# TI-Nspire™/TI-Nspire™ CX Manual de Referência

Este manual do utilizador aplica-se ao software TI-Nspire™ versão 3.2. Para obter a versão mais recente da documentação, visite education.ti.com/guides.

## *Informações importantes*

Excepto se indicado expressamente na Licença que acompanha um programa, Texas Instruments não dá garantia, explícita ou implícita, incluindo mas não se limitando a quaisquer garantias de comercialização e adequação a um fim particular, relativamente a quaisquer programas ou materiais de documentação e disponibiliza estes materiais unicamente numa base "tal qual". Em nenhum caso, a Texas Instruments será responsável perante alguém por danos especiais, colaterais, incidentais, ou consequenciais em ligação com a ou provenientes da compra ou utilização destas matérias, e a responsabilidade única e exclusiva da Texas Instruments, independentemente da forma de actuação, não excederá a quantia estabelecida na licença do programa. Além disso, a Texas Instruments não será responsável por qualquer queixa de qualquer tipo apresentada contra a utilização destes materiais por terceiros.

### Licença

Consulte a íntegra da licença instalada em **C:\Program Files\TI Education\<TI-Nspire™ Product Name>\license**.

# Índice

## Informações importantes

## Modelos de expressão


Modelo de fracção ..... 1
Modelo de expoente ..... 1
Modelo de raiz quadrada ..... 1
Modelo de raiz de índice N ..... 1
Modelo de expoente e ..... 2
Modelo de log ..... 2
Modelo de Função por ramos (2 ramos) ..... 2
Modelo de Função por ramos (N ramos) ..... 2
Modelo do sistema de 2 equações ..... 3
Modelo do sistema de N equações ..... 3
Modelo do valor absoluto ..... 3
Modelo gg°mm'ss.ss'' ..... 3
Modelo da matriz (2 x 2) ..... 3
Modelo da matriz (1 x 2) ..... 3
Modelo da matriz (2 x 1) ..... 4
Modelo da matriz (m x n) ..... 4
Modelo da soma (Σ) ..... 4
Modelo do produto (Π) ..... 4
Modelo da primeira derivada ..... 5
Modelo da segunda derivada ..... 5
Modelo do integral definido ..... 5


## Lista alfabética

### A


abs() ..... 6
amortTbl() ..... 6
and ..... 6
angle() ..... 7
ANOVA ..... 7
ANOVA2way ..... 8
Ans ..... 9
approx() ..... 10
▸approxFraction() ..... 10
approxRational() ..... 10
arccos() ..... 10
arccosh() ..... 10
arccot() ..... 10
arccoth() ..... 11
arccsc() ..... 11
arccsch() ..... 11
arcsec() ..... 11
arcsech() ..... 11
arcsin() ..... 11
arcsinh() ..... 11
arctan() ..... 11
arctanh() ..... 11
augment() ..... 11
avgRC() ..... 12


### B


bal() ..... 12
▸Base2 ..... 12
▸Base10 ..... 13
▸Base16 ..... 14
binomCdf() ..... 14
binomPdf() ..... 14


### C


ceiling() ..... 14
centralDiff() ..... 15
char() ..... 15
$\chi^2$ 2way ..... 15
$\chi^2$ Cdf() ..... 16
$\chi^2$ GOF ..... 16
$\chi^2$ Pdf() ..... 16
ClearAZ ..... 16
ClrErr ..... 17
colAugment() ..... 17
colDim() ..... 17
colNorm() ..... 17
completeSquare() ..... 18
conj() ..... 18
constructMat() ..... 18
CopyVar ..... 18
corrMat() ..... 19
cos() ..... 19
$\cos^{-1}$() ..... 20
cosh() ..... 21
$\cosh^{-1}$() ..... 21
cot() ..... 21
$\cot^{-1}$() ..... 22
coth() ..... 22
$\coth^{-1}$() ..... 22
count() ..... 22
countif() ..... 23
cPolyRoots() ..... 23
crossP() ..... 23
csc() ..... 24
$\csc^{-1}$() ..... 24
csch() ..... 24
$\text{csch}^{-1}$() ..... 24
CubicReg ..... 25
cumulativeSum() ..... 25
Cycle ..... 26
▸Cylind ..... 26


### D


dbd() ..... 26
▸DD ..... 27
▸Decimal ..... 27
Define ..... 27
Define LibPriv ..... 28
Define LibPub ..... 29
deltaList() ..... 29
DelVar ..... 29
delVoid() ..... 29
det() ..... 30
diag() ..... 30
dim() ..... 30
Disp ..... 31
▸DMS ..... 31
dotP() ..... 31


### E


e^() ..... 32

eff() ..... 32
eigVc() ..... 32
eigVl() ..... 33
Else ..... 33
ElseIf ..... 33
EndFor ..... 33
EndFunc ..... 33
EndIf ..... 33
EndLoop ..... 33
EndPrgm ..... 33
EndTry ..... 33
EndWhile ..... 34
euler() ..... 34
Exit ..... 34
exp() ..... 35
expr() ..... 35
ExpReg ..... 35

## F

factor() ..... 36
FCdf() ..... 36
Fill ..... 37
FiveNumSummary ..... 37
floor() ..... 37
For ..... 38
format() ..... 38
fPart() ..... 38
FPdf() ..... 39
freqTable▸list() ..... 39
frequency() ..... 39
FTest_2Samp ..... 40
Func ..... 40

## G

gcd() ..... 41
geomCdf() ..... 41
geomPdf() ..... 41
getDenom() ..... 41
getLangInfo() ..... 42
getLockInfo() ..... 42
getMode() ..... 42
getNum() ..... 43
getType() ..... 43
getVarInfo() ..... 44
Goto ..... 44
▸Grad ..... 45

## I

identity() ..... 45
If ..... 45
ifFn() ..... 46
imag() ..... 46
Indirecta ..... 47
inString() ..... 47
int() ..... 47
intDiv() ..... 47
interpolate() ..... 48
inv $\chi^2$() ..... 48
invF() ..... 48
invNorm() ..... 48
invt() ..... 48
iPart() ..... 49
irr() ..... 49
isPrime() ..... 49
isVoid() ..... 49

## L

Lbl ..... 50
lcm() ..... 50
left() ..... 50
libShortcut() ..... 51
LinRegBx ..... 51
LinRegMx ..... 52
LinRegtIntervals ..... 53
LinRegtTest ..... 54
linSolve() ..... 55
ΔList() ..... 55
list▸mat() ..... 55
ln() ..... 55
LnReg ..... 56
Local ..... 57
Lock ..... 57
log() ..... 57
Logistic ..... 58
LogisticD ..... 59
Loop ..... 60
LU ..... 60

## M

mat▸list() ..... 60
max() ..... 61
mean() ..... 61
median() ..... 61
MedMed ..... 62
mid() ..... 63
min() ..... 63
mirr() ..... 64
mod() ..... 64
mRow() ..... 64
mRowAdd() ..... 64
MultReg ..... 65
MultRegIntervals ..... 65
MultRegTests ..... 66

## N

nand ..... 67
nCr() ..... 67
nDerivative() ..... 68
newList() ..... 68
newMat() ..... 68
nfMax() ..... 68
nfMin() ..... 69
nInt() ..... 69
nom() ..... 69
nor ..... 69
norm() ..... 70
normCdf() ..... 70
normPdf() ..... 70
not ..... 70
nPr() ..... 71
npv() ..... 72
nSolve() ..... 72

## O

OneVar ..... 73
or (ou) ..... 74

ord() ..... 74

## P

P▶Rx() ..... 74
P▶Ry() ..... 75
PassErr ..... 75
piecewise() ..... 75
poissCdf() ..... 75
poissPdf() ..... 75
▶Polar ..... 76
polyEval() ..... 76
polyRoots() ..... 76
PowerReg ..... 77
Prgm ..... 78
prodSeq() ..... 78
Produto (Pl) ..... 78
product() ..... 78
propFrac() ..... 79

## Q

QR ..... 79
QuadReg ..... 80
QuartReg ..... 81

## R

R▶Pθ() ..... 82
R▶Pr() ..... 82
▶Rad ..... 82
rand() ..... 82
randBin() ..... 83
randInt() ..... 83
randMat() ..... 83
randNorm() ..... 83
randPoly() ..... 83
randSamp() ..... 83
RandSeed ..... 84
real() ..... 84
▶Rect ..... 84
ref() ..... 85
remain() ..... 85
Request ..... 86
RequestStr ..... 87
Return ..... 87
right() ..... 87
rk23() ..... 88
root() ..... 88
rotate() ..... 89
round() ..... 89
rowAdd() ..... 90
rowDim() ..... 90
rowNorm() ..... 90
rowSwap() ..... 90
rref() ..... 90

## S

sec() ..... 91
sec$^{-1}$() ..... 91
sech() ..... 91
sech$^{-1}$() ..... 91
seq() ..... 92
seqGen() ..... 92
seqn() ..... 93
setMode() ..... 93
shift() ..... 94
sign() ..... 95
simult() ..... 95
sin() ..... 96
sin$^{-1}$() ..... 96
sinh() ..... 97
sinh$^{-1}$() ..... 97
SinReg ..... 98
SortA ..... 98
SortD ..... 99
▶Sphere ..... 99
sqrt () ..... 99
stat.results ..... 100
stat.values ..... 101
stDevPop() ..... 101
stDevSamp() ..... 101
Stop (Parar) ..... 102
Store (Guardar) ..... 102
string() ..... 102
subMat() ..... 102
Sigma (Soma) ..... 102
sum() ..... 102
sumIf() ..... 103
sumSeq() ..... 103
system() ..... 103

## T

T (transpor) ..... 103
tan() ..... 104
tan$^{-1}$() ..... 104
tanh() ..... 105
tanh$^{-1}$() ..... 105
tCdf() ..... 106
Text ..... 106
Then ..... 106
tInterval ..... 106
tInterval_2Samp ..... 107
tPdf() ..... 107
trace() ..... 108
Try ..... 108
tTest ..... 109
tTest_2Samp ..... 109
tvmFV() ..... 110
tvmI() ..... 110
tvmN() ..... 110
tvmPmt() ..... 110
tvmPV() ..... 110
TwoVar ..... 111

## U

unitV() ..... 112
unLock ..... 113

## V

varPop() ..... 113
varSamp() ..... 113

## W

warnCodes() ..... 114
when() ..... 114
While ..... 114

## X

xor (xou) ..... 115

## Z

zInterval ..... 115
zInterval_1Prop ..... 116
zInterval_2Prop ..... 116
zInterval_2Samp ..... 117
zTest ..... 117
zTest_1Prop ..... 118
zTest_2Prop ..... 118
zTest_2Samp ..... 119

## Símbolos

+ (adicionar) ..... 120
– (subtrair) ..... 120
· (multiplicar) ..... 121
/ (dividir) ..... 121
^ (potência) ..... 122
x 2 (quadrado) ..... 122
.+ (ponto adicionar) ..... 123
.– (ponto subtracção) ..... 123
. · (ponto mult.) ..... 123
. / (ponto dividir) ..... 123
.^ (ponto potência) ..... 123
- (negação) ..... 124
% (percentagem) ..... 124
= (igual) ..... 125
≠ (diferente) ..... 125
< (menor que) ..... 126
≤ (igual ou menor que) ..... 126
> (maior que) ..... 126
≥ (igual ou maior que) ..... 126
⇒ (implicação lógica) ..... 127
⇔ (implicação lógica dupla, XNOR) ..... 127
! (factorial) ..... 127
& (acrescentar) ..... 127
d() (derivada) ..... 128
∫() (integral) ..... 128
√() (raiz quadrada) ..... 128
Π () (prodSeq) ..... 129
Σ() (sumSeq) ..... 129
ΣInt() ..... 130
ΣPrn() ..... 130
# (indirecta) ..... 131
E (notação científica) ..... 131
g (gradianos) ..... 131
r (radianos) ..... 131
° (graus) ..... 132
°, ', '' (grau/minuto/segundo) ..... 132
∠ (ângulo) ..... 132
_ (carácter de sublinhado como um elemento vazio) ..... 132
10^() ..... 133
^-1 (recíproco) ..... 133
| (operador de limite) ..... 133
→ (guardar) ..... 134
:= (atribuir) ..... 134
© (comentário) ..... 135
0b, 0h ..... 135

## Elementos (nulos) vazios

Cálculos que envolvam elementos nulos ..... 136
Argumentos da lista que contenham elementos nulos ..... 136

## Atalhos para introduzir expressões matemáticas

## Hierarquia do EOS™ (Equation Operating System)

## Mensagens e códigos de erros

## Códigos de aviso e mensagens

## Apoio técnico, manutenção e garantia dos produtos Texas Instruments

# Manual de Referência TI-Nspire™

Este manual lista os modelos, as funções, os comandos e os operadores disponíveis para avaliar expressões matemáticas.

## *Modelos de expressão*

Os modelos de expressão oferecem uma forma simples para introduzir expressões matemáticas em notação matemática padronizada. Quando introduzir um modelo, aparece na linha de entrada com pequenos blocos em posições em que pode introduzir elementos. Um cursor mostra o elemento que pode introduzir.

Utilize as teclas de setas ou prima [tab] para mover o cursor para a posição de cada elemento e escreva um valor ou uma expressão para o elemento. Prima [enter] ou [ctrl] [enter] para avaliar a expressão.

**Modelo de fracção** — **Teclas** [ctrl] [÷]

**Nota:** Consulte também **/ (dividir)**, página 121.

Exemplo:

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

**Modelo de expoente** — **Tecla** [^]

**Nota:** Escreva o primeiro valor, prima [^] e, em seguida, escreva o expoente. Para colocar o cursor na base, prima a seta direita ( ▶ ).

**Nota:** Consulte também **^ (potência)**, página 122.

Exemplo:

$$2^3 \qquad 8$$

**Modelo de raiz quadrada** — **Teclas** [ctrl] [x²]

**Nota:** Consulte também **√() (raiz quadrada)**, página 128.

Exemplo:

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

$$\sqrt{\{9,16,4\}} \qquad \{3,4,2\}$$

**Modelo de raiz de índice N** — **Teclas** [ctrl] [^]

**Nota:** Consulte também **raiz()**, página 88.

Exemplo:

$$\sqrt[3]{8} \qquad 2$$

$$\sqrt[3]{\{8,27,15\}} \qquad \{2,3,2.46621\}$$

### Modelo de expoente e

**Tecla** $e^x$

$e^{\square}$

Exponencial natural $e$ elevado à potência

**Nota:** Consulte também e **^()**, página 32.

Exemplo:

$e^1$ 2.71828182846

### Modelo de log

**Teclas** ctrl $10^x$

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

Calcule o log para uma base especificada. Para uma predefinição de base 10, omita a base.

**Nota:** Consulte também **log()**, página 57.

Exemplo:

$\log_4(2.)$ 0.5

### Modelo de Função por ramos (2 ramos)

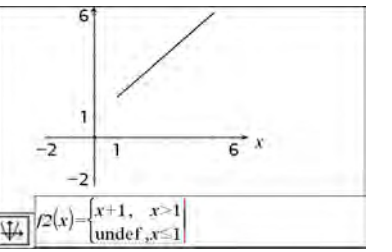
**Catálogo>**

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

Permite criar expressões e condições para uma função por ramos de 2 ramos. Para adicionar um ramo, clique no modelo e repita o modelo.

**Nota:** Consulte também **piecewise()**, página 75.

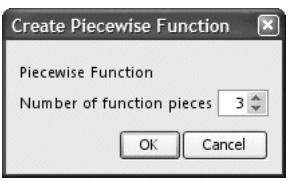
Exemplo:




### Modelo de Função por ramos (N ramos)

**Catálogo>**

Permite criar expressões e condições para uma função por ramos de $N$ -ramos. Para adicionar um ramo, clique no modelo e repita o modelo.




**Nota:** Consulte também **piecewise()**, página 75.

Exemplo:
Consulte o exemplo para o modelo de Função por ramos (2 ramos).

**Modelo do sistema de 2 equações**

**Catálogo>**

$\left\{\square\right.$
$\square$

Cria um sistema de duas equações lineares. Para adicionar uma linha a um sistema existente, clique no modelo e repita o modelo.

**Nota:** Consulte também **sistema()**, página 103.

Exemplo:

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

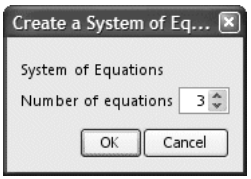
$\text{solve}\left(\begin{cases} y=x^2-2 \\ x+2\cdot y=-1 \end{cases}, x, y\right)$

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

**Modelo do sistema de N equações**

**Catálogo>**

Permite criar um sistema de N equações lineares. Pede *N*.




**Nota:** Consulte também **sistema()**, página 103.

Exemplo:
Consulte o exemplo do modelo do sistema de equações (2 equações).

**Modelo do valor absoluto**

**Catálogo>**

$|\square|$ **Nota:** Consulte também **abs()**, página 6.

Exemplo:

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

**Modelo gg°mm'ss.ss''**

**Catálogo>**

$\square°\square'\square''$

Permite introduzir ângulos na forma **gg** ° **mm** ' **ss.ss** '', em que **gg** é o número de graus decimais, **mm** é o número de minutos e **ss.ss** é o número de segundos.

Exemplo:

$30°15'10'' \qquad 0.528011$

**Modelo da matriz (2 x 2)**

**Catálogo>**

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

Cria uma matriz 2 x 2.

Exemplo:

$\begin{bmatrix}1 & 2 \\ 3 & 4\end{bmatrix}\cdot 5 \qquad \begin{bmatrix}5 & 10 \\ 15 & 20\end{bmatrix}$

**Modelo da matriz (1 x 2)**

**Catálogo>**

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

Exemplo:

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

**Modelo da matriz (2 x 1)**

**Catálogo>**

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

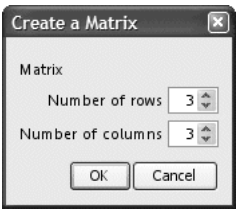
Exemplo:

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

**Modelo da matriz (m x n)**

**Catálogo>**

O modelo aparece depois de lhe ser pedido para especificar o número de linhas e colunas.




**Nota:** Se criar uma matriz com um grande número de linhas e colunas, pode demorar alguns momentos a aparecer.

Exemplo:

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

**Modelo da soma (Σ)**

**Catálogo>**

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

Exemplo:

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

**Nota:** Consulte também **Σ() (sumSeq)**, página 129.

**Modelo do produto (Π)**

**Catálogo>**

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

Exemplo:

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

**Nota:** Consulte também **Π () (prodSeq)**, página 129.

**Modelo da primeira derivada**

**Catálogo >**

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

Pode utilizar o modelo da primeira derivada para calcular a primeira derivada num ponto numericamente com métodos de diferenciação automáticos.

**Nota:** Consulte também **d() (derivada)** , página 128.

Exemplo:

$$\frac{d}{dx}(|x|)|x=0 \qquad \text{undef}$$

**Modelo da segunda derivada**

**Catálogo >**

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

Pode utilizar o modelo da segunda derivada para calcular a segunda derivada num ponto numericamente com métodos de diferenciação automáticos.

**Nota:** Consulte também **d() (derivada)** , página 128.

Exemplo:

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

**Modelo do integral definido**

**Catálogo>**

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

Pode utilizar o modelo do integral definido para definir o integral definido numericamente com o mesmo método de nInt().

**Nota:** Consulte também **nInt()** , página 69.

Exemplo:

$$\int_0^{10} x^2 \, dx \qquad 333.333$$

# Lista alfabética

Os itens cujos nomes não sejam alfabéticos (como +, !, e >) são listados no fim desta secção, começando na página 120. Salvo indicação em contrário, todos os exemplos desta secção foram efectuados no modo de reinicialização predefinido e todas as variáveis são assumidas como indefinidas.

## A

### abs()

**Catálogo >**

**abs(***Valor1***)** ⇒ *valor*
**abs(***Lista1***)** ⇒ *lista*
**abs(***Matriz1***)** ⇒ *matriz*

Devolve o valor absoluto do argumento.

**Nota:** Consulte também **Modelo do valor absoluto**, página 3.

Se o argumento for um número complexo, devolve o módulo do número.

**Nota:** Todas as variáveis indefinidas são tratadas como variáveis reais.

$$\left|\left\{\frac{\pi}{2},\frac{-\pi}{3}\right\}\right| \qquad \{1.5708, 1.0472\}$$

$$|2-3\cdot i| \qquad 3.60555$$

### amortTbl()

**Catálogo >**

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

Função de amortização que devolve uma matriz como uma tabela de amortização para um conjunto de argumentos TVM.

*NPmt* é o número de pagamentos a incluir na tabela. A tabela começa com o primeiro pagamento.

*N*, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 111.

- Se omitir *Pmt*, predefine-se para *Pmt* = **tvmPmt (***N, I, PV, FV, PpY, CpY, PmtAt***)**.
- Se omitir *FV*, predefine-se para *FV* =0.
- As predefinições para *PpY*, *CpY* e *PmtAt* são iguais às predefinições para as funções TVM.

*ValorArredondado* especifica o número de casas decimais para arredondamento. Predefinição=2.

As colunas da matriz de resultados são por esta ordem: Número de pagamentos, montante pago para juros, montante para capital e saldo.

O saldo apresentado na linha *n* é o saldo após o pagamento *n*.

Pode utilizar a matriz de saída como entrada para as outras funções de amortização **Σint()** e **Σprn()**, página 130 e **bal()**, página 12.

$$\text{amortTbl}(12,60,10,5000,,,12,12)$$

| | | | |
|---|---|---|---|
| 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 |

### and

**Catálogo >**

*ExprBooleana1* **and** *ExprBooleana2* ⇒ *Expressão booleana*
*ListaBooleana1* **and** *ListaBooleana2* ⇒ *Lista booleana*
*MatrizBooleana1* **and** *MatrizBooleana2* ⇒ *Matriz booleana*

Devolve falso, verdadeiro ou uma forma simplificada da entrada original.

## and

Catálogo >

*Inteiro1* **and** *Inteiro2* ⇒ *número inteiro*

Compara dois números inteiros reais bit a bit com uma operação **and**. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se ambos os bits forem 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respectivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 10).

Se introduzir um número inteiro decimal muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado.

No modo base Hex:

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

**Importante:** Zero, não a letra O.

No modo base Bin:

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

No modo base Dec:

| | |
|---|---|
| 37 and 0b100 | 4 |

**Nota:** Uma entrada binária pode ter até 64 dígitos (não contando com o prefixo 0b). Uma entrada hexadecimal pode ter até 16 dígitos.

## angle()

Catálogo >

**angle(***Valor1***)** ⇒ *valor*

Devolve o ângulo do argumento, interpretando o argumento como um número complexo.

No modo de ângulo Graus:

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

No modo de ângulo Gradianos:

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

No modo de ângulo Radianos:

| | |
|---|---|
| $\text{angle}(1+i)$ | 0.785398 |
| $\text{angle}(\{1+2\cdot i, 3+0\cdot i, 0-4\cdot i\})$ | $\{1.10715, 0., -1.5708\}$ |

**angle(***Lista1***)** ⇒ *lista*
**angle(***Matriz1***)** ⇒ *matriz*

Devolve uma lista ou matriz de ângulos dos elementos em *Lista1* ou *Matriz1*, interpretando cada elemento como um número complexo que representa um ponto de coordenada rectangular bidimensional.

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

## ANOVA

Catálogo >

**ANOVA** *Lista1*, *Lista2* [, *Lista3*, ..., *Lista20* ][, *Marcador*]

Efectua uma análise de variação de uma via para comparar as médias de 2 a 20 populações. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

*Marcador* =0 para Dados, *Marcador* =1 para Estatística

| Variável de saída | Descrição |
|---|---|
| stat.F | Valor da estatística F |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.df | Graus de liberdade dos grupos |
| stat.SS | Soma dos quadrados dos grupos |
| stat.MS | Quadrados médios para os grupos |
| stat.dfError | Graus de liberdade dos erros |

| Variável de saída | Descrição |
|---|---|
| stat.SSError | Soma dos quadrados dos erros |
| stat.MSError | Quadrado médio para os erros |
| stat.sp | Desvio padrão associado |
| stat.xbarlist | Média da entrada das listas |
| stat.CLowerList | Intervalos de confiança de 95% para a média de cada lista de entrada |
| stat.CUpperList | Intervalos de confiança de 95% para a média de cada lista de entrada |

**ANOVA2way** **Catálogo >**

**ANOVA2way** *Lista1*, *Lista2* [, *Lista3*, …, *Lista10* ][, *LinhaNiv*]

Calcula uma análise de variação bidirecional através da comparação das médias de 2 a 10 populações. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

*LinhaNiv*=0 para Bloco

*LinhaNiv*=2,3,...,*Len*-1, para Dois fatores, em que *Len*=comprimento(*Lista1*)=comprimento(*Lista2*) = = comprimento(*Lista10*) e *Len* / *LinhaNiv* ∈ {2,3, }

Saídas: Design do bloco

| Variável de saída | Descrição |
|---|---|
| stat.F | F estatística do factor da coluna |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.df | Graus de liberdade do factor da coluna |
| stat.SS | Soma dos quadrados do factor da coluna |
| stat.MS | Quadrados médios para o factor da coluna |
| stat.FBloco | F estatística para o factor |
| stat.PValBlock | Menor probabilidade de rejeição da hipótese nula |
| stat.dfBlock | Graus de liberdade para factor |
| stat.SSBlock | Soma dos quadrados para o factor |
| stat.MSBlock | Quadrados médios para o factor |
| stat.dfError | Graus de liberdade dos erros |
| stat.SSError | Soma dos quadrados dos erros |
| stat.MSError | Quadrados médios para os erros |
| stat.s | Desvio padrão do erro |

Saídas do factor da coluna

| Variável de saída | Descrição |
|---|---|
| stat.Fcol | F estatística do factor da coluna |

| Variável de saída | Descrição |
|---|---|
| stat.PValCol | Valor da probabilidade do factor da coluna |
| stat.dfCol | Graus de liberdade do factor da coluna |
| stat.SSCol | Soma dos quadrados do factor da coluna |
| stat.MSCol | Quadrados médios para o factor da coluna |

Saídas do factor da linha

| Variável de saída | Descrição |
|---|---|
| stat.FLinha | F estatística do factor da linha |
| stat.PValRow | Valor da probabilidade do factor da linha |
| stat.dfRow | Graus de liberdade do factor da linha |
| stat.SSRow | Soma dos quadrados do factor da linha |
| stat.MSRow | Quadrados médios para o factor da linha |

Saídas de interacção

| Variável de saída | Descrição |
|---|---|
| stat.FInteragir | Festatística da interacção |
| stat.PValInteract | Valor da probabilidade da interacção |
| stat.dfInteract | Graus de liberdade da interacção |
| stat.SSInteract | Soma de quadrados da interacção |
| stat.MSInteract | Quadrados médios para interacção |

Saídas de erros

| Variável de saída | Descrição |
|---|---|
| stat.dfError | Graus de liberdade dos erros |
| stat.SSError | Soma dos quadrados dos erros |
| stat.MSError | Quadrados médios para os erros |
| s | Desvio padrão do erro |

## Ans

**Teclas** ctrl (−)

**Ans** ⇒ *valor*

Devolve o resultado da expressão avaliada mais recentemente.

| | |
|---|---|
| 56 | 56 |
| 56+4 | 60 |
| 60+4 | 64 |

### approx()

Catálogo >

**approx(***Valor1***)** ⇒ *número*

Devolve a avaliação do argumentos como uma expressão com valores decimais, quando possível, independentemente do modo **Auto ou Aproximado** actual.

Isto é equivalente a introduzir o argumento e a introduzir ctrl enter.

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

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

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

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

$\text{approx}\begin{bmatrix}\frac{1}{3} & \frac{1}{9}\end{bmatrix}$ → $\begin{bmatrix}0.333333 & 0.111111\end{bmatrix}$

**approx(***Lista1***)** ⇒ *lista*
**approx(***Matriz1***)** ⇒ *matriz*

Devolve uma lista ou uma *matriz* em que cada elemento foi avaliado para um valor decimal, quando possível.

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

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

### ▸approxFraction()

Catálogo >

*Valor* **▸approxFraction([***Tol***])** ⇒ *valor*
*Lista* **▸approxFraction([***Tol***])** ⇒ *lista*
*Matriz* **▸approxFraction([***Tol***])** ⇒ *matriz*

Devolve a entrada como uma fracção com uma tolerância de *Tol*. Se omitir *Tol*, é utilizada uma tolerância de 5.E-14.

**Nota:** Pode introduzir esta função através da escrita de **@>approxFraction(...)** no teclado do computador.

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

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

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

### approxRational()

Catálogo >

**approxRational(***Valor*[**,** *Tol*]**)** ⇒ *valor*
**approxRational(** *Lista* [ **,** *Tol*] **)** ⇒ *lista*
**approxRational(** *Matriz* [ **,** *Tol*] **)** ⇒ *matriz*

Devolve o argumento como uma fracção com uma tolerância de *Tol*. Se omitir *Tol*, é utilizada uma tolerância de 5.E-14.

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

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

### arccos()

Consulte cos⁻¹(), página 20.

### arccosh()

Consulte cosh⁻¹(), página 21.

### arccot()

Consulte cot⁻¹(), página 22.

**arccoth()** Consulte $\coth^{-1}()$, página 22.

**arccsc()** Consulte $\csc^{-1}()$, página 24.

**arccsch()** Consulte $\text{csch}^{-1}()$, página 24.

**arcsec()** Consulte $\sec^{-1}()$, página 91.

**arcsech()** Consulte $\text{sech}^{-1}()$, página 91.

**arcsin()** Consulte $\sin^{-1}()$, página 96.

**arcsinh()** Consulte $\sinh^{-1}()$, página 97.

**arctan()** Consulte $\tan^{-1}()$, página 104.

**arctanh()** Consulte $\tanh^{-1}()$, página 105.

## augment()

**Catálogo >**

**augment(***Lista1, Lista2***)** ⇒ *lista*

Devolve uma nova lista que é a *Lista2* acrescentada ao fim da *Lista1*.

**augment(***Matriz1***,** *Matriz2***)** ⇒ *matriz*

Devolve uma nova lista que é a *Matriz2* acrescentada ao fim da *Matriz1*. Quando utilizar o carácter "," as matrizes têm de ter dimensões de colunas iguais, e a *Matriz2* é acrescentada à *Matriz1* como novas colunas. Não altere *Matriz1* ou *Matriz2*.

$\text{augment}(\{1,-3,2\},\{5,4\})$ $\quad \{1,-3,2,5,4\}$

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

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

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

### avgRC()

Catálogo >

**avgRC(***Expr1*, *Var* [=*Valor*] [, *Passo*]**)** ⇒ *expressão*
**avgRC(***Expr1*, *Var* [=*Valor*] [, *Lista1*]**)** ⇒ *lista*
**avgRC(***Lista1*, *Var* [=*Valor*] [, *Passo*]**)** ⇒ *lista*
**avgRC(***Matriz1*, *Var* [=*Valor*] [, *Passo*]**)** ⇒ *matriz*

Devolve o quociente de diferença de avanço (taxa de câmbio média).

*Expr1* pode ser um nome de função definido pelo utilizador (ver **Func**).

Ao especificar o *Valor*, substitui qualquer atribuição de variável anterior ou qualquer substituição atual "|" para a variável.

*Passo* é o valor do passo. Se omitir *Passo*, predefine-se para 0,001.

Não se esqueça de que a função similar **centralDiff()** utiliza o quociente de diferença central.

$x:=2$ → 2

$\text{avgRC}(x^2-x+2,x)$ → 3.001

$\text{avgRC}(x^2-x+2,x,.1)$ → 3.1

$\text{avgRC}(x^2-x+2,x,3)$ → 6

## B

### bal()

Catálogo >

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

**bal(***NPmt*, *TabelaDeDepreciação***)** ⇒ *valor*

Função de amortização que calcula o saldo do plano após um pagamento especificado.

*N*, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 111.

*NPmt* especifica o número de pagamentos a partir dos quais quer os dados calculados.

*N*, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 111.

- Se omitir *Pmt*, predefine-se para *Pmt* = **tvmPmt(***N, I, PV, FV, PpY, CpY, PmtAt***)**.
- Se omitir *FV*, predefine-se para *FV* =0.
- As predefinições para *PpY*, *CpY* e *PmtAt* são iguais às predefinições para as funções TVM.

*ValorArredondado* especifica o número de casas decimais para arredondamento. Predefinição=2.

**bal(***NPmt*, *TabelaDeDepreciação***)** calcula o saldo após o número de pagamentos *NPmt*, baseado na tabela de amortização *TabelaDeDepreciação*. O argumento *TabelaDeDepreciação* tem de ser uma matriz no forma descrita em **amortTbl()**, página 6.

**Nota:** Consulte também **ΣInt()** e **ΣPrn()**, página 130.

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

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

$$\begin{bmatrix} 0 & 0. & 0. & 5000. \\ 1 & -23.35 & -825.63 & 4174.37 \\ 2 & -19.49 & -829.49 & 3344.88 \\ 3 & -15.62 & -833.36 & 2511.52 \\ 4 & -11.73 & -837.25 & 1674.27 \\ 5 & -7.82 & -841.16 & 833.11 \\ 6 & -3.89 & -845.09 & -11.98 \end{bmatrix}$$

$\text{bal}(4,tbl)$ → 1674.27

### ▶Base2

Catálogo >

*NúmeroInteiro1* **▶Base2** ⇒ *número inteiro*

**Nota:** Pode introduzir este operador através da escrita de **@>Base2** no teclado do computador.

Converte *NúmeroInteiro1* para um número binário. Os números binários ou hexadecimais têm sempre um prefixo 0b ou 0h, respectivamente.

256▶Base2 → 0b100000000

0h1F▶Base2 → 0b11111

## ▸Base2

**Catálogo >**

Zero, não a letra O, seguido por b ou h.

0b *NúmeroBinário*
0h *NúmeroHexadecimal*

Um número binário pode ter até 64 dígitos. Um número hexadecimal pode ter até 16 dígitos.

Sem um prefixo, *NúmeroInteiro1* é tratado como decimal (base 10). O resultado aparece em binário, independentemente do modo base.

Os números negativos aparecem no formato de "complemento de dois". Por exemplo,

−1 aparece como
0hFFFFFFFFFFFFFFFF no modo base Hex
0b111...111 (64 1's) no modo base Binário

$-2^{63}$ aparece como
0h8000000000000000 no modo base Hex
0b100...000 (63 zeros) no modo base Binário

Se introduzir um número inteiro na base 10 fora do intervalo de uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Considere os seguintes exemplos de valores fora do intervalo.

$2^{63}$ torna-se $-2^{63}$ e aparece como
0h8000000000000000 no modo base Hex
0b100...000 (63 zeros) no modo base Binário

$2^{64}$ torna-se 0 e aparece como
0h0 no modo base Hex
0b0 no modo base Binário

$-2^{63} - 1$ torna-se $2^{63} - 1$ e aparece como
0h7FFFFFFFFFFFFFFF no modo base Hex
0b111...111 (64 1's) no modo base Binário

## ▸Base10

**Catálogo >**

*NúmeroInteiro1* **▸Base10** ⇒ *número inteiro*

**Nota:** Pode introduzir este operador através da escrita de **@>Base10** no teclado do computador.

Converte *NúmeroInteiro1* para um número decimal (base 10). Uma entrada binária ou hexadecimal têm de ter sempre um prefixo 0b ou 0h, respectivamente.

0b *NúmeroBinário*
0h *NúmeroHexadecimal*

Zero, não a letra O, seguido por b ou h.

Um número binário pode ter até 64 dígitos. Um número hexadecimal pode ter até 16 dígitos.

