TEXAS INSTRUMENTS

TI-nspire™

TI-Nspire™ CAS Referenshandbok

Denna handbok avser TI-Nspire™ programvara version 4.3. För att erhålla den senaste versionen av dokumentationen, besök education.ti.com/guides.

Viktigt information

Med undantag för vad som uttryckligen anges i den licens som medföljer ett program lämnar Texas Instruments inga garantier, vare sig uttryckliga eller underförstådda, inklusive garantier avseende säljbarhet eller lämplighet för visst ändamål beträffande något program- eller bokmaterial, och tillhandahåller sådant material "i befintligt skick". Under inga omständigheter skall Texas Instruments hållas ansvarigt för några speciella, indirekta eller tillfälliga skador eller följdskador i samband med inköpet eller användningen av materialet, och Texas Instruments:s enda och uteslutande skadeståndsskyldighet, oberoende av anspråkets form, skall inte överstiga det belopp som anges i licensen för programmet. Inte heller skall Texas Instruments hållas ansvarigt för anspråk av något som helst slag beträffande användningen av materialet av annan part.

Licens

Se den fullständiga licensen som installerats i
C:\Program Files\TI Education\<TI-Nspire™ Product Name>\license.

© 2006 - 2016 Texas Instruments Incorporated

Innehåll

Viktigt information 2

Mallar för uttryck **5**

Alfabetisk lista **12**

A 12
B 21
C 24
D 50
E 61
F 71
G 81
I 87
L 95
M 112
N 120
O 129
P 131
Q 140
R 144
S 157
T 183
U 198
V 199
W 200
X 202
Z 203

Symboler **212**

Tomma element **238**

Kortkommandon för att mata in matematiska uttryck **240**

EOS™-hierarki (Equation Operating System) **242**

Felkoder och meddelanden **244**

Varningskoder och meddelanden **253**

Service och Support **255**

Service och garanti för TI-produkter 255

Innehållsförteckning **257**

Mallar för uttryck

Mallar för uttryck erbjuder ett enkelt sätt att mata skriva in uttryck med matematiska standardtecken. När du matar in en mall visas den på inmatningsraden med små block i positioner där du kan skriva in element. En markör visar vilket element du kan skriva in.

Använd piltangenterna eller tryck på [tab] för att flytta till varje elements position, och skriv ett värde eller uttryck för det aktuella elementet. Tryck på [enter] eller [ctrl] [enter] för att utvärdera uttrycket.

Mall för Bråk

[ctrl] [÷] tangenter

Obs: Se även **/ (dela)**, på sidan 214.

Exempel:

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

Mall för Exponent

[^] tangent

Obs: Skriv in det första värdet, tryck på [^] och skriv sedan in exponenten. För att återföra markören till basraden, tryck på högerpilen (▶).

Obs: Se även **^ (potens)**, på sidan 214.

Exempel:

2^3 8

Mall för Kvadratrot

[ctrl] [x²] tangenter

Obs: Se även **√() (kvadratrot)**, på sidan 225.

Exempel:

$\sqrt{4}$ 2

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

Mall för N:te rot

[ctrl] [^] tangenter

Obs: Se även **root()**, på sidan 154.

Exempel:

Mall för N:te rot

ctrl ^ tangenter

$\sqrt[3]{8}$ → 2

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

e exponent mall

e^x tangent

$e^{\square}$

Basen för den naturliga logaritmen *e* upphöjd till

Obs: Se även **e^()**, på sidan 61.

Exempel:

e^1 → e

$e^{1.}$ → 2.71828182846

Mall för Log

ctrl 10^x tangenter

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

Beräknar logaritmen till en specificerad bas. För en förinställning av bas 10, utelämna basen.

Obs: Se även **log()**, på sidan 107.

Exempel:

$\log_4(2.)$ → 0.5

Stegvis mall (2 steg)

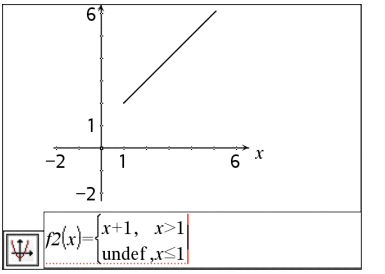
Katalog >

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

Låter dig skapa uttryck och villkor för en stegvis funktion med två steg.- För att lägga till ett steg, klicka i mallen och upprepa mallen.

Obs: Se även **stegvis()**, på sidan 133.

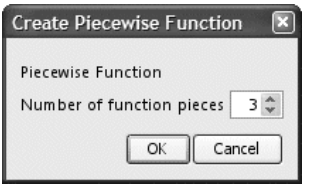
Exempel:



Stegvis mall (N steg)

Katalog >

Låter dig skapa uttryck och villkor för en stegvis funktion med N steg.- Promptar för *N*.



Obs: Se även **stegvis()**, på sidan 133.

Exempel:

Se exemplet på Stegvis mall (2 steg).

Mall för System med 2 ekvationer

Katalog >

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

Skapar ett ekvationssystem med två ekvationer. För att lägga till en rad i ett befintligt system, klicka i mallen och upprepa mallen.

Obs: Se även **system()**, på sidan 182.

Exempel:

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

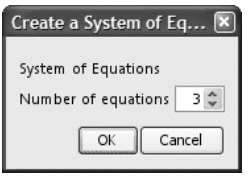
$\text{solve}\left(\begin{cases} y=x^2-2 \\ x+2\cdot y=-1 \end{cases}, x, y\right)$

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

Mall för System med N ekvationer

Katalog >

Låter dig skapa ett ekvationssystem med *N*ekvationer. Promptar för *N*.



Obs: Se även **system()**, på sidan 182.

Exempel:

Se exemplet på mall för Ekvationssystem (2 ekvationer).

Mall för Absolutbelopp

Katalog >

$|\square|$ **Obs:** Se även **abs()**, på sidan 12.

Exempel:

Mall för Absolutbelopp

Katalog >

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

Mall för dd°mm'ss.ss"

Katalog >

□°□'□"

Låter dig skriva in vinklar i formatet **dd°mm'ss.ss''**, där **dd** är antalet decimala grader, **mm** är antalet minuter och **ss.ss** är antalet sekunder.

Exempel:

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

Matrismall (2 x 2)

Katalog >

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

Skapar en 2 x 2-matris.

Exempel:

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

Matrismall (1 x 2)

Katalog >

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

Exempel:

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

Matrismall (2 x 1)

Katalog >

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

Exempel:

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

Matrismall (m x n)

Katalog >

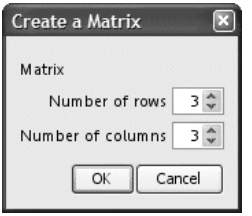
Mallen visas när du har uppmanats att specificera antalet rader och kolumner.

Exempel:

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

Matrismall (m x n)

Katalog >



Obs: Om du skapar en matris med många rader och kolumner kan det ta några sekunder innan den visas.

Mall för Summa (Σ)

Katalog >

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

Exempel:

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

Obs: Se även **Σ() (sumSeq)**, på sidan 226.

Mall för Produkt (Π)

Katalog >

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

Exempel:

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

Obs: Se även **Π() (prodSeq)**, på sidan 225.

Mall för förstaderivata

Katalog >

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

Exempel:

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

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

Mallen för förstaderivata kan också användas för att beräkna förstaderivatan i en punkt.

Obs: Se även **d() (derivata)**, på sidan 222.

Andraderivata, mall

Katalog >

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

Mallen för andraderivata kan också användas för att beräkna andraderivatan i en punkt.

Obs: Se även **d() (derivata)**, på sidan 222.

Exempel:

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

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

Mall för N:te derivata

Katalog >

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

Obs: Se även **d() (derivata)**, på sidan 222.

Exempel:

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

Mall för Bestämd integral

Katalog >

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

Obs: Se även **∫() integral()**, på sidan 212.

Exempel:

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

Mall för obestämd integral

Katalog >

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

Obs: Se även **∫() integral()**, på sidan 212.

Exempel:

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

Mall för Gränsvärde

Katalog >

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

Använd − eller (−) för det vänstra gränsvärdet. Använd + för det högra gränsvärdet.

Exempel:

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

Mall för Gränsvärde

Katalog >

Obs: Se även **gränsvärde()**, på sidan 97.

Alfabetisk lista

Poster som inte är alfabetiska (till exempel, +, ! och >) listas i slutet av detta avsnitt och börjar, på sidan 212. Om inget annat anges har alla exempel i detta avsnitt utförts i det förinställda återställningsläget och alla variabler betraktas som odefinierade.

A

abs() Katalog >

abs(*Expr1***)⇒***uttryck*

abs(*List1***)⇒***lista*

abs(*Matrix1***)⇒***matris*

Ger argumentets absolutbelopp.

Obs: Se även **Mall för Absolutbelopp**, på sidan 7.

Om argumentet är ett komplext tal erhålls talets modul.

Obs: Alla odefinierade variabler behandlas som reella variabler.

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

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

$$|z| \qquad |z|$$

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

amortTbl() Katalog >

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

Amorteringsfunktion som ger en matris i form av en amorteringstabell för en uppsättning av TVM-argument.

NPmt är antalet inbetalningar som skall inkluderas i tabellen. Tabellen börjar med den första inbetalningen.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 196.

- Om du utelämnar *Pmt* används förinställningen *Pmt*=**tvmPmt (***N,I,PV,FV,PpY,CpY,PmtAt***)**.
- Om du utelämnar *FV* används förinställningen *FV*=0.
- Förinställningarna av *PpY*, *CpY* och

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

PmtAt är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

Kolumnerna i resultatmatrisen har följande ordning: Inbetalningsnummer, räntebelopp, kapitalbelopp och balans.

Balansen som visas på rad *n* är balansen efter inbetalning *n*.

Du kan använda resultatmatrisen som indata för de andra amorteringsfunktionerna **ΣInt()** och **ΣPrn()**, se på sidan 226, och **bal()**, se på sidan 21.

and (och) Katalog >

BooleanExpr1 **and** *BooleanExpr2⇒Booleskt uttryck*

BooleanList1 **and** *BooleanList2⇒Boolesk lista*

BooleanMatrix1 **and** *BooleanMatrix2⇒Boolesk matris*

Ger resultatet sant eller falskt eller en förenklad form av den ursprungliga inmatningen.

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

Integer1 **and** *Integer2⇒heltal*

Jämför två reella heltal bit för bit med en **och**-operation. Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om båda bitarna är 1, annars blir resultatet 0. Det erhållna värdet representerar bitresultatet och visas enligt Bas-läget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

I hexadecimalt basläge:

0h7AC36 and 0h3D5F	0h2C16

Viktigt: Noll, inte bokstaven O.

I binärt basläge:

0b100101 and 0b100	0b100

I decimalt basläge:

37 and 0b100	4

and (och)

Katalog >

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk moduloberäkning för att få ned värdet till lämplig nivå.

Obs: En binär inmatning kan ha upp till 64 siffror (exklusive prefixet 0b). En hexadecimal inmatning kan ha upp till 16 siffror.

angle()

Katalog >

angle(*Expr1***)⇒***uttryck*

Ger argumentets vinkel med argumentet tolkat som ett komplext tal.

Obs: Alla odefinierade variabler behandlas som reella variabler.

I vinkelläget Grader:

$\text{angle}(0+2\cdot i)$ → 90

I vinkelläget Nygrader:

$\text{angle}(0+3\cdot i)$ → 100

I vinkelläget Radianer:

$\text{angle}(1+i)$ → $\frac{\pi}{4}$

$\text{angle}(z)$ → $\frac{-\pi\cdot(\text{sign}(z)-1)}{2}$

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

vinkel(*List1***)⇒***lista*

vinkel(*Matrix1***)⇒***matris*

Ger en lista eller matris över vinklarna hos elementen i *List1* eller *Matrix1*, där varje element tolkas som ett komplext tal som representerar en tvådimensionell rektangulär koordinatpunkt.

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

ANOVA

Katalog >

ANOVA *List1*,*List2*[,*List3*,...,*List20*][,*Flag*]

Utför en 1-vägs variansanalys för att jämföra medelvärdena hos 2 till 20 populationer. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

ANOVA

Katalog >

Flag=0 för Data, *Flag*=1 för Statistik

Resultatvariabel	Beskrivning
stat.F	Värdet på F-statistiken
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos grupperna
stat.SS	Kvadratsumma hos grupperna
stat.MS	Kvadratmedelvärde hos grupperna
stat.dfError	Frihetsgrader hos felen
stat.SSError	Kvadratsumma hos felen
stat.MSError	Kvadratmedelvärde hos felen
stat.sp	Sammanslagen (pooled) standardavvikelse
stat.xbarlist	Medelvärdet på listornas indata
stat.CLowerList	95 % konfidensintervall för medelvärdet hos varje indatalista
stat.CUpperList	95 % konfidensintervall för medelvärdet hos varje indatalista

ANOVA2way

Katalog >

ANOVA 2-vägs*Lista1*,*Lista2*
[,*Lista3*,...,*Lista10*][,*LevRow*]

Beräknar en 2-vägs variansanalys för att jämföra medelvärdena hos 2 till 10 populationer. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

NivRad=0 för Block

NivRad=2,3,...,*Län*-1, för Två Faktorer, där *Län*=längd(*Lista1*)=längd(*Lista2*) = ... = längd(*Lista10*) och *Län* / *NivRad* ∈{2,3,...}

Utdata: Block Design

Resultatvariabel	Beskrivning
stat.F	F statistik för kolumnfaktorn
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas

Resultatvariabel	Beskrivning
stat.df	Frihetsgrader hos kolumnfaktorn
stat.SS	Kvadratsumma hos kolumnfaktorn
stat.MS	Kvadratmedelvärde hos kolumnfaktorn
Statistik.FBlock	F statistik för faktor
stat.PValBlock	Lägsta sannolikhet vid vilken nollhypotesen kan förkastas
stat.dfBlock	Frihetsgrader hos faktor
stat.SSBlock	Kvadratsumma hos faktor
stat.MSBlock	Kvadratmedelvärde hos faktor
stat.dfError	Frihetsgrader hos felen
stat.SSError	Kvadratsumma hos felen
stat.MSError	Kvadratmedelvärde hos felen
stat.s	Standardavvikelse hos felet

Utdata för KOLUMNFAKTOR

Resultatvariabel	Beskrivning
stat.Fcol	F statistik för kolumnfaktorn
stat.PValCol	Sannolikhetsvärde på kolumnfaktorn
stat.dfCol	Frihetsgrader hos kolumnfaktorn
stat.SSCol	Kvadratsumma hos kolumnfaktorn
stat.MSCol	Kvadratmedelvärde hos kolumnfaktorn

Utdata för RADFAKTOR

Resultatvariabel	Beskrivning
stat.FRow	F statistik för radfaktorn
stat.PValRow	Sannolikhetsvärde på radfaktorn
stat.dfRow	Frihetsgrader hos radfaktorn
stat.SSRow	Kvadratsumma hos radfaktorn
stat.MSRow	Kvadratmedelvärde hos radfaktorn

Utdata för INTERAKTION

Resultatvariabel	Beskrivning
stat.FInteract	F statistik för interaktionen
stat.PValInteract	Sannolikhetsvärde på interaktionen
stat.dfInteract	Frihetsgrader hos interaktionen
stat.SSInteract	Kvadratsumma hos interaktionen
stat.MSInteract	Kvadratmedelvärde hos interaktionen

Utdata för FEL

Resultatvariabel	Beskrivning
stat.dfError	Frihetsgrader hos felen
stat.SSError	Kvadratsumma hos felen
stat.MSError	Kvadratmedelvärde hos felen
s	Standardavvikelse hos felet

Ans (svar)

ctrl (–) tangenter

Ans⇒*värde*

Ger resultatet på det senast beräknade uttrycket.

56	56
56+4	60
60+4	64

approx()

Katalog >

approx(*Expr1*)⇒*uttryck*

Visar resultatet av beräkningen av argumentet som ett uttryck med decimala värden, när så är möjligt, oavsett den aktuella inställningen av **Auto eller Ungefärlig**.

Detta motsvarar att skriva in argumentet och trycka på ctrl enter.

approx(*List1*)⇒*lista*

approx(*Matrix1*)⇒*matris*

Ger en lista eller *matris* där varje element har beräknats till ett decimalt värde, när så är möjligt.

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

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

▸approxFraction() Katalog >

Expr▸**approxFraction(**[*Tol*]**)**⇒*uttryck*

List ▸**approxFraction(**[*Tol*]**)**⇒*lista*

Matrix ▸**approxFraction(**[*Tol*]**)**⇒*matris*

Ger indata som ett bråk med hjälp av toleransen hos *Tol*. Om *Tol* utelämnas används en tolerans på 5.E-14.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`@>approxFraction(…)`**.

$\frac{1}{2}+\frac{1}{3}+\tan(\pi)$ → 0.833333

0.83333333333333▸approxFraction(5.E-14) → $\frac{5}{6}$

{π,1.5}▸approxFraction(5.E-14) → $\left\{\frac{5419351}{1725033},\frac{3}{2}\right\}$

approxRational() Katalog >

approxRational(*Expr*[**,** *Tol*]**)**⇒*uttryck*

approxRational(*List*[**,** *Tol*]**)**⇒*lista*

approxRational(*Matrix*[**,** *Tol*]**)**⇒*matris*

Ger argumentet som ett bråk med hjälp av toleransen hos *Tol*. Om *Tol* utelämnas används en tolerans på 5.E-14.

approxRational(0.333,5·10^-5) → $\frac{333}{1000}$

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

arccos() **Se cos⁻¹(), på sidan 36.**

arccosh() **Se cosh⁻¹(), på sidan 37.**

arccot() **Se cot⁻¹(), på sidan 38.**

arccoth() **Se coth⁻¹(), på sidan 39.**

arccsc() **Se csc⁻¹(), på sidan 41.**

arccsch() **Se csch^{-1}(), på sidan 42.**

arcLen() **Katalog >**

arcLen(*Expr1*,*Var*,*Start*,*End*) ⇒*uttryck*

Ger båglängden hos *Expr1* från *Start* till *End* med hänsyn till variabeln *Var*.

Båglängden beräknas som en integral baserat på ett definierat funktionsläge.

arcLen(*List1*,*Var*,*Start*,*End*)⇒*lista*

Ger en lista på båglängden hos varje element i *List1* från *Start* till *End* med hänsyn till *Var*.

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

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

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

arcsec() **Se sec^{-1}(), på sidan 157.**

arcsech() **Se sech^{-1}(), på sidan 158.**

arcsin() **Se sin^{-1}(), på sidan 169.**

arcsinh() **Se sinh^{-1}(), på sidan 170.**

arctan() **Se tan^{-1}(), på sidan 184.**

arctanh() **Se tanh^{-1}(), på sidan 185.**

augment() Katalog >

augment(*List1, List2***)⇒***lista*

Ger en ny lista med *List2* inlagd i slutet på *List1*.

augment(*Matrix1***, *Matrix2***)⇒***matris*

Ger en ny matris med *Matrix2* fogad till *Matrix1*. När kommatecknet (,) används måste matriserna ha samma raddimensioner och *Matrix2* fogas till *Matrix1* som nya kolumner. Ändrar inte *Matrix1* eller *Matrix2*.

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

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

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

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

avgRC() Katalog >

avgRC(*Expr1***, *Var* [=*Value*] [, *Step*])⇒***uttryck***

avgRC(*Expr1***, *Var* [=*Value*] [, *List1*])⇒***lista***

avgRC(*List1***, *Var* [=*Value*] [, *Step*])⇒***lista***

avgRC(*Matrix1***, *Var* [=*Value*] [, *Step*])⇒***matris***

Ger differenskvoten i positiv riktning.

Expr1 kan vara ett användardefinierat funktionsnamn (se **Func**).

När *Värde* specificeras överstyr detta värde eventuella tidigare variabeltilldelningar eller aktuella ersättningar av typ "|" för variabeln.

Step är stegvärdet. Om *Step* utelämnas används förinställningen 0.001.

Observera att den liknande funktionen **centralDiff()** använder den symmetriska differenskvoten.

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

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

avgRC(x^2-x+2,*x*) $2\cdot(x-0.4995)$

avgRC(x^2-x+2,*x*,0.1) $2\cdot(x-0.45)$

avgRC(x^2-x+2,*x*,3) $2\cdot(x+1)$

bal() Katalog >

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

bal(*NPmt*,*amortTable***)**⇒*värde*

Amorteringsfunktion som beräknar planerad balans efter en specificerad inbetalning.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 196.

NPmt anger numret på den inbetalning efter vilken du vill att data skall beräknas.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 196.

- Om du utelämnar *Pmt* används förinställningen *Pmt*=**tvmPmt (***N*,*I*,*PV*,*FV*,*PpY*,*CpY*,*PmtAt***)**.
- Om du utelämnar *FV* används förinställningen *FV*=0.
- Förinställningarna av *PpY*, *CpY* och *PmtAt* är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

bal(*NPmt*,*amortTable***)** beräknar lånebalansen efter inbetalning nummer *NPmt*, baserat på amorteringstabell *amortTable*. Argumentet *amortTable* måste vara en matris i den form som beskrivs under **amortTbl()**, på sidan 12.

Obs: Se även **Σint()** och **Σprn()**, på sidan 226.

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

bal(4,*tbl*) 1674.27

▸Base2 Katalog >

Integer1 **▸Base2**⇒*heltal*

256▸Base2 0b100000000

0h1F▸Base2 0b11111

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>Base2**.

Omvandlar *Integer1* till ett binärt tal. Binära och hexadecimala tal har alltid prefixet 0b respektive 0h. Noll, inte bokstaven O, följt av b eller h.

0b *binärtTal*

0h *hexadecimaltTal*

Ett binärt tal kan ha upp till 64 siffror. Ett hexadecimalt tal kan ha upp till 16 siffror.

Utan prefix behandlas *Integer1* som ett decimalt tal (bas 10). Resultatet visas i binär form, oavsett Bas-läget.

Negativa tal visas i "tvåkomplement"-form. Exempel,

−1 visas som 0hFFFFFFFFFFFFFFFF i Hexadecimalt basläge 0b111...111 (64 1's) i Binärt basläge

-2^{63} visas som 0h8000000000000000 i Hexadecimalt basläge 0b100...000 (63 zeros) i Binärt basläge

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. Se följande exempel på värden utanför området.

2^{63} blir -2^{63} och visas som 0h8000000000000000 i Hexadecimalt basläge 0b100...000 (63 nollor) i Binärt basläge

2^{64} blir 0 och visas som 0h0 i Hexadecimalt basläge 0b0 i Binärt basläge

$-2^{63} - 1$ blir $2^{63} - 1$ och visas som 0h7FFFFFFFFFFFFFFF i Hexadecimalt basläge 0b111...111 (64 ettor) i Binärt basläge

▸Base10 Katalog >

Integer1 **▸Base10**⇒*heltal*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>Base10**.

Omvandlar *Integer1* till ett decimalt tal (bas 10). En binär eller hexadecimal inmatning måste alltid ha prefixet 0b respektive 0h.

0b *binaryNumber*

0h *hexadecimalNumber*

Noll, inte bokstaven O, följt av b eller h.

Ett binärt tal kan ha upp till 64 siffror. Ett hexadecimalt tal kan ha upp till 16 siffror.

Utan prefix behandlas *Integer1* som ett decimalt tal. Resultatet visas i decimal form, oavsett Bas-läget.

0b10011▸Base10	19
0h1F▸Base10	31

▸Base16 Katalog >

Integer1 **▸Base16**⇒*heltal*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>Base16**.

Konverterar *Integer1* till ett hexadecimalt tal. Binära och hexadecimala tal har alltid prefixet 0b respektive 0h.

0b *binaryNumber*

0h *hexadecimalNumber*

Noll, inte bokstaven O, följt av b eller h.

Ett binärt tal kan ha upp till 64 siffror. Ett hexadecimalt tal kan ha upp till 16 siffror.

Utan prefix behandlas *Integer1* som ett decimalt tal (bas 10). Resultatet visas i hexadecimal form, oavsett Bas-läget.

256▸Base16	0h100
0b111100001111▸Base16	0hF0F

▸Base16 Katalog >

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. För mer information, se ▸**Base2**, på sidan 21.

binomCdf() Katalog >

binomCdf(*n*,*p*)⇒*lista*

binomCdf(*n*,*p*,*lowBound*,*upBound*)⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

binomCdf(*n*,*p*,*upBound*)för P(0≤X ≤*upBound*)⇒*tal* om *upBound* är ett tal, *lista* om *upBound* är en lista

Beräknar en kumulativ sannolikhet för den diskreta binomialfördelningen med *n* antal försök och sannolikheten *p* för att lyckas vid varje försök.

För P(X ≤ *upBound*), sätt *lowBound*=0

binomPdf() Katalog >

binomPdf(*n*,*p*)⇒*lista*

binomPdf(*n*,*p*,*XVal*)⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar en sannolikhet för den diskreta binomialfördelningen med *n* antal försök och sannolikheten *p* för att lyckas vid varje försök.

C

ceiling() Katalog >

ceiling(*Expr1*)⇒*heltal*

Ger det närmaste heltal som är ≥ argumentet.

ceiling(.456) 1.

ceiling() Katalog >

Argumentet kan vara ett reellt eller ett komplext tal.

Obs: Se även **floor()**.

ceiling(*List1***)⇒***lista***

ceiling(*Matrix1***)⇒***matris***

Ger en lista eller matris över taket för varje element.

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

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

centralDiff() Katalog >

centralDiff(*Uttr1***,***Var* [=*Värde*] [,*Steg*])⇒*uttryck*

centralDiff(*Uttr1***,***Var* [,*Steg*])**|***Var*=*Värde*⇒*uttryck*

centralDiff(*Uttr1***,***Var* [=*Värde*] [,*Lista*])⇒*lista*

centralDiff(*Lista1***,***Var* [=*Värde*] [,*Steg*])⇒*lista*

centralDiff(*Matris1***,***Var* [=*Värde*] [,*Steg*])⇒*matris*

Ger den numeriska derivatan genom att använda formeln för symmetrisk differenskvot.

När *Värde* specificeras överstyr detta värde eventuella tidigare variabeltilldelningar eller aktuella ersättningar av typ "|" för variabeln.

Steg är stegvärdet. Om *Steg* utelämnas används förinställningen 0.001.

När du använder *Lista1* eller *Matris1* utförs operationen på värdena i listan eller matriselementen.

Obs: Se även **avgRC()** och **d()**.

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

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

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

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

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

cFactor(*Expr1*[**,***Var*]**)⇒***uttryck*

cFactor(*List1*[**,***Var*]**)⇒***lista*

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

cFactor(*Expr1***)** ger en faktorisering av *Expr1* baserad på uttryckets alla variabler med en gemensam nämnare.

Expr1 faktoriseras så långt det går till linjära, rationella faktorer även om detta inför nya icke-reella tal. Detta alternativ är lämpligt om du vill ha en faktorisering baserad på mer än en variabel.

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

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

$\text{cFactor}\left(x^2+3\right)$ x^2+3

$\text{cFactor}\left(x^2+a\right)$ x^2+a

cFactor(*Expr1***,***Var***)** ger en faktorisering av *Expr1* baserad på variabeln *Var*.

Expr1 faktoriseras så långt det går till faktorer som är linjära i *Var*, med kanske icke-reella konstanter, även om detta inför irrationella konstanter eller deluttryck som är irrationella i andra variabler.

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

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

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

Faktorerna och deras termer sorteras med *Var* som huvudvariabel. Liknande potenser av *Var* samlas in i varje faktor. Inkludera *Var* om faktorisering baserad på endast denna variabel behövs och du är villig att acceptera irrationella uttryck i andra variabler för att öka faktoriseringen baserad på *Var*. Viss tillfällig faktorisering kan ske vad gäller andra variabler.

Med inställningen Auto i läge **Auto eller Ungefärlig** medger inkludering av *Var* också en uppskattning med koefficienter med flytande decimalkomma när irrationella koefficienter inte explicit kan uttryckas kortfattat med termerna i de inbyggda funktionerna. Även med endast en variabel kan inkludering av *Var* ge en mer fullständig faktorisering.

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

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

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Obs: Se även **faktor()**.

char()

Katalog >

char(*Integer***)⇒***tecken***

Ger en teckensträng som innehåller tecknet med numret *Integer* från handenhetens teckenuppsättning. Det giltiga området för *Integer* är 0–65535.

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

charPoly()

Katalog >

charPoly(*squareMatrix,Var***)⇒***polynom*

charPoly(*squareMatrix,Expr***)⇒***polynom*

charPoly
(*squareMatrix1,Matrix2***)⇒***polynom*

Ger det karakteristiska polynomet för *squareMatrix*. Det karakteristiska polynomet för *n*×*n* matris A, betecknat $p_A(\lambda)$, är polynomet definierat av

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

där I betecknar enhetsmatrisen *n*×*n*.

squareMatrix1 och *squareMatrix2* måste ha samma dimensioner.

$m := \begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 0 \\ -2 & 2 & 5 \end{bmatrix}$	$\begin{bmatrix} 1 & 3 & 0 \\ 2 & -1 & 0 \\ -2 & 2 & 5 \end{bmatrix}$
charPoly(*m*,*x*)	$-x^3+5\cdot x^2+7\cdot x-35$
charPoly(*m*,x^2+1)	$-x^6+2\cdot x^4+14\cdot x^2-24$
charPoly(*m*,*m*)	0

$\chi^2$2way

Katalog >

$\chi^2$2way *ObsMatrix*

chi22way *ObsMatrix*

Beräknar ett χ^2-test för association på 2-vägstabellen över antal i den observerade matrisen *ObsMatrix*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

För information om effekten av tomma element i en matris, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.χ^2	Chi-kvadratstatistik: summa (observerad - förväntad)^2/förväntad

Resultatvariabel	Beskrivning
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos chi-kvadratstatistiken
stat.ExpMat	Matris över förväntad elementräknetabell, baserad på nollhypotesen
stat.CompMat	Matris över elementbidrag till chi-kvadratstatistiken

χ^2Cdf()

Katalog >

χ^2**Cdf(***lowBound*,*upBound*,*df***)**⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

chi2Cdf(*lowBound*,*upBound*,*df***)**⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

Beräknar sannolikheten för χ^2-fördelning mellan *lowBound* och *upBound* för den specificerade frihetsgraden *df*.

För P(X *upBound*), sätt *lowBound* = 0.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

χ^2GOF

Katalog >

χ^2**GOF** *obsList*,*expList*,*df*

chi2GOF *obsList*,*expList*,*df*

Utför ett test för att bekräfta att urvalsdata är från en population som följer en specificerad fördelning. *obsList* är en lista med data och måste innehålla heltal. En sammanfattning av resultaten visas i variabeln *stat.results*, på sidan 177.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.χ^2	Chi-kvadratstatistik: summa (observerad - förväntad)^2/förväntad
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos chi-kvadratstatistiken
stat.CompList	Elementbidrag till chi-kvadratstatistiken

χ^2Pdf()

Katalog >

χ^2Pdf(*XVal*,*df*)⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

chi2Pdf(*XVal*,*df*)⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar värde hos täthetsfunktionen (pdf) för χ^2-fördelningen vid ett specificerat *XVal*-värde för den specificerade frihetsgraden *df*.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

ClearAZ

Katalog >

ClearAZ

Rensar alla variabler som har ett enda tecken i det aktuella problemet.

Om en eller flera variabler är låsta visar detta kommando ett felmeddelande och tar endast bort olåsta variabler. Se **unLock**, på sidan 199.

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

ClrErr

Katalog >

ClrErr

Rensar felstatusen och ställer in systemvariabeln *errCode* på noll.

För ett exempel på **ClrErr**, se exempel 2 under kommandot **Try**, på sidan 192.

ClrErr

Katalog >

Villkoret **Else** i blocket **Try...Else...EndTry** bör använda **ClrErr** eller **PassErr**. Om felet skall processas eller ignoreras, använd **ClrErr**. Om det är okänt hur felet skall hanteras, använd **PassErr** för att skicka felet vidare till nästa felhanterare. Om det inte finns någon ytterligare felhanterare för **Try...Else...EndTry** visas feldialogrutan som normal.

Obs: Se även **PassErr**, på sidan 132 och **Try**, på sidan 192.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

colAugment()

Katalog >

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

Ger en ny matris med *Matrix2* fogad till *Matrix1*. Matriserna måste ha samma kolumndimensioner och *Matrix2* fogas till *Matrix1* som nya rader. Ändrar inte *Matrix1* eller *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()

Katalog >

colDim(*Matrix***)⇒*uttryck***

Ger antalet kolumner i *Matrix*.

Obs: Se även **rowDim()**.

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

colNorm()

Katalog >

colNorm(*Matrix***)⇒*uttryck***

Ger maximum av summorna av absolutbeloppen på elementen i kolumnerna i *Matrix*.

Obs: Odefinierade matriselement är ej tillåtna. Se även **rowNorm()**.

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

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

comDenom(*Expr1*[,*Var*]**)**⇒*uttryck*

comDenom(*List1*[,*Var*]**)**⇒*lista*

comDenom(*Matrix1*[,*Var*]**)**⇒*matris*

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

comDenom(*Expr1***)** ger en reducerad kvot mellan en fullt expanderad täljare och en fullt expanderad nämnare.

comDenom(*Expr1*,*Var***)** ger en reducerad kvot mellan täljare och nämnare som expanderats enligt *Var*. Termerna och deras faktorer sorteras med *Var* som huvudvariabel. Liknande potenser av *Var* samlas in. Viss tillfällig faktorisering kan ske av de insamlade koefficienterna. Jämfört med att utesluta *Var* sparar detta ofta tid, minne och skärmutrymme, vilket gör uttrycket mer begripligt. Det gör också att påföljande operationer på resultatet går snabbare och med mindre risk att minnet tar slut.

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

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

Om *Var* inte förekommer i *Expr1* ger **comDenom(***Expr1*,*Var***)** en reducerad kvot mellan en oexpanderad täljare och en oexpanderad nämnare. Sådana resultat sparar i regel ännu mer tid, minne och skärmutrymme. Sådana delvis faktoriserade resultat gör också att påföljande operationer på resultatet går mycket snabbare och med mycket mindre risk att minnet tar slut.

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

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

Även om det inte finns någon nämnare är funktionen **comden** ofta ett snabbt sätt att erhålla en delvis faktorisering om **factor()** är för långsam eller om den utarmar minnet.

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

Tips: Mata in denna **comden()** funktionsdefinition och prova den rutinmässigt som ett alternativ till **comDenom()** och **factor()**.

completeSquare ()

Katalog >

completeSquare(ExprOrEqn, Var)Þuttryck eller ekvation

completeSquare(*ExprOrEqn*, *Var^Power***)⇒***uttryck eller ekvation*

completeSquare(*ExprOrEqn*, *Var1, Var2 [,...]***)⇒***uttryck eller ekvation*

completeSquare(*ExprOrEqn*, {*Var1, Var2 [,...]*}**)⇒***uttryck eller ekvation*

Konverterar ett kvadratiskt polynomuttryck på formen a·x²+b·x+c till formen a·(x-h)²+k

– eller –

Konverterar en andragradsekvation på formen a•x2+b•x+c=d till formen a·(x-h)²=k

Det första argumentet måste vara ett kvadratiskt uttryck eller en andragradsekvation i standardform med avseende på det andra argumentet.

Det andra argumentet måste vara en enda envariabelterm eller en enda envariabelterm upphöjd till en rationell potens, till exempel x, y² eller z^(1/3).

Den tredje och fjärde syntaxen försöker att fullborda kvadraten avseende variablerna Var1, Var2 [,...]).

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

conj()

Katalog >

conj(*Expr1***)⇒***uttryck*

conj(*List1***)⇒***lista*

conj(*Matrix1***)⇒***matris*

Ger argumentets komplexkonjugat.

Obs: Alla odefinierade variabler behandlas som reella variabler.

conj($1+2\cdot i$) $1-2\cdot i$

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

conj(z) z

conj($x+i\cdot y$) $x-y\cdot i$

constructMat() Katalog >

constructMat
(*Expr***,***Var1***,***Var2***,***numRows***,***numCols***)**
⇒*matris*

Ger en matris baserad på argumenten.

Expr är ett uttryck i variablerna *Var1* och *Var2*. Element i den resulterande matrisen skapas genom att utvärdera *Expr* för varje ökat värde på *Var1* och *Var2*.

Var1 ökas automatiskt från **1** till och med *numRows*. Inom varje rad ökas *Var2* från **1** till och med *numCols*.

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

CopyVar Katalog >

CopyVar *Var1***,** *Var2*

CopyVar *Var1***.,** *Var2***.**

CopyVar *Var1, Var2* kopierar värdet på variabel *Var1* till variabel *Var2*, och skapar *Var2* vid behov. Variabeln *Var1* måste ha ett värde.

Om *Var1* är namnet på en befintlig användardefinierad funktion kopieras definitionen på denna funktion till funktionen *Var2*. Funktionen *Var1* måste vara definierad.

Var1 måste uppfylla kraven för namngivning av variabler eller måste vara ett indirection-uttryck som förenklas till ett variabelnamn som uppfyller kraven.

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.* kopierar alla led i variabelgruppen *Var1.* till gruppen *Var2.* och skapar *Var2.* vid behov.

Var1. måste vara namnet på en befintlig variabelgrupp, till exempel, den statistiska *stat.nn*-resultat eller variabler skapade med funktionen **LibShortcut()**. Om *Var2.* redan finns ersätter detta kommando alla led som är gemensamma för båda grupperna och lägger till de led som inte redan finns. Om en eller flera medlemmar av *Var2.* är låsta lämnas alla medlemmar av *Var2.* oförändrade.

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()

Katalog >

corrMat(*List1*,*List2*[,...[,*List20*]]**)**

Beräknar korrelationsmatrisen för den sammanfogade matrisen [*List1, List2, ..., List20*].

▸cos

Katalog >

Expr **▸cos**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>cos**.

Representerar *Expr* i termer av cosinus. Detta är en omvandlingsoperator för visning. Den kan endast användas i slutet av inmatningsraden.

▸cos reducerar alla potenser av sin(...) modulo 1−cos(...)^2 så att eventuella återstående potenser av cos(...) har exponenter i området (0, 2). Resultatet kommer sålunda att vara fritt från sin(...) om, och endast om, sin(...) finns i det givna uttrycket vid jämna potenser.

Obs: Denna omvandlingsoperator stöds inte i vinkellägena Grader och Nygrader. Innan du använder den, kontrollera att vinkelläget är inställt på Radianer och att *Expr* inte innehåller explicita referenser till vinklar i grader eller nygrader.

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

cos()

trig tangent

cos(*Expr1***)⇒***uttryck*

cos(*List1***)⇒***lista*

cos(*Expr1***)** ger argumentets cosinus som ett uttryck.

cos(*List1***)** ger en lista på cosinus för alla element i *List1*.

I vinkelläget Grader:

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

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

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

I vinkelläget Nygrader:

Obs: Argumentet tolkas som en vinkel i grader, nygrader eller radianer enligt det inställda vinkelläget. Du kan använda °, G eller r för att tillfälligt överstyra vinkelläget.

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

I vinkelläget Radianer:

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

$\cos(45°)$ $\frac{\sqrt{2}}{2}$

cos(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med cosinus för *squareMatrix1*. Detta är inte detsamma som att beräkna cosinus för varje element.

När en skalär funktion f(A) används på *squareMatrix1* (A) beräknas resultatet med algoritmen:

Beräkna egenvärdena (λ_i) och egenvektorerna (V_i) för A.

squareMatrix1 måste vara möjlig att diagonalisera. Den får inte heller ha symboliska variabler som inte har tilldelats ett värde.

Forma matriserna:

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

Då är $A = X\,B\,X^{-1}$ och $f(A) = X\,f(B)\,X^{-1}$. Exempelvis $\cos(A) = X\cos(B)\,X^{-1}$ där:

cos(B) =

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

Alla beräkningar utförs med flyttalsaritmetik.

I vinkelläget Radianer:

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

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

cos⁻¹() — trig tangent

cos⁻¹(*Expr1***)⇒***uttryck***

cos⁻¹(*List1***)⇒***lista***

cos⁻¹(*Expr1***)** ger den vinkel vars cosinus är *Expr1* som ett uttryck.

cos⁻¹(*List1***)** ger en lista på invers cosinus för varje element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **arccos(...)**.

cos⁻¹(*squareMatrix1***)⇒***squareMatrix***

Ger matrisen med invers cosinus för *squareMatrix1*. Detta är inte detsamma som att beräkna invers cosinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Grader:

$\cos^{-1}(1)$ → 0

I vinkelläget Nygrader:

$\cos^{-1}(0)$ → 100

I vinkelläget Radianer:

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

I vinkelläget Radianer och i Rektangulärt komplext format:

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

cosh() — Katalog >

cosh(*Expr1***)⇒***uttryck***

cosh(*List1***)⇒***lista***

cosh(*Expr1***)** ger argumentets hyperboliska cosinus som ett uttryck.

cosh(*List1***)** ger en lista på hyperbolisk cosinus för varje element i *List1*.

cosh(*squareMatrix1***)⇒***kvadratMatris*

I vinkelläget Grader:

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

I vinkelläget Radianer:

cosh()

Katalog >

Ger matrisen med hyperbolisk cosinus för *squareMatrix1*. Detta är inte detsamma som att beräkna hyperbolisk cosinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

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

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

$\cosh^{-1}()$

Katalog >

cosh^{-1}(*Expr1***)⇒***uttryck***

cosh^{-1}(*List1***)⇒***lista***

$\cosh^{-1}(1)$ → 0

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

cosh^{-1}(*Expr1***)** ger argumentets inversa hyperboliska cosinus som ett uttryck.

cosh^{-1}(*List1***)** ger en lista på invers hyperbolisk cosinus för varje element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **arccosh(...)**.

cosh^{-1}(*squareMatrix1***)⇒***kvadratMatris***

Ger matrisen med invers hyperbolisk cosinus för *squareMatrix1*. Detta är inte detsamma som att beräkna invers hyperbolisk cosinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer och i Rektangulärt komplext format:

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

cot()

trig tangent

cot(*Expr1***)** ⇒ *uttryck*

cot(*List1***)** ⇒ *lista*

I vinkelläget Grader:

$\cot(45)$ → 1

cot()

trig tangent

Ger cotangens för *Expr1* eller en lista på cotangens för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i grader, nygrader eller radianer enligt det inställda vinkelläget. Du kan använda °, G eller r för att tillfälligt överstyra vinkelläget.

I vinkelläget Nygrader:

$\cot(50)$ → 1

I vinkelläget Radianer:

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

cot⁻¹()

trig tangent

cot⁻¹(*Expr1***)⇒***uttryck*

cot⁻¹(*List1***)⇒***lista*

Ger den vinkel vars cotangens är *Expr1* eller en lista på invers cotangens för varje element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arccot(…)`**.

I vinkelläget Grader:

$\cot^{-1}(1)$ → 45

I vinkelläget Nygrader:

$\cot^{-1}(1)$ → 50

I vinkelläget Radianer:

$\cot^{-1}(1)$ → $\frac{\pi}{4}$

coth()

Katalog >

coth(*Expr1***)⇒***uttryck*

coth(*List1***)⇒***lista*

Ger hyperbolisk cotangens för *Expr1* eller en lista på hyperbolisk cotangens för alla element i *List1*.

$\coth(1.2)$ → 1.19954

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

coth⁻¹() Katalog >

coth⁻¹(*Expr1***)⇒***uttryck***

coth⁻¹(*List1***)⇒***lista***

Ger den inversa hyperboliska cotangensen för *Expr1* eller en lista på invers hyperbolisk cotangens för alla element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arccoth(...)`**.

$\coth^{-1}(3.5)$ 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() Katalog >

count(*Value1orList1* [,*Value2orList2* [,...]]**)⇒***värde*

Ger det totala ackumulerade antalet element i argumenten som utvärderas till numeriska värden.

Varje argument kan vara ett uttryck, ett värde, en lista eller en matris. Du kan blanda datatyper och använda argument med olika dimensioner.

För en lista, en matris, eller ett område av celler, utvärderas varje element för att bestämma om det skall inkluderas i räkningen.

I applikationen Listor och kalkylblad kan du använda ett område av celler i stället för ett argument.

Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

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

I det sista exemplet räknas endast 1/2 och $3+4*i$. De återstående argumenten, förutsatt att x är odefinierad, utvärderas inte till numeriska värden.

countif() Katalog >

countif(*List*,*Criteria***)⇒***värde*

Ger det totala ackumulerade antalet element i *List* som uppfyller specificerade *Criteria*.

Criteria kan vara:

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

Räknar antalet element som är lika med 3.

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

countif() Katalog >

- Ett värde, ett uttryck eller en sträng. Som exempel räknar **3** endast de element i *List* som förenklas till värdet 3.
- Ett booleskt uttryck som innehåller symbolen **?** fungerar som platshållare för varje element. Som exempel räknar **?<5** endast de element i *List* som är lägre än 5.

I applikationen Listor och kalkylblad kan du använda ett område av celler i stället för *List*.

Tomma element i listan ignoreras. För mer information om tomma element, se på sidan 238.

Obs: Se även **sumIf()**, på sidan 182 och **frequency()**, på sidan 78.

Räknar antalet element som är lika med "def."

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

Räknar antalet element som är lika med x. I detta exempel förutsätts att variabeln x är odefinierad.

countIf({1,3,5,7,9}, ?<5) → 2

Räknar 1 och 3.

countIf({1,3,5,7,9}, 2<?<8) → 3

Räknar 3, 5 och 7.

countIf({1,3,5,7,9}, ?<4 or ?>6) → 4

Räknar 1, 3, 7 och 9.

cPolyRoots() Katalog >

cPolyRoots(*Poly*, *Var***)⇒***lista***

cPolyRoots(*ListaPåKoeff***)⇒***lista***

Den första syntaxen, **cPolyRoots(***Poly*, *Var***)**, ger en lista på komplexa rötter i polynomet *Poly* med avseende på *Var*.

Poly måste vara ett polynom i en variabel.

Den andra syntaxen, **cPolyRoots (***ListaPåKoeff***)**, ger en lista på komplexa rötter för koefficienterna i *ListaPåKoeff*.

Obs: Se även **polyRoots()**, på sidan 137.

polyRoots(y^3+1, y) → {-1}

cPolyRoots(y^3+1, y) → {-1, 1/2 − (√3/2)·i, 1/2 + (√3/2)·i}

polyRoots(x^2+2·x+1, x) → {-1, -1}

cPolyRoots({1,2,1}) → {-1, -1}

crossP() Katalog >

crossP(*List1***,** *List2***)⇒***lista*

Ger vektorprodukten av *List1* och *List2* som en lista.

List1 och *List2* måste ha samma dimension och dimensionen måste vara antingen 2 eller 3.

crossP(*Vector1***,** *Vector2***)⇒***vektor*

Ger en rad- eller kolumnvektor (beroende på argumenten) som är vektorprodukten av *Vector1* och *Vector2*.

Både *Vector1* och *Vector2* måste vara radvektorer eller båda måste vara kolumnvektorer. Båda vektorerna måste ha samma dimension och dimensionen måste vara antingen 2 eller 3.

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

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

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

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

csc() trig tangent

csc(*Expr1***)⇒***uttryck*

csc(*List1***)⇒***lista*

Ger cosekanten för *Expr1* eller en lista på cosekanten för alla element i *List1*.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

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

I vinkelläget Radianer:

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

csc⁻¹() trig tangent

csc⁻¹(*Expr1***)** ⇒ *uttryck*

csc⁻¹(*List1***)** ⇒ *lista*

Ger den vinkel vars cosekant är *Expr1* eller en lista på den inversa cosekanten för varje element i *List1*.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

csc⁻¹() — tangent

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **arccsc(...)**.

$\csc^{-1}(1)$ → 100

I vinkelläget Radianer:

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

csch() — Katalog >

csch(*Expr1***)** ⇒ *uttryck*

csch(*List1***)** ⇒ *lista*

Ger den hyperboliska cosekanten för *Expr1* eller en lista på den hyperboliska cosekanten för alla element i *List1*.

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

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

csch⁻¹() — Katalog >

csch⁻¹(*Expr1***)** ⇒ *uttryck*

csch⁻¹(*List1***)** ⇒ *lista*

Ger den inversa hyperboliska cosekanten för *Expr1* eller en lista på den inversa hyperboliska cosekanten för alla element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **arccsch(...)**.

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

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

cSolve() — Katalog >

cSolve(*Equation***,** *Var***)**⇒*Booleskt uttryck*

cSolve(*Equation***,** *Var*=*Guess***)**⇒*Booleskt uttryck*

cSolve(*Inequality***,** *Var***)**⇒*Booleskt uttryck*

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

$\text{solve}(x^3=-1, x)$ → $x=-1$

Ger möjliga komplexa lösningar på en ekvation eller olikhet för *Var*. Målet är att producera "kandidater" för alla reella och icke-reella lösningar. Även om *Equation* är reell medger **cSolve()** icke-reella resultat i Real Complex Format.

Även om alla odefinierade variabler som inte slutar med ett understrykningstecken (_) behandlas som om de vore reella kan **cSolve()** lösa polynomekvationer för komplexa lösningar.

cSolve() ställer temporärt in området på komplext under lösningen även om det aktuella området är reellt. I det komplexa området använder rationella potenser med udda nämnare principaldelen framför den reella delen. Följaktligen är lösningar från **solve()** på ekvationer som innehåller sådana rationella potenser inte nödvändigtvis deluppsättningar av lösningarna från **cSolve()**.

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

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

cSolve() börjar med exakta symboliska metoder. **cSolve()** använder om nödvändigt också iterativ ungefärlig komplex polynomfaktoruppdelning.

I läge Display Digits (Visa siffror) för Fix 2:

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

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

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

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

Obs: Se även **cZeros()**, **solve()** och **zeros()**.

Obs: Om *Equation* är ett icke-polynom med funktioner såsom **abs()**, **angle()**, **conj()**, **real()** eller **imag()** bör du placera ett understrykningstecken (tryck på ctrl ␣) i slutet på *Var*. Som förinställning behandlas en variabel som ett reellt värde.

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Om du använder *var_* behandlas variabeln som komplex.

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

Du bör också använda *var_* för övriga variabler i *Equation* som kan ha icke-reella värden. Annars kan du få oväntade resultat.

cSolve(*Eqn1* **and** *Eqn2* [**and…**], *VarOrGuess1*, *VarOrGuess2* [**, … **]) ⇒*Booleskt uttryck*

cSolve(*SystemOfEqns*, *VarOrGuess1*,

VarOrGuess2 [**, …**]) ⇒*Booleskt uttryck*

Ger möjliga komplexa lösningar på ekvationssystem där varje *VarOrGuess* specificerar en variabel som du vill lösa.

Du kan som alternativ specificera en initial gissning för en variabel. Varje *VarOrGuess* måste ha formen:

variable

– eller –

variable = *reellt eller icke-reellt tal*

Som exempel är x giltigt och likaså x=3+*i*.

Om alla ekvationer är polynom och om du INTE specificerar några initiala gissningar använder **cSolve()** eliminationsmetoden Gröbner/Buchberger för att försöka bestämma **alla** komplexa lösningar.

