TEXAS INSTRUMENTS

TI-nspire™

TI-Nspire™ CAS Referenzhandbuch

Dieser Leitfaden îst gültig für die TI-Nspire™ Software-Version 4.3. 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 Dokumentationen, 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.

Inhaltsverzeichnis

Wichtige Informationen 2
Inhaltsverzeichnis 3

Vorlagen für Ausdrücke 5

Alphabetische Auflistung 12

A 12
B 21
C 26
D 54
E 66
F 77
G 88
I 95
L 104
M 121
N 130
O 140
P 143
Q 153
R 156
S 171
T 199
U 216
V 216
W 218
X 220
Z 221

Sonderzeichen 231

Leere (ungültige) Elemente 259

Tastenkürzel zum Eingeben mathematischer Ausdrücke 261

Auswertungsreihenfolge in EOS™ (Equation Operating System) 263

Fehlercodes und -meldungen 265

Warncodes und -meldungen 274

Allgemeine Hinweise 276

Hinweise zu TI Produktservice und Garantieleistungen 276

Index **277**

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 233.

Beispiel:

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

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 234.

Beispiel:

$$2^3 \qquad 8$$

Vorlage Quadratwurzel

[ctrl] [x²] Tasten

Hinweis: Siehe auch **√() (Quadratwurzel)**, Seite 245.

Beispiel:

$$\sqrt{4} \qquad 2$$

$$\sqrt{\{9,a,4\}} \qquad \{3,\sqrt{(a)},2\}$$

Vorlage n-te Wurzel

ctrl ^ Tasten

$\sqrt[\square]{\square}$ **Hinweis:** Siehe auch **root()**, Seite 167.

Beispiel:

$\sqrt[3]{8}$ → 2

$\sqrt[3]{\{8,27,b\}}$ → $\left\{2,3,b^{\frac{1}{3}}\right\}$

Vorlage e Exponent

e^x Tasten

$e^{\square}$

Potenz zur natürlichen Basis *e*

Hinweis: Siehe auch **e^()**, Seite 66.

Example:

e^1 → e

$e^{1.}$ → 2.71828182846

Vorlage Logarithmus

ctrl 10^x Taste

$\log_{\square}(\square)$

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

Hinweis: Siehe auch **log()**, Seite 117.

Beispiel:

$\log_4(2.)$ → 0.5

Vorlage Stückweise (2 Teile)

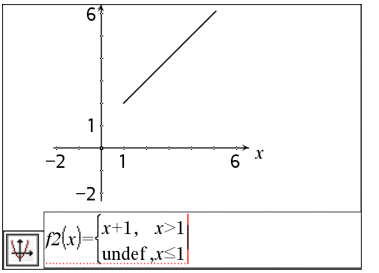
Katalog > $\frac{\square}{\square}$

$\begin{cases}\square, & \square \\ \square, & \square\end{cases}$

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 145.

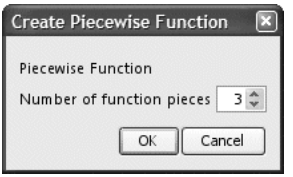
Beispiel:



Vorlage Stückweise (n Teile)

Katalog >

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



Hinweis: Siehe auch **piecewise()**, Seite 145.

Beispiel:

Siehe Beispiel für die Vorlage Stückweise (2 Teile).

Vorlage System von 2 Gleichungen

Katalog >

$\begin{cases} \square \\ \square \end{cases}$

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 199.

Beispiel:

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

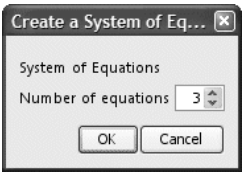
$\text{solve}\left(\begin{cases} y=x^2-2 \\ x+2\cdot y=-1 \end{cases}, x, y\right)$

$x=\frac{-3}{2} \text{ and } y=\frac{1}{4} \text{ or } x=1 \text{ and } y=-1$

Vorlage System von n Gleichungen

Katalog >

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



Hinweis: Siehe auch **system()**, Seite 199.

Beispiel:

Siehe Beispiel für die Vorlage Gleichungssystem (2 Gleichungen).

Vorlage Absolutwert

Katalog >

$|\square|$ **Hinweis:** Siehe auch **abs()**, Seite 12.

Beispiel:

Vorlage Absolutwert

Katalog >

$$\left|\left\{2,-3,4,-4^3\right\}\right| \qquad \{2,3,4,64\}$$

Vorlage dd°mm'ss.ss"

Katalog >

$\square°\square'\square"$

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

Beispiel:

$$30°15'10" \qquad \frac{10891\cdot\pi}{64800}$$

Vorlage Matrix (2 x 2)

Katalog >

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

Erzeugt eine 2 x 2 Matrix.

Beispiel:

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

Vorlage Matrix (1 x 2)

Katalog >

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

Beispiel:

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

Vorlage Matrix (2 x 1)

Katalog >

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

Beispiel:

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

Vorlage Matrix (m x n)

Katalog >

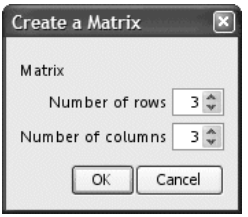
Die Vorlage wird angezeigt, nachdem Sie aufgefordert wurden, die Anzahl der Zeilen und Spalten anzugeben.

Beispiel:

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

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:

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

Hinweis: Siehe auch **Σ() (sumSeq)**, Seite 246.

Vorlage Produkt (Π)

Katalog >

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

Beispiel:

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

Hinweis: Siehe auch **Π() (prodSeq)**, Seite 246.

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 243.

$$\frac{d}{dx}\left(x^3\right) \qquad 3\cdot x^2$$

$$\frac{d}{dx}\left(x^3\right)\Big|_{x=3} \qquad 27$$

Vorlage Zweite Ableitung

Katalog >

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

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

Hinweis: Siehe auch **d() (Ableitung)**, Seite 243.

Beispiel:

$$\frac{d^2}{dx^2}\left(x^3\right) \qquad 6\cdot x$$

$$\frac{d^2}{dx^2}\left(x^3\right)\Big|_{x=3} \qquad 18$$

Vorlage n-te Ableitung

Katalog >

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

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

Hinweis: Siehe auch **d() (Ableitung)**, Seite 243.

Beispiel:

$$\frac{d^3}{dx^3}\left(x^3\right)\Big|_{x=3} \qquad 6$$

Vorlage Bestimmtes Integral

Katalog >

$$\int_{\square}^{\square} \square \, d\square$$

Hinweis: Siehe auch **∫() integral()**, Seite 231.

Beispiel:

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

Vorlage Unbestimmtes Integral

Katalog >

$$\int \square \, d\square$$

Beispiel:

Vorlage Unbestimmtes Integral

Katalog >

Hinweis: Siehe auch **∫() integral()**, Seite 231.

$$\int x^2 \, dx \qquad \frac{x^3}{3}$$

Vorlage Limes

Katalog >

$$\lim_{\square \to \square^{\square}} (\square)$$

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

Hinweis: Siehe auch **limit()**, Seite 106.

Beispiel:

$$\lim_{x \to 5^{\square}} (2 \cdot x + 3) \qquad 13$$

Alphabetische Auflistung

Elemente, deren Namen nicht alphabetisch sind (wie +, !, und >) finden Sie am Ende dieses Abschnitts (Seite 231). 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(*Ausdr1***)⇒***Ausdruck*

abs(*Liste1***)⇒***Liste*

abs(*Matrix1***)⇒***Matrix*

Gibt den Absolutwert des Arguments zurück.

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.

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

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

$|z| \quad |z|$

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

amortTbl() — Katalog >

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

Amortisationsfunktion, die eine Matrix als Amortisationstabelle 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 213) beschrieben.

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

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

- 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 247, und **bal()**, Seite 21, verwenden.

Boolescher Ausdr1 **and** *Boolescher Ausdr2*⇒*Boolescher Ausdruck*

Boolesche Liste1 **and** *Boolesche Liste2* ⇒*Boolesche Liste*

Boolesche Matrix1 **and** *Boolesche Matrix2*⇒*Boolesche Matrix*

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

$x\geq 3 \text{ and } x\geq 4$ $\quad x\geq 4$

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

*Ganzzahl1***and***Ganzzahl2*⇒*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; anderenfalls 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 $\quad$ 0h2C16

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

0b100101 and 0b100 $\quad$ 0b100

and (und) Katalog >

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.

Im Dec-Modus:

37 and 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.

angle() (Winkel) Katalog >

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 Grad-Modus:

$\text{angle}(0+2\cdot i)$	90

Im Neugrad-Modus:

$\text{angle}(0+3\cdot i)$	100

Im Bogenmaß-Modus:

$\text{angle}(1+i)$	$\frac{\pi}{4}$
$\text{angle}(z)$	$\frac{-\pi\cdot(\text{sign}(z)-1)}{2}$
$\text{angle}(x+i\cdot y)$	$\frac{\pi\cdot\text{sign}(y)}{2}-\tan^{-1}\left(\frac{x}{y}\right)$
$\text{angle}(\{1+2\cdot i,3+0\cdot i,0-4\cdot i\})$	$\left\{\frac{\pi}{2}-\tan^{-1}\left(\frac{1}{2}\right),0,\frac{-\pi}{2}\right\}$

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.

ANOVA

Katalog >

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 193)

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

Ausgabevariable	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)

Katalog >

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 193)

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
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(*Liste1***)⇒***Liste*

approx(*Matrix1***)⇒***Matrix*

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

$\text{approx}\left(\frac{1}{3}\right)$ 0.333333

$\text{approx}\left(\left\{\frac{1}{3},\frac{1}{9}\right\}\right)$ {0.333333,0.111111}

$\text{approx}(\{\sin(\pi),\cos(\pi)\})$ {0.,-1.}

$\text{approx}([\sqrt{2}\quad\sqrt{3}])$ [1.41421 1.73205]

$\text{approx}\left(\begin{bmatrix}\frac{1}{3} & \frac{1}{9}\end{bmatrix}\right)$ [0.333333 0.111111]

$\text{approx}(\{\sin(\pi),\cos(\pi)\})$ {0.,-1.}

$\text{approx}([\sqrt{2}\quad\sqrt{3}])$ [1.41421 1.73205]

▸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

$0.83333333333333\blacktriangleright\text{approxFraction}(5.\text{E}^{-}14)$ $\frac{5}{6}$

$\{\pi,1.5\}\blacktriangleright\text{approxFraction}(5.\text{E}^{-}14)$ $\left\{\frac{5419351}{1725033},\frac{3}{2}\right\}$

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.

$\text{approxRational}\left(0.333,5\cdot10^{-5}\right)$ $\frac{333}{1000}$

$\text{approxRational}\left(\{0.2,0.33,4.125\},5.\text{E}^{-}14\right)$ $\left\{\frac{1}{5},\frac{33}{100},\frac{33}{8}\right\}$

arccos() **Siehe cos⁻¹(), Seite 39**

arccosh() **Siehe cosh⁻¹(), Seite 40.**

arccot() **Siehe cot⁻¹(), Seite 41.**

arccoth() **Siehe coth⁻¹(), Seite 42.**

arccsc() **Siehe csc⁻¹(), Seite 44.**

arccsch() **Siehe csch⁻¹(), Seite 45.**

arcLen() (Bogenlänge) Katalog >

arcLen(*Ausdr1*,*Var*,*Start*,*Ende***)** ⇒*Ausdruck*

Gibt 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.

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

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

arcLen(*Liste1*,*Var*,*Start*,*Ende***)**⇒*Liste*

Gibt eine Liste der Bogenlängen für jedes Element von *Liste1* zwischen *Start* und *Ende* bezüglich der Variablen *Var* zurück.

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

arcsec() **Siehe sec⁻¹(), Seite 171.**

arcsech() **Siehe $\text{sech}^{-1}()$, Seite 172.**

arcsin() **Siehe $\sin^{-1}()$, Seite 183.**

arcsinh() **Siehe $\sinh^{-1}()$, Seite 184.**

arctan() **Siehe $\tan^{-1}()$, Seite 200.**

arctanh() **Siehe $\tanh^{-1}()$, Seite 202.**

augment() (Erweitern) Katalog >

augment(*Liste1, Liste2***)⇒***Liste***

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

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

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(*m1*,*m2*) $\begin{bmatrix}1 & 2 & 5\\ 3 & 4 & 6\end{bmatrix}$

avgRC() (Durchschnittliche Änderungsrate)

Katalog >

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

avgRC(*Ausdr1*, *Var* [=*Wert*] [, *Liste1*])⇒*Liste*

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

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

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.

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

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

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

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

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

B

bal()

Katalog >

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

bal(*NPmt*,*AmortTabelle*)⇒*Wert*

Amortisationsfunktion, die den Saldo nach einer angegebenen Zahlung berechnet.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* und *PmtAt* werden in der TVM-Argumentetabelle (Seite 213) beschrieben.

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

tbl:=amortTbl(6,6,5.75,5000,,12,12)

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

$\text{bal}(4,tbl)$ → 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 213) 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 247.

▸Base2 **Katalog >**

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*

256▸Base2	0b100000000
0h1F▸Base2	0b11111

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 binär angezeigt.

Negative Zahlen werden als Binärkomplement angezeigt. Beispiel:

−1 wird angezeigt als

0hFFFFFFFFFFFFFFFF 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

▸Base2 Katalog >

$-2^{63} - 1$ 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*

256▸Base16	0h100
0b111100001111▸Base16	0hF0F

▸Base16 Katalog >

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.

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() Katalog >

binomCdf(*n*,*p***)**⇒*Liste*

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≤X ≤*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 ≤ *obereGrenze*) setzen Sie *untereGrenze*=0

binomPdf() Katalog >

binomPdf(*n*,*p***)**⇒*Liste*

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

binomPdf()

Katalog >

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)

Katalog >

ceiling(*Ausdr1***)⇒***Ganzzahl*

Gibt die erste ganze Zahl zurück, die ≥ 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.

$\text{ceiling}(.456)$ → $1.$

$\text{ceiling}(\{-3.1,1,2.5\})$ → $\{-3.,1,3.\}$

$\text{ceiling}\begin{bmatrix} 0 & -3.2\cdot i \\ 1.3 & 4 \end{bmatrix}$ → $\begin{bmatrix} 0 & -3.\cdot i \\ 2. & 4 \end{bmatrix}$

centralDiff()

Katalog >

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.

$\text{centralDiff}(\cos(x),x,h)$ → $\frac{-(\cos(x-h)-\cos(x+h))}{2\cdot h}$

$\lim_{h\to 0}(\text{centralDiff}(\cos(x),x,h))$ → $-\sin(x)$

$\text{centralDiff}(x^3,x,0.01)$ → $3.\cdot(x^2+0.000033)$

$\text{centralDiff}(\cos(x),x)|x=\frac{\pi}{2}$ → $-1.$

$\text{centralDiff}(x^2,x,\{0.01,0.1\})$ → $\{2.\cdot x,2.\cdot x\}$

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.

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

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

cFactor() (Komplexer Faktor) Katalog >

cFactor(*Ausdr1*[**,***Var*]**)**⇒*Ausdruck*

cFactor(*Liste1*[**,***Var*]**)**⇒*Liste*

cFactor(*Matrix1*[**,***Var*]**)**⇒*Matrix*

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

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.

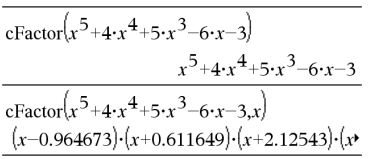
$\text{cFactor}\left(a^3\cdot x^2+a\cdot x^2+a^3+a,x\right)$	$a\cdot\left(a^2+1\right)\cdot(x-i)\cdot(x+i)$
$\text{cFactor}\left(x^2+3,x\right)$	$\left(x+\sqrt{3}\cdot i\right)\cdot\left(x-\sqrt{3}\cdot i\right)$
$\text{cFactor}\left(x^2+a,x\right)$	$\left(x+\sqrt{a}\cdot -i\right)\cdot\left(x+\sqrt{a}\cdot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

cFactor() (Komplexer Faktor) Katalog >

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 **factor()**.



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

char() (Zeichenstring) Katalog >

char(*Ganzzahl***)⇒***Zeichen*

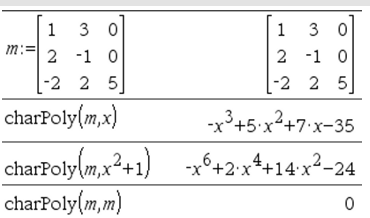
Gibt ein Zeichenstring zurück, das das Zeichen mit der Nummer *Ganzzahl* aus dem Zeichensatz des Handhelds enthält. Der gültige Wertebereich für *Ganzzahl* ist 0–65535.

charPoly() Katalog >

charPoly
(*Quadratmatrix,Var***)⇒***Polynomausdruck*

charPoly(*Quadratmatrix,*
*Ausdr***)⇒***Polynomausdruck*

charPoly
(
Quadratmatrix1,Matrix2



charPoly()

Katalog >

)⇒*Polynomausdruck*

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

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

definierte Polynom, wobei *I* die *n*×*n*-Einheitsmatrix kennzeichnet.

Quadratmatrix1 und *Quadratmatrix2* müssen dieselbe Dimension haben.

χ²2way

Katalog >

χ²2way *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 193.)

Informationen zu den Auswirkungen leerer Elemente in einer Matrix finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	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

χ²Cdf()

Katalog >

χ²Cdf

χ^2Cdf()

Katalog >

(
untereGrenze
,*obereGrenze*,*Freigrad***)**⇒*Zahl*, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

chi2Cdf
(
untereGrenze
,*obereGrenze*,*Freiheitsgrad***)**⇒*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($X \le$ *obereGrenze*) setzen Sie *untereGrenze*= 0.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

χ^2GOF

Katalog >

χ^2GOF *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 193)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	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^2Pdf()

Katalog >

χ^2Pdf(*XWert*,*FreiGrad*)⇒*Zahl*, wenn *Xwert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

chi2Pdf(*XWert*,*FreiGrad*)⇒*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 259).

ClearAZ (LöschAZ)

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 216

$5 \rightarrow b$	5
b	5
ClearAZ	*Done*
b	b

ClrErr (LöFehler)

Katalog >

ClrErr

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

Ein Beispiel für **ClrErr** finden Sie als Beispiel 2 im Abschnitt zum Befehl **Versuche (Try)**, Seite 209.

ClrErr (LöFehler)

Katalog >

Das **Else** im Block **Try...Else...EndTry** muss **ClrErr** oder **PassErr** (**ÜbgebFehler**) 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 144, und **Try**, Seite 209.

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*.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\rightarrow m1 \qquad \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

$$\begin{bmatrix}5 & 6\end{bmatrix}\rightarrow m2 \qquad \begin{bmatrix}5 & 6\end{bmatrix}$$

$$\text{colAugment}(m1,m2) \qquad \begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix}$$

colDim() (Spaltendimension)

Katalog >

colDim(*Matrix***)⇒*Ausdruck***

Gibt die Anzahl der Spalten von *Matrix* zurück.

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

$$\text{colDim}\left(\begin{bmatrix}0 & 1 & 2\\ 3 & 4 & 5\end{bmatrix}\right) \qquad 3$$

colNorm() (Spaltennorm)

Katalog >

colNorm(*Matrix***)⇒*Ausdruck***

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

$$\text{colNorm}(mat) \qquad 9$$

colNorm() (Spaltennorm) Katalog >

Gibt das Maximum der Summen der absoluten Elementwerte der Spalten von *Matrix* zurück.

Hinweis: Undefinierte Matrixelemente sind nicht zulässig. Siehe auch **rowNorm()**.

comDenom() (Gemeinsamer Nenner) Katalog >

comDenom(*Ausdr1*[**,***Var*]**)**⇒*Ausdruck*

comDenom(*Liste1*[**,***Var*]**)**⇒*Liste*

comDenom(*Matrix1*[**,***Var*]**)**⇒*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. Gleichartige 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 Speicherplatz ausgeht.

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

comDenom() (Gemeinsamer Nenner) Katalog >

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

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

$comden(1234\cdot x^2\cdot(y^3-y)+2468\cdot x\cdot(y^2-1))$
$1234\cdot x\cdot(x\cdot y+2)\cdot(y^2-1)$

completeSquare () Katalog >

completeSquare(*AusdrOdGl*, *Var***)⇒***Ausdruck oder Gleichung*

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

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

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

Konvertiert einen quadratischen Polynomausdruck der Form a·x²+b·x+c in die Form a·(x-h)²+k

- oder -

Konvertiert eine quadratische Gleichung der Form a·x²+b·x+c=d in die Form a·(x-h)²=k

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

completeSquare$(x^2+2\cdot x+3,x)$ $(x+1)^2+2$

completeSquare$(x^2+2\cdot x=3,x)$ $(x+1)^2=4$

completeSquare$(x^6+2\cdot x^3+3,x^3)$ $(x^3+1)^2+2$

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

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

completeSquare$(x^2+2\cdot x\cdot y,x,y)$ $(x+y)^2-y^2$

completeSquare ()

Katalog >

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*

conj(*Liste1***)⇒***Liste*

conj(*Matrix1***)⇒***Matrix*

Gibt das komplex Konjugierte des Arguments zurück.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt.

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

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

$\text{conj}(z)$ → z

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

constructMat()

Katalog >

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

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*.

$\text{constructMat}\left(\frac{1}{i+j}, i, j, 3, 4\right)$ →

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

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 Variablennamen vereinfachen lässt, der den Regeln entspricht.

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

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.

aa.a:=45	45
aa.b:=6.78	6.78
CopyVar *aa.*,*bb.*	*Done*
getVarInfo()	*aa.a* "NUM" "" 0 *aa.b* "NUM" "" 0 *bb.a* "NUM" "" 0 *bb.b* "NUM" "" 0

corrMat() (Korrelationsmatrix) Katalog >

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

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

▸cos Katalog >

Ausdr **▸cos**

▸cos

Katalog > 📖

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

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

▸cos reduziert alle Potenzen von

sin(...) modulo 1−cos(...)^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.

$(\sin(x))^2 \blacktriangleright \cos \quad 1-(\cos(x))^2$

cos() (Kosinus)

trig Taste

cos(*Ausdr1***)⇒***Ausdruck*

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 vorübergend aufzuheben.

Im Grad-Modus:

$\cos\left(\frac{\pi}{4}{}_r\right) \quad \frac{\sqrt{2}}{2}$

$\cos(45) \quad \frac{\sqrt{2}}{2}$

$\cos(\{0,60,90\}) \quad \left\{1,\frac{1}{2},0\right\}$

Im Neugrad-Modus:

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

Im Bogenmaß-Modus:

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

$\cos(45°)$ $\frac{\sqrt{2}}{2}$

cos(*Quadratmatrix1***)⇒***Quadratmatrix*

Gibt den Matrix-Kosinus von *Quadratmatrix1* zurück. Dies ist nicht gleichbedeutend mit der Berechnung des Kosinus jedes einzelnen Elements.

Wenn eine skalare Funktion f(A) auf *Quadratmatrix1* (A) angewendet wird, erfolgt die Berechnung des Ergebnisses durch den Algorithmus:

Berechnung der Eigenwerte (λi) und Eigenvektoren (Vi) von A.

Quadratmatrix1 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:

$$\cos\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$$

$$\begin{bmatrix} 0.212493 & 0.205064 & 0.121389 \\ 0.160871 & 0.259042 & 0.037126 \\ 0.248079 & -0.090153 & 0.218972 \end{bmatrix}$$

cos⁻¹() (Arkuskosinus)

Taste

cos⁻¹(*Ausdr1***)⇒***Ausdruck***

cos⁻¹(*Liste1***)⇒***Liste***

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 Grad-Modus:

$\cos^{-1}(1) \quad 0$

Im Neugrad-Modus:

$\cos^{-1}(0) \quad 100$

Im Bogenmaß-Modus:

$\cos^{-1}(\{0,0.2,0.5\}) \quad \left\{\frac{\pi}{2}, 1.36944, 1.0472\right\}$

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

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

$$\begin{bmatrix}1.73485+0.064606\cdot i & -1.49086+2.10514\\ -0.725533+1.51594\cdot i & 0.623491+0.77836\\ -2.08316+2.63205\cdot i & 1.79018-1.27182\cdot\end{bmatrix}$$

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

cosh() (Cosinus hyperbolicus)

Katalog >

cosh(*Ausdr1***)⇒***Ausdruck***

cosh(*Liste1***)⇒***Liste***

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 Grad-Modus:

$\cosh\left(\left\{\left(\frac{\pi}{4}\right)^r\right\}\right) \quad \cosh(45)$

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()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

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

$$\begin{bmatrix}421.255 & 253.909 & 216.905\\ 327.635 & 255.301 & 202.958\\ 226.297 & 216.623 & 167.628\end{bmatrix}$$

$\cosh^{-1}()$ (Arkuskosinus hyperbolicus)

Katalog >

$\cosh^{-1}$(*Ausdr1***)⇒***Ausdruck***

$\cosh^{-1}$(*Liste1***)⇒***Liste***

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

$$\cosh^{-1}(\{1,2.1,3\}) \qquad \{0,1.37286,\cosh^{-1}(3)\}$$

$\cosh^{-1}$(*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":

$$\cosh^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}2.52503+1.73485\cdot i & -0.009241-1.4908\\ 0.486969-0.725533\cdot i & 1.66262+0.623491\\ -0.322354-2.08316\cdot i & 1.26707+1.79018\end{bmatrix}$$

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

cot() (Kotangens)

trig Taste

cot(*Ausdr1***)** ⇒ *Ausdruck*

Im Grad-Modus:

cot() (Kotangens)

trig Taste

cot(*Liste1***)** ⇒ *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 vorübergend aufzuheben.

$\cot(45)$ → 1

Im Neugrad-Modus:

$\cot(50)$ → 1

Im Bogenmaß-Modus:

$\cot(\{1,2.1,3\})$ → $\left\{\frac{1}{\tan(1)}, -0.584848, \frac{1}{\tan(3)}\right\}$

$\cot^{-1}()$ (Arkuskotangens)

trig Taste

$\cot^{-1}$(*Ausdr1***)**⇒*Ausdruck*

$\cot^{-1}$(*Liste1***)**⇒*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.

Im Grad-Modus:

$\cot^{-1}(1)$ → 45

Im Neugrad-Modus:

$\cot^{-1}(1)$ → 50

Im Bogenmaß-Modus:

$\cot^{-1}(1)$ → $\frac{\pi}{4}$

coth() (Kotangens hyperbolicus)

Katalog >

coth(*Ausdr1***)**⇒*Ausdruck*

coth(*Liste1***)**⇒*Liste*

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

$\coth(1.2)$ → 1.19954

$\coth(\{1,3.2\})$ → $\left\{\frac{1}{\tanh(1)}, 1.00333\right\}$

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 259).

$\text{count}(2,4,6)$ 3

$\text{count}(\{2,4,6\})$ 3

$\text{count}\left(2,\{4,6\},\begin{bmatrix}8 & 10\\ 12 & 14\end{bmatrix}\right)$ 7

$\text{count}\left(\frac{1}{2},3+4\cdot i,\text{undef},\text{"hello"},x+5.,\text{sign}(0)\right)$ 2

Im letzten Beispiel werden nur 1/2 und 3+4*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.

$\text{countIf}(\{1,3,\text{"abc"},\text{undef},3,1\},3)$ 2

Zählt die Anzahl der Elemente, die 3 entsprechen.

countIf() Katalog >

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 259).

Hinweis: Siehe auch **sumIf()**, Seite 198, und **frequency()**, Seite 86.

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() Katalog >

cPolyRoots(*Poly*,*Var***)⇒***Liste*

cPolyRoots(*KoeffListe***)⇒***Liste*

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

Poly muss dabei ein Polynom in einer Variablen sein.

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

Hinweis: Siehe auch **polyRoots()**, Seite 149.

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·x+1,x) {-1,-1}

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

crossP() (Kreuzprodukt)

Katalog >

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.

$\text{crossP}(\{a1,b1\},\{a2,b2\})$
$\{0,0,a1\cdot b2-a2\cdot b1\}$

$\text{crossP}(\{0.1,2.2,-5\},\{1,-0.5,0\})$
$\{-2.5,-5.,-2.25\}$

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.

$\text{crossP}([1\ \ 2\ \ 3],[4\ \ 5\ \ 6])$ $[-3\ \ 6\ \ -3]$

$\text{crossP}([1\ \ 2],[3\ \ 4])$ $[0\ \ 0\ \ -2]$

csc() (Kosekans)

trig Taste

csc(*Ausdr1***)⇒***Ausdruck*

csc(*Liste1***)⇒***Liste*

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

Im Grad-Modus:

$\csc(45)$ $\sqrt{2}$

Im Neugrad-Modus:

$\csc(50)$ $\sqrt{2}$

Im Bogenmaß-Modus:

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

csc^{-1}() (Inverser Kosekans)

trig Taste

csc^{-1}(*Ausdr1***)** ⇒ *Ausdruck*

csc^{-1}(*Liste1***)** ⇒ *Liste*

Im Grad-Modus:

$\csc^{-1}(1)$ 90

Im Neugrad-Modus:

csc^{-1}() (Inverser Kosekans)

trig Taste

Gibt entweder den Winkel, dessen Kosekans *Ausdr1* entspricht, oder eine Liste der inversen Kosekans 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 **arccsc(...)** eintippen.

$\csc^{-1}(1)$ → 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(*Ausdr1***)** ⇒ *Ausdruck*

csch(*Liste1***)** ⇒ *Liste*

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

$\operatorname{csch}(3)$ → $\frac{1}{\sinh(3)}$

$\operatorname{csch}(\{1,2.1,4\})$ → $\left\{\frac{1}{\sinh(1)},0.248641,\frac{1}{\sinh(4)}\right\}$

csch^{-1}() (Inverser Kosekans hyperbolicus)

Katalog >

csch^{-1}(*Ausdr1***)** ⇒ *Ausdruck*

csch^{-1}(*Liste1***)** ⇒ *Liste*

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

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

$\operatorname{csch}^{-1}(1)$ → $\sinh^{-1}(1)$

$\operatorname{csch}^{-1}(\{1,2.1,3\})$ → $\left\{\sinh^{-1}(1),0.459815,\sinh^{-1}\left(\frac{1}{3}\right)\right\}$

cSolve() (Komplexe Lösung)

Katalog >

cSolve(*Gleichung***,** *Var***)⇒***Boolescher Ausdruck*

cSolve(*Gleichung***,** *Var=Schätzwert***)⇒***Boolescher Ausdruck*

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

$\text{solve}(x^3=-1,x)$ → $x=-1$

cSolve(*Ungleichung*, *Var*)⇒*Boolescher Ausdruck*

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.

$\text{cSolve}\left(x^{\frac{1}{3}}=-1,x\right)$ false

$\text{solve}\left(x^{\frac{1}{3}}=-1,x\right)$ $x=-1$

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

Im Modus Angezeigte Ziffern auf Fix 2:

$\text{exact}\left(\text{cSolve}\left(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3=0,x\right)\right)$

$x\cdot\left(x^4+4\cdot x^3+5\cdot x^2-6\right)=3$

$\text{cSolve}(Ans,x)$

$x=-1.11+1.07\cdot i$ or $x=-1.11-1.07\cdot i$ or $x=-2.$

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 (ctrl ␣ drücken) hinter *Var* setzen. Standardmäßig wird eine Variable als reeller Wert behandelt.

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

Bei Verwendung von *var*_ wird die Variable als komplex behandelt.

$\text{cSolve}(\text{conj}(z\_)=1+i,z\_)$ $z\_=1-i$

Sie sollten *var_* auch für alle anderen Variablen in *Gleichung* verwenden, die nicht-reelle Werte haben könnten. Anderenfalls 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.

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

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

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

Um das ganze Ergebnis zu sehen, drücken Sie ▲ und verwenden dann ◀ und ▶, 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.

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

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

$$\text{cSolve}\left(u\_\cdot v\_-u\_=v\_ \text{ and } v\_^2=-u\_,\{u\_,v\_,w\_\}\right)$$
$$u\_=\frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i \text{ and } v\_=\frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i \text{ and } w\_=c8 \text{ or } u\_$$

Bei Gleichungssystemen aus Polynomen kann die Berechnungsdauer 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.

$$\text{cSolve}\left(u\_+v\_=e^{w\_} \text{ and } u\_-v\_=i,\{u\_,v\_\}\right)$$
$$u\_=\frac{e^{w\_}+i}{2} \text{ and } v\_=\frac{e^{w\_}-i}{2}$$

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 Zahlen vereinfachbar sein.

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

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

$$\text{cSolve}\left(e^{z}-=w\_ \text{ and } w\_=z\_^2,\{w\_,z\_=1+i\}\right)$$
$$w\_=0.149606+4.8919\cdot i \text{ and } z\_=1.58805+1.$$

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

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 193.)

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 259).

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
stat.R^2	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
stat.XReg	Liste der Datenpunkte in der modifizierten *X List*, die schließlich in der Regression mit den Beschränkungen für *Häufigkeit*, *Kategorieliste* und *Mit Kategorien* verwendet wurde

Ausgabevariable	Beschreibung
stat.YReg	Liste der Datenpunkte in der modifizierten *Y List*, die schließlich in der Regression mit den Beschränkungen für *Häufigkeit*, *Kategorieliste* und *Mit Kategorien* verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

cumulativeSum() (kumulierteSumme) Katalog >

cumulativeSum(*Liste1***)⇒***Liste***

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

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

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 259).

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

cumulativeSum(*m1*)

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

Cycle (Zyklus) Katalog >

Cycle (Zyklus)

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 Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

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

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

$[2\ \ 2\ \ 3]\blacktriangleright\text{Cylind} \qquad \left[2\cdot\sqrt{2}\ \ \angle\frac{\pi}{4}\ \ 3\right]$

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] [␣] drücken) hinter *Var* setzen. Standardmäßig wird eine 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. Anderenfalls erhalten Sie möglicherweise unerwartete Ergebnisse.

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

Im Modus Angezeigte Ziffern auf Fix 3:

$\text{cZeros}\left(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3,x\right)$
$\left\{-1.1138+1.07314\cdot i,-1.1138-1.07314\cdot i,-2.\right\}$

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

$\text{cZeros}\left(\text{conj}(z\_)-1-i,z\_\right) \qquad \{1-i\}$

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.