Sem um prefixo, *NúmeroInteiro1* é tratado como decimal. O resultado aparece em decimal, independentemente do modo base.

| | |
|---|---|
| 0b10011▸Base10 | 19 |
| 0h1F▸Base10 | 31 |

### ▶Base16

**Catálogo >**

*NúmeroInteiro1* **▶Base16** ⇒ *número inteiro*

**Nota:** Pode introduzir este operador através da escrita de **@>Base16** no teclado do computador.

Converte *NúmeroInteiro1* para um número hexadecimal. Os números binários ou hexadecimais têm sempre um prefixo 0b ou 0h, respectivamente.

0b *NúmeroBinário*
0h *NúmeroHexadecimal*

Zero, não a letra O, seguido por b ou h.

Um número binário pode ter até 64 dígitos. Um número hexadecimal pode ter até 16 dígitos.

Sem um prefixo, *NúmeroInteiro1* é tratado como decimal (base 10). O resultado aparece em hexadecimal, independentemente do modo base.

Se introduzir um número inteiro na base 10 muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Para mais informações, consulte **▶Base2**, página 12.

| | |
|---|---|
| 256▶Base16 | 0h100 |
| 0b111100001111▶Base16 | 0hF0F |

### binomCdf()

**Catálogo >**

**binomCdf(***n***,** *p***)** ⇒ *número*

**binomCdf(***n***,***p***,***LimiteInferior***,***LimiteSuperior***)** ⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

**binomCdf(***n***,***p***,***LimiteSuperior***)** para P(0≤X≤*LimiteSuperior*) ⇒ *número* se *LimiteSuperior* for um número, *lista* se *LimiteSuperior* for uma lista

Calcula uma probabilidade cumulativa para a distribuição binomial discreta com *n* número de tentativas e a probabilidade *p* de sucesso de cada tentativa.

Para P(X ≤ *LimiteSuperior*), defina *LimiteInferior*=0

### binomPdf()

**Catálogo >**

**binomPdf(***n***,** *p***)** ⇒ *número*

**binomPdf(***n***,** *p***,** *ValX***)** ⇒ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula uma probabilidade para a distribuição binomial discreta com o *n* número de tentativas e a probabilidade *p* de sucesso de cada tentativa.

# C

### ceiling()

**Catálogo >**

**ceiling(***Valor1***)** ⇒ *valor*

Devolve o número inteiro mais próximo que é ≥ o argumento.

O argumento pode ser um número complexo ou real.

**Nota:** Consulte também **floor()**.

| | |
|---|---|
| ceiling(.456) | 1. |

## ceiling()

Catálogo >

**ceiling(***Lista1***)** ⇒ *lista*
**ceiling(***Matriz1***)** ⇒ *matriz*

Devolve uma lista ou matriz do ceiling de cada elemento.

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

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

## centralDiff()

Catálogo >

**centralDiff(***Expr1***,***Var* [=*Valor*][**,***Passo*]**)** ⇒ *expressão*
**centralDiff(***Expr1***,***Var* [**,***Passo*]**)|***Var*=*Valor* ⇒ *expressão*
**centralDiff(***Expr1***,***Var* [=*Valor*][**,***Lista*]**)** ⇒ *lista*
**centralDiff(***Lista1***,***Var* [=*Valor* [**,***Passo*]**)** ⇒ *lista*
**centralDiff(***Matriz1***,***Var* [=*Valor* [**,***Passo*]**)** ⇒ *matriz*

Devolve a derivada numérica com a fórmula do quociente da diferença central.

Ao especificar o *Valor*, substitui qualquer atribuição de variável anterior ou qualquer substituição atual "|" para a variável.

*Passo* é o valor do passo. Se omitir *Passo*, predefine-se para 0,001.

Quando utilizar *Lista1* ou *Matriz1* , a operação é mapeada através dos valores da lista ou dos elementos da matriz.

**Nota:** Consulte também **avgRC()**.

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

## char()

Catálogo >

**char(***Número inteiro***)** ⇒ *carácter*

Devolve uma cadeia de caracteres com o carácter numerado *Número inteiro* a partir do conjunto de caracteres da unidade portátil. O intervalo válido para o *Número inteiro* é 0–65535.

$\text{char}\left(38\right)$ → "&"

$\text{char}\left(65\right)$ → "A"

## $\chi^2$ 2way

Catálogo >

$\chi^2$ **2way** *MatrizObs*
**chi22way** *MatrizObs*

Calcula um teste $\chi^2$ para associação à tabela de contagens bidireccional na matriz observada *MatrizObs*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Para mais informações sobre o efeito dos elementos vazios numa matriz, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.$\chi^2$ | Estatística do Qui quadrado: soma (observada - prevista) $^2$ /prevista |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.df | Graus de liberdade para a estatística do Qui quadrado |
| stat.ExpMat | Matriz da tabela de contagem de elementos previsto, assumindo a hipótese nula |
| stat.CompMat | Matriz de contribuições da estatística do Qui quadrado dos elementos |

## $\chi^2$ Cdf()

**Catálogo >**

$\chi^2$**Cdf(***LimiteInferior*,*LimiteSuperior*,*df***)** ⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

**chi2Cdf(***LimiteInferior*,*LimiteSuperior*,*df***)** ⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

Calcula a probabilidade de distribuição $\chi^2$ entre *LimiteInferior* e *LimiteSuperior* para os graus de liberdade especificados *df*.

Para P($X \le$ *LimiteSuperior*), defina *LimiteInferior* = 0.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

## $\chi^2$ GOF

**Catálogo >**

$\chi^2$ **GOF** *Lista obs*, *Lista exp*, *df*

**chi2GOF** *Lista obs*, *Lista exp*, *df*

Efectua um teste para confirmar que os dados da amostra são de uma população que está em conformidade com uma distribuição especificada. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.$\chi^2$ | Estatística do Qui quadrado: soma((observada - prevista) $^2$ /prevista |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.df | Graus de liberdade para a estatística do Qui quadrado |
| stat.CompList | Matriz de contribuições da estatística do Qui quadrado dos elementos |

## $\chi^2$ Pdf()

**Catálogo >**

$\chi^2$**Pdf(***ValX*,*df***)** ⇒ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

**chi2Pdf(***ValX*,*df***)** ⇒ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula a função de densidade de probabilidade (pdf) para a distribuição $\chi^2$ num valor *ValX* especificado para os graus de liberdade especificados *df*.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

## ClearAZ

**Catálogo >**

**ClearAZ**

Apaga todas as variáveis de um carácter no espaço do problema actual.

Se uma ou mais variáveis estiverem bloqueadas, este comando mostra uma mensagem de erro e só elimina as variáveis desbloqueadas. Consulte **unLock**, página 113.

| | |
|---|---|
| $5 \rightarrow b$ | 5 |
| $b$ | 5 |
| ClearAZ | *Done* |
| $b$ | "Error: Variable is not defined" |

## ClrErr

Catálogo >

**ClrErr**

Apaga o estado de erro e define a variável do sistema *errCode* para zero.

A proposição **Else** do bloco **Try...Else...EndTry** deve utilizar **ClrErr** ou **PassErr**. Se tiver de processar ou ignorar o erro, utilize **ClrErr**. Se não souber o que fazer com o erro, utilize **PassErr** para o enviar para a rotina de tratamento de erros seguinte. Se não existirem mais rotinas de tratamento de erros **Try...Else...EndTry** pendente, a caixa de diálogo de erros aparecerá como normal.

**Nota:** Consulte também **PassErr**, página 75, e **Try**, página 108.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

Para ver um exemplo de **ClrErr**, consulte o exemplo 2 no comando **Try**, página 108.

## colAugment()

Catálogo >

**colAugment(***Matriz1***,** *Matriz2***)** ⇒ *matriz*

Devolve uma nova lista que é a *Matriz2* acrescentada ao fim da *Matriz1*. As matrizes têm de ter dimensões de colunas iguais, e a *Matriz2* é acrescentada à *Matriz1* como novas colunas. Não altere *Matriz1* ou *Matriz2*.

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

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

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

## colDim()

Catálogo >

**colDim(***Matriz***)** ⇒ *expressão*

Devolve o número de colunas contidas em *Matriz*.

**Nota:** Consulte também **rowDim()**.

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

## colNorm()

Catálogo >

**colNorm(***Matriz***)** ⇒ *expressão*

Devolve o máximo das somas dos valores absolutos dos elementos nas colunas em *Matriz*.

**Nota:** Os elementos da matriz indefinidos não são permitidos. Consulte também **rowNorm()**.

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

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

## completeSquare()

Catálogo >

**completeSquare(***ExprOrEqn***,** *Var***)** ⇒ *expressão ou equação*

**completeSquare(***ExprOrEqn***,** *Var*^*Power***)** ⇒ *expressão ou equação*

**completeSquare(***ExprOrEqn***,** *Var1 Var2 [ ...]***)** ⇒ *expressão ou equação*

**completeSquare(***ExprOrEqn***, {***Var1 Var2 [ ...]***})** ⇒ *expressão ou equação*

Converte uma expressão polinomial quadrática da forma $a{\cdot}x^2+b{\cdot}x+c$ para a forma $a{\cdot}(x-h)^2+k$

ou

Converte uma equação do 2º grau da forma a·x2+b·x+c=d para a forma a·(x-h)2=k

O primeiro argumento tem de ser uma expressão quadrática ou equação na forma padrão, em relação ao segundo argumento.

O segundo argumento tem de ser um único termo de uma só variável ou um único termo de uma só variável elevado a uma potência racional, por exemplo x, $y^2$ ou $z^{(1/3)}$.

A terceira e quarta expressões de sintaxe para concluir o quadrado nas variáveis *Var1*, *Var2* [,… ]).

completeSquare$\left(x^2+2{\cdot}x+3,x\right)$ $(x+1)^2+2$

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

completeSquare$\left(x^6+2{\cdot}x^3+3,x^3\right)$ $\left(x^3+1\right)^2+2$

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

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

completeSquare$\left(x^2+2{\cdot}x{\cdot}y,x,y\right)$ $(x+y)^2-y^2$

## conj()

Catálogo >

**conj(***Valor1***)** ⇒ *valor*
**conj(***Lista1***)** ⇒ *lista*
**conj(***Matriz1***)** ⇒ *matriz*

Devolve o conjugado complexo do argumento.

**Nota:** Todas as variáveis indefinidas são tratadas como variáveis reais.

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

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

## constructMat()

Catálogo >

**constructMat(***Expr***,***Var1***,***Var2***,***NúmLinhas***,***NúmColunas***)** ⇒ *matriz*

Devolve uma matriz de acordo com os argumentos.

*Expr* é uma expressão nas variáveis *Var1* e *Var2*. Os elementos da matriz resultante são formados através da avaliação de *Expr* para cada valor incrementado de *Var1* e *Var2*.

*Var1* é incrementada automaticamente de **1** a *NúmLinhas*. Em cada linha, *Var2* é incrementada de **1** a *NúmColunas*.

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

Catálogo >

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

**CopyVar** *Var1*, *Var2* copia o valor da variável *Var1* à variável *Var2*, criando *Var2*, se for necessário. A variável *Var1* tem de ter um valor.

Se *Var1* for o nome de uma função definida pelo utilizador existente, copia a definição dessa função para a função *Var2*. A função *Var1* tem de ser definida.

*Var1* tem de cumprir os requisitos de nomeação de variáveis ou tem de ser uma expressão indirecta que se simplifica para um nome de variável que cumpra os requisitos.

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

Catálogo >

**CopyVar** *Var1*., *Var2*. copia todos os membros da *Var1*. grupo de variáveis para a *Var2*. grupo, criando *Var2*. se for necessário.

*Var1*. tem de ser o nome de um grupo de variáveis existentes, como, por exemplo, o da estatística *stat*.*nn* resultados ou variáveis criados com a função **LibShortcut()**. Se *Var2*. já existe, este comando substitui todos os membros comuns a ambos os grupos e adiciona os membros que já não existam. Se um ou mais membros de *Var2*. estiverem bloqueados, todos os membros de *Var2*. ficam inalteráveis.

| | |
|---|---|
| *aa.a*:=45 | 45 |
| *aa.b*:=6.78 | 6.78 |
| *aa.c*:=8.9 | 8.9 |
| getVarInfo() | *aa.a* "NUM" "" <br> *aa.b* "NUM" "" <br> *aa.c* "NUM" "" |
| CopyVar *aa.*,*bb.* | *Done* |
| getVarInfo() | *aa.a* "NUM" "" <br> *aa.b* "NUM" "" <br> *aa.c* "NUM" "" <br> *bb.a* "NUM" "" <br> *bb.b* "NUM" "" <br> *bb.c* "NUM" "" |

## corrMat()

Catálogo >

**corrMat(***Lista1***,** *Lista2* [, …[, *Lista20* ]]**)**

Calcula a matriz de correlação para a matriz aumentada [ *Lista1*, *Lista2*, ..., *Lista20* ].

## cos()

Tecla

**cos(***Valor1***)** ⇒ *valor*
**cos(***Lista1***)** ⇒ *lista*

**cos(***Valor1***)** devolve o co-seno do argumento como um valor.

**cos(***Lista1***)** devolve uma lista de co-senos de todos os elementos na *Lista1*.

**Nota:** O argumento é interpretado como um ângulo express em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual. Pode utilizar °, ᴳ ou ʳ para substituir o modo de ângulo temporariamente.

No modo de ângulo Graus:

| | |
|---|---|
| $\cos\left(\left\{\frac{\pi}{4}\right\}^r\right)$ | 0.707107 |
| $\cos(45)$ | 0.707107 |
| $\cos(\{0,60,90\})$ | {1.,0.5,0.} |

No modo de ângulo Gradianos:

| | |
|---|---|
| $\cos(\{0,50,100\})$ | {1.,0.707107,0.} |

No modo de ângulo Radianos:

| | |
|---|---|
| $\cos\left(\frac{\pi}{4}\right)$ | 0.707107 |
| $\cos(45°)$ | 0.707107 |

## cos()

**Tecla** trig

**cos(***MatrizQuadrada1***)** ⇒ *Matriz quadrada*

Devolve o co-seno da matriz da *MatrizQuadrada1*. Isto não é o mesmo que calcular o co-seno de cada elemento.

Quando uma função escalar f(A) operar na *MatrizQuadrada1* (A), o resultado é calculado pelo algoritmo:

Calcule os valores próprios ($\lambda_i$) e os vectores próprios ($V_i$) de A.

*MatrizQuadrada1* tem de ser diagnolizável. Também não pode ter variáveis simbólicas sem um valor.

Forme as matrizes:

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

$A = X\,B\,X^{-1}$ e $f(A) = X\,f(B)\,X^{-1}$. Por exemplo, $\cos(A) = X\cos(B)\,X^{-1}$ em que:

cos(B) =

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

Todos os cálculos são efectuados com a aritmética de ponto flutuante.

No modo de ângulo Radianos:

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

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

## cos⁻¹()

**Tecla** trig

**cos⁻¹(***Valor1***)** ⇒ *valor*
**cos⁻¹(***Lista1***)** ⇒ *lista*

**cos⁻¹(***Valor1***)** devolve o ângulo cujo co-seno é *Valor1*.

**cos⁻¹(***Lista1***)** devolve uma lista de co-senos inversos de cada elemento de *Lista1*.

**Nota:** O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

**Nota:** Pode introduzir esta função através da escrita de **arccos(…)** no teclado.

**cos⁻¹(***MatrizQuadrada1***)** ⇒ *Matriz quadrada*

Devolve o co-seno inverso da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o co-seno inverso de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Graus:

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

No modo de ângulo Gradianos:

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

No modo de ângulo Radianos:

$$\cos^{-1}(\{0,0.2,0.5\})$$

$$\{1.5708, 1.36944, 1.0472\}$$

No modo de ângulo Radianos e Formato complexo rectangular:

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

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

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

## cosh()

Catálogo >

**cosh(***Valor1***)** ⇒ *valor*
**cosh(***Lista1***)** ⇒ *lista*

**cosh(***Valor1***)** devolve o co-seno hiperbólico do argumento.

**cosh** (*Lista1*) devolve uma lista dos co-senos hiperbólicos de cada elemento de *Lista1*.

**cosh (***MatrizQuadrada1***)** ⇒ *Matriz quadrada*

Devolve o co-seno hiperbólico da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o co-seno hiperbólico de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

$\cosh\left(\left(\frac{\pi}{4}\right)^r\right)$ → 1.74671E19

No modo de ângulo Radianos:

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

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

## cosh⁻¹()

Catálogo >

**cosh⁻¹(***Valor1***)** ⇒ *valor*
**cosh⁻¹(***Lista1***)** ⇒ *lista*

**cosh⁻¹**(*Valor1*) devolve o co-seno hiperbólico inverso do argumento.

**cosh⁻¹**(*Lista1*) devolve uma lista dos co-senos hiperbólicos inversos de cada elemento de *Lista1*.

**Nota:** Pode introduzir esta função através da escrita de **arccosh(** … **)** no teclado.

**cosh⁻¹(***MatrizQuadrada1***)** ⇒ *Matriz quadrada*

Devolve o co-seno hiperbólico inverso da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o co-seno hiperbólico inverso de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

$\cosh^{-1}(1)$ → 0

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

No modo de ângulo Radianos e Formato complexo rectangular:

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

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

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

## cot()

Tecla trig

**cot(***Valor1***)** ⇒ *valor*
**cot(***Lista1***)** ⇒ *lista*

Devolve a co-tangente de *Valor1* ou devolve uma lista das co-tangentes de todos os elementos em *Lista1*.

**Nota:** O argumento é interpretado como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual. Pode utilizar °, ᴳ ou ʳ para substituir o modo de ângulo temporariamente.

**Nota:** Pode introduzir esta função através da escrita de **arccot(** … **)** no teclado.

No modo de ângulo Graus:

$\cot(45)$ → 1

No modo de ângulo Gradianos:

$\cot(50)$ → 1

No modo de ângulo Radianos:

$\cot(\{1,2.1,3\})$ → $\{0.642093,-0.584848,-7.01525\}$

### cot⁻¹()

**Tecla** trig

**cot⁻¹(***Valor1***)** ⇒ *valor*
**cot⁻¹(***Lista1***)** ⇒ *lista*

Devolve o ângulo cuja co-tangente é *Valor1* ou devolve uma lista com as co-tangentes inversas de cada elemento de *Lista1*.

**Nota:** O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

No modo de ângulo Graus:

$\cot^{-1}(1)$ → 45

No modo de ângulo Gradianos:

$\cot^{-1}(1)$ → 50

No modo de ângulo Radianos:

$\cot^{-1}(1)$ → .785398

### coth()

**Catálogo >**

**coth(***Valor1***)** ⇒ *valor*
**coth(***Lista1***)** ⇒ *lista*

Devolve a co-tangente hiperbólica de *Valor1* ou devolve uma lista das co-tangentes hiperbólicas de todos os elementos de *Lista1*.

$\coth(1.2)$ → 1.19954

$\coth(\{1,3.2\})$ → $\{1.31304,1.00333\}$

### coth⁻¹()

**Catálogo >**

**coth⁻¹(***Valor1***)** ⇒ *valor*
**coth⁻¹(***Lista1***)** ⇒ *lista*

Devolve a co-tangente hiperbólica inversa de *Valor1* ou devolve uma lista com as co-tangentes hiperbólicas inversas de cada elemento de *Lista1*.

**Nota:** Pode introduzir esta função através da escrita de **arccoth(**…**)** no teclado.

$\coth^{-1}(3.5)$ → 0.293893

$\coth^{-1}(\{-2,2.1,6\})$ → $\{-0.549306,0.518046,0.168236\}$

### count()

**Catálogo >**

**count(***Valor1ouLista1* [, *Valor2ouLista2* [,…]]**)** ⇒ *valor*

Devolve a contagem acumulada de todos os elementos nos argumentos que se avaliam para valores numéricos.

Cada argumento pode ser uma expressão, valor, lista ou matriz. Pode misturar tipos de dados e utilizar argumentos de várias dimensões.

Para uma lista, matriz ou intervalo de dados, cada elemento é avaliado para determinar se deve ser incluído na contagem.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de qualquer argumento.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

$\text{count}(2,4,6)$ → 3

$\text{count}(\{2,4,6\})$ → 3

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

## countif()

Catálogo >

**countif(***Lista, Critérios***)** ⇒ *valor*

Devolve a contagem acumulada de todos os elementos em *Lista* que cumpram os *critérios* especificados.

*Critérios* podem ser:

- Um valor, uma expressão ou uma cadeia. Por exemplo, **3** conta apenas aqueles elementos em *Lista* que se simplificam para o valor 3.
- Uma expressão booleana com o símbolo **?** como um identificador para cada elemento. Por exemplo, **?<5** conta apenas aqueles elementos em *Lista* inferiores a 5.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de *Lista*.

Os elementos (nulos) vazios da lista são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

**Nota:** Consulte também **sumIf()**, página 103 e **frequency()**, página 39.

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

Conta o número de elementos igual a 3.

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

Conta o número de elementos igual a "def."

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

Conta 1 e 3.

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

Conta 3, 5, e 7.

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

Conta 1, 3, 7 e 9.

## cPolyRoots()

Catálogo >

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

**cPolyRoots(***ListaDeCoeficientes***)** ⇒ *lista*

A primeira sintaxe, **cPolyRoots(***Poly***,***Var***)**, devolve uma lista de raízes complexas do polinómio *Poly* na variável *Var*.

*Poly* tem de ser um polinómio na forma expandida. Não utilize formatos não expandidos, como, por exemplo, $y^2 \cdot y+1$ ou $x \cdot x+2 \cdot x+1$

A segunda sintaxe, **cPolyRoots(***ListaDeCoeficientes***)**, devolve uma lista de raízes complexas para os coeficientes em *ListaDeCoeficientes*.

**Nota:** Consulte também **polyRoots()**, página 76.

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

cPolyRoots($y^3+1,y$) {-1,0.5−0.866025·*i*,0.5+0.866025·*i*}

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

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

## crossP()

Catálogo >

**crossP(***Lista1***,** *Lista2***)** ⇒ *lista*

Devolve o produto cruzado de *Lista1* e *Lista2* como uma lista.

*Lista1* e *Lista2* têm de ter dimensões iguais e a dimensão tem de ser 2 ou 3.

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

Devolve um vector da linha ou coluna (dependendo dos argumentos) que é o produto cruzado de *Vector1* e *Vector2*.

*Vector1* e *Vector2* têm de ser vectores de linhas ou ambos têm de ser vectores de colunas. Ambos os vectores têm de ter dimensões iguais e a dimensão tem de ser 2 ou 3.

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

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

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

### csc()

**Tecla** trig

**csc(***Valor1***)** ⇒ *valor*
**csc(***Lista1***)** ⇒ *lista*

Devolve a co-secante de *Valor1* ou devolve uma lista com as co-secantes de todos os elementos em *Lista1*.

No modo de ângulo Graus:

$\csc(45)$ 1.41421

No modo de ângulo Gradianos:

$\csc(50)$ 1.41421

No modo de ângulo Radianos:

$\csc\left\{\left\{1,\frac{\pi}{2},\frac{\pi}{3}\right\}\right\}$ $\{1.1884,1.,1.1547\}$

### csc$^{-1}$()

**Tecla** trig

**csc$^{-1}$(***Valor1***)** ⇒ *valor*
**csc$^{-1}$(***Lista1***)** ⇒ *lista*

Devolve o ângulo cuja co-secante é *Valor1* ou devolve uma lista com as co-secantes inversas de cada elemento de *Lista1*.

**Nota:** O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

**Nota:** Pode introduzir esta função através da escrita de **arccsc(**…**)** no teclado.

No modo de ângulo Graus:

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

No modo de ângulo Gradianos:

$\csc^{-1}(1)$ 100

No modo de ângulo Radianos:

$\csc^{-1}(\{1,4,6\})$ $\{1.5708,0.25268,0.167448\}$

### csch()

**Catálogo >**

**csch(***Valor1***)** ⇒ *valor*
**csch(***Lista1***)** ⇒ *lista*

Devolve a co-secante hiperbólica de *Valor1* ou devolve uma lista das co-secantes hiperbólicas de todos os elementos de *Lista1*.

$\operatorname{csch}(3)$ 0.099822

$\operatorname{csch}(\{1,2.1,4\})$ $\{0.850918,0.248641,0.036644\}$

### csch$^{-1}$()

**Catálogo >**

**csch$^{-1}$(***Valor***)** ⇒ *valor*
**csch$^{-1}$(***Lista1***)** ⇒ *lista*

Devolve a co-secante hiperbólica inversa de *Valor1* ou devolve uma lista com as cosesecantes hiperbólicas inversas de cada elemento de *Lista1*.

**Nota:** Pode introduzir esta função através da escrita de **arccsch(**…**)** no teclado.

$\operatorname{csch}^{-1}(1)$ 0.881374

$\operatorname{csch}^{-1}(\{1,2.1,3\})$ $\{0.881374,0.459815,0.32745\}$

## CubicReg

Catálogo >

**CubicReg** *X*, *Y*[, [*Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão polinomial cúbica $y = a \cdot x^3+b \cdot x^2+c \cdot x+d$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros ≥ 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: $a \cdot x^3+b \cdot x^2+c \cdot x+d$ |
| stat.a, stat.b, stat.c, stat.d | Coeficientes de regressão |
| stat.$R^2$ | Coeficiente de determinação |
| stat.Resid | Resíduos da regressão |
| stat.XReg | Lista de pontos de dados na *Lista X* modificada utilizada na regressão com base em restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de pontos de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

## cumulativeSum()

Catálogo >

**cumulativeSum(***Lista1***)** ⇒ *lista*

Devolve uma lista das somas acumuladas dos elementos em *Lista1*, começando no elemento 1.

**cumulativeSum(***Matriz1***)** ⇒ *matriz*

Devolve uma matriz das somas cumulativas dos elementos em *Matriz1*. Cada elemento é a soma cumulativa da coluna de cima a baixo.

Um elemento (nulo) vazio em *Lista1* ou em *Matriz1* produz um elemento nulo na matriz ou lista resultante. Para mais informações sobre os elementos vazios, consulte a página 136.

$\text{cumulativeSum}(\{1,2,3,4\}) \quad \{1,3,6,10\}$

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

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

### Cycle

Catálogo >

**Cycle**

Transfere o controlo imediatamente para a iteração seguinte do ciclo actual (**For**, **While** ou **Loop**).

**Cycle** não é permitido fora das três estruturas em espiral (**For**, **While** ou **Loop**).

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

Lista de funções que soma os números inteiros de 1 a 100 ignorando 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

Catálogo >

*Vector* **▸Cylind**

**Nota:** Pode introduzir este operador através da escrita de **@>Cylind** no teclado do computador.

Apresenta o vector da linha ou coluna em forma cilíndrica [r, ∠θ, z].

*Vector* tem de ter exactamente três elementos. Pode ser uma linha ou coluna.

```
[2  2  3]▸Cylind
                [2.82843  ∠0.785398  3.]
```

## D

### dbd()

Catálogo >

**dbd(***data1*,*data2***)** ⇒ *valor*

Devolve o número de dias entre *data1* e *data2* com o método de contagem de dias actual.

*data1* e *data2* podem ser números ou listas de números no intervalo das datas no calendário padrão. Se *data1* e *data2* forem listas, têm de ter o mesmo comprimento.

*data1* e *data2* têm de estar entre os anos 1950 e 2049.

Pode introduzir as datas num de dois formatos. A colocação decimal diferencia-se entre os formatos de data.

MM.AAAA (formato utilizado nos Estados Unidos)
DDMM.AA (formato utilizado na 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

**Catálogo >**

*Expr1* **▸DD** ⇒ *valor*
*Lista1* **▸DD** ⇒ *lista*
*Matriz1* **▸DD** ⇒ *matriz*

**Nota:** Pode introduzir este operador através da escrita de **@>DD** no teclado do computador.

Devolve o decimal equivalente do argumento expresso em graus. O argumento é um número, uma lista ou uma matriz que é interpretada pela definição do modo ângulo em gradianos, radianos ou graus.

No modo de ângulo Graus:

$(1.5°)\blacktriangleright DD$ → $1.5°$

$(45°22'14.3")\blacktriangleright DD$ → $45.3706°$

$(\{45°22'14.3",60°0'0"\})\blacktriangleright DD$ → $\{45.3706°,60°\}$

No modo de ângulo Gradianos:

$1\blacktriangleright DD$ → $\frac{9}{10}^{g}$

No modo de ângulo Radianos:

$(1.5)\blacktriangleright DD$ → $85.9437°$

## ▸Decimal

**Catálogo >**

*Número1* **▸Decimal** ⇒ *valor*
*Lista1* **▸Decimal** ⇒ *valor*
*Matriz1* **▸Decimal** ⇒ *valor*

**Nota:** Pode introduzir este operador através da escrita de **@>Decimal** no teclado do computador.

Mostra o argumento em forma decimal. Este operador só pode ser utilizado no fim da linha de entrada.

$\frac{1}{3}\blacktriangleright Decimal$ → $0.333333$

## Define

**Catálogo >**

**Define** *Var* = *Expressão*
**Define** *Função*(*Parâm1*, *Parâm2*, ...) = *Expressão*

Define a variável *Var* ou a função *Função* definida pelo utilizador.

Os parâmetros como, por exemplo, *Parâm1*, fornecem marcadores para argumentos de passagem para a função. Quando chamar uma função definida pelo utilizador, tem de fornecer os argumentos (por exemplo, valores ou variáveis) correspondentes aos parâmetros. Quando chamada, a função avalia a *Expressão* com os argumentos fornecidos.

*Var* e *Função* não podem ter o nome de uma variável do sistema, um comando ou uma função integrada.

**Nota:** Esta forma de **Define** é equivalente à execução da expressão: *expressão* ➞ *Função*(*Parâm1*,*Parâm2*).

Define $g(x,y)=2\cdot x-3\cdot y$ → *Done*

$g(1,2)$ → $-4$

$1\to a:\ 2\to b:\ g(a,b)$ → $-4$

Define $h(x)=when(x<2,2\cdot x-3,-2\cdot x+3)$ → *Done*

$h(-3)$ → $-9$

$h(4)$ → $-5$

## Define

Catálogo >

**Define** *Função*(*Parâm1*, *Parâm2*, ...) **= Func**
*Bloco*
**EndFunc**

**Define** *Programa*(*Parâm1*, *Parâm2*, ...) **= Prgm**
*Bloco*
**EndPrgm**

Desta forma, o programa ou a função definida pelo utilizador pode executar um bloco de várias afirmações.

*Bloco* pode ser uma afirmação ou uma série de afirmações em linhas separadas. *O bloco* pode também incluir expressões e instruções (como, por exemplo, **If**, **Then**, **Else** e **For**).

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições em diferentes linhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

**Nota:** Consulte também **Define LibPriv**, página 28, e **Define LibPub**, página 29.

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

g(3,-7)                                  3
```

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

g(3,-7)
                          3 greater than -7
                                      Done
```

## Define LibPriv

Catálogo >

**Define LibPriv** *Var* **=** *Expressão*

**Define LibPriv** *Função*(*Parâm1*, *Parâm2*, ...) **=** *Expressão*

**Define LibPriv** *Função*(*Parâm1*, *Parâm2*, ...) **= Func**
*Bloco*
**EndFunc**

**Define LibPriv** *Programa*(*Parâm1*, *Parâm2*, ...) **= Prgm**
*Bloco*
**EndPrgm**

Funciona da mesma forma que **Define**, excepto com um programa, uma função ou uma variável da biblioteca privada. As funções e os programas privados não aparecem no Catálogo.

**Nota:** Consulte também **Define**, página 27, e **Define LibPub**, página 29.

## Define LibPub

Catálogo >

**Define LibPub** *Var* **=** *Expressão*

**Define LibPub** *Função*(*Parâm1*, *Parâm2*, ...) **=** *Expressão*

**Define LibPub** *Função*(*Parâm1*, *Parâm2*, ...) **= Func**
*Bloco*
**EndFunc**

**Define LibPub** *Programa*(*Parâm1*, *Parâm2*, ...) **= Prgm**
*Bloco*
**EndPrgm**

Funciona da mesma forma que **Define**, excepto com um programa, uma função ou uma variável da biblioteca pública. As funções e os programas públicos aparecem no Catálogo depois de guardar e actualizar a biblioteca.

**Nota:** Consulte também **Define**, página 27, e **Define LibPriv**, página 28.

## deltaList()

Consulte **ΔList()**, página 55.

## DelVar

Catálogo >

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

**DelVar** *Var***.**

Elimina a variável ou o grupo de variáveis especificado da memória.

Se uma ou mais variáveis estiverem bloqueadas, este comando mostra uma mensagem de erro e só elimina as variáveis desbloqueadas. Consulte **unLock**, página 113.

$2 \rightarrow a$ 2

$(a+2)^2$ 16

DelVar $a$ *Done*

$(a+2)^2$ "Error: Variable is not defined"

**DelVar** *Var***.** elimina todos os membros da *Var***.** grupo de variáveis (como, por exemplo, as estatísticas *stat***.***nn* resultados ou variáveis criados com a função **LibShortcut()**). O ponto (**.**) nesta forma do comando **DelVar** limita-o à eliminação do grupo de variáveis; a variável simples *Var* não é afectada.

*aa.a*:=45 45

*aa.b*:=5.67 5.67

*aa.c*:=78.9 78.9

getVarInfo() $\begin{bmatrix} aa.a & "NUM" & "\square" \\ aa.b & "NUM" & "\square" \\ aa.c & "NUM" & "\square" \end{bmatrix}$

DelVar *aa.* *Done*

getVarInfo() "NONE"

## delVoid()

Catálogo >

**delVoid(***Lista1***)** ⇒ *lista*

Devolve uma lista com o conteúdo de *Lista1* com todos os elementos (nulos) vazios removidos.

Para mais informações sobre os elementos vazios, consulte a página 136.

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

## det()

Catálogo >

**det(***MatrizQuadrada*[**,** *Tolerância*]**)** ⇒ *expressão*

Apresenta o determinante de *MatrizQuadrada*.

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior à *Tolerância*. Esta tolerância é utilizada apenas se a matriz tiver entradas de ponto flutuante e não contiver nenhuma variável simbólica sem nenhum valor atribuído. Caso contrário, *Tolerância* é ignorada.

- Se utilizar ctrl enter ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética de ponto flutuante.
- Se *Tolerância* for omitida ou não utilizada, a tolerância predefinida é calculada da seguinte forma:

  5E−14 · **max(dim(***MatrizQuadrada***))** · **rowNorm(***MatrizQuadrada***)**

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

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

$\det(mat1)$ → $0$

$\det(mat1,.1)$ → $1.\text{E}20$

## diag()

Catálogo >

**diag(***Lista***)** ⇒ *matriz*
**diag(***MatrizLinha***)** ⇒ *matriz*
**diag(***MatrizColuna***)** ⇒ *matriz*

Devolve uma matriz com os valores da matriz ou da lista de argumentos na diagonal principal.

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

**diag(***MatrizQuadrada***)** ⇒ *MatrizLinha*

Devolve uma matriz da linha com elementos da diagonal principal de *MatrizQuadrada*.

*MatrizQuadrada* tem de ser quadrada.

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

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

## dim()

Catálogo >

**dim(***Lista***)** ⇒ *número inteiro*

Devolve a dimensão de *Lista*.

$\dim(\{0,1,2\})$ → $3$

**dim(***Matriz***)** ⇒ *lista*

Devolve as dimensões da matriz como uma lista de dois elementos {linhas, colunas}.

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

**dim(***Cadeia***)** ⇒ *número inteiro*

Devolve o número de caracteres contidos na cadeia de caracteres *Cadeia*.