Komplexa lösningar kan innehålla både reella och icke-reella lösningar, som i exemplet till höger.

Obs: I följande exempel används ett understrykningstecken (tryck på ctrl ␣) så att variablerna behandlas som komplexa.

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Ekvationssystem som innehåller polynom kan ha extra variabler som saknar värden, men representerar givna numeriska värden som kan ersättas senare.

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Du kan också inkludera lösningsvariabler som inte visas i ekvationerna. Dessa lösningar visar hur familjer av lösningar kan innehålla godtyckliga konstanter med formen *c*k där *k* är ett heltalssuffix från 1 till och med 255.

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

cSolve() Katalog >

För polynomsystem kan beräkningstiden och användningen av minne i hög grad bero på i vilken ordning du listar lösningsvariabler. Om ditt första val utarmar minnet, eller tar på ditt tålamod, kan du försöka med att arrangera om variablerna i ekvationerna och/eller i listan *VarOrGuess*.

Om du inte inkluderar några gissningar och om någon ekvation är ett icke-polynom i någon variabel, men alla ekvationer är linjära i alla lösningsvariabler, använder **cSolve()** Gauss eliminationsmetod för att försöka bestämma alla lösningar.

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

Om ett system är varken polynomt i alla dess variabler eller linjärt i dess lösningsvariabler bestämmer **cSolve()** högst en lösning med en ungefärlig iterativ metod. För att göra detta måste antalet lösningsvariabler vara lika med antalet ekvationer och alla övriga variabler i ekvationerna måste förenklas till tal.

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

En icke-reell gissning är ofta nödvändig för att bestämma en icke-reell lösning. För konvergens kan en gissning behöva vara ganska nära en lösning.

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

CubicReg Katalog >

CubicReg *X*, *Y*[, [*Freq*] [, *Category*, *Include*]]

Utför en tredjegrads regressionsanalysy = a ·x3+b · x2+c ·x+dpå listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*, på sidan 177.

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot x^3+b \cdot x^2+c \cdot x+d$
stat.a, stat.b, stat.c, stat.d	Regressionskoefficienter
stat.R^2	Determinationskoefficient
stat.Resid	Residualer från regressionsanalysen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

cumulativeSum() Katalog >

cumulativeSum(*List1***)⇒***lista***

Ger en lista på de kumulativa summorna av elementen i *List1* och börjar med element 1.

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

cumulativeSum(*Matrix1***)⇒***matris***

Ger en matris över de kumulativa summorna av elementen i *Matrix1*. Varje element är den kumulativa summan i kolumnen räknat uppifrån och ned.

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

$$\text{cumulativeSum}(m1) \qquad \begin{bmatrix}1 & 2\\4 & 6\\9 & 12\end{bmatrix}$$

cumulativeSum() Katalog >

Ett tomt element i *List1* eller *Matrix1* ger ett tomt element i den resulterande listan eller matrisen. För mer information om tomma element, se på sidan 238.

Cycle Katalog >

Cycle

Överför omedelbart kontroll till nästa iteration i den aktuella slingan (**For**, **While** eller **Loop**).

Cycle tillåts inte utanför de tre slingstrukturerna (**For**, **While** eller **Loop**).

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Funktion som listar heltal från 1 till 100 utom 50.

```
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 Katalog >

Vector **▸Cylind**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>Cylind**.

Visar rad- eller kolumnvektorn i cylindrisk form [r,∠θ, z].

Vector måste ha exakt tre element. Den kan vara antingen en rad eller en kolumn.

```
[2  2  3]▸Cylind                    [2·√2  ∠π/4  3]
```

cZeros() Katalog >

cZeros(*Expr***,** *Var***)⇒***lista*

Ger en lista på möjliga reella och icke-reella värden på *Var* som gör *Expr*=0. **cZeros()** gör detta genom att beräkna **exp▸list(cSolve(***Expr*=0,*Var***),***Var***)**. Annars är **cZeros()** lika **zeros()**.

Obs: Se även **cSolve()**, **solve()** och **zeros()**.

I läge Display Digits (Visa siffror) för Fix 3:

```
cZeros(x^5+4·x^4+5·x^3−6·x−3,x)
{−1.1138+1.07314·i,−1.1138−1.07314·i,−2.▸
```

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

$\text{cZeros}(\text{conj}(z\_)-1-i,z\_) \quad \{1-i\}$

Obs: Om *Expr* är ett icke-polynom med funktioner såsom **abs()**, **angle()**, **conj()**, **real()** eller **imag()** bör du placera ett understrykningstecken (tryck på ctrl ␣) i slutet på *Var*. Som förinställning behandlas en variabel som ett reellt värde. Om du använder *var_* behandlas variabeln som komplex.

Du bör också använda *var_* för övriga variabler i *Expr* som kan ha icke-reella värden. Annars kan du få oväntade resultat.

cZeros({*Expr1***,** *Expr2* [**,** ...] **},**
{*VarOrGuess1***,***VarOrGuess2* [**,** ...]
})⇒*matris*

Ger möjliga positioner där uttrycken är noll samtidigt. Varje *VarOrGuess* specificerar en okänd vars värde du söker.

Du kan som alternativ specificera en initial gissning för en variabel. Varje *VarOrGuess* måste ha formen:

variable

– eller –

variable = reellt eller icke-reellt tal

Som exempel är x giltigt och likaså x=3+*i*.

Om alla ekvationer är polynom och om du INTE specificerar några initiala gissningar använder **cZeros()** eliminationsmetoden Gröbner/Buchberger för att försöka bestämma **alla** komplexa nollställen.

Komplexa nollställen kan innehålla både reella och icke-reella nollställen, som i exemplet till höger.

Obs: I följande exempel används ett understrykningstecken _ (tryck på ctrl ␣) så att variablerna behandlas som komplexa.

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

Varje rad i den resulterande matrisen representerar ett alternativt nollställe, med komponenterna ordnade på samma sätt som i listan *VarOrGuess*. För att extrahera en rad, indexera matrisen med [*row*].

Extrahera rad 2:

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

System av polynom kan ha extra variabler som saknar värden, men representerar givna numeriska värden som kan ersättas senare.

$$\text{cZeros}\left(\left\{u\_\cdot v\_-u\_-c\_\cdot v\_, v\_^2+u\_\right\}, \left\{u\_, v\_\right\}\right)$$

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

Du kan också inkludera okända variabler som inte visas i uttrycken. Dessa nollställen visar hur familjer av nollställen kan innehålla godtyckliga konstanter med formen *c*k, där k är ett heltalssuffix från 1 till och med 255.

$$\text{cZeros}\left(\left\{u\_\cdot v\_-u\_-v\_, v\_^2+u\_\right\}, \left\{u\_, v\_, w\_\right\}\right)$$

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

För polynomsystem kan beräkningstiden och användningen av minne i hög grad bero på i vilken ordning du listar okända. Om ditt första val utarmar minnet, eller tar på ditt tålamod, kan du försöka med att arrangera om variablerna i uttrycken och/eller i listan *VarOrGuess*.

Om du inte inkluderar några gissningar och om något uttryck är ett icke-polynom i någon variabel, men alla uttryck är linjära i alla okända, använder **cZeros()** Gauss eliminationsmetod för att försöka bestämma alla nollställen.

$$\text{cZeros}\left(\left\{u\_+v\_-e^{w\_}, u\_-v\_-i\right\}, \left\{u\_, v\_\right\}\right)$$

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

Om ett system är varken polynomt i alla dess variabler eller linjärt i dess okända bestämmer **cZeros()** högst ett nollställe med en ungefärlig iterativ metod. För att göra detta måste antalet okända vara lika med antalet uttryck och alla övriga variabler i uttrycken måste förenklas till tal.

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

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

En icke-reell gissning är ofta nödvändig för att bestämma ett icke-reellt nollställe. För konvergens kan en gissning behöva vara ganska nära ett nollställe.

$$\text{cZeros}\left(\left\{e^{z}-w\_, w\_-z\_^2\right\}, \left\{w\_, z\_=1+i\right\}\right)$$

$$\begin{bmatrix} 0.149606+4.8919\cdot i & 1.58805+1.54022\cdot i \end{bmatrix}$$

dbd()

Katalog >

dbd(*date1,date2***)⇒***värde*

Ger antalet dagar mellan *date1* och *date2* med dagräkningsmetoden.

date1 och *date2* kan vara tal eller listor på tal inom den normala kalendern. Om både *date1* och *date2* är listor måste de vara lika långa.

date1 och *date2* måste vara mellan 1950 och 2049.

Du kan mata in datumen i ett av två format. Decimalplaceringen skiljer sig mellan de två datumformaten.

MM.DDYY (format som ofta används i USA)

DDMM.YY (format som ofta används i Europa)

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

Katalog >

Expr1 **▸DD⇒***värde*

List1 **▸DD⇒***lista*

Matrix1 **▸DD⇒***matris*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>DD**.

Ger den decimala ekvivalenten till argumentet uttryckt i grader. Argumentet är ett tal, en lista eller en matris som tolkas av vinkelläget i nygrader, radianer eller grader.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

1▸DD	$\frac{9}{10}^{\circ}$

I vinkelläget Radianer:

(1.5)▸DD	85.9437°

▸Decimal

Katalog >

Expression1 **▸Decimal**⇒*uttryck*

List1 **▸Decimal**⇒*uttryck*

Matrix1 **▸Decimal**⇒*uttryck*

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Decimal`**.

Visar argumentet i decimal form. Operatorn kan endast användas i slutet av inmatningsraden.

$\frac{1}{3}$▸Decimal	0.333333

Define (Definiera)

Katalog >

Define *Var* **=** *Uttryck*

Define *Function*(*Param1*, *Param2*, ...) **=** *Uttryck*

Definierar variabeln *Var* eller den användardefinierade funktionen *Function*.

Parametrar såsom *Param1* utgör platshållare för att överföra argument till funktionen. När du anropar en användardefinierad funktion måste du ha argument (till exempel, värden eller variabler) som överensstämmer med parametrarna. När funktionen anropas beräknar den *Expression* med de givna argumenten.

Var och *Function* får inte vara namnet på en systemvariabel eller inbyggd funktion eller ett kommando.

Obs: Denna form av **Define** är ekvivalent med att exekvera uttrycket: *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 $h(x)=\text{when}(x<2,2\cdot x-3,-2\cdot x+3)$	*Done*
$h(-3)$	-9
$h(4)$	-5

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

Block

EndFunc

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

$g(3,-7)$	3

Define (Definiera) Katalog >

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

Block

EndPrgm

I denna form kan den användardefinierade funktionen eller programmet exekvera ett block av flera påståenden.

Block kan vara antingen ett enstaka påstående eller en serie av påståenden på separata rader. *Block* kan även inkludera uttryck och instruktioner (till exempel, **If**, **Then**, **Else** och **For**).

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Obs: Se även **Define LibPriv**, på sidan 52 och **Define LibPub**, på sidan 53.

```
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
```

Define LibPriv Katalog >

Define LibPriv *Var* = *Uttryck*

Define LibPriv *Function*(*Param1*, *Param2*, ...) = *Uttryck*

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

Block

EndFunc

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

Block

EndPrgm

Fungerar på samma sätt som **Define** förutom att en privat biblioteksvariabel, funktion eller program definieras. Privata funktioner och program visas inte i Katalogen.

Define LibPriv **Katalog >**

Obs: Se även **Define**, på sidan 51 och **Define LibPub**, på sidan 53.

Define LibPub **Katalog >**

Define LibPub *Var* **=** *Uttryck*

Define LibPub *Function*(*Param1*, *Param2*, ...) **=** *Uttryck*

Define LibPub *Function*(*Param1*, *Param2*, ...) **= Func**

Block

EndFunc

Define LibPub *Program*(*Param1*, *Param2*, ...) **= Prgm**

Block

EndPrgm

Fungerar på samma sätt som **Define** förutom att en allmän biblioteksvariabel, funktion eller program definieras. Allmänna funktioner och program visas i Katalogen när biblioteket har sparats och uppdaterats.

Obs: Se även **Define**, på sidan 51 och **Define LibPriv**, på sidan 52.

deltaList() **Se ΔList(), på sidan 103.**

deltaTmpCnv() **Se ΔtmpCnv(), på sidan 190.**

DelVar

Katalog >

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

DelVar *Var*.

Tar bort den specificerade variabeln eller variabelgruppen från minnet.

Om en eller flera variabler är låsta visar detta kommando ett felmeddelande och tar endast bort olåsta variabler. Se **unLock**, på sidan 199.

DelVar *Var*. tar bort alla led i variabelgruppen *Var*. variabelgrupp (till exempel, den statistiska *stat.nn*-resultaten eller variabler skapade med funktionen **LibShortcut()**). Punkten (.) i denna form av **DelVar**-kommandot begränsar kommandot till borttagning av en variabelgrupp: den enkla variabeln *Var* påverkas inte.

$2 \rightarrow a$ → 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() →

aa.a	"NUM"	""
aa.b	"NUM"	""
aa.c	"NUM"	""

DelVar *aa.* → *Done*

getVarInfo() → "NONE"

delVoid()

Katalog >

delVoid(*List1***)⇒***lista***

Ger en lista med innehållet i *List1* med alla tomma element borttagna.

För mer information om tomma element, se på sidan 238.

delVoid({1,void,3}) → {1,3}

derivative()

Se *d*(), på sidan 222.

deSolve()

Katalog >

deSolve(*1stOr2ndOrderODE***,** *Var***,** *depVar***) ⇒***en allmän lösning*

Ger en ekvation som explicit eller implicit specificerar en generell lösning på den ordinära differentialekvationen (ODE) av 1:a eller 2:a ordningen. I ODE:

- Använd en primsymbol (tryck på ?!▸) för

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

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

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

DelVar *temp* → *Done*

att beteckna förstaderivatan av den beroende variabeln med avseende på den oberoende variabeln.

- Använd två primsymboler för att beteckna motsvarande andraderivata.

Primsymbolen används endast för derivata inom deSolve(). Använd **d()** i övriga fall.

Den generella lösningen på en ekvation av 1:a ordningen innehåller en godtycklig konstant med formen *c*k, där k är ett heltalssuffix från 1 till och med 255. Lösningen på en ekvation av 2:a ordningen innehåller två sådana konstanter.

Använd **solve()** på en implicit lösning om du vill försöka konvertera den till en eller flera ekvivalenta explicita lösningar.

När du jämför dina resultat med lösningar i läroböcker eller egna lösningar för hand, tänk på att olika metoder inför godtyckliga konstanter på olika ställen i beräkningen, vilket kan ge olika generella lösningar.

deSolve(*1stOrderODE***and***initCond***,** *Var***,** *depVar***)** ⇒*en partikulärlösning*

Ger en partikulärlösning som uppfyller *1stOrderODE* och *initCond*. Detta är vanligen enklare än att bestämma en allmän lösning, ersätta initiala värden, lösa den godtyckliga konstanten och sedan ersätta värdet i den allmänna lösningen.

initCond är en ekvation med formen:

depVar (*initialIndependentValue*) = *initialDependentValue*

initialIndependentValue och *initialDependentValue* kan vara variabler såsom x0 och y0 som saknar lagrade värden. Implicit derivering kan underlätta verifieringen av implicita lösningar.

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

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

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

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

$\text{deSolve}\left(ode \text{ and } y\left(0\right)=0,x,y\right)\rightarrow soln$ $\quad \frac{-\left(2\cdot\sin\left(y\right)+y^2\right)}{2}=-\left(e^x-1\right)\cdot e^{-x}\cdot\sin\left(y\right)$

$soln|x=0$ and $y=0$ $\quad$ true

$ode|y'=\text{impDif}\left(soln,x,y\right)$ $\quad$ true

DelVar *ode*,*soln* $\quad$ *Done*

deSolve() Katalog >

deSolve
(*2ndOrderODE* **and** *initCond1* **and** *initCond2*, *Var*, *depVar***)⇒***en partikulärlösning*

Ger en partikulärlösning som uppfyller *2nd Order ODE* och har ett specificerat värde på den beroende variabeln och dess förstaderivata i en punkt.

För *initCond1*, använd formen:

depVar (*initialIndependentValue*) = *initialDependentValue*

För *initCond2*, använd formen:

depVar (*initialIndependentValue*) = *initial1stDerivativeValue*

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

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

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

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

deSolve
(*2ndOrderODE* **and** *bndCond1* **and** *bndCond2*, *Var*, *depVar***)⇒***en partikulärlösning*

Ger en partikulärlösning som uppfyller *2ndOrderODE* och har specificerade värden i två punkter.

$\text{deSolve}\left(y''=x \text{ and } y(0)=1 \text{ and } y'(2)=3, x, y\right)$

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

$\text{deSolve}\left(y''=2\cdot y' \text{ and } y(3)=1 \text{ and } y'(4)=2, x, y\right)$

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

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

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

det() Katalog >

det(*squareMatrix*[**,** *Tolerance*]**)⇒***uttryck*

Ger determinanten för *squareMatrix*.

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tolerance*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tolerance*.

- Om du använder ctrl enter eller ställer in **Auto eller Ungefärlig** på Approximate

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

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

$\det(mat1,.1)$ → $1.\text{E}20$

utförs beräkningarna med flyttalsaritmetik.

- Om *Tolerance* utelämnas eller inte används beräknas standardtoleransen som:

  5E-14 ·**max(dim(***squareMatrix***))** · **rowNorm(***squareMatrix***)**

diag()

Katalog >

diag(*List***)⇒***matris*

diag(*rowMatrix***)⇒***matris*

diag(*columnMatrix***)⇒***matris*

Ger en matris med värdena i argumentlistan eller matrisen i dess huvuddiagonal.

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

diag(*squareMatrix***)⇒***radMatris*

Ger en radmatris som innehåller elementen från huvuddiagonalen hos *squareMatrix*.

squareMatrix måste vara kvadratisk.

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

$$\text{diag}(Ans) \qquad \begin{bmatrix}4 & 2 & 9\end{bmatrix}$$

dim()

Katalog >

dim(*List***)⇒***heltal*

Ger dimensionen på *List*.

$$\dim\{0,1,2\} \qquad 3$$

dim(*Matrix***)⇒***lista*

Ger dimensionerna på en matris som en lista med två element {rader, kolumner}.

$$\dim\begin{bmatrix}1 & -1\\ 2 & -2\\ 3 & 5\end{bmatrix} \qquad \{3,2\}$$

dim(*String***)⇒***heltal*

Ger antalet tecken i teckensträngen *String*.

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

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

Disp

Katalog >

Disp *exprOrString1* [**,** *exprOrString2*] ...

Visar argumenten i *Calculator*-historiken. Argumenten visas i ordningsföljd med utslutning som separatorer.

Huvudsakligen användbart i program och funktioner för att säkerställa visningen av mellanliggande beräkningar.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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

Katalog >

Expr ▸*DMS*

List ▸**DMS**

Matrix ▸**DMS**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>DMS**.

Tolkar argumentet som en vinkel och visar motsvarande DMS-värde (DDDDDD°MM'SS.ss''). Se °, ', '', på sidan 230 för DMS-format (grad, minuter, sekunder).

Obs: ▸DMS konverterar från radianer till grader vid användning i läget radianer. Om inmatningen följs av en gradsymbol ° sker ingen konvertering. Du kan bara använda ▸**DMS** i slutet av en inmatningsrad.

I vinkelläget Grader:

```
(45.371)▸DMS                    45°22'15.6"
{{45.371,60}}▸DMS               {45°22'15.6",60°}
```

domain() (område)

Katalog >

domain(*Uttr1*, *Var*)⇒*uttryck*

Ger området *Uttr1* med avseende på *Var*.

domain() kan användas för att undersöka områden hos funktioner. Det är begränsat till reella och ändliga områden.

Denna funktionalitet har begränsningar på grund av brister i förenkling av datoriserad algebra och algoritmlösare.

Vissa funktioner kan inte användas som argument för **domain()**, oavsett om de visas explicit eller inom användardefinierade variabler och funktioner. I följande exempel kan inte uttrycket förenklas eftersom ∫() är en otillåten funktion.

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

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

dominantTerm()

Katalog >

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

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

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

Ger den dominanta termen i en potensserierepresentation av *Expr1* expanderat kring *Point*. Den dominanta termen är den term som snabbast ökar i storlek nära *Var* = *Point*. Den resulterande potensen av (*Var* − *Point*) kan ha en negativ och/eller exponent i bråkform. Koefficienten för denna potens kan inkludera logaritmer av (*Var* − *Point*) och andra funktioner av *Var* som domineras av alla potenser av (*Var* − *Point*) som har samma exponenttecken.

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

dominantTerm() Katalog >

Point förinställs till 0. *Point* kan vara ∞ eller −∞, varvid den dominanta termen kommer att vara den term som har den största exponenten av *Var* snarare än den minsta exponenten av *Var*.

dominantTerm(…) ger "**dominantTerm(…)**" om den inte kan bestämma en sådan representation, till exempel, för essentiella singulariteter såsom **sin(**1/*z***)** vid *z*=0, $e^{-1/z}$ vid z=0, eller e^{z} vid z = ∞ eller −∞.

Om serien eller någon av dess derivator har en språngdiskontinuitet vid *Point* innehåller resultatet troligen deluttryck i formen sign (…) eller abs(…) för en reell expansionsvariabel, eller $(-1)^{\text{floor(…angle(…)…)}}$ för en komplex expansionsvariabel, vilken slutar med "_". Om du tänker använda den dominanta termen endast för värden på ena sidan av *Point*, bifoga då till **dominantTerm(…)** den lämpliga av "| *Var* > *Point*", "| *Var* < *Point*", "| "*Var* ≥ *Point*" eller "*Var* ≤ *Point*" för att erhålla ett enklare resultat.

dominantTerm() fördelas över 1:a-argumentlistor och matriser.

dominantTerm() kan användas när du vill veta det enklaste uttrycket som är asymptotiskt med ett annat uttryck såsom *Var* → *Point*. **dominantTerm()** kan också användas när det inte är uppenbart vilken grad den första icke-nolltermen i en serie kommer att ha, och du inte vill gissa varken iterativt eller interaktivt eller använda en programslinga.

Obs: Se även **series()**, på sidan 161.

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

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

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

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

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

dotP() Katalog >

dotP(*List1***,** *List2***)⇒***uttryck*

Ger "prick"-produkten av två listor.

dotP(*Vector1***,** *Vector2***)⇒***uttryck*

Ger "prick"-produkten av två vektorer.

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

$$\text{dotP}(\{1,2\},\{5,6\}) \qquad 17$$

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

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

Båda måste vara radvektorer eller båda måste vara kolumnvektorer.

E

e^() — e^x-tangent

e^(*Expr1***)⇒***uttryck*

Ger **e** upphöjt till potensen *Expr1*.

Obs: Se även **e exponentmall**, på sidan 6.

Obs: Att trycka på [e^x] för att visa e^(skiljer sig från att trycka på tecknet [E] på tangentbordet.

Du kan skriva in ett komplext tal i den polära formen $re^{i\theta}$. Använd dock denna form endast i vinkelläget Radianer: den orsakar ett områdesfel i vinkelläget Grader eller Nygrader.

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

e^(*List1***)⇒***lista*

Ger **e** upphöjt till potensen för varje element i *List1*.

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

e^(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med exponenten för *squareMatrix1*. Detta är inte detsamma som att beräkna e upphöjt till potensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

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

eff() — Katalog >

eff(*nominalRate***,***CpY***)⇒***värde*

Finansiell funktion som konverterar den nominella räntan *nominalRate* till en årlig effektiv ränta, given av *CpY* som antalet ränteperioder per år.

eff(5.75,12)	5.90398

eff() Katalog >

nominalRate måste vara ett reellt tal och *CpY* måste vara ett reellt tal > 0.

Obs: Se även **nom()**, på sidan 124.

eigVc() Katalog >

eigVc(*squareMatrix***)⇒***matris*

Ger en matris som innehåller egenvektorerna för en reell eller komplex *squareMatrix* där varje kolumn i resultatet motsvarar ett egenvärde. Observera att en egenvektor inte är unik: den kan vara skalad med vilken konstant faktor som helst. Egenvektorerna är normaliserade, vilket innebär att om V = [x_1, x_2, ..., x_n], så är:

$x_1^2 + x_2^2 + ... + x_n^2 = 1$

squareMatrix balanseras först med liknande transformationer tills rad- och kolumnnormerna har så nära samma värde som möjligt. *squareMatrix* reduceras sedan till övre Hessenberg-form och egenvektorerna beräknas med en Schur-faktorisering.

I Rektangulärt komplext format:

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

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

eigVl() Katalog >

eigVl(*squareMatrix***)⇒***lista*

Ger en lista på egenvärdena för en reell eller komplex *squareMatrix*.

squareMatrix balanseras först med likhetstransformationer tills rad- och kolumnnormerna har så nära samma värde som möjligt. *squareMatrix* reduceras sedan till övre Hessenberg-form och egenvektorerna beräknas från den övre Hessenberg-matrisen.

I läget Rektangulärt komplext format:

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

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Else Se If, på sidan 88.

ElseIf

Katalog >

If *BooleanExpr1* **Then**

Block1

ElseIf *BooleanExpr2* **Then**

Block2

⋮

ElseIf *BooleanExprN* **Then**

BlockN

EndIf

⋮

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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

Se For, på sidan 76.

EndFunc

Se Func, på sidan 80.

EndIf

Se If, på sidan 88.

EndLoop

Se Loop, på sidan 111.

EndPrgm

Se Prgm, på sidan 139.

EndTry **Se Try, på sidan 192.**

EndWhile **Se While, på sidan 202.**

euler ()

Katalog >

euler(*Expr*, *Var*, *depVar*, {*Var0, VarMax*}, *depVar0*, *VarStep* [, *eulerStep*]**)** ⇒*matris*

euler(*SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *eulerStep*]**)** ⇒*matris*

euler(*ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}**x** *ListOfDepVars0*, *VarStep* [, *eulerStep*]**)** ⇒*matris*

Använder Eulers metod för att lösa systemet

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

med *depVar*(*Var0*)=*depVar0* på intervallet [*Var0*,*VarMax*]. Ger en matris vars första rad definierar resultatvärdena för *Var* och vars andra rad definierar den första lösningskomponenten vid motsvarande *Var*-värden, och så vidare.

Expr är det högra ledet som definierar den ordinära differentialekvationen (ODE).

SystemOfExpr är systemet av högerled som definierar systemet av ODE:er (motsvarar ordningen av oberoende variabler i *ListOfDepVars*).

ListOfExpr är en lista på högerled som definierar systemet av ODE:er (motsvarar ordningen av oberoende variabler i *ListOfDepVars*).

Var är den oberoende variabeln.

ListOfDepVars är en lista på oberoende variabler.

Differentialekvation:

y'=0,001*y*(100-y) och y(0)=10

$$\text{euler}(0.001\cdot y\cdot(100-y),t,y,\{0,100\},10,1)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9 & 11.8712 & 12.9174 & 14.042 \end{bmatrix}$$

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Jämför ovanstående resultat med CAS exakta lösning som erhållits med deSolve() och seqGen():

$$\text{deSolve}(y'=0.001\cdot y\cdot(100-y) \text{ and } y(0)=10,t,y)$$

$$y=\frac{100.\cdot(1.10517)^t}{(1.10517)^t+9.}$$

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

$$\{10.,10.9367,11.9494,13.0423,14.2189\}$$

Ekvationssystem:

$$\begin{cases} y1'=-y1+0.1\cdot y1\cdot y2 \\ y2'=3\cdot y2-y1\cdot y2 \end{cases}$$

med $y1(0)=2$ och $y2(0)=5$

euler ()

Katalog >

{*Var0*, *VarMax*} är en lista med två element som instruerar funktionen att integrera från *Var0* till *VarMax*.

ListOfDepVars0 är en lista på startvärden för oberoende variabler.

VarStep är ett tal skilt från noll så att **sign (***VarStep***)** = **sign(***VarMax-Var0***)** och lösningar ges vid *Var0*+i·*VarStep* för alla i=0,1,2,... sådana att *Var0*+i·*VarStep* är i intervallet [*var0*,*VarMax*] (det kanske inte finns ett lösningsvärde vid *VarMax*).

eulerStep är ett positivt heltal (förinställs på 1) som definierar antalet euler-steg mellan resultatvärden. Den verkliga stegstorleken som används av euler-metoden är *VarStep*/*eulerStep*.

$$\text{euler}\left\{\begin{bmatrix} -y1+0.1\cdot y1\cdot y2 \\ 3\cdot y2-y1\cdot y2 \end{bmatrix}, t, \{y1, y2\}, \{0,5\}, \{2,5\}, 1\right\}$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. & 5. \\ 2. & 1. & 1. & 3. & 27. & 243. \\ 5. & 10. & 30. & 90. & 90. & -2070. \end{bmatrix}$$

eval ()

Hubb-meny

eval(*Expr***)** ⇒ *string*

eval() är giltig endast i TI-Innovator™ Hub Kommandoargument i programmeringskommandona **Get**, **GetStr** och **Send**. Programmet beräknar uttrycket *Expr* och ersätter **eval()**-meddelandet med resultatet som en teckensträng.

Argumentet *Expr* måste generera ett reellt tal.

Ställ in den blå komponenten hos RGB-lysdioden på halv intensitet.

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

Återställ den blå komponenten till OFF.

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

Argumentet eval() måste generera ett reellt tal.

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

Program för att tona in den röda komponenten

eval ()

Hubb-meny

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

Kör programmet.

```
fadein()                                   Done
```

Även om **eval()** inte visar dess resultat kan du visa den resulterande Hubb-kommandosträngen efter att ha kört kommandot genom att kontrollera någon av följande speciella variabler.

iostr.SendAns
iostr.GetAns
iostr.GetStrAns

Obs: Se även **Get** (på sidan 82), **GetStr** (på sidan 85), och **Send** (på sidan 159).

```
n:=0.25                                    0.25
m:=8                                       8
n·m                                        2.
Send "SET COLOR.BLUE ON TIME eval(n·m)"
                                           Done
iostr.SendAns    "SET COLOR.BLUE ON TIME 2"
```

exact()

Katalog >

exact(*Expr1* [**,** *Tolerance*]**)⇒***uttryck*

exact(*List1* [**,** *Tolerance*]**)⇒***lista*

exact(*Matrix1* [**,** *Tolerance*]**)⇒***matris*

Använder det aritmetiska läget Exact för att, när så är möjligt, ge argumentet som ett rationellt tal.

Tolerance specificerar konverteringens tolerans: förinställningen är 0 (noll).

```
exact(0.25)                                1/4
exact(0.333333)                            333333/1000000
exact(0.333333,0.001)                      1/3
exact(3.5·x+y)                             7·x/2+y
exact({0.2,0.33,4.125})                    {1/5, 33/100, 33/8}
```

Exit

Katalog >

Exit

Avslutar det aktuella blocket **For**, **While** eller **Loop**.

Funktionslista:

Exit

Katalog >

Exit tillåts inte utanför de tre slingstrukturerna (**For**, **While** eller **Loop**).

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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 >

Expr ▸**exp**

Representerar *Expr* i termer av basen för den naturliga logaritmen *e*. Detta är en omvandlingsoperator för visning. Den kan endast användas i slutet av inmatningsraden.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>exp**.

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

$$2\cdot\sinh(x)\blacktriangleright\exp \qquad e^{x}-e^{-x}$$

exp()

e^{x}-tangent

exp(*Expr1***)⇒***uttryck*

Ger **e** upphöjt till potensen *Expr1*.

Obs: Se även **e** exponentmall, på sidan 6.

$$e^{1} \qquad e$$

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

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

Du kan skriva in ett komplext tal i den polära formen $re^{i\theta}$. Använd dock denna form endast i vinkelläget Radianer: den orsakar ett områdesfel i vinkelläget Grader eller Nygrader.

exp(*List1***)⇒***lista*

Ger **e** upphöjt till potensen för varje element i *List1*.

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

exp(*squareMatrix1***)⇒***kvadratMatris*

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

exp()

e^x-tangent

Ger matrisen med exponenten för *squareMatrix1*. Detta är inte detsamma som att beräkna **e** upphöjt till potensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

exp▸list()

Katalog >

exp▸list(*Expr*,*Var*)⇒*lista*

Undersöker *Expr* för ekvationer som är separerade med ordet "eller," och ger en lista med de högra sidorna av ekvationerna med formen *Var*=*Expr*. Detta ger dig en enkel metod att extrahera vissa lösningsvärden i resultaten från funktionerna **solve()**, **cSolve()**, **fMin()** och **fMax()**.

Obs: exp▸list() behövs inte med funktionerna **zeros** och **cZeros()** eftersom de direkt ger en lista på lösningsvärden.

Du kan infoga denna funktion med datorns tangentbord genom att skriva **exp@>list(...)**.

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

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

expand()

Katalog >

expand(*Expr1* [**,** *Var*])⇒*uttryck*

expand(*List1* [**,***Var*])⇒*lista*

expand(*Matrix1* [**,***Var*])⇒*matris*

expand(*Expr1***)** ger *Expr1* utvecklat med avseende på alla dess variabler. Utvecklingen är en polynomutveckling för polynom och partiell bråkutveckling för rationella uttryck.

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

$x^2+2\cdot x\cdot y+2\cdot x+y^2+2\cdot y+1$

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

$\frac{1}{x-1}-\frac{1}{x}+\frac{1}{y-1}-\frac{1}{y}$

Målsättningen med **expand()** är att transformera *Expr1* till en summa av och/eller skillnad mellan enkla termer. Som kontrast är målsättningen med **factor()** att transformera *Expr1* till en produkt av och/eller kvot mellan enkla faktorer.

expand(*Expr1*,*Var***)** ger *Expr1* utvecklat med avseende på *Var*. Liknande potenser av *Var* samlas in. Termerna och deras faktorer sorteras med *Var* som huvudvariabel. Viss tillfällig faktorisering eller utveckling kan ske av de insamlade koefficienterna. Jämfört med att utesluta *Var* sparar detta ofta tid, minne och skärmutrymme, vilket gör uttrycket mer begripligt.

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

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

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

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

Även med endast en variabel kan användning av *Var* göra den faktorisering av nämnare som används för partialbråksutveckling mer fullständig.

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

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

Tips: För rationella uttryck är **propFrac()** ett snabbare, men mindre extremt, alternativ till **expand()**.

Obs: Se även **comDenom()** för en expanderad täljare genom en expanderad nämnare.

expand(*Expr1*,[*Var*]**)** fördelar också logaritmer och rationella potenser oberoende av *Var*. För ökad fördelning av logaritmer och rationella potenser kan olikhetsbegränsningar vara nödvändiga för att säkerställa att vissa faktorer är naturliga tal.

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

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

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

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

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

expand(*Expr1*, [*Var*]**)** fördelar också absolutvärden, **sign()** och exponentialer oberoende av *Var*.

Obs: Se även **tExpand()** för utveckling av trigonometriska utryck.

expr()

Katalog >

expr(*String***)⇒***uttryck*

Ger teckensträngen i *String* som ett uttryck och exekverar det omedelbart.

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

Katalog >

ExpReg *X*, *Y* [, [*Freq*] [, *Category*, *Include*]]

Beräknar den exponentiella regressioneny = a ·(b)ˣpå listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a ·(b)ˣ
stat.a, stat.b	Regressionskoefficienter

Resultatvariabel	Beskrivning
stat.r^2	Koefficient för linjär bestämning av transformerade data
stat.r	Korrelationskoefficient för transformerade data (x, ln(y))
stat.Resid	Residualer associerade med den exponentiella modellen
stat.ResidTrans	Residualer associerade med linjär anpassning av transformerade data
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

F

factor() Katalog >

factor(*Expr1*[**,** *Var*]**)⇒***uttryck*

factor(*List1*[**,***Var*]**)⇒***lista*

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

factor(*Expr1***)** ger en faktorisering av *Expr1* baserad på uttryckets alla variabler med en gemensam nämnare.

Expr1 faktoriseras så långt det går till linjära, rationella faktorer utan att införa nya icke-reella deluttryck. Detta alternativ är lämpligt om du vill ha en faktorisering baserad på mer än en variabel.

factor(*Expr1*,*Var***)** ger en faktorisering av *Expr1* baserad på variabeln *Var*.

Expr1 faktoriseras så långt det går till reella faktorer som är linjära i *Var*, även om detta inför irrationella konstanter eller deluttryck som är irrationella i andra variabler.

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

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

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

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

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

Faktorerna och deras termer sorteras med *Var* som huvudvariabel. Liknande potenser av *Var* samlas in i varje faktor. Inkludera *Var* om faktorisering baserad på endast denna variabel behövs och du är villig att acceptera irrationella uttryck i andra variabler för att öka faktoriseringen baserad på *Var*. Viss tillfällig faktorisering kan ske vad gäller andra variabler.

Med inställningen Auto i läge **Auto eller Ungefärlig** medger inkludering av *Var* en uppskattning med koefficienter i flyttalsform när irrationella koefficienter inte explicit kan uttryckas kortfattat med termerna i de inbyggda funktionerna. Även med endast en variabel kan inkludering av *Var* ge en mer fullständig faktorisering.

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

$\text{factor}(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3,x)$
$(x-0.964673)\cdot(x+0.611649)\cdot(x+2.12543)\cdot(x$

Obs: Se även **comDenom()** för ett snabbt sätt att erhålla partiell faktorisering när **factor()** inte är snabb nog eller om den utarmar minnet.

Obs: Se även **cFactor()** för faktorisering hela vägen till komplexa koefficienter i sökningen efter linjära faktorer.

factor(*rationalNumber***)** ger det rationella talet faktoriserat i primtal. För sammansatta tal ökar beräkningstiden exponentiellt med antalet siffror i den näst största faktorn. Som exempel kan faktorisering av ett 30-siffrigt heltal ta mer än en dag och faktorisering av ett 100-siffrigt tal kan ta mer än 100 år.

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

För att stoppa en beräkning manuellt,

- **Handenhet:** Håll ned [on] och tryck på [enter] upprepade gånger.
- **Windows®:** Håll ned **F12** och tryck på **Enter** upprepade gånger.
- **Macintosh®:** Håll ned **F5** och tryck på **Enter** upprepade gånger.
- **iPad®:** Appen visar en uppmaning. Du kan fortsätta att vänta eller avbryta.

factor()

Katalog >

Om du endast vill bestämma om ett tal är ett primtal, använd **isPrime()** i stället. Detta går mycket fortare, särskilt om *rationalNumber* inte är ett primtal och om den näst största faktorn har mer än fem siffror.

FCdf()

Katalog >

FCdf
(
lowBound
,*upBound*,*dfNumer*,*dfDenom***)**⇒*number* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

FCdf
(
lowBound
,*upBound*,*dfNumer*,*dfDenom***)**⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

Beräknar sannolikheten för F -fördelningen mellan *undre gräns* och *övre gräns* för det specificerade *dfNämn* (frihetsgrader) och *dfTälj*.

För P($X \le$ *övrGräns*), ange *undrGräns* = 0.

Fill

Katalog >

Fill *Expr, matrixVar*⇒*matris*

Ersätter varje element i variabeln *matrixVar* med *Expr*.

matrixVar måste redan existera.

Fill *Expr, listVar*⇒*lista*

Ersätter varje element i variabeln *listVar* med *Expr*.

listVar måste redan existera.

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

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*[,[*Freq*][,*Category*,*Include*]]

Ger en förkortad version av envariabelstatistiken för listan *X*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

X representerar en lista på aktuella data.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*-värde. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Ett tomt element i någon av listorna *X*, *Freq* eller *Category* resulterar i ett tomrum för motsvarande element i dessa listor. För mer information om tomma element, se på sidan 238.

Resultatvariabel	Beskrivning
stat.MinX	Minsta x-värde
stat.Q_1X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q_3X	Övre kvartil för x
stat.MaxX	Största x-värde

floor()

Katalog >

floor(*Expr1***)**⇒*heltal*

floor(−2.14) −3.

Ger det största heltal som är ≤ argumentet. Denna funktion är identisk med **int()**.

Argumentet kan vara ett reellt eller ett komplext tal.

floor() Katalog >

floor(*List1***)⇒***lista***

floor(*Matrix1***)⇒***matris***

Ger en lista eller matris med golvvärden för varje element.

Obs: Se även **ceiling()** och **int()**.

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

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

fMax() Katalog >

fMax(*Expr, Var***)⇒***Booleskt uttryck***

fMax(*Expr***,** *Var***,***lowBound***)**

fMax(*Expr***,** *Var***,***lowBound***,***upBound***)**

fMax(*Expr***,** *Var***) | ***lowBound*≤*Var*≤*upBound*

Ger ett booleskt uttryck som specificerar möjliga värden på *Var* som maximerar *Expr* eller lokaliserar dess lägsta övre gräns.

Du kan använda ("|")-operatorn begränsning för att begränsa lösningsintervallet och/eller specificera andra begränsningar.

Med inställningen Approximate i läge **Auto eller Ungefärlig** söker **fMax()** iterativt efter ett ungefärligt lokalt maximum. Detta går ofta fortare, särskilt om du använder operatorn "|" för att begränsa sökningen till ett relativt litet intervall som innehåller exakt ett lokalt maximum.

Obs: Se även **fMin()** och **max()**.

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

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

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

fMin() Katalog >

fMin(*Expr, Var***)⇒***Booleskt uttryck***

fMin(*Expr***,** *Var***,***lowBound***)**

fMin(*Expr***,** *Var***,***lowBound***,***upBound***)**

fMin(*Expr***,** *Var***) | ***lowBound*≤*Var*≤*upBound*

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

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

fMin()

Katalog >

Ger ett booleskt uttryck som specificerar möjliga värden på *Var* som minimerar *Expr* eller lokaliserar dess största nedre gräns.

Du kan använda ("|")-operatorn begränsning för att begränsa lösningsintervallet och/eller specificera andra begränsningar.

Med inställningen Approximate i läge **Auto eller Ungefärlig** söker **fMin()** iterativt efter ett ungefärligt lokalt minimum. Detta går ofta fortare, särskilt om du använder operatorn "|" för att begränsa sökningen till ett relativt litet intervall som innehåller exakt ett lokalt minimum.

Obs: Se även **fMax()** och **min()**.

For

Katalog >

For *Var*, *Low*, *High* [, *Step*]

Block

EndFor

Exekverar iterativt påståendena i *Block* för varje värde på *Var*, från *Low* till *High*, i steg enligt *Step*.

Var får inte vara en systemvariabel.

Step kan vara positivt eller negativt. Det förinställda värdet är 1.

Block kan vara antingen ett enstaka påstående eller en serie av påståenden separerade med tecknet ":".

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

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

g()                                5050
```

format() Katalog >

format(*Expr*[, *formatString*]**)⇒***sträng*

Ger *Expr* som en teckensträng baserad på formatmallen.

Expr måste förenklas till ett tal.

formatString är en sträng och måste ha formen: “F[n]”, “S[n]”, “E[n]”, “G[n][c]”, där [ ] indikerar frivilliga delar.

F[n]: Fast format. n är antalet siffror som skall visas efter decimalpunkten.

S[n]: Scientific format (Grundpotensform). n är antalet siffror som skall visas efter decimalpunkten.

E[n]: Engineering format. n är antalet siffror efter den första signifikanta siffran. Exponenten justeras till en multipel av tre och decimalpunkten flyttas åt höger med noll, en eller två siffror.

G[n][c]: Samma som fast format, men separerar också siffor till vänster om basen i grupper om tre. c specificerar det gruppseparerande tecknet och är förinställt på kommatecken. Om c är en punkt visas basen som ett kommatecken.

[Rc]: Samtliga ovanstående specifikationssymboler kan förses med suffix med Rc-basflaggan, där c är ett enstaka tecken som specificerar vad som skall ersättas för baspunkten.

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"

fPart() Katalog >

fPart(*Expr1***)⇒***uttryck*

fPart(*List1***)⇒***lista*

fPart(*Matrix1***)⇒***matris*

Ger argumentets bråkdel.

Ger, för en lista eller matris, elementens bråkdelar.

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

fPart()

Katalog >

Argumentet kan vara ett reellt eller ett komplext tal.

FPdf()

Katalog >

FPdf(*XVal*,*dfNumer*,*dfDenom***)**⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar sannolikheten för F-fördelning vid *XVal* för specificerad *dfNumer* (frihetsgrader) och *dfDenom*.

freqTable▸list()

Katalog >

freqTable▸list
(*List1*,*freqIntegerList***)**⇒*lista*

Ger en lista som innehåller elementen från *List1* expanderad enligt frekvenserna i *freqIntegerList*. Denna funktion kan användas för att skapa en frekvenstabell för applikationen Data & Statistik.

List1 kan vara vilken giltig lista som helst.

freqIntegerList måste ha samma dimensioner som *List1* och får endast innehålla icke-negativa heltalselement. Varje element specificerar antalet gånger motsvarande *List1*-element kommer att upprepas i resultatlistan. Ett värde på noll utesluter motsvarande *List1*-element.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`freqTable@>list(...)`**.

Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

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})
{1,2,2,2,4}

frequency()

Katalog >

frequency(*List1*,*binsList***)**⇒*lista*

Ger en lista med antalet element i *List1*. Talen baseras på områden (bins = staplar) som du definierar i *binsList*.

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}

frequency() Katalog >

Om *binsList* är {b(1), b(2), …, b(n)} är de specificerade områdena {**?**≤b(1), b(1)<**?**≤b(2),…,b(n-1)<**?**≤b(n), b(n)>**?**}. Den resulterande listan är ett element längre än *binsList*.

Varje element i resultatet motsvarar antalet element från *List1* som är i området för denna stapel. Uttryckt enligt funktionen **countIf()** är resultatet { countIf (list, **?**≤b(1)), countIf(list, b(1)<**?**≤b(2)), …, countIf(list, b(n-1)<**?**≤b(n)), countIf(list, b(n)>**?**)}.

Element i *List1* som inte kan "placeras i en stapel" ignoreras. Även tomma element ignoreras. För mer information om tomma element, se på sidan 238.

I applikationen Listor och kalkylblad kan du använda ett område av celler i stället för båda argumenten.

Obs: Se även **countIf()**, på sidan 39.

Förklaring av resultat:

2 element från *Datalist* är ≤2.5

4 element från *Datalist* är >2.5 och ≤4.5

3 element från *Datalist* är >4.5

Elementet "hello" är en sträng och kan inte placeras i någon av de definierade staplarna.

FTest_2Samp Katalog >

FTest_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*]]]

FTest_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*]]]

(Indatalista)

FTest_2Samp *sx1*,*n1*,*sx2*,*n2*[,*Hypoth*]

FTest_2Samp *sx1*,*n1*,*sx2*,*n2*[,*Hypoth*]

(Summary stats indata)

Utför ett 2-sampel F test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

eller H_a: σ1 > σ2, sätt *Hypoth*>0

För H_a: σ1 ≠ σ2 (förinställning), sätt *Hypoth* =0

För H_a: σ1 < σ2, sätt *Hypoth*<0

FTest_2Samp

Katalog >

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.F	Beräknad Û-statistik för datasekvensen
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.dfNumer	täljare, frihetsgrader = n1-1
stat.dfDenom	nämnare, frihetsgrader = n2-1
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i *List 1* och *List 2*
stat.x1_bar stat.x2_bar	Medelvärden hos urvalet i datasekvenserna i *List 1* och *List 2*
stat.n1, stat.n2	Storlek på urvalen

Func

Katalog >

Func

Block

EndFunc

Mall för att skapa en användardefinierad funktion.

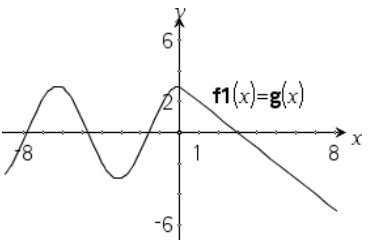
Block kan vara ett enstaka påstående, en serie av påståenden separerade med tecknet “:” eller en serie av påståenden på separata rader. Funktionen kan använda instruktionen **Return** för att ge ett specifikt resultat.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Definiera en stegvis funktion:

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

Resultat från plottning av g(x)



gcd() Katalog >

gcd(*Value1, Value2***)⇒***uttryck*

Ger den största gemensamma delaren för de två argumenten. **gcd** för två bråk är **gcd** för deras täljare dividerat med **lcm** för deras nämnare.

I läge Auto eller Approximate (Ungefärlig) är **gcd** 1.0 för bråktal i flyttalsform.

$\gcd(18,33)$ → 3

gcd(*List1, List2***)⇒***lista*

Ger största gemensamma delare för motsvarande element i *List1* och *List2*.

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

gcd(*Matrix1, Matrix2***)⇒***matris*

Ger största gemensamma delare för motsvarande element i *Matrix1* och *Matrix2*.

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

geomCdf() Katalog >

geomCdf(*p***,***lowBound***,***upBound***)⇒***tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

geomCdf(*p***,***upBound***)** för P(1≤X ≤*upBound*)**⇒***tal* om *upBound* är ett tal, *lista* om *upBound* är en lista

Beräknar en kumulativ geometrisk sannolikhet från *lowBound* till *upBound* med den specificerade sannolikheten *p* för att lyckas.

För P(X ≤ *upBound*), sätt *lowBound* = 1.

geomPdf() Katalog >

geomPdf(*p***,***XVal***)⇒***tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

geomPdf()

Katalog >

Beräknar en sannolikhet vid *XVal*, vid vilket försök i försöksomgången som man lyckas första gången, för den diskreta geometriska fördelningen med den specificerade sannolikheten p för att lyckas.

Get

Hubb-meny

Get[*promtString*,] *var*[, *statusVar*]

Get[*promtString*,] *func*(*arg1*, *...argn*) [, *statusVar*]

Programmeringskommando: Hämtar ett värde från en ansluten TI-Innovator™ Hub och tilldelar värdet till variabeln *var*.

Värdet måste begäras:

- På förhand genom ett **Send "READ ..."** -kommando.

  — eller —

- Genom att bädda in en **"READ ..."** -begäran som alternativt *promptString*-argument. Med denna metod kan du använda ett enda kommando för att begära värdet och hämta det.

Implicit förenkling äger rum. Till exempel tolkas en mottagen sträng ”123” som ett numeriskt värde. För att bevara strängen, använd **GetStr** istället för **Get**.

Om du inkluderar det valfria argumentet *statusVar*, tilldelas det ett värde baserat på om operationen har lyckats. Värdet noll betyder att inga data mottogs.

I den andra syntaxen kan ett program använda argumentet *func*() för att lagra den mottagna strängen som en funktionsdefinition. Denna syntax fungerar som om programmet exekverade kommandot:

Define *func*(*arg1*, *...argn*) = *received string*

Exempel: Begär nuvarande värde från hubbens inbyggda ljusnivåsensor. Använd **Get** för att hämta värdet och tilldela det till variabeln *lightval*.

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

Bädda in READ-begäran i **Get**-kommandot.

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

Get

Hubb-meny

Programmet kan sedan använda den definierade funktionen *func*().

Obs: Du kan använda kommandot **Get** i ett användardefinierat program, men inte i en funktion.