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

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

$$\mathrm{cZeros}\left(\left\{u\_\cdot v\_-u\_-v\_, v\_^2+u\_\right\}, \left\{u\_, v\_\right\}\right) \quad \begin{bmatrix} 0 & 0 \\ \frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i & \frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i \\ \frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i & \frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i \end{bmatrix}$$

Zeile 2 extrahieren:

$$Ans[2] \quad \begin{bmatrix} \frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i & \frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i \end{bmatrix}$$

$$\mathrm{cZeros}\left(\left\{u\_\cdot v\_-u\_-c\_\cdot v\_, v\_^2+u\_\right\}, \left\{u\_, v\_\right\}\right) \quad \begin{bmatrix} 0 & 0 \\ \frac{-\left(\sqrt{1-4\cdot c\_}-1\right)^2}{4} & \frac{-\left(\sqrt{1-4\cdot c\_}-1\right)}{2} \\ \frac{-\left(\sqrt{1-4\cdot c\_}+1\right)^2}{4} & \frac{\sqrt{1-4\cdot c\_}+1}{2} \end{bmatrix}$$

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.

$$\text{cZeros}\left(\left\{u\_\cdot v\_-u\_-v\_,v\_^2+u\_\right\},\left\{u\_,v\_,w\_\right\}\right)$$
$$\begin{bmatrix} 0 & 0 & c4 \\ \frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i & \frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i & c4 \\ \frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i & \frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i & c4 \end{bmatrix}$$

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 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.

$$\text{cZeros}\left(\left\{u\_+v\_-e^{w\_},u\_-v\_-i\right\},\left\{u\_,v\_\right\}\right)$$
$$\begin{bmatrix} \frac{e^{w\_}+i}{2} & \frac{e^{w\_}-i}{2} \end{bmatrix}$$

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.

$$\text{cZeros}\left(\left\{e^{z}-w\_,w\_-z\_^2\right\},\left\{w\_,z\_\right\}\right)$$
$$\begin{bmatrix} 0.494866 & -0.703467 \end{bmatrix}$$

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

$$\text{cZeros}\left(\left\{e^{z}-w\_,w\_-z\_^2\right\},\left\{w\_,z\_=1+i\right\}\right)$$
$$\begin{bmatrix} 0.149606+4.8919\cdot i & 1.58805+1.54022\cdot i \end{bmatrix}$$

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)

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*

Liste1 ▸**DD**⇒*Liste*

Matrix1 ▸**DD**⇒*Matrix*

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

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

Im Grad-Modus:

(1.5°)▸DD	1.5°
(45°22'14.3")▸DD	45.3706°
{{45°22'14.3",60°0'0"}}▸DD	{45.3706°,60°}

Im Neugrad-Modus:

1▸DD	$\frac{9}{10}°$

Im Bogenmaß-Modus:

(1.5)▸DD	85.9437°

▸Decimal (Dezimal)

Katalog >

Ausdr1 ▸*Decimal*⇒*Ausdruck*

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.

$\frac{1}{3}$▸Decimal	0.333333

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 Befehls sein.

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

Define $g(x,y)=2\cdot x-3\cdot y$	*Done*
$g(1,2)$	-4
$1\to a$: $2\to b$: $g(a,b)$	-4
Define $h(x)=$when$(x<2,2\cdot x-3,-2\cdot x+3)$	*Done*
$h(-3)$	-9
$h(4)$	-5

Definie

Katalog >

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

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

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 56, und **Definiere LibPub (Define LibPub)**, Seite 57.

```
Define g(x,y)=Func                    Done
                If x>y Then
                Return x
                Else
                Return y
                EndIf
                EndFunc
g(3,-7)                                  3
```

```
Define g(x,y)=Prgm
                If x>y Then
                Disp x," greater than ",y
                Else
                Disp x," not greater than ",y
                EndIf
                EndPrgm
                                      Done
g(3,-7)
                    3 greater than -7
                                      Done
```

Definiere LibPriv (Define LibPriv)

Katalog >

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

Definiere LibPriv (Define LibPriv) **Katalog >**

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 55, und **Definiere LibPub (Define LibPub)**, Seite 57.

Definiere LibPub (Define LibPub) **Katalog >**

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 55, und **Definiere LibPriv (Define LibPriv)**, Seite 56.

deltaList() **Siehe ΔList(), Seite 113.**

deltaTmpCnv() **Siehe ΔtmpCnv(), Seite 207.**

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 216.

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 a	*Done*
$(a+2)^2$	$(a+2)^2$

$aa.a := 45$	45
$aa.b := 5.67$	5.67
$aa.c := 78.9$	78.9
getVarInfo()	$\begin{bmatrix} aa.a & \text{"NUM"} & \text{"}\square\text{"} \\ aa.b & \text{"NUM"} & \text{"}\square\text{"} \\ aa.c & \text{"NUM"} & \text{"}\square\text{"} \end{bmatrix}$
DelVar $aa.$	*Done*
getVarInfo()	"NONE"

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 259).

delVoid({1,void,3})	{1,3}

derivative()

Siehe *d*(), Seite 243.

deSolve(*ODE1.Oder2.Ordnung*, *Var*, *abhängigeVar*) ⇒*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 *c*k, 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.

$\text{deSolve}\left(y''+2\cdot y'+y=x^2,x,y\right)$

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

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

$\frac{d^2}{dx^2}\left(temp\right)+2\cdot\frac{d}{dx}\left(temp\right)+temp-x^2 \qquad 0$

DelVar *temp* *Done*

$\text{deSolve}\left(y'=\left(\cos\left(y\right)\right)^2\cdot x,x,y\right) \qquad \tan\left(y\right)=\frac{x^2}{2}+c4$

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

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

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.

$\sin(y)=\left(y\cdot e^{x}+\cos(y)\right)\cdot y' \rightarrow ode$

$\sin(y)=\left(e^{x}\cdot y+\cos(y)\right)\cdot y'$

$\text{deSolve}\left(ode \text{ and } y(0)=0,x,y\right) \rightarrow soln$

$\frac{-\left(2\cdot\sin(y)+y^{2}\right)}{2}=-\left(e^{x}-1\right)\cdot e^{-x}\cdot\sin(y)$

$soln \mid x=0 \text{ and } y=0$	true
$ode \mid y'=\text{impDif}(soln,x,y)$	true
DelVar *ode*,*soln*	*Done*

deSolve
(
ODE2.Ordnung
and
Anfangsbedingung1
and*Anfangsbedingung2*, *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

$\text{deSolve}\left(y''=y^{\frac{-1}{2}} \text{ and } y(0)=0 \text{ and } y'(0)=0,t,y\right)$

$\frac{2\cdot y^{\frac{3}{4}}}{3}=t$

$\text{solve}(Ans,y)$

$y=\frac{2^{\frac{2}{3}}\cdot(3\cdot t)^{\frac{4}{3}}}{4} \text{ and } t\geq 0$

deSolve() (Lösung) Katalog >

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

deSolve
(
ODE2.Ordnung
and*Randbedingung1***and***Randbedingung2***,**
*Var***,** *abhängigeVar***)⇒***eine spezielle Lösung*

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

$\text{deSolve}(y''=x \text{ and } y(0)=1 \text{ and } y'(2)=3,x,y)$
$y=\frac{x^3}{6}+x+1$

$\text{deSolve}(y''=2\cdot y' \text{ and } y(3)=1 \text{ and } y'(4)=2,x,y)$
$y=e^{2\cdot x-8}-e^{-2}+1$

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

det() (Matrixdeterminante) Katalog >

det(*Quadratmatrix***[,**
*Toleranz***])⇒***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. Anderenfalls 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 ·**max(dim(***Quadratmatrix***))** ·

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

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

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

$\begin{bmatrix}1.\text{E}20 & 1\\ 0 & 1\end{bmatrix}\rightarrow mat1$ $\begin{bmatrix}1.\text{E}20 & 1\\ 0 & 1\end{bmatrix}$

$\det(mat1)$ 0

$\det(mat1,.1)$ $1.\text{E}20$

det() (Matrixdeterminante) Katalog >

rowNorm(*Quadratmatrix***)**

diag() (Matrixdiagonale) Katalog >

diag(*Liste***)⇒***Matrix***

diag(*Zeilenmatrix***)⇒***Matrix***

diag(*Spaltenmatrix***)⇒***Matrix***

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

$\text{diag}([2\ \ 4\ \ 6]) \rightarrow \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$

diag(*Quadratmatrix***)⇒***Zeilenmatrix*

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

Quadratmatrix muss eine quadratische Matrix sein.

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

$\text{diag}(Ans) \rightarrow [4\ \ 2\ \ 9]$

dim() (Dimension) Katalog >

dim(*Liste***)⇒***Ganzzahl*

Gibt die Dimension von *Liste* zurück.

$\dim(\{0,1,2\}) \rightarrow 3$

dim(*Matrix***)⇒***Liste*

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

$\dim\left(\begin{bmatrix} 1 & -1 \\ 2 & -2 \\ 3 & 5 \end{bmatrix}\right) \rightarrow \{3,2\}$

dim(*String***)⇒***Ganzzahl*

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

$\dim(\text{"Hello"}) \rightarrow 5$

$\dim(\text{"Hello "\&"there"}) \rightarrow 11$

Disp (Zeige) Katalog >

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 ð
                                        241 ñ
                                        242 ò
                                        243 ó
                                        Done
```

▶DMS (GMS) Katalog >

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 251) 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()

Katalog >

domain(*Ausdr1*, *Var*) ⇒*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 ∫() eine nicht zulässige Funktion ist.

$\text{domain}\left(\int_1^x \frac{1}{t}\,dt, x\right) \blacktriangleright \text{domain}\left(\int_1^x \frac{1}{t}\,dt, x\right)$

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

dominanterTerm (), dominantTerm()

Katalog >

dominantTerm(*Expr1*, *Var* [, *Point*])⇒*expression*

dominantTerm(*Expr1*, *Var* [, *Point*]) | *Var*>*Point*⇒*expression*

dominantTerm(*Expr1*, *Var* [, *Point*]) | *Var*<*Point* ⇒*expression*

$\text{dominantTerm}(\tan(\sin(x))-\sin(\tan(x)), x)$	$\frac{x^7}{30}$
$\text{dominantTerm}\left(\frac{1-\cos(x-1)}{(x-1)^3}, x, 1\right)$	$\frac{1}{2\cdot(x-1)}$
$\text{dominantTerm}\left(x^{-2}\cdot\tan\left(x^{\frac{1}{3}}\right), x\right)$	$\frac{1}{x^{\frac{5}{3}}}$
$\text{dominantTerm}\left(\ln\left(x^x-1\right)\cdot x^{-2}, x\right)$	$\frac{\ln(x\cdot\ln(x))}{x^2}$

Gibt 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 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 −∞ 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 = ∞ oder −∞.

Wenn die Folge oder eine ihrer Ableitungen eine Sprungstelle bei *Point* hat, enthält das Ergebnis wahrscheinlich Unterausdrücke der Form sign(…) oder abs(…) für eine reelle Expansionsvariable oder $(-1)^{\text{floor}(\ldots\text{angle}(\ldots)\ldots)}$ 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* ≥ *Point*" oder "*Var* ≤ *Point*" an, um ein einfacheres Ergebnis zu erhalten.

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

$\text{dominantTerm}\left(e^{\frac{-1}{z}},z\right)$

$\text{dominantTerm}\left(e^{\frac{-1}{z}},z,0\right)$

$\text{dominantTerm}\left(\left(1+\frac{1}{n}\right)^{n},n,\infty\right)$ e

$\text{dominantTerm}\left(\tan^{-1}\left(\frac{1}{x}\right),x,0\right)$ $\frac{\pi\cdot\text{sign}(x)}{2}$

$\text{dominantTerm}\left(\tan^{-1}\left(\frac{1}{x}\right),x\right)\,|\,x>0$ $\frac{\pi}{2}$

dominanterTerm (), dominantTerm()

Katalog >

dominantTerm() können Sie verwenden, wenn Sie den einfachsten möglichen Ausdruck wissen möchten, der asymptotisch zu einem anderen Ausdruck wie *Var* → *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 175.

dotP() (Skalarprodukt)

Katalog >

dotP(*Liste1***,** *Liste2***)⇒***Ausdruck*

Gibt das Skalarprodukt zweier Listen zurück.

dotP(*Vektor1***,** *Vektor2***)⇒***Ausdruck*

Gibt das Skalarprodukt zweier Vektoren zurück.

Es müssen beide Zeilenvektoren oder beide Spaltenvektoren sein.

$\mathrm{dotP}(\{a,b,c\},\{d,e,f\})$ $\quad a\cdot d+b\cdot e+c\cdot f$

$\mathrm{dotP}(\{1,2\},\{5,6\})$ $\quad 17$

$\mathrm{dotP}([a\ \ b\ \ c],[d\ \ e\ \ f])$ $\quad a\cdot d+b\cdot e+c\cdot f$

$\mathrm{dotP}([1\ \ 2\ \ 3],[4\ \ 5\ \ 6])$ $\quad 32$

E

e^()

Taste e^x

e^(*Ausdr1***)⇒***Ausdruck*

Gibt **e** hoch *Ausdr1* zurück.

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

Hinweis: Das Drücken von e^x 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 $re^{i\theta}$ eingeben. Verwenden Sie diese aber nur im Winkelmodus Bogenmaß, da die Form im Grad- oder Neugrad-Modus einen Bereichsfehler verursacht.

e^1 $\quad e$

$e^{1.}$ $\quad 2.71828$

e^{3^2} $\quad e^9$

e^() — e^x Taste

e^(*Liste1***)⇒***Liste***

Gibt **e** hoch jedes Element der *Liste1* zurück.

$e^{\{1,1.,0.5\}}$ → $\{e, 2.71828, 1.64872\}$

e^(*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()**.

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

eff() — Katalog >

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

$\text{eff}(5.75,12)$ → 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 134.

eigVc() (Eigenvektor) — Katalog >

eigVc(*Quadratmatrix***)⇒***Matrix***

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 = $[x_1, x_2, \ldots, x_n]$, dann:

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

Im Komplex-Formatmodus "kartesisch":

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

$\text{eigVc}(m1)$

$\begin{bmatrix}-0.800906 & 0.767947 & (\\ 0.484029 & 0.573804+0.052258\cdot i & 0.5738\\ 0.352512 & 0.262687+0.096286\cdot i & 0.2626\end{bmatrix}$

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

eigVc() (Eigenvektor) — Katalog >

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.

eigVl() (Eigenwert) — Katalog >

eigVl(*Quadratmatrix***)⇒***Liste*

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

Im Komplex-Formatmodus "kartesisch":

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

$$\text{eigVl}(m1)$$

$$\{-4.40941, 2.20471+0.763006\cdot i, 2.20471-0.$$

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

Else — Siehe If, Seite 96.

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 83.**

EndFunc **Siehe Func, Seite 87.**

EndIf **Siehe If, Seite 96.**

EndLoop **Siehe Loop, Seite 120.**

EndWhile **Siehe While, Seite 220.**

EndPrgm **Siehe Prgm, Seite 151.**

EndTry **Siehe Try, Seite 209.**

euler ()

Katalog >

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

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

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

Verwendet die Euler-Methode zum Lösen des Systems

Differentialgleichung:

y'=0.001*y*(100-y) und y(0)=10

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

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

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

mit $abhVar(Var0)=abhVar0$ auf dem Intervall [$Var0$,$VarMax$]. Gibt eine Matrix zurück, deren erste Zeile die Ausgabewerte von Var definiert 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, 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 $VarMax$).

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

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

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

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 $y2(0)=5$

$$\text{euler}\left(\begin{cases} -y1+0.1\cdot y1\cdot y2 \\ 3\cdot y2-y1\cdot y2 \end{cases}, t, \{y1, y2\}, \{0,5\}, \{2,5\}, 1\right)$$
$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. & 5. \\ 2. & 1. & 1. & 3. & 27. & 243. \\ 5. & 10. & 30. & 90. & 90. & -2070. \end{bmatrix}$$

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*.

eval () — Hub-Menü

eval(*Expr***)** ⇒ *Zeichenfolge*

eval() ist nur im TI-Innovator™ Hub Befehlsargument von Programmierbefehlen **Get**, **GetStr** und **Send** gültig. Die Software wertet den Ausdruck *Expr* aus und ersetzt die Anweisung **eval()** mit dem Ergebnis als Zeichenfolge.

Das Argument *Expr* muss zu einer reellen Zahl vereinfachbar sein.

Stellen Sie das blaue Element von RGB LED auf halbe Intensität ein.

```
lum:=127                                   127
Send "SET COLOR.BLUE eval(lum)"            Done
```

Setzen Sie das blaue Element auf AUS zurück.

```
Send "SET COLOR.BLUE OFF"                  Done
```

Argument eval() muss zu einer reellen Zahl vereinfachbar sein.

```
Send "SET LED eval("4") TO ON"
                        "Error: Invalid data type"
```

Programm zum Einblenden des roten Elements

```
Define fadein()=
Prgm
For i,0,255,10
  Send "SET COLOR.RED eval(i)"
    Wait 0.1
  EndFor
  Send "SET COLOR.RED OFF"
EndPrgm
```

Führen Sie das Programm aus.

```
fadein()                                   Done
```

eval ()

Hub-Menü

Obwohl **eval()** sein Ergebnis nicht anzeigt, können Sie die resultierende Hub-Zeichenfolge nach Ausführen des Befehls durch Prüfung einer beliebigen der folgenden speziellen Variablen anzeigen.

iostr.SendAns
iostr.GetAns
iostr.GetStrAns

Hinweis: Siehe auch **Get** (Seite 89), **GetStr** (Seite 93) und **Send** (Seite 172).

exact() (Exakt)

Katalog >

exact(*Ausdr1* [**,** *Toleranz*])⇒*Ausdruck*

exact(*Liste1* [**,** *Toleranz*])⇒*Liste*

exact(*Matrix1* [**,** *Toleranz*])⇒*Matrix*

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

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

$\text{exact}(0.25)$	$\frac{1}{4}$
$\text{exact}(0.333333)$	$\frac{333333}{1000000}$
$\text{exact}(0.333333,0.001)$	$\frac{1}{3}$
$\text{exact}(3.5\cdot x+y)$	$\frac{7\cdot x}{2}+y$
$\text{exact}(\{0.2,0.33,4.125\})$	$\left\{\frac{1}{5},\frac{33}{100},\frac{33}{8}\right\}$

Exit (Abbruch)

Katalog >

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

Katalog >

Ausdr ▸**exp**

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

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

$\frac{d}{dx}\left(e^{x}+e^{-x}\right)$ → $2\cdot\sinh(x)$

$2\cdot\sinh(x)$▸exp → $e^{x}-e^{-x}$

exp() (e hoch x)

e^x Taste

exp(*Ausdr1***)⇒***Ausdruck*

Gibt **e** hoch *Ausdr1* zurück.

e^{1} → e

$e^{1.}$ → 2.71828

e^{3^2} → e^{9}

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

Sie können eine komplexe Zahl in der polaren Form rei θ eingeben. Verwenden Sie diese aber nur im Winkelmodus Bogenmaß, da die Form im Grad- oder Neugrad-Modus einen Bereichsfehler verursacht.

exp(*Liste1***)⇒***Liste*

Gibt **e** hoch jedes Element der *Liste1* zurück.

$e^{\{1,1.,0.5\}}$ → $\{e,2.71828,1.64872\}$

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()**.

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

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.

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

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

expand() (Entwickle) Katalog >

expand(*Ausdr1* [**,** *Var*])⇒*Ausdruck*

expand(*Liste1* [**,***Var*])⇒*Liste*

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.

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

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

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.

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

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

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

$\text{expand}(Ans,x)$ $\frac{1}{x-1}-\frac{1}{x}+\frac{1}{y\cdot(y-1)}$

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

$\text{expand}\left(\frac{x^3+x^2-2}{x^2-2}\right)$ $\frac{2\cdot x}{x^2-2}+x+1$

$\text{expand}(Ans,x)$ $\frac{1}{x-\sqrt{2}}+\frac{1}{x+\sqrt{2}}+x+1$

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.

$\ln(2\cdot x\cdot y)+\sqrt{2\cdot x\cdot y}$ $\ln(2\cdot x\cdot y)+\sqrt{2\cdot x\cdot y}$

$\text{expand}(Ans)$ $\ln(x\cdot y)+\sqrt{2}\cdot\sqrt{x\cdot y}+\ln(2)$

$\text{expand}(Ans)\,|\,y\geq 0$ $\ln(x)+\sqrt{2}\cdot\sqrt{x}\cdot\sqrt{y}+\ln(y)+\ln(2)$

$\text{sign}(x\cdot y)+|x\cdot y|+e^{2\cdot x+y}$ $e^{2\cdot x+y}+\text{sign}(x\cdot y)+|x\cdot y|$

$\text{expand}(Ans)$ $\text{sign}(x)\cdot\text{sign}(y)+|x|\cdot|y|+\left(e^x\right)^2\cdot e^y$

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

Hinweis: Siehe auch **tExpand()** zur 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 193.)

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 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: $a \cdot (b)^x$
stat.a, stat.b	Regressionskoeffizienten

Ausgabevariable	Beschreibung
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 *X List*, die schließlich in der Regression mit den Beschränkungen für *Häufigkeit*, *Kategorieliste* und *Mit Kategorien* verwendet wurde
stat.YReg	Liste der Datenpunkte in der modifizierten *Y List*, die schließlich in der Regression mit den Beschränkungen für *Häufigkeit*, *Kategorieliste* und *Mit Kategorien* verwendet wurde
stat.FreqReg	Liste der Häufigkeiten für *stat.XReg* und *stat.YReg*

F

factor() (Faktorisiere) — Katalog >

factor(*Ausdr1***[, *Var*])⇒*Ausdruck***

factor(*Liste1***[,*Var*])⇒*Liste***

factor(*Matrix1***[,*Var*])⇒*Matrix***

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

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.

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 werden.

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

$\text{factor}(x^2+1)$ → x^2+1

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

$\text{factor}(x^2-3)$ → x^2-3

$\text{factor}(x^2-a)$ → x^2-a

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

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

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

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.

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

$x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3$

$\text{factor}(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3,x)$

$(x-0.964673)\cdot(x+0.611649)\cdot(x+2.12543)\cdot(x^$

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.

$\text{factor}(152417172689)$ $123457\cdot1234577$

$\text{isPrime}(152417172689)$ false

So halten Sie eine Berechnung manuell an:

- **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*.

FCdf() Katalog >

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

Fill (Füllen) Katalog >

Fill $Ausdr, MatrixVar \Rightarrow Matrix$

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

$MatrixVar$ muss bereits vorhanden sein.

Fill $Ausdr, ListeVar \Rightarrow Liste$

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

$ListeVar$ muss bereits vorhanden sein.

$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} \rightarrow amatrix$ $\quad$ $\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$

Fill 1.01,*amatrix* $\quad$ *Done*

amatrix $\quad$ $\begin{bmatrix}1.01 & 1.01\\1.01 & 1.01\end{bmatrix}$

$\{1,2,3,4,5\} \rightarrow alist$ $\quad$ $\{1,2,3,4,5\}$

Fill 1.01,*alist* $\quad$ *Done*

alist $\quad$ $\{1.01,1.01,1.01,1.01,1.01\}$

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 193.)

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.

FiveNumSummary

Katalog >

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 259).

Ausgabevariable	Beschreibung
stat.MinX	Minimum der x-Werte
stat.Q_1X	1. Quartil von x
stat.MedianX	Median von x
stat.Q_3X	3. Quartil von x
stat.MaxX	Maximum der x-Werte

floor() (Untergrenze)

Katalog >

floor(*Ausdr1***)⇒***Ganzzahl*

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

Das Argument kann eine reelle oder eine komplexe Zahl sein.

floor(*Liste1***)⇒***Liste*

floor(*Matrix1***)⇒***Matrix*

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

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

$\text{floor}(-2.14)$ $-3.$

$\text{floor}\left(\left\{\frac{3}{2},0,-5.3\right\}\right)$ $\{1.,0.,-6.\}$

$\text{floor}\begin{bmatrix}1.2 & 3.4\\ 2.5 & 4.8\end{bmatrix}$ $\begin{bmatrix}1. & 3.\\ 2. & 4.\end{bmatrix}$

fMax() (Funktionsmaximum)

Katalog >

fMax(*Ausdr, Var***)⇒***Boolescher Ausdruck*

fMax(*Ausdr***,** *Var***,***UntereGrenze***)**

fMax(*Ausdr***,** *Var***,***UntereGrenze***,***ObereGrenze***)**

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

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

$\text{fMax}\left(.5\cdot x^3-x-2,x\right)$ $x=\infty$

fMax() (Funktionsmaximum) Katalog >

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.

$\text{fMax}(0.5 \cdot x^3 - x - 2, x) \mid x \le 1 \qquad x = -0.816497$

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) Katalog >

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.

$\text{fMin}(1-(x-a)^2-(x-b)^2, x) \qquad x = -\infty \text{ or } x = \infty$

$\text{fMin}(0.5 \cdot x^3 - x - 2, x) \mid x \ge 1 \qquad x = 1.$

fMin() (Funktionsminimum) Katalog >

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 kleinesIntervall zu beschränken, das genau ein lokales Minimum enthält.

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

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.

Define $g()$=Func Local *tempsum*,*step*,*i* $0 \rightarrow tempsum$ $1 \rightarrow step$ For *i*,1,100,*step* $tempsum+i \rightarrow tempsum$ EndFor EndFunc	*Done*
$g()$	5050

format() (Format) Katalog >

format(*Ausdr*[, *FormatString*]**)**⇒*String*

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

Ausdr muss zu einer Zahl vereinfachbar sein.

format(1.234567,"f3")	"1.235"
format(1.234567,"s2")	"1.23E0"
format(1.234567,"e3")	"1.235E0"
format(1.234567,"g3")	"1.235"
format(1234.567,"g3")	"1,234.567"
format(1.234567,"g3,r:")	"1:235"

format() (Format) Katalog >

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 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(*Ausdr1***)⇒***Ausdruck*

fPart(*Liste1***)⇒***Liste*

fPart(*Matrix1***)⇒***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.

fPart(-1.234)	-0.234
fPart({1,-2.3,7.003})	{0,-0.3,0.003}

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.

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 259).

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}

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.

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)<**?**≤b(2)), …, countIf(Liste, b(n-1)<**?**≤b(n)), countIf(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 259).

Innerhalb der Lists & Spreadsheet Applikation können Sie für beide Argumente Zellenbereiche verwenden.

Hinweis: Siehe auch **countIf()**, Seite 42.

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}

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 platziert werden.

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)

F Test_2Samp (Zwei-Stichproben F-Test)

Katalog >

Führt einen F -Test mit zwei Stichproben durch. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 193.)

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 259).

Ausgabevariable	Beschreibung
Statistik.F	Berechnete Û Statistik für die Datenfolge
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.dfNumer	Freiheitsgrade des Zählers = n1-1
stat.dfDenom	Freiheitsgrade des Nenners = n2-1
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in *Liste 1* und *Liste 2*
stat.x1_bar stat.x2_bar	Stichprobenmittelwerte der Datenfolgen in *Liste 1* und *Liste 2*
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.

Definieren Sie eine stückweise definierte Funktion:

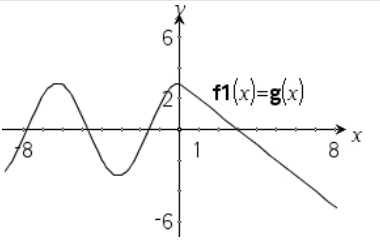
```
Define g(x)=Func                    Done
              If x<0 Then
              Return 3·cos(x)
              Else
              Return 3−x
              EndIf
              EndFunc
```

Ergebnis der graphischen Darstellung g(x)

Func

Katalog >

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



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.

In den Modi Auto oder Approximiert ist der **gcd** von Fließkommabrüchen 1,0.

gcd(*Liste1, Liste2***)⇒***Liste*

$\gcd(\{12,14,16\},\{9,7,5\})$ → $\{3,7,1\}$

Gibt die größten gemeinsamen Teiler der einander entsprechenden Elemente von *Liste1* und *Liste2* zurück.

gcd(*Matrix1, Matrix2***)⇒***Matrix*

$\gcd\left(\begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix},\begin{bmatrix}4 & 8\\ 12 & 16\end{bmatrix}\right)$ → $\begin{bmatrix}2 & 4\\ 6 & 8\end{bmatrix}$

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≤X ≤*obereGrenze*)⇒*Zahl*, wenn *obereGrenze* eine Zahl ist, *Liste*, wenn *obereGrenze* eine Liste ist

geomCdf() Katalog >

Berechnet die kumulative geometrische Wahrscheinlichkeit von *UntereGrenze* bis *ObereGrenze* mit der angegebenen Erfolgswahrscheinlichkeit *p*.

Für P(X ≤ *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 Hub-Menü

Get[*EingabeString*,]*Var*[**,** *statusVar*]

Get[*EingabeString*,] *Fkt***(***arg1***,** *...argn***)** [**,** *statusVar*]

Programmierbefehl: Ruft einen Wert von einem verbundenen TI-Innovator™ Hub ab und weist den Wert der Variablen *var* zu.

Der Wert muss angefordert werden:

- Im Voraus durch einen Befehl **Send "READ ..."** .

  – oder –

- Durch Einbetten einer Anforderung **"READ ..."** als optionales Argument von *promptString*. Bei dieser Methode können Sie einen einzelnen Befehl verwenden, um den Wert anzufordern und abzurufen.

Beispiel: Fordern Sie den aktuellen Wert des integrierten Lichtpegelsensors des Hub an. Verwenden Sie **Get**, um den Wert abzurufen, und weisen Sie ihn der Variablen *lightval* zu.

Send "READ BRIGHTNESS"	*Done*
Get *lightval*	*Done*
lightval	0.347922

Betten Sie die Anforderung READ in den Befehl **Get** ein.

Get "READ BRIGHTNESS",*lightval*	*Done*
lightval	0.378441

Implizite Vereinfachung findet statt. Zum Beispiel wird eine empfangene Zeichenfolge „123“ als numerischer Wert interpretiert. Um die Zeichenfolge beizubehalten, verwenden Sie **GetStr** statt **Get**.

Wenn Sie das optionale Argument von *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 von *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 einem benutzerdefinierten Programm, aber nicht in einer Funktion verwenden.

Hinweis: Siehe auch **GetStr**, Seite 93 und **Send**, Seite 172.

getDenom() (Nenner holen) Katalog >

getDenom(*Ausdr1***)⇒***Ausdruck*

Transformiert das Argument in einen Ausdruck mit gekürztem gemeinsamem Nenner und gibt dann den Nenner zurück.

$\text{getDenom}\left(\frac{x+2}{y-3}\right)$ → $y-3$

$\text{getDenom}\left(\frac{2}{7}\right)$ → 7

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

getLangInfo() Katalog >

getLangInfo()⇒*Zeichenkette*

getLangInfo() → "en"

getLangInfo() Katalog >

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 116, und**unLock**, Seite 216.

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(*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 176.

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
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.

$\text{getNum}\left(\frac{x+2}{y-3}\right)$	$x+2$
$\text{getNum}\left(\frac{2}{7}\right)$	2
$\text{getNum}\left(\frac{1}{x}+\frac{1}{y}\right)$	$x+y$

GetStr — Hub-Menü

GetStr[*EingabeString*,] *Var*[**,** *statusVar*]

GetStr[*EingabeString*,] *Fkt***(***arg1***,** *...argn***)** [**,** *statusVar*]

Programmierbefehl: Verhält sich genauso wie der Befehl **Get**, der abgerufene Wert wird aber immer als Zeichenfolge interpretiert. Der Befehl **Get** interpretiert die Antwort hingegen als Ausdruck, es sei denn, sie ist in Anführungszeichen ("") gesetzt.

Hinweis: Siehe auch **Get**, Seite 89 und **Send**, Seite 172.

Zum Beispiel siehe **Get**.

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.

$\{1,2,3\}\rightarrow temp$	$\{1,2,3\}$
$\text{getType}(temp)$	"LIST"
$3\cdot i\rightarrow temp$	$3\cdot i$
$\text{getType}(temp)$	"EXPR"
DelVar *temp*	*Done*
$\text{getType}(temp)$	"NONE"

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.

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.

getVarInfo()	"NONE"
Define $x=5$	*Done*
Lock x	*Done*
Define LibPriv $y=\{1,2,3\}$	*Done*
Define LibPub $z(x)=3\cdot x^2-x$	*Done*
getVarInfo()	$\begin{bmatrix} x & \text{"NUM"} & \text{"}\square\text{"} & 1 \\ y & \text{"LIST"} & \text{"LibPriv "} & 0 \\ z & \text{"FUNC"} & \text{"LibPub "} & 0 \end{bmatrix}$
getVarInfo(*tmp3*)	"Error: Argument must be a string"
getVarInfo("tmp3")	$\begin{bmatrix} volcyl2 & \text{"NONE"} & \text{"LibPub "} & 0 \end{bmatrix}$

$a:=1$	1
$b:=[1\ \ 2]$	$[1\ \ 2]$
$c:=[1\ \ 3\ \ 7]$	$[1\ \ 3\ \ 7]$
$vs:=$getVarInfo()	$\begin{bmatrix} a & \text{"NUM"} & \text{"}\square\text{"} & 0 \\ b & \text{"MAT"} & \text{"}\square\text{"} & 0 \\ c & \text{"MAT"} & \text{"}\square\text{"} & 0 \end{bmatrix}$
$vs[1]$	$[1\ \ \text{"NUM"}\ \ \text{"}\square\text{"}\ \ 0]$
$vs[1,1]$	1
$vs[2]$	"Error: Invalid list or matrix"
$vs[2,1]$	$[1\ \ 2]$

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 Beispiels: 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*

Wandelt *Ausdr1* ins Winkelmaß Neugrad um.

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

Im Grad-Modus:

```
(1.5)▸Grad                    (1.66667)ᵍ
```

Im Bogenmaß-Modus:

```
(1.5)▸Grad                    (95.493)ᵍ
```

I

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 *Boolescher Ausdr*
Anweisung

If *Boolescher Ausdr* **Then**
Block
EndIf

Wenn *Boolescher Ausdr* wahr ergibt, wird die Einzelanweisung *Anweisung* oder der Anweisungsblock *Block* ausgeführt und danach mit EndIf fortgefahren.

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.