$\dim(\text{"Hello"})$ → $5$

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

## Disp

Catálogo >

**Disp** [ *exprOuCadeia1* ] [**,** *exprOuCadeia2* ] ...

Mostra os argumentos no histórico da Calculadora. Os argumentos são apresentados em sucessão com espaços pequenos como separadores.

Útil principalmente em programas e funções para garantir a visualização de cálculos intermédios.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
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

Catálogo >

*Valor* **▸DMS**
*Lista* **▸DMS**
*Matriz* **▸DMS**

**Nota:** Pode introduzir este operador através da escrita de **@>DMS** no teclado do computador.

Interpreta o argumento como um ângulo e mostra o número DMS equivalente (DDDDDD °MM ' SS.ss ''). Consulte °, ', '' na página 132 para o formato DMS (grau, minutos, segundos).

**Nota:** ▸DMS converterá de radianos para graus quando utilizado em modo de radianos. Se a entrada for seguida por um símbolo de grau °, não ocorrerá nenhuma conversão. Pode utilizar o **▸DMS** apenas no fim de uma linha de entrada.

No modo de ângulo Graus:

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

## dotP()

Catálogo >

**dotP(***Lista1***,** *Lista2***)** ⇒ *expressão*

Devolve o produto do "ponto" de duas listas.

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

Devolve o produto do "ponto" de dois vectores.

Ambos têm de ser vectores da linha ou da coluna.

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

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

# E

## e^()

**Tecla** $e^x$

**e^(***Valor1***)** $\Rightarrow$ *valor*

Devolve e elevado à potência *Valor1*.

**Nota:** Consulte também e **modelo do expoente**, página 2.

**Nota:** Premir $e^x$ para ver e ^( é diferente de premir o carácter **E** no teclado.

Pode introduzir um número complexo na forma polar $re^{i\theta}$. No entanto, utilize esta forma apenas no modo de ângulo Radianos; causa um erro de domínio no modo de ângulo Graus ou Gradianos.

$e^1$ → 2.71828

$e^{3^2}$ → 8103.08

**e^(***Lista1***)** $\Rightarrow$ *lista*

Devolve e elevado à potência de cada elemento em *Lista1*.

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

**e^(***MatrizQuadrada1***)** $\Rightarrow$ *MatrizQuadrada*

Devolve a matriz exponencial de *MatrizQuadrada1*. Isto não é o mesmo que calcular e elevado à potência de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

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

## eff()

**Catálogo >**

**eff(***TaxaNominal*,*CpY***)** $\Rightarrow$ *valor*

Função financeira que converte a taxa de juro nominal *TaxaNominal* para uma taxa efectiva anual, dando *CpY* como o número de período compostos por ano.

*TaxaNominal* tem de ser um número real e *CpY* tem de ser um número real > 0.

**Nota:** Consulte também **nom()**, página 69.

eff(5.75,12) → 5.90398

## eigVc()

**Catálogo >**

**eigVc(***MatrizQuadrada***)** $\Rightarrow$ *matriz*

Devolve uma matriz com os vectores próprios para uma *MatrizQuadrada* real ou complexa, em que cada coluna do resultado corresponde a um valor próprio. Não se esqueça de que um vector próprio não é único; pode ser dimensionado por qualquer factor constante. Os vectores próprios são normalizados, significando que se $V = [x_1, x_2, \ldots, x_n]$:

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

*MatrizQuadrada* é primeiro equilibrada com tranformações de similaridade até as normas das colunas e linhas estarem o mais perto possível do mesmo valor. A *MatrizQuadrada* é reduzida para a forma Hessenberg superior e os vectores próprios são calculados através de uma factorização Schur.

No Formato complexo rectangular:

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

eigVc(*m1*)

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

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

## eigVl()

**Catálogo >**

**eigVl(***MatrizQuadrada***)** ⇒ *lista*

Devolve uma lista dos valores próprios de uma *MatrizQuadrada* real ou complexa.

*MatrizQuadrada* é primeiro equilibrada com tranformações de similaridade até as normas das colunas e linhas estarem o mais perto possível do mesmo valor. A *MatrizQuadrada* é reduzida para a forma Hessenberg superior e os valores próprios são calculados a partir da matriz Hessenberg superior.

No modo de formato complexo rectangular:

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

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

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

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

## Else

Consulte If, página 45.

## ElseIf

**Catálogo >**

**Se** *ExprBooleana1*
*Block1*
**ElseIf** *BooleanExpr2*
*Block2*
⋮
**ElseIf** *ExprBooleanaN*
*BlockN*
EndIf
⋮

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
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

Consulte For, página 38.

## EndFunc

Consulte Func, página 40.

## EndIf

Consulte If, página 45.

## EndLoop

Consulte Loop, página 60.

## EndPrgm

Consulte Prgm, página 78.

## EndTry

Consulte Try, página 108.

## euler()

**Catálogo >**

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

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

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

Utiliza o método de Euler para resolver o sistema

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

com *depVar*(*Var0*)=*depVar0* no intervalo [*Var0*,*VarMax*]. Apresenta uma matriz cuja primeira linha define os valores de saída *Var* e cuja segunda linha define o valor da primeira componente da solução nos valores *Var* correspondentes, e assim por diante.

*Expr* é o lado direito que define a equação diferencial ordinária (EDO).

*SystemOfExpr* é o sistema de lados direitos que definem o sistema de EDOs (corresponde à ordem de variáveis dependentes em *ListOfDepVars*).

*ListOfExpr* é uma lista de segundos membros que definem o sistema de EDOs (corresponde à ordem de variáveis dependentes em *ListOfDepVars*).

*Var* é a variável independente.

*ListOfDepVars* é uma lista de variáveis dependentes.

{*Var0*, *VarMax*} é uma lista de dois elementos que informa a função para integrar de *Var0* a *VarMax*.

*ListOfDepVars0* é uma lista de valores iniciais para variáveis dependentes.

*VarStep* é um número diferente de zero tal como **sign(***VarStep***)** = **sign(***VarMax*-*Var0***)** e as soluções regressam a *Var0*+i·*VarStep* para todos os i=0,1,2,… tal como *Var0*+i·*VarStep* está em [*var0*,*VarMax*] (pode não existir um valor de solução em *VarMax*).

*eulerStep* é um número inteiro positivo (passa para 1) que define o número de passos Euler entre os valores de saída. O tamanho de passo real utilizado pelo método Euler é *VarStep* / *eulerStep*.

Equação diferencial:
y'=0.001*y*(100-y) e y(0)=10

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

Para ver o resultado completo, prima ▲ e, de seguida, utilize ◀ e ▶ para mover o cursor.

Sistema de equações:
$\begin{cases} y1' = -y1 + 0.1\cdot y1\cdot y2 \\ y2' = 3\cdot y2 - y1\cdot y2 \end{cases}$
com *y1*(0)=2 e *y2*(0)=5

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

## Exit

**Catálogo >**

**Exit**

Sai do bloco **For**, **While** ou **Loop** actual.

**Exit** não é permitido fora das três estruturas circulares (**For**, **While** ou **Loop**).

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo [↵] em vez de [enter] no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

Listagem de funções:

```
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()

**Tecla** $e^x$

**exp(***Valor1***)** ⇒ *valor*

Devolve **e** elevado à potência *Valor1*.

**Nota:** Consulte também **e** modelo do expoente, página 2.

Pode introduzir um número complexo na forma polar $re^{i\theta}$. No entanto, utilize esta forma apenas no modo de ângulo Radianos; causa um erro de domínio no modo de ângulo Graus ou Gradianos.

$e^1$ → 2.71828

$e^{3^2}$ → 8103.08

**exp(***Lista1***)** ⇒ *lista*

Devolve **e** elevado à potência de cada elemento em *Lista1*.

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

**exp(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve a matriz exponencial de *MatrizQuadrada1*. Isto não é o mesmo que calcular **e** elevado à potência de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

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

## expr()

**Catálogo >**

**expr(***Cadeia***)** ⇒ *expressão*

Devolve a cadeia de caracteres contidos em *Cadeia* como uma expressão e executa-a imediatamente.

"Define cube(x)=x^3" → *funcstr*

"Define cube(x)=x^3"

expr(*funcstr*) → *Done*

*cube*(2) → 8

## ExpReg

**Catálogo >**

**ExpReg** *X*, *Y* [, [*Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão exponencial $y = a \cdot (b)^x$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: $a \cdot (b)^x$ |
| stat.a, stat.b | Parâmetros da regressão |
| stat.$r^2$ | Coeficiente de determinação linear para dados transformados |

| Variável de saída | Descrição |
|---|---|
| stat.r | Coeficiente de correlação para dados transformados (x, ln(y)) |
| stat.Resid | Resíduos associados ao modelo exponencial |
| stat.ResidTrans | Residuais associados ao ajuste linear de dados transformados |
| stat.XReg | Lista de pontos de dados na *Lista X* modificada utilizada na regressão com base em restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de pontos de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

# *F*

## factor()

**Catálogo >**

**factor(***NúmeroRacional***)** devolve o número racional em primos. Para números compostos, o tempo de cálculo cresce exponencialmente com o número de dígitos no segundo maior factor. Por exemplo, a decomposição em factores de um número inteiro de 30 dígitos pode demorar mais de um dia e a decomposição em factores de um número de 100 dígitos pode demorarar mais de um século.

Para parar um cálculo manualmente,

- **Windows®:** Prima continuamente a tecla **F12** e prima repetidamente **Enter**.
- **Macintosh®:** Prima continuamente a tecla **F5** e prima repetidamente **Enter**.
- **Unidade portátil:** Prima continuamente a tecla [on] e prima repetidamente [enter].

Se quiser apenas determinar se um número é primo, utilize **isPrime()**. É muito mais rápido, em especial, se o *NúmeroRacional* não for primo e o segundo maior factor tiver mais de cinco dígitos.

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

## FCdf()

**Catálogo >**

**FCdf(***LimiteInferior***,** *LimiteSuperior***,** *dfNumer***,** *dfDenom***)**
⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

**FCdf(***LimiteInferior***,** *LimiteSuperior***,** *dfNumer***,** *dfDenom***)**
⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

Calcula a probabilidade da distribuição F entre *LimiteInferior* e *LimiteSuperior* para o *dfNumer* (graus de liberdade) e *dfDenom* especificados.

Para P($X \le$ *LimiteSuperior*), definir *LimiteInferior* = 0.

## Fill

Catálogo >

**Fill** *Valor, VarMatriz* ⇒ *matriz*

Substitui cada elemento na variável *VarMatriz* por *Valor*.

*matrixVar* já tem de existir.

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

$$\text{Fill } 1.01, amatrix \qquad Done$$

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

**Fill** *Valor, VarLista* ⇒ *lista*

Substitui cada elemento na variável *VarLista* por *Valor*.

*VarLista* já tem de existir.

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

$$\text{Fill } 1.01, alist \qquad Done$$

$$alist \qquad \{1.01,1.01,1.01,1.01,1.01\}$$

## FiveNumSummary

Catálogo >

**FiveNumSummary** *X*[,[*Freq*][,*Categoria*,*Incluir*]]

Fornece uma versão abreviada da estatística de 1 variável na lista *X*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

*X* representa uma lista de dados.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada valor *X* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos para os valores *X* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Um elemento (nulo) vazio em qualquer das listas *X*, *Freq* ou *Category* resulta num nulo para o elemento correspondente de todas essas listas. Para mais informações sobre os elementos vazios, consulte a página 136.

| Variável de saída | Descrição |
|---|---|
| stat.MinX | Mínimo dos valores x |
| stat.$Q_1$X | 1º quartil de x |
| stat.MedianX | Mediana de x |
| stat.$Q_3$X | 3º quartil de x |
| stat.MaxX | Máximo dos valores x |

## floor()

Catálogo >

**floor(***Valor1***)** ⇒ *número inteiro*

Devolve o maior número inteiro que é ≤ o argumento. Esta função é idêntica a **int()**.

O argumento pode ser um número complexo ou real.

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

## floor()

Catálogo >

**floor(***Lista1***)** ⇒ *lista*
**floor(***Matriz1***)** ⇒ *matriz*

Devolve uma lista ou matriz do floor de cada elemento.

**Nota:** Consulte também **ceiling()** e **int()**.

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

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

## For

Catálogo >

**For** *Var*, *Baixo*, *Alto* [, *Passo* ]
*Bloco*
**EndFor**

Executa as declarações em *Bloco* iteravitamente para cada valor de *Var*, de *Baixo* para *Alto*, em incrementos de *Passo*.

*Var* não tem de ser uma variável do sistema.

*Passo* pode ser positivo ou negativo. O valor predefinido é 1.

*Bloco* pode ser uma declaração ou uma série de declarações separadas pelo carácter ":".

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
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()

Catálogo >

**format(***Valor* [, *CadeiaFormato* ]**)** ⇒ *cadeia*

Devolve *Valor* como uma cadeia de caracteres com base no modelo do formato.

*CadeiaFormato* é uma cadeia e tem de estar na forma: "F[n]", "S[n]", "E[n]", "G[n][c]", em que [ ] indica porções opcionais.

F[n]: Formato fixo. n é o número de dígitos para visualizar o ponto decimal.

S[n]: Formato científico. n é o número de dígitos para visualizar o ponto decimal.

E[n]: Formato de engenharia. n é o número de dígitos após o primeiro dígito significante. O exponente é ajustado para um múltiplo de três e o ponto decimal é movido para a direita zero, um ou dois dígitos.

G[n][c]: Igual ao formato fixo mas também separa os dígitos à esquerda da raiz em grupos de três. c especifica o carácter do separador de grupos e predefine para uma vírgula. Se c for um ponto, a raiz será apresentada como uma vírgula.

[Rc]: Qualquer um dos especificadores acima pode ser sufixado com o marcador de raiz Rc, em que c é um carácter que especifica o que substituir pelo ponto da raiz.

| | |
|---|---|
| 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()

Catálogo >

**fPart(***Expr1***)** ⇒ *expressão*
**fPart(***Lista1***)** ⇒ *lista*
**fPart(***Matriz1***)** ⇒ *matriz*

Devolve a parte fraccionária do argumento.

Para uma lista ou matriz, devolve as partes fraccionárias dos elementos.

O argumento pode ser um número complexo ou real.

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

## FPdf()

**Catálogo >**

**FPdf(***ValX***,** *dfNumer***,** *dfDenom***)** ⇒ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula a probabilidade da distribuição F no *ValX* para o *dfNumer* (graus de liberdade) e o *dfDenom* especificados.

## freqTable▸list()

**Catálogo >**

**freqTable▸list(***Lista1***,***ListaNúmerosInteirosFreq***)** ⇒ *lista*

Apresenta uma lista com os elementos de *Lista1* expandida de acordo com as frequências em *ListaNúmerosInteirosFreq*. Esta função pode ser utilizada para construir uma tabela de frequência para a aplicação Dados e Estatística.

*Lista1* pode ser qualquer lista válida.

*ListaNúmerosInteirosFreq* tem de ter a mesma dimensão da *Lista1* e só deve conter elementos de números inteiros não negativos. Cada elemento especifica o número de vezes que o elemento de *Lista1* correspondente é repetido na lista de resultados. Um valor de zero exclui o elemento de *Lista1* correspondente.

**Nota:** Pode introduzir esta função através da escrita de **`freqTable@>list(`**...**`)`** no teclado do computador.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

freqTable▸list({1,2,3,4},{1,4,3,1}) → {1,2,2,2,2,3,3,3,4}

freqTable▸list({1,2,3,4},{1,4,0,1}) → {1,2,2,2,2,4}

## frequency()

**Catálogo >**

**frequency(***Lista1***,***Listabins***)** ⇒ *lista*

Devolve uma lista que contém as contagens dos elementos em *Lista1*. As contagens são baseadas em intervalos (bins) definidos em *Listabins*.

Se *Listabins* for {b(1), b(2), …, b(n)}, os intervalos especificados são {**?**≤ b(1), b(1)<**?**≤ b(2),…,b(n-1)<**?**≤ b(n), b(n)>**?**}. A lista resultante é um elemento maior que *Listabins*.

Cada elemento do resultado corresponde ao número de elementos de *Lista1* que estão no intervalo desse lote. Expresso em termos da função **countIf()**, o resultado é { countIf(list, **?**≤ b(1)), countIf(lista, b(1)<**?**≤ b(2)), …, countIf(lista, b(n-1)<**?**≤ b(n)), countIf(lista, b(n)>**?**)}.

Elementos de *Lista1* que não podem ser "colocados num lote" são ignorados.

Elementos de *Lista1* que não podem ser "colocados num lote" são ignorados. Os elementos (nulos) vazios também são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de ambos os argumentos.

**Nota:** Consulte também **countIf()**, página 23.

*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**}

Explicação do resultado:
**2** elementos da *Lista de dados* são ≤ 2.5
**4** elementos da *Lista de dados* são >2.5 e ≤ 4.5
**3** elementos da *Lista de dados* são >4.5
O elemento "hello" é uma cadeia e não pode ser colocado em nenhum lote definido.

## FTest_2Samp

**Catálogo >**

**FTest_2Samp** *Lista1*, *Lista2* [, *Freq1* [, *Freq2* [, *Hipótese* ]]]
**FTest_2Samp** *Lista1*, *Lista2* [, *Freq1* [, *Freq2* [, *Hipótese* ]]]

(Entrada da lista de dados)

**FTest_2Samp** *sx1*, *n1*, *sx2*, *n2* [, *Hipótese*]
**FTest_2Samp** *sx1*, *n1*, *sx2*, *n2* [, *Hipótese*]

(Entrada estatística do resumo)

Efectua um teste F de duas amostras. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

ou $H_a$: $\sigma 1 > \sigma 2$, defina *Hipótese*>0
Para $H_a$: $\sigma 1 \neq \sigma 2$ (predefinição), defina *Hipótese* =0
Para $H_a$: $\sigma 1 < \sigma 2$, defina *Hipótese*<0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.F | Estatística F calculada para a sequência de dados |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.dfNumer | graus de liberdade do "numerador" = n1-1 |
| stat.dfDenom | graus de liberdade do "denominador"= n2-1 |
| stat.sx1, stat.sx2 | Desvios padrão da amostra das sequências de dados em *Lista 1* e *Lista 2* |
| stat.x1_bar<br>stat.x2_bar | Médias da amostra das sequência de dados em *Lista 1* e *Lista 2* |
| stat.n1, stat.n2 | Tamanho das amostras |

## Func

**Catálogo >**

**Func**
*Bloco*
**EndFunc**

Modelo para criar uma função definida pelo utilizador.

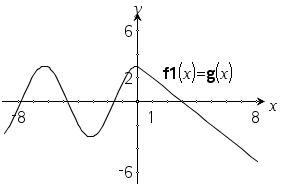
*Bloco* pode ser uma declaração, uma série de declarações separadas pelo carácter ":" ou uma série de declarações em linhas separadas. A função pode utilizar a função **Return** para devolver um resultado específicos.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

Definir uma função por ramos:

Define $g(x)$=Func — *Done*
If $x<0$ Then
Return $3\cdot\cos(x)$
Else
Return $3-x$
EndIf
EndFunc

Resultado do gráfico g(x)



# G

## gcd()

**Catálogo >**

**gcd(***Valor1*, *Valor2***)** ⇒ *expressão*

Devolve o máximo divisor comum dos dois argumentos. O **gcd** de duas fracções é o **gcd** dos numeradores divididos pelo **lcm** dos denominadores.

No modo Auto ou Aproximado, o **gcd** dos números do ponto flutuante fraccionária é 1.0.

$\gcd(18,33)$ → 3

**gcd(***Lista1*, *Lista2***)** ⇒ *lista*

Devolve os máximos divisores comuns dos elementos correspondentes em *Lista1* e *Lista2*.

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

**gcd(***Matriz1*, *Matriz2***)** ⇒ *matriz*

Devolve os máximos divisores comuns dos elementos correspondentes em *Matriz1* e *Matriz2*.

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

## geomCdf()

**Catálogo >**

**geomCdf(***p*,*LimiteInferior*,*LimiteSuperior***)** ⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

**geomCdf(***p*,*LimiteSuperior***)** para P(1≤X≤*LimiteSuperior*) ⇒ *número* se *LimiteSuperior* for um número, *lista* se *LimiteSuperior* for uma lista

Calcula uma probabilidade geométrica cumulativa do *LimiteInferior* ao *LimiteSuperior* com a probabilidade de sucesso especificada *p*.

Para P(X ≤ *LimiteSuperior*), defina *LimiteInferior* = 1.

## geomPdf()

**Catálogo >**

**geomPdf(***p*, *ValX***)** ⇒ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula uma probabilidade em *ValX*, o número da tentativa em que ocorre o primeiro sucesso, para a distribuição geométrica discreta com a probabilidade de sucesso especificada p.

## getDenom()

**Catálogo >**

**getDenom(***Fracção1***)** ⇒ *valor*

Transforma o argumento numa expressão que tem um denominador comum simplificado e, em seguida, devolve o denominador.

$x:=5: y:=6$ → 6

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

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

$\text{getDenom}\left(\frac{1}{x}+\frac{y^2+y}{y^2}\right)$ → 30

## getLangInfo()

Catálogo >

**getLangInfo()** ⇒ *abreviatura*

Apresenta uma abreviatura do nome do idioma activo. Por exemplo, pode utilizá-lo num programa ou função para determinar o idioma actual.

Inglês = "en"
Dinamarquês = "da"
Alemão = "de"
Finlandês = "fi"
Francês = "fr"
Italiano = "it"
Holandês = "nl"
Flamengo = "nl_BE"
Norueguês = "no"
Português = "pt"
Espanhol = "es"
Sueco = "sv"

| | |
|---|---|
| getLangInfo() | "en" |

## getLockInfo()

Catálogo >

**getLockInfo**(*Var*) ⇒ *valor*

Devolve o estado de bloqueio/desbloqueio actual da variável *Var*.

*valor* =**0**: *Var* está desbloqueada ou não existe.
*valor* =**1**: *Var* está bloqueada e não pode ser modificada nem eliminada.

Consulte **Lock**, página 57, e **unLock**, página 113.

| | |
|---|---|
| *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()

Catálogo >

**getMode(***NúmeroInteiroNomeModo***)** ⇒ *valor*
**getMode(0)** ⇒ *lista*

**getMode(***NúmeroInteiroNomeModo***)** devolve um valor que representa a definição actual do modo *NúmeroInteiroNomeModo*.

**getMode(0)** devolve uma lista com os pares de números. Cada par é composto por um número inteiro do modo e um número inteiro da definição.

Para uma listagem dos modos e das definições, consulte a tabela abaixo.

Se guardar as definições com **getMode(0)** → *var*, pode utilizar **setMode(***var***)** num programa ou função para restaurar temporariamente as definições na execução da função ou do programa. Consulte **setMode()**, página 93.

| | |
|---|---|
| getMode(0) | {1,1,2,1,3,1,4,1,5,1,6,1,7,1} |
| getMode(1) | 1 |
| getMode(7) | 1 |

| Nome do modo | Número inteiro do modo | Números inteiros da definição |
|---|---|---|
| Ver dígitos | **1** | **1** =Flutuante, **2** =Flutuante1, **3** =Flutuante2, **4** =Flutuante3, **5** =Flutuante4, **6** =Flutuante5, **7** =Flutuante6, **8** =Flutuante7, **9** =Flutuante8, **10** =Flutuante9, **11** =Flutuante10, **12** =Flutuante11, **13** =Flutuante12, **14** =Fixo0, **15** =Fixo1, **16** =Fixo2, **17** =Fixo3, **18** =Fixo4, **19** =Fixo5, **20** =Fixo6, **21** =Fixo7, **22** =Fixo8, **23** =Fixo9, **24** =Fixo10, **25** =Fixo11, **26** =Fixo12 |
| Ângulo | **2** | **1** =Radianos, **2** =Graus, **3** =Gradianos |
| Formato exponencial | **3** | **1** =Normal, **2** =Científica, **3** =Engenharia |
| Real ou Complexo | **4** | **1** =Real, **2** =Rectangular, **3** =Polar |
| Auto or Aprox. | **5** | **1** =Auto, **2** =Aproximado |
| Formato vectorial | **6** | **1** =Rectangular, **2** =Cilíndrico, **3** =Esférico |
| Base | **7** | **1** =Decimal, **2** =Hex, **3** =Binário |

## getNum()

**Catálogo >**

**getNum(***Fracção1***)** ⇒ *valor*

Transforma o argumento numa expressão que tem um denominador comum simplificado e, em seguida, devolve o numerador.

| | |
|---|---|
| $x:=5:\ y:=6$ | 6 |
| $\text{getNum}\left(\frac{x+2}{y-3}\right)$ | 7 |
| $\text{getNum}\left(\frac{2}{7}\right)$ | 2 |
| $\text{getNum}\left(\frac{1}{x}+\frac{1}{y}\right)$ | 11 |

## getType()

**Catálogo >**

**getType(***var***)** ⇒ *cadeia de texto*

Apresenta uma cadeia de texto que indica o tipo de dados da variável *var*.

Se *var* não tiver sido definido, apresenta a cadeia de texto "NENHUM".

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

## getVarInfo()

**Catálogo >**

**getVarInfo()** ⇒ *matriz* ou *palavra*

**getVarInfo(***CadeiaDoNomeDaBiblioteca***)** ⇒ *matriz* ou *palavra*

**getVarInfo()** devolve uma matriz de informações (nome da variável, tipo, acessibilidade da biblioteca e estado de bloqueio/desbloqueio) para todas as variáveis e os objectos da biblioteca definidos no problema actual.

Se não definir nenhuma variável, **getVarInfo()** apresenta a palavra

**getVarInfo(***NomeDaBiblioteca***)** apresenta uma matriz com informações para todos os objectos da biblioteca definidos na biblioteca *CadeiaDoNomeDaBiblioteca*. *CadeiaDoNomeDaBiblioteca* tem de ser uma palavra (texto entre aspas) ou uma variável da frase.

Se a biblioteca *CadeiaDoNomeDaBiblioteca* não existir, ocorre um erro.

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

Veja o exemplo do lado esquerdo, em que o resultado de **getVarInfo()** é atribuído à variável *vs*. A tentar de apresentação da linha 2 ou da linha 3 de *vs* apresenta uma mensagem de erro de "Matriz ou lista inválida" porque pelo menos um dos elementos nessas linhas (variável *b*, por exemplo) reavalia-se para uma matriz.

Este erro pode também ocorrer quando utilizar *Ans* para reavaliar um resultado **getVarInfo()**.

O sistema apresenta o erro acima porque a versão actual do software não suporta uma estrutura de matriz generalizada em que um elemento de uma matriz pode ser uma matriz ou uma lista.

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

## Goto

**Catálogo >**

**Goto** *NomeDefinição*

Transfere o controlo para a definição *NomeDefinição*.

*NomeDefinição* tem de ser definido na mesma função com uma instrução **Lbl**.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
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

**Catálogo >**

*Expr1* **▸Grad** ⇒ *expressão*

Converte *Expr1* para medição do ângulo de gradianos.

**Nota:** Pode introduzir este operador através da escrita de **@>Grad** no teclado do computador.

No modo de ângulo Graus:

$(1.5)\blacktriangleright\text{Grad}$ $(1.66667)^g$

No modo de ângulo Radianos:

$(1.5)\blacktriangleright\text{Grad}$ $(95.493)^g$

# I

## identity()

**Catálogo >**

**identity(***Número inteiro***)** ⇒ *matriz*

Devolve a matriz de identidade com uma dimensão de *Número inteiro*.

*Número inteiro* tem de ser um número inteiro positivo.

$\text{identity}(4)$

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

## If

**Catálogo >**

**If** *Declaração*
*ExprBooleana*

**If** *ExprBooleana* **Then**
*Bloco*
**EndIf**

Se a *ExprBooleana* for avaliada como verdadeira, executa a declaração individual *Declaração* ou o bloco de declarações *Bloco* antes de continuar a execução.

Se a *ExprBooleana* for avaliada como falsa, continua a execução sem executar a declaração ou o bloco de declarações.

*Bloco* pode ser uma declaração ou uma sequência de declarações separadas pelo carácter ":".

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

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

**If** *ExprBooleana* **Then**
*Bloco1*
**Else**
*Bloco2*
**EndIf**

Se a *ExprBooleana* for avaliada como verdadeira, executa o *Bloco1* e ignora o *Bloco2*.

Se a *ExprBooleana* for avaliada como falsa, ignora o *Bloco1*, mas executa o *Bloco2*.

*Bloco1* e *Bloco2* podem ser uma declaração única.

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

## If

Catálogo >

**If** *ExprBooleana1* **Then**
  *Bloco1*
**ElseIf** *ExprBooleana2* **Then**
  *Bloco2*
⋮
**ElseIf** *ExprBooleanaN* **Then**
  *BlocoN*
**EndIf**

Permite a derivação. Se a *ExprBooleana1* for avaliada como verdadeira, executa o *Bloco1*. Se a *ExprBooleana1* for avaliada como falsa, avalia a *ExprBooleana2*, etc.

```
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()

Catálogo >

**ifFn(***ExprBooleana***,** *Value_If_true* **[,** *Value_If_false* **[,** *Value_If_unknown* **]])** ⇒ *expressão, lista ou matriz*

Avalia a expressão booleana *ExprBooleana* (ou cada elemento da *ExprBooleana*) e produz um resultado com base nas seguintes regras:

- *ExprBooleana* pode testar um valor individual, uma lista ou uma matriz.
- Se um elemento da *ExprBooleana* for avaliado como verdadeiro, devolve o elemento correspondente de *Value_If_true*.
- Se um elemento da *ExprBooleana* for avaliada como falsa, devolve o elemento correspondente de *Value_If_false*. Se omitir *Value_If_false*, devolve undef.
- Se um elemento da *ExprBooleana* não for verdadeiro nem falso, devolve o elemento correspondente *Value_If_unknown*. Se omitir *Value_If_unknown*, devolve undef.
- Se o segundo, o terceiro ou o quarto argumento da função **ifFn()** for uma expressão individual, o teste booleano é aplicado a todas as posições da *ExprBooleana*.

**Nota:** Se a declaração *ExprBooleana* simplificada envolver uma lista ou matriz, todos os outros argumentos da lista ou matriz têm de ter as mesmas dimensões e o resultado terá as mesmas dimensões.

```
ifFn({1,2,3}<2.5,{5,6,7},{8,9,10})
                                    {5,6,10}
```

O valor do teste de **1** é inferior a 2.5, por esta razão, o elemento *Value_If_True* correspondente de **5** é copiado para a lista de resultados.

O valor do teste de **2** é inferior a 2.5, por esta razão, o elemento *Value_If_True* correspondente de **6** é copiado para a lista de resultados.

O valor do teste de **3** não é inferior a 2.5, por esta razão, o elemento *Value_If_False* correspondente de **10** é copiado para a lista de resultados.

```
ifFn({1,2,3}<2.5,4,{8,9,10})          {4,4,10}
```

*Value_If_true* é um valor individual e corresponde a qualquer posição seleccionada.

```
ifFn({1,2,3}<2.5,{5,6,7})             {5,6,undef}
```

*Value_If_false* não é especificado. Undef é utilizado.

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

Um elemento seleccionado de *Value_If_true*. Um elemento seleccionado de *Value_If_unknown*.

## imag()

Catálogo >

**imag(***Value1***)** ⇒ *valor*

Devolve a parte imaginária do argumento.

**Nota:** Todas as variáveis indefinidas são tratadas como variáveis reais. Consulte também **real()**, página 84

```
imag(1+2·i)                              2
```

### imag()

Catálogo >

**imag(***Lista1***)** ⇒ *lista*

Devolve uma lista de partes imaginárias dos elementos.

**imag(***Matriz1***)** ⇒ *matriz*

Devolve uma matriz das partes imaginárias dos elementos.

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

$\text{imag}\begin{bmatrix} 1 & 2 \\ i\cdot 3 & i\cdot 4 \end{bmatrix}$ → $\begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix}$

**Indirecta** Consulte **#()**, página 131.

### inString()

Catálogo >

**inStrin g(***CadeiaDeOrigem***,** *CadeiaDeOrigem* **[,** *Início* **])** ⇒ *número inteiro*

Devolve a posição do carácter na cadeia *CadeiaDeOrigem* em que começa a primeira ocorrência da cadeia *CadeiaSecundária*.

*Início*, se incluído, especifica a posição do carácter na *CadeiaDeOrigem* em que começa a procura. Predefinição = 1 (o primeiro carácter de *CadeiaDeOrigem*).

Se *CadeiaDeOrigem* não contiver *CadeiaSecundária* ou *Início* for > o comprimento de *CadeiaDeOrigem*, devolve zero.

inString("Hello there","the") → 7

inString("ABCEFG","D") → 0

### int()

Catálogo >

**int(***Valor***)** ⇒ *número inteiro*
**int(***Lista1***)** ⇒ *lista*
**int(***Matriz1***)** ⇒ *matriz*

Devolve o maior número inteiro que é igual ou inferior ao argumento. Esta função é idêntica a **floor()**.

O argumento pode ser um número complexo ou real.

Para uma lista ou matriz, devolve o maior número inteiro de cada elemento.

int(-2.5) → -3.

int([-1.234 0 0.37]) → [-2. 0 0.]

### intDiv()

Catálogo >

**intDiv(***Número1***,** *Número2***)** ⇒ *número inteiro*
**intDiv(***Lista1***,** *Lista2***)** ⇒ *lista*
**intDiv(***Matriz1***,** *Matriz2***)** ⇒ *matriz*

Devolve a parte do número inteiro assinada de (*Número1* ÷ *Número2*).

Para listas e matrizes, devolve a parte do número inteiro assinada de (argumento 1 ÷ argumento 2) para cada par de elementos.

intDiv(-7,2) → -3

intDiv(4,5) → 0

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

## interpolate()

Catálogo >

**interpolate(***xValue***,** *xList***,** *yList***,** *yPrimeList***)** ⇒ *lista*

Esta função efectua o seguinte:

Dado *xList*, *yList*=**f(***xList***)** e *yPrimeList*=**f'(***xList***)** para alguma função **f** desconhecida, é utilizada uma interpolante cúbica para aproximar a função **f** em *xValue*. Presume-se que *xList* é uma lista de números estritamente crescentes ou decrescentes, mas esta função pode apresentar um valor mesmo quando não o seja. Esta função percorre *xList* procurando por um intervalo [*xList*[i], *xList*[i+1]] que contenha *xValue*. Se encontrar tal intervalo, apresenta um valor interpolado para **f(***xValue***)**; caso contrário, apresenta **undef**.

*xList*, *yList* e *yPrimeList* têm de ter a mesma dimensão ≥ 2 e conter expressões que simplificam para números.

*xValue* pode ser um número ou uma lista de números.

Equação diferencial:
$y'=-3\cdot y+6\cdot t+5$ e $y(0)=5$

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

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

Para ver o resultado completo, prima ▲ e, de seguida, utilize ◀ e ▶ para mover o cursor.

Utilize a função de interpolação() para calcular os valores de função para xvaluelist:

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

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

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

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

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

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

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

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

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

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

## inv $\chi^2$()

Catálogo >

**inv** $\chi^2$ **(***Área***,** *df***)**
**invChi2(***Área***,** *df***)**

Calcula a função de probabilidade acumulada inversa $\chi^2$ (Qui quadrado) especificada pelo grau de liberdade, *df* para uma determinada *Área* debaixo da curva.

## invF()

Catálogo >

**invF(***Área***,** *dfNumer***,** *dfDenom***)**
**invF(***Área***,** *dfNumer***,** *dfDenom***)**

calcula a função de distribuição cunulativa inversa F especificada pelo *dfNumer* e o *dfDenom* para uma determinada *Área* debixo da curva.