Obs: Se även **GetStr**, på sidan 85 och **Send**, på sidan 159.

getDenom()

Katalog >

getDenom(*Expr1***)⇒***uttryck*

Transformerar argumentet till ett uttryck med reducerad gemensam nämnare och ger sedan dess nämnare.

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

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

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

getLangInfo()

Katalog >

getLangInfo()⇒*sträng*

Ger en sträng som motsvarar det korta namnet på det aktuella aktiva språket. Du kan exempelvis använda den i ett program eller i en funktion för att bestämma det aktuella språket.

getLangInfo() → "en"

Engelska = “en”
Danska = “da”
Tyska = “de”
Finska = “fi”
Franska = “fr”
Italienska = “it”
Holländska = “nl”
Belgisk holländska = “nl_BE”
Norska = “no”
Portugisiska = “pt”
Spanska = “es”
Svenska = “sv”

getLockInfo() Katalog >

getLockInfo(*Var***)⇒***värde*

Ger den aktuella låsta/olåsta statusen för variabeln *Var*.

värde = **0**: *Var* är olåst eller finns inte.

värde = **1**: *Var* är låst och kan inte modifieras eller tas bort.

Se **Lock**, på sidan 107 och**unLock**, på sidan 199.

a:=65	65
Lock *a*	*Done*
getLockInfo(*a*)	1
a:=75	"Error: Variable is locked."
DelVar *a*	"Error: Variable is locked."
Unlock *a*	*Done*
a:=75	75
DelVar *a*	*Done*

getMode() Katalog >

getMode(*ModeNameInteger***)⇒***värde*

getMode(0)⇒*lista*

getMode(*ModeNameInteger***)** ger ett värde som representerar den aktuella lägesinställningen för *ModeNameInteger*.

getMode(0) ger en lista på talpar. Varje par består av ett lägesheltal och ett inställningsheltal.

Se nedan för en lista på lägen och deras inställningar.

Om du sparar inställningarna med **getMode(0)** → *var* kan du använda **setMode(***var***)** i en funktion eller ett program för att temporärt återställa inställningarna endast inom exekveringen av funktionen eller programmet. Se **setMode()**, på sidan 162.

getMode(0)	{1,7,2,1,3,1,4,1,5,1,6,1,7,1,8,1}
getMode(1)	7
getMode(8)	1

Lägets namn	Lägesheltal	Heltal för inställningar
Display Digits (Antal siffror)	1	**1**=Float, **2**=Float1, **3**=Float2, **4**=Float3, **5**=Float4, **6**=Float5, **7**=Float6, **8**=Float7, **9**=Float8, **10**=Float9, **11**=Float10, **12**=Float11, **13**=Float12, **14**=Fix0, **15**=Fix1, **16**=Fix2, **17**=Fix3, **18**=Fix4, **19**=Fix5, **20**=Fix6, **21**=Fix7, **22**=Fix8, **23**=Fix9, **24**=Fix10, **25**=Fix11, **26**=Fix12
Angle (Vinkel)	2	**1**=Radian, **2**=Degree, **3**=Gradian

Lägets namn	Läges-heltal	Heltal för inställningar
Exponential Format (Exponentiellt format)	3	**1**=Normal, **2**=Scientific, **3**=Engineering
Real or Complex (Reellt eller Komplext)	4	**1**=Real, **2**=Rectangular, **3**=Polar
Auto or Approx. (Auto eller Ungefärlig)	5	**1**=Auto, **2**=Approximate, **3**=Exact
Vector Format (Vektorformat)	6	**1**=Rectangular, **2**=Cylindrical, **3**=Spherical
Base (Bas)	7	**1**=Decimal, **2**=Hex, **3**=Binary
Unit System (Enhetssystem)	8	**1**=SI, **2**=Eng/US

getNum() Katalog >

getNum(*Expr1***)⇒***uttryck*

Transformerar argumentet till ett uttryck med reducerad gemensam nämnare och ger sedan dess täljare.

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

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

$\text{getNum}\left(\frac{1}{x}+\frac{1}{y}\right)$ $x+y$

GetStr Hubb-meny

GetStr[*promtString*,] *var*[**,** *statusVar*]

GetStr[*promtString*,] *func*(*arg1*, *...argn*) [**,** *statusVar*]

Programmeringskommando: Fungerar precis som kommandot **Get**, förutom att det mottagna värdet alltid tolkas som en sträng. I motsats tolkar kommandot **Get** svaret som ett uttryck såvida det inte är omgivet av citationstecken ("").

Obs: Se även **Get**, på sidan 82 och **Send**, på sidan 159.

Se **Get** för exempel.

getType() Katalog >

getType(*var***)⇒***sträng*

Ger en sträng som anger datatypen för variabeln *var*.

Om *var* inte har definierats erhålls strängen "NONE".

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

getVarInfo() Katalog >

getVarInfo()⇒*matris* eller *sträng*

getVarInfo(*LibNameString***)⇒***matris* eller *sträng*

getVarInfo() ger en matris med information (variabelnamn, typ, åtkomlighet till bibliotek och låst/olåst status) för alla variabler och biblioteksobjekt som är definierade i det aktuella problemet.

Om inga variabler är definierade ger **getVarInfo()** strängen "NONE".

getVarInfo(*LibNameString***)**ger en matris med information för alla biblioteksobjekt som är definierade i biblioteket *LibNameString*. *LibNameString* måste vara en sträng (text omsluten med citationstecken) eller en strängvariabel.

Om biblioteket *LibNameString* inte finns uppstår ett fel.

Se exemplet till vänster där resultatet av **getVarInfo()** tilldelas variabeln *vs*. Ett försök att visa rad 2 eller rad 3 av *vs* ger ett "Ogiltig lista eller matris"-fel eftersom minst ett av elementen i dessa rader (till exempel, variabel *b*) omvärderas till en matris.

Detta fel kan också inträffa när *Ans* används för att utvärdera ett **getVarInfo()**-resultat på nytt.

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

getVarInfo() Katalog >

Systemet ger ovanstående fel eftersom den aktuella versionen av programvaran inte stöder en generaliserad matrisstruktur där ett element i en matris kan vara antingen en matris eller en lista.

Goto Katalog >

Goto *labelName*

Överför kontroll till etiketten *labelName*.

labelName måste definieras i samma funktion med en **Lbl**-instruktion.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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 Katalog >

Expr1 ▸ **Grad**⇒*uttryck*

Konverterar *Expr1* till en vinkelmätning i nygrader.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Grad`**.

I vinkelläget Grader:

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

I vinkelläget Radianer:

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

I

identity() Katalog >

identity(*Integer***)**⇒*matris*

Ger enhetsmatrisen med ett mått på *Integer* (Heltal).

Integer måste vara positivt heltal.

```
identity(4)
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
```

If *BooleanExpr*

Statement

If *BooleanExpr* **Then**

Block

EndIf

If *BooleanExpr* utvärderar till sant och exekverar sedan det enstaka påståendet *Statement* eller blocket av påståenden *Block* innan exekveringen fortsätter.

If *BooleanExpr* utvärderar till falskt och fortsätter exekveringen utan att exekvera påståendet eller blocket av påståenden.

Block kan vara antingen ett enstaka påstående eller en serie av påståenden separerade med tecknet ":".

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

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

$g(-2)$	4

If *BooleanExpr* **Then**

Block1

Else

Block2

EndIf

If *BooleanExpr* utvärderar till sant, exekverar *Block1* och hoppar sedan över *Block2*.

If *BooleanExpr* utvärderar till falskt, hoppar över *Block1* och exekverar *Block2*.

Block1 och *Block2* kan vara enstaka påståenden.

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

$g(12)$	12
$g(-12)$	12

If

Katalog >

If *BooleanExpr1* **Then**

Block1

ElseIf *BooleanExpr2* **Then**

Block2

⋮

ElseIf *BooleanExprN* **Then**

BlockN

EndIf

Medger förgrening. If *BooleanExpr1* utvärderar till sant och exekverar *Block1*. If *BooleanExpr1* utvärderar till falskt, utvärderar *BooleanExpr2*, och så vidare.

```
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(*BooleanExpr*,*Value_If_true* **[,***Value_If_false* **[,***Value_If_unknown***]])**⇒*uttryck, lista eller matris*

Utvärderar det booleska uttrycket *BooleanExpr* (eller varje element från *BooleanExpr*) och producerar ett resultat baserat på följande regler:

- *BooleanExpr* kan testa ett enstaka värde, en lista eller en matris.
- Om ett element i *BooleanExpr* utvärderas som sant erhålls motsvarande element från *Value_If_true*.
- Om ett element i *BooleanExpr* utvärderas som falskt erhålls motsvarande element från *Value_If_false*. Om du utelämnar *Value_If_false* erhålls undef.
- Om ett element i *BooleanExpr* är varken sant eller falskt erhålls motsvarande element från *Value_If_unknown*. Om du utelämnar *Value_If_unknown* erhålls undef.
- Om det andra, tredje eller fjärde argumentet i funktionen **ifFn()** är ett enstaka uttryck tillämpas det booleska

```
ifFn({1,2,3}<2.5,{5,6,7},{8,9,10})
                                    {5,6,10}
```

Testvärdet på **1** är mindre än 2.5, varför dess motsvarande

Value_If_True element **5** kopieras till resultatlistan.

Testvärdet på **2** är mindre än 2.5, varför dess motsvarande

Value_If_True element **6** kopieras till resultatlistan.

Testvärdet på **3** är inte mindre än 2.5, varför dess motsvarande *Value_If_False* element **10** kopieras till resultatlistan.

```
ifFn({1,2,3}<2.5,4,{8,9,10})        {4,4,10}
```

Value_If_true är ett enstaka värde och motsvarar varje vald position.

ifFn()

Katalog >

testet på varje position i *BooleanExpr*.

Obs: Om det förenklade påståendet *BooleanExpr* inbegriper en lista eller matris måste alla övriga list- eller matrisargument ha samma dimensioner, och resultatet får då samma dimensioner.

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

Value_If_false är ej specificerat. Undef används.

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

Ett valt element från *Value_If_true*. Ett valt element från *Value_If_unknown*.

imag()

Katalog >

imag(*Expr1***)⇒***uttryck*

Ger argumentets imaginärdel.

Obs: Alla odefinierade variabler behandlas som reella variabler. Se även **real()**, på sidan 147

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

$\text{imag}(z)$ 0

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

imag(*List1***)⇒***lista*

Ger en lista på elementens imaginärdelar.

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

imag(*Matrix1***)⇒***matris*

Ger en matris på elementens imaginärdelar.

$\text{imag}\begin{bmatrix} a & b \\ i\cdot c & i\cdot d \end{bmatrix}$ $\begin{bmatrix} 0 & 0 \\ c & d \end{bmatrix}$

impDif()

Katalog >

impDif(*Equation***, *Var*, *dependVar*[,*Ord*])**
⇒*uttryck*

där ordern *Ord* förinställs på 1.

Beräknar den implicita derivatan för ekvationer i vilka en variabel är implicit definierad med termer från en annan.

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

Indirection

Se #(), på sidan 228.

inString()

Katalog >

inString(*srcString***,** *subString***[,** *Start***])⇒***heltal*

Ger teckenpositionen i strängen *srcString* där den första förekomsten av strängen *subString* börjar.

Start, om inkluderad, specificerar teckenpositionen inom *srcString* där sökningen börjar. Förinställning = 1 (det första tecknet i *srcString*).

Återgår till noll om *srcString* inte innehåller *subString* eller om *Start* är > längden på *srcString*.

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

int()

Katalog >

int(*Expr***)⇒***heltal*

int(*List1***)⇒***lista*

int(*Matrix1***)⇒***matris*

Ger det största heltalet som är mindre än eller lika med argumentet. Denna funktion är identisk med **floor()**.

Argumentet kan vara ett reellt eller ett komplext tal.

Ger, för en lista eller matris, det största heltalet för varje element.

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

intDiv()

Katalog >

intDiv(*Number1***,** *Number2***)⇒***heltal*

intDiv(*List1***,** *List2***)⇒***lista*

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

Ger heltalsdelen av (*Number1* ÷ *Number2*).

Ger, för listor och matriser, heltalsdelen av (argument 1 ÷ argument 2) för varje elementpar.

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

integral

Se ∫↓⇐⇒ π σ δα ◆♦◆-

interpolate ()

Katalog >

interpolate(*xValue*, *xList*, *yList*, *yPrimeList***)**⇒*lista*

Denna funktion gör följande:

Baserat på *xList*, *yList*=**f(***xList***)** och *yPrimeList*=**f'(***xList***)** för en okänd funktion **f** används en kubisk interpolering för att approximera funktionen **f** vid *xValue*. Det förutsätts att *xList* är en lista på monotont ökande eller minskande tal, men denna funktion kan ge ett värde även när listan inte uppfyller förutsättningarna. Denna funktion går igenom *xList* och söker ett intervall [*xList*[i], *xList*[i+1]] som innehåller *xValue*. Om funktionen hittar ett sådant intervall ger den ett interpolerat värde på **f(***xValue***)**, annars ger den **undef.**

xList, *yList* och *yPrimeList* måste ha samma dimension ≥ 2 och innehålla uttryck som förenklas till tal.

xValue kan vara en odefinierad variabel, ett tal eller en lista på tal.

Differentialekvation:

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

```
rk:=rk23(-3·y+6·t+5,t,y,{0,10},5,1)
[0.      1.        2.        3.        4.
 5.   3.19499   5.00394   6.99957   9.00593   10
```

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Använd funktionen interpolate() för att beräkna funktionsvärdena för xvaluelist:

```
xvaluelist:=seq(i,i,0,10,0.5)
{0,0.5,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,5.5,6.,6.5,
xlist:=mat▸list(rk[1])
{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.}
ylist:=mat▸list(rk[2])
{5.,3.19499,5.00394,6.99957,9.00593,10.9978
yprimelist:=-3·y+6·t+5|y=ylist and t=xlist
{-10.,1.41503,1.98819,2.00129,1.98221,2.006
interpolate(xvaluelist,xlist,ylist,yprimelist)
{5.,2.67062,3.19499,4.02782,5.00394,6.00011
```

invχ2()

Katalog >

invχ2(*Area*,*df***)**

invChi2(*Area*,*df***)**

Beräknar den inversa kumulativa sannolikhetsfunktionen χ^2 (chi-kvadrat) specificerad av frihetsgraden *df* för en given *Area* under kurvan.

invF()

Katalog >

invF(*Area*,*dfNumer*,*dfDenom***)**

invF() Katalog >

invF(*Area*,*dfNumer*,*dfDenom***)**

Beräknar den inversa kumulativa fördelningsfunktionen F specificerad av *dfNumer* och *dfDenom* för en given *Area* under kurvan.

invNorm() Katalog >

invNorm(*Area*[,μ[,σ]]**)**

Beräknar den inversa kumulativa normalfördelningsfunktionen för en given *Area* under normalfördelningskurvan specificerad av μ och σ.

invt() Katalog >

invt(*Area*,*df***)**

Beräknar den inversa kumulativa sannolikhetsfunktionen student-t specifierad av Degree of Freedom (Frihetsgrad), *df*, för en given *Area* under kurvan.

iPart() Katalog >

iPart(*Number***)⇒***heltal*

iPart(*List1***)⇒***lista*

iPart(*Matrix1***)⇒***matris*

Ger argumentets heltalsdel.

Ger, för listor och matriser, heltalsdelen för varje element.

Argumentet kan vara ett reellt eller ett komplext tal.

$\text{iPart}(-1.234)$ → $-1.$

$\text{iPart}\left(\left\{\frac{3}{2},-2.3,7.003\right\}\right)$ → $\{1.,-2.,7.\}$

irr() Katalog >

irr(*CF0,CFList* [,*CFFreq*]**)⇒***value*

Finansiell funktion som beräknar internräntan på en investering.

CF0 är det initiala kassaflödet vid tidpunkt 0 och måste vara ett reellt tal.

CFList är en lista på kassaflödesbelopp efter det initiala kassaflödet CF0.

CFFreq är en frivillig lista i vilken varje element specificerar frekvensen för ett grupperat (konsekutivt) kassaflödesbelopp, vilket är det motsvarande elementet i *CFList*. Förinställningen är 1. Om du vill mata in värden måste de vara positiva heltal < 10.000.

Obs: Se även **mirr()**, på sidan 116.

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

isPrime() Katalog >

isPrime(*Number***)⇒***Booleskt konstantuttryck*

Ger sant eller falskt för att indikera om *number* är ett heltal ≥ 2 som är jämnt delbart endast med sig självt och 1.

Om *Number* överskrider cirka 306 siffror och saknar faktorer ≤1021 visar **isPrime (***Number***)** ett felmeddelande.

Om du endast vill bestämma om *Number* är ett primtal, använd **isPrime()** i stället för **factor()**. Detta går mycket fortare, särskilt om *Number* inte är ett primtal och dess näst största faktor har mer än cirka fem siffror.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

$\text{isPrime}(5)$	true
$\text{isPrime}(6)$	false

Funktion för att hitta nästa primtal efter ett specificerat tal:

Define $nextprim(n)=$Func Loop $n+1\rightarrow n$ If $\text{isPrime}(n)$ Return n EndLoop EndFunc	*Done*
$nextprim(7)$	11

isVoid() Katalog >

isVoid(*Var***)⇒***Booleskt konstantuttryck*

isVoid(*Expr***)⇒***Booleskt konstantuttryck*

isVoid(*List***)⇒***lista på Booleska konstantuttryck*

Ger sant eller falskt för att indikera huruvida argumentet är en tom datatyp.

För mer information om tomma element, se på sidan 238.

$a:=\_$	$\_$
$\text{isVoid}(a)$	true
$\text{isVoid}(\{1,\_,3\})$	$\{\text{false},\text{true},\text{false}\}$

L

Lbl Katalog >

Lbl *labelName*

Definierar en etikett med namnet *labelName* inom en funktion.

Du kan använda en **Goto** *labelName*-instruktion för att överföra kontroll till instruktionen direkt efter etiketten.

labelName måste uppfylla samma krav på namngivning som ett variabelnamn.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Define $g()$=Func	*Done*
Local *temp*,*i*	
$0\rightarrow temp$	
$1\rightarrow i$	
Lbl *top*	
$temp+i\rightarrow temp$	
If $i<10$ Then	
$i+1\rightarrow i$	
Goto *top*	
EndIf	
Return *temp*	
EndFunc	
$g()$	55

lcm() Katalog >

lcm(*Number1***, *Number2***)⇒***uttryck*

lcm(*List1***, *List2***)⇒***lista*

lcm(*Matrix1***, *Matrix2***)⇒***matris*

Ger den minsta gemensamma multipeln för de två argumenten. **lcm** för två bråk är **lcm** för deras täljare dividerat med **gcd** för deras nämnare. **lcm** för tal i flyttalsform är deras produkt.

$\text{lcm}(6,9)$	18
$\text{lcm}\left(\left\{\frac{1}{3},-14,16\right\},\left\{\frac{2}{15},7,5\right\}\right)$	$\left\{\frac{2}{3},14,80\right\}$

lcm() Katalog >

Ger, för två listor eller matriser, den minsta gemensamma multipeln för de motsvarande elementen.

left() Katalog >

left(*sourceString***[, *Num*])⇒*sträng***

Ger *Num*-tecknen längst till vänster i teckensträngen *sourceString*.

Om du utelämnar *Num* erhålls alla i *sourceString*.

left(*List1***[, *Num*])⇒*lista***

Ger *Num*-elementen längst till vänster i *List1*.

Om du utelämnar *Num* erhålls alla i *List1*.

left(*Comparison***)⇒*uttryck***

Ger den vänstra sidan av en ekvation eller olikhet.

left("Hello",2) "He"

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

left(x<3) x

libShortcut() Katalog >

libShortcut(*LibNameString***, *ShortcutNameString***

[, *LibPrivFlag*])⇒*lista på variabler*

Skapar en variabelgrupp i det aktuella problemet som innehåller referenser till alla objekt i det specificerade biblioteksdokumentet *libNameString*. Lägger också till gruppmedlemmarna på menyn Variables. Du kan sedan referera till varje objekt med hjälp av dess *ShortcutNameString*.

Ställ *LibPrivFlag*=**0** för att utesluta privata biblioteksobjekt (förinställning)

Ställ *LibPrivFlag*=**1** för att inkludera privata biblioteksobjekt

För att kopiera en variabelgrupp, se **CopyVar**, på sidan 33.

Detta exempel förutsätter ett korrekt lagrat och uppdaterat biblioteksdokument med namnet **linalg2** och som innehåller objekt definierade som *clearmat*, *gauss1* och *gauss2*.

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}

libShortcut() Katalog >

För att ta bort en variabelgrupp, se **DelVar**, på sidan 54.

limit() eller lim() Katalog >

limit(*Expr1***,** *Var***,** *Point* [**,***Direction*]**)⇒***uttryck*

limit(*List1***,** *Var***,** *Point* [**,** *Direction*]**)⇒***lista*

limit(*Matrix1***,** *Var***,** *Point* [**,** *Direction*]**)⇒***matris*

Ger det begärda gränsvärdet.

Obs: Se även **Limit template**, på sidan 10.

Direction (Riktning): negativ=från vänster, positiv=från höger, annars=båda. (Om *Direction* utelämnas förinställs den till båda.)

Gränsvärden vid positiv ∞ och vid negativ ∞ konverteras alltid till ensidiga gränsvärden från den ändliga sidan.

Beroende på omständigheterna ger **limit()** sig självt eller undef när den inte kan bestämma ett unikt gränsvärde. Detta innebär nödvändigtvis inte att ett unikt gränsvärde inte existerar. “undef” innebär att resultatet antingen är ett okänt tal med ändlig eller oändlig storlek, eller hela uppsättningen av sådana tal.

limit() använder metoder såsom L’Hopital’s regel, så det finns unika gränsvärden som den inte kan bestämma. Om *Expr1* innehåller odefinierade variabler utöver *Var* kan du behöva begränsa dem för att erhålla ett mer kortfattat resultat.

$\lim_{x\to 5}(2\cdot x+3)$	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() eller lim()

Katalog >

Gränsvärden kan vara mycket känsliga för avrundningsfel. Undvik om möjligt inställningen Approximate (Ungefärlig) i läge **Auto eller Ungefärlig** och ungefärliga tal när du beräknar gränsvärden. Annars har gränsvärden som skulle vara noll eller ha oändlig storlek sannolikt inte dessa egenskaper, och gränsvärden som skulle ha ändlig icke nollstorlek kanske inte har detta.

LinRegBx

Katalog >

LinRegBx *X*,*Y*[,[*Freq*][,*Category*,*Include*]]

Utför den linjära regressionsanalysen y = a+b·xpå listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a+b·x
stat.a, stat.b	Regressionskoefficienter

Resultatvariabel	Beskrivning
stat.r^2	Determinationskoefficient
stat.r	Korrelationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

LinRegMx

Katalog >

LinRegMx *X*,*Y*[,[*Freq*][,*Category*,*Include*]]

Beräknar den linjära regressionen y = m ·x+b på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: m ·x+b
stat.m, stat.b	Regressionskoefficienter

Resultatvariabel	Beskrivning
stat.r^2	Determinationskoefficient
stat.r	Korrelationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

LinRegtIntervals

Katalog >

LinRegtIntervals *X*,*Y*[,*F*[,**0**[,*CLev*]]]

För Slope (Lutning). Beräknar ett nivå-C-konfidensintervall för lutningen.

LinRegtIntervals *X*,*Y*[,*F*[,**1**,*Xval*[,*CLev*]]]

För Response (Svar). Beräknar ett prognostiserat y-värde, ett nivå-C-prediktionsintervall för en enstaka observation och ett nivå-C-konfidensintervall för medelvärdet på svaret.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

F är en frivillig lista på frekvensvärden. Varje element i *F* specificerar förekomsten av varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a+b ·x

Resultatvariabel	Beskrivning
stat.a, stat.b	Regressionskoefficienter
stat.df	Frihetsgrader
stat.r^2	Determinationskoefficient
stat.r	Korrelationskoefficient
stat.Resid	Residualer från regressionen

Endast för typen Slope

Resultatvariabel	Beskrivning
[stat.CLower, stat.CUpper]	Konfidensintervall för lutningen
stat.ME	Konfidensintervall - felmarginal
stat.SESlope	Standardfel hos lutning
stat.s	Standardfel hos linjen

Endast för typen Response

Resultatvariabel	Beskrivning
[stat.CLower, stat.CUpper]	Konfidensintervall för medelvärdet på svaret
stat.ME	Konfidensintervall - felmarginal
stat.SE	Standardfel hos medelsvar
[stat.LowerPred, stat.UpperPred]	Prediktionsintervall för en enstaka observation
stat.MEPred	Prognostiseringsintervall - felmarginal
stat.SEPred	Standardfel för prognostisering
stat.$\hat{y}$	a + b ·XVal

LinRegtTest

Katalog >

LinRegtTest X,Y[,$Freq$[,$Hypoth$]]

Utför en linjär regressionsanalys på listorna X och Y och ett t-test på lutningens värde β samt korrelationskoefficienten ρ för ekvationen $y=\alpha+\beta x$. Det testar nollhypotesen $H_0:\beta=0$ (equivalently, $\rho=0$) mot en av tre alternativa hypoteser.

Alla listor måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Hypoth är ett valfritt värde som specificerar en av tre alternativa hypoteser mot vilka nollhypotesen (H_0:β=ρ=0) kommer att testas.

För H_a: β≠0 och ρ≠0 (förinställning), ställ *Hypoth*=0

För H_a: β<0 och ρ<0, ställ *Hypoth*<0

För H_a: β>0 och ρ>0, ställ *Hypoth*>0

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	**Beskrivning**
stat.RegEqn	Regressionsekvation: a + b · x
stat.t	*t*-Statistik för signifikanstest
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader
stat.a, stat.b	Regressionskoefficienter
stat.s	Standardfel hos linjen
stat.SESlope	Standardfel hos lutning
stat.r^2	Determinationskoefficient
stat.r	Korrelationskoefficient
stat.Resid	Residualer från regressionen

linSolve() Katalog >

linSolve(*SystemAvLinjäraEkv*, *Var1*, *Var2*, *...***)⇒***lista*

linSolve(*LinjärEkv1* **och** *LinjärEkv2* **och** *...*, *Var1*, *Var2*, *...***)⇒***lista*

linSolve({*LinjärEkv1*, *LinjärEkv2*, *...***},** *Var1*, *Var2*, *...***)** ⇒*lista*

linSolve(*SystemAvLinjäraEkv*, **{***Var1*, *Var2*, *...***})** ⇒*lista*

linSolve(*LinjärEkv1* **och** *LinjärEkv2* **och** *...*, **{***Var1*, *Var2*, *...***})**⇒*lista*

linSolve({*LinjärEkv1*, *LinjärEkv2*, *...***},** **{***Var1*, *Var2*, *...***})** ⇒*lista*

Ger en lista på lösningar för variablerna *Var1*, *Var2*, *...*

Det första argumentet måste beräknas till ett system av linjära ekvationer eller en enda linjär ekvation. Annars inträffar ett argumentfel.

Som exempel leder beräkningen **`linSolve(x=1 och x=2,x)`** till ett "Argumentfel"-resultat.

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

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

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

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

ΔList() Katalog >

ΔList(*List1***)**⇒*lista*

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`deltaList(...)`**.

Ger en lista på skillnaderna mellan konsekutiva element i *List1*. Varje element i *List1* subtraheras från nästa element i *List1*. Den resulterande listan är alltid ett element kortare än den ursprungliga *List1*.

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

list▸mat() Katalog >

list▸mat(*List* **[,** *elementsPerRow***])⇒***matris*

Ger en matris fylld rad efter rad med elementen från *List*.

elementsPerRow, om inkluderad, specificerar antalet element per rad. Förinställningen är antalet element i *List* (en rad).

Om *List* inte fyller den resulterande listan läggs nollor till.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`list@>mat(...)`**.

$\text{list▸mat}(\{1,2,3\})$ $\quad [1\ 2\ 3]$

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

▸ln Katalog >

Expr **▸ln⇒***uttryck*

Medför att inmatningen *Expr* konverteras till ett uttryck som endast innehåller naturliga logaritmer (ln).

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>ln`**.

$\left(\log_{10}(x)\right)\text{▸ln}$ $\quad \frac{\ln(x)}{\ln(10)}$

ln() ctrl eˣ tangenter

ln(*Expr1***)⇒***uttryck*

ln(*List1***)⇒***lista*

Ger argumentets naturliga logaritm.

Ger, för en lista, elementens naturliga logaritmer.

$\ln(2.)$ $\quad 0.693147$

Om det komplexa formatläget är Real:

$\ln(\{-3,1.2,5\})$

"Error: Non−real calculation"

Om det komplexa formatläget är Rectangular:

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

ln()

ctrl e^x tangenter

ln(*squareMatrix1***)⇒***kvadratMatris***

Ger matrisen med naturlig logaritm för *squareMatrix1*. Detta är inte detsamma som att beräkna den naturliga logaritmen för varje element. För information om beräkningsmetoden, se **cos()**.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer och i Rektangulärt komplext format:

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

$$\begin{bmatrix}1.83145+1.73485\cdot i & 0.009193-1.49086 & \\ 0.448761-0.725533\cdot i & 1.06491+0.623491 & \\ -0.266891-2.08316\cdot i & 1.12436+1.79018\cdot & \end{bmatrix}$$

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

LnReg

Katalog >

LnReg X**,** Y[**,** [$Freq$] [**,** $Category$**,** $Include$]]

Utför en en logaritmisk regressionsanalys y = a+b·ln(x) på listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom $Include$ måste ha samma dimensioner.

X och Y är listor på oberoende och beroende variabler.

$Freq$ är en frivillig lista på frekvensvärden. Varje element i $Freq$ specificerar frekvensen för varje motsvarande X- och Y-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

$Category$ är en lista på kategorikoder för motsvarande X- och Y-data.

$Include$ är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: a+b ·ln(x)
stat.a, stat.b	Regressionskoefficienter
stat.r^2	Koefficient för linjär bestämning av transformerade data
stat.r	Korrelationskoefficient för transformerade data (ln(x), y)
stat.Resid	Residualer associerade med den logaritmiska modellen
stat.ResidTrans	Residualer associerade med linjär anpassning av transformerade data
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

Local

Katalog >

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

Betecknar specificerade *vars* som lokala variabler. Dessa variabler existerar endast under utvärderingen av ett uttryck och tas bort när exekveringen av uttrycket är klar.

Obs: Lokala variabler sparar minne eftersom de endast existerar tillfälligt. De stör heller inga befintliga globala variabelvärden. Lokala variabler måste användas för **For**-slingor och för att temporärt spara värden i en flerradig funktion eftersom modifieringar av globala värden inte är tillåtna i en funktion.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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***.**

Låser den specificerade variabeln eller variabelgruppen. Låsta variabler kan inte modifieras eller tas bort.

Du kan inte låsa eller låsa upp systemvariabeln *Ans* och du kan inte låsa systemvariabelgrupperna *stat.* och *tvm.*

Obs: Kommandot **Lås** (**Lock**) rensar Ångra/Upprepa-historiken när det används på olåsta variabler.

Se **unLock**, på sidan 199 och**getLockInfo()**, på sidan 84.

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()

ctrl 10ˣ tangenter

log(*Expr1*[**,***Expr2*]**)⇒***uttryck*

log(*List1*[**,***Expr2*]**)⇒***lista*

Ger bas-*Expr2*-logaritmen för det första argumentet.

Obs: Se även **Log template**, på sidan 6.

Ger, för en lista, bas-*Expr2*-logaritmen för elementen.

Om det andra argumentet utelämnas används 10 som bas.

$\log_{10}(2.)$	0.30103
$\log_{4}(2.)$	0.5
$\log_{3}(10)-\log_{3}(5)$	$\log_{3}(2)$

Om det komplexa formatläget är Real:

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

Om det komplexa formatläget är Rectangular:

$$\log_{10}(\{-3,1.2,5\})$$
$$\left\{\log_{10}(3)+1.36438\cdot i,0.079181,\log_{10}(5)\right\}$$

log(*squareMatrix1* [**,***Expr*]**)⇒***kvadratMatris*

I vinkelläget Radianer och i Rektangulärt komplext format:

log()

ctrl 10^x tangenter

Ger matrisen med bas-*Expr*-logaritm för *squareMatrix1*. Detta är inte detsamma som att beräkna bas-*Expr*-logaritmen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

Om basargumentet utelämnas används 10 som bas.

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

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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

▶logbase

Katalog >

Expr ▶**logbase(***Expr1***)**⇒*uttryck*

Medför att inmatningen Expression förenklas till ett uttryck med basen *Expr1*.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>logbase(**...**)**.

$$\log_3(10)-\log_5(5)\blacktriangleright\text{logbase}(5) \qquad \frac{\log_5\left(\frac{10}{3}\right)}{\log_5(3)}$$

Logistic

Katalog >

Logistic *X*, *Y*[, [*Freq*] [, *Category*, *Include*]]

Utför den logistiska regressionsanalyseny = (c/(1+a ·e^-bx))på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $c/(1+a \cdot e^{-bx})$
stat.a, stat.b, stat.c	Regressionskoefficienter
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

LogisticD *X*, *Y* [, [*Iterations*], [*Freq*] [, *Category*, *Include*]]

Utför den logistiska regressionsanalysen y = $(c/(1+a \cdot e^{-bx})+d)$ på listorna *X* och *Y* med frekvensen *Freq*, med ett specificerat antal *Iterationer*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Iterations är ett valfritt värde som specificerar det maximala antalet gånger en lösning kommer att provas. Om denna utelämnas används 64. Normalt ger större värden bättre noggrannhet, men längre exekveringstider, och vice versa.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $c/(1+a \cdot e^{-bx})+d)$
stat.a, stat.b, stat.c, stat.d	Regressionskoefficienter
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

Loop

Loop

Block

EndLoop

Exekverar påståendena i *Block* upprepade gånger. Observera att slingan upprepas i all oändlighet såvida inte en **Goto**- eller **Exit**-instruktion exekveras inom *Block*.

Block är en serie av påståenden separerade med tecknet ":".

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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

LU *Matris*, *lMatris*, *uMatris*, *pMatris[,Tol]*

Beräknar uppdelningen Doolittle LU (undre-övre) av en reell eller komplex matris. Den undertriangulära matrisen lagras i *lMatris*, den övertriangulära matrisen lagras i *uMatris* och permutationsmatrisen (som beskriver radväxlingarna som har gjorts under beräkningen) lagras i *pMatris*.

lMatris · *uMatris* = *pMatris* · *matris*

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning med tal i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du använder ctrl enter eller ställer in **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E-14 ·max(dim(*Matrix*)) ·rowNorm (*Matrix*)

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

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

lower

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

upper

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

perm

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

LU

Katalog >

Faktoriseringsalgoritmen **LU** använder partiell pivotering med radutbyten.

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

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

lower

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

upper

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

perm

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

M

mat▸list()

Katalog >

mat▸list(*Matrix***)⇒***lista*

Ger en lista med elementen i *Matrix*. Elementen kopieras från *Matrix* rad för rad.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`mat@>list(`**...**`)`**.

$\text{mat▸list}\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ → $\{1,2,3\}$

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

$\text{mat▸list}(m1)$ → $\{1,2,3,4,5,6\}$

max()

Katalog >

max(*Expr1*, *Expr2***)⇒***uttryck*

max(*List1*, *List2***)⇒***lista*

max(*Matrix1*, *Matrix2***)⇒***matris*

Ger de två argumentens maximum. Ger, om argumenten är två listor eller matriser, en lista eller matris som innehåller maximumvärdet för varje par av motsvarande element.

$\max(2.3,1.4)$ → 2.3

$\max(\{1,2\},\{-4,3\})$ → $\{1,3\}$

max(*List***)⇒***uttryck*

Ger maximumelementet i *list*.

$\max(\{0,1,-7,1.3,0.5\})$ → 1.3

max(*Matrix1***)⇒***matris*

Ger en radvektor som innehåller maximumelementet för varje kolumn i *Matrix1*.

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

max() Katalog >

Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

Obs: Se även **fMax()** och **min()**.

mean() Katalog >

mean(*List*[**,** *freqList*]**)**⇒*uttryck*

Ger medelvärdet för elementen i *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

mean(*Matrix1*[**,** *freqMatrix*]**)**⇒*matris*

Ger en radvektor med medelvärdena för alla kolumner i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

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

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

I vektorformatet Rectangular:

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

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

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

median() Katalog >

median(*List*[**,** *freqList*]**)**⇒*uttryck*

Ger medianen för elementen i *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

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

Ger en radvektor som innehåller medianerna för kolumnerna i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

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

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

median() Katalog >

Obs:

- Alla inmatningar i listan eller matrisen måste förenklas till tal.
- Tomma element i listan eller matrisen ignoreras. För mer information om tomma element, se på sidan 238.

MedMed Katalog >

MedMed *X*,*Y* [, *Freq*] [, *Category*, *Include*]]

Beräknar median-median-linjeny = (m ·x+b)på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Ekvation för median-median-linje m ·x+b
stat.m, stat.b	Modellkoefficienter
stat.Resid	Residualer från median-median-linjen

Resultatvariabel	Beskrivning
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

mid() Katalog >

mid(*sourceString***, *Start*[, *Count*])⇒*sträng***

Ger *Count*-tecknen från teckensträngen *sourceString* och börjar med teckennumret *Start*.

Ger, om *Count* utelämnas eller är större än dimensionen på *sourceString*, alla tecken från *sourceString* med start från teckennumret *Start*.

Count måste vara ≥ 0. Om *Count* = 0 erhålls en tom sträng.

```
mid("Hello there",2)                "ello there"
mid("Hello there",7,3)              "the"
mid("Hello there",1,5)              "Hello"
mid("Hello there",1,0)              ""
```

mid(*sourceList***, *Start* [, *Count*])⇒*lista***

Ger *Count*-elementen från *sourceList* och börjar med elementnumret *Start*.

Ger, om *Count* utelämnas eller är större än dimensionen på *sourceList*, alla element från *sourceList* med start från elementnumret *Start*.

Count måste vara ≥ 0. Om Count = 0 erhålls en tom lista.

```
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(*sourceStringList***, *Start*[, *Count*])⇒*lista***

Ger *Count*-strängarna från stränglistan *sourceStringList* och börjar med elementnumret *Start*.

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

min() Katalog >

min(*Expr1***, *Expr2*)⇒*uttryck***

min(*List1, List2*)⇒*lista***

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

min() Katalog >

min(*Matrix1, Matrix2***)⇒***matris***

Ger de två argumentens minimum. Ger, om argumenten är två listor eller matriser, en lista eller matris som innehåller minimumvärdet för varje par av motsvarande element.

min(*List***)⇒***uttryck***

Ger minimumelementet för *List*.

min(*Matrix1***)⇒***matris***

Ger en radvektor som innehåller minimumelementet för varje kolumn i *Matrix1*.

Obs: Se även **fMin()** och **max()**.

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

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

mirr() Katalog >

mirr
(*financeRate***,***reinvestRate***,***CF0***,***CFList*
[**,***CFFreq*])

Finansiell funktion som beräknar den modifierade internräntan på en investering.

financeRate är den räntesats som du betalar på kassaflödesbeloppen.

reinvestRate är den räntesats vid vilken kassaflödena återinvesteras.

CF0 är det initiala kassaflödet vid tidpunkt 0 och måste vara ett reellt tal.

CFList är en lista på kassaflödesbelopp efter det initiala kassaflödet CF0.

CFFreq är en frivillig lista i vilken varje element specificerar frekvensen för ett grupperat (konsekutivt) kassaflödesbelopp, vilket är det motsvarande elementet i *CFList*. Förinställningen är 1. Om du vill mata in värden måste de vara positiva heltal < 10.000.

Obs: Se även **irr()**, på sidan 94.

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

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

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

mod() Katalog >

mod(*Expr1*, *Expr2*)⇒*uttryck*

mod(*List1*, *List2*)⇒*lista*

mod(*Matrix1*, *Matrix2*)⇒*matris*

Ger det första argumentet modulo det andra argumentet definierat av identiteterna:

mod(x,0) = x

mod(x,y) = x − y floor(x/y)

När det andra argumentet är skilt från noll är resultatet periodiskt i det argumentet. Resultatet är antingen noll eller har samma tecken som det andra argumentet.

Ger, om argumenten är två listor eller matriser, en lista eller matris som innehåller modulen för varje par av motsvarande element.

Obs: Se även **remain()**, på sidan 149

$\mathrm{mod}(7,0)$ → 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() Katalog >

mRow(*Expr*, *Matrix1*, *Index***)⇒***matris*

Ger en kopia av *Matrix1* med varje element i rad *Index* i *Matrix1* multiplicerat med *Expr*.

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

mRowAdd() Katalog >

mRowAdd(*Expr*, *Matrix1*, *Index1*, *Index2***)** ⇒*matris*

Ger en kopia av *Matrix1* med varje element i rad *Index2* i *Matrix1* ersatt med:

Expr · *rad Index1* + *rad Index2*

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

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

MultReg Katalog >

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

MultReg

Katalog >

Beräknar den multipla linjära regressionen i lista *Y* på listorna *X1*, *X2*, ..., *X10*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor måste ha samma dimensioner.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238.

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: b0+b1 ·x1+b2 ·x2+ ...
stat.b0, stat.b1, ...	Regressionskoefficienter
stat.R^2	Koefficient för multipel bestämning
stat.$\hat{y}$List	$\hat{y}$List = b0+b1 ·x1+ ...
stat.Resid	Residualer från regressionsanalysen

MultRegIntervals

Katalog >

MultRegIntervals *Y*, *X1*[,*X2*[,*X3*,... [,*X10*]]],*XValList*[,*CLevel*]

Beräknar ett prognostiserat y-värde, ett nivå-C-prediktionsintervall för en enstaka observation och ett nivå-C-konfidensintervall för medelvärdet på svaret.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor måste ha samma dimensioner.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: b0+b1 ·x1+b2 ·x2+ ...
stat.$\hat{y}$	En punktuppskattning: $\hat{y}$ = b0 + b1 · xl + ... för *XValList*
stat.dfError	Fel hos frihetsgrader
stat.CLower, stat.CUpper	Konfidensintervall för ett medelvärde på svaret

Resultatvariabel	Beskrivning
stat.ME	Konfidensintervall - felmarginal
stat.SE	Standardfel hos medelvärdet på svaret
stat.LowerPred, stat.UpperrPred	Prediktionsintervall för en enstaka observation
stat.MEPred	Prediktionsintervall - felmarginal
stat.SEPred	Standardfel för prognostisering
stat.bList	Lista på regressionskoefficienter, {b0,b1,b3,…}
stat.Resid	Residualer från regressionen

MultRegTests

Katalog >

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

Ett multipelt linjärt regressionstest utför en multipel linjär regressionsanalys på givna data och ger den globala F-teststatistiken och t-teststatistiken för koefficienterna.

En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Utdata

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: b0+b1 ·x1+b2 ·x2+ …
stat.F	Global F-teststatistik
stat.PVal	P-värde associerat med global F-statistik
stat.R^2	Koefficient för multipel bestämning
stat.AdjR^2	Justerad koefficient för multipel bestämning
stat.s	Standardavvikelse hos felet
stat.DW	Durbin-Watson-statistik: används för att bestämma om modellen innehåller autokorrelation av första ordningen
stat.dfReg	Frihetsgrader hos regressionen

Resultatvariabel	Beskrivning
stat.SSReg	Regressionens kvadratsumma
stat.MSReg	Regression medelkvadrat
stat.dfError	Fel hos frihetsgrader
stat.SSError	Felens kvadratsumma
stat.MSError	Felens medelkvadrat
stat.bList	{b0,b1,…} Lista på koefficienter
stat.tList	Lista på t-statistik för varje koefficient i bList
stat.PList	Lista på P-värden för varje t-statistik
stat.SEList	Lista på standardfel för koefficienter i bList
stat.$\hat{y}$List	$\hat{y}$List = b0+b1 ·x1+ . . .
stat.Resid	Residualer från regressionen
stat.sResid	Standardiserade residualer: erhållna genom att dividera en residual med dess standardavvikelse
stat.CookDist	Cooks avstånd: mått på den influens som en observation har, baserat på residual och stigning
stat.Leverage	Mått på hur långt värdena i den oberoende variabeln är från sina medelvärden

N

nand

ctrl = knappar

BooleskUttr1 **nand** *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 **nand** *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 **nand** *BooleskMatris2* ger *Boolesk matris*

Ger negation av en logisk **and** uppgift på de två argumenten. Ger resultatet sant, falskt eller en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

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

nand

ctrl = knappar

Heltal1 **nand** *Heltal2* ⇒ *heltal*

Jämför två reella heltal bit för bit med en **nand**-operation Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om båda bitarna är 1, annars blir resultatet 0. Det erhållna värdet representerar bitresultatet och visas enligt Bas-läget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 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()

Katalog >

nCr(*Expr1*, *Expr2* **)** ⇒ *uttryck*

För heltal *Expr1* och *Expr2* med *Expr1* ≥ *Expr2* ≥ 0 är **nCr()** antalet kombinationer av *Expr1* saker tagna *Expr2* åt gången. (Detta kallas också en binomial koefficient.) Båda argumenten kan vara heltal eller symboliska uttryck.

nCr(*Expr*, **0)** ⇒ **1**

nCr(*Expr*, *negInteger* **)** ⇒ **0**

nCr(*Expr*, *posInteger* **)** ⇒ *Expr* ·(*Expr*−1)**...** **(** *Expr*−*posInteger***+1)/** *posInteger***!**

nCr(*Expr*, *nonInteger* **)** ⇒ *expression***!/** **((** *Expr*−*nonInteger***)!** ·*nonInteger***!)**

nCr(*List1*, *List2* **)** ⇒ *lista*

Ger en lista på kombinationer baserat på motsvarande elementpar i de två listorna. Argumenten måste ha samma liststorlek.

nCr(*Matrix1*, *Matrix2* **)** ⇒ *matris*

Ger en matris över kombinationer baserat på motsvarande elementpar i de två matriserna. Argumenten måste ha samma matrisstorlek.

$$\mathrm{nCr}(z,3) \qquad \frac{z\cdot(z-2)\cdot(z-1)}{6}$$

$$Ans|z=5 \qquad 10$$

$$\mathrm{nCr}(z,c) \qquad \frac{z!}{c!\cdot(z-c)!}$$

$$\frac{Ans}{\mathrm{nPr}(z,c)} \qquad \frac{1}{c!}$$

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

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

nDerivative() Katalog >

nDerivative(*Uttr1*,*Var*=*Värde*[,*Ordning*]**)⇒***värde*

nDerivative(*Uttr1*,*Var*[,*Ordning*]**) |** *Var*=*Värde***⇒***värde*

Ger den numeriska derivatan beräknad med automatiska deriveringsmetoder.

När *Värde* specificeras överstyr detta värde eventuella tidigare variabeltilldelningar eller aktuella ersättningar av typ "|" för variabeln.

Ordning för derivatan måste vara **1** eller **2**.

$\text{nDerivative}(\lvert x\rvert, x=1)$	1
$\text{nDerivative}(\lvert x\rvert, x)\lvert x=0$	undef
$\text{nDerivative}(\sqrt{x-1}, x)\lvert x=1$	undef

newList() Katalog >

newList(*numElements***)⇒***lista*

Ger en lista med dimensionen på *numElements*. Varje element är noll.

$\text{newList}(4)$	$\{0,0,0,0\}$

newMat() Katalog >

newMat(*numRows*, *numColumns***)⇒***matris*

Ger en matris med nollor med dimensionen *numRows* gånger *numColumns*.

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

nfMax() Katalog >

nfMax(*Expr*, *Var***)⇒***värde*

nfMax(*Expr*, *Var, undrGräns***)⇒***värde*

nfMax(*Expr*, *Var, undrGräns*, *övrGräns***)⇒***värde*

nfMax(*Expr, Var***) |** *undrGräns*≤*Var*≤*övrGräns***⇒***värde*

Ger ett möjligt numeriskt värde på variabeln *Var* där lokalt maximum för *Expr* inträffar.

$\text{nfMax}(-x^2-2\cdot x-1, x)$	-1.
$\text{nfMax}(0.5\cdot x^3-x-2, x, -5, 5)$	5.

nfMax() Katalog >

Om du inför *undrGräns* och *övrGräns* söker funktionen i det stängda intervallet [*undrGräns*,*övrGräns*] efter lokalt maximum.

Obs: Se även **fMax()** och **d()**.

nfMin() Katalog >

nfMin(*Expr***,** *Var***)⇒***värde*

nfMin(*Expr***,** *Var, undrGräns***)⇒***värde*

nfMin(*Expr***,** *Var, undrGräns***,** *övrGräns***)⇒***värde*

nfMin(*Expr, Var***) |** *undrGräns*≤*Var* ≤*övrGräns*⇒*värde*

Ger ett möjligt numeriskt värde på variabeln *Var* där lokalt minimum för *Expr* inträffar.

Om du inför *undrGräns* och *övrGräns* söker funktionen i det stängda intervallet [*undrGräns*,*övrGräns*] efter lokalt minimum.

Obs: Se även **fMin()** och **d()**.

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

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

nInt() Katalog >

nInt(*Expr1, Var, Lower, Upper***)⇒***uttryck*

Om integranden *Expr1* inte innehåller någon variabel utöver *Var*, och om *Lower* och *Upper* är konstanter, positiv ∞ eller negativ ∞, ger **nInt()** en uppskattning av ∫**(***Expr1, Var, Lower, Upper***)**. Denna uppskattning är ett vägt genomsnitt av vissa sampelvärden hos integranden i intervallet *Lower*<*Var*<*Upper*.

Målsättningen är sex signifikanta siffror. Den adaptiva algoritmen bestämmer när det verkar sannolikt att målet har uppnåtts, eller när det verkar osannolikt att ytterligare sampling ger en nämnvärd förbättring.

$\text{nInt}(e^{-x^2},x,-1,1)$ 1.49365

$\text{nInt}(\cos(x),x,-\pi,\pi+1.\text{E}-12)$ $-1.04144\text{E}-12$

$\int_{-\pi}^{\pi+10^{-12}}\cos(x)\,dx$ $-\sin\left(\frac{1}{1000000000000}\right)$

nInt()

Katalog >

En varning (“Questionable accuracy”) visas när det verkar som om målet inte har uppnåtts.

Man kan kapsla in **nInt()** för att utföra multipel numerisk integrering. Integrationsgränser kan bero på integrationsvariabler utanför gränserna.

Obs: Se även ∫**()**, på sidan 212.

$$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x\cdot y}}{\sqrt{x^2-y^2}},y,-x,x\right),x,0,1\right) \qquad 3.30423$$

nom()

Katalog >

nom(*effectiveRate*,*CpY*)⇒*värde*

Finansiell funktion som konverterar den årliga effektiva räntan *effectiveRate* till en nominell ränta, given av *CpY* som antalet sammansatta ränteperioder per år.

effectiveRate måste vara ett reellt tal och *CpY* måste vara ett reellt tal > 0.