```
Define g(x)=Func                    Done
              If x<0 Then
              Return x^2
              EndIf
              EndFunc
g(-2)                               4
```

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.

```
Define g(x)=Func                    Done
              If x<0 Then
              Return -x
              Else
              Return x
              EndIf
              EndFunc
g(12)                               12
g(-12)                              12
```

If

Katalog >

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)=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
g(-4)                                  4
g(10)                                  3
```

ifFn()

Katalog >

ifFn(*BoolescherAusdruck***,***Wert_wenn_wahr* **[,***Wert_wenn_falsch* **[,***Wert_wenn_unbekannt***]])⇒***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.
- 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*

```
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 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.

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 dieselbe(n) Dimension(en) haben, und auch das Ergebnis wird dieselben(n) Dimension(en) haben.

$\text{ifFn}(\{1,2,3\}<2.5,\{5,6,7\})$ $\{5,6,\text{undef}\}$

Wert_wenn_falsch ist nicht spezifiziert. Undef wird verwendet.

$\text{ifFn}(\{2,"a"\}<2.5,\{6,7\},\{9,10\},"err")$ $\{6,"err"\}$

Ein aus *Wert_wenn_wahr* ausgewähltes Element. Ein aus *Wert_wenn_unbekannt* ausgewähltes Element.

imag() (Imaginärteil) Katalog >

imag(*Ausdr1***)⇒***Ausdruck*

Gibt den Imaginärteil des Arguments zurück.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt. Siehe auch **real()**, Seite 159

$\text{imag}(1+2\cdot i)$ 2

$\text{imag}(z)$ 0

$\text{imag}(x+i\cdot y)$ y

imag(*Liste1***)⇒***Liste*

Gibt eine Liste der Imaginärteile der Elemente zurück.

$\text{imag}(\{-3,4-i,i\})$ $\{0,-1,1\}$

imag(*Matrix1***)⇒***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*

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.

$\text{impDif}(x^2+y^2=100,x,y)$ $\frac{-x}{y}$

Umleitung **Siehe #(), Seite 249.**

inString() (In String) Katalog >

inString(*Quellstring*, *Teilstring*[, *Start*])⇒*Ganzzahl*

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.

inString("Hello there","the")	7
inString("ABCEFG","D")	0

int() (Ganze Zahl) Katalog >

int(*Ausdr***)⇒***Ganzzahl*

int(*Liste1***)⇒***Liste*

int(*Matrix1***)⇒***Matrix*

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.

int(-2.5)	-3.
int([-1.234 0 0.37])	[-2. 0 0.]

intDiv() (Ganzzahl teilen) Katalog >

intDiv(*Zahl1*, *Zahl2***)⇒***Ganzzahl*

intDiv(*Liste1*, *Liste2***)⇒***Liste*

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

intDiv(-7,2)	-3
intDiv(4,5)	0
intDiv({12,-14,-16},{5,4,-3})	{2,-3,5}

intDiv() (Ganzzahl teilen) Katalog >

Gibt den mit Vorzeichen versehenen ganzzahligen Teil von (*Zahl1* ÷ *Zahl2*) zurück.

Für eine Liste oder Matrix wird für jedes Elementpaar der mit Vorzeichen versehene ganzzahlige Teil von (Argument 1 ÷ Argument 2) zurückgegeben.

integral Siehe ∫(), Seite 231.

interpolate () Katalog >

interpolate(*xWert*, *xListe*, *yListe*, *yStrListe***)⇒***Liste*

Diese Funktion tut folgendes:

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 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.

Differentialgleichung:

$y'=-3\cdot y+6\cdot t+5$ und $y(0)=5$

$rk:=\text{rk23}(-3\cdot y+6\cdot t+5,t,y,\{0,10\},5,1)$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 5. & 3.19499 & 5.00394 & 6.99957 & 9.00593 \end{bmatrix}$$

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

Verwenden Sie die Funktion interpolate(), um die Funktionswerte für die Liste xWert zu berechnen:

$xvaluelist:=\text{seq}(i,i,0,10,0.5)$

$\{0,0.5,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,5.5,6.,6.5,$

$xlist:=\text{mat}\blacktriangleright\text{list}(rk[1])$

$\{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.\}$

$ylist:=\text{mat}\blacktriangleright\text{list}(rk[2])$

$\{5.,3.19499,5.00394,6.99957,9.00593,10.9978$

$yprimelist:=-3\cdot y+6\cdot t+5|y=ylist \text{ and } t=xlist$

$\{-10.,1.41503,1.98819,2.00129,1.98221,2.006$

$\text{interpolate}(xvaluelist,xlist,ylist,yprimelist)$

$\{5.,2.67062,3.19499,4.02782,5.00394,6.00011$

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 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(*Liste1***)⇒***Liste*

iPart(*Matrix1***)⇒***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.

iPart(-1.234)	-1.
iPart({3/2,-2.3,7.003})	{1,-2.,7.}

irr() Katalog >

irr(*CF0,CFListe* [,*CFFreq*]**)⇒***Wert*

Finanzfunktion, die den internen Zinsfluss einer Investition berechnet.

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 Ganzzahlen < 10.000 sein.

Hinweis: Siehe auch **mirr()**, Seite 126.

list1:={6000,-8000,2000,-3000}	{6000,-8000,2000,-3000}
list2:={2,2,2,1}	{2,2,2,1}
irr(5000,*list1*,*list2*)	-4.64484

isPrime() (Primzahltest) Katalog >

isPrime(*Zahl***)⇒***Boolescher konstanter Ausdruck*

Gibt "wahr" oder "falsch" zurück, um anzuzeigen, ob es sich bei *Zahl* um eine ganze Zahl ≥ 2 handelt, die nur durch sich selbst oder 1 ganzzahlig teilbar ist.

isPrime(5)	true
isPrime(6)	false

Funktion zum Auffinden der nächsten Primzahl nach einer angegebenen Zahl:

isPrime() (Primzahltest) Katalog >

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

```
Define nextprim(n)=Func                    Done
                  Loop
                  n+1→n
                  If isPrime(n)
                  Return n
                  EndLoop
                  EndFunc
nextprim(7)                                11
```

isVoid() Katalog >

isVoid(*Var***)⇒***Boolescher konstanter Ausdruck*

isVoid(*Ausdr***)⇒***Boolescher konstanter Ausdruck*

isVoid(*Liste***)⇒***Liste Boolescher konstanter Ausdrücke*

Gibt wahr oder falsch zurück, um anzuzeigen, ob das Argument ein ungültiger Datentyp ist.

Weitere Informationen zu ungültigen Elementen finden Sie (Seite 259).

```
a:=_                                       _
isVoid(a)                                  true
isVoid({1,_,3})                            {false,true,false}
```

Lbl (Marke)

Katalog >

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 Variablennamen.

Hinweis zum Eingeben des Beispiels: 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
```

lcm() (Kleinstes gemeinsames Vielfaches)

Katalog >

lcm(*Zahl1***,** *Zahl2***)⇒***Ausdruck*

lcm(*Liste1***,** *Liste2***)⇒***Liste*

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.

```
lcm(6,9)                                          18
lcm({1/3,-14,16},{2/15,7,5})          {2/3,14,80}
```

left() (Links)

Katalog >

left(*Quellstring***[,** *Anz***])⇒***String*

```
left("Hello",2)                                   "He"
```

left() (Links) Katalog >

Gibt *Anz* Zeichen zurück, die links in der Zeichenkette *Quellstring* enthalten sind.

Wenn Sie *Anz* weglassen, wird der gesamte *Quellstring* zurückgegeben.

left(*Liste1*[**,** *Anz*]**)**⇒*Liste*

Gibt *Anz* Elemente zurück, die links in *Liste1* enthalten sind.

Wenn Sie *Anz* weglassen, wird die gesamte *Liste1* zurückgegeben.

left(*Vergleich***)**⇒*Ausdruck*

Gibt die linke Seite einer Gleichung oder Ungleichung zurück.

left({1,3,-2,4},3) {1,3,-2}

left(x<3) x

libShortcut() Katalog >

libShortcut(*BiblioNameString***,** *VerknNameString*

[**,** *BiblioPrivMerker*]**)**⇒*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 36).

Informationen zum Löschen einer Variablengruppe finden Sie unter **DelVar** (Seite 58).

Dieses Beispiel setzt ein richtig gespeichertes und aktualisiertes Bibliotheksdokument namens **linalg2** voraus, das als *clearmat*, *gauss1* und *gauss2* definierte Objekte enthält.

getVarInfo("linalg2")

clearmat	"FUNC"	"LibPub "
gauss1	"PRGM"	"LibPriv "
gauss2	"FUNC"	"LibPub "

libShortcut("linalg2","la") {*la.clearmat*,*la.gauss2*}

libShortcut("linalg2","la",1) {*la.clearmat*,*la.gauss1*,*la.gauss2*}

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.

$\lim_{x\to 5}(2\cdot x+3)$	13
$\lim_{x\to 0^+}\left(\frac{1}{x}\right)$	∞
$\lim_{x\to 0}\left(\frac{\sin(x)}{x}\right)$	1
$\lim_{h\to 0}\left(\frac{\sin(x+h)-\sin(x)}{h}\right)$	$\cos(x)$
$\lim_{n\to\infty}\left(\left(1+\frac{1}{n}\right)^n\right)$	*e*

$\lim_{x\to\infty}(a^x)$	undef
$\lim_{x\to\infty}(a^x)\mid a>1$	∞
$\lim_{x\to\infty}(a^x)\mid a>0 \text{ and } a<1$	0

limit() oder lim() (Limes) Katalog >

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 müssten, dies nicht sind und umgekehrt endliche Grenzwerte ungleich Null nicht erkannt werden.

LinRegBx Katalog >

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 193.)

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 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: a+b ·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 *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*

LinRegMx

Katalog >

LinRegMx *X*,*Y*[,[*Häuf*][,*Kategorie*,*Mit*]]

Berechnet die lineare Regression y = m ·x+b auf Liste *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

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.

LinRegMx

Katalog >

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: m ·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 *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*

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 193.)

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 Ganzzahlen ≥ 0 sein.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: a+b ·x
stat.a, stat.b	Regressionskoeffizienten
stat.df	Freiheitsgrade
stat.r^2	Bestimmungskoeffizient
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

Ausgabevariable	Beschreibung
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 ($H_0:\beta=\rho=0$) untersucht wird.

Für H_a: β 0 und ρ 0 (Standard) setzen Sie *Hypoth*=0

Für H_a: $\beta<0$ und $\rho<0$ setzen Sie *Hypoth*<0

Für H_a: $\beta>0$ und $\rho>0$ setzen Sie *Hypoth*>0

Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: a + b · 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*, ...**)**⇒*Liste*

linSolve(*LineareGl1* **and** *LineareGl2* **and** ..., *Var1*, *Var2*, ...**)**⇒*Liste*

linSolve({*LineareGl1*, *LineareGl2*, ...**},** *Var1*, *Var2*, ...**)** ⇒*Liste*

linSolve(*SystemLinearerGl*, **{***Var1*, *Var2*, ...**})** ⇒*Liste*

linSolve(*LineareGl1* **and** *LineareGl2* **and** ..., **{***Var1*, *Var2*, ...**})**⇒*Liste*

linSolve({*LineareGl1*, *LineareGl2*, ...**},** **{***Var1*, *Var2*, ...**})** ⇒*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. Anderenfalls tritt ein Argumentfehler auf.

Die Auswertung von **linSolve**(x=1 and x=2,x) führt beispielsweise zu dem Ergebnis "Argumentfehler" .

$\text{linSolve}\left(\begin{Bmatrix}2\cdot x+4\cdot y=3\\5\cdot x-3\cdot y=7\end{Bmatrix},\{x,y\}\right)\quad\left\{\frac{37}{26},\frac{1}{26}\right\}$

$\text{linSolve}\left(\begin{Bmatrix}2\cdot x=3\\5\cdot x-3\cdot y=7\end{Bmatrix},\{x,y\}\right)\quad\left\{\frac{3}{2},\frac{1}{6}\right\}$

$\text{linSolve}\left(\begin{Bmatrix}apple+4\cdot pear=23\\5\cdot apple-pear=17\end{Bmatrix},\{apple,pear\}\right)\quad\left\{\frac{13}{3},\frac{14}{3}\right\}$

$\text{linSolve}\left(\begin{Bmatrix}apple\cdot 4+\frac{pear}{3}=14\\-apple+pear=6\end{Bmatrix},\{apple,pear\}\right)\quad\left\{\frac{36}{13},\frac{114}{13}\right\}$

Δlist() (Listendifferenz)

Katalog >

Δlist(*Liste1***)⇒***Liste*

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*.

$\Delta\text{List}(\{20,30,45,70\})$ $\{10,15,25\}$

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.

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.

$\text{list}\blacktriangleright\text{mat}(\{1,2,3\})$ $\begin{bmatrix}1 & 2 & 3\end{bmatrix}$

$\text{list}\blacktriangleright\text{mat}(\{1,2,3,4,5\},2)$ $\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 0\end{bmatrix}$

▸ln (Natürlicher Logarithmus)

Katalog >

Ausdr **▸ln⇒***Ausdruck*

Führt dazu, dass der eingegebene *Ausdr* in einen Ausdruck umgewandelt wird, der nur natürliche Logarithmen (ln) enthält.

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

$\left(\log_{10}(x)\right)\blacktriangleright\ln$ $\frac{\ln(x)}{\ln(10)}$

ln() (Natürlicher Logarithmus)

ctrl **e^x** **Tasten**

ln(*Ausdr1***)⇒***Ausdruck*

ln(*Liste1***)⇒***Liste*

Gibt den natürlichen Logarithmus des Arguments zurück.

Gibt für eine Liste die natürlichen Logarithmen der einzelnen Elemente zurück.

$\ln(2.)$ → 0.693147

Bei Komplex-Formatmodus reell:

$\ln(\{-3,1.2,5\})$ → "Error: Non−real calculation"

Bei Komplex-Formatmodus kartesisch:

$\ln(\{-3,1.2,5\})$ → $\{\ln(3)+\pi\cdot i, 0.182322, \ln(5)\}$

ln(*Quadratmatrix1***)⇒***Quadratmatrix*

Ergibt den natürlichen Matrix-Logarithmus von *Quadratmatrix1*. Dies ist nicht gleichbedeutend mit der Berechnung des natürlichen Logarithmus jedes einzelnen Elements. Näheres zum Berechnungsverfahren finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\ln\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}1.83145+1.73485\cdot i & 0.009193-1.49086\\ 0.448761-0.725533\cdot i & 1.06491+0.623491\\ -0.266891-2.08316\cdot i & 1.12436+1.79018\cdot\end{bmatrix}$$

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·ln(x) auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

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 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: a+b·ln(x)
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 >

Lock*Var1* [**,** *Var2*] [**,** *Var3*] ...

Lock*Var***.**

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 216, und getLockInfo(), Seite 91.**

```
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 ctrl 10^x

log(*Ausdr1*[**,***Ausdr2*]**)⇒***Ausdruck*

log(*Liste1*[**,***Ausdr2*]**)⇒***Liste*

Gibt für den Logarithmus des Arguments zur Basis *Ausdr2* zurück.

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.

$\log_{10}(2.)$ 0.30103

$\log_{4}(2.)$ 0.5

$\log_{3}(10)-\log_{3}(5)$ $\log_{3}(2)$

Bei Komplex-Formatmodus reell:

$\log_{10}(\{-3,1.2,5\})$ Error: *Non–real result*

Bei Komplex-Formatmodus kartesisch:

$\log_{10}(\{-3,1.2,5\})$

$\{\log_{10}(3)+1.36438\cdot i, 0.079181, \log_{10}(5)\}$

log(*Quadratmatrix1*
[**,***Ausdr2*]**)⇒***Quadratmatrix*

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.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\log_{10}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$

$$\begin{bmatrix}0.795387+0.753438\cdot i & 0.003993-0.6474\\ 0.194895-0.315095\cdot i & 0.462485+0.2707\\ -0.115909-0.904706\cdot i & 0.488304+0.7774\end{bmatrix}$$

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

▶logbase

Katalog >

Ausdr1 **▶logbase(***Ausdr2***)⇒***Ausdruck*

Führt dazu, dass der eingegebene Ausdruck zu einem Ausdruck mit der Basis *Ausdr2* vereinfacht wird.

$\log_{3}(10)-\log_{5}(5)\blacktriangleright\text{logbase}(5)$ $\dfrac{\log_{5}\left(\frac{10}{3}\right)}{\log_{5}(3)}$

▸logbase

Katalog >

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 Regressiony = (c/(1+a ·e^{-bx}))auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

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 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: c/(1+a ·e^{-bx})
stat.a, stat.b, stat.c	Regressionskoeffizienten
stat.Resid	Residuen von der Regression

Ausgabevariable	Beschreibung
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*

LogisticD

Katalog >

LogisticD *X*, *Y* [, [*Iterationen*], [*Häuf*] [, *Kategorie*, *Mit*]]

Berechnet die logistische Regression y = (c/(1+a ·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 193.)

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 259).

Ausgabevariable	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 *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*

Loop (Schleife) Katalog >

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 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
```

LU (Untere/obere Matrixzerlegung) Katalog >

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 · *uMatrix* = *pMatrix* · *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. Anderenfalls wird *Tol* ignoriert.

- 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 ·max(dim(*Matrix*)) ·rowNorm (*Matrix*)

Der **LU**-Faktorisierungsalgorithmus verwendet partielle Pivotisierung mit Zeilentausch.

$\begin{bmatrix}6 & 12 & 18\\5 & 14 & 31\\3 & 8 & 18\end{bmatrix} \rightarrow m1$ $\quad \begin{bmatrix}6 & 12 & 18\\5 & 14 & 31\\3 & 8 & 18\end{bmatrix}$

LU *m1*,*lower*,*upper*,*perm* — *Done*

lower $\quad \begin{bmatrix}1 & 0 & 0\\\frac{5}{6} & 1 & 0\\\frac{1}{2} & \frac{1}{2} & 1\end{bmatrix}$

upper $\quad \begin{bmatrix}6 & 12 & 18\\0 & 4 & 16\\0 & 0 & 1\end{bmatrix}$

perm $\quad \begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$

$\begin{bmatrix}m & n\\o & p\end{bmatrix} \rightarrow m1$ $\quad \begin{bmatrix}m & n\\o & p\end{bmatrix}$

LU *m1*,*lower*,*upper*,*perm* — *Done*

lower $\quad \begin{bmatrix}1 & 0\\\frac{m}{o} & 1\end{bmatrix}$

upper $\quad \begin{bmatrix}o & p\\0 & n-\frac{m\cdot p}{o}\end{bmatrix}$

perm $\quad \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$

M

mat▸list() (Matrix in Liste) Katalog >

mat▸list(*Matrix***)⇒***Liste*

mat▸list([1 2 3]) — {1,2,3}

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

mat▸list(*m1*) — {1,2,3,4,5,6}

mat▸list() (Matrix in Liste) Katalog >

Gibt eine Liste zurück, die mit den Elementen aus *Matrix* gefüllt wurde. Die Elemente werden Zeile für Zeile aus *Matrix* kopiert.

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*

Gibt das größte Element von *Liste* zurück.

$\max(\{0,1,-7,1.3,0.5\})$ → 1.3

max(*Matrix1***)⇒***Matrix*

Gibt einen Zeilenvektor zurück, der das größte Element jeder Spalte von *Matrix1* enthält.

$\max\begin{pmatrix}1 & -3 & 7\\ -4 & 0 & 0.3\end{pmatrix}$ → $[1\ 0\ 7]$

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 259).

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

mean() (Mittelwert) Katalog >

mean(*Liste*[**,** *Häufigkeitsliste*]**)⇒***Ausdruck*

Gibt den Mittelwert der Elemente in *Liste* zurück.

$\text{mean}(\{0.2,0,1,-0.3,0.4\})$ → 0.26

$\text{mean}(\{1,2,3\},\{3,2,1\})$ → $\frac{5}{3}$

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

mean() (Mittelwert) Katalog >

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 259).

Im Vektorformat kartesisch:

$$\text{mean}\begin{bmatrix}0.2 & 0\\ -1 & 3\\ 0.4 & -0.5\end{bmatrix} \quad \begin{bmatrix}-0.133333 & 0.833333\end{bmatrix}$$

$$\text{mean}\begin{bmatrix}\frac{1}{5} & 0\\ -1 & 3\\ \frac{2}{5} & \frac{-1}{2}\end{bmatrix} \quad \begin{bmatrix}\frac{-2}{15} & \frac{5}{6}\end{bmatrix}$$

$$\text{mean}\left(\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix},\begin{bmatrix}5 & 3\\ 4 & 1\\ 6 & 2\end{bmatrix}\right) \quad \begin{bmatrix}\frac{47}{15} & \frac{11}{3}\end{bmatrix}$$

median() (Median) Katalog >

median(*Liste*[**,** *freqList*]**)**⇒*Ausdruck*

Gibt den Medianwert der Elemente in *Liste* zurück.

Jedes *freqList*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

$$\text{median}(\{0.2,0,1,-0.3,0.4\}) \quad 0.2$$

median(*Matrix1*[**,** *freqMatrix*]**)**⇒*Matrix*

Gibt einen Zeilenvektor zurück, der die Medianwerte der einzelnen Spalten von *Matrix1*enthält.

Jedes *freqMatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

$$\text{median}\begin{bmatrix}0.2 & 0\\ 1 & -0.3\\ 0.4 & -0.5\end{bmatrix} \quad \begin{bmatrix}0.4 & -0.3\end{bmatrix}$$

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 259).

MedMed Katalog >

MedMed *X*,*Y* [**,** *Häuf*] [**,** *Kategorie*, *Mit*]]

Berechnet die Median-Median-Liniey = (m ·x+b)auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

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 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Median-Median-Linien-Gleichung: m ·x+b
stat.m, stat.b	Modellkoeffizienten
stat.Resid	Residuen von der Median-Median-Linie
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*

mid() (Teil-String)

Katalog >

mid(*Quellstring*, *Start*[, *Anzahl*])⇒*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("Hello there",2)          "ello there"
mid("Hello there",7,3)        "the"
mid("Hello there",1,5)        "Hello"
mid("Hello there",1,0)        ""
```

mid(*Quellliste*, *Start* [, *Anzahl*])⇒*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({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(*QuellstringListe*, *Start*[, *Anzahl*])⇒*Liste*

Gibt *Anzahl* Strings aus der Stringliste *QuellstringListe* ab dem Element mit der Nummer *Start* zurück.

```
mid({"A","B","C","D"},2,2)
                          {"B","C"}
```

min() (Minimum)

Katalog >

min(*Ausdr1*, *Ausdr2*)⇒*Ausdruck*

min(*Liste1, Liste2*)⇒*Liste*

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(2.3,1.4)                  1.4
min({1,2},{-4,3})             {-4,2}
```

min() (Minimum) Katalog >

min(*Liste***)⇒***Ausdruck***

Gibt das kleinste Element von *Liste* zurück.

$\min(\{0,1,-7,1.3,0.5\})$ -7

min(*Matrix1***)⇒***Matrix***

Gibt einen Zeilenvektor zurück, der das kleinste Element jeder Spalte von *Matrix1* enthält.

$\min\left(\begin{bmatrix}1 & -3 & 7\\ -4 & 0 & 0.3\end{bmatrix}\right)$ $\begin{bmatrix}-4 & -3 & 0.3\end{bmatrix}$

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

mirr() Katalog >

mirr
(
Finanzierungsrate
,*Reinvestitionsrate***,***CF0***,***CFListe*
[,*CFFreq***])**

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 Ganzzahlen < 10.000 sein.

$list1:=\{6000,-8000,2000,-3000\}$ $\{6000,-8000,2000,-3000\}$

$list2:=\{2,2,2,1\}$ $\{2,2,2,1\}$

$\text{mirr}(4.65,12,5000,list1,list2)$ 13.41608607

Hinweis: Siehe auch **irr()**, Seite 102.

mod() (Modulo) Katalog >

mod(*Ausdr1*, *Ausdr2*)⇒*Ausdruck*

mod(*Liste1*, *Liste2*)⇒*Liste*

mod(*Matrix1*, *Matrix2*)⇒*Matrix*

Gibt das erste Argument modulo das zweite Argument gemäß der folgenden Identitäten zurück:

mod(x,0) = x

mod(x,y) = x − y 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 Elementpaars enthält.

Hinweis: Siehe auch **remain()**, Seite 162

$\mathrm{mod}(7,0)$	7
$\mathrm{mod}(7,3)$	1
$\mathrm{mod}(-7,3)$	2
$\mathrm{mod}(7,-3)$	-2
$\mathrm{mod}(-7,-3)$	-1
$\mathrm{mod}(\{12,-14,16\},\{9,7,-5\})$	$\{3,0,-4\}$

mRow() (Matrixzeilenoperation) Katalog >

mRow(*Ausdr*, *Matrix1*, *Index*)⇒*Matrix*

Gibt eine Kopie von *Matrix1* zurück, in der jedes Element der Zeile *Index* von *Matrix1* mit *Ausdr* multipliziert ist.

$$\mathrm{mRow}\left(\frac{-1}{3},\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},2\right) \qquad \begin{bmatrix}1 & 2\\ -1 & \frac{-4}{3}\end{bmatrix}$$

mRowAdd() (Matrixzeilenaddition) Katalog >

mRowAdd(*Ausdr*, *Matrix1*, *Index1*, *Index2*) ⇒*Matrix*

Gibt eine Kopie von *Matrix1* zurück, wobei jedes Element in Zeile *Index2* von *Matrix1* ersetzt wird durch:

Ausdr × Zeile *Index1* + Zeile *Index2*

$$\mathrm{mRowAdd}\left(-3,\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},1,2\right) \qquad \begin{bmatrix}1 & 2\\ 0 & -2\end{bmatrix}$$

$$\mathrm{mRowAdd}\left(n,\begin{bmatrix}a & b\\ c & d\end{bmatrix},1,2\right) \qquad \begin{bmatrix}a & b\\ a\cdot n+c & b\cdot n+d\end{bmatrix}$$

MultReg Katalog >

MultReg *Y*, *X1*[,*X2*[,*X3*,…[,*X10*]]]

MultReg

Katalog >

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 193.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: b0+b1 ·x1+b2 ·x2+ ...
stat.b0, stat.b1, ...	Regressionskoeffizienten
stat.R^2	Multiples Bestimmtheitsmaß
stat.$\hat{y}$List	$\hat{y}$List = b0+b1 ·x1+ ...
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 193.)

Alle Listen müssen die gleiche Dimension besitzen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: b0+b1 ·x1+b2 ·x2+ ...
stat.$\hat{y}$	Eine Punktschätzung: $\hat{y}$ = b0 + b1 · xl + ... für *XWertListe*

Ausgabevariable	Beschreibung
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, stat.UpperrPred	Vorhersageintervall für eine einzelne Beobachtung
stat.MEPred	Vorhersageintervall-Fehlertoleranz
stat.SEPred	Standardfehler für Vorhersage
stat.bList	Liste der Regressionskoeffizienten, {b0,b1,b2,…}
stat.Resid	Residuen von der Regression

MultRegTests

Katalog >

MultRegTests *Y*, *X1*[,*X2*[,*X3*,…[,*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 193.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgaben

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: b0+b1 ·x1+b2 ·x2+ …
stat.F	Globale F-Testgröße
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

Ausgabevariable	Beschreibung
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.$\hat{y}$List	$\hat{y}$List = b0+b1 ·x1+ . . .
stat.Resid	Residuen von der Regression
stat.sResid	Standardisierte Residuen; wird durch Division eines Residuums 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

ctrl = Tasten

BoolescherAusdr1 **nand** *BoolescherAusd2*
ergibt *Boolescher Ausdruck*

$x \geq 3$ and $x \geq 4$	$x \geq 4$
$x \geq 3$ nand $x \geq 4$	$x < 4$

BoolescheListe1 **nand** *BoolescheListe2*
ergibt *Boolesche Liste*

BoolescheMatrix1 **nand** *BoolescheMatrix2*
ergibt *Boolesche Matrix*

nand

ctrl **=** **Tasten**

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* ≥ *Ausdr2* ≥ 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(*Ausdr***, 0)** ⇒ **1**

nCr(*Ausdr***,** *negGanzzahl***)** ⇒ **0**

nCr(*Ausdr***,** *posGanzzahl***)** ⇒ *Ausdr* · **(***Ausdr*−1**)... (***Ausdr*−*posGanzzahl***+1)/** *posGanzzahl***!**

nCr(*Ausdr***,** *keineGanzzahl***)** ⇒ *Ausdr***!/**

nCr(z,3)	$\frac{z\cdot(z-2)\cdot(z-1)}{6}$
Ans|z=5	10
nCr(z,c)	$\frac{z!}{c!\cdot(z-c)!}$
$\frac{Ans}{nPr(z,c)}$	$\frac{1}{c!}$

nCr() (Kombinationen)

Katalog >

((
Ausdr−keineGanzzahl)**!** ·*keineGanzzahl***!)**

nCr(*Liste1***,** *Liste2***)⇒***Liste*

Gibt eine Liste von Binomialkoeffizienten auf der Basis der entsprechenden Elementpaare der beiden Listen zurück. Die Argumente müssen Listen gleicher Größe sein.

$\mathrm{nCr}(\{5,4,3\},\{2,4,2\}) \quad \{10,1,3\}$

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.

$$\mathrm{nCr}\left(\begin{bmatrix}6 & 5\\ 4 & 3\end{bmatrix},\begin{bmatrix}2 & 2\\ 2 & 2\end{bmatrix}\right) \quad \begin{bmatrix}15 & 10\\ 6 & 3\end{bmatrix}$$

nDerivative()

Katalog >

nDerivative(*Ausdr1***,***Var***=***Wert*
[**,***Ordnung*]**)⇒***Wert*

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.

$\mathrm{nDerivative}(|x|,x=1) \quad 1$

$\mathrm{nDerivative}(|x|,x)|x=0 \quad \mathrm{undef}$

$\mathrm{nDerivative}(\sqrt{x-1}\,,x)|x=1 \quad \mathrm{undef}$

newList() (Neue Liste)

Katalog >

newList(*AnzElemente***)⇒***Liste*

Gibt eine Liste der Dimension *AnzElemente* zurück. Jedes Element ist Null.

$\mathrm{newList}(4) \quad \{0,0,0,0\}$

newMat() (Neue Matrix)

Katalog >

newMat(*AnzZeil*, *AnzSpalt*)⇒*Matrix*

Gibt eine Matrix der Dimension *AnzZeil* mal *AnzSpalt* zurück, wobei die Elemente Null sind.

$\text{newMat}(2,3)$ → $\begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}$

nfMax() (Numerisches Funktionsmaximum)

Katalog >

nfMax(*Ausdr*, *Var*)⇒*Wert*

nfMax(*Ausdr*, *Var*, *UntereGrenze*)⇒*Wert*

nfMax(*Ausdr*, *Var*, *UntereGrenze*, *ObereGrenze*)⇒*Wert*

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 Invervall [*UntereGrenze*,*ObereGrenze*] für das lokale Maximum.

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

$\text{nfMax}(-x^2-2\cdot x-1,x)$ → $-1.$

$\text{nfMax}(0.5\cdot x^3-x-2,x,-5,5)$ → $5.$

nfMin() (Numerisches Funktionsminimum)

Katalog >

nfMin(*Ausdr*, *Var*)⇒*Wert*

nfMin(*Ausdr*, *Var*, *UntereGrenze*)⇒*Wert*

nfMin(*Ausdr*, *Var*, *UntereGrenze*, *ObereGrenze*)⇒*Wert*

nfMin(*Ausdr*, *Var*) | *UntereGrenze*≤*Var*≤*ObereGrenze*⇒*Wert*

Gibt einen möglichen numerischen Wert der Variablen *Var* zurück, wobei das lokale Minimum von *Ausdr* auftritt.

$\text{nfMin}(x^2+2\cdot x+5,x)$ → $-1.$

$\text{nfMin}(0.5\cdot x^3-x-2,x,-5,5)$ → $-5.$

nfMin() (Numerisches Funktionsminimum)

Katalog >

Wenn Sie *UntereGrenze* und *ObereGrenze* ersetzen, sucht die Funktion in dem geschlossenen Invervall [*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 ∫**(***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.

$\text{nInt}\left(\cos(x),x,-\pi,\pi+1.\text{E}^{-}12\right)$ $-1.04144\text{E}^{-}12$

$\int_{-\pi}^{\pi+10^{-12}} \cos(x)\,dx - \sin\left(\frac{1}{1000000000000}\right)$

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.

$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x\cdot y}}{\sqrt{x^2-y^2}},y,-x,x\right),x,0,1\right)$ 3.30423

Hinweis: Siehe auch ∫**()**, Seite 231.

nom()

Katalog >

nom(*Effektivzins,CpY***)⇒***Wert*

$\text{nom}(5.90398,12)$ 5.75

nom()

Katalog >

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 67.

nor

ctrl = Tasten

BoolescherAusd1 **nor** *BoolescherAusdr2*
ergibt *Boolescher Ausdruck*

BoolescheListe1 **nor** *BoolescheListe2*
ergibt *Boolesche Liste*

BoolescheMatrix1 **nor** *BoolescheMatrix2*
ergibt *Boolesche Matrix*

$x\geq 3$ or $x\geq 4$	$x\geq 3$
$x\geq 3$ nor $x\geq 4$	$x<3$

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.

Ganzzahl1 **nor** *Ganzzahl2* ⇒ *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}

nor

ctrl = Tasten

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***)⇒***Ausdruck*

norm(*Vektor***)⇒***Ausdruck*

Gibt die Frobeniusnorm zurück.

$\text{norm}\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right)$	$\sqrt{a^2+b^2+c^2+d^2}$
$\text{norm}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	$\sqrt{30}$
$\text{norm}\left(\begin{bmatrix} 1 & 2 \end{bmatrix}\right)$	$\sqrt{5}$
$\text{norm}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$	$\sqrt{5}$

normalLine()

Katalog >

normalLine(*Ausdr1***,***Var***,***Punkt***)⇒***Ausdruck*

normalLine
(*Ausdr1***,***Var***=***Punkt***)⇒***Ausdruck*

Gibt 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.

