablen `stat.results` gespeichert. (Seite 186.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

Ausgaben

Ausgabeverable	Beschreibung
stat.RegEqn	Regressionsgleichung: $b_0+b_1 \cdot x_1+b_2 \cdot x_2+\dots$
stat.F	Globale F -Testgröße

Ausgabeveriable	Beschreibung
stat.PVal	Mit globaler F -Statistik verknüpfter P-Wert
stat.R ²	Multiples Bestimmtheitsmaß
stat.AdjR ²	Angepasster Koeffizient des multiplen Bestimmtheitsmaßes
stat.s	Standardabweichung des Fehlers
stat.DW	Durbin-Watson-Statistik; bestimmt, ob in dem Modell eine Autokorrelation erster Ordnung vorhanden ist
stat.dfReg	Regressions-Freiheitsgrade
stat.SSReg	Summe der Regressionsquadrate
stat.MSReg	Mittlere Regressionsstreuung
stat.dfError	Fehler-Freiheitsgrade
stat.SSError	Summe der Fehlerquadrate
stat.MSError	Mittleres Fehlerquadrat
stat.bList	{b0,b1,...} Liste der Koeffizienten
stat.tList	Liste der t-Testgrößen, eine für jeden Koeffizienten in b-Liste
stat.PList	Liste der P-Werte für jede t-Testgröße
stat.SEList	Liste der Standardfehler für Koeffizienten in b-Liste
stat.yList	\hat{y} List = $b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Residuen von der Regression
stat.sResid	Standardisierte Residuen; wird durch Division eines Residiums durch die Standardabweichung ermittelt
stat.CookDist	Cookscher Abstand; Maß für den Einfluss einer Beobachtung auf der Basis von Residuum und Hebelwert
stat.Leverage	Maß für den Abstand der Werte der unabhängigen Variable von den Mittelwerten (Hebelwerte)

N

nand

Tasten

BoolescherAusdr1 **nand** BoolescherAusdr2
ergibt Boolescher Ausdruck

$x \geq 3 \text{ and } x \geq 4 \quad x \geq 4$

$x \geq 3 \text{ nand } x \geq 4 \quad x < 4$

BoolescheListe1 **nand** BoolescheListe2
ergibt Boolesche Liste

*BoolescheMatrix1*nand*BoolescheMatrix2*
ergibt Boolesche Matrix

Gibt die Negation einer logischen **and** Operation auf beiden Argumenten zurück.
Gibt „wahr“, „falsch“ oder eine vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

*Ganzzahl1*nand*Ganzzahl2*⇒*Ganzzahl*

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer **nand**-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 64-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis dann 1, wenn beide Bits 1 sind; anderenfalls ist das Ergebnis 0. Der zurückgegebene Wert stellt die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 10).

3 and 4	0
3 nand 4	-1
{1,2,3} and {3,2,1}	{1,2,1}
{1,2,3} nand {3,2,1}	{-2,-3,-2}

nCr() (Kombinationen)

Katalog >

nCr(Ausdr1, Ausdr2)⇒Ausdruck

Für ganzzahlige *Ausdr1* und *Ausdr2* mit $Ausdr1 \geq Ausdr2 \geq 0$ ist **nCr()** die Anzahl der Möglichkeiten, *Ausdr1* Elemente aus *Ausdr2* Elementen auszuwählen (auch als Binomialkoeffizient bekannt). Beide Argumente können ganze Zahlen oder symbolische Ausdrücke sein.

$nCr(z,3)$	$\frac{z \cdot (z-1) \cdot (z-2)}{6}$
$Ans z=5$	10
$nCr(z,c)$	$\frac{z!}{c!(z-c)!}$
$\frac{Ans}{nPr(z,c)}$	$\frac{1}{c!}$

nCr(Ausdr, 0)⇒1

nCr(Ausdr, negGanzzahl)⇒0

nCr() (Kombinationen)**Katalog > **

**nCr(Ausdr, posGanzzahl)⇒ Ausdr ·
(Ausdr-1)...(Ausdr-posGanzzahl+1)/
posGanzzahl!**

**nCr(Ausdr, keineGanzzahl)⇒ Ausdr!/
((Ausdr-keineGanzzahl)! · keineGanzzahl!)**

nCr(Liste1, Liste2)⇒Liste

nCr({5,4,3},{2,4,2})

{10,1,3}

Gibt eine Liste von Binomialkoeffizienten auf der Basis der entsprechenden Elementpaare der beiden Listen zurück. Die Argumente müssen Listen gleicher Größe sein.

nCr(Matrix1, Matrix2)⇒Matrix

Gibt eine Matrix von Binomialkoeffizienten auf der Basis der entsprechenden Elementpaare der beiden Matrizen zurück. Die Argumente müssen Matrizen gleicher Größe sein.

nCr([6, 5], [2, 2])

[15, 10]
[6, 3]

nDerivative()**Katalog > **

**nDerivative(Ausdr1,Var=Wert[,Ordnung])
⇒Wert**

nDerivative(|x|,x=1)

1

nDerivative(|x|,x)|x=0

undef

nDerivative(sqrt(x-1),x)|x=1

undef

**nDerivative(Ausdr1,Var[,Ordnung]) |
Var=Wert⇒Wert**

Gibt die numerische Ableitung zurück, berechnet durch automatische Ableitungsmethoden.

Wenn Wert angegeben ist, setzt er jede vorausgegangene Variablenzuweisung oder jede aktuelle „|“ Ersetzung für die Variable außer Kraft.

Ordnung der Ableitung muss 1 oder 2 sein.

newList() (Neue Liste)**Katalog > **

newList(AnzElemente)⇒Liste

newList(4)

{0,0,0,0}

Gibt eine Liste der Dimension AnzElemente zurück. Jedes Element ist Null.

newMat() (Neue Matrix)

Katalog > 

newMat(AnzZeil, AnzSpalt)⇒Matrix

Gibt eine Matrix der Dimension *AnzZeil* mal *AnzSpalt* zurück, wobei die Elemente Null sind.

newMat(2,3)

[0 0 0]
[0 0 0]

nfMax() (Numerisches Funktionsmaximum)

Katalog > 

nfMax(Ausdr, Var)⇒Wert

nfMax(Ausdr, Var, UntereGrenze)⇒Wert

nfMax(Ausdr, Var, UntereGrenze, ObereGrenze)⇒Wert

nfMax($x^2 - 2 \cdot x - 1, x$)

-1.

nfMax($0.5 \cdot x^3 - x - 2, x, -5, 5$)

5.

**nfMax(Ausdr, Var) | UntereGrenze≤Var
≤ObereGrenze⇒Wert**

Gibt einen möglichen numerischen Wert der Variablen *Var* zurück, wobei das lokale Maximum von *Ausdr* auftritt.

Wenn Sie *UntereGrenze* und *ObereGrenze* ersetzen, sucht die Funktion in dem geschlossenen Intervall [*UntereGrenze*,*ObereGrenze*] für das lokale Maximum.

Hinweis: Siehe auch **fMax()** und **d()**.

nfMin() (Numerisches Funktionsminimum)

Katalog > 

nfMin(Ausdr, Var)⇒Wert

nfMin(Ausdr, Var, UntereGrenze)⇒Wert

nfMin(Ausdr, Var, UntereGrenze, ObereGrenze)⇒Wert

nfMin($x^2 + 2 \cdot x + 5, x$)

-1.

nfMin($0.5 \cdot x^3 - x - 2, x, -5, 5$)

-5.

**nfMin(Ausdr, Var) | UntereGrenze≤Var
≤ObereGrenze⇒Wert**

Gibt einen möglichen numerischen Wert der Variablen *Var* zurück, wobei das lokale Minimum von *Ausdr* auftritt.

Wenn Sie *UntereGrenze* und *ObereGrenze*

nfMin() (Numerisches Funktionsminimum)

Katalog >

ersetzen, sucht die Funktion in dem geschlossenen Intervall $[UntereGrenze, ObereGrenze]$ für das lokale Minimum.

Hinweis: Siehe auch fMin() und d().

nInt() (Numerisches Integral)

Katalog >

nInt(Ausdr1, Var, Untere, Obere)
⇒Ausdruck

$$\text{nInt}\left(e^{-x^2}, x, -1, 1\right)$$

1.49365

Wenn der Integrand *Ausdr1* außer *Var* keine anderen Variablen enthält und wenn *Untere* und *Obere* Konstanten oder positiv ∞ oder negativ ∞ sind, gibt nInt() eine Näherung für $\int(Ausdr1, Var, Untere, Obere)$ zurück. Diese Näherung ist der gewichtete Durchschnitt von Stichprobenwerten des Integranden im Intervall *Untere*<*Var*<*Obere*.

Das Berechnungsziel sind sechs signifikante Stellen. Der angewendete Algorithmus beendet die Weiterberechnung, wenn das Ziel hinreichend erreicht ist oder wenn weitere Stichproben wahrscheinlich zu keiner sinnvollen Verbesserung führen.

Wenn es scheint, dass das Berechnungsziel nicht erreicht wurde, wird die Meldung "Zweifelhafte Genauigkeit" angezeigt.

Sie können nInt() verschachteln, um mehrere numerische Integrationen durchzuführen. Die Integrationsgrenzen können von außerhalb liegenden Integrationsvariablen abhängen.

Hinweis: Siehe auch J(), Seite 223.

$$\begin{aligned} \text{nInt}\left(\cos(x), x, -\pi, \pi+1.\text{e}-12\right) &= 1.04144\text{e}-12 \\ \int_{\pi+10^{-12}}^{\pi+10^{-12}} \cos(x) dx &\approx -\sin\left(\frac{1}{1000000000000}\right) \end{aligned}$$

$$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x \cdot y}}{\sqrt{x^2-y^2}}, y, -x, x\right), x, 0, 1\right) = 3.30423$$

nom()

Katalog >

nom(Effektivzins, CpY)⇒Wert

$$\text{nom}(5.90398, 12)$$

5.75

Finanzfunktion zur Umrechnung des jährlichen Effektivzinssatzes *Effektivzins* in einen Nominalzinssatz, wobei *CpY* als Anzahl der Verzinsungsperioden pro Jahr

gegeben ist.

Effektivzins muss eine reelle Zahl sein und
CpY muss eine reelle Zahl > 0 sein.

Hinweis: Siehe auch **eff()**, Seite 66.

nor**ctrl****=****Tasten**

BoolescherAusdr1norBoolescherAusdr2
ergibt Boolescher Ausdruck

 $x \geq 3 \text{ or } x \geq 4$ $x \geq 3$ $x \geq 3 \text{ nor } x \geq 4$ $x < 3$

BoolescheListe1norBoolescheListe2
ergibt Boolesche Liste

BoolescheMatrix1norBoolescheMatrix2
ergibt Boolesche Matrix

Gibt die Negation einer logischen **or**
Operation auf beiden Argumenten zurück.
Gibt „wahr“ oder „falsch“ oder eine
vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die
Ergebnisse des Vergleichs der einzelnen
Elemente zurückgegeben.

Ganzzahl1norGanzzahl2 \Rightarrow Ganzzahl

Vergleicht zwei reelle ganze Zahlen mit
Hilfe einer **nor**-Operation Bit für Bit. Intern
werden beide ganzen Zahlen in binäre 64-
Bit-Zahlen mit Vorzeichen konvertiert. Beim
Vergleich der sich entsprechenden Bits ist
das Ergebnis dann 1, wenn beide Bits 1
sind; anderenfalls ist das Ergebnis 0. Der
zurückgegebene Wert stellt die Bit-
Ergebnisse dar und wird im jeweiligen
Basis-Modus angezeigt.

3 or 4	7
3 nor 4	-8
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} nor {3,2,1}	{-4,-3,-4}

Sie können die ganzen Zahlen in jeder Basis
eingeben. Für eine binäre oder
hexadezimale Eingabe ist das Präfix **0b**
bzw. **0h** zu verwenden. Ohne Präfix werden
ganze Zahlen als dezimal behandelt
(Basis 10).

norm()**Katalog >** **norm(Matrix)** \Rightarrow Ausdruck**norm(Vektor)** \Rightarrow Ausdruck

Gibt die Frobeniusnorm zurück.

norm($\begin{bmatrix} a & b \\ c & d \end{bmatrix}$)	$\sqrt{a^2+b^2+c^2+d^2}$
norm($\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$)	$\sqrt{30}$
norm($\begin{bmatrix} 1 & 2 \end{bmatrix}$)	$\sqrt{5}$
norm($\begin{bmatrix} 1 \\ 2 \end{bmatrix}$)	$\sqrt{5}$

normalLine()**Katalog >** **normalLine(Ausdr1,Var,Punkt)** \Rightarrow Ausdruck**normalLine(Ausdr1,Var=Punkt)** \Rightarrow AusdruckGibt die Normale zu der durch *Ausdr1* dargestellten Kurve an dem in *Var=Punkt* angegebenen Punkt zurück.Stellen Sie sicher, dass die unabhängige Variable nicht definiert ist. Wenn zum Beispiel $f1(x):=5$ und $x:=3$ ist, gibt **normalLine(f1(x),x,2)** "false" zurück.

normalLine(x^2 ,x,1)	$\frac{3}{2} - \frac{x}{2}$
normalLine($(x-3)^2 - 4$,x,3)	$x=3$
normalLine(x^3 ,x=0)	0
normalLine($\sqrt{ x }$,x=0)	undef

normCdf()**(Normalverteilungswahrscheinlichkeit)****Katalog >** **normCdf(untereGrenze,obereGrenze[,μ[,σ]])** \Rightarrow Zahl,
wenn *untereGrenze* und *obereGrenze* Zahlen sind,
Liste, wenn *untereGrenze* und *obereGrenze* Listen
sindBerechnet die Normalverteilungswahrscheinlichkeit zwischen *untereGrenze* und *obereGrenze* für die angegebenen μ (Standard = 0) und σ (Standard = 1).Für $P(X \leq obereGrenze)$ setzen Sie *untereGrenze* = $-\infty$.**normPdf() (Wahrscheinlichkeitsdichte)****Katalog >** **normPdf(XWert[,μ[,σ]])** \Rightarrow Zahl, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste istBerechnet die Wahrscheinlichkeitsdichtefunktion für die Normalverteilung an einem bestimmten *XWert*

für die vorgegebenen μ und σ .

not (nicht)

Katalog >

not

Boolescher Ausdrkl \Rightarrow *Boolescher Ausdruck*

Gibt „wahr“ oder „falsch“ oder eine vereinfachte Form des Arguments zurück.

not *Ganzzahl* \Rightarrow *Ganzzahl*

$\text{not}(2 \geq 3)$	true
$\text{not}(x < 2)$	$x \geq 2$
not not <i>innocent</i>	<i>innocent</i>

Im Hex-Modus:

Gibt das Einerkomplement einer reellen ganzen Zahl zurück. Intern wird *Ganzzahl1* in eine 32-Bit-Dualzahl mit Vorzeichen umgewandelt. Für das Einerkomplement werden die Werte aller Bits umgekehrt (so dass 0 zu 1 wird und umgekehrt). Die Ergebnisse werden im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen mit jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix wird die ganze Zahl als dezimal behandelt (Basis 10).

Geben Sie eine dezimale ganze Zahl ein, die für eine 64-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter **►Base2**, Seite 22.

Im Hex-Modus:

not 0h7AC36 0hEEEEEEEEE853C9

Im Bin-Modus

0b100101►Base10 37

not 0b10010

not 0b100101►Base10

-38

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

Hinweis: Eine binäre Eingabe kann bis zu 64 Stellen haben (das Präfix 0b wird nicht mitgezählt). Eine hexadezimale Eingabe kann bis zu 16 Stellen aufweisen.

nPr() (Permutationen)

Katalog > [a-z]

nPr(Ausdr1, Ausdr2) \Rightarrow Ausdruck

Für ganzzahlige Ausdr1 und Ausdr2 mit $Ausdr1 \geq Ausdr2 \geq 0$ ist **nPr()** die Anzahl der Möglichkeiten, Ausdr1 Elemente unter Berücksichtigung der Reihenfolge aus Ausdr2 Elementen auszuwählen. Beide Argumente können ganze Zahlen oder symbolische Ausdrücke sein.

$nPr(z, 3)$	$z \cdot (z-1) \cdot (z-2)$
$Ans z=5$	60
$nPr(z, -3)$	$\frac{1}{(z+1) \cdot (z+2) \cdot (z+3)}$
$nPr(z, c)$	$\frac{z!}{(z-c)!}$
$Ans \cdot nPr(z-c, -c)$	1

$$\mathbf{nPr}(Ausdr, 0) \Rightarrow 1$$

nPr() (Permutationen)

Katalog >

nPr(Ausdr, negGanzzahl) \Rightarrow $1 / ((Ausdr+1) \cdot (Ausdr+2) \cdots (Ausdr-negGanzzahl))$

nPr(Ausdr, posGanzzahl) \Rightarrow Ausdr $\cdot (Ausdr-1) \cdots (Ausdr-posGanzzahl+1)$

nPr(Ausdr, keineGanzzahl) \Rightarrow Ausdr! / (Ausdr-keineGanzzahl)!

nPr(Liste1, Liste2) \Rightarrow Liste

nPr({5,4,3},{2,4,2}) {20,24,6}

Gibt eine Liste der Permutationen auf der Basis der entsprechenden Elementpaare der beiden Listen zurück. Die Argumente müssen Listen gleicher Größe sein.

nPr(Matrix1, Matrix2) \Rightarrow Matrix

nPr([6 5][2 2], [30 20]
[4 3][2 2], [12 6])

Gibt eine Matrix der Permutationen auf der Basis der entsprechenden Elementpaare der beiden Matrizen zurück. Die Argumente müssen Matrizen gleicher Größe sein.

npv()

Katalog >

npv(Zinssatz,CFO,CFListe,[CFFreq])

list1:={6000,-8000,2000,-3000}
{6000,8000,2000,-3000}
list2:={2,2,2,1} {2,2,2,1}
npv(10,5000,list1,list2) 4769.91

Finanzfunktion zur Berechnung des Nettobarwerts; die Summe der Barwerte für die Bar-Zuflüsse und -Abflüsse. Ein positives Ergebnis für npv zeigt eine rentable Investition an.

Zinssatz ist der Satz, zu dem die Cash-Flows (der Geldpreis) für einen Zeitraum.

CFO ist der Anfangs-Cash-Flow zum Zeitpunkt 0; dies muss eine reelle Zahl sein.

CFListe ist eine Liste der Cash-Flow-Beträge nach dem anfänglichen Cash-Flow CF0.

CFFreq ist eine Liste, in der jedes Element die Häufigkeit des Auftretens für einen gruppierten (fortlaufenden) Cash-Flow-Betrag angibt, der das entsprechende Element von CFListe ist. Der Standardwert ist 1; wenn Sie Werte eingeben, müssen diese positive Ganzahlen < 10.000 sein.

nSolve() (Numerische Lösung)

nSolve(Gleichung,Var[=Schätzwert])
 \Rightarrow Zahl oder Fehler_String

**nSolve(Gleichung,Var
[=Schätzwert],UntereGrenze) \Rightarrow Zahl oder Fehler_String**

**nSolve(Gleichung,Var
[=Schätzwert],UntereGrenze,ObereGrenze)
 \Rightarrow Zahl oder Fehler_String**

**nSolve(Gleichung,Var[=Schätzwert]) |
UntereGrenze \leq Var \leq ObereGrenze \Rightarrow Zahl
oder Fehler_String**

Ermittelt iterativ eine reelle numerische Näherungslösung von *Gleichung* für deren eine Variable. Geben Sie die Variable an als:

Variable

– oder –

Variable = reelle Zahl

Beispiel: x ist gültig und x=3 ebenfalls.

nSolve() ist häufig sehr viel schneller als **solve()** oder **zeros()**, insbesondere, wenn zusätzlich der Operator “|” benutzt wird, um die Suche auf ein relativ kleines Intervall zu beschränken, das genau eine einzige Lösung enthält.

nSolve() versucht entweder einen Punkt zu ermitteln, wo der Unterschied zwischen tatsächlichem und erwartetem Wert Null ist oder zwei relativ nahe Punkte, wo der Restfehler entgegengesetzte Vorzeichen besitzt und nicht zu groß ist. Wenn nSolve() dies nicht mit einer kleinen Anzahl von Versuchen erreichen kann, wird die Zeichenkette “Keine Lösung gefunden” zurückgegeben.

Hinweis: Siehe auch **cSolve()**, **cZeros()**, **solve()** und **zeros()**.

nSolve($x^2 + 5 \cdot x - 25 = 9, x$)	3.84429
nSolve($x^2 = 4, x = -1$)	-2.
nSolve($x^2 = 4, x = 1$)	2.

Hinweis: Existieren mehrere Lösungen, können Sie mit Hilfe einer Schätzung eine bestimmte Lösung suchen.

nSolve($x^2 + 5 \cdot x - 25 = 9, x < 0$)	-8.84429
nSolve($\frac{(1+r)^{24} - 1}{r} = 26, r$) $r > 0$ and $r < 0.25$	0.006886
nSolve($x^2 = -1, x$)	"No solution found"

OneVar (Eine Variable)**Katalog > ****OneVar [1,]X1,[Häufigkeit],[,Kategorie,Mit]]****OneVar [n,]X1,X2[X3[,...,X20]]]**

Berechnet die 1-Variablenstatistik für bis zu 20 Listen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

Die X-Argumente sind Datenlisten.

Häufigkeit ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häufigkeit* gibt die Häufigkeit für jeden entsprechenden *X*-Wert an. Der Standardwert ist 1. Alle Elemente müssen Ganzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Ein leeres (ungültiges) Element in einer der Listen *X*, *Freq* oder *Kategorie* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Ein leeres (ungültiges) Element in einer der Listen *X1* bis *X20* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

Ausgabevariable	Beschreibung
stat. \bar{x}	Mittelwert der x-Werte
stat. Σx	Summe der x-Werte
stat. Σx^2	Summe der x^2 -Werte
stat.sx	Stichproben-Standardabweichung von x
stat. x	Populations-Standardabweichung von x
stat.n	Anzahl der Datenpunkte

Ausgabeveriable	Beschreibung
stat.MinX	Minimum der x-Werte
stat.Q ₁ X	1. Quartil von x
stat.MedianX	Median von x
stat.Q ₃ X	3. Quartil von x
stat.MaxX	Maximum der x-Werte
stat.SSX	Summe der Quadrate der Abweichungen der x-Werte vom Mittelwert

or (oder)

Katalog >

BoolescherAusdr1 **or** BoolescherAusdr2
ergibt Boolescher Ausdruck

$x \geq 3$ or $x \geq 4$

$x \geq 3$

BoolescheListe1 **or** BoolescheListe2 ergibt
Boolesche Liste

BoolescheMatrix1 **or** BoolescheMatrix2
ergibt Boolesche Matrix

Gibt „wahr“ oder „falsch“ oder eine vereinfachte Form des ursprünglichen Terms zurück.

Gibt „wahr“ zurück, wenn ein Ausdruck oder beide Ausdrücke zu „wahr“ ausgewertet werden. Gibt nur dann „falsch“ zurück, wenn beide Ausdrücke „falsch“ ergeben.

Define g(x)=Func
If $x \leq 0$ or $x \geq 5$
Goto end
Return $x \cdot 3$
Lbl end
EndFunc

g(3) 9
g(0) A function did not return a value

Hinweis: Siehe xor.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Ganzzahl1 **or** Ganzzahl2 \Rightarrow Ganzzahl

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer or-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 32-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis dann 1, wenn eines der Bits 1 ist; das Ergebnis ist nur dann 0, wenn beide Bits 0 sind. Der zurückgegebene Wert stellt

Im Hex-Modus:

0h7AC36 or 0h3D5F 0h7BD7F

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

0b100101 or 0b100 0b100101

die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 10).

Geben Sie eine dezimale ganze Zahl ein, die für eine 64-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter **►Base2**, Seite 22.

Hinweis: Siehe xor.

ord() (Numerischer Zeichencode)

ord(String)⇒Ganzzahl

ord(Liste l)⇒Liste

Gibt den Zahlenwert (Code) des ersten Zeichens der Zeichenkette *String* zurück. Handelt es sich um eine Liste, wird der Code des ersten Zeichens jedes Listenelements zurückgegeben.

ord("hello")	104
char(104)	"h"
ord(char[24])	24
ord({ "alpha", "beta" })	{ 97,98 }

P

►Rx() (Kartesische x-Koordinate)

►Rx(*rAusdr*, *θAusdr*)⇒Ausdruck

►Rx(*rListe*, *θListe*)⇒Liste

►Rx(*rMatrix*, *θMatrix*)⇒Matrix

Gibt die äquivalente x-Koordinate des Paares (*r*, *θ*) zurück.

Hinweis: Das *θ*-Argument wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Ist das Argument ein

Im Bogenmaß-Modus:

P►Rx(<i>r</i> , <i>θ</i>)	$\cos(\theta) \cdot r$
P►Rx(4,60°)	2
P►Rx({ -3,10,1.3 }, { $\frac{\pi}{3}$, $\frac{-\pi}{4}$, 0 })	$\left\{ \frac{-3}{2}, 5 \cdot \sqrt{2}, 1.3 \right\}$

P>Rx() (Kartesische x-Koordinate)

Katalog >

Ausdruck, können Sie $^{\circ}$, G oder r benutzen, um die Winkelmoduseinstellung temporär zu ändern.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **P@>Rx (...)** eintippen.

P>Ry() (Kartesische y-Koordinate)

Katalog >

P>Ry(rAusdr, θAusdr)⇒Ausdruck

P>Ry(rListe, θListe)⇒Liste

P>Ry(rMatrix, θMatrix)⇒Matrix

Gibt die äquivalente y-Koordinate des Paars (r, θ) zurück.

Hinweis: Das θ -Argument wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Ist das Argument ein Ausdruck, können Sie $^{\circ}$, G oder r benutzen, um die Winkelmoduseinstellung temporär zu ändern.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **P@>Ry (...)** eintippen.

Im Bogenmaß-Modus:

$$\begin{aligned} \text{P}\text{Ry}(r, \theta) &= \frac{\sin(\theta) \cdot r}{2 \cdot \sqrt{3}} \\ \text{P}\text{Ry}(4, 60^\circ) &= \frac{\sin(60^\circ) \cdot 4}{2 \cdot \sqrt{3}} \\ &= \frac{\frac{\sqrt{3}}{2} \cdot 4}{2 \cdot \sqrt{3}} \\ &= 2 \end{aligned}$$

PassErr (ÜbgebFeh)

Katalog >

PassErr

Übergibt einen Fehler an die nächste Stufe.

Ein Beispiel zu **PassErr** finden Sie im Beispiel 2 unter Befehl **Versuche (Try)**, Seite 202.

Wenn die Systemvariable *Fehlercode (errCode)* Null ist, tut **PassErr** nichts.

Das **Else** im Block **Try...Else...EndTry** muss **ClrErr** oder **PassErr** verwenden. Wenn der Fehler verarbeitet oder ignoriert werden soll, verwenden Sie **ClrErr**. Wenn nicht bekannt ist, was mit dem Fehler zu tun ist, verwenden Sie **PassErr**, um ihn an den nächsten Error Handler zu übergeben. Wenn keine weiteren **Try...Else...EndTry** Error Handler unerledigt sind, wird das Fehlerdialogfeld als normal angezeigt.

Hinweis: Siehe auch **Löehler**, Seite 31, und **Versuche**, Seite 202.

Hinweis zum Eingeben des Beispiels: Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

piecewise() (Stückweise)

piecewise(Ausdr1 [, Bedingung1 [, Ausdr2 [, Bedingung2 [, ...]]]])

Gibt Definitionen für einestückweise definierte Funktion in Form einer Liste zurück. Sie können auch mit Hilfe einer Vorlagestückweise Definitionen erstellen.

Define $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, & x \leq 0 \end{cases}$	Done
$p(1)$	1
$p(-1)$	undef

Hinweis: Siehe auch **Vorlage Stückweise**, Seite 7.

poissCdf()

poissCdf(λ , untereGrenze, obereGrenze) \Rightarrow Zahl, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

poissCdf(λ , obereGrenze) (für $P(0 \leq X \leq \text{obereGrenze}) \Rightarrow$ Zahl, wenn *obereGrenze* eine Zahl ist, Liste, wenn *obereGrenze* eine Liste ist

Berechnet die kumulative Wahrscheinlichkeit für die diskrete Poisson-Verteilung mit dem vorgegebenen Mittelwert λ .

Für $P(X \leq \text{obereGrenze})$ setzen Sie *untereGrenze* = 0

poissPdf()

poissPdf(λ , XWert) \Rightarrow Zahl, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeit für die diskrete Poisson-Verteilung mit dem vorgegebenen Mittelwert λ .

Polar

Katalog >

Vektor ►Polar

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @►Polar eintippen.

Zeigt *Vektor* in Polarform $[r \angle \theta]$ an. Der Vektor muss die Dimension 2 besitzen und kann eine Zeile oder eine Spalte sein.

Hinweis: ►Polar ist eine Anzeigeformatanweisung, keine Konvertierungsfunktion. Sie können sie nur am Ende einer Eingabezeile benutzen, und sie nimmt keine Aktualisierung von ans vor.

Hinweis: Siehe auch ►Rect, Seite 153.

komplexerWert ►Polar

Zeigt komplexerVektor in Polarform an.

- Der Grad-Modus für Winkel gibt $(r \angle \theta)$ zurück.
- Der Bogenmaß-Modus für Winkel gibt $r e^{i\theta}$ zurück.

komplexerWert kann jede komplexe Form haben. Eine $r e^{i\theta}$ -Eingabe verursacht jedoch im Winkelmodus Grad einen Fehler.

Hinweis: Für eine Eingabe in Polarform müssen Klammern $(r \angle \theta)$ verwendet werden.

$[1 \ 3.]$ ►Polar	$[3.16228 \ \angle 1.24905]$
$[x \ y]$ ►Polar	$\left[\sqrt{x^2+y^2} \ \angle \frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right) \right]$

Im Bogenmaß-Modus:

$(3+4 \cdot i)$ ►Polar	$e^{i \cdot \left(\frac{\pi}{2} - \tan^{-1}\left(\frac{3}{4}\right) \right) \cdot .5}$
$\left(4 \angle \frac{\pi}{3}\right)$ ►Polar	$e^{\frac{i \cdot \pi}{3} \cdot .4}$

Im Neugrad-Modus:

$(4 \cdot i)$ ►Polar	$(4 \angle 100)$
----------------------	------------------

Im Grad-Modus:

$(3+4 \cdot i)$ ►Polar	$(5 \angle 90 - \tan^{-1}\left(\frac{3}{4}\right))$
------------------------	---

polyCoeffs()

Katalog >

polyCoeffs(Poly [,Var])⇒Liste

Gibt eine Liste der Koeffizienten des Polynoms Poly mit Bezug auf die Variable Var zurück.

Poly muss ein Polynomausdruck in Var sein. Wir empfehlen, Var nicht wegzulassen, außer wenn Poly ein Ausdruck in einer einzelnen Variablen ist.

polyCoeffs($4 \cdot x^2 - 3 \cdot x + 2, x$)	{4, -3, 2}
--	------------

polyCoeffs($(x-1)^2 \cdot (x+2)^3$)	{1, 4, 1, -10, -4, 8}
---------------------------------------	-----------------------

Entwickelt das Polynom und wählt x für die

weggelassene Variable *Var*.

$\text{polyCoeffs}((x+y+z)^2, x)$	$\{1, 2 \cdot (y+z), (y+z)^2\}$
$\text{polyCoeffs}((x+y+z)^2, y)$	$\{1, 2 \cdot (x+z), (x+z)^2\}$
$\text{polyCoeffs}((x+y+z)^2, z)$	$\{1, 2 \cdot (x+y), (x+y)^2\}$

polyDegree()

polyDegree(Poly [,Var])⇒Wert

Gibt den Grad eines Polynomausdrucks *Poly* in Bezug auf die Variable *Var* zurück. Wenn Sie *Var* weglassen, wählt die Funktion **polyDegree()** einen Standardwert aus den im Polynom *Poly* enthaltenen Variablen aus.

Poly muss ein Polynom ausdruck in *Var* sein. Wir empfehlen, *Var* nicht wegzulassen, außer wenn *Poly* ein Ausdruck in einer einzelnen Variablen ist.

$\text{polyDegree}(5)$	0
$\text{polyDegree}(\ln(2)+\pi, x)$	0

Konstante Polynome

$\text{polyDegree}(4 \cdot x^2 - 3 \cdot x + 2, x)$	2
$\text{polyDegree}((x-1)^2 \cdot (x+2)^3)$	5

$\text{polyDegree}((x+y^2+z^3)^2, x)$	2
$\text{polyDegree}((x+y^2+z^3)^2, y)$	4

$\text{polyDegree}((x-1)^{10000}, x)$	10000
---------------------------------------	-------

Der Grad kann auch extrahiert werden, wenn dies für die Koeffizienten nicht möglich ist. Dies liegt daran, dass der Grad extrahiert werden kann, ohne das Polynom zu entwickeln.

polyEval() (Polynom auswerten)**Katalog >** **polyEval(Liste1, Ausdr1)⇒Ausdruck****polyEval(Liste1, Liste2)⇒Ausdruck**

Interpretiert das erste Argument als Koeffizienten eines nach fallenden Potenzen geordneten Polynoms und gibt das Polynom bezüglich des zweiten Arguments zurück.

$\text{polyEval}(\{a,b,c\},x)$	$a \cdot x^2 + b \cdot x + c$
$\text{polyEval}(\{1,2,3,4\},2)$	26
$\text{polyEval}(\{1,2,3,4\},\{2,-7\})$	{26,-262}

polyGcd()**Katalog >** **polyGcd(Ausdr1,Ausdr2)⇒Ausdruck**

Gibt den größten gemeinsamen Teiler der beiden Argumente zurück.

Ausdr1 und Ausdr2 müssen Polynomausdrücke sein.

Listen-, Matrix- und Boolesche Argumente sind nicht zulässig.

$\text{polyGcd}(100,30)$	10
$\text{polyGcd}(x^2-1,x-1)$	$x-1$
$\text{polyGcd}(x^3-6 \cdot x^2+11 \cdot x-6,x^2-6 \cdot x+8)$	$x-2$

polyQuotient()**Katalog >** **polyQuotient(Poly1,Poly2 [,Var])**
⇒Ausdruck

Gibt den Polynomquotienten von Poly1 geteilt durch Polynom Poly2 bezüglich der angegebenen Variable Var zurück.

Poly1 und Poly2 müssen Polynomausdrücke in Var sein. Wir empfehlen, Var nicht wegzulassen, außer wenn Poly1 und Poly2 Ausdrücke in denselben einzelnen Variablen sind.

$\text{polyQuotient}(x-1,x-3)$	1
$\text{polyQuotient}(x-1,x^2-1)$	0
$\text{polyQuotient}(x^2-1,x-1)$	$x+1$
$\text{polyQuotient}(x^3-6 \cdot x^2+11 \cdot x-6,x^2-6 \cdot x+8)$	x

$\text{polyQuotient}((x-y) \cdot (y-z),x+y+z,x)$	$y-z$
$\text{polyQuotient}((x-y) \cdot (y-z),x+y+z,y)$	$2 \cdot x - y + 2 \cdot z$

$\text{polyQuotient}((x-y) \cdot (y-z),x+y+z,z)$	$-(x-y)$
--	----------

polyRemainder()

Katalog >

polyRemainder(Poly1,Poly2 [,Var])

\Rightarrow Ausdruck

Gibt den Rest des Polynoms *Poly1* geteilt durch Polynom *Poly2* bezüglich der angegebenen Variablen *Var* zurück.

Poly1 und *Poly2* müssen

Polynomausdrücke in *Var* sein. Wir empfehlen, *Var* nicht wegzulassen, außer wenn *Poly1* und *Poly2* Ausdrücke in derselben einzelnen Variablen sind.

polyRemainder($x-1,x-3$)	2
polyRemainder($x-1,x^2-1$)	$x-1$
polyRemainder($x^2-1,x-1$)	0

polyRemainder($(x-y) \cdot (y-z),x+y+z,x$)	$(y-z) \cdot (2 \cdot y + z)$
polyRemainder($(x-y) \cdot (y-z),x+y+z,y$)	$-2 \cdot x^2 - 5 \cdot x \cdot z - 2 \cdot z^2$
polyRemainder($(x-y) \cdot (y-z),x+y+z,z$)	$(x-y) \cdot (x+2 \cdot y)$

polyRoots()

Katalog >

polyRoots(Poly,Var) \Rightarrow Liste

polyRoots(KoeffListe) \Rightarrow Liste

Die erste Syntax **polyRoots(Poly,Var)** gibt eine Liste mit reellen Wurzeln des Polynoms *Poly* bezüglich der Variablen *Var* zurück. Wenn keine reellen Wurzeln existieren, wird eine leere Liste zurückgegeben: { }.

Poly muss dabei ein Polynom in einer Variablen sein.

Die zweite Syntax **polyRoots(KoeffListe)** liefert eine Liste mit reellen Wurzeln für die Koeffizienten in *KoeffListe*.

Hinweis: Siehe auch **cPolyRoots()**, Seite 42.

polyRoots(y^3+1,y)	{-1}
cPolyRoots(y^3+1,y)	$\left\{-1, \frac{1}{2} - \frac{\sqrt{3}}{2} \cdot i, \frac{1}{2} + \frac{\sqrt{3}}{2} \cdot i\right\}$
polyRoots($x^2+2 \cdot x+1,x$)	{-1,-1}
polyRoots({1,2,1})	{-1,-1}

PowerReg

Katalog >

PowerReg X,Y [, Häuf] [, Kategorie, Mit]

Berechnet die Potenzregression $y = (a \cdot (x)^b)$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot (x)^b$
stat.a, stat.b	Regressionskoeffizienten
stat.r ²	Koeffizient der linearen Bestimmtheit für transformierte Daten
stat.r	Korrelationskoeffizient für transformierte Daten ($\ln(x)$, $\ln(y)$)
stat.Resid	Mit dem Potenzmodell verknüpfte Residuen
stat.ResidTrans	Residuen für die lineare Anpassung transformierter Daten
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

EndPrgm

Vorlage zum Erstellen eines benutzerdefinierten Programms. Muss mit dem Befehl **Definiere (Define)**, **Definiere LibPub (Define LibPub)** oder **Definiere LibPriv (Define LibPriv)** verwendet werden.