Obs: Se även **eff()**, på sidan 61.

$$\text{nom}(5.90398,12) \qquad 5.75$$

nor

ctrl **=** **knappar**

BooleskUttr1 **nor** *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 **nor** *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 **nor** *BooleskMatris2* ger *Boolesk matris*

Ger negation av en logisk **or** operation på de två argumenten. Ger resultatet sant, falskt eller en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

$$x\geq 3 \text{ or } x\geq 4 \qquad x\geq 3$$

$$x\geq 3 \text{ nor } x\geq 4 \qquad x<3$$

nor

ctrl = knappar

Heltal1 **nor** *Heltal2* ⇒ *heltal*

Jämför två reella heltal bit för bit med en **nor**-operation Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om båda bitarna är 1, annars blir resultatet 0. Det erhållna värdet representerar bitresultatet och visas enligt Bas-läget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

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}

norm()

Katalog >

norm(*Matrix* **)** ⇒ *uttryck*

norm(*Vector* **)** ⇒ *uttryck*

Ger Frobenius norm.

$\text{norm}\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right)$	$\sqrt{a^2+b^2+c^2+d^2}$
$\text{norm}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	$\sqrt{30}$
$\text{norm}\left(\begin{bmatrix} 1 & 2 \end{bmatrix}\right)$	$\sqrt{5}$
$\text{norm}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$	$\sqrt{5}$

normalLine()

Katalog >

normalLine(*Expr1* **,** *Var* **,** *Point* **)** ⇒ *uttryck*

normalLine(*Expr1* **,** *Var* **=** *Point* **)** ⇒ *uttryck*

Ger normalen för kurvan representerad av *Expr1* vid punkten som specificeras i *Var=Point*.

Kontrollera att den oberoende variabeln inte är definierad. Om exempelvis f1(x):=5 och x:=3 ger **normalLine(**f1(x),x,2**)** "falskt".

$\text{normalLine}\left(x^2,x,1\right)$	$\frac{3}{2}-\frac{x}{2}$
$\text{normalLine}\left((x-3)^2-4,x,3\right)$	$x=3$
$\text{normalLine}\left(x^{\frac{1}{3}},x=0\right)$	0
$\text{normalLine}\left(\sqrt{\lvert x\rvert},x=0\right)$	undef

normCdf()

Katalog >

normCdf(*lowBound* **,** *upBound* **[,** μ **[,** σ **]])** ⇒ *tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

normCdf() Katalog >

Beräknar sannolikheten vid en normalfördelning mellan *lowBound* och *upBound* för specificerad μ (förinställning=0) och σ (förinställning=1).

För P(X ≤ *upBound*), sätt *lowBound* = −∞.

normPdf() Katalog >

normPdf(*XVal*[,μ[,σ]]**)**⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar värde hos täthetsfunktionen för normalfördelning vid ett specificerat *XVal*-värde för specificerad μ och σ.

not (icke) Katalog >

not *BooleanExpr*⇒*Booleskt uttryck*

Ger en sann, falsk eller förenklad form av argumentet.

not(2≥3)	true
not(*x*<2)	*x*≥2
not not *innocent*	*innocent*

not *Integer1*⇒*heltal*

Ger ettkomplementet till ett reellt heltal. Internt omvandlas *Integer1* till ett 64-bitars binärt tal. Värdet på varje bit växlas (0 blir 1 och vice versa) för ettans komplement. Resultaten visas enligt det inställda basläget.

I hexadecimalt basläge:

Viktigt: Noll, inte bokstaven O.

not 0h7AC36	0hFFFFFFFFFF853C9

Du kan skriva in heltalet i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalet som ett decimalt tal (bas 10).

I binärt basläge:

0b100101▶Base10	37
not 0b100101	
0b11111111111111111111111111111111▶	
not 0b100101▶Base10	−38

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk moduloperation för att få ned värdet till lämplig nivå. För mer information, se ▶**Base2**, på sidan 21.

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Obs: En binär inmatning kan ha upp till 64 siffror (exklusive prefixet 0b). En hexadecimal inmatning kan ha upp till 16 siffror.

nPr() Katalog >

nPr(*Expr1***,** *Expr2***)⇒***uttryck*

För heltal *Expr1* och *Expr2* med *Expr1* ≥ *Expr2* ≥ 0 är **nPr()** antalet permutationer av *Expr1* saker tagna *Expr2* åt gången. Båda argumenten kan vara heltal eller symboliska uttryck.

nPr(*Expr***, 0)⇒1**

nPr(*Expr***,** *negInteger***)⇒1/((***Expr***+1)** · **(***Expr***+2)**... **(***expression−negInteger***))**

nPr(*Expr***,** *posInteger***)⇒***Expr* ·**(***Expr***−1)...** **(***Expr−posInteger***+1)**

nPr(*Expr***,** *nonInteger***)⇒***Expr***! / (***Expr−nonInteger***)!**

nPr(*List1***,** *List2***)⇒***lista*

Ger en lista på permutationer baserat på motsvarande elementpar i de två listorna. Argumenten måste ha samma liststorlek.

nPr(*Matrix1***,** *Matrix2***)⇒***matris*

Ger en matris över permutationer baserat på motsvarande elementpar i de två matriserna. Argumenten måste ha samma matrisstorlek.

$\mathrm{nPr}(z,3)$	$z\cdot(z-2)\cdot(z-1)$
$Ans\|z=5$	60
$\mathrm{nPr}(z,-3)$	$\frac{1}{(z+1)\cdot(z+2)\cdot(z+3)}$
$\mathrm{nPr}(z,c)$	$\frac{z!}{(z-c)!}$
$Ans\cdot\mathrm{nPr}(z-c,-c)$	1

$\mathrm{nPr}(\{5,4,3\},\{2,4,2\})$	$\{20,24,6\}$

$\mathrm{nPr}\left(\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}\right)$	$\begin{bmatrix}30&20\\12&6\end{bmatrix}$

npv() Katalog >

npv(*InterestRate***,***CFO***,***CFList***[,***CFFreq***])**

Finansiell funktion som beräknar nettovärdet, dvs. summan av aktuella värden för kassainflöden och kassautflöden. Ett positivt resultat för npv indikerar en vinstgivande investering.

InterestRate är räntan med vilken kassaflödena (kapitalanskaffningskostnaderna) diskonteras under en period.

CF0 är det initiala kassaflödet vid tidpunkt 0 och måste vara ett reellt tal.

$list1:=\{6000,-8000,2000,-3000\}$	$\{6000,-8000,2000,-3000\}$
$list2:=\{2,2,2,1\}$	$\{2,2,2,1\}$
$\mathrm{npv}(10,5000,list1,list2)$	4769.91

CFList är en lista på kassaflödesbelopp efter det initiala kassaflödet *CF0*.

CFFreq är en lista i vilken varje element specificerar frekvensen för ett grupperat (konsekutivt) kassaflödesbelopp, vilket är det motsvarande elementet i *CFList*. Förinställningen är 1. Om du vill mata in värden måste de vara positiva heltal < 10.000.

nSolve(*Equation*,*Var* **[=***Guess***])**⇒*tal eller fel_sträng*

nSolve(*Equation*,*Var* **[=***Guess***]**,*lowBound***)** ⇒*tal eller fel_sträng*

nSolve(*Equation*,*Var* **[=***Guess***]**,*lowBound*,*upBound***)** ⇒*tal eller fel_sträng*

nSolve(*Equation*,*Var* **[=***Guess***]) |** *lowBound* ≤*Var*≤*upBound* ⇒*tal eller fel_sträng*

Söker iterativt efter en ungefärlig reell numerisk lösning på *Equation* för dess variabel. Specificera variabeln som:

variabel

– eller –

variabel = *reellt tal*

Som exempel är x giltigt och likaså x=3.

nSolve() är ofta mycket snabbare än **solve()** eller **zeros()**, särskilt om operatorn "|" används för att begränsa sökningen till ett litet intervall som innehåller exakt en enkel lösning.

nSolve() försöker att bestämma antingen en punkt där residualen är noll eller två relativt närliggande punkter där residualen har motsatta tecken och inte är överdrivet stor. Om detta inte kan uppnås med ett måttligt antal samplingspunkter erhålls strängen "no solution found".

$\text{nSolve}\left(x^2+5\cdot x-25=9,x\right)$	3.84429
$\text{nSolve}\left(x^2=4,x=-1\right)$	−2.
$\text{nSolve}\left(x^2=4,x=1\right)$	2.

Obs: Om det finns flera lösningar kan du använda en gissning för att lättare hitta en viss lösning.

$\text{nSolve}\left(x^2+5\cdot x-25=9,x\right)\mid x<0$	−8.84429
$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26,r\right)\mid r>0 \text{ and } r<0.25$	0.006886
$\text{nSolve}\left(x^2=-1,x\right)$	"No solution found"

Obs: Se även **cSolve()**, **cZeros()**, **solve()** och **zeros()**.

O

OneVar [**1,**]*X*[**,**[*Freq*][**,***Category*,*Include*]]

OneVar [*n***,**]*X1*,*X2*[*X3*[**,**...[**,***X20*]]]

Beräknar 1-variabelstatistik på upp till 20 listor. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X-argumenten är datalistor.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*-värde. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Ett tomt element i någon av listorna *X*, *Freq* eller *Category* resulterar i ett tomrum för motsvarande element i dessa listor. Ett tomt element i någon av listorna *X1* till och med *X20* resulterar i ett tomrum för motsvarande element i dessa listor. För mer information om tomma element, se på sidan 238.

Resultatvariabel	**Beskrivning**
stat.$\bar{x}$	Medelvärde av x-värden
stat.Σx	Summa av x-värden

Resultatvariabel	Beskrivning
stat.Σx^2	Summa av x^2-värden
stat.sx	Standardavvikelse för x (sampling)
stat. x	Standardavvikelse för x (population)
stat.n	Antal datapunkter
stat.MinX	Minsta x-värde
stat.Q_1X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q_3X	Övre kvartil för x
stat.MaxX	Största x-värde
stat.SSX	Kvadratsumma av avvikelser från medelvärdet på x

or (eller) Katalog >

BooleskUttr1 **or** *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 **or** *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 **or** *BooleskMatris2* ger *Boolesk matris*

Ger resultatet sant eller falskt eller en förenklad form av den ursprungliga inmatningen.

Ger sant om ettdera eller båda uttrycken förenklas till sant. Ger resultatet falskt om båda uttrycken utvärderas som falska.

Obs: Se **xor**.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Integer1 **or** *Integer2*⇒*heltal*

$x\geq 3$ 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*

I hexadecimalt basläge:

0h7AC36 or 0h3D5F → 0h7BD7F

Viktigt: Noll, inte bokstaven O.

or (eller) Katalog >

Jämför två reella heltal bit för bit med en or-operation. Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om båda bitarna är 1. Resultatet blir 0 endast om båda bitarna är 0. Det erhållna värdet representerar bitresultaten och visas enligt det inställda basläget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. För mer information, se ▸**Base2**, på sidan 21.

Obs: Se **xor.**

I binärt basläge:

0b100101 or 0b100	0b100101

Obs: En binär inmatning kan ha upp till 64 siffror (exklusive prefixet 0b). En hexadecimal inmatning kan ha upp till 16 siffror.

ord() Katalog >

ord(*String***)⇒***integer*

ord(*List1***)⇒***lista*

Ger den numeriska koden för det första tecknet i teckensträngen *String* eller en lista på de första tecknen i varje listelement.

ord("hello")	104
char(104)	"h"
ord(char(24))	24
ord({"alpha","beta"})	{97,98}

P

P▸Rx() Katalog >

P▸Rx(*rExpr***, θ***Expr***)⇒***uttryck*

P▸Rx(*rList***, θ***List***)⇒***lista*

P▸Rx(*rMatrix***, θ***Matrix***)⇒***matris*

Ger den ekvivalenta x-koordinaten för paret (r, θ).

I vinkelläget Radianer:

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▶Rx() Katalog >

Obs: Argumentet θ tolkas som en vinkel i antingen grader, nygrader eller i radianer beroende på det aktuella vinkelläget. Om argumentet är ett uttryck kan du använda °, G eller r för att tillfälligt överstyra vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **P@>Rx(...)**.

P▶Ry() Katalog >

P▶Ry(*rExpr*, θ*Expr*)⇒*uttryck*

P▶Ry(*rList*, θ*List*)⇒*lista*

P▶Ry(*rMatrix*, θ*Matrix*)⇒*matris*

Ger den ekvivalenta y-koordinaten för paret (r, θ).

Obs: Argumentet θ tolkas som en vinkel i antingen grader, radianer eller nygrader beroende på det aktuella vinkelläget. Om argumentet är ett uttryck kan du använda °, G eller r för att tillfälligt överstyra vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **P@>Ry(...)**.

I vinkelläget Radianer:

$$\text{P▶Ry}(r,\theta) \qquad \sin(\theta)\cdot r$$

$$\text{P▶Ry}(4,60°) \qquad 2\cdot\sqrt{3}$$

$$\text{P▶Ry}\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right) \qquad \left\{\frac{-3\cdot\sqrt{3}}{2},-5\cdot\sqrt{2},0.\right\}$$

PassErr Katalog >

PassErr

Flyttar ett fel till nästa nivå.

Om systemvariabeln *errCode* är noll utför **PassErr** ingenting.

För ett exempel på **PassErr**, se exempel 2 under kommandot **Try**, på sidan 192.

PassErr

Katalog >

Villkoret **Else** i blocket **Try...Else...EndTry** bör använda **ClrErr** eller **PassErr**. Om felet skall processas eller ignoreras, använd **ClrErr**. Om det är okänt hur felet skall hanteras, använd **PassErr** för att skicka felet vidare till nästa felhanterare. Om det inte finns någon ytterligare felhanterare för **Try...Else...EndTry** visas feldialogrutan som normal.

Obs: Se även **ClrErr**, på sidan 29 och **Try**, på sidan 192.

Kommentarer om inmatningen av exemplet:I applikationen Räknare kan du skriva in flerradiga definitioner genom att trycka på ↵ i stället för enter i slutet av varje linje. På datorns tangentbord, håll ned **Alt** och tryck på **Enter**.

piecewise()

Katalog >

piecewise(*Expr1* [**,** *Condition1* [**,** *Expr2* [**,** *Condition2* [**,** ...]]]]**)**

Ger definitioner för en stegvis funktion i form av en lista. Du kan också skapa stegvisa definitioner med hjälp av en mall.

Obs: Se även **Piecewise template**, på sidan 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**,***lowBound***,***upBound***)**⇒*tal* om *lowBound* och *upBound* är tal, *lista* om *lowBound* och *upBound* är listor

poissCdf(λ**,***upBound***)**(för P(0≤X ≤*upBound*)⇒*tal* om *upBound* är ett tal, lista om *upBound* är en lista

Beräknar en kumulativ sannolikhet för den diskreta Poisson-fördelningen med det specificerade medelvärdet λ.

För P(X ≤ *upBound*), sätt *lowBound*=0

poissPdf() Katalog >

poissPdf(λ**,***XVal***)**⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar en sannolikhet för den diskreta Poisson-fördelningen med det specificerade medelvärdet λ.

▸Polar Katalog >

Vector **▸Polar**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Polar`**.

Visar *vector* i polär form [r∠θ]. Vektorn måste ha dimensionen 2 och kan vara en rad eller en kolumn.

Obs: ▸Polar är en visa format-instruktion, inte en konverteringsfunktion. Du kan endast använda den i slutet av en inmatningsrad, och den uppdaterar inte *ans*.

Obs: Se även **▸Rect**, på sidan 147.

$[1 \quad 3.] \blacktriangleright \text{Polar} \qquad [3.16228 \quad \angle 1.24905]$

$[x \quad y] \blacktriangleright \text{Polar} \qquad \left[\sqrt{x^2+y^2} \quad \angle \frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right)\right]$

complexValue **▸Polar**

Visar *complexVector* i polär form.

- Vinkelläget Grader ger (r∠θ).
- Vinkelläget Radianer ger $re^{i\theta}$.

complexValue kan ha valfri komplex form. En inmatning av $re^{i\theta}$ orsakar dock ett fel i vinkelläget Grader.

Obs: Du måste använda parenteserna för en (r∠θ) polär inmatning.

I vinkelläget Radianer:

$(3+4\cdot i) \blacktriangleright \text{Polar} \qquad e^{i\cdot\left(\frac{\pi}{2}-\tan^{-1}\left(\frac{3}{4}\right)\right)}\cdot 5$

$\left(\left(4 \angle \frac{\pi}{3}\right)\right) \blacktriangleright \text{Polar} \qquad e^{\frac{i\cdot\pi}{3}}\cdot 4$

I vinkelläget Nygrader:

$(4\cdot i) \blacktriangleright \text{Polar} \qquad (4 \angle 100)$

I vinkelläget Grader:

$(3+4\cdot i) \blacktriangleright \text{Polar} \qquad \left(5 \angle 90-\tan^{-1}\left(\frac{3}{4}\right)\right)$

polyCoeffs() Katalog >

polyCoeffs(*Poly* [,*Var*]**)⇒***lista*

Ger en lista på koefficienterna för polynomet *Poly* med avseende på variabeln *Var*.

Poly måste vara ett polynomuttryck i *Var*. Vi rekommenderar att du inte utelämnar *Var* såvida inte *Poly* är ett uttryck i bara en variabel.

$\text{polyCoeffs}(4\cdot x^2-3\cdot x+2,x)$ $\{4,-3,2\}$

$\text{polyCoeffs}((x-1)^2\cdot(x+2)^3)$ $\{1,4,1,-10,-4,8\}$

Expanderar polynomet och väljer x för den utelämnade *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*]**)⇒***värde*

Ger graden för polynomuttrycket *Poly* med avseende på variabeln *Var*. Om du utelämnar *Var* väljer funktionen **polyDegree()** en förinställning från variablerna i polynomet *Poly*.

Poly måste vara ett polynomuttryck i *Var*. Vi rekommenderar att du inte utelämnar *Var* såvida inte *Poly* är ett uttryck i en singulär variabel.

$\text{polyDegree}(5)$ 0

$\text{polyDegree}(\ln(2)+\pi,x)$ 0

Konstanta polynom

$\text{polyDegree}(4\cdot x^2-3\cdot x+2,x)$ 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

polyDegree()

Katalog >

Graden kan extraheras även om koefficienterna inte kan extrahera den. Detta beror på att graden kan extraheras utan att utveckla polynomet.

polyEval()

Katalog >

polyEval(*List1***, *Expr1*)⇒*uttryck***

polyEval(*List1***, *List2*)⇒*uttryck***

Tolkar det första argumentet som koefficienten för ett polynom med fallande ordning och ger polynomet utvärderat för det andra argumentets värde.

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(*Expr1*,*Expr2*)⇒*uttryck***

Ger den största gemensamma delaren för de två argumenten.

Expr1 och *Expr2* måste vara polynomuttryck.

Listargument, matrisargument och booleska argument är ej tillåtna.

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*]**)⇒*uttryck***

Ger kvoten på polynomet *Poly1* dividerat med polynomet *Poly2* med avseende på den specificerade variabeln *Var*.

Poly1 och *Poly2* måste vara polynomuttryck i *Var*. Vi rekommenderar att du inte utelämnar *Var* såvida inte *Poly1* och *Poly2* är uttryck i samma variabel.

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·x-y+2·z
polyQuotient((x-y)·(y-z),x+y+z,z)	-(x-y)

polyRemainder()

Katalog >

polyRemainder(*Poly1*,*Poly2* [,*Var*]**)**⇒*uttryck*

Ger resten på polynomet *Poly1* dividerat med polynomet *Poly2* med avseende på den specificerade variabeln *Var*.

Poly1 och *Poly2* måste vara polynomuttryck i *Var*. Vi rekommenderar att du inte utelämnar *Var* såvida inte *Poly1* och *Poly2* är uttryck i samma variabel.

polyRemainder$(x-1,x-3)$ → 2

polyRemainder$(x-1,x^2-1)$ → $x-1$

polyRemainder$(x^2-1,x-1)$ → 0

polyRemainder$((x-y)\cdot(y-z),x+y+z,x)$ → $-(y-z)\cdot(2\cdot y+z)$

polyRemainder$((x-y)\cdot(y-z),x+y+z,y)$ → $-2\cdot x^2-5\cdot x\cdot z-2\cdot z^2$

polyRemainder$((x-y)\cdot(y-z),x+y+z,z)$ → $(x-y)\cdot(x+2\cdot y)$

polyRoots()

Katalog >

polyRoots(*Poly*,*Var***)** ⇒*lista*

polyRoots(*ListaPåKoeff***)** ⇒*lista*

Den första syntaxen, **polyRoots(***Poly*,*Var***)**, ger en lista på reella rötter i polynomet *Poly* med avseende på variabeln *Var*. Om inga reella rötter existerar ger den en tom lista: { }.

Poly måste vara ett polynom i en variabel.

Den andra syntaxen, **polyRoots (***ListaPåKoeff***)** ger en lista på reella rötter för koefficienterna i *ListaPåKoeff*.

Obs: Se även **cPolyRoots()**, på sidan 40.

polyRoots(y^3+1,y) → $\{-1\}$

cPolyRoots(y^3+1,y) → $\left\{-1,\frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i,\frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i\right\}$

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

polyRoots$(\{1,2,1\})$ → $\{-1,-1\}$

PowerReg

Katalog >

PowerReg *X*,*Y* [, *Freq*] [, *Category*, *Include*]]

Utför potensregressionsanalys y = (a ·(x)^b)på listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	**Beskrivning**
stat.RegEqn	Regressionsekvation: a ·(x)^b
stat.a, stat.b	Regressionskoefficienter
stat.r^2	Koefficient för linjär bestämning av transformerade data
stat.r	Korrelationskoefficient för transformerade data (ln(x), ln(y))
stat.Resid	Residualer associerade med potensmodellen
stat.ResidTrans	Residualer associerade med linjär anpassning av transformerade data
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

Prgm

Katalog >

Prgm

Block

EndPrgm

Mall för att skapa ett användardefinierat program. Måste användas med kommandot **Define**, **Define LibPub** eller **Define LibPriv**.

Block kan vara ett enstaka påstående, en serie av påståenden separerade med tecknet ":" eller en serie av påståenden på separata rader.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Beräkna GCD och visa mellanliggande resultat.

```
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()

Se Π(), på sidan 225.

Product (PI)

Se Π(), på sidan 225.

product()

Katalog >

product(*List*[**,** *Start*[**,** *end*]]**)⇒***uttryck*

Ger produkten av elementen i *List*. *Start* och *end* är valfria. De specificerar ett område med element.

product(*Matrix1*[**,** *Start*[**,** *end*]]**)⇒***matris*

Ger en radvektor som innehåller produkterna av elementen i kolumnerna i *Matrix1*. *Start* och *end* är valfria. De specificerar ett område med rader.

```
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]
```

Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

propFrac(*Expr1*[**,** *Var*]**)⇒***uttryck*

propFrac(*rational_number***)** ger *rational_number* som summan av ett heltal och ett bråk med samma tecken och med större nämnare än täljare.

propFrac(*rational_expression*,*Var***)** ger summan av egentliga bråk och ett polynom med avseende på *Var*. Graden hos *Var* i nämnaren överskrider graden hos *Var* i täljaren i varje egentligt bråk. Liknande potenser av *Var* samlas in. Termerna och deras faktorer sorteras med *Var* som huvudvariabel.

Om *Var* utelämnas utförs en utveckling av ett egentligt bråk med avseende på den mest betydande variabeln. Koefficienterna för polynomdelen görs sedan egentliga, först med avseende på deras mest betydande variabel och så vidare.

För rationella uttryck är **propFrac()** ett snabbare, men mindre extremt alternativ till **expand()**.

Du kan använda funktionen **propFrac()** för att representera blandade bråk och demonstrera addition och subtraktion av blandade bråk.

$\text{propFrac}\left(\frac{4}{3}\right)$ $1+\frac{1}{3}$

$\text{propFrac}\left(\frac{-4}{3}\right)$ $-1-\frac{1}{3}$

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

$\text{propFrac}(Ans)$ $\frac{1}{x+1}+x+\frac{1}{y+1}+y$

$\text{propFrac}\left(\frac{11}{7}\right)$ $1+\frac{4}{7}$

$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right)$ $8+\frac{37}{44}$

$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$ $-2-\frac{29}{44}$

Q

QR *Matrix***,** *qMatName***,** *rMatName*[**,** *Tol*]

Siffran för flytande komma (9.) i m1 medför att resultat beräknas med flyttalsaritmetik.

QR

Katalog >

Beräknar faktoriseringen Householder QR av en reell eller komplex matris. De resulterande Q- och R-matriserna lagras i specificerad *MatNames*. Q-matrisen är unitär. R-matrisen är övertriangulär.

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du använder ctrl enter eller ställer in **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E⁻14 ·max(dim(*Matrix*)) ·rowNorm (*Matrix*)

QR-faktoriseringen beräknas numeriskt med Householder-transformationer. Den symboliska lösningen beräknas med Gram-Schmidt. Kolumnerna i *qMatName* är de ortonormerade basvektorerna som täcker utrymmet som definieras av *matrix*.

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9.\end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9.\end{bmatrix}$$

$$\text{QR } m1,qm,rm \qquad \textit{Done}$$

$$qm \qquad \begin{bmatrix}0.123091 & 0.904534 & 0.408248\\ 0.492366 & 0.301511 & -0.816497\\ 0.86164 & -0.301511 & 0.408248\end{bmatrix}$$

$$rm \qquad \begin{bmatrix}8.12404 & 9.60114 & 11.0782\\ 0. & 0.904534 & 1.80907\\ 0. & 0. & 0.\end{bmatrix}$$

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

$$\text{QR } m1,qm,rm \qquad \textit{Done}$$

$$qm \qquad \begin{bmatrix}\frac{m}{\sqrt{m^2+o^2}} & \frac{-\text{sign}(m\cdot p-n\cdot o)\cdot o}{\sqrt{m^2+o^2}}\\ \frac{o}{\sqrt{m^2+o^2}} & \frac{m\cdot\text{sign}(m\cdot p-n\cdot o)}{\sqrt{m^2+o^2}}\end{bmatrix}$$

$$rm \qquad \begin{bmatrix}\sqrt{m^2+o^2} & \frac{m\cdot n+o\cdot p}{\sqrt{m^2+o^2}}\\ 0 & \frac{|m\cdot p-n\cdot o|}{\sqrt{m^2+o^2}}\end{bmatrix}$$

QuadReg

Katalog >

QuadReg *X*,*Y* [, *Freq*] [, *Category*, *Include*]]

Utför den kvadratiska regressionsanalysen y = a ·x²+b ·x+cpå listorna *X* och *Y* med frekvensen *Freq*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se "Tomma element", på sidan 238.

Resultatvariabel	Beskrivning
stat.RegEqn	Regressionsekvation: $a \cdot x^2+b \cdot x+c$
stat.a, stat.b, stat.c	Regressionskoefficienter
stat.R^2	Determinationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

QuartReg

Katalog >

QuartReg *X*,*Y* [, *Freq*] [, *Category*, *Include*]]

Utför en fjärdegrads regressionsanalys y = $a \cdot x^4+b \cdot x^3+c \cdot x^2+d \cdot x+e$ på listorna X och Y med frekvensen $Freq$. En sammanfattning av resultaten visas i variabeln $stat.results$. (Se på sidan 177.)

Alla listor utom $Include$ måste ha samma dimensioner.

X och Y är listor på oberoende och beroende variabler.

$Freq$ är en frivillig lista på frekvensvärden. Varje element i $Freq$ specificerar frekvensen för varje motsvarande X- och Y-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

$Category$ är en lista på kategorikoder för motsvarande X- och Y-data.

$Include$ är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	**Beskrivning**
stat.RegEqn	Regressionsekvation: $a \cdot x^4+b \cdot x^3+c \cdot x^2+d \cdot x+e$
stat.a, stat.b, stat.c, stat.d, stat.e	Regressionskoefficienter
stat.R^2	Determinationskoefficient
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade $X\ List$ som används i regressionen baserat på begränsningar i $Freq$, $Category\ List$ och $Include\ Categories$
stat.YReg	Lista på datapunkter i den modifierade $Y\ List$ som används i regressionen baserat på begränsningar i $Freq$, $Category\ List$ och $Include\ Categories$
stat.FreqReg	Lista på frekvenser som motsvarar $stat.XReg$ och $stat.YReg$

R▶Pθ() Katalog >

R▶Pθ (*xExpr*, *yExpr*)⇒*uttryck*

R▶Pθ (*xList*, *yList*)⇒*lista*

R▶Pθ (*xMatrix*, *yMatrix*)⇒*matris*

Ger den ekvivalenta θ-koordinaten för argumentparen (*x*,*y*).

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`R@>Ptheta(...)`**.

I vinkelläget Grader:

$$R\blacktriangleright P\theta(x,y) \qquad 90\cdot\text{sign}(y)-\tan^{-1}\left(\frac{x}{y}\right)$$

I vinkelläget Nygrader:

$$R\blacktriangleright P\theta(x,y) \qquad 100\cdot\text{sign}(y)-\tan^{-1}\left(\frac{x}{y}\right)$$

I vinkelläget Radianer:

$$R\blacktriangleright P\theta(3,2) \qquad \tan^{-1}\left(\frac{2}{3}\right)$$

$$R\blacktriangleright P\theta\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right) \qquad \begin{bmatrix}0 & \tan^{-1}\left(\frac{16}{\pi}\right)+\frac{\pi}{2} & 0.643501\end{bmatrix}$$

R▶Pr() Katalog >

R▶Pr (*xExpr*, *yExpr*)⇒*uttryck*

R▶Pr (*xList*, *yList*)⇒*lista*

R▶Pr (*xMatrix*, *yMatrix*)⇒*matris*

Ger den ekvivalenta r-koordinaten för argumentparen (*x*,*y*).

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`R@>Pr(...)`**.

I vinkelläget Radianer:

$$R\blacktriangleright Pr(3,2) \qquad \sqrt{13}$$

$$R\blacktriangleright Pr(x,y) \qquad \sqrt{x^2+y^2}$$

$$R\blacktriangleright Pr\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right) \qquad \begin{bmatrix}3 & \frac{\sqrt{\pi^2+256}}{4} & 2.5\end{bmatrix}$$

▶Rad Katalog >

Expr1▶*Rad*⇒*uttryck*

Konvertar argumentet till en vinkel i radianer.

I vinkelläget Grader:

$$(1.5)\blacktriangleright \text{Rad} \qquad (0.02618)^r$$

▸Rad

Katalog >

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>Rad**.

I vinkelläget Nygrader:

$(1.5)\blacktriangleright\text{Rad}$	$(0.023562)^r$

rand()

Katalog >

rand()⇒*uttryck*

rand(*#Trials***)**⇒*lista*

rand() ger ett slumpvärde mellan 0 och 1.

rand(*#Trials***)** ger en lista med *#Trials* slumpvärden mellan 0 och 1.

Bestämmer slumptalsfröet.

RandSeed 1147	*Done*
rand(2)	{0.158206,0.717917}

randBin()

Katalog >

randBin(*n***,** *p***)**⇒*uttryck*

randBin(*n***,** *p***,** *#Trials***)**⇒*lista*

randBin(*n***,** *p***)** ger ett reellt slumptal från en specificerad binomialfördelning.

randBin(*n***,** *p***,** *#Trials***)** ger en lista med *#Trials* reella slumptal från en specificerad binomialfördelning.

randBin(80,0.5)	42
randBin(80,0.5,3)	{41,32,39}

randInt()

Catalog >

randInt(*lowBound***,***upBound***)**⇒*uttryck*

randInt(*lowBound***,***upBound***,***#Trials***)**⇒*lista*

randInt(*lowBound***,***upBound***)** ger ett slumptal med heltalsvärde inom det område som specificeras av heltalsgränserna *lowBound* och *upBound*.

randInt(*lowBound***,***upBound***,***#Trials***)** ger en lista med *#Trials* slumptal med heltalsvärden inom det specificerade området.

randInt(3,10)	5
randInt(3,10,4)	{9,7,5,8}

randMat()

Katalog >

randMat(*numRows***, *numColumns*)⇒*matris***

Ger en matris med heltal mellan -9 och 9 med specificerad dimension.

Båda argumenten måste förenklas till heltal.

RandSeed 1147	*Done*
randMat(3,3)	$\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$

Obs: Värdena i denna matris ändras varje gång du trycker på enter.

randNorm()

Katalog >

randNorm(μ**, σ)⇒*uttryck***

randNorm(μ**, σ, *#Trials*)⇒*lista***

Ger ett decimalt tal från den specificerade normalfördelningen. Det kan vara ett reellt tal, men det blir kraftigt koncentrerat i intervallet [μ−3 ·σ, μ+3 ·σ].

randNorm(μ**, σ, *#Trials*)** ger en lista med *#Trials* decimaltal från den specificerade normalfördelningen.

RandSeed 1147	*Done*
randNorm(0,1)	0.492541
randNorm(3,4.5)	−3.54356

randPoly()

Katalog >

randPoly(*Var*, *Order*)⇒*uttryck*

Ger ett polynom i *Var* med specificerad *Order*. Koefficienterna är slumpheltal i området −9 t.o.m. 9. Den första koefficienten är inte noll.

Order måste vara 0–99.

RandSeed 1147	*Done*
randPoly(*x*,5)	$-2\cdot x^5+3\cdot x^4-6\cdot x^3+4\cdot x-6$

randSamp()

Katalog >

randSamp(*List*,*numTrials*[,*noRepl*])⇒*lista*

Ger en lista på ett slumpmässigt urval av *numTrials* försök från *List* med ett alternativ för urvalsutbyte (*noRepl*=0) eller inget urvalsutbyte (*noRepl*=1). Förinställningen är med urvalsutbyte.

Define *list3*={1,2,3,4,5}	*Done*
Define *list4*=randSamp(*list3*,6)	*Done*
list4	{2,3,4,3,1,2}

RandSeed

Katalog >

RandSeed *Number*

Om *Number* = 0 ställs fröna in på fabriksinställningarna för slumptalsgeneratorn. Om *Number* ≠ 0 används det för att generera två frön, vilka lagras i systemvariablerna seed1 och seed2.

RandSeed 1147	*Done*
rand()	0.158206

real()

Katalog >

real(*Expr1***)**⇒*uttryck*

Ger argumentets reella del.

Obs: Alla odefinierade variabler behandlas som reella variabler. Se även **imag()**, på sidan 90.

real(2+3·*i*)	2
real(*z*)	*z*
real(*x*+*i*·*y*)	*x*

real(*List1***)**⇒*lista*

Ger de reella delarna av alla element.

real({*a*+*i*·*b*, 3, *i*})	{*a*, 3, 0}

real(*Matrix1***)**⇒*matris*

Ger de reella delarna av alla element.

$$\text{real}\left(\begin{bmatrix} a+i\cdot b & 3 \\ c & i \end{bmatrix}\right) \qquad \begin{bmatrix} a & 3 \\ c & 0 \end{bmatrix}$$

▶Rect

Katalog >

Vector **▶Rect**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva `@>Rect`.

Visar *Vector* i rektangulär form [x, y, z]. Vektorn måste ha dimensionen 2 eller 3 och kan vara en rad eller en kolumn.

Obs: **▶Rect** är en visa format-instruktion, inte en konverteringsfunktion. Du kan endast använda den i slutet av en inmatningsrad, och den uppdaterar inte *ans*.

Obs: Se även **▶Polar**, på sidan 134.

$$\left(\begin{bmatrix} 3 & \angle\frac{\pi}{4} & \angle\frac{\pi}{6} \end{bmatrix}\right) \blacktriangleright \text{Rect}$$

$$\begin{bmatrix} \frac{3\cdot\sqrt{2}}{4} & \frac{3\cdot\sqrt{2}}{4} & \frac{3\cdot\sqrt{3}}{2} \end{bmatrix}$$

$$\begin{bmatrix} a & \angle b & \angle c \end{bmatrix}$$

$$\begin{bmatrix} a\cdot\cos(b)\cdot\sin(c) & a\cdot\sin(b)\cdot\sin(c) & a\cdot\cos(c) \end{bmatrix}$$

complexValue **▶Rect**

I vinkelläget Radianer:

▸Rect

Katalog >

Visar *complexValue* i rektangulär form a+bi. *complexValue* kan ha valfri komplex form. En inmatning av re$^{i\theta}$ orsakar dock ett fel i vinkelläget Grader.

Obs: Du måste använda parenteserna för en (r∠θ) polär inmatning.

$\left(4 \cdot e^{\frac{\pi}{3}}\right)$▸Rect $\quad 4 \cdot e^{\frac{\pi}{3}}$

$\left(\left(4 \angle \frac{\pi}{3}\right)\right)$▸Rect $\quad 2+2 \cdot \sqrt{3} \cdot i$

I vinkelläget Nygrader:

$((1 \angle 100))$▸Rect $\quad i$

I vinkelläget Grader:

$((4 \angle 60))$▸Rect $\quad 2+2 \cdot \sqrt{3} \cdot i$

Obs: För att skriva in tecknet ∠, välj det från symbollistan i Catalog.

ref()

Katalog >

ref(*Matrix1***[, *Tol*])⇒*matris***

Ger radtrappstegsformen av *Matrix1*.

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du använder ctrl enter eller ställer in **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E⁻14 ·max(dim(*Matrix1*)) ·rowNorm (*Matrix1*)

Undvik odefinierade element i *Matris1*. De kan leda till oväntade resultat.

Om till exempel *a* är odefinierat i följande uttryck visas ett varningsmeddelande och resultatet ges som:

$\text{ref}\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}$ $\quad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$

$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow m1$ $\quad \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$\text{ref}(m1)$ ⚠ $\quad \begin{bmatrix} 1 & \frac{d}{c} \\ 0 & 1 \end{bmatrix}$

ref()

Katalog >

$$\text{ref}\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Varningen visas på grund av att det generaliserade elementet 1/*a* inte skulle vara giltigt för *a*=0.

Du kan undvika detta genom att i förväg lagra ett värde i *a* eller genom att använda ("|")-operatorn begränsning för att ersätta ett värde såsom visas i följande exempel.

$$\text{ref}\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} | a=0 \qquad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Obs: Se även **rref()**, på sidan 157.

remain()

Katalog >

remain(*Expr1***,** *Expr2***)⇒***uttryck*

remain(*List1***,** *List2***)⇒***lista*

remain(*Matrix1***,** *Matrix2***)⇒***matris*

Ger resten på det första argumentet med avseende på det andra argumentet definierat av identiteterna:

remain(x,0) x

remain(x,y) x−y·iPart(x/y)

Som en följd av detta, obeservera att **remain(**−x,y**)** −**remain(**x,y**)**. Resultatet är antingen noll eller har samma tecken som det första argumentet.

Obs: Se även **mod()**, på sidan 117.

remain(7,0)	7
remain(7,3)	1
remain(−7,3)	−1
remain(7,−3)	1
remain(−7,−3)	−1
remain({12,−14,16},{9,7,−5})	{3,0,1}

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

Request

Katalog >

Request*promptString*, *var*[, *DispFlag* [, *statusVar*]]

Definiera ett program:

```
Definiera begär_demo()=Prgm
```

Request*promptString*, *func*(*arg1*, *...argn*) [, *DispFlag* [, *statusVar*]]

Programmeringskommando: Pausar programmet och visar en dialogruta med meddelandet *frågeSträng* och en svarsruta där användaren ska skriva in svaret.

När användaren skriver in svaret och klickar på **OK** tilldelas innehållet i svarsrutan till variabeln *var*.

Om användaren klickar på **Cancel** (Avbryt) fortsätter programmet utan att acceptera någon inmatning. Programmet använder det tidigare värdet på *var* om *var* redan var definierad.

Det valfria argumentet *DispFlagga* kan vara ett valfritt uttryck.

- Om *DispFlagga* utelämnas eller beräknas till **1** visas promptmeddelandet och användarens svar i Räknarens historik.
- Om *DispFlagga* beräknas till **0** visas inte prompten och svaret i historiken.

Det vafria argumentet *statusVar* ger programmet ett sätt att bestämma hur användaren lämnade dialogrutan. Observera att *statusVar* kräver argumentet *DispFlag*.

- Om användaren klickade på **OK** eller tryckte på **Enter** eller **Ctrl+Enter** ställs variabeln *statusVar* in på värdet **1**.
- Annars ställs variabeln *statusVar* in på värdet **0**.

Argumentet *funk*() låter programmet lagra användarens svar som en funktionsdefinition. Denna syntax fungerar som om användaren exekverade kommandot:

Definiera *funk*(*arg1*, *...argn*) = *användarens svar*

```
  Begär "Radie: ",r
  Disp "Area = ",pi*r^2
EndPrgm
```

Kör programmet och skriv in ett svar:

```
begär_demo()
```

Resultat efter tryckning på **OK**:

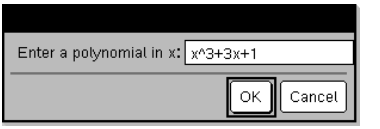
```
Radie: 6/2
Area= 28.2743
```

Definiera ett program:

```
Definiera polynom()=Prgm
  Begär "Skriv in ett polynom i
x:",p(x)
  Disp "Reella rötter
är:",polyRoots(p(x),x)
EndPrgm
```

Kör programmet och skriv in ett polynom:

```
polynom()
```



Resultat efter tryckning på **OK**:

```
Skriv in ett polynom i x: x^3+3x+1
Reella rötter är: {-0.322185}
```

Request

Katalog >

Programmet kan sedan använda den definierade funktionen *funk*(). Strängen *frågeSträng* bör vägleda användaren till att skriva in ett lämpligt *användarens svar* som fullbordar funktionsdefinitionen.

Obs: Du kan använda kommandot **Request** inom ett användardefinierat program, men inte inom en funktion.

För att stoppa ett program som innehåller ett **Request**-kommando inne i en oändlig slinga:

- **Handenhet:** Håll ned on och tryck på enter upprepade gånger.
- **Windows®:** Håll ned **F12** och tryck på **Enter** upprepade gånger.
- **Macintosh®:** Håll ned **F5** och tryck på **Enter** upprepade gånger.
- **iPad®:** Appen visar en uppmaning. Du kan fortsätta att vänta eller avbryta.

Obs: Se även **RequestStr**, på sidan 151.

RequestStr

Katalog >

RequestStr*frågeSträng*, *var*[, *DispFlagga*]

Programmeringskommando: Fungerar på precis samma sätt som den första syntaxen i kommandot **Request**, förutom att användarens svar alltid tolkas som en sträng. Kommandot Request tolkar svaret som ett uttryck såvida inte användaren omsluter det med citationstecken (“”).

Obs: Du kan använda kommandot **RequestStr** inom ett användardefinierat program, men inte inom en funktion.

För att stoppa ett program som innehåller ett **RequestStr**-kommando inne i en oändlig slinga:

- **Handenhet:** Håll ned on och tryck på enter upprepade gånger.
- **Windows®:** Håll ned **F12** och tryck på

Definiera ett program:

```
Definiera begärStr_demo()=Prgm
  BegärStr “Ditt namn:”,namn,0
  Disp “Svaret har “,dim(namn),”
tecken.”
EndPrgm
```

Kör programmet och skriv in ett svar:

```
begärStr_demo()
```

RequestStr

Katalog >

Enter upprepade gånger.

- **Macintosh®:** Håll ned **F5** och tryck på **Enter** upprepade gånger.
- **iPad®:** Appen visar en uppmaning. Du kan fortsätta att vänta eller avbryta.

Obs: Se även **Request**, på sidan 149.

Resultat efter tryckning på **OK** (observera att argumentet *DispFlagga* för **0** förbiser prompten och svarar från historiken):

```
begärStr_demo()
Svaret har 5 tecken.
```

Return

Katalog >

Return [*Expr*]

Ger *Expr* som resultatet av funktionen. Använd inom ett **Func**...**EndFunc**-block.

Obs: Använd **Return** utan ett argument inom ett **Prgm**...**EndPrgm**-block för att gå ur ett program.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
Define factorial(nn)=
Func
Local answer,counter
1→answer
For counter,1,nn
answer·counter→answer
EndFor
Return answer
EndFunc
```

factorial(3) 6

right()

Katalog >

right(*List1*[**,** *Num*]**)⇒***lista*

Ger *Num*-elementen längst till höger i *List1*.

Om du utelämnar *Num* erhålls alla i *List1*.

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

right(*sourceString*[**,** *Num*]**)⇒***sträng*

Ger *Num*-tecknen längst till höger i teckensträngen *sourceString*.

Om du utelämnar *Num* erhålls alla i *sourceString*.

right("Hello",2) "lo"

right(*Comparison***)⇒***uttryck*

Ger den högra sidan av en ekvation eller olikhet.

right(x<3) 3

rk23 ()

Katalog >

rk23(*Expr***,** *Var***,** *depVar***, {***Var0***,** *VarMax***},**

Differentialekvation:

depVar0, *VarStep* **[**, *diftol*]**)**⇒*matris*

rk23(*SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *diftol*]**)** ⇒*matris*

rk23(*ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *diftol*]**)** ⇒*matris*

Använder Runge-Kuttas metod för att lösa systemet.

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

med *depVar*(*Var0*)=*depVar0* på intervallet [*Var0*,*VarMax*]. Ger en matris vars första rad definierar resultatvärdena för *Var*, definierade enligt *VarStep*. Den andra raden definierar värdet på den första lösningskomponenten vid motsvarande värden på *Var*, och så vidare.

Expr är det högra ledet som definierar den ordinära differentialekvationen (ODE).

SystemOfExpr är ett system av högerled som definierar systemet av ODE:er (motsvarar ordningen av oberoende variabler i *ListOfDepVars*).

ListOfExpr är en lista på högerled som definierar systemet av ODE:er (motsvarar ordningen av oberoende variabler i *ListOfDepVars*).

Var är den oberoende variabeln.

ListOfDepVars är en lista på oberoende variabler.

{*Var0*, *VarMax*} är en lista med två element som instruerar funktionen att integrera från *Var0* till *VarMax*.

ListOfDepVars0 är en lista på startvärden för oberoende variabler.

y'=0,001*y*(100-y) och 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

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Samma ekvation med *diftol* inställd på 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

Jämför ovanstående resultat med CAS exakta lösning som erhållits med deSolve() och seqGen():

```
deSolve(y'=0.001·y·(100−y) and y(0)=10,t,y)
```

$$y=\frac{100.\cdot(1.10517)^t}{(1.10517)^t+9.}$$

```
seqGen(100.·(1.10517)^t/((1.10517)^t+9.),t,y,{0,100})
```

{10.,10.9367,11.9494,13.0423,14.2189,15.48

Ekvationssystem:

$$\begin{cases} y1' = -y1 + 0.1 \cdot y1 \cdot y2 \\ y2' = 3 \cdot y2 - y1 \cdot y2 \end{cases}$$

med *y1*(0)=2 och *y2*(0)=5

```
rk23({-y1+0.1·y1·y2, 3·y2−y1·y2},t,{y1,y2},{0,5},{2,5},1)
```

0.	1.	2.	3.	4.
2.	1.94103	4.78694	3.25253	1.82848
5.	16.8311	12.3133	3.51112	6.27245

rk23 ()

Katalog >

Om *VarStep* utvärderas till ett tal skilt från noll ges sign(*VarStep*) = sign(*VarMax*-*Var0*) och lösningar vid *Var0*+i***VarStep* för alla i=0,1,2,... sådana att *Var0*+i***VarStep* är i intervallet [*var0*,*VarMax*] (kanske inte ger ett lösningsvärde vid *VarMax*).

Om *VarStep* utvärderas till noll ges lösningar vid "Runge-Kutta"-värdena för *Var*.

diftol är feltoleransen (förinställs på 0,001).

root()

Katalog >

root(*Expr***)⇒** *rot*

root(*Expr1***, *Expr2***)⇒** *rot*

root(*Expr***) ger kvadratroten ur** *Expr***.**

root(*Expr1***, *Expr2***)** ger *Expr2*-roten ur *Expr1*. *Expr1* kan vara en reell eller komplex konstant i flyttalsform, ett heltal eller komplex rationell konstant eller ett allmänt symboliskt uttryck.

Obs: Se även **Nth root template**, på sidan 5.

$\sqrt[3]{8}$	2
$\sqrt[3]{3}$	$3^{\frac{1}{3}}$
$\sqrt[3]{3.}$	1.44225

rotate()

Katalog >

rotate(*Integer1*[**,***#ofRotations*]**)⇒***heltal*

Roterar bitarna i ett binärt heltal. Du kan skriva in *Integer1* i valfri talbas. Det omvandlas automatiskt till 64-bitars binär form. Om storleken på *Integer1* är alltför stor för denna form för en symmetrisk modulooperation talet inom området. För mer information, se ▶**Base2**, på sidan 21.

Om *#ofRotations* är positiv sker rotationen åt vänster. Om *#ofRotations* är negativ sker rotationen åt höger. Förinställningen är ⁻1 (rotera en bit åt höger).

Vid exempelvis höger rotation:

I binärt basläge:

rotate(0b11111111111111111111111111111111)	0b10000000000000000000000000000001
rotate(256,1)	0b1000000000

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

I hexadecimalt basläge:

rotate(0h78E)	0h3C7
rotate(0h78E,⁻2)	0h80000000000001E3
rotate(0h78E,2)	0h1E38

rotate() Katalog >

Varje bit roteras åt höger.

0b00000000000000011110101100000110101

Biten längst till höger roteras till positionen längst till vänster.

ger:

0b10000000000000011110101100000110101

Resultatet visas enligt det inställda basläget.

Viktigt: När du skriver in ett binärt eller hexadecimalt tal, använd alltid prefixet 0b eller 0h (noll, inte bokstaven O).

rotate(*List1*[**,***#ofRotations*]**)⇒***lista*

Ger en kopia av *List1* roterad åt höger eller vänster av *#ofRotations*-elementen. Ändrar inte *List1*.

Om *#ofRotations* är positiv sker rotationen åt vänster. Om *#ofRotations* är negativ sker rotationen åt höger. Förinställningen är −1 (rotera ett element åt höger).

I decimalt basläge:

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*[**,***#ofRotations*]**)⇒***sträng*

Ger en kopia av *String1* roterad åt höger eller vänster av *#ofRotations*-tecknen. Ändrar inte *String1*.

Om *#ofRotations* är positiv sker rotationen åt vänster. Om *#ofRotations* är negativ sker rotationen åt höger. Förinställningen är −1 (rotera ett tecken åt höger).

rotate("abcd")	"dabc"
rotate("abcd",-2)	"cdab"
rotate("abcd",1)	"bcda"

round() Katalog >

round(*Expr1*[**,** *digits*]**)⇒***uttryck*

Ger argumentet avrundat till det specificerade antalet siffror efter decimalpunkten.

digits måste vara ett heltal i området 0–12. Om *digits* inte inkluderas erhålls argumentet avrundat till 12 signifikanta siffror.

Obs: Läget Display digits (Visa siffror) kan påverka hur detta visas.

round(1.234567,3)	1.235

round() Katalog >

round(*List1*[**,** *digits*]**)⇒***lista*

Ger en lista på elementen avrundade till det specificerade antalet siffror.

round(*Matrix1*[**,** *digits*]**)⇒***matris*

Ger en matris över elementen avrundade till det specificerade antalet siffror.