## invNorm()

Catálogo >

**invNorm(***Área* [**,** μ**,** σ]**)**

Calcula a função de distribuição normal acumulada inversa para uma determinada *Área* debaixo da curva de distribuição normal especificada por μ e σ.

## invt()

Catálogo >

**invt(***Área***,** *df***)**

Calcula a função de probabilidade student-t acumulada inversa especificada pelo grau de liberdade, *df* para uma determinada *Área* debaixo da curva.

### iPart()

Catálogo >

**iPart(***Número***)** ⇒ *número inteiro*
**iPart(***Lista1***)** ⇒ *lista*
**iPart(***Matriz1***)** ⇒ *matriz*

Devolve a parte do número inteiro do argumento.

Para listas e matrizes, devolve a parte do número inteiro de cada elemento.

O argumento pode ser um número complexo ou real.

| | |
|---|---|
| $\text{iPart}(-1.234)$ | $-1.$ |
| $\text{iPart}\left(\left\{\frac{3}{2},-2.3,7.003\right\}\right)$ | $\{1.,-2.,7.\}$ |

### irr()

Catálogo >

**irr(***CF0, ListaCF* [, *FreqCF* ]**)** ⇒ *valor*

Função financeira que calcula a taxa de retorno interna de um investimento.

*CF0* é o cash flow inicial no momento 0; tem de ser um número real.

*ListaCF* é uma lista de montantes de cash flow após o cash flow inicial CF0.

*FreqCF* é uma lista opcional em que cada elemento especifica a frequência da ocorrência para um montante de cash flow agrupado (consecutivo), que é o elemento correspondente de *ListaCF*. A predefinição é 1; se introduzir valores, têm de ser números inteiros positivos < 10,000.

**Nota:** Consulte também **mirr()**, página 64.

| | |
|---|---|
| $list1:=\{6000,-8000,2000,-3000\}$ | $\{6000,-8000,2000,-3000\}$ |
| $list2:=\{2,2,2,1\}$ | $\{2,2,2,1\}$ |
| $\text{irr}(5000,list1,list2)$ | $-4.64484$ |

### isPrime()

Catálogo >

**isPrime(***Número***)** ⇒ *Expressão constante booleana*

Devolve verdadeiro ou falso para indicar se o *número* é um número inteiro ≥ 2 que é divisível apenas por si e 1.

Se o *Número* exceder cerca de 306 dígitos e não tiver factores ≤ 1021, **isPrime(***Número***)** mostra uma mensagem de erro.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

| | |
|---|---|
| $\text{isPrime}(5)$ | true |
| $\text{isPrime}(6)$ | false |

Função para localizar o número primeiro seguinte após um número especificado:

| | |
|---|---|
| Define $nextprim(n)$=Func<br>Loop<br>$n+1\rightarrow n$<br>If $\text{isPrime}(n)$<br>Return $n$<br>EndLoop<br>EndFunc | *Done* |
| $nextprim(7)$ | 11 |

### isVoid()

Catálogo >

**isVoid(***Var***)** ⇒ *Expressão constante booleana*
**isVoid(***Expr***)** ⇒ *Expressão constante booleana*
**isVoid(***Lista***)** ⇒ *lista de Expressões constantes booleanas*

Devolve verdadeiro ou falso para indicar se o argumento é um tipo de dados nulos.

Para mais informações sobre elementos nulos, consulte 136.

| | |
|---|---|
| $a:=\_$ | $\_$ |
| $\text{isVoid}(a)$ | true |
| $\text{isVoid}(\{1,\_,3\})$ | $\{\text{false,true,false}\}$ |

# L

## Lbl

Catálogo >

**Lbl** *NomeDefinição*

Define uma definição com o nome *NomeDefinição* numa função.

Pode utilizar uma instrução **Goto** *NomeDefinição* para transferir o controlo para a instrução imediatamente a seguir à definição.

*NomeDefinição* tem de cumprir os mesmos requisitos de nomeação do nome de uma variável.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
Define g()=Func                    Done
    Local temp,i
    0→temp
    1→i
    Lbl top
    temp+i→temp
    If i<10 Then
    i+1→i
    Goto top
    EndIf
    Return temp
    EndFunc
g()                                 55
```

## lcm()

Catálogo >

**lcm(***Número1***,** *Número2***)** ⇒ *expressão*
**lcm(***Lista1***,** *Lista2***)** ⇒ *lista*
**lcm(***Matriz1***,** *Matriz2***)** ⇒ *matriz*

Devolve o mínimo múltiplo comum dos dois argumentos. O **lcm** de duas fracções é o **lcm** dos numeradores divididos pelo **gcd** dos denominadores. O **lcm** dos números de ponto flutuante fraccionários é o produto.

Para duas listas ou matrizes, devolve os mínimos múltiplos comuns dos elementos correspondentes.

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

## left()

Catálogo >

**left(***CadeiaDeOrigem* [**,** *Num* ]**)** ⇒ *cadeia*

Devolve os caracteres *Num* mais à esquerda contidos na cadeia de caracteres *CadeiaDeOrigem*.

Se omitir *Num*, devolve todos os caracteres de *CadeiaDeOrigem*.

**left(***Lista1* [**,** *Num* ]**)** ⇒ *lista*

Devolve os elementos *Num* mais à esquerda em *Lista1*.

Se omitir *Num*, devolve todos os elementos de *Lista1*.

**left(***Comparação***)** ⇒ *expressão*

Devolve o lado esquerdo de uma equação ou desigualdade.

```
left("Hello",2)                    "He"
left({1,3,-2,4},3)                 {1,3,-2}
```

## libShortcut()

**Catálogo >**

**libShortcut(***CadeiaDoNomeDaBiblioteca***,** *CadeiaDoNomeDoAtalho*
**[,** *MarcadorDeBibPriv***])** ⇒ *lista de variáveis*

Cria um grupo de variáveis no problema actual que contém referências a todos os objectos no documento da biblioteca especificado *CadeiaDoNomeDaBiblioteca*. Adiciona também os membros do grupo ao menu Variáveis. Pode referir-se a cada objecto com a *CadeiaDoNomeDoAtalho*.

Definir *MarcadorDeBibliotecaPrivada*=**0** para excluir objectos da biblioteca privada (predefinição)
Definir *MarcadorDeBibliotecaPrivada*=**1** para incluir objectos da biblioteca privada

Para copiar um grupo de variáveis, consulte **CopyVar** na página 18. Para eliminar um grupo de variáveis, consulte **DelVar** na página 29.

Este exemplo assume um documento de biblioteca actualizado e guardado adequadamente denominado **linalg2** que contém objectos definidos como *clearmat*, *gauss1* e *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*}

## LinRegBx

**Catálogo >**

**LinRegBx** *X*,*Y*[,[*Freq* [,*Categoria*,*Incluir*]]

Calcula a regressão linear y = a+b • x a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: a+b • x |
| stat.a, stat.b | Parâmetros de regressão |
| stat.$r^2$ | Coeficiente de determinação |
| stat.r | Coeficiente de correlação |
| stat.Resid | Resíduos da regressão |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

## LinRegMx

Catálogo >

**LinRegMx** *X*,*Y*[,[*Freq*][,*Categoria*,*Incluir*]]

Calcula a regressão linear y = m • x+b a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: m • x+b |
| stat.m, stat.b | Parâmetros de regressão |
| stat.$r^2$ | Coeficiente de determinação |
| stat.r | Coeficiente de correlação |
| stat.Resid | Resíduos da regressão |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

**LinRegtIntervals** *X*, *Y*[, *F*[, **0**[, *NívC*]]]

Para declive. Calcula o intervalo de confiança de nível C do declive.

**LinRegtIntervals** *X*, *Y*[, *F*[, **1**, *ValX*[, *NívC*]]]

Para resposta. Calcula um valor y previsto, um intervalo de previsão de nível C para uma observação, e um intervalo de confiança de nível C para a resposta média.

Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão.

*X* e *Y* são listas de variáveis independentes e dependentes.

*F* é uma lista opcional de valores de frequência. Cada elemento em *F* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros ≥ 0.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: a+b • x |
| stat.a, stat.b | Parâmetros de regressão |
| stat.df | Graus de liberdade |
| stat.$r^2$ | Coeficiente de determinação |
| stat.r | Coeficiente de correlação |
| stat.Resid | Resíduos da regressão |

Apenas para o tipo de declive

| Variável de saída | Descrição |
|---|---|
| [stat.CLower, stat.CUpper] | Intervalo de confiança para o declive |
| stat.ME | Margem de erro do intervalo de confiança |
| stat.SESlope | Erro padrão do declive |
| stat.s | Erro padrão sobre a linha |

Apenas para o tipo de resposta

| Variável de saída | Descrição |
|---|---|
| [stat.CLower, stat.CUpper] | Intervalo de confiança para a resposta média |
| stat.ME | Margem de erro do intervalo de confiança |
| stat.SE | Erro padrão da resposta média |

| Variável de saída | Descrição |
|---|---|
| [stat.LowerPred, stat.UpperPred] | Intervalo de previsão para uma observação |
| stat.MEPred | Margem de erro do intervalo de previsão |
| stat.SEPred | Erro padrão para previsão |
| stat.$\hat{y}$ | a + b • XVal |

## LinRegtTest

**Catálogo >**

**LinRegtTest** *X*,*Y*[,*Freq*[,*Hipótese*]]

Calcula uma regressão linear a partir das listas *X* e *Y* e um teste *t* no valor do declive β e o coeficiente de correlação ρ para a equação $y=\alpha+\beta x$. Testa a hipótese nula $H_0:\beta=0$ (equivalentemente, $\rho=0$) em relação a uma das três hipóteses alternativas.

Todas as listas têm de ter a mesma dimensão.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Hipótese* é um valor opcional que especifica uma de três hipóteses alternativas em relação à qual a hipótese nula ($H_0:\beta=\rho=0$) será testada.

Para $H_a$: $\beta\neq0$ e $\rho\neq0$ (predefinição), defina *Hipótese*=0
Para $H_a$: $\beta<0$ e $\rho<0$, defina *Hipótese*<0
Para $H_a$: $\beta>0$ e $\rho>0$, defina *Hipótese*>0

Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: a + b • x |
| stat.t | *t*-Estatística para teste de importância |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.df | Graus de liberdade |
| stat.a, stat.b | Parâmetros de regressão |
| stat.s | Erro padrão sobre a linha |
| stat.SESlope | Erro padrão do declive |
| stat.$r^2$ | Coeficiente de determinação |
| stat.r | Coeficiente de correlação |
| stat.Resid | Resíduos da regressão |

## linSolve()

**Catálogo >**

**linSolve(** *SistemaDeEquaçõesLineares*, *Var1*, *Var2*, ...**)** ⇒ *lista*
**linSolve(***EquaçãoLinear1* **e** *EquaçãoLinear2* **e** ..., *Var1*, *Var2*, ...**)** ⇒ *lista*
**linSolve({***EquaçãoLinear1*, *EquaçãoLinear2*, ...**}**, *Var1*, *Var2*, ...**)** ⇒ *lista*
**linSolve(***SistemaDeEquaçõesLineares*, **{***Var1*, *Var2*, ...**})** ⇒ *lista*
**linSolve(***EquaçãoLinear1* **e** *EquaçãoLinear2* **e** ..., **{***Var1*, *Var2*, ...**})** ⇒ *lista*
**linSolve({***EquaçãoLinear1*, *EquaçãoLinear2*, ...**}**, **{***Var1*, *Var2*, ...**})** ⇒ *lista*

Devolve uma lista de soluções para as variáveis *Var1*, *Var2*, ...

O primeiro argumento tem de avaliar um sistema de equações do 1º grau ou uma equação individual do 1º grau. Caso contrário, ocorre um erro de argumento.

Por exemplo, a avaliação de **linSolve(**x=1 e x=2,x**)** produz um resultado de "Erro de argumento".

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

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

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

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

## ΔList()

**Catálogo >**

**ΔList(***Lista1***)** ⇒ *lista*

**Nota:** Pode introduzir esta função através da escrita de **deltaList(**...**)** no teclado.

Devolve uma lista com as diferenças entre os elementos consecutivos em *Lista1*. Cada elemento de *Lista1* é subtraído do elemento seguinte de *Lista1*. A lista resultante é sempre um elemento mais pequeno que a *Lista1* original.

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

## list▸mat()

**Catálogo >**

**list▸mat(***Lista* [**,** *elementosPorLinha* ]**)** ⇒ *matriz*

Devolve uma matriz preenchida linha por linha com os elementos da *Lista*.

*elementosPorLinha*, se incluído, especifica o número de elementos por linha. A predefinição é o número de elementos em *Lista* (uma linha).

Se a *Lista* não preencher a matriz resultante, são adicionados zeros.

**Nota:** Pode introduzir esta função através da escrita de **list@>mat(**...**)** no teclado do computador.

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

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

## ln()

**Teclas** ctrl ex

**ln(***Valor1***)** ⇒ *valor*
**ln(***Lista1***)** ⇒ *lista*

Devolve o logaritmo natural do argumento.

Para uma lista, devolve os logaritmos naturais dos elementos.

$\ln(2.)$ 0.693147

Se o modo do formato complexo for Real:

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

Se o modo do formato complexo for Rectangular:

$\ln(\{-3,1.2,5\})$ $\{1.09861+3.14159\cdot i,0.182322,1.60944\}$

## ln()

**Teclas** ctrl $e^x$

**ln(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve o logaritmo natural da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o logaritmo natural de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()** em.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos e Formato complexo rectangular:

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

$$\begin{bmatrix}1.83145+1.73485\cdot i & 0.009193-1.49086\\ 0.448761-0.725533\cdot i & 1.06491+0.623491\\ -0.266891-2.08316\cdot i & 1.12436+1.79018\cdot\end{bmatrix}$$

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

## LnReg

**Catálogo >**

**LnReg** *X*, *Y*[, [*Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão logarítmica y = a+b • ln(x) a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: a+b • ln(x) |
| stat.a, stat.b | Parâmetros de regressão |
| stat.$r^2$ | Coeficiente de determinação linear para dados transformados |
| stat.r | Coeficiente de correlação para dados transformados (ln(x), y) |
| stat.Resid | Resíduos associados ao modelo logarítmico |
| stat.ResidTrans | Resíduos associados ao ajuste linear dos dados transformados |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

## Local

**Catálogo >**

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

Declara as *vars* especificadas como variáveis locais. Essas variáveis só existem durante a avaliação de uma função e são eliminadas quando a função terminar a execução.

**Nota:** As variáveis locais poupam memória porque só existem temporariamente. Também não perturbam nenhum valor da variável global existente. As variáveis locais têm de ser utilizadas para ciclos **For** e guardar temporariamente os valores numa função multilinhas visto que as modificações nas variáveis globais não são permitidas numa função.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
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

**Catálogo >**

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

Bloqueia as variáveis ou o grupo de variáveis especificadas. Não pode eliminar ou modificar as variáveis bloqueadas.

Não pode bloquear ou desbloquear a variável do sistema *Ans*, e não pode bloquear os grupos de variáveis do sistema *stat***.** ou *tvm***.**

**Nota:** O comando **Bloquear** (**Lock**) apaga o histórico de Anular/Repetir quando aplicado a variáveis desbloqueadas.

Consulte **unLock**, página 113, e **getLockInfo()**, página 42.

```
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()

**Teclas** ctrl 10^x

**log (***Valor1* [**,** *Valor2* ]**)** ⇒ *valor*
**log (***Lista1* [**,** *Valor2* ]**)** ⇒ *lista*

Devolve o logaritmo -*Valor2* base do primeiro argumento.

**Nota:** Consulte também **Modelo do logaritmo**, página 2.

Para uma lista, devolve o logaritmo -*Valor2* base dos elementos.

Se omitir o segundo argumento, 10 é utilizado como a base.

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

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

$$\log_{3}(10)-\log_{3}(5) \qquad 0.63093$$

Se o modo do formato complexo for Real:

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

"Error: Non−real calculation"

Se o modo do formato complexo for Rectangular:

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

$$\{0.477121+1.36438\cdot i,0.079181,0.69897\}$$

## log()

**Teclas** ctrl 10^x

**log (***MatrizQuadrada1* **[,** *Valor* **])** ⇒ *MatrizQuardrada*

Devolve o logaritmo *Valor* base da matriz de *MatrizQuadrada1*. Isto não é mesmo que calcular o logaritmo *Valor* base de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

Se omitir o argumento base, 10 é utilizado como a base.

No modo de ângulo Radianos e Formato complexo rectangular:

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

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

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

## Logistic

**Catálogo >**

**Logistic** *X*, *Y*[, [*Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão logística $y = (c/(1+a \cdot e^{-bx}))$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: $c/(1+a \cdot e^{-bx})$ |
| stat.a, stat.b, stat.c | Parâmetros de regressão |
| stat.Resid | Resíduos da regressão |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

**LogisticD** *X*, *Y* [, [*Repetições*], [*Freq*] [, *Categoria*, *Incluir*] ]

Calcula a regressão logística $y = (c/(1+a \cdot e^{-bx})+d)$ a partir das listas *X* e *Y* com a frequência *Freq*, utilizando um número especificado de *repetições*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Iterações* é um valor opcional que especifica o número máximo de vezes que uma solução será tentada. Se for omitido, 64 é utilizado. Em geral, valores maiores resultam numa melhor precisão, mas maiores tempos de execução, e vice-versa.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: $c/(1+a \cdot e^{-bx})+d)$ |
| stat.a, stat.b, stat.c, stat.d | Parâmetros de regressão |
| stat.Resid | Resíduos da regressão |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

### Loop

Catálogo >

**Ciclo**
*Bloco*
**EndLoop**

Executa repetidamente as declarações em *Bloco*. Não se esqueça de que o ciclo será executado continuamente, excepto se executar a instrução **Ir para** ou **Sair** no *Bloco*.

*Bloco* é uma sequência de declarações separadas pelo carácter ":".

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
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

Catálogo >

**LU** *Matriz*, *MatrizI*, *Matrizu*, *Matrizp[, Tol]*

Calcula a decomposição LU (inferior-superior) Doolittle LU de uma matriz complexa ou real. A matriz triangular inferior é guardada em *MatrizI*, a matriz triangular superior em *Matrizu* e a matriz de permutações (que descreve as trocas de linhas durante o cálculo) em *Matrizp*.

*MatrizI* · *Matrizu* = *Matrizp* · *matriz*

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância só é utilizada se a matriz tiver entradas de ponto flutuante e não contiver variáveis simbólicas sem um valor atribuído. Caso contrário, *Tol* é ignorado.

- Se utilizar ctrl enter ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética do ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida for calculada como:
  5E⁻14 · max(dim(*Matriz*)) · rowNorm(*Matriz*)

O algortimo de factorização **LU** utiliza a articulação parcial com as trocas de linhas.

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

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

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

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

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

## *M*

### mat▸list()

Catálogo >

**mat▸lis t(***Matriz***)** ⇒ *lista*

Devolve uma lista preenchida com os elementos em *Matriz*. Os elementos são copiados de *Matriz* linha por linha.

**Nota:** Pode introduzir esta função através da escrita de **mat@>list(** **...)** no teclado do computador.

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

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

$$\text{mat}\blacktriangleright\text{list}(m1) \qquad \{1,2,3,4,5,6\}$$

## max()

Catálogo >

**max(***Valor1*, *Valor2***)** ⇒ *expressão*
**max(***Lista1*, *Lista2***)** ⇒ *lista*
**max(***Matriz1*, *Matriz2***)** ⇒ *matriz*

Devolve o máximo dos dois argumentos. Se os argumentos forem duas listas ou matrizes, devolve uma lista ou matriz com o valor máximo de cada par dos elementos correspondentes.

**max(***Lista***)** ⇒ *expressão*

Devolve o elemento máximo em *lista*.

**max(***Matriz1***)** ⇒ *matriz*

Devolve um vector da linha com o elemento máximo de cada coluna em *Matriz1*.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

**Nota:** Consulte também **min()**.

$\max(2.3,1.4)$ → $2.3$

$\max(\{1,2\},\{-4,3\})$ → $\{1,3\}$

$\max(\{0,1,-7,1.3,0.5\})$ → $1.3$

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

## mean()

Catálogo >

**mean(***Lista* [**,** *freList* ]**)** ⇒ *expressão*

Devolve a média dos elementos em *Lista*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

**mean(***Matriz1* [**,** *MatrizFreq* ]**)** ⇒ *matriz*

Devolve um vector da linha da média de todas as colunas em *Matriz1*.

Cada elemento de *MatrizFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

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

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

No Formato de vector 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()

Catálogo >

**median(***Lista*[**,** *ListaFreq*]**)** ⇒ *expressão*

Devolve a mediana dos elementos em *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

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

## median()

**Catálogo >**

**median(***Matriz1***[, ***MatrizFreq***])** ⇒ *matriz*

Devolve um vector em linha com as medianas das colunas da *Matriz1*.

Cada elemento de *MatrizFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

**Notas:**

- Todas as entradas da lista ou matriz têm de ser simplificadas para números.
- Os elementos (nulos) vazios da lista ou matriz são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

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

## MedMed

**Catálogo >**

**MedMed** *X*,*Y* **[,** *Freq***] [,** *Categoria***,** *Incluir***]]**

Calcula a recta média-média y = (m • x+b) a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação da recta mediana-mediana: m • x+b |
| stat.m, stat.b | Parâmetros do modelo |
| stat.Resid | Resíduos da recta mediana-mediana |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

## mid()

Catálogo >

**mid(***CadeiaDeOrigem***,** *Início* **[,** *Contagem* **])** ⇒ *cadeia*

Devolve os caracteres *Contagem* a partir da cadeia de caracteres *CadeiaDeOrigem*, começando pelo número de caracteres *Início*.

Se *Contagem* for omitida ou maior que a dimensão de *CadeiaDeOrigem*, devolve todos os caracteres de *CadeiaDeOrigem*, começando pelo número de caracteres *Início*.

*Contagem* tem de ser ≥ 0. Se *Contagem* = 0, devolve uma cadeia vazia.

```
mid("Hello there",2)          "ello there"
mid("Hello there",7,3)        "the"
mid("Hello there",1,5)        "Hello"
mid("Hello there",1,0)        ""
```

**mid(***ListaDeOrigem***,** *Início* **[,** *Contagem* **])** ⇒ *lista*

Devolve os elementos *Contagem* de *ListaDeOrigem*, começando pelo número de elementos *Início*.

Se *Contagem* for omitida ou maior que a dimensão de *ListaDeOrigem*, devolve todos os elementos de *ListaDeOrigem*, começando pelo número de elementos *Início*.

*Contagem* tem de ser ≥ 0. Se Contagem = 0, devolve uma lista vazia.

```
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(***ListaDaCadeiaDeOrigem***,** *Início* **[,** *Contagem* **])** ⇒ *lista*

Devolve as cadeias *Contagem* da lista de cadeias *ListaDaCadeiaDeOrigem*, começando pelo número de elementos *Início*.

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

## min()

Catálogo >

**min(***Valor1***,** *Valor2***)** ⇒ *expressão*
**min(***Lista1*, *Lista2***)** ⇒ *lista*
**min(***Matriz1*, *Matriz2***)** ⇒ *matriz*

Devolve o mínimo dos dois argumentos. Se os argumentos forem duas listas ou matrizes, devolve uma lista ou matriz com o valor mínimo de cada par dos elementos correspondentes.

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

**min(***Lista***)** ⇒ *expressão*

Devolve o elemento mínimo de *Lista*.

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

**min(***Matriz1***)** ⇒ *matriz*

Devolve um vector da linha com o elemento mínimo de cada coluna em *Matriz1*.