Block kann eine einzelne Anweisung, eine Reihe von durch das Zeichen ":" voneinander getrennten Anweisungen oder eine Reihe von Anweisungen in separaten Zeilen sein.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define *proggcd(a,b)*=Prgm

```
Local d
While b≠0
d:=mod(a,b)
a:=b
b:=d
Disp a," ",b
EndWhile
Disp "GCD=",a
EndPrgm
```

*Done**proggcd(4560,450)*

450	60
60	30
30	0
GCD=30	

*Done***prodSeq()****Siehe $\Pi()$, Seite 238.****Product (Π) (Produkt)****Siehe $\Pi()$, Seite 238.****product() (Produkt)**

Katalog >

product([Liste[, Start[, Ende]])⇒Ausdruck

Gibt das Produkt der Elemente von *Liste* zurück. *Start* und *Ende* sind optional. Sie geben einen Elementebereich an.

product([Matrix1[, Start[, Ende]])⇒Matrix

Gibt einen Zeilenvektor zurück, der die Produkte der Elemente aus den Spalten von *Matrix1* enthält. *Start* und *Ende* sind optional. Sie geben einen Zeilenbereich an.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

product({1,2,3,4}) 24

product({2,x,y}) 2·x·y

product({4,5,8,9},2,3) 40

product	$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$	[28 80 162]
---------	---	-------------

product	$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},1,2$	[4 10 18]
---------	---	-----------

propFrac() (Echter Bruch)

Katalog >

propFrac(Ausdr[*, Var*]) \Rightarrow Ausdruck

propFrac(rationale_Wert) gibt rationale_Wert als Summe einer ganzen Zahl und eines Bruchs zurück, der das gleiche Vorzeichen besitzt und dessen Nenner größer ist als der Zähler.

propFrac(rationaler_Ausdruck, Var) gibt die Summe der echten Brüche und ein Polynom bezüglich Var zurück. Der Grad von Var im Nenner übersteigt in jedem echten Bruch den Grad von Var im Zähler. Gleichartige Potenzen von Var werden zusammengefasst. Die Terme und Faktoren werden mit Var als der Hauptvariablen sortiert.

Wird Var weggelassen, wird eine Entwicklung des echten Bruchs bezüglich der wichtigsten Hauptvariablen vorgenommen. Die Koeffizienten des Polynomteils werden dann zuerst bezüglich der wichtigsten Hauptvariablen entwickelt usw.

Für rationale Ausdrücke ist **propFrac()** eine schnellere, aber weniger weitgehende Alternative zu **expand()**.

Mit der Funktion **propFrac()** können Sie gemischte Brüche darstellen und die Addition und Subtraktion bei gemischten Brüchen demonstrieren.

propFrac($\frac{4}{3}$)	$1\frac{1}{3}$
propFrac($\frac{-4}{3}$)	$-1\frac{1}{3}$

propFrac($\frac{x^2+x+1}{x+1} + \frac{y^2+y+1}{y+1}, x$)	$\frac{1}{x+1} + x + \frac{y^2+y+1}{y+1}$
propFrac(Ans)	$\frac{1}{x+1} + x + \frac{1}{y+1} + y$

Q

QR

Katalog >

QR Matrix, qMatrix, rMatrix[, Tol]

Berechnet die Householdersche QR-Faktorisierung einer reellen oder komplexen Matrix. Die sich ergebenden Q- und R-Matrizen werden in den angegebenen Matrix gespeichert. Die Q-Matrix ist unitär.

propFrac($\frac{11}{7}$)	$1\frac{4}{7}$
propFrac($3 + \frac{1}{11} + 5 + \frac{3}{4}$)	$8\frac{37}{44}$
propFrac($3 + \frac{1}{11} - \left(5 + \frac{3}{4}\right)$)	$-2\frac{29}{44}$

QR

Katalog >

Bei der R-Matrix handelt es sich um eine obere Dreiecksmatrix.

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \xrightarrow{\text{m1}} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

QR m1,qm,rm Done

$$qm \quad \begin{bmatrix} 0.123091 & 0.904534 & 0.408248 \\ 0.492366 & 0.301511 & -0.816497 \\ 0.86164 & -0.301511 & 0.408248 \end{bmatrix}$$

$$rm \quad \begin{bmatrix} 8.12404 & 9.60114 & 11.0782 \\ 0. & 0.904534 & 1.80907 \\ 0. & 0. & 0. \end{bmatrix}$$

- Wenn Sie **ctrl enter** verwenden oder den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
 $5E-14 \cdot \max(\dim(Matrix)) \cdot \text{rowNorm}(Matrix)$

Die QR-Faktorisierung wird anhand von Householderschen Transformationen numerisch berechnet. Die symbolische Lösung wird mit dem Gram-Schmidt-Verfahren berechnet. Die Spalten in *qMatName* sind die orthonormalen Basisvektoren, die den durch *Matrix* definierten Raum aufspannen.

$$\begin{bmatrix} m & n \\ o & p \end{bmatrix} \xrightarrow{\text{m1}} \begin{bmatrix} m & n \\ o & p \end{bmatrix}$$

QR m1,qm,rm Done

$$qm \quad \begin{bmatrix} \frac{m}{\sqrt{m^2+o^2}} & \frac{-\text{sign}(m \cdot p - n \cdot o) \cdot o}{\sqrt{m^2+o^2}} \\ \frac{o}{\sqrt{m^2+o^2}} & \frac{m \cdot \text{sign}(m \cdot p - n \cdot o)}{\sqrt{m^2+o^2}} \end{bmatrix}$$

$$rm \quad \begin{bmatrix} \frac{\sqrt{m^2+o^2}}{m} & \frac{m \cdot n + o \cdot p}{\sqrt{m^2+o^2}} \\ 0 & \frac{|m \cdot p - n \cdot o|}{\sqrt{m^2+o^2}} \end{bmatrix}$$

QuadReg

Katalog >

QuadReg X,Y[, Häuf] [, Kategorie, Mit]]

Berechnet die quadratische polynomiale Regression $y = a \cdot x^2 + b \cdot x + c$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot x^2 + b \cdot x + c$
stat.a, stat.b, stat.c	Regressionskoeffizienten
stat.R ²	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

QuartReg *X, Y [, Häuf] [, Kategorie, Mit]*]

Berechnet die polynomiale Regression vierter Ordnung = $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$ auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung

der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

Ausgabeverable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$
stat.a, stat.b, stat.c, stat.d, stat.e	Regressionskoeffizienten
stat.R ²	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X</i> -Liste, die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y</i> -Liste, die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien</i> verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

R►Pθ() (Polarkoordinate)**Katalog >** **R►Pθ (xAusdr, yAusdr)⇒Ausdruck**

Im Grad-Modus:

R►Pθ (xListe, yListe)⇒Liste**R►Pθ (xMatrix, yMatrix)⇒Matrix**

$$\text{R►Pθ}(x,y) = \frac{90 \cdot \text{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right)}{1}$$

Gibt die äquivalente θ-Koordinate des Paares (x,y) zurück.**Hinweis:** Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.**Hinweis:** Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **R@>Ptheta (...) eintippen.**

Im Neugrad-Modus:

R►Pθ(x,y)

$$\frac{100 \cdot \text{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right)}{1}$$

Im Bogenmaß-Modus:

R►Pθ(3,2)

$$\frac{\tan^{-1}\left(\frac{2}{3}\right)}{1}$$

$$\begin{aligned} \text{R►Pθ}\left[\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix}\right] \\ \left[0 \quad \tan^{-1}\left(\frac{16}{\pi}\right) + \frac{\pi}{2} \quad 0.643501 \right] \end{aligned}$$

R►Pr() (Polarkoordinate)**Katalog >** **R►Pr (xAusdr, yAusdr)⇒Ausdruck**

Im Bogenmaß-Modus:

R►Pr (xListe, yListe)⇒Liste**R►Pr (xMatrix, yMatrix)⇒Matrix**

$$\text{R►Pr}(x,y) = \sqrt{x^2+y^2}$$

Gibt die äquivalente r-Koordinate des Paares (x,y) zurück.**Hinweis:** Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **R@>Pr (...) eintippen.**

$$\text{R►Pr}(3,2) = \sqrt{13}$$

$$\begin{aligned} \text{R►Pr}\left[\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix}\right] \\ \left[3 \quad \frac{\sqrt{\pi^2+256}}{4} \quad 2.5 \right] \end{aligned}$$

►Rad (Bogenmaß)**Katalog >** **Ausdr!►Rad⇒Ausdruck**

Im Grad-Modus:

Wandelt das Argument ins Winkelmaß Bogenmaß um.

(1.5)►Rad

$$(0.02618)^r$$

►Rad (Bogenmaß)

Katalog >

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>Rad eintippen.

Im Neugrad-Modus:

{1.5}►Rad

{0.023562}^r

rand() (Zufallszahl)

Katalog >

rand()⇒Ausdruck

rand(#Trials)⇒Liste

rand() gibt einen Zufallswert zwischen 0 und 1 zurück.

rand(#Trials) gibt eine Liste zurück, die #Trials Zufallswerte zwischen 0 und 1 enthält.

Setzt Ausgangsbasis für Zufallszahlengenerierung.

RandSeed 1147

Done

rand(2)

{0.158206,0.717917}

randBin() (Zufallszahl aus Binomialverteilung)

Katalog >

randBin(n, p)⇒Ausdruck

randBin(80,0.5)

42

randBin(n, p, #Trials)⇒Liste

randBin(80,0.5,3)

{41,32,39}

randBin(n, p) gibt eine reelle Zufallszahl aus einer angegebenen Binomialverteilung zurück.

randBin(n, p, #Trials) gibt eine Liste mit #Trials reellen Zufallszahlen aus einer angegebenen Binomialverteilung zurück.

randInt() (Ganzzahlige Zufallszahl)

Katalog >

randInt(lowBound,upBound)⇒Ausdruck

randInt(3,10)

5

randInt(lowBound,upBound,#Trials)
⇒Liste

randInt(3,10,4)

{9,7,5,8}

randInt(lowBound,upBound) gibt eine ganzzahlige Zufallszahl innerhalb der durch UntereGrenze (lowBound) und ObereGrenze (upBound) festgelegten Grenzen zurück.

randInt() (Ganzzahlige Zufallszahl)

Katalog >

randInt(*lowBound, upBound, #Trials*) gibt eine Liste mit *#Trials* ganzzahligen Zufallszahlen innerhalb des festgelegten Bereichs zurück.

randMat() (Zufallsmatrix)

Katalog >

randMat(*AnzZeil, AnzSpalt*) \Rightarrow Matrix

Gibt eine Matrix der angegebenen Dimension mit ganzzahligen Werten zwischen -9 und 9 zurück.

Beide Argumente müssen zu ganzen Zahlen vereinfachbar sein.

RandSeed 1147

Done

randMat(3,3)

$$\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$$

Hinweis: Die Werte in dieser Matrix ändern sich mit jedem Drücken von **enter**.

randNorm() (Zufallsnorm)

Katalog >

randNorm(μ , σ) \Rightarrow Ausdruck

randNorm(μ , σ , #Versuche) \Rightarrow Liste

Gibt eine Dezimalzahl aus der Gaußschen Normalverteilung zurück. Dies könnte eine beliebige reelle Zahl sein, die Werte konzentrieren sich jedoch stark in dem Intervall [$\mu - 3 \cdot \sigma$, $\mu + 3 \cdot \sigma$].

randNorm(μ , σ , #Versuche) gibt eine Liste mit #Versuche Dezimalzahlen aus der angegebenen Normalverteilung zurück.

RandSeed 1147

Done

randNorm(0,1)

0.492541

randNorm(3,4.5)

-3.54356

randPoly() (Zufallspolynom)

Katalog >

randPoly(*Var, Ordnung*) \Rightarrow Ausdruck

Gibt ein Polynom in *Var* der angegebenen *Ordnung* zurück. Die Koeffizienten sind zufällige ganze Zahlen im Bereich -9 bis 9. Der führende Koeffizient ist nie Null.

Ordnung muss zwischen 0 und 99 betragen.

RandSeed 1147

Done

randPoly(*x,5*)

$$-2 \cdot x^5 + 3 \cdot x^4 - 6 \cdot x^3 + 4 \cdot x - 6$$

randSamp() (Zufallsstichprobe)

Katalog >

randSamp(*List*,#*Trials*[,*noRepl*]) \Rightarrow *Liste*

Gibt eine Liste mit einer Zufallsstichprobe von #*Trials* Versuchen aus *Liste* (*List*) zurück mit der Möglichkeiten, Stichproben zu ersetzen (*noRepl*=0) oder nicht zu ersetzen (*noRepl*=1). Die Vorgabe ist mit Stichprobenersatz.

Define <i>list3</i> = $\{1,2,3,4,5\}$	Done
Define <i>list4</i> = randSamp (<i>list3</i> ,6)	Done
<i>list4</i>	$\{2,3,4,3,1,2\}$

RandSeed (Zufallszahl)

Katalog >

RandSeed *Zahl*

Zahl = 0 setzt die Ausgangsbasis ("seed") für den Zufallszahlengenerator auf die Werkseinstellung zurück. Bei *Zahl* \neq 0 werden zwei Basen erzeugt, die in den Systemvariablen seed1 und seed2 gespeichert werden.

RandSeed 1147	Done
rand()	0.158206

real() (Reell)

Katalog >

real(*AusdrI*) \Rightarrow *Ausdruck*

Gibt den Realteil des Arguments zurück.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt. Siehe auch **imag()**, Seite 94.

real(*Listel*) \Rightarrow *Liste*

Gibt für jedes Element den Realteil zurück.

real(*Matrixl*) \Rightarrow *Matrix*

Gibt für jedes Element den Realteil zurück.

real($2+3 \cdot i$)	2
real(z)	z
real($x+i \cdot y$)	x

real($\{a+i \cdot b, 3, i\}$)	$\{a, 3, 0\}$
---------------------------------	---------------

real($\begin{bmatrix} a+i \cdot b & 3 \\ c & i \end{bmatrix}$)	$\begin{bmatrix} a & 3 \\ c & 0 \end{bmatrix}$
--	--

►Rect (Kartesisch)

Katalog >

Vektor ►Rect

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>**Rect** eintippen.

Zeigt *Vektor* in der kartesischen Form [x, y, z] an. Der Vektor muss die Dimension 2 oder 3 besitzen und kann eine Zeile oder

$\begin{bmatrix} 3 & \angle \frac{\pi}{4} & \angle \frac{\pi}{6} \end{bmatrix}$	►Rect
$\begin{bmatrix} 3 \cdot \sqrt{2} \\ 4 \\ 4 \end{bmatrix}$	$\begin{bmatrix} 3 \cdot \sqrt{2} & 3 \cdot \sqrt{2} & 3 \cdot \sqrt{3} \\ 4 & 4 & 2 \end{bmatrix}$
$\begin{bmatrix} a & \angle b & \angle c \\ a \cdot \cos(b) \cdot \sin(c) & a \cdot \sin(b) \cdot \sin(c) & a \cdot \cos(c) \end{bmatrix}$	

eine Spalte sein.

Hinweis: ►Rect ist eine Anzeigeformatanweisung, keine Konvertierungsfunktion. Sie können sie nur am Ende einer Eingabezeile benutzen, und sie nimmt keine Aktualisierung von ans vor.

Hinweis: Siehe auch ►Polar, Seite 140.

komplexerWert ►Rect

Zeigt komplexerWert in der kartesischen Form $a+bi$ an. KomplexerWert kann jede komplexe Form haben. Eine $re^{i\theta}$ -Eingabe verursacht jedoch im Winkelmodus Grad einen Fehler.

Hinweis: Für eine Eingabe in Polarform müssen Klammern ($r \angle \theta$) verwendet werden.

Im Bogenmaß-Modus:

$$\begin{array}{c} \left(\begin{array}{c} \frac{\pi}{3} \\ 4 \cdot e^3 \end{array} \right) \blacktriangleright \text{Rect} \\ \hline \left(\begin{array}{c} \frac{\pi}{3} \\ 4 \angle \frac{\pi}{3} \end{array} \right) \blacktriangleright \text{Rect} \end{array} \quad \frac{\pi}{3} \quad 2+2\sqrt{3} \cdot i$$

Im Neugrad-Modus:

$$\left(\begin{array}{c} 1 \angle 100 \end{array} \right) \blacktriangleright \text{Rect} \quad i$$

Im Grad-Modus:

$$\left(\begin{array}{c} 4 \angle 60 \end{array} \right) \blacktriangleright \text{Rect} \quad 2+2\sqrt{3} \cdot i$$

Hinweis: Wählen Sie zur Eingabe von \angle das Symbol aus der Sonderzeichenpalette des Katalogs aus.

ref() (Diagonalform)

ref(*MatrixI[, Tol]*) \Rightarrow Matrix

Gibt die Diagonalform von *MatrixI* zurück.

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

$$\text{ref} \left(\begin{array}{cccc} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{array} \right) \quad \begin{array}{cccc} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{array}$$

- Wenn Sie **ctrl enter** verwenden oder den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:

$$5E^{-14} \cdot \max(\dim(Matrix\,I)) \cdot \text{rowNorm}(Matrix\,I)$$

$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$
$\text{ref}(m1)$	$\begin{bmatrix} 1 & \frac{d}{c} \\ 0 & 1 \end{bmatrix}$

⚠

Vermeiden Sie nicht definierte Elemente in *Matrix1*. Sie können zu unerwarteten Ergebnissen führen.

Wenn z. B. im folgenden Ausdruck *a* nicht definiert ist, erscheint eine Warnmeldung und das Ergebnis wird wie folgt angezeigt:

$\text{ref}\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
---	---

Die Warnung erscheint, weil das verallgemeinerte Element $1/a$ für $a=0$ nicht zulässig wäre.

Sie können dieses Problem umgehen, indem Sie zuvor einen Wert in *a* speichern oder wie im folgenden Beispiel gezeigt eine Substitution mit dem womit-Operator „|“ vornehmen.

$\text{ref}\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} a=0$	$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$
---	---

Hinweis: Siehe auch **rref()**, Seite 164.

remain() (Rest)

Katalog >

remain(Ausdr1, Ausdr2)⇒Ausdruck

remain(Liste1, Liste2)⇒Liste

remain(Matrix1, Matrix2)⇒Matrix

Gibt den Rest des ersten Arguments bezüglich des zweiten Arguments gemäß folgender Definitionen zurück:

remain(x,0) x

remain(x,y) x-y · iPart(x/y)

Als Folge daraus ist zu beachten, dass **remain(-x,y) = -remain(x,y)**. Das Ergebnis ist entweder Null oder besitzt das gleiche Vorzeichen wie das erste Argument.

Hinweis: Siehe auch **mod()**, Seite 122.

remain(7,0)	7
remain(7,3)	1
remain(-7,3)	-1
remain(7,-3)	1
remain(-7,-3)	-1
remain({12,-14,16},{9,7,-5})	{3,0,1}

$$\text{remain}\left[\begin{bmatrix} 9 & -7 \\ 6 & 4 \end{bmatrix}, \begin{bmatrix} 4 & 3 \\ 4 & -3 \end{bmatrix}\right] = \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$

Request

Katalog >

RequestEingabeString, var[, FlagAnz [, statusVar]]

RequestEingabeString, func(arg1, ...argn) [, FlagAnz [, statusVar]]

Programmierbefehl: Pausiert das Programm und zeigt ein Dialogfeld mit der Meldung *EingabeString* sowie einem Eingabefeld für die Antwort des Benutzers an.

Wenn der Benutzer eine Antwort eingibt und auf **OK** klickt, wird der Inhalt des Eingabefelds in die Variable *var* geschrieben.

Falls der Benutzer auf **Abbrechen** klickt, wird das Programm fortgesetzt, ohne Eingaben zu übernehmen. Das Programm verwendet den vorherigen *var*-Wert, soweit *var* bereits definiert wurde.

Bei dem optionalen Argument *FlagAnz* kann es sich um einen beliebigen Ausdruck handeln.

- Wenn *FlagAnz* fehlt oder den Wert **1**

Definieren Sie ein Programm:

Define request_demo()=Prgm

Request "Radius: ",r

Disp "Fläche =",pi*r^2

EndPrgm

Starten Sie das Programm und geben Sie eine Antwort ein:

request_demo()



Ergebnis nach Auswahl von **OK**:

Radius: 6/2

Fläche = 28.2743

ergibt, werden die Eingabeaufforderung und die Benutzerantwort im Calculator-Protokoll angezeigt.

- Wenn *FlagAnz* den Wert **0** ergibt, werden die Aufforderung und die Antwort nicht im Protokoll angezeigt.

Das optionale Argument *statusVar* ermöglicht es dem Programm, zu bestimmen, wie der Benutzer das Dialogfeld verlassen hat. Beachten Sie bitte, dass *statusVar* das Argument *FlagAnz* erfordert.

- Wenn der Benutzer auf **OK** geklickt oder die **Eingabetaste** bzw. **Strg+Eingabetaste** gedrückt hat, wird die Variable *statusVar* auf den Wert **1** gesetzt.
- Andernfalls wird die Variable *statusVar* auf den Wert **0** gesetzt.

Mit dem Argument *func()* kann ein Programm die Benutzerantwort als Funktionsdefinition speichern. Diese Syntax verhält sich so, als hätte der Benutzer den folgenden Befehl ausgeführt:

*Define Fkt(Arg1, ...Argn) =
Benutzerantwort*

Anschließend kann das Programm die so definierte Funktion *Fkt()* nutzen. Die Meldung *EingabeString* sollte dem Benutzer die nötigen Informationen geben, damit dieser eine passende *Benutzerantwort* zur Vervollständigung der Funktionsdefinition eingeben kann.

Hinweis: Sie können den Befehl **Request** in benutzerdefinierten Programmen, aber nicht in Funktionen verwenden.

So halten Sie ein Programm an, das einen Befehl **Request** in einer Endlosschleife enthält:

- **Handheld:** Halten Sie die Taste  gedrückt und drücken Sie mehrmals

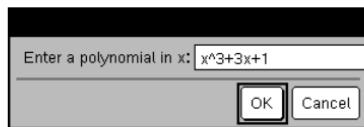
Definieren Sie ein Programm:

```
Define polynomial()=Prgm
Request "Polynom in x eingeben:",p(x)
Disp "Reelle Wurzeln:",polyRoots(p(x),x)
```

EndPrgm

Starten Sie das Programm und geben Sie eine Antwort ein:

polynomial()



Ergebnis nach Auswahl von **OK**:

Polynom in x eingeben: x^3+3x+1

Reelle Wurzeln: {-0.322185}

[enter].

- **Windows®:** Halten Sie die Taste **F12** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **Macintosh®:** Halten Sie die Taste **F5** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **iPad®:** Die App zeigt eine Eingabeaufforderung an. Sie können weiter warten oder abbrechen.

Hinweis: Siehe auch **RequestStr**, Seite 158.

RequestStr

RequestStr*EingabeString, Var[, FlagAnz]*

Programmierbefehl: Verhält sich genauso wie die erste Syntax des Befehls **Request**, die Benutzerantwort wird aber immer als String interpretiert. Der Befehl **Request** interpretiert die Antwort hingegen als Ausdruck, es sei denn, der Benutzer setzt sie in Anführungszeichen ("").

Hinweis: Sie können den Befehl **RequestStr** in benutzerdefinierten Programmen, aber nicht in Funktionen verwenden.

So halten Sie ein Programm an, das einen Befehl **RequestStr** in einer Endlosschleife enthält:

- **Handheld:** Halten Sie die Taste **[on]** gedrückt und drücken Sie mehrmals **[enter]**.
- **Windows®:** Halten Sie die Taste **F12** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **Macintosh®:** Halten Sie die Taste **F5** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **iPad®:** Die App zeigt eine Eingabeaufforderung an. Sie können weiter warten oder abbrechen.

Hinweis: Siehe auch **Request**, Seite 156.

Definieren Sie ein Programm:

```
Define requestStr_demo()=Prgm
  RequestStr "Ihr Name:",name,0
  Disp "Die Antwort hat ",dim(name)," Zeichen."
EndPrgm
```

Starten Sie das Programm und geben Sie eine Antwort ein:

requestStr_demo()



Ergebnis nach Auswahl von **OK** (Hinweis: Wegen *DispFlag = 0* werden Eingabeaufforderung und Antwort nicht im Protokoll angezeigt):

requestStr_demo()

Die Antwort hat 5 Zeichen.

Return (Rückgabe)

Katalog >

Return [Ausdr]

Gibt *Ausdr* als Ergebnis der Funktion zurück. Verwendbar in einem Block **Func...EndFunc**.

Hinweis: Verwenden Sie **Zurück (Return)** ohne Argument innerhalb eines **Prgm...EndPrgm** Blocks, um ein Programm zu beenden.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define **factorial (nn)=**

Func

Local *answer,counter*1 → *answer*For *counter,1,nn**answer·counter* → *answer*

EndFor

Return *answer*

EndFunc

factorial (3)

6

right() (Rechts)

Katalog >

right(Liste1[, Anz])⇒Liste**right({1,3,2,4},3)**

{3,2,4}

Gibt *Anz* Elemente zurück, die rechts in *Liste1* enthalten sind.

Wenn Sie *Anz* weglassen, wird die gesamte *Liste1* zurückgegeben.

right(Quellstring[, Anz])⇒String**right("Hello",2)**

"lo"

Gibt *Anz* Zeichen zurück, die rechts in der Zeichenkette *Quellstring* enthalten sind.

Wenn Sie *Anz* weglassen, wird der gesamte *Quellstring* zurückgegeben.

right(Vergleich)⇒Ausdruck**right(x<3)**

3

Gibt die rechte Seite einer Gleichung oder Ungleichung zurück.

rk23 ()

Katalog >

rk23(Ausdr, Var, abhVar, {Var0, VarMax}, abhVar0, VarSchritt [, diftol])⇒Matrix

Differentialgleichung:

 $y' = 0.001 * y * (100 - y)$ und $y(0) = 10$

rk23 ()

rk23(*AusdrSystem*, *Var*, *ListeAbhVar*, {*Var0*, *VarMax*}, *ListeAbhVar0*, *VarSchritt* [, *diftol*]) \Rightarrow Matrix

rk23(*AusdrListe*, *Var*, *ListeAbhVar*, {*Var0*, *VarMax*}, *ListeAbhVar0*, *VarSchritt* [, *diftol*]) \Rightarrow Matrix

Verwendet die Runge-Kutta-Methode zum Lösen des Systems

$$\frac{d \ depVar}{d \ Var} = Expr(Var, depVar)$$

mit *abhVar*(*Var0*)=*abhVar0* auf dem Intervall [*Var0*,*VarMax*]. Gibt eine Matrix zurück, deren erste Zeile die Ausgabewerte von *Var* definiert, wie durch *VarSchritt* definiert. Die zweite Zeile definiert den Wert der ersten Lösungskomponente an den entsprechenden *Var* Werten usw.

Ausdr ist die rechte Seite, die die gewöhnliche Differentialgleichung (ODE) definiert.

AusdrSystem ist ein System rechter Seiten, welche das ODE-System definieren (entspricht der Ordnung abhängiger Variablen in *ListeAbhVar*).

AusdrListe ist eine Liste rechter Seiten, welche das ODE-System definieren (entspricht der Ordnung abhängiger Variablen in *ListeAbhVar*).

Var ist die unabhängige Variable.

ListeAbhVar ist eine Liste abhängiger Variablen.

{*Var0*, *VarMax*} ist eine Liste mit zwei Elementen, die die Funktion anweist, von *Var0* zu *VarMax* zu integrieren.

ListeAbhVar0 ist eine Liste von Anfangswerten für abhängige Variablen.

rk23(0.001·y·(100-y),t,y,{0,100},10,1)

0.	1.	2.	3.	4.
10.	10.9367	11.9493	13.042	14.2

Um das ganze Ergebnis zu sehen, drücken Sie und verwenden dann und , um den Cursor zu bewegen.

Dieselbe Gleichung mit *diftol* auf 1.E-6

rk23(0.001·y·(100-y),t,y,{0,100},10,1,1.E-6)

0.	1.	2.	3.	4.
10.	10.9367	11.9495	13.0423	14.2189

Vergleichen Sie das vorstehende Ergebnis mit der exakten CAS-Lösung, die Sie erhalten, wenn Sie deSolve() und seqGen() verwenden:

deSolve(y'=0.001·y·(100-y) and y(0)=10,t,y)
 $y=\frac{100 \cdot (1.10517)^t}{(1.10517)^t+9}$

seqGen($\frac{100 \cdot (1.10517)^t}{(1.10517)^t+9}$)
{10.,10.9367,11.9494,13.0423,14.2189,15.48}

Gleichungssystem:

$$\begin{cases} y1'=-y1+0.1 \cdot y1 \cdot y2 \\ y2'=3 \cdot y2-y1 \cdot y2 \end{cases}$$

mit $y1(0)=2$ und $y2(0)=5$

rk23($\begin{cases} y1'+0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2-y1 \cdot y2 \end{cases}$,t,{y1,y2},{0,5},{2,5},1)

0.	1.	2.	3.	4.
2.	1.94103	4.78694	3.25253	1.82848
5.	16.8311	12.3133	3.51112	6.27245

Wenn *VarSchritt* eine Zahl ungleich Null ergibt: Zeichen(*VarSchritt*) = Zeichen (*VarMax-Var0*) und Lösungen werden an *Var0+i*VarSchritt* für alle i=0,1,2,... zurückgegeben, sodass *Var0+i*VarSchritt* in [*var0*,*VarMax*] ist (möglicherweise gibt es keinen Lösungswert an *VarMax*).

Wenn *VarSchritt* Null ergibt, werden Lösungen an den „Runge-Kutta“ *Var*-Werten zurückgegeben.

diftol ist die Fehlertoleranz (standardmäßig 0.001).

root() (Wurzel)

Katalog > 

root(Ausdr) \Rightarrow root

$$\sqrt[3]{8} = 2$$

root(Ausdr1, Ausdr2) \Rightarrow Wurzel

$$\frac{3}{\sqrt[3]{3}} = \frac{1}{3}$$

root(Ausdr) gibt die Quadratwurzel von Ausdr zurück.

$$\begin{array}{r} & 3^3 \\ \hline 3\overline{)1} & 144225 \end{array}$$

root(Ausdr1, Ausdr2) gibt die *Ausdr2*. Wurzel von *Ausdr1* zurück. *Ausdr1* kann eine reelle oder eine komplexe Gleitkommakonstante, eine ganze Zahl oder eine komplexe rationale Konstante oder ein allgemeiner symbolischer Ausdruck sein.

Hinweis: Siehe auch Vorlage n-te Wurzel, Seite 6.

rotate() (Rotieren)

Katalog > 

rotate(Ganzzahl1[,#Rotationen])
⇒Ganzzahl

Im Bin-Modus:

Rotiert die Bits in einer binären ganzen Zahl. *Ganzzahl1* kann mit jeder Basis eingegeben werden und wird automatisch in eine 64-Bit-Dualform konvertiert. Ist der Absolutwert von *Ganzzahl1* für diese Form zu groß, wird eine symmetrische Modulo-Operation ausgeführt, um sie in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter [Base2](#).

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

Seite 22.

Ist *#Rotationen* positiv, erfolgt eine Rotation nach links. Ist *#Rotationen* negativ, erfolgt eine Rotation nach rechts. Vorgabe ist -1 (ein Bit nach rechts rotieren).

Beispielsweise in einer Rechtsrotation:

Jedes Bit rotiert nach rechts.

0b000000000000001111010110000110101

Bit ganz rechts rotiert nach ganz links.

Es ergibt sich:

0b10000000000000111101011000011010

Das Ergebnis wird gemäß dem jeweiligen Basis-Modus angezeigt.

rotate(Liste1[,#Rotationen])⇒Liste

Gibt eine um *#Rotationen* Elemente nach rechts oder links rotierte Kopie von *Liste1* zurück. Verändert *Liste1* nicht.

Ist *#Rotationen* positiv, erfolgt eine Rotation nach links. Ist *#Rotationen* negativ, erfolgt eine Rotation nach rechts. Vorgabe ist -1 (ein Element nach rechts rotieren).

rotate(String1[,#Rotationen])⇒String

Gibt eine um *#Rotationen* Zeichen nach rechts oder links rotierte Kopie von *String1* zurück. Verändert *String1* nicht.

Ist *#Rotationen* positiv, erfolgt eine Rotation nach links. Ist *#Rotationen* negativ, erfolgt eine Rotation nach rechts. Vorgabe ist -1 (ein Zeichen nach rechts rotieren).

Im Hex-Modus:

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h8000000000000001E3
rotate(0h78E,2)	0h1E38

Wichtig: Geben Sie eine Dual- oder Hexadezimalzahl stets mit dem Präfix 0b bzw. 0h ein (Null, nicht der Buchstabe O).

Im Dec-Modus:

rotate({1,2,3,4})	{4,1,2,3}
rotate({1,2,3,4},-2)	{3,4,1,2}
rotate({1,2,3,4},1)	{2,3,4,1}

rotate("abcd")	"dabc"
rotate("abcd",-2)	"cdab"
rotate("abcd",1)	"bceda"

round(Ausdr1[, Stellen])⇒Ausdruck

round(1.234567,3) 1.235

Gibt das Argument gerundet auf die angegebene Anzahl von Stellen nach dem Dezimaltrennzeichen zurück.

round() (Runden)

Katalog >

Stellen muss eine ganze Zahl im Bereich 0–12 sein. Wird *Stellen* weggelassen, wird das Argument auf 12 signifikante Stellen gerundet.

Hinweis: Die Anzeige des Ergebnisses kann von der Einstellung “Angezeigte Ziffern” beeinflusst werden.

round(Liste1[, Stellen])⇒Liste

Gibt eine Liste von Elementen zurück, die auf die angegebene Stellenzahl gerundet wurden.

round(Matrix1[, Stellen])⇒Matrix

Gibt eine Matrix von Elementen zurück, die auf die angegebene Stellenzahl gerundet wurden.

$$\text{round}(\{\pi, \sqrt{2}, \ln(2)\}, 4) \\ \{3.1416, 1.4142, 0.6931\}$$

$$\text{round}\left(\begin{bmatrix} \ln(5) & \ln(3) \\ \pi & e^1 \end{bmatrix}, 1\right) \\ \begin{bmatrix} 1.6 & 1.1 \\ 3.1 & 2.7 \end{bmatrix}$$

rowAdd() (Zeilenaddition)

Katalog >

rowAdd(Matrix1, rIndex1, rIndex2)
⇒Matrix

Gibt eine Kopie von *Matrix1* zurück, in der die Zeile *rIndex2* durch die Summe der Zeilen *rIndex1* und *rIndex2* ersetzt ist.

$$\text{rowAdd}\left(\begin{bmatrix} 3 & 4 \\ -3 & -2 \end{bmatrix}, 1, 2\right) \\ \begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix}$$

$$\text{rowAdd}\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}, 1, 2\right) \\ \begin{bmatrix} a & b \\ a+c & b+d \end{bmatrix}$$

rowDim() (Zeilendimension)

Katalog >

rowDim(Matrix)⇒Ausdruck

Gibt die Anzahl der Zeilen von *Matrix* zurück.

Hinweis: Siehe auch **colDim()**, Seite 32.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1 \\ \text{rowDim}(m1) \\ 3$$

rowNorm() (Zeilennorm)

Katalog >

rowNorm(Matrix)⇒Ausdruck

Gibt das Maximum der Summen der Absolutwerte der Elemente der Zeilen von *Matrix* zurück.

Hinweis: Alle Matrixelemente müssen zu Zahlen vereinfachbar sein. Siehe auch

$$\text{rowNorm}\left(\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}\right) \\ 25$$

colNorm(), Seite 32.

rowSwap() (Zeilentausch)

rowSwap(*Matrix1*, *rIndex1*, *rIndex2*)
⇒*Matrix*

Gibt *Matrix1* zurück, in der die Zeilen *rIndex1* und *rIndex2* vertauscht sind.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow mat$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
rowSwap(<i>mat</i> , 1, 3)	$\begin{bmatrix} 5 & 6 \\ 3 & 4 \\ 1 & 2 \end{bmatrix}$

rref() (Reduzierte Diagonalform)

rref(*Matrix1*[, *Tol*])⇒*Matrix*

Gibt die reduzierte Diagonalform von *Matrix1* zurück.

$\text{rref}\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 0 & 0 & \frac{66}{71} \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$
--	---

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

rref($\begin{bmatrix} a & b \\ c & d \end{bmatrix}$)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
--	--

- Wenn Sie **ctrl enter** verwenden oder den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
 $5E-14 \cdot \max(\dim(\text{Matrix1})) \cdot \text{rowNorm}(\text{Matrix1})$

Hinweis: Siehe auch **ref()**, Seite 154.

sec() (Sekans)
 Taste
sec(Ausdr1) ⇒ Ausdruck**sec(Liste1) ⇒ Liste**

Gibt den Sekans von *Ausdr1* oder eine Liste der Sekans aller Elemente in *Liste1* zurück.

Hinweis: Der als Argument angegebene Winkel wird gemäß der aktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, ° oder ‰ benutzen, um den Winkelmodus vorübergend aufzuheben.

Im Grad-Modus:

$$\begin{array}{l} \sec(45) \\ \sec(\{1,2,3,4\}) \quad \left\{ \frac{1}{\cos(1)}, 1.00081, \frac{1}{\cos(4)} \right\} \end{array} \quad \sqrt{2}$$

sec⁻¹() (Arkussekans)
 Taste
sec⁻¹(Ausdr1) ⇒ Ausdruck**sec⁻¹(Liste1) ⇒ Liste**

Gibt entweder den Winkel, dessen Sekans *Ausdr1* entspricht, oder eine Liste der inversen Sekans aller Elemente in *Liste1* zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arcsec** (...) eintippen.