$\text{normalLine}\left(x^2,x,1\right)$	$\frac{3}{2}-\frac{x}{2}$
$\text{normalLine}\left((x-3)^2-4,x,3\right)$	$x=3$
$\text{normalLine}\left(\begin{matrix} \frac{1}{3} \\ x \end{matrix},x=0\right)$	0
$\text{normalLine}\left(\sqrt{\lvert x\rvert},x=0\right)$	undef

normCdf() (Normalverteilungswahrscheinlichkeit)

Katalog >

normCdf(*untereGrenze***,***obereGrenze***[,**μ [,σ]]**)⇒***Zahl*, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

normCdf() (Normalverteilungswahrscheinlichkeit)

Katalog >

Berechnet die Normalverteilungswahrscheinlichkeit zwischen *untereGrenze* und *obereGrenze* für die angegebenen μ (Standard = 0) und σ (Standard = 1).

Für P(X ≤ *obereGrenze*) setzen Sie *untereGrenze* = −∞.

normPdf() (Wahrscheinlichkeitsdichte)

Katalog >

normPdf(*XWert*[,μ[,σ]]**)**⇒*Zahl*, wenn *XWert* eine Zahl ist, *Liste*, wenn *XWert* eine Liste ist

Berechnet die Wahrscheinlichkeitsdichtefunktion für die Normalverteilung an einem bestimmten *XWert* für die vorgegebenen μ und σ.

not (nicht)

Katalog >

not
BoolescherAusdr1⇒*BoolescherAusdruck*

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

not *Ganzzahl1*⇒*Ganzzahl*

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).

not(2≥3)	true
not(*x*<2)	*x*≥2
not not *innocent*	*innocent*

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O.

not 0h7AC36	0hFFFFFFFFFFF853C9

Im Bin-Modus:

0b100101▸Base10	37
not 0b100101	0b11111111111111111111111111111111▸
not 0b100101▸Base10	−38

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

not (nicht) — Katalog >

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: 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 >

nPr(*Ausdr1*, *Ausdr2*)⇒*Ausdruck*

Für ganzzahlige *Ausdr1* und *Ausdr2* mit *Ausdr1* ≥ *Ausdr2* ≥ 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.

$\mathrm{nPr}(z,3) \quad z\cdot(z-2)\cdot(z-1)$

$Ans\,|\,z=5 \quad 60$

$\mathrm{nPr}(z,^{-}3) \quad \frac{1}{(z+1)\cdot(z+2)\cdot(z+3)}$

$\mathrm{nPr}(z,c) \quad \frac{z!}{(z-c)!}$

$Ans\cdot\mathrm{nPr}(z-c,^{-}c) \quad 1$

nPr(*Ausdr*, **0)⇒1**

nPr(*Ausdr*, *negGanzzahl***)⇒ 1/((***Ausdr***+1) · (***Ausdr***+2)... (***Ausdr*−*negGanzzahl***))**

nPr(*Ausdr*, *posGanzzahl***) ⇒** *Ausdr* **· (***Ausdr*−**1)... (***Ausdr*−*posGanzzahl***+1)**

nPr(*Ausdr*, *keineGanzzahl***) ⇒***Ausdr***! / (***Ausdr*−*keineGanzzahl***)!**

nPr(*Liste1*, *Liste2***)⇒***Liste*

Gibt eine Liste der Permutationen auf der Basis der entsprechenden Elementpaare der beiden Listen zurück. Die Argumente müssen Listen gleicher Größe sein.

$\mathrm{nPr}(\{5,4,3\},\{2,4,2\}) \quad \{20,24,6\}$

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

Gibt eine Matrix der Permutationen auf der Basis der entsprechenden Elementpaare der beiden Matrizen zurück. Die Argumente müssen Matrizen gleicher Größe sein.

$\mathrm{nPr}\left(\begin{bmatrix}6 & 5\\ 4 & 3\end{bmatrix},\begin{bmatrix}2 & 2\\ 2 & 2\end{bmatrix}\right) \quad \begin{bmatrix}30 & 20\\ 12 & 6\end{bmatrix}$

npv() Katalog >

npv(*Zinssatz***,***CFO***,***CFListe*[,*CFFreq*]**)**

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.

CF0 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 Ganzzahlen < 10.000 sein.

$list1:=\{6000,-8000,2000,-3000\}$	$\{6000,-8000,2000,-3000\}$
$list2:=\{2,2,2,1\}$	$\{2,2,2,1\}$
$\text{npv}(10,5000,list1,list2)$	4769.91

nSolve() (Numerische Lösung) Katalog >

nSolve(*Gleichung***,***Var*
[=*Schätzwert***])**⇒*Zahl oder Fehler_String*

nSolve(*Gleichung***,***Var*
[=*Schätzwert***],***UntereGrenze***)** ⇒*Zahl oder Fehler_String*

nSolve(*Gleichung***,***Var*
[=
*Schätzwert***],***UntereGrenze***,***ObereGrenze***)**
⇒*Zahl oder Fehler_String*

nSolve(*Gleichung***,***Var***[=***Schätzwert***]) |**
UntereGrenze≤*Var*≤*ObereGrenze* ⇒*Zahl oder Fehler_String*

Ermittelt iterativ eine reelle numerische Näherungslösung von *Gleichung* für deren eine Variable. Geben Sie die Variable an als:

Variable

$\text{nSolve}(x^2+5\cdot x-25=9,x)$	3.84429
$\text{nSolve}(x^2=4,x=-1)$	-2.
$\text{nSolve}(x^2=4,x=1)$	2.

Hinweis: Existieren mehrere Lösungen, können Sie mit Hilfe einer Schätzung eine bestimmte Lösung suchen.

– 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().**

$\text{nSolve}(x^2+5\cdot x-25=9, x) \mid x<0$ $\quad -8.84429$

$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26, r\right) \mid r>0 \text{ and } r<0.25$ $\quad 0.006886$

$\text{nSolve}(x^2=-1, x)$ $\quad$ "No solution found"

O

OneVar [**1,**]*X*[**,**[*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 193.)

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 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. 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 259).

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
stat.MinX	Minimum der x-Werte
stat.Q_1X	1. Quartil von x
stat.MedianX	Median von x
stat.Q_3X	3. Quartil von x
stat.MaxX	Maximum der x-Werte
stat.SSX	Summe der Quadrate der Abweichungen der x-Werte vom Mittelwert

BoolescherAusd1 **or** *BoolescherAusdr2* ergibt *Boolescher Ausdruck*

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.

Hinweis: Siehe **xor**.

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

$x \geq 3$ or $x \geq 4$ → $x \geq 3$

Define $g(x)$=Func → *Done*
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*

Ganzzahl1 **or** *Ganzzahl2* ⇒ *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 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).

Im Hex-Modus:

0h7AC36 or 0h3D5F → 0h7BD7F

Wichtig: Null, nicht Buchstabe O.

Im Bin-Modus:

0b100101 or 0b100 → 0b100101

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.

or (oder) Katalog >

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) Katalog >

ord(*String***)⇒***Ganzzahl*

ord(*Liste1***)⇒***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

P▸Rx() (Kartesische x-Koordinate) Katalog >

P▸Rx(*rAusdr***, θ***Ausdr***)⇒***Ausdruck***

P▸Rx(*rListe***, θ***Liste***)⇒***Liste***

P▸Rx(*rMatrix***, θ***Matrix***)⇒***Matrix***

Gibt die äquivalente x-Koordinate des Paars (r, θ) zurück.

Hinweis: Das θ-Argument wird gemäß deraktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Ist das Argument ein Ausdruck, können Sie °, 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.

Im Bogenmaß-Modus:

P▸Rx(r,θ)	cos(θ)·r
P▸Rx(4,60°)	2
P▸Rx({-3,10,1.3},{π/3,-π/4,0})	{-3/2,5·√2,1.3}

P▶Ry() (Kartesische y-Koordinate) Katalog >

P▶Ry(*rAusdr*, θ*Ausdr*)⇒*Ausdruck*

P▶Ry(*rListe*, θ*Liste*)⇒*Liste*

P▶Ry(*rMatrix*, θ*Matrix*)⇒*Matrix*

Gibt die äquivalente y-Koordinate des Paars (r, θ) zurück.

Hinweis: Das θ-Argument wird gemäß deraktuellen Winkelmoduseinstellung als Grad, Neugrad oder Bogenmaß interpretiert. Ist das Argument ein Ausdruck, können Sie °, 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:

$\mathrm{P}\blacktriangleright\mathrm{Ry}(r,\theta) \qquad \sin(\theta)\cdot r$

$\mathrm{P}\blacktriangleright\mathrm{Ry}(4,60°) \qquad 2\cdot\sqrt{3}$

$\mathrm{P}\blacktriangleright\mathrm{Ry}\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right) \qquad \left\{\frac{-3\cdot\sqrt{3}}{2},-5\cdot\sqrt{2},0.\right\}$

PassErr (ÜbgebFeh) Katalog >

PassErr

Übergibt einen Fehler an die nächste Stufe.

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 **LöFehler**, Seite 31, und **Versuche**, Seite 209.

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

Ein Beispiel zu **PassErr** finden Sie im Beispiel 2 unter Befehl **Versuche (Try)**, Seite 209.

piecewise() (Stückweise) Katalog >

piecewise(*Ausdr1* **[,** *Bedingung1* **[,** *Ausdr2* **[,** *Bedingung2* **[, ...]]]])**

Gibt Definitionen für eine stückweise definierte Funktion in Form einer Liste zurück. Sie können auch mit Hilfe einer Vorlage stückweise Definitionen erstellen.

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

Define $p(x)=\begin{cases} x, & x>0 \\ \text{undef}, & x\le 0 \end{cases}$ *Done*

$p(1)$ 1

$p(-1)$ undef

poissCdf() Katalog >

poissCdf
(λ**,***untereGrenze***,***obereGrenze***)**⇒*Zahl*, wenn *untereGrenze* und *obereGrenze* Zahlen sind, *Liste*, wenn *untereGrenze* und *obereGrenze* Listen sind

poissCdf(λ**,***obereGrenze***)**(für P(0≤X ≤*obereGrenze*)⇒*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 ≤ *obereGrenze*) setzen Sie *untereGrenze* = 0

poissPdf() Katalog >

poissPdf(λ**,***XWert***)**⇒*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.

$[1 \quad 3.]$ ▸Polar $[3.16228 \quad \angle 1.24905]$

$[x \quad y]$ ▸Polar $\left[\sqrt{x^2+y^2} \quad \angle \frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right)\right]$

▸Polar

Katalog >

Zeigt *Vektor* in Polarform [r∠θ] 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 160.

komplexerWert **▸Polar**

Zeigt *komplexerVektor* in Polarform an.

- Der Grad-Modus für Winkel gibt (r∠θ) zurück.
- Der Bogenmaß-Modus für Winkel gibt re^iθ zurück.

komplexerWert kann jede komplexe Form haben. Eine re^iθ-Eingabe verursacht jedoch im Winkelmodus Grad einen Fehler.

Hinweis: Für eine Eingabe in Polarform müssen Klammern (r∠θ) verwendet werden.

Im Bogenmaß-Modus:

$(3+4\cdot i)\blacktriangleright\text{Polar} \qquad e^{i\cdot\left(\frac{\pi}{2}-\tan^{-1}\left(\frac{3}{4}\right)\right)}\cdot 5$

$\left(\left(4\angle\frac{\pi}{3}\right)\right)\blacktriangleright\text{Polar} \qquad e^{\frac{i\cdot\pi}{3}}\cdot 4$

Im Neugrad-Modus:

$(4\cdot i)\blacktriangleright\text{Polar} \qquad (4\angle 100)$

Im Grad-Modus:

$(3+4\cdot i)\blacktriangleright\text{Polar} \qquad \left(5\angle 90-\tan^{-1}\left(\frac{3}{4}\right)\right)$

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.

$\text{polyCoeffs}\left(4\cdot x^2-3\cdot x+2,x\right) \qquad \{4,-3,2\}$

$\text{polyCoeffs}\left((x-1)^2\cdot(x+2)^3\right) \qquad \{1,4,1,-10,-4,8\}$

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() Katalog >

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 Polynomausdruck 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.

polyEval({a,b,c},x)	$a \cdot x^2 + b \cdot x + c$
polyEval({1,2,3,4},2)	26
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.

polyGcd(100,30)	10
polyGcd(x^2-1,$x-1$)	$x-1$
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 derselben einzelnen Variablen sind.

polyQuotient($x-1$,$x-3$)	1
polyQuotient($x-1$,x^2-1)	0
polyQuotient(x^2-1,$x-1$)	$x+1$
polyQuotient($x^3-6 \cdot x^2+11 \cdot x-6$,$x^2-6 \cdot x+8$)	x

polyQuotient((x−y)·(y−z),x+y+z,x)	$y-z$
polyQuotient((x−y)·(y−z),x+y+z,y)	$2 \cdot x-y+2 \cdot z$

polyQuotient((x−y)·(y−z),x+y+z,z)	$-(x-y)$

polyRemainder() Katalog >

polyRemainder(*Poly1*,*Poly2* [,*Var*]**)**⇒*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.

$\text{polyRemainder}(x-1,x-3)$ → 2

$\text{polyRemainder}(x-1,x^2-1)$ → $x-1$

$\text{polyRemainder}(x^2-1,x-1)$ → 0

$\text{polyRemainder}((x-y)\cdot(y-z),x+y+z,x)$ → $-(y-z)\cdot(2\cdot y+z)$

$\text{polyRemainder}((x-y)\cdot(y-z),x+y+z,y)$ → $-2\cdot x^2-5\cdot x\cdot z-2\cdot z^2$

$\text{polyRemainder}((x-y)\cdot(y-z),x+y+z,z)$ → $(x-y)\cdot(x+2\cdot y)$

polyRoots() Katalog >

polyRoots(*Poly*,*Var***)** ⇒*Liste*

polyRoots(*KoeffListe***)** ⇒*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 43.

$\text{polyRoots}(y^3+1,y)$ → $\{-1\}$

$\text{cPolyRoots}(y^3+1,y)$ → $\left\{-1,\frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i,\frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i\right\}$

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

$\text{polyRoots}(\{1,2,1\})$ → $\{-1,-1\}$

PowerReg Katalog >

PowerReg *X*,*Y* [, *Häuf*] [, *Kategorie*, *Mit*]]

Berechnet die Potenzregressiony = (a · (x)^b)auf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

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 259).

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 *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*

Prgm

Katalog >

Prgm
Block
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.

GCD berechnen und Zwischenergebnisse anzeigen.

```
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 Π(), Seite 246.

Product (PI) (Produkt)

Siehe Π(), Seite 246.

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.

```
product({1,2,3,4})                      24
product({2,x,y})                        2·x·y
product({4,5,8,9},2,3)                  40

product([1 2 3; 4 5 6; 7 8 9])          [28 80 162]
product([1 2 3; 4 5 6; 7 8 9],1,2)      [4 10 18]
```

product() (Produkt) Katalog >

Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 259).

propFrac() (Echter Bruch) Katalog >

propFrac(*Ausdr1*[**,** *Var*]**)**⇒*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.

$\text{propFrac}\left(\frac{4}{3}\right)$ $1+\frac{1}{3}$

$\text{propFrac}\left(\frac{-4}{3}\right)$ $-1-\frac{1}{3}$

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.

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

$\text{propFrac}(Ans)$ $\frac{1}{x+1}+x+\frac{1}{y+1}+y$

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.

$\text{propFrac}\left(\frac{11}{7}\right)$ $1+\frac{4}{7}$

$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right)$ $8+\frac{37}{44}$

$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$ $-2-\frac{29}{44}$

QR

Katalog >

QR *Matrix*, *qMatrix*, *rMatrix*[, *Tol*]

Berechnet die Householdersche QR-Faktorisierung einer reellen oder komplexen Matrix. Die sich ergebenden Q- und R-Matrzen werden in den angegebenen *Matrix* gespeichert. Die Q-Matrix ist unitär. 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. Anderenfalls wird *Tol* ignoriert.

- 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 ·max(dim(*Matrix*)) ·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.

Die Fließkommazahl (9,) in m1 bewirkt, dass das Ergebnis in Fließkommaform berechnet wird.

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9.\end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9.\end{bmatrix}$$

$$\text{QR } m1,qm,rm \qquad \textit{Done}$$

$$qm \qquad \begin{bmatrix}0.123091 & 0.904534 & 0.408248\\ 0.492366 & 0.301511 & -0.816497\\ 0.86164 & -0.301511 & 0.408248\end{bmatrix}$$

$$rm \qquad \begin{bmatrix}8.12404 & 9.60114 & 11.0782\\ 0. & 0.904534 & 1.80907\\ 0. & 0. & 0.\end{bmatrix}$$

$$\begin{bmatrix}m & n\\ o & p\end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix}m & n\\ o & p\end{bmatrix}$$

$$\text{QR } m1,qm,rm \qquad \textit{Done}$$

$$qm \qquad \begin{bmatrix}\frac{m}{\sqrt{m^2+o^2}} & \frac{-\text{sign}(m\cdot p-n\cdot o)\cdot o}{\sqrt{m^2+o^2}}\\ \frac{o}{\sqrt{m^2+o^2}} & \frac{m\cdot\text{sign}(m\cdot p-n\cdot o)}{\sqrt{m^2+o^2}}\end{bmatrix}$$

$$rm \qquad \begin{bmatrix}\sqrt{m^2+o^2} & \frac{m\cdot n+o\cdot p}{\sqrt{m^2+o^2}}\\ 0 & \frac{|m\cdot p-n\cdot o|}{\sqrt{m^2+o^2}}\end{bmatrix}$$

QuadReg *X*,*Y* [, *Häuf*] [, *Kategorie*, *Mit*]]

Berechnet die quadratische polynomiale Regressiony = a ·x²+b ·x+cauf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

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 259).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: a ·x²+b ·x+c
stat.a, stat.b, stat.c	Regressionskoeffizienten
stat.R²	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
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*

QuartReg

Katalog >

QuartReg *X*,*Y* [, *Häuf*] [, *Kategorie*, *Mit*]]

Berechnet die polynomiale Regression vierter Ordnungy = a ·x4+b ·x3+c · x2+d ·x+eauf Listen *X* und *Y* mit der Häufigkeit *Häuf*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

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 259).

Ausgabevariable	**Beschreibung**
stat.RegEqn	Regressionsgleichung: a ·x4+b ·x3+c · x2+d ·x+e
stat.a, stat.b, stat.c, stat.d, stat.e	Regressionskoeffizienten

Ausgabevariable	Beschreibung
stat.R^2	Bestimmungskoeffizient
stat.Resid	Residuen von der Regression
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*

R

R▸Pθ() (Polarkoordinate)

Katalog >

R▸Pθ (*xAusdr***,** *yAusdr***)⇒***Ausdruck*

R▸Pθ (*xListe***,** *yListe***)⇒***Liste*

R▸Pθ (*xMatrix***,** *yMatrix***)⇒***Matrix*

Gibt die äquivalente θ-Koordinate des Paars (*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 Grad-Modus:

$$R\blacktriangleright P\theta(x,y) \qquad 90\cdot\text{sign}(y)-\tan^{-1}\left(\frac{x}{y}\right)$$

Im Neugrad-Modus:

$$R\blacktriangleright P\theta(x,y) \qquad 100\cdot\text{sign}(y)-\tan^{-1}\left(\frac{x}{y}\right)$$

Im Bogenmaß-Modus:

$$R\blacktriangleright P\theta(3,2) \qquad \tan^{-1}\left(\frac{2}{3}\right)$$

$$R\blacktriangleright P\theta\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right) \qquad \begin{bmatrix}0 & \tan^{-1}\left(\frac{16}{\pi}\right)+\frac{\pi}{2} & 0.643501\end{bmatrix}$$

R▸Pr() (Polarkoordinate)

Katalog >

R▸Pr (*xAusdr***,** *yAusdr***)⇒***Ausdruck*

R▸Pr (*xListe***,** *yListe***)⇒***Liste*

R▸Pr (*xMatrix***,** *yMatrix***)⇒***Matrix*

Im Bogenmaß-Modus:

R▸Pr() (Polarkoordinate)

Katalog >

Gibt die äquivalente r-Koordinate des Paars (*x*,*y*) zurück.

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

$\mathrm{R}\blacktriangleright\mathrm{Pr}(3,2) \quad \sqrt{13}$

$\mathrm{R}\blacktriangleright\mathrm{Pr}(x,y) \quad \sqrt{x^2+y^2}$

$\mathrm{R}\blacktriangleright\mathrm{Pr}\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right)$

$\begin{bmatrix}3 & \frac{\sqrt{\pi^2+256}}{4} & 2.5\end{bmatrix}$

▸Rad (Bogenmaß)

Katalog >

Ausdr1▸**Rad**⇒*Ausdruck*

Wandelt das Argument ins Winkelmaß Bogenmaß um.

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

Im Grad-Modus:

$(1.5)\blacktriangleright\mathrm{Rad} \quad (0.02618)^r$

Im Neugrad-Modus:

$(1.5)\blacktriangleright\mathrm{Rad} \quad (0.023562)^r$

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(*n***,** *p***,** *#Trials***)**⇒*Liste*

randBin(*n***,** *p***)** gibt eine reelle Zufallszahl aus einer angegebenen Binomialverteilung zurück.

randBin(80,0.5)	42
randBin(80,0.5,3)	{41,32,39}

randBin() (Zufallszahl aus Binomialverteilung)

Katalog >

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 (*lowBound*,*upBound*,*#Trials***)**⇒*Liste*

randInt(*lowBound*,*upBound***)** gibt eine ganzzahlige Zufallszahl innerhalb der durch *UntereGrenze (lowBound)* und *ObereGrenze (upBound)* festgelegten Grenzen zurück.

randInt(*lowBound*,*upBound*,*#Trials***)** gibt eine Liste mit *#Trials* ganzzahligen Zufallszahlen innerhalb des festgelegten Bereichs zurück.

randInt(3,10)	5
randInt(3,10,4)	{9,7,5,8}

randMat() (Zufallsmatrix)

Katalog >

randMat(*AnzZeil*, *AnzSpalt***)**⇒*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**)**⇒*Ausdruck*

randNorm(μ, σ, *#Versuche***)**⇒*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 [μ−3 ·σ, μ+3 ·σ].

RandSeed 1147	*Done*
randNorm(0,1)	0.492541
randNorm(3,4.5)	−3.54356

randNorm() (Zufallsnorm) Katalog >

randNorm(μ**,** σ**,** *#Versuche***)** gibt eine Liste mit *#Versuche* Dezimalzahlen aus der angegebenen Normalverteilung zurück.

randPoly() (Zufallspolynom) Katalog >

randPoly(*Var***,** *Ordnung***)⇒***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*])**⇒***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 *list3*={1,2,3,4,5}	*Done*
Define *list4*=randSamp(*list3*,6)	*Done*
list4	{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* ≠ 0 werden zwei Basen erzeugt, die in den Systemvariablen seed1 und seed2 gespeichert werden.

RandSeed 1147	*Done*
rand()	0.158206

real() (Reell) Katalog >

real(*Ausdr1***)⇒***Ausdruck*

Gibt den Realteil des Arguments zurück.

Hinweis: Alle undefinierten Variablen werden als reelle Variablen behandelt. Siehe auch **imag()**, Seite 98.

real(2+3·*i*)	2
real(*z*)	*z*
real(*x*+*i*·*y*)	*x*

real() (Reell) Katalog >

real(*Liste1***)⇒***Liste*

Gibt für jedes Element den Realteil zurück.

$\text{real}(\{a+i\cdot b,3,i\})$ $\{a,3,0\}$

real(*Matrix1***)⇒***Matrix*

Gibt für jedes Element den Realteil zurück.

$\text{real}\begin{bmatrix} a+i\cdot b & 3 \\ c & i \end{bmatrix}$ $\begin{bmatrix} a & 3 \\ c & 0 \end{bmatrix}$

▶Rect (Kartesisch) Katalog >

Vektor **▶Rect**

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

$\begin{bmatrix} 3 & \angle\frac{\pi}{4} & \angle\frac{\pi}{6} \end{bmatrix}\blacktriangleright\text{Rect}$

$\begin{bmatrix} \frac{3\cdot\sqrt{2}}{4} & \frac{3\cdot\sqrt{2}}{4} & \frac{3\cdot\sqrt{3}}{2} \end{bmatrix}$

Zeigt *Vektor* in der kartesischen Form [x, y, z] an. Der Vektor muss die Dimension 2 oder 3 besitzen und kann eine Zeile oder eine Spalte sein.

$[a \quad \angle b \quad \angle c]$

$[a\cdot\cos(b)\cdot\sin(c) \quad a\cdot\sin(b)\cdot\sin(c) \quad a\cdot\cos(c)]$

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 145.

komplexerWert **▶Rect**

Im Bogenmaß-Modus:

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.

$\left(4\cdot e^{\frac{\pi}{3}}\right)\blacktriangleright\text{Rect}$ $4\cdot e^{\frac{\pi}{3}}$

$\left(\left(4\angle\frac{\pi}{3}\right)\right)\blacktriangleright\text{Rect}$ $2+2\cdot\sqrt{3}\cdot i$

Hinweis: Für eine Eingabe in Polarform müssen Klammern ($r\angle\theta$) verwendet werden.

Im Neugrad-Modus:

$((1\angle 100))\blacktriangleright\text{Rect}$ i

Im Grad-Modus:

$((4\angle 60))\blacktriangleright\text{Rect}$ $2+2\cdot\sqrt{3}\cdot i$

Hinweis: Wählen Sie zur Eingabe von ∠ das Symbol aus der Sonderzeichenpalette des Katalogs aus.

ref(*Matrix1*[**,** *Tol*]**)⇒***Matrix*

Gibt die Diagonalform von *Matrix1* 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. Anderenfalls wird *Tol* ignoriert.

- 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 ·max(dim(*Matrix1*)) ·rowNorm (*Matrix1*)

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:

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Die Warnung erscheint, weil das verallgemeinerte Element $1/a$ für $a=0$ nicht zulässig wäre.

$$\text{ref}\left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \to m1 \qquad \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\text{ref}(m1) \qquad \begin{bmatrix} 1 & \frac{d}{c} \\ 0 & 1 \end{bmatrix}$$

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.

$$\text{ref}\left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) | a=0 \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Hinweis: Siehe auch **rref()**, Seite 170.

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 127.

$\text{remain}(7,0)$	7
$\text{remain}(7,3)$	1
$\text{remain}(-7,3)$	-1
$\text{remain}(7,-3)$	1
$\text{remain}(-7,-3)$	-1
$\text{remain}(\{12,-14,16\},\{9,7,-5\})$	$\{3,0,1\}$

$$\text{remain}\left(\begin{bmatrix} 9 & -7 \\ 6 & 4 \end{bmatrix},\begin{bmatrix} 4 & 3 \\ 4 & -3 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$

Request

Katalog >

Request*EingabeString*, *var*[, *FlagAnz* [, *statusVar*]]

Request*EingabeString*, *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.

Definieren Sie ein Programm:

```
Define request_demo()=Prgm
  Request "Radius: ",r
  Disp "Fläche = ",pi*r²
EndPrgm
```

Starten Sie das Programm und geben Sie eine Antwort ein:

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** 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.

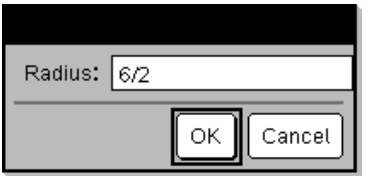
Das optionale Argument *statusVar* ermöglicht es dem Programm, zu bestimmen, wie der Benutzer das Dialogfeld verlassen hat. Beachten Sie bitte, dass *statusVar* das Argument *FlagAnz* erfordert.

- Wenn der Benutzer auf **OK** geklickt oder die **Eingabetaste** bzw. **Strg+Eingabetaste** gedrückt hat, wird die Variable *statusVar* auf den Wert **1** gesetzt.
- Anderenfalls 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*

```
request_demo()
```



Ergebnis nach Auswahl von **OK**:

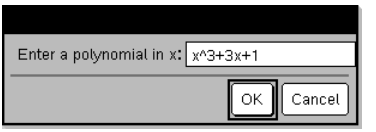
```
Radius: 6/2
Fläche = 28.2743
```

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}
```

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 [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 **RequestStr, Seite 164.**

RequestStr

Katalog >

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.

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()
```

RequestStr

Katalog >

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 162.

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*

Gibt *Anz* Elemente zurück, die rechts in *Liste1* enthalten sind.

Wenn Sie *Anz* weglassen, wird die gesamte *Liste1* zurückgegeben.

```
right({1,3,-2,4},3)             {3,-2,4}
```

right() (Rechts)

Katalog >

right(*Quellstring***[, *Anz*])⇒*String***

Gibt *Anz* Zeichen zurück, die rechts in der Zeichenkette *Quellstring* enthalten sind.

Wenn Sie *Anz* weglassen, wird der gesamte *Quellstring* zurückgegeben.

right("Hello",2) "lo"

right(*Vergleich***)⇒*Ausdruck***

Gibt die rechte Seite einer Gleichung oder Ungleichung zurück.

right(x<3) 3

rk23 ()

Katalog >

rk23(*Ausdr*, *Var*, *abhVar*, {*Var0*, *VarMax*}, *abhVar0*, *VarSchritt* [, *diftol*])⇒*Matrix*

rk23(*AusdrSystem*, *Var*, *ListeAbhVar*, {*Var0*, *VarMax*}, *ListeAbhVar0*, *VarSchritt* [, *diftol*]) ⇒*Matrix*

rk23(*AusdrListe*, *Var*, *ListeAbhVar*, {*Var0*, *VarMax*}, *ListeAbhVar0*, *VarSchritt* [, *diftol*]) ⇒*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*).

Differentialgleichung:

y'=0.001*y*(100-y) und y(0)=10

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)

$$y=\frac{100.\cdot(1.10517)^t}{(1.10517)^t+9.}$$

rk23 ()

Katalog >

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.

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).

$$\text{seqGen}\left(\frac{100.\cdot(1.10517)^{t}}{(1.10517)^{t}+9.},t,y,\{0,100\}\right)$$

$$\{10.,10.9367,11.9494,13.0423,14.2189,15.48$$

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 $y2(0)=5$

$$\text{rk23}\left(\begin{cases} -y1+0.1\cdot y1\cdot y2 \\ 3\cdot y2-y1\cdot y2 \end{cases},t,\{y1,y2\},\{0,5\},\{2,5\},1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 2. & 1.94103 & 4.78694 & 3.25253 & 1.82848 \\ 5. & 16.8311 & 12.3133 & 3.51112 & 6.27245 \end{bmatrix}$$

root() (Wurzel)

Katalog >

root(*Ausdr***)**⇒ *root*

root(*Ausdr1***, *Ausdr2***)**⇒ *Wurzel*

root(*Ausdr***) gibt die Quadratwurzel von** *Ausdr* **zurück.**

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.

$$\sqrt[3]{8} \qquad 2$$

$$\sqrt[3]{3} \qquad 3^{\frac{1}{3}}$$

$$\sqrt[3]{3.} \qquad 1.44225$$

rotate(*Ganzzahl1* [**,***#Rotationen*]**)⇒***Ganzzahl*

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**, 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.

0b00000000000000111101011000011010 1

Bit ganz rechts rotiert nach ganz links.

Es ergibt sich:

0b10000000000000011110101100001101 0

Das Ergebnis wird gemäß dem jeweiligen Basis-Modus angezeigt.

Im Bin-Modus:

rotate(0b11111111111111111111111111111111)	0b10000000000000000000000000000001
rotate(256,1)	0b1000000000

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

Im Hex-Modus:

rotate(0h78E)	0h3C7
rotate(0h78E,⁻2)	0h80000000000001E3
rotate(0h78E,2)	0h1E38

Wichtig: Geben Sie eine Dual- oder Hexadezimalzahl stets mit dem Präfix 0b bzw. 0h ein (Null, nicht der Buchstabe O).

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).

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(*String1*[**,***#Rotationen*]**)⇒***String*

Gibt eine um *#Rotationen* Zeichen nach rechts oder links rotierte Kopie von *String1* zurück. Verändert *String1* nicht.

rotate("abcd")	"dabc"
rotate("abcd",⁻2)	"cdab"
rotate("abcd",1)	"bcda"

rotate() (Rotieren) Katalog >

Ist *#Rotationen* positiv, erfolgt eine Rotation nach links. ist *#Rotationen* negativ, erfolgt eine Rotation nach rechts. Vorgabe ist ⁻1 (ein Zeichen nach rechts rotieren).

round() (Runden) Katalog >

round(*Ausdr1***[, *Stellen*])⇒*Ausdruck***

Gibt das Argument gerundet auf die angegebene Anzahl von Stellen nach dem Dezimaltrennzeichen zurück.

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(1.234567,3) 1.235

round(*Liste1***[, *Stellen*])⇒*Liste***

Gibt eine Liste von Elementen zurück, die auf die angegebene Stellenzahl gerundet wurden.

round({π,√2,ln(2)},4) {3.1416,1.4142,0.6931}

round(*Matrix1***[, *Stellen*])⇒*Matrix***

Gibt eine Matrix von Elementen zurück, die auf die angegebene Stellenzahl gerundet wurden.

round([ln(5) ln(3); π e^1],1) [1.6 1.1; 3.1 2.7]

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.

rowAdd([3 4; -3 -2],1,2) [3 4; 0 2]

rowAdd([a b; c d],1,2) [a b; a+c b+d]

rowDim() (Zeilendimension)

Katalog >

rowDim(*Matrix***)⇒***Ausdruck*

Gibt die Anzahl der Zeilen von *Matrix* zurück.

Hinweis: Siehe auch **colDim()**, Seite 32.

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

$$\text{rowDim}(m1) \qquad 3$$

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 **colNorm()**, Seite 32.

$$\text{rowNorm}\left(\begin{bmatrix}-5 & 6 & -7\\3 & 4 & 9\\9 & -9 & -7\end{bmatrix}\right) \qquad 25$$

rowSwap() (Zeilentausch)

Katalog >

rowSwap(*Matrix1***,** *rIndex1***,** *rIndex2***)⇒***Matrix*

Gibt *Matrix1* zurück, in der die Zeilen *rIndex1* und *rIndex2* vertauscht sind.

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

$$\text{rowSwap}(mat,1,3) \qquad \begin{bmatrix}5 & 6\\3 & 4\\1 & 2\end{bmatrix}$$

rref() (Reduzierte Diagonalform)

Katalog >

rref(*Matrix1***[,** *Tol***]⇒***Matrix*

Gibt die reduzierte Diagonalform von *Matrix1* zurück.

$$\text{rref}\left(\begin{bmatrix}-2 & -2 & 0 & -6\\1 & -1 & 9 & -9\\-5 & 2 & 4 & -4\end{bmatrix}\right) \qquad \begin{bmatrix}1 & 0 & 0 & \frac{66}{71}\\0 & 1 & 0 & \frac{147}{71}\\0 & 0 & 1 & \frac{-62}{71}\end{bmatrix}$$

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. Anderenfalls wird *Tol* ignoriert.

$$\text{rref}\left(\begin{bmatrix}a & b\\c & d\end{bmatrix}\right) \qquad \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

- 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 ·max(dim(*Matrix1*)) ·rowNorm (*Matrix1*)

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

S

sec() (Sekans)

trig **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 °, g oder r benutzen, um den Winkelmodus vorübergehend aufzuheben.

Im Grad-Modus:

$\sec(45)$ $\sqrt{2}$

$\sec(\{1,2.3,4\})$ $\left\{\frac{1}{\cos(1)}, 1.00081, \frac{1}{\cos(4)}\right\}$

sec⁻¹() (Arkussekans)

trig **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.

Im Grad-Modus:

$\sec^{-1}(1)$ 0

Im Neugrad-Modus:

$\sec^{-1}(\sqrt{2})$ 50

Im Bogenmaß-Modus:

sec⁻¹() (Arkussekans)

trig Taste

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

$\sec^{-1}(\{1,2,5\})$ $\quad \left\{0, \frac{\pi}{3}, \cos^{-1}\left(\frac{1}{5}\right)\right\}$

sech() (Sekans hyperbolicus)

Katalog >

sech(*Ausdr1***)** ⇒ *Ausdruck*

sech(*Liste1***)** ⇒ *Liste*

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

$\operatorname{sech}(3)$ $\quad \frac{1}{\cosh(3)}$

$\operatorname{sech}(\{1,2.3,4\})$ $\quad \left\{\frac{1}{\cosh(1)}, 0.198522, \frac{1}{\cosh(4)}\right\}$

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":

$\operatorname{sech}^{-1}(1)$ $\quad 0$

$\operatorname{sech}^{-1}(\{1,-2,2.1\})$ $\quad \left\{0, \frac{2\cdot\pi}{3}\cdot i, 8.\text{E}{-15}+1.07448\cdot i\right\}$

Send

Hub-Menü

Send *exprOrString1*[**,** *exprOrString2*] ...

Programmierbefehl: Sendet einen oder mehrere TI-Innovator™ Hub Befehle an den verbundenen Hub.

exprOrString muss ein gültiger TI-Innovator™ Hub Befehl sein. Normalerweise enthält *exprOrString* einen Befehl **"SET ..."** zum Steuern eines Geräts oder einen Befehl **"READ ..."** zum Anfordern von Daten.

Die Argumente werden hintereinander an den Hub gesendet.

Beispiel: Schalten Sie das blaue Element der integrierten RGB LED 0,5 Sekunden lang ein.