$\text{round}(\{\pi,\sqrt{2},\ln(2)\},4)$
$\{3.1416,1.4142,0.6931\}$

$\text{round}\begin{pmatrix}\begin{bmatrix}\ln(5) & \ln(3)\\ \pi & e^1\end{bmatrix},1\end{pmatrix}$ $\begin{bmatrix}1.6 & 1.1\\ 3.1 & 2.7\end{bmatrix}$

rowAdd() Katalog >

rowAdd(*Matrix1***,** *rIndex1***,** *rIndex2***)⇒***matris*

Ger en kopia av *Matrix1* med rad *rIndex2* ersatt av summan av raderna *rIndex1* och *rIndex2*.

$\text{rowAdd}\begin{pmatrix}\begin{bmatrix}3 & 4\\ -3 & -2\end{bmatrix},1,2\end{pmatrix}$ $\begin{bmatrix}3 & 4\\ 0 & 2\end{bmatrix}$

$\text{rowAdd}\begin{pmatrix}\begin{bmatrix}a & b\\ c & d\end{bmatrix},1,2\end{pmatrix}$ $\begin{bmatrix}a & b\\ a+c & b+d\end{bmatrix}$

rowDim() Katalog >

rowDim(*Matrix***)⇒***uttryck*

Ger antalet rader i *Matrix*.

Obs: Se även **colDim()**, på sidan 30.

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

$\text{rowDim}(m1)$ 3

rowNorm() Katalog >

rowNorm(*Matrix***)⇒***uttryck*

Ger maximum av summorna av absolutbeloppen på elementen i raderna i *Matrix*.

Obs: Alla matriselement måste förenklas till tal. Se även **colNorm()**, på sidan 30.

$\text{rowNorm}\begin{pmatrix}\begin{bmatrix}-5 & 6 & -7\\ 3 & 4 & 9\\ 9 & -9 & -7\end{bmatrix}\end{pmatrix}$ 25

rowSwap() Katalog >

rowSwap(*Matrix1***,** *rIndex1***,** *rIndex2***)⇒***matris*

Ger *Matrix1* med raderna *rIndex1* och *rIndex2* växlade.

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

$\text{rowSwap}(mat,1,3)$ $\begin{bmatrix}5 & 6\\ 3 & 4\\ 1 & 2\end{bmatrix}$

rref() Katalog >

rref(*Matrix1*[**,** *Tol*]**)⇒***matris*

Ger den reducerade radtrappstegsformen av *Matrix1*.

$$\text{rref}\begin{bmatrix}-2 & -2 & 0 & -6\\ 1 & -1 & 9 & -9\\ -5 & 2 & 4 & -4\end{bmatrix} \qquad \begin{bmatrix}1 & 0 & 0 & \frac{66}{71}\\ 0 & 1 & 0 & \frac{147}{71}\\ 0 & 0 & 1 & \frac{-62}{71}\end{bmatrix}$$

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

$$\text{rref}\begin{bmatrix}a & b\\ c & d\end{bmatrix} \qquad \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

- Om du använder ctrl enter eller ställer in **Auto eller Ungefärlig** på Approximate utförs beräkningarna med flyttalsaritmetik.
- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E⁻14 ·max(dim(*Matrix1*)) ·rowNorm (*Matrix1*)

Obs: Se även **ref()**, på sidan 148.

S

sec() trig tangent

sec(*Expr1***) ⇒** *uttryck*

sec(*List1***) ⇒** *lista*

Ger sekansfunktionen för *Expr1* eller en lista på sekansfunktionerna för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i grader, nygrader eller radianer enligt det inställda vinkelläget. Du kan använda °, G eller r för att tillfälligt överstyra vinkelläget.

I vinkelläget Grader:

$$\sec(45) \qquad \sqrt{2}$$

$$\sec(\{1,2.3,4\}) \qquad \left\{\frac{1}{\cos(1)}, 1.00081, \frac{1}{\cos(4)}\right\}$$

sec⁻¹() trig tangent

sec⁻¹(*Expr1***) ⇒** *uttryck*

I vinkelläget Grader:

sec⁻¹() — trig tangent

sec⁻¹(*List1***)** ⇒ *lista*

Ger den vinkel vars sekansfunktion är *Expr1* eller en lista på de inversa sekansfunktionerna för alla element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **arcsec(...)**.

$\sec^{-1}(1)$ → 0

I vinkelläget Nygrader:

$\sec^{-1}(\sqrt{2})$ → 50

I vinkelläget Radianer:

$\sec^{-1}(\{1,2,5\})$ → $\left\{0, \frac{\pi}{3}, \cos^{-1}\left(\frac{1}{5}\right)\right\}$

sech() — Katalog >

sech(*Expr1***)** ⇒ *uttryck*

sech(*List1***)** ⇒ *lista*

Ger den hyperboliska sekansfunktionen för *Expr1* eller en lista på de hyperboliska sekansfunktionerna för elementen i *List1*.

$\operatorname{sech}(3)$ → $\frac{1}{\cosh(3)}$

$\operatorname{sech}(\{1,2.3,4\})$ → $\left\{\frac{1}{\cosh(1)}, 0.198522, \frac{1}{\cosh(4)}\right\}$

sech⁻¹() — Katalog >

sech⁻¹(*Expr1***)** ⇒ *uttryck*

sech⁻¹ (*List1***)** ⇒ *lista*

Ger den inversa hyperboliska sekansfunktionen för *Expr1* eller en lista på den inversa hyperboliska sekansfunktionen för alla element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **arcsech(...)**.

I vinkelläget Radianer och i Rektangulärt komplext läge:

$\operatorname{sech}^{-1}(1)$ → 0

$\operatorname{sech}^{-1}(\{1,-2,2.1\})$ → $\left\{0, \frac{2\cdot\pi}{3}\cdot i, 8.\text{E}{-}15 + 1.07448\cdot i\right\}$

Send — Hubb-meny

Send*UttrEllerSträng1* [**,** *UttrEllerSträng2*] ...

Programmeringskommando: Skickar ett eller flera TI-Innovator™ Hub kommandon till en ansluten hubb.

exprOrString måste vara ett giltigt TI-Innovator™ Hub Kommando. *exprOrString* innehåller vanligen ett **"SET ..."** -kommando för att styra en enhet eller ett **"READ ..."** -kommando för att begära data.

Argumenten skickas till hubben i turordning.

Obs: Du kan använda kommandot **Send** i ett användardefinierat program, men inte i en funktion.

Obs: Se även **Get** (på sidan 82), **GetStr** (på sidan 85) och **eval()** (på sidan 65).

Exempel: Slå på den blå komponenten i den inbyggda RGB-lysdioden i 0,5 sekunder.

```
Send "SET COLOR.BLUE ON TIME .5"
                                Done
```

Exempel: Begär nuvarande värde från hubbens inbyggda ljusnivåsensor. Ett **Get**-kommando hämtar värdet och tilldelar det till variabeln *lightval*.

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

Exempel: Skicka en beräknad frekvens till hubbens inbyggda högtalare. Använd den speciella variabeln *iostr.SendAns* för att visa hubb-kommandot med det beräknade uttrycket.

$n:=50$	50
$m:=4$	4
Send "SET SOUND eval(m· n)"	*Done*
iostr.SendAns	"SET SOUND 200"

seq() — Katalog >

seq(*Expr***,** *Var***,** *Low***,** *High***[,** *Step***])⇒***lista*

Ökar *Var* från *Low* till *High* i steg om *Step*, utvärderar *Expr* och ger resultaten som en lista. De ursprungliga innehållen i *Var* är fortfarande där när **seq()** har beräknats.

Det förinställda värdet på *Step* = 1.

$\text{seq}\left(n^2,n,1,6\right)$	$\{1,4,9,16,25,36\}$
$\text{seq}\left(\frac{1}{n},n,1,10,2\right)$	$\left\{1,\frac{1}{3},\frac{1}{5},\frac{1}{7},\frac{1}{9}\right\}$
$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$	$\frac{1968329}{1270080}$

Obs: För att få ett närmevärde,

Handenhet: Tryck på ctrl enter.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘**+Enter**.
iPad®: Håll ned **enter** och välj ≈.

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$	1.54977

seqGen() Katalog >

seqGen(*Expr*, *Var*, *depVar*, {*Var0*, *VarMax*}[, *ListOfInitTerms* [, *VarStep* [, *CeilingValue*]]]) ⇒*lista*

Genererar en lista på termer för talföljden *depVar*(*Var*)=*Expr* enligt följande: Ökar den oberoende variabeln *Var* från *Var0* till *VarMax* i steg om *VarStep*, utvärderar *depVar*(*Var*) för motsvarande värden på *Var* med formeln *Expr* och *ListOfInitTerms* och ger resultaten som en lista.

seqGen(*ListOrSystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*} [, *MatrixOfInitTerms* [, *VarStep* [, *CeilingValue*]]]) ⇒*matris*

Genererar en matris med termer för ett system (eller en lista) av talföljder *ListOfDepVars* (*Var*)=*ListOrSystemOfExpr* enligt följande: Ökar den oberoende variabeln *Var* från *Var0* till *VarMax* i steg om *VarStep*, utvärderar *ListOfDepVars*(*Var*) för motsvarande värden på *Var* med formeln *ListOrSystemOfExpr* och *MatrixOfInitTerms* och ger resultaten som en matris.

De ursprungliga innehållen i *Var* är oförändrade när **seqGen()** har beräknats.

Det förinställda värdet på *VarStep* = **1**.

Genererar de första 5 termerna i talföljden $u(n) = u(n\text{-}1)^2/2$, med $u(1)$=**2** och *VarStep*=**1**.

$$\text{seqGen}\left\{\frac{(u(n-1))^2}{n}, n, u, \{1,5\}, \{2\}\right\}$$
$$\left\{2, 2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\}$$

Exempel i vilket Var0=2:

$$\text{seqGen}\left\{\frac{u(n-1)+1}{n}, n, u, \{2,5\}, \{3\}\right\}$$
$$\left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\}$$

Exempel i vilket den initiala termen är symbolisk:

$$\text{seqGen}\{u(n-1)+2, n, u, \{1,5\}, \{a\}\}$$
$$\{a, a+2, a+4, a+6, a+8\}$$

System med två talföljder:

$$\text{seqGen}\left\{\left\{\frac{1}{n}, \frac{u2(n-1)}{2}+u1(n-1)\right\}, n, \{u1,u2\}, \{1,5\}, \begin{bmatrix} \_ \\ 2 \end{bmatrix}\right\}$$
$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

Obs: Tomrummet (_) i den initiala termmatrisen ovan används för att indikera att den initiala termen för u1(n) beräknas med den explicita talföljdsformeln u1(n)=1/n.

seqn() Katalog >

seqn(*Expr*(**u**, *n* [, *ListOfInitTerms*[, *nMax* [, *CeilingValue*]]])⇒*lista*

Genererar de första 6 termerna i talföljden $u(n) = u(n\text{-}1)/2$, med $u(1)$=**2**.

seqn() Katalog >

Genererar en lista på termer för en talföljd, $u(n)=Expr(u, n)$, enligt följande: Ökar n från 1 till $nMax$ i steg om 1, utvärderar $u(n)$ för motsvarande värden på n med formeln $Expr(u, n)$ och $ListOfInitTerms$ och ger resultaten som en lista.

seqn(*Expr***(***n* [**,** *nMax* [**,** *CeilingValue*]]**)**⇒*lista*

Genererar en lista på termer för en icke-rekursiv talföljd, $u(n)=Expr(n)$, enligt följande: Ökar n från 1 till $nMax$ i steg om 1, utvärderar $u(n)$ för motsvarande värden på n med formeln $Expr(n)$ och ger resultaten som en lista.

Om $nMax$ saknas ställs $nMax$ in på 2500

Om $nMax$=0 ställs $nMax$ in på 2500

Obs: seqn() anropar **seqGen( )** med $n0$=**1** och $nstep$=**1**

$$\text{seqn}\left(\frac{u(n-1)}{n},\{2\},6\right) \qquad \left\{2,1,\frac{1}{3},\frac{1}{12},\frac{1}{60},\frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2},6\right) \qquad \left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16},\frac{1}{25},\frac{1}{36}\right\}$$

series() Catalog >

series(*Expr1***,** *Var***,** *Order* [**,** *Point*]**)**⇒*uttryck*

series(*Expr1***,** *Var***,** *Order* [**,** *Point*]**) |** *Var***>***Point*⇒*uttryck*

series(*Expr1***,** *Var***,** *Order* [**,** *Point*]**) |** *Var***<***Point*⇒*uttryck*

Ger en generaliserad trunkerad potensserierepresentation av $Expr1$ utvecklat kring $Point$ genom $Order$-graden $Order$ kan vara ett rationellt tal vilket som helst. De resulterande potenserna av ($Var - Point$) kan inkludera negativa och/eller exponenter i bråkform. Koefficienterna för dessa potenser kan inkludera logaritmer av ($Var - Point$) och andra funktioner av Var som domineras av alla potenser av ($Var - Point$) som har samma exponenttecken.

$$\text{series}\left(\frac{1-\cos(x-1)}{(x-1)^2},x,4,1\right) \qquad \frac{1}{2}-\frac{(x-1)^2}{24}+\frac{(x-1)^4}{720}$$

$$\text{series}\left(\frac{-1}{e^{z_-}},z_-,1\right) \qquad z_-1$$

$$\text{series}\left(\left(1+\frac{1}{n}\right)^n,n,2,\infty\right) \qquad e-\frac{e}{2\cdot n}+\frac{11\cdot e}{24\cdot n^2}$$

$$\text{series}\left(\tan^{-1}\left(\frac{1}{x}\right),x,5\right)\Big|x>0 \qquad \frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}$$

$$\text{series}\left(\int\frac{\sin(x)}{x}\,dx,x,6\right) \qquad x-\frac{x^3}{18}+\frac{x^5}{600}$$

$$\text{series}\left(\int_0^x \sin(x\cdot\sin(t))\,dt,x,7\right) \qquad \frac{x^3}{2}-\frac{x^5}{24}-\frac{29\cdot x^7}{720}$$

$$\text{series}\left(\left(1+e^x\right)^2,x,2,1\right) \qquad (e+1)^2+2\cdot e\cdot(e+1)\cdot(x-1)+e\cdot(2\cdot e+1)\cdot(x-1)^2$$

Point förinställs till 0. *Point* kan vara ∞ eller −∞, varvid utvecklingen sker genom *Order*-graden i 1/(*Var* − *Point*).

series(...) ger "**series(...)**" om den inte kan bestämma en sådan representation, till exempel, för väsentliga singulariteter såsom **sin(**1/*z***)** vid *z*=0, $e^{-1/z}$ vid z=0, eller e^{z} vid z = ∞ eller −∞.

Om serien eller någon av dess derivator har en språngdiskontinuitet vid *Point* innehåller resultatet troligen deluttryck i formen sign (...) eller abs(...) för en reell expansionsvariabel, eller $(-1)^{\text{floor(...angle(...)...)}}$ för en komplex expansionsvariabel, vilken slutar med "_". Om du tänker använda serien endast för värden på ena sidan av *Point*, lägg då till den lämpliga av "| *Var* > *Point*", "| *Var* < *Point*", "| "*Var* ≥ *Point*" eller "*Var* ≤ *Point*" för att erhålla ett enklare resultat.

series() kan ge symboliska approximationer av obestämda och bestämda integraler, för vilka symboliska lösningar annars inte kan erhållas.

series() fördelas över 1:a-argumentlistor och matriser.

series() är en generaliserad version av **taylor()**.

Som det sista exemplet till höger illustrerar kan visningsrutinerna nedåt i resultatet som produceras av series(...) arrangera om termer så att den dominanta termen inte är längst till vänster.

Obs: Se även dominantTerm(), på sidan 59.

setMode() Katalog >

setMode(*modeNameInteger***,** *settingInteger***)** ⇒*heltal*

setMode(*list***)** ⇒*lista på heltal*

Visa ungefärligt värde på π med förinställningen för Display Digits (Visa siffror) och visa sedan π med inställningen Fix2. Kontrollera sedan att förinställningen har återställts efter exekveringen av programmet.

Endast giltigt inom en funktion eller ett program.

setMode(*modeNameInteger***,** *settingInteger***)** ställer temporärt in läget *modeNameInteger* på den nya inställningen *settingInteger* och ger ett heltal som motsvarar den ursprungliga inställningen på det läget. Ändringen är begränsad till den tid det tar att exekvera programmet/funktionen.

modeNameInteger specificerar vilket läge du vill ställa in. Det måste vara något av de lägesheltal som anges i nedanstående tabell.

settingInteger specificerar den nya inställningen för läget. Det måste vara något av de inställningsheltal som anges i nedanstående tabell för det läge som du ställer in.

setMode(*list***)** låter dig ändra flera inställningar. *list* innehåller par av lägesheltal och inställningsheltal.. **setMode (***list***)** ger en liknande lista vars heltalspar representerar de ursprungliga lägena och inställningarna.

Om du har sparat alla lägesinställningar med **getMode(0)** → *var* kan du använda **setMode(***var***)** för att återställa dessa inställningar tills funktionen eller programmet avslutas. Se **getMode()**, på sidan 84.

Obs: De aktuella lägesinställningarna förs vidare till anropade delrutiner. Om någon delrutin ändrar en lägesinställning förloras lägesändringen när kontrollen återgår till den anropande delrutinen.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
Define prog1()=Prgm                    Done
  Disp approx(π)
  setMode(1,16)
  Disp approx(π)
  EndPrgm

prog1()
                                       3.14159
                                       3.14

                                       Done
```

Lägets namn	Läges-heltal	Heltal för inställningar
Display Digits (Visa siffror)	1	**1**=Float, **2**=Float1, **3**=Float2, **4**=Float3, **5**=Float4, **6**=Float5, **7**=Float6, **8**=Float7, **9**=Float8, **10**=Float9, **11**=Float10, **12**=Float11, **13**=Float12, **14**=Fix0, **15**=Fix1, **16**=Fix2, **17**=Fix3, **18**=Fix4, **19**=Fix5, **20**=Fix6, **21**=Fix7, **22**=Fix8, **23**=Fix9, **24**=Fix10, **25**=Fix11, **26**=Fix12
Angle (Vinkel)	2	**1**=Radian, **2**=Degree, **3**=Gradian
Exponential Format (Exponentiellt format)	3	**1**=Normal, **2**=Scientific, **3**=Engineering
Real or Complex (Reellt eller Komplext)	4	**1**=Real, **2**=Rectangular, **3**=Polar
Auto or Approx. (Auto eller Ungefärlig)	5	**1**=Auto, **2**=Approximate, **3**=Exact
Vector Format (Vektorformat)	6	**1**=Rectangular, **2**=Cylindrical, **3**=Spherical
Base (Bas)	7	**1**=Decimal, **2**=Hex, **3**=Binary
Unit System (Enhetssystem)	8	**1**=SI, **2**=Eng/US

shift() Katalog >

shift(*Integer1*[**,***#ofShifts*]**)**⇒*heltal*

Skiftar bitarna i ett binärt heltal. Du kan skriva in *Integer1* i valfri talbas. Det omvandlas automatiskt till 64-bitars binär form. Om storleken på *Integer1* är alltför stor för denna form för en symmetrisk modulooperation talet inom området. För mer information, se ▶**Base2**, på sidan 21.

Om *#ofShifts* är positiv sker skiftningen åt vänster. Om *#ofShifts* är negativ sker skiftningen åt höger. Förinställningen är −1 (skifta en bit åt höger).

I binärt basläge:

shift(0b1111010110000110101)	0b111101011000011010
shift(256,1)	0b1000000000

I hexadecimalt basläge:

shift(0h78E)	0h3C7
shift(0h78E,-2)	0h1E3
shift(0h78E,2)	0h1E38

Viktigt: För att skriva in ett binärt eller hexadecimalt tal, använd alltid prefixet 0b eller 0h (noll, inte bokstaven O).

Vid en skiftning åt höger "droppas" biten längst till höger och 1 infogas som denna bit. Vid skiftning åt vänster "droppas" biten längst till vänster och 0 infogas som denna bit.

Vid exempelvis skiftning åt höger:

Varje bit skiftas åt höger.

0b00000000000000111101011000011010

Infogar 0 om biten längst till vänster är 0 eller 1 om denna bit är 1.

ger:

0b00000000000000011110101100001101O

Resultatet visas enligt det inställda basläget. Inledande nollor visas inte.

shift(*List1* [**,***#ofShifts*]**)⇒***lista*

Ger en kopia av *List1* skiftad åt höger eller vänster av *#ofShifts*-elementen. Ändrar inte *List1*.

Om *#ofShifts* är positiv sker skiftningen åt vänster. Om *#ofShifts* är negativ sker skiftningen åt höger. Förinställningen är −1 (skifta ett element åt höger).

Element som infogas i början eller slutet av *list* av skiftningen tilldelas symbolen "undef".

I decimalt basläge:

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* [**,***#ofShifts*]**)⇒***sträng*

Ger en kopia av *String1* skiftad åt höger eller vänster av *#ofShifts*-tecknen. Ändrar inte *String1*.

Om *#ofShifts* är positiv sker skiftningen åt vänster. Om *#ofShifts* är negativ sker skiftningen åt höger. Förinställningen är −1 (skifta ett tecken åt höger).

Tecken som infogas i början eller slutet av *string* av skiftningen får ett mellanslag.

shift("abcd")	" abc"
shift("abcd",-2)	" ab"
shift("abcd",1)	"bcd "

sign() Katalog >

sign(*Expr1***)⇒***uttryck*

sign(*List1***)⇒***lista*

sign(*Matrix1***)⇒***matris*

Ger, för ett reellt eller komplext *Expr1*, *Expr1*/**abs(***Expr1***)** när *Expr1*≠ 0.

Ger 1 om *Expr1* är positivt.

Ger ⁻1 om *Expr1* är negativt.

sign(0) ger ±1 om det komplexa formatläget är Real, annars ger det sig självt.

sign(0) representerar enhetscirkeln i det komplexa området.

Ger, för en lista eller matris, tecknen för alla element.

$\text{sign}(-3.2) \quad -1.$

$\text{sign}(\{2,3,4,-5\}) \quad \{1,1,1,-1\}$

$\text{sign}(1+|x|) \quad 1$

Om det komplexa formatläget är Real:

$\text{sign}([-3 \quad 0 \quad 3]) \quad [-1 \quad \pm 1 \quad 1]$

simult() Katalog >

simult(*coeffMatrix***,** *constVector***[,** *Tol***])⇒***matris*

Ger en kolumnvektor som innehåller lösningarna för ett system av linjära ekvationer.

Obs: Se även **linSolve()**, på sidan 103.

coeffMatrix måste vara en kvadratmatis som innehåller koefficienterna för ekvationerna.

constVector måste ha samma antal rader (samma dimension) som *coeffMatrix* och innehålla konstanterna.

Alternativt behandlas varje matriselement som noll om dess absolutvärde är mindre än *Tol*. Denna tolerans används endast om matrisen har inmatning i flyttalsform och inte innehåller några symboliska variabler som inte har tilldelats ett värde. Annars ignoreras *Tol*.

- Om du ställer in läget **Auto eller Ungefärlig** på Approximate utförs

Lös för x och y:

x + 2y = 1

3x + 4y = ⁻1

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

Lösningen är x=⁻3 och y=2.

Lös:

ax + by = 1

cx + dy = 2

$\begin{bmatrix}a & b\\ c & d\end{bmatrix} \rightarrow matx1 \quad \begin{bmatrix}a & b\\ c & d\end{bmatrix}$

$\text{simult}\left(matx1, \begin{bmatrix}1\\ 2\end{bmatrix}\right) \quad \begin{bmatrix}\frac{-(2\cdot b-d)}{a\cdot d-b\cdot c}\\ \frac{2\cdot a-c}{a\cdot d-b\cdot c}\end{bmatrix}$

beräkningarna med flyttalsaritmetik.

- Om *Tol* utelämnas eller inte används beräknas standardtoleransen som: 5E⁻14 ·max(dim(*coeffMatrix*)) ·rowNorm(*coeffMatrix*)

simult(*coeffMatrix***,** *constMatrix***[,** *Tol***])⇒***matris*

Löser multipla system av linjära ekvationer där varje system har samma koefficienter för variablerna, men olika konstanter.

Varje kolumn i *constMatrix* måste innehålla konstanterna för ett ekvationssystem. Varje kolumn i den resulterande matrisen innehåller lösningen på motsvarande system.

Lös:

$x + 2y = 1$

$3x + 4y = -1$

$x + 2y = 2$

$3x + 4y = -3$

$$\text{simult}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}, \begin{bmatrix}1 & 2\\ -1 & -3\end{bmatrix}\right) \qquad \begin{bmatrix}-3 & -7\\ 2 & \frac{9}{2}\end{bmatrix}$$

För det första systemet är lösningen x=⁻3 och y=2. För det andra systemet är lösningen x=⁻7 och y=9/2.

▸sin **Katalog >**

Expr **▸sin**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@>sin**.

Representerar *Expr* i termer av sinus. Detta är en omvandlingsoperator för visning. Den kan endast användas i slutet av inmatningsraden.

▸sin reducerar alla potenser av cos(...) modulo 1−sin(...)^2 så att eventuella återstående potenser av sin(...) har exponenter i området (0, 2). Resultatet kommer sålunda att vara fritt från cos(...) om, och endast om, cos(...) endast inträffar i det givna uttrycket vid jämna potenser.

$$\cos(x)^2 \blacktriangleright \text{sin} \qquad 1-\sin(x)^2$$

▸sin

Katalog >

Obs: Denna omvandlingsoperator stöds inte i vinkellägena Grader och Nygrader. Innan du använder den, kontrollera att vinkelläget är inställt på Radianer och att *Expr* inte innehåller explicita referenser till vinklar i grader eller nygrader.

sin()

tangent

sin(*Expr1***)⇒***uttryck*

sin(*List1***)⇒***lista*

sin(*Expr1***)** Ger sinus för argumentet som ett uttryck.

sin(*List1***)** ger en lista på sinus för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i antingen grader, nygrader eller radianer beroende på det aktuella vinkelläget. Du kan använda °,ᴳ eller ʳ för att tillfälligt överstyra vinkelläget.

I vinkelläget Grader:

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

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

$\sin(\{0,60,90\})$ $\quad \left\{0,\frac{\sqrt{3}}{2},1\right\}$

I vinkelläget Nygrader:

$\sin(50)$ $\quad \frac{\sqrt{2}}{2}$

I vinkelläget Radianer:

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

$\sin(45°)$ $\quad \frac{\sqrt{2}}{2}$

sin(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med sinus för *squareMatrix1*. Detta är inte detsamma som att beräkna sinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

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

$$\begin{bmatrix}0.9424 & -0.04542 & -0.031999\\ -0.045492 & 0.949254 & -0.020274\\ -0.048739 & -0.00523 & 0.961051\end{bmatrix}$$

$\sin^{-1}()$ — tangent

**$\sin^{-1}$(*Expr1*)⇒*uttryck*

**$\sin^{-1}$(*List1*)⇒*lista*

$\sin^{-1}$(*Expr1*) ger den vinkel vars sinus är *Expr1* som ett uttryck.

$\sin^{-1}$(*List1*) ger en lista på invers sinus för varje element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arcsin(...)`**.

$\sin^{-1}$(*squareMatrix1*)⇒*kvadratMatris*

Ger matrisen med invers sinus för *squareMatrix1*. Detta är inte detsamma som att beräkna invers sinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Grader:

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

I vinkelläget Nygrader:

$\sin^{-1}(1)$ 100

I vinkelläget Radianer:

$\sin^{-1}(\{0,0.2,0.5\})$ $\{0,0.201358,0.523599\}$

I vinkelläget Radianer och i Rektangulärt komplext format:

$$\sin^{-1}\begin{bmatrix}1 & 5\\ 4 & 2\end{bmatrix}$$
$$\begin{bmatrix}-0.174533-0.12198\cdot i & 1.74533-2.35591\cdot i\\ 1.39626-1.88473\cdot i & 0.174533-0.593162\cdot i\end{bmatrix}$$

sinh() — Katalog >

sinh(*Expr1*)⇒*uttryck*

sinh(*List1*)⇒*lista*

sinh (*Expr1*) ger argumentets hyperboliska sinus som ett uttryck.

sinh (*List1*) ger en lista på hyperboliska sinus för varje element i *List1*.

sinh(*squareMatrix1*)⇒*kvadratMatris*

Ger matrisen med hyperbolisk sinus för *squareMatrix1*. Detta är inte detsamma som att beräkna hyperbolisk sinus för varje element. Se **cos()** för information om beräkningsmetoden.

$\sinh(1.2)$ 1.50946

$\sinh(\{0,1.2,3.\})$ $\{0,1.50946,10.0179\}$

I vinkelläget Radianer:

$$\sinh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}$$
$$\begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

sinh⁻¹(*Expr1***)⇒***uttryck*

sinh⁻¹(*List1***)⇒***lista*

sinh⁻¹(*Expr1***)** ger argumentets inversa hyperboliska sinus som ett uttryck.

sinh⁻¹(*List1***)** ger en lista på invers hyperbolisk sinus för varje element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`arcsinh(…)`**.

$\sinh^{-1}(0)$ → 0

$\sinh^{-1}(\{0,2.1,3\})$ → $\{0,1.48748,\sinh^{-1}(3)\}$

sinh⁻¹(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med invers hyperbolisk sinus för *squareMatrix1*. Detta är inte detsamma som att beräkna invers hyperbolisk sinus för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet innehåller alltid tal med flytande komma.

I vinkelläget Radianer:

$$\sinh^{-1}\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right) \rightarrow \begin{bmatrix}0.041751 & 2.15557 & 1.1582\\ 1.46382 & 0.926568 & 0.112557\\ 2.75079 & -1.5283 & 0.57268\end{bmatrix}$$

SinReg *X*, *Y* [, [*Iterations*],[*Period*] [, *Category*, *Include*]]

Utför en trigonometrisk regressionsanalys på listorna *X* och *Y*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Iterations är ett värde som specificerar det maximala antalet gånger (1 till och med 16) en lösning kommer att provas. Om denna utelämnas används 8. Normalt ger större värden bättre noggrannhet, men längre exekveringstider, och vice versa.

Period specificerar en uppskattad period. Om denna utelämnas bör skillnaden mellan värdena i X vara lika och i ordningsföljd. Om du specificerar *Period* kan skillnaderna mellan x-värden vara olika.

Category är en lista på kategorikoder för motsvarande X- och Y-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Resultatet av **SinReg** är alltid i radianer oavsett det inställda vinkelläget.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	**Beskrivning**
stat.RegEqn	Regressionsekvation: a ·sin(bx+c)+d
stat.a, stat.b, stat.c, stat.d	Regressionskoefficienter
stat.Resid	Residualer från regressionen
stat.XReg	Lista på datapunkter i den modifierade *X List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.YReg	Lista på datapunkter i den modifierade *Y List* som används i regressionen baserat på begränsningar i *Freq*, *Category List* och *Include Categories*
stat.FreqReg	Lista på frekvenser som motsvarar *stat.XReg* och *stat.YReg*

solve() Katalog >

solve(*Equation***,** *Var***)⇒***Booleskt uttryck*

solve(*Equation***,** *Var*=*Guess***)⇒***Booleskt uttryck*

$$\text{solve}\left(a\cdot x^2+b\cdot x+c=0,x\right)$$
$$x=\frac{\sqrt{b^2-4\cdot a\cdot c}-b}{2\cdot a} \text{ or } x=\frac{-\left(\sqrt{b^2-4\cdot a\cdot c}+b\right)}{2\cdot a}$$

solve(*Inequality*, *Var*)⇒*Booleskt uttryck*

Ger möjliga reella lösningar på en ekvation eller olikhet för *Var*. Målet är att ge "kandidater" för alla lösningar. Det kan dock finnas ekvationer eller lösningar för vilka antalet lösningar är oändligt.

Möjliga lösningar kanske inte är reella ändliga lösningar för vissa kombinationer av värden för odefinierade variabler.

$Ans|a=1 \text{ and } b=1 \text{ and } c=1$

$$x=\frac{-1}{2}+\frac{\sqrt{3}}{2}\cdot i \text{ or } x=\frac{-1}{2}-\frac{\sqrt{3}}{2}\cdot i$$

Med inställningen Auto i läge **Auto eller Ungefärlig** är målet att producera exakta lösningar när de är koncisa, och kompletterade med iterativa sökningar med ungefärlig aritmetik när exakta lösningar är opraktiska.

$\text{solve}\left((x-a)\cdot e^{x}=-x\cdot(x-a),x\right)$

$x=a \text{ or } x=-0.567143$

På grund av den förinställda annulleringen av den största gemensamma delaren från täljaren och nämnaren i kvoter kan lösningar vara lösningar endast som gränsvärden från en sida eller båda sidorna.

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

För olikheter av typ ≥, ≤, < eller > är explicita lösningar osannolika såvida inte olikheten är linjär och endast innehåller *Var*.

$\text{solve}\left(5\cdot x-2\geq 2\cdot x,x\right)$ $\quad x\geq\frac{2}{3}$

I läget Exact erhålls delar som inte kan lösas som en implicit ekvation eller olikhet.

$\text{exact}\left(\text{solve}\left((x-a)\cdot e^{x}=-x\cdot(x-a),x\right)\right)$

$e^{x}+x=0 \text{ or } x=a$

Använd ("|")-operatorn begränsning för att begränsa lösningsintervallet och/eller andra variabler som ingår i ekvationen eller olikheten När du finner en lösning i ett intervall kan du använda olikhetsoperatorerna för att utesluta det intervallet från efterföljande sökningar.

I vinkelläget Radianer:

$\text{solve}\left(\tan(x)=\frac{1}{x},x\right)|x>0 \text{ and } x<1$

$x=0.860334$

ger resultatet falskt när inga reella lösningar hittas. ger resultatet sant om **solve()** kan fastställa att varje ändligt reellt värde på *Var* satisfierar ekvationen eller olikheten.

$\text{solve}(x=x+1,x)$ false

$\text{solve}(x=x,x)$ true

Eftersom **solve()** alltid ger ett booleskt resultat kan du använda "and", "or" och "inte" för att kombinera resultat från **solve()** med varandra eller med andra booleska uttryck.

$2 \cdot x-1 \le 1$ and solve$(x^2 \ne 9, x)$ $\quad x \ne -3$ and $x \le 1$

Lösningar kan innehålla en ny, unik odefinierad konstant med formen *n*j där j är ett heltal i intervallet 1–255. Sådana variabler betecknar ett godtyckligt heltal.

I vinkelläget Radianer:

solve$(\sin(x)=0, x)$ $\quad x = \mathbf{n}1 \cdot \pi$

I läge Real betecknar bråktalspotenser med udda nämnare endast den rella delen. Annars betecknar flera delade uttryck såsom rationella potenser, logaritmer och inversa trigonometriska funktioner endast huvudintervallet. Följaktligen ger **solve()** endast lösningar som motsvarar den reella delen eller huvudintervallet.

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

solve$(\sqrt{x}=-2, x)$ $\quad$ false

solve$(-\sqrt{x}=-2, x)$ $\quad x=4$

Obs: Se även **cSolve()**, **cZeros()**, **nSolve()** och **zeros()**.

solve(*Eqn1* **and** *Eqn2* [**and**...]**,** *VarOrGuess1***,** *VarOrGuess2* [**,** ...]**)⇒***Booleskt uttryck*

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 $y=-1$

solve(*SystemOfEqns***,** *VarOrGuess1***,** *VarOrGuess2* [**,** ...]**)⇒***Booleskt uttryck*

solve({*Eqn1***,** *Eqn2* [,...]} {*VarOrGuess1***,** *VarOrGuess2* [**,** ...]}**) ⇒***Booleskt uttryck*

Ger möjliga reella lösningar på ekvationssystem där varje *VarOrGuess* specificerar en variabel som du vill lösa ut.

Du kan separera ekvationerna med operatorn **and** (och) eller också kan du mata in ett *SystemOfEqns* (Ekvationssystem) med en mall från Katalogen. Antalet *VarOrGuess*-argument måste vara lika med antalet ekvationer. Du kan som alternativ specificera en initial gissning för en variabel. Varje *VarOrGuess* måste ha formen:

variable

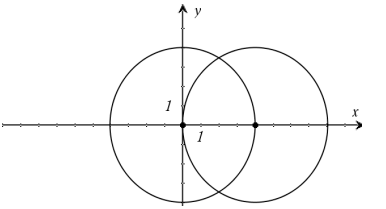
– eller –

variable = reellt eller icke-reellt tal

Som exempel är x giltigt och likaså x=3.

Om alla ekvationer är polynom och om du INTE specificerar några initiala gissningar använder **solve()** eliminationsmetoden Gröbner/Buchberger för att försöka bestämma alla reella lösningar.

Lås oss som exempel anta att du har en cirkel med radien r i origo och en annan cirkel med radien r där centrum är där den första cirkeln korsar den positiva x-axeln. Använd **solve()** för att finna skärningspunkterna.



Såsom illustreras med r i exemplet till höger kan samtidiga polynomekvationer ha extra variabler som saknar värden, men representerar givna numeriska värden som kan ersättas senare.

$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y\}\right)$$
$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ or } x=\frac{r}{2} \text{ and } y=\frac{-\sqrt{3}\cdot r}{2}$$

Du kan även (eller i stället) inkludera lösningsvariabler som inte visas i ekvationerna. Du kan exempelvis inkludera z som en lösningsvariabel för att utöka föregående exempel till två parallella överkorsande cylindrar med radien r.

$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y,z\}\right)$$
$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ and } z=c1 \text{ or } x=\frac{r}{2} \text{ and } y\blacktriangleright$$

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

Cylinderlösningarna illustrerar hur familjer av lösningar kan innehålla godtyckliga konstanter med formen *c*k där k är ett heltalssuffix från 1 till och med 255.

För polynomsystem kan beräkningstiden och användningen av minne i hög grad bero på i vilken ordning du listar lösningsvariabler. Om ditt första val utarmar minnet, eller tar på ditt tålamod, kan du försöka med att arrangera om variablerna i ekvationerna och/eller i listan *VarOrGuess*.

solve()

Katalog >

Om du inte inkluderar några gissningar och om någon ekvation är ett icke-polynom i någon variabel, men alla ekvationer är linjära i lösningsvariablerna, använder **solve ()** Gauss eliminationsmetod för att försöka bestämma alla reella lösningar.

Om ett system är varken polynomt i alla dess variabler eller linjärt i dess lösningsvariabler bestämmer **solve()** högst en lösning med en ungefärlig iterativ metod. För att göra detta måste antalet lösningsvariabler vara lika med antalet ekvationer och alla övriga variabler i ekvationerna måste förenklas till tal.

Varje lösningsvariabel startar vid dess gissade värde om det finns något, annars startar den vid 0.0.

Använd gissningar för att söka ytterligare lösningar en i taget. För konvergens kan en gissning behöva vara ganska nära en lösning.

$\text{solve}\left(x+e^{z}\cdot y=1 \text{ and } x-y=\sin(z),\{x,y\}\right)$

$x=\frac{e^{z}\cdot\sin(z)+1}{e^{z}+1} \text{ and } y=\frac{-(\sin(z)-1)}{e^{z}+1}$

$\text{solve}\left(e^{z}\cdot y=1 \text{ and } -y=\sin(z),\{y,z\}\right)$

$y=2.812E{-}10 \text{ and } z=21.9911 \text{ or } y=0.001871$ ▸

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

$\text{solve}\left(e^{z}\cdot y=1 \text{ and } -y=\sin(z),\{y,z=2\cdot\pi\}\right)$

$y=0.001871 \text{ and } z=6.28131$

SortA

Katalog >

SortA *List1*[**,** *List2*] [**,** *List3*] ...

SortA *Vector1*[**,** *Vector2*] [**,** *Vector3*] ...

Sorterar elementen i det första argumentet i stigande ordning.

Om du inkluderar ytterligare argument sorteras elementen i varje argument så att deras nya positioner matchar de nya positionerna för elementen i det första argumentet.

Alla argument måste vara namn på listor eller vektorer. Alla argument måste ha samma dimensioner.

Tomma element inom det första argumentet flyttas till botten. För mer information om tomma element, se på sidan 238.

$\{2,1,4,3\}\rightarrow list1$	$\{2,1,4,3\}$
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

Katalog >

SortD *List1*[**,** *List2*] [**,** *List3*] ...

SortD *Vector1*[**,***Vector2*] [**,***Vector3*] ...

Identisk med **SortA** med undantag för att **SortD** sorterar elementen i fallande ordning.

Tomma element inom det första argumentet flyttas till botten. För mer information om tomma element, se på sidan 238.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD *list1*,*list2*	*Done*
list1	$\{4,3,2,1\}$
list2	$\{3,4,1,2\}$

▸Sphere

Katalog >

Vector **▸Sphere**

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **`@>Sphere`**.

Visar rad- eller kolumnvektorn i sfärisk form [ρ ∠θ ∠ϕ].

Vector måste ha dimensionen 3 och kan vara antingen en rad- eller en kolumnvektor.

Obs: ▸Sphere är en visa format-instruktion, inte en konverteringsfunktion. Du kan endast använda den i slutet av en inmatningsrad.

Obs: För att få ett närmevärde,

Handenhet: Tryck på ctrl enter.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘+**Enter**.
iPad®: Håll ned **enter** och välj ≈.

$$\begin{bmatrix}1 & 2 & 3\end{bmatrix} \blacktriangleright \text{Sphere}$$
$$\begin{bmatrix}3.74166 & \angle 1.10715 & \angle 0.640522\end{bmatrix}$$

Obs: För att få ett närmevärde,

Handenhet: Tryck på ctrl enter.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘+**Enter**.
iPad®: Håll ned **enter** och välj ≈.

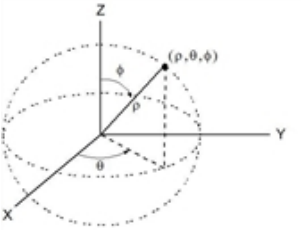
$$\left(\begin{bmatrix}2 & \angle\frac{\pi}{4} & 3\end{bmatrix}\right) \blacktriangleright \text{Sphere}$$
$$\begin{bmatrix}3.60555 & \angle 0.785398 & \angle 0.588003\end{bmatrix}$$

Tryck på enter

$$\left(\begin{bmatrix}2 & \angle\frac{\pi}{4} & 3\end{bmatrix}\right) \blacktriangleright \text{Sphere}$$
$$\begin{bmatrix}\sqrt{13} & \angle\frac{\pi}{4} & \angle\sin^{-1}\left(\frac{2\cdot\sqrt{13}}{13}\right)\end{bmatrix}$$

▸Sphere

Katalog >



sqrt()

Katalog >

sqrt(*Expr1***)⇒***uttryck*

sqrt(*List1***)⇒***lista*

Ger kvadratroten ur argumentet.

Ger, för en lista, kvadraträtterna ur alla element i *List1*.

Obs: Se även **Square root template**, på sidan 5.

$\sqrt{4}$ → 2

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

stat.results

Katalog >

stat.results

Visar resultaten av en statistisk beräkning.

Resultaten visas som en uppsättning av namn-värdepar. De specifika namnen som visas beror på den/det senast utvärderade statistiska funktionen/kommandot.

Du kan kopiera ett namn eller ett värde och klistra in det på andra platser.

Obs: Undvik att definiera variabler med samma namn som de variabler vilka används för statistisk analys. I vissa fall kan ett feltillstånd uppstå. Variabelnamn som används för statistisk analys listas i nedanstående tabell.

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}"

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}$Diff	stat.Σy²	stat.$\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			

Obs: Varje gång applikationen Listor och kalkylblad beräknar statistiska resultat kopierar den "stat.*"-gruppvariabler till en "stat#.*"-grupp där # är ett tal som ökas automatiskt. Detta låter dig bibehålla tidigare resultat medan du utför flera beräkningar.

stat.values

Katalog >

stat.values

Se exemplet **stat.results**.

stat.values

Katalog >

Visar en matris över värdena beräknade för den/det senast utvärderade statistiska funktionen/kommandot.

Till skillnad från **stat.results** utelämnar **stat.values** namnen som är associerade med värdena.

Du kan kopiera ett värde och klistra in det på andra platser.

stDevPop()

Katalog >

stDevPop(*List***[, ***freqList***])⇒***uttryck***

Ger populationens standardavvikelse för elementen i *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

Obs: *List* måste innehålla minst två element. Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

I vinkelläget Radianer och i Auto-läge:

$$\text{stDevPop}(\{a,b,c\}) \qquad \frac{\sqrt{2\cdot(a^2-a\cdot(b+c)+b^2-b\cdot c+c^2)}}{3}$$

$$\text{stDevPop}(\{1,2,5,-6,3,-2\}) \qquad \frac{\sqrt{465}}{6}$$

$$\text{stDevPop}(\{1.3,2.5,-6.4\},\{3,2,5\}) \qquad 4.11107$$

stDevPop(*Matrix1***[, ***freqMatrix***])⇒***matris***

Ger en radvektor med populationens standardavvikelser för kolumnerna i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

Obs: *Matrix1* måste innehålla minst två rader. Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

$$\text{stDevPop}\begin{bmatrix}1 & 2 & 5\\ -3 & 0 & 1\\ 5 & 7 & 3\end{bmatrix} \qquad \begin{bmatrix}\frac{4\cdot\sqrt{6}}{3} & \frac{\sqrt{78}}{3} & \frac{2\cdot\sqrt{6}}{3}\end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix}-1.2 & 5.3\\ 2.5 & 7.3\\ 6 & -4\end{bmatrix},\begin{bmatrix}4 & 2\\ 3 & 3\\ 1 & 7\end{bmatrix}\right) \qquad \begin{bmatrix}2.52608 & 5.21506\end{bmatrix}$$

stDevSamp() Katalog >

stDevSamp(*List*[**,** *freqList*]**)**⇒*uttryck*

Ger urvalets standardavvikelse för elementen i *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

Obs: *List* måste innehålla minst två element. Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

$$\text{stDevSamp}(\{a,b,c\}) \quad \frac{\sqrt{3\cdot(a^2-a\cdot(b+c)+b^2-b\cdot c+c^2)}}{3}$$

$$\text{stDevSamp}(\{1,2,5,{}^{-}6,3,{}^{-}2\}) \quad \frac{\sqrt{62}}{2}$$

$$\text{stDevSamp}(\{1.3,2.5,{}^{-}6.4\},\{3,2,5\}) \quad 4.33345$$

stDevSamp(*Matrix1*[**,** *freqMatrix*]**)**⇒*matris*

Ger en radvektor med urvalets standardavvikelser för kolumnerna i *Matrix1*.

Varje *freqMatrix*-element räknar antalet förekomster av motsvarande element i *Matrix1*.

Obs: *Matrix1* måste innehålla minst två rader. Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

$$\text{stDevSamp}\left(\begin{bmatrix}1 & 2 & 5\\ {}^{-}3 & 0 & 1\\ 5 & 7 & 3\end{bmatrix}\right) \quad \begin{bmatrix}4 & \sqrt{13} & 2\end{bmatrix}$$

$$\text{stDevSamp}\left(\begin{bmatrix}{}^{-}1.2 & 5.3\\ 2.5 & 7.3\\ 6 & {}^{-}4\end{bmatrix},\begin{bmatrix}4 & 2\\ 3 & 3\\ 1 & 7\end{bmatrix}\right) \quad \begin{bmatrix}2.7005 & 5.44695\end{bmatrix}$$

Stop Katalog >

Stop

Programmeringskommando: Avslutar programmet.

Stop är ej tillåtet i funktioner.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
i:=0                                  0
Define prog1()=Prgm                   Done
                For i,1,10,1
                If i=5
                Stop
                EndFor
                EndPrgm
prog1()                               Done
i                                     5
```

Store Se → (store), på sidan 236.

string() Katalog >

string(*Expr***)⇒***sträng*

Förenklar *Expr* och ger resultatet som en teckensträng.

string(1.2345)	"1.2345"
string(1+2)	"3"
string(cos(x)+√(3))	"cos(x)+√(3)"

subMat() Katalog >

subMat(*Matrix1***[, ***startRow***] [, ***startCol***] [, ***endRow***] [, ***endCol***]) ⇒***matris***

Ger specificerad undermatris av *Matrix1*.

Förvalsinställningar: *startRow*=1, *startCol*=1, *endRow*=sista rad, *endCol*=sista kolumn.

$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\rightarrow m1$	$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$
subMat(*m1*,2,1,3,2)	$\begin{bmatrix}4&5\\7&8\end{bmatrix}$
subMat(*m1*,2,2)	$\begin{bmatrix}5&6\\8&9\end{bmatrix}$

Sum (Sigma) Se Σ(), på sidan 226.

sum() Katalog >

sum(*List***[, ***Start***[, ***End***]])⇒***uttryck***

Ger summan av elementen i *List*.

Start och *end* är valfria. De specificerar ett område med element.

Varje tomt argument ger ett tomt resultat. Tomma element i *List* ignoreras. För mer information om tomma element, se på sidan 238.

sum({1,2,3,4,5})	15
sum({*a*,2·*a*,3·*a*})	6·*a*
sum(seq(*n*,*n*,1,10))	55
sum({1,3,5,7,9},3)	21

sum(*Matrix1***[, ***Start***[, ***End***]])⇒***matris***

Ger en radvektor som innehåller summorna av elementen i kolumnerna i *Matrix1*.

Start och *end* är valfria. De specificerar ett område med rader.

Varje tomt argument ger ett tomt resultat. Tomma element i *Matrix1* ignoreras. För mer information om tomma element, se på sidan 238.

$\text{sum}\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$	$\begin{bmatrix}5&7&9\end{bmatrix}$
$\text{sum}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$	$\begin{bmatrix}12&15&18\end{bmatrix}$
$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2,3\right)$	$\begin{bmatrix}11&13&15\end{bmatrix}$

sumIf() Katalog >

sumIf(*List*,*Criteria*[, *SumList*]**)**⇒*värde*

Ger den totala kumulerade summan av alla element i *List* som uppfyller specificerade *Criteria*. Du kan också specificera en alternativ lista, *sumList*, för att ge elementen som skall kumuleras.

List kan vara ett uttryck, en lista eller en matris. *SumList*, om specificerad, måste ha samma dimension(er) som *List*.

Criteria kan vara:

- Ett värde, ett uttryck eller en sträng. Som exempel ackumulerar **34** endast de element i *List* som förenklas till värdet 34.
- Ett booleskt uttryck som innehåller symbolen **?** fungerar som platshållare för varje element. Som exempel ackumulerar **?<10** endast de element i *List* som är lägre än 10.

När ett *List*-element uppfyller *Criteria* adderas elementet till den ackumulerade summan. Om du inkluderar *sumList* adderas i stället motsvarande element från *sumList* till summan.

I applikationen Listor och kalkylblad kan du använda ett område av celler i stället för *List* och *sumList*.

Tomma element ignoreras. För mer information om tomma element, se på sidan 238.

Obs: Se även **countIf()**, på sidan 39.