Im Grad-Modus:

$$\sec^{-1}(1) \quad 0$$

Im Neugrad-Modus:

$$\sec^{-1}(\sqrt{2}) \quad 50$$

Im Bogenmaß-Modus:

$$\sec^{-1}(\{1,2,5\}) \quad \left\{ 0, \frac{\pi}{3}, \cos^{-1}\left(\frac{1}{5}\right) \right\}$$

sech() (Sekans hyperbolicus)**Katalog > ****sech(Ausdr1) ⇒ Ausdruck**

$$\text{sech}(3) \quad \frac{1}{\cosh(3)}$$

sech(Liste1) ⇒ Liste

$$\text{sech}(\{1,2,3,4\}) \quad \left\{ \frac{1}{\cosh(1)}, 0.198522, \frac{1}{\cosh(4)} \right\}$$

Gibt den hyperbolischen Sekans von *Ausdr1* oder eine Liste der hyperbolischen Sekans der Elemente in *Liste1* zurück.

sech⁻¹() (Arkussekans hyperbolicus)

Katalog >

sech⁻¹(Ausdr1) ⇒ Ausdruck

sech⁻¹ (Liste1) ⇒ Liste

Gibt den inversen hyperbolischen Sekans von *Ausdr1* oder eine Liste der inversen hyperbolischen Sekans aller Elemente in *Liste1* zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arcsech (...)** eintippen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus “kartesisch”:

$\text{sech}^{-1}(1)$	0
$\text{sech}^{-1}\{1, -2, 2, 1\}$	$\left\{0, \frac{2\pi}{3}, i, 8 \cdot 10^{-15} + 1.07448 \cdot i\right\}$

Send

Katalog >

Send exprOrString1 [, exprOrString2] ...

Sendet ein oder mehrere Argumente *AusdruckOderString* an ein Gerät, das an den USB-Anschluss angeschlossen ist.

Das angeschlossene USB-Gerät muss für die Kommunikation mit TI-Nspire™ Handhelds oder der Software konfiguriert sein.

Die Argumente werden hintereinander gesendet.

Hinweis: Sie können den Befehl **Send** in einem benutzerdefinierten Programm, aber nicht in einer Funktion verwenden.

Hinweis: Siehe auch **Get**, page 85 und **GetStr**, page 89.

Beispiel: Dies schaltet eine grüne LED 0,5 Sekunden lang ein und anschließend aus.

Send “SET GREEN 1 ON”

Wait 0.5

Send “SET GREEN 1 OFF”

seq() (Folge)

Katalog >

seq(Ausdr, Var, Von, Bis[, Schritt])⇒Liste

Erhöht Var in durch Schritt festgelegten Stufen von Von bis Bis, wertet Ausdr aus und gibt die Ergebnisse als Liste zurück. Der ursprüngliche Inhalt von Var ist nach Beendigung von **seq()** weiterhin vorhanden.

Der Vorgabewert für *Schritt* ist 1.

$\text{seq}\left(n^2, n, 1, 6\right)$	{1, 4, 9, 16, 25, 36}
$\text{seq}\left(\frac{1}{n}, n, 1, 10, 2\right)$	$\left\{1, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}\right\}$
$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)$	$\frac{1968329}{1270080}$

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie **ctrl enter**.

Windows®: Drücken Sie Strg+Eingabetaste.

Macintosh®: Drücken ⌘+Eingabetaste.

iPad®: Halten Sie die Eingabetaste gedrückt und wählen Sie aus.

$$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right) \quad 1.54977$$

seqGen()

seqGen(Ausdr, Var, abhVar, {Var0, VarMax}[], ListeAnfTerme [, VarSchritt [, ObergrWert]]]) \Rightarrow Liste

Generiert eine Term-Liste für die Folge $abhVar(Var)=Ausdr$ wie folgt: Erhöht die unabhängige Variable Var von $Var0$ bis $VarMax$ um $VarSchritt$, wertet $abhVar(Var)$ für die entsprechenden Werte von Var mithilfe der Formel $Ausdr$ und der $ListeAnfTerme$ aus und gibt die Ergebnisse als Liste zurück.

seqGen(SystemListeOderAusdr, Var, ListeAbhVar, {Var0, VarMax} [, MatrixAnfTerme [, VarSchritt [, ObergrWert]]]) \Rightarrow Matrix

Generiert eine Term-Matrix für ein System (oder eine Liste) von Folgen $ListeAbhVar(Var)=SystemListeOderAusdr$ wie folgt: Erhöht die unabhängige Variable Var von $Var0$ bis $VarMax$ um $VarSchritt$, wertet $ListeAbhVar(Var)$ für die entsprechenden Werte von Var mithilfe der Formel $SystemListeOderAusdr$ und der $MatrixAnfTerme$ aus und gibt die Ergebnisse als Matrix zurück.

Der ursprüngliche Inhalt von Var ist nach Beendigung von **seqGen()** weiterhin vorhanden.

Der Standardwert für $VarSchritt$ ist 1.

Generieren Sie die ersten 5 Terme der Folge $u(n)=u(n-1)^2/2$ mit $u(1)=2$ und $VarSchritt=1$.

$$\text{seqGen}\left(\frac{(u(n-1))^2}{n}, n, u, \{1, 5\}, \{2\}\right)$$

$$\left\{2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\}$$

Beispiel mit $Var0=2$:

$$\text{seqGen}\left(\frac{u(n-1)+1}{n}, n, u, \{2, 5\}, \{3\}\right)$$

$$\left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\}$$

Beispiel, in dem der Anfangsterm symbolisch ist:

$$\text{seqGen}(u(n-1)+2, n, u, \{1, 5\}, \{a\})$$

$$\{a, a+2, a+4, a+6, a+8\}$$

System zweiter Folgen:

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u2(n-1)}{2} + u1(n-1)\right\}, n, \{u1, u2\}, \{1, 5\}, \begin{bmatrix} - \\ 2 \end{bmatrix}\right)$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

Hinweis: Die Lücke () in der oben

aufgeführten Anfangsterm-Matrix zeigt an, dass der Anfangsterm für $u1(n)$ mit der expliziten Folge-Formel $u1(n)=1/n$ berechnet wird.

seqn(Ausdr{u, n [, ListeAnfTerme[, nMax [, ObergrWert]}])⇒Liste

Generiert eine Term-Liste für eine Folge $u(n)=\text{Ausdr}(u, n)$ wie folgt: Erhöht n von 1 bis $nMax$ um 1, wertet $u(n)$ für die entsprechenden Werte von n mithilfe der Formel $\text{Ausdr}(u, n)$ und *ListeAnfTerme* aus und gibt die Ergebnisse als Liste zurück.

seqn(Ausdr{n [, nMax [, ObergrWert]})
⇒Liste

Generiert eine Term-Liste für eine nichtrekursive Folge $u(n)=\text{Ausdr}(n)$ wie folgt: Erhöht n von 1 bis $nMax$ um 1, wertet $u(n)$ für die entsprechenden Werte von n mithilfe der Formel $\text{Ausdr}(n)$ aus und gibt die Ergebnisse als Liste zurück.

Wenn $nMax$ fehlt, wird $nMax$ auf 2500 gesetzt

Wenn $nMax=0$, wird $nMax$ auf 2500 gesetzt

Hinweis: seqn() gibt seqGen() mit $n0=1$ und $nSchritt =1$ an

Generieren Sie die ersten 6 Terme der Folge $u(n) = u(n-1)/2$ mit $u(1)=2$.

$$\text{seqn}\left(\frac{u(n-1)}{n}, \{2\}, 6\right) \\ \left\{ 2, 1, \frac{1}{3}, \frac{1}{12}, \frac{1}{60}, \frac{1}{360} \right\}$$

$$\text{seqn}\left(\frac{1}{n^2}, 6\right) \\ \left\{ 1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36} \right\}$$

series(Expr1, Var, Order [, Point])
⇒Ausdruck

$$\text{series}\left(\frac{1-\cos(x-1)}{(x-1)^2}, x, 4, 1\right) \\ \frac{1}{2} - \frac{(x-1)^2}{24} + \frac{(x-1)^4}{720}$$

series(Expr1, Var, Order [, Point]) |
Var>Point⇒Ausdruck

$$\text{series}\left(\frac{-1}{e^{z_-}}, z, -1\right) \\ z_- - 1$$

series(Expr1, Var, Order [, Point]) |
Var<Point⇒Ausdruck

$$\text{series}\left(\left(1 + \frac{1}{n}\right)^n, n, 2, \infty\right) \\ e - \frac{e}{2 \cdot n} + \frac{11 \cdot e}{24 \cdot n^2}$$

series()

Gibt eine verallgemeinerte endliche Potenzreihe von *Expr1* entwickelt um *Point* bis Grad *Order* zurück. *Order* kann jede beliebige rationale Zahl sein. Die resultierenden Potenzen von (*Var* – *Point*) können negative und/oder Bruchexponenten beinhalten. Die Koeffizienten dieser Potenzen können Logarithmen von (*Var* – *Point*) und andere Funktionen von *Var* beinhalten, die von allen Potenzen von (*Var* – *Point*) mit demselben Exponentenzeichen dominieren.

Point ist vorgegeben als 0. *Point* kann ∞ oder $-\infty$ sein; in diesen Fällen ist die Entwicklung durch Grad *Order* in $1/(Var - Point)$.

series(...) gibt “**series(...)**” zurück, wenn sie keine Darstellung bestimmen kann wie für wesentliche Singularitäten wie z.B. $\sin(1/z)$ bei $z=0$, $e^{-1/z}$ bei $z=0$ oder e^z bei $z = \infty$ oder $-\infty$.

Wenn die Reihe oder eine ihrer Ableitungen eine Sprungstelle bei *Point* hat, enthält das Ergebnis wahrscheinlich Unterausdrücke der Form $\text{sign}(\dots)$ oder $\text{abs}(\dots)$ für eine reelle Expansionsvariable oder $(-1)^{\text{floor}(\dots)}$ für eine komplexe

Expansionsvariable, die mit “_” endet.

Wenn Sie die Folge nur für Werte auf einer Seite von *Point* verwenden möchten, hängen Sie je nach Bedarf “| *Var* > *Point*”, “| *Var* < *Point*”, “| *Var* ≥ *Point*” oder “*Var* ≤ *Point*” an, um ein einfacheres Ergebnis zu erhalten.

series() kann symbolische Approximationen für unbestimmte Integrale und bestimmte Integrale bereitstellen, für die anders keine symbolischen Lösungen erreicht werden können.

series() wird über Listen und Matrizen mit erstem Argument verteilt.

series() ist eine verallgemeinerte Version von **taylor()**.

series($\tan^{-1}\left(\frac{1}{x}\right), x, 5$) $x > 0$	$\frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5}$
series($\int \frac{\sin(x)}{x} dx, x, 6$)	$x - \frac{x^3}{18} + \frac{x^5}{600}$
series($\int_0^x \sin(x \cdot \sin(t)) dt, x, 7$)	$\frac{x^3}{2} - \frac{x^5}{24} - \frac{29 \cdot x^7}{720}$

series($(1+e^x)^2, x, 2, 1$) $(e+1)^2 + 2 \cdot e \cdot (e+1) \cdot (x-1) + e \cdot (2 \cdot e+1) \cdot (x-1)^2$

Wie im letzten nebenstehenden Beispiel demonstriert, können die Anzeigeroutinen hinter dem von `series(...)` erzeugten Ergebnis Terme so umstellen, dass der dominante Term nicht ganz links steht.

Hinweis: Siehe auch **dominantTerm()**, Seite 63.

setMode

setMode(*ModusNameGanzzahl*,
*GanzzahlFestlegen***)** \Rightarrow *Ganzzahl*

setMode(Liste) \Rightarrow *Liste mit ganzen Zahlen*

Nur gültig innerhalb einer Funktion oder eines Programms.

setMode(*ModusNameGanzzahl*,
*GanzzahlFestlegen***)** schaltet den Modus *ModusNameGanzzahl* vorübergehend in *GanzzahlFestlegen* und gibt eine ganze Zahl entsprechend der ursprünglichen Einstellung dieses Modus zurück. Die Änderung ist auf die Dauer der Ausführung des Programms / der Funktion begrenzt.

ModusNameGanzzahl gibt an, welchen Modus Sie einstellen möchten. Hierbei muss es sich um eine der Modus-Ganzzahlen aus der nachstehenden Tabelle handeln.

GanzzahlFestlegen gibt die neue Einstellung für den Modus an. Für den Modus, den Sie festlegen, müssen Sie eine der in der nachstehenden Tabelle aufgeführten Einstellungs-Ganzzahlen verwenden.

setMode(Liste) dient zum Ändern mehrerer Einstellungen. *Liste* enthält Paare von Modus- und Einstellungs-Ganzzahlen. **setMode(Liste)** gibt eine ähnliche Liste zurück, deren Ganzzahlen-Paare die ursprünglichen Modi und Einstellungen angeben.

Zeigen Sie den Näherungswert von π an, indem Sie die Standardeinstellung für Zahlen anzeigen (Display Digits) verwenden, und zeigen Sie dann π mit einer Einstellung von Fix 2 an. Kontrollieren Sie, dass der Standardwert nach Beendigung des Programms wiederhergestellt wird.

Define <i>prog1()</i> =Prgm	<i>Done</i>
Disp approx(π)	
setMode(1,16)	
Disp approx(π)	
EndPrgm	
<hr/>	
<i>prog1()</i>	
	3.14159
	3.14
<hr/>	
	<i>Done</i>

Wenn Sie alle Moduseinstellungen mit **getMode(0) → var** gespeichert haben, können Sie **setMode(var)** verwenden, um diese Einstellungen wiederherzustellen, bis die Funktion oder das Programm beendet wird. Siehe **getMode()**, Seite 88.

Hinweis: Die aktuellen Moduseinstellungen werden an aufgerufene Subroutinen weitergegeben. Wenn eine der Subroutinen eine Moduseinstellung ändert, geht diese Modusänderung verloren, wenn die Steuerung zur aufrufenden Routine zurückkehrt.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Modus Name	Modus Ganzzahl	Einstellen von Ganzzahlen
Angezeigte Ziffern	1	1=Fließ, 2=Fließ 1, 3=Fließ 2, 4=Fließ 3, 5=Fließ 4, 6=Fließ 5, 7=Fließ 6, 8=Fließ 7, 9=Fließ 8, 10=Fließ 9, 11=Fließ 10, 12=Fließ 11, 13=Fließ 12, 14=Fix 0, 15=Fix 1, 16=Fix 2, 17=Fix 3, 18=Fix 4, 19=Fix 5, 20=Fix 6, 21=Fix 7, 22=Fix 8, 23=Fix 9, 24=Fix 10, 25=Fix 11, 26=Fix 12
Winkel	2	1=Bogenmaß, 2=Grad, 3=Neugrad
Exponentialformat	3	1=Normal, 2=Wissenschaftlich, 3=Technisch
Reell oder komplex	4	1=Reell, 2=Kartesisch, 3=Polar
Auto oder Approx.	5	1=Auto, 2=Approximiert, 3=Exakt
Vektorformat	6	1=Kartesisch, 2=Zylindrisch, 3=Sphärisch
Basis	7	1=Dezimal, 2=Hex, 3=Binär
Einheitensystem	8	1=SI, 2=Eng/US

shift() (Verschieben)

shift(Ganzzahl1[,#Verschiebungen]) Im Bin-Modus:
→Ganzzahl

Verschiebt die Bits in einer binären ganzen Zahl. *Ganzzahl1* kann mit jeder Basis eingegeben werden und wird automatisch in eine 64-Bit-Dualform konvertiert. Ist der Absolutwert von *Ganzzahl1* für diese Form zu groß, wird eine symmetrische Modulo-Operation ausgeführt, um sie in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie unter **►Base2**, Seite 22.

Ist **#Verschiebungen** positiv, erfolgt die Verschiebung nach links. Ist **#Verschiebungen** negativ, erfolgt die Verschiebung nach rechts. Vorgabe ist -1 (ein Bit nach rechts verschieben).

In einer Rechtsverschiebung wird das ganz rechts stehende Bit abgeschnitten und als ganz links stehendes Bit eine 0 oder 1 eingesetzt. Bei einer Linksverschiebung wird das Bit ganz links abgeschnitten und 0 als letztes Bit rechts eingesetzt.

Beispielsweise in einer Rechtsverschiebung:

Alle Bits werden nach rechts verschoben.

0b000000000000000111101011000011010

Setzt 0 ein, wenn Bit ganz links 0 ist, und 1, wenn Bit ganz links 1 ist.

Es ergibt sich:

0b000000000000000111101011000011010

Das Ergebnis wird gemäß dem jeweiligen Basis-Modus angezeigt. Führende Nullen werden nicht angezeigt.

shift(Liste1 [,#Verschiebungen])⇒Liste

Gibt eine um **#Verschiebungen** Elemente nach rechts oder links verschobene Kopie von *Liste1* zurück. Verändert *Liste1* nicht.

Ist **#Verschiebungen** positiv, erfolgt die Verschiebung nach links. Ist **#Verschiebungen** negativ, erfolgt die Verschiebung nach rechts. Vorgabe ist -1

shift(0b1111010110000110101)	{ undef,1,2,3 }
shift(256,1)	0b1000000000

Im Hex-Modus:

shift(0h78E)	0h3C7
shift(0h78E,-2)	0h1E3
shift(0h78E,2)	0h1E38

Wichtig: Geben Sie eine Dual- oder Hexadezimalzahl stets mit dem Präfix 0b bzw. 0h ein (Null, nicht der Buchstabe O).

Im Dec-Modus:

shift({1,2,3,4})	{ undef,1,2,3 }
shift({1,2,3,4},-2)	{ undef,undef,1,2 }
shift({1,2,3,4},2)	{ 3,4,undef,undef }

shift() (Verschieben)

Katalog >

(ein Element nach rechts verschieben).

Dadurch eingeführte neue Elemente am Anfang bzw. am Ende von *Liste* werden auf "undef" gesetzt.

shift(String1 [,#Verschiebungen])⇒String

Gibt eine um #Verschiebungen Zeichen nach rechts oder links verschobene Kopie von *Liste1* zurück. Verändert *String1* nicht.

shift("abcd")	" abc"
shift("abcd",-2)	" ab"
shift("abcd",1)	"bcd "

Ist #Verschiebungen positiv, erfolgt die Verschiebung nach links. Ist #Verschiebungen negativ, erfolgt die Verschiebung nach rechts. Vorgabe ist -1 (ein Zeichen nach rechts verschieben).

Dadurch eingeführte neue Zeichen am Anfang bzw. am Ende von *String* werden auf ein Leerzeichen gesetzt.

sign() (Zeichen)

Katalog >

sign(Ausdr1)⇒Ausdruck

sign(-3.2)	-1.
------------	-----

sign(Liste1)⇒Liste

sign({2,3,4,-5})	{1,1,1,-1}
------------------	------------

sign(Matrix1)⇒Matrix

sign(1+ x)	1
-------------	---

Gibt für reelle und komplexe Ausdr1 Ausdr1/abs(Ausdr1) zurück, wenn Ausdr1 ≠ 0.

Bei Komplex-Formatmodus Reell:

Gibt 1 zurück, wenn Ausdr1 positiv ist.

sign([-3 0 3])	[-1 ±1 1]
----------------	-----------

Gibt -1 zurück, wenn Ausdr1 negativ ist.

Gibt 0 zurück, wenn Ausdr1 gleich null ist.

sign(0) gibt ±1 zurück, wenn als Komplex-Formatmodus Reell eingestellt ist; anderenfalls gibt es sich selbst zurück.

sign(0) stellt im komplexen Bereich den Einheitskreis dar.

Gibt für jedes Element einer Liste bzw. Matrix das Vorzeichen zurück.

simult() (Gleichungssystem)

Katalog >

simult(KoeffMatrix, KonstVektor[, Tol])

Auflösen nach x und y:

\Rightarrow Matrix

Ergibt einen Spaltenvektor, der die Lösungen für ein lineares Gleichungssystem enthält.

Hinweis: Siehe auch linSolve(), Seite 108.

KoeffMatrix muss eine quadratische Matrix sein, die die Koeffizienten der Gleichung enthält.

KonstVektor muss die gleiche Zeilenanzahl (gleiche Dimension) besitzen wie *KoeffMatrix* und die Konstanten enthalten.

Sie haben die Option, dass jedes Matrixelement als Null behandelt wird, wenn dessen absoluter Wert geringer als *Tol* ist. Diese Toleranz wird nur dann verwendet, wenn die Matrix Fließkommaelemente aufweist und keinerlei symbolische Variablen ohne zugewiesene Werte enthält. Andernfalls wird *Tol* ignoriert.

- Wenn Sie den Modus **Auto oder Näherung** auf Approximiert einstellen, werden Berechnungen in Fließkomma-Arithmetik durchgeführt.
- Wird *Tol* weggelassen oder nicht verwendet, so wird die Standardtoleranz folgendermaßen berechnet:
 $5E-14 \cdot \max(\dim(KoeffMatrix))$
 $\cdot \text{rowNorm}(KoeffMatrix)$

simult(KoeffMatrix, KonstMatrix[, Tol])

 \Rightarrow Matrix

Löst mehrere lineare Gleichungssysteme, die alle dieselben Gleichungskoeffizienten, aber unterschiedliche Konstanten haben.

Jede Spalte in *KonstMatrix* muss die Konstanten für ein Gleichungssystem enthalten. Jede Spalte in der sich ergebenden Matrix enthält die Lösung für das entsprechende System.

$$x + 2y = 1$$

$$3x + 4y = -1$$

$$\begin{array}{c} \text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) \\ \hline \begin{bmatrix} -3 \\ 2 \end{bmatrix} \end{array}$$

Die Lösung ist $x=-3$ und $y=2$.

Auflösen:

$$ax + by = 1$$

$$cx + dy = 2$$

$$\begin{array}{c} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow \text{matx1} \\ \text{simult}\left(\text{matx1}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \\ \hline \begin{bmatrix} \frac{-2 \cdot b - d}{a \cdot d - b \cdot c} \\ \frac{2 \cdot a - c}{a \cdot d - b \cdot c} \end{bmatrix} \end{array}$$

Auflösen:

$$x + 2y = 1$$

$$3x + 4y = -1$$

$$x + 2y = 2$$

$$3x + 4y = -3$$

simult() (Gleichungssystem)

Katalog >

$$\text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & -3 \end{bmatrix}\right) = \begin{bmatrix} -3 & 7 \\ 2 & 9/2 \end{bmatrix}$$

Für das erste System ist $x=-3$ und $y=2$. Für das zweite System ist $x=7$ und $y=9/2$.

►sin

Katalog >

Ausdr ►sin

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>sin eintippen.

Drückt Ausdr durch Sinus aus. Dies ist ein Anzeigewandlungsoperator. Er kann nur am Ende der Eingabezeile verwendet werden.

►sin reduziert alle Potenzen von $\cos(\dots)$ modulo $1-\sin(\dots)^2$, so dass alle verbleibenden Potenzen von $\sin(\dots)$ Exponenten im Bereich $(0, 2)$ haben. Deshalb enthält das Ergebnis dann und nur dann kein $\cos(\dots)$, wenn $\cos(\dots)$ im gegebenen Ausdruck nur bei geraden Potenzen auftritt.

Hinweis: Dieser Umrechnungsoperator wird im Winkelmodus Grad oder Neugrad (Gon) nicht unterstützt. Bevor Sie ihn verwenden, müssen Sie sicherstellen, dass der Winkelmodus auf Radian eingestellt ist und Ausdr keine expliziten Verweise auf Winkel in Grad oder Neugrad enthält.

sin() (Sinus)

Taste

$\sin(Ausdr1) \Rightarrow Ausdruck$

Im Grad-Modus:

$\sin(Liste1) \Rightarrow Liste$

$\sin(Ausdr1)$ gibt den Sinus des Arguments als Ausdruck zurück.

$\sin(Liste1)$ gibt eine Liste zurück, die für jedes Element von $Liste1$ den Sinus enthält.

sin() (Sinus)

trig Taste

Hinweis: Das Argument wird entsprechend dem aktuellen Winkelmodus als Winkel in Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, G oder r benutzen, um die Winkelmoduseinstellung temporär zu ändern.

$\sin\left(\frac{\pi}{4}\right)$	$\frac{\sqrt{2}}{2}$
$\sin(45)$	$\frac{\sqrt{2}}{2}$
$\sin(\{0,60,90\})$	$\left\{0, \frac{\sqrt{3}}{2}, 1\right\}$

Im Neugrad-Modus:

$\sin(50)$	$\frac{\sqrt{2}}{2}$
------------	----------------------

Im Bogenmaß-Modus:

$\sin\left(\frac{\pi}{4}\right)$	$\frac{\sqrt{2}}{2}$
$\sin(45^\circ)$	$\frac{\sqrt{2}}{2}$

sin(Quadratmatrix I)⇒Quadratmatrix

Gibt den Matrix-Sinus von *Quadratmatrix I* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Sinus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

$\sin\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.9424 & -0.04542 & -0.031999 \\ -0.045492 & 0.949254 & -0.020274 \\ -0.048739 & -0.00523 & 0.961051 \end{bmatrix}$
--	--

sin⁻¹() (Arkussinus)

trig Taste

sin⁻¹(Ausdr I)⇒Ausdruck

Im Grad-Modus:

$\sin^{-1}(1)$	90
----------------	----

sin⁻¹(Ausdr I) gibt den Winkel, dessen Sinus *Ausdr I* ist, als Ausdruck zurück.

Im Neugrad-Modus:

$\sin^{-1}(1)$	100
----------------	-----

sin⁻¹() (Arkussinus)

trig Taste

Sinus zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie `arcsin (...)` eintippen.

sin⁻¹(Quadratmatrix I)⇒Quadratmatrix

Gibt den inversen Matrix-Sinus von *Quadratmatrix I* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Sinus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

$$\sin^{-1}(\{0,0.2,0.5\}) \quad \{0,0.201358,0.523599\}$$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\begin{aligned}\sin^{-1}\left(\begin{bmatrix} 1 & 5 \\ 4 & 2 \end{bmatrix}\right) \\ \begin{bmatrix} -0.174533-0.12198 \cdot i & 1.74533-2.35591 \cdot i \\ 1.39626-1.88473 \cdot i & 0.174533-0.593162 \cdot i \end{bmatrix}\end{aligned}$$

sinh() (Sinus hyperbolicus)

Katalog >

sinh(Ausdr I)⇒Ausdruck

$$\sinh(1.2) \quad 1.50946$$

sinh(Liste I)⇒Liste

$$\sinh(\{0,1,2,3\}) \quad \{0,1.50946,10.0179\}$$

sinh (Ausdr I) gibt den Sinus hyperbolicus des Arguments als Ausdruck **zurück**.

sinh (Liste I) gibt in Form einer Liste für jedes Element aus *Liste I* den Sinus hyperbolicus zurück.

sinh(Quadratmatrix I)⇒Quadratmatrix

Gibt den Matrix-Sinus hyperbolicus von *Quadratmatrix I* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Sinus hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

$$\begin{aligned}\sinh\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) \\ \begin{bmatrix} 360.954 & 305.708 & 239.604 \\ 352.912 & 233.495 & 193.564 \\ 298.632 & 154.599 & 140.251 \end{bmatrix}\end{aligned}$$

sinh⁻¹() (Arkussinus hyperbolicus)

Katalog >

sinh⁻¹(Ausdr I)⇒Ausdruck

$\sinh^{-1}(0)$ 0

sinh⁻¹(Liste I)⇒Liste

$\sinh^{-1}(\{0,2,1,3\})$ {0,1.48748,sinh⁻¹(3)}

sinh⁻¹(Ausdr I) gibt den inversen Sinus hyperbolicus des Arguments als Ausdruck zurück.

sinh⁻¹(Liste I) gibt in Form einer Liste für jedes Element aus *Liste I* den inversen Sinus hyperbolicus zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arcsinh (...) eintippen.**

sinh⁻¹(Quadratmatrix I)⇒Quadratmatrix

Gibt den inversen Matrix-Sinus hyperbolicus von *Quadratmatrix I* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Sinus hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Im Bogenmaß-Modus:

$$\sinh^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} 0.041751 & 2.15557 & 1.1582 \\ 1.46382 & 0.926568 & 0.112557 \\ 2.75079 & -1.5283 & 0.57268 \end{pmatrix}$$

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

SinReg

Katalog >

SinReg X, Y [, [Iterationen],[Periode] [, Kategorie, Mit]]

Berechnet die sinusförmige Regression auf Listen *X* und *Y*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Iterationen ist ein Wert, der angibt, wie viele Lösungsversuche (1 bis 16) maximal unternommen werden. Bei Auslassung wird 8 verwendet. Größere Werte führen in der Regel zu höherer Genauigkeit, aber auch zu längeren Ausführungszeiten, und umgekehrt.

Periode gibt eine geschätzte Periode an. Bei Auslassung sollten die Werte in *X* sequentiell angeordnet und die Differenzen zwischen ihnen gleich sein. Wenn Sie *Periode* jedoch angeben, können die Differenzen zwischen den einzelnen x-Werten ungleich sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Die Ausgabe von **SinReg** erfolgt unabhängig von der Winkelmoduseinstellung immer im Bogenmaß (rad).

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabeveriable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot \sin(bx+c)+d$
stat.a, stat.b, stat.c, stat.d	Regressionskoeffizienten
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten <i>X-Liste</i> , die in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
stat.YReg	Liste der Datenpunkte in der modifizierten <i>Y-Liste</i> , die schließlich in der Regression mit den Beschränkungen für <i>Häuf</i> , <i>Kategorieliste</i> und <i>Mit-Kategorien verwendet wurde</i>
stat.FreqReg	Liste der Häufigkeiten für <i>stat.XReg</i> und <i>stat.YReg</i>

solve() (Löse)

solve(Gleichung, Var)⇒Boolescher Ausdruck

$$\begin{aligned} &\text{solve}(a \cdot x^2 + b \cdot x + c = 0, x) \\ &x = \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a} \text{ or } x = \frac{-\sqrt{b^2 - 4 \cdot a \cdot c} + b}{2 \cdot a} \end{aligned}$$

solve(Gleichung, Var=Schätzwert)
⇒Boolescher Ausdruck

solve(Ungleichung, Var)⇒Boolescher Ausdruck

Gibt mögliche reelle Lösungen einer Gleichung oder Ungleichung für *Var* zurück. Das Ziel ist, Kandidaten für alle Lösungen zu erhalten. Es kann jedoch Gleichungen oder Ungleichungen geben, für die es eine unendliche Anzahl von Lösungen gibt.

Für manche Wertekombinationen undefinierter Variablen kann es sein, dass mögliche Lösungen nicht reell und endlich sind.

Ist der Modus **Auto oder Näherung** auf Auto eingestellt, ist das Ziel die Ermittlung exakter kompakter Lösungen, wobei ergänzend eine iterative Suche mit Näherungslösungen benutzt wird, wenn exakte Lösungen sich als unpraktisch erweisen.

Da Quotienten standardmäßig mit dem größten gemeinsamen Teiler von Zähler und Nenner gekürzt werden, kann es sein, dass Lösungen nur in den Grenzwerten von einer oder beiden Seiten liegen.

Für Ungleichungen der Typen \geq , \leq , $<$ oder $>$ sind explizite Lösungen unwahrscheinlich, es sei denn, die Ungleichung ist linear und enthält nur *Var*.

Ist der Modus **Auto oder Näherung** auf Exakt eingestellt, werden nicht lösbar Teile als implizite Gleichung oder Ungleichung zurückgegeben.

Verwenden Sie den womit-Operator „|“ zur Beschränkung des Lösungsintervalls und/oder zur Einschränkung anderer Variablen, die in der Gleichung bzw. Ungleichung vorkommen. Wenn Sie eine Lösung in einem Intervall gefunden haben, können Sie die Ungleichungsoperatoren benutzen, um dieses Intervall aus nachfolgenden Suchläufen auszuschließen.

Wenn keine reellen Lösungen ermittelt werden können, wird „falsch“ zurückgegeben. „wahr“ wird zurückgegeben, wenn **solve()** feststellt,

Ans| $a=1$ and $b=1$ and $c=1$

$$x = \frac{-1 + \sqrt{3}}{2} \cdot i \text{ or } x = \frac{-1 - \sqrt{3}}{2} \cdot i$$

solve $((x-a) \cdot e^x = x \cdot (x-a), x)$

$$x=a \text{ or } x=0.567143$$

$$(x+1) \cdot \frac{x-1}{x-1} + x - 3$$

$$2 \cdot x - 2$$

solve $(5 \cdot x - 2 \geq 2 \cdot x, x)$

$$x \geq \frac{2}{3}$$

exact(solve $((x-a) \cdot e^x = x \cdot (x-a), x)$)

$$e^x + x = 0 \text{ or } x = a$$

Im Bogenmaß-Modus:

solve $\left(\tan(x) = \frac{1}{x}, x\right) | x > 0 \text{ and } x < 1$

$$x = 0.860334$$

solve $(x = x + 1, x)$

false

solve $(x = x, x)$

true

dass jeder endliche reelle Wert von *Var* die Gleichung bzw. Ungleichung erfüllt.

Da **solve()** stets ein Boolesches Ergebnis liefert, können Sie "and", "or" und "not" verwenden, um Ergebnisse von **solve()** miteinander oder mit anderen Booleschen Ausdrücken zu verknüpfen.

Lösungen können eine neue unbestimmte Konstante der Form *nj* enthalten, wobei *j* eine ganze Zahl im Intervall 1–255 ist. Eine solche Variable steht für eine beliebige ganze Zahl.

Im reellen Modus zeigen Bruchpotenzen mit ungeradem Nenner nur das reelle Intervall. Ansonsten zeigen zusammengesetzte Ausdrücke wie Bruchpotenzen, Logarithmen und inverse trigonometrische Funktionen nur das Hauptintervall. Demzufolge liefert **solve()** nur Lösungen, die diesem einen reellen oder Hauptintervall entsprechen.

Hinweis: Siehe auch **cSolve()**, **cZeros()**, **nSolve()** und **zeros()**.

**solve(Glch1 and Glch2 [and...],
VarOderSchätzwert1,
VarOderSchätzwert2 [, ...])**
⇒Boolescher Ausdruck

**solve(Gleichungssystem,
VarOderSchätzwert1,
VarOderSchätzwert2 [, ...])**
⇒Boolescher Ausdruck

**solve({Glch1, Glch2 [...]})
{VarOderSchätzwert1,
VarOderSchätzwert2 [, ...]})**
⇒Boolescher Ausdruck

Gibt mögliche reelle Lösungen eines algebraischen Gleichungssystems zurück, in dem jedes Argument *VarOderSchätzwert* eine Variable darstellt, nach der Sie die Gleichungen auflösen möchten.

$$2 \cdot x - 1 \leq 1 \text{ and solve}(x^2 = 9, x) \quad x \neq -3 \text{ and } x \leq 1$$

Im Bogenmaß-Modus:

$$\text{solve}(\sin(x) = 0, x) \quad x = n1 \cdot \pi$$

$\text{solve}\left(\frac{1}{x^3} = -1, x\right)$	$x = -1$
$\text{solve}(\sqrt{x} = 2, x)$	false
$\text{solve}(-\sqrt{x} = 2, x)$	$x = 4$

$$\text{solve}(y = x^2 - 2 \text{ and } x + 2 \cdot y = -1, \{x, y\}) \\ x = \frac{-3}{2} \text{ and } y = \frac{1}{4} \text{ or } x = 1 \text{ and } y = -1$$

Sie können die Gleichungen mit dem Operator **and** trennen oder mit einer Vorlage aus dem Katalog ein *Gleichungssystem* eingeben. Die Anzahl der *VarOderSchätzwert*-Argumente muss der Anzahl der Gleichungen entsprechen. Sie haben die Option, eine Ausgangsschätzung für eine Variable anzugeben. Jedes Argument *VarOderSchätzwert* muss die folgende Form haben:

Variable

- oder -

Variable = reelle oder nicht-reelle Zahl

Beispiel: x ist gültig und $x = 3$ ebenfalls.

Wenn alle Gleichungen Polynome sind und Sie KEINE Anfangsschätzwerte angeben, dann verwendet **solve()** das lexikalische Gröbner/Buchbergersche Eliminationsverfahren beim Versuch, alle reellen Lösungen zu bestimmen.

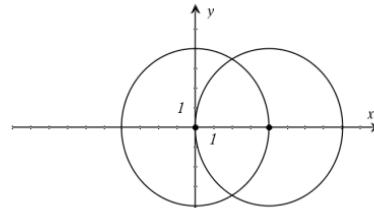
Betrachten wir z.B. einen Kreis mit dem Radius r und dem Ursprung als Mittelpunkt und einen weiteren Kreis mit Radius r und dem Schnittpunkt des ersten Kreises mit der positiven x -Achse als Mittelpunkt.

Verwenden Sie **solve()** zur Bestimmung der Schnittpunkte.

Wie in nebenstehendem Beispiel durch r demonstriert, können Gleichungssysteme zusätzliche Variablen ohne Wert aufweisen, die aber für numerische Werte stehen, welche später eingesetzt werden können.

Sie können auch (oder stattdessen) Lösungsvariablen angeben, die in den Gleichungen nicht erscheinen. Geben Sie zum Beispiel z als eine Lösungsvariable an, um das vorangehende Beispiel auf zwei parallele, sich schneidende Zylinder mit dem Radius r auszudehnen.

Die Zylinder-Lösungen verdeutlichen, dass



$$\begin{aligned} \text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y\}\right) \\ x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3} \cdot r}{2} \text{ or } x=\frac{r}{2} \text{ and } y=-\frac{\sqrt{3} \cdot r}{2} \end{aligned}$$

$$\begin{aligned} \text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y,z\}\right) \\ x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3} \cdot r}{2} \text{ and } z=c1 \text{ or } x=\frac{r}{2} \text{ and } y= \end{aligned}$$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

Lösungsfamilien "beliebige" Konstanten der Form ck , enthalten können, wobei k ein ganzzahliger Index im Bereich 1 bis 255 ist.

Bei Gleichungssystemen aus Polynomen kann die Berechnungsduer oder Speicherbelastung stark von der Reihenfolge abhängen, in welcher Sie die Lösungsvariablen angeben. Übersteigt Ihre erste Wahl die Speicherkapazität oder Ihre Geduld, versuchen Sie, die Variablen in der Gleichung und/oder *VarOderSchätzwert*-Liste umzuordnen.

Wenn Sie keine Schätzwerte angeben und eine Gleichung in einer Variablen nicht-polynomisch ist, aber alle Gleichungen in allen Lösungsvariablen linear sind, so verwendet **solve()** das Gaußsche Eliminationsverfahren beim Versuch, alle reellen Lösungen zu bestimmen.