```
Send "SET COLOR.BLUE ON TIME .5"
                                Done
```

Beispiel: Fordern Sie den aktuellen Wert des integrierten Lichtpegelsensors des Hub an. Ein Befehl **Get** ruft den Wert ab und weist ihn der Variablen *lightval* zu.

Send "READ BRIGHTNESS"	*Done*
Get *lightval*	*Done*
lightval	0.347922

Send — Hub-Menü

Hinweis: Sie können den Befehl **Send** in einem benutzerdefinierten Programm, aber nicht in einer Funktion verwenden.

Hinweis: Siehe auch **Get** (Seite 89), **GetStr** (Seite 93) und **eval()** (Seite 71).

Beispiel: Senden Sie eine berechnete Frequenz an den integrierten Lautsprecher des Hub. Verwenden Sie die spezielle Variable *iostr.SendAns*, um den Hub-Befehl mit dem ausgewerteten Ausdruck anzuzeigen.

$n:=50$	50
$m:=4$	4
Send "SET SOUND eval(m · n)"	*Done*
iostr.SendAns	"SET SOUND 200"

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.

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$	1.54977

seqGen() — Katalog >

seqGen(*Ausdr*, *Var*, *abhVar*, {*Var0*, *VarMax*}[, *ListeAnfTerme* [, *VarSchritt* [, *ObergrWert*]]]**)** ⇒*Liste*

Generieren Sie die ersten 5 Terme der Folge $u(n) = u(n\text{-}1)^2/2$ mit $u(1)$=**2** und *VarSchritt*=**1**.

$\text{seqGen}\left(\frac{\left(u(n-1)\right)^2}{n},n,u,\{1,5\},\{2\}\right)$
$\left\{2,2,\frac{4}{3},\frac{4}{9},\frac{16}{405}\right\}$

seqGen() Katalog >

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*]]]**)** ⇒*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**.

Beispiel mit Var0=2:

$$\text{seqGen}\left(\frac{u(n-1)+1}{n},n,u,\{2,5\},\{3\}\right) \quad \left\{3,\frac{4}{3},\frac{7}{12},\frac{19}{60}\right\}$$

Beispiel, in dem der Anfangsterm symbolisch ist:

$$\text{seqGen}\left(u(n-1)+2,n,u,\{1,5\},\{a\}\right) \quad \{a,a+2,a+4,a+6,a+8\}$$

System zweiter Folgen:

$$\text{seqGen}\left(\left\{\frac{1}{n},\frac{u2(n-1)}{2}+u1(n-1)\right\},n,\{u1,u2\},\{1,5\},\begin{bmatrix}\_\\2\end{bmatrix}\right) \quad \begin{bmatrix}1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5}\\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24}\end{bmatrix}$$

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() Katalog >

seqn(*Ausdr*(*u*, *n* [, *ListeAnfTerme*[, *nMax* [, *ObergrWert*]]]**)**⇒*Liste*

Generiert eine Term-Liste für eine Folge *u*(*n*)=*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 *Ausdr*(*u*, *n*) und *ListeAnfTerme* aus und gibt die Ergebnisse als Liste zurück.

seqn(*Ausdr*(*n* [, *nMax* [, *ObergrWert*]]**)**⇒*Liste*

Generieren Sie die ersten 6 Terme der Folge *u*(*n*) = *u*(*n*-1)/2 mit *u*(1)=**2**.

$$\text{seqn}\left(\frac{u(n-1)}{n},\{2\},6\right) \quad \left\{2,1,\frac{1}{3},\frac{1}{12},\frac{1}{60},\frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2},6\right) \quad \left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16},\frac{1}{25},\frac{1}{36}\right\}$$

seqn()

Katalog >

Generiert eine Term-Liste für eine nichtrekursive Folge $u(n)=Ausdr(n)$ wie folgt: Erhöht n von 1 bis $nMax$ um 1, wertet $u(n)$ für die entsprechenden Werte von n mithilfe der Formel $Ausdr(n)$ 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**

series()

Katalog >

series(*Expr1*, *Var*, *Order* [, *Point*]**)⇒***Ausdruck*

series(*Expr1*, *Var*, *Order* [, *Point*]**) |** *Var***>***Point*⇒*Ausdruck*

series(*Expr1*, *Var*, *Order* [, *Point*]**) |** *Var***<***Point*⇒*Ausdruck*

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 dominiert werden.

Point ist vorgegeben als 0. *Point* kann ∞ oder −∞ sein; in diesen Fällen ist die Entwicklung durch Grad *Order* in 1/(*Var* − *Point*).

$$\text{series}\left(\frac{1-\cos(x-1)}{(x-1)^2},x,4,1\right) \qquad \frac{1}{2}-\frac{(x-1)^2}{24}+\frac{(x-1)^4}{720}$$

$$\text{series}\left(\frac{-1}{e^{z-}},z-,1\right) \qquad z-1$$

$$\text{series}\left(\left(1+\frac{1}{n}\right)^n,n,2,\infty\right) \qquad e-\frac{e}{2\cdot n}+\frac{11\cdot e}{24\cdot n^2}$$

$$\text{series}\left(\tan^{-1}\left(\frac{1}{x}\right),x,5\right)\left|x>0\right. \qquad \frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}$$

$$\text{series}\left(\int\frac{\sin(x)}{x}\,dx,x,6\right) \qquad x-\frac{x^3}{18}+\frac{x^5}{600}$$

$$\text{series}\left(\int_0^x \sin(x\cdot\sin(t))\,dt,x,7\right) \qquad \frac{x^3}{2}-\frac{x^5}{24}-\frac{29\cdot x^7}{720}$$

$$\text{series}\left(\left(1+e^x\right)^2,x,2,1\right)$$
$$(e+1)^2+2\cdot e\cdot(e+1)\cdot(x-1)+e\cdot(2\cdot e+1)\cdot(x-1)^2$$

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 = ∞ oder −∞.

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

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 64.

setMode

Katalog >

setMode(*ModusNameGanzzahl***,** *GanzzahlFestlegen***)** ⇒*Ganzzahl*

setMode(*Liste***)** ⇒*Liste mit ganzen Zahlen*

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.

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.

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 92.

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.

```
Define prog1()=Prgm                    Done
  Disp approx(π)
  setMode(1,16)
  Disp approx(π)
  EndPrgm

prog1()
                                    3.14159
                                       3.14
                                       Done
```

setMode

Katalog >

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)

Katalog >

shift(*Ganzzahl1* [**,***#Verschiebungen*]**)⇒***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.

Im Bin-Modus:

shift(0b1111010110000110101)	0b111101011000011010
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).

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.

0b00000000000000111101011000011010

Setzt 0 ein, wenn Bit ganz links 0 ist, und 1, wenn Bit ganz links 1 ist.

Es ergibt sich:

0b00000000000000011110101100011010

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 (ein Element nach rechts verschieben).

Dadurch eingeführte neue Elemente am Anfang bzw. am Ende von *Liste* werden auf “undef” gesetzt.

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(*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 "

shift() (Verschieben) Katalog >

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(*Liste1***)⇒***Liste*

sign(*Matrix1***)⇒***Matrix*

Gibt für reelle und komplexe *Ausdr1* *Ausdr1*/**abs(***Ausdr1***)** zurück, wenn *Ausdr1*≠ 0.

Gibt 1 zurück, wenn *Ausdr1* positiv ist.

Gibt ⁻1 zurück, wenn *Ausdr1* negativ 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.

$\text{sign}(-3.2)$ → $-1.$

$\text{sign}(\{2,3,4,-5\})$ → $\{1,1,1,-1\}$

$\text{sign}(1+|x|)$ → 1

Bei Komplex-Formatmodus Reell:

$\text{sign}\begin{bmatrix}-3 & 0 & 3\end{bmatrix}$ → $\begin{bmatrix}-1 & \pm 1 & 1\end{bmatrix}$

simult() (Gleichungssystem) Katalog >

simult(*KoeffMatrix***,** *KonstVektor***[,** *Tol***])⇒***Matrix*

Ergibt einen Spaltenvektor, der die Lösungen für ein lineares Gleichungssystem enthält.

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

KoeffMatrix muss eine quadratische Matrix sein, die die Koeffizienten der Gleichung enthält.

Auflösen nach x und y:

$x + 2y = 1$

$3x + 4y = -1$

$\text{simult}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}, \begin{bmatrix}1\\ -1\end{bmatrix}\right)$ → $\begin{bmatrix}-3\\ 2\end{bmatrix}$

Die Lösung ist x=⁻3 und y=2.

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. Anderenfalls 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 ·max(dim(*KoeffMatrix*)) ·rowNorm(*KoeffMatrix*)

Auflösen:

ax + by = 1

cx + dy = 2

$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow matx1$ $\quad$ $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$\text{simult}\left(matx1, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$ $\quad$ $\begin{bmatrix} \frac{-(2 \cdot b - d)}{a \cdot d - b \cdot c} \\ \frac{2 \cdot a - c}{a \cdot d - b \cdot c} \end{bmatrix}$

simult(*KoeffMatrix***,** *KonstMatrix***[,** *Tol***])⇒***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.

Auflösen:

x + 2y = 1

3x + 4y = ⁻1

x + 2y = 2

3x + 4y = ⁻3

$\text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & -3 \end{bmatrix}\right)$ $\quad$ $\begin{bmatrix} -3 & -7 \\ 2 & \frac{9}{2} \end{bmatrix}$

Für das erste System ist x=⁻3 und y=2. Für das zweite System ist x=⁻7 und y=9/2.

Ausdr **▸sin**

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

$(\cos(x))^2 \blacktriangleright \sin$ $\quad$ $1-(\sin(x))^2$

▸sin

Katalog > 📖

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

▸sin reduziert alle Potenzen von cos(...) modulo 1−sin(...)^2, so dass alle verbleibenden Potenzen von sin(...) Exponenten im Bereich (0, 2) haben. Deshalb enthält das Ergebnis dann und nur dann kein cos(...), wenn cos(...) 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)

trig Taste

sin(*Ausdr1***)⇒***Ausdruck*

sin(*Liste1***)⇒***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.

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:

$\sin\left(\frac{\pi}{4}r\right)$ → $\frac{\sqrt{2}}{2}$

$\sin(45)$ → $\frac{\sqrt{2}}{2}$

$\sin(\{0,60,90\})$ → $\left\{0,\frac{\sqrt{3}}{2},1\right\}$

Im Neugrad-Modus:

$\sin(50)$ → $\frac{\sqrt{2}}{2}$

Im Bogenmaß-Modus:

sin() (Sinus)

trig Taste

$\sin\left(\frac{\pi}{4}\right)$ $\frac{\sqrt{2}}{2}$

$\sin(45°)$ $\frac{\sqrt{2}}{2}$

sin(*Quadratmatrix1***)⇒***Quadratmatrix***

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

Quadratmatrix1 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

$\sin^{-1}$(*Ausdr1***)⇒***Ausdruck***

$\sin^{-1}$(*Liste1***)⇒***Liste***

$\sin^{-1}$(*Ausdr1***)** gibt den Winkel, dessen Sinus *Ausdr1* ist, als Ausdruck zurück.

$\sin^{-1}$(*Liste1***)** gibt in Form einer Liste für jedes Element aus *Liste1* den inversen 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.

Im Grad-Modus:

$\sin^{-1}(1)$ 90

Im Neugrad-Modus:

$\sin^{-1}(1)$ 100

Im Bogenmaß-Modus:

$\sin^{-1}(\{0,0.2,0.5\})$ $\{0,0.201358,0.523599\}$

$\sin^{-1}$(*Quadratmatrix1***)⇒***Quadratmatrix***

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

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\sin^{-1}\left(\begin{bmatrix}1 & 5\\ 4 & 2\end{bmatrix}\right)$$

$$\begin{bmatrix}-0.174533-0.12198\cdot i & 1.74533-2.35591\cdot i\\ 1.39626-1.88473\cdot i & 0.174533-0.593162\cdot i\end{bmatrix}$$

sin⁻¹() (Arkussinus)

trig Taste

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

sinh() (Sinus hyperbolicus)

Katalog >

sinh(*Ausdr1***)⇒***Ausdruck*

sinh(*Liste1***)⇒***Liste*

sinh (*Ausdr1***)** gibt den Sinus hyperbolicus des Arguments als Ausdruck **zurück.**

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

sinh(*Quadratmatrix1***)⇒***Quadratmatrix*

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$\sinh(1.2)$ → 1.50946

$\sinh(\{0,1.2,3.\})$ → $\{0,1.50946,10.0179\}$

Im Bogenmaß-Modus:

$$\sinh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

sinh⁻¹() (Arkussinus hyperbolicus)

Katalog >

sinh⁻¹(*Ausdr1***)⇒***Ausdruck*

sinh⁻¹(*Liste1***)⇒***Liste*

sinh⁻¹(*Ausdr1***)** gibt den inversen Sinus hyperbolicus des Arguments als Ausdruck zurück.

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

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

sinh⁻¹(*Quadratmatrix1***)⇒***Quadratmatrix*

$\sinh^{-1}(0)$ → 0

$\sinh^{-1}(\{0,2.1,3\})$ → $\{0,1.48748,\sinh^{-1}(3)\}$

Im Bogenmaß-Modus:

sinh⁻¹() (Arkussinus hyperbolicus) Katalog >

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$$\sinh^{-1}\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

$$\begin{bmatrix}0.041751 & 2.15557 & 1.1582\\ 1.46382 & 0.926568 & 0.112557\\ 2.75079 & -1.5283 & 0.57268\end{bmatrix}$$

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 193.)

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 259).

Ausgabevariable	Beschreibung
stat.RegEqn	Regressionsgleichung: a ·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 *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*

solve() (Löse) Katalog >

solve(*Gleichung*, *Var***)⇒***Boolescher Ausdruck*

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.

$$\text{solve}\left(a\cdot x^2+b\cdot x+c=0,x\right)$$
$$x=\frac{\sqrt{b^2-4\cdot a\cdot c}-b}{2\cdot a} \text{ or } x=\frac{-\left(\sqrt{b^2-4\cdot a\cdot c}+b\right)}{2\cdot a}$$

Für manche Wertekombinationen undefinierter Variablen kann es sein, dass mögliche Lösungen nicht reell und endlich sind.

$Ans|a=1 \text{ and } b=1 \text{ and } c=1$

$$x=\frac{-1}{2}+\frac{\sqrt{3}}{2}\cdot i \text{ or } x=\frac{-1}{2}-\frac{\sqrt{3}}{2}\cdot i$$

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.

$\text{solve}((x-a)\cdot e^x=-x\cdot(x-a),x)$

$x=a \text{ or } x=-0.567143$

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.

$(x+1)\cdot\frac{x-1}{x-1}+x-3$ $\quad 2\cdot x-2$

Für Ungleichungen der Typen ≥, ≤, < oder > sind explizite Lösungen unwahrscheinlich, es sei denn, die Ungleichung ist linear und enthält nur *Var*.

$\text{solve}(5\cdot x-2\geq 2\cdot x,x)$ $\quad x\geq\frac{2}{3}$

Ist der Modus **Auto oder Näherung** auf Exakt eingestellt, werden nicht lösbare Teile als implizite Gleichung oder Ungleichung zurückgegeben.

$\text{exact}(\text{solve}((x-a)\cdot e^x=-x\cdot(x-a),x))$

$e^x+x=0 \text{ or } x=a$

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.

Im Bogenmaß-Modus:

$\text{solve}\left(\tan(x)=\frac{1}{x},x\right)|x>0 \text{ and } x<1$

$x=0.860334$

Wenn keine reellen Lösungen ermittelt werden können, wird “falsch” zurückgegeben. “wahr” wird zurückgegeben, wenn **solve()** feststellt, dass jeder endliche reelle Wert von *Var* die Gleichung bzw. Ungleichung erfüllt.

$\text{solve}(x=x+1,x)$ false

$\text{solve}(x=x,x)$ true

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.

$2 \cdot x - 1 \le 1$ and solve$(x^2 \ne 9, x)$ $\quad x \ne -3$ and $x \le 1$

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 Bogenmaß-Modus:

solve$(\sin(x)=0, x)$ $\quad x = n1 \cdot \pi$

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.

solve$\left(x^{\frac{1}{3}} = -1, x\right)$ $\quad x = -1$

solve$(\sqrt{x} = -2, x)$ $\quad$ false

solve$(-\sqrt{x} = -2, x)$ $\quad x = 4$

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.

solve$(y = x^2 - 2$ and $x + 2 \cdot y = -1, \{x, y\})$

$x = \frac{-3}{2}$ and $y = \frac{1}{4}$ or $x = 1$ 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

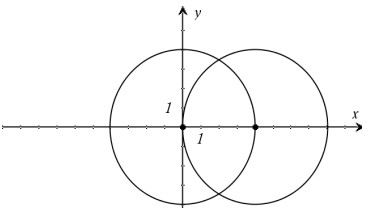
- oder -

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.

$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y\}\right)$$
$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ or } x=\frac{r}{2} \text{ and } y=\frac{-\sqrt{3}\cdot r}{2}$$

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.

$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y,z\}\right)$$
$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ and } z=c1 \text{ or } x=\frac{r}{2} \text{ and } y\blacktriangleright$$

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

Die Zylinder-Lösungen verdeutlichen, dass Lösungsfamilien "beliebige" Konstanten der Form *c*k, enthalten können, wobei k ein ganzzahliger Index im Bereich 1 bis 255 ist.

Bei Gleichungssystemen aus Polynomen kann die Berechnungsdauer 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.

$$\text{solve}\left(x+e^{z}\cdot y=1 \text{ and } x-y=\sin(z),\{x,y\}\right)$$
$$x=\frac{e^{z}\cdot\sin(z)+1}{e^{z}+1} \text{ and } y=\frac{-(\sin(z)-1)}{e^{z}+1}$$

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.

$$\text{solve}\left(e^{z}\cdot y=1 \text{ and } -y=\sin(z),\{y,z\}\right)$$
$$y=2.812\text{E}{-10} \text{ and } z=21.9911 \text{ or } y=0.001871\blacktriangleright$$

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

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.

$$\text{solve}\left(e^{z}\cdot y=1 \text{ and } -y=\sin(z),\{y,z=2\cdot\pi\}\right)$$
$$y=0.001871 \text{ and } z=6.28131$$

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 259).

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA *list1*	*Done*
list1	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA *list2*,*list1*	*Done*
list2	$\{1,2,3,4\}$
list1	$\{4,3,2,1\}$

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 259).

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD *list1*,*list2*	*Done*
list1	$\{4,3,2,1\}$
list2	$\{3,4,1,2\}$

▸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 [ρ ∠θ ∠ϕ] an.

Hinweis: Erzwingen eines Näherungsergebnisses,

▸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.

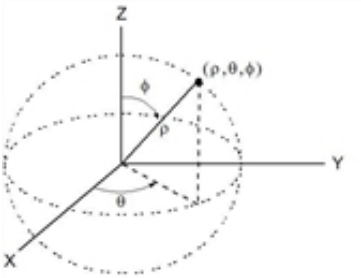
Handheld: Drücken Sie ctrl enter.
Windows®: Drücken Sie **Strg+Eingabetaste**.
Macintosh®: Drücken ⌘+**Eingabetaste**.
iPad®: Halten Sie die **Eingabetaste** gedrückt und wählen Sie ≈ aus.

$$[1 \quad 2 \quad 3]\blacktriangleright\text{Sphere}$$
$$[3.74166 \quad \angle 1.10715 \quad \angle 0.640522]$$

$$\left(\left[2 \quad \angle\frac{\pi}{4} \quad 3\right]\right)\blacktriangleright\text{Sphere}$$
$$[3.60555 \quad \angle 0.785398 \quad \angle 0.588003]$$

Drücken Sie enter.

$$\left(\left[2 \quad \angle\frac{\pi}{4} \quad 3\right]\right)\blacktriangleright\text{Sphere}$$
$$\left[\sqrt{13} \quad \angle\frac{\pi}{4} \quad \angle\sin^{-1}\left(\frac{2\cdot\sqrt{13}}{13}\right)\right]$$



sqrt() (Quadratwurzel) Katalog >

sqrt(*Ausdr1***)⇒***Ausdruck*

sqrt(*Liste1***)⇒***Liste*

Gibt die Quadratwurzel des Arguments zurück.

Bei einer Liste wird die Quadratwurzel für jedes Element von *Liste1* zurückgegeben.

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

sqrt() (Quadratwurzel) Katalog >

Hinweis: Siehe auch **Vorlage Quadratwurzel**, Seite 5.

stat.results Katalog >

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.

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
"r²"	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}"

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.

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR²	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.r²	stat.SSY
stat.b1	stat.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.σx	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.σy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.σx1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.σx2	stat.UpperVal
stat.b8	stat.F	stat.n	stat.Σx	stat.$\bar{x}$
stat.b9	stat.FBlock	Stat.$\hat{p}$	stat.Σx²	stat.$\bar{x}$1
stat.b10	stat.Fcol	stat.$\hat{p}$1	stat.Σxy	stat.$\bar{x}$2
stat.bList	stat.FInteract	stat.$\hat{p}$2	stat.Σy	stat.$\bar{x}$Diff

stat.χ^2	stat.FreqReg	stat.$\hat{p}$Diff	stat.Σy^2	stat.$\bar{x}$List
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.$\bar{y}$
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat.$\hat{y}$
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat.$\hat{y}$List
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

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.

```
Siehe stat.results.
```

stDevPop() (Populations-Standardabweichung)

Katalog >

stDevPop(*Liste*[**,** *Häufigkeitsliste*]**)**⇒*Ausdruck*

Ergibt die Populations-Standardabweichung der Elemente in *Liste*.

Jedes *Häufigkeitsliste*-Element gewichtet die Elemente von *Liste* in der gegebenen Reihenfolge entsprechend.

Im Bogenmaß- und automatischen Modus:

$$\text{stDevPop}(\{a,b,c\}) \qquad \frac{\sqrt{2\cdot(a^2-a\cdot(b+c)+b^2-b\cdot c+c^2)}}{3}$$

$$\text{stDevPop}(\{1,2,5,-6,3,-2\}) \qquad \frac{\sqrt{465}}{6}$$

$$\text{stDevPop}(\{1.3,2.5,-6.4\},\{3,2,5\}) \qquad 4.11107$$

stDevPop() (Populations-Standardabweichung)

Katalog >

Hinweis: *Liste* muss mindestens zwei Elemente haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 259).

stDevPop(*Matrix1*[**,** *Häufigkeitsmatrix*]**)⇒***Matrix*

Ergibt einen Zeilenvektor der Populations-Standardabweichungen der Spalten in *Matrix1*.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Matrix1* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 259).

$$\text{stDevPop}\begin{bmatrix}1 & 2 & 5\\ -3 & 0 & 1\\ 5 & 7 & 3\end{bmatrix} \quad \begin{bmatrix}\frac{4\cdot\sqrt{6}}{3} & \frac{\sqrt{78}}{3} & \frac{2\cdot\sqrt{6}}{3}\end{bmatrix}$$

$$\text{stDevPop}\begin{bmatrix}-1.2 & 5.3\\ 2.5 & 7.3\\ 6 & -4\end{bmatrix},\begin{bmatrix}4 & 2\\ 3 & 3\\ 1 & 7\end{bmatrix} \quad \begin{bmatrix}2.52608 & 5.21506\end{bmatrix}$$

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 259).

$$\text{stDevSamp}(\{a,b,c\}) \quad \frac{\sqrt{3\cdot\left(a^2-a\cdot(b+c)+b^2-b\cdot c+c^2\right)}}{3}$$

$$\text{stDevSamp}(\{1,2,5,-6,3,-2\}) \quad \frac{\sqrt{62}}{2}$$

$$\text{stDevSamp}(\{1.3,2.5,-6.4\},\{3,2,5\}) \quad 4.33345$$

stDevSamp() (Stichproben-Standardabweichung)

Katalog >

stDevSamp(*Matrix1*[**,** *Häufigkeitsmatrix*]**)⇒***Matrix*

Ergibt einen Zeilenvektor der Stichproben-Standardabweichungen der Spalten in *Matrix1*.

Jedes *Häufigkeitsmatrix*-Element gewichtet die Elemente von *Matrix1* in der gegebenen Reihenfolge entsprechend.

Hinweis: *Matrix1* muss mindestens zwei Zeilen haben. Leere (ungültige) Elemente werden ignoriert. Weitere Informationen zu leeren Elementen finden Sie (Seite 259).

$$\text{stDevSamp}\begin{bmatrix}1 & 2 & 5\\ -3 & 0 & 1\\ 5 & 7 & 3\end{bmatrix} \quad \begin{bmatrix}4 & \sqrt{13} & 2\end{bmatrix}$$

$$\text{stDevSamp}\left(\begin{bmatrix}-1.2 & 5.3\\ 2.5 & 7.3\\ 6 & -4\end{bmatrix},\begin{bmatrix}4 & 2\\ 3 & 3\\ 1 & 7\end{bmatrix}\right) \quad \begin{bmatrix}2.7005 & 5.44695\end{bmatrix}$$

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()=Prgm                    Done
  For i,1,10,1
  If i=5
  Stop
  EndFor
  EndPrgm
prog1()                                Done
i                                      5
```

Store (Speichern)

Siehe → (speichern), Seite 257.

string() (String)

Katalog >

string(*Ausdr***)⇒***String*

Vereinfacht *Ausdr* und gibt das Ergebnis als Zeichenkette zurück.

```
string(1.2345)                         "1.2345"
string(1+2)                            "3"
string(cos(x)+√(3))                    "cos(x)+√(3)"
```

subMat() (Untermatrix)

Katalog >

subMat(*Matrix1*[**,** *vonZei*] [**,** *vonSpl*] [**,** *bisZei*] [**,** *bisSpl*]**)** ⇒*Matrix*

Gibt die angegebene Untermatrix von *Matrix1* zurück.

Vorgaben: *vonZei*=1, *vonSpl*=1, *bisZei*=letzte Zeile, *bisSpl*=letzte Spalte.

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\rightarrow m1 \qquad \begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$$

$$\text{subMat}(m1,2,1,3,2) \qquad \begin{bmatrix}4&5\\7&8\end{bmatrix}$$

$$\text{subMat}(m1,2,2) \qquad \begin{bmatrix}5&6\\8&9\end{bmatrix}$$

Summe (Sigma)

Siehe Σ(), Seite 246.

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 259).

$$\text{sum}(\{1,2,3,4,5\}) \qquad 15$$

$$\text{sum}(\{a,2\cdot a,3\cdot a\}) \qquad 6\cdot a$$

$$\text{sum}(\text{seq}(n,n,1,10)) \qquad 55$$

$$\text{sum}(\{1,3,5,7,9\},3) \qquad 21$$

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 259).

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

$$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right) \qquad \begin{bmatrix}12&15&18\end{bmatrix}$$

$$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2,3\right) \qquad \begin{bmatrix}11&13&15\end{bmatrix}$$

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 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 259).

Hinweis: Siehe auch **countIf()**, Seite 42.

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

sumSeq() **Siehe Σ(), Seite 246.**

system() (System)

Katalog >

system(*Ausdr1* **[,** *Ausdr2* **[,** *Ausdr3* **[, ...]]])**

system(*Glch1* **[,** *Glch2* **[,** *Glch3* **[, ...]]])**

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.

$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=8 \end{cases}, x, y\right)$ $\quad x=4 \text{ and } y=-4$

T

T (Transponierte)

Katalog >

*Matrix1***T**⇒*matrix*

Gibt die komplex konjugierte, transponierte Matrix von *Matrix1* zurück.

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

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

$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^T \quad \begin{bmatrix} a & c \\ b & d \end{bmatrix}$

$\begin{bmatrix} 1+i & 2+i \\ 3+i & 4+i \end{bmatrix}^T \quad \begin{bmatrix} 1-i & 3-i \\ 2-i & 4-i \end{bmatrix}$

tan() (Tangens)

trig Taste

tan(*Ausdr1***)**⇒*Ausdruck*

tan(*Liste1***)**⇒*Liste*

tan(*Ausdr1***)** gibt den Tangens des Arguments als Ausdruck zurück.

tan(*Liste1***)** gibt in Form einer Liste für jedes Element in *Liste1* 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:

$\tan\left(\frac{\pi}{4}_r\right) \quad 1$

$\tan(45) \quad 1$

$\tan(\{0,60,90\}) \quad \{0,\sqrt{3},\text{undef}\}$

Im Neugrad-Modus:

$\tan\left(\frac{\pi}{4}_r\right) \quad 1$

$\tan(50) \quad 1$

$\tan(\{0,50,100\}) \quad \{0,1,\text{undef}\}$

Im Bogenmaß-Modus:

tan() (Tangens)

trig Taste

$\tan\left(\frac{\pi}{4}\right)$ → 1

$\tan(45°)$ → 1

$\tan\left(\left\{\pi, \frac{\pi}{3}, -\pi, \frac{\pi}{4}\right\}\right)$ → $\{0, \sqrt{3}, 0, 1\}$

tan(*Quadratmatrix1***)⇒***Quadratmatrix*

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

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

Im Bogenmaß-Modus:

$$\tan\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}-28.2912 & 26.0887 & 11.1142\\ 12.1171 & -7.83536 & -5.48138\\ 36.8181 & -32.8063 & -10.4594\end{bmatrix}$$

$\tan^{-1}$() (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.

Im Grad-Modus:

$\tan^{-1}(1)$ → 45

Im Neugrad-Modus:

$\tan^{-1}(1)$ → 50

Im Bogenmaß-Modus:

$\tan^{-1}(\{0, 0.2, 0.5\})$ → $\{0, 0.197396, 0.463648\}$

$\tan^{-1}$(*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()**.

Im Bogenmaß-Modus:

$$\tan^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}-0.083658 & 1.26629 & 0.62263\\ 0.748539 & 0.630015 & -0.070012\\ 1.68608 & -1.18244 & 0.455126\end{bmatrix}$$

$\tan^{-1}()$ (Arkustangens)

trig Taste

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

tangentLine()

Katalog >

tangentLine
(*Ausdr1*,*Var*,*Punkt***)**⇒*Ausdruck*

tangentLine
(*Ausdr1*,*Var***=***Punkt***)**⇒*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.

$\text{tangentLine}(x^2,x,1)$	$2\cdot x-1$
$\text{tangentLine}((x-3)^2-4,x=3)$	-4
$\text{tangentLine}\left(x^{\frac{1}{3}},x=0\right)$	$x=0$
$\text{tangentLine}(\sqrt{x^2-4},x=2)$	undef
$x:=3:\ \text{tangentLine}(x^2,x,1)$	5

tanh() (Tangens hyperbolicus)

Katalog >

tanh(*Ausdr1***)**⇒*Ausdruck*

tanh(*Liste1***)**⇒*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***)**⇒*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:

$$\tanh\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

$$\begin{bmatrix}-0.097966 & 0.933436 & 0.425972\\ 0.488147 & 0.538881 & -0.129382\\ 1.28295 & -1.03425 & 0.428817\end{bmatrix}$$

tanh⁻¹() (Arkustangens hyperbolicus) Katalog >

tanh⁻¹(*Ausdr1***)⇒***Ausdruck*

tanh⁻¹(*Liste1***)⇒***Liste*

tanh⁻¹(*Ausdr1***)** gibt den inversen Tangens 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.