```
min([ 1  -3  7 ])             [-4  -3  0.3]
   ([-4   0  0.3])
```

**Nota:** Consulte também **max()**.

### mirr()

Catálogo >

**mirr(***TaxaDeFinanciamento***,** *TaxaDeReinvestimento***,** *CF0***,** *ListaCF* [**,** *FreqCF* ]**)**

Função financeira que devolve a taxa de retorno interna modificada de um investimento.

*TaxaDeFinanciamento* é a taxa de juro que é paga sobre os montantes de cash flow.

*TaxaDeReinvestimento* é a taxa de juro em que os cash flows são reinvestidos.

*CF0* é o cash flow inicial no momento 0; tem de ser um número real.

*ListaCF* é uma lista de montantes de cash flow após o cash flow inicial CF0.

*FreqCF* é uma lista opcional em que cada elemento especifica a frequência da ocorrência para um montante de cash flow agrupado (consecutivo), que é o elemento correspondente de *ListaCF*. A predefinição é 1; se introduzir valores, têm de ser números inteiros positivos < 10,000.

**Nota:** Consulte também **irr()**, página 49.

*list1*:={6000,⁻8000,2000,⁻3000} {6000,⁻8000,2000,⁻3000}

*list2*:={2,2,2,1} {2,2,2,1}

mirr(4.65,12,5000,*list1*,*list2*) 13.41608607

### mod()

Catálogo >

**mod(***Valor1***,** *Valor2***)** ⇒ *expressão*
**mod(***Lista1***,** *Lista2***)** ⇒ *lista*
**mod(***Matriz1***,** *Matriz2***)** ⇒ *matriz*

Devolve o primeiro módulo de argumentos do segundo argumento conforme definido pelas identidades:

mod(x,0) = x
mod(x,y) = x − y floor(x/y)

Quando o segundo argumento for diferente de zero, o resultado é periódico nesse argumento. O resultado é zero ou tem o mesmo sinal do segundo argumento.

Se os argumentos forem duas listas ou matrizes, devolve uma lista ou matriz com o módulo de cada par de elementos correspondentes.

**Nota:** Consulte também **remain()**, página 85

mod(7,0) 7

mod(7,3) 1

mod(⁻7,3) 2

mod(7,⁻3) ⁻2

mod(⁻7,⁻3) ⁻1

mod({12,⁻14,16},{9,7,⁻5}) {3,0,⁻4}

### mRow()

Catálogo >

**mRow(***Valor***,** *Matriz1***,** *Índice***)** ⇒ *matriz*

Devolve uma cópia de *Matriz1* com cada elemento na linha *Índice* de *Matriz1* multiplicado por *Valor*.

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

### mRowAdd()

Catálogo >

**mRowAdd(***Valor***,** *Matriz1***,** *Índice1***,** *Índice2***)** ⇒ *matriz*

Devolve uma cópia de *Matriz1* com cada elemento na linha *Índice2* de *Matriz1* substituído por:

*Valor* · linha *Índice1* + linha *Índice2*

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

## MultReg

Catálogo >

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

Calcula a regressão linear múltipla da lista *Y* nas listas *X1*, *X2*, …, *X10*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter dimensões iguais.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: b0+b1 • x1+b2 • x2+ ... |
| stat.b0, stat.b1, ... | Parâmetros de regressão |
| stat.$R^2$ | Coeficiente de determinação múltipla |
| stat.$\hat{y}$Lista | $\hat{y}$Lista = b0+b1 • x1+ ... |
| stat.Resid | Resíduos da regressão |

## MultRegIntervals

Catálogo >

**MultRegIntervals** *Y*, *X1*[,*X2*[,*X3*,...[,*X10*]]],*ListaValX*[,*NívelC*]

Calcula um valor y previsto, um intervalo de previsão de nível C para uma observação, e um intervalo de confiança de nível C para a resposta média.

Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter dimensões iguais.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: b0+b1 • x1+b2 • x2+ ... |
| stat.$\hat{y}$ | Um ponto prevê: $\hat{y}$ = b0 + b1 • xl + ... para *ListaDeValoresX* |
| stat.dfError | Erro dos graus de liberdade |
| stat.CLower, stat.CUpper | Intervalo de confiança para uma resposta média |
| stat.ME | Margem de erro do intervalo de confiança |
| stat.SE | Erro padrão da resposta média |
| stat.LowerPred, stat.UpperrPred | Intervalo de previsão para uma observação |
| stat.MEPred | Margem de erro do intervalo de previsão |
| stat.SEPred | Erro padrão para previsão |
| stat.bList | Lista de parâmetros de regressão, {b0,b1,b2,...} |

| Variável de saída | Descrição |
|---|---|
| stat.Resid | Resíduais da regressão |

**MultRegTests** **Catálogo >**

**MultRegTests** *Y, X1*[*,X2*[*,X3*[*,...*[*,X10*]]]

O teste de regressão linear calcula uma regressão linear múltipla a partir dos dados fornecidos e fornece a estatística do teste $F$ global e estatística do teste $t$ para os coeficientes.

Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

Saídas

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: b0+b1 • x1+b2 • x2+ ... |
| stat.F | Estatística do teste $F$ global |
| stat.PVal | Valor P associado à estatística $F$ global |
| stat.$R^2$ | Coeficiente de determinação múltipla |
| stat.Adj$R^2$ | Coeficiente ajustado de determinação múltipla |
| stat.s | Desvio padrão do erro |
| stat.DW | Estatística Durbin-Watson; utilizada para determinar se a correlação automática de primeira ordem está presente no modelo |
| stat.dfReg | Graus de liberdade da regressão |
| stat.SSReg | Soma de quadrados da regressão |
| stat.MSReg | Quadrado médio da regressão |
| stat.dfError | Erro dos graus de liberdade |
| stat.SSError | Erro da soma de quadrados |
| stat.MSError | Erro do quadrado médio |
| stat.bList | {b0,b1,...} Lista de parâmetros |
| stat.tList | Lista da estatística t, um para cada coeficiente na bList |
| stat.PList | Lista de valores P para cada estatística t |
| stat.SEList | Lista de erros padrão para coeficientes na bList |
| stat.$\hat{\mathbf{y}}$Lista | $\hat{\mathbf{y}}$Lista = b0+b1 • x1+ . . . |
| stat.Resid | Resíduos da regressão |
| stat.sResid | Resíduos normalizados; obtido através da divisão de um resíduo pelo desvio padrão |
| stat.CookDist | Distância de Cook; medição da influência de uma observação com base no residual e optimização |
| stat.Leverage | Medição da distância entre os valores independentes e os valores médios |

### nand

**Teclas** ctrl =

*ExprBooleana1* **nand** *ExprBooleana2* devolve *expressão booleana*
*ListaBooleana1* **nand** *ListaBooleana2* devolve *lista booleana*
*MatrizBooleana1* **nand** *MatrizBooleana2* devolve *matriz booleana*

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

Devolve a negação de uma operação **and** lógica nos dois argumentos. Devolve falso, verdadeiro ou uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

*NúmeroInteiro1* **nand** *NúmeroInteiro2* ⇒ *número inteiro*

Compara dois números inteiros reais bit a bit com uma operação **nand**. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se ambos os bits forem 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

| | |
|---|---|
| 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} |

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respetivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 10).

### nCr()

**Catálogo >**

**nCr(***Valor1***,** *Valor2***)** ⇒ *expressão*

Para o número inteiro *Valor1* e *Valor2* com *Valor1* ≥ *Valor2* ≥ 0, **nCr()** é o número de combinações de coisas de *Valor1* retiradas de *Valor2* de uma vez. (Isto também é conhecido como um coeficiente binomial.)

| | |
|---|---|
| $\text{nCr}(z,3)\|z=5$ | 10 |
| $\text{nCr}(z,3)\|z=6$ | 20 |

**nCr(***Valor***, 0)** ⇒ **1**

**nCr(***Valor***,** *NúmeroInteiroNeg***)** ⇒ **0**

**nCr(***Valor***,** *NúmeroInteiroPos***)** ⇒ *Valor* **·(***Valor* **−1)...**
**(***Valor* **−***NúmeroInteiroPos* **+1)/** *NúmeroInteiroPos***!**

**nCr(***Valor***,** *NúmeroNãoInteiro***)** ⇒ *expressão* **!/**
**((***Valor* **−***NúmeroNãoInteiro***)!·** *NúmeroNãoInteiro* **!)**

**nCr(***Lista1***,** *Lista2***)** ⇒ *lista*

| | |
|---|---|
| $\text{nCr}(\{5,4,3\},\{2,4,2\})$ | $\{10,1,3\}$ |

Devolve uma lista de combinações com base nos pares de elementos correspondentes nas duas listas. Os argumentos têm de ter o mesmo tamanho de listas.

**nCr(***Matriz1***,** *Matriz2***)** ⇒ *matriz*

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

Devolve uma matriz de combinações com base nos pares de elementos correspondentes nas duas matrizes. Os argumentos têm de ter o mesmo tamanho de matrizes.

### nDerivative()

Catálogo >

**nDerivative(***Expr1***,***Var*=*Valor*[**,***Ordem*]**)** ⇒ *valor*
**nDerivative(***Expr1***,***Var*[**,***Ordem*]**)** | *Var*=*Valor* ⇒ *valor*

Devolve a derivada numérica calculada com os métodos de diferenciação automáticos.

Ao especificar o *Valor*, substitui qualquer atribuição de variável anterior ou qualquer substituição atual "|" para a variável.

Se a variável *Var* não contiver um valor numérico, tem de fornecer *Valor*.

*Ordem* da derivada tem de ser **1** ou **2**.

**Nota:** O algoritmo **nDerivative()** tem uma limitação: funciona recursivamente através da expressão não simplificada, computação do valor numérico da primeira derivada (e a segunda, se aplicável) e a avaliação de cada subexpressão, que pode conduzir a um resultado imprevisto.

Considere o exemplo da direita. A primeira derivada de x·(x^2+x)^(1/3) em x=0 é igual a 0. No entanto, como a primeira derivada da subexpressão (x^2+x)^(1/3) está indefinida em x=0, e este valor é utilizado para calcular a derivada da expressão total, **nDerivative()** reporta o resultado como indefinido e apresenta uma mensagem de aviso.

Se encontrar esta limitação, verifique a solução graficamente. Pode também tentar com **centralDiff()**.

$\text{nDerivative}(|x|,x=1)$ → 1

$\text{nDerivative}(|x|,x)|x=0$ → undef

$\text{nDerivative}(\sqrt{x-1},x)|x=1$ → undef

$\text{nDerivative}\left(x\cdot(x^2+x)^{\frac{1}{3}},x,1\right)|x=0$ → undef

$\text{centralDiff}\left(x\cdot(x^2+x)^{\frac{1}{3}},x\right)|x=0$ → 0.000033

### newList()

Catálogo >

**newLis t(***ElementosNum***)** ⇒ *lista*

Devolve uma lista com uma dimensão de *ElementosNum*. Cada elemento é zero.

$\text{newList}(4)$ → {0,0,0,0}

### newMat()

Catálogo >

**newMa t(***LinhaNum***,** *ColunasNum***)** ⇒ *matriz*

Devolve uma matriz de zeros com a dimensão *LinhasNum* por *ColunasNum*.

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

### nfMax()

Catálogo >

**nfMax(***Expr***,** *Var***)** ⇒ *valor*
**nfMax(***Expr***,** *Var***,** *LimiteInferior***)** ⇒ *valor*
**nfMax(***Expr***,** *Var***,** *LimiteInferior***,** *LimiteSuperior***)** ⇒ *valor*
**nfMax(***Expr***,** *Var***)** | *LimiteInferior*≤*Var*≤*LimiteSuperior* ⇒ *valor*

Devolve um valor numérico candidato da variável *Var* em que ocorre o máximo local de *Expr*.

Se fornecer um *LimiteInferior* e um *LimiteSuperior*, a função procura o máximo local no intervalo fechado [*LimiteInferior*,*LimiteSuperior*].

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

$\text{nfMax}(0.5\cdot x^3-x-2,x,-5,5)$ → −0.816497

## nfMin()

Catálogo >

**nfMin(***Expr***,** *Var***)** ⇒ *valor*
**nfMin(***Expr***,** *Var, LimiteInferior***)** ⇒ *valor*
**nfMin(***Expr***,** *Var, LimiteInferior***,** *LimiteSuperior***)** ⇒ *valor*
**nfMin(***Expr, Var***) |** *LimiteInferior*≤*Var*≤*LimiteSuperior* ⇒ *valor*

Devolve um valor numérico candidato da variável *Var* em que ocorre o mínimo local de *Expr*.

Se fornecer um *LimiteInferior* e um *LimiteSuperior*, a função procura o mínimo local no intervalo fechado [*LimiteInferior*,*LimiteSuperior*].

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

$\text{nfMin}(0.5\cdot x^3-x-2,x,-5,5)$ → $0.816497$

## nInt()

Catálogo >

**nInt(***Expr1, Var, Inferior, Superior***)** ⇒ *expressão*

Se a expressão a integrar *Expr1* não contiver nenhuma variável para além de *Var* e se *Inferior* e *Superior* forem constantes, ∞ positivo ou ∞ negativo, **nInt()** devolve uma aproximação de ∫**(***Expr1, Var, Inferior, Superior***)**. Esta aproximação é uma média ponderada de alguns valores de amostra da expressão a integrar no intervalo *Inferior* < *Var* < *Superior*.

$\text{nInt}(e^{-x^2},x,-1,1)$ → $1.49365$

O objectivo é obter seis dígitos significativos. O algoritmo adaptável termina quando parecer que o objectivo foi alcançado ou quando parecer improvável que as amostras adicionais produzam uma melhoria acentuada.

Aparece um aviso ("Precisão questionável") quando parecer que o objectivo não foi alcançado.

$\text{nInt}(\cos(x),x,-\pi,\pi+1.\text{E}{-12})$ → $-1.04144\text{E}{-12}$

Nest **nInt()** para fazer integração numérica múltipla. Os limites da integração podem depender das variáveis de integração fora dos limites.

$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x\cdot y}}{\sqrt{x^2-y^2}},y,-x,x\right),x,0,1\right)$ → $3.30423$

## nom()

Catálogo >

**nom(***TaxaEfectiva,CpY***)** ⇒ *valor*

Função financeira que converte a taxa de juro efectiva anual *TaxaEfectiva* para uma taxa nominal, dando *CpY* como o número de períodos compostos por ano.

*TaxaEfectiva* tem de ser um número real e *CpY* tem de ser um número real > 0.

**Nota:** Consulte também **eff()**, página 32.

$\text{nom}(5.90398,12)$ → $5.75$

## nor

Teclas ctrl =

*ExprBooleana1* **nor** *ExprBooleana2* devolve *expressão booleana*
*ListaBooleana1* **nor** *ListaBooleana2* devolve *lista booleana*
*MatrizBooleana1* **nor** *MatrizBooleana2* devolve *matriz booleana*

Devolve a negação de uma operação **or** lógica nos dois argumentos. Devolve falso, verdadeiro ou uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

$x\geq 3 \text{ or } x\geq 4$ → $x\geq 3$

$x\geq 3 \text{ nor } x\geq 4$ → $x<3$

### nor

**Teclas** ctrl =

*NúmeroInteiro1* **nor** *NúmeroInteiro2* ⇒ *número inteiro*

Compara dois números inteiros reais bit a bit com uma operação **nor**. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se ambos os bits forem 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respetivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 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()

**Catálogo >**

**norm(***Matriz***)** ⇒ *expressão*
**norm(***Vector***)** ⇒ *expressão*

Apresenta a norma Frobenius.

| | |
|---|---|
| $\text{norm}\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$ | 5.47723 |
| $\text{norm}\begin{pmatrix}1 & 2\end{pmatrix}$ | 2.23607 |
| $\text{norm}\begin{pmatrix}1\\ 2\end{pmatrix}$ | 2.23607 |

### normCdf()

**Catálogo >**

**normCdf(***LimiteInferior***,***LimiteSuperior***[,μ[,σ]])** ⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

Calcula a probabilidade de distribuição normal entre *LimiteInferior* e *LimiteSuperior* para os μ (predefinição=0) e σ (predefinição=1) especificados.

Para P(X ≤ *LimiteSuperior*), defina *LimiteInferior* = **−9E999**.

### normPdf()

**Catálogo >**

**normPdf(***ValX* **[,μ[,σ]])** ⇒ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula a função de densidade de probabilidade para a distribuição normal num valor *ValX* especificado para μ e σ especificados.

### not

**Catálogo >**

**no t** *ExprBooleana* ⇒ *Expressão booleana*

Devolve falso, verdadeiro ou uma forma simplificada do argumento.

| | |
|---|---|
| not (2≥3) | true |
| not 0hB0▶Base16 | 0hFFFFFFFFFFFFFF4F |
| not not 2 | 2 |

## not

Catálogo >

**não** *NúmeroInteiro1* ⇒ *número inteiro*

Devolve um complemento de um número inteiro real. Internalmente, *NúmeroInteiro1* é convertido para um número de binário de 64 bits. O valor de cada bit é mudado (0 torna-se 1 e vice-versa) para um complemento. Os resultados aparecem de acordo com o modo base.

Pode introduzir o número em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respectivamente. Sem um prefixo, o número inteiro é tratado como decimal (base 10).

Se introduzir um número inteiro na base 10 muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Para mais informações, consulte ▶**Base2**, página 12.

No modo base Hex:

**Importante:** Zero, não a letra O.

| | |
|---|---|
| not 0h7AC36 | 0hFFFFFFFFFFF853C9 |

No modo base Bin:

| | |
|---|---|
| 0b100101▶Base10 | 37 |
| not 0b100101 | |
| 0b1111111111111111111111111111111111111111▶ | |
| not 0b100101▶Base10 | −38 |

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

**Nota:** Uma entrada binária pode ter até 64 dígitos (não contando com o prefixo 0b). Uma entrada hexadecimal pode ter até 16 dígitos.

## nPr()

Catálogo >

**nPr(***Valor1***,** *Valor2***)** ⇒ *expressão*

Para o número inteiro *Valor1* e *Valor2* com *Valor1* ≥ *Valor2* ≥ 0, **nPr()** é o número de permutações de coisas de *Valor1* retiradas de *Valor2* de uma vez.

**nPr(***Valor***, 0)** ⇒ **1**

**nPr(***Valor***,** *NúmeroInteiroNeg***)** ⇒ **1/((***Valor* **+1)** · **(***Valor* **+2)** ... **(***Valor* **−***NúmeroInteiroNeg***))**

**nPr(***Valor***,** *NúmeroInteiroPos***)**

⇒ *Valor* · **(***Valor* **−1)... (***Valor* **−***NúmeroInteiroPos* **+1)**

**nPr(***Valor***,** *NúmeroNãoInteiro***)**

⇒ *Valor***! / (***Valor* **−***NúmeroNãoInteiro***)!**

**nPr(***Lista1***,** *Lista2***)** ⇒ *lista*

Devolve uma lista de permutações com base nos pares de elementos correspondentes nas duas listas. Os argumentos têm de ter o mesmo tamanho de listas.

**nPr(***Matriz1***,** *Matriz2***)** ⇒ *matriz*

Devolve uma matriz de permutações com base nos pares de elementos correspondentes nas duas matrizes. Os argumentos têm de ter a a mesma matriz de tamanhos.

$$\mathrm{nPr}(z,3)\,|\,z=5 \qquad 60$$

$$\mathrm{nPr}(z,3)\,|\,z=6 \qquad 120$$

$$\mathrm{nPr}(\{5,4,3\},\{2,4,2\}) \qquad \{20,24,6\}$$

$$\mathrm{nPr}\left(\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}\right) \qquad \begin{bmatrix}30&20\\12&6\end{bmatrix}$$

$$\mathrm{nPr}(\{5,4,3\},\{2,4,2\}) \qquad \{20,24,6\}$$

$$\mathrm{nPr}\left(\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}\right) \qquad \begin{bmatrix}30&20\\12&6\end{bmatrix}$$

## npv()

Catálogo >

**npv(***TaxaDeJuro***,** *CFO***,** *ListaCF* [**,** *FreqCF* ]**)**

Função financeira que calcula o valor líquido actual; a soma dos valores actuais de entradas e saídas do cash flow. Um resultado positivo para npv indica um investimento lucrativo.

*TaxaDeJuro* é a taxa a descontar dos cash flows (o custo do dinheiro) durante um período.

*CF0* é o cash flow inicial no momento 0; tem de ser um número real.

*ListaCF* é uma lista de montantes de cash flow após o cash flow inicial *CF0*.

*FreqCF* é uma lista em que cada elemento especifica a frequência da ocorrência para um montante de cash flow agrupado (consecutivo), que é o elemento correspondente de *ListaCF*. A predefinição é 1; se introduzir valores, têm de ser números inteiros positivos < 10,000.

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

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

$\text{npv}(10,5000,list1,list2)$ 4769.91

## nSolve()

Catálogo >

**nSolve(***Equação***,** *Var* **[= ***Tentativa* **])** ⇒ *número ou erro da cadeia*

**nSolve(***Equação***,** *Var* **[= ***Tentativa* **],** *LimiteInferior***)** ⇒ *número ou erro da cadeia*

**nSolve(***Equação***,** *Var* **[= ***Tentativa* **],** *LimiteInferior***,** *LimiteSuperior***)** ⇒ *número ou erro da cadeia*

**nSolve(***Equação***,** *Var* **[= ***Tentativa* **]) |** *LimiteInferior*≤*Var*≤*LimiteSuperior* ⇒ *número ou erro da cadeia*

Procura iterativamente uma solução numérica real aproximada para *Equação* para uma variável. Especifique a variável como:

*variável*
– ou –
*variável = número real*

Por exemplo, x é válido e logo é x=3.

**nSolve()** tenta determinar se um ponto em que o residual é zero ou dois pontos relativamente próximos em que o residual tem sinais opostos e a magnitude do residual não é excessiva. Se não conseguir atingir isto com um número modesto de pontos de amostra, devolve a cadeira "nenhuma solução encontrada."

$\text{nSolve}(x^2+5\cdot x-25=9,x)$ 3.84429

$\text{nSolve}(x^2=4,x=-1)$ -2.

$\text{nSolve}(x^2=4,x=1)$ 2.

**Nota:** Se existirem várias soluções, pode utilizar uma tentativa para ajudar a encontrar uma solução particular.

$\text{nSolve}(x^2+5\cdot x-25=9,x)|x<0$ -8.84429

$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26,r\right)|r>0 \text{ and } r<0.25$ 0.006886

$\text{nSolve}(x^2=-1,x)$ "No solution found"

# O

**OneVar** **Catálogo >**

**OneVar** [ **1,** ] *X* [**,** [ *Freq* ][**,** *Categoria***,** *Incluir* ]]
**OneVar** [ *n***,** ] *X1***,** *X2* [ *X3* [**,** …[**,** *X20* ]]]

Calcula a estatística de 1 variável até 20 listas. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*Os argumentos X* são listas de dados.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada valor *X* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos para os valores *X* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Um elemento (nulo) vazio em qualquer das listas *X*, *Freq* ou *Category* resulta num nulo para o elemento correspondente de todas essas listas. Um elemento vazio em qualquer uma das listas de *X1* a *X20* resulta num vazio para o elemento correspondente de todas essas listas. Para mais informações sobre os elementos vazios, consulte a página 136.

| Variável de saída | Descrição |
|---|---|
| stat.$\bar{x}$ | Média dos valores x |
| stat.$\Sigma x$ | Soma dos valores x |
| stat.$\Sigma x^2$ | Soma dos valores x $^2$ |
| stat.sx | Desvio padrão da amostra de x |
| stat.$\sigma x$ | Desvio padrão da população de x |
| stat.n | Número de pontos de dados |
| stat.MinX | Mínimo dos valores x |
| stat.Q $_1$X | 1º quartil de x |
| stat.MedianX | Mediana de x |
| stat.Q $_3$X | 3º quartil de x |
| stat.MaxX | Máximo de valores x |
| stat.SSX | Soma de quadrados de desvios da média de x |

### or (ou)

Catálogo >

*ExprBooleana1* **or** *ExprBooleana2* devolve *expressão booleana*
*ListaBooleana1* **or** *ListaBooleana2* devolve *lista booleana*
*MatrizBooleana1* **or** *MatrizBooleana2* devolve *matriz booleana*

Devolve falso, verdadeiro ou uma forma simplificada da entrada original.

Devolve verdadeiro se uma ou ambas as expressões forem simplificadas para verdadeiro. Devolve falso apenas se ambas as expressões forem avaliadas para falso.

**Nota:** Consulte **xor**.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

*NúmeroInterior1* **or** *NúmeroInterior2* ⇒ *número inteiro*

Compara dois números inteiros reais bit a bit com uma operação or. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se ambos os bits forem 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respectivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 10).

Se introduzir um número inteiro na base 10 muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Para mais informações, consulte **▶Base2**, página 12.

**Nota:** Consulte **xor**.

| 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* |

No modo base Hex:

| 0h7AC36 or 0h3D5F | 0h7BD7F |
|---|---|

**Importante:** Zero, não a letra O.

No modo base Bin:

| 0b100101 or 0b100 | 0b100101 |
|---|---|

**Nota:** Uma entrada binária pode ter até 64 dígitos (não contando com o prefixo 0b). Uma entrada hexadecimal pode ter até 16 dígitos.

### ord()

Catálogo >

**ord(***Cadeia***)** ⇒ *número inteiro*
**ord(***Lista1***)** ⇒ *lista*

Devolve o código numérico do primeiro carácter na cadeia de caracteres *Cadeia* ou uma lista dos primeiros caracteres de cada elemento da lista.

| ord("hello") | 104 |
|---|---|
| char(104) | "h" |
| ord(char(24)) | 24 |
| ord({"alpha","beta"}) | {97,98} |

## P

### P▶Rx()

Catálogo >

**P▶Rx(***rExpr***, θ***Expr***)** ⇒ *expressão*
**P▶Rx(***rList***, θ***List***)** ⇒ *lista*
**P▶Rx(***rMatrix***, θ***Matrix***)** ⇒ *matriz*

Devolve a coordenada x equivalente do par (r, θ).

**Nota:** O argumento θ é interpretado como um ângulo expresso em graus, gradianos ou radianos de acordo com o modo de ângulo actual. Se o argumento for uma expressão, pode utilizar °, G ou r para substituir a definição do modo de ângulo temporariamente.

**Nota:** Pode introduzir esta função através da escrita de **P@>Rx(** ... **)** no teclado do computador.

No modo de ângulo Radianos:

| P▶Rx(4,60°) | 2. |
|---|---|
| P▶Rx({−3,10,1.3},{π/3,−π/4,0}) | {−1.5,7.07107,1.3} |

## P▶Ry()

**Catálogo >**

**P▶Ry(***rValue*, θ*Value*) ⇒ *valor*
**P▶Ry(***rList*, θ*List*) ⇒ *lista*
**P▶Ry(***rMatrix*, θ*Matrix*) ⇒ *matriz*

Devolve a coordenada y equivalente do par (r, θ).

**Nota:** O argumento θ é interpretado como um ângulo expresso em graus, gradianos ou radianos de acordo com o modo de ângulo actual.

**Nota:** Pode introduzir esta função através da escrita de **P@>Ry(** ... **)** no teclado do computador.

No modo de ângulo Radianos:

$\mathrm{P}\blacktriangleright\mathrm{Ry}(4,60°)$ 3.4641

$\mathrm{P}\blacktriangleright\mathrm{Ry}\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right)$
$\{-2.59808,-7.07107,0\}$

## PassErr

**Catálogo >**

**PassErr**

Passa um erro para o nível seguinte.

Se a variável do sistema *errCode* for zero, **PassErr** não faz nada.

A proposição **Else** do bloco **Try...Else...EndTry** deve utilizar **ClrErr** ou **PassErr**. Se tiver de processar ou ignorar o erro, utilize **ClrErr**. Se não souber o que fazer com o erro, utilize **PassErr** para o enviar para a rotina de tratamento de erros seguinte. Se não existirem mais rotinas de tratamento de erros **Try...Else...EndTry** pendente, a caixa de diálogo de erros aparecerá como normal.

**Nota:** Consulte também **ClrErr**, na página 17, e **Try**, na página 108.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições em diferentes linhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, mantenha premida a tecla **Alt** e prima **Enter**.

Para ver um exemplo de **PassErr**, consulte o exemplo 2 no comando **Try**, página 108.

## piecewise()

**Catálogo >**

**piecewise(***Expr1* [**,** *Condição1* [**,** *Expr2* [**,** *Condição2* [**,** … ]]]]**)**

Devolve as definições para uma função piecewise na forma de uma lista. Pode também criar definições piecewise com um modelo.

**Nota:** Consulte também **Modelo de Função por ramos**, página 2.

Define $p(x)=\begin{cases}x, & x>0\\ \text{undef}, & x\le 0\end{cases}$ *Done*

$p(1)$ 1

$p(-1)$ undef

## poissCdf()

**Catálogo >**

**poissCdf(**λ**,***LimiteInferior*,*LimiteSuperior***)** ⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas
**poissCdf(**λ**,***LimiteSuperior***)** (para P(0≤X≤*LimiteSuperior*) ⇒ *número* se *LimiteSuperior* for um número, lista se *LimiteSuperior* for uma lista

Calcula uma probabilidade cumulativa para a distribuição Poisson discreta com a média especificada λ.

Para P(X ≤ *LimiteSuperior*), defina *LimiteInferior*=0

## poissPdf()

**Catálogo >**

**poissPdf(**λ**,** *ValX***)** ⇒ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula uma probabilidade para a distribuição Poisson discreta com a média especificada λ.

## ▸Polar

Catálogo >

*Vector* **▸Polar**

**Nota:** Pode introduzir este operador através da escrita de **@>Polar** no teclado do computador.

Apresenta o *vector* em forma polar [r ∠θ]. O vector tem de ser de dimensão 2 e pode ser uma linha ou uma coluna.

**Nota: ▸Polar** é uma instrução de formato de visualização, não uma função de conversão. Só pode utilizá-la no fim de uma linha de entrada e não actualiza *ans*.

**Nota:** Consulte também **▸Rect**, página 84.

$[1 \quad 3.]\blacktriangleright\text{Polar}$ $\quad [3.16228 \quad \angle 71.5651]$

*ValorComplexo* **▸Polar**

Apresenta *VectorComplexo* em forma polar.

- O modo de ângulo Graus devolve (r ∠θ).
- O modo de ângulo Radianos devolve $re^{i\theta}$.

*ValorComplexo* pode ter qualquer forma complexa. No entanto, uma entrada $re^{i\theta}$ provoca um erro no modo de ângulo Graus.

**Nota:** Tem de utilizar os parêntesis para uma entrada polar (r ∠θ).

No modo de ângulo Radianos:

$(3+4\cdot i)\blacktriangleright\text{Polar}$ $\quad e^{.927295\cdot i}\cdot 5$

$\left(\left(4\angle\frac{\pi}{3}\right)\right)\blacktriangleright\text{Polar}$ $\quad e^{1.0472\cdot i}\cdot 4.$

No modo de ângulo Gradianos:

$(4\cdot i)\blacktriangleright\text{Polar}$ $\quad (4\angle 100)$

No modo de ângulo Graus:

$(3+4\cdot i)\blacktriangleright\text{Polar}$ $\quad (5\angle 53.1301)$

## polyEval()

Catálogo >

**polyEval(***Lista1***,** *Expr1***)** ⇒ *expressão*
**polyEval(***Lista1***,** *Lista2***)** ⇒ *expressão*

Interpreta o primeiro argumento como o coeficiente de um polinómio de grau descendente e devolve o polinómio avaliado para o valor do segundo argumento.

$\text{polyEval}(\{1,2,3,4\},2)$ $\quad 26$

$\text{polyEval}(\{1,2,3,4\},\{2,-7\})$ $\quad \{26,-262\}$

## polyRoots()

Catálogo >

**polyRoots(***Poli***,***Var***)** ⇒ *lista*
**polyRoots(***ListaDeCoeficientes***)** ⇒ *lista*

A primeira sintaxe, **polyRoots(***Poli***,***Var***)**, devolve uma lista de raízes reais do polinómio *Poly* em relação à variável *Var*. Se não existirem raízes reais, devolve uma lista vazia: {}.

*Poly* tem de ser um polinómio na forma expandida. Não utilize formatos não expandidos, como, por exemplo, $y^2\cdot y+1$ ou $x\cdot x+2\cdot x+1$

A segunda sintaxe, **polyRoots(***ListaDeCoeficientes***)**, devolve uma lista de raízes reais para os coeficientes em *ListaDeCoeficientes*.

**Nota:** Consulte também **cPolyRoots()**, página 23.

$\text{polyRoots}(y^3+1,y)$ $\quad \{-1\}$

$\text{cPolyRoots}(y^3+1,y)$ $\quad \{-1,0.5-0.866025\cdot i,0.5+0.866025\cdot i\}$

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

$\text{polyRoots}(\{1,2,1\})$ $\quad \{-1,-1\}$

**PowerReg** *X*,*Y* [, *Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão de potência $y = (a \cdot (x)^b)$ nas listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: $a \cdot (x)^b$ |
| stat.a, stat.b | Parâmetros de regressão |
| stat.$r^2$ | Coeficiente de determinação linear para dados transformados |
| stat.r | Coeficiente de correlação para dados transformados (ln(x), ln(y)) |
| stat.Resid | Resíduos associados ao modelo de potência |
| stat.ResidTrans | Resíduos associados ao ajuste linear dos dados transformados |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

## Prgm

**Catálogo >**

**Prgm**
*Bloco*
**EndPrgm**

Modelo para criar um programa definido pelo utilizador. Tem de ser utilizado o comando **Define**, **Define BibPub** ou **Define BibPriv**.

*Bloco* pode ser uma afirmação, uma série de afirmações separadas pelo carácter ":" ou uma série de afirmações em linhas separadas.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

Calcule o GCD e visualize os resultados intermédios.

```
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()

Consulte Π ( ) , página 129.

## Produto (PI)

Consulte Π (), página 129.

## product()

**Catálogo >**

**product(***Lista* [**,** *Início* [**,** *fim* ]]**)** ⇒ *expressão*

Apresenta o produto dos elementos contidos na *Lista*. *Início* e *Fim* são opcionais. Especificam um intervalo de elementos.

```
product({1,2,3,4})                          24
product({4,5,8,9},2,3)                      40
```

**product(***Matriz1* [**,** *Início* [**,** *fim* ]]**)** ⇒ *matriz*

Devolve um vector da linha com os produtos dos elementos nas colunas de *Matriz1*. *Início* e *Fim* são opcionais. Especificam um intervalo de linhas.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

```
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]
```

## propFrac()

**Catálogo >**

**propFrac(***Valor1* **[,** *Var* **])** ⇒ *valor*

**propFrac(***rational_number***)** devolve *rational_number* como a soma de um número inteiro e uma fracção com o mesmo sinal e uma magnitudade do denominador maior que a magnitude do numerador.

$$\text{propFrac}\left(\frac{4}{3}\right) \qquad 1+\frac{1}{3}$$

$$\text{propFrac}\left(\frac{-4}{3}\right) \qquad -1-\frac{1}{3}$$

**propFrac(***rational_expression***,** *Var***)** devolve a soma das fracções adequadas e um polinómio em relação a *Var*. O grau de *Var* no denominador excede o grau de *Var* no numerador em cada fracção adequada. As potências similares de *Var* são recolhidas. Os termos e os factores são ordenados com *Var* como variável principal.

Se omitir *Var*, uma expansão da fracção adequada é efectuada em relação à variável principal. Os coeficientes da parte polinominal são efectuados adequadamente em relação à primeira variável principal, etc.

Pode utilizar a função **propFrac()** para representar as fracções mistas e demonstrar a adição e a subtracção de fracções mistas.

$$\text{propFrac}\left(\frac{11}{7}\right) \qquad 1+\frac{4}{7}$$

$$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right) \qquad 8+\frac{37}{44}$$

$$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right) \qquad -2-\frac{29}{44}$$

# Q

## QR

**Catálogo >**

**QR** *Matriz***,** *MatrizQ***,** *MatrizR* **[,** *Tol***]**

Calcula a factorização QR Householder de uma matriz complexa ou real. As matrizes Q e R resultantes são guardados nos *Matriz* especificados. A matriz Q é unitária. A matriz R é triangular superior.

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância só é utilizada se a matriz tiver entradas de ponto flutuante e não contiver variáveis simbólicas sem um valor atribuído. Caso contrário, *Tol* é ignorado.

- Se utilizar **ctrl** **enter** ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética do ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida for calculada como:
  5E-14 · max(dim(*Matriz*)) · rowNorm(*Matriz*)

A factorização QR é calculada numericamente com as transformações Householder. A solução simbólica é calculada com Gram-Schmidt. As colunas em *qMatName* são vectores de base ortonormal que ligam o espaço definido pela *matriz*.

O número de ponto flutuante (9.) em m1 faz com que os resultados sejam calculados na forma de ponto flutuante.

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9.\end{bmatrix} \rightarrow m1 \qquad \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9.\end{bmatrix}$$

QR *m1*,*qm*,*rm* — *Done*

$$qm = \begin{bmatrix}0.123091 & 0.904534 & 0.408248\\ 0.492366 & 0.301511 & -0.816497\\ 0.86164 & -0.301511 & 0.408248\end{bmatrix}$$

$$rm = \begin{bmatrix}8.12404 & 9.60114 & 11.0782\\ 0. & 0.904534 & 1.80907\\ 0. & 0. & 0.\end{bmatrix}$$

ClearAZ — *Done*

**QuadReg** *X*,*Y* [, *Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão polinomial quadrática $y = a \cdot x^2+b \cdot x+c$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter dimensões iguais, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são incluídos no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: $a \cdot x^2+b \cdot x+c$ |
| stat.a, stat.b, stat.c | Parâmetros de regressão |
| stat.$R^2$ | Coeficiente de determinação |
| stat.Resid | Resíduos da regressão |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

**QuartReg** *X*,*Y* [, *Freq*] [, *Categoria*, *Incluir*]]

Calcula a regressão polinomial quártica $y = a \cdot x^4+b \cdot x^3+c \cdot x^2+d \cdot x+e$ a partir das listas *X* e *Y* com a frequência *Freq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: $a \cdot x^4+b \cdot x^3+c \cdot x^2+d \cdot x+e$ |
| stat.a, stat.b, stat.c, stat.d, stat.e | Parâmetros de regressão |
| stat.$R^2$ | Coeficiente de determinação |
| stat.Resid | Resíduos da regressão |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

# R

## R▸Pθ()

**Catálogo >**

**R▸Pθ(***xValue***,** *yValue***)** ⇒ *valor*
**R▸Pθ(***xList***,** *yList***)** ⇒ *lista*
**R▸Pθ(***xMatrix***,** *yMatrix***)** ⇒ *matriz*

Devolve a coordenada θ equivalente dos argumentos dos pares (*x*,*y*).

**Nota:** O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

**Nota:** Pode introduzir esta função através da escrita de **R@>Ptheta(** ... **)** no teclado do computador.

No modo de ângulo Graus:

$R▸P\theta(2,2)$ 45.

No modo de ângulo Gradianos:

$R▸P\theta(2,2)$ 50.

No modo de ângulo Radianos:

$R▸P\theta(3,2)$ 0.588003

$R▸P\theta\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right)$

$\begin{bmatrix}0. & 2.94771 & 0.643501\end{bmatrix}$

## R▸Pr()

**Catálogo >**

**R▸Pr(***xValue***,** *yValue***)** ⇒ *valor*
**R▸Pr(***xList***,** *yList***)** ⇒ *lista*
**R▸Pr(***xMatrix***,** *yMatrix***)** ⇒ *matriz*

Devolve a coordenada r equivalente dos argumentos dos pares (*x*,*y*).

**Nota:** Pode introduzir esta função através da escrita de **R@>Pr(** ... **)** no teclado do computador.

No modo de ângulo Radianos:

$R▸Pr(3,2)$ 3.60555

$R▸Pr\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right)$

$\begin{bmatrix}3 & 4.07638 & \frac{5}{2}\end{bmatrix}$

## ▸Rad

**Catálogo >**

*Valor1* **▸Rad** ⇒ *valor*

Converte o argumento para a medição do ângulo de radianos.

**Nota:** Pode introduzir esta função através da escrita de **@>Rad** no teclado do computador.

No modo de ângulo Graus:

$(1.5)▸Rad$ $(0.02618)^r$

No modo de ângulo Gradianos:

$(1.5)▸Rad$ $(0.023562)^r$

## rand()

**Catálogo >**

**rand()** ⇒ *expressão*
**rand(***#Tentativas***)** ⇒ *lista*

**rand()** devolve um valor aleatório entre 0 e 1.

**rand(***#Tentativas***)** devolve uma lista com # valores aleatórios entre 0 e 1.

Define a semente do número aleatório.

RandSeed 1147 *Done*

rand(2) {0.158206,0.717917}

## randBin()

**Catálogo >**

**randBin(***n*, *p***)** ⇒ *expressão*
**randBin(***n*, *p*, *#Tentativas***)** ⇒ *lista*

**randBin(***n*, *p***)** devolve um número real aleatório de uma distribuição binomial especificada.

**randBin(***n*, *p*, *#Trials***)** devolve uma lista com números reais aleatórios *#Tentativas* de uma distribuição binomial especificada.

| | |
|---|---|
| randBin(80,.5) | 34. |
| randBin(80,.5,3) | {47.,41.,46.} |

## randInt()

**Catálogo >**

**randInt(***LimiteInferior*, *LimiteSuperior***)** ⇒ *expressão*
**randInt(***LimiteInferior*, *LimiteSuperior*, *#Tentativas***)** ⇒ *lista*

**randInt(***LimiteInferior*, *LimiteSuperior***)** devolve um número inteiro aleatório no intervalo especificado pelos limites dos números inteiros *LimiteInferior* e *LimiteSuperior*.

**randInt(***LimiteInferior*, *LimiteSuperior*, *#Tentativas***)** devolve uma lista com # números inteiros aleatórios no intervalo especificado.

| | |
|---|---|
| randInt(3,10) | 7. |
| randInt(3,10,4) | {8.,9.,4.,4.} |

## randMat()

**Catálogo >**

**randMat(***LinhasNum*, *ColunasNum***)** ⇒ *matriz*

Devolve uma matriz de números inteiros entre -9 e 9 da dimensão especificada.

Ambos os argumentos têm de ser simplificados para números inteiros.

| | |
|---|---|
| RandSeed 1147 | *Done* |
| randMat(3,3) | $\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$ |

**Nota:** Os valores desta matriz mudam sempre que prime enter.

## randNorm()

**Catálogo >**

**randNorm(**μ, σ**)** ⇒ *expressão*
**randNorm(**μ, σ, *#Tentativas***)** ⇒ *lista*

Devolve um número decimal da distribuição normal específica. Pode ser qualquer número real, mas estará fortemente concentrado no intervalo [ μ− 3 · σ, μ+3 · σ].

**randNorm(**μ, σ, *#Tentativas***)** devolve uma lista com números decimais *#Tentativas* de uma distribuição normal especificada.

| | |
|---|---|
| RandSeed 1147 | *Done* |
| randNorm(0,1) | 0.492541 |
| randNorm(3,4.5) | −3.54356 |

## randPoly()

**Catálogo >**

**randPoly(***Var*, *Ordem***)** ⇒ *expressão*

Devolve um polinómio em *Var* da *Ordem* especificada. Os coeficientes são números inteiros aleatórios no intervalo de −9 a 9. O coeficiente superiornão será zero.

*Ordem* tem de ser 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()

**Catálogo >**

**randSamp(***Lista*,*#Tentativas*[,*SemSubstituição*]**)** ⇒ *lista*

Devolve uma lista com uma amostra aleatória de tentativas *#Tentativas* de *Lista* com uma opção para substituição da amostra (*SemSubstituição*=0) ou sem substituição da amostra (*SemSubstituição*=1). A predefinição é com substituição da amostra.

| | |
|---|---|
| Define *list3*={1,2,3,4,5} | *Done* |
| Define *list4*=randSamp(*list3*,6) | *Done* |
| *list4* | {5.,1.,3.,3.,4.,4.} |

## RandSeed

Catálogo >

**RandSeed** *Número*

Se *Número* = 0, define as sementes para as predefinições de fábrica para o gerador de números aleatórios. Se *Número* ≠ 0, é utilizado para gerar duas sementes, que são guardadas nas variáveis do sistema seed1 e seed2.

RandSeed 1147 → *Done*

rand() → 0.158206

## real()

Catálogo >

**real(***Valor1***)** ⇒ *valor*

Devolve a parte real do argumento.

**Nota:** Todas as variáveis indefinidas são tratadas como variáveis reais. Consulte também **imag()**, página 46.

real(2+3·*i*) → 2

**real(***Lista1***)** ⇒ *lista*

Devolve as partes reais de todos os elementos.

real({1+3·*i*,3,*i*}) → {1,3,0}

**real(***Matriz1***)** ⇒ *matriz*

Devolve as partes reais de todos os elementos.

$$\text{real}\begin{bmatrix} 1+3\cdot i & 3 \\ 2 & i \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 3 \\ 2 & 0 \end{bmatrix}$$

## ▸Rect

Catálogo >

*Vector* **▸Rect**

**Nota:** Pode introduzir este operador através da escrita de **@>Rect** no teclado do computador.

Apresenta o *Vector* na forma rectangular [x, y, z]. O vector tem de ser de dimensão 2 ou 3 e pode ser uma linha ou uma coluna.

$$\begin{bmatrix} 3 & \angle\frac{\pi}{4} & \angle\frac{\pi}{6} \end{bmatrix} \blacktriangleright \text{Rect} \rightarrow \begin{bmatrix} 1.06066 & 1.06066 & 2.59808 \end{bmatrix}$$

**Nota:** **▸Rect** é uma instrução de formato de visualização, não uma função de conversão. Só pode utilizá-la no fim de uma linha de entrada e não actualiza *ans*.

**Nota:** Consulte também **▸Polar**, página 76.

*ValorComplexo* **▸Rect**

Apresenta o *ValorComplexo* na forma rectangular a+bi. O *ValorComplexo* pode ter qualquer forma complexa. No entanto, uma entrada $re^{i\theta}$ provoca um erro no modo de ângulo Graus.

**Nota:** Tem de utilizar os parêntesis para uma entrada polar (r ∠θ).

No modo de ângulo Radianos:

$$\left(4\cdot e^{\frac{\pi}{3}}\right) \blacktriangleright \text{Rect} \rightarrow 11.3986$$

$$\left(4 \angle \frac{\pi}{3}\right) \blacktriangleright \text{Rect} \rightarrow 2.+3.4641\cdot i$$

No modo de ângulo Gradianos:

$$(1 \angle 100) \blacktriangleright \text{Rect} \rightarrow i$$

No modo de ângulo Graus:

$$(4 \angle 60) \blacktriangleright \text{Rect} \rightarrow 2.+3.4641\cdot i$$

**Nota:** Para escrever ∠, seleccione-o na lista de símbolos no Catálogo.

## ref()

Catálogo >

**ref(***Matriz1* **[,** *Tol* **])** ⇒ *matriz*

Devolve a forma de escalão-linha de *Matriz1*.

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância só é utilizada se a matriz tiver entradas de ponto flutuante e não contiver variáveis simbólicas sem um valor atribuído. Caso contrário, *Tol* é ignorado.

- Se utilizar ctrl enter ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética do ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida for calculada como:
  5E−14 · max(dim(*Matriz1*)) · rowNorm(*Matriz1*)

Evite elementos indefinidos em *Matriz1*. Podem conduzir a resultados imprevistos.