Wenn ein System weder in all seinen Variablen polynomial noch in seinen Lösungsvariablen linear ist, dann bestimmt **solve()** mindestens eine Lösung anhand eines iterativen näherungsweisen Verfahrens. Hierzu muss die Anzahl der Lösungsvariablen gleich der Gleichungsanzahl sein, und alle anderen Variablen in den Gleichungen müssen zu Zahlen vereinfachbar sein.

Jede Lösungsvariable beginnt bei dem entsprechenden geschätzten Wert, falls vorhanden; ansonsten beginnt sie bei 0,0.

Suchen Sie anhand von Schätzwerten nach einzelnen zusätzlichen Lösungen. Für Konvergenz sollte eine Schätzung ziemlich nahe bei einer Lösung liegen.

$$\begin{aligned} \text{solve}\left(x+e^z \cdot y=1 \text{ and } x-y=\sin(z), \{x,y\}\right) \\ x = \frac{e^z \cdot \sin(z)+1}{e^z+1} \text{ and } y = \frac{-\sin(z)-1}{e^z+1} \end{aligned}$$

$$\begin{aligned} \text{solve}\left(e^z \cdot y=1 \text{ and } y=\sin(z), \{y,z\}\right) \\ y=2.812e-10 \text{ and } z=21.9911 \text{ or } y=0.001871 \end{aligned}$$

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

$$\begin{aligned} \text{solve}\left(e^z \cdot y=1 \text{ and } y=\sin(z), \{y,z=2 \cdot \pi\}\right) \\ y=0.001871 \text{ and } z=6.28131 \end{aligned}$$

SortA (In aufsteigender Reihenfolge sortieren)

Katalog >

SortA *Liste1[, Liste2] [, Liste3] ...*

SortA *Vektor1[, Vektor2] [, Vektor3] ...*

Sortiert die Elemente des ersten Arguments in aufsteigender Reihenfolge.

Bei Angabe von mehr als einem Argument werden die Elemente der zusätzlichen Argumente so sortiert, dass ihre neue Position mit der neuen Position der Elemente des ersten Arguments übereinstimmt.

Alle Argumente müssen Listen- oder Vektornamen sein. Alle Argumente müssen die gleiche Dimension besitzen.

Leere (ungültige) Elemente im ersten Argument werden nach unten verschoben. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA <i>list1</i>	<i>Done</i>
<i>list1</i>	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA <i>list2,list1</i>	<i>Done</i>
<i>list2</i>	$\{1,2,3,4\}$
<i>list1</i>	$\{4,3,2,1\}$

SortD (In absteigender Reihenfolge sortieren)

Katalog >

SortD *Liste1[, Liste2] [, Liste3] ...*

SortD *Vektor1[,Vektor2] [,Vektor3] ...*

Identisch mit **SortA** mit dem Unterschied, dass **SortD** die Elemente in absteigender Reihenfolge sortiert.

Leere (ungültige) Elemente im ersten Argument werden nach unten verschoben. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD <i>list1,list2</i>	<i>Done</i>
<i>list1</i>	$\{4,3,2,1\}$
<i>list2</i>	$\{3,4,1,2\}$

►Sphere (Kugelkoordinaten)

Katalog >

Vektor ►Sphere

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @>**Sphere** eintippen.

Zeigt den Zeilen- oder Spaltenvektor in Kugelkoordinaten [$\rho \angle\theta \angle\phi$] an.

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie **ctrl** **enter**.

Windows®: Drücken Sie **Strg+Eingabetaste**.

Macintosh®: Drücken **⌘+Eingabetaste**.

iPad®: Halten Sie die **Eingabetaste** gedrückt

►Sphere (Kugelkoordinaten)

Katalog >

Vektor muss die Dimension 3 besitzen und kann ein Zeilen- oder ein Spaltenvektor sein.

Hinweis: ►Sphere ist eine Anzeigeformatanweisung, keine Konvertierungsfunktion. Sie können sie nur am Ende einer Eingabezeile benutzen.

und wählen Sie aus.

[1 2 3] ►Sphere

[3.74166 ∠1.10715 ∠0.640522]

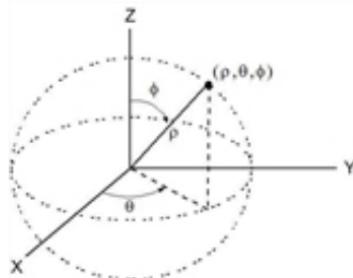
$\left(\begin{matrix} 2 & \angle \frac{\pi}{4} & 3 \end{matrix} \right)$ ►Sphere

[3.60555 ∠0.785398 ∠0.588003]

Drücken Sie .

$\left(\begin{matrix} 2 & \angle \frac{\pi}{4} & 3 \end{matrix} \right)$ ►Sphere

$\left[\sqrt{13} \angle \frac{\pi}{4} \angle \sin^{-1}\left(\frac{2 \cdot \sqrt{13}}{13}\right) \right]$



sqrt() (Quadratwurzel)

Katalog >

sqrt(Ausdr1)⇒Ausdruck

$\sqrt{4}$ 2

sqrt(Liste1)⇒Liste

$\sqrt{\{9,a,4\}}$ $\{3,\sqrt{a},2\}$

Gibt die Quadratwurzel des Arguments zurück.

Bei einer Liste wird die Quadratwurzel für jedes Element von *Liste1* zurückgegeben.

Hinweis: Siehe auch **Vorlage Quadratwurzel**, Seite 5.

stat.results

stat.results

Zeigt Ergebnisse einer statistischen Berechnung an.

Die Ergebnisse werden als Satz von Namen-Wert-Paaren angezeigt. Die angezeigten Namen hängen von der zuletzt ausgewerteten Statistikfunktion oder dem letzten Befehl ab.

Sie können einen Namen oder einen Wert kopieren und ihn an anderen Positionen einfügen.

Hinweis: Definieren Sie nach Möglichkeit keine Variablen, die dieselben Namen haben wie die für die statistische Analyse verwendeten Variablen. In einigen Fällen könnte ein Fehler auftreten. Namen von Variablen, die für die statistische Analyse verwendet werden, sind in der Tabelle unten aufgelistet.

Katalog >

xlist:= { 1,2,3,4,5 } { 1,2,3,4,5 }

ylist:= { 4,8,11,14,17 } { 4,8,11,14,17 }

LinRegMx xlist, ylist, 1: stat.results

"Title"	"Linear Regression (mx+b)"
"RegEqn"	"m*x+b"
"m"	3.2
"b"	1.2
"r^2"	0.996109
"r"	0.998053
"Resid"	"{...}"

stat.values	"Linear Regression (mx+b)"
	"m*x+b"
	3.2
	1.2
	0.996109
	0.998053
	"{-0.4,0.4,0.2,0.,-0.2}"

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR ²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r ²	stat.SSY
stat.b1	stat.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSIInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.ox	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.oy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.ox1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.ox2	stat.UpperVal
stat.b8	stat.F	stat.n	stat.Σx	stat.Ȑ
stat.b9	stat.FBlock	Stat.ŷ	stat.Σx ²	stat.Ȑ1
stat.b10	stat.Fcol	stat.ŷ1	stat.Σxy	stat.Ȑ2
stat.bList	stat.FInteract	stat.ŷ2	stat.Σy	stat.ȐDiff
stat.χ ²	stat.FreqReg	stat.ŷDiff	stat.Σy ²	stat.ȐList
stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.CompList	stat.LowerVal	stat.PValCol	stat.SEPred	stat.Ȑ

stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat.y
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat.yList
stat.CUpper	stat.MaxY	stat.Q1X	stat.sp	stat.YReg
stat.CUpperList	stat.ME	stat.Q1Y	stat.SS	
stat.d	stat.MedianX			

Hinweis: Immer, wenn die Applikation 'Lists & Spreadsheet' statistische Ergebnisse berechnet, kopiert sie die Gruppenvariablen "stat." in eine "stat#"-Gruppe, wobei # eine automatisch inkrementierte Zahl ist. Damit können Sie vorherige Ergebnisse beibehalten, während mehrere Berechnungen ausgeführt werden.

stat.values

Katalog >

stat.values

Siehe stat.results.

Zeigt eine Matrix der Werte an, die für die zuletzt ausgewertete Statistikfunktion oder den letzten Befehl berechnet wurden.

Im Gegensatz zu **stat.results** lässt **stat.values** die den Werten zugeordneten Namen aus.

Sie können einen Wert kopieren und ihn an anderen Positionen einfügen.

stDevPop() (Populations-Standardabweichung)

Katalog >

stDevPop(Liste[, Häufigkeitsliste])
⇒ Ausdruck

Im Bogenmaß- und automatischen Modus:

$$\text{stDevPop}(\{a,b,c\}) = \sqrt{\frac{2(a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2)}{3}}$$

$$\text{stDevPop}(\{1,2,5,-6,3,-2\}) = \sqrt{\frac{465}{6}}$$

$$\text{stDevPop}(\{1.3,2.5, 6.4\}, \{3,2,5\}) = 4.11107$$

Ergibt die Populations-Standardabweichung der Elemente in *Liste*.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Liste* muss mindestens zwei Elemente haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

stDevPop() (Populations-Standardabweichung)

Katalog >

stDevPop(*Matrix I*[, Häufigkeitsmatrix])
⇒Matrix

Ergibt einen Zeilenvektor der Populations-Standardabweichungen der Spalten in *Matrix I*.

Jedes Häufigkeitsmatrix-Element gewichtet die Elemente von *Matrix I* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Matrix I* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

$$\begin{array}{c} \text{stDevPop} \begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{pmatrix} \begin{pmatrix} 4\sqrt{6} & \sqrt{78} & 2\sqrt{6} \\ 3 & 3 & 3 \end{pmatrix} \\ \hline \text{stDevPop} \begin{pmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{pmatrix} \\ \hline [2.52608 \quad 5.21506] \end{array}$$

stDevSamp() (Stichproben-Standardabweichung)

Katalog >

stDevSamp(*Liste*[, Häufigkeitsliste])
⇒Ausdruck

Ergibt die Stichproben-Standardabweichung der Elemente in *Liste*.

Jedes Häufigkeitsliste-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Liste* muss mindestens zwei Elemente haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

$$\begin{array}{c} \text{stDevSamp}\{\{a,b,c\}\} \\ \frac{\sqrt{3 \cdot (a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2)}}{3} \\ \hline \text{stDevSamp}\{\{1,2,5,-6,3,-2\}\} \\ \frac{\sqrt{62}}{2} \\ \hline \text{stDevSamp}\{\{1,3,2,5,-6,4\}, \{3,2,5\}\} \\ 4.33345 \end{array}$$

stDevSamp(*Matrix I*[, Häufigkeitsmatrix])
⇒Matrix

Ergibt einen Zeilenvektor der Stichproben-Standardabweichungen der Spalten in *Matrix I*.

Jedes Häufigkeitsmatrix-Element gewichtet die Elemente von *Matrix I* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Matrix I* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

$$\begin{array}{c} \text{stDevSamp} \begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{pmatrix} \begin{bmatrix} 4 & \sqrt{13} & 2 \end{bmatrix} \\ \hline \text{stDevSamp} \begin{pmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{pmatrix} \begin{pmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{pmatrix} \\ \hline [2.7005 \quad 5.44695] \end{array}$$

Stop (Stopp)

Katalog >

Stop

Programmierbefehl: Beendet das Programm.

Stop ist in Funktionen nicht zulässig.

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

$i := 0$	0
Define $prog1() = \text{Prgm}$	<i>Done</i>
For $i, 1, 10, 1$	
If $i = 5$	
Stop	
EndFor	
EndPrgm	

$prog1()$	Done
i	5

Store (Speichern)

Siehe → (speichern), Seite 249.

string() (String)

Katalog >

$\text{string}(Ausdr) \Rightarrow \text{String}$

Vereinfacht $Ausdr$ und gibt das Ergebnis als Zeichenkette zurück.

$\text{string}(1.2345)$	"1.2345"
$\text{string}(1+2)$	"3"
$\text{string}(\cos(x)+\sqrt{3})$	" $\cos(x)+\sqrt{3}$ "

subMat() (Untermatrix)

Katalog >

$\text{subMat}[\text{Matrix } 1[, \text{vonZei}] [, \text{vonSpl}] [, \text{bisZei}] [, \text{bisSpl}]] \Rightarrow \text{Matrix}$

Gibt die angegebene Untermatrix von $\text{Matrix } 1$ zurück.

Vorgaben: $\text{vonZei}=1$, $\text{vonSpl}=1$, $\text{bisZei}=\text{letzte Zeile}$, $\text{bisSpl}=\text{letzte Spalte}$.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
$\text{subMat}(m1, 2, 1, 3, 2)$	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
$\text{subMat}(m1, 2, 2)$	$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

Summe (Sigma)

Siehe $\Sigma()$, Seite 238.

sum() (Summe)

Katalog >

sum(Liste[, Start[, Ende]])⇒Ausdruck

Gibt die Summe der Elemente in *Liste* zurück.

Start und *Ende* sind optional. Sie geben einen Elementebereich an.

Ein ungültiges Argument erzeugt ein ungültiges Ergebnis. Leere (ungültige) Elemente in *Liste* werden ignoriert.
Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

sum(Matrix1[, Start[, Ende]])⇒Matrix

Gibt einen Zeilenvektor zurück, der die Summen der Elemente aus den Spalten von *Matrix1* enthält.

Start und *Ende* sind optional. Sie geben einen Zeilenbereich an.

Ein ungültiges Argument erzeugt ein ungültiges Ergebnis. Leere (ungültige) Elemente in *Matrix1* werden ignoriert.
Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

sum({1,2,3,4,5})	15
sum({a,2·a,3·a})	6·a
sum(seq(n,n,1,10))	55
sum({1,3,5,7,9},3)	21

sumIf()

Katalog >

sumIf(Liste,Kriterien[, SummeListe])
⇒Wert

Gibt die kumulierte Summe aller Elemente in *Liste* zurück, die die angegebenen *Kriterien* erfüllen. Optional können Sie eine Alternativliste, *SummeListe*, angeben, an die die Elemente zum Kumulieren weitergegeben werden sollen.

Liste kann ein Ausdruck, eine Liste oder eine Matrix sein. *SummeListe* muss, sofern sie verwendet wird, dieselben Dimension (en) haben wie *Liste*.

Kriterien können sein:

- Ein Wert, ein Ausdruck oder eine Zeichenfolge. So kumuliert beispielsweise **34** nur solche Elemente in *Liste*, die

sum({1 2 3 4 5 6})	[5 7 9]
sum({1 2 3 4 5 6 7 8 9})	[12 15 18]
sum({1 2 3 4 5 6 7 8 9},{2,3})	[11 13 15]

sumIf({1,2,e,3,π,4,5,6},2.5<?<4.5)	e+π+7
sumIf({1,2,3,4},2<?<5,{10,20,30,40})	70

vereinfacht den Wert 34 ergeben.

- Ein Boolescher Ausdruck, der das Sonderzeichen ? als Platzhalter für jedes Element verwendet. Beispielsweise zählt ?<10 nur solche Elemente in *Liste* zusammen, die kleiner als 10 sind.

Wenn ein Element in *Liste* die *Kriterien* erfüllt, wird das Element zur Kumulationssumme hinzugerechnet. Wenn Sie *SummeListe* hinzufügen, wird stattdessen das entsprechende Element aus *SummeListe* zur Summe hinzugerechnet.

In der Lists & Spreadsheet Applikation können Sie anstelle von *Liste* und *SummeListe* auch einen Zellenbereich verwenden.

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

Hinweis: Siehe auch **countIf()**, Seite 41.

sumSeq()

Siehe $\Sigma()$, Seite 238.

system() (System)

system(Ausdr1 [, Ausdr2 [, Ausdr3 [, ...]]])

system(Glch1 [, Glch2 [, Glch3 [, ...]]])

$$\text{solve} \begin{cases} x+y=0 \\ x-y=8 \end{cases} \quad x=4 \text{ and } y=-4$$

Gibt ein Gleichungssystem zurück, das als Liste formatiert ist. Sie können ein Gleichungssystem auch mit Hilfe einer Vorlage erstellen.

Hinweis: Siehe auch **Gleichungssystem**, Seite 7.

T (Transponierte)**Katalog > ** $\text{Matrix} I \mathbf{T} \Rightarrow \text{matrix}$

Gibt die komplexe konjugierte, transponierte Matrix von *MatrixI* zurück.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie @t eintippen.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$
$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^\tau$	$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$
$\begin{bmatrix} 1+i & 2+i \\ 3+i & 4+i \end{bmatrix}^\tau$	$\begin{bmatrix} 1-i & 3-i \\ 2-i & 4-i \end{bmatrix}$

tan() (Tangens)**trig Taste** $\text{tan}(\text{AusdrI}) \Rightarrow \text{Ausdruck}$ $\text{tan}(\text{ListeI}) \Rightarrow \text{Liste}$

$\text{tan}(\text{AusdrI})$ gibt den Tangens des Arguments als Ausdruck zurück.

$\text{tan}(\text{ListeI})$ gibt in Form einer Liste für jedes Element in *ListeI* den Tangens zurück.

Hinweis: Das Argument wird entsprechend dem aktuellen Winkelmodus als Winkel in Grad, Neugrad oder Bogenmaß interpretiert. Sie können °, G oder r benutzen, um die Winkelmoduseinstellung temporär zu ändern.

Im Grad-Modus:

$\tan\left(\frac{\pi}{4}\right)$	1
$\tan(45)$	1
$\tan(\{0,60,90\})$	$\{0,\sqrt{3},\text{undef}\}$

Im Neugrad-Modus:

$\tan\left(\frac{\pi}{4}\right)$	1
$\tan(50)$	1
$\tan(\{0,50,100\})$	$\{0,1,\text{undef}\}$

Im Bogenmaß-Modus:

$\tan\left(\frac{\pi}{4}\right)$	1
$\tan(45^\circ)$	1
$\tan\left(\left\{-\frac{\pi}{3}, -\frac{\pi}{4}\right\}\right)$	$\{0,\sqrt{3},0,1\}$

Im Bogenmaß-Modus:

 $\text{tan}(\text{QuadratmatrixI}) \Rightarrow \text{Quadratmatrix}$

Gibt den Matrix-Tangens von *QuadratmatrixI* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Tangens jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im

tan() (Tangens)

trig Taste

Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$$\tan^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{pmatrix}$$

tan⁻¹() (Arkustangens)

trig Taste

tan⁻¹(Ausdr1)⇒Ausdruck

tan⁻¹(Liste1)⇒Liste

tan⁻¹(Ausdr1) gibt den Winkel, dessen Tangens *Ausdr1* ist, als Ausdruck zurück.

tan⁻¹(Liste1) gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Tangens zurück.

Hinweis: Das Ergebnis wird gemäß der aktuellen Winkelmoduseinstellung in Grad, in Neugrad oder im Bogenmaß zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arctan (...) eintippen**.

tan⁻¹(Quadratmatrix1)⇒Quadratmatrix

Gibt den inversen Matrix-Tangens von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Tangens jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Grad-Modus:

$$\tan^{-1}(1)$$
 45

Im Neugrad-Modus:

$$\tan^{-1}(1)$$
 50

Im Bogenmaß-Modus:

$$\tan^{-1}(\{0,0.2,0.5\}) = \{0,0.197396,0.463648\}$$

Im Bogenmaß-Modus:

$$\tan^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{pmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{pmatrix}$$

tangentLine()**tangentLine(Ausdr1,Var,Punkt)** \Rightarrow Ausdruck**tangentLine(Ausdr1,Var=Punkt)** \Rightarrow Ausdruck

Gibt die Tangente zu der durch *Ausdr1* dargestellten Kurve an dem in *Var=Punkt* angegebenen Punkt zurück.

Stellen Sie sicher, dass die unabhängige Variable nicht definiert ist. Wenn zum Beispiel $f1(x):=5$ und $x:=3$ ist, gibt **tangentLine(f1(x),x,2)** "false" zurück.

tangentLine($x^2,x,1$)	$2 \cdot x - 1$
tangentLine($(x-3)^2-4,x=3$)	-4
tangentLine($x^3,x=0$)	$x=0$
tangentLine($\sqrt{x^2-4},x=2$)	undef
x:=3: tangentLine($x^2,x,1$)	5

tanh() (Tangens hyperbolicus)**tanh(Ausdr1)** \Rightarrow Ausdruck**tanh(Liste1)** \Rightarrow Liste

tanh(Ausdr1) gibt den Tangens hyperbolicus des Arguments als Ausdruck zurück.

tanh(Liste1) gibt in Form einer Liste für jedes Element aus *Liste1* den Tangens hyperbolicus zurück.

tanh(Quadratmatrix1) \Rightarrow Quadratmatrix

Gibt den Matrix-Tangens hyperbolicus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Tangens hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

tanh(1.2)	0.833655
tanh({0,1})	{0,tanh(1)}

Im Bogenmaß-Modus:

tanh $\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$	$\begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$
--	---

tanh⁻¹(I) (Arkustangens hyperbolicus)**tanh⁻¹(Ausdr1)** \Rightarrow Ausdruck

Im Komplex-Formatmodus "kartesisch":

tanh⁻¹(Liste1) \Rightarrow Liste**tanh⁻¹(Ausdr1)** gibt den inversen Tangens

tanh⁻¹⁽⁾ (Arkustangens hyperbolicus)

Katalog >

hyperbolicus des Arguments als Ausdruck zurück.

tanh⁻¹(Liste1) gibt in Form einer Liste für jedes Element aus *Liste1* den inversen Tangens hyperbolicus zurück.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **arctanh (...) eintippen.**

tanh⁻¹(Quadratmatrix1)⇒Quadratmatrix

Gibt den inversen Matrix-Tangens hyperbolicus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des inversen Tangens hyperbolicus jedes einzelnen Elements. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

tanh ⁻¹ (0)	0
tanh ⁻¹ ({1,2,1,3})	$\left\{ \text{undef}, 0.518046 - 1.5708 \cdot i, \frac{\ln(2)}{2} - \frac{\pi}{2} \cdot i \right\}$

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

tanh ⁻¹ ($\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$)	$\begin{bmatrix} -0.099353 + 0.164058 \cdot i & 0.267834 - 1.4908 \\ -0.087596 - 0.725533 \cdot i & 0.479679 - 0.94730 \\ 0.511463 - 2.08316 \cdot i & -0.878563 + 1.7901 \end{bmatrix}$
---	--

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

taylor() (Taylor-Polynom)

Katalog >

taylor(Ausdr1, Var, Ordnung[, Punkt])
⇒Ausdruck

Gibt das angeforderte Taylor-Polynom zurück. Das Polynom enthält alle ganzzahligen Potenzen von (*Var minus Punkt*) mit nicht verschwindenden Koeffizienten von Null bis *Ordnung*. **taylor()** gibt sich selbst zurück, wenn es keine endliche Potenzreihe dieser Ordnung gibt oder negative oder Bruchexponenten erforderlich wären. Benutzen Sie Substitution und/oder die temporäre Multiplikation mit einer Potenz (*Var minus Punkt*), um allgemeinere Potenzreihen zu ermitteln.

Punkt ist vorgegeben als Null und ist der Entwicklungspunkt.

taylor($e^{\sqrt{x}}$,x,2)	taylor($e^{\sqrt{x}}$,x,2,0)
taylor(e^t ,t,4) $t=\sqrt{x}$	$\frac{3}{24} + \frac{x^2}{6} + \frac{x}{2} + \sqrt{x} + 1$
taylor($\frac{1}{x(x-1)}$,x,3)	taylor($\frac{1}{x(x-1)}$,x,3,0)
expand($\frac{\text{taylor}(\frac{x}{x(x-1)},x,4)}{x}$)	$-x^3 - x^2 - x - \frac{1}{x} - 1$

tCdf(*UntGrenze*,*ObGrenze*,*FreiGrad*)⇒Zahl, wenn
UntGrenze und *ObGrenze* Zahlen sind, Liste, wenn
UntGrenze und *ObGrenze* Listen sind

Berechnet für eine Student-*t*-Verteilung mit vorgegebenen Freiheitsgraden *FreiGrad* die Intervallwahrscheinlichkeit zwischen *UntGrenze* und *ObGrenze*.

Für $P(X \leq \text{obereGrenze})$ setzen Sie *untereGrenze* = $-\infty$.

tCollect() (Trigonometrische Zusammenfassung)

tCollect(*Ausdr1*)⇒*Ausdruck*

Gibt einen Ausdruck zurück, in dem Produkte und ganzzahlige Potenzen von Sinus und Cosinus in eine lineare Kombination von Sinus und Cosinus von Winkelvielfachen, Winkelsummen und Winkeldifferenzen umgewandelt sind. Diese Transformation wandelt trigonometrische Polynome in eine lineare Kombination um.

In manchen Fällen führt **tCollect()** zum Erfolg, wo die vorgegebene trigonometrische Vereinfachung nicht zum Erfolg führt. **tCollect()** bewirkt in beinahe allen Fällen eine Umkehrung von Transformationen, die mit **tExpand()** vorgenommen wurden. Manchmal lässt sich ein Ausdruck vereinfachen, wenn man in getrennten Schritten **tExpand()** auf ein Ergebnis von **tCollect()** anwendet (oder umgekehrt).

$$\frac{\text{tCollect}((\cos(\alpha))^2)}{2} = \frac{\cos(2\cdot\alpha)+1}{2}$$

$$\frac{\text{tCollect}(\sin(\alpha)\cdot\cos(\beta))}{2} = \frac{\sin(\alpha-\beta)+\sin(\alpha+\beta)}{2}$$

tExpand() (Trigonometrische Entwicklung)

tExpand(*Ausdr1*)⇒*Ausdruck*

Gibt einen Ausdruck zurück, in dem Sinus und Cosinus von ganzzahligen

$$\frac{\text{tExpand}(\sin(3\cdot\phi))}{4\cdot\sin(\phi)\cdot(\cos(\phi))^2-\sin(\phi)}$$

$$\frac{\text{tExpand}(\cos(\alpha-\beta))}{\cos(\alpha)\cdot\cos(\beta)+\sin(\alpha)\cdot\sin(\beta)}$$

Winkelvielfachen, Winkelsummen und Winkeldifferenzen entwickelt sind. Aufgrund der Identität $(\sin(x))^2 + (\cos(x))^2 = 1$ sind viele äquivalente Ergebnisse möglich. Ein Ergebnis kann sich daher von einem in anderen Publikationen angegebenen unterscheiden.

In manchen Fällen führt **tExpand()** zum Erfolg, wo die vorgegebene trigonometrische Vereinfachung nicht zum Erfolg führt. **tExpand()** bewirkt in beinahe allen Fällen eine Umkehrung von Transformationen, die mit **tCollect()** vorgenommen wurden. Manchmal lässt sich ein Ausdruck vereinfachen, wenn man in getrennten Schritten **tCollect()** auf ein Ergebnis von **tExpand()** anwendet (oder umgekehrt).

Hinweis: Die Skalierung von $\pi/180$ im Winkelmodus "Grad" behindert die Erkennung entwickelbarer Formen durch **tExpand()**. Die besten Ergebnisse werden bei Benutzung von **tExpand()** im Bogenmaß-Modus erzielt.

Text

Text *EingabeString[, FlagAnz]*

Programmierbefehl: Pausiert das Programm und zeigt die Zeichenkette *EingabeString* in einem Dialogfeld an.

Wenn der Benutzer **OK** auswählt, wird die Programmausführung fortgesetzt.

Bei dem optionalen Argument *FlagAnz* kann es sich um einen beliebigen Ausdruck handeln.

- Wenn *FlagAnz* fehlt oder den Wert **1** ergibt, wird die Textmeldung im Calculator-Protokoll angezeigt.
- Wenn *FlagAnz* den Wert **0** ergibt, wird die Meldung nicht im Protokoll angezeigt.

Wenn das Programm eine Eingabe vom Benutzer benötigt, verwenden Sie stattdessen **Request**, Seite 156, oder **RequestStr**, Seite 158.

Definieren Sie ein Programm, das fünfmal anhält und jeweils eine Zufallszahl in einem Dialogfeld anzeigt.

Schließen Sie in der Vorlage **Prgm...EndPrgm** jede Zeile mit **[enter]** ab anstatt mit **[enter]**. Auf der Computertastatur halten Sie **Alt** gedrückt und drücken die **Eingabetaste**.

Define text_demo()=Prgm

For i,1,5

strinfo:="Random number " & string(rand(i))

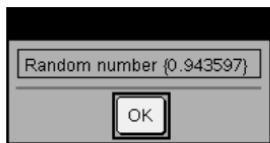
Hinweis: Sie können diesen Befehl in benutzerdefinierten Programmen, aber nicht in Funktionen verwenden.

Text strinfo
EndFor
EndPrgm

Starten Sie das Programm:

`text_demo()`

Muster eines Dialogfelds:

**tInterval**

tInterval *Liste[, Häuf[, KNiv]]*

(Datenlisteneingabe)

tInterval *Ȑ, sx, n[, KNiv]*

(Zusammenfassende statistische Eingabe)

Berechnet das Konfidenzintervall *t*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 186.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekannten Populationsmittelwert
stat.Ȑ	Stichprobenmittelwert der Datenfolge aus der zufälligen Normalverteilung

AusgabevARIABLE	Beschreibung
stat.ME	Fehlertoleranz
stat.df	Freiheitsgrade
stat.ox	Stichproben-Standardabweichung
stat.n	Länge der Datenfolge mit Stichprobenmittelwert

tInterval_2Samp (Zwei-Stichproben-t-Konfidenzintervall)

Katalog > 

tInterval_2Samp Liste1, Liste2[, Häufigkeit1
[, Häufigkeit2[, KStufe[, Verteilt]]]]

(Datenlisteneingabe)

tInterval_2Samp $\bar{x}_1, sx_1, n_1, \bar{x}_2, sx_2, n_2[, KStufe
[, Verteilt]]]$

(Zusammenfassende statistische Eingabe)

Berechnet ein t -Konfidenzintervall für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Verteilt=1 verteilt Varianzen; *Verteilt=0* verteilt keine Varianzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat. $\bar{x}_1-\bar{x}_2$	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.df	Freiheitsgrade
stat. \bar{x}_1 , stat. \bar{x}_2	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat.ox1, stat.ox2	Stichproben-Standardabweichungen für <i>Liste 1</i> und <i>Liste 2</i>
stat.n1, stat.n2	Anzahl der Stichproben in Datenfolgen
stat.sp	Die verteilte Standardabweichung. Wird berechnet, wenn <i>Verteilt = JA</i> .

tmpCnv() (Konvertierung von Temperaturwerten)

Katalog > 

tmpCnv(Ausdr \cdot TempEinh, \cdot TempEinh2)
⇒ Ausdruck \cdot TempEinh2

Konvertiert einen durch *Ausdr* definierten Temperaturwert von einer Einheit in eine andere. Folgende Temperatureinheiten sind gültig:

\cdot °C Celsius

\cdot °F Fahrenheit

\cdot °K Kelvin

\cdot °R Rankine

Wählen Sie zur Eingabe von ° das Symbol aus der Sonderzeichenpalette des Katalogs aus.

Zur Eingabe von _ drücken Sie **ctrl** **[]**.

100 °C wird zum Beispiel in 212 °F konvertiert.

Zur Konvertierung eines Temperaturbereichs verwenden Sie hingegen **ΔtmpCnv()**.

tmpCnv(100 · °C, °F)	212 · °F
tmpCnv(32 · °F, °C)	0 · °C
tmpCnv(0 · °C, °K)	273.15 · °K
tmpCnv(0 · °F, °R)	459.67 · °R

Hinweis: Sie können den Katalog verwenden, um Temperatureinheiten auszuwählen.

ΔtmpCnv() (Konvertierung von Temperaturbereichen)

Katalog > 

ΔtmpCnv(Ausdr \cdot tempEinh, \cdot tempEinh2)
⇒ Ausdruck \cdot tempEinh2

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **deltaTmpCnv (...)** eintippen.

Konvertiert einen durch *Ausdr* definierten Temperaturbereich (Differenz zwischen zwei Temperaturwerten) von einer Einheit in eine andere. Folgende Temperatureinheiten sind gültig:

\cdot °C Celsius

\cdot °F Fahrenheit

Wählen Sie zur Eingabe von Δ das Symbol aus der Sonderzeichenpalette des Katalogs aus.

ΔtmpCnv(100 · °C, °F)	180 · °F
ΔtmpCnv(180 · °F, °C)	100 · °C
ΔtmpCnv(100 · °C, °K)	100 · °K
ΔtmpCnv(100 · °F, °R)	100 · °R
ΔtmpCnv(1 · °C, °F)	1.8 · °F

Hinweis: Sie können den Katalog verwenden, um Temperatureinheiten auszuwählen.

ΔtmpCnv() (Konvertierung von Temperaturbereichen)

Katalog >

_°K Kelvin

_°R Rankine

Wählen Sie zur Eingabe von ° das Symbol aus der Sonderzeichenpalette oder geben Sie @d ein.

Zur Eingabe von _ drücken Sie **ctrl** .

1_°C und 1_°K haben denselben Absolutwert, ebenso wie 1_°F und 1_°R. 1_°C ist allerdings 9/5 so groß wie 1_°F.

Ein 100_°C Bereich (von 0_°C bis 100_°C) ist beispielsweise einem 180_°F Bereich äquivalent.

Zur Konvertierung eines bestimmten Temperaturwerts verwenden Sie hingegen **tmpCnv()**.

tPdf()

Katalog >

tPdf(*XWert,FreiGrad*)⇒Zahl, wenn *XWert* eine Zahl ist, Liste, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeitsdichtefunktion (Pdf) einer Student-*t*-Verteilung an einem bestimmten *x*-Wert für die vorgegebenen Freiheitsgrade *FreiGrad*.

trace()

Katalog >

trace(*Quadratmatrix*)⇒Ausdruck

Gibt die Spur (Summe aller Elemente der Hauptdiagonalen) von *Quadratmatrix* zurück.

trace $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$	15
trace $\begin{pmatrix} a & 0 \\ 1 & a \end{pmatrix}$	2· <i>a</i>

```
Try
block1
Else
block2
EndTry
```

Führt *Block1* aus, bis ein Fehler auftritt.
 Wenn in *Block1* ein Fehler auftritt, wird die
 Programmausführung an *Block2*
 übertragen. Die Systemvariable *Fehlercode*
(errCode) enthält den Fehlercode, der es
 dem Programm ermöglicht, eine
 Fehlerwiederherstellung durchzuführen.
 Eine Liste der Fehlercodes finden Sie unter
 "Fehlercodes und -meldungen" (Seite 259).

Block1 und *Block2* können einzelne
 Anweisungen oder Reihen von
 Anweisungen sein, die durch das Zeichen
 ":" voneinander getrennt sind.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von
 mehrzeiligen Programm- und
 Funktionsdefinitionen finden Sie im
 Abschnitt „Calculator“ des
 Produkthandbuchs.

Beispiel 2

Um die Befehle **Versuche (Try)**, **LöFehler (ClrErr)** und **Ügebefehl (PassErr)** im Betrieb zu sehen, geben Sie das rechts gezeigte Programm *eigenvals()* ein. Sie starten das Programm, indem Sie jeden der folgenden Ausdrücke eingeben.

$$\text{eigenvals}\left(\begin{bmatrix} -3 \\ -41 \\ 5 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix}\right)$$

$$\text{eigenvals}\left(\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$$

Hinweis: Siehe auch **LöFehler**, Seite 31, und **Ügebefehl**, Seite 138.

```
Define prog1()=Prgm
Try
z:=z+1
Disp "z incremented."
Else
Disp "Sorry, z undefined."
EndTry
EndPrgm
```

Done

z:=1:prog1()

z incremented.

Done

DelVar z:prog1()

Sorry, z undefined.

Done

Definiere eigenvals(a,b)=Prgm

© Programm *eigenvals(A,B)* zeigt die
 Eigenwerte von A·B an

Try

Disp "A= ",a

Disp "B= ",b

Disp " "

Disp "Eigenwerte von A·B sind:",eigVl(a*b)

Else

If errCode=230 Then

Disp "Fehler: Produkt von A·B muss eine
 quadratische Matrix sein"

ClrErr

```

Else
PassErr
EndIf
EndTry
EndPrgm

```

tTest**tTest** μ_0 ,*Liste*[,Häufigkeit[,*Hypoth*]]

(Datenlisteneingabe)

tTest μ_0 , \bar{x} ,*sx*,*n*,[*Hypoth*]

(Zusammenfassende statistische Eingabe)

Führt einen Hypothesen-Test für einen einzelnen, unbekannten Populationsmittelwert μ durch, wenn die Populations-Standardabweichung σ unbekannt ist. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Siehe Seite 186.)

Getestet wird $H_0: \mu = \mu_0$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu < \mu_0$ setzen Sie *Hypoth*<0

Für $H_a: \mu \neq \mu_0$ (Standard) setzen Sie *Hypoth*=0

Für $H_a: \mu > \mu_0$ setzen Sie *Hypoth*>0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabeveriable	Beschreibung
stat.t	$(\bar{x} - \mu_0) / (\text{stdev} / \sqrt{n})$
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade
stat. \bar{x}	Stichprobenmittelwert der Datenfolge in <i>Liste</i>
stat.sx	Stichproben-Standardabweichung der Datenfolge

AusgabevARIABLE	Beschreibung
stat.n	Stichprobenumfang

tTest_2Samp (t-Test für zwei Stichproben)

Katalog > 

tTest_2Samp Liste1, Liste2[, Häufigkeit1
[, Häufigkeit2[, Hypoth[, Verteilt]]]]

(Datenlisteneingabe)

tTest_2Samp $\bar{x}_1, sx1, n1, \bar{x}_2, sx2, n2[, Hypoth
[, Verteilt]]]$

(Zusammenfassende statistische Eingabe)

Berechnet einen *t*-Test für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Getestet wird $H_0: \mu_1 = \mu_2$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu_1 < \mu_2$ setzen Sie *Hypoth<0*

Für $H_a: \mu_1 \neq \mu_2$ (Standard) setzen Sie *Hypoth=0*

Für $H_a: \mu_1 > \mu_2$ setzen Sie *Hypoth>0*

Verteilt=1 verteilt Varianzen

Verteilt=0 verteilt keine Varianzen

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.t	Für die Differenz der Mittelwerte berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade für die t-Statistik
stat. \bar{x}_1 , stat. \bar{x}_2	Stichprobenmittelwerte der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.n1, stat.n2	Stichprobenumfang
stat.sp	Die verteilte Standardabweichung. Wird berechnet, wenn <i>Verteilt=1</i> .

tvmFV()**Katalog >**

tvmFV($N, I, PV, Pmt, [PpY], [CpY], [PmtAt]$)
 \Rightarrow Wert

tvmFV(120,5,0,-500,12,12)

77641.1

Finanzfunktion, die den Geld-Endwert berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 206) beschrieben. Siehe auch **amortTbl()**, Seite 12.

tvmI()**Katalog >**

tvmI($N, PV, Pmt, FV, [PpY], [CpY], [PmtAt]$)
 \Rightarrow Wert

tvmI(240,100000,-1000,0,12,12)

10.5241

Finanzfunktion, die den jährlichen Zinssatz berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 206) beschrieben. Siehe auch **amortTbl()**, Seite 12.

tvmN()**Katalog >**

tvmN($I, PV, Pmt, FV, [PpY], [CpY], [PmtAt]$)
 \Rightarrow Wert

tvmN(5,0,-500,77641,12,12)

120.