Im Komplex-Formatmodus "kartesisch":

$\tanh^{-1}(0) \quad 0$

$\tanh^{-1}(\{1,2.1,3\}) \quad \left\{\text{undef}, 0.518046-1.5708\cdot i, \frac{\ln(2)}{2}-\frac{\pi}{2}\cdot i\right\}$

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

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$\tanh^{-1}\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right)$$

$$\begin{bmatrix}-0.099353+0.164058\cdot i & 0.267834-1.4908 \\ -0.087596-0.725533\cdot i & 0.479679-0.9473 \\ 0.511463-2.08316\cdot i & -0.878563+1.7901\end{bmatrix}$$

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.

$\text{taylor}\left(e^{\sqrt{x}},x,2\right) \quad \text{taylor}\left(e^{\sqrt{x}},x,2,0\right)$

$\text{taylor}\left(e^{t},t,4\right)\Big|t=\sqrt{x} \quad \frac{x^2}{24}+\frac{x^3}{6}+\frac{x}{2}+\sqrt{x}+1$

$\text{taylor}\left(\frac{1}{x\cdot(x-1)},x,3\right) \quad \text{taylor}\left(\frac{1}{x\cdot(x-1)},x,3,0\right)$

$\text{expand}\left(\frac{\text{taylor}\left(\frac{x}{x\cdot(x-1)},x,4\right)}{x},x\right) \quad -x^3-x^2-x-\frac{1}{x}-1$

taylor() (Taylor-Polynom)

Katalog >

Punkt ist vorgegeben als Null und ist der Entwicklungspunkt.

tCdf()

Katalog >

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 ≤ *obereGrenze*) setzen Sie *untereGrenze* = −∞.

tCollect() (Trigonometrische Zusammenfassung)

Katalog >

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 getrenntenSchritten **tExpand()** auf ein Ergebnis von **tCollect()** anwendet (oder umgekehrt).

$\text{tCollect}\left(\left(\cos\left(\alpha\right)\right)^2\right)$ $\quad\dfrac{\cos\left(2\cdot\alpha\right)+1}{2}$

$\text{tCollect}\left(\sin\left(\alpha\right)\cdot\cos\left(\beta\right)\right)$ $\quad\dfrac{\sin\left(\alpha-\beta\right)+\sin\left(\alpha+\beta\right)}{2}$

tExpand() (Trigonometrische Entwicklung)

Katalog >

tExpand(*Ausdr1***)⇒***Ausdruck*

Gibt einen Ausdruck zurück, in dem Sinus und Cosinus von ganzzahligen 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 getrenntenSchritten **tCollect()** auf ein Ergebnis von **tExpand()** anwendet (oder umgekehrt).

Hinweis: Die Skalierung von π/180 im Winkelmodus "Grad" behindert die Erkennung entwickelbarer Formen durch **tExpand()**. Die besten Ergebnisse werden bei Benutzung von **tExpand()** im Bogenmaß-Modus erzielt.

$\text{tExpand}(\sin(3\cdot\varphi))$ $\quad 4\cdot\sin(\varphi)\cdot(\cos(\varphi))^2-\sin(\varphi)$

$\text{tExpand}(\cos(\alpha-\beta))$ $\quad \cos(\alpha)\cdot\cos(\beta)+\sin(\alpha)\cdot\sin(\beta)$

Text

Katalog >

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.

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 ↵ ab anstatt mit enter. Auf der Computertastatur halten Sie **Alt** gedrückt und drücken die **Eingabetaste**.

```
Define text_demo()=Prgm
  For i,1,5
```

- 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 162, oder**RequestStr**, Seite 164.

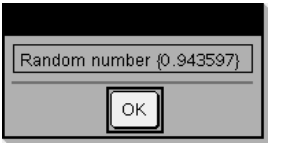
Hinweis: Sie können diesen Befehl in benutzerdefinierten Programmen, aber nicht in Funktionen verwenden.

```
    strinfo:="Random number " &
string(rand(i))
    Text strinfo
    EndFor
EndPrgm
```

Starten Sie das Programm:

```
text_demo()
```

Muster eines Dialogfelds:



Then **Siehe If, Seite 96.**

tInterval **Katalog >**

tInterval *Liste*[**,***Häuf*[**,***KNiv*]]

(Datenlisteneingabe)

tInterval $\bar{x}$**,***sx*,*n***[,***KNiv***]**

(Zusammenfassende statistische Eingabe)

Berechnet das Konfidenzintervall *t*. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekannten Populationsmittelwert

Ausgabevariable	Beschreibung
stat.$\bar{x}$	Stichprobenmittelwert der Datenfolge aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.df	Freiheitsgrade
stat.σx	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*,*sx1*,*n1*,$\bar{x}$*2*,*sx2*,*n2*[,*KStufe*[,*Verteilt*]]

(Zusammenfassende statistische Eingabe)

Berechnet ein *t*-Konfidenzintervall für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 193.)

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 259).

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

Ausgabevariable	Beschreibung
stat.σx1, stat.σx2	Stichproben-Standardabweichungen für *Liste 1* und *Liste 2*
stat.n1, stat.n2	Anzahl der Stichproben in Datenfolgen
stat.sp	Die verteilte Standardabweichung. Wird berechnet, wenn *Verteilt* = JA.

tmpCnv() (Konvertierung von Temperaturwerten)

Katalog >

tmpCnv(*Ausdr_°TempEinh***, _°***TempEinh2***)** ⇒*Ausdruck _°TempEinh2*

Konvertiert einen durch *Ausdr* definierten Temperaturwert von einer Einheit in eine andere. Folgende Temperatureinheiten sind gültig:

_°C Celsius

_°F Fahrenheit

_°K Kelvin

_°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 Δ**tmpCnv()**.

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_°tempEinh***, _°***tempEinh2***)** ⇒*Ausdruck _°tempEinh2*

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

Wählen Sie zur Eingabe von Δ das Symbol aus der Sonderzeichenpalette des Katalogs aus.

ΔtmpCnv() (Konvertierung von Temperaturbereichen)

Katalog >

Konvertiert einen durch *Ausdr* definierten Temperaturbereich (Differenz zwischen zwei Temperaturwerten) von einer Einheit in eine andere. Folgende Temperatureinheiten sind gültig:

_°C Celsius

_°F Fahrenheit

_°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()**.

$\Delta\text{tmpCnv}(100\cdot\_°C,\_°F)$	$180.\cdot\_°F$
$\Delta\text{tmpCnv}(180\cdot\_°F,\_°C)$	$100.\cdot\_°C$
$\Delta\text{tmpCnv}(100\cdot\_°C,\_°K)$	$100.\cdot\_°K$
$\Delta\text{tmpCnv}(100\cdot\_°F,\_°R)$	$100.\cdot\_°R$
$\Delta\text{tmpCnv}(1\cdot\_°C,\_°F)$	$1.8\cdot\_°F$

Hinweis: Sie können den Katalog verwenden, um Temperatureinheiten auszuwählen.

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.

$\text{trace}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$ → 15

$\text{trace}\begin{bmatrix}a&0\\1&a\end{bmatrix}$ → $2\cdot a$

Try (Versuche) Katalog >

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 265).

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.

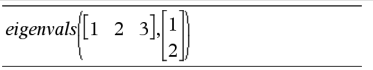
```
Define prog1()=Prgm
  Try
  z:=z+1
  Disp "z incremented."
  Else
  Disp "Sorry, z undefined."
  EndTry
  EndPrgm
                                  Done
z:=1:prog1()
                    z incremented.
                                  Done
DelVar z:prog1()
                    Sorry, z undefined.
                                  Done
```

Beispiel 2

Um die Befehle **Versuche (Try)**, **LöFehler (ClrErr)** und **ÜbgebFeh (PassErr)** im Betrieb zu sehen, geben Sie das rechts gezeigte Programm eigenvals() ein. Sie starten das Programm, indem Sie jeden der folgenden Ausdrücke eingeben.

$eigenvals\left(\begin{bmatrix}-3\\-41\\5\end{bmatrix},\begin{bmatrix}-1&2&-3.1\end{bmatrix}\right)$

```
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)
```



Hinweis: Siehe auch **LöFehler**, Seite 31, und **ÜbgebFeh**, Seite 144.

```
Else
  If errCode=230 Then
    Disp "Fehler: Produkt von A·B muss eine quadratische Matrix sein"
    ClrErr
  Else
    PassErr
  EndIf
EndTry
EndPrgm
```


tTest $\mu 0$**,***Liste*[**,***Häufigkeit*[**,***Hypoth*]]

(Datenlisteneingabe)

tTest $\mu 0$**,**$\bar{x}$**,***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 193.)

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 259).

Ausgabevariable	Beschreibung
stat.t	$(\bar{x} - \mu 0) / (\text{stdev} / \text{sqrt}(n))$
stat.PVal	Kleinste Signifikanzebene, bei der die Nullhypothese verworfen werden kann
stat.df	Freiheitsgrade
stat.$\bar{x}$	Stichprobenmittelwert der Datenfolge in *Liste*
stat.sx	Stichproben-Standardabweichung der Datenfolge
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 193.)

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 259).

Ausgabevariable	Beschreibung
stat.t	Für die Differenz der Mittelwerte berechneter Standardwert

Ausgabevariable	Beschreibung
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 *Liste 1* und *Liste 2*
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in *Liste 1* und *Liste 2*
stat.n1, stat.n2	Stichprobenumfang
stat.sp	Die verteilte Standardabweichung. Wird berechnet, wenn *Verteilt*=1.

tvmFV()

Katalog >

tvmFV(*N*,*I*,*PV*,*Pmt*,[*PpY*],[*CpY*],[*PmtAt*]**)**⇒*Wert*

Finanzfunktion, die den Geld-Endwert berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 213) beschrieben. Siehe auch **amortTbl()**, Seite 12.

tvmFV(120,5,0,-500,12,12) 77641.1

tvmI()

Katalog >

tvmI(*N*,*PV*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)**⇒*Wert*

Finanzfunktion, die den jährlichen Zinssatz berechnet.

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 213) beschrieben. Siehe auch **amortTbl()**, Seite 12.

tvmI(240,100000,-1000,0,12,12) 10.5241

tvmN()

Katalog >

tvmN(*I*,*PV*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)**⇒*Wert*

Finanzfunktion, die die Anzahl der Zahlungsperioden berechnet.

tvmN(5,0,-500,77641,12,12) 120.

tvmN()

Katalog >

Hinweis: Die in den TVM-Funktionen verwendeten Argumente werden in der Tabelle der TVM-Argumente (Seite 213) beschrieben. Siehe auch **amortTbl()**, Seite 12.

tvmPmt()

Katalog >

tvmPmt(*N*,*I*,*PV*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)⇒***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 213) beschrieben. Siehe auch **amortTbl()**, Seite 12.

tvmPV()

Katalog >

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 213) 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

TVM-Argumente*	Beschreibung	Datentyp
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 (Zwei Variable) Katalog >

TwoVar *X*, *Y*[, [*Häuf*] [, *Kategorie*, *Mit*]]

Berechnet die 2-Variablen-Statistik. Eine Zusammenfassung der Ergebnisse wird in der Variablen *stat.results* gespeichert. (Seite 193.)

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 259).

Ausgabevariable	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
Stat. xy	Summe der x ·y-Werte
stat.r	Korrelationskoeffizient
stat.MinX	Minimum der x-Werte
stat.Q_1X	1. Quartil von x
stat.MedianX	Median von x
stat.Q_3X	3. Quartil von x
stat.MaxX	Maximum der x-Werte
stat.MinY	Minimum der y-Werte
stat.Q_1Y	1. Quartil von y
stat.MedY	Median von y
stat.Q_3Y	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

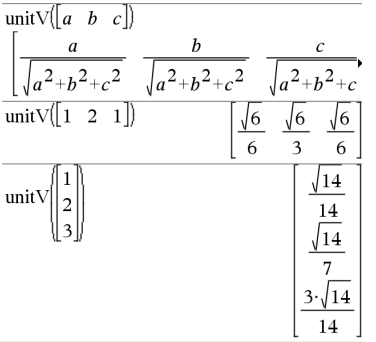
unitV() (Einheitsvektor)

Katalog >

unitV(*Vektor1***)⇒***Vektor*

Gibt je nach der Form von *Vektor1* entweder einen Zeilen- oder einen Spalteneinheitsvektor zurück.

Vektor1 muss eine einzeilige oder eine einspaltige Matrix sein.



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. Gesperrte Variablen können nicht geändert oder gelöscht werden.

Siehe **Lock**, Seite 116, und **getLockInfo()**, Seite 91.

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*

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.

varPop({5,10,15,20,25,30})	$\frac{875}{12}$
Ans·1.	72.9167

varPop() (Populationsvarianz) Katalog >

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 259).

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.

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 259).

$\text{varSamp}(\{a,b,c\})$ $\quad \frac{a^2-a\cdot(b+c)+b^2-b\cdot c+c^2}{3}$

$\text{varSamp}(\{1,2,5,-6,3,-2\})$ $\quad \frac{31}{2}$

$\text{varSamp}(\{1,3,5\},\{4,6,2\})$ $\quad \frac{68}{33}$

varSamp(*Matrix1*[**,** *Häufigkeitsmatrix*]**)⇒***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 259).

Hinweis: *Matrix1* muss mindestens zwei Zeilen enthalten.

$\text{varSamp}\left(\begin{bmatrix}1 & 2 & 5\\ -3 & 0 & 1\\ .5 & .7 & 3\end{bmatrix}\right)$ $\quad \begin{bmatrix}4.75 & 1.03 & 4\end{bmatrix}$

$\text{varSamp}\left(\begin{bmatrix}-1.1 & 2.2\\ 3.4 & 5.1\\ -2.3 & 4.3\end{bmatrix},\begin{bmatrix}6 & 3\\ 2 & 4\\ 5 & 1\end{bmatrix}\right)$ $\quad \begin{bmatrix}3.91731 & 2.08411\end{bmatrix}$

Wait

Katalog >

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 rundet diesen Wert auf die nächsten 0,1 Sekunden auf.

Zum Abbrechen eines **Wait** das gerade durchgeführt wird,

- **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: Sie können den Befehl **Wait** in einem benutzerdefinierten Programm, aber nicht in einer Funktion verwenden.

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.

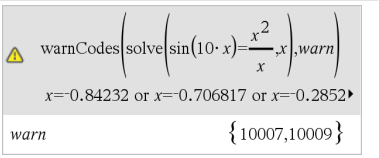
```
Send "SET GREEN 1 ON"
Wait 0.5
Send "SET GREEN 1 OFF"
```

warnCodes ()

Katalog >

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.



warnCodes ()

Katalog >

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 274).

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

when() (Wenn)

Katalog >

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.

$\text{when}(x<0, x+3) \mid x=5$ undef

$\text{when}(n>0, n\cdot factoral(n-1), 1) \rightarrow factoral(n)$ *Done*

$factoral(3)$ 6

$3!$ 6

While

Katalog >

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 $sum\_of\_recip(n)$=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 $tempsum$
EndFunc

Done

$sum\_of\_recip(3)$ $\frac{11}{6}$

X

xor (Boolesches exklusives oder)

Katalog >

*BoolescherAusd1***xor***BoolescherAusdr2*
ergibt *Boolescher Ausdruck*

*BoolescheListe1***xor***BoolescheListe2*
ergibt *Boolesche Liste*

*BoolescheMatrix1***xor***BoolescheMatrix2*
ergibt *Boolesche Matrix*

Gibt wahr zurück, wenn *Boolescher Ausdr1* wahr und *Boolescher Ausdr2* 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 142.

true xor true	false
5>3 xor 3>5	true

Ganzzahl1 **xor** *Ganzzahl2* ⇒ *Ganzzahl*

Im Hex-Modus:

Wichtig: Null, nicht Buchstabe O

0h7AC36 xor 0h3D5F	0h79169

xor (Boolesches exklusives oder) Katalog >

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 unter ▸**Base2**, Seite 22.

Hinweis: Siehe **or**, Seite 142.

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.

Z

zeros() (Nullstellen) Katalog >

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 **exp▸list(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.

$$\text{zeros}\left(a\cdot x^2+b\cdot x+c,x\right)$$
$$\left\{\frac{\sqrt{b^2-4\cdot a\cdot c}-b}{2\cdot a},\frac{-\left(\sqrt{b^2-4\cdot a\cdot c}+b\right)}{2\cdot a}\right\}$$

$$a\cdot x^2+b\cdot x+c|x=Ans[2] \qquad 0$$

$$\text{exact}\left(\text{zeros}\left(a\cdot\left(e^x+x\right)\cdot\left(\text{sign}\left(x\right)-1\right),x\right)\right) \qquad \{\}$$

$$\text{exact}\left(\text{solve}\left(a\cdot\left(e^x+x\right)\cdot\left(\text{sign}\left(x\right)-1\right)=0,x\right)\right)$$
$$e^x+x=0 \text{ or } x>0 \text{ or } a=0$$

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:

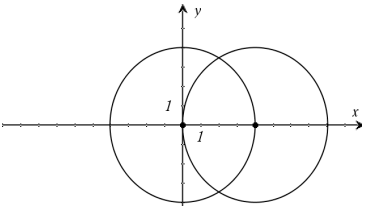
Variable

– oder –

Variable = reell oder nicht-reelle Zahl

Beispiel: x ist gültig und x=3 ebenfalls.

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.

$$\text{zeros}\left\{\left\{x^2+y^2-r^2,(x-r)^2+y^2-r^2\right\},\{x,y\}\right\}$$

$$\begin{bmatrix} \frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2} \\ \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} \end{bmatrix}$$

Zeile 2 extrahieren:

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.

$Ans[2]$

$$\begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} \end{bmatrix}$$

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.

$\text{zeros}\left(\left\{x^2+y^2-r^2,(x-r)^2+y^2-r^2\right\},\{x,y,z\}\right)$

$$\begin{bmatrix} \frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2} & c1 \\ \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} & c1 \end{bmatrix}$$

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 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.

$\text{zeros}\left(\left\{x+e^z\cdot y-1,x-y-\sin(z)\right\},\{x,y\}\right)$

$$\begin{bmatrix} \frac{e^z\cdot\sin(z)+1}{e^z+1} & \frac{-(\sin(z)-1)}{e^z+1} \end{bmatrix}$$

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.

$\text{zeros}\left(\left\{e^z\cdot y-1,-y-\sin(z)\right\},\{y,z\}\right)$

$$\begin{bmatrix} 0.041458 & 3.18306 \\ 0.001871 & 6.28131 \\ 4.76\text{E}{-11} & 1796.99 \\ 2.\text{E}{-13} & 254.469 \end{bmatrix}$$

zeros() (Nullstellen)

Katalog >

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.

$$\text{zeros}\left(\left\{e^{z}\cdot y-1,\ -y-\sin(z)\right\},\{y,z=2\cdot\pi\}\right)$$
$$\left[0.001871 \quad 6.28131\right]$$

zInterval (z-Konfidenzintervall)

Katalog >

zInterval σ,*Liste*[,*Häufigkeit*[,*KStufe*]]

(Datenlisteneingabe)

zInterval σ,$\bar{x}$,*n* [,*KStufe*]

(Zusammenfassende statistische Eingabe)

Berechnet ein *z*-Konfidenzintervall. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 193.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall für den unbekannten Populationsmittelwert
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 *Liste*

zInterval_1Prop (z-Konfidenzintervall für eine Proportion)

Katalog >

zInterval_1Prop *x*,*n* [,*KStufe*]

zInterval_1Prop (z-Konfidenzintervall für eine Proportion)

Katalog >

Berechnet ein *z*-Konfidenzinterval für eine Proportion. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 193.)

x ist eine nicht negative Ganzzahl.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	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 193.)

x1 und *x2* sind nicht negative Ganzzahlen.

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

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

Ausgabevariable	Beschreibung
stat.n1	Stichprobenumfang in Datenfolge eins
stat.n2	Stichprobenumfang in Datenfolge zwei

zInterval_2Samp (z-Konfidenzintervall für zwei Stichproben)

Katalog >

zInterval_2Samp σ_1,σ_2 ,*Liste1*,*Liste2* [,*Häufigkeit1*[,*Häufigkeit2*,[*KStufe*]]]

(Datenlisteneingabe)

zInterval_2Samp σ_1,σ_2,$\bar{x}$*1*,*n1*,$\bar{x}$*2*,*n2* [,*KStufe*]

(Zusammenfassende statistische Eingabe)

Berechnet ein *z*-Konfidenzintervall für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 193.)

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

Ausgabevariable	Beschreibung
stat.CLower, stat.CUpper	Konfidenzintervall mit dem Konfidenzniveau der Verteilungswahrscheinlichkeit
stat.$\bar{x}$1-$\bar{x}$2	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat.ME	Fehlertoleranz
stat.$\bar{x}$1, stat.$\bar{x}$2	Stichprobenmittelwerte der Datenfolgen aus der zufälligen Normalverteilung
stat.σx1, stat.σx2	Stichproben-Standardabweichungen für *Liste 1* und *Liste 2*
stat.n1, stat.n2	Anzahl der Stichproben in Datenfolgen
stat.r1, stat.r2	Bekannte Populations-Standardabweichungen für Datenfolge *Liste 1* und *Liste 2*

zTest

Katalog >

zTest μ*0*,σ,*Liste*,[*Häufigkeit*[,*Hypoth*]]

(Datenlisteneingabe)

zTest

Katalog >

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 193.)

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 259).

Ausgabevariable	Beschreibung
stat.z	$(\bar{x} - \mu 0) / (\sigma / \text{sqrt}(n))$
stat.P Value	Kleinste Wahrscheinlichkeit, bei der die Nullhypothese verworfen werden kann
stat.$\bar{x}$	Stichprobenmittelwert der Datenfolge in *Liste*
stat.sx	Stichproben-Standardabweichung der Datenfolge. Wird nur für *Daten*eingabe zurückgegeben.
stat.n	Stichprobenumfang

zTest_1Prop (z-Test für eine Proportion)

Katalog >

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 193.)

x ist eine nicht negative Ganzzahl.

Getestet wird H_0: $p = p0$ in Bezug auf eine der folgenden Alternativen:

zTest_1Prop (z-Test für eine Proportion)

Katalog >

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 259).

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 (z-Test für zwei Proportionen)

Katalog >

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 193.)

x1 und *x2* sind nicht negative Ganzzahlen.

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: $p < p0$ setzen Sie *Hypoth*<0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter "Leere (ungültige) Elemente" (Seite 259).

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 (z-Test für zwei Stichproben)

Katalog >

zTest_2Samp σ_1,σ_2 ,*Liste1*,*Liste2* [,*Häufigkeit1*[,*Häufigkeit2*[,*Hypoth*]]]

(Datenlisteneingabe)

zTest_2Samp σ_1,σ_2,$\bar{x}1$,*n1*,$\bar{x}2$,*n2*[,*Hypoth*]

(Zusammenfassende statistische Eingabe)

Berechnet einen *z*-Test für zwei Stichproben. Eine Zusammenfassung der Ergebnisse wird in der Variable *stat.results* gespeichert. (Seite 193.)

Getestet wird H_0: $\mu1 = \mu2$ in Bezug auf eine der folgenden Alternativen:

Für H_a: $\mu1 < \mu2$ setzen Sie *Hypoth*<0

Für H_a: $\mu1 \neq \mu2$ (Standard) setzen Sie *Hypoth*=0

Für H_a: $\mu1 > \mu2$ setzen Sie *Hypoth*>0

Informationen zu den Auswirkungen leerer Elemente in einer Liste finden Sie unter “Leere (ungültige) Elemente” (Seite 259).

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 *Liste 1* und *Liste 2*

Ausgabevariable	Beschreibung
stat.sx1, stat.sx2	Stichproben-Standardabweichungen der Datenfolgen in *Liste 1* und *Liste 2*
stat.n1, stat.n2	Stichprobenumfang

Sonderzeichen

+ (addieren)

Taste: ⊞

Ausdr1 **+** *Ausdr2*⇒*Ausdruck*

Gibt die Summe der beiden Argumente zurück.

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.

$\left\{22,\pi,\frac{\pi}{2}\right\}\rightarrow l1$	$\left\{22,\pi,\frac{\pi}{2}\right\}$
$\left\{10{,}5,\frac{\pi}{2}\right\}\rightarrow l2$	$\left\{10{,}5,\frac{\pi}{2}\right\}$
$l1+l2$	$\left\{32,\pi+5,\pi\right\}$
$Ans+\left\{\pi,-5,-\pi\right\}$	$\left\{\pi+32,\pi,0\right\}$
$\begin{bmatrix}a & b\\ c & d\end{bmatrix}+\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}a+1 & b\\ c & d+1\end{bmatrix}$

Ausdr **+** *Liste1*⇒*Liste*

Liste1 **+** *Ausdr*⇒*Liste*

Gibt eine Liste zurück, die die Summen von *Ausdr* plus jedem Element der *Liste1* enthält.

$15+\left\{10,15,20\right\}$	$\left\{25,30,35\right\}$
$\left\{10,15,20\right\}+15$	$\left\{25,30,35\right\}$

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.

$20+\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$	$\begin{bmatrix}21 & 2\\ 3 & 24\end{bmatrix}$

−(subtrahieren)

Taste: ⊟

Ausdr1 − *Ausdr2*⇒*Ausdruck*

Gibt *Ausdr1* minus *Ausdr2* zurück.

6−2	4
$\pi-\frac{\pi}{6}$	$\frac{5\cdot\pi}{6}$

−(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.

$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\}$ → $\left\{12,\pi-5,0\right\}$

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

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.

$15-\left\{10,15,20\right\}$ → $\left\{5,0,-5\right\}$

$\left\{10,15,20\right\}-15$ → $\left\{-5,0,5\right\}$

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.

$20-\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ → $\begin{bmatrix}19 & -2\\ -3 & 16\end{bmatrix}$

Hinweis: Verwenden Sie **.**− (Punkt Minus) zum Subtrahieren eines Ausdrucks von jedem Element.

·(multiplizieren) — Taste ×

Ausdr1•*Ausdr2*⇒*Ausdruck*

Gibt das Produkt der beiden Argumente zurück.

$2\cdot 3.45$ → 6.9

$x\cdot y\cdot x$ → $x^2\cdot y$

Liste1•*Liste2*⇒*Liste*

Gibt eine Liste zurück, die die Produkte der entsprechenden Elemente aus *Liste1* und *Liste2* enthält.

$\left\{1.,2,3\right\}\cdot\left\{4,5,6\right\}$ → $\left\{4.,10,18\right\}$

$\left\{\frac{2}{a},\frac{3}{2}\right\}.\cdot\left\{a^2,\frac{b}{3}\right\}$ → $\left\{2\cdot a,\frac{b}{2}\right\}$

·(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.

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}.\begin{bmatrix}a & d\\ b & e\\ c & f\end{bmatrix} \qquad \begin{bmatrix}a+2\cdot b+3\cdot c & d+2\cdot e+3\cdot f\\ 4\cdot a+5\cdot b+6\cdot c & 4\cdot d+5\cdot e+6\cdot f\end{bmatrix}$$

Ausdr•Liste1⇒Liste

Liste1•Ausdr⇒Liste

Gibt eine Liste zurück, die die Produkte von *Ausdr* und jedem Element der *Liste1* enthält.

$$\pi\cdot\{4,5,6\} \qquad \{4\cdot\pi,5\cdot\pi,6\cdot\pi\}$$

Ausdr•Matrix1⇒Matrix

Matrix1•Ausdr⇒Matrix

Gibt eine Matrix zurück, die die Produkte von *Ausdr* und jedem Element der *Matrix1* enthält.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\cdot 0.01 \qquad \begin{bmatrix}0.01 & 0.02\\ 0.03 & 0.04\end{bmatrix}$$

$$\lambda\cdot\text{identity}(3) \qquad \begin{bmatrix}\lambda & 0 & 0\\ 0 & \lambda & 0\\ 0 & 0 & \lambda\end{bmatrix}$$

Hinweis: Verwenden Sie **.** ·(Punkt-Multiplikation) zum Multiplizieren eines Ausdrucks mit jedem Element.

/(dividieren)

÷Taste

Ausdr1 / Ausdr2⇒Ausdruck

Gibt *Ausdr1* dividiert durch *Ausdr2* zurück.

Hinweis: Siehe auch **Vorlage Bruch**, Seite 5.

$$\frac{2}{3.45} \qquad .57971$$

$$\frac{x^3}{x} \qquad x^2$$

Liste1 / Liste2⇒Liste

Gibt eine Liste der Elemente von *Liste1* dividiert durch *Liste2* zurück.

Die Listen müssen die gleiche Dimension besitzen.

$$\frac{\{1.,2,3\}}{\{4,5,6\}} \qquad \left\{0.25,\frac{2}{5},\frac{1}{2}\right\}$$

Ausdr / Liste1 ⇒ Liste

Liste1 / Ausdr ⇒ Liste

$$\frac{a}{\{3,a,\sqrt{a}\}} \qquad \left\{\frac{a}{3},1,\sqrt{a}\right\}$$

$$\frac{\{a,b,c\}}{a\cdot b\cdot c} \qquad \left\{\frac{1}{b\cdot c},\frac{1}{a\cdot c},\frac{1}{a\cdot b}\right\}$$

/ (dividieren)

÷ Taste

Gibt eine Liste der Elemente von *Ausdr* dividiert durch *Liste1* oder *Liste1* dividiert durch *Ausdr* zurück.

Matrix1 / *Ausdr* ⇒ *Matrix*

Gibt eine Matrix zurück, die die Quotienten *Matrix1*/*Ausdr* enthält.

Hinweis: Verwenden Sie **. /** (Punkt-Division) zum Dividieren eines Ausdrucks durch jedes Element.

$$\frac{\begin{bmatrix} a & b & c \end{bmatrix}}{a \cdot b \cdot c} \qquad \begin{bmatrix} \frac{1}{b \cdot c} & \frac{1}{a \cdot c} & \frac{1}{a \cdot b} \end{bmatrix}$$

^ (Potenz)

^ Taste

Ausdr1 **^** *Ausdr2* ⇒ *Ausdruck*

Liste1 **^** *Liste2* ⇒ *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.

$$4^2 \qquad 16$$

$$\{a,2,c\}^{\{1,b,3\}} \qquad \{a,2^b,c^3\}$$

Ausdr **^** *Liste1* ⇒ *Liste*

Gibt *Ausdr* hoch den Elementen von *Liste1* zurück.

$$p^{\{a,2,-3\}} \qquad \left\{p^a, p^2, \frac{1}{p^3}\right\}$$

Liste1 **^** *Ausdr* ⇒ *Liste*

Gibt die Elemente von *Liste1* hoch *Ausdr* zurück.

$$\{1,2,3,4\}^{-2} \qquad \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}\right\}$$

^ (Potenz)

[^] Taste

Quadratmatrix1 **^** *Ganzzahl* ⇒ *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.

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^2$ → $\begin{bmatrix}7 & 10\\ 15 & 22\end{bmatrix}$

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1}$ → $\begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-2}$ → $\begin{bmatrix}\frac{11}{2} & \frac{-5}{2}\\ \frac{-15}{4} & \frac{7}{4}\end{bmatrix}$

x² (Quadrat)

[x²] Taste

*Ausdr1*² ⇒ *Ausdruck*

Gibt das Quadrat des Arguments zurück.

*Liste1*² ⇒ *Liste*

Gibt eine Liste zurück, die die Produkte der Elemente in *Liste1* enthält.

*Quadratmatrix1*² ⇒ *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

$\{2,4,6\}^2$ → $\{4,16,36\}$

$\begin{bmatrix}2 & 4 & 6\\ 3 & 5 & 7\\ 4 & 6 & 8\end{bmatrix}^2$ → $\begin{bmatrix}40 & 64 & 88\\ 49 & 79 & 109\\ 58 & 94 & 130\end{bmatrix}$

$\begin{bmatrix}2 & 4 & 6\\ 3 & 5 & 7\\ 4 & 6 & 8\end{bmatrix}.^\wedge 2$ → $\begin{bmatrix}4 & 16 & 36\\ 9 & 25 & 49\\ 16 & 36 & 64\end{bmatrix}$

.+ (Punkt-Addition)

[.] [+] Tasten

Matrix1 **.+** *Matrix2* ⇒ *Matrix*

Ausdr **.+** *Matrix1* ⇒ *Matrix*

Matrix1 **.+** *Matrix2* gibt eine Matrix zurück, die 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* ⇒ *Matrix*

Ausdr **.−** *Matrix1* ⇒ *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.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \qquad \begin{bmatrix} a-c & -2 \\ b-d & -2 \end{bmatrix}$$

$$x .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \qquad \begin{bmatrix} x-c & x-4 \\ x-d & x-5 \end{bmatrix}$$

.· (Punkt-Mult.)

. × Tasten

Matrix1 **.·** *Matrix2* ⇒ *Matrix*

Ausdr **.·** *Matrix1* ⇒ *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.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .\cdot \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} a\cdot c & 8 \\ 5\cdot b & 3\cdot d \end{bmatrix}$$

$$x .\cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \begin{bmatrix} a\cdot x & b\cdot x \\ c\cdot x & d\cdot x \end{bmatrix}$$

. / (Punkt-Division)

. ÷ Tasten

Matrix1 **. /** *Matrix2* ⇒ *Matrix*

Ausdr **. /** *Matrix1* ⇒ *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.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} ./ \left(\begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \right) \qquad \begin{bmatrix} \frac{a}{c} & \frac{1}{2} \\ \frac{b}{5} & \frac{3}{d} \end{bmatrix}$$

$$x ./ \left(\begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \right) \qquad \begin{bmatrix} \frac{x}{c} & \frac{x}{4} \\ \frac{x}{5} & \frac{x}{d} \end{bmatrix}$$

.^ (Punkt-Potenz)

.^ Tasten

Matrix1 **.^** *Matrix2* ⇒ *Matrix*

Ausdr **.^** *Matrix1* ⇒ *Matrix*

Matrix1 **.^** *Matrix2* gibt eine Matrix zurück, in der jedes Element aus *Matrix2* Exponent des entsprechenden Elements aus *Matrix1* ist.