Por exemplo, se *a* for indefinido na expressão seguinte, aparece uma mensagem de aviso e o resultado aparece como:

$$\text{ref}\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

O aviso aparece porque o elemento generalizado 1/*a* não seria válido para *a*=0.

Pode evitar isto guardando um valor para *a* anteriormente ou utilizando o operador de limite ("|") para substituir um valor, conforme indicado no exemplo seguinte.

$$\text{ref}\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mid a = 0 \Rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

**Nota:** Consulte também **rref()**, página 90.

$$\text{ref}\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix} \qquad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

## remain()

Catálogo >

**remain(***Valor1***,** *Valor2***)** ⇒ *valor*
**remain(***Lista1***,** *Lista2***)** ⇒ *lista*
**remain(***Matriz1***,** *Matriz2***)** ⇒ *matriz*

Devolve o resto do primeiro argumento em relação ao segundo argumento conforme definido pelas identidades:

remain(x,0) x
remain(x,y) x − y · iPart(x/y)

Por consequência, não se esqueça de que **remain(**−x,y**)** − **remain(**x,y**)**. O resultado é zero ou tem o mesmo sinal do primeiro argumento.

**Nota:** Consulte também **mod()**, página 64.

| | |
|---|---|
| $\text{remain}(7,0)$ | $7$ |
| $\text{remain}(7,3)$ | $1$ |
| $\text{remain}(-7,3)$ | $-1$ |
| $\text{remain}(7,-3)$ | $1$ |
| $\text{remain}(-7,-3)$ | $-1$ |
| $\text{remain}(\{12,-14,16\},\{9,7,-5\})$ | $\{3,0,1\}$ |

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

**Request** *promptString*, *var*[, *DispFlag* [, *statusVar*]]
**Request** *promptString*, *func*(*arg1*, *...argn*)
[, *DispFlag* [, *statusVar*]]

Programar comando Interrompe o programa e mostra uma caixa de diálogo com a mensagem *CadeiaDePedido* e uma caixa de entrada para a resposta do utilizador.

Quando o utilizador escrever uma resposta e clicar em **OK**, os conteúdos da caixa de entrada são atribuídos à variável *var*.

Se o utilizador clicar em **Cancelar**, o programa continua sem aceitar qualquer entrada. O programa utiliza o valor anterior de *var* se *var* já tiver definida.

O argumento *MostrarMarcador* opcional pode ser qualquer expressão.

- Se omitir *MostrarMarcador* e avaliar para **1**, a mensagem do pedido e a resposta do utilizador aparecem no histórico da Calculadora.
- Se *MostrarMarcador* avaliar para **0**, o pedido e a resposta não aparecem no histórico.

O argumento *statusVar* opcional proporciona uma forma de determinar como o utilizador ignorou a caixa de diálogo. Atente que *statusVar* requer o argumento *DispFlag*.

- Se o utilizador tiver clicado em **OK** ou premido **Enter** ou **Ctrl+Enter**, a variável *statusVar* é definida para um valor de **1**.
- Caso contrário, a variável *statusVar* é definida para um valor de **0**.

O argumento *func*() permite a um programa guardar a resposta do utilizador como uma definição da função. Esta sintaxe funciona como se o utilizador executasse o comando:

Definir *func*(*arg1*, *...argn*) = *resposta do utilizador*

O programa pode utilizar a função definida *func*(). A *CadeiaDoPedido* deve orientar o utilizador para introduzir a *resposta do utilizador* adequada que complete a definição da função.

**Nota:** Pode utilizar o comando **Request** num programa definido pelo utilizador, mas não dentro de uma função.

Para parar um programa que contém um comando **Request** dentro de um ciclo infinito:

- **Windows®:** Prima continuamente a tecla **F12** e prima repetidamente **Enter**.
- **Macintosh®:** Prima continuamente a tecla **F5** e prima repetidamente **Enter**.
- **Unidade portátil:** Prima continuamente a tecla [on] e prima repetidamente [enter].

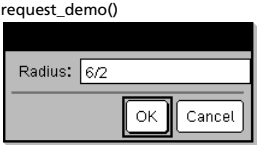
**Nota:** Consulte também **RequestStr**, página 87.

Definir um programa:

```
Definir request_demo()=Prgm
  Pedir "Raio: ",r
  Disp "Área = ",pi*r²
EndPrgm
```

Executar o programa e escrever uma resposta:

```
request_demo()
```




Resultado depois de seleccionar **OK**:

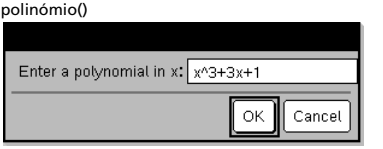
```
Raio:  6/2
Área= 28.2743
```

Definir um programa:

```
Definir polynomial()=Prgm
  Pedir "Introduzir um polinómio em x:",p(x)
  Mostrar "Raízes reais são:",polyRoots(p(x),x)
EndPrgm
```

Executar o programa e escrever uma resposta:

```
polinómio()
```




Resultado depois de seleccionar **OK**:

```
Introduzir um polinómio em x: x^3+3x+1
Raízes reais são: {-0.322185}
```

## RequestStr

Catálogo >

**RequestStr** *CadeiaDoPedido*, *var*[, *MostrarMarcador*]

Programar comando: Opera de modo idêntico à primeira sintaxe do comando **Request**, excepto se a resposta do utilizador for sempre interpretada como uma cadeia. Por contraste, o comando **Request** interpreta a resposta como uma expressão, excepto se o utilizador a colocar entre aspas (" ").

**Nota:** Pode utilizar o comando **RequestStr** num programa definido pelo utilizador, mas não numa função.

Para parar um programa que contém um comando **RequestStr** dentro de um ciclo infinito:

- **Windows®:** Prima continuamente a tecla **F12** e prima repetidamente **Enter**.
- **Macintosh®:** Prima continuamente a tecla **F5** e prima repetidamente **Enter**.
- **Unidade portátil:** Prima continuamente a tecla [on] e prima repetidamente [enter].

**Nota:** Consulte também **Request**, página 86.

Definir um programa:

```
Definir requestStr_demo()=Prgm
 RequestStr "Nome:",nome,0
 Mostrar "Resposta tem ",caracteres(nome),"
ocultos."
EndPrgm
```

Executar o programa e escrever uma resposta:

```
requestStr_demo()
```


Resultado depois de seleccionar **OK** (Não se esqueça de que o argumento *MostrarMarcador* de **0** omite o pedido e a resposta do histórico):

```
requestStr_demo()
                    A resposta tem 5 caracteres.
```

## Return

Catálogo >

**Return** [ *Expr*]

Devolve *Expr* como resultado da função. Utilize num bloco **Func** ... **EndFunc**.

**Nota:** Utilize **Voltar** sem um argumento num bloco **Prgm**...**EndPrgm** para sair de um programa.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo [↵] em vez de [enter] no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
Define factoral(nn)=Func
                      Local answer,count
                      1→answer
                      For count,1,nn
                      answer·count→answer
                      EndFor
                      Return answer
                      EndFunc
                                        Done
factoral(3)                                6
```

## right()

Catálogo >

**right(***Lista1* [**,** *Num* ]**)** ⇒ *lista*

Devolve os elementos *Num* mais à direita contidos em *Lista1*.

Se omitir *Num*, devolve todos os elementos de *Lista1*.

**right(***sourceString* [**,** *Num* ]**)** ⇒ *cadeia*

Devolve os caracteres *Num* mais à direita na cadeia de caracteres *sourceString*.

Se omitir *Num*, devolve todos os caracteres de *sourceString*.

**right(***Comparação***)** ⇒ *expressão*

Devolve o lado direito de uma equação ou desigualdade.

```
right({1,3,-2,4},3)                {3,-2,4}

right("Hello",2)                        "lo"
```

## rk23()

Catálogo >

**rk23(***Expr*, *Var*, *depVar*, {*Var0*, *VarMax*}, *depVar0*, *VarStep* [, *diftol*]**)** ⇒ *matriz*

**rk23(***SystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *diftol*]**)** ⇒ *matriz*

**rk23(***ListOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*}, *ListOfDepVars0*, *VarStep* [, *diftol*]**)** ⇒ *matriz*

Utiliza o método Runge-Kutta para resolver o sistema

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

com *depVar*(*Var0*)=*depVar0* no intervalo [*Var0*,*VarMax*]. Apresenta uma matriz cuja primeira fila define os valores de saída *Var* conforme definido por *VarStep*. A segunda fila define o valor da primeira componente da solução nos valores *Var* correspondentes, e assim por diante.

*Expr* é o segundo membro que define a equação diferencial ordinária (EDO).

*SystemOfExpr* é o sistema de segundos membros que definem o sistema de EDOs (corresponde à ordem de variáveis dependentes em *ListOfDepVars*).

*ListOfExpr* é uma lista de segundos membros que definem o sistema de EDOs (corresponde à ordem de variáveis dependentes em *ListOfDepVars*).

*Var* é a variável independente.

*ListOfDepVars* é uma lista de variáveis dependentes.

{*Var0*, *VarMax*} é uma lista de dois elementos que informa a função para integrar de *Var0* a *VarMax*.

*ListOfDepVars0* é uma lista de valores iniciais para variáveis dependentes.

Se *VarStep* avalia para um número diferente de zero: sinal(*VarStep*) = sinal(*VarMax*-*Var0*) e soluções são apresentadas em *Var0*+i**VarStep* para todos os i=0,1,2,… tal como *Var0*+i**VarStep* está em [*var0*,*VarMax*] (pode não obter um valor de solução em *VarMax*).

se *VarStep* avaliar para zero, as soluções são apresentadas nos valores *Var* Runge-Kutta".

*diftol* é a tolerância de erro (passa para 0.001).

Equação diferencial:
y'=0.001*y*(100-y) e y(0)=10

$$\text{rk23}\left(0.001\cdot y\cdot(100-y),t,y,\{0,100\},10,1\right)$$

| | | | | |
|---|---|---|---|---|
| 0. | 1. | 2. | 3. | 4. |
| 10. | 10.9367 | 11.9493 | 13.042 | 14.2 |

Para ver o resultado completo, prima ▲ e, de seguida, utilize ◀ e ▶ para mover o cursor.

Mesma equação com *diftol* definido para 1.E-6

$$\text{rk23}\left(0.001\cdot y\cdot(100-y),t,y,\{0,100\},10,1,1.\text{E}{-6}\right)$$

| | | | | |
|---|---|---|---|---|
| 0. | 1. | 2. | 3. | 4. |
| 10. | 10.9367 | 11.9495 | 13.0423 | 14.2189 |

Sistema de equações:
$\begin{cases} y1' = -y1 + 0.1\cdot y1\cdot y2 \\ y2' = 3\cdot y2 - y1\cdot y2 \end{cases}$
com *y1*(0)=2 e *y2*(0)=5

$$\text{rk23}\left(\begin{cases} -y1+0.1\cdot y1\cdot y2 \\ 3\cdot y2-y1\cdot y2 \end{cases},t,\{y1,y2\},\{0,5\},\{2,5\},1\right)$$

| | | | | |
|---|---|---|---|---|
| 0. | 1. | 2. | 3. | 4. |
| 2. | 1.94103 | 4.78694 | 3.25253 | 1.82848 |
| 5. | 16.8311 | 12.3133 | 3.51112 | 6.27245 |

## root()

Catálogo >

**root(***Valor***)** ⇒ *raiz*

**root(***Valor1*, *Valor2***)** ⇒ *raiz*

**root(***Valor***)** devolve a raiz quadrada de *Valor*.

**root(***Valor1*, *Valor2***)** devolve a raiz de *Valor2* de *Valor1*. *Valor1* pode ser uma constante de ponto flutuante complexa ou uma constante racional complexa ou número inteiro.

**Nota:** Consulte também **Modelo da raiz de índice N**, página 1.

$$\sqrt[3]{8} \qquad 2$$

$$\sqrt[3]{3} \qquad 1.44225$$

## rotate()

Catálogo >

**rotate(***NúmeroInteiro1* **[,** *#deRotações* **])** ⇒ *número inteiro*

Roda os bits num número inteiro binário. Pode introduzir *NúmeroInteiro1* em qualquer base numérica; é convertido automaticamente para uma forma binária de 64 bits assinada. Se a magnitude de *NúmeroInteiro1* for muito grande para esta forma, uma operação do módulo simétrico coloca-o no intervalo. Para mais informações, consulte ▶**Base2**, página 12.

Se *#deRotações* for positivo, a rotação é para a esquerda. Se *#deRotações* for negativo, a rotação é para a direita. A predefinição é −1 (rodar um bit para a direita).

Por exemplo, numa rotação para a direita:

Cada bit roda para a direita.

0b00000000000000111101011000011010 1

O bit mais à direita roda para o extremo esquerdo.

produz:

0b10000000000000011110101100001101 0

O resultado aparece de acordo com o modo base.

**rotate(***Lista1* **[,** *#deRotações* **])** ⇒ *lista*

Devolve uma cópia de *Lista1* rodada para a direita ou para a esquerda pelos elementos *#deRotações*. Não altere *Lista1*.

Se *#deRotações* for positivo, a rotação é para a esquerda. Se *#deRotações* for negativo, a rotação é para a direita. A predefinição é −1 (rodar um elemento para a direita).

**rotate(***Cadeia1* **[,** *#deRotações* **])** ⇒ *cadeia*

Devolve uma cópia de *Cadeia1* rodada para a direita ou para a esquerda pelos caracteres *#deRotações*. Não altere *Cadeia1*.

Se *#deRotações* for positivo, a rotação é para a esquerda. Se *#deRotações* for negativo, a rotação é para a direita. A predefinição é −1 (rodar um carácter para a direita).

No modo base Bin:

rotate(0b11111111111111111111111111111111) → 0b10000000000000000000000000000001

rotate(256,1) → 0b1000000000

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

No modo base Hex:

rotate(0h78E) → 0h3C7

rotate(0h78E,-2) → 0h80000000000001E3

rotate(0h78E,2) → 0h1E38

**Importante:** Para introduzir um número binário ou hexadecimal, utilize sempre o prefixo 0b ou 0h (zero, não a letra O).

No modo base Dec:

rotate({1,2,3,4}) → {4,1,2,3}

rotate({1,2,3,4},-2) → {3,4,1,2}

rotate({1,2,3,4},1) → {2,3,4,1}

rotate("abcd") → "dabc"

rotate("abcd",-2) → "cdab"

rotate("abcd",1) → "bcda"

## round()

Catálogo >

**round(***Valor1* **[,** *dígitos* **])** ⇒ *valor*

Devolve o argumento arredondado para o número especificado de dígitos após o ponto decimal.

*dígitos* tem de ser um número inteiro no intervalo 0–12. Se *dígitos* não for incluído, devolve o argumento arredondado para 12 dígitos significantes.

**Nota:** A visualização do modo de dígitos pode afectar como este é apresentado.

round(1.234567,3) → 1.235

**round(***Lista1* **[,** *dígitos* **])** ⇒ *lista*

Devolve uma lista dos elementos arredondado para o número especificado de dígitos.

round({π,√2,ln(2)},4) → {3.1416,1.4142,0.6931}

**round(***Matriz1* **[,** *dígitos* **])** ⇒ *matriz*

Devolve uma matriz dos elementos arredondados para o número especificado de dígitos.

$$\text{round}\left(\begin{bmatrix}\ln(5) & \ln(3)\\ \pi & e^{1}\end{bmatrix},1\right) \rightarrow \begin{bmatrix}1.6 & 1.1\\ 3.1 & 2.7\end{bmatrix}$$

## rowAdd()

Catálogo >

**rowAdd(***Matriz1***,** *rIndex1***,** *rIndex2***)** ⇒ *matriz*

Devolve uma cópia de *Matriz1* com a linha *rIndex2* substituída pela soma das linhas *rIndex1* e *rIndex2*.

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

## rowDim()

Catálogo >

**rowDim(***Matriz***)** ⇒ *expressão*

Devolve o número de linhas em *Matriz*.

**Nota:** Consulte também **colDim()**, página 17.

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

$$\text{rowDim}(m1) \qquad 3$$

## rowNorm()

Catálogo >

**rowNorm(***Matriz***)** ⇒ *expressão*

Devolve o máximo das somas dos valores absolutos dos elementos nas linhas em *Matriz*.

**Nota:** Todos os elementos da matriz têm de ser simplificados para números. Consulte também **colNorm()**, página 17.

$$\text{rowNorm}\left(\begin{bmatrix}-5 & 6 & -7\\ 3 & 4 & 9\\ 9 & -9 & -7\end{bmatrix}\right) \qquad 25$$

## rowSwap()

Catálogo >

**rowSwap(***Matriz1***,** *rIndex1***,** *rIndex2***)** ⇒ *matriz*

Devolve *Matriz1* com as linhas *rIndex1* e *rIndex2* trocadas.

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

$$\text{rowSwap}(mat,1,3) \qquad \begin{bmatrix}5 & 6\\ 3 & 4\\ 1 & 2\end{bmatrix}$$

## rref()

Catálogo >

**rref(***Matriz1* **[,** *Tol* **])** ⇒ *matriz*

Devolve a forma de escalão-linha reduzida de *Matriz1*.

$$\text{rref}\left(\begin{bmatrix}-2 & -2 & 0 & -6\\ 1 & -1 & 9 & -9\\ -5 & 2 & 4 & -4\end{bmatrix}\right) \qquad \begin{bmatrix}1 & 0 & 0 & \frac{66}{71}\\ 0 & 1 & 0 & \frac{147}{71}\\ 0 & 0 & 1 & \frac{-62}{71}\end{bmatrix}$$

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância só é utilizada se a matriz tiver entradas de ponto flutuante e não contiver variáveis simbólicas sem um valor atribuído. Caso contrário, *Tol* é ignorado.

- Se utilizar ctrl enter ou definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética do ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida for calculada como:
  5E−14 • max(dim(*Matriz1*)) • rowNorm(*Matriz1*)

**Nota:** Consulte também **ref()**, página 85.

# S

## sec()

**Tecla** trig

**sec(***Valor1***)** ⇒ *valor*
**sec(***Lista1***)** ⇒ *lista*

Devolve a secante de *Valor1* ou devolve uma lista com as secantes de todos os elementos em *Lista1*.

**Nota:** O argumento é interpretado como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual. Pode utilizar °, G ou r para substituir o modo de ângulo temporariamente.

No modo de ângulo Graus:

$\sec(45)$ 1.41421

$\sec(\{1,2.3,4\})$ $\{1.00015,1.00081,1.00244\}$

## sec$^{-1}$()

**Tecla** trig

**sec$^{-1}$(***Valor1***)** ⇒ *valor*
**sec$^{-1}$(***Lista1***)** ⇒ *lista*

Devolve o ângulo cuja secante é *Valor1* ou devolve uma lista com as secantes inversas de cada elemento de *Lista1*.

**Nota:** O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

**Nota:** Pode introduzir esta função através da escrita de **arcsec(…)** no teclado do computador.

No modo de ângulo Graus:

$\sec^{-1}(1)$ 0

No modo de ângulo Gradianos:

$\sec^{-1}(\sqrt{2})$ 50

No modo de ângulo Radianos:

$\sec^{-1}(\{1,2,5\})$ $\{0,1.0472,1.36944\}$

## sech()

**Catálogo >**

**sech(***Valor1***)** ⇒ *valor*
**sech(***Lista1***)** ⇒ *lista*

Devolve a secante hiperbólica de *Valor1* ou devolve uma lista com as secantes hiperbólicas dos elementos *Lista1*.

$\text{sech}(3)$ 0.099328

$\text{sech}(\{1,2.3,4\})$ $\{0.648054,0.198522,0.036619\}$

## sech$^{-1}$()

**Catálogo >**

**sech$^{-1}$(***Valor1***)** ⇒ *valor*
**sech$^{-1}$(***Lista1***)** ⇒ *lista*

Devolve a secante hiperbólica inversa de *Valor1* ou devolve uma lista com as secantes hiperbólicas inversas de cada elemento de *Lista1*.

**Nota:** Pode introduzir esta função através da escrita de **arcsech(…)** no teclado do computador.

No modo de ângulo Radianos e Formato complexo rectangular:

$\text{sech}^{-1}(1)$ 0

$\text{sech}^{-1}(\{1,-2,2.1\})$ $\{0,2.0944\cdot i,8.\text{E}^{-}15+1.07448\cdot i\}$

## seq()

Catálogo >

**seq(***Expr*, *Var*, *Baixo*, *Alto* [, *Passo* ]) ⇒ *lista*

Incrementa *Var* de *Baixo* até *Alto* por um incremento de *Passo*, avalia *Expr* e apresenta os resultados como uma lista. O conteúdo original de *Var* ainda está aqui após a conclusão de **seq()**.

O valor predefinido para *Passo* = 1.

$\text{seq}\left(n^2,n,1,6\right)$ $\quad \{1,4,9,16,25,36\}$

$\text{seq}\left(\frac{1}{n},n,1,10,2\right)$ $\quad \left\{1,\frac{1}{3},\frac{1}{5},\frac{1}{7},\frac{1}{9}\right\}$

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$ $\quad \frac{1968329}{1270080}$

Prima **Ctrl**+**Enter** ctrl enter (Macintosh®: ⌘+**Enter**) para avaliar:

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$ $\quad 1.54977$

## seqGen()

Catálogo >

**seqGen(***Expr*, *Var*, *depVar*, {*Var0*, *VarMax*}[, *ListOfInitTerms* [, *VarStep* [, *CeilingValue*]]]) ⇒ *lista*

Gera uma lista de termos para sequência *depVar*(*Var*)=*Expr* da seguinte forma: Incrementa a variável independente *Var* de *Var0* até *VarMax* por *VarStep*, avalia *depVar*(*Var*) para os valores correspondentes de *Var* utilizando a fórmula *Expr* e *ListOfInitTerms* e apresenta os resultados como uma lista.

**seqGen(***ListOrSystemOfExpr*, *Var*, *ListOfDepVars*, {*Var0*, *VarMax*} [, *MatrixOfInitTerms* [, *VarStep* [, *CeilingValue*]]]) ⇒ *matriz*

Gera uma matriz de termos de um sistema (ou lista) de sequências *ListOfDepVars*(*Var*)=*ListOrSystemOfExpr* da seguinte forma: Incrementa a variável independente *Var* de *Var0* até *VarMax* por *VarStep*, avalia *ListOfDepVars*(*Var*) para os valores correspondentes de *Var* utilizando a fórmula *ListOrSystemOfExpr* e *MatrixOfInitTerms* e apresenta os resultados como uma matriz.

O conteúdo original de *Var* está inalterado após a conclusão de **seqGen()**.

O valor predefinido para *VarStep* = **1**.

Gere o primeiros 5 termos da sequência $u(n) = u(n\text{-}1)^2/2$, com $u(1)$=**2** e *VarStep*=**1**.

$\text{seqGen}\left(\frac{\left(u(n-1)\right)^2}{n},n,u,\{1,5\},\{2\}\right)$

$\left\{2,2,\frac{4}{3},\frac{4}{9},\frac{16}{405}\right\}$

Exemplo no qual Var0=2:

$\text{seqGen}\left(\frac{u(n-1)+1}{n},n,u,\{2,5\},\{3\}\right)$

$\left\{3,\frac{4}{3},\frac{7}{12},\frac{19}{60}\right\}$

Sistema de duas sequências:

$\text{seqGen}\left(\left\{\frac{1}{n},\frac{u2(n-1)}{2}+u1(n-1)\right\},n,\{u1,u2\},\{1,5\},\begin{bmatrix}\_\\2\end{bmatrix}\right)$

$\begin{bmatrix}1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5}\\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24}\end{bmatrix}$

Nota: O Vazio (_) na matriz do termo inicial acima, é utilizado para indicar que o 1º termo para u1(n) é calculado utilizando a fórmula de sucessão u1(n)=1/n.

## seqn()

**Catálogo >**

**seqn(***Expr***(u, ***n* **[, ***ListOfInitTerms***[, ***nMax* **[, ***CeilingValue***]]]) ⇒** *lista*

Gera uma lista de termos para uma sucessão *u(n)=Expr(u, n)*, da seguinte forma: Incrementa *n* a partir de 1 até *nMax* por 1, avalia *u*(*n*) para os valores correspondentes de *n* utilizando a fórmula *Expr*(*u*, *n*) e *ListOfInitTerms* e apresenta os resultados como uma lista.

**seqn(***Expr***(***n* **[, ***nMax* **[, ***CeilingValue***]]) ⇒** *lista*

Gera uma lista de termos para uma sucessão não recorrente *u(n)=Expr(n)*, da seguinte forma: Incrementa *n* a partir de 1 até *nMax* por 1, avalia *u*(*n*) para os valores correspondentes de *n* utilizando a fórmula *Expr(n)* e apresenta os resultados como uma lista.

Se *nMax* estiver em falta, *nMax* é definido para 2500

Se *nMax*=0, *nMax* é definido para 2500

**Nota: seqn()** chamadas **seqGen( )** com *n0*=**1** e *nstep* =**1**

Gere o primeiros 6 termos da sequência *u*(*n*) = *u*(*n*-1)/2, com *u*(1)=**2**.

$$\text{seqn}\left(\frac{u(n-1)}{n},\{2\},6\right) \qquad \left\{2,1,\frac{1}{3},\frac{1}{12},\frac{1}{60},\frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2},6\right) \qquad \left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16},\frac{1}{25},\frac{1}{36}\right\}$$

## setMode()

**Catálogo >**

**setMode(***NúmeroInteiroNomeModo***,** *NúmeroInteiroDefinição***)** ⇒ *número inteiro*
**setMode(***lista***)** ⇒ *lista de números inteiros*

Válido apenas numa função ou num programa.

**setMode(***NúmeroInteiroNomeModo***,** *NúmeroInteiroDefinição***)** define temporariamente o modo *NúmeroInteiroNomeModo* para a nova definição *NúmeroInteiroDefinição* e devolve um número inteiro correspondente à definição original desse modo. A alteração é limitada à duração da execução do programa/função.

*NúmeroInteiroNomeModo* especifica que modo quer definir. Tem de ser um dos números inteiros do modo da tabela abaixo.

*NúmeroInteiroDefinição* especifica a nova definição do modo. Tem de ser um dos números inteiros da definição listados abaixo para o modo específico que está a definir.

**setMode(***lista***)** permite alterar várias definições. *lista* contém os pares de números inteiros do modo e da lista. **setMode(***lista***)** devolve uma lista similar cujos pares de números inteiros representam as definições e os modos originais.

Se guardou todas as definições do modo com **getMode(0) ➞** *var*, pode utilizar **setMode(***var***)** para restaurar essas definições até sair da função ou do programa. Consulte **getMode()**, página 42.

**Nota:** As definições do modo actual são passadas para subrotinas. Se uma subrotina alterar uma definição do modo, a alteração do modo perder-se--á quando o controlo voltar à rotina.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo [↵] em vez de [enter] no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

Apresente o valor aproximado de π com a predefinição para Ver dígitos e apresente π com uma definição de Fix2. Certifique-se de que a predefinição é restaurada após a execução do programa.