Finanzfunktion, die die Anzahl der Zahlungsperioden berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 206) beschrieben. Siehe auch **amortTbl()**, Seite 12.

tvmPmt()**Katalog >**

tvmPmt($N, I, PV, FV, [PpY], [CpY], [PmtAt]$)
 \Rightarrow Wert

tvmPmt(60,4,30000,0,12,12)

-552.496

Finanzfunktion, die den Betrag der einzelnen Zahlungen berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 206) beschrieben. Siehe auch **amortTbl()**, Seite 12.

tvmPV(N,I,Pmt,FV,[PpY],[CpY],[PmtAt])
⇒Wert

tvmPV(48,4,-500,30000,12,12) -3426.7

Finanzfunktion, die den Barwert berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 206) beschrieben. Siehe auch **amortTbl()**, Seite 12.

TVM-Argumente*	Beschreibung	Datentyp
N	Anzahl der Zahlungsperioden	reelle Zahl
I	Jahreszinssatz	reelle Zahl
PV	Barwert	reelle Zahl
Pmt	Zahlungsbetrag	reelle Zahl
FV	Endwert	reelle Zahl
PpY	Zahlungen pro Jahr, Standard=1	Ganzzahl > 0
CpY	Verzinsungsperioden pro Jahr, Standard=1	Ganzzahl > 0
PmtAt	Zahlung fällig am Ende oder am Anfang der jeweiligen Zahlungsperiode, Standard=Ende	Ganzzahl (0=Ende, 1=Anfang)

* Die Namen dieser TVM-Argumente ähneln denen der TVM-Variablen (z.B. **tvm.pv** und **tvm.pmt**), die vom Finanzlöser der *Calculator* Applikation verwendet werden. Die Werte oder Ergebnisse der Argumente werden jedoch von den Finanzfunktionen nicht unter den TVM-Variablen gespeichert.

TwoVar X, Y[, Häuf] [, Kategorie, Mit]]

Berechnet die 2-Variablen-Statistik. Eine Zusammenfassung der Ergebnisse wird in der

Variablen *stat.results* gespeichert. (Seite 186.)

Alle Listen außer *Mit* müssen die gleiche Dimension besitzen.

X und *Y* sind Listen von unabhängigen und abhängigen Variablen.

Häuf ist eine optionale Liste von Häufigkeitswerten. Jedes Element in *Häuf* gibt die Häufigkeit für jeden entsprechenden *X*- und *Y*-Datenpunkt an. Der Standardwert ist 1. Alle Elemente müssen Ganzzahlen ≥ 0 sein.

Kategorie ist eine Liste von Kategoriecodes für die entsprechenden *X* und *Y* Daten.

Mit ist eine Liste von einem oder mehreren Kategoriecodes. Nur solche Datenelemente, deren Kategoriecode in dieser Liste enthalten ist, sind in der Berechnung enthalten.

Ein leeres (ungültiges) Element in einer der Listen *X*, *Freq* oder *Kategorie* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Ein leeres (ungültiges) Element in einer der Listen *X1* bis *X20* führt zu einem Fehler im entsprechenden Element aller dieser Listen. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

Ausgabeverable	Beschreibung
stat. \bar{x}	Mittelwert der x-Werte
stat. x	Summe der x-Werte
stat. x2	Summe der x2-Werte
stat.sx	Stichproben-Standardabweichung von x
stat. x	Populations-Standardabweichung von x
stat. n	Anzahl der Datenpunkte
stat. \bar{y}	Mittelwert der y-Werte
stat. y	Summe der y-Werte
stat. y ²	Summe der y2-Werte
stat.sy	Stichproben-Standardabweichung von y
stat. y	Populations-Standardabweichung von y

AusgabevARIABLE	Beschreibung
Stat. xy	Summe der $x \cdot y$ -Werte
stat.r	Korrelationskoeffizient
stat.MinX	Minimum der x-Werte
stat.Q ₁ X	1. Quartil von x
stat.MedianX	Median von x
stat.Q ₃ X	3. Quartil von x
stat.MaxX	Maximum der x-Werte
stat.MinY	Minimum der y-Werte
stat.Q ₁ Y	1. Quartil von y
stat.MedY	Median von y
stat.Q ₃ Y	3. Quartil von y
stat.MaxY	Maximum der y-Werte
stat.(x-)²	Summe der Quadrate der Abweichungen der x-Werte vom Mittelwert
stat.(y-)²	Summe der Quadrate der Abweichungen der y-Werte vom Mittelwert

U

unitV() (Einheitsvektor)

Katalog >

unitV(Vektor l)⇒Vektor

Gibt je nach der Form von *Vektor l* entweder einen Zeilen- oder einen Spalteneinheitsvektor zurück.

Vektor l muss eine einzeilige oder eine einspaltige Matrix sein.

$$\begin{array}{c}
 \text{unitV}([a \ b \ c]) \\
 \left[\begin{array}{c} a \\ \sqrt{a^2+b^2+c^2} \end{array} \right] \quad \left[\begin{array}{c} b \\ \sqrt{a^2+b^2+c^2} \end{array} \right] \quad \left[\begin{array}{c} c \\ \sqrt{a^2+b^2+c^2} \end{array} \right] \\
 \text{unitV}([1 \ 2 \ 1]) \\
 \left[\begin{array}{c} \frac{\sqrt{6}}{6} \\ \frac{\sqrt{6}}{3} \\ \frac{\sqrt{6}}{6} \end{array} \right] \\
 \text{unitV} \left(\begin{array}{c} 1 \\ 2 \\ 3 \end{array} \right) \\
 \left[\begin{array}{c} \frac{\sqrt{14}}{14} \\ \frac{14}{14} \\ \frac{\sqrt{14}}{14} \\ \frac{7}{3 \cdot \sqrt{14}} \\ \frac{3 \cdot \sqrt{14}}{14} \end{array} \right]
 \end{array}$$

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

unLock

Katalog >

unLock*Var1 [, Var2] [, Var3] ...*

unLock*Var.*

Entsperrt die angegebenen Variablen bzw. die Variablengruppe. Gesperzte Variablen können nicht geändert oder gelöscht werden.

Siehe **Lock**, Seite 112, und **getLockInfo()**, Seite 88.

<i>a:=65</i>	65
<i>Lock a</i>	<i>Done</i>
<i>getLockInfo(a)</i>	1
<i>a:=75</i>	"Error: Variable is locked."
<i>DelVar a</i>	"Error: Variable is locked."
<i>Unlock a</i>	<i>Done</i>
<i>a:=75</i>	75
<i>DelVar a</i>	<i>Done</i>

V

varPop() (Populationsvarianz)

Katalog >

varPop*[Liste, Häufigkeitsliste]*
⇒Ausdruck

Ergibt die Populationsvarianz von *Liste* zurück.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Liste* muss mindestens zwei Elemente enthalten.

Wenn ein Element in einer der Listen leer (ungültig) ist, wird dieses Element ignoriert. Das entsprechende Element in der anderen Liste wird ebenfalls ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

<i>varPop({5,10,15,20,25,30})</i>	875
	12

<i>Ans·1.</i>	72.9167
---------------	---------

varSamp() (Stichproben-Varianz)

Katalog >

varSamp*[Liste, Häufigkeitsliste]*
⇒Ausdruck

Ergibt die Stichproben-Varianz von *Liste*.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Liste* muss mindestens zwei Elemente enthalten.

<i>varSamp({a,b,c})</i>	
$\frac{a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2}{3}$	
<i>varSamp({1,2,5,-6,3,-2})</i>	31
	2
<i>varSamp({1,3,5},{4,6,2})</i>	68
	33

Wenn ein Element in einer der Listen leer (ungültig) ist, wird dieses Element ignoriert. Das entsprechende Element in der anderen Liste wird ebenfalls ignoriert.
Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

varSamp(*Matrix1*[, *Häufigkeitsmatrix*])
 \Rightarrow Matrix

Gibt einen Zeilenvektor zurück, der die Stichproben-Varianz jeder Spalte von *Matrix1* enthält.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

Wenn ein Element in einer der Matrizen leer (ungültig) ist, wird dieses Element ignoriert. Das entsprechende Element in der anderen Matrix wird ebenfalls ignoriert.
Weitere Informationen zu leeren Elementen finden Sie (Seite 252).

Hinweis: *Matrix1* muss mindestens zwei Zeilen enthalten.

varSamp	$\begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{pmatrix}$	$\begin{bmatrix} 4.75 & 1.03 & 4 \end{bmatrix}$
varSamp	$\begin{pmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{pmatrix}$	$\begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}$ $\begin{bmatrix} 3.91731 & 2.08411 \end{bmatrix}$

W

Wait

Wait *ZeitInSekunden*

Setzt die Ausführung für einen Zeitraum von *ZeitInSekunden* aus.

Wait ist besonders nützlich bei einem Programm, das eine kurze Verzögerung benötigt, damit die angeforderten Daten verfügbar werden.

Das Argument *ZeitInSekunden* muss ein Ausdruck sein, der zu einem Dezimalwert im Bereich von 0 bis 100 vereinfacht wird. Der Befehl runden diesen Wert auf die nächsten 0,1 Sekunden auf.

Zum Abbrechen eines **Wait** das gerade durchgeführt wird,

Um 4 Sekunden zu warten:

Wait 4

Um 1/2 Sekunde zu warten:

Wait 0.5

Um 1,3 Sekunden mithilfe der Variablen *seccount* zu warten:

seccount:=1.3

Wait seccount

Dieses Beispiel schaltet eine grüne LED 0,5 Sekunden lang ein und anschließend aus.

Send “SET GREEN 1 ON”

Wait 0.5

Send “SET GREEN 1 OFF”

- **Handheld:** Halten Sie die Taste gedrückt und drücken Sie mehrmals .
- **Windows®:** Halten Sie die Taste **F12** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **Macintosh®:** Halten Sie die Taste **F5** gedrückt und drücken Sie mehrmals die **Eingabetaste**.
- **iPad®:** Die App zeigt eine Eingabeaufforderung an. Sie können weiter warten oder abbrechen.

Hinweis: Sie können den Befehl **Wait** in einem benutzerdefinierten Programm, aber nicht in einer Funktion verwenden.

warnCodes ()

warnCodes(Ausdr1, StatusVar)⇒Ausdruck

Wertet den Ausdruck *Ausdr1* aus, gibt das Ergebnis zurück und speichert die Codes aller erzeugten Warnungen in der Listenvariablen *StatusVar*. Wenn keine Warnungen erzeugt werden, weist diese Funktion *StatusVar* eine leere Liste zu.

Ausdr1 kann jeder in TI-Nspire™ oder TI-Nspire™ CAS gültige mathematische Ausdruck sein. *Ausdr1* kann kein Befehl und keine Zuweisung sein.

StatusVar muss ein gültiger Variablenname sein.

Eine Liste der Warncodes und der zugehörigen Meldungen finden Sie (Seite 268).

warnCodes(solve($\sin(10 \cdot x) = \frac{x^2}{x}, x$), warn)

 $x = -0.84232 \text{ or } x = -0.706817 \text{ or } x = -0.2852$

warn	$\{10007, 10009\}$
------	--------------------

Um das ganze Ergebnis zu sehen, drücken Sie \blacktriangle und verwenden dann \blacktriangleleft und \triangleright , um den Cursor zu bewegen.

when() (Wenn)

when(Bedingung, wahresErgebnis [, falschesErgebnis][, unbekanntesErgebnis])⇒Ausdruck

Gibt *wahresErgebnis*, *falschesErgebnis* oder

unbekanntesErgebnis zurück, je nachdem, ob die *Bedingung* wahr, falsch oder unbekannt ist. Gibt die Eingabe zurück, wenn zu wenige Argumente angegeben werden.

Lassen Sie sowohl *falschesErgebnis* als auch *unbekanntesErgebnis* weg, um einen Ausdruck nur für den Bereich zu bestimmen, in dem *Bedingung* wahr ist.

Geben Sie **undef** für *falschesErgebnis* an, um einen Ausdruck zu bestimmen, der nur in einem Intervall graphisch dargestellt werden soll.

when() ist hilfreich für die Definition rekursiver Funktionen.

when($x < 0, x + 3$) $x = 5$	undef
---------------------------------	-------

when($n > 0, n \cdot factorial(n - 1), 1$) → <i>factorial</i> (n)	Done
---	------

<i>factorial</i> (3)	6
----------------------	---

While

While *Bedingung*

Block

EndWhile

Führt die in *Block* enthaltenen Anweisungen so lange aus, wie *Bedingung* wahr ist.

Block kann eine einzelne Anweisung oder eine Serie von Anweisungen sein, die durch ":" getrennt sind.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define <i>sum_of_recip</i> (n) = Func Local $i, tempsum$ $1 \rightarrow i$ $0 \rightarrow tempsum$ While $i \leq n$ $tempsum + \frac{1}{i} \rightarrow tempsum$ $i + 1 \rightarrow i$ EndWhile Return <i>tempsum</i> EndFunc	Done
---	------

<i>sum_of_recip</i> (3)	11
-------------------------	----

xor (Boolesches exklusives oder)**Katalog > **

*BoolescherAusdr1***xor***BoolescherAusdr2*
ergibt Boolescher Ausdruck

true xor true	false
5>3 xor 3>5	true

*BoolescheListe1***xor***BoolescheListe2*
ergibt Boolesche Liste

*BoolescheMatrix1***xor***BoolescheMatrix2*
ergibt Boolesche Matrix

Gibt wahr zurück, wenn *BoolescherAusdr1* wahr und *BoolescherAusdr2* falsch ist und umgekehrt.

Gibt falsch zurück, wenn beide Argumente wahr oder falsch sind. Gibt einen vereinfachten Booleschen Ausdruck zurück, wenn eines der beiden Argumente nicht zu wahr oder falsch ausgewertet werden kann.

Hinweis: Siehe or, Seite 136.

Ganzzahl1 xor Ganzzahl2 \Rightarrow *Ganzzahl*

Vergleicht zwei reelle ganze Zahlen mit Hilfe einer xor-Operation Bit für Bit. Intern werden beide ganzen Zahlen in binäre 32-Bit-Zahlen mit Vorzeichen konvertiert. Beim Vergleich der sich entsprechenden Bits ist das Ergebnis 1, wenn eines der Bits (nicht aber beide) 1 ist; das Ergebnis ist 0, wenn entweder beide Bits 0 oder beide Bits 1 sind. Der zurückgegebene Wert stellt die Bit-Ergebnisse dar und wird im jeweiligen Basis-Modus angezeigt.

Sie können die ganzen Zahlen in jeder Basis eingeben. Für eine binäre oder hexadezimale Eingabe ist das Präfix 0b bzw. 0h zu verwenden. Ohne Präfix werden ganze Zahlen als dezimal behandelt (Basis 10).

Geben Sie eine dezimale ganze Zahl ein, die für eine 64-Bit-Dualform mit Vorzeichen zu groß ist, dann wird eine symmetrische Modulo-Operation ausgeführt, um den Wert in den erforderlichen Bereich zu bringen. Weitere Informationen finden Sie

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O

0h7AC36 xor 0h3D5F	0h79169
--------------------	---------

Im Bin-Modus:

0b100101 xor 0b100	0b100001
--------------------	----------

Hinweis: Eine binäre Eingabe kann bis zu 64 Stellen haben (das Präfix 0b wird nicht mitgezählt). Eine hexadezimale Eingabe kann bis zu 16 Stellen aufweisen.

unter **Base2**, Seite 22.

Hinweis: Siehe **or**, Seite 136.

Z

zeros() (Nullstellen)

zeros(Ausdr, Var)⇒Liste

zeros(Ausdr, Var= Schätzwert)⇒Liste

Gibt eine Liste möglicher reeller Werte für *Var* zurück, die *Ausdr*=0 ergeben. **zeros()** erreicht dies durch Berechnung von **expolist(solve(Ausdr=0,Var),Var)**.

Für manche Zwecke ist die Ergebnisform von **zeros()** günstiger als die von **solve()**. Allerdings kann die Ergebnisform von **zeros()** folgende Lösungen nicht ausdrücken: implizite Lösungen, Lösungen, für die Ungleichungen erforderlich sind, sowie Lösungen, die nicht *Var* betreffen.

Hinweis: Siehe auch **cSolve()**, **cZeros()** und **solve()**.

zeros({Ausdr1, Ausdr2},

**{VarOderSchätzwert1,
VarOderSchätzwert2 [, ...]})⇒Matrix**

Gibt mögliche reelle Nullstellen für die simultanen algebraischen Ausdrücke zurück, wobei jeder *VarOderSchätzwert* einen gesuchten unbekannten Wert angibt.

Sie haben die Option, eine Ausgangsschätzung für eine Variable anzugeben. *VarOderSchätzwert* muss immer die folgende Form haben:

Variable

– oder –

Variable = *reell* oder *nicht-reelle Zahl*

Beispiel: x ist gültig und x=3 ebenfalls.

$$\text{zeros}\left(a \cdot x^2 + b \cdot x + c, x\right) \\ \left\{ \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a}, \frac{-\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a} \right\}$$

$$a \cdot x^2 + b \cdot x + c | x = \text{Ans}[2] \quad 0$$

$$\text{exact}\left(\text{zeros}\left(a \cdot (e^x + x) \cdot (\text{sign}(x) - 1), x\right)\right) \quad \{\square\}$$

$$\text{exact}\left(\text{solve}\left(a \cdot (e^x + x) \cdot (\text{sign}(x) - 1) = 0, x\right)\right)$$

$$e^x + x = 0 \text{ or } x > 0 \text{ or } a = 0$$

zeros() (Nullstellen)

Katalog >

Wenn alle Ausdrücke Polynome sind und Sie KEINE Anfangsschätzwerte angeben, dann verwendet **zeros()** das lexikalische Gröbner/Buchbergersche Eliminationsverfahren beim Versuch, alle reellen Nullstellen zu bestimmen.

Betrachten wir z.B. einen Kreis mit dem Radius r und dem Ursprung als Mittelpunkt und einen weiteren Kreis mit Radius r und dem Schnittpunkt des ersten Kreises mit der positiven x-Achse als Mittelpunkt.

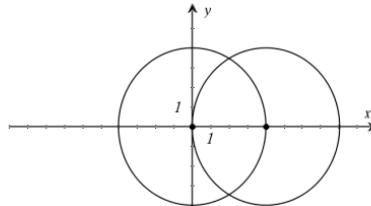
Verwenden Sie **zeros()** zur Bestimmung der Schnittpunkte.

Wie in nebenstehendem Beispiel durch r demonstriert, können simultane polynomische Ausdrücke zusätzliche Variablen ohne Wert aufweisen, die aber für numerische Werte stehen, welche später eingesetzt werden können.

Jede Zeile der sich ergebenden Matrix stellt eine alternative Nullstelle dar, wobei die Komponenten in derselben Reihenfolge wie in der *VarOderSchätzwert*-Liste angeordnet sind. Um eine Zeile zu erhalten ist die Matrix nach [Zeile] zu indizieren.

Sie können auch (oder stattdessen) Unbekannte angeben, die in den Ausdrücken nicht erscheinen. Geben Sie zum Beispiel z als eine Unbekannte an, um das vorangehende Beispiel auf zwei parallele, sich schneidende Zylinder mit dem Radius r auszudehnen. Die Zylinder-Nullstellen verdeutlichen, dass Nullstellenfamilien "beliebige" Konstanten der Form ck enthalten können, wobei k ein ganzzahliger Index im Bereich 1 bis 255 ist.

Bei polynomialen Gleichungssystemen kann die Berechnungsdauer oder Speicherbelastung stark von der Reihenfolge abhängen, in der Sie die Unbekannten angeben. Übersteigt Ihre erste Wahl die Speicherkapazität oder Ihre Geduld, versuchen Sie, die Variablen in den



$$\text{zeros}\left(\left\{x^2+y^2-r^2, (x-r)^2+y^2-r^2\right\}, \{x,y\}\right)$$

$$\begin{bmatrix} \frac{r}{2} & \frac{-\sqrt{3} \cdot r}{2} \\ \frac{r}{2} & \frac{\sqrt{3} \cdot r}{2} \end{bmatrix}$$

Zeile 2 extrahieren:

$$\text{Ans}[2]$$

$$\begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3} \cdot r}{2} \end{bmatrix}$$

$$\text{zeros}\left(\left\{x^2+y^2-r^2, (x-r)^2+y^2-r^2\right\}, \{x,y,z\}\right)$$

$$\begin{bmatrix} \frac{r}{2} & \frac{-\sqrt{3} \cdot r}{2} & \text{c1} \\ \frac{r}{2} & \frac{\sqrt{3} \cdot r}{2} & \text{c1} \end{bmatrix}$$

zeros() (Nullstellen)

Ausdrücken und/oder der *VarOderSchätzwert*-Liste umzuordnen.

Wenn Sie keine Schätzwerte angeben und ein Ausdruck in einer Variablen kein Polynom ist, aber alle Ausdrücke in ihren Unbekannten linear sind, so verwendet **zeros()** das Gaußsche Eliminationsverfahren beim Versuch, alle reellen Nullstellen zu bestimmen.

Wenn ein System weder in all seinen Variablen polynomial noch in seinen Unbekannten linear ist, dann bestimmt **zeros()** mindestens eine Nullstelle anhand eines iterativen Näherungsverfahrens. Hierzu muss die Anzahl der Unbekannten gleich der Ausdruckanzahl sein, und alle anderen Variablen in den Ausdrücken müssen zu Zahlen vereinfachbar sein.

Jede Unbekannte beginnt bei dem entsprechenden geschätzten Wert, falls vorhanden; ansonsten beginnt sie bei 0,0.

Suchen Sie anhand von Schätzwerten nach einzelnen zusätzlichen Nullstellen. Für Konvergenz sollte ein Schätzwert ziemlich nahe bei der Nullstelle liegen.

$$\text{zeros}\left(\left\{x+e^z \cdot y - 1, x - y - \sin(z)\right\}, \{x, y\}\right)$$

$$\begin{bmatrix} e^z \cdot \sin(z) + 1 & (\sin(z) - 1) \\ e^z + 1 & e^z + 1 \end{bmatrix}$$

$$\text{zeros}\left(\left\{e^z \cdot y - 1, y - \sin(z)\right\}, \{y, z\}\right)$$

$$\begin{bmatrix} 0.041458 & 3.18306 \\ 0.001871 & 6.28131 \\ 4.76 \cdot 10^{-11} & 1796.99 \\ 2. \cdot 10^{-13} & 254.469 \end{bmatrix}$$

$$\text{zeros}\left(\left\{e^z \cdot y - 1, y - \sin(z)\right\}, \{y, z=2 \cdot \pi\}\right)$$

$$\begin{bmatrix} 0.001871 & 6.28131 \end{bmatrix}$$

zInterval (z-Konfidenzintervall)

zInterval $\sigma, \text{Liste}[, \text{Häufigkeit}, KStufe]]$

(Datenlisteneingabe)

zInterval $\sigma, \bar{x}, n [, KStufe]$

(Zusammenfassende statistische Eingabe)

Berechnet ein *z*-Konfidenzintervall. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.CLower,	Konfidenzintervall für den unbekannten Populationsmittelwert

Ausgabeverable	Beschreibung
stat.CUpper	
stat. \bar{x}	Stichprobenmittelwert der Datenfolge aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.sx	Stichproben-Standardabweichung
stat.n	Länge der Datenfolge mit Stichprobenmittelwert
stat. σ	Bekannte Populations-Standardabweichung für Datenfolge <i>Liste</i>

zInterval_1Prop (z-Konfidenzintervall für eine Proportion)

Katalog > 

zInterval_1Prop $x, n [, KStufe]$

Berechnet ein z -Konfidenzintervall für eine Proportion. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

x ist eine nicht negative Ganzzahl.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

Ausgabeverable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat. \hat{p}	Die berechnete Erfolgsproportion
stat.ME	Fehlertoleranz
stat.n	Anzahl der Stichproben in Datenfolge

zInterval_2Prop (z-Konfidenzintervall für zwei Proportionen)

Katalog > 

zInterval_2Prop $x1, n1, x2, n2 [, KStufe]$

Berechnet das z -Konfidenzintervall für zwei Proportionen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

$x1$ und $x2$ sind nicht negative Ganzzahlen.

zInterval_2Prop (z-Konfidenzintervall für zwei Proportionen)

Katalog > 

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

AusgabevARIABLE	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat. \hat{p} Diff	Die geschätzte Differenz zwischen den Proportionen
stat.ME	Fehlertoleranz
stat. \hat{p}_1	Geschätzte erste Stichprobenproportion
stat. \hat{p}_2	Geschätzte zweite Stichprobenproportion
stat.n1	Stichprobenumfang in Datenfolge eins
stat.n2	Stichprobenumfang in Datenfolge zwei

zInterval_2Samp (z-Konfidenzintervall für zwei Stichproben)

Katalog > 

zInterval_2Samp $\sigma_1, \sigma_2, \text{Liste1}, \text{Liste2}, [\text{Häufigkeit1}, \text{Häufigkeit2}, [\text{KStufe}]]$

(Datenlisteneingabe)

zInterval_2Samp $\sigma_1, \sigma_2, \bar{x}_1, n1, \bar{x}_2, n2, [\text{KStufe}]$

(Zusammenfassende statistische Eingabe)

Berechnet ein z-Konfidenzintervall für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 252).

AusgabevARIABLE	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat. $\bar{x}_1 - \bar{x}_2$	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz

AusgabevARIABLE	Beschreibung
stat. \bar{x} 1, stat. \bar{x} 2	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat. σ x1, stat. σ x2	Stichproben-Standardabweichungen für <i>Liste 1</i> und <i>Liste 2</i>
stat.n1, stat.n2	Anzahl der Stichproben in Datenfolgen
stat.r1, stat.r2	Bekannte Populations-Standardabweichungen für Datenfolge <i>Liste 1</i> und <i>Liste 2</i>

zTest

Katalog > 

zTest $\mu_0, \sigma, Liste, [Häufigkeit, Hypoth]$

(Datenlisteneingabe)

zTest $\mu_0, \sigma, \bar{x}, n, [Hypoth]$

(Zusammenfassende statistische Eingabe)

Führt einen *z*-Test mit der Häufigkeit *Häufigkeitsliste* durch. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Getestet wird $H_0: \mu = \mu_0$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu < \mu_0$ setzen Sie *Hypoth<0*

Für $H_a: \mu \neq \mu_0$ (Standard) setzen Sie *Hypoth=0*

Für $H_a: \mu > \mu_0$ setzen Sie *Hypoth>0*

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.z	$(\bar{x} - \mu_0) / (\sigma / \sqrt{n})$
stat.P Value	Kleinste Wahrscheinlichkeit, bei der die Nullhypothese verworfen werden kann
stat. \bar{x}	Stichprobenmittelwert der Datenfolge in <i>Liste</i>
stat.sx	Stichproben-Standardabweichung der Datenfolge. Wird nur für <i>Dateneingabe</i> zurückgegeben.
stat.n	Stichprobenumfang

zTest_1Prop $p0,x,n[,Hypoth]$

Berechnet einen z -Test für eine Proportion. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Siehe Seite 186.)

x ist eine nicht negative Ganzzahl.

Getestet wird $H_0: p = p0$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: p > p0$ setzen Sie *Hypoth>0*

Für $H_a: p \neq p0$ (*Standard*) setzen Sie *Hypoth=0*

Für $H_a: p < p0$ setzen Sie *Hypoth<0*

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.p0	Hypothetische Populations-Standardabweichung
stat.z	Für die Proportion berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat. \hat{p}	Geschätzte Stichprobenproportion
stat.n	Stichprobenumfang

zTest_2Prop $x1,n1,x2,n2[,Hypoth]$

Berechnet einen z -Test für zwei Proportionen. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

$x1$ und $x2$ sind nicht negative Ganzahlen.

Getestet wird $H_0: p1 = p2$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: p1 > p2$ setzen Sie *Hypoth>0*

Für $H_a: p1 \neq p2$ (*Standard*) setzen Sie *Hypoth=0*

Für $H_a: p1 < p2$ setzen Sie *Hypoth<0*

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige)

Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.z	Für die Differenz der Proportionen berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat. \hat{p} 1	Geschätzte erste Stichprobenproportion
stat. \hat{p} 2	Geschätzte zweite Stichprobenproportion
stat. \hat{p}	Geschätzte verteilte Stichprobenproportion
stat.n1, stat.n2	Stichprobenanzahl in Versuchen 1 und 2

zTest_2Samp $\sigma_1, \sigma_2, \text{Liste1}, \text{Liste2}[\text{Häufigkeit1}, \text{Häufigkeit2}, \text{Hypoth}]$

(Datenlisteneingabe)

zTest_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2, \text{Hypoth}$

(Zusammenfassende statistische Eingabe)

Berechnet einen z-Test für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 186.)

Getestet wird $H_0: \mu_1 = \mu_2$ in Bezug auf eine der folgenden Alternativen:

Für $H_a: \mu_1 < \mu_2$ setzen Sie *Hypoth*<0

Für $H_a: \mu_1 \neq \mu_2$ (Standard) setzen Sie *Hypoth*=0

Für $H_a: \mu_1 > \mu_2$ setzen Sie *Hypoth*>0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 252).

AusgabevARIABLE	Beschreibung
stat.z	Für die Differenz der Mittelwerte berechneter Standardwert
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat. \bar{x} 1, stat. \bar{x} 2	Stichprobenmittelwerte der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in <i>Liste 1</i> und <i>Liste 2</i>

Ausgabeveriable	Beschreibung
stat.n1, stat.n2	Stichprobenumfang

Sonderzeichen

+ (addieren)

Ausdr1 + Ausdr2⇒Ausdruck

Gibt die Summe der beiden Argumente zurück.

Taste

56	56
56+4	60
60+4	64
64+4	68
68+4	72

Liste1 + Liste2⇒Liste

Matrix1 + Matrix2⇒Matrix

Gibt eine Liste (bzw. eine Matrix) zurück, die die Summen der entsprechenden Elemente von *Liste1* und *Liste2* (oder *Matrix1* und *Matrix2*) enthält.

Die Argumente müssen die gleiche Dimension besitzen.

Ausdr + Liste1⇒Liste

Liste1 + Ausdr⇒Liste

Gibt eine Liste zurück, die die Summen von *Ausdr* plus jedem Element der *Liste1* enthält.

Ausdr + Matrix1⇒Matrix

Matrix1 + Ausdr⇒Matrix

Gibt eine Matrix zurück, in der *Ausdr* zu jedem Element der Diagonalen von *Matrix1* addiert ist. *Matrix1* muss eine quadratische Matrix sein.

Hinweis: Verwenden Sie .+ (Punkt Plus) zum Addieren eines Ausdrucks zu jedem Element.

$\left\{ 22, \pi, \frac{\pi}{2} \right\} \rightarrow l1$	$\left\{ 22, \pi, \frac{\pi}{2} \right\}$
$\left\{ 10,5, \frac{\pi}{2} \right\} \rightarrow l2$	$\left\{ 10,5, \frac{\pi}{2} \right\}$
$l1+l2$	$\{ 32, \pi+5, \pi \}$
$Ans + \{ \pi, -5, -\pi \}$	$\{ \pi+32, \pi, 0 \}$
$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} a+1 & b \\ c & d+1 \end{bmatrix}$

$15 + \{ 10, 15, 20 \}$	$\{ 25, 30, 35 \}$
$\{ 10, 15, 20 \} + 15$	$\{ 25, 30, 35 \}$

-(subtrahieren)

Ausdr1 - Ausdr2⇒Ausdruck

Gibt *Ausdr1* minus *Ausdr2* zurück.

Taste

$6-2$	4
$\pi - \frac{\pi}{6}$	$\frac{5\pi}{6}$

- (subtrahieren) Taste*Liste1 - Liste2*⇒*Liste**Matrix1 - Matrix2*⇒*Matrix*

Subtrahiert die einzelnen Elemente aus *Liste2* (oder *Matrix2*) von denen in *Liste1* (oder *Matrix1*) und gibt die Ergebnisse zurück.

Die Argumente müssen die gleiche Dimension besitzen.

Ausdr - Liste1⇒*Liste**Liste1 - Ausdr*⇒*Liste*

Subtrahiert jedes Element der *Liste1* von *Ausdr* oder subtrahiert *Ausdr* von jedem Element der *Liste1* und gibt eine Liste der Ergebnisse zurück.

Ausdr - Matrix1⇒*Matrix**Matrix1 - Ausdr*⇒*Matrix*

Ausdr - Matrix1 gibt eine Matrix zurück, die *Ausdr* multipliziert mit der Einheitsmatrix minus *Matrix1* ist. *Matrix1* muss eine quadratische Matrix sein.

Matrix1 - Ausdr gibt eine Matrix zurück, die *Ausdr* multipliziert mit der Einheitsmatrix subtrahiert von *Matrix1* ist. *Matrix1* muss eine quadratische Matrix sein.

Hinweis: Verwenden Sie .- (Punkt Minus) zum Subtrahieren eines Ausdrucks von jedem Element.

$\left\{ 22, \pi, \frac{\pi}{2} \right\} - \left\{ 10,5, \frac{\pi}{2} \right\}$	$\{ 12, \pi - 5, 0 \}$
$\begin{bmatrix} 3 & 4 \end{bmatrix} - \begin{bmatrix} 1 & 2 \end{bmatrix}$	$\begin{bmatrix} 2 & 2 \end{bmatrix}$

$15 - \{ 10, 15, 20 \}$	$\{ 5, 0, -5 \}$
$\{ 10, 15, 20 \} - 15$	$\{ -5, 0, 5 \}$

$20 - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$
---	--

· (multiplizieren) Taste*Ausdr1•Ausdr2*⇒*Ausdruck*

Gibt das Produkt der beiden Argumente zurück.

Liste1•Liste2⇒*Liste*

Gibt eine Liste zurück, die die Produkte der entsprechenden Elementen aus *Liste1* und *Liste2* enthält.

$2 \cdot 3 \cdot 45$	6.9
----------------------	-----

$x \cdot y \cdot x$	$x^2 \cdot y$
---------------------	---------------

$\{ 1, 2, 3 \} \cdot \{ 4, 5, 6 \}$	$\{ 4, 10, 18 \}$
$\begin{bmatrix} 2 & 3 \\ a & 2 \end{bmatrix} \cdot \begin{bmatrix} a^2 & b \\ 2 & 3 \end{bmatrix}$	$\begin{bmatrix} 2 \cdot a, \frac{b}{2} \end{bmatrix}$

·(multiplizieren)

Taste

Die Listen müssen die gleiche Dimension besitzen.

Matrix1·*Matrix2*⇒*Matrix*

Gibt das Matrizenprodukt von *Matrix1* und *Matrix2* zurück.

Die Spaltenanzahl von *Matrix1* muss gleich die Zeilenanzahl von *Matrix2* sein.

Ausdr·*Liste1*⇒*Liste*

Liste1·*Ausdr*⇒*Liste*

Gibt eine Liste zurück, die die Produkte von *Ausdr* und jedem Element der *Liste1* enthält.

Ausdr·*Matrix1*⇒*Matrix*

Matrix1·*Ausdr*⇒*Matrix*

Gibt eine Matrix zurück, die die Produkte von *Ausdr* und jedem Element der *Matrix1* enthält.

Hinweis: Verwenden Sie . ·(Punkt-Multiplikation) zum Multiplizieren eines Ausdrucks mit jedem Element.

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix} = \begin{bmatrix} a+2\cdot b+3\cdot c & d+2\cdot e+3\cdot f \\ 4\cdot a+5\cdot b+6\cdot c & 4\cdot d+5\cdot e+6\cdot f \end{bmatrix}$$

$$\pi \cdot \{4, 5, 6\} = \{4 \cdot \pi, 5 \cdot \pi, 6 \cdot \pi\}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01 = \begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$$

$$1 \cdot \text{identity}(3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

/ (dividieren)

Taste

Ausdr1/*Ausdr2*⇒*Ausdruck*

Gibt *Ausdr1* dividiert durch *Ausdr2* zurück.

Hinweis: Siehe auch **Vorlage Bruch**, Seite 5.

$$\frac{2}{3.45} = 0.57971$$

$$\frac{x^3}{x} = x^2$$

$$\frac{\{1, 2, 3\}}{\{4, 5, 6\}} = \left\{ \frac{1}{2}, \frac{2}{3}, \frac{1}{2} \right\}$$

Liste1/*Liste2*⇒*Liste*

Gibt eine Liste der Elemente von *Liste1* dividiert durch *Liste2* zurück.

Die Listen müssen die gleiche Dimension besitzen.

Ausdr / *Liste1* ⇒ *Liste*

Liste1 / *Ausdr* ⇒ *Liste*

Gibt eine Liste der Elemente von *Ausdr*

$$\frac{a}{\{3, a, \sqrt{a}\}} = \left\{ \frac{a}{3}, 1, \sqrt{a} \right\}$$

$$\frac{\{a, b, c\}}{a \cdot b \cdot c} = \left\{ \frac{1}{b \cdot c}, \frac{1}{a \cdot c}, \frac{1}{a \cdot b} \right\}$$

/ (dividieren)

 Taste

dividiert durch *Liste1* oder *Liste1* dividiert durch *Ausdr* zurück.