Ausdr **.^** *Matrix1* gibt eine Matrix zurück, in der jedes Element aus *Matrix1* Exponent von *Ausdr* ist.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .^ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} a^c & 16 \\ b^5 & 3^d \end{bmatrix}$$

$$x .^ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} x^c & x^4 \\ x^5 & x^d \end{bmatrix}$$

-(Negation)

(−) Taste

-*Ausdr1* ⇒ *Ausdruck*

-*Liste1* ⇒ *Liste*

-*Matrix1* ⇒ *Matrix*

Gibt die Negation des Arguments zurück.

Bei einer Liste oder Matrix werden alle Elementen negiert zurückgegeben.

Ist das Argument eine binäre oder hexadezimale ganze Zahl, ergibt die Negation das Zweierkomplement.

-2.43	-2.43
-{-1,0.4,1.2E19}	{1,-0.4,-1.2E19}
-*a*· -*b*	*a*· *b*

Im Bin-Modus:

Wichtig: Null, nicht Buchstabe O

```
-0b100101
 0b11111111111111111111111111111011
```

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

% (Prozent)

ctrl **📖** Tasten

Ausdr1 **%** ⇒ *Ausdruck*

Liste1 **%** ⇒ *Liste*

Matrix1 **%** ⇒ *Matrix*

$$\frac{argument}{100}$$

Ergibt

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.

% (Prozent)

ctrl Tasten

Bei einer Liste oder einer Matrix wird eine Liste/Matrix zurückgegeben, in der jedes Element durch 100 dividiert ist.

13%	0.13
{{1,10,100}}%	{0.01,0.1,1.}

= (gleich)

= Taste

Ausdr1 = *Ausdr2*⇒*Boolescher Ausdruck*

Liste1 = *Liste2*⇒ *Boolesche Liste*

Matrix1 = *Matrix2*⇒ *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.

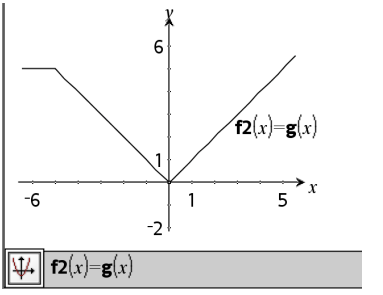
Hinweis zum Eingeben des Beispiels: Anweisungen für die Eingabe von mehrzeiligen Programm- und Funktionsdefinitionen finden Sie im Abschnitt „Calculator“ des Produkthandbuchs.

Beispielfunktion mit den mathematischen Vergleichssymbolen: =, ≠, <, ≤, >, ≥

```
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

Ergebnis der graphischen Darstellung g(x)



≠ (ungleich)

ctrl = Tasten

Ausdr1 ≠ *Ausdr2* ⇒ *Boolescher Ausdruck*

Liste1 ≠ *Liste2* ⇒ *Boolesche Liste*

Matrix1 ≠ *Matrix2* ⇒ *Boolesche Matrix*

Siehe Beispiel bei “=” (gleich).

≠ (ungleich)

ctrl = Tasten

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* ⇒ *Boolescher Ausdruck*

Liste1 **<** *Liste2* ⇒ *Boolesche Liste*

Matrix1 **<** *Matrix2* ⇒ *Boolesche Matrix*

Siehe Beispiel bei "=" (gleich).

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.

≤ (kleiner oder gleich)

ctrl = Tasten

Ausdr1 ≤ *Ausdr2* ⇒ *Boolescher Ausdruck*

Liste1 ≤ *Liste2* ⇒ *Boolesche Liste*

Matrix1 ≤ *Matrix2* ⇒ *Boolesche Matrix*

Siehe Beispiel bei "=" (gleich).

≤ (kleiner oder gleich)

ctrl **=** **Tasten**

Gibt wahr zurück, wenn *Ausdr1* bei Auswertung kleiner oder gleich *Ausdr2* ist.

Gibt falsch zurück, wenn *Ausdr1* bei Auswertung 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* ⇒ *Boolescher Ausdruck*

Liste1 **>** *Liste2* ⇒ *Boolesche Liste*

Matrix1 **>** *Matrix2* ⇒ *Boolesche Matrix*

Siehe Beispiel bei "=" (gleich).

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.

≥ (größer oder gleich)

ctrl **=** **Tasten**

Ausdr1 ≥ *Ausdr2* ⇒ *Boolescher Ausdruck*

Liste1 ≥ *Liste2* ⇒ *Boolesche Liste*

Matrix1 ≥ *Matrix2* ⇒ *Boolesche Matrix*

Siehe Beispiel bei "=" (gleich).

≥ (größer oder gleich)

ctrl = Tasten

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 Vergleichs der einzelnen Elemente zurückgegeben.

Hinweis: Sie können diesen Operator über die Tastatur Ihres Computers eingeben, indem Sie das Tastenkürzel **>=**

⇒ (logische Implikation)

ctrl = Tasten

BoolescherAusd1 ⇒ *BoolescherAusdr2* ergibt *Boolescher Ausdruck*

BoolescheListe1 ⇒ *BooleschеLiset2* ergibt *Boolesche Liste*

BooleschеMatrix1 ⇒ *BooleschеMatrix2* ergibt *Boolesche Matrix*

Ganzzahl1 ⇒ *Ganzzahl2* ergibt *Ganzzahl*

5>3 or 3>5	true
5>3 ⇒ 3>5	false
3 or 4	7
3 ⇒ 4	-4
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} ⇒ {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 **=>**

⇔ (logische doppelte Implikation, XNOR)

ctrl **=** **Tasten**

BoolescherAusdr1 ⇔ *BoolescherAusdr2* ergibt *Boolescher Ausdruck*

BoolescheListe1 ⇔ *BooleschеLiset2* ergibt *Boolesche Liste*

BooleschеMatrix1 ⇔ *BooleschеMatrix2* ergibt *Boolesche Matrix*

Ganzzahl1 ⇔ *Ganzzahl2* ergibt *Ganzzahl*

5>3 xor 3>5	true
5>3 ⇔ 3>5	false
3 xor 4	7
3 ⇔ 4	-8
{1,2,3} xor {3,2,1}	{2,0,2}
{1,2,3} ⇔ {3,2,1}	{-3,-1,-3}

Gibt die Negation einer **XOR** boleschen 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***!** ⇒ *Ausdruck*

*Liste1***!** ⇒ *Liste*

*Matrix1***!** ⇒ *Matrix*

5!	120
{{5,4,3}}!	{120,24,6}
$\left\{\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right\}!$	$\begin{bmatrix}1 & 2\\ 6 & 24\end{bmatrix}$

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* ⇒ *String*

"Hello "&"Nick"	"Hello Nick"

Gibt einen String zurück, der durch Anfügen von *String2* an *String1* gebildet wurde.

d(*Ausdr1*, *Var*[, *Ordnung*]**)⇒***Ausdruck*

d(*Liste1*, *Var*[, *Ordnung*]**)⇒***Liste*

d(*Matrix1*, *Var*[, *Ordnung*]**)⇒***Matrix*

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.

$\frac{d}{dx}\left(f(x)\cdot g(x)\right)$	$\frac{d}{dx}\left(f(x)\right)\cdot g(x)+\frac{d}{dx}\left(g(x)\right)\cdot f(x)$
$\frac{d}{dy}\left(\frac{d}{dx}\left(x^2\cdot y^3\right)\right)$	$6\cdot y^2\cdot x$
$\frac{d}{dx}\left(\left\{x^2,x^3,x^4\right\}\right)$	$\left\{2\cdot x,3\cdot x^2,4\cdot x^3\right\}$

∫(*Ausdr1*, *Var*[, *Untere*, *Obere*]) ⇒ *Ausdruck*

∫(*Ausdr1*, *Var*[, *Konstante*]) ⇒ *Ausdruck*

Gibt das Integral von *Ausdr1* bezüglich der Variablen *Var* von *Untere* bis *Obere* zurück.

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

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.

$\int x^2 \, dx$ $\quad \frac{x^3}{3}$

$\int(a \cdot x^2, x, c)$ $\quad \frac{a \cdot x^3}{3} + c$

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.

∫() gibt sich selbst zurück bei Stücken von *Ausdr1*, die es nicht als explizite endliche Kombination seiner integrierten Funktionen und Operatoren bestimmen kann.

$\int b \cdot e^{-x^2} + \frac{a}{x^2 + a^2} \, dx$ $\quad b \cdot \int e^{-x^2} \, dx + \tan^{-1}\left(\frac{x}{a}\right)$

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.

∫() (Integral)

Katalog >

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 Integration unzulässig ist oder fehlschlägt.

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.

$$\int_{-1}^{1} e^{-x^2}\,dx \qquad 1.49365$$

∫() können verschachtelt werden, um Mehrfach-Integrale zu bearbeiten. Die Integrationsgrenzen können von außerhalb liegenden Integrationsvariablen abhängen.

Hinweis: Siehe auch **nInt()**, Seite 134.

$$\int_0^a \int_0^x \ln(x+y)\,dy\,dx$$

$$\frac{a^2\cdot\ln(a)}{2}+\frac{a^2\cdot(4\cdot\ln(2)-3)}{4}$$

√() (Quadratwurzel)

ctrl x²-Tasten

√ (*Ausdr1*)⇒*Ausdruck*

√ (*Liste1*)⇒*Liste*

Gibt die Quadratwurzel des Arguments zurück.

Bei einer Liste wird die Quadratwurzel für jedes Element von *Liste1* zurückgegeben.

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

Hinweis: Siehe auch **Vorlage Quadratwurzel**, Seite 5.

$$\sqrt{4} \qquad 2$$

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

Π() (ProdSeq)

Katalog >

Π(*Ausdr1*, *Var*, *Von*, *Bis*)⇒*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.

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

$\prod_{k=1}^{n}\left(k^2\right)$ → $(n!)^2$

$\prod_{n=1}^{5}\left(\left\{\frac{1}{n},n,2\right\}\right)$ → $\left\{\frac{1}{120},120,32\right\}$

Π(*Ausdr1*, *Var*, *Von*, *Von*−1)⇒1

Π(*Ausdr1*, *Var*, *Von*, *Bis*) ⇒**1/**Π(*Ausdr1*, *Var*, *Bis*+*1*, *Von*−*1*) if *Bis* < *Von*−*1*

$\prod_{k=4}^{3}\left(k\right)$ → 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.

$\prod_{k=4}^{1}\left(\frac{1}{k}\right)$ → 6

$\prod_{k=4}^{1}\left(\frac{1}{k}\right)\cdot\prod_{k=2}^{4}\left(\frac{1}{k}\right)$ → $\frac{1}{4}$

Σ() (SumSeq)

Katalog >

Σ(*Ausdr1*, *Var*, *Von*, *Bis*)⇒*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.

$\sum_{n=1}^{5}\left(\frac{1}{n}\right)$ → $\frac{137}{60}$

$\sum_{k=1}^{n}\left(k^2\right)$ → $\frac{n\cdot(n+1)\cdot(2\cdot n+1)}{6}$

$\sum_{n=1}^{\infty}\left(\frac{1}{n^2}\right)$ → $\frac{\pi^2}{6}$

Σ() (SumSeq) — Katalog >

Σ(*Ausdr1*, *Var*, *Von*, *Von−1***)⇒**0

Σ(*Ausdr1*, *Var*, *Von*, *Bis***)** ⇒-**Σ(***Ausdr1*, *Var*, *Bis+1*, *Von*−1**)** if *Bis* < *Von−1*

$$\sum_{k=4}^{3}(k) \qquad 0$$

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.

$$\sum_{k=4}^{1}(k) \qquad -5$$

$$\sum_{k=4}^{1}(k)+\sum_{k=2}^{4}(k) \qquad 4$$

ΣInt() — Katalog >

ΣInt(*NPmt1*, *NPmt2*, *N*, *I*, *PV*,[*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*WertRunden*]**)⇒***Wert*

ΣInt(*NPmt1*,*NPmt2*,*AmortTabelle***)⇒***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 213) 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.

ΣInt(1,3,12,4.75,20000,,12,12) -213.48

tbl:=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

ΣInt(1,3,*tbl*) -213.48

ΣInt() Katalog >

ΣInt(*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 ΣPrn() auf dieser und **Bal()**, Seite 21.

ΣPrn() Katalog >

ΣPrn(*NPmt1***,** *NPmt2***,** *N***,** *I***,** *PV***,** [*Pmt*]**,** [*FV*]**,** [*PpY*]**,** [*CpY*]**,** [*PmtAt*]**,** [*WertRunden*]**)⇒***Wert*

ΣPrn(*NPmt1***,***NPmt2***,***AmortTabelle***)⇒***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 213) 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.

ΣPrn(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

ΣPrn(1,3,*tbl*) −4916.28

ΣPrn() Katalog >

ΣPrn(*NPmt1,NPmt2,AmortTabelle***)** berechnet die Summe der gezahlten Tilgungsbeträge auf der Grundlage der Amortisationstabelle *AmortTabelle*. Das Argument *AmortTabelle* muss eine Matrix in der unter **amortTbl()**, Seite 12, beschriebenen Form sein.

Hinweis: Siehe auch ΣInt() auf dieser und **Bal()**, Seite 21.

(Umleitung) ctrl Tasten

*varNameString*

Greift auf die Variable namens *VarNameString* zu. So können Sie innerhalb einer Funktion Variablen unter Verwendung von Strings erzeugen.

#("x"&"y"&"z")	*xyz*

Erzeugt oder greift auf die Variable xyz zu.

$10 \rightarrow r$	10
"r" $\rightarrow s1$	"r"
#*s1*	10

Gibt den Wert der Variable (r) zurück, dessen Name in Variable s1 gespeichert ist.

E (Wissenschaftliche Schreibweise) EE Taste

*Mantisse***E***Exponent*

Gibt eine Zahl in wissenschaftlicher Schreibweise ein. Die Zahl wird als *Mantisse* × $10^{Exponent}$ interpretiert.

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 **2.3@E4** ein, um 2.3**E**4 einzugeben.

23000.	23000.
2300000000.+4.1E15	4.1E15
$3 \cdot 10^4$	30000

g (Neugrad)

π▸ Taste

*Ausdr1***g**⇒*Ausdruck*

*Ausdr1***g**⇒*Ausdruck*

*Liste1***g**⇒*Liste*

*Matrix1***g**⇒*Matrix*

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Bogenmaß-Modus einen Winkel in Neugrad anzugeben.

Im Winkelmodus Bogenmaß wird *Ausdr1* mit π/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.

Im Grad-, Neugrad- oder Bogenmaß-Modus:

$\cos(50^g)$ $\frac{\sqrt{2}}{2}$

$\cos(\{0,100^g,200^g\})$ $\{1,0,-1\}$

r (Bogenmaß)

π▸ Taste

Ausdr1^r⇒*Ausdruck*

Liste1^r⇒*Liste*

Matrix1^r⇒*Matrix*

Diese Funktion gibt Ihnen die Möglichkeit, im Grad- oder Neugrad-Modus einen Winkel im Bogenmaß anzugeben.

Im Winkelmodus Grad wird das Argument mit 180/π multipliziert.

Im Winkelmodus Bogenmaß wird das Argument unverändert zurückgegeben.

Im Neugrad-Modus wird das Argument mit 200/π multipliziert.

Tipp: Verwenden Sie ^{r} in einer Funktionsdefinition, wenn Sie bei Ausführung der Funktion das Bogenmaß frei von der Winkelmoduseinstellung erzwingen möchten.

Im Winkelmodus Grad, Neugrad oder Bogenmaß:

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

$\cos\left(\left\{0^r,\frac{\pi}{12}^r,-(\pi)^r\right\}\right)$ $\left\{1,\frac{(\sqrt{3}+1)\cdot\sqrt{2}}{4},-1\right\}$

ʳ (Bogenmaß)

π Taste

Hinweis: Sie können dieses Sonderzeichen über die Tastatur Ihres Computers eingeben, indem Sie **@r** eintippen.

° (Grad)

π Taste

Ausdr1°⇒Ausdruck

Liste1°⇒Liste

Matrix1°⇒Matrix

Diese Funktion gibt Ihnen die Möglichkeit, im Neugrad- oder Bogenmaß-Modus einen Winkel in Grad anzugeben.

Im Winkelmodus Bogenmaß wird das Argument mit π/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:

$\cos(45°)$ → $\frac{\sqrt{2}}{2}$

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.

$\cos\left\{\left\{0,\frac{\pi}{4},90°,30.12°\right\}\right\}$ → $\{1.,0.707107,0.,0.864976\}$

°, ', " (Grad/Minute/Sekunde)

ctrl 🕮 Tasten

dd°mm'ss.ss"⇒Ausdruck

*dd*Eine positive oder negative Zahl

*mm*Eine nicht negative Zahl

*ss.ss*Eine 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.

Im Grad-Modus:

25°13'17.5"	25.2215
25°30'	$\frac{51}{2}$

°, ', " (Grad/Minute/Sekunde)

ctrl 🕮 Tasten

Hinweis: Nach ss.ss werden zwei Apostrophe ('') gesetzt, kein Anführungszeichen (").

∠ (Winkel)

ctrl 🕮 Tasten

[*Radius*,∠θ_*Winkel*]⇒*Vektor*

(Eingabe polar)

[*Radius*,∠θ_*Winkel*,*Z_Koordinate*]⇒*Vektor*

(Eingabe zylindrisch)

[*Radius*,∠θ_*Winkel*,∠θ_*Winkel*]⇒*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

$$[5 \angle 60° \angle 45°] \quad \left[\frac{5\cdot\sqrt{2}}{4} \quad \frac{5\cdot\sqrt{6}}{4} \quad \frac{5\cdot\sqrt{2}}{2}\right]$$

zylindrisch

$$[5 \angle 60° \angle 45°] \quad \left[\frac{5\cdot\sqrt{2}}{2} \quad \angle\frac{\pi}{3} \quad \frac{5\cdot\sqrt{2}}{2}\right]$$

sphärisch

$$[5 \angle 60° \angle 45°] \quad \left[5 \quad \angle\frac{\pi}{3} \quad \angle\frac{\pi}{4}\right]$$

(*Größe* ∠ *Winkel*)⇒*komplexerWert*

(Eingabe polar)

Dient zur Eingabe eines komplexen Werts in polarer (r∠θ) Form. Der *Winkel* wird gemäß der aktuellen Winkelmoduseinstellung interpretiert.

Im Winkelmodus Bogenmaß und Komplex-Formatmodus "kartesisch":

$$5+3\cdot i-\left(10 \angle \frac{\pi}{4}\right) \quad 5-5\cdot\sqrt{2}+\left(3-5\cdot\sqrt{2}\right)\cdot i$$

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 ≈ aus.

$$5+3\cdot i-\left(10 \angle \frac{\pi}{4}\right) \quad -2.07107-4.07107\cdot i$$

' (Ableitungsstrich)

?!▸ Taste

Variable '

Variable ''

Gibt in einer Differentialgleichung einen Ableitungsstrich ein. Ein Ableitungsstrich kennzeichnet eine Differentialgleichung erster Ordnung, zwei Ableitungsstriche kennzeichnen eine Differentialgleichung zweiter Ordnung usw.

$$\text{deSolve}\left(y''=y^{\frac{-1}{2}} \text{ and } y(0)=0 \text{ and } y'(0)=0, t, y\right)$$
$$\frac{2\cdot y^{\frac{3}{4}}}{3}=t$$

_ (Unterstrich als leeres Element)

Siehe “Leere (ungültige) Elemente” , Seite 259.

_ (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.

$3\cdot\_m\blacktriangleright\_ft$	$9.84252\cdot\_ft$

Hinweis: Das Umrechnungssymbol ▸ können Sie im Katalog finden. Klicken Sie auf $\int\Sigma$ und dann auf **Mathematische Operatoren**.

*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.

z sei undefiniert:

$\text{real}(z)$	z
$\text{real}(z\_)$	$\text{real}(z\_)$
$\text{imag}(z)$	0
$\text{imag}(z\_)$	$\text{imag}(z\_)$

Hinweis: Eine komplexe Zahl kann ohne

_ (Unterstrich als Einheiten-Bezeichner)

ctrl ␣ Tasten

Unterstrich _ in Variablen gespeichert werden. Bei Berechnungen wie **cSolve()** und **cZeros()** empfiehlt sich allerdings die Verwendung von _, um beste Ergebnisse zu erzielen.

▶ (konvertieren)

ctrl 🕮 Tasten

Ausdr_Einheit1 ▶ *_Einheit2*⇒*Ausdr_Einheit2*

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 ctrl ␣.

Hinweis: Verwenden Sie zum Konvertieren von Temperatureinheiten **tmpCnv()** und **ΔtmpCnv()**. Der Konvertierungsoperator ▶ ist nicht für Temperatureinheiten anwendbar.

$3 \cdot \_m \blacktriangleright \_ft$ $\quad 9.84252 \cdot \_ft$

10^()

Katalog > 🕮

10^(*Ausdr1***)**⇒*Ausdruck*

10^(*Liste1***)**⇒*Liste*

Gibt 10 hoch Argument zurück.

Bei einer Liste wird 10 hoch jedem Element von *Liste1* zurückgegeben.

$10^{1.5}$ $\quad 31.6228$

$10^{\{0,-2,2,a\}}$ $\quad \left\{1,\frac{1}{100},100,10^{a}\right\}$

10^()

Katalog >

10^(*Quadratmatrix1***)⇒***Quadratmatrix*

Ergibt 10 hoch *Quadratmatrix1*. Dies ist nicht gleichbedeutend mit der Berechnung von 10 hoch jedem Element. Näheres zur Berechnungsmethode finden Sie im Abschnitt **cos()**.

Quadratmatrix1 muss diagonalisierbar sein. Das Ergebnis enthält immer Fließkommazahlen.

$$10^{\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}}$$

$$\begin{bmatrix}1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6\\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6\\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6\end{bmatrix}$$

^-1(Kehrwert)

Katalog >

Ausdr1 **^-1**⇒*Ausdruck*

Liste1 **^-1**⇒*Liste*

Gibt den Kehrwert des Arguments zurück.

Bei einer Liste wird für jedes Element von *Liste1* der Kehrwert zurückgegeben.

Quadratmatrix1 **^-1**⇒*Quadratmatrix*

Gibt die Inverse von *Qudratmatrix1* zurück.

Quadratmatrix1 muss eine nicht-singuläre quadratische Matrix sein.

$$(3.1)^{-1} \qquad 0.322581$$

$$\{a,4,-0.1,x,-2\}^{-1} \qquad \left\{\frac{1}{a},\frac{1}{4},-10.,\frac{1}{x},\frac{-1}{2}\right\}$$

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ a & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}\frac{-2}{a-2} & \frac{1}{a-2}\\ \frac{a}{2\cdot(a-2)} & \frac{-1}{2\cdot(a-2)}\end{bmatrix}$$

| (womit-Operator)

ctrl Tasten

Ausdr **|** *BoolescherAusdr1* [**and***BoolescherAusdr2*]...

Ausdr **|** *BoolescherAusdr1* [**or***BoolescherAusdr2*]...

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.

$$x+1|x=3 \qquad 4$$

$$x+y|x=\sin(y) \qquad \sin(y)+y$$

$$x+y|\sin(y)=x \qquad x+y$$

| (womit-Operator)

ctrl Tasten

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.

$x^3-2\cdot x+7\rightarrow f(x)$ *Done*

$f(x)|x=\sqrt{3}$ $\sqrt{3}+7$

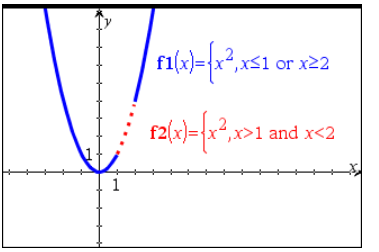
$(\sin(x))^2+2\cdot\sin(x)-6|\sin(x)=d$ $d^2+2\cdot d-6$

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.

$\text{solve}(x^2-1=0,x)|x>0 \text{ and } x<2$ $x=1$

$\sqrt{x}\cdot\sqrt{\frac{1}{x}}|x>0$ 1

$\sqrt{x}\cdot\sqrt{\frac{1}{x}}$ $\sqrt{\frac{1}{x}}\cdot\sqrt{x}$



Ausschließungen verwenden den relationalen Operator „ungleich“ (/= oder ≠), 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.

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

→ (speichern)

ctrl var **Taste**

Ausdr → *Var*

Liste → *Var*

Matrix → *Var*

Expr → *Funktion*(*Param1*,...)

List → *Funktion*(*Param1*,...)

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.

$\frac{\pi}{4} \rightarrow myvar$	$\frac{\pi}{4}$
$2\cdot\cos(x) \rightarrow y1(x)$	*Done*
$\{1,2,3,4\} \rightarrow lst5$	$\{1,2,3,4\}$
$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \rightarrow matg$	$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$
"Hello" → *str1*	"Hello"

:= (zuweisen)

ctrl Tasten

Var := *Ausdr*

Var := *Liste*

Var := *Matrix*

Function(*Param1*,...) := *Ausdr*

Function(*Param1*,...) := *Liste*

Function(*Param1*,...) := *Matrix*

Wenn Variable *Var* noch nicht existiert, wird *Var* erzeugt und auf *Ausdr*, *Liste* oder *Matrix* initialisiert.

$myvar := \frac{\pi}{4}$	$\frac{\pi}{4}$
$y1(x) := 2\cdot\cos(x)$	*Done*
$lst5 := \{1,2,3,4\}$	$\{1,2,3,4\}$
$matg := \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$	$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$
$str1 :=$ "Hello"	"Hello"

:= (zuweisen)

ctrl 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)

ctrl 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)=Func
    © Declare variables
    Local i,result
    result:=0
    For i,1,n,1 ©Loop n times
    result:=result+i^2
    EndFor
    Return result
    EndFunc
                                Done
g(3)                            14
```

0b, 0h

0 B Tasten, 0 H Tasten

0b *binäre_Zahl*

0h *hexadezimale_Zahl*

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 Dec-Modus:

```
0b10+0hF+10                     27
```