```
Define prog1()=Prgm                    Done
                Disp π
                setMode(1,16)
                Disp π
                EndPrgm
prog1()
                                       3.14159
                                       3.14
                                       Done
```

| Nome do modo | Número inteiro do modo | Números inteiros da definição |
|---|---|---|
| Ver dígitos | **1** | **1** =Flutuante, **2** =Flutuante1, **3** =Flutuante2, **4** =Flutuante3, **5** =Flutuante4, **6** =Flutuante5, **7** =Flutuante6, **8** =Flutuante7, **9** =Flutuante8, **10** =Flutuante9, **11** =Flutuante10, **12** =Flutuante11, **13** =Flutuante12, **14** =Fixo0, **15** =Fixo1, **16** =Fixo2, **17** =Fixo3, **18** =Fixo4, **19** =Fixo5, **20** =Fixo6, **21** =Fixo7, **22** =Fixo8, **23** =Fixo9, **24** =Fixo10, **25** =Fixo11, **26** =Fixo12 |
| Ângulo | **2** | **1** =Radianos, **2** =Graus, **3** =Gradianos |
| Formato exponencial | **3** | **1** =Normal, **2** =Científica, **3** =Engenharia |
| Real ou Complexo | **4** | **1** =Real, **2** =Rectangular, **3** =Polar |
| Auto or Aprox. | **5** | **1** =Auto, **2** =Aproximado |
| Formato vectorial | **6** | **1** =Rectangular, **2** =Cilíndrico, **3** =Esférico |
| Base | **7** | **1** =Decimal, **2** =Hex, **3** =Binário |

## shift()

**Catálogo >**

**shift(***NúmeroInteiro1* **[,** *#deDeslocações* **])** ⇒ *número inteiro*

Desloca os bits num número inteiro binário. Pode introduzir *NúmeroInteiro1* em qualquer base numérica; é convertido automaticamente para uma forma binária de 64 bits assinada. Se a magnitude de *NúmeroInteiro1* for muito grande para esta forma, uma operação do módulo simétrico coloca-o no intervalo. Para mais informações, consulte **▶Base2**, página 12.

Se *#deDeslocações* for positivo, a deslocação é para a esquerda. Se *#deDeslocações* for negativo, a deslocação é para a direita. A predefinição é −1 (deslocar um bit para a direita).

Numa deslocação para a direita, o bit mais à direita cai e 0 ou 1 é inserido para corresponder ao bit mais à esquerda. Numa deslocação para a esquerda, o bit mais à esquerda cai e 0 é inserido como o bit mais à direita.

Por exemplo, numa deslocação para a direita:

Cada bit desloca-se para a direita.

0b00000000000000011110101100011010

Insere 0 se o bit mais à esquerda for 0
ou 1 se o bit mais à esquerda for 1.

produz:

0b00000000000000011110101100011010

O resultado aparece de acordo com o modo base. Os zeros à esquerda não aparecem.

No modo base Bin:

| | |
|---|---|
| shift(0b1111010110000110101) | 0b1111010110000011010 |
| shift(256,1) | 0b1000000000 |

No modo base Hex:

| | |
|---|---|
| shift(0h78E) | 0h3C7 |
| shift(0h78E,−2) | 0h1E3 |
| shift(0h78E,2) | 0h1E38 |

**Importante:** Para introduzir um número binário ou hexadecimal, utilize sempre o prefixo 0b ou 0h (zero, não a letra O).

**shift(***Lista1* **[,** *#deDeslocações* **])** ⇒ *lista*

Devolve uma cópia de *Lista1* deslocada para a direita ou para a esquerda pelos elementos *#deDeslocações*. Não altere *Lista1*.

Se *#deDeslocações* for positivo, a deslocação é para a esquerda. Se *#deDeslocações* for negativo, a deslocação é para a direita. A predefinição é −1 (deslocar um elemento para a direita).

Os elementos introduzidos no início ou no fim de *lista* pela deslocação são definidos para o símbolo "undef".

No modo base Dec:

| | |
|---|---|
| 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()

**Catálogo >**

**shift(***Cadeia1* **[, *#deDeslocações* ]) ⇒ *cadeia***

Devolve uma cópia de *Cadeia1* rodada para a direita ou para a esquerda pelos caracteres *#deDeslocações*. Não altere *Cadeia1*.

Se *#deDeslocações* for positivo, a deslocação é para a esquerda. Se *#deDeslocações* for negativo, a deslocação é para a direita. A predefinição é −1 (deslocar um carácter para a direita).

Os caracteres introduzidos no início ou no fim de *lista* pela deslocação são definidos para um espaço.

shift("abcd") " abc"
shift("abcd",-2) " ab"
shift("abcd",1) "bcd "

## sign()

**Catálogo >**

**sign(***Valor1***) ⇒ *valor***
**sign(***Lista1***) ⇒ *lista***
**sign(***Matriz1***) ⇒ *matriz***

Para *Valor1* real ou complexo, devolve *Valor1* / **abs(***Valor1***)** quando *Valor1* ≠ 0.

Devolve 1 se *Valor1* for positivo.
Devolve −1 se *Valor1* for negativo.
**sign(0)** devolve ±1 se o modo do formato complexo for Real; caso contrário, devolve-se a si próprio.

**sign(0)** representa o círculo no domínio complexo.

Para uma lista ou matriz, devolve os sinais de todos os elementos.

sign(-3.2) -1
sign({2,3,4,-5}) {1,1,1,-1}

Se o modo do formato complexo for Real:

sign([-3 0 3]) [-1 undef 1]

## simult()

**Catálogo >**

**simult(***MatrizCoef***,** *VectorConst* **[,** *Tol* **]) ⇒ *matriz***

Devolve um vector da coluna que contém as soluções para um sistema de equações lineares.

Nota: Consulte também **linSolve()**, página 55.

*MatrizCoef* tem de ser uma matriz quadrada que contenha os coeficientes das equações.

*VectorConst* tem de ter o mesmo número de linhas (a mesma dimensão) que *MatrizCoef* e conter as constantes.

Opcionalmente, qualquer elemento da matriz é tratado como zero se o valor absoluto for inferior a *Tol*. Esta tolerância só é utilizada se a matriz tiver entradas de ponto flutuante e não contiver variáveis simbólicas sem um valor atribuído. Caso contrário, *Tol* é ignorado.

- Se definir o modo **Auto ou Aproximado** para Aproximado, os cálculos são efectuados com a aritmética de ponto flutuante.
- Se *Tol* for omitido ou não utilizado, a tolerância predefinida for calculada como:
  5E−14 • max(dim(*MatrizCoef*)) • rowNorm(*MatrizCoef*)

Resolver para x e y:
x + 2y = 1
3x + 4y = −1

simult([[1 2],[3 4]],[1,-1]) [-3 2]

A solução é x= −3 e y=2.

Resolver:
ax + by = 1
cx + dy = 2

[[1 2],[3 4]]→matx1 [[1 2],[3 4]]
simult(matx1,[1,2]) [0 1/2 2]

**simult(***MatrizCoef***,** *MatrizConst* **[,** *Tol* **]) ⇒ *matriz***

Resolve vários sistema de equações lineares, em que cada sistema tem os mesmo coeficientes de equações, mas constantes diferentes.

Cada coluna em *MatrizConst* tem de conter as constantes para um sistema de equações. Cada coluna da matriz resultante contém a solução para o sistema correspondente.

Resolver:
x + 2y = 1
3x + 4y = −1

x + 2y = 2
3x + 4y = −3

simult([[1 2],[3 4]],[[1 2],[-1 -3]]) [[-3 -7],[2 9/2]]

Para o primeiro sistema, x= −3 e y=2. Para o segundo sistema, x= −7 e y=9/2.

## sin()

**Tecla** trig

**sin(***Valor1***)** ⇒ *valor*
**sin(***Lista1***)** ⇒ *lista*

**sin(***Valor1***)** devolve o seno do argumento.

**sin(***Lista1***)** devolve uma lista de senos de todos os elementos em *Lista1*.

**Nota:** O argumento é interpretado como um ângulo expresso em graus, gradianos ou radianos, de acordo com o modo de ângulo actual. Pode utilizar °, G ou r para substituir a definição do modo de ângulo temporariamente.

No modo de ângulo Graus:

$\sin\left(\left(\frac{\pi}{4}\right)^r\right)$ 0.707107

$\sin(45)$ 0.707107

$\sin(\{0,60,90\})$ $\{0.,0.866025,1.\}$

No modo de ângulo Gradianos:

$\sin(50)$ 0.707107

No modo de ângulo Radianos:

$\sin\left(\frac{\pi}{4}\right)$ 0.707107

$\sin(45°)$ 0.707107

**sin(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve o seno da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o seno de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos:

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

$$\begin{bmatrix}0.9424 & -0.04542 & -0.031999\\ -0.045492 & 0.949254 & -0.020274\\ -0.048739 & -0.00523 & 0.961051\end{bmatrix}$$

## sin⁻¹()

**Tecla** trig

**sin⁻¹(***Valor1***)** ⇒ *valor*
**sin⁻¹(***Lista1***)** ⇒ *lista*

**sin⁻¹(***Valor1***)** devolve o ângulo cujo seno é *Valor1*.

**sin⁻¹(***Lista1***)** devolve uma lista de senos inversos de cada elemento de *Lista1*.

**Nota:** O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

**Nota:** Pode introduzir esta função através da escrita de **arcsin(…)** no teclado do computador.

**sin⁻¹(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve o seno inverso da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o seno inverso de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Graus:

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

No modo de ângulo Gradianos:

$\sin^{-1}(1)$ 100

No modo de ângulo Radianos:

$\sin^{-1}(\{0,0.2,0.5\})$ $\{0,0.201358,0.523599\}$

Nos modos de ângulo Radianos e Formato complexo rectangular:

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

$$\begin{bmatrix}-0.164058-0.064606\cdot i & 1.49086-2.1051\cdot i\\ 0.725533-1.51594\cdot i & 0.947305-0.7783\cdot i\\ 2.08316-2.63205\cdot i & -1.79018+1.2718\cdot i\end{bmatrix}$$

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

## sinh()

**Catálogo >**

**sinh(***Numver1***)** ⇒ *valor*
**sinh(***Lista1***)** ⇒ *lista*

**sinh(***Valor1***)** devolve o seno hiperbólico do argumento.

**sinh(***Lista1***)** devolve uma lista dos senos hiperbólicos de cada elemento de *Lista1*.

**sinh(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve o seno hiperbólico da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o seno hiperbólico de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

$\sinh(1.2)$ → 1.50946

$\sinh(\{0,1.2,3.\})$ → $\{0,1.50946,10.0179\}$

No modo de ângulo Radianos:

$$\sinh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

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

**Catálogo >**

**$\sinh^{-1}$(***Valor1***)** ⇒ *valor*
**$\sinh^{-1}$(***Lista1***)** ⇒ *lista*

**$\sinh^{-1}$(***Valor1***)** devolve o seno hiperbólico inverso do argumento.

**$\sinh^{-1}$(***Lista1***)** devolve uma lista de senos hiperbólicos inversos de cada elemento de *Lista1*.

**Nota:** Pode introduzir esta função através da escrita de **arcsinh(** … **)** no teclado.

**$\sinh^{-1}$(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve o seno hiperbólico inverso da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular o seno hiperbólico inverso de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

$\sinh^{-1}(0)$ → 0

$\sinh^{-1}(\{0,2.1,3\})$ → $\{0,1.48748,1.81845\}$

No modo de ângulo Radianos:

$$\sinh^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}0.041751 & 2.15557 & 1.1582\\ 1.46382 & 0.926568 & 0.112557\\ 2.75079 & -1.5283 & 0.57268\end{bmatrix}$$

## SinReg

Catálogo >

**SinReg** *X*, *Y* [, [*Repetições*],[ *Ponto*] [, *Categoria*, *Incluir*] ]

Calcula a regressão sinusoidal nas listas *X* e *Y*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis independentes e dependentes.

*Iterações* é um valor opcional que especifica o número máximo de vezes (de 1 a 16) que uma solução será tentada. Se for omitido, 8 é utilizado. Em geral, valores maiores resultam numa melhor precisão, mas maiores tempos de execução, e vice-versa.

*Período* especifica um período previsto. Se for omitido, a diferença entre os valores em *X* deve ser igual e por ordem sequencial. Se especificar *Período*, as diferenças entre os valores x podem ser desiguais.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são considerados no cálculo.

A saída de **SinReg** é sempre em radianos, independentemente da definição do modo de ângulo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.RegEqn | Equação de regressão: a • sin(bx+c)+d |
| stat.a, stat.b, stat.c, stat.d | Parâmetros de regressão |
| stat.Resid | Resíduos da regressão |
| stat.XReg | Lista de dados na *Lista X* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.YReg | Lista de dados na *Lista Y* modificada utilizada na regressão com base nas restrições de *Freq*, *Lista de categorias* e *Incluir categorias* |
| stat.FreqReg | Lista de frequências correspondentes a *stat.XReg* e *stat.YReg* |

## SortA

Catálogo >

**SortA** *Lista1* [, *Lista2* ] [, *Lista3* ] ...
**SortA** *Vector1* [, *Vector2* ] [, *Vector3* ] ...

Ordena os elementos do primeiro argumento por ordem crescente.

Se incluir argumentos adicionais, ordena os elementos para que as novas posições correspondam às novas posições dos elementos no primeiro argumento.

Todos os argumentos têm de ter nomes de listas ou vectores. Todos os argumentos têm de ter dimensões iguais.

Os elementos (nulos) vazios do primeiro argumento movem-se para a parte inferior. Para mais informações sobre os elementos vazios, consulte a página 136.

| | |
|---|---|
| $\{2,1,4,3\} \rightarrow list1$ | $\{2,1,4,3\}$ |
| 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

Catálogo >

**SortD** *Lista1* [**,** *Lista2* ] [**,** *Lista3* ] ...
**SortD** *Vector1* [**,** *Vector* ] [**,** *Vector3* ] ...

Idêntico a **SortA**, excepto que **SortD** ordena os elementos por ordem decrescente.

Os elementos (nulos) vazios do primeiro argumento movem-se para a parte inferior. Para mais informações sobre os elementos vazios, consulte a página 136.

$\{2,1,4,3\} \rightarrow list1 \quad \{2,1,4,3\}$

$\{1,2,3,4\} \rightarrow list2 \quad \{1,2,3,4\}$

SortD *list1*,*list2* *Done*

*list1* $\{4,3,2,1\}$

*list2* $\{3,4,1,2\}$

## ▸Sphere

Catálogo >

*Vector* **▸Sphere**

**Nota:** Pode introduzir esta função através da escrita de **@>Sphere** no teclado.

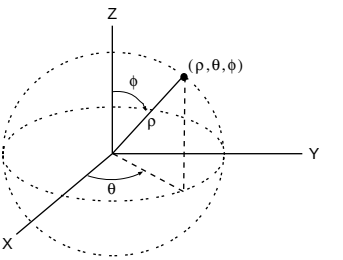
Apresenta o vector da linha ou coluna em forma esférica [ρ ∠θ ∠ϕ].

*O vector* tem de ser de dimensão 3 e pode ser um vector da linha ou coluna.

**Nota: ▸Sphere** é uma instrução de formato de visualização, não uma função de conversão. Só pode utilizá-la no fim da linha de entrada.

$[1 \quad 2 \quad 3] \blacktriangleright \text{Sphere}$
$[3.74166 \quad \angle 1.10715 \quad \angle 0.640522]$

$\left\langle \left[ 2 \quad \angle \frac{\pi}{4} \quad 3 \right] \right\rangle \blacktriangleright \text{Sphere}$
$[3.60555 \quad \angle 0.785398 \quad \angle 0.588003]$




## sqrt ()

Catálogo >

**sqrt(***Valor1***)** ⇒ *valor*
**sqrt(***Lista1***)** ⇒ *lista*

Devolve a raiz quadrada do argumento.

Para uma lista, devolve as raízes quadradas de todos os elementos em *Lista1***.**

**Nota:** Consulte também **Modelo de raiz quadrada**, página 1.

$\sqrt{4} \quad 2$

$\sqrt{\{9,2,4\}} \quad \{3,1.41421,2\}$

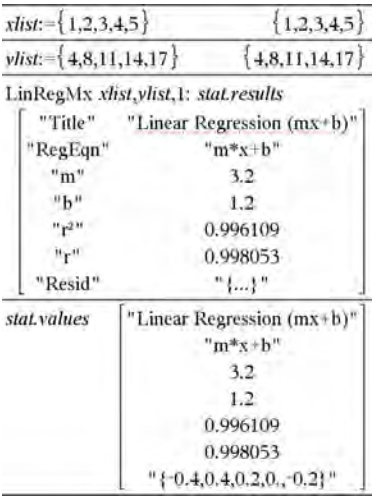
## stat.results

Apresenta os resultados de um cálculo estatístico.

Os resultados aparecem como um conjunto de pares de valores de nomes. Os nomes específicos apresentados estão dependentes do comando ou da função estatística avaliada mais recentemente.

Pode copiar um nome ou um valor e colá-lo noutra localização.

**Nota:** Evite definir variáveis que utilizem os mesmos nomes das variáveis utilizadas para análise estatística. Em alguns casos, pode ocorrer uma condição de erro. Os nomes das variáveis utilizados para análise estatística são listados na tabela abaixo.




stat.a
stat.AdjR²
stat.b
stat.b0
stat.b1
stat.b2
stat.b3
stat.b4
stat.b5
stat.b6
stat.b7
stat.b8
stat.b9
stat.b10
stat.bList
stat.$\chi^2$
stat.c
stat.CLower
stat.CLowerList
stat.CompList
stat.CompMatrix
stat.CookDist
stat.CUpper
stat.CUpperList
stat.d
stat.dfDenom
stat.dfBlock
stat.dfCol
stat.dfError
stat.dfInteract
stat.dfReg
stat.dfNumer
stat.dfRow
stat.DW
stat.e
stat.ExpMatrix
stat.F
stat.FBlock
stat.Fcol
stat.FInteract
stat.FreqReg
stat.Frow
stat.Leverage
stat.LowerPred
stat.LowerVal
stat.m
stat.MaxX
stat.MaxY
stat.ME
stat.MedianX
stat.MedianY
stat.MEPred
stat.MinX
stat.MinY
stat.MS
stat.MSBlock
stat.MSCol
stat.MSError
stat.MSInteract
stat.MSReg
stat.MSRow
stat.n
stat.$\hat{p}$
stat.$\hat{p}$1
stat.$\hat{p}$2
stat.$\hat{p}$Diff
stat.PList
stat.PVal
stat.PValBlock
stat.PValCol
stat.PValInteract
stat.PValRow
stat.Q1X
stat.Q1Y
stat.Q3X
stat.Q3Y
stat.r
stat.$r^2$
stat.RegEqn
stat.Resid
stat.ResidTrans
stat.$\sigma$x
stat.$\sigma$y
stat.$\sigma$x1
stat.$\sigma$x2
stat.$\Sigma$x
stat.$\Sigma x^2$
stat.$\Sigma$xy
stat.$\Sigma$y
stat.$\Sigma y^2$
stat.s
stat.SE
stat.SEList
stat.SEPred
stat.sResid
stat.SEslope
stat.sp
stat.SS
stat.SSBlock
stat.SSCol
stat.SSX
stat.SSY
stat.SSError
stat.SSInteract
stat.SSReg
stat.SSRow
stat.tList
stat.UpperPred
stat.UpperVal
stat.$\bar{x}$
stat.$\bar{x}$1
stat.$\bar{x}$2
stat.$\bar{x}$Diff
stat.$\bar{x}$List
stat.XReg
stat.XVal
stat.XValList
stat.$\bar{y}$
stat.$\hat{y}$
stat.$\hat{y}$ List
stat.YReg

**Nota:** Sempre que a aplicação Listas e Folha de Cálculo calcula parâmetros estatísticos, copia as variáveis do grupo "stat." para um grupo "stat#.", em que # é um número que é incrementado automaticamente. Isto permite manter os resultados anteriores durante a execução de vários cálculos.

**stat.values**

Apresenta uma matriz dos valores calculados para o comando ou a função estatística avaliada mais recentemente.

Ao contrário de **stat.results**, **stat.valu** omite os nomes associados aos valores.

Pode copiar um valor e colá-lo noutras localizações.

Consulte o exemplo de **stat.results**.



**stDevPop(***Lista* **[,** *ListFreq* **])** ⇒

Devolve o desvio padrão da população dos elementos em *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

**Nota:** *Lista* tem de ter pelo menos dois elementos. Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

Nos modos auto e de ângulo Radianos:

$\text{stDevPop}(\{1,2,5,-6,3,-2\})$ 3.59398

$\text{stDevPop}(\{1.3,2.5,-6.4\},\{3,2,5\})$ 4.11107

**stDevPop(***Matriz1* **[,** *MatrizFreq* **])** ⇒ *matriz*

Devolve um vector da linha dos desvios padrão da população das colunas em *Matriz1*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

**Nota:** *Matriz1* tem de ter pelo menos duas linhas. Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

$$\text{stDevPop}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right) \quad \begin{bmatrix} 3.26599 & 2.94392 & 1.63299 \end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix},\begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) \quad \begin{bmatrix} 2.52608 & 5.21506 \end{bmatrix}$$



**stDevSamp(***Lista* **[,** *ListaFreq* **])** ⇒ *expressão*

Devolve o desvio padrão da amostra dos elementos em *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

**Nota:** *Lista* tem de ter pelo menos dois elementos. Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

$\text{stDevSamp}(\{1,2,5,-6,3,-2\})$ 3.937

$\text{stDevSamp}(\{1.3,2.5,-6.4\},\{3,2,5\})$ 4.33345

**stDevSamp(***Matriz1* **[,** *MatrizFreq* **])** ⇒ *matriz*

Devolve um vector da coluna dos desvios padrão da amostra das colunas em *Matriz1*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

**Nota:** *Matriz1* tem de ter pelo menos duas linhas. Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

$$\text{stDevPop}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right) \quad \begin{bmatrix} 3.26599 & 2.94392 & 1.63299 \end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix},\begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) \quad \begin{bmatrix} 2.52608 & 5.21506 \end{bmatrix}$$

### Stop (Parar)

Catálogo >

**Stop**

Programar comando: Termina o programa.

**Stop** não é permitido em funções.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
i:=0                                   0
Define prog1()=Prgm                    Done
               For i,1,10,1
               If i=5
               Stop
               EndFor
               EndPrgm
prog1()                                Done
i                                      5
```

### Store (Guardar)

Consulte → **(guardar)**, página 134.

### string()

Catálogo >

**strin g(***Expr***)** ⇒ *cadeia*

Simplifica *Expr* e devolve o resultado como uma cadeia de caracteres.

```
string(1.2345)                         "1.2345"
string(1+2)                            "3"
```

### subMat()

Catálogo >

**subMa t(***Matriz1* [**,** *LinhaInicial* ] [**,** *ColInicial* ] [**,** *LinhaFinal* ] [**,** *ColFinal* ]**)**
⇒ *matrix*

Devolve a submatriz especificada de *Matriz1*.

Predefinições: *LinhaInicial* =1, *ColInicial* =1, *LinhaFinal* =última linha, *ColFinal* =última coluna.

$$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}\rightarrow m1 \qquad \begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$$

$$\text{subMat}(m1,2,1,3,2) \qquad \begin{bmatrix}4&5\\7&8\end{bmatrix}$$

$$\text{subMat}(m1,2,2) \qquad \begin{bmatrix}5&6\\8&9\end{bmatrix}$$

### Sigma (Soma)

Consulte Σ**()**, página 129.

### sum()

Catálogo >

**sum(***Lista* [**,** *Início* [**,** *Fim* ]]**)** ⇒ *expressão*

Devolve a soma dos elementos em *Lista*.

*Início* e *Fim* são opcionais. Especificam um intervalo de elementos.

Qualquer argumento vazio produz um resultado vazio. Os elementos (nulos) vazios da *Lista* são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

```
sum({1,2,3,4,5})                       15
sum({a,2·a,3·a})
                   "Error: Variable is not defined"
sum(seq(n,n,1,10))                     55
sum({1,3,5,7,9},3)                     21
```

### sum()

Catálogo >

**sum(***Matrix1* **[,** *Início* **[,** *Fim* **]])** ⇒ *matriz*

Devolve um vector da linha com as somas dos elementos nas colunas em *Matriz1*.

*Início* e *Fim* são opcionais. Especificam um intervalo de linhas.

Qualquer argumento vazio produz um resultado vazio. Os elementos (nulos) vazios da *Matrix1* são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

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

$\text{sum}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$ → $\begin{bmatrix}12&15&18\end{bmatrix}$

$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2,3\right)$ → $\begin{bmatrix}11&13&15\end{bmatrix}$

### sumIf()

Catálogo >

**sumIf(***Lista***,** *Critérios* **[,** *ListaDeSomas* **])** ⇒ *valor*

Devolve a soma acumulada de todos os elementos em *Lista* que satisfazem os *Critérios* especificados. Opcionalmente, pode especificar uma lista alternativa, *ListaDeSomas*, para fornecer os elementos a acumular.

*Lista* pode ser uma expressão, lista ou matriz. *ListaDeSomas*, se especificada, tem de ter as mesmas dimensões que *Lista*.

*Critérios* podem ser:

- Um valor, uma expressão ou uma cadeia. Por exemplo, **34** acumula apenas os elementos em *Lista* que são simplificados para o valor 34.
- Uma expressão booleana com o símbolo **?** como um identificador para cada elemento. Por exemplo, **?<10** acumula apenas os elementos em *Lista* que são inferiores a 10.

Quando um elementos da *Lista* cumprir os *Critérios*, o elemento é adicionado à soma acumulada. Se incluir *ListaDeSomas*, o elemento correspondente de *ListaDeSomas* é adicionado à soma.

Na aplicação Listas e Folha de cálculo, pode utilizar um intervalo de células no lugar de *Lista* e de *ListaDeSomas*.

Os elementos (nulos) vazios são ignorados. Para mais informações sobre os elementos vazios, consulte a página 136.

**Nota:** Consulte também **countIf()**, página 23.

$\text{sumIf}(\{1,2,e,3,\pi,4,5,6\},2.5<?<4.5)$ → 12.859874482

$\text{sumIf}(\{1,2,3,4\},2<?<5,\{10,20,30,40\})$ → 70

### sumSeq()

Consulte Σ(), página 129.

### system()

Catálogo >

**system(***Valor1* **[,** *Valor2* **[,** *Valor3* **[,** *...***]]])**

Devolve um sistema de equações formatado como uma lista. Pode também criar um sistema com um modelo.

## T

### T (transpor)

Catálogo >

*Matriz1*$^{\mathbf{T}}$ ⇒ *matriz*

Apresenta a transposta dos conjugados dos complexo da *Matriz1*.

**Nota:** Pode introduzir este operador através da escrita de **@t** no teclado do computador.

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

## tan()

**Tecla** trig

**tan(***Valor1***)** ⇒ *valor*
**tan(***Lista1***)** ⇒ *lista*

**tan(***Valor1***)** devolve a tangente do argumento.

**tan(***Lista1***)** devolve uma lista das tangentes de todos os elementos em *Lista1*.

**Nota:** O argumento é interpretado como um ângulo expresso em graus, gradianos ou radianos, de acordo com o modo de ângulo actual. Pode utilizar °, G ou r para substituir a definição do modo de ângulo temporariamente.

No modo de ângulo Graus:

$\tan\left(\left(\frac{\pi}{4}\right)^{r}\right)$ → 1.

$\tan(45)$ → 1.

$\tan(\{0,60,90\})$ → $\{0.,1.73205,\text{undef}\}$

No modo de ângulo Gradianos:

$\tan\left(\left(\frac{\pi}{4}\right)^{r}\right)$ → 1.

$\tan(50)$ → 1.

$\tan(\{0,50,100\})$ → $\{0.,1.,\text{undef}\}$

No modo de ângulo Radianos:

$\tan\left(\frac{\pi}{4}\right)$ → 1.

$\tan(45°)$ → 1.

$\tan\left(\left\{\pi,\frac{\pi}{3},-\pi,\frac{\pi}{4}\right\}\right)$ → $\{0.,1.73205,0.,1.\}$

**tan(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve a tangente da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular a tangente de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos:

$$\tan\left(\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix}\right) \rightarrow \begin{bmatrix}-28.2912 & 26.0887 & 11.1142\\ 12.1171 & -7.83536 & -5.48138\\ 36.8181 & -32.8063 & -10.4594\end{bmatrix}$$

## $\tan^{-1}()$

**Tecla** trig

**$\tan^{-1}$(***Valor1***)** ⇒ *valor*
**$\tan^{-1}$(***Lista1***)** ⇒ *lista*

**$\tan^{-1}$(***Valor1***)** devolve o ângulo cuja tangente é *Valor1*.

**$\tan^{-1}$(***Lista1***)** devolve uma lista das tangentes inversas de cada elemento de *Lista1*.

**Nota:** O resultado é devolvido como um ângulo expresso em graus, gradianos ou radianos, de acordo com a definição do modo de ângulo actual.

**Nota:** Pode introduzir esta função através da escrita de **arctan(**…**)** no teclado.

No modo de ângulo Graus:

$\tan^{-1}(1)$ → 45

No modo de ângulo Gradianos:

$\tan^{-1}(1)$ → 50

No modo de ângulo Radianos:

$\tan^{-1}(\{0,0.2,0.5\})$ → $\{0,0.197396,0.463648\}$

## tan⁻¹()

**Tecla** trig

**tan⁻¹(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve a tangente inversa da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular a tangente inversa de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos:

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

$$\begin{bmatrix}-0.083658 & 1.26629 & 0.62263\\ 0.748539 & 0.630015 & -0.070012\\ 1.68608 & -1.18244 & 0.455126\end{bmatrix}$$

## tanh()

**Catálogo >**

**tanh(***Valor1***)** ⇒ *valor*
**tanh(***Lista1***)** ⇒ *lista*

**tanh(***Valor1***)** devolve a tangente hiperbólica do argumento.

**tanh(***Lista1***)** devolve uma lista das tangentes hiperbólicas de cada elemento de *Lista1*.

$$\tanh(1.2) \qquad 0.833655$$

$$\tanh(\{0,1\}) \qquad \{0.,0.761594\}$$

**tanh(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve a tangente hiperbólica da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular a tangente hiperbólica de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos:

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

$$\begin{bmatrix}-0.097966 & 0.933436 & 0.425972\\ 0.488147 & 0.538881 & -0.129382\\ 1.28295 & -1.03425 & 0.428817\end{bmatrix}$$

## tanh⁻¹()

**Catálogo >**

**tanh⁻¹(***Valor1***)** ⇒ *valor*
**tanh⁻¹(***Lista1***)** ⇒ *lista*

**tanh⁻¹(***Valor1***)** devolve a tangente hiperbólica inversa do argumento como uma expressão.

**tanh⁻¹(***Lista1***)** devolve uma lista das tangentes hiperbólicas inversas de cada elemento de *Lista1*.

**Nota:** Pode introduzir esta função através da escrita de **arctanh(** … **)** no teclado.

No Formato complexo rectangular:

$$\tanh^{-1}(0) \qquad 0.$$

$$\tanh^{-1}(\{1,2.1,3\})$$

$$\{\text{undef},0.518046-1.5708\cdot i,0.346574-1.570$$

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

**tanh⁻¹(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve a tangente hiperbólica inversa da matriz de *MatrizQuadrada1*. Isto não é o mesmo que calcular a tangente hiperbólica inversa de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

No modo de ângulo Radianos e Formato complexo rectangular:

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

$$\begin{bmatrix}-0.099353+0.164058\cdot i & 0.267834-1.4908\\ -0.087596-0.725533\cdot i & 0.479679-0.9473\\ 0.511463-2.08316\cdot i & -0.878563+1.7901\end{bmatrix}$$

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

## tCdf()

**Catálogo >**

**tCdf(***LimiteInferior***,** *LimiteSuperior***,** *dfs***)** ⇒ *número* se *LimiteInferior* e *LimiteSuperior* forem números, *lista* se *LimiteInferior* e *LimiteSuperior* forem listas

Calcula a probabilidade da distribuição Student-*t* entre *LimiteInferior* e *LimiteSuperior* para os graus de liberdade especificados *df*.

Para P(X ≤ *LimiteSuperior*), defina *LimiteInferior* = **−9E999**.

## Text

**Catálogo >**

**Text** *CadeiaDePedido* **[,** *MostrarMarcador***]**

Programar comando: Interrompe o programa e mostra a cadeia de caracteres *CadeiaDoPedido* numa caixa de diálogo.

Quando o utilizador seleccionar **OK**, a execução do programa continua.

O argumento *marcador* opcional pode ser qualquer expressão.

- Se omitir *MostrarMarcador* e avaliar para **1**, a mensagem de texto é adicionada ao histórico da Calculadora.
- Se *MostrarMarcador* avaliar para **0**, a mensagem de texto não é adicionada ao histórico.

Se o programa necessitar de uma resposta escrita do utilizador, consulte **Request**, página 86, ou **RequestStr**, página 87.

**Nota:** Pode utilizar este comando num programa definido pelo utilizador, mas não numa função.

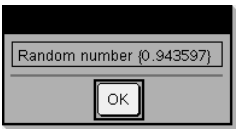
Defina um programa que interrompa a visualização após cinco números aleatórios numa caixa de diálogo.
No modelo Prgm...EndPrgm, complete cada linha, premindo ↵ em vez de **enter**. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
Define text_demo()=Prgm
 For i,1,5
  strinfo:="Random number" & string(rand(i))
  Text strinfo
 EndFor
EndPrgm
```

Executar o programa:

```
text_demo()
```

Amostra de uma caixa de diálogo:




## Then

Consulte **If,** página 45.

## tInterval

**Catálogo >**

**tInterval** *Lista***[,***Freq***[,***NívelC***]]**

(Entrada da lista de dados)

**tInterval** $\bar{x}$**,***sx***,***n***[,***NívelC***]**

(Entrada estatística do resumo)

Calcula um intervalo de confiança *t*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.CLower, stat.CUpper | Intervalo de confiança para uma média de população desconhecida |
| stat.$\bar{x}$ | Média da amostra da sequência de dados da distribuição aleatória normal |
| stat.ME | Margem de erro |
| stat.df | Graus de liberdade |

| Variável de saída | Descrição |
|---|---|
| stat.σx | Desvio padrão da amostra |
| stat.n | Comprimento da sequência de dados com a média da amostra |

### tInterval_2Samp

Catálogo >

**tInterval_2Samp** *Lista1*, *Lista2* [, *Freq1* [, *Freq2* [, *NívelC* [, *Combinado* ]]]]

(Entrada da lista de dados)

**tInterval_2Samp** $\bar{x}1$, *sx1*, *n1*, $\bar{x}2$, *sx2*, *n2* [, *NívelC* [, *Combinado* ]]

(Entrada estatística do resumo)

Calcula um intervalo de confiança *t* de duas amostras. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

*Combinado* = **1** combina variações; *Combinado* = **0** não combina variações.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.CLower, stat.CUpper | Intervalo de confiança com probabilidade da distribuição do nível de confiança |
| stat.$\bar{x}1$ - $\bar{x}2$ | Médias das amostras das sequências de dados da distribuição aleatória normal |
| stat.ME | Margem de erro |
| stat.df | Graus de liberdade |
| stat.$\bar{x}1$, stat.$\bar{x}2$ | Médias das amostras das sequências de dados da distribuição aleatória normal |
| stat.σx1, stat.σx2 | Desvios padrão das amostras para *Lista 1* e *Lista 2* |
| stat.n1, stat.n2 | Número de amostras em sequências de dados |
| stat.sp | Desvio padrão combinado. Calculado quando *Combinado* = SIM. |

### tPdf()

Catálogo >

**tPdf(***ValX*, *df***)** ⇒ *número* se *ValX* for um número, *lista* se *ValX* for uma lista

Calcula a função de densidade da probabilidade (pdf) para a distribuição Student-*t* num valor *x* especificado com os graus de liberdade especificados *df*.

## trace()

Catálogo >

**trace(***MatrizQuadrada***)** ⇒ *valor*

Apresenta o traço (soma de todos os elementos na diagonal principal) de *MatrizQuadrada*.

$$\text{trace}\left(\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}\right) \quad 15$$

$$a:=12 \quad 12$$

$$\text{trace}\left(\begin{bmatrix}a & 0\\ 1 & a\end{bmatrix}\right) \quad 24$$

## Try

Catálogo >

**Try**
*bloco1*
**Else**
*bloco2*
**EndTry**

Executa o *bloco1* excepto se ocorrer um erro. A execução do programa transfere-se para *bloco2* se ocorrer um erro em *bloco1*. A variável do sistema *errCode* contém o código de erro para permitir que o programa efectue a recuperação do erro. Para obter uma lista de códigos de erros, consulte *"Mensagens e códigos de erros"*, página 142.

*bloco1* e *bloco2* podem ser uma única palavra ou uma série de palavras separadas pelo carácter ":".

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições em diferentes linhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
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
```

### Exemplo 2

Para ver os comandos **Try**, **ClrErr** e **PassErr** na operação, introduza o programa de valores próprios() apresentado à direita. Execute o programa através da execução de cada uma das seguintes expressões.

$$eigenvals\left(\begin{bmatrix}-3\\ -41\\ 5\end{bmatrix},\begin{bmatrix}-1 & 2 & -3.1\end{bmatrix}\right)$$

**Nota:** Consulte também **ClrErr**, página 17, e **PassErr**, página 75.

```
Definir valores próprios(a,b)=Prgm
© Os valores próprios do programa(A,B) mostra os valores
próprios de A·B
Ensaio
  Disp "A= ",a
  Disp "B= ",b
  Disp " "
  Disp "Valores próprios de A·B são:",eigVl(a*b)
Else
  If errCode=230 Then
    Disp "Error: Produto de A·B tem de ser uma matriz
quadrada"
    ClrErr
  Else
    PassErr
  EndIf
EndTry
EndPrgm
```

## tTest

Catálogo >

**tTest** $\mu 0$, *Lista* [, *Freq* [, *Hipótese* ]]

(Entrada da lista de dados)

**tTest** $\mu 0$, $\bar{x}$, *sx*, *n*, [ *Hipótese*]

(Entrada estatística do resumo)

Efectua um teste da hipótese para uma média da população desconhecida $\mu$ quando o desvio padrão da população $\sigma$ for desconhecido. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Teste $H_0$: $\mu = \mu 0$, em relação a uma das seguintes:

Para $H_a$: $\mu < \mu 0$, defina *Hipótese*<0
Para $H_a$: $\mu \neq \mu 0$ (predefinição), defina *Hipótese*=0
Para $H_a$: $\mu > \mu 0$, defina *Hipótese*>0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.t | $(\bar{x} - \mu 0) / (\text{stdev} / \text{sqrt}(n))$ |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.df | Graus de liberdade |
| stat.$\bar{x}$ | Média da amostra da sequência de dados em *Lista* |
| stat.sx | Desvio padrão da amostra da sequência de dados |
| stat.n | Tamanho da amostra |

## tTest_2Samp

Catálogo >

**tTest_2Samp** *Lista1*, *Lista2* [, *Freq1* [, *Freq2* [, *Hipótese* [, *Combinado* ]]]]

(Entrada da lista de dados)

**tTest_2Samp** $\bar{x}1$, *sx1*, *n1*, $\bar{x}2$, *sx2*, *n2* [, *Hipótese* [, *Combinado* ]]

(Entrada estatística do resumo)

Calcula um teste *t* de duas amostras. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Teste $H_0$: $\mu 1 = \mu 2$, em relação a uma das seguintes:

Para $H_a$: $\mu 1 < \mu 2$, defina *Hipótese*<0
Para $H_a$: $\mu 1 \neq \mu 2$ (predefinição), defina *Hipótese*=0
Para $H_a$: $\mu 1 > \mu 2$, defina *Hipótese*>0

*Combinado*=**1** combina as variâncias
*Combinado*=**0** não combina as variâncias

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.t | Valor normal padrão calculado para a diferença de médias |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.df | Graus de liberdade para a t-statistic |
| stat.$\bar{x}$1, stat.$\bar{x}$2 | Médias da amostra das sequências de dados em *Lista 1* e *Lista 2* |
| stat.sx1, stat.sx2 | Desvios padrão da amostra das sequências de dados em *Lista 1* e *Lista 2* |
| stat.n1, stat.n2 | Tamanho das amostras |
| stat.sp | Desvio padrão combinado. Calculado quando *Combinado* =1. |

### tvmFV()

**Catálogo >**

**tvmFV(***N*, *I*, *PV*, *Pmt*, [ *PpY* ], [ *CpY* ], [ *PmtAt* ]**)** ⇒ *valor*

Função financeira que calcula o valor futuro do dinheiro.

**Nota:** Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 111. Consulte também **amortTbl()**, página 6.

```
tvmFV(120,5,0,-500,12,12)                77641.1
```

### tvmI()

**Catálogo >**

**tvmI(***N*, *PV*, *Pmt*, *FV*, [ *PpY* ], [ *CpY* ], [ *PmtAt* ]**)** ⇒ *valor*

Função financeira que calcula a taxa de juro por ano.

**Nota:** Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 111. Consulte também **amortTbl()**, página 6.

```
tvmI(240,100000,-1000,0,12,12)           10.5241
```

### tvmN()

**Catálogo >**

**tvmN(***I*, *PV*, *Pmt*, *FV*, [ *PpY* ], [ *CpY* ], [ *PmtAt* ]**)** ⇒ *valor*

Função financeira que calcula o número de períodos de pagamento.

**Nota:** Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 111. Consulte também **amortTbl()**, página 6.

```
tvmN(5,0,-500,77641,12,12)                120.
```

### tvmPmt()

**Catálogo >**

**tvmPmt(***N*, *I*, *PV*, *FV*, [ *PpY* ], [ *CpY* ], [ *PmtAt* ]**)** ⇒ *valor*

Função financeira que calcula o montante de cada pagamento.

**Nota:** Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 111. Consulte também **amortTbl()**, página 6.

```
tvmPmt(60,4,30000,0,12,12)               -552.496
```

### tvmPV()

**Catálogo >**

**tvmPV(***N*, *I*, *Pmt*, *FV*, [ *PpY* ], [ *CpY* ], [ *PmtAt* ]**)** ⇒ *valor*

Função financeira que calcula o valor actual.

**Nota:** Os argumentos utilizados nas funções TVM são descritos na tabela de argumentos TVM, página 111. Consulte também **amortTbl()**, página 6.

```
tvmPV(48,4,-500,30000,12,12)             -3426.7
```

| Argumento TVM* | Descrição | Tipo de dados |
|---|---|---|
| *N* | Número de períodos de pagamento | número real |
| *I* | Taxa de juro anual | número real |
| *PV* | Valor actual | número real |
| *Pmt* | Montante do pagamento | número real |
| *FV* | Valor actual | número real |
| *PpY* | Pagamentos por ano, predefinição=1 | número inteiro > 0 |
| *CpY* | Períodos compostos por ano, predefinição=1 | número inteiro > 0 |
| *PmtAt* | Pagamento devido no fim ou no início de cada período, predefiniçãot=fim | número inteiro (0=fim, 1=início) |

***** Estes nomes dos argumentos do valor temporal do dinheiro são similares aos nomes das variáveis TVM (como **tvm.pv** e **tvm.pmt**) que são utilizados pelo resolutor financeiro da aplicação Calculadora. No entanto, as funções financeiras não guardam os resultados ou os valores dos argumentos nas variáveis TVM.

## TwoVar

**Catálogo >**

**TwoVar** *X*, *Y*[, [*Freq*] [, *Categoria*, *Incluir*]]

Calcula a estatística TwoVar. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Todas as listas têm de ter a mesma dimensão, excepto para *Incluir*.

*X* e *Y* são listas de variáveis dependentes e independentes.

*Freq* é uma lista opcional de valores de frequência. Cada elemento em *Freq* especifica a frequência de ocorrência para cada ponto de dados *X* e *Y* correspondente. O valor predefinido é 1. Todos os elementos têm de ser números inteiros 0.

*Categoria* é uma lista de códigos de categorias numéricos ou de cadeias para os dados *X* e *Y* correspondentes.