Matrix1 / *Ausdr* \Rightarrow *Matrix*

Gibt eine Matrix zurück, die die Quotienten *Matrix1* / *Ausdr* enthält.

Hinweis: Verwenden Sie . / (Punkt-Division) zum Dividieren eines Ausdrucks durch jedes Element.

$$\frac{\begin{bmatrix} a & b & c \end{bmatrix}}{a \cdot b \cdot c} = \begin{bmatrix} \frac{1}{b \cdot c} & \frac{1}{a \cdot c} & \frac{1}{a \cdot b} \end{bmatrix}$$

\wedge (Potenz)

 Taste

Ausdr1 \wedge *Ausdr2* \Rightarrow *Ausdruck*

Liste1 \wedge *Liste2* \Rightarrow *Liste*

Gibt das erste Argument hoch dem zweiten Argument zurück.

Hinweis: Siehe auch **Vorlage Exponent**, Seite 5.

Bei einer Liste wird jedes Element aus *Liste1* hoch dem entsprechenden Element aus *Liste2* zurückgegeben.

Im reellen Bereich benutzen Bruchpotenzen mit gekürztem ungeradem Nenner den reellen statt den Hauptzeig im komplexen Modus.

Ausdr \wedge *Liste1* \Rightarrow *Liste*

Gibt *Ausdr* hoch den Elementen von *Liste1* zurück.

Liste1 \wedge *Ausdr* \Rightarrow *Liste*

Gibt die Elemente von *Liste1* hoch *Ausdr* zurück.

$$4^2 = 16$$

$$\left\{ a, 2, c \right\}^{\{1, b, 3\}} = \left\{ a, 2^b, c^3 \right\}$$

$$p^{\{a, 2, -3\}} = \left\{ p^a, p^2, \frac{1}{p^3} \right\}$$

$$\left\{ 1, 2, 3, 4 \right\}^{-2} = \left\{ 1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16} \right\}$$

\wedge (Potenz)

Taste

Quadratmatrix1 \wedge *Ganzzahl* \Rightarrow Matrix

Gibt *Quadratmatrix1* hoch *Ganzzahl* zurück.

Quadratmatrix1 muss eine quadratische Matrix sein.

Ist *Ganzzahl* = -1, wird die inverse Matrix berechnet.

Ist *Ganzzahl* < -1, wird die inverse Matrix hoch der entsprechenden positiven Zahl berechnet.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2$	$\begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1}$	$\begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$
$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2}$	$\begin{bmatrix} \frac{11}{4} & -\frac{5}{4} \\ 2 & 2 \\ -\frac{15}{4} & \frac{7}{4} \end{bmatrix}$

x^2 (Quadrat)

Taste

Ausdr1 2 \Rightarrow *Ausdruck*

Gibt das Quadrat des Arguments zurück.

Liste1 2 \Rightarrow *Liste*

Gibt eine Liste zurück, die die Produkte der Elemente in *Liste1* enthält.

Quadratmatrix1 2 \Rightarrow Matrix

Gibt das Matriz-Quadrat von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Quadrats jedes einzelnen Elements. Verwenden Sie $.^2$, um das Quadrat jedes einzelnen Elements zu berechnen.

4^2	16
$\{2,4,6\}^2$	$\{4,16,36\}$
$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2$	$\begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$
$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix} .^2$	$\begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$

.+ (Punkt-Addition)

Tasten

Matrix1 .+ *Matrix2* \Rightarrow Matrix

Ausdr .+ *Matrix1* \Rightarrow Matrix

Matrix1 .+ *Matrix2* gibt eine Matrix zurück, die Summe jedes Elementpaars von *Matrix1* und *Matrix2* ist.

Ausdr .+ *Matrix1* gibt eine Matrix zurück, die die Summe von Ausdruck und jedem Element von *Matrix1* ist.

$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} a+c & 6 \\ b+5 & d+3 \end{bmatrix}$
$x .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\begin{bmatrix} x+c & x+4 \\ x+5 & x+d \end{bmatrix}$

. - (Punkt-Subt.)

Tasten

Matrix1 . - Matrix2 \Rightarrow Matrix

Ausdr . - Matrix1 \Rightarrow Matrix

Matrix1 . - Matrix2 gibt eine Matrix zurück,
die die Differenz jedes Elementpaars von
Matrix1 und *Matrix2* ist.

Ausdr . - Matrix1 gibt eine Matrix zurück,
die die Differenz von *Ausdr* und jedem
Element von *Matrix1* ist.

$$\begin{array}{c|cc|cc} & \boxed{a} & \boxed{2} & \boxed{-} & \boxed{c} & \boxed{4} \\ \hline & \boxed{b} & \boxed{3} & & \boxed{d} & \boxed{5} \end{array} \quad \begin{array}{c|cc} & \boxed{a-c} & \boxed{-2} \\ \hline & \boxed{b-d} & \boxed{-2} \end{array}$$

$$x . - \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \quad \begin{bmatrix} x-c & x-4 \\ x-d & x-5 \end{bmatrix}$$

. · (Punkt-Mult.)

Tasten

Matrix1 . · Matrix2 \Rightarrow Matrix

Ausdr . · Matrix1 \Rightarrow Matrix

Matrix1 . · Matrix2 gibt eine Matrix zurück,
die das Produkt jedes Elementpaars von
Matrix1 und *Matrix2* ist.

Ausdr . · Matrix1 gibt eine Matrix zurück,
die das Produkt von *Ausdr* und jedem
Element von *Matrix1* ist.

$$\begin{array}{c|cc|cc} & \boxed{a} & \boxed{2} & \cdot & \boxed{c} & \boxed{4} \\ \hline & \boxed{b} & \boxed{3} & & \boxed{5} & \boxed{d} \end{array} \quad \begin{array}{c|cc} & \boxed{a \cdot c} & \boxed{8} \\ \hline & \boxed{5 \cdot b} & \boxed{3 \cdot d} \end{array}$$

$$x . \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \begin{bmatrix} a \cdot x & b \cdot x \\ c \cdot x & d \cdot x \end{bmatrix}$$

. / (Punkt-Division)

Tasten

Matrix1 . / Matrix2 \Rightarrow Matrix

Ausdr . / Matrix1 \Rightarrow Matrix

Matrix1 . / Matrix2 gibt eine Matrix
zurück, die der Quotient jedes
Elementpaars von *Matrix1* und *Matrix2* ist.

Ausdr . / Matrix1 gibt eine Matrix zurück,
die der Quotient von *Ausdr* und jedem
Element von *Matrix1* ist.

$$\begin{array}{c|cc|cc} & \boxed{a} & \boxed{2} & \diagup & \boxed{c} & \boxed{4} \\ \hline & \boxed{b} & \boxed{3} & & \boxed{5} & \boxed{d} \end{array} \quad \begin{array}{c|cc} & \boxed{a} & \boxed{1} \\ \hline & \boxed{c} & \boxed{2} \\ & \boxed{b} & \boxed{3} \\ \hline & \boxed{5} & \boxed{d} \end{array}$$

$$x . \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \quad \begin{bmatrix} x & x \\ c & 4 \\ \frac{x}{5} & \frac{x}{d} \end{bmatrix}$$

.^ (Punkt-Potenz)

. ^ Tasten

Matrix1 \wedge *Matrix2* \Rightarrow *Matrix*

$$\begin{array}{c} \left[\begin{matrix} a & 2 \\ b & 3 \end{matrix} \right] \cdot \begin{matrix} c & 4 \\ 5 & d \end{matrix} \\ \hline x \cdot \begin{matrix} c & 4 \\ 5 & d \end{matrix} \end{array} \quad \begin{array}{c} \left[\begin{matrix} a^c & 16 \\ b^5 & 3d \end{matrix} \right] \\ \hline \begin{matrix} x^c & x^4 \\ x^5 & xd \end{matrix} \end{array}$$

Ausdr \wedge *MatrixI* \Rightarrow *Matrix*

Matrix1 \wedge *Matrix2* gibt eine Matrix zurück, in der jedes Element aus *Matrix2* Exponent des entsprechenden Elements aus *Matrix1* ist.

Ausdr .ⁿ *Matrix1* gibt eine Matrix zurück, in der jedes Element aus *Matrix1* Exponent von *Ausdr* ist.

- (Negation)

(-) Taste

$\neg Ausdr1 \Rightarrow Ausdruck$

-2.43 -2.43

-Liste1 \Rightarrow Liste

$$-\{-1, 0, 4, 1, 2\} \times 10^9 \quad \{1, -0, 4, -1, 2\} \times 10^9$$

-Matrix \downarrow *Matrix*

$\neg q : \neg b$ $q : b$

Gibt die Negation des Arguments zurück.

Bei einer Liste oder Matrix werden alle Elemente negiert zurückgegeben.

Im Bin-Modus:

Ist das Argument eine binäre oder hexadezimale ganze Zahl, ergibt die Negation das Zweierkomplement.

Wichtig: Null, nicht Buchstabe O

Um das ganze Ergebnis zu sehen, drücken Sie **▲** und verwenden dann **◀** und **▶**, um den Cursor zu bewegen.

% (Prozent)

ctrl **Tasten**

Ausdr1 % \Rightarrow Ausdruck

Liste 1 % \Rightarrow *Liste*

Matrix1 % \Rightarrow *Matrix*

Hinweis: Erzwingen eines Näherungsergebnisses.

Handheld: Drücken Sie **ctrl** **enter**.

Windows®: Drücken Sie Strg+Eingabetaste.

Macintosh®: Drücken ⌘+Eingabetaste.

iPad®: Halten Sie die **Eingabetaste** gedrückt

Ergibt 100

% (Prozent)

Tasten

Bei einer Liste oder einer Matrix wird eine Liste/Matrix zurückgegeben, in der jedes Element durch 100 dividiert ist.

und wählen Sie aus.

13%

0.13

$\{\{1,10,100\}\}\%$

{0.01,0.1,1.}

= (gleich)

Taste

Ausdr1 = Ausdr2 \Rightarrow Boolescher Ausdruck

Liste1 = Liste2 \Rightarrow Boolesche Liste

Matrix1 = Matrix2 \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn Ausdr1 bei Auswertung gleich Ausdr2 ist.

Gibt falsch zurück, wenn Ausdr1 bei Auswertung ungleich Ausdr2 ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis zum Eingeben des Beispiels:

Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

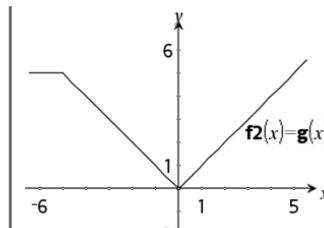
Beispiefunktion mit den mathematischen Vergleichssymbolen: $=, \neq, <, \leq, >, \geq$

Define $g(x) = \text{Func}$

```
If  $x \leq -5$  Then
  Return 5
Elseif  $x > -5$  and  $x < 0$  Then
  Return  $-x$ 
Elseif  $x \geq 0$  and  $x \neq 10$  Then
  Return  $x$ 
Elseif  $x = 10$  Then
  Return 3
EndIf
EndFunc
```

Done

Ergebnis der graphischen Darstellung $g(x)$



f2(x) = g(x)

\neq (ungleich)

Tasten

Ausdr1 \neq Ausdr2 \Rightarrow Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

Liste1 \neq Liste2 \Rightarrow Boolesche Liste

\neq (ungleich)

ctrl **=** Tasten

$Matrix1 \neq Matrix2 \Rightarrow Boolesche\ Matrix$

Gibt wahr zurück, wenn $Ausdr1$ bei Auswertung ungleich $Ausdr2$ ist.

Gibt falsch zurück, wenn $Ausdr1$ bei Auswertung gleich $Ausdr2$ ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie $/=$ eintippen

$<$ (kleiner als)

ctrl **=** Tasten

$Ausdr1 < Ausdr2 \Rightarrow Boolescher\ Ausdruck$

Siehe Beispiel bei " $=$ " (gleich).

$Liste1 < Liste2 \Rightarrow Boolesche\ Liste$

$Matrix1 < Matrix2 \Rightarrow Boolesche\ Matrix$

Gibt wahr zurück, wenn $Ausdr1$ bei Auswertung kleiner als $Ausdr2$ ist.

Gibt falsch zurück, wenn $Ausdr1$ bei Auswertung größer oder gleich $Ausdr2$ ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

\leq (kleiner oder gleich)

ctrl **=** Tasten

$Ausdr1 \leq Ausdr2 \Rightarrow Boolescher\ Ausdruck$

Siehe Beispiel bei " $=$ " (gleich).

$Liste1 \leq Liste2 \Rightarrow Boolesche\ Liste$

$Matrix1 \leq Matrix2 \Rightarrow Boolesche\ Matrix$

Gibt wahr zurück, wenn $Ausdr1$ bei Auswertung kleiner oder gleich $Ausdr2$ ist.

Gibt falsch zurück, wenn $Ausdr1$ bei Auswertung

\leq (kleiner oder gleich)

ctrl **=** Tasten

größer als *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel **<=**

$>$ (größer als)

ctrl **=** Tasten

Ausdr1 > Ausdr2 \Rightarrow Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

Liste1 > Liste2 \Rightarrow Boolesche Liste

Matrix1 > Matrix2 \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung größer als *Ausdr2* ist.

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung kleiner oder gleich *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

\geq (größer oder gleich)

ctrl **=** Tasten

Ausdr1 \geq Ausdr2 \Rightarrow Boolescher Ausdruck

Siehe Beispiel bei “=” (gleich).

Liste1 \geq Liste2 \Rightarrow Boolesche Liste

Matrix1 \geq Matrix2 \Rightarrow Boolesche Matrix

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung größer oder gleich *Ausdr2* ist.

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung kleiner oder gleich *Ausdr2* ist.

In allen anderen Fällen wird eine vereinfachte Form der Gleichung zurückgegeben.

Bei Listen und Matrizen werden die Ergebnisse des

\geq (größer oder gleich)

ctrl = Tasten

Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel \geq =

\Rightarrow (logische Implikation)

ctrl = Tasten

BoolescherAusdr1 \Rightarrow BoolescherAusdr2
ergibt Boolescher Ausdruck

BoolescheListe1 \Rightarrow BoolescheListe2
ergibt Boolesche Liste

BoolescheMatrix1 \Rightarrow BoolescheMatrix2
ergibt Boolesche Matrix

Ganzzahl1 \Rightarrow Ganzzahl2 ergibt Ganzzahl

5>3 or 3>5	true
5>3 \Rightarrow 3>5	false
3 or 4	7
3 \Rightarrow 4	-4
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} \Rightarrow {3,2,1}	{-1,-1,-3}

Wertet den Ausdruck **not** <Argument1> **or** <Argument2> aus und gibt „wahr“, „falsch“ oder eine vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel \Rightarrow =

\Leftrightarrow (logische doppelte Implikation, XNOR)

ctrl = Tasten

BoolescherAusdr1 \Leftrightarrow BoolescherAusdr2
ergibt Boolescher Ausdruck

BoolescheListe1 \Leftrightarrow BoolescheListe2
ergibt Boolesche Liste

BoolescheMatrix1 \Leftrightarrow BoolescheMatrix2
ergibt Boolesche Matrix

Ganzzahl1 \Leftrightarrow Ganzzahl2 ergibt Ganzzahl

5>3 xor 3>5	true
5>3 \Leftrightarrow 3>5	false
3 xor 4	7
3 \Leftrightarrow 4	-8
{1,2,3} xor {3,2,1}	{2,0,2}
{1,2,3} \Leftrightarrow {3,2,1}	{-3,-1,-3}

\Leftrightarrow (logische doppelte Implikation, XNOR)

ctrl = Tasten

Gibt die Negation einer **XOR** booleschen Operation auf beiden Argumenten zurück.
Gibt „wahr“, „falsch“ oder eine vereinfachte Form des Arguments zurück.

Bei Listen und Matrizen werden die Ergebnisse des Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie $<=>$ drücken

! (Fakultät)

?!▶ Taste

Ausdr1! \Rightarrow Ausdruck

$5!$ 120

Liste1! \Rightarrow Liste

$\{\{5,4,3\}\}!$ {120,24,6}

Matrix1! \Rightarrow Matrix

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}!$ $\begin{bmatrix} 1 & 2 \\ 6 & 24 \end{bmatrix}$

Gibt die Fakultät des Arguments zurück.

Bei Listen und Matrizen wird eine Liste/Matrix mit der Fakultät der einzelnen Elemente zurückgegeben.

&

/k Tasten

String1 & String2 \Rightarrow String

"Hello " & "Nick" "Hello Nick"

Gibt einen String zurück, der durch Anfügen von *String2* an *String1* gebildet wurde.

d() (Ableitung)

Katalog >

$d(Ausdr1, Var[, Ordnung]) \Rightarrow$ Ausdruck

$\frac{d}{dx}(f(x) \cdot g(x))$ $\frac{d}{dx}(f(x)) \cdot g(x) + \frac{d}{dx}(g(x)) \cdot f(x)$

$d(Liste1, Var[, Ordnung]) \Rightarrow$ Liste

$\frac{d}{dy} \left(\frac{d}{dx}(x^2 \cdot y^3) \right)$ $6 \cdot y^2 \cdot x$

$d(Matrix1, Var[, Ordnung]) \Rightarrow$ Matrix

$\frac{d}{dx} \left(\{x^2, x^3, x^4\} \right)$ $\{2 \cdot x, 3 \cdot x^2, 4 \cdot x^3\}$

Gibt die erste Ableitung des ersten Arguments bezüglich der Variablen *Var* zurück.

Ordnung (sofern angegeben) muss eine

ganze Zahl sein. Ist die Ordnung kleiner als Null, ist das Ergebnis eine Anti-Ableitung (Integration).

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **derivative (...)** eintippen.

d() folgt nicht dem normalen Auswertungsmechanismus, seine Argumente vollständig zu vereinfachen und dann die Funktionsdefinition auf diese vollständig vereinfachten Argumente anzuwenden. Stattdessen führt **d()** die folgenden Schritte aus:

1. Vereinfachung des zweiten Arguments nur so weit, dass es nicht zu einer Nichtvariablen führt.
2. Vereinfachung des ersten Arguments nur so weit, dass es jeden gespeicherten Wert für die in Schritt 1 bestimmte Variable neu aufruft.
3. Bestimmung der symbolischen Ableitung des Ergebnisses von Schritt 2 bezüglich der Variablen aus Schritt 1.

Wenn die Variable aus Schritt 1 einen gespeicherten Wert oder einen Wert hat, der durch den womit-Operator („|“) spezifiziert ist, wird dieser Wert im Ergebnis aus Schritt 3 ersetzt.

Hinweis: Siehe auch **Erste Ableitung**, Seite 9; **Zweite Ableitung**, Seite 10; und **n-te Ableitung**, Seite 10.

$\int(Ausdr1, Var[, Untere, Obere]) \Rightarrow Ausdruck$

$\int(Ausdr1, Var[, Konstante]) \Rightarrow Ausdruck$

Gibt das Integral von *Ausdr1* bezüglich der Variablen *Var* von *Untere* bis *Obere* zurück.

$$\int_a^b x^2 dx = \frac{b^3}{3} - \frac{a^3}{3}$$

Hinweis: Siehe auch **Vorlage Bestimmtes Integral** und **Vorlage Unbestimmtes Integral**, Seite 10.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **Integral (...)** eintippen.

Gibt ein unbestimmtes Integral zurück, wenn *UntGreenze* und *ObGreenze* nicht angegeben werden. Eine symbolische Integrationskonstante wird weggelassen, sofern Sie nicht das Argument *Konstante* einfügen.

Gleichwertig gültige unbestimmte Integrale können durch eine numerische Konstante voneinander abweichen. Eine solche Konstante kann verborgen sein - insbesondere, wenn ein unbestimmtes Integral logarithmische oder inverse trigonometrische Funktionen enthält. Außerdem werden manchmal stückweise konstante Ausdrücke hinzugefügt, um einem unbestimmten Integral über ein größeres Intervall Gültigkeit zu verleihen als bei der üblichen Formel.

$\int()$ gibt sich selbst zurück bei Stücken von *AusdrI*, die es nicht als explizite endliche Kombination seiner integrierten Funktionen und Operatoren bestimmen kann.

Sind sowohl *UntGreenze* als auch *ObGreenze* angegeben, wird versucht, Unstetigkeiten oder unstetige Ableitungen im Intervall *UntGreenze* < *Var* < *ObGreenze* zu finden, um das Intervall an diesen Stellen unterteilen zu können.

Ist der Modus **Auto oder Näherung** auf Auto eingestellt, wird eine numerische Integration vorgenommen, wo dies möglich ist, wenn kein unbestimmtes Integral oder kein Grenzwert ermittelt werden kann.

Bei der Einstellung **Approximiert** wird die numerische Integration, wo möglich, zuerst versucht. Unbestimmte Integrale werden nur dann gesucht, wenn die numerische

$$\begin{aligned} \int x^2 dx & \frac{x^3}{3} \\ \int(a \cdot x^2, x, c) & \frac{a \cdot x^3}{3} + c \end{aligned}$$

$$\int b \cdot e^{-x^2} + \frac{a}{x^2+a^2} dx \quad b \cdot \int e^{-x^2} dx + \tan^{-1}\left(\frac{x}{a}\right)$$

Hinweis: Erzwingen eines Näherungsergebnisses,

$\int()$ (Integral)

Katalog > 

Integration unzulässig ist oder fehlschlägt.

Handheld: Drücken Sie **ctrl** **enter**.

Windows®: Drücken Sie **Strg+Eingabetaste**.

Macintosh®: Drücken **⌘+Eingabetaste**.

iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie  aus.

$$\int_{-1}^1 e^{-x^2} dx \quad 1.49365$$

$\int()$ können verschachtelt werden, um Mehrfach-Integrale zu bearbeiten. Die Integrationsgrenzen können von außerhalb liegenden Integrationsvariablen abhängen.

Hinweis: Siehe auch **nInt()**, Seite 129.

$$\int_0^a \int_0^x \ln(x+y) dy dx$$

$$\frac{a^2 \cdot \ln(a)}{2} + \frac{a^2 \cdot (4 \cdot \ln(2) - 3)}{4}$$

$\sqrt()$ (Quadratwurzel)

ctrl **x²** **Tasten**

$\sqrt(AusdrI) \Rightarrow Ausdruck$

$$\sqrt{4} \quad 2$$

$\sqrt(ListeI) \Rightarrow Liste$

$$\sqrt{\{9,a,4\}} \quad \{3,\sqrt{a},2\}$$

Gibt die Quadratwurzel des Arguments zurück.

Bei einer Liste wird die Quadratwurzel für jedes Element von *ListeI* zurückgegeben.

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **sqrt(...)** **eintippen**.

Hinweis: Siehe auch **Vorlage Quadratwurzel**, Seite 5.

$\Pi()$ (ProdSeq)

Katalog >

$\Pi(Ausdr1, Var, Von, Bis) \Rightarrow Ausdruck$

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **prodSeq**(...) eintippen.

Wertet *Ausdr1* für jeden Wert von *Var* zwischen *Von* und *Bis* aus und gibt das Produkt der Ergebnisse zurück.

Hinweis: Siehe auch **Vorlage Produkt** (Π), Seite 9.

$\Pi(Ausdr1, Var, Von, Von-1) \Rightarrow 1$

$\Pi(Ausdr1, Var, Von, Bis) \Rightarrow 1/\Pi(Ausdr1, Var, Bis+1, Von-1)$ if *Bis* < *Von-1*

Die verwendeten Produktformeln wurden ausgehend von der folgenden Quelle entwickelt:

Ronald L. Graham, Donald E. Knuth, Oren Patashnik: *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley 1994.

$$\frac{\prod_{n=1}^5 \left(\frac{1}{n} \right)}{120}$$

$$\frac{\prod_{k=1}^n (k^2)}{(n!)^2}$$

$$\frac{\prod_{n=1}^5 \left(\left\{ \frac{1}{n}, n, 2 \right\} \right)}{\left\{ \frac{1}{120}, 120, 32 \right\}}$$

$$\frac{\prod_{k=4}^3 (k)}{1}$$

$$\frac{\prod_{k=4}^1 \left(\frac{1}{k} \right)}{6}$$

$$\frac{\prod_{k=4}^1 \left(\frac{1}{k} \right) \cdot \prod_{k=2}^4 \left(\frac{1}{k} \right)}{\frac{1}{4}}$$

$\Sigma()$ (SumSeq)

Katalog >

$\Sigma(Ausdr1, Var, Von, Bis) \Rightarrow Ausdruck$

Hinweis: Sie können diese Funktion über die Tastatur Ihres Computers eingeben, indem Sie **sumSeq**(...) eintippen.

Wertet *Ausdr1* für jeden Wert von *Var* zwischen *Von* und *Bis* aus und gibt die Summe der Ergebnisse zurück.

Hinweis: Siehe auch **Vorlage Summe**, Seite 9.

$$\frac{\sum_{n=1}^5 \left(\frac{1}{n} \right)}{60}$$

$$\frac{\sum_{k=1}^n (k^2)}{\frac{n \cdot (n+1) \cdot (2 \cdot n+1)}{6}}$$

$$\frac{\sum_{n=1}^{\infty} \left(\frac{1}{n^2} \right)}{\frac{\pi^2}{6}}$$

$\Sigma()$ (SumSeq)

Katalog >

$$\Sigma(Ausdr1, Var, Von, Von-I) \Rightarrow 0$$

$$\Sigma(Ausdr1, Var, Von, Bis) \Rightarrow -\Sigma(Ausdr1, Var, Bis+1, Von-1) \text{ if } Bis < Von-I$$

$$\sum_{k=4}^3 \{k\}$$

0

Die verwendeten Summenformeln wurden ausgehend von der folgenden Quelle entwickelt:

Ronald L. Graham, Donald E. Knuth, Oren Patashnik: *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley 1994.

$$\begin{aligned} &\sum_{k=4}^1 \{k\} \\ &\sum_{k=4}^1 \{k\} + \sum_{k=2}^4 \{k\} \end{aligned}$$

-5

4

$\Sigma\text{Int}()$

Katalog >

$$\Sigma\text{Int}(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [WertRunden]) \Rightarrow Wert$$

$$\Sigma\text{Int}(1, 3, 12, 4.75, 20000, 12, 12) \quad -213.48$$

$$\Sigma\text{Int}(NPmt1, NPmt2, AmortTabelle) \Rightarrow Wert$$

Amortisationsfunktion, die die Summe der Zinsen innerhalb eines angegebenen Zahlungsbereichs berechnet.

$NPmt1$ und $NPmt2$ definieren Anfang und Ende des Zahlungsbereichs.

$N, I, PV, Pmt, FV, PpY, CpY$ und $PmtAt$ werden in der TVM-Argumentetabelle (Seite 206) beschrieben.

- Wenn Sie Pmt nicht angeben, wird standardmäßig $Pmt=\text{tvmPmt}$ ($N, I, PV, FV, PpY, CpY, PmtAt$) eingesetzt.
- Wenn Sie FV nicht angeben, wird standardmäßig $FV=0$ eingesetzt.
- Die Standardwerte für PpY, CpY und $PmtAt$ sind dieselben wie bei den TVM-Funktionen.

$WertRunden$ legt die Anzahl der Dezimalstellen für das Runden fest. Standard=2.

$$tbl:=\text{amortTbl}(12, 12, 4.75, 20000, 12, 12)$$

0	0.	0.	20000.
1	-77.49	-1632.43	18367.6
2	-71.17	-1638.75	16728.8
3	-64.82	-1645.1	15083.7
4	-58.44	-1651.48	13432.2
5	-52.05	-1657.87	11774.4
6	-45.62	-1664.3	10110.1
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02

$$\Sigma\text{Int}(1, 3, tbl) \quad -213.48$$

$\Sigma\text{Int}(NPmt1, NPmt2, AmortTable)$
 berechnet die Summe der Zinsen auf der Grundlage der Amortisationstabelle *AmortTabelle*. Das Argument *AmortTabelle* muss eine Matrix in der unter **amortTbl()**, Seite 12, beschriebenen Form sein.

Hinweis: Siehe auch $\Sigma\text{Prn}()$ auf dieser und **Bal()**, Seite 21.

 $\Sigma\text{Prn}()$

$\Sigma\text{Prn}(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [WertRunden]) \Rightarrow Wert$

$\Sigma\text{Prn}(NPmt1, NPmt2, AmortTable) \Rightarrow Wert$

Amortisationsfunktion, die die Summe der Tilgungszahlungen innerhalb eines angegebenen Zahlungsbereichs berechnet.

NPmt1 und *NPmt2* definieren Anfang und Ende des Zahlungsbereichs.

N, I, PV, Pmt, FV, PpY, CpY und *PmtAt* werden in der TVM-Argumentetabelle (Seite 206) beschrieben.

- Wenn Sie *Pmt* nicht angeben, wird standardmäßig *Pmt=tvmPmt* (*N,I,PV,FV,PpY,CpY,PmtAt*) eingesetzt.
- Wenn Sie *FV* nicht angeben, wird standardmäßig *FV=0* eingesetzt.
- Die Standardwerte für *PpY*, *CpY* und *PmtAt* sind dieselben wie bei den TVM-Funktionen.

WertRunden legt die Anzahl der Dezimalstellen für das Runden fest. Standard=2.

$\Sigma\text{Prn}(NPmt1, NPmt2, AmortTable)$
 berechnet die Summe der gezahlten Tilgungsbeträge auf der Grundlage der Amortisationstabelle *AmortTabelle*. Das Argument *AmortTabelle* muss eine Matrix

$\Sigma\text{Prn}(1,3,12,4.75,20000,,12,12)$ -4916.28

tbl:=amortTbl(12,12,4.75,20000,,12,12)

0	0.	0.	20000.
1	-77.49	-1632.43	18367.57
2	-71.17	-1638.75	16728.82
3	-64.82	-1645.1	15083.72
4	-58.44	-1651.48	13432.24
5	-52.05	-1657.87	11774.37
6	-45.62	-1664.3	10110.07
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02

$\Sigma\text{Prn}(1,3,tbl)$ -4916.28

in der unter `amortTbl()`, Seite 12,
beschriebenen Form sein.

Hinweis: Siehe auch `\SigmaInt()` auf dieser und
`Bal()`, Seite 21.

(Umleitung)

`# varNameString`

Greift auf die Variable namens `VarNameString` zu. So können Sie innerhalb einer Funktion Variablen unter Verwendung von Strings erzeugen.

  Tasten

`#("x"&"y"&"z")` `xyz`

Erzeugt oder greift auf die Variable `xyz` zu.

<code>10→r</code>	10
<code>"r"→s1</code>	<code>"r"</code>
<code>#s1</code>	10

Gibt den Wert der Variable (`r`) zurück,
dessen Name in Variable `s1` gespeichert ist.

E (Wissenschaftliche Schreibweise)

`MantisseEEExponent`

Gibt eine Zahl in wissenschaftlicher Schreibweise ein. Die Zahl wird als `Mantisse × 10Exponent` interpretiert.

 Taste

23000.	23000.
2300000000.+4.1e15	4.1e15
3·10 ⁴	30000

Tipp: Wenn Sie eine Potenz von 10 eingeben möchten, ohne ein Dezimalwertergebnis zu verursachen, verwenden Sie `10^Ganzzahl`.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie `@E` eintippen. Tippen Sie zum Beispiel `2 . 30@E4` ein, um `2.3E4` einzugeben.

g (Neugrad)

`AusdrIg⇒Ausdruck`

`AusdrIg⇒Ausdruck`

 Taste

Im Grad-, Neugrad- oder Bogenmaß-Modus:

`Listelg⇒Liste`

g (Neugrad)

Taste

Matrix 1g⇒*Matrix*

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Bogenmaß-Modus einen Winkel in Neugrad anzugeben.

$$\begin{array}{ll} \cos(50^\circ) & \frac{\sqrt{2}}{2} \\ \cos(\{0, 100^\circ, 200^\circ\}) & \{1, 0, -1\} \end{array}$$

Im Winkelmodus Bogenmaß wird *Ausdr1* mit $\pi/200$ multipliziert.

Im Winkelmodus Grad wird *Ausdr1* mit $g/100$ multipliziert.

Im Neugrad-Modus wird *Ausdr1* unverändert zurückgegeben.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie @g eintippen.

r (Bogenmaß)

Taste

Ausdr1r⇒*Ausdruck*

Liste 1r⇒*Liste*

Matrix 1r⇒*Matrix*

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Neugrad-Modus einen Winkel im Bogenmaß anzugeben.

Im Winkelmodus Grad, Neugrad oder Bogenmaß:

$$\begin{array}{ll} \cos\left(\frac{\pi}{4^r}\right) & \frac{\sqrt{2}}{2} \\ \cos\left(\left\{0^\circ, \frac{\pi}{12}, -(\pi)^r\right\}\right) & \left\{1, \frac{(\sqrt{3}+1)\cdot\sqrt{2}}{4}, -1\right\} \end{array}$$

Im Winkelmodus Grad wird das Argument mit $180/\pi$ multipliziert.

Im Winkelmodus Bogenmaß wird das Argument unverändert zurückgegeben.

Im Neugrad-Modus wird das Argument mit $200/\pi$ multipliziert.

Tipp: Verwenden Sie r in einer Funktionsdefinition, wenn Sie bei Ausführung der Funktion das Bogenmaß frei von der Winkelmoduseinstellung erzwingen möchten.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie @x eintippen.

° (Grad)

Taste

Ausdr l° ⇒Ausdruck

Liste l° ⇒Liste

Matrix l° ⇒Matrix

Diese Funktion gibt Ihnen die Möglichkeit, im Neugrad- oder Bogenmaß-Modus einen Winkel in Grad anzugeben.

Im Winkelmodus Bogenmaß wird das Argument mit $\pi/180$ multipliziert.

Im Winkelmodus Grad wird das Argument unverändert zurückgegeben.

Im Winkelmodus Neugrad wird das Argument mit $10/9$ multipliziert.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie @d eintippen.

Im Winkelmodus Grad, Neugrad oder Bogenmaß:

$$\cos(45^{\circ}) = \frac{\sqrt{2}}{2}$$

Im Winkelmodus Bogenmaß:

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie **ctrl enter**.

Windows®: Drücken Sie **Strg+Eingabetaste**.

Macintosh®: Drücken **⌘+Eingabetaste**.

iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie **≈** aus.

$$\cos\left(\left\{0, \frac{\pi}{4}, 90^{\circ}, 30.12^{\circ}\right\}\right) = \{1., 0.707107, 0., 0.864976\}$$

°, ', " (Grad/Minute/Sekunde)

ctrl **Tasten**

dd°mm'mm.ss"⇒Ausdruck

Im Grad-Modus:

daEine positive oder negative Zahl

$$25^{\circ}13'17.5'' = 25.2215$$

mmEine nicht negative Zahl

$$25^{\circ}30' = \frac{51}{2}$$

ss.ssEine nicht negative Zahl

Gibt $dd+(mm/60)+(ss.ss/3600)$ zurück.

Mit einer solchen Eingabe auf der 60er-Basis können Sie:

- Einen Winkel unabhängig vom aktuellen Winkelmodus in Grad/Minuten/Sekunden eingeben.
- Uhrzeitangaben in Stunden/Minuten/Sekunden vornehmen.

Hinweis: Nach ss.ss werden zwei Apostrophe ('') gesetzt, kein Anführungszeichen (").

\angle (Winkel)

ctrl Tasten

[Radius, \angle Winkel] \Rightarrow Vektor

(Eingabe polar)

[Radius, \angle Winkel, Z Koordinate]

\Rightarrow Vektor

(Eingabe zylindrisch)

[Radius, \angle Winkel, \angle Winkel] \Rightarrow Vektor

(Eingabe sphärisch)

Gibt Koordinaten als Vektor zurück, wobei die aktuelle Einstellung für Vektorformat gilt: kartesisch, zylindrisch oder sphärisch.

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie @ eintippen.

Im Bogenmaß-Modus mit Vektorformat eingestellt auf:

kartesisch

$$\left[5 \ \angle 60^\circ \ \angle 45^\circ \right] \quad \left[\frac{5\sqrt{2}}{4} \ \frac{5\sqrt{6}}{4} \ \frac{5\sqrt{2}}{2} \right]$$

zylindrisch

$$\left[5 \ \angle 60^\circ \ \angle 45^\circ \right] \quad \left[\frac{5\sqrt{2}}{2} \ \angle \frac{\pi}{3} \ \frac{5\sqrt{2}}{2} \right]$$

sphärisch

$$\left[5 \ \angle 60^\circ \ \angle 45^\circ \right] \quad \left[5 \ \angle \frac{\pi}{3} \ \angle \frac{\pi}{4} \right]$$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$5+3 \cdot i \cdot \left(10 \angle \frac{\pi}{4} \right) \quad 5-5 \cdot \sqrt{2} + (3-5 \cdot \sqrt{2}) \cdot i$$

Hinweis: Erzwingen eines Näherungsergebnisses,

Handheld: Drücken Sie **ctrl enter**.

Windows®: Drücken Sie **Strg+Eingabetaste**.

Macintosh®: Drücken **⌘+Eingabetaste**.

iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie **\approx** aus.

$$5+3 \cdot i \cdot \left(10 \angle \frac{\pi}{4} \right) \quad -2.07107 - 4.07107 \cdot i$$

' (Ableitungsstrich)

Variable '

Variable ''

Gibt in einer Differentialgleichung einen Ableitungsstrich ein. Ein Ableitungsstrich kennzeichnet eine Differentialgleichung erster Ordnung, zwei Ableitungsstriche kennzeichnen eine Differentialgleichung zweiter Ordnung usw.

$$\text{deSolve} \left(y'' = y^{\frac{-1}{2}} \text{ and } y(0)=0 \text{ and } y'(0)=0, t, y \right)$$

$$\frac{3}{2 \cdot y^{\frac{1}{4}}} = t$$

_ (Unterstrich als leeres Element)

Siehe "Leere (ungültige) Elemente", Seite 252.