$\text{sumIf}(\{1,2,e,3,\pi,4,5,6\},2.5<?<4.5)$

$e+\pi+7$

$\text{sumIf}(\{1,2,3,4\},2<?<5,\{10,20,30,40\})$

70

sumSeq() Se Σ(), på sidan 226.

system() Katalog >

system(*Eqn1* [**,** *Eqn2* [**,** *Eqn3* [**,** ...]]]**)**

system(*Expr1* [**,** *Expr2* [**,** *Expr3* [**,** ...]]]**)**

$\text{solve}\left(\begin{cases}x+y=0\\x-y=8\end{cases},x,y\right)$ $x=4$ and $y=-4$

system()

Katalog >

Ger ett ekvationssystem formaterat som en lista. Du kan också skapa ett system med en mall.

Obs: Se även **System of equations**, på sidan 7.

T

T(transponera)

Katalog >

*Matrix1*T⇒*matris*

Ger den komplexkonjugerade transponeringen av *Matrix1*.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva `@t`.

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

$$\begin{bmatrix}a & b\\ c & d\end{bmatrix}^{T} \qquad \begin{bmatrix}a & c\\ b & d\end{bmatrix}$$

$$\begin{bmatrix}1+i & 2+i\\ 3+i & 4+i\end{bmatrix}^{T} \qquad \begin{bmatrix}1-i & 3-i\\ 2-i & 4-i\end{bmatrix}$$

tan()

tangent

tan(*Expr1***)**⇒*uttryck*

tan(*List1***)**⇒*lista*

tan(*Expr1***)** ger tangensen för argumentet som ett uttryck.

tan(*List1***)** ger en lista på tangensen för alla element i *List1*.

Obs: Argumentet tolkas som en vinkel i antingen grader, nygrader eller radianer beroende på det aktuella vinkelläget. Du kan använda °, G eller r för att tillfälligt överstyra vinkelläget.

I vinkelläget Grader:

$$\tan\left(\frac{\pi}{4}r\right) \qquad 1$$

$$\tan(45) \qquad 1$$

$$\tan(\{0,60,90\}) \qquad \{0,\sqrt{3},\text{undef}\}$$

I vinkelläget Nygrader:

$$\tan\left(\frac{\pi}{4}r\right) \qquad 1$$

$$\tan(50) \qquad 1$$

$$\tan(\{0,50,100\}) \qquad \{0,1,\text{undef}\}$$

I vinkelläget Radianer:

tan()

trig tangent

$\tan\left(\frac{\pi}{4}\right)$ → 1

$\tan(45°)$ → 1

$\tan\left(\left\{\pi, \frac{\pi}{3}, -\pi, \frac{\pi}{4}\right\}\right)$ → $\{0, \sqrt{3}, 0, 1\}$

tan(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med tangensen för *squareMatrix1*. Detta är inte detsamma som att beräkna tangensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

$$\tan\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$$

tan⁻¹()

trig tangent

tan⁻¹(*Expr1***)⇒***uttryck*

tan⁻¹(*List1***)⇒***lista*

tan⁻¹(*Expr1***)** ger den vinkel vars tangens är *Expr1* som ett uttryck.

tan⁻¹(*List1***)** ger en lista på den inversa tangensen för varje element i *List1*.

Obs: Resultatet erhålls som en vinkel i grader, nygrader eller radianer beroende på det aktuella vinkelläget.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **arctan(…)**.

I vinkelläget Grader:

$\tan^{-1}(1)$ → 45

I vinkelläget Nygrader:

$\tan^{-1}(1)$ → 50

I vinkelläget Radianer:

$\tan^{-1}(\{0, 0.2, 0.5\})$ → $\{0, 0.197396, 0.463648\}$

tan⁻¹(*squareMatrix1***)⇒***kvadratMatris*

Ger matrisen med invers tangens för *squareMatrix1*. Detta är inte detsamma som att beräkna invers tangens för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer:

$$\tan^{-1}\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{bmatrix}$$

tangentLine() Katalog >

tangentLine(*Expr1*,*Var*,*Point*)⇒*uttryck*

tangentLine(*Expr1*,*Var*=*Point*)⇒*uttryck*

Ger tangenten för kurvan representerad av *Expr1* vid punkten som specificeras i *Var*=*Point*.

Kontrollera att den oberoende variabeln inte är definierad. Om exempelvis f1(x):=5 och x:=3 ger **tangentLine(**f1(x),x,2**)** "falskt".

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

$\text{tangentLine}((x-3)^2-4,x=3)$ → -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() Katalog >

tanh(*Expr1***)**⇒*uttryck*

tanh(*List1***)**⇒*lista*

tanh(*Expr1***)** ger den hyperboliska tangensen för argumentet som ett uttryck.

tanh(*List1***)** ger en lista på den hyperboliska tangensen för varje element i *List1*.

tanh(*squareMatrix1***)**⇒*kvadratMatris*

Ger matrisen med hyperbolisk tangens för *squareMatrix1*. Detta är inte detsamma som att beräkna hyperbolisk tangens för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

$\tanh(1.2)$ → 0.833655

$\tanh(\{0,1\})$ → $\{0,\tanh(1)\}$

I vinkelläget Radianer:

$$\tanh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}-0.097966 & 0.933436 & 0.425972\\ 0.488147 & 0.538881 & -0.129382\\ 1.28295 & -1.03425 & 0.428817\end{bmatrix}$$

tanh⁻¹() Katalog >

tanh⁻¹(*Expr1***)**⇒*uttryck*

tanh⁻¹(*List1***)**⇒*lista*

tanh⁻¹(*Expr1***)** ger argumentets inversa hyperboliska tangens som ett uttryck.

tanh⁻¹(*List1***)** ger en lista på invers hyperbolisk tangens för varje element i *List1*.

I Rektangulärt komplext format:

$\tanh^{-1}(0)$ → 0

$\tanh^{-1}(\{1,2.1,3\})$ → $\left\{\text{undef},0.518046-1.5708\cdot i,\frac{\ln(2)}{2}-\frac{\pi}{2}\cdot i\right\}$

tanh⁻¹() Katalog >

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **arctanh(...)**.

tanh⁻¹(*squareMatrix1***)⇒***kvadratMatris***

Ger matrisen med invers hyperbolisk tangens för *squareMatrix1*. Detta är inte detsamma som att beräkna invers hyperbolisk tangens för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

I vinkelläget Radianer och i Rektangulärt komplext format:

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

$$\begin{bmatrix}-0.099353+0.164058\cdot i & 0.267834-1.4908\\ -0.087596-0.725533\cdot i & 0.479679-0.9473\\ 0.511463-2.08316\cdot i & -0.878563+1.7901\end{bmatrix}$$

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

taylor() Katalog >

taylor(*Expr1***, *Var*, *Order*[, *Point*])⇒***uttryck***

Ger det begärda Taylorpolynomet. Polynomet inkluderar icke-nolltermer av heltalsgrader från noll till och med *Order* i (*Var* minus *Point*). **taylor()** ger sig själv om det inte finns någon trunkerad potensserie av denna ordning, eller om det skulle kräva negativa exponenter eller exponenter i bråkform. Använd substitution och/eller temporär multiplikation med en potens av (*Var* minus *Point*) för att bestämma mer allmänna potensserier.

Point förinställs till noll och utgör expansionspunkten.

$$\text{taylor}\left(e^{\sqrt{x}},x,2\right) \qquad \text{taylor}\left(e^{\sqrt{x}},x,2,0\right)$$

$$\text{taylor}\left(e^{t},t,4\right)\Big|t=\sqrt{x} \qquad \frac{x^2}{24}+\frac{x^{\frac{3}{2}}}{6}+\frac{x}{2}+\sqrt{x}+1$$

$$\text{taylor}\left(\frac{1}{x\cdot(x-1)},x,3\right) \qquad \text{taylor}\left(\frac{1}{x\cdot(x-1)},x,3,0\right)$$

$$\text{expand}\left(\frac{\text{taylor}\left(\frac{x}{x\cdot(x-1)},x,4\right)}{x},x\right) \qquad -x^3-x^2-x-\frac{1}{x}-1$$

tCdf() Katalog >

tCdf(*lowBound***,***upBound***,***df***)⇒***tal*** om *lowBound* och *upBound* är tal, ***lista*** om *lowBound* och *upBound* är listor

Beräknar sannolikheten för Student-*t*-fördelning mellan *lowBound* och *upBound* för den specificerade frihetsgraden *df*.

tCdf()

Katalog >

För P(X ≤ *upBound*), sätt *lowBound* = $-\infty$.

tCollect()

Katalog >

tCollect(*Expr1***)⇒***uttryck*

Ger ett uttryck i vilket produkter och heltalspotenser av sinus och cosinus konverteras till en linjär kombination av sinus och cosinus för multipla vinklar, vinkelsummor och vinkelskillnader. Transformationen konverterar trigonometriska polynom till en linjär kombination.

Ibland uppnår **tCollect()** dina mål när den förinställda trigonometriska förenklingen inte gör det. **tCollect()** tenderar att omkasta transformationer utförda av **tExpand()**. Ibland ger två separata steg med **tExpand()** på ett resultat från **tCollect()**, eller vice versa, en förenkling av ett uttryck.

$\text{tCollect}(\cos(\alpha)^2)$ $\quad \frac{\cos(2\cdot\alpha)+1}{2}$

$\text{tCollect}(\sin(\alpha)\cdot\cos(\beta))$ $\quad \frac{\sin(\alpha-\beta)+\sin(\alpha+\beta)}{2}$

tExpand()

Katalog >

tExpand(*Expr1***)⇒***uttryck*

Ger ett uttryck i vilket sinus och cosinus för multipla heltalsvinklar, vinkelsummor och vinkelskillnader utvecklas. På grund av identiteten (sin(x))2+(cos(x))2=1 finns det många möjliga ekvivalenta resultat. Som en följd härav kan ett resultat skilja sig från ett resultat som visas i andra publikationer.

Ibland uppnår **tExpand()** dina mål när den förinställda trigonometriska förenklingen inte gör det. **tExpand()** tenderar att omkasta transformationer utförda av **tCollect()**. Ibland ger två separata steg med **tCollect()** på ett resultat från **tExpand()**, eller vice versa, en förenkling av ett uttryck.

Obs: I läget Grader stör skalning med π/180 förmågan hos **tExpand()** att känna igen former som kan utvecklas. För bästa resultat bör **tExpand()** användas i läget Radianer.

$\text{tExpand}(\sin(3\cdot\varphi))$ $\quad 4\cdot\sin(\varphi)\cdot\cos(\varphi)^2-\sin(\varphi)$

$\text{tExpand}(\cos(\alpha-\beta))$ $\quad \cos(\alpha)\cdot\cos(\beta)+\sin(\alpha)\cdot\sin(\beta)$

Text

Katalog >

Text*frågeSträng*[**,** *DispFlagga*]

Programmeringskommando: Pausar programmet och visar teckensträngen *frågeSträng* i en dialogruta.

När användaren väljer **OK** fortsätter exekveringen av programmet.

Det valfria argumentet *flagga* kan vara ett valfritt uttryck.

- Om *DispFlagga* utelämnas eller beräknas till **1** sparas textmeddelandet i Räknarens historik.
- Om *DispFlag* beräknas till **0** sparas textmeddelandet inte i historiken.

Om programmet behöver ett svar som skrivs in av användaren, se **Request**, på sidan 149 eller **RequestStr**, på sidan 151.

Obs: Du kan använda detta kommando inom ett användardefinierat program, men inte inom en funktion.

Definiera ett program som pausar för att visa vart och ett av fem slumptal i en dialogruta.

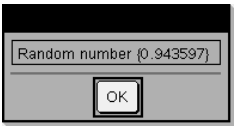
Inom mallen Prgm...EndPrgm template, fullborda varje rad genom att trycka på ↵ i stället för enter. På datorns tangentbord, håll ned **Alt** och tryck på **Enter**.

```
Define text_demo()=Prgm
  For i,1,5
    strinfo:="Random number " &
string(rand(i))
    Text strinfo
  EndFor
EndPrgm
```

Kör programmet:

```
text_demo()
```

Exempel på en dialogruta:



Then

Se If, på sidan 88.

tInterval

Katalog >

tInterval *List*[**,***Freq*[**,***CLevel*]]

(Indatalista)

tInterval $\bar{x}$**,***sx*,*n*[**,***CLevel*]

(Summary stats indata)

tInterval

Katalog >

Beräknar ett *t*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall för ett okänt populationsmedelvärde
stat.$\bar{x}$	Urvalsmedelvärde för datasekvensen från den slumpmässiga normalfördelningen
stat.ME	Felmarginal
stat.df	Frihetsgrader
stat.σx	Standardavvikelse för urvalet
stat.n	Längden på datasekvenser med urvalsmedelvärde

tInterval_2Samp

Katalog >

tInterval_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*CLevel*[,*Pooled*]]]]

(Indatalista)

tInterval_2Samp $\bar{x}$*1*,*sx1*,*n1*,$\bar{x}$*2*,*sx2*,*n2*[,*CLevel*[,*Pooled*]]

(Summary stats indata)

Beräknar ett 2-sampel *t*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Pooled=**1** slår samman varianser, *Pooled*=**0** slår inte samman varianser.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall innehållande konfidensnivån för fördelningens sannolikhet

Resultatvariabel	Beskrivning
stat.$\bar{x}1$-$\bar{x}2$	Urvalsmedelvärden för datasekvenserna från den slumpmässiga normalfördelningen
stat.ME	Felmarginal
stat.df	Frihetsgrader
stat.$\bar{x}1$, stat.$\bar{x}2$	Urvalsmedelvärden för datasekvenserna från den slumpmässiga normalfördelningen
stat.σx1, stat.σx2	Urvalets standardavvikelser för *List 1* och *List 2*
stat.n1, stat.n2	Antalet samplingar i datasekvenser
stat.sp	Den sammanslagna standardavvikelsen. Beräknas när *Pooled* = YES.

tmpCnv() Katalog >

tmpCnv(*Expr_°tempUnit***, _°***tempUnit2***)**
⇒*uttryck _°tempUnit2*

Konverterar ett temperaturvärde specificerat av *Expr* från en enhet till en annan. Giltiga temperaturenheter är:

_°C Celsius

_°F Fahrenheit

_°K Kelvin

_°R Rankine

För att skriva in symbolen °, välj den på symbollistan i Katalogen.

För att skriva in symbolen _, tryck på ctrl ␣.

Som exempel konverteras 100_°C till 212_°F.

För att konvertera ett temperaturintervall, använd Δ**tmpCnv()** i stället.

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

Obs: Du kan använda Katalogen för att välja temperaturenheter.

ΔtmpCnv() Katalog >

ΔtmpCnv(*Expr_°tempUnit***, _°***tempUnit2***)**
⇒*uttryck _°tempUnit2*

För att skriva in symbolen Δ, välj den på symbollistan i Katalogen.

ΔtmpCnv() Katalog >

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`deltaTmpCnv(...)`**.

Omvandlar ett temperaturintervall (skillnaden mellan två temperaturvärden) specificerat av *Expr* från en enhet till en annan. Giltiga temperaturenheter är:

_°C Celsius

_°F Fahrenheit

_°K Kelvin

_°R Rankine

För att mata in °, välj den på symbolpaletten eller skriv **`@d`**.

För att skriva in symbolen _, tryck på ctrl ␣.

1_°C och 1_°K har samma storlek och likaså 1_°F och 1_°R. 1_°C är dock 9/5 större än 1_°F.

Som exempel motsvarar ett 100_°C-område (från 0_°C till 100_°C) ett 180_°F-område.

För att konvertera ett visst temperaturvärde i stället för ett intervall, använd **tmpCnv()**.

$\Delta\text{tmpCnv}(100\cdot\_°\text{C},\_°\text{F})$	$180.\cdot\_°\text{F}$
$\Delta\text{tmpCnv}(180\cdot\_°\text{F},\_°\text{C})$	$100.\cdot\_°\text{C}$
$\Delta\text{tmpCnv}(100\cdot\_°\text{C},\_°\text{K})$	$100.\cdot\_°\text{K}$
$\Delta\text{tmpCnv}(100\cdot\_°\text{F},\_°\text{R})$	$100.\cdot\_°\text{R}$
$\Delta\text{tmpCnv}(1\cdot\_°\text{C},\_°\text{F})$	$1.8\cdot\_°\text{F}$

Obs: Du kan använda Katalogen för att välja temperaturenheter.

tPdf() Katalog >

tPdf(*XVal*,*df***)**⇒*tal* om *XVal* är ett tal, *lista* om *XVal* är en lista

Beräknar värde hos täthetsfunktionen (pdf) för Student-*t*-fördelningen vid ett specificerat *x*-värde med den specificerade frihetsgraden *df*.

trace() Katalog >

trace(*squareMatrix***)⇒***uttryck*

Ger spåret (summan av alla elementen på huvuddiagonalen) av *squareMatrix*.

$\text{trace}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\right)$	15
$\text{trace}\left(\begin{bmatrix}a&0\\1&a\end{bmatrix}\right)$	$2\cdot a$

Try Katalog >

Try

block1

Else

block2

EndTry

Exekverar *block1* såvida inte ett fel uppstår. Programexekveringen överförs till *block2* om ett fel uppstår i *block1*. Systemvariabeln *errCode* innehåller felkoden som låter programmet utföra återhämtning efter fel. För en lista på felkoder, se "*Felkoder och meddelanden*" (på sidan 244).

block1 och *block2* kan vara antingen ett enstaka påstående eller en serie av påståenden separerade med tecknet ":".

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

```
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
```

Exempel 2

För att se kommandona **Try**, **ClrErr** och **PassErr** i arbete, mata in programmet eigenvals() som visas till höger. Kör programmet genom att exekvera vart och ett av följande uttryck.

$$eigenvals\left(\begin{bmatrix}-3\\-41\\5\end{bmatrix},\begin{bmatrix}-1&2&-3.1\end{bmatrix}\right)$$

```
Definiera eigenvals(a,b)=Prgm

© Programmet eigenvals(A,B) visar
egenvärdena för A·B

Try

  Disp "A= ",a

  Disp "B= ",b

  Disp " "

  Disp "Eigenvalues A·B are:",eigVl(a*b)
```

Try

Katalog >

$$eigenvals\left(\begin{bmatrix}1 & 2 & 3\end{bmatrix},\begin{bmatrix}1\\2\end{bmatrix}\right)$$

Obs: Se även **ClrErr**, på sidan 29 och **PassErr**, på sidan 132.

```
Else
  If errCode=230 Then
    Disp "Error: Product of A·B must be a square matrix"
    ClrErr
  Else
    PassErr
  EndIf
EndTry
EndPrgm
```

tTest

Katalog >

tTest $\mu 0$**,***List*[**,***Freq*[**,***Hypoth*]]

(Indatalista)

tTest $\mu 0$**,**$\bar{x}$**,***sx*,*n*,[*Hypoth*]

(Summary stats indata)

Utför ett hypotestest för ett okänt populationsmedelvärde, μ, när populationens standardavvikelse, σ, är okänd. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Testa H_0: μ = μ0, mot en av följande:

För H_a: μ < μ0, ställ *Hypoth*<0

För H_a: μ ≠ μ0 (förinställning), ställ *Hypoth*=0

För H_a: μ > μ0, ställ *Hypoth*>0

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.t	$(\bar{x} - \mu 0) / (\text{stdev} / \text{sqrt}(n))$
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader
stat.$\bar{x}$	Urvalsmedelvärde för datasekvensen i *List*
stat.sx	Urvalets standardavvikelse för datasekvensen
stat.n	Urvalets storlek

tTest_2Samp

Katalog >

tTest_2Samp *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*[,*Pooled*]]]]

(Indatalista)

tTest_2Samp $\bar{x}$*1*,*sx1*,*n1*,$\bar{x}$*2*,*sx2*,*n2*[,*Hypoth*[,*Pooled*]]

(Summary stats indata)

Beräknar ett 2-sampel *t*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Testa H_0: $\mu 1 = \mu 2$, mot en av följande:

För H_a: $\mu 1 < \mu 2$, ställ *Hypoth*<0

För H_a: $\mu 1 \neq \mu 2$ (förinställning), ställ *Hypoth*=0

För H_a: $\mu 1 > \mu 2$, ställ *Hypoth*>0

Pooled=**1** slår samman varianser,

Pooled=**0** slår inte samman varianser.

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.t	Standardnormalvärde beräknat för skillnaden mellan medelvärden
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.df	Frihetsgrader hos t-statistiken

Resultatvariabel	Beskrivning
stat.$\bar{x}$1, stat.$\bar{x}$2	Medelvärden hos urvalet i datasekvenserna i *List 1* och *List 2*
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i *List 1* och *List 2*
stat.n1, stat.n2	Storlek på urvalen
stat.sp	Den sammanslagna standardavvikelsen. Beräknad när *Pooled*=1.

tvmFV()

Katalog >

tvmFV(*N*,*I*,*PV*,*Pmt*,[*PpY*],[*CpY*],[*PmtAt*]**)**⇒*värde*

Finansiell funktion som beräknar det framtida värdet på pengar.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 196. Se även **amortTbl()**, på sidan 12.

tvmFV(120,5,0,-500,12,12) 77641.1

tvmI()

Katalog >

tvmI(*N*,*PV*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)**⇒*värde*

Finansiell funktion som beräknar räntesatsen per år.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 196. Se även **amortTbl()**, på sidan 12.

tvmI(240,100000,-1000,0,12,12) 10.5241

tvmN()

Katalog >

tvmN(*I*,*PV*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)**⇒*värde*

Finansiell funktion som beräknar antalet betalningsperioder.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 196. Se även **amortTbl()**, på sidan 12.

tvmN(5,0,-500,77641,12,12) 120.

tvmPmt()

Katalog >

tvmPmt(*N*,*I*,*PV*,*FV*,[*PpY*],[*CpY*],[*PmtAt*])⇒*värde*

tvmPmt(60,4,30000,0,12,12) -552.496

Finansiell funktion som beräknar beloppet på varje betalning.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 196. Se även **amortTbl()**, på sidan 12.

tvmPV()

Katalog >

tvmPV(*N*,*I*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*])⇒*värde*

tvmPV(48,4,-500,30000,12,12) -3426.7

Finansiell funktion som beräknar nuvärdet.

Obs: Argumenten som används i TVM-funktionerna beskrivs i tabellen över TVM-argumenten, på sidan 196. Se även **amortTbl()**, på sidan 12.

TVM-argument*	**Beskrivning**	**Datatyp**
N	Antal betalningsperioder	reellt tal
I	Årlig räntesats	reellt tal
PV	Nuvärde	reellt tal
Pmt	Betalningsbelopp	reellt tal
FV	Framtida värde	reellt tal
PpY	Antal betalningar per år, förinställning = 1	heltal > 0
CpY	Ränteperioder per år, förinställning = 1	heltal > 0
PmtAt	Betalning som skall betalas i slutet (end) eller i början (beginning) av varje period, förinställning = end	heltal (0 = end, 1 = beginning)

***** Dessa argumentnamn avseende tidsjusterat pengavärde påminner om namnen på TVM-variablerna (till exempel, **tvm.pv** och **tvm.pmt**) som används av Finance Solver i applikationen *Calculator*. Finansiella funktioner lagrar dock inte sina argumentvärden eller resultat i TVM-variablerna.

TwoVar

Katalog >

TwoVar *X*, *Y*[, [*Freq*] [, *Category*, *Include*]]

Utför tvåvariabelstatistik. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Alla listor utom *Include* måste ha samma dimensioner.

X och *Y* är listor på oberoende och beroende variabler.

Freq är en frivillig lista på frekvensvärden. Varje element i *Freq* specificerar frekvensen för varje motsvarande *X*- och *Y*-datapunkt. Det förinställda värdet är 1. Alla element måste vara heltal ≥ 0.

Category är en lista på kategorikoder för motsvarande *X*- och *Y*-data.

Include är en lista på en eller flera av kategorikoderna. Endast de dataobjekt vars kategorikod är med på listan tas med i beräkningen.

Ett tomt element i någon av listorna *X*, *Freq* eller *Category* resulterar i ett tomrum för motsvarande element i dessa listor. Ett tomt element i någon av listorna *X1* till och med *X20* resulterar i ett tomrum för motsvarande element i dessa listor. För mer information om tomma element, se på sidan 238.

Resultatvariabel	**Beskrivning**
stat.$\bar{x}$	Medelvärde av x-värden
stat. x	Summa av x-värden
stat. x2	Summa av x2-värden
stat.sx	Standardavvikelse för x (sampling)
stat. x	Standardavvikelse för x (population)
stat.n	Antal datapunkter
stat.$\bar{y}$	Medelvärde av y-värden
stat. y	Summa av y-värden
stat. y^2	Summa av y2-värden

Resultatvariabel	Beskrivning
stat.sy	Standardavvikelse för y (sampling)
stat. y	Populationens standardavvikelse för y
stat. xy	Summa av x ·y-värden
stat.r	Korrelationskoefficient
stat.MinX	Minsta x-värde
stat.Q_1X	Undre kvartil för x
stat.MedianX	Median för x
stat.Q_3X	Övre kvartil för x
stat.MaxX	Största x-värde
stat.MinY	Minsta y-värde
stat.Q_1Y	Undre kvartil för y
stat.MedY	Median för y
stat.Q_3Y	Övre kvartil för y
stat.MaxY	Största y-värde
stat. (x-)^2	Kvadratsumma för avvikelser från medelvärdet på x
stat. (y-)^2	Kvadratsumma för avvikelser från medelvärdet på y

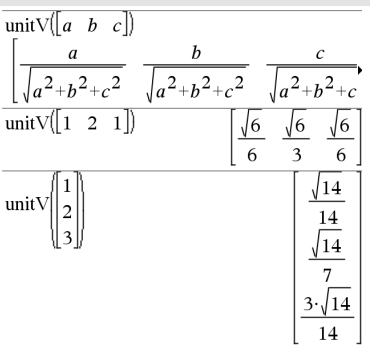
U

unitV() Katalog >

unitV(*Vector1***)⇒***vektor*

Ger en radenhets- eller kolumnenhetsvektor beroende på formen hos *Vector1*.

Vector1 måste vara antingen en enradsmatris eller en enkolumnsmatris.



unitV() Katalog >

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

unLock Katalog >

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

unLock *Var***.**

Låser upp den specificerade variabeln eller variabelgruppen. Låsta variabler kan inte modifieras eller tas bort.

Se **Lock**, på sidan 107 och **getLockInfo()**, på sidan 84.

$a:=65$	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() Katalog >

varPop(*List*[**,** *freqList*]**)**⇒*uttryck*

Ger populationsvariansen för *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

Obs: *List* måste innehålla minst två element.

Om ett element i endera listan är tomt ignoreras detta element och även motsvarande element i den andra listan ignoreras. För mer information om tomma element, se på sidan 238.

varPop$\{\{5,10,15,20,25,30\}\}$	$\frac{875}{12}$
$Ans \cdot 1.$	72.9167

varSamp() Katalog >

varSamp(*List*[**,** *freqList*]**)⇒***uttryck*

Ger sampelvariansen för *List*.

Varje *freqList*-element räknar antalet förekomster av motsvarande element i *List*.

Obs: *List* måste innehålla minst två element.

Om ett element i endera listan är tomt ignoreras detta element och även motsvarande element i den andra listan ignoreras. För mer information om tomma element, se på sidan 238.

$\text{varSamp}(\{a,b,c\})$ $\quad \frac{a^2-a\cdot(b+c)+b^2-b\cdot c+c^2}{3}$

$\text{varSamp}(\{1,2,5,-6,3,-2\})$ $\quad \frac{31}{2}$

$\text{varSamp}(\{1,3,5\},\{4,6,2\})$ $\quad \frac{68}{33}$

varSamp(*Matrix1*[**,** *freqMatrix*]**)⇒***matris*

Ger en radvektor som innehåller sampelvariansen för varje kolumn i *Matrix1*.

Varje *freqMatrix*-element räknar antalet konsekutiva förekomster av motsvarande element i *Matrix1*.

Obs: *Matrix1* måste innehålla minst två rader.

Om ett element i endera matrisen är tomt ignoreras detta element och även motsvarande element i den andra matrisen ignoreras. För mer information om tomma element, se på sidan 238.

$\text{varSamp}\left(\begin{bmatrix}1 & 2 & 5\\ -3 & 0 & 1\\ .5 & .7 & 3\end{bmatrix}\right)$ $\quad \begin{bmatrix}4.75 & 1.03 & 4\end{bmatrix}$

$\text{varSamp}\left(\begin{bmatrix}-1.1 & 2.2\\ 3.4 & 5.1\\ -2.3 & 4.3\end{bmatrix},\begin{bmatrix}6 & 3\\ 2 & 4\\ 5 & 1\end{bmatrix}\right)$ $\quad \begin{bmatrix}3.91731 & 2.08411\end{bmatrix}$

W

Wait Katalog >

Wait *tidISekunder*

Fördröjer exekvering för en period som varar *tidISekunder* sekunder.

Wait är speciellt användbart i ett program som behöver en kort fördröjning för att begärda data ska bli tillgängliga.

Argumentet *tidISekunder* måste vara ett uttryck som förenklas till ett decimalvärde i intervallet 0 till 100. Kommandot avrundar detta värde till närmaste 0,1 sekunder.

För att vänta 4 sekunder:

Wait 4

För att vänta 1/2 sekund:

Wait 0.5

För att vänta 1,3 sekunder med hjälp av variabeln *sekantal*:

sekantal:=1.3
Wait sekantal

Wait

Katalog >

För att avbryta en **Wait** som pågår,

- **Handenhet:** Håll ned [on] och tryck på [enter] upprepade gånger.
- **Windows®:** Håll ned **F12** och tryck på **Enter** upprepade gånger.
- **Macintosh®:** Håll ned **F5** och tryck på **Enter** upprepade gånger.
- **iPad®:** Appen visar en uppmaning. Du kan fortsätta att vänta eller avbryta.

Obs: Du kan använda kommandot **Wait** i ett användardefinierat program, men inte inom en funktion.

I detta exempel tänds en grön lysdiod i 0,5 sekunder och släcks sedan.

```
Send "SET GREEN 1 ON"
Wait 0.5
Send "SET GREEN 1 OFF"
```

warnCodes ()

Katalog >

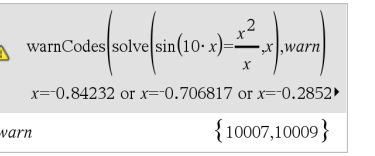
warnCodes(*Expr1*, *StatusVar*)⇒*uttryck*

Utvärderar uttrycket *Expr1*, ger resultatet och lagrar eventuellt genererade varningskoder i listvariabeln *StatusVar*. Om inga varningar genereras tilldelar denna funktion *StatusVar* en tom lista.

Expr1 kan vara ett valfritt giltigt matematiskt uttryck i TI-Nspire™ eller TI-Nspire™ CAS. Du kan inte använda ett kommando eller en tilldelning som *Expr1*.

StatusVar måste vara ett giltigt variabelnamn.

För en lista på varningskoder och tillhörande meddelanden, se på sidan 253.



För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

when()

Katalog >

when(*Condition*, *trueResult* [, *falseResult*] [, *unknownResult*]) ⇒*uttryck*

Ger *trueResult*, *falseResult* eller *unknownResult* beroende på om *Condition* är sant, falskt eller okänt. Ger indata om det finns alltför få argument för att specificera ett lämpligt resultat.

when()

Katalog >

Utelämna både *falseResult* och *unknownResult* för att skapa ett uttryck som är definierat endast i området där *Condition* är sant.

Använd ett **undef** *falseResult* för att definiera ett uttryck som plottas endast i ett intervall.

when() är användbar för att definiera rekursiva funktioner.

$\text{when}(x<0, x+3) \mid x=5$	undef

$\text{when}(n>0, n\cdot factoral(n-1), 1) \rightarrow factoral(n)$	*Done*
$factoral(3)$	6
$3!$	6

While

Katalog >

While *Condition*

Block

EndWhile

Exekverar påståendena i *Block* så länge *Condition* är sant.

Block kan vara antingen ett enstaka påstående eller en serie av påståenden separerade med tecknet ":".

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Define $sum\_of\_recip(n)=$	Func
	Local $i, tempsum$
	$1 \rightarrow i$
	$0 \rightarrow tempsum$
	While $i \le n$
	$tempsum + \frac{1}{i} \rightarrow tempsum$
	$i+1 \rightarrow i$
	EndWhile
	Return $tempsum$
	EndFunc
	Done
$sum\_of\_recip(3)$	$\frac{11}{6}$

X

xor

Katalog >

BooleskUttr1 **xor** *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 **xor** *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 **xor** *BooleskMatris2* ger *Boolesk matris*

true xor true	false
5>3 xor 3>5	true

Ger resultatet sant om *BooleanExpr1* är sant och *BooleanExpr2* är falskt, eller vice versa.

Ger resultatet falskt om båda argumenten är sanna eller falska. Ger ett förenklat booleskt uttryck om något av argumenten inte kan lösas om till sant eller falskt.

Obs: Se **eller**, på sidan 130.

Integer1 **xor** *Integer2* ⇒ *heltal*

Jämför två reella heltal bit för bit med en **xor**-operation. Internt omvandlas båda heltalen till 64-bitars binära tal. När motsvarande bitar jämförs blir resultatet 1 om en bit (men inte båda) är 1. Resultatet blir 0 om båda bitarna är 0 eller 1. Det erhållna värdet representerar bitresultaten och visas enligt det inställda basläget.

Du kan skriva in heltalen i valfri talbas. För en binär eller hexadecimal inmatning måste du använda prefixet 0b respektive 0h. Utan prefix behandlas heltalen som decimala (bas 10).

Om du skriver in ett decimalt heltal som är alltför stort för att anges i 64-bitars binär form används en symmetrisk modulooperation för att få ned värdet till lämplig nivå. För mer information, se ▶**Base2**, på sidan 21.

Obs: Se **eller**, på sidan 130.

I hexadecimalt basläge:

Viktigt: Noll, inte bokstaven O.

0h7AC36 xor 0h3D5F	0h79169

I binärt basläge:

0b100101 xor 0b100	0b100001

Obs: En binär inmatning kan ha upp till 64 siffror (exklusive prefixet 0b). En hexadecimal inmatning kan ha upp till 16 siffror.

Z

zeros() **Katalog >**

zeros(*Expr***,** *Var***)⇒***lista*

zeros(*Expr***,** *Var*=*Guess***)⇒***lista*

Ger en lista på möjliga reella värden på *Var* som gör *Expr*=0. **zeros()** gör detta genom att beräkna **exp▶list(solve(***Expr*=0,*Var***),***Var***)**.

$$\text{zeros}\left(a\cdot x^2+b\cdot x+c,x\right)$$
$$\left\{\frac{\sqrt{b^2-4\cdot a\cdot c}-b}{2\cdot a},\frac{-\left(\sqrt{b^2-4\cdot a\cdot c}+b\right)}{2\cdot a}\right\}$$

$$a\cdot x^2+b\cdot x+c\,|\,x=Ans[2] \qquad 0$$

För vissa ändamål är resultatformen för **zeros()** mer praktisk än den för **solve()**. Resultatformen för **zeros()** kan dock inte uttrycka implicita lösningar, lösningar som kräver olikheter eller lösningar som inte inbegriper *Var*.

$\text{exact}\left(\text{zeros}\left(a\cdot\left(e^x+x\right)\cdot\left(\text{sign}\left(x\right)-1\right),x\right)\right)$ $\{\square\}$

$\text{exact}\left(\text{solve}\left(a\cdot\left(e^x+x\right)\cdot\left(\text{sign}\left(x\right)-1\right)=0,x\right)\right)$

$e^x+x=0 \text{ or } x>0 \text{ or } a=0$

Obs: Se även **cSolve()**, **cZeros()** och **solve()**.

zeros({*Expr1***,** *Expr2***}, {***VarOrGuess1***,** *VarOrGuess2* **[,** *...* **]})⇒***matris*

Ger möjliga reella nollställen för de samtidiga algebraiska uttrycken där varje *VarOrGuess* specificerar en okänd vars värde du söker.

Du kan som alternativ specificera en initial gissning för en variabel. Varje *VarOrGuess* måste ha formen:

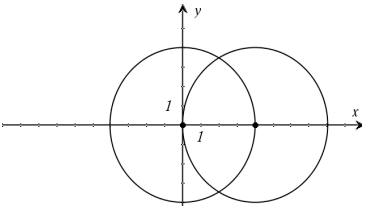
variable

– eller –

variable = reellt eller icke-reellt tal

Som exempel är x giltigt och likaså x=3.

Om alla uttryck är polynom och om du INTE specificerar några initiala gissningar använder **zeros()** eliminationsmetoden Gröbner/Buchberger för att försöka bestämma alla reella nollställen.



Lås oss som exempel anta att du har en cirkel med radien r i origo och en annan cirkel med radien r där centrum är där den första cirkeln korsar den positiva x-axeln. Använd **zeros()** för att finna skärningspunkterna.

$\text{zeros}\left(\left\{x^2+y^2-r^2,\left(x-r\right)^2+y^2-r^2\right\},\left\{x,y\right\}\right)$

$\begin{bmatrix}\frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2}\\ \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2}\end{bmatrix}$

Såsom illustreras med r i exemplet till höger kan system av polynomuttryck ha extra variabler som saknar värden, men representerar givna numeriska värden som kan ersättas senare.

Varje rad i den resulterande matrisen representerar ett alternativt nollställe, med komponenterna ordnade på samma sätt som i listan *VarOrGuess*. För att extrahera en rad, indexera matrisen med [*row*].

Du kan även (eller i stället) inkludera okända som inte visas i uttrycken. Du kan exempelvis inkludera z som en okänd för att utöka föregående exempel till två parallella överkorsande cylindrar med radien r. Cylinderlösningarna illustrerar hur familjer av nollställen kan innehålla godtyckliga konstanter med formen ck där k är ett heltalssuffix från 1 till och med 255.

För polynomsystem kan beräkningstiden och användningen av minne i hög grad bero på i vilken ordning du listar okända. Om ditt första val utarmar minnet, eller tar på ditt tålamod, kan du försöka med att arrangera om variablerna i uttrycken och/eller i listan *VarOrGuess*.

Om du inte inkluderar några gissningar och om något uttryck är ett icke-polynom i någon variabel, men alla uttryck är linjära i de okända, använder **zeros()** Gauss eliminationsmetod för att försöka bestämma alla reella nollställen.

Om ett system är varken polynomt i alla dess variabler eller linjärt i dess okända bestämmer **zeros()** högst ett nollställe med en ungefärlig iterativ metod. För att göra detta måste antalet okända vara lika med antalet uttryck och alla övriga variabler i uttrycken måste förenklas till tal.

Varje okänd startar vid dess gissade värde om det finns något, annars startar den vid 0.0.

Använd gissningar för att söka ytterligare nollställen ett i taget. För konvergens kan en gissning behöva vara ganska nära ett nollställe.

Extrahera rad 2:

$$Ans[2] \qquad \begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} \end{bmatrix}$$

$$\text{zeros}\left(\left\{x^2+y^2-r^2,(x-r)^2+y^2-r^2\right\},\{x,y,z\}\right)$$

$$\begin{bmatrix} \frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2} & c1 \\ \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} & c1 \end{bmatrix}$$

$$\text{zeros}\left(\left\{x+e^z\cdot y-1,x-y-\sin(z)\right\},\{x,y\}\right)$$

$$\begin{bmatrix} \frac{e^z\cdot\sin(z)+1}{e^z+1} & \frac{-(\sin(z)-1)}{e^z+1} \end{bmatrix}$$

$$\text{zeros}\left(\left\{e^z\cdot y-1,-y-\sin(z)\right\},\{y,z\}\right)$$

$$\begin{bmatrix} 0.041458 & 3.18306 \\ 0.001871 & 6.28131 \\ 4.76\text{E}{-11} & 1796.99 \\ 2.\text{E}{-13} & 254.469 \end{bmatrix}$$

$$\text{zeros}\left(\left\{e^z\cdot y-1,-y-\sin(z)\right\},\{y,z=2\cdot\pi\}\right)$$

$$\begin{bmatrix} 0.001871 & 6.28131 \end{bmatrix}$$

zInterval

zInterval σ,*List*[,*Freq*[,*CLevel*]]

zInterval

Katalog >

(Indatalista)

zInterval σ,$\bar{x}$,*n* [,*CLevel*]

(Summary stats indata)

Beräknar ett *z*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall för ett okänt populationsmedelvärde
stat.$\bar{x}$	Urvalsmedelvärde för datasekvensen från den slumpmässiga normalfördelningen
stat.ME	Felmarginal
stat.sx	Standardavvikelse för urvalet
stat.n	Längden på datasekvenser med urvalsmedelvärde
stat.σ	Känd populationsstandardavvikelse för datasekvensen *List*

zInterval_1Prop

Katalog >

zInterval_1Prop *x*,*n* [,*CLevel*]

Beräknar ett 1-proportion *z*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

x är ett icke negativt heltal.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall innehållande konfidensnivån för fördelningens sannolikhet
stat.$\hat{p}$	Den beräknade proportionen lyckade försök

Resultatvariabel	Beskrivning
stat.ME	Felmarginal
stat.n	Antalet samplingar i datasekvens

zInterval_2Prop

Katalog >

zInterval_2Prop *x1*,*n1*,*x2*,*n2*[,*CLevel*]

Beräknar ett 2-proportion *z*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

x1 och *x2* är icke negativa heltal.

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall innehållande konfidensnivån för fördelningens sannolikhet
stat.$\hat{p}$Diff	Den beräknade skillnaden mellan proportioner
stat.ME	Felmarginal
stat.$\hat{p}$1	Proportionsuppskattning för första urvalet
stat.$\hat{p}$2	Proportionsuppskattning för andra urvalet
stat.n1	Urvalets storlek i datasekvens ett
stat.n2	Urvalets storlek i datasekvens två

zInterval_2Samp

Katalog >

zInterval_2Samp σ_1,σ_2,*List1*,*List2*[,*Freq1*[,*Freq2*,[*CLevel*]]]

(Indatalista)

zInterval_2Samp σ_1,σ_2,$\bar{x}1$,*n1*,$\bar{x}2$,*n2*[,*CLevel*]

(Summary stats indata)

Beräknar ett 2-sampel *z*-konfidensintervall. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

zInterval_2Samp

Katalog >

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.CLower, stat.CUpper	Konfidensintervall innehållande konfidensnivån för fördelningens sannolikhet
stat.$\bar{x}1$-$\bar{x}2$	Urvalsmedelvärden för datasekvenserna från den slumpmässiga normalfördelningen
stat.ME	Felmarginal
stat.$\bar{x}1$, stat.$\bar{x}2$	Urvalsmedelvärden för datasekvenserna från den slumpmässiga normalfördelningen
stat.σx1, stat.σx2	Urvalets standardavvikelser för *List 1* och *List 2*
stat.n1, stat.n2	Antalet samplingar i datasekvenser
stat.r1, stat.r2	Kända populationsstandardavvikelser för datasekvenserna *List 1* och *List 2*

zTest

Katalog >

zTest μ*0*,σ,*List*,[*Freq*[,*Hypoth*]]

(Indatalista)

zTest μ*0*,σ,$\bar{x}$,*n*[,*Hypoth*]

(Summary stats indata)

Utför ett *z*-test med frekvensen *freqlist*. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Testa H_0: μ = μ0, mot en av följande:

För H_a: μ < μ0, ställ *Hypoth*<0

För H_a: μ ≠ μ0 (förinställning), ställ *Hypoth*=0

För H_a: μ > μ0, ställ *Hypoth*>0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.z	$(\bar{x} - \mu 0) / (\sigma / \text{sqrt}(n))$
stat.P Value	Lägsta sannolikhet vid vilken nollhypotesen kan förkastas
stat.$\bar{x}$	Urvalsmedelvärde för datasekvensen i *List*
stat.sx	Urvalets standardavvikelse för datasekvensen. Ges endast för *Data*-inmatningen.
stat.n	Urvalets storlek

zTest_1Prop

Katalog >

zTest_1Prop *p0*,*x*,*n*[,*Hypoth*]

Beräknar ett 1-proportion *z*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

x är ett icke negativt heltal.

Testa H_0: $p = p0$ mot en av följande:

För H_a: $p > p0$, ställ *Hypoth*>0

För H_a: $p \neq p0$ *(förinställning)*, ställ *Hypoth*=0

För H_a: $p < p0$, ställ *Hypoth*<0

För information om effekten av tomma element i en lista, se “Tomma element” (på sidan 238).

Resultatvariabel	Beskrivning
stat.p0	Hypotetisk populationsproportion
stat.z	Standardnormalvärde beräknat för proportionen
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.$\hat{p}$	Uppskattad urvalsproportion
stat.n	Urvalets storlek

zTest_2Prop

Katalog >

zTest_2Prop *x1*,*n1*,*x2*,*n2*[,*Hypoth*]

zTest_2Prop

Katalog >

Beräknar ett 2-proportion *z*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

x1 och *x2* är icke negativa heltal.

Testa H_0: *p1 = p2* mot en av följande:

För H_a: *p1 > p2*, ställ *Hypoth*>0

För H_a: *p1 ≠ p2 (förinställning)*, ställ *Hypoth*=0

För H_a: *p < p0*, ställ *Hypoth*<0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.z	Standardnormalvärde beräknat för skillnaden mellan proportioner
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.$\hat{p}1$	Proportionsuppskattning för första urvalet
stat.$\hat{p}2$	Proportionsuppskattning för andra urvalet
stat.$\hat{p}$	Uppskattning av sammanslagna urvalsproportioner
stat.n1, stat.n2	Antal samplingar i försöksomgång 1 och 2

zTest_2Samp

Katalog >

zTest_2Samp σ_1,σ_2, *List1*,*List2*[,*Freq1*[,*Freq2*[,*Hypoth*]]]

(Indatalista)

zTest_2Samp σ_1,σ_2,$\bar{x}1$,*n1*,$\bar{x}2$,*n2*[,*Hypoth*]

(Summary stats indata)

Beräknar ett 2-sampel *z*-test. En sammanfattning av resultaten visas i variabeln *stat.results*. (Se på sidan 177.)

Testa H_0: $\mu1 = \mu2$, mot en av följande:

För H_a: $\mu1 < \mu2$, ställ *Hypoth*<0

För H_a: $\mu 1 \neq \mu 2$ (förinställning), ställ *Hypoth*=0

För H_a: $\mu 1 > \mu 2$, ställ *Hypoth*>0

För information om effekten av tomma element i en lista, se "Tomma element" (på sidan 238).

Resultatvariabel	Beskrivning
stat.z	Standardnormalvärde beräknat för skillnaden mellan medelvärden
stat.PVal	Lägsta signifikansnivå vid vilken nollhypotesen kan förkastas
stat.$\bar{x}1$, stat.$\bar{x}2$	Medelvärden hos urvalet i datasekvenserna i *List1* och *List2*
stat.sx1, stat.sx2	Standardavvikelser hos urvalet i datasekvenserna i *List1* och *List2*
stat.n1, stat.n2	Storlek på urvalen

Symboler

+ (addera) — +tangent

Expr1 **+** *Expr2*⇒*uttryck*

Ger summan av de två argumenten.

56	56
$56+4$	60
$60+4$	64
$64+4$	68
$68+4$	72

List1 **+** *List2*⇒*lista*

Matrix1 **+** *Matrix2*⇒*matris*

Ger en lista (eller matris) som innehåller summorna av motsvarande element i *List1* och *List2* (eller *Matrix1* och *Matrix2*).

Argumenten måste ha samma dimensioner.

$\left\{22,\pi,\frac{\pi}{2}\right\}\rightarrow l1$	$\left\{22,\pi,\frac{\pi}{2}\right\}$
$\left\{10,5,\frac{\pi}{2}\right\}\rightarrow l2$	$\left\{10,5,\frac{\pi}{2}\right\}$
$l1+l2$	$\{32,\pi+5,\pi\}$
$Ans+\{\pi,-5,-\pi\}$	$\{\pi+32,\pi,0\}$
$\begin{bmatrix}a & b\\ c & d\end{bmatrix}+\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}a+1 & b\\ c & d+1\end{bmatrix}$

Expr **+** *List1*⇒*lista*

List1 **+** *Expr*⇒*lista*

Ger en lista på summorna av *Expr* och varje element i *List1*.

$15+\{10,15,20\}$	$\{25,30,35\}$
$\{10,15,20\}+15$	$\{25,30,35\}$

Expr **+** *Matrix1*⇒*matrixs*

Matrix1 **+** *Expr*⇒*matris*

Ger en matris med *Expr* adderat till varje element i diagonalen av *Matrix1*. *Matrix1* måste vara kvadratisk.

Obs: Använd **.+** (punkt plus) för att lägga till ett uttryck till varje element.

$20+\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$	$\begin{bmatrix}21 & 2\\ 3 & 24\end{bmatrix}$

−(subtrahera) — −tangent

Expr1 − *Expr2*⇒*uttryck*

Ger *Expr1* minus *Expr2*.

$6-2$	4
$\pi-\frac{\pi}{6}$	$\frac{5\cdot\pi}{6}$

List1 − *List2*⇒*lista*

Matrix1 − *Matrix2*⇒*matris*

$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\}$	$\{12,\pi-5,0\}$
$[3\ \ 4]-[1\ \ 2]$	$[2\ \ 2]$

−(subtrahera) — −tangent

Subtraherar varje element i *List2* (eller *Matrix2*) från motsvarande element i *List1* (eller *Matrix1*) och ger resultatet.

Argumenten måste ha samma dimensioner.

Expr − *List1*⇒*lista*

List1 − *Expr*⇒*lista*

Subtraherar varje element i *List1* från *Expr* eller subtraherar *Expr* från varje element i *List1* och ger en lista på resultaten.

$15-\{10,15,20\}$ → $\{5,0,-5\}$

$\{10,15,20\}-15$ → $\{-5,0,5\}$

Expr − *Matrix1*⇒*matris*

Matrix1 − *Expr*⇒*matris*

Expr − *Matrix1* ger en matris över *Expr* gånger enhetsmatrisen minus *Matrix1*. *Matrix1* måste vara kvadratisk.

Matrix1 − *Expr* ger en matris över *Expr* gånger enhetsmatrisen subtraherad från *Matrix1*. *Matrix1* måste vara kvadratisk.

$20-\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ → $\begin{bmatrix}19 & -2\\ -3 & 16\end{bmatrix}$

Obs: Använd **.**− (punkt minus) för att subtrahera ett uttryck från varje element.

·(multiplicera) — ×tangent

Expr1 ·*Expr2*⇒*uttryck*

Ger produkten av de två argumenten.

$2\cdot 3.45$ → 6.9

$x\cdot y\cdot x$ → $x^2\cdot y$

List1·*List2*⇒*lista*

Ger en lista som innehåller produkterna av motsvarande element i *List1* och *List2*.

$\{1.,2,3\}\cdot\{4,5,6\}$ → $\{4.,10,18\}$

$\left\{\frac{2}{a},\frac{3}{2}\right\}\cdot\left\{a^2,\frac{b}{3}\right\}$ → $\left\{2\cdot a,\frac{b}{2}\right\}$

Listorna måste ha samma dimensioner.