Im Bin-Modus:

```
0b10+0hF+10                     0b11011
```

Im Hex-Modus:

```
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 58, und **isVoid()**, Seite 103.

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 ␣.

Kalkulationen mit ungültigen Elementen

Bei der Mehrzahl aller Kalkulationen, die ein ungültiges Element enthalten, wird das Ergebnis ebenfalls ungültig sein. Sonderfälle sind nachstehend aufgeführt.

$\lvert \_ \rvert$	_
$\gcd(100,\_)$	_
$3+\_$	_
$\{5,\_,10\}-\{3,6,9\}$	$\{2,\_,1\}$

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
$\text{cumulativeSum}(\{1,2,\_,4,5\})$	$\{1,3,\_,7,12\}$
$\text{cumulativeSum}\left(\begin{bmatrix}1 & 2\\ 3 & \_\\ 5 & 6\end{bmatrix}\right)$	$\begin{bmatrix}1 & 2\\ 4 & \_\\ 9 & 8\end{bmatrix}$

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	$\{1,3,4,5,2\}$

$\{1,2,3,\_,5\} \rightarrow list1$	$\{1,2,3,\_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD *list1*,*list2*	*Done*
list1	$\{5,3,2,1,\_\}$
list2	$\{5,3,2,1,4\}$

In Regressionen sorgt ein ungültiges Element in einer Liste X oder Y dafür, dass auch das entsprechende Element im Residuum ungültig ist.

$l1:=\{1,2,3,4,5\}$: $l2:=\{2,\_,3,5,6.6\}$	$\{2,\_,3,5,6.6\}$
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.

$l1:=\{1,3,4,5\}$: $l2:=\{2,3,5,6.6\}$	$\{2,3,5,6.6\}$
$cat:=\{\text{"M","M","F","F"}\}$: $incl:=\{\text{"F"}\}$	$\{\text{"F"}\}$
LinRegMx *l1*,*l2*,1,*cat*,*incl*	*Done*
stat.Resid	$\{\_,\_,0.,0.\}$
stat.XReg	$\{\_,\_,4.,5.\}$
stat.YReg	$\{\_,\_,5.,6.6\}$
stat.FreqReg	$\{\_,\_,1.,1.\}$

Eine Häufigkeit von 0 in Regressionen führt dazu, dass das entsprechende Element im Residuum ungültig ist.

$l1:=\{1,3,4,5\}$: $l2:=\{2,3,5,6.6\}$	$\{2,3,5,6.6\}$
LinRegMx *l1*,*l2*,$\{1,0,1,1\}$	*Done*
stat.Resid	$\{0.069231,\_,-0.276923,0.207692\}$
stat.XReg	$\{1.,\_,4.,5.\}$
stat.YReg	$\{2.,\_,5.,6.6\}$
stat.FreqReg	$\{1.,\_,1.,1.\}$

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(...)**
$\int$()	**integral(...)**
Σ() (Vorlage Summe)	**sumSeq(...)**
Π() (Vorlage Produkt)	**prodSeq(...)**
$\sin^{-1}$(), $\cos^{-1}$(), ...	**arcsin(...), arccos(...)**, ...
ΔListe()	**deltaList(...)**
ΔtmpCnv()	**deltaTmpCnv(...)**

Von der Computertastatur

Sonderzeichen:	Tastenkürzel:
c1, *c2*, ... (Konstanten)	**@c1**, **@c2**, ...
n1, *n2*, ... (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)	**@<**
▶ (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 (°,',"), Fakultät (!), Prozent (%), Bogenmaß (^{r}), Tiefstellen ([ ]), Transponieren (^{T})
5	Potenzieren, Potenzoperator (^)
6	Negation ($^-$)
7	Stringverkettung (&)
8	Multiplikation (•), Division (/)
9	Addition (+), Subtraktion (-)
10	Gleichheitsbeziehungen: gleich (=), ungleich (≠ oder /=), kleiner als (<), kleiner oder gleich (≤ oder <=), größer als (>), größer oder gleich (≥ oder >=)
11	Logisches Nicht: **not**
12	Logische Konjunktion: **and**
13	Logisch **or**
14	**xor**, **nor**, **nand**
15	logische Implikation, (⇒)
16	Logische doppelte Implikation, XNOR (⇔)
17	womit-Operator („| “)
18	Speichern (→)

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. Anderenfalls 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→r und “r”→s1, dann #s1=10.

Postfix-Operatoren

Postfix-Operatoren sind Operatoren, die direkt nach einem Argument stehen, zum Beispiel 5!, 25% oder 60°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 −x2 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 209).

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: • 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	Variablenname doppelt vergeben
280	Else und ElseIf 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: • sto → • := • **Definiere**

Fehlercode	Beschreibung
	um Variablen Werte zuzuweisen.
965	Betriebssystem nicht lizensiert
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: • **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: • 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 • 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: • 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: • 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	Bibliothaksvariable nicht gefunden: • Vergewissern Sie sich, dass sich die Bibliotheksvariable im ersten Problem in der Bibliothek befindet. • Überprüfen Sie, dass die Bibliothaksvariable als LibPub oder LibPriv definiert wurde.

Fehlercode	Beschreibung
	• Aktualisieren Sie die Bibliotheken. Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation
1210	Unzulässiger Name für Bibliothekskurzform. Vergewissern Sie sich, dass der Name: • keinen Punkt enthält • nicht mit einem Unterstrich beginnt • nicht länger ist als 16 Zeichen • nicht reserviert ist Weitere Einzelheiten finden Sie im Abschnitt Bibliotheken der Dokumentation.
1220	Bereichsfehler: Die Funktionen tangentLine und normalLine unterstützen nur Funktionen mit reellen Werten.
1230	Bereichsfehler. Im Grad- und Neugradmodus werden die trigonometrischen Konversionsoperatoren nicht unterstützt.
1250	Argumentfehler System linearer Gleichungen verwenden. Beispiel für ein System zweier linearer Gleichungen mit den Variablen x und y: 3x+7y=5 2y-5x=-1
1260	Argumentfehler: 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.
1270	Argumentfehler Ordnung der Ableitung muss gleich 1 oder 2 sein.
1280	Argumentfehler Verwenden Sie ein Polynom in entwickelter Form in einer Variablen.
1290	Argumentfehler Verwenden Sie ein Polynom in einer Variablen.
1300	Argumentfehler Die Koeffizienten des Polynoms müssen numerische Werte ergeben.

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 218).

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(): • 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	^0 oder undef^0 durch 1 ersetzt
10014	undef^0 durch 1 ersetzt
10015	1^ oder 1^undef durch 1 ersetzt
10016	1^undef durch 1 ersetzt
10017	Überlauf durch ∞ oder ⁻∞ ρσ
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 231

!

!, Fakultät 242

"

", Sekunden-Schreibweise 251

#

#, Umleitung 249
#, Umleitungsoperator 264

%

%, Prozent 237

*

*, multiplizieren 232

.

.-, Punkt-Subtraktion 236
.*, Punkt-Multiplikation 236
./, Punkt-Division 236
.^, Punkt-Potenz 237
.+, Punkt-Addition 235

/

/, dividieren 233

:

:=, zuweisen 257

^

^⁻¹, Kehrwert 255
^, Potenz 234

_

_, Einheitenbezeichnung 253

|

|, womit-Operator 255

'

', Ableitungsstrich 253
', Minuten-Schreibweise 251

+

+, addieren 231

<

<, kleiner als 239

=

=, gleich 238
≠, ungleich[*] 238

>

>, größer als 240

∏

∏, Produkt 246

∑

∑( ), Summe 246
∑Int( ) 247
∑Prn( ) 248

√

√( ), Quadratwurzel 245

∠

∠, winkel 252

∫

∫, Integral 244

≤

≤, kleiner oder gleich 239

≥

≥, größer oder gleich 240

▸

▸, Einheiten konvertieren[*] 254
▸, in Neugrad umwandeln 95
▸approxFraction( ) 18
▸Base10, Anzeige als ganze Dezimalzahl[Base10] 24
▸Base16, Hexadezimaldarstellung[Base16] 24
▸Base2, Binärdarstellung[Base2] 22
▸cos, durch Kosinus ausdrücken[cos] 36
▸Cylind, Anzeige als Zylindervektor[Cylind (Zylindervektor)] 51
▸DD, Anzeige als Dezimalwinkel[DD (Dezimalwinkel)] 54
▸Decimal, Anzeige als Dezimalzahl[Decimal] 55
▸DMS, Anzeige als Grad/Minute/Sekunde[DMS (GMS)] 63
▸exp, ausdrücken durch e[exp] 73
▸Polar, Anzeige als Polarvektor[Polar] 145
▸Rad, in Bogenmaß umwandeln 157
▸Rect, Anzeige als kartesischer Vektor[Rect (Kartesisch)] 160
▸sin, durch Sinus ausdrücken[sin] 181
▸Sphere, Anzeige als sphärischer Vektor[Sphere (Kugelkoordinaten)] 191
▸tmpCnv() (Konvertierung von Temperaturbereichen)[tmpCnv] 207

⇒

⇒, logische Implikation 241, 261

→

→, speichern 257

⇔

⇔, logische doppelte Implikation[*] 242

©

©, Kommentar 258

°

°, Grad-Schreibweise 251
°, Grad/Minute/Sekunde 251

0

0b, binäre Anzeige 258
0h, hexadezimale Anzeige 258

1

10^( ), Potenz von zehn 254

A

Abbruch, Exit 72
Ableitung oder n-te Ableitung
  Vorlage für 10
Ableitungen
  erste Ableitung, d( ) 243
  numerische Ableitung, nDeriv( ) 133
  numerische Ableitung, nDerivative( ) 132
Ableitungsstrich, ... 253
Ableitungsstrich, ', ... 253
abrufen/zurückgeben
  Variableninformationen, getVarInfo( ) 90
Abrufen/zurückgeben
  Variableninformationen, getVarInfo( ) 94
abs( ), Absolutwert 12
Absolutwert
  Vorlage für 7-8
addieren, + ... 231
als kartesischen Vektor anzeigen, ▸Rect 160
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 172
Anzeige als
  binär, ▸Base2 22

Dezimalwinkel, ▸DD 54
ganze Dezimalzahl, ▸Base10 24
Grad/Minute/Sekunde, ▸DMS 63
hexadezimal, ▸Base16 24
kartesischer Vektor, ▸Rect 160
Polarvektor, ▸Polar 145
sphärischer Vektor, ▸Sphere 191
Zylindervektor, ▸Cylind 51
Anzeige als sphärischer Vektor, ▸Sphere 191
Anzeige als Zylindervektor, ▸Cylind 51
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 213
Arkuskosinus, $\cos^{-1}$( ) 39
Arkussinus, $\sin^{-1}$( ) 183
Arkustangens, $\tan^{-1}$( ) 200
augment( ), erweitern/verketten 20
Ausdrücke
Ausdruck in Liste, exp▸list( ) 74
String in Ausdruck, expr( ) 76, 117
Ausschließung mit „|“ Operator 255
Auswertungsreihenfolge 263
avgRC( ), durchschnittliche Änderungsrate 21

B

Befehl Stopp 196
benutzerdefinierte Funktionen 55
benutzerdefinierte Funktionen und Programme 56-57

Bestimmtes Integral
Vorlage für ... 10
Bibliothek
erstelle Tastaturbefehle für Objekte ... 105
binär
Anzeige, 0b ... 258
Darstellung, ▸Base2 ... 22
binomCdf( ) ... 25
binomPdf( ) ... 25
Bogenlänge, arcLen( ) ... 19
Bogenmaß, r ... 250
Boolean operators
and ... 13
Boolesch
und, and ... 13
Boolesche Operatoren
⇒ ... 241, 261
⇔ ... 242
nand ... 130
nicht ... 137
nor ... 135
oder ... 142
xor ... 220
Brüche
propFrac (Echter Bruch) ... 152
Vorlage für ... 5

C

Cdf( ) ... 79
ceiling( ), Obergrenze ... 26
centralDiff( ) ... 26
cFactor( ), komplexer Faktor ... 27
char( ), Zeichenstring ... 28
charPoly( ) ... 28
$\chi^2$2way ... 29
ClearAZ ... 31
colAugment ... 32
colDim( ), Spaltendimension der Matrix ... 32
colNorm( ), Spaltennorm der Matrix ... 32
comDenom( ), gemeinsamer Nenner ... 33
completeSquare( ), complete square ... 34
conj( ), Komplex Konjugierte ... 35
constructMat( ), Matrix erstellen ... 35

corrMat( ), Korrelationsmatrix 36
cos⁻¹, Arkuskosinus 39
cos( ), Kosinus 37
cosh⁻¹( ), Arkuskosinus hyperbolicus 40
cosh( ), Cosinus hyperbolicus 39
cot⁻¹( ), Arkuskotangens 41
cot( ), Kotangens 40
coth⁻¹( ), Arkuskotangens hyperbolicus 42
coth( ), Kotangens hyperbolicus 41
countIf( ), Elemente in einer Liste bedingt zählen 42
cPolyRoots() 43
crossP( ), Kreuzprodukt 44
csc⁻¹( ), inverser Kosekans 44
csc( ), Kosekans 44
csch⁻¹( ), inverser Kosekans hyperbolicus 45
csch( ), Kosekans hyperbolicus 45
cSolve( ), komplexe Lösung 45
CubicReg, kubische Regression 49
Cycle, Zyklus 50
cZeros( ), komplexe Nullstellen 51

D

d( ), erste Ableitung 243
Daten anzeigen, Anz 172
Daten anzeigen, Disp 63
dbd( ), Tage zwischen Daten 54
Define, definiere 55
Definiere 55
Definiere LibPriv (Define LibPriv) 56
Definiere LibPub (Define LibPub) 57
Definiere, Define 55
definieren
    öffentliche Funktion / öffentliches Programm 57
    private Funktion oder Programm 56
Definitionsbereichsfunktion, domain( ) 64
deltaList() 57
deltaTmpCnv() 57
DelVar, Variable löschen 58
delVoid( ), ungültige Elemente entfernen 58
derivative() 58
deSolve( ), Lösung 59
det( ), Matrixdeterminante 61

Dezimal
Anzeige als ganze Zahl, ▸Base10 24
Winkelanzeige, ▸DD 54
diag( ), Matrixdiagonale 62
Diagonalform, ref( ) 161
dim( ), Dimension 62
Dimension, dim( ) 62
dividieren, / 233
domain( ), Definitionsbereichsfunktion 64
dominant term, dominantTerm( ) 64
dominantTerm( ), dominant term 64
dotP( ), Skalarprodukt 66
drehe( ) 167
drehen, drehe( ) 167
durchschnittliche Änderungsrate, avgRC( ) 21

E

e Exponent
Vorlage für 6
e hoch x, e^( ) 66, 73
e, ausdrücken durch 73
E, Exponent 249
e^( ), e hoch x 66
echter Bruch, propFrac 152
eff( ), Nominal- in Effektivsatz konvertieren 67
Effektivsatz, eff( ) 67
Eigenvektor, eigVc( ) 67
Eigenwert, eigVl( ) 68
eigVc( ), Eigenvektor 67
eigVl( ), Eigenwert 68
Eingabe, Input 98
Einheiten
konvertieren 254
Einheitsmatrix, identity( ) 95
Einheitsvektor, unitV( ) 216
Einstellungen, hole aktuellen 92
Elemente in einer Liste bedingt zählen, countIf( ) 42
Elemente in einer Liste zählen, zähle( ) 42
else if, ElseIf 68
else, Else 96
ElseIf, else if 68
end
for, EndFor 83

if, EndIf ... 96
Schleife, EndLoop ... 120
while, EndWhile ... 220
end if, EndIf ... 96
end while, EndWhile ... 220
Ende
Funktion, EndFunc ... 87
Ende der Schleife, EndLoop ... 120
EndWhile, end while ... 220
Entfernen
ungültige Elemente aus Liste ... 58
Entwickle, expand( ) ... 74
EOS (Equation Operating System) ... 263
Equation Operating System (EOS) ... 263
Ergebnis
ausdrücken durch e ... 73
durch Kosinus ausdrücken ... 36
durch Sinus ausdrücken ... 181
Ergebnisse mit zwei Variablen, TwoVar ... 214
Ergebnisse, Statistik ... 193
Ergebniswerte, Statistik ... 194
Ersetzung durch „|“ Operator ... 255
erste Ableitung
Vorlage für ... 9
erweitern/verketten, augment( ) ... 20
euler( ), Euler function ... 69
exact( ), Exakt ... 72
Exakt, exact( ) ... 72
Exit, Abbruch ... 72
exp( ), e hoch x ... 73
exp▸list( ), Ausdruck in Liste ... 74
expand( ), Entwickle ... 74
Exponent, E ... 249
Exponenten
Vorlage für ... 5
Exponentielle Regression, ExpReg ... 76
expr( ), String in Ausdruck ... 76, 117
ExpReg, exponentielle Regression ... 76

F

factor( ), Faktorisiere ... 77
Faktorisiere, factor( ) ... 77
Fakultät, ! ... 242

Fehler übergeben, ÜbgebFeh 144
Fehler und Fehlerbehebung
Fehler löschen, LöFehler 31
Fehler übergeben, ÜbgebFeh 144
festlegen
Modus, setMode( ) 176
Fill, Matrix füllen 80
Finanzfunktionen, tvmFV( ) 212
Finanzfunktionen, tvmI( ) 212
Finanzfunktionen, tvmN( ) 212
Finanzfunktionen, tvmPmt( ) 213
Finanzfunktionen, tvmPV( ) 213
FiveNumSummary 80
floor( ), Untergrenze 81
fMax( ), Funktionsmaximum 81
fMin( ), Funktionsminimum 82
Folge, seq( ) 173
Folge, series( ) 175
For 83
for, For 83
For, for 83
format( ), Formatstring 83
Formatstring, format( ) 83
fpart( ), Funktionsteil 84
freqTable( ) 85
Frobeniusnorm, norm( ) 136
Func, Funktion 87
Func, Programmfunktion 87
Funktion beenden, EndFunc 87
Funktionen
benutzerdefiniert 55
Maximum, fMax( ) 81
Minimum, fMin( ) 82
Programmfunktion, Func 87
Teil, fpart( ) 84
Funktionen und Variablen
kopieren 36

G

g, Neugrad 250
ganze Zahl, int( ) 99
Ganzzahl teilen, intDiv( ) 99
ganzzahliger Teil, iPart( ) 102

gcd( ), größter gemeinsamer Teiler 88
gehe zu, Goto 95
gemeinsamer Nenner, comDenom( ) 33
geomCdf( ) 88
geomPdf( ) 89
Get 89
getDenom( ), Nenner holen/zurückgeben 90
getLangInfo( ), Sprachinformationen abrufen/zurückgeben 90
getLockInfo( ), testet den Gesperrt-Status einer Variablen oder Variablengruppe 91
getMode( ), getMode-Einstellungen 92
getNum( ), Zähler holen/zurückgeben 93
GetStr 93
getType( ), get type of variable 93
getVarInfo( ), Variableninformationen abrufen/zurückgeben 94
gleich, = ... 238
Gleichungssystem (2 Gleichungen)
Vorlage für 7
Gleichungssystem (n Gleichungen)
Vorlage für 7
Gleichungssystem, simult( ) 180
Goto, gehe zu 95
Grad-/Minuten-/Sekundenanzeige, ▸DMS 63
Grad-Schreibweise, ° 251
größer als, > 240
Größer oder gleich, ≥ 240
größter gemeinsamer Teiler, gcd( ) 88
Gruppen, Gesperrt-Status testen 91
Gruppen, sperren und entsperren 116, 216

H

Häufigkeit( ) 86
hexadezimal
Anzeige, ▸Base16 24
Anzeige, 0h 258
holen/zurückgeben
Nenner, getDenom( ) 90
Zähler, getNum( ) 93
Hyperbolisch
Arkuskosinus, $\cosh^{-1}$( ) 40
Arkussinus, $\sinh^{-1}$( ) 184
Arkustangens, $\tanh^{-1}$( ) 202
Cosinus, cosh( ) 39
Sinus, sinh( ) 184

Tangens, tanh( ) 201

I

identity( ), Einheitsmatrix 95
if, If 96
If, if 96
ifFn( ) 97
imag( ), Imaginärteil 98
Imaginärteil, imag( ) 98
ImpDif( ), implizite Ableitung 98
implizite Ableitung, Impdif( ) 98
in String, inString( ) 99
Input, Eingabe 98
inString( ), in String 99
int( ), ganze Zahl 99
intDiv( ), Ganzzahl teilen 99
Integral, ∫ 244
interpolate( ), interpolate 100
inverse kumulative Normalverteilung (invNorm( )) 101
invF( ) 101
invNorm( ), inverse kumulative Normalverteilung 101
invt( ) 101
Invχ^2( ) 101
iPart( ), ganzzahliger Teil 102
irr( ), interner Zinsfluss
  interner Zinsfluss, irr( ) 102
isPrime( ), Primzahltest 102
isVoid( ), Test auf Ungültigkeit 103

K

kartesische x-Koordinate, P▸Rx( ) 143
kartesische y-Koordinate, P▸Ry( ) 144
Kehrwert, ^$^{-1}$ 255
kleiner als, < 239
Kleiner oder gleich, ≤ 239
kleinstes gemeinsames Vielfaches, lcm 104
Kombinationen, nCr( ) 131
Kommentar, © 258
komplex
  Faktor, cFactor( ) 27
  Konjugierte, conj( ) 35
  Lösung, cSolve( ) 45

Nullstellen, cZeros( ) 51
Konstante
in solve( ) 188
Konstanten
in cSolve( ) 48
in cZeros( ) 53
in deSolve( ) 59
in solve( ) 190
Tastenkürzel für 261
konvertieren
Einheiten 254
Korrelationsmatrix, corrMat( ) 36
Kosinus / Cos
ausdrücken durch 36
Kosinus, cos( ) 37
Kotangens, cot( ) 40
Kreuzprodukt, crossP( ) 44
kubische Regression, CubicReg 49
kumulierte Summe, cumulativeSum( ) 50
kumulierteSumme( ), kumulierte Summe 50

L

Lbl, Marke 104
lcm, kleinstes gemeinsames Vielfaches 104
leere (ungültige) Elemente 259
left( ), links 104
LibPriv 56
LibPub 57
libShortcut( ), erstelle Tastaturbefehle für Bibliotheksobjekte 105
Limes
lim( ) (Limes) 106
limit( ) (Limes) 106
Vorlage für 11
limit( ) oder lim( ), Limes 106
lineare Regression, LinRegAx 108
lineare Regression, LinRegBx 107
Lineare Regression, LinRegBx 109
links, left( ) 104
LinRegBx, lineare Regression 107
LinRegMx, lineare Regression 108
LinRegtIntervals, lineare Regression 109
LinRegtTest 111
linSolve() 112

list▸mat( ), Liste in Matrix 113
Liste in Matrix, list▸mat( ) 113
Liste, Elemente bedingt zählen 42
Liste, Elemente zählen in 42
Listen
Ausdruck in Liste, exp▸list( ) 74
Differenzen in einer Liste, Δlist( ) 113
erweitern/verketten, augment( ) 20
in absteigender Reihenfolge sortieren, SortD 191
in aufsteigender Reihenfolge sortieren, SortA 191
Kreuzprodukt, crossP( ) 44
kumulierte Summe, cumulativeSum( ) 50
leere Elemente in 259
Liste in Matrix, list▸mat( ) 113
Matrix in Liste, mat▸list( ) 121
Maximum, max( ) 122
Minimum, min( ) 125
neu, newList( ) 132
Produkt, product( ) 151
Skalarprodukt, dotP( ) 66
Summe, sum( ) 197
Summierung, sum( ) 198
Teil-String, mid( ) 125
ln( ), natürlicher Logarithmus 114
LnReg, logarithmische Regression 114
Local, lokale Variable 116
Lock, Variable oder Variablengruppe sperren 116
LöFehler, Fehler löschen 31
Logarithmen 114
Logarithmische Regression, LnReg 114
Logarithmus
Vorlage für 6
logische doppelte Implikation, ⇔ 242
logische Implikation, ⇒ 241, 261
Logistic, logistische Regression 118
LogisticD, logistische Regression 119
Logistische Regression, Logistic 118
Logistische Regression, LogisticD 119
lokal, Local 116
lokale Variable, Local 116
Loop, Schleife 120
löschen
Variable, DelVar 58

Löschen
Fehler, LöFehler ... 31
ungültige Elemente aus Liste ... 58
Löse, solve( ) ... 186
Lösung, deSolve( ) ... 59
LU, untere/obere Matrixzerlegung ... 121

M

Marke, Lbl ... 104
mat▸list( ), Matrix in Liste ... 121
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()( ) ... 35
Matrix in Liste, mat▸list( ) ... 121
Matrizen
Determinante, det( ) ... 61
Diagonale, diag( ) ... 62
Diagonalform, ref( ) ... 161
Dimension, dim( ) ... 62
Eigenvektor, eigVc( ) ... 67
Eigenwert, eigVl( ) ... 68
Einheitsmatrix, identity( ) ... 95
erweitern/verketten, augment( ) ... 20
füllen, Fill ... 80
kumulierte Summe, cumulativeSum( ) ... 50
Liste in Matrix, list▸mat( ) ... 113
Matrix in Liste, mat▸list( ) ... 121
Matrixzeilenmultiplikation und -addition, mRowAdd( ) ... 127
Maximum, max( ) ... 122
Minimum, min( ) ... 125
neu, newMat( ) ... 133
Produkt, product( ) ... 151
Punkt-Addition, .+ ... 235
Punkt-Division, ./ ... 236
Punkt-Multiplikation, .* ... 236
Punkt-Potenz, .^ ... 237
Punkt-Subtraktion, .- ... 236

QR-Faktorisierung, QR 153
reduzierte Diagonalform, rref( ) 170
Spaltendimension, colDim( ) 32
Spaltennorm, colNorm( ) 32
Summe, sum( ) 197
Summierung, sum( ) 198
Transponierte, T 199
untere/obere Matrixzerlegung, LU 121
Untermatrix, subMat( ) 197, 199
Zeilenaddition, rowAdd( ) 169
Zeilendimension, rowDim( ) 170
Zeilennorm, rowNorm( ) 170
Zeilenoperation, mRow( ) 127
Zeilentausch, rowSwap( ) 170
Zufall, randMat( ) 158
max( ), Maximum 122
Maximum, max( ) 122
mean( ), Mittelwert 122
median( ), Median 123
Median, median( ) 123
MedMed, Mittellinienregression 123
mid( ), Teil-String 125
min( ), Minimum 125
Minimum, min( ) 125
Minuten-Schreibweise, ... 251
mirr( ), modifizierter interner Zinsfluss 126
mit, | 255
Mittellinienregression, MedMed 123
Mittelwert, mean( ) 122
mod( ), Modulo 127
Modi
festlegen, setMode( ) 176
Modifizierter interner Zinsfluss, mirr( ) 126
Modulo, mod( ) 127
Moduseinstellungen, getMode( ) 92
mRow( ), Matrixzeilenoperation 127
mRowAdd( ), Matrixzeilenmultiplikation und -addition 127
Multipler linearer Regressions-t-Test 129
multiplizieren, * 232
MultReg (Mehrfachregression) 127
MultRegIntervals( ) (Mehrfachregressionsintervall) 128
MultRegTests( ) 129

N

n-te Wurzel
- Vorlage für ... 6

nand, Boolescher Operator ... 130
natürlicher Logarithmus, ln( ) ... 114
nCr( ), Kombinationen ... 131
nDerivative( ), numerische Ableitung ... 132
Negation, Eingabe von negativen Zahlen ... 264
Nenner ... 33
Nettobarwert, npv ( ) ... 139
neu
- Liste, newList( ) ... 132
- Matrix, newMat( ) ... 133

Neugrad-Schreibweise, g ... 250
newList( ), neue Liste ... 132
newMat( ), neue Matrix ... 133
nfMax( ), numerisches Funktionsmaximum ... 133
nfMin( ), numerisches Funktionsminimum ... 133
nicht, Boolescher Operator ... 137
nInt( ), numerisches Integral ... 134
nom), Effektivzins in Nominalzins konvertieren ... 134
Nominalzinssatz, nom( ) ... 134
nor, Boolescher Operator ... 135
norm( ), Frobeniusnorm ... 136
Normale, normalLine( ) ... 136
normalLine( ) ... 136
Normalverteilungswahrscheinlichkeit, normCdf( ) ... 136
normCdf( ) (Normalverteilungswahrscheinlichkeit) ... 136
normPdf( ) (Wahrscheinlichkeitsdichte) ... 137
nPr( ), Permutationen ... 138
npv( ), Nettobarwert ... 139
nSolve( ), numerische Lösung ... 139
Nullstellen, zeroes( ) ... 221
numerisch
- Ableitung, nDeriv( ) ... 133
- Ableitung, nDerivative( ) ... 132
- Integral, nInt( ) ... 134
- Lösung, nSolve( ) ... 139

O

Obergrenze, ceiling( ) ... 26, 43

Objekte
erstelle Tastaturbefehle für Bibliothek 105
oder (Boolesch), oder 142
oder, Boolescher Operator 142
OneVar, Statistik mit einer Variable 140
Operatoren
Auswertungsreihenfolge 263
ord( ), numerischer Zeichencode 143

P

P▸Rx( ), kartesische x-Koordinate 143
P▸Ry( ), kartesische y-Koordinate 144
Pdf( ) 85
Permutationen, nPr( ) 138
piecewise( ) (Stückweise) 145
poissCdf( ) 145
poissPdf( ) 145
polar
Koordinate, R▸Pr( ) 156
Koordinate, R▸Pθ( ) 156
Vektoranzeige, ▸Polar 145
polyCoef( ) 146
polyDegree( ) 147
polyEval( ), Polynom auswerten 148
polyGcd( ) 148-149
Polynom auswerten, polyEval( ) 148
Polynome
auswerten, polyEval( ) 148
Zufall, randPoly( ) 159
PolyRoots() 149
Potenz von zehn, 10^( ) 254
Potenz, ^ 234
Potenzregression, PowerReg 149, 162, 164, 204
PowerReg, Potenzregression 149
Prgm, Definiere Programm 151
Primzahltest, isPrime( ) 102
prodSeq() 151
product( ), Produkt 151
Produkt ∏( )
Vorlage für 9
Produkt, ∏( ) 246
Produkt, product( ) 151

Programme
öffentliche Bibliothek definieren 57
Private Bibliothek definieren 56
Programme und Programmieren
E/A-Bildschirm anzeigen, Anz 172
E/A-Bildschirm anzeigen, Zeige 63
Fehler löschen, LöFehler 31
programmieren
Daten anzeigen, Disp 63
Definiere Programm, Prgm 151
Fehler übergeben, ÜbgebFeh 144
Programmierung
Daten anzeigen, Anz 172
propFrac, echter Bruch 152
Prozent, % 237
Punkt
Addition, .+ 235
Division, ./ 236
Multiplikation, .* 236
Potenz, .^ 237
Subtraktion, .- 236

Q

QR-Faktorisierung, QR 153
QR,QR-Faktorisierung 153
Quadratische Regression, QuadReg 154
Quadratwurzel
Vorlage für 5
Quadratwurzel, √( ) 245
Quadratwurzel, ‡( ) 192
QuadReg, quadratische Regression 154
QuartReg, Regression vierter Ordnung 155

R

r, Bogenmaß 250
R▸Pr( ), Polarkoordinate 156
R▸Pθ( ), Polarkoordinate 156
rand( ), Zufallszahl 157
randBin, Zufallszahl 157
randInt( ), ganzzahlige Zufallszahl 158
randMat( ), Zufallsmatrix 158
randNorm( ), Zufallsnorm 158
randPoly( ), Zufallspolynom 159

randSamp( ) (Zufallsstichprobe) 159
RandSeed, Zufallszahl 159
real( ), reell 159
rechts, right( ) 165
reduzierte Diagonalform, rref( ) 170
reell, real( ) 159
ref( ), Diagonalform 161
Regression vierter Ordnung, QuartReg 155
Regressionen
exponentielle, ExpReg 76
kubische, CubicReg 49
lineare Regression, LinRegAx 108
lineare Regression, LinRegBx 107
Lineare Regression, LinRegBx 109
logarithmische, LnReg 114
Logistic (Logistisch) 118
logistische, Logistic 119
Mittellinie, MedMed 123
MultReg (Mehrfachregression) 127
Potenzregression, PowerReg 149, 162, 164, 204
quadratische, QuadReg 154
sinusförmige, SinReg 185
vierter Ordnung, QuartReg 155
remain( ), Rest 162
Request 162
RequestStr 164
Rest, remain( ) 162
Return, Rückgabe 165
right( ), rechts 165
right, right( ) 34, 69, 100, 166, 218
rk23( ), Runge Kutta function 166
rotate( ), rotieren 168
rotieren, rotate( ) 168
round( ), runden 169
rowAdd( ), Matrixzeilenaddition 169
rowDim( ), Zeilendimension der Matrix 170
rowNorm( ), Zeilennorm der Matrix 170
rowSwap( ), Matrixzeilentausch 170
rref( ), reduzierte Diagonalform 170
Rückgabe, Return 165
runden, round( ) 169

S

Schleife, Loop 120
Schreibweise Grad/Minute/Sekunde 251
sec^{-1}( ), Arkussekans 171
sec( ), Sekans 171
sech^{-1}( ), Arkussekans hyperbolicus 172
sech( ), Sekans hyperbolicus 172
Sekunden-Schreibweise, " 251
seq( ), Folge 173
seqGen( ) 173
seqn( ) 174
sequence, seq( ) 173-174
series( ), Folge 175
setMode( ), Modus festlegen 176
shift( ), verschieben 178
sign( ), Zeichen 180
simult( ), Gleichungssystem 180
sin^{-1}( ), Arkussinus 183
sin( ), Sinus 182
sinh^{-1}( ), Arkussinus hyperbolicus 184
sinh( ), Sinus hyperbolicus 184
SinReg, sinusförmige Regression 185
Sinus
  ausdrücken durch 181
Sinus, sin( ) 182
Sinusförmige Regression, SinReg 185
Skalar
  Produkt, dotP( ) 66
solve( ), Löse 186
SortA, in aufsteigender Reihenfolge sortieren 191
SortD, in absteigender Reihenfolge sortieren 191
sortieren
  in absteigender Reihenfolge sortieren, SortD 191
  in aufsteigender Reihenfolge, SortA 191
speichern
  Symbol, → 257
Sprache
  Sprachinformation abrufen 90
sqrt( ), Quadratwurzel 192
Standardabweichung, stdDev( ) 194-195, 216
stat.results 193
stat.values 194

Statistik
Ergebnisse mit zwei Variablen, TwoVar ... 214
Fakultät, ! ... 242
Kombinationen, nCr( ) ... 131
Median, median( ) ... 123
Mittelwert, mean( ) ... 122
Permutationen, nPr( ) ... 138
Standardabweichung, stdDev( ) ... 194-195, 216
Statistik mit einer Variable, OneVar ... 140
Varianz, variance( ) ... 217
Zufallsnorm, randNorm( ) ... 158
Zufallszahl, RandSeed ... 159
Statistik mit einer Variable, OneVar ... 140
stdDevPop( ), Populations-Standardabweichung ... 194
stdDevSamp( ), Stichproben-Standardabweichung ... 195
String
Dimension, dim( ) ... 62
Länge ... 62
string( ), Ausdruck in String ... 196
Stringlänge ... 62
strings
right, right( ) ... 34, 69, 100, 166, 218
Strings
Ausdruck in String, string( ) ... 196
Format, format( ) ... 83
Formatieren ... 83
in, InString ... 99
links, left( ) ... 104
rechts, right( ) ... 165
rotieren, rotate( ) ... 168
String in Ausdruck, expr( ) ... 76, 117
Teil-String, mid( ) ... 125
Umleitung, # ... 249
verschieben, shift( ) ... 178
Zeichencode, ord( ) ... 143
Zeichenstring, char( ) ... 28
Stückweise definierte Funktion (2 Teile)
Vorlage für ... 6
Stückweise definierte Funktion (n Teile)
Vorlage für ... 7
Student-t-Wahrscheinlichkeitsdichte, tPdf( ) ... 208
subMat( ), Untermatrix ... 197, 199
subtrahieren, - ... 231

sum( ), Summe 197
sumIf( ) 198
Summe ∑( )
Vorlage für 9
Summe der Tilgungszahlungen 248
Summe der Zinszahlungen 247
Summe, ∑( ) 246
Summe, sum( ) 197
sumSeq() 198

T

t test, t-Test 210
T, Transponierte 199
Tage zwischen Daten, dbd( ) 54
$\tan^{-1}$( ), Arkustangens 200
tan( ), Tangens 199
Tangens, tan( ) 199
Tangente, tangentLine( ) 201
tangentLine( ) 201
$\tanh^{-1}$( ), Arkustangens hyperbolicus 202
tanh( ), Tangens hyperbolicus 201
Tastenkürzel 261
Tastenkürzel, Tastatur 261
Taylor-Polynom, taylor( ) 202
taylor( ), Taylor-Polynom 202
tCdf(), Wahrscheinlichkeit einer Student t-Verteilung 203
tCollect( ), trigonometrische Zusammenfassung 203
Teil-String, mid( ) 125
Test auf Ungültigkeit, isVoid( ) 103
Test_2S, Zwei-Stichproben F-Test 86
tExpand( ), trigonometrische Entwicklung 204
Text, Befehl 204
tInterval, Konfidenzintervall t 205
tInterval_2Samp, ZweiStichproben-t-Konfidenzintervall 206
tmpCnv() (Konvertierung von Temperaturwerten) 207
trace( ) 209
Transponierte, T 199
trigonometrische Entwicklung, tExpand( ) 204
trigonometrische Zusammenfassung, tCollect( ) 203
Try, Befehl zur Fehlerbehandlung 209
tTest, t-Test 210
tTest_2Samp, Zwei-Stichproben-t-Test 211

TVM-Argumente 213
tvmFV( ) 212
tvmI( ) 212
tvmN( ) 212
tvmPmt( ) 213
tvmPV( ) 213
TwoVar, Ergebnisse mit zwei Variablen 214

U

ÜbgebFeh, Fehler übergeben 144
Umleitung, # 249
Umleitungsoperator (#) 264
umwandeln
▸Grad (Neugrad) 95
▸Rad 157
unbestimmtes Integral
Vorlage für 10
ungleich, ≠ 238
ungültig, testen auf 103
ungültige Elemente 259
ungültige Elemente, entfernen 58
unitV( ), Einheitsvektor 216
unLock, Variable oder Variablengruppe entsperren 216
Untergrenze, floor( ) 81
Untermatrix, subMat( ) 197, 199
Unterstrich, _ 253

V

Variable
Name aus String erstellen 264
Variable oder Funktion kopieren, CopyVar 36
Variablen
alle einbuchstabigen löschen 31
lokal, Local 116
löschen, DelVar 58
Variablen und Funktionen
kopieren 36
Variablen und Variablengruppen entsperren 216
Variablen und Variablengruppen sperren 116
Variablen, sperren und entsperren 91, 116, 216
Varianz, variance( ) 217
varPop( ) (Populationsvarianz) 216
varSamp( ), Stichproben-Varianz 217

Vektoren
Anzeige als Zylindervektor, ▸Cylind 51
Einheit, unitV( ) 216
Kreuzprodukt, crossP( ) 44
Skalarprodukt, dotP( ) 66
verschieben, shift( ) 178
Verteilungsfunktionen
binomCdf( ) 25
binomPdf( ) 25
invNorm( ) 101
invt( ) 101
Invχ^2( ) 101
normCdf( ) (Normalverteilungswahrscheinlichkeit) 136
normPdf( ) (Wahrscheinlichkeitsdichte) 137
poissCdf( ) 145
poissPdf( ) 145
tCdf( ) 203
tPdf( ) 208
$\chi^2$2way( ) 29
χ^2Cdf( ) 29
χ^2GOF( ) 30
χ^2Pdf( ) 31
Vorlagen
Ableitung oder n-te Ableitung 10
Absolutwert 7-8
Bestimmtes Integral 10
Bruch 5
e Exponent 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 $\prod$( ) 9
Quadratwurzel 5
Stückweise definierte Funktion (2 Teile) 6
Stückweise definierte Funktion (n Teile) 7

Summe ∑( ) 9
unbestimmtes Integral 10
zweite Ableitung 10

W

Wahrscheinlichkeit einer Student-t-Verteilung, tCdf( ) 203
Wahrscheinlichkeitsdichte, normPdf( ) 137
Warncodes und -meldungen 274
warnCodes( ), Warning codes 218
Warte-Befehl 218
wenn, when( ) 219
when( ), wenn 219
while, While 220
While, while 220
winkel, ∠ 252
Winkel, angle( ) 14
womit-Operator „|“ 255
womit-Operator, Auswerungsreihenfolge 263

X

x², Quadrat 235
XNOR 242
xor, Boolesches exklusives oder 220

Z

Zähle Tage zwischen Daten, dbd( ) 54
zähle( ), Elemente in einer Liste zählen 42
Zeichen
String, char( ) 28
Zeichencode, ord( ) 143
Zeichen, sign( ) 180
Zeichenfolgen
drehen, drehe( ) 167
zum Erstellen von Variablennamen verwenden 264
Zeichenstring, char( ) 28
Zeige, Daten anzeigen 63
Zeitwert des Geldes, Anzahl Zahlungen 212
Zeitwert des Geldes, Barwert 213
Zeitwert des Geldes, Endwert 212
Zeitwert des Geldes, Zahlungsbetrag 213
Zeitwert des Geldes, Zinsen 212
zeroes( ), Nullstellen 221

zInterval, z-Konfidenzintervall 224
zInterval_1Prop, z-Konfidenzintervall für eine Proportion 224
zInterval_2Prop, z-Konfidenzintervall für zwei Proportionen 225
zInterval_2Samp, z-Konfidenzintervall für zwei Stichproben 226
zTest 226
zTest_1Prop, z-Test für eine Proportion 227
zTest_2Prop, z-Test für zwei Proportionen 228
zTest_2Samp, z-Test für zwei Stichproben 229
Zufall
  Matrix, randMat( ) 158
  Norm, randNorm( ) 158
  Polynom, randPoly( ) 159
  Zahl, RandSeed 159
Zufallsstichprobe 159
zuweisen, := 257
Zwei-Stichproben F-Test 86
zweite Ableitung
  Vorlage für 10
Zyklus, Cycle 50

Δ

Δlist( ), Listendifferenz 113

χ

χ^2Cdf( ) 29
χ^2GOF 30
χ^2Pdf( ) 31