_ (Unterstrich als Einheiten-Bezeichner)

ctrl Tasten

Ausdr_Einheit

Kennzeichnet die Einheiten für einen *Ausdr.*. Alle Einheitennamen müssen mit einem Unterstrich beginnen.

Sie können entweder vordefinierte Einheiten verwenden oder Ihre eigenen erstellen. Eine Liste vordefinierter Einheiten finden Sie im Katalog auf der Registerkarte Einheiten-Konversion (Unit Conversions). Sie können Einheitennamen aus dem Katalog auswählen oder sie direkt eingeben.

Variable_

Besitzt *Variable* keinen Wert, so wird sie behandelt, als würde sie eine komplexe Zahl darstellen. Die Variable wird ohne das Zeichen *_* standardmäßig als reell behandelt.

Besitzt *Variable* einen Wert, so wird das Zeichen *_* ignoriert und *Variable* behält ihren ursprünglichen Datentyp bei.

Hinweis: Eine komplexe Zahl kann ohne

3 · _m ► _ft 9.84252 · _ft

Hinweis: Das Umrechnungssymbol ► können Sie im Katalog finden. Klicken Sie auf und dann auf **Mathematische Operatoren**.

z sei undefiniert:

real(z)	<i>z</i>
real(z_)	real(z_)
imag(z)	0
imag(z_)	imag(z_)

_ (Unterstrich als Einheiten-Bezeichner)

Tasten

Unterstrich _ in Variablen gespeichert werden. Bei Berechnungen wie **cSolve()** und **cZeros()** empfiehlt sich allerdings die Verwendung von _ um beste Ergebnisse zu erzielen.

► (konvertieren)

Tasten

Ausdr_Einheit1 ► _Einheit2⇒Ausdr_Einheit2

3·_m►_ft

9.84252·_ft

Konvertiert einen Ausdruck von einer Einheit in eine andere.

Der Unterstrich _ kennzeichnet die Einheiten. Diese Einheiten müssen sich in derselben Kategorie befinden, z.B. Länge oder Fläche

Eine Liste vordefinierter Einheiten finden Sie im Katalog auf der Registerkarte Einheiten-Konversion (Unit Conversions):

- Sie können einen Einheitennamen aus der Liste auswählen.
- Sie können den Konversionsoperator, ►, vom Listenanfang verwenden.

Sie können die Einheitennamen auch manuell eingeben. Um bei der Eingabe von Einheitennamen auf dem Handheld "_" einzugeben, drücken Sie .

Hinweis: Verwenden Sie zum Konvertieren von Temperatureinheiten **tmpCnv()** und **ΔtmpCnv()**. Der Konvertierungsoperator ► ist nicht für Temperatureinheiten anwendbar.

10^()

Katalog >

10^(Ausdr1)⇒Ausdruck

$10^{1.5}$ 31.6228

10^(Liste1)⇒Liste

$10^{\{0,-2,2,a\}}$ $\left\{1,\frac{1}{100},100,10^a\right\}$

Gibt 10 hoch Argument zurück.

Bei einer Liste wird 10 hoch jedem Element von *Liste1* zurückgegeben.

10^(Quadratmatrix I) \Rightarrow Quadratmatrix

Ergibt 10 hoch *Quadratmatrix I*. Dies ist nicht gleichbedeutend mit der Berechnung von 10 hoch jedem Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

$$\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1.14336\text{e}7 & 8.17155\text{e}6 & 6.67589\text{e}6 \\ 9.95651\text{e}6 & 7.11587\text{e}6 & 5.81342\text{e}6 \\ 7.65298\text{e}6 & 5.46952\text{e}6 & 4.46845\text{e}6 \end{bmatrix}$$

Quadratmatrix I muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

^-1(Kehrwert)

Ausdr1 **^-1** \Rightarrow Ausdruck

$$(3.1)^{-1} \quad 0.322581$$

Listel **^-1** \Rightarrow Liste

$$\left\{ a, 4, -0.1, x, -2 \right\}^{-1} \quad \left\{ \frac{1}{a}, \frac{1}{4}, -10, \frac{1}{x}, \frac{-1}{2} \right\}$$

Gibt den Kehrwert des Arguments zurück.

Bei einer Liste wird für jedes Element von *Listel* der Kehrwert zurückgegeben.

Quadratmatrix I **^-1** \Rightarrow *Quadratmatrix*

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \quad \begin{bmatrix} -2 & 1 \\ 3 & -1 \\ 2 & 2 \end{bmatrix}$$

Gibt die Inverse von *Quadratmatrix I* zurück.

Quadratmatrix I muss eine nicht-singuläre quadratische Matrix sein.

$$\begin{bmatrix} 1 & 2 \\ a & 4 \end{bmatrix}^{-1} \quad \begin{bmatrix} -2 & 1 \\ \frac{a}{a-2} & \frac{-1}{a-2} \\ \frac{a}{2 \cdot (a-2)} & \frac{-1}{2 \cdot (a-2)} \end{bmatrix}$$

| (womit-Operator)

Ausdr | BoolescherAusdr1

$$x+1|x=3 \quad 4$$

[andBoolescherAusdr2]...

$$x+y|x=\sin(y) \quad \sin(y)+y$$

Ausdr | BoolescherAusdr1

$$[orBoolescherAusdr2]...$$

Das womit-Symbol („|“) dient als binärer Operator. Der Operand links von | ist ein Ausdruck. Der Operand rechts von | gibt eine oder mehrere Relationen an, die auf die Vereinfachung des Ausdrucks einwirken sollen. Bei Angabe mehrerer Relationen nach dem | sind diese jeweils mit logischen „and“ oder „or“ Operatoren miteinander zu verketten.

$$x+1|x=3$$

$$x+y|x=\sin(y)$$

$$x+y|\sin(y)=x$$

4

$\sin(y)+y$

$x+y$

Der womit-Operator erfüllt drei

Grundaufgaben:

- Ersetzung
- Intervallbeschränkung
- Ausschließung

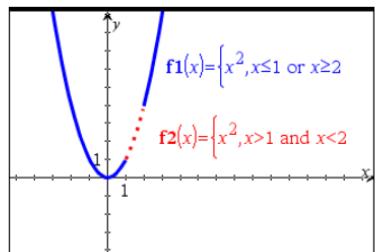
Ersetzungen werden in Form einer Gleichung angegeben, wie etwa $x=3$ oder $y=\sin(x)$. Am wirksamsten ist eine Ersetzung, wenn die linke eine einfache Variable ist. Ausdr | Variable = Wert bewirkt, dass jedes Mal, wenn Variable in Ausdr vorkommt, Wert ersetzt wird.

Intervallbeschränkungen werden in Form einer oder mehrerer mit logischen „and“ oder „or“ Operatoren verknüpfte Ungleichungen angegeben.

Intervallbeschränkungen ermöglichen auch Vereinfachungen, die andernfalls ungültig oder nicht berechenbar wären.

$x^3 - 2 \cdot x + 7 \rightarrow f(x)$	Done
$f(x) x=\sqrt{3}$	$\sqrt{3} + 7$
$(\sin(x))^2 + 2 \cdot \sin(x) - 6 \sin(x)=d$	$d^2 + 2 \cdot d - 6$

solve($x^2 - 1 = 0, x$) $ x > 0 \text{ and } x < 2$	$x=1$
$\sqrt{x} \cdot \sqrt{\frac{1}{x}} > 0$	1
$\sqrt{x} \cdot \sqrt{\frac{1}{x}}$	$\sqrt{\frac{1}{x}} \cdot \sqrt{x}$



Ausschließungen verwenden den relationalen Operator „ungleich“ ($/=$ oder \neq), um einen bestimmten Wert bei der Operation auszuschließen. Sie dienen hauptsächlich zum Ausschließen einer exakten Lösung bei Verwendung von **cSolve()**, **cZeros()**, **fMax()**, **fMin()**, **solve()**, **zeros()** usw.

solve($x^2 - 1 = 0, x$) $ x \neq 1$	$x=-1$
--	--------

→ (speichern)

ctrl

var

Taste*Ausdr → Var*

$$\frac{\pi}{4} \rightarrow myvar \quad \frac{\pi}{4}$$

Liste → Var

$$2 \cdot \cos(x) \rightarrow yI(x) \quad Done$$

Matrix → Var

$$\{1,2,3,4\} \rightarrow lst5 \quad \{1,2,3,4\}$$

Expr → Funktion(Param1,...)

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \rightarrow matg \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

List → Funktion(Param1,...)

$$"Hello" \rightarrow str1 \quad "Hello"$$

Matrix → Funktion(Param1,...)

**Wenn Variable *Var* noch nicht existiert,
wird *Var* erzeugt und auf *Ausdr*, *Liste* oder
Matrix initialisiert.**

Wenn *Var* existiert und nicht gesperrt oder
geschützt ist, wird der Variableninhalt
durch *Ausdr*, *Liste* oder *Matrix* ersetzt.

Tipp: Wenn Sie symbolische Rechnungen
mit undefinierten Variablen vornehmen
möchten, sollten Sie vermeiden, Werte in
Variablen mit häufig benutzten
Einzeichennamen abzuspeichern (etwa den
Variablen a, b, c, x, y, z usw.).

Hinweis: Sie können diesen Operator über
die Tastatur Ihres Computers eingeben,
indem Sie das Tastenkürzel **=:** eintippen.
Geben Sie zum Beispiel **pi/4 =: myvar**
ein.

:= (zuweisen)

ctrl

var

Tasten*Var := Ausdr*

$$myvar := \frac{\pi}{4} \quad \frac{\pi}{4}$$

Var := Liste

$$yI(x) := 2 \cdot \cos(x) \quad Done$$

Var := Matrix

$$lst5 := \{1,2,3,4\} \quad \{1,2,3,4\}$$

Function(Param1,...) := Ausdr

$$matg := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Function(Param1,...) := Liste

$$str1 := "Hello" \quad "Hello"$$

Function(Param1,...) := Matrix

Wenn Variable *Var* noch nicht existiert,
wird *Var* erzeugt und auf *Ausdr*, *Liste* oder
Matrix initialisiert.

:= (zuweisen)

Tasten

Wenn *Var* existiert und nicht gesperrt oder geschützt ist, wird der Variableninhalt durch *Ausdr*, *Liste* bzw. *Matrix* ersetzt.

Tipp: Wenn Sie symbolische Rechnungen mit undefinierten Variablen vornehmen möchten, sollten Sie vermeiden, Werte in Variablen mit häufig benutzten Einzeichennamen abzuspeichern (etwa den Variablen a, b, c, x, y, z usw.).

© (Kommentar)

Tasten

© [Text]

© verarbeitet *Text* als Kommentarzeile und ermöglicht so die Eingabe von Anmerkungen zu von Ihnen erstellten Funktionen und Programmen.

© kann an den Zeilenanfang oder an eine beliebige Stelle der Zeile gesetzt werden. Alles, was rechts von © bis zum Zeilenende steht, gilt als Kommentar.

Hinweis zum Eingeben des Beispiels:
Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Define $g(n) = \text{Func}$

© Declare variables
Local i, result
result := 0
For i, 1, n, 1 ©Loop n times
result := result + i²
EndFor
Return result
EndFunc

Done

$g(3)$

14

0b, 0h

0 B Tasten, 0 H Tasten

0b binäre_Zahl

Im Dec-Modus:

0h hexadezimale_Zahl

0b10+0hF+10 27

Kennzeichnet eine Dual- bzw. Hexadezimalzahl. Zur Eingabe einer Dual- oder Hexadezimalzahl muss unabhängig vom jeweiligen Basis-Modus das Präfix 0b bzw. 0h verwendet werden. Eine Zahl ohne Präfix wird als dezimal behandelt (Basis 10).

Die Ergebnisse werden im jeweiligen Basis-Modus angezeigt.

Im Bin-Modus:

0b10+0hF+10 0b11011

Im Hex-Modus:

0b, 0h

0 [B] Tasten, 0 [H] Tasten

0b10+0hF+10

0h1B

Leere (ungültige) Elemente

Bei der Analyse von Daten der realen Welt liegt möglicherweise nicht immer ein vollständiger Datensatz vor. TI-Nspire™ CAS lässt leere bzw. ungültige Datenelemente zu, sodass Sie mit den nahezu vollständigen Daten fortfahren können anstatt von vorn anfangen oder unvollständige Fälle verwerfen zu müssen.

Ein Beispiel für Daten mit leeren Elementen finden Sie im Kapitel Lists & Spreadsheet unter „*Tabellendaten grafisch darstellen*“.

Mit der Funktion **delVoid()** können Sie leere Elemente aus einer Liste löschen. Die Funktion **isVoid()** sucht nach leeren Elementen. Einzelheiten finden Sie unter **delVoid()**, Seite 57, und **isVoid()**, Seite 99.

Hinweis: Um ein leeres Element manuell in einen mathematischen Ausdruck einzugeben, geben Sie „_“ oder das Schlüsselwort **void** ein. Das Schlüsselwort **void** wird bei der Auswertung des Ausdrucks automatisch in das Symbol „_“ konvertiert. Um „_“ auf dem Handheld einzugeben, drücken Sie **ctrl** **enter**.

Kalkulationen mit ungültigen Elementen

Bei der Mehrzahl aller Kalkulationen, die ein ungültiges Element enthalten, wird das Ergebnis ebenfalls ungültig sein.
Sonderfälle sind nachstehend aufgeführt.

$\lfloor _\rfloor$	=
$\text{gcd}(100, _)$	=
$3 + _$	=
$\{5, _, 10\} - \{3, 6, 9\}$	$\{2, _, 1\}$

Listenargumente, die ungültige Elemente enthalten

Die folgenden Funktionen und Befehle ignorieren (überspringen) ungültige Elemente, die in Listenargumenten gefunden werden.

count, countIf, cumulativeSum, freqTable>list, frequency, max, mean, median, product, stDevPop, stDevSamp, sum, sumIf, varPop und varSamp sowie Regressionskalkulationen, **OneVar**, **TwoVar** und **FiveNumSummary** Statistiken, Konfidenzintervalle und statistische Tests

$\text{sum}(\{2, _, 3, 5, 6, 6\})$	16.6
$\text{median}(\{1, 2, _, _, _, 3\})$	2
$\text{cumulativeSum}(\{1, 2, _, 4, 5\})$	$\{1, 3, _, 7, 12\}$
$\text{cumulativeSum}\begin{bmatrix} 1 & 2 \\ 3 & _ \\ 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 \\ 4 & _ \\ 9 & 8 \end{bmatrix}$

Listenargumente, die ungültige Elemente enthalten

SortA und **SortD** verschieben alle ungültigen Elemente im ersten Argument nach unten.

$\{5,4,3,_,1\} \rightarrow list1$	$\{5,4,3,_,1\}$
$\{5,4,3,2,1\} \rightarrow list2$	$\{5,4,3,2,1\}$
SortA $list1, list2$	Done
$list1$	$\{1,3,4,5,_\}$
$list2$	$\{1,3,4,5,2\}$

In Regressionen sorgt ein ungültiges Element in einer Liste X oder Y dafür, dass auch das entsprechende Element im Residuum ungültig ist.

$\{1,2,3,_,5\} \rightarrow list1$	$\{1,2,3,_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD $list1, list2$	Done
$list1$	$\{5,3,2,1,_\}$
$list2$	$\{5,3,2,1,4\}$

$l1:=\{1,2,3,4,5\}; l2:=\{2,_,3,5,6,6\}$	$\{2,_,3,5,6,6\}$
LinRegMx $l1, l2$	Done
$stat.Resid$	$\{0.434286,_, -0.862857, -0.011429, 0.44\}$
$stat.XReg$	$\{1,_, 3, 4, 5, _\}$
$stat.YReg$	$\{2,_, 3, 5, 6, 6\}$
$stat.FreqReg$	$\{1,_, 1, 1, 1, _\}$

Eine ausgelassene Kategorie in Regressionen sorgt dafür, dass das entsprechende Element im Residuum ungültig ist.

$l1:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
$cat:=\{"M", "M", "F", "F"\}; incl:=\{"F"\}$	$\{"F"\}$
LinRegMx $l1, l2, 1, cat, incl$	Done
$stat.Resid$	$\{_, _, 0, 0, _\}$
$stat.XReg$	$\{_, _, 4, 5, _\}$
$stat.YReg$	$\{_, _, 5, 6, 6\}$
$stat.FreqReg$	$\{_, _, 1, 1, _\}$

Listenargumente, die ungültige Elemente enthalten

Eine Häufigkeit von 0 in Regressionen führt dazu, dass das entsprechende Element im Residuum ungültig ist.

$I1 := \{1, 3, 4, 5\}; I2 := \{2, 3, 5, 6, 6\}$	$\{2, 3, 5, 6, 6\}$
LinRegMx $I1, I2, \{1, 0, 1, 1\}$	Done
stat.Resid	$\{0.069231, \dots, -0.276923, 0.207692\}$
stat.XReg	$\{1, \dots, 4, 5\}$
stat.YReg	$\{2, \dots, 5, 6, 6\}$
stat.FreqReg	$\{1, \dots, 1, 1\}$

Tastenkürzel zum Eingeben mathematischer Ausdrücke

Tastenkürzel ermöglichen es Ihnen, Elemente mathematischer Ausdrücke über die Tastatur einzugeben anstatt über den Katalog oder die Sonderzeichenpalette. Um beispielsweise den Ausdruck $\sqrt{6}$ einzugeben, können Sie **sqrt(6)** in die Eingabezeile eingeben. Wenn Sie **enter** drücken, ändert sich der Ausdruck **sqrt(6)** in $\sqrt{6}$. Einige Tastenkürzel sind sowohl für die Eingabe über das Handheld als auch über die Computertastatur nützlich. Andere sind hauptsächlich für die Computertastatur hilfreich.

Von Handheld oder Computertastatur

Sonderzeichen:	Tastenkürzel:
π	pi
θ	theta
∞	infinity
\leq	<=
\geq	>=
\neq	/=
\Rightarrow (logische Implikation)	=>
\Leftrightarrow (logische doppelte Implikation, XNOR)	<=>
\rightarrow (Operator speichern)	=:
$ $ (Absolutwert)	abs(...)
$\sqrt()$	sqrt(...)
d()	derivative(...)
f()	integral(...)
$\Sigma()$ (Vorlage Summe)	sumSeq(...)
$\Pi()$ (Vorlage Produkt)	prodSeq(...)
$\sin^{-1}(), \cos^{-1}(), \dots$	arcsin(...), arccos(...), ...
Δ Liste()	deltaList(...)
Δ tmpCnv()	deltaTmpCnv(...)

Von der Computertastatur

Sonderzeichen:	Tastenkürzel:
$c1, c2, \dots$ (Konstanten)	@c1, @c2, ...
$n1, n2, \dots$ (ganzzahlige Konstanten)	@n1, @n2, ...

Sonderzeichen:	Tastenkürzel:
i (imaginäre Konstante)	@i
e (natürlicher Logarithmus zur Basis e)	@e
E (wissenschaftliche Schreibweise)	@E
T (Transponierte)	@t
r (Bogenmaß)	@r
$^{\circ}$ (Grad)	@d
g (Neugrad)	@g
\angle (Winkel)	@<
\blacktriangleright (Umwandlung)	@>
►Decimal, ►approxFraction() usw.	@>Decimal, @>approxFraction() usw.

Auswertungsreihenfolge in EOS™ (Equation Operating System)

Dieser Abschnitt beschreibt das Equation Operating System (EOS™), das von der TI-Nspire™ CAS Technologie genutzt wird. Zahlen, Variablen und Funktionen werden in einer einfachen Abfolge eingegeben. Die EOS™ Software wertet Ausdrücke und Gleichungen anhand der gesetzten Klammern und der im Folgenden beschriebenen Priorität der Operatoren aus.

Auswertungsreihenfolge

Ebene	Operator
1	Klammern: rund (), eckig [], geschweift { }
2	Umleitung (#)
3	Funktionsaufrufe
4	Postfix-Operatoren: Grad-Minuten-Sekunden ($^{\circ}, ", !$), Fakultät (!), Prozent (%), Bogenmaß (Γ), Tiefstellen ([]), Transponieren (T)
5	Potenzieren, Potenzoperator (^)
6	Negation (-)
7	Stringverkettung (&)
8	Multiplikation (\cdot), Division (/)
9	Addition (+), Subtraktion (-)
10	Gleichheitsbeziehungen: gleich (=), ungleich (\neq oder $/=$), kleiner als (<), kleiner oder gleich (\leq oder $<=$), größer als (>), größer oder gleich (\geq oder $>=$)
11	Logisches Nicht: not
12	Logische Konjunktion: and
13	Logisch or
14	xor, nor, nand
15	logische Implikation, (\Rightarrow)
16	Logische doppelte Implikation, XNOR (\Leftrightarrow)
17	womit-Operator („ “)
18	Speichern (\rightarrow)

Klammern (rund, eckig, geschweift)

Alle Berechnungen, die in Klammern – runde, eckige oder geschweifte – gesetzt sind, werden als erste ausgewertet. Ein Beispiel: Im Ausdruck $4(1+2)$ wertet die EOS™ Software zunächst $1+2$ aus, da dieser Teil des Ausdrucks in Klammern steht. Das Ergebnis 3 wird dann mit 4 multipliziert.

Die Anzahl der öffnenden und schließenden Klammern eines jeden Typs muss innerhalb eines Ausdrucks oder einer Gleichung jeweils übereinstimmen. Andernfalls wird eine Fehlermeldung mit dem fehlenden Element angezeigt. Beim Ausdruck $(1+2)/(3+4$ erscheint beispielsweise die Fehlermeldung „) fehlt“.

Hinweis: In der TI-Nspire™ CAS Software können Sie Ihre eigenen Funktionen definieren. Daher wird eine Variable, auf die ein Ausdruck in Klammern folgt, als Funktionsaufruf und nicht wie sonst implizit als Multiplikation interpretiert. Der Ausdruck $a(b+c)$ steht beispielsweise für den Wert der Funktion a mit dem Argument $b+c$. Um den Ausdruck $b+c$ mit der Variablen a zu multiplizieren, verwenden Sie die explizite Multiplikation: $a*(b+c)$.

Umleitung

Der Umleitungsoperator # wandelt eine Zeichenfolge (String) in einen Variablen- oder Funktionsnamen um. Mit #("x"&"y"&"z") wird beispielsweise der Variablenname xyz erstellt. Mithilfe der Umleitung können Sie auch Variablen aus einem Programm heraus erstellen und modifizieren. Beispiel: Wenn $10 \rightarrow r$ und $"r" \rightarrow s1$, dann $#s1=10$.

Postfix-Operatoren

Postfix-Operatoren sind Operatoren, die direkt nach einem Argument stehen, zum Beispiel 5!, 25% oder $60^{\circ}15'45''$. Argumente, auf die ein Postfix-Operator folgt, werden auf der vierten Prioritätsebene ausgewertet. Beispiel: Im Ausdruck $4^3!$ wird zuerst 3! ausgewertet. Das Ergebnis 6 wird dann als Exponent für 4 verwendet, und das Endergebnis ist 4096.

Potenz

Potenzen (^) und elementweise Potenzen (.^) werden von rechts nach links ausgewertet. Der Ausdruck 2^3^2 wird zum Beispiel wie $2^{(3^2)}$ ausgewertet, hat also das Ergebnis 512. Er unterscheidet sich damit vom Ausdruck $(2^3)^2$ mit dem Ergebnis 64.

Negation

Zum Eingeben einer negativen Zahl drücken Sie [(-)] und geben dann die Zahl ein. Postfix-Operatoren und Potenzen werden vor der Negation ausgewertet. Das Ergebnis von $-x^2$ ist zum Beispiel eine negative Zahl; $-9^2 = -81$. Um eine negative Zahl zu quadrieren, verwenden Sie Klammern: $(-9)^2$, Ergebnis 81.

Einschränkung („|“)

Das Argument nach dem womit-Operator „|“ stellt eine Reihe von Einschränkungen dar, die beeinflussen, wie das Argument vor dem Operator ausgewertet wird.

Fehlercodes und -meldungen

Wenn ein Fehler auftritt, wird sein Code der Variablen *errCode* zugewiesen. Benutzerdefinierte Programme und Funktionen können *errCode* auswerten, um die Ursache eines Fehlers zu bestimmen. Ein Beispiel für die Benutzung von *errCode* finden Sie als Beispiel 2 unter dem Befehl **Versuche (Try)** (Seite 202).

Hinweis: Einigen Fehlerbedingungen gelten nur für TI-Nspire™ CAS Produkte, andere gelten nur für TI-Nspire™ Produkte.

Fehlercode	Beschreibung
10	Funktion ergab keinen Wert
20	Test ergab nicht WAHR oder FALSCH. Generell können nicht definierte Variablen nicht verglichen werden. Beispielsweise würde der Test 'if a<b' diesen Fehler auslösen, wenn entweder a oder b zum Zeitpunkt der Ausführung der If-Anweisung nicht definiert ist.
30	Argument darf kein Verzeichnisname sein.
40	Argumentfehler
50	Argumente passen nicht Zwei oder mehr Argumente müssen vom gleichen Typ sein.
60	Argument muss Boolescher Ausdruck oder ganze Zahl sein
70	Argument muss Dezimalzahl sein
90	Argument muss Liste sein
100	Argument muss Matrix sein
130	Argument muss String sein
140	Argument muss Variablenname sein. Vergewissern Sie sich, dass der Name: <ul style="list-style-type: none">• nicht mit einer Ziffer beginnt• keine Leerzeichen oder Sonderzeichen enthält• keine unzulässigen Unterstriche oder Punkte enthält• die maximale Zeichenlänge nicht überschreitet Weitere Einzelheiten finden Sie im Abschnitt Calculator in der Dokumentation.
160	Argument muss Ausdruck sein
165	Batteriespannung zu niedrig zum Senden/Empfangen Setzten Sie vor dem Senden oder Empfangen neue Batterien ein.
170	Grenze

Fehlercode	Beschreibung
	Um das Suchintervall zu definieren, muss die untere Grenze kleiner sein als die obere Grenze.
180	Abbruch Die Taste esc oder on wurde gedrückt, während eine lange Berechnung oder ein Programm ausgeführt wurde.
190	Zirkuläre Definition Diese Meldung wird angezeigt, um zu verhindern, dass durch unendliches Ersetzen von Variablenwerten bei der Vereinfachung der Platz im Hauptspeicher nicht ausreicht. Dieser Fehler wird beispielsweise durch 'a+1->a' ausgelöst, wenn a eine nicht definierte Variable ist.
200	Zusammengesetzter Ausdruck ungültig Diese Fehlermeldung würde zum Beispiel durch 'solve(3x^2-4=0,x) x<0 or x>5' ausgelöst werden, weil die Einschränkung durch "oder (or)" anstatt "und (and)" getrennt wird.
210	Ungültiger Datentyp Ein Argument weist einen falschen Datentyp auf.
220	Abhängiger Grenzwert
230	Dimension Ein Listen- oder Matrixindex ist ungültig. Wenn beispielsweise die Liste {1,2,3,4} in L1 gespeichert wird, ist L1[5] ein Dimensionsfehler, weil L1 nur vier Elemente enthält.
235	Dimensionsfehler. Nicht genügend Elemente in den Listen.
240	Dimensionsfehler Zwei oder mehr Argumente müssen die gleiche Dimension haben. So ist beispielsweise [1,2]+[1,2,3] ein Dimensionsfehler, weil die Matrizen eine unterschiedliche Anzahl von Elementen enthalten.
250	Division durch Null
260	Bereichsfehler Ein Argument muss in einem festgelegten Bereich sein. rand(0) ist zum Beispiel nicht gültig.
270	Variablename doppelt vergeben
280	Else und Elself außerhalb If..EndIf-Block ungültig
290	Zu EndTry fehlt passende Else-Anweisung
295	Zu viele Iterationen

Fehlercode	Beschreibung
300	2- oder 3-elementige Liste bzw. Matrix erwartet
310	Das erste Argument von nSolve muss eine Gleichung in einer einzigen Variablen sein. Es darf keine andere Variable ohne Wert außer der interessierenden Variablen enthalten.
320	1. Argument von Löse oder cLöse muss Gleichung/Ungleichung sein Löse($3x-4, x$) ist beispielsweise ungültig, weil das erste Argument keine Gleichung ist.
345	Einheiten passen nicht zusammen
350	Index nicht im gültigen Bereich
360	Umleitungs-String kein gültiger Variablenname
380	Undefinierte Antw Entweder hat die vorangegangene Berechnung keine Antw (Ans) erzeugt oder es fand keine vorangegangene Berechnung statt.
390	Zuweisung ungültig
400	Zuweisungswert ungültig
410	Befehl ungültig
430	Ungültig für aktuelle Modus-Einstellungen
435	Schätzwert ungültig
440	Implizierte Multiplikation ungültig Beispielsweise ist ' $x(x+1)$ ' ungültig, während ' $x*(x+1)$ ' eine korrekte Syntax ist. So wird eine Verwechslung zwischen impliziter Multiplikation und Funktionsaufrufen vermieden.
450	In Funktion oder aktuellem Ausdruck ungültig In einer benutzerdefinierten Funktion sind nur bestimmte Befehle zulässig.
490	In Try..EndTry Block ungültig
510	Liste oder Matrix ungültig
550	Ungültig außerhalb Funktion oder Programm Einige Befehle sind nur in einer Funktion oder einem Programm gültig. Beispielsweise kann Lokal (Local) nur in einer Funktion oder einem Programm verwendet werden.
560	Nur in Loop..EndLoop-, For..EndFor- oder While..EndWhile-Block gültig Beispielsweise ist der Befehl Abbruch (Exit) nur in diesen Schleifenblöcken gültig.
565	Nur in einem Programm gültig

Fehlercode	Beschreibung
570	Ungültiger Pfadname \var ist beispielsweise ungültig.
575	Polarkomplex ungültig
580	Programmaufruf ungültig Programme können nicht innerhalb von Funktionen oder Ausdrücken wie z.B. '1+p(x)' aufgerufen werden, wenn p ein Programm ist.
600	Tabelle ungültig
605	Verwendung der Einheiten ungültig
610	Variablenname in Lokal-Anweisung ungültig
620	Variablen- bzw. Funktionsname ungültig
630	Variablenverweis ungültig
640	Vektorsyntax ungültig
650	Kabelübertragung gestört Eine Übertragung zwischen zwei Geräten wurde nicht abgeschlossen. Überprüfen Sie, dass das Kabel an beiden Seiten fest angeschlossen ist.
665	Diagonalisierung der Matrix nicht möglich
670	Wenig Speicher 1. Löschen Sie Daten in diesem Dokument 2. Speichern und schließen Sie dieses Dokument Wenn 1 und 2 fehlschlagen, nehmen Sie die Batterien heraus und setzen Sie sie wieder ein
672	Ressourcenauslastung
673	Ressourcenauslastung
680	fehlt (
690	fehlt)
700	fehlt "
710	fehlt]
720	fehlt }
730	Anfang oder Ende des Blocks fehlt
740	Then im If..EndIf-Block fehlt

Fehlercode	Beschreibung
750	Name verweist nicht auf Funktion oder Programm
765	Keine Funktionen ausgewählt
780	Keine Lösung gefunden
800	Nicht-reelles Ergebnis Wenn die Software beispielsweise in der Einstellung Reell (Real) ist, ist $\sqrt{(-1)}$ ungültig. Um komplexe Berechnungen zu ermöglichen, ändern Sie die Moduseinstellung 'Reell oder Komplex' (Real or Complex) in KARTESISCH (RECTANGULAR) oder POLAR (POLAR).
830	Überlauf
850	Programm nicht gefunden Ein Programmverweis in einem anderen Programm wurde während der Ausführung im angegebenen Pfad nicht gefunden.
855	Zufallsfunktionen sind im Graphikmodus nicht zulässig
860	Rekursion zu tief
870	Reservierter Name oder Systemvariable
900	Argumentfehler Das Median-Median-Modell konnte nicht auf den Datensatz angewendet werden.
910	Syntaxfehler
920	Text nicht gefunden
930	Zu wenig Argumente Der Funktion oder dem Befehl fehlen ein oder mehr Argumente.
940	Zu viele Argumente Der Ausdruck oder die Gleichung enthält eine überschüssige Anzahl von Argumenten und kann nicht ausgewertet werden.
950	Zu viele Indizierungen
955	Zu viele undefinierte Variable
960	Variable ist nicht definiert Der Variablen wurde kein Wert zugewiesen. Verwenden Sie einen der folgenden Befehle: <ul style="list-style-type: none">• sto →• :=• Definiere

Fehlercode	Beschreibung
	um Variablen Werte zuzuweisen.
965	Betriebssystem nicht lizenziert
970	Variable ist aktiv, daher keine Verweise oder Änderungen zulässig
980	Variable ist geschützt
990	Ungültiger Variablenname Stellen Sie sicher, dass der Name die maximale Zeichenlänge nicht überschreitet
1000	Fenstervariable nicht im Bereich
1010	Zoom
1020	Interner Fehler
1030	Verletzung des Zugriffsschutzes auf geschützten Speicher
1040	Nicht unterstützte Funktion. Für diese Funktion ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1045	Nicht unterstützter Operator. Für diesen Operator ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1050	Nicht unterstütztes Merkmal. Für diesen Operator ist ein Computer-Algebra-System erforderlich. Probieren Sie TI-Nspire™ CAS.
1060	Das Eingabeargument muss numerisch sein. Nur Eingaben, die numerische Werte enthalten, sind zulässig.
1070	Argument der trig. Funktion ist zu groß für eine exakte Vereinfachung
1080	Keine Unterstützung von Antw (Ans). Diese Applikation unterstützt nicht Antw (Ans).
1090	Funktion ist nicht definiert. Verwenden Sie einen der folgenden Befehle: <ul style="list-style-type: none">• Definiere• :=• sto → um eine Funktion zu definieren.
1100	Nicht-reelle Berechnung Wenn die Software beispielsweise in der Einstellung Reell (Real) ist, ist $\sqrt{(-1)}$ ungültig. Um komplexe Berechnungen zu ermöglichen, ändern Sie die Moduseinstellung 'Reell oder Komplex' (Real or Complex) in KARTESISCH (RECTANGULAR) oder POLAR (POLAR).
1110	Ungültige Grenzen
1120	Keine Zeichenänderung

Fehlercode	Beschreibung
1130	Argument kann weder eine Liste noch eine Matrix sein
1140	Argumentfehler Das erste Argument muss ein Polynomausdruck im zweiten Argument sein. Wenn das zweite Argument ausgelassen wird, versucht die Software, eine Voreinstellung auszuwählen.
1150	Argumentfehler Die ersten zwei Argumente müssen Polynomausdrücke im dritten Argument sein. Wenn das dritte Argument ausgelassen wird, versucht die Software, eine Voreinstellung auszuwählen.
1160	Bibliotheks-Pfadname ungültig Ein Pfadname muss in der Form xxx\yyy angegeben werden, wobei: <ul style="list-style-type: none"> • Der xxx Teil kann 1 bis 16 Zeichen haben. • Der yyy Teil kann 1 bis 15 Zeichen haben. Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1170	Verwendung des Bibliotheks-Pfadnamens ungültig <ul style="list-style-type: none"> • Ein Wert kann einem Pfadnamen nicht mit Definiere (Define), := oder sto → zugewiesen werden. • Ein Pfadname kann nicht als lokale Variable festgelegt oder als Parameter in einer Funktions- oder Programmdefinition verwendet werden.
1180	Bibliotheks-Variablenname ungültig. Vergewissern Sie sich, dass der Name: <ul style="list-style-type: none"> • keinen Punkt enthält • nicht mit einem Unterstrich beginnt • nicht länger ist als 15 Zeichen Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1190	Bibliotheks-Dokument nicht gefunden: <ul style="list-style-type: none"> • Vergewissern Sie sich, dass sich die Bibliothek im Ordner MyLib befindet. • Aktualisieren Sie die Bibliotheken. Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1200	Bibliotheksvariable nicht gefunden: <ul style="list-style-type: none"> • Vergewissern Sie sich, dass sich die Bibliotheksvariable im ersten Problem in der Bibliothek befindet. • Überprüfen Sie, dass die Bibliotheksvariable als LibPub oder LibPriv definiert wurde.

Fehlercode	Beschreibung
	<ul style="list-style-type: none"> • Aktualisieren Sie die Bibliotheken. <p>Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation</p>
1210	<p>Unzulässiger Name für Bibliothekskurzform.</p> <p>Vergewissern Sie sich, dass der Name:</p> <ul style="list-style-type: none"> • keinen Punkt enthält • nicht mit einem Unterstrich beginnt • nicht länger ist als 16 Zeichen • nicht reserviert ist <p>Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation.</p>
1220	<p>Bereichsfehler:</p> <p>Die Funktionen tangentLine und normalLine unterstützen nur Funktionen mit reellen Werten.</p>
1230	<p>Bereichsfehler.</p> <p>Im Grad- und Neugradmodus werden die trigonometrischen Konversionsoperatoren nicht unterstützt.</p>
1250	<p>Argumentfehler</p> <p>System linearer Gleichungen verwenden.</p> <p>Beispiel für ein System zweier linearer Gleichungen mit den Variablen x und y:</p> $3x+7y=5$ $2y-5x=-1$
1260	<p>Argumentfehler:</p> <p>Das erste Argument von nfMin oder nfMax muss ein Ausdruck in einer einzigen Variablen sein. Es darf keine andere Variable ohne Wert außer der interessierenden Variablen enthalten.</p>
1270	<p>Argumentfehler</p> <p>Ordnung der Ableitung muss gleich 1 oder 2 sein.</p>
1280	<p>Argumentfehler</p> <p>Verwenden Sie ein Polynom in entwickelter Form in einer Variablen.</p>
1290	<p>Argumentfehler</p> <p>Verwenden Sie ein Polynom in einer Variablen.</p>
1300	<p>Argumentfehler</p> <p>Die Koeffizienten des Polynoms müssen numerische Werte ergeben.</p>

Fehlercode	Beschreibung
1310	Argumentfehler: Eine Funktion konnte für ein oder mehrere Argumente nicht ausgewertet werden.
1380	Argumentfehler: Verschachtelte Aufrufe der domain() Funktion sind nicht erlaubt.

Warncodes und -meldungen

Über die Funktion **warnCodes()** können Sie die bei der Auswertung eines Ausdrucks erzeugten Warnungen speichern. In dieser Tabelle sind alle numerischen Warncodes und die zugehörigen Meldungen aufgelistet.