Matrix1 ·*Matrix2*⇒*matris*

Ger en matris över produkten av *Matrix1* och *Matrix2*.

Antalet kolumner i *Matrix1* måste vara lika med antalet rader i *Matrix2*.

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}.\begin{bmatrix}a & d\\ b & e\\ c & f\end{bmatrix}$ →

$$\begin{bmatrix}a+2\cdot b+3\cdot c & d+2\cdot e+3\cdot f\\ 4\cdot a+5\cdot b+6\cdot c & 4\cdot d+5\cdot e+6\cdot f\end{bmatrix}$$

Expr ·*List1*⇒*lista*

$\pi\cdot\{4,5,6\}$ → $\{4\cdot\pi,5\cdot\pi,6\cdot\pi\}$

·(multiplicera)

×tangent

List1 ·*Expr*⇒*lista*

Ger en lista på produkterna av *Expr* och varje element i *List1*.

Expr ·*Matrix1*⇒*matris*

Matrix1 ·*Expr*⇒*matris*

Ger en matris över produkterna av *Expr* och varje element i *Matrix1*.

Obs: Använd **.**·(punkt multiplicera) för att multiplicera ett uttryck med varje element.

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\cdot 0.01 \qquad \begin{bmatrix}0.01 & 0.02\\ 0.03 & 0.04\end{bmatrix}$

$\lambda\cdot\text{identity}(3) \qquad \begin{bmatrix}\lambda & 0 & 0\\ 0 & \lambda & 0\\ 0 & 0 & \lambda\end{bmatrix}$

/ (dividera)

÷tangent

Expr1 / *Expr2*⇒*uttryck*

Ger kvoten av *Expr1* dividerat med *Expr2*.

Obs: Se även **Fraction template**, på sidan 5.

$\frac{2}{3.45} \qquad .57971$

$\frac{x^3}{x} \qquad x^2$

List1 / *List2*⇒*lista*

Ger en lista på kvoterna av *List1* dividerat med *List2*.

Listorna måste ha samma dimensioner.

$\frac{\{1.,2,3\}}{\{4,5,6\}} \qquad \left\{0.25,\frac{2}{5},\frac{1}{2}\right\}$

Expr / *List1* ⇒ *lista*

List1 / *Expr* ⇒ *lista*

Ger en lista på kvoterna av *Expr* dividerat med *List1* eller*List1* dividerat med *Expr*.

$\frac{a}{\{3,a,\sqrt{a}\}} \qquad \left\{\frac{a}{3},1,\sqrt{a}\right\}$

$\frac{\{a,b,c\}}{a\cdot b\cdot c} \qquad \left\{\frac{1}{b\cdot c},\frac{1}{a\cdot c},\frac{1}{a\cdot b}\right\}$

Matrix1 / *Expr* ⇒ *matris*

Ger en matris över kvoterna av *Matrix1*/*Expr*.

$\frac{[a\ \ b\ \ c]}{a\cdot b\cdot c} \qquad \begin{bmatrix}\frac{1}{b\cdot c} & \frac{1}{a\cdot c} & \frac{1}{a\cdot b}\end{bmatrix}$

Obs: Använd **. /** (punkt dividera) för att dividera ett uttryck med varje element.

^ (potens)

^tangent

Expr1 ^ *Expr2* ⇒ *uttryck*

List1 **^** *List2* ⇒ *lista*

$4^2 \qquad 16$

$\{a,2,c\}^{\{1,b,3\}} \qquad \left\{a,2^b,c^3\right\}$

^ (potens)

^ tangent

Ger det första argumentet upphöjt till det andra argumentets potens.

Obs: Se även **Exponent template**, på sidan 5.

Ger, för en lista, elementen i *List1* upphöjda till potensen för motsvarande element i *List2*.

I det reella området använder potenser i bråkform som har reducerade exponenter med udda nämnare den reella delen kontra principaldelen för komplext läge.

Expr ^ *List1* ⇒ *lista*

Ger *Expr* upphöjt till potensen för elementen i *List1*.

List1 ^ *Expr* ⇒ *lista*

Ger elementen i *List1* upphöjda till potensen för *Expr*.

squareMatrix1 **^** *integer* ⇒ *matris*

Ger *squareMatrix1* upphöjd till potensen för *integer* (heltal).

squareMatrix1 måste vara en kvadratisk matris.

Om *integer* = ⁻1 beräknas den inversa matrisen.

Om *integer* < ⁻1 beräknas den inversa matrisen till en lämplig positiv potens.

$p^{\{a,2,-3\}}$ → $\left\{p^a, p^2, \frac{1}{p^3}\right\}$

$\{1,2,3,4\}^{-2}$ → $\left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}\right\}$

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^2$ → $\begin{bmatrix}7 & 10\\ 15 & 22\end{bmatrix}$

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1}$ → $\begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-2}$ → $\begin{bmatrix}\frac{11}{2} & \frac{-5}{2}\\ \frac{-15}{4} & \frac{7}{4}\end{bmatrix}$

x² (kvadrat)

x² tangent

*Expr1*² ⇒ *uttryck*

Ger kvadraten på argumentet.

*List1*² ⇒ *lista*

Ger en lista med kvadraterna på elementen i *List1*.

*squareMatrix1*² ⇒ *matris*

4^2 → 16

$\{2,4,6\}^2$ → $\{4,16,36\}$

$\begin{bmatrix}2 & 4 & 6\\ 3 & 5 & 7\\ 4 & 6 & 8\end{bmatrix}^2$ → $\begin{bmatrix}40 & 64 & 88\\ 49 & 79 & 109\\ 58 & 94 & 130\end{bmatrix}$

$\begin{bmatrix}2 & 4 & 6\\ 3 & 5 & 7\\ 4 & 6 & 8\end{bmatrix}.^\wedge 2$ → $\begin{bmatrix}4 & 16 & 36\\ 9 & 25 & 49\\ 16 & 36 & 64\end{bmatrix}$

x² (kvadrat)

x² tangent

Ger en matris över kvadraten på *squareMatrix1*. Detta är inte detsamma som att beräkna kvadraten på varje element. Använd .^2 för att beräkna kvadraten på varje element.

.+ (punkt addera)

. + tangenter

Matrix1 **.+** *Matrix2* ⇒ *matris*

Expr **.+** *Matrix1* ⇒ *matris*

Matrix1 **.+** *Matrix2* ger en matris som är summan av varje par av motsvarande element i *Matrix1* och *Matrix2*.

Expr **.+** *Matrix1* ger en matris som är summan av *Expr* och varje element i *Matrix1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \quad \begin{bmatrix} a+c & 6 \\ b+5 & d+3 \end{bmatrix}$$

$$x .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \quad \begin{bmatrix} x+c & x+4 \\ x+5 & x+d \end{bmatrix}$$

.− (punkt subtrahera)

. − tangenter

Matrix1 **.−** *Matrix2* ⇒ *matrixs*

Expr **.−***Matrix1* ⇒ *matris*

Matrix1 **.−***Matrix2* ger en matris som är skillnaden mellan varje par av motsvarande element i *Matrix1* och *Matrix2*.

Expr **.−***Matrix1* ger en matris som är skillnaden mellan *Expr* och varje element i *Matrix1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \quad \begin{bmatrix} a-c & -2 \\ b-d & -2 \end{bmatrix}$$

$$x .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \quad \begin{bmatrix} x-c & x-4 \\ x-d & x-5 \end{bmatrix}$$

. ·(punkt multiplicera)

. × tangenter

Matrix1 **.·** *Matrix2* ⇒ *matris*

Expr **.·** *Matrix1* ⇒ *matris*

Matrix1 **.·** *Matrix2* ger en matris som är produkten av varje par av motsvarande element i *Matrix1* och *Matrix2*.

Expr **.·** *Matrix1* ger en matris som innehåller produkterna av *Expr* och varje element i *Matrix1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .\cdot \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \quad \begin{bmatrix} a\cdot c & 8 \\ 5\cdot b & 3\cdot d \end{bmatrix}$$

$$x .\cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \begin{bmatrix} a\cdot x & b\cdot x \\ c\cdot x & d\cdot x \end{bmatrix}$$

. / (punkt dividera)

. ÷ tangenter

Matrix1 **. /** *Matrix2* ⇒ *matris*

Expr **. /** *Matrix1* ⇒ *matris*

Matrix1 **. /** *Matrix2* ger en matris som är kvoten av varje par av motsvarande element i *Matrix1* och *Matrix2*.

Expr **. /** *Matrix1* ger en matris som är kvoten av *Expr* och varje element i *Matrix1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} ./ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} \frac{a}{c} & \frac{1}{2} \\ \frac{b}{5} & \frac{3}{d} \end{bmatrix}$$

$$x ./ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} \frac{x}{c} & \frac{x}{4} \\ \frac{x}{5} & \frac{x}{d} \end{bmatrix}$$

.^ (punkt potens)

. ^ tangenter

Matrix1 **.^** *Matrix2* ⇒ *matris*

Expr **. ^** *Matrix1* ⇒ *matris*

Matrix1 **.^** *Matrix2* ger en matris där varje element i *Matrix2* är exponenten för motsvarande element i *Matrix1*.

Expr **.^** *Matrix1* ger en matris där varje element i *Matrix1* är exponenten för *Expr*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .^ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} a^c & 16 \\ b^5 & 3^d \end{bmatrix}$$

$$x .^ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} x^c & x^4 \\ x^5 & x^d \end{bmatrix}$$

-(negation)

(−) tangent

-*Expr1* ⇒ *uttryck*

-*List1* ⇒ *lista*

-*Matrix1* ⇒ *matris*

Ger argumentets negation.

Ger, för en lista eller matris, alla elementen negerade.

Om argumentet är ett binärt eller hexadecimalt heltal ger negationen tvåkomplementet.

-2.43	-2.43
-{-1,0.4,1.2E19}	{1,-0.4,-1.2E19}
-*a*· -*b*	*a*· *b*

I binärt basläge:

Viktigt: Noll, inte bokstaven O.

```
-0b100101
0b1111111111111111111111111111111▸
```

För att se hela resultatet, tryck på ▲ och använd sedan ◀ och ▶ för att flytta markören.

% (procent)

ctrl 🕮 tangenter

Expr1 **%** ⇒ *uttryck*

List1 **%** ⇒ *lista*

Matrix1 **%** ⇒ *matris*

argument
Ger 100

Ger, för en lista eller matris, en lista eller matris med varje element dividerat med 100.

Obs: För att få ett närmevärde,

Handenhet: Tryck på ctrl enter.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘**+Enter**.
iPad®: Håll ned **enter** och välj ≈.

13%	0.13
{{1,10,100}}%	{0.01,0.1,1.}

= (lika med)

= tangent

Expr1 **=** *Expr2*⇒*Booleskt uttryck*

List1 **=** *List2*⇒ *Boolesk lista*

Matrix1 **=** *Matrix2*⇒ *Boolesk matris*

Ger resultatet sant om *Expr1* bestäms vara lika med *Expr2*.

Ger resultatet falskt om *Expr1* bestäms inte vara lika med *Expr2*.

Allt annat ger en förenklad form av ekvationen.

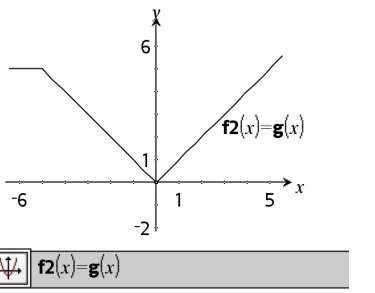
Ger, för listor och matriser, jämförelser element för element.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitioner.

Exempel på funktion som använder matchande testsymboler: =, ≠, <, ≤, >, ≥

```
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
```

Resultat från plottning av g(x)



≠ (inte lika med)

ctrl = tangenter

Expr1 ≠ *Expr2* ⇒ *Booleskt uttryck*

List1 ≠ *List2* ⇒ *Boolesk lista*

Matrix1 ≠ *Matrix2* ⇒ *Boolesk matris*

Ger resultatet sant om *Expr1* bestäms inte vara lika med *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara lika med *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **/=**

Se exemplet "=" (lika med).

< (mindre än)

ctrl = tangenter

Expr1 < *Expr2* ⇒ *Booleskt uttryck*

List1 < *List2* ⇒ *Boolesk lista*

Matrix1 < *Matrix2* ⇒ *Boolesk matris*

Ger resultatet sant om *Expr1* bestäms vara mindre än *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara större än eller lika med *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Se exemplet "=" (lika med).

≤ (mindre än eller lika med)

ctrl = tangenter

Expr1 ≤ *Expr2* ⇒ *Booleskt uttryck*

List1 ≤ *List2* ⇒ *Boolesk lista*

Matrix1 ≤ *Matrix2* ⇒ *Boolesk matris*

Se exemplet "=" (lika med).

≤ (mindre än eller lika med)

ctrl = tangenter

Ger resultatet sant om *Expr1* bestäms vara mindre än eller lika med *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara större än *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **<=**

> (större än)

ctrl = tangenter

$Expr1 > Expr2 \Rightarrow Booleskt\ uttryck$

Se exemplet "=" (lika med).

$List1 > List2 \Rightarrow Boolesk\ lista$

$Matrix1 > Matrix2 \Rightarrow Boolesk\ matris$

Ger resultatet sant om *Expr1* bestäms vara större än *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara mindre än eller lika med *Expr2*.

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

≥ (större än eller lika med)

ctrl = tangenter

$Expr1 \geq Expr2 \Rightarrow Booleskt\ uttryck$

Se exemplet "=" (lika med).

$List1 \geq List2 \Rightarrow Boolesk\ lista$

$Matrix1 \geq Matrix2 \Rightarrow Boolesk\ matris$

Ger resultatet falskt om *Expr1* bestäms vara större än eller lika med *Expr2*.

Ger resultatet falskt om *Expr1* bestäms vara mindre än *Expr2*.

≥ (större än eller lika med)

ctrl = tangenter

Allt annat ger en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **>=**

⇒ (logisk implikation)

ctrl = knappar

BooleskUttr1 ⇒ *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 ⇒ *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 ⇒ *BooleskMatris2* ger *Boolesk matris*

Heltal1 ⇒ *Heltal2* ger *Heltal*

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}

Beräknar uttrycket **not <argument1> or <argument2>** och ger resultatet sant, falskt eller en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **=>**

⇔ (logisk dubbel implikation, XNOR)

ctrl = knappar

BooleskUttr1 ⇔ *BooleskUttr2* ger *Booleskt uttryck*

BooleskLista1 ⇔ *BooleskLista2* ger *Boolesk lista*

BooleskMatris1 ⇔ *BooleskMatris2* ger *Boolesk matris*

Heltal1 ⇔ *Heltal2* ger *Heltal*

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}

⇔ (logisk dubbel implikation, XNOR) ctrl = knappar

Ger negation av en **XOR** Boolesk operation på de två argumenten. Ger resultatet sant, falskt eller en förenklad form av ekvationen.

Ger, för listor och matriser, jämförelser element för element.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **<=>**

! (fakultet) ?!▸ tangent

Expr1! ⇒ *uttryck*

List1! ⇒ *lista*

Matrix1! ⇒ *matris*

5!	120
$(\{5,4,3\})!$	$\{120,24,6\}$
$\left(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\right)!$	$\begin{bmatrix}1 & 2\\6 & 24\end{bmatrix}$

Ger argumentets fakultet.

Ger, för en lista eller matris, en lista eller matris med elementens fakulteter.

& (lägg till) ctrl 📖 tangenter

String1 **&** *String2* ⇒ *sträng*

"Hello "&"Nick"	"Hello Nick"

Ger en textsträng som är *String2* bifogad till *String1*.

d() (derivata) Katalog > 📖

d(*Uttr1***,** *Var*[**,** *Ordning*]**)**⇒*uttryck*

d(*Lista1***,** *Var*[**,** *Ordning*]**)**⇒*lista*

d(*Matris1***,** *Var*[**,** *Ordning*]**)**⇒*matris*

$\frac{d}{dx}(f(x)\cdot g(x))$	$\frac{d}{dx}(f(x))\cdot g(x)+\frac{d}{dx}(g(x))\cdot f(x)$
$\frac{d}{dy}\left(\frac{d}{dx}\left(x^2\cdot y^3\right)\right)$	$6\cdot y^2\cdot x$
$\frac{d}{dx}\left(\left\{x^2,x^3,x^4\right\}\right)$	$\left\{2\cdot x,3\cdot x^2,4\cdot x^3\right\}$

Ger förstaderivatan av det första argumentet med avseende på variabeln *Var*.

Ordning, om inkluderad, måste vara ett heltal. Om ordningen är mindre än noll blir resultatet en antiderivata.

d() (derivata)

Katalog >

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`derivative(...)`**.

d() följer inte den normala utvärderingsmekanismen för att fullt förenkla dess argument och sedan tillämpa funktionsdefinitionen på dessa fullt förenklade argument. I stället utför **d()** följande steg:

1. Förenklar det andra argumentet endast i sådan utsträckning att det inte leder till en icke-variabel.
2. Förenklar det första argumentet endast i sådan utsträckning att det inte hämtar något lagrat värde för den variabel som bestäms av steg 1.
3. Bestämmer den symboliska derivatan av resultatet från steg 2 med avseende på variabeln från steg 1.

Om variabeln från steg 1 har ett lagrat värde eller ett värde specificerat med ("|")-operatorn begränsning, substituera det värdet i resultatet från steg 3.

Obs: Se även **Förstaderivata**, på sidan 9, **Andraderivata**, på sidan 10 eller **N:te derivata**, på sidan 10.

∫() (integrera)

Katalog >

∫(*Uttr1*, *Var*[, *Undre*, *Övre*]) ⇒ *uttryck*

∫(*Uttr1*, *Var*[, *Konstant*]) ⇒ *uttryck*

Ger integralen av *Uttr1* med avseende på *Var* från *Undre* till *Övre*.

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

Obs: Se även **Mall för bestämd integral** eller **Mall för obestämd integral**, på sidan 10.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`integral(...)`**.

Om *Lower* och *Upper* utelämnas erhålls en primitiv funktion. En symbolisk integrationskonstant utelämnas såvida du inte anger argumentet *Constant*.

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

$\int(a \cdot x^2, x, c) \qquad \frac{a \cdot x^3}{3} + c$

Jämbördigt giltiga antiderivator kan skilja med en numerisk konstant. En sådan konstant kan vara maskerad, särskilt när en antiderivata innehåller logaritmer eller inversa trigonometriska funktioner. Dessutom läggs ibland stegvisa konstanta uttryck till för att göra en antiderivata giltig över ett större intervall än med den vanliga formeln.

∫() returnerar sig själv för steg av *Expr1* som inte kan bestämmas som en explicit ändlig kombination av dess inbyggda funktioner och operatorer.

$\int b \cdot e^{-x^2} + \frac{a}{x^2 + a^2} \, dx \qquad b \cdot \int e^{-x^2} \, dx + \tan^{-1}\left(\frac{x}{a}\right)$

När du anger *Lower* och *Upper* görs ett försök att hitta eventuella diskontinuiteter eller diskontinuerliga derivator i intervallet *Lower* < *Var* < *Upper* för att dela upp intervallet vid dessa ställen.

Med inställningen Auto i läge **Auto eller Ungefärlig** används numerisk integrering där så är tillämpligt när en antiderivata eller ett gränsvärde inte kan bestämmas.

Med inställningen Approximate görs först ett försök med numerisk integrering om detta är tillämpligt. Antiderivata söks endast där sådan numerisk integrering inte är tillämplig eller misslyckas.

Obs: För att få ett närmevärde,

Handenhet: Tryck på ctrl enter.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘+**Enter**.
iPad®: Håll ned **enter** och välj ≈.

$\int_{-1}^{1} e^{-x^2} \, dx \qquad 1.49365$

∫() (integrera)

Katalog >

∫() kan kapslas in för att göra multipelintegraler. Integrationsgränser kan bero på integrationsvariabler utanför gränserna.

Obs: Se även **nInt()**, på sidan 123.

$$\int_0^a \int_0^x \ln(x+y)\, dy\, dx \qquad \frac{a^2\cdot\ln(a)}{2}+\frac{a^2\cdot(4\cdot\ln(2)-3)}{4}$$

√() (kvadratrot)

ctrl **x²** **tangenter**

√ (*Expr1*)⇒*uttryck*

√ (*List1*)⇒*lista*

Ger kvadratroten ur argumentet.

Ger, för en lista, kvadratrötterna ur alla element i *List1*.

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **sqrt(...)**

Obs: Se även **Square root template**, på sidan 5.

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

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

Π() (prodSeq)

Katalog >

Π(*Expr1*, *Var*, *Low*, *High*)⇒*uttryck*

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **prodSeq(...)**.

Utvärderar *Expr1* för varje värde på *Var* från *Low* till *High* och ger produkten av resultaten.

Obs: Se även **Product template (Π)**, på sidan 9.

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

$$\prod_{k=1}^{n}\left(k^2\right) \qquad (n!)^2$$

$$\prod_{n=1}^{5}\left(\left\{\frac{1}{n},n,2\right\}\right) \qquad \left\{\frac{1}{120},120,32\right\}$$

Π(*Expr1*, *Var*, *Low*, *Low*−1)⇒1

Π(*Expr1*, *Var*, *Low*, *High*) ⇒**1/**Π(*Expr1*, *Var*, *High*+1, *Low*−1) om *High* < *Low*−1

$$\prod_{k=4}^{3}(k) \qquad 1$$

Produktformlerna som används har härletts från följande referens:

Π() (prodSeq)

Katalog >

Ronald L. Graham, Donald E. Knuth och Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$$\prod_{k=4}^{1}\left(\frac{1}{k}\right) \qquad 6$$

$$\prod_{k=4}^{1}\left(\frac{1}{k}\right)\cdot\prod_{k=2}^{4}\left(\frac{1}{k}\right) \qquad \frac{1}{4}$$

Σ() (sumSeq)

Katalog >

Σ(*Expr1***,** *Var***,** *Low***,** *High***)⇒***uttryck*

Obs: Du kan infoga denna funktion med datorns tangentbord genom att skriva **`sumSeq(...)`**.

Utvärderar *Uttr1* för varje värde på *Var* från *Låg* till *Hög* och ger summan av resultaten.

Obs: Se även **Sum template**, på sidan 9.

$$\sum_{n=1}^{5}\left(\frac{1}{n}\right) \qquad \frac{137}{60}$$

$$\sum_{k=1}^{n}\left(k^2\right) \qquad \frac{n\cdot(n+1)\cdot(2\cdot n+1)}{6}$$

$$\sum_{n=1}^{\infty}\left(\frac{1}{n^2}\right) \qquad \frac{\pi^2}{6}$$

Σ(*Expr1***,** *Var***,** *Low***,** *Low*−*1***)⇒**0

Σ(*Expr1***,** *Var***,** *Low***,** *High***) ⇒**-**Σ(***Expr1***,** *Var***,** *High*+*1***,** *Low*−**1)** om *High* < *Low*−*1*

$$\sum_{k=4}^{3}(k) \qquad 0$$

Summaformlerna som används har härletts från följande referens:

Ronald L. Graham, Donald E. Knuth och 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*]**,** [*roundValue*]**)⇒***värde*

ΣInt(*NPmt1***,***NPmt2***,***amortTable***)⇒***värde*

ΣInt(1,3,12,4.75,20000,,12,12) -213.48

ΣInt() Katalog >

Amorteringsfunktion som beräknar räntesumman under ett specificerat område av betalningar.

NPmt1 och *NPmt2* definierar start och slut på betalningsområdet.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 196.

- Om du utelämnar *Pmt* används förinställningen *Pmt*=**tvmPmt (***N,I,PV,FV,PpY,CpY,PmtAt***)**.
- Om du utelämnar *FV* används förinställningen *FV*=0.
- Förinställningarna av *PpY*, *CpY* och *PmtAt* är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

ΣInt(*NPmt1,NPmt2,amortTable***)** beräknar räntesumman baserat på amorteringstabellen *amortTable*. Argumentet *amortTable* måste vara en matris i den form som beskrivs under **amortTbl()**, på sidan 12.

Obs: Se även ΣPrn() nedan och **Bal()**, på sidan 21.

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

ΣPrn() Katalog >

ΣPrn(*NPmt1*, *NPmt2*, *N*, *I*, *PV*, [*Pmt*], [*FV*], [*PpY*], [*CpY*], [*PmtAt*], [*roundValue*]**)**⇒*värde*

ΣPrn(*NPmt1,NPmt2,amortTable***)**⇒*värde*

Amorteringsfunktion som beräknar kapitalsumman under ett specificerat område av betalningar.

NPmt1 och *NPmt2* definierar start och slut på betalningsområdet.

ΣPrn(1,3,12,4.75,20000,,12,12) -4916.28

ΣPrn() Katalog > 📖

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* och *PmtAt* beskrivs i tabellen över TVM-argument, se på sidan 196.

- Om du utelämnar *Pmt* används förinställningen *Pmt*=**tvmPmt (***N,I,PV,FV,PpY,CpY,PmtAt***)**.
- Om du utelämnar *FV* används förinställningen *FV*=0.
- Förinställningarna av *PpY*, *CpY* och *PmtAt* är desamma som för TVM-funktionerna.

roundValue anger antalet decimaler för avrundning. Förinställning: 2.

ΣPrn(*NPmt1,NPmt2,amortTable***)** beräknar summan av betalt kapital baserat på amorteringstabellen *amortTable*. Argumentet *amortTable* måste vara en matris i den form som beskrivs under **amortTbl()**, på sidan 12.

Obs: Se även ΣInt() ovan och **Bal()**, på sidan 21.

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

(indirection) ctrl 📖 tangenter

*varNameString*

Avser variabeln vars namn är *varNameString*. Detta låter dig använda strängar för att skapa variabelnamn inom en funktion.

#("x"&"y"&"z") *xyz*

Skapar eller avser variabeln xyz.

10→*r*	10
"r"→*s1*	"r"
#*s1*	10

Ger värdet på variabeln (r) vars namn är lagrat i variabeln s1.

E (grundpotensform) EE tangent

*mantissa***E***exponent*

Skriver in ett tal i grundpotensform. Talet tolkas som *mantissa* × 10^exponent.

23000.	23000.
2300000000.+4.1E15	4.1E15
$3 \cdot 10^4$	30000

E (grundpotensform)

EE tangent

Tips: Om du vill skriva in en potens av 10 utan att orsaka ett decimalt resultat, använd 10^*integer*.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **@E**. Skriv exempelvis **2.3@E4** för att mata in 2.3E4.

g (nygrad)

π▸ tangent

*Expr1*g⇒*uttryck*

*List1*g⇒*lista*

*Matrix1*g⇒*matris*

Denna funktion ger dig ett sätt att specificera en vinkel i nygrader när du är i läge Grader eller Radianer.

Multiplicerar i vinkelläget Radianer *Expr1* med π/200.

Multiplicerar i vinkelläget Grader *Expr1* med g/100.

Ger i läget Nygrader *Expr1* oförändrat.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva **@g**.

I vinkelläge Grader, Nygrader eller Radianer:

$$\cos(50^g) \qquad \frac{\sqrt{2}}{2}$$

$$\cos(\{0,100^g,200^g\}) \qquad \{1,0,-1\}$$

r (radian)

π▸ tangent

*Expr1*r ⇒*uttryck*

*List1*r ⇒*lista*

*Matrix1*r ⇒*matris*

Denna funktion ger dig ett sätt att specificera en vinkel i radianer när du är i läge Grader eller Nygrader.

Multiplicerar i vinkelläget Grader argumentet med 180/π.

I vinkelläge Grader, Nygrader eller Radianer:

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

$$\cos\left(\left\{0^r,\frac{\pi}{12}^r,-(\pi)^r\right\}\right) \qquad \left\{1,\frac{(\sqrt{3}+1)\cdot\sqrt{2}}{4},-1\right\}$$

ʳ (radian)

π tangent

Ger i vinkelläget Radianer argumentet oförändrat.

Multiplicerar i vinkelläget Nygrader argumentet med 200/π.

Tips: Använd ʳ om du vill framtvinga radianer i en funktionsdefinition oavsett det inställda läget när funktionen används.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva **@r**.

° (grader)

π tangent

Expr1°⇒*uttryck*

List1°⇒*lista*

Matrix1°⇒*matris*

Denna funktion ger dig ett sätt att specificera en vinkel i grader när du är i läge Nygrader eller Radianer.

Multiplicerar i vinkelläget Radianer argumentet med π/180.

Ger i vinkelläget Grader argumentet oförändrat.

Multiplicerar i vinkelläget Nygrader argumentet med 10/9.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva **@d**.

I vinkelläge Grader, Nygrader eller Radianer:

$\cos(45°)$ → $\frac{\sqrt{2}}{2}$

I vinkelläget Radianer:

Obs: För att få ett närmevärde,

Handenhet: Tryck på ctrl enter.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘+**Enter**.
iPad®: Håll ned **enter** och välj ≈.

$\cos\left\{\left\{0,\frac{\pi}{4},90°,30.12°\right\}\right\}$ → $\{1.,0.707107,0.,0.864976\}$

°, ', " (grad/minut/sekund)

ctrl 🕮 tangenter

dd°*mm*'*ss.ss*"⇒*uttryck*

*dd*Ett positivt eller negativt tal

*mm*Ett icke negativt tal

*ss.ss*Ett icke negativt tal

Ger *dd*+(*mm*/60)+(*ss.ss*/3600).

I vinkelläget Grader:

25°13'17.5"	25.2215
25°30'	$\frac{51}{2}$

°, ', " (grad/minut/sekund)

ctrl 🕮 tangenter

Med detta bas-60 inmatningsformat kan du:

- Skriva in en vinkel i grader/minuter/sekunder oavsett det inställda vinkelläget.
- Skriva in tid som timmar/minuter/sekunder.

Obs: Avsluta ss.ss med två apostrofer (''), inte med citationstecken (").

∠ (vinkel)

ctrl 🕮 tangenter

[*Radius*,∠θ_*Angle*]⇒*vektor*

(polär indata)

[*Radius*,∠θ_*Angle*,*Z_Coordinate*]⇒*vektor*

(cylindrisk indata)

[*Radius*,∠θ_*Angle*,∠θ_*Angle*]⇒*vektor*

(sfärisk indata)

Ger koordinater som en vektor beroende på det inställda vektorformatläget: rektangulär, cylindrisk eller sfärisk.

Obs: Du kan infoga denna symbol med datorns tangentbord genom att skriva **@<**.

I vinkelläget Radianer och med vektorformatet inställt på:

rektangulär

$$[5 \angle 60° \angle 45°] \quad \left[\frac{5\cdot\sqrt{2}}{4} \quad \frac{5\cdot\sqrt{6}}{4} \quad \frac{5\cdot\sqrt{2}}{2}\right]$$

cylindrisk

$$[5 \angle 60° \angle 45°] \quad \left[\frac{5\cdot\sqrt{2}}{2} \angle \frac{\pi}{3} \quad \frac{5\cdot\sqrt{2}}{2}\right]$$

sfärisk

$$[5 \angle 60° \angle 45°] \quad \left[5 \angle \frac{\pi}{3} \angle \frac{\pi}{4}\right]$$

(*Magnitude* ∠ *Angle*)⇒*complexValue*

(polär indata)

Matar in ett komplext värde i (r∠θ) polär form. *Angle* tolkas enligt det inställda vinkelläget.

I vinkelläget Radianer och i Rektangulärt komplext format:

$$5+3\cdot i-\left(10 \angle \frac{\pi}{4}\right) \quad 5-5\cdot\sqrt{2}+\left(3-5\cdot\sqrt{2}\right)\cdot i$$

Obs: För att få ett närmevärde,

∠ (vinkel)

ctrl **tangenter**

Handenhet: Tryck på **ctrl** **enter**.
Windows®: Tryck på **Ctrl+Enter**.
Macintosh®: Tryck på ⌘**+Enter**.
iPad®: Håll ned **enter** och välj ≈.

$$5+3\cdot i-\left(10\angle\frac{\pi}{4}\right) \qquad -2.07107-4.07107\cdot i$$

' (prim)

?! **tangent**

variable '

variable ''

Skriver in en primsymbol i en differentialekvation. En enda primsymbol betecknar en differentialekvation av första ordningen, två primsymboler betecknar en ekvation av andra ordningen, och så vidare.

$$\text{deSolve}\left(y''=y^{\frac{-1}{2}} \text{ and } y(0)=0 \text{ and } y'(0)=0, t, y\right) \qquad \frac{2\cdot y^{\frac{3}{4}}}{3}=t$$

_ (understrykningstecken som ett tomt element)

Se "Tomma element" (på sidan 238).

_ (understrykningstecken som enhetsbenämnare)

ctrl **␣** **tangenter**

*Expr*_*Unit*

Anger enheterna för ett *Expr*. Alla enhetsnamn måste börja med en understrykning.

Du kan använda fördefinierade enheter eller skapa dina egna enheter. För en lista på fördefinierade enheter, öppna Katalogen och ta fram fliken Omvandling av enheter. Du kan välja enhetsnamn från Katalogen eller skriva in enhetsnamnen direkt.

$$3\cdot\_\text{m}\blacktriangleright\_\text{ft} \qquad 9.84252\cdot\_\text{ft}$$

Obs: Du finner konverteringssymbolen ▶ i Katalogen. Klicka på **∫Σ** och sedan på **Math Operators**.

*Variable*_

När *Variable* inte har något värde behandlas den som om den representerar ett komplext tal. Som förinställning behandlas variabeln som reell utan _ .

Förutsatt att z är odefinierad:

_ (understrykningstecken som enhetsbenämnare)

ctrl ␣ tangenter

Om *Variable* har ett värde ignoreras _ och *Variable* behåller dess ursprungliga datatyp.

Obs: Du kan lagra ett komplext tal i en variabel utan att använda _ . För bästa resultat i beräkningar såsom **cSolve()** och **cZeros()** rekommenderas dock _ .

$\text{real}(z)$	z
$\text{real}(z\_)$	$\text{real}(z\_)$
$\text{imag}(z)$	0
$\text{imag}(z\_)$	$\text{imag}(z\_)$

▶ (konvertera)

ctrl 🕮 tangenter

Expr_Unit1 ▶ *_Unit2⇒Expr_Unit2*

Omvandlar ett uttryck från en enhet till en annan.

Understrykningstecknet _ betecknar enheterna. Enheterna måste tillhöra samma kategori, till exempel, Längd eller Area.

För en lista på fördefinierade enheter, öppna Katalogen och ta fram fliken Omvandling av enheter:

- Du kan välja ett enhetsnamn på listan.
- Du kan välja omvandlingsoperator, ▶, längst upp i listan.

Du kan också skriva in enhetsnamnen manuellt. För att skriva "_" när du skriver enhetsnamn på handenheten, tryck på ctrl ␣.

Obs: För att konvertera temperaturenheter, använd **tmpCnv()** och **ΔtmpCnv()**. Konverteringsoperatorn ▶ hanterar inte temperaturenheter.

$3\cdot\_\text{m}\blacktriangleright\_\text{ft}$	$9.84252\cdot\_\text{ft}$

10^()

Katalog > 🕮

10^ (*Expr1***)⇒***uttryck*

10^ (*List1***)⇒***lista*

Ger 10 upphöjd till argumentets potens.

$10^{1.5}$	31.6228
$10^{\{0,-2,2,a\}}$	$\left\{1,\frac{1}{100},100,10^{a}\right\}$

10^()

Katalog >

Ger, för en lista, 10 upphöjd till potensen för elementen i *List1*.

10^(*squareMatrix1***)⇒***kvadratMatris*

Ger 10 upphöjd till potensen för *squareMatrix1*. Detta är inte detsamma som att beräkna 10 upphöjd till potensen för varje element. Se **cos()** för information om beräkningsmetoden.

squareMatrix1 måste vara möjlig att diagonalisera. Resultatet visas alltid i flyttalsform.

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

$$\begin{bmatrix}1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6\\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6\\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6\end{bmatrix}$$

^-1(inverterat värde)

Katalog >

Expr1 **^-1⇒***uttryck*

List1 **^-1⇒***lista*

Ger argumentets inverterade värde.

Ger, för en lista, de inverterade värdena på elementen i *List1*.

squareMatrix1 **^-1⇒***kvadratMatris*

Ger inversen av *squareMatrix1*.

squareMatrix1 måste vara en icke singulär kvadratisk matris.

$$(3.1)^{-1} \qquad 0.322581$$

$$\{a,4,-0.1,x,-2\}^{-1} \qquad \left\{\frac{1}{a},\frac{1}{4},-10.,\frac{1}{x},\frac{-1}{2}\right\}$$

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ a & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}\frac{-2}{a-2} & \frac{1}{a-2}\\ \frac{a}{2\cdot(a-2)} & \frac{-1}{2\cdot(a-2)}\end{bmatrix}$$

| (operatorn begränsning)

ctrl tangenter

Uttr **|** *BooleskUttr1* [**and***BooleskUttr2*]...

Uttr **|** *BooleskUttr1* [**or***BooleskUttr2*]...

$$x+1|x=3 \qquad 4$$

$$x+y|x=\sin(y) \qquad \sin(y)+y$$

$$x+y|\sin(y)=x \qquad x+y$$

| (operatorn begränsning)

ctrl 📖 tangenter

("|")-symbolen begränsning fungerar som en binär operator. Operanden till vänster om | är ett uttryck. Operanden till höger om | specificerar ett eller flera relationer som är avsedda att påverka förenklingen av uttrycket. Flera relationer efter | måste förbindas med ett logiskt "**and**" eller "**or**"-operatorer.

Operatorn begränsning ger tre bastyper av funktionalitet:

- Substitutioner
- Intervallbegränsningar
- Uteslutningar

Substitutioner är i form av en likhet såsom x=3 eller y=sin(x). För bästa effektivitet bör den vänstra sidan vara en enkel variabel. *Uttr* | *Variabel* = *värde* ersätter *värde* vid varje förekomst av *Variabel* i *Uttr*.

$x^3-2\cdot x+7\rightarrow f(x)$ *Done*

$f(x)\,|\,x=\sqrt{3}$ $\sqrt{3}+7$

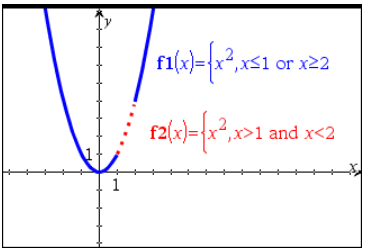
$(\sin(x))^2+2\cdot\sin(x)-6\,|\,\sin(x)=d$ $d^2+2\cdot d-6$

Intervallbegränsningar tar formen av en eller flera olikheter som förbinds med logiska "**and**" eller "**or**"-operatorer. Intervallbegränsningar medger också förenklingar som annars kan vara ogiltiga eller ej beräkningsbara.

$\text{solve}(x^2-1=0,x)\,|\,x>0 \text{ and } x<2$ $x=1$

$\sqrt{x}\cdot\sqrt{\frac{1}{x}}\,|\,x>0$ 1

$\sqrt{x}\cdot\sqrt{\frac{1}{x}}$ $\sqrt{\frac{1}{x}}\cdot\sqrt{x}$



Uteslutningar använder jämförelseoperatorn "inte lika med" (/= eller ≠) för att utesluta ett specifikt värde från övervägning. De används främst för att utesluta en exakt lösning när t.ex. **cSolve()**, **cZeros()**, **fMax()**, **fMin()**, **solve()** **zeros()** osv. används.

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

→ (lagra)

ctrl var tangent

Expr → *Var*

List → *Var*

Matrix → *Var*

Expr → *Function*(*Param1*,...)

List → *Function*(*Param1*,...)

Matrix → *Function*(*Param1*,...)

Om variabeln *Var* inte finns så skapas den och initialiseras till *Expr*, *List* eller *Matrix*.

Om variabeln *Var* redan finns, och inte är låst eller skyddad, ersätts dess innehåll med *Expr*, *List* eller *Matrix*.

Tips: Om du tänker göra symboliska beräkningar med odefinierade variabler, undvik att lagra någonting i enbokstaviga variabler som ofta används, till exempel, a, b, c, x, y, z, och så vidare.

Obs: Du kan infoga denna operator med datorns tangentbord genom att skriva **=:** som kortkommando. Skriv exempelvis **`pi/4 =: minvar`**.

$\frac{\pi}{4} \rightarrow myvar$ $\quad \frac{\pi}{4}$

$2 \cdot \cos(x) \rightarrow y1(x)$ $\quad$ *Done*

$\{1,2,3,4\} \rightarrow lst5$ $\quad \{1,2,3,4\}$

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

"Hello" → *str1* $\quad$ "Hello"

:= (tilldela)

ctrl tangenter

Var := *Expr*

Var := *List*

Var := *Matrix*

Function(*Param1*,...) := *Expr*

Function(*Param1*,...) := *List*

Function(*Param1*,...) := *Matrix*

Om variabeln *Var* inte finns så skapas *Var* och initialiseras till *Expr*, *List* eller *Matrix*.

Om *Var* redan finns, och inte är låst eller skyddad, ersätts dess innehåll med *Expr*, *List* eller *Matrix*.

$myvar := \frac{\pi}{4}$ $\quad \frac{\pi}{4}$

$y1(x) := 2 \cdot \cos(x)$ $\quad$ *Done*

$lst5 := \{1,2,3,4\}$ $\quad \{1,2,3,4\}$

$matg := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ $\quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

str1 := "Hello" $\quad$ "Hello"

:= (tilldela)

ctrl tangenter

Tips: Om du tänker göra symboliska beräkningar med odefinierade variabler, undvik att lagra någonting i enbokstaviga variabler som ofta används, till exempel, a, b, c, x, y, z, och så vidare.

© (kommentar)

ctrl tangenter

© [*text*]

© hanterar *text* som en kommentarsrad så att du kan kommentera funktioner och program som du skapar.

© kan vara i början eller var som helst på raden. Allting till höger om **©**, till slutet av raden, utgör kommentaren.

Obs för att mata in exemplet: Se avsnittet Räknare i produkthandboken för instruktioner om hur du anger multiline-program och funktionsdefinitionen.

```
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 tangenter, 0 H tangenter

0b *binaryNumber*

0h *hexadecimalNumber*

Betecknar ett binärt respektive ett hexadecimalt tal. För att skriva in ett binärt eller hexadecimalt tal måste du använda prefixet 0b eller 0h oavsett det inställda basläget. Utan prefix behandlas ett tal som ett decimalt tal (bas 10).

Resultaten visas enligt det inställda basläget.

I decimalt basläge:

```
0b10+0hF+10                          27
```

I binärt basläge:

```
0b10+0hF+10                     0b11011
```

I hexadecimalt basläge:

```
0b10+0hF+10                        0h1B
```

Tomma element

Vid analys av verkliga data kanske du inte alltid har en komplett datauppsättning. TI-Nspire™ CAS tillåter tomma dataelement så att du kan fortsätta med de nästan kompletta uppgifterna i stället för att behöva börja om från början eller kassera ofullständiga fall.

Du finner ett exempel på data som inbegriper tomma element i kapitlet Listor och kalkylblad under "*Plotta kalkylbladsdata*."

Med funktionen delVoid() kan du ta bort tomma element från en lista. Med funktionen isVoid() kan du testa förekomst av tomma element. För mer information, se delVoid(), på sidan 54 och isVoid(), på sidan 95.

Obs: För att manuellt mata in ett tomt element i ett matematiskt uttryck, skriv in "_" eller nyckelordet **tomrum**. Nyckelordet **tomrum** omvandlas automatiskt till symbolen "_" när uttrycket utvärderas. För att skriva in "_" på handenheten, tryck på ctrl ␣.

Beräkningar som innehåller tomma element

Flertalet beräkningar som inbegriper en tom inmatning producerar ett tomt resultat. Se specialfall nedan.

$|\_|$ → _

$\gcd(100,\_)$ → _

$3+\_$ → _

$\{5,\_,10\}-\{3,6,9\}$ → $\{2,\_,1\}$

Lista argument som innehåller tomma element

Följande funktioner och kommandon ignorerar (hoppar över) tomma element som påträffas i listargument:

count, **countIf**, **cumulativeSum**, **freqTable▸lista**, **frekvens**, **max**, **medelvärde**, **median**, **produkt**, **stDevPop**, **stDevSamp**, **summa**, **sumIf**, **varPop** och **varSamp** samt regressionsberäkningar, **OneVar**, **TwoVar** och **FiveNumSummary** statistik, konfidensintervaller och stat-tester

$\text{sum}(\{2,\_,3,5,6.6\})$ → 16.6

$\text{median}(\{1,2,\_,\_,\_,3\})$ → 2

$\text{cumulativeSum}(\{1,2,\_,4,5\})$ → $\{1,3,\_,7,12\}$

$\text{cumulativeSum}\begin{bmatrix}1 & 2\\ 3 & \_\\ 5 & 6\end{bmatrix}$ → $\begin{bmatrix}1 & 2\\ 4 & \_\\ 9 & 8\end{bmatrix}$

SortA och **SortD** flyttar alla tomma element inom det första argumentet till botten.

$\{5,4,3,\_,1\} \rightarrow list1$ → $\{5,4,3,\_,1\}$

$\{5,4,3,2,1\} \rightarrow list2$ → $\{5,4,3,2,1\}$

SortA *list1*,*list2* → *Done*

list1 → $\{1,3,4,5,\_\}$

list2 → $\{1,3,4,5,2\}$

Lista argument som innehåller tomma element

$\{1,2,3,\_,5\} \rightarrow list1$	$\{1,2,3,\_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD *list1*,*list2*	*Done*
list1	$\{5,3,2,1,\_\}$
list2	$\{5,3,2,1,4\}$

I regressionsberäkningar introducerar ett tomrum i en X- eller Y-lista ett tomrum i motsvarande element för residualen.

$l1:=\{1,2,3,4,5\}$: $l2:=\{2,\_,3,5,6.6\}$	$\{2,\_,3,5,6.6\}$
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	$\{1.,\_,1.,1.,1.\}$

En utelämnad kategori i regressionsberäkningar introducerar ett tomrum i motsvarande element för residualen.

$l1:=\{1,3,4,5\}$: $l2:=\{2,3,5,6.6\}$	$\{2,3,5,6.6\}$
$cat:=\{"M","M","F","F"\}$: $incl:=\{"F"\}$	$\{"F"\}$
LinRegMx *l1*,*l2*,1,*cat*,*incl*	*Done*
stat.Resid	$\{\_,\_,0.,0.\}$
stat.XReg	$\{\_,\_,4.,5.\}$
stat.YReg	$\{\_,\_,5.,6.6\}$
stat.FreqReg	$\{\_,\_,1.,1.\}$

En frekvens på 0 i regressionsberäkningar introducerar ett tomrum i motsvarande element för residualen.

$l1:=\{1,3,4,5\}$: $l2:=\{2,3,5,6.6\}$	$\{2,3,5,6.6\}$
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	$\{1.,\_,1.,1.\}$

Kortkommandon för att mata in matematiska uttryck

Med kortkommandon kan du mata in element i matematiska uttryck genom att skriva i stället för att använda Katalogen eller Symbolpaletten. För att exempelvis mata in uttrycket $\sqrt{6}$ kan du skriva **sqrt(6)** på inmatningsraden. När du trycker på enter ändras uttrycket **sqrt(6)** till $\sqrt{6}$. Vissa kortkommandon kan användas från både handenheten och datorns tangentbord. Andra är i första hand användbara från datorns tangentbord.

Från handenheten eller datorns tangentbord

För att mata in detta:	Skriv detta kortkommando:
π	pi
θ	theta
∞	**infinity**
$\leq$	**<=**
$\geq$	**>=**
$\neq$	**/=**
$\Rightarrow$ (logisk implikation)	=>
$\Leftrightarrow$ (logisk dubbel implikation, XNOR)	<=>
$\rightarrow$ (lagra operator)	**=:**
| | (absolutbelopp)	**abs(...)**
$\sqrt{}$**()**	**sqrt(...)**
$d()$	**derivative(...)**
$\int()$	**integral(...)**
$\Sigma()$ (mallen Summa)	**sumSeq(...)**
$\Pi()$ (mallen Produkt)	**prodSeq(...)**
sin^{-1}(), cos^{-1}(), ...	**arcsin(...)**, **arccos(...)**, ...
Δ**List()**	**deltaList(...)**
Δ**tmpCnv()**	**deltaTmpCnv(...)**

Från datorns tangentbord

För att mata in detta:	Skriv detta kortkommando:
c1, c2, ... (konstanter)	**@c1**, **@c2**, ...
n1, n2, ...	**@n1**, **@n2**, ...

För att mata in detta:	Skriv detta kortkommando:
(heltalskonstanter)	
i (imaginära enheten)	**@i**
e (naturlig logaritmbas e)	**@e**
E (grundpotensform)	**@E**
T (transponera)	**@t**
r (radianer)	**@r**
° (grader)	**@d**
g (nygrader)	**@g**
∠ (vinkel)	**@<**
▶ (omvandling)	**@>**
▶Decimal, **▶approxFraction()**, och så vidare.	**@>Decimal**, **@>approxFraction()**, och så vidare.

EOS™-hierarki (Equation Operating System)

Detta avsnitt beskriver det ekvationsoperativsystem (EOS™) som används av TI-Nspire™ CAS Inlärningsteknologi för matematik och naturvetenskap. Tal, variabler och funktioner matas in i en enkel och direkt sekvens. Programvaran EOS™ beräknar uttryck och ekvationer genom parentesgruppering och enligt de prioriteringsregler som beskrivs nedan.

Ordning vid utvärdering

Nivå	Operator
1	Parenteser ( ), hakparenteser [ ], klammerparenteser { }
2	Indirection (#)
3	Funktionsanrop
4	Postoperatorer: grader-minuter-sekunder (°,',"), fakultet (!), procent (%), radian (^{r}), index ([ ]), transponering (^{T})
5	Exponentberäkning, potensoperator (^)
6	Negation ($^-$)
7	Strängsammanlänkning (&)
8	Multiplikation (•), division (/)
9	Addition (+), subtraktion (-)
10	Likhetsförhållanden: lika med (=), inte lika med (≠ eller /=), mindre än (<), mindre än eller lika med (≤ eller <=), större än (>), större än eller lika med (≥ eller >=)
11	Logiskt **not**
12	Logiskt **and**
13	Logisk **or**
14	**xor**, **nor**, **nand**
15	Logisk implikation (⇒)
16	Logisk dubbel implikation, XNOR⇔
17	(“|”)-operatorn begränsning
18	Spara (→)

Parenteser, hakparenteser och klammerparenteser

Alla beräkningar inom ett par av parenteser, hakparenteser eller klammerparenteser utvärderas först. I exempelvis uttrycket 4(1+2) beräknar EOS™ först den del av uttrycket som är inom parentesen, 1+2, och multiplicerar sedan resultatet, 3, med 4.

Antalet inledande respektive avslutande parenteser, hakparenteser och klammerparenteser måste vara lika inom ett uttryck eller en ekvation. Annars visas ett felmeddelande som anger det element som saknas. Till exempel visar (1+2)/(3+4 felmeddelandet "Missing)".

Obs: Eftersom TI-Nspire™ CAS programvara ger dig möjlighet att definiera dina egna funktioner betraktas ett variabelnamn, följt av ett uttryck inom parentes, som ett "funktionsanrop" i stället för en indirekt multiplikation. Till exempel är a(b+c) funktionen a utvärderad med b+c. För att multiplicera uttrycket b+c med variabeln a, använd explicit multiplikation: a*(b+c).

Indirection

Den indirekta operatorn (#) konverterar en sträng till ett variabel- eller funktionsnamn. Till exempel skapar #("x"&"y"&"z") variabelnamnet xyz. "Indirection" ger också möjlighet att skapa och modifiera variabler inne i ett program. Om exempelvis 10→r och "r"→s1 erhålls #s1=10.

Postoperatorer

Postoperatorer är operatorer som kommer direkt efter ett argument, t.ex. 5!, 25% eller 60°15' 45". Argument som följs av en postoperator utvärderas på den fjärde prioritetsnivån. I exempelvis uttrycket 4^3! utvärderas 3! först. Resultatet, 6, blir sedan exponenten för 4, vilket ger 4096.

Exponentberäkning

Exponentberäkning (^) och exponentberäkning element för element (.^) utvärderas från höger till vänster. Till exempel utvärderas uttrycket 2^3^2 på samma sätt som 2^(3^2). Man får resultatet 512. Detta är inte detsamma som (2^3)^2, vilket ger resultatet 64.

Negation

För att skriva in ett negativt tal, tryck på (−) följt av talet. Postoperationer och exponentberäkningar utförs före negationer. Till exempel är resultatet av −x2 ett negativt tal och $-9^2 = -81$. Använd parenteser för att beräkna kvadraten på ett negativt tal. Till exempel ger $(-9)^2$ resultatet 81.

Begränsning ("|")

Argumentet som följer efter ("|")-operatorn begränsning ger en uppsättning av begränsningar som påverkar utvärderingen av argumentet som föregår operatorn.

Felkoder och meddelanden

När ett fel inträffar placeras dess felkod i variabeln *errCode*. Användardefinierade program och funktioner kan undersöka *errCode* för att bestämma orsaken till ett fel. För ett exempel på användningen av *errCode*, se exempel 2 under kommandot **Try**, på sidan 192.

Obs: Vissa feltillstånd avser endast TI-Nspire™ CAS-produkter medan andra endast avser TI-Nspire™-produkter.

Felkod	Beskrivning
10	A function did not return a value (En funktion gav inget värde)
20	A test did not resolve to TRUE or FALSE. (Ett test gav varken TRUE eller FALSE.) I regel kan odefinierade variabler inte jämföras. Som exempel orsakar testet "Om a<b" detta fel om a eller b är odefinierade när Om-påståendet exekveras.
30	Argument cannot be a folder name. (Argumentet får inte vara ett mappnamn.)
40	Argument error (Argumentfel)
50	Argument mismatch (Fel typ av argument) Två eller flera argument måste vara av samma typ.
60	Argument must be a Boolean expression or integer (Argumentet måste vara ett booleskt uttryck eller ett heltal)
70	Argument must be a decimal number (Argumentet måste vara ett decimaltal)
90	Argument must be a list (Argumentet måste vara en lista)
100	Argument must be a matrix (Argumentet måste vara en matris)
130	Argument must be a string (Argumentet måste vara en sträng)
140	Argument must be a variable name (Argumentet måste vara ett variabelnamn). Kontrollera att namnet: • inte börjar med en siffra • inte innehåller mellanslag eller specialtecken • inte innehåller understrykningstecken eller punkt på ogiltigt sätt • inte överskrider max. längd Se avsnittet Calculator (Räknare) i dokumentationen för mer information.
160	Argument must be an expression (Argumentet måste vara ett uttryck)
165	Batteries too low for sending or receiving (Batterierna är för svaga för sändning eller mottagning) Sätt i nya batterier före sändning eller mottagning.
170	Bound (Gräns)

Felkod	Beskrivning
	Den nedre gränsen måste vara mindre än den övre gränsen för att definiera sökintervallet.
180	Avsluta Tangenten esc eller on trycktes ned under en lång beräkning eller under programexekvering.
190	Circular definition (Cirkulär definition) Detta meddelande visas för att inte minnet skall ta slut under ändlös ersättning av variabelvärden i samband med förenkling. Till exempel orsakar a+1->a, där a är en odefinierad variabel, detta fel.
200	Invalid constraint (Ogiltig begränsning) Som exempel skulle solve(3x^2-4=0,x) | x<0 eller x>5 ge detta felmeddelande eftersom begränsningen är separerad av "eller" i stället för "och".
210	Invalid data type (Ogiltig datatyp) Ett argument är av fel datatyp.
220	Dependent limit (Beroende gräns)
230	Dimension Ett index för en lista eller för en matris är inte giltigt. Om exempelvis listan {1,2,3,4} lagras i L1 är L1[5] ett dimensionsfel eftersom L1 endast innehåller fyra element.
235	Dimension mismatch. (Dimensioner överensstämmer inte.) Inte tillräckligt med element i listorna.
240	Dimension mismatch (Dimensioner överensstämmer inte) Två eller flera argument måste ha samma dimension. Till exempel är [1,2]+[1,2,3] ett dimensionsfel eftersom matriserna innehåller olika antal element.
250	Divide by zero (Division med noll)
260	Domain error (Områdesfel) Ett argument måste vara inom ett specificerat område. Exempelvis är rand(0) inte giltigt.
270	Duplicate variable name (Dubblerade variabelnamn)
280	Else and ElseIf invalid outside of If…EndIf block (Else och ElseIf är ogiltiga utanför If..EndIf-block)
290	EndTry is missing the matching Else statement (EndTry saknas i det matchande Else-påståendet)
295	Excessive iteration (För många iterationer)

Felkod	Beskrivning
300	Expected 2 or 3-element list or matrix (En 2- eller 3-elements lista eller matris förväntades)
310	Första argumentet till nSolve måste vara en ekvation i en variabel. Det får inte innehålla någon variabel utan värde förutom den variabel som används i beräkningen.
320	First argument of solve or cSolve must be an equation or inequality (Första argumentet till solve eller cSolve måste vara en ekvation eller en olikhet) Till exempel är solve(3x^2-4,x) ogiltigt eftersom första argumentet inte är en ekvation.
345	Inconsistent units (Enheterna är oförenliga)
350	Index out of range (Index ligger utanför giltigt område)
360	Indirection string is not a valid variable name (Indirection-strängen är inte ett giltigt variabelnamn)
380	Undefined Ans (Odefinierad Ans) Antingen skapades inte Ans av den föregående beräkningen eller också har ingen tidigare beräkning matats in.
390	Invalid assignment (Ogiltig tilldelning)
400	Invalid assignment value (Ogiltigt tilldelningsvärde)
410	Invalid command (Ogiltigt kommando)
430	Invalid for the current mode settings (Ogiltigt för de aktuella inställningarna)
435	Invalid guess (Ogiltig gissning)
440	Invalid implied multiply (Ogiltig indirekt multiplikation) Till exempel är x(x+1) ogiltig medan x*(x+1) har korrekt syntax. Detta är för att undvika förväxling mellan indirekt multiplikation och funktionsanrop.
450	Invalid in a function or current expression (Ogiltigt i en funktion eller i det aktuella uttrycket) Endast vissa kommandon är giltiga i en användardefinierad funktion.
490	Invalid in Try..EndTry block (Ogiltigt i Try..EndTry-block)
510	Invalid list or matrix (Ogiltig lista eller matris)
550	Invalid outside function or program (Ogiltigt utanför funktion eller program) Ett antal kommandon är inte giltiga utanför en funktion eller ett program. Exempelvis kan Local bara användas inne i en funktion eller ett program.
560	Invalid outside Loop..EndLoop, For..EndFor, or While..EndWhile blocks (Ogiltigt utanför Loop..EndLoop, For..EndFor, eller While..EndWhile-block) Exempelvis kan kommandot Exit bara användas innanför dessa slingor.

Felkod	Beskrivning
565	Invalid outside program (Ogiltig utanför program)
570	Invalid pathname (Ogiltigt sökvägsnamn) Till exempel är \var ogiltigt.
575	Invalid polar complex (Ogiltigt polärt komplext tal)
580	Invalid program reference (Ogiltig programreferens) Man får inte referera till Program inom funktioner eller uttryck såsom 1+p(x) där p är ett program.
600	Invalid table (Ogiltig tabell)
605	Invalid use of units (Ogiltig användning av enheter)
610	Invalid variable name in a Local statement (Ogiltigt variabelnamn i ett Local-påstående)
620	Invalid variable or function name (Ogiltigt variabel- eller funktionsnamn)
630	Invalid variable reference (Ogiltig variabelreferens)
640	Invalid vector syntax (Ogiltig syntax för vektor)
650	Link transmission (Länköverföring) En överföring mellan två enheter slutfördes inte. Kontrollera att anslutningskabeln är ordentligt ansluten i båda ändarna.
665	Matrix not diagonalizable (Matrisen är inte diagonaliserbar)
670	Low Memory (Ont om minne) 1. Ta bort lite data från detta dokument 2. Spara och stäng detta dokument Om dessa steg inte löser problemet, plocka ur batterierna och sätt i dem igen
672	Resource exhaustion (Resursutmattning)
673	Resource exhaustion (Resursutmattning)
680	Missing ((Saknar “ (“)
690	Missing) (Saknar “) “)
700	Missing " (Saknar “ " “)
710	Missing] (Saknar “] “)
720	Missing } (Saknar “ } “)
730	Missing start or end of block syntax (Saknar start eller slut i blockets syntax)

Felkod	Beskrivning
740	Missing Then in the If..EndIf block (Saknar "Then" i If..EndIf-blocket)
750	Name is not a function or program (Namnet är inte en funktion eller ett program)
765	No functions selected (Inga funktioner är valda)
780	No solution found (Ingen lösning funnen)
800	Non-real result (Resultatet är inte ett reellt tal) Om exempelvis programvaran har inställningen Real så är $\sqrt{(-1)}$ ogiltigt. För att medge komplexa resultat, ändra lägesinställningen "Real or Complex" till RECTANGULAR eller POLAR.
830	Overflow (För stort värde)
850	Program not found (Program hittades inte) En programreferens inne i ett annat program kunde ej hittas via angiven sökväg under exekveringen.
855	Rand type functions not allowed in graphing (Funktioner av slumptyp tillåts ej vid grafritning)
860	Recursion too deep (Rekursionen för djup)
870	Reserved name or system variable (Reserverat namn eller systemvariabel)
900	Argument error (Argumentfel) Median-median-modellen kunde inte användas på datauppsättningen.
910	Syntaxfel
920	Text not found (Text hittades inte)
930	Too few arguments (För få argument) Funktionen eller kommandot saknar ett eller flera argument.
940	Too many arguments (För många argument) Uttrycket eller ekvationen innehåller för många argument och kan inte utvärderas.
950	Too many subscripts (För många nedsänkta tecken)
955	Too many undefined variables (För många odefinierade variabler)
960	Variable is not defined (Variabel ej definierad) Inget värde är tillskrivet variabeln. Använd något av följande kommandon: • sto → • := • **Define (Definiera)**

Felkod	Beskrivning
	för att tilldela variabler värden.
965	Unlicensed OS (Olicensierat OS)
970	Variable in use so references or changes are not allowed (Variabeln används varför referenser eller ändringar ej tillåts)
980	Variable is protected (Variabeln är skyddad)
990	Invalid variable name (Ogiltigt variabelnamn) Kontrollera att namnet inte överskrider max. längd.
1000	Window variables domain (Fönstervariabeldomän)
1010	Zoom
1020	Internal error (Internt fel)
1030	Protected memory violation (Minnesskydd)
1040	Unsupported function. (Funktionen stöds ej.) Denna funktion kräver Computer Algebra System. Prova TI-Nspire™ CAS.
1045	Unsupported operator. (Operatorn stöds ej.) Denna operator kräver Computer Algebra System. Prova TI-Nspire™ CAS.
1050	Unsupported feature. (Funktionen stöds ej.) Denna operator kräver Computer Algebra System. Prova TI-Nspire™ CAS.
1060	Input argument must be numeric. (Inmatade argument måste vara numeriska.) Endast inmatningar som innehåller numeriska värden är tillåtna.
1070	Trig function argument too big for accurate reduction (Trig-funktionens argument är för stort för en noggrann reduktion)
1080	Unsupported use of Ans.This application does not support Ans. (Användning av Ans stöds inte. Denna applikation stöder inte Ans.)
1090	Function is not defined. (Funktion ej definierad.) Använd något av följande kommandon: • **Define (Definiera)** • **:=** • sto → för att definiera en funktion.
1100	Non-real calculation (Icke-reell beräkning) Om exempelvis programvaran har inställningen Real så är $\sqrt{(-1)}$ ogiltigt. För att medge komplexa resultat, ändra lägesinställningen “Real or Complex” till RECTANGULAR eller POLAR.
1110	Invalid bounds (Ogiltiga gränser)

Felkod	Beskrivning
1120	No sign change (Ingen teckenändring)
1130	Argument cannot be a list or matrix (Argumentet får inte vara en lista eller en matris)
1140	Argument error (Argumentfel) Det första argumentet måste vara ett polynomuttryck i det andra argumentet. Om det andra argumentet utelämnas försöker programvaran välja ett förinställt argument.
1150	Argument error (Argumentfel) De första två argumenten måste vara polynomuttryck i det tredje argumentet. Om det tredje argumentet utelämnas försöker programvaran välja ett förinställt argument.
1160	Invalid library pathname (Ogiltigt sökvägsnamn för bibliotek) Ett sökvägsnamn måste vara i formen *xxx**yyy*, där: • Delen *xxx* kan ha 1 till 16 tecken. • Delen *yyy* kan ha 1 till 15 tecken. Se avsnittet Bibliotek i dokumentationen för mer information.
1170	Invalid use of library pathname (Ogiltig användning av sökvägsnamn för bibliotek) • Ett sökvägsnamn får inte ges ett värde med **Define**, **:=** eller sto →. • Ett sökvägsnamn får inte anges som en Local-variabel eller användas som en parameter i en funktions- eller programdefinition.
1180	Invalid library variable name. (Ogiltigt namn på biblioteksvariabel.) Kontrollera att namnet: • inte innehåller en punkt • inte börjar med ett understrykningstecken • inte överskrider 15 tecken Se avsnittet Bibliotek i dokumentationen för mer information.
1190	Biblioteksdokument kunde ej hittas: • Kontrollera att biblioteket är i MyLib-mappen. • Uppdatera bibliotek. Se avsnittet Bibliotek i dokumentationen för mer information.
1200	Biblioteksvariabel kunde ej hittas: • Kontrollera att biblioteksvariabeln finns i det första problemet i biblioteket. • Kontrollera att biblioteksvariabeln har definierats som LibPub eller LibPriv. • Uppdatera bibliotek. Se avsnittet Bibliotek i dokumentationen för mer information.

Felkod	Beskrivning
1210	Ogiltigt genvägsnamn till bibliotek. Kontrollera att namnet inte: • innehåller en punkt • börjar med ett understrykningstecken • överskrider 16 tecken • är ett reserverat namn Se avsnittet Bibliotek i dokumentationen för mer information.
1220	Områdesfel: Funktionerna tangentLine och normalLine stöder endast funktioner med reella värden.
1230	Områdesfel. Trigonometriska omvandlingsoperatorer stöds inte i vinkellägena Grader och Nygrader.
1250	Argumentfel Använd ett linjärt ekvationssystem. Exempel på ett linjärt ekvationssystem med variablerna x och y: 3x+7y=5 2y-5x=-1
1260	Argumentfel: Det första argumentet till nfMin eller nfMax måste vara ett uttryck i en variabel. Det får inte innehålla någon variabel utan värde förutom den variabel som används i beräkningen.
1270	Argumentfel Derivatans ordning måste vara lika med 1 eller 2.
1280	Argumentfel Använd ett polynom i expanderad form i en variabel.
1290	Argumentfel Använd ett polynom i en variabel.
1300	Argumentfel Koefficienterna i polynomet måste beräknas till numeriska värden.
1310	Argumentfel: En funktion kunde inte utvärderas för ett eller flera av dess argument.
1380	Argumentfel:

Felkod	Beskrivning
	Nästlade anrop till funktionen domän() är inte tillåtna.

Varningskoder och meddelanden

Du kan använda funktionen **warnCodes()** för att lagra de varningskoder som genereras genom att utvärdera ett uttryck. Denna tabell listar varje numerisk varningskod och tillhörande meddelande.

För ett exempel på lagring av varningskoder, se **warnCodes()**, på sidan 201.

Varningskod	Meddelande
10000	Operationen kan ge falska lösningar.
10001	Derivering av en ekvation kan ge en falsk ekvation.
10002	Tvivelaktig lösning
10003	Tvivelaktig noggrannhet
10004	Operationen kan förlora lösningar.
10005	cSolve kan ange fler nollställen.
10006	Solve kan ange fler nollställen.
10007	Flera lösningar kan finnas. Försök att ange lämpliga nedre och övre gränser och/eller en gissning. Exempel som använder solve(): • solve(Ekvation, Var=Gissning)|undrGräns<Var<övrGräns • solve(Ekvation, Var)|undrGräns<Var<övrGräns • solve(Ekvation, Var=Gissning)
10008	Resultatets definitionsområde kan vara mindre än inmatningens definitionsområde.
10009	Resultatets definitionsområde kan vara större än inmatningens definitionsområde.
10012	Icke-reell beräkning
10013	∞^0 eller undef^0 ersattes med 1
10014	undef^0 ersattes med 1
10015	1^∞ eller 1^undef ersattes med 1
10016	1^undef ersattes med 1
10017	För stort värde ersattes med ∞ eller ⁻∞
10018	Operationen kräver och ger ett 64-bitars värde.
10019	Resursutmattning, förenklingen kan vara ofullständig.
10020	Trig-funktionens argument är för stort för en noggrann reduktion.
10021	Inmatningen innehåller en odefinierad parameter. Resultatet kanske inte är giltigt för samtliga möjliga parametervärden.

Varningskod	Meddelande
10022	Att ange lämpliga övre och undre gränser kan ge en lösning.
10023	Skalär har multiplicerats med enhetsmatrisen.
10024	Resultat erhållet med ungefärlig aritmetik.
10025	Ekvivalens kan inte verifieras i EXAKT läge.
10026	Begränsning kan ignoreras. Specificera begränsning med formen "\" 'Variable MathTestSymbol Constant' eller en sammansättning av dessa former, till exempel 'x<3 och x>-12'

Service och Support

Service och garanti för TI-produkter

TI-produkter och service	Mer information om TI-produkter och service kan du få via E-post eller genom att besöka TI på deras Internetadress. e-post: ti-cares@ti.com internetadress: education.ti.com
Service och garanti	Information om garantitid och garantivillkor eller om produktservice finns i garantibeviset som medföljer denna produkt. Du kan också kontakta din lokala återförsäljare/distributör för Texas Instruments.

Innehållsförteckning

'

', minutnotation 230
', prim 232

–

–, subtrahera[*] 212

!

!, fakultet 222

"

", sekundnotation 230

#

#, indirection 228
#, indirection, operator 243

%

%, procent 218

&

&, lägg till 222

*

*, multiplicera 213

.

.-, punkt subtraktion 216
.*, punkt multiplikation 216
./, punkt division 217
.^, punkt potens 217
.+, punkt addition 216

:

:=, tilldela 236

^

^⁻¹, inverterat värde 234
^, potens 214

_

_, enhetsbeteckning 232

|

|, operatorn begränsning 234

+

+, addera 212

/

/, dividera[*] 214

=

≠, inte lika med[*] 219
=, lika med 218

>

>, större än 220

∏

∏, produkt[*] 225

∑

∑( ), summa[*] 226
∑Int( ) 226
∑Prn( ) 227

√

√, kvadratrot[*] 225

∫

∫, integrera[*] 223

≤

≤, mindre än eller lika med 219

≥

≥, större än eller lika med 220

▸

▸, konvertera enheter[*] 233
▸, konvertera till nygradvinkel[Grad] 87
▸approxFraction( ) 18
▸Base10, visa som decimalt heltal[Bas 10] 23
▸Base16, visa som hexadecimal[Bas 16] 23
▸Base2, visa som binär[Bas 2] 21
▸cos, visning i termer av cosinus[cos] 34
▸Cylind, visa som cylindrisk vektor[Cylind] 47
▸DD, visa som decimal vinkel[DD] 50
▸Decimal, visa resultat som decimal[Decimal] 51
▸DMS, visa som grad/minut/sekund[DMS] 58
▸exp, visning i termer av e[exp] 67
▸Polar, visa som polär vektor[Polär] 134
▸Rad, konvertera till radianvinkel[Rad] 144
▸Rect, visa som rektangulär vektor[Rect] 147
▸sin, visning i termer av sinus[sin] 167
▸Sphere, visa som sfärisk vektor[sfär] 176

→

→, lagra 236

⇒

⇒, logisk implikation[*] 221, 240

⇔

⇔, logisk dubbel implikation[*] 221

©

©, kommentar 237

°

°, grader/minuter/sekunder[*] 230
°, gradnotation[*] 230

0

0b, binär indikator 237
0h, hexadecimal indikator 237

1

10^( ), tiopotens 233

2

2-sampel F Test 79

A

abs( ), absolutbelopp 12
absolutbelopp
mall 7-8
addera, + ... 212
amorteringstabell, amortTbl( ) 12, 21
amortTbl( ), amorteringstabell 12, 21
and, Booleska operatorer 13
andraderivata
mall för 10
angle( ), vinkel 14
annars, Else 88
ANOVA 2-vägs, 2-vägs variansanalys 15
ANOVA, 1-vägs variansanalys 14
Ans, sista svar 17
antal dagar mellan datum, dbd( ) 50
approx( ), ungefärlig 17
approxRational( ) 18
arccos() 18
arccosh() 18
arccosinus, $\cos^{-1}$( ) 36
arccot() 18
arccoth() 18
arccsc() 18
arccsch() 19
arcLen( ), båglängd 19
arcsec() 19
arcsech() 19
arcsin() 19
arcsinh() 19
arcsinus, $\sin^{-1}$( ) 169

arctan() 19
arctangens, tan^{-1}( ) 184
arctanh() 19
argument i TVM-funktioner 196
augment( ), sammanfoga/slå ihop 20
avgRC( ), genomsnittlig ändringsfrekvens 20
avrundning, round( ) 155
avsluta, Exit 66

B

beräkning, ordning vid 242
bestämd integral
  mall 10
bibliotek
  skapa genvägar till objekt 96
binomCdf( ) 24
binomPdf( ) 24
binär
  indikator, 0b 237
  visa, ▸Base2 21
blandade bråk, använda propFrac(› med 140
Booleska operatorer
  ⇒ 221, 240
  ⇔ 221
  and 13
  eller 130
  icke 126
  negerad 120
  nor 124
  xor 202
bråk
  mall 5
  propFrac 140
båglängd, arcLen( ) 19

C

Cdf( ) 73
ceiling( ), tak 24
centralDiff( ) 25
cFactor( ), komplex faktor 26
char( ), teckensträng 27
charPoly( ) 27
$\chi^2$2way 27

χ²Cdf( ) 28
χ²GOF 28
χ²Pdf( ) 29
ClearAZ 29
ClrErr, rensa fel 29
colAugment 30
colDim( ), matriskolumndimension 30
colNorm( ), matrixskolumnnorm 30
comDenom( ), gemensam nämnare 31
completeSquare( ), complete square 32
conj( ), komplexkonjugat 32
constructMat( ), konstruera matris 33
corrMat( ), korrelationsmatris 34
cos⁻¹, arccosinus 36
cos( ), cosinus 34
cosh⁻¹( ), hyperbolisk arccosinus 37
cosh( ), hyperbolisk cosinus 36
cosine
visning i termer av 34
cosinus, cos( ) 34
cot⁻¹( ), arccotangens 38
cot( ), cotangens 37
cotangens, cot( ) 37
coth⁻¹( ), hyperbolisk arccotangens 39
coth( ), hyperbolisk cotangens 38
count( ), räkna poster i en lista 39
countif( ), villkorligt räkna poster i en lista 39
cPolyRoots() 40
crossP( ), vektorprodukt 41
csc⁻¹( ), invers cosekant 41
csc( ), cosekant 41
csch⁻¹( ), invers hyperbolisk cosekant 42
csch( ), hyperbolisk cosekant 42
cSolve( ), lösa komplext 42
CubicReg, kubisk regression 45
cumulativeSum( ), kumulativ summa 46
Cycle, cykel 47
cykel, Cycle 47
cZeros( ), komplexa nollor 47

D

d( ), förstaderivata 222

days between dates (dagar mellan datum), dbd( ) 50
dbd( ), days between dates (dagar mellan datum) 50
decimal
heltalsvisning, ▸Base10 23
vinkelvisning, ▸DD 50
Define 51
Define LibPriv 52
Define LibPub 53
define, Define 51
Define, define 51
defining
private function or program 52
public function or program 53
dela heltal, intDiv( ) 91
delmatris, subMat( ) 182
deltaList() 53
deltaTmpCnv() 53
DelVar, ta bort variabel 54
delVoid( ), ta bort tomma element 54
derivata
förstaderivata, d( ) 222
numerisk derivata, nDeriv( ) 122-123
numerisk derivata, nDerivative( ) 122
derivata eller n:te derivata
mall 10
derivative() 54
deSolve( ), lösning 54
det( ), matrisdeterminant 56
diag( ), diagonal matris 57
dim( ), dimension 57
dimension, dim( ) 57
Disp, visa data 58, 159
dividera, / 214
dominant term, dominantTerm( ) 59
dominantTerm( ), dominant term 59
dotP( ), skalärprodukt 60

E

e exponent
mall 6
e till en potens, e^( ) 61, 67
E, exponent 228
e, visning av uttryck i termer av 67

e^( ), e till en potens 61
eff), konvertera nominell till effektiv ränta 61
effektiv ränta, eff( ) 61
egentligt bråk, propFrac 140
egenvektor, eigVc( ) 62
egenvärde, eigVl( ) 62
eigVc( ), egenvektor 62
eigVl( ), egenvärde 62
Ekvationsoperativsystem (EOS) 242
ekvationssystem (2 ekvationer)
    mall 7
ekvationssystem (N ekvationer)
    mall 7
eller (boolesk), eller 130
eller, Boolesk operator 130
else if, ElseIf 63
ElseIf, else if 63
end
    for, EndFor 76
    funktion, EndFunc 80
    om, EndIf 88
end function, EndFunc 80
EndTry, slut försök 192
EndWhile, slut medan 202
enheter
    konvertera 233
enhetsvektor, unitV( ) 198
envariabelstatistik, OneVar 129
EOS (Ekvationsoperativsystem) 242
euler( ), Euler function 64
exact( ), exakt 66
exakt, exact( ) 66
Exit, avsluta 66
exp( ), e till en potens 67
exp▸list( ), uttryck till lista 68
expand( ), expandera 68
expandera, expand( ) 68
exponent, E 228
exponenter
    mall 5
exponentiell regession, ExpReg 70
expr( ), sträng till uttryck 70, 108
ExpReg, exponentiell regession 70

F

factor( ), faktor 71
faktor, factor( ) 71
fakultet, ! 222
fel och felsökning
  flytta fel, PassErr 132
  rensa fel, ClrErr 29
Fill, fylla matris 73
finansiella funktioner, tvmFV( ) 195
finansiella funktioner, tvmI( ) 195
finansiella funktioner, tvmN( ) 195
finansiella funktioner, tvmPmt( ) 196
finansiella funktioner, tvmPV( ) 196
FiveNumSummary 74
floor( ), golv 74
flytta fel, PassErr 132
fMax( ), funktionsmaximum 75
fMin( ), funktionsminimum 75
For 76
for, For 76
For, for 76
format( ), formatsträng 77
formatsträng, format( ) 77
fpart( ), funktionsdel 77
freqTable( ) 78
frequency( ) 78
Frobenius norm, norm( ) 125
Func, funktion 80
Func, programfunktion 80
functions
  user-defined 51
funktioner
  del, fpart( ) 77
  maximum, fMax( ) 75
  minimum, fMin( ) 75
  programfunktion, Func 80
funktioner och variabler
  kopiera 33
förstaderivata
  mall för 9
försök, Try 192

G

g, nygrader 229
gcd( ), största gemensamma delare 81
ge, Return 152
gemensam nämnare, comDenom( ) 31
genomsnittlig ändringsfrekvens, avgRC( ) 20
geomCdf( ) 81
geomPdf( ) 81
Get 82
getDenom( ), hämta/ge nämnare 83
getLangInfo( ), hämta/ge språkinformation 83
getLockInfo( ), testar låsstatus hos en variabel eller variabelgrupp 84
getMode( ), hämta lägesinställningar 84
getNum( ), hämta/ge tal 85
GetStr 85
getType( ), get type of variable 86
getVarInfo( ), hämta/ge variabelinformation 86
golv, floor( ) 74
Goto, gå till 87
grader/minuter/sekunder 230
gradnotation, ° 230
grupper, låsa och låsa upp 107, 199
grupper, testa låsstatus 84
gränsvärde
  lim( ) 97
  limit( ) 97
  mall 10
gå till, Goto 87

H

heltal, int( ) 91
heltalsdel, iPart( ) 93
hexadecimal
  indikator, 0h 237
  visa, ▸Base16 23
hyperbolisk
  arccosinus, $\cosh^{-1}$( ) 37
  arcsinus, $\sinh^{-1}$( ) 170
  arctangens, $\tanh^{-1}$( ) 185
  cosinus, cosh( ) 36
  sinus, sinh( ) 169
  tangens, tanh( ) 185

hämta/ge
nämnare, getDenom( ) 83
tal, getNum( ) 85
variabelinformation, getVarInfo( ) 83, 86
höger, right( ) 152

I

icke, Booleska operatorer 126
identitetsmatris, identity( ) 87
identity( ), identitetsmatris 87
If, om 88
ifFn( ) 89
imag( ), imaginärdelen 90
imaginärdelen, imag( ) 90
ImpDif( ), implicit derivata 90
implicit derivata, Impdif( ) 90
indata, Input 90
indirection, # 228
indirection, operator (#) 243
inom sträng, inString( ) 91
Input, indata 90
inString( ), inom sträng 91
inställningar, hämta aktuella 84
int( ), heltal 91
intDiv( ), dela heltal 91
inte lika med, ≠ 219
integrera, ‰ 223
interpolate( ), interpolate 92
invers kumulativ normalfördelning (invNorm( ) 93
invers, ^⁻¹ 234
inverterat värde, ^⁻¹ 234
invF( ) 92
invNorm( ), invers kumulativ normalfördelning 93
invt( ) 93
Invχ²( ) 92
iPart( ), heltalsdel 93
irr( ), internränta
internränta, irr( ) 94
isPrime( ), primtest 94
isVoid( ), testa för tomrum 95

K

kombinationer, nCr( ) 121
Kommandot Wait 200
kommentar, © 237
komplex
faktor, cFactor( ) 26
komplexa
nollor, cZeros( ) 47
komplext
konjugat, conj( ) 32
lösa, cSolve( ) 42
konstant
isolve( ) 173
konstanter
genvägar för 240
icSolve( ) 44
icZeros( ) 49
ideSolve( ) 55
isolve( ) 174
konstruera matris, constructMat( ) 33
konvertera
4Grad 87
4Rad 144
enheter 233
kopiera variabel eller funktion, CopyVar 33
korrelationsmatris, corrMat( ) 34
kortkommandon 240
kortkommandon, tangentbord 240
kubisk regression, CubicReg 45
kumulativ summa, cumulativeSum( ) 46
kvadratisk regression, QuadReg 141
kvadratrot
mall 5
kvadratrot, √( ) 177, 225
kvartär regression, QuartReg 142

L

lagra
symbol, & 236
Lbl, märka 95
lcm, minsta gemensamma multipel 95
left( ), vänster 96
LibPriv 52

LibPub 53
libShortcut( ), skapa genvägar till biblioteksobjekt 96
lika med, = ... 218
limit( ) eller lim( ), gränsvärde 97
linjär regression, LinRegAx 99
linjär regression, LinRegBx 98, 100
LinRegBx, linjär regression 98
LinRegMx, linjär regression 99
LinRegtIntervals, linjär regression 100
LinRegtTest 101
linSolve() 103
Δlist( ), listskillnad 103
list▸mat( ), lista till matris 104
lista till matris, list▸mat( ) 104
lista, räkna poster 39
lista, villkorligt räkna poster 39
listor
kumulativ summa, cumulativeSum( ) 46
lista till matris, list▸mat( ) 104
matris till lista, mat▸list( ) 112
maximum, max( ) 112
med tomma element 238
minimum, min( ) 115
mittsträng, mid( ) 115
ny, newList( ) 122
produkt, product( ) 139
sammanfoga/slå ihop, augment( ) 20
skalärprodukt, dotP( ) 60
skillnad, @list( ) 103
skillnader i en lista, @list( ) 103
sortera fallande, SortD 176
sortera stigande, SortA 175
summering, sum( ) 181-182
uttryck till lista, exp▸list( ) 68
vektorprodukt, crossP( ) 41
ln( ), naturlig logaritm 104
LnReg, logaritmisk regression 105
Local, lokal variabel 106
Log
mall 6
logaritmer 104
logaritmisk regression, LnReg 105
logisk dubbel implikation, ⇔ 221

logisk implikation, ⇒ 221, 240
Logistic, logistisk regression 108
LogisticD, logistisk regression 109
logistisk regression, Logistic 108
logistisk regression, LogisticD 109
lokal variabel, Local 106
lokal, Local 106
Loop, slinga 111
LU, matris undre-övre uppdelning 111
låsa upp variabler eller variabelgrupper 199
låsa variabler eller variabelgrupper 107
Låsa, låsa variabel eller variabelgrupp 107
lägen
    inställning, setMode( ) 162
lägesinställningar, getMode( ) 84
lägg till, & 222
längd på sträng 57
lös, solve( ) 171
lösning, deSolve( ) 54

M

mallar
    absolutbelopp 7-8
    andraderivata 10
    bestämd integral 10
    bråk 5
    derivata eller n:te derivata 10
    e exponent 6
    ekvationssystem (2 ekvationer) 7
    ekvationssystem (N ekvationer) 7
    exponent 5
    förstaderivata 9
    gränsvärde 10
    kvadratrot 5
    Log 6
    matris (1 × 2) 8
    matris (2 × 1) 8
    matris (2 × 2) 8
    matris (m × n) 8
    n:te rot 5
    obestämd integral 10
    produkt (P) 9
    stegvis funktion (2 steg) 6

stegvis funktion (N steg) 7
summa (G) 9
mat▸list( ), matris till lista 112
matris (1 × 2)
mall 8
matris (2 × 1)
mall 8
matris (2 × 2)
mall 8
matris (m × n)
mall 8
matris till lista, mat▸list( ) 112
matriser
delmatris, subMat( ) 182
determinant, det( ) 56
diagonal, diag( ) 57
dimension, dim( ) 57
egenvektor, eigVc( ) 62
egenvärde, eigVl( ) 62
fylla, Fill 73
identitet, identity( ) 87
kolumndimension, colDim( ) 30
kolumnnorm, colNorm( ) 30
kumulativ summa, cumulativeSum( ) 46
lista till matris, list▸mat( ) 104
matris till lista, mat▸list( ) 112
maximum, max( ) 112
minimum, min( ) 115
ny, newMat( ) 122
produkt, product( ) 139
punkt addition, .+ 216
punkt division, .P 217
punkt multiplikation, .* 216
punkt potens, .^ 217
punkt subtration, .N 216
QR-faktorisering, QR 140
radaddition, rowAdd( ) 156
raddimension, rowDim( ) 156
radmultiplikation och addition, mRowAdd( ) 117
radnorm, rowNorm( ) 156
radoperation, mRow( ) 117
radtrappstegsform, ref( ) 148
radväxling, rowSwap( ) 156

reducerad radtrappstegsform, rref( ) 157
sammanfoga/slå ihop, augment( ) 20
slump, randMat( ) 146
summering, sum( ) 181-182
transponera, T 183
undermatris, subMat( ) 181
undre-övre uppdelning, LU 111
max( ), maximum 112
maximum, max( ) 112
mean( ), medelvärde 113
med, ... 234
medan, While 202
medelvärde, mean( ) 113
median( ), median 113
median, median( ) 113
medium-medium-linjeregression, MedMed 114
MedMed, medium-medium-linjeregression 114
mid( ), mittsträng 115
min( ), minimum 115
mindre än eller lika med, { 219
minimum, min( ) 115
minsta gemensamma multipel, lcm 95
minutnotation, ... 230
mirr( ), modifierad internränta 116
mittsträng, mid( ) 115
mod( ), modul 117
modifierad internränta, mirr( ) 116
modul, mod( ) 117
mRow( ), matrisradoperation 117
mRowAdd( ), matrisradmultiplikation och addition 117
Multipelt linjärt regression-t-test 119
multiplicera, * 213
MultReg 117
MultRegIntervals( ) 118
MultRegTests( ) 119
märka, Lbl 95

N

n:te rot
mall 5
nand, Boolesk operator 120
naturlig logaritm, ln( ) 104

nCr( ), kombinationer 121
nDerivative( ), numerisk derivata 122
negation, skriva in negativa tal 243
nettovärde, npv( ) 127
newList( ), ny lista 122
newMat( ), ny matris 122
nfMax( ), numeriskt funktionsmaximum 122
nfMin( ), numeriskt funktionsminimum 123
nInt( ), numerisk integral 123
nollställen, zeroes( ) 203
nom), konvertera effektiv till nominell ränta 124
nominell ränta, nom( ) 124
nor, Booleska operatorer 124
norm( ), Frobenius norm 125
normal, normalLine( ) 125
normalLine( ) 125
normCdf( ) 125
normPdf( ) 126
nPr( ), permutationer 127
npv( ), nettovärde 127
nSolve( ), numerisk lösning 128
numeric
derivata, nDeriv( ) 122
numerisk
derivata, nDeriv( ) 123
derivata, nDerivative( ) 122
integral, nInt( ) 123
lösning, nSolve( ) 128
ny
lista, newList( ) 122
matris, newMat( ) 122
nygradsnotation, g 229
nämnare 31
när, when( ) 201

O

obestämd integral
mall 10
objekt
skapa genvägar till bibliotek 96
område( ), områdesfunktion 59
områdesfunktion, område( ) 59
OneVar, envariabelstatistik 129

operatorer
ordning vid beräkning 242
operatorn begränsning "|" 234
operatorn begränsning, ordning vid utvärdering 242
ord( ), numerisk teckenkod 131
fördelningsfunktioner
binomCdf( ) 24
binomPdf( ) 24
$\chi^2$2way( ) 27
invNorm( ) 93
invt( ) 93
Invχ^2( ) 92
normCdf( ) 125
normPdf( ) 126
χ^2Cdf( ) 28
poissCdf( ) 133
poissPdf( ) 134
tCdf( ) 186
tPdf( ) 191
χ^2GOF( ) 28
χ^2Pdf( ) 29

P

P▸Rx( ), rektangulär x-koordinat 131
P▸Ry( ), rektangulär y-koordinat 132
PassErr, flytta fel 132
Pdf( ) 78
permutationer, nPr( ) 127
piecewise( ) 133
poissCdf( ) 133
poissPdf( ) 134
polyCoef( ) 135
polyDegree( ) 135
polyEval( ), utvärdera polynom 136
polyGcd( ) 136-137
polynom
slump, randPoly( ) 146
utvärdera, polyEval( ) 136
PolyRoots() 137
polär
koordinat, R▸Pr( ) 144
koordinat, R▸Pθ( ) 144
visa vektor, ▸Polar 134

potens, ^ 214
potensregression, PowerReg 137, 149, 151, 188
PowerReg, potensregression 137
Prgm, definiera program 139
prim, ... 232
primtalstest, isPrime( ) 94
procent, % 218
prodSeq() 139
product( ), produkt 139
produkt (P)
    mall 9
produkt, product( ) 139
produkt, Π( ) 225
program och programmering
    försök, Try 192
    rensa fel, ClrErr 29
    slut försök, EndTry 192
    slut program, EndPrgm 139
    visa I/O- skärm, Disp 159
    visa I/O-fönster, Disp 58
programmera
    definiera program, Prgm 139
    flytta fel, PassErr 132
    visa data, Disp 58
programmering
    visa data, Disp 159
programs
    defining private library 52
    defining public library 53
propFrac, egentligt bråk 140
punkt
    addition, .+ 216
    division, .P 217
    multiplikation, .* 216
    potens, .^ 217
    subtraktion, .N 216

Q

QR-faktorisering, QR 140
QR, QR-faktorisering 140
QuadReg, kvadratisk regression 141
QuartReg, kvartär regression 142

R

R, radian 229
R▸Pr( ), polär koordinat 144
R▸Pθ( ), polär koordinat 144
radian, R 229
radtrappstegsform, ref( ) 148
rand( ), random number 145
randBin, random number 145
randInt( ), random integer 145
randMat( ), slumpmatris 146
randNorm( ), slumpnorm 146
randPoly( ), slumppolynom 146
randSamp( ) 146
RandSeed, slumptalsfrö 147
real( ), reell 147
reducerad radtrappstegsform, rref( ) 157
reell, real( ) 147
ref( ), radtrappstegsform 148
regression
  exponentiell, ExpReg 70
  kubisk, CubicReg 45
regressioner
  kvadratisk, QuadReg 141
  kvartär, QuartReg 142
  linjär regression, LinRegAx 99
  linjär regression, LinRegBx 98, 100
  logaritmisk, LnReg 105
  Logistic 108
  logistisk, Logistic 109
  medium-medium-linje, MedMed 114
  MultReg 117
  potensregression, PowerReg 137, 149, 151, 188
  sinus, SinReg 170
rektangulär vektorvisning, ▸Rect 147
rektangulär x-koordinat, P▸Rx( ) 131
rektangulär y-koordinat, P▸Ry( ) 132
remain( ), rest 149
rensa
  fel, ClrErr 29
Request 149
RequestStr 151
rest, remain( ) 149

result
visning i termer av cosinus ... 34
visning i termer av e ... 67
visning i termer av sinus ... 167
resultat, statistik ... 177
resultvärden, statistik ... 178
Return, ge ... 152
right( ), höger ... 152
right, right( ) ... 32, 64, 92, 152, 201
rk23( ), Runge Kutta function ... 152
rotate( ), rotera ... 154
rotera, rotate( ) ... 154
round( ), avrundning ... 155
rowAdd( ), matrisradaddition ... 156
rowDim( ), matrisraddimension ... 156
rowNorm( ), matrisradnorm ... 156
rowSwap( ), växla matrisrad ... 156
rref( ), reducerad radtrappstegsform ... 157
räkna poster i en lista villkorligt , countif( ) ... 39
räkna poster i en lista, count( ) ... 39

S

sammanfoga/slå ihop, augment( ) ... 20
samtidiga ekvationer, simult( ) ... 166
sannolik täthet, normPdf( ) ... 126
sannolikhet för normalfördelning, normCdf( ) ... 125
sannolikhet för student-t-fördelning, tCdf( ) ... 186
sec^{-1}( ), invers sekansfunktion ... 157
sec( ), sekansfunktion ... 157
$sech^{-1}$( ), invers hyperbolisk sekansfunktion ... 158
sech( ), hyperbolisk sekansfunktion ... 158
sekundnotation, " ... 230
seq( ), talföljd ... 159
seqGen( ) ... 160
seqn( ) ... 160
sequence, seq( ) ... 160
series( ), series ... 161
series, series( ) ... 161
setMode( ), ställ in läge ... 162
sfärisk vektorvisning, ▸Sphere ... 176
shift( ), skifta ... 164
sign( ), tecken ... 166

simult( ), samtidiga ekvationer 166
sin⁻¹( ), arcsinus 169
sin( ), sinus 168
sine
visning av uttryck i termer av 167
sinh⁻¹( ), hyperbolisk arcsinus 170
sinh( ), hyperbolisk sinus 169
SinReg, sinusregression 170
sinus, sin( ) 168
sinusregression, SinReg 170
skalär
produkt, dotP( ) 60
skifta, shift( ) 164
slinga, Loop 111
slump
matris, randMat( ) 146
norm, randNorm( ) 146
polynom, randPoly( ) 146
talfrö, RandSeed 147
slumpsampel 146
slut
försök, EndTry 192
medan, EndWhile 202
program, EndPrgm 139
slinga, EndLoop 111
slut medan, EndWhile 202
slut om, EndIf 88
slut slinga, EndLoop 111
solve( ), lös 171
SortA, sortera stigande 175
SortD, sortera fallande 176
sortera
fallande, SortD 176
stigande, SortA 175
språk
hämta språkinformation 83
sqrt( ), kvadratrot 177
standardavvikelse, stdDev( ) 179-180, 199
stat.results 177
stat.values 178
statistik
envariabelstatistik, OneVar 129
fakultet, ! 222
kombinationer, nCr( ) 121

medelvärde, mean( ) 113
median, median( ) 113
permutationer, nPr( ) 127
slumpnorm, randNorm( ) 146
slumptalsfrö, RandSeed 147
standardavvikelse, stdDev( ) 179-180, 199
tvåvariabelresultat, TwoVar 196
varians, variance( ) 200
stdDevPop( ), standardavvikelse för population 179
stdDevSamp( ), standardavvikelse för urval 180
stegvis funktion (2 steg)
mall 6
stegvis funktion (N steg)
mall 7
Stoppkommando 180
string( ), uttryck till sträng 181
strings
right, right( ) 32, 64, 92, 152, 201
sträng
dimension, dim( ) 57
längd 57
strängar
använda för att skapa variabelnamn 243
format, format( ) 77
formatera 77
höger, right( ) 152
indirection, # 228
inom, InString 91
lägg till, & 222
mittsträng, mid( ) 115
numerisk teckenkod, ord( ) 131
rotera, rotate( ) 154
skifta, shift( ) 164
sträng till uttryck, expr( ) 70, 108
teckensträng, char( ) 27
uttryck till sträng, string( ) 181
vänster, left( ) 96
ställ in
läge, setMode( ) 162
större än eller lika med, | 220
större än, > 220
största gemensamma delare, gcd( ) 81
subMat( ), delmatris 182

subMat( ), undermatris 181
substitution med "|"-operatorn 234
subtrahera, - 212
sum( ), summering 181
sumIf( ) 182
summa (G)
    mall 9
summa av kapitalbetalningar 227
summa av räntebetalningar 226
summa, ∑( ) 226
summering, sum( ) 181
sumSeq() 182
svar (sista), Ans 17

T

t-test, tTest 193
T, transponera 183
ta bort
    tomma element från lista 54
    variabel, DelVar 54
tak, ceiling( ) 24-25, 40
talföljd, seq( ) 159
$\tan^{-1}$( ), arctangens 184
tan( ), tangens 183
tangens, tan( ) 183
tangent, tangentLine( ) 185
tangentLine( ) 185
$\tanh^{-1}$( ), hyperbolisk arctangens 185
tanh( ), hyperbolisk tangens 185
taylor( ), Taylorpolynom 186
Taylorpolynom, taylor( ) 186
tCdf( ), studentt fördelningssannolikhet 186
tCollect( ), trigonometrisk insamling 187
tecken
    numerisk kod, ord( ) 131
    sträng, char( ) 27
tecken, sign( ) 166
teckensträng, char( ) 27
Test_2S, 2-sample F test 79
testa för tomrum, isVoid( ) 95
tExpand( ), trigonometrisk utveckling 187
Text, kommando 188
tidsjusterat pengavärde, antal betalningsperioder 195

tidsjusterat pengavärde, betalningsbelopp ... 196
tidsjusterat pengavärde, framtida värde ... 195
tidsjusterat pengavärde, nuvärde ... 196
tidsjusterat pengavärde, ränta ... 195
TInterval, t konfidensintervall ... 188
tInterval_2Samp, 2sampel t-konfidensintervall ... 189
tiopotens, 10^( ) ... 233
ΔtmpCnv() [tmpCnv] ... 190
tmpCnv() ... 190
tomma element ... 238
tomma element, ta bort ... 54
tomrum, testa för ... 95
tPdf( ), studentt täthetsfunktion ... 191
trace( ) ... 192
transponera, T ... 183
trigonometrisk insamling, tCollect( ) ... 187
trigonometrisk utveckling, tExpand( ) ... 187
Try, felhanteringskommando ... 192
Try, försök ... 192
tTest, t-test ... 193
tTest_2Samp, 2-sampel t-test ... 194
TVM-argument ... 196
tvmFV( ) ... 195
tvmI( ) ... 195
tvmN( ) ... 195
tvmPmt( ) ... 196
tvmPV( ) ... 196
TwoVar, tvåvariabelresultat ... 196
tvåvariabelresultat, TwoVar ... 196
täthetsfunktion för student-t, tPdf( ) ... 191

U

undermatris, subMat( ) ... 181
understrykning, _ ... 232
ungefärlig, approx( ) ... 17
unitV( ), enhetsvektor ... 198
unLock, låsa upp variabel eller variabelgrupp ... 199
user-defined functions ... 51
user-defined functions and programs ... 52-53
uteslutning med "|"-operatorn ... 234
uttryck
    sträng till uttryck, expr( ) ... 70, 108

uttryck till lista, exp▸list( ) 68
utvärdera polynom, polyEval( ) 136

V

variabel
skapa namn från en teckensträng 243
variabler
lokal, Local 106
rensa alla enstaka bokstäver 29
ta bort, DelVar 54
variabler och funktioner
kopiera 33
variabler, låsa och låsa upp 84, 107, 199
varians, variance( ) 200

W

warnCodes( ), Warning codes 201

V

varningskoder och meddelanden 253

V

varPop( ) 199

V

varSamp( ), sampelvarians 200

V

vektorer
enhet, unitV( ) 198
skalärprodukt, dotP( ) 60
vektorprodukt, crossP( ) 41
visa cylindrisk vektor, ▸Cylind 47

V

vektorprodukt, crossP( ) 41
when( ), när 201
While, medan 202

V

vinkel, angle( ) 14

V

visa cylindrisk vektor, ▸Cylind 47

V

visa data, Disp 58, 159

V

visa som
- binär, ▸Base2 21
- cylindrisk vektor, ▸Cylind 47
- decimal vinkel, ▸DD 50
- decimalt heltal, ▸Base10 23
- grad/minut/sekund, ▸DMS 58
- hexadecimal, ▸Base16 23
- polär vektor, ▸Polar 134
- rektangulär vektor, ▸Rect 147
- sfärisk vektor, ▸Sphere 176

V

visning i grad/minut/sekund, ▸DMS 58

V

vänster, left( ) 96

X

x², kvadrat 215
XNOR 221
xor, Booleskt exklusivt eller 202

Z

zeroes( ), nollställen 203
zInterval, z konfidensintervall 205
zInterval_1Prop, 1-proportion z-konfidensintervall 206
zInterval_2Prop, 2-proportion z-konfidensintervall 207
zInterval_2Samp, 2-sampel z-konfidensintervall 207
zTest 208
zTest_1Prop, 1-proportion z-test 209

zTest_2Prop, 2-proportion z-test 209
zTest_2Samp, 2-sampel z-test 210