Ein Beispiel zum Speichern von Warncodes finden Sie unter **warnCodes()** (Seite 211).

Warncode	Meldung
10000	Operation könnte falsche Lösungen erzeugen.
10001	Differenzieren einer Gleichung kann eine falsche Gleichung erzeugen.
10002	Zweifelhafte Lösung
10003	Zweifelhafte Genauigkeit
10004	Operation könnte Lösungen unterdrücken.
10005	clöse (cSolve) liefert u.U. mehrere Nullstellen.
10006	Löse (Solve) liefert u.U. mehrere Nullstellen.
10007	Weitere Lösungen möglich. Versuchen Sie, Ober- und Untergrenzen und/oder einen Schätzwert anzugeben. Beispiele mit solve(): <ul style="list-style-type: none">• solve(Gleichung, Var=Schätzwert) UntereGrenze<Var<ObereGrenze• solve(Gleichung, Var) UntereGrenze<Var<ObereGrenze• solve(Gleichung, Var=Schätzwert)
10008	Definitionsbereich des Ergebnisses kann kleiner sein als der der Eingabe.
10009	Definitionsbereich des Ergebnisses kann größer sein als der der Eingabe.
10012	Nicht-reelle Berechnung
10013	$\wedge 0$ oder $\text{undef} \wedge 0$ durch 1 ersetzt
10014	$\text{undef} \wedge 0$ durch 1 ersetzt
10015	1^\wedge oder 1^\wedgeundef durch 1 ersetzt
10016	1^\wedgeundef durch 1 ersetzt
10017	Überlauf durch ∞ oder $-\infty$ $\rho\sigma$
10018	Operation verlangt und liefert 64 Bit Wert.
10019	Ressourcen ausgeschöpft, Vereinfachung könnte unvollständig sein.
10020	Argument der trig. Funktion ist zu groß für eine exakte Vereinfachung.
10021	Eingabe enthält einen nicht definierten Parameter. Ergebnis gilt möglicherweise nicht für alle möglichen Parameterwerte.

Warncode	Meldung
10022	Eventuell erhalten Sie eine Lösung, wenn Sie geeignete Ober- und Untergrenzen festlegen.
10023	Skalar wurde mit Einheitsmatrix multipliziert.
10024	Ergebnis über approximierte Arithmetik erhalten.
10025	Äquivalenz kann im Modus EXAKT nicht verifiziert werden.
10026	Einschränkung wird möglicherweise ignoriert. Geben Sie Einschränkungen in der Form "\ 'Variable Konstante MatheTestSymbol' oder einer Verbindung dieser Formen an, z. B. 'x<3 und x>-12'

Allgemeine Hinweise

Hinweise zu TI Produktservice und Garantieleistungen

Informationen über Produkte und Dienstleistungen von TI Wenn Sie mehr über das Produkt- und Serviceangebot von TI wissen möchten, senden Sie uns eine E-Mail oder besuchen Sie uns im World Wide Web.

E-Mail-Adresse: ti-cares@ti.com

Internet-Adresse: education.ti.com

Service- und Garantiehinweise Informationen über die Garantiebedingungen oder über unseren Produktservice finden Sie in der Garantieerklärung, die dem Produkt beiliegt. Sie können diese Unterlagen auch bei Ihrem Texas Instruments Händler oder Distributor anfordern.

Index

-, subtrahieren	223
!	
!, Fakultät	234
"	
", Sekunden-Schreibweise	243
#	
#, Umleitung	241
#, Umleitungsoperator	258
%	
%, Prozent	229
*	
*, multiplizieren	224
.	
.-, Punkt-Subtraktion	228
.*, Punkt-Multiplikation	228
./, Punkt-Division	228
.^, Punkt-Potenz	229
.+, Punkt-Addition	227
/	
/, dividieren	225
:	
:=, zuweisen	249
^	
^-1, Kehrwert	247
^, Potenz	226

→ Einheitenbezeichnung	245
, womit-Operator	247
'	
', Ableitungsstrich	245
', Minuten-Schreibweise	243
+	
+ , addieren	223
<	
<, kleiner als	231
=	
=, gleich	230
≠	
≠, ungleich[*]	230
>	
>, größer als	232
Π	
Π, Produkt	238
Σ	
Σ(), Summe	238
ΣInt()	239
ΣPrn()	240
√	
√(), Quadratwurzel	237
∠	
∠, winkel	244

ʃ, Integral	235
≤	
≤, kleiner oder gleich	231
≥	
≥, größer oder gleich	232
▶	
►, Einheiten konvertieren[*]	246
►, in Neugrad umwandeln	91
►approxFraction()	18
►Base10, Anzeige als ganze Dezimalzahl[Base10]	24
►Base16, Hexadezimaldarstellung[Base16]	24
►Base2, Binärdarstellung[Base2]	22
►cos, durch Kosinus ausdrücken[cos]	35
►Cylind, Anzeige als Zylindervektor[Cylind (Zylindervektor)]	50
►DD, Anzeige als Dezimalwinkel[DD (Dezimalwinkel)]	53
►Decimal, Anzeige als Dezimalzahl[Dezimal]	54
►DMS, Anzeige als Grad/Minute/Sekunde[DMS (GMS)]	62
►exp, ausdrücken durch e[exp]	70
►Polar, Anzeige als Polarvektor[Polar]	140
►Rad, in Bogenmaß umwandeln	150
►Rect, Anzeige als kartesischer Vektor[Rect (Kartesisch)]	153
►sin, durch Sinus ausdrücken[sin]	175
►Sphere, Anzeige als sphärischer Vektor[Sphere (Kugelkoordinaten)]	184
►tmpCnv() (Konvertierung von Temperaturbereichen)[tmpCnv]	200
⇒	
⇒, logische Implikation	233, 255
→	
→, speichern	249
↔	
↔, logische doppelte Implikation[*]	233
©	
©, Kommentar	250

°, Grad-Schreibweise	243
°, Grad/Minute/Sekunde	243
0	
Ob, binäre Anzeige	250
Oh, hexadezimale Anzeige	250
1	
10^(), Potenz von zehn	246
A	
Abbruch, Exit	70
Ableitung oder n-te Ableitung	
Vorlage für	10
Ableitungen	
erste Ableitung, d()	234
numerische Ableitung, nDeriv()	128
numerische Ableitung, nDerivative()	127
Ableitungsstrich,	245
Ableitungsstrich, '	245
abrufen/zurückgeben	
Variableninformationen, getVarInfo()	87
Abrufen/zurückgeben	
Variableninformationen, getVarInfo()	90
abs(), Absolutwert	12
Absolutwert	
Vorlage für	7-8
addieren, +	223
als kartesischen Vektor anzeigen, ►Rect	153
Amortisationstabelle, amortTbl()	12, 21
amortTbl(), Amortisationstabelle	12, 21
and, Boolean operator	13
and, Boolesches und	13
angle(), Winkel	14
ANOVA, einfache Varianzanalyse	15
ANOVA2way, zweifache Varianzanalyse	15
Ans, letzte Antwort	17
Antwort (letzte), Ans	17
Anz, Daten anzeigen	166
Anzeige als	
binär, ►Base2	22

Dezimalwinkel, ►DD	53
ganze Dezimalzahl, ►Base10	24
Grad/Minute/Sekunde, ►DMS	62
hexadezimal, ►Base16	24
kartesischer Vektor, ►Rect	153
Polarvektor, ►Polar	140
sphärischer Vektor, ►Sphere	184
Zylindervektor, ►Cylind	50
Anzeige als sphärischer Vektor, ►Sphere	184
Anzeige als Zylindervektor, ►Cylind	50
approx(), approximieren	18
approximieren, approx()	18
approxRational()	18
arccos()	19
arccosh()	19
arccot()	19
arccoth()	19
arccsc()	19
arccsch()	19
arcLen(), Bogenlänge	19
arcsec()	19
arcsech()	20
arcsin()	20
arcsinh()	20
arctan()	20
arctanh()	20
Argumente in TVM-Funktionen	206
Arkuskosinus, $\cos^{-1}()$	38
Arkussinus, $\sin^{-1}()$	176
Arkustangens, $\tan^{-1}()$	193
augment(), erweitern/verketten	20
Ausdrücke	
Ausdruck in Liste, exp►list()	71
String in Ausdruck, expr()	73, 113
Ausschließung mit „ “ Operator	247
Auswertungsreihenfolge	257
avgRC(), durchschnittliche Änderungsrate	21

B

Befehl Stopp	189
benutzerdefinierte Funktionen	54
benutzerdefinierte Funktionen und Programme	55-56

Bestimmtes Integral	
Vorlage für	10
Bibliothek	
erstelle Tastaturlbefehle für Objekte	101
binär	
Anzeige, Ob	250
Darstellung, ►Base2	22
binomCdf()	25
binomPdf()	25
Bogenlänge, arcLen()	19
Bogenmaß, r	242
Boolean operators	
and	13
Boolesch	
und, and	13
Boolesche Operatoren	
⇒	233, 255
↔	233
nand	125
nicht	132
nor	130
oder	136
xor	213
Brüche	
propFrac (Echter Bruch)	146
Vorlage für	5

C

Cdf()	76
ceiling(), Obergrenze	25
centralDiff()	26
cFactor(), komplexer Faktor	27
char(), Zeichenstring	28
charPoly()	28
χ ² 2way	29
ClearAZ	31
colAugment	31
colDim(), Spaltendimension der Matrix	32
colNorm(), Spaltennorm der Matrix	32
comDenom(), gemeinsamer Nenner	32
completeSquare(), complete square	33
conj(), Komplex Konjugierte	34
constructMat(), Matrix erstellen	34

corrMat(), Korrelationsmatrix	35
\cos^{-1} , Arkuskosinus	38
$\cos()$, Kosinus	36
$\cosh^{-1}()$, Arkuskosinus hyperbolicus	39
$\cosh()$, Cosinus hyperbolicus	38
$\cot^{-1}()$, Arkuskotangens	40
$\cot()$, Kotangens	39
$\coth^{-1}()$, Arkuskotangens hyperbolicus	41
$\coth()$, Kotangens hyperbolicus	40
countIf(), Elemente in einer Liste bedingt zählen	41
cPolyRoots()	42
crossP(), Kreuzprodukt	43
$\csc^{-1}()$, inverser Kosekans	43
$\csc()$, Kosekans	43
$\csch^{-1}()$, inverser Kosekans hyperbolicus	44
$\csch()$, Kosekans hyperbolicus	44
cSolve(), komplexe Lösung	45
CubicReg, kubische Regression	48
Cycle, Zyklus	49
cZeros(), komplexe Nullstellen	50

D

d(), erste Ableitung	234
Daten anzeigen, Anz	166
Daten anzeigen, Disp	62
dbd(), Tage zwischen Daten	53
Define, definiere	54
Definiere	54
Definiere LibPriv (Define LibPriv)	55
Definiere LibPub (Define LibPub)	56
Definiere, Define	54
definieren	
öffentliche Funktion / öffentliches Programm	56
private Funktion oder Programm	55
Definitionsreichsfunktion, domain()	63
deltaList()	56
deltaTmpCnv()	56
DelVar, Variable löschen	57
delVoid(), ungültige Elemente entfernen	57
derivative()	57
deSolve(), Lösung	58
det(), Matrixdeterminante	60

Dezimal	
Anzeige als ganze Zahl, ►Base10	24
Winkelanzeige, ►DD	53
diag(), Matrixdiagonale	61
Diagonalf orm, ref()	154
dim(), Dimension	61
Dimension, dim()	61
dividieren, /	225
domain(), Definitionsbereichsfunktion	63
dominant term, dominantTerm()	63
dominantTerm(), dominant term	63
dotP(), Skalarprodukt	65
drehen()	161
drehen, drehen()	161
durchschnittliche Änderungsrate, avgRC()	21

E

e Exponent	
Vorlage für	6
e hoch x, e^()	65, 70
e, ausdrücken durch	70
E, Exponent	241
E/A-Bildschirm anzeigen, Anz	166
e^(), e hoch x	65
echter Bruch, propFrac	146
eff(), Nominal- in Effektivsatz konvertieren	66
Effektivsatz, eff()	66
Eigenvektor, eigVc()	66
Eigenwert, eigVl()	66
eigVc(), Eigenvektor	66
eigVl(), Eigenwert	66
Eingabe, Input	95
Einheiten	
konvertieren	246
Einheitsmatrix, identity()	92
Einheitsvektor, unitV()	208
Einstellungen, hole aktuellen	88
Elemente in einer Liste bedingt zählen, countIf()	41
Elemente in einer Liste zählen, zähle()	41
else if, Elseif	67
else, Else	92
Elseif, else if	67

end	
for, EndFor	80
if, EndIf	92
Schleife, EndLoop	116
while, EndWhile	212
end if, EndIf	92
end while, EndWhile	212
Ende	
Funktion, EndFunc	84
Ende der Schleife, EndLoop	116
EndWhile, end while	212
Entfernen	
ungültige Elemente aus Liste	57
Entwickeln, expand()	71
EOS (Equation Operating System)	257
Equation Operating System (EOS)	257
Ergebnis	
ausdrücken durch e	70
durch Kosinus ausdrücken	35
durch Sinus ausdrücken	175
Ergebnisse mit zwei Variablen, TwoVar	206
Ergebnisse, Statistik	186
Ergebniswerte, Statistik	187
Ersetzung durch „ “ Operator	247
erste Ableitung	
Vorlage für	9
erweitern/verketten, augment()	20
euler(), Euler function	68
exact(), Exakt	69
Exakt, exact()	69
Exit, Abbruch	70
exp(), e hoch x	70
exp(list()), Ausdruck in Liste	71
expand(), Entwickeln	71
Exponent, E	241
Exponenten	
Vorlage für	5
Exponentielle Regression, ExpReg	73
expr(), String in Ausdruck	73, 113
ExpReg, exponentielle Regression	73
F	
factor(), Faktorisieren	74

Faktorisiere, factor()	74
Fakultät, !	234
Fehler übergeben, ÜbgebFeh	138
Fehler und Fehlerbehebung	
Fehler löschen, LöFehler	31
Fehler übergeben, ÜbgebFeh	138
festlegen	
Modus, setMode()	170
Fill, Matrix füllen	77
Finanzfunktionen, tvmFV()	205
Finanzfunktionen, tvmI()	205
Finanzfunktionen, tvmN()	205
Finanzfunktionen, tvmPmt()	205
Finanzfunktionen, tvmPV()	206
FiveNumSummary	77
floor(), Untergrenze	78
fMax(), Funktionsmaximum	78
fMin(), Funktionsminimum	79
Folge, seq()	166
Folge, series()	168
For	80
for, For	80
For, for	80
format(), Formatstring	80
Formatstring, format()	80
fpart(), Funktionsteil	81
freqTable()	82
Frobeniusnorm, norm()	131
Func, Funktion	84
Func, Programmfunktion	84
Funktion beenden, EndFunc	84
Funktionen	
benutzerdefiniert	54
Maximum, fMax()	78
Minimum, fMin()	79
Programmfunktion, Func	84
Teil, fpart()	81
Funktionen und Variablen	
kopieren	35
G	
g, Neugrad	241
ganze Zahl, int()	95

Ganzzahl teilen, intDiv()	96
ganzzahliger Teil, iPart()	98
gcd(), größter gemeinsamer Teiler	84
gehe zu, Goto	91
gemeinsamer Nenner, comDenom()	32
geomCdf()	85
geomPdf()	85
Get	85
getDenom(), Nenner holen/zurückgeben	87
getLangInfo(), Sprachinformationen abrufen/zurückgeben	87
getLockInfo(), testet den Gesperrt-Status einer Variablen oder Variablengruppe	88
getMode(), getMode-Einstellungen	88
getNum(), Zähler holen/zurückgeben	89
GetStr	89
getType(), get type of variable	90
getVarInfo(), Variableninformationen abrufen/zurückgeben	90
gleich, =	230
Gleichungssystem (2 Gleichungen)	
Vorlage für	7
Gleichungssystem (n Gleichungen)	
Vorlage für	7
Gleichungssystem, simult()	173
Goto, gehe zu	91
Grad-/Minuten-/Sekundenanzeige, ►DMS	62
Grad-Schreibweise, °	243
größer als, >	232
Größer oder gleich, ≥	232
größter gemeinsamer Teiler, gcd()	84
Gruppen, Gesperrt-Status testen	88
Gruppen, sperren und entsperren	112, 209

H

Häufigkeit()	82
hexadezimal	
Anzeige, ►Base16	24
Anzeige, 0h	250
holen/zurückgeben	
Nenner, getDenom()	87
Zähler, getNum()	89
Hyperbolisch	
Arkuskosinus, cosh ⁻¹ ()	39
Arkussinus, sinh ⁻¹ ()	178
Arkustangens, tanh ⁻¹ ()	194

Cosinus, cosh()	38
Sinus, sinh()	177
Tangens, tanh()	194

I

identity(), Einheitsmatrix	92
if, If	92
If, if	92
ifFn()	93
imag(), Imaginärteil	94
Imaginärteil, imag()	94
ImpDif(), implizite Ableitung	95
implizite Ableitung, Impdif()	95
in String, inString()	95
Input, Eingabe	95
inString(), in String	95
int(), ganze Zahl	95
intDiv(), Ganzzahlteilen	96
Integral, ∫	235
interpolate(), interpolate	96
inverse kumulative Normalverteilung (invNorm())	97
invF()	97
invNorm(), inverse kumulative Normalverteilung	97
invt()	98
Invy ² ()	97
iPart(), ganzzahliger Teil	98
irr(), interner Zinsfluss	
interner Zinsfluss, irr()	98
isPrime(), Primzahltest	99
isVoid(), Test auf Ungültigkeit	99

K

kartesische x-Koordinate, P►Rx()	137
kartesische y-Koordinate, P►Ry()	138
Kehrwert, \wedge^{-1}	247
kleiner als, <	231
Kleiner oder gleich, ≤	231
kleinstes gemeinsames Vielfaches, lcm	100
Kombinationen, nCr()	126
Kommentar, ©	250
komplex	
Faktor, cFactor()	27

Konjugierte, conj()	34
Lösung, cSolve()	45
Nullstellen, cZeros()	50
Konstante	
in solve()	181
Konstanten	
in cSolve()	47
in cZeros()	52
in deSolve()	58
in solve()	182
Tastenkürzel für	255
konvertieren	
Einheiten	246
Korrelationsmatrix, corrMat()	35
Kosinus / Cos	
ausdrücken durch	35
Kosinus, cos()	36
Kotangens, cot()	39
Kreuzprodukt, crossP()	43
kubische Regression, CubicReg	48
kumulierte Summe, cumulativeSum()	49
kumulierteSumme(), kumulierte Summe	49

L

Lbl, Marke	100
lcm, kleinstes gemeinsames Vielfaches	100
leere (ungültige) Elemente	252
left(), links	101
LibPriv	55
LibPub	56
libShortcut(), erstelle Tasturbefehle für Bibliotheksobjekte	101
Limes	
lim() (Limes)	102
limit() (Limes)	102
Vorlage für	11
limit() oder lim(), Limes	102
lineare Regression, LinRegAx	104
lineare Regression, LinRegBx	103
Lineare Regression, LinRegBx	105
links, left()	101
LinRegBx, lineare Regression	103
LinRegMx, lineare Regression	104
LinRegIntervals, lineare Regression	105

LinRegtTest	106
linSolve()	108
list►mat(), Liste in Matrix	109
Liste in Matrix, list►mat()	109
Liste, Elemente bedingt zählen	41
Liste, Elemente zählen in	41
Listen	
Ausdruck in Liste, exp►list()	71
Differenzen in einer Liste, Alist()	108
erweitern/verketten, augment()	20
in absteigender Reihenfolge sortieren, SortD	184
in aufsteigender Reihenfolge sortieren, SortA	184
Kreuzprodukt, crossP()	43
kumulierte Summe, cumulativeSum()	49
leere Elemente in	252
Liste in Matrix, list►mat()	109
Matrix in Liste, mat►list()	117
Maximum, max()	117
Minimum, min()	121
neu, newList()	127
Produkt, product()	145
Skalarprodukt, dotP()	65
Summe, sum()	190
Summierung, sum()	190
Teil-String, mid()	120
In(), natürlicher Logarithmus	109
LnReg, logarithmische Regression	110
Local, lokale Variable	111
Lock, Variable oder Variablengruppe sperren	112
LöFehler, Fehler löschen	31
Logarithmen	109
Logarithmische Regression, LnReg	110
Logarithmus	
Vorlage für	6
logische doppelte Implikation, \Leftrightarrow	233
logische Implikation, \Rightarrow	233, 255
Logistic, logistische Regression	113
LogisticD, logistische Regression	114
Logistische Regression, Logistic	113
Logistische Regression, LogisticD	114
lokal, Local	111
lokale Variable, Local	111
Loop, Schleife	116

löschen	
Variable, DelVar	57
Löschen	
Fehler, LöFehler	31
ungültige Elemente aus Liste	57
Löse, solve()	179
Lösung, deSolve()	58
LU, untere/obere Matrixzerlegung	116

M

Marke, Lbl	100
mat►list(), Matrix in Liste	117
Matrix (1×2)	
Vorlage für	8
Matrix (2×1)	
Vorlage für	8
Matrix (2×2)	
Vorlage für	8
Matrix (m × n)	
Vorlage für	8
Matrix erstellen, constructMat()	34
Matrix in Liste, mat►list()	117
Matrizen	
Determinante, det()	60
Diagonale, diag()	61
Diagonalform, ref()	154
Dimension, dim()	61
Eigenvektor, eigVc()	66
Eigenwert, eigVl()	66
Einheitsmatrix, identity()	92
erweitern/verketten, augment()	20
füllen, Fill	77
kumulierte Summe, cumulativeSum()	49
Liste in Matrix, list►mat()	109
Matrix in Liste, mat►list()	117
Matrixzeilenmultiplikation und -addition, mRowAdd()	123
Maximum, max()	117
Minimum, min()	121
neu, newMat()	128
Produkt, product()	145
Punkt-Addition, .+	227
Punkt-Division, ./	228
Punkt-Multiplikation, .*	228

Punkt-Potenz, \wedge	229
Punkt-Subtraktion, $-$	228
QR-Faktorisierung, QR	146
reduzierte Diagonalf orm, rref()	164
Spaltendimension, colDim()	32
Spaltennorm, colNorm()	32
Summe, sum()	190
Summierung, sum()	190
Transponierte, T	192
untere/obere Matrixzerlegung, LU	116
Untermatrix, subMat()	189, 191
Zeilenaddition, rowAdd()	163
Zeilendimension, rowDim()	163
Zeilennorm, rowNorm()	163
Zeilenoperation, mRow()	122
Zeilentausch, rowSwap()	164
Zufall, randMat()	152
max(), Maximum	117
Maximum, max()	117
mean(), Mittelwert	118
median(), Median	118
Median, median()	118
MedMed, Mittellinienregression	119
mid(), Teil-String	120
min(), Minimum	121
Minimum, min()	121
Minuten-Schreibweise,	243
mirr(), modifizierter interner Zinsfluss	121
mit,	247
Mittellinienregression, MedMed	119
Mittelwert, mean()	118
mod(), Modulo	122
Modi	
festlegen, setMode()	170
Modifizierter interner Zinsfluss, mirr()	121
Modulo, mod()	122
Moduseinstellungen, getMode()	88
mRow(), Matrixzeilenoperation	122
mRowAdd(), Matrixzeilenmultiplikation und -addition	123
Multipler linearer Regressions-t-Test	124
multiplizieren, *	224
MultReg (Mehrfachregression)	123
MultRegIntervals() (Mehrfachregressionsintervall)	123

N

n-te Wurzel	
Vorlage für	6
nand, Boolescher Operator	125
natürlicher Logarithmus, ln()	109
nCr(), Kombinationen	126
nDerivative(), numerische Ableitung	127
Negation, Eingabe von negativen Zahlen	258
Nenner	32
Nettobarwert, npv()	133
neu	
Liste, newList()	127
Matrix, newMat()	128
Neugrad-Schreibweise, g	241
newList(), neue Liste	127
newMat(), neue Matrix	128
nfMax(), numerisches Funktionsmaximum	128
nfMin(), numerisches Funktionsminimum	128
nicht, Boolescher Operator	132
nInt(), numerisches Integral	129
nom), Effektivzins in Nominalzins konvertieren	129
Nominalzinssatz, nom()	129
nor, Boolescher Operator	130
norm(), Frobeniusnorm	131
Normale, normalLine()	131
normalLine()	131
Normalverteilungswahrscheinlichkeit, normCdf()	131
normCdf() (Normalverteilungswahrscheinlichkeit)	131
normPdf() (Wahrscheinlichkeitsdichte)	131
nPr(), Permutationen	132
npv(), Nettobarwert	133
nSolve(), numerische Lösung	134
Nullstellen, zeroes()	214
numerisch	
Ableitung, nDeriv()	128
Ableitung, nDerivative()	127
Integral, nInt()	129
Lösung, nSolve()	134

O

Obergrenze, ceiling()	25-26, 42
Objekte	
erstelle Tastaturlbefehle für Bibliothek	101
oder (Boolesch), oder	136
oder, Boolescher Operator	136
OneVar, Statistik mit einer Variable	135
Operatoren	
Auswertungsreihenfolge	257
ord(), numerischer Zeichencode	137

P

P►Rx(), kartesische x-Koordinate	137
P►Ry(), kartesische y-Koordinate	138
Pdf()	81
Permutationen, nPr()	132
piecewise()(Stückweise)	139
poissCdf()	139
poissPdf()	139
polar	
Koordinate, R►Pr()	150
Koordinate, R►Pθ()	150
Vektoranzeige, ►Polar	140
polyCoef()	140
polyDegree()	141
polyEval(), Polynom auswerten	142
polyGcd()	142-143
Polynom auswerten, polyEval()	142
Polynome	
auswerten, polyEval()	142
Zufall, randPoly()	152
PolyRoots()	143
Potenz von zehn, 10^()	246
Potenz, ^	226
Potenzregression, PowerReg	143, 156, 158, 197
PowerReg, Potenzregression	143
Prgm, Definiere Programm	144
Primzahltest, isPrime()	99
prodSeq()	145
product(), Produkt	145
Produkt $\prod()$	
Vorlage für	9

Produkt, $\prod()$	238
Produkt, product()	145
Programme	
öffentliche Bibliothek definieren	56
Private Bibliothek definieren	55
Programme und Programmieren	166
E/A-Bildschirm anzeigen, Zeige	62
Fehler löschen, LöFehler	31
programmieren	
Daten anzeigen, Disp	62
Definiere Programm, Prgm	144
Fehler übergeben, ÜbgebFeh	138
Programmierung	166
propFrac, echter Bruch	146
Prozent, %	229
Punkt	
Addition, .+	227
Division, ./	228
Multiplikation, .*	228
Potenz, .^	229
Subtraktion, -	228

Q

QR-Faktorisierung, QR	146
QR,QR-Faktorisierung	146
Quadratische Regression, QuadReg	147
Quadratwurzel	
Vorlage für	5
Quadratwurzel, $\sqrt()$	237
Quadratwurzel, $\pm()$	185
QuadReg, quadratische Regression	147
QuartReg, Regression vierter Ordnung	148

R

r, Bogenmaß	242
R \mapsto Pr(), Polarkoordinate	150
R \mapsto Pθ(), Polarkoordinate	150
rand(), Zufallszahl	151
randBin, Zufallszahl	151
randInt(), ganzzahlige Zufallszahl	151
randMat(), Zufallsmatrix	152
randNorm(), Zufallsnorm	152

randPoly(), Zufallspolynom	152
randSamp() (Zufallsstichprobe)	153
RandSeed, Zufallszahl	153
real(), reell	153
rechts, right()	159
reduzierte Diagonalform, rref()	164
reell, real()	153
ref(), Diagonalform	154
Regression vierter Ordnung, QuartReg	148
Regressionen	
exponentielle, ExpReg	73
kubische, CubicReg	48
lineare Regression, LinRegAx	104
lineare Regression, LinRegBx	103
Lineare Regression, LinRegBx	105
logarithmische, LnReg	110
Logistic (Logistisch)	113
logistische, Logistic	114
Mittellinie, MedMed	119
MultReg (Mehrfachregression)	123
Potenzregression, PowerReg	143, 156, 158, 197
quadratische, QuadReg	147
sinusförmige, SinReg	178
vierter Ordnung, QuartReg	148
remain(), Rest	156
Request	156
RequestStr	158
Rest, remain()	156
Return, Rückgabe	159
right(), rechts	159
right, right()	33, 68, 96, 159, 211
rk23(), Runge Kutta function	159
rotate(), rotieren	161
rotieren, rotate()	161
round(), runden	162
rowAdd(), Matrixzeilenaddition	163
rowDim(), Zeilendimension der Matrix	163
rowNorm(), Zeilennorm der Matrix	163
rowSwap(), Matrixzeilentausch	164
rref(), reduzierte Diagonalform	164
Rückgabe, Return	159
runden, round()	162

Schleife, Loop	116
Schreibweise Grad/Minute/Sekunde	243
$\sec^{-1}()$, Arkussekans	165
$\sec()$, Sekans	165
$\operatorname{sech}^{-1}()$, Arkussekans hyperbolicus	166
$\operatorname{sech}()$, Sekans hyperbolicus	165
Sekunden-Schreibweise, "	243
seq(), Folge	166
seqGen()	167
seqn()	168
sequence, seq()	167-168
series(), Folge	168
setMode(), Modus festlegen	170
shift(), verschieben	171
sign(), Zeichen	173
simult(), Gleichungssystem	173
$\sin^{-1}()$, Arkussinus	176
$\sin()$, Sinus	175
$\sinh^{-1}()$, Arkussinus hyperbolicus	178
$\sinh()$, Sinus hyperbolicus	177
SinReg, sinusförmige Regression	178
Sinus	
ausdrücken durch	175
Sinus, $\sin()$	175
Sinusförmige Regression, SinReg	178
Skalar	
Produkt, dotP()	65
solve(), Löse	179
SortA, in aufsteigender Reihenfolge sortieren	184
SortD, in absteigender Reihenfolge sortieren	184
sortieren	
in absteigender Reihenfolge sortieren, SortD	184
in aufsteigender Reihenfolge, SortA	184
speichern	
Symbol, \rightarrow	249
Sprache	
Sprachinformation abrufen	87
sqrt(), Quadratwurzel	185
Standardabweichung, stdDev()	187-188, 209
stat.results	186
stat.values	187

Statistik	
Ergebnisse mit zwei Variablen, TwoVar	206
Fakultät, !	234
Kombinationen, nCr()	126
Median, median()	118
Mittelwert, mean()	118
Permutationen, nPr()	132
Standardabweichung, stdDev()	187-188, 209
Statistik mit einer Variable, OneVar	135
Varianz, variance()	209
Zufallsnorm, randNorm()	152
Zufallszahl, RandSeed	153
Statistik mit einer Variable, OneVar	135
stdDevPop(), Populations-Standardabweichung	187
stdDevSamp(), Stichproben-Standardabweichung	188
String	
Dimension, dim()	61
Länge	61
string(), Ausdruck in String	189
Stringlänge	61
strings	
right, right()	33, 68, 96, 159, 211
Strings	
Ausdruck in String, string()	189
Format, format()	80
Formatieren	80
in, InString	95
links, left()	101
rechts, right()	159
rotieren, rotate()	161
String in Ausdruck, expr()	73, 113
Teil-String, mid()	120
Umleitung, #	241
verschieben, shift()	171
Zeichencode, ord()	137
Zeichenstring, char()	28
Stückweise definierte Funktion (2 Teile)	
Vorlage für	6
Stückweise definierte Funktion (n Teile)	
Vorlage für	7
Student-t-Wahrscheinlichkeitsdichte, tPdf()	201
subMat(), Untermatrix	189, 191
subtrahieren, -	223

sum(), Summe	190
sumif()	190
Summe $\Sigma()$	
Vorlage für	9
Summe der Tilgungszahlungen	240
Summe der Zinszahlungen	239
Summe, $\Sigma()$	238
Summe, sum()	190
sumSeq()	191

T

t test, t-Test	203
T, Transponierte	192
Tagen zwischen Daten, dbd()	53
tan ⁻¹⁽⁾ , Arkustangens	193
tan(), Tangens	192
Tangens, tan()	192
Tangente, tangentLine()	194
tangentLine()	194
tanh ⁻¹⁽⁾ , Arkustangens hyperbolicus	194
tanh(), Tangens hyperbolicus	194
Tastenkürzel	255
Tastenkürzel, Tastatur	255
Taylor-Polynom, taylor()	195
taylor(), Taylor-Polynom	195
tCdf(), Wahrscheinlichkeit einer Student t-Verteilung	196
tCollect(), trigonometrische Zusammenfassung	196
Teil-String, mid()	120
Test auf Ungültigkeit, isVoid()	99
Test_2S, Zwei-Stichproben F-Test	83
tExpand(), trigonometrische Entwicklung	196
Text, Befehl	197
tInterval, Konfidenzintervall t	198
tInterval_2Samp, ZweiStichproben-t-Konfidenzintervall	199
tmpCnv() (Konvertierung von Temperaturwerten)	200
trace()	201
Transponierte, T	192
trigonometrische Entwicklung, tExpand()	196
trigonometrische Zusammenfassung, tCollect()	196
Try, Befehl zur Fehlerbehandlung	202
tTest, t-Test	203
tTest_2Samp, Zwei-Stichproben-t-Test	204

TVM-Argumente	206
tvmFV()	205
tvmI()	205
tvmN()	205
tvmPmt()	205
tvmPV()	206
TwoVar, Ergebnisse mit zwei Variablen	206

Ü

Ügebefehl, Fehler übergeben	138
-----------------------------------	-----

U

Umleitung, #	241
Umleitungsoperator (#)	258
umwandeln	
►Grad (Neugrad)	91
►Rad	150
unbestimmtes Integral	
Vorlage für	10
ungleich, ≠	230
ungültig, testen auf	99
ungültige Elemente	252
ungültige Elemente, entfernen	57
unitV(), Einheitsvektor	208
unLock, Variable oder Variablengruppe entsperren	209
Untergrenze, floor()	78
Untermatrix, subMat()	189, 191
Unterstrich, _	245

V

Variable	
Name aus String erstellen	258
Variable oder Funktion kopieren, CopyVar	35
Variablen	
alle einbuchstabigen löschen	31
lokal, Local	111
löschen, DelVar	57
Variablen und Funktionen	
kopieren	35
Variablen und Variablengruppen entsperren	209
Variablen und Variablengruppen sperren	112
Variablen, sperren und entsperren	88, 112, 209

Varianz, variance()	209
varPop() (Populationsvarianz)	209
varSamp(), Stichproben-Varianz	209
Vektoren	
Anzeige als Zylindervektor, ►Cylind	50
Einheit, unitV()	208
Kreuzprodukt, crossP()	43
Skalarprodukt, dotP()	65
verschieben, shift()	171
Verteilungsfunktionen	
binomCdf()	25
binomPdf()	25
invNorm()	97
invt()	98
Invχ ² ()	97
normCdf() (Normalverteilungswahrscheinlichkeit)	131
normPdf() (Wahrscheinlichkeitsdichte)	131
poissCdf()	139
poissPdf()	139
tCdf()	196
tPdf()	201
χ ² 2way()	29
χ ² Cdf()	29
χ ² GOF()	30
χ ² Pdf()	30
Vorlagen	
Ableitung oder n-te Ableitung	10
Absolutwert	7-8
Bestimmtes Integral	10
Bruch	5
eExponent	6
erste Ableitung	9
Exponent	5
Gleichungssystem (2 Gleichungen)	7
Gleichungssystem (n Gleichungen)	7
Limes	11
Logarithmus	6
Matrix (1 × 2)	8
Matrix (2 × 1)	8
Matrix (2 × 2)	8
Matrix (m × n)	8
n-te Wurzel	6
Produkt Π()	9

Quadratwurzel	5
Stückweise definierte Funktion (2 Teile)	6
Stückweise definierte Funktion (n Teile)	7
Summe $\sum()$	9
unbestimmtes Integral	10
zweite Ableitung	10

W

Wahrscheinlichkeit einer Student-t-Verteilung, tCdf()	196
Wahrscheinlichkeitsdichte, normPdf()	131
Warncodes und -meldungen	268
warnCodes(), Warning codes	211
Warte-Befehl	210
wenn, when()	211
when(), wenn	211
while, While	212
While, while	212
winkel, \angle	244
Winkel, angle()	14
womit-Operator „ “	247
womit-Operator, Auswertungsreihenfolge	257

X

x^2 , Quadrat	227
XNOR	233
xor, Boolesches exklusives oder	213

Z

Zähle Tage zwischen Daten, dbd()	53
zähl(), Elemente in einer Liste zählen	41
Zeichen	
String, char()	28
Zeichencode, ord()	137
Zeichen, sign()	173
Zeichenfolgen	
drehen, drehe()	161
zum Erstellen von Variablennamen verwenden	258
Zeichenstring, char()	28
Zeige, Daten anzeigen	62
Zeitwert des Geldes, Anzahl Zahlungen	205
Zeitwert des Geldes, Barwert	206
Zeitwert des Geldes, Endwert	205

Zeitwert des Geldes, Zahlungsbetrag	205
Zeitwert des Geldes, Zinsen	205
zeroes(), Nullstellen	214
zInterval, z-Konfidenzintervall	216
zInterval_1Prop, z-Konfidenzintervall für eine Proportion	217
zInterval_2Prop, z-Konfidenzintervall für zwei Proportionen	217
zInterval_2Samp, z-Konfidenzintervall für zwei Stichproben	218
zTest	219
zTest_1Prop, z-Test für eine Proportion	220
zTest_2Prop, z-Test für zwei Proportionen	220
zTest_2Samp, z-Test für zwei Stichproben	221
Zufall	
Matrix, randMat()	152
Norm, randNorm()	152
Polynom, randPoly()	152
Zahl, RandSeed	153
Zufallsstichprobe	153
zuweisen, :=	249
Zwei-Stichproben F-Test	83
zweite Ableitung	
Vorlage für	10
Zyklus, Cycle	49

Δ

Δlist(), Listendifferenz	108
---------------------------------	-----

X

χ²Cdf()	29
χ²GOF	30
χ²Pdf()	30