*Incluir* é uma lista de um ou mais códigos de categorias. Apenas os itens de dados cujo código de categoria está incluído nesta lista são incluídos no cálculo.

Um elemento (nulo) vazio em qualquer das listas *X*, *Freq* ou *Category* resulta num nulo para o elemento correspondente de todas essas listas. Um elemento vazio em qualquer uma das listas de *X1* a *X20* resulta num vazio para o elemento correspondente de todas essas listas. Para mais informações sobre os elementos vazios, consulte a página 136.

| Variável de saída | Descrição |
|---|---|
| stat.$\bar{x}$ | Média dos valores x |
| stat.$\Sigma$x | Soma dos valores x |
| stat.$\Sigma$x2 | Soma de valores x2 |
| stat.sx | Desvio padrão da amostra de x |
| stat.$\sigma$x | Desvio padrão da população de x |
| stat.n | Número de pontos de dados |

| Variável de saída | Descrição |
|---|---|
| stat.$\bar{y}$ | Média de valores y |
| stat.$\Sigma y$ | Soma de valores y |
| stat.$\Sigma y^2$ | Soma de valores y2 |
| stat.sy | Desvio padrão da amostra de y |
| stat.$\sigma y$ | Desvio padrão da população de y |
| stat.$\Sigma xy$ | Soma de valores x · y |
| stat.r | Coeficiente de correlação |
| stat.MinX | Mínimo dos valores x |
| stat.$Q_1X$ | 1º quartil de x |
| stat.MedianX | Mediana de x |
| stat.$Q_3X$ | 3º quartil de x |
| stat.MaxX | Máximo de valores x |
| stat.MinY | Mínimo dos valores y |
| stat.$Q_1Y$ | 1º quartil de y |
| stat.MedY | Mediana de y |
| stat.$Q_3Y$ | 3º quartil de y |
| stat.MaxY | Máximo de valores y |
| stat.$\Sigma(x - \bar{x})^2$ | Soma de quadrados de desvios da média de x |
| stat.$\Sigma(y - \bar{y})^2$ | Soma de quadrados de desvios da média de y |

# *U*

**unitV()** **Catálogo >**

**unitV(***Vector1***)** ⇒ *vector*

Devolve um vector unitário da linha ou da coluna na forma de *Vector1*.

*Vector1* tem de ser uma matriz de coluna ou linha individual.

$$\text{unitV}\begin{bmatrix}1 & 2 & 1\end{bmatrix} \quad \begin{bmatrix}0.408248 & 0.816497 & 0.408248\end{bmatrix}$$

$$\text{unitV}\begin{bmatrix}1\\2\\3\end{bmatrix} \quad \begin{bmatrix}0.267261\\0.534522\\0.801784\end{bmatrix}$$

### unLock

Catálogo >

**unLock** *Var1*[**,** *Var2*] [**,** *Var3*] ...
**unLock** *Var*.

Desbloqueia as variáveis ou o grupo de variáveis especificadas. Não pode eliminar ou modificar as variáveis bloqueadas.

Consulte **Lock**, página 57, e **getLockInfo()**, página 42.

| | |
|---|---|
| $a:=65$ | 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()

Catálogo >

**varPop(***Lista* [**,** *ListFreq* ]**)** ⇒ *expressão*

Devolve a variação da população de *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

**Nota:** *Lista* tem de conter pelo menos dois elementos.

Se um elemento numa das listas estiver vazio (nulo), esse elemento é ignorado e o elemento correspondente na outra lista também é ignorado. Para mais informações sobre os elementos vazios, consulte a página 136.

| | |
|---|---|
| varPop$(\{5,10,15,20,25,30\})$ | 72.9167 |

### varSamp()

Catálogo >

**varSamp(***Lista* [**,** *ListaFreq* ]**)** ⇒ *expressão*

Devolve a variação da amostra de *Lista*.

Cada elemento de *ListFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Lista*.

**Nota:** *Lista* tem de conter pelo menos dois elementos.

Se um elemento numa das listas estiver vazio (nulo), esse elemento é ignorado e o elemento correspondente na outra lista também é ignorado. Para mais informações sobre os elementos vazios, consulte a página 136.

| | |
|---|---|
| varSamp$(\{1,2,5,-6,3,-2\})$ | $\frac{31}{2}$ |
| varSamp$(\{1,3,5\},\{4,6,2\})$ | $\frac{68}{33}$ |

**varSamp(***Matriz1* [**,** *MatrizFreq* ]**)** ⇒ *matriz*

Devolve um vector da coluna com a variação da amostra de cada coluna em *Matriz1*.

Cada elemento de *ListaFreq* conta o número de ocorrências consecutivas do elemento correspondente em *Matriz1*.

**Nota:** *Matriz1* tem de conter pelo menos duas linhas.

Se um elemento numa das matrizes estiver vazio (nulo), esse elemento é ignorado e o elemento correspondente na outra matriz também é ignorado. Para mais informações sobre os elementos vazios, consulte a página 136.

| | |
|---|---|
| varSamp$\begin{pmatrix}1 & 2 & 5\\ -3 & 0 & 1\\ .5 & .7 & 3\end{pmatrix}$ | $\begin{bmatrix}4.75 & 1.03 & 4\end{bmatrix}$ |
| varSamp$\left(\begin{bmatrix}-1.1 & 2.2\\ 3.4 & 5.1\\ -2.3 & 4.3\end{bmatrix},\begin{bmatrix}6 & 3\\ 2 & 4\\ 5 & 1\end{bmatrix}\right)$ | $\begin{bmatrix}3.91731 & 2.08411\end{bmatrix}$ |

## warnCodes()

Catálogo >

**warnCodes(***Expr1***,** *StatusVar***)** ⇒ *expressão*

Avalia a expressão *Expr1*, apresenta o resultado e guarda os códigos de quaisquer avisos gerados na variável da lista *StatusVar*. Se não forem gerados avisos, esta função atribui a *StatusVar* uma lista vazia.

*Expr1* pode ser qualquer expressão matemática TI-Nspire™ ou TI-Nspire™ CAS válida. Não pode utilizar um comando ou atribuição como *Expr1*.

*StatusVar* tem de ser um nome de variável válido.

Para uma lista dos códigos de aviso e mensagens associadas, consulte a página 147.

$\text{warnCodes}\left(\text{solve}\left(\sin(10\cdot x)=\frac{x^2}{x},x\right),warn\right)$

$x=-0.84232 \text{ or } x=-0.706817 \text{ or } x=-0.285234 \text{ or } x=$

| | |
|---|---|
| *warn* | {10007, 10009} |

Para ver o resultado completo, prima ▲ e, de seguida, utilize ◀ e ▶ para mover o cursor.

## when()

Catálogo >

**when(***Condição***,** *ResultadoVerdadeiro* **[,** *ResultadoFalso* **][,** *ResultadoDesconhecido* **])**
⇒ *expressão*

Devolve *ResultadoVerdadeiro*, *ResultadoFalso* ou *ResultadoDesconhecido*, dependendo se a *Condição* é verdadeira, falsa ou desconhecida. Devolve a entrada se existirem poucos argumentos para especificar o resultado adequado.

Omite *ResultadoFalso* e *ResultadoDesconhecido* para definir uma expressão apenas na região em que a *Condição* é verdadeira.

Utilize um **undef** *ResultadoFalso* para definir uma expressão representada graficamente apenas num intervalo.

**when()** é útil para definir funções recursivas.

| | |
|---|---|
| $\text{when}(x<0,x+3)\|x=5$ | undef |

$\text{when}(n>0,n\cdot factoral(n-1),1)\rightarrow factoral(n)$

*Done*

| | |
|---|---|
| $factoral(3)$ | 6 |
| $3!$ | 6 |

## While

Catálogo >

**While** *Condição*
*Bloco*
**EndWhile**

Executa as declarações em *Bloco* desde que *Condição* seja verdadeira.

*Bloco* pode ser uma declaração ou uma sequência de declarações separadas pelo carácter ":".

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
Define sum_of_recip(n)=Func
                         Local i,tempsum
                         1→i
                         0→tempsum
                         While i≤n
                         tempsum+1/i→tempsum
                         i+1→i
                         EndWhile
                         Return tempsum
                         EndFunc
                                                Done
```

| | |
|---|---|
| $sum\_of\_recip(3)$ | $\frac{11}{6}$ |

# X

### xor (xou)

**Catálogo >**

*ExprBooleana1* **xor** *ExprBooleana2* devolve *expressão booleana*
*ListaBooleana1* **xor** *ListaBooleana2* devolve *lista booleana*
*MatrizBooleana1* **xor** *MatrizBooleana2* devolve *matriz booleana*

Devolve verdadeiro se *ExprBooleana1* for verdadeira e *ExprBooleana2* for falsa ou vice-versa.

Devolve falso se ambos os argumentos forem verdadeiros ou falsos. Devolve uma expressão booleana simplificada se não for possível resolver um dos argumentos para verdadeiro ou falso.

**Nota:** Consulte **or**, página 74.

| | |
|---|---|
| true xor true | false |
| 5>3 xor 3>5 | true |

*NúmeroInteiro1* **xor** *NúmeroInteiro2* ⇒ *número inteiro*

Compara dois números inteiros reais bit a bit com uma operação **xor**. Internamente, ambos os números inteiros são convertidos para números binários de 64 bits assinados. Quando os bits correspondentes forem comparados, o resultado é 1 se um dos bits (mas não ambos) for 1; caso contrário, o resultado é 0. O valor devolvido representa os resultados dos bits e aparece de acordo com o modo base.

Pode introduzir os números inteiros em qualquer base numérica. Para uma entrada binária ou hexadecimal, tem de utilizar o prefixo 0b ou 0h, respectivamente. Sem um prefixo, os números inteiros são tratados como decimais (base 10).

Se introduzir um número inteiro na base 10 muito grande para uma forma binária de 64 bits assinada, é utilizada uma operação de módulo simétrico para colocar o valor no intervalo adequado. Para mais informações, consulte ▶**Base2**, página 12.

**Nota:** Consulte **or**, página 74.

No modo base Hex:
**Importante:** Zero, não a letra O.

| | |
|---|---|
| 0h7AC36 xor 0h3D5F | 0h79169 |

No modo base Bin:

| | |
|---|---|
| 0b100101 xor 0b100 | 0b100001 |

**Nota:** Uma entrada binária pode ter até 64 dígitos (não contando com o prefixo 0b). Uma entrada hexadecimal pode ter até 16 dígitos.

# Z

### zInterval

**Catálogo >**

**zInterval** σ**,** *Lista* [**,** *Freq* [**,** *NívelC* ]]

(Entrada da lista de dados)

**zInterval** σ**,** $\bar{x}$**,** *n* [**,** *NívelC*]

(Entrada estatística do resumo)

Calcula um intervalo de confiança *z*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.CLower, stat.CUpper | Intervalo de confiança para uma média de população desconhecida |
| stat.$\bar{x}$ | Média da amostra da sequência de dados da distribuição aleatória normal |
| stat.ME | Margem de erro |
| stat.sx | Desvio padrão da amostra |

| Variável de saída | Descrição |
|---|---|
| stat.n | Comprimento da sequência de dados com a média da amostra |
| stat.$\sigma$ | Desvio padrão da população conhecido para a sequência de dados *Lista* |

### zInterval_1Prop

Catálogo > 

**zInterval_1Prop** *x*, *n* [, *NívelC*]

Calcula um intervalo de confiança *z* de uma proporção. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

*x* é um número inteiro não negativo.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.CLower, stat.CUpper | Intervalo de confiança com probabilidade da distribuição do nível de confiança |
| stat.$\hat{p}$ | Proporção calculada de sucessos |
| stat.ME | Margem de erro |
| stat.n | Número de amostras na sequência de dados |

### zInterval_2Prop

Catálogo > 

**zInterval_2Prop** *x1*, *n1*, *x2*, *n2* [, *NívelC*]

Calcula um intervalo de confiança *z* de duas proporções. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

*x1* e *x2* são números inteiros não negativos.

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.CLower, stat.CUpper | Intervalo de confiança com probabilidade da distribuição do nível de confiança |
| stat.$\hat{p}$Diff | Diferença calculada entre proporções |
| stat.ME | Margem de erro |
| stat.$\hat{p}$1 | Primeira previsão da proporção da amostra |
| stat.$\hat{p}$2 | Segunda previsão da proporção da amostra |
| stat.n1 | Tamanho da amostra na sequência de dados um |
| stat.n2 | Tamanho da amostra na sequência de dados dois |

### zInterval_2Samp

Catálogo >

**zInterval_2Samp** $\sigma_1$**,** $\sigma_2$**,** *Lista1***,** *Lista2* **[,** *Freq1* **[,** *Freq2***, [** *NívelC* **]]]**

(Entrada da lista de dados)

**zInterval_2Samp** $\sigma_1$**,** $\sigma_2$**,** $\bar{x}$*1***,** *n1***,** $\bar{x}$*2***,** *n2* **[,** *NívelC*]

(Entrada estatística do resumo)

Calcula um intervalo de confiança *z* de duas amostras. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.CLower, stat.CUpper | Intervalo de confiança com probabilidade da distribuição do nível de confiança |
| stat.$\bar{x}$1 - $\bar{x}$2 | Médias das amostras das sequências de dados da distribuição aleatória normal |
| stat.ME | Margem de erro |
| stat.$\bar{x}$1, stat.$\bar{x}$2 | Médias das amostras das sequências de dados da distribuição aleatória normal |
| stat.$\sigma$x1, stat.$\sigma$x2 | Desvios padrão da amostra para *Lista 1* e *Lista 2* |
| stat.n1, stat.n2 | Número de amostras em sequências de dados |
| stat.r1, stat.r2 | Desvios padrão da população conhecidos para sequência de dados *Lista 1* e *Lista 2* |

### zTest

Catálogo >

**zTest** $\mu$*0***,** $\sigma$**,** *Lista***, [** *Freq* **[,** *Hipótese* **]]**

(Entrada da lista de dados)

**zTest** $\mu$*0***,** $\sigma$**,** $\bar{x}$**,** *n* **[,** *Hipótese*]

(Entrada estatística do resumo)

Efectua um teste *z* com a frequência *listfreq*. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Teste $H_0$: $\mu = \mu 0$, em relação a uma das seguintes:

Para $H_a$: $\mu < \mu 0$, defina *Hipótese*<0
Para $H_a$: $\mu \neq \mu 0$ (predefinição), defina *Hipótese*=0
Para $H_a$: $\mu > \mu 0$, defina *Hipótese*>0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.z | ($\bar{x} - \mu 0$) / ($\sigma$ / sqrt(n)) |
| stat.P Value | Menor probabilidade de rejeição da hipótese nula |
| stat.$\bar{x}$ | Média da amostra da sequência de dados em *Lista* |

| Variável de saída | Descrição |
|---|---|
| stat.sx | Desvio padrão da amostra da sequência de dados. Apenas devolvido para a entrada *Dados*. |
| stat.n | Tamanho da amostra |

### zTest_1Prop

Catálogo > 

**zTest_1Prop** *p0*, *x*, *n* [, *Hipótese*]

Calcula um teste *z* de uma proporção. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

*x* é um número inteiro não negativo.

Teste $H_0$: *p = p0* em relação a uma das seguintes:

Para $H_a$: *p > p0*, defina *Hipótese*>0
Para $H_a$: *p ≠ p0 (predefinição)*, defina *Hipótese*=0
Para $H_a$: *p < p0*, defina *Hipótese*<0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.p0 | Proporção da população suposta |
| stat.z | Valor normal padrão calculado para a proporção |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.$\hat{p}$ | Proporção da amostra prevista |
| stat.n | Tamanho da amostra |

### zTest_2Prop

Catálogo > 

**zTest_2Prop** *x1*, *n1*, *x2*, *n2* [, *Hipótese*]

Calcula um teste *z* de duas proporções. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

*x1* e *x2* são números inteiros não negativos.

Teste $H_0$: *p1 = p2* em relação a uma das seguintes:

Para $H_a$: *p1 > p2*, defina *Hipótese*>0
Para $H_a$: *p1 ≠ p2 (predefinição)*, defina *Hipótese*=0
Para $H_a$: *p < p0*, defina *Hipótese*<0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.z | Valor normal padrão calculado para a diferença de proporções |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.$\hat{p}$1 | Primeira previsão da proporção da amostra |
| stat.$\hat{p}$2 | Segunda previsão da proporção da amostra |

| Variável de saída | Descrição |
|---|---|
| stat.$\hat{p}$ | Previsão da proporção da amostra combinada |
| stat.n1, stat.n2 | Números de amostras retiradas das tentativas 1 e 2 |

### zTest_2Samp

**Catálogo >**

**zTest_2Samp** $\sigma_1$**,** $\sigma_2$**,** *Lista1***,** *Lista2* **[,** *Freq1* **[,** *Freq2* **[,** *Hipótese* **]]]**

(Entrada da lista de dados)

**zTest_2Samp** $\sigma_1$**,** $\sigma_2$**,** $\bar{x}1$**,** *n1***,** $\bar{x}2$**,** *n2* **[,** *Hipótese*]

(Entrada estatística do resumo)

Calcula um teste *z* de duas amostras. Um resumo dos resultados é guardado na variável *stat.results*. (Consulte a página 100.)

Teste $H_0$: $\mu1 = \mu2$, em relação a uma das seguintes:

Para $H_a$: $\mu1 < \mu2$, defina *Hipótese*<0
Para $H_a$: $\mu1 \neq \mu2$ (predefinição), defina *Hipótese*=0
Para $H_a$: $\mu1 > \mu2$, *Hipótese*>0

Para mais informações sobre o efeito dos elementos vazios numa lista, consulte "Elementos (nulos) vazios" na página 136.

| Variável de saída | Descrição |
|---|---|
| stat.z | Valor normal padrão calculado para a diferença de médias |
| stat.PVal | Menor nível de significância para o qual a hipótese nula pode ser rejeitada |
| stat.$\bar{x}1$, stat.$\bar{x}2$ | Médias das amostras das sequências de dados em *Lista1* e *Lista2* |
| stat.sx1, stat.sx2 | Desvios padrão da amostra das sequências de dados em *Lista1* e *Lista2* |
| stat.n1, stat.n2 | Tamanho das amostras |

# Símbolos

**+ (adicionar)** **Tecla +**

*Valor1* **+** *Valor2* ⇒ *valor*

Devolve a soma dos dois argumentos.

$56 \quad 56$

$56+4 \quad 60$

$60+4 \quad 64$

$64+4 \quad 68$

$68+4 \quad 72$

*Lista1* + *Lista2* ⇒ *lista*
*Matriz1* + *Matriz2* ⇒ *matriz*

Devolve uma lista (ou matriz) com as somas dos elementos correspondentes em *Lista1* e *Lista2* (ou *Matriz1* e *Matriz2*).

As dimensões dos argumentos têm de ser iguais.

$\left\{22,\pi,\frac{\pi}{2}\right\} \rightarrow l1 \quad \{22,3.14159,1.5708\}$

$\left\{10,5,\frac{\pi}{2}\right\} \rightarrow l2 \quad \{10,5,1.5708\}$

$l1+l2 \quad \{32,8.14159,3.14159\}$

*Valor* + *Lista1* ⇒ *lista*
*Lista1* + *Valor* ⇒ *lista*

Devolve uma lista com as somas de *Valor* e de cada elemento em *Lista1*.

$15+\{10,15,20\} \quad \{25,30,35\}$

$\{10,15,20\}+15 \quad \{25,30,35\}$

*Valor* + *Matriz1* ⇒ *matriz*
*Matriz1* + *Valor* ⇒ *matriz*

Devolve uma matriz com *Valor* adicionado a cada elemento na diagonal de *Matriz1*. *Matriz1* tem de ser quadrada.

$20+\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix} \quad \begin{bmatrix}21 & 2\\ 3 & 24\end{bmatrix}$

**Nota:** Utilize **.+** (ponto mais) para adicionar uma expressão a cada elemento.

**− (subtrair)** **Tecla −**

*Valor1* − *Valor2* ⇒ *valor*

Devolve *Valor1* menos *Valor2*.

$6-2 \quad 4$

$\pi-\frac{\pi}{6} \quad 2.61799$

*Lista1* − *Lista2* ⇒ *lista*
*Matriz1* − *Matriz2* ⇒ *matriz*

Subtrai cada elmento em *Lista2* (ou *Matriz2*) do elemento correspondente em *Lista1* (ou *Matriz1*) e devolve os resultados.

As dimensões dos argumentos têm de ser iguais.

$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\} \quad \{12,-1.85841,0.\}$

$[3 \quad 4]-[1 \quad 2] \quad [2 \quad 2]$

*Valor* − *Lista1* ⇒ *lista*
*Lista1* − *Valor* ⇒ *lista*

Subtrai cada elemento de *Lista1* de *Valor* ou subtrai *Valor* de cada elemento de *Lista1* e devolve uma lista dos resultados.

$15-\{10,15,20\} \quad \{5,0,-5\}$

$\{10,15,20\}-15 \quad \{-5,0,5\}$

## – (subtrair)

**Tecla** −

*Valor* – *Matriz1* ⇒ *matriz*

*Matriz1* – *Valor* ⇒ *matriz*

*Valor* – *Matriz1* devolve uma matriz de *Valor* vezes a matriz de identidade menos *Matriz1*. *Matriz1* tem de ser quadrada.

*Matriz1* – *Valor* devolve uma matriz de *Valor* vezes a matriz de identidade subtraída de *Matriz1*. *Matriz1* tem de ser quadrada.

**Nota:** Utilize **.–** (ponto menos) para subtrair uma expressão de cada elemento.

$20-\begin{bmatrix}1&2\\3&4\end{bmatrix}$ $\quad\begin{bmatrix}19&-2\\-3&16\end{bmatrix}$

## · (multiplicar)

**Tecla** ×

*Valor1* · *Valor2* ⇒ *valor*

Devolve o produto dos dois argumentos.

$2\cdot 3.45$ $\quad 6.9$

*Lista1* · *Lista2* ⇒ *lista*

Devolve uma lista com os produtos dos elementos correspondentes em *Lista1* e *Lista2*.

As dimensões das listas têm de ser iguais.

$\{1.,2,3\}\cdot\{4,5,6\}$ $\quad\{4,10,18\}$

*Matriz1* · *Matriz2* ⇒ *matriz*

Devolve o produto da matriz de *Matriz1* e *Matriz2*.

O número de colunas em *Matriz1* tem de ser igual ao número de linhas em *Matriz2*.

$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}\cdot\begin{bmatrix}7&8\\7&8\\7&8\end{bmatrix}$ $\quad\begin{bmatrix}42&48\\105&120\end{bmatrix}$

*Valor* · *Lista1* ⇒ *lista*

*Lista1* · *Valor* ⇒ *lista*

Devolve uma lista com os produtos de *Valor* e de cada elemento em *Lista1*.

$\pi\cdot\{4,5,6\}$ $\quad\{12.5664,15.708,18.8496\}$

*Valor* · *Matriz1* ⇒ *matriz*

*Matriz1* · *Valor* ⇒ *matriz*

Devolve uma matriz com os produtos de *Valor* e de cada elemento em *Matriz1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix}\cdot 0.01$ $\quad\begin{bmatrix}0.01&0.02\\0.03&0.04\end{bmatrix}$

$6\cdot\text{identity}(3)$ $\quad\begin{bmatrix}6&0&0\\0&6&0\\0&0&6\end{bmatrix}$

**Nota:** Utilize **.·** (ponto multiplicar) para multiplicar uma expressão por cada elemento.

## / (dividir)

**Tecla** ÷

*Valor1* / *Valor2* ⇒ *valor*

Devolve o quociente de *Valor1* dividido pelo *Valor2*.

**Nota:** Consulte também **Modelo da fracção**, página 1.

$\frac{2}{3.45}$ $\quad .57971$

*Lista1* / *Lista2* ⇒ *lista*

Devolve uma lista com os quocientes de *Lista1* divididos pela *Lista2*.

As dimensões das listas têm de ser iguais.

$\frac{\{1.,2,3\}}{\{4,5,6\}}$ $\quad\left\{0.25,\frac{2}{5},\frac{1}{2}\right\}$

*Valor* / *Lista1* ⇒ *lista*

*Lista1* / *Valor* ⇒ *lista*

Devolve uma lista com os quocientes de *Valor* divididos pela *Lista1* ou de *Lista1* divididos pelo *Valor*.

$\frac{6}{\{3,6,\sqrt{6}\}}$ $\quad\{2,1,2.44949\}$

$\frac{\{7,9,2\}}{7\cdot 9\cdot 2}$ $\quad\left\{\frac{1}{18},\frac{1}{14},\frac{1}{63}\right\}$

### / (dividir)

**Tecla** ÷

*Valor* / *Matriz1* ⇒ *matriz*

*Matriz1* / *Valor* ⇒ *matriz*

Devolve uma matriz com os quocientes de *Matriz1* / *Valor*.

**Nota:** Utilize **.** / (ponto dividir) para dividir uma expressão por cada elemento.

$$\frac{[7\ \ 9\ \ 2]}{7\cdot 9\cdot 2} \qquad \left[\frac{1}{18}\ \ \frac{1}{14}\ \ \frac{1}{63}\right]$$

### ^ (potência)

**Tecla** ^

*Valor1* ^ *Valor2* ⇒ *valor*

*Lista1* ^ *Lista2* ⇒ *lista*

Devolve o primeiro argumento elevado à potência do segundo argumento.

**Nota:** Consulte também **Modelo do expoente**, página 1.

Para uma lista, devolve os elementos em *Lista1* elevados à potência dos elementos correspondentes em *Lista2*.

No domínio real, as potências fraccionárias que tenham expoentes simplificados com denominadores ímpares utilizam a derivação real versus a derivação principal para o modo complexo.

$$4^2 \qquad 16$$

$$\{2,4,6\}^{\{1,2,3\}} \qquad \{2,16,216\}$$

*Valor* ^ *Lista1* ⇒ *lista*

Devolve *Valor* elevado à potência dos elementos em *Lista1*.

*Lista1* ^ *Valor* ⇒ *lista*

Devolve os elementos em *Lista1* elevados à potência de *Valor*.

$$\pi^{\{1,2,-3\}} \qquad \{3.14159,9.8696,0.032252\}$$

$$\{1,2,3,4\}^{-2} \qquad \left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16}\right\}$$

*MatrizQuadrada1* ^ *número inteiro* ⇒ *matriz*

Devolve *MatrizQuadrada1* elevada à potência do *número inteiro*.

*MatrizQuadrada1* tem de ser uma matriz quadrada.

Se *número inteiro* = −1, calcula a matriz inversa.
Se *número inteiro* < −1, calcula a matriz inversa para uma potência positiva adequada.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^2 \qquad \begin{bmatrix}7 & 10\\ 15 & 22\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-2} \qquad \begin{bmatrix}\frac{11}{2} & \frac{-5}{2}\\ \frac{-15}{4} & \frac{7}{4}\end{bmatrix}$$

### x² (quadrado)

**Tecla** x²

*Valor1* ² ⇒ *valor*

Devolve o quadrado do argumento.

*Lista1* ² ⇒ *lista*

Devolve uma lista com os quadrados dos elementos em *Lista1*.

*MatrizQuadrada1* ² ⇒ *matriz*

Devolve a matriz quadrada de *MatrizQuadrada1*. Isto não é o mesmo que calcular o quadrado de cada elemento. Utilize .^2 para calcular o quadrado de cada elemento.

### .+ (ponto adicionar)

**Teclas** . +

*Matriz1* **.+** *Matriz2* ⇒ *matriz*
*Valor* **.+** *Matriz1* ⇒ *matriz*

*Matriz1* **.+** *Matriz2* devolve uma matriz que é a soma de cada par dos elementos correspondentes em *Matriz1* e *Matriz2*.

*Valor* **.+** *Matriz1* devolve uma matriz que é a soma de *Valor* e de cada elemento em *Matriz1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix} .+ \begin{bmatrix}10&30\\20&40\end{bmatrix} \quad \begin{bmatrix}11&32\\23&44\end{bmatrix}$

$5 .+ \begin{bmatrix}10&30\\20&40\end{bmatrix} \quad \begin{bmatrix}15&35\\25&45\end{bmatrix}$

### .– (ponto subtracção)

**Teclas** . –

*Matriz1* **.–** *Matriz2* ⇒ *matriz*
*Valor* **.–***Matriz1* ⇒ *matriz*

*Matriz1* **.–***Matriz2* devolve uma matriz que é a diferença entre cada par de elementos correspondentes em *Matriz1* e *Matriz2*.

*Valor* **.–***Matriz1* devolve uma matriz que é a diferença de *Valor* e de cada elemento em *Matriz1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix} .- \begin{bmatrix}10&20\\30&40\end{bmatrix} \quad \begin{bmatrix}-9&-18\\-27&-36\end{bmatrix}$

$5 .- \begin{bmatrix}10&20\\30&40\end{bmatrix} \quad \begin{bmatrix}-5&-15\\-25&-35\end{bmatrix}$

### .· (ponto mult.)

**Teclas** . ×

*Matriz1* **.·** *Matriz2* ⇒ *matriz*
*Valor* **.·** *Matriz1* ⇒ *matriz*

*Matriz1* **.·** *Matriz2* devolve uma matriz que é o produto de cada par dos elementos correspondentes em *Matriz1* e *Matriz2*.

*Valor* **.·** *Matriz1* devolve uma matriz com os produtos de *Valor* e de cada elemento em *Matriz1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix} .\cdot \begin{bmatrix}10&20\\30&40\end{bmatrix} \quad \begin{bmatrix}10&40\\90&160\end{bmatrix}$

$5 .\cdot \begin{bmatrix}10&20\\30&40\end{bmatrix} \quad \begin{bmatrix}50&100\\150&200\end{bmatrix}$

### ./ (ponto dividir)

**Teclas** . ÷

*Matriz1* **./** *Matriz2* ⇒ *matriz*
*Valor* **./** *Matriz1* ⇒ *matriz*

*Matriz1* **./** *Matriz2* devolve uma matriz que é o quociente de cada par de elementos correspondente em *Matriz1* e *Matriz2*.

*Valor* **./** *Matriz1* devolve uma matriz que é o quociente de *Valor* e de cada elemento em *Matriz1*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix} ./ \left(\begin{bmatrix}10&20\\30&40\end{bmatrix}\right) \quad \begin{bmatrix}\frac{1}{10}&\frac{1}{10}\\\frac{1}{10}&\frac{1}{10}\end{bmatrix}$

$5 ./ \left(\begin{bmatrix}10&20\\30&40\end{bmatrix}\right) \quad \begin{bmatrix}\frac{1}{2}&\frac{1}{4}\\\frac{1}{6}&\frac{1}{8}\end{bmatrix}$

### .^ (ponto potência)

**Teclas** . ^

*Matriz1* **.^** *Matriz2* ⇒ *matriz*
*Valor* **. ^** *Matriz1* ⇒ *matriz*

*Matriz1* **.^** *Matriz2* devolve uma matriz em que cada elemento em *Matriz2* é o expoente para o elemento correspondente em *Matriz1*.

*Valor* **.^** *Matriz1* devolve uma matriz em que cada elemento em *Matriz1* é o expoente para *Valor*.

$\begin{bmatrix}1&2\\3&4\end{bmatrix} .\wedge \begin{bmatrix}0&2\\3&-1\end{bmatrix} \quad \begin{bmatrix}1&4\\27&\frac{1}{4}\end{bmatrix}$

$5 .\wedge \begin{bmatrix}0&2\\3&-1\end{bmatrix} \quad \begin{bmatrix}1&25\\125&\frac{1}{5}\end{bmatrix}$

### - (negação)

**Tecla** (−)

-*Valor1* ⇒ *valor*
-*Lista1* ⇒ *lista*
-*Matriz1* ⇒ *matriz*

Devolve a negação do argumento.

Para uma lista ou matriz, devolve todos os elementos negados.

Se o argumento for um número inteiro binário ou hexadecimal, a negação dá o complemento de dois.

| | |
|---|---|
| $-2.43$ | $-2.43$ |
| $-\{-1,0.4,1.2\text{E}19\}$ | $\{1.,-0.4,-1.2\text{E}19\}$ |

No modo base Bin:

**Importante:** Zero, não a letra O

| | |
|---|---|
| 0b100101▶Dec | 37 |
| $-$0b100101 | |
| 0b1111111111111111111111111111111▶ | |
| *Ans*▶Dec | −37 |

Para ver o resultado completo, prima ▲ e utilize ◀ e ▶ para mover o cursor.

### % (percentagem)

**Teclas** ctrl 🕮

*Valor1* **%** ⇒ *valor*
*Lista1* **%** ⇒ *lista*
*Matriz1* **%** ⇒ *matriz*

Devolve $\frac{argument}{100}$

Para uma lista ou matriz, devolve uma lista ou matriz com cada elemento dividido por 100.

Prima **Ctrl**+**Enter** ctrl enter (Macintosh®: ⌘+**Enter**) para avaliar:

| | |
|---|---|
| 13% | 0.13 |

Prima **Ctrl**+**Enter** ctrl enter (Macintosh®: ⌘+**Enter**) para avaliar:

| | |
|---|---|
| $\{\{1,10,100\}\}\%$ | $\{0.01,0.1,1.\}$ |

### = (igual)

**Tecla** [=]

*Expr1* = *Expr2* ⇒ *Expressão booleana*
*Lista1* = *Lista2* ⇒ *Lista booleana*
*Matriz1* = *Matriz2* ⇒ *Matriz booleana*

Devolve verdadeiro se *Expr1* for determinada para ser igual a *Expr2*.

Devolve falso se *Expr1* for determinada para ser diferente a *Expr2*.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

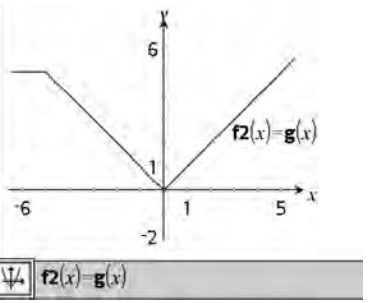
**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo [↵] em vez de [enter] no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

Exemplo de função que utiliza os símbolos de teste matemático: =, ≠, <, ≤, >, ≥

Define $g(x)$=Func
If $x\leq -5$ Then
Return 5
ElseIf $x>-5$ and $x<0$ Then
Return $-x$
ElseIf $x\geq 0$ and $x\neq 10$ Then
Return $x$
ElseIf $x=10$ Then
Return 3
EndIf
EndFunc

*Done*

Resultado do gráfico g(x)




### ≠ (diferente)

**Teclas** [ctrl] [=]

*Expr1* ≠ *Expr2* ⇒ *Expressão booleana*
*Lista1* ≠ *Lista2* ⇒ *Lista booleana*
*Matriz1* ≠ *Matriz2* ⇒ *Matriz booleana*

Devolve verdadeiro se *Expr1* for determinada para ser diferente a *Expr2*.

Devolve falso se *Expr1* for determinada para ser igual a *Expr2*.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

**Nota:** Pode introduzir este operador através da escrita de **/=** no teclado.

Consulte exemplo "=" (igual).

### < (menor que)

**Teclas** ctrl =

*Expr1* **<** *Expr2* ⇒ *Expressão booleana*
*Lista1* **<** *Lista2* ⇒ *Lista booleana*
*Matriz1* **<** *Matriz2* ⇒ *Matriz booleana*

Devolve verdadeiro se *Expr1* for determinada para ser menor que *Expr2*.

Devolve falso se *Expr1* for determinada para ser igual ou maior que *Expr2*.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

Consulte exemplo "=" (igual).

### ≤ (igual ou menor que)

**Teclas** ctrl =

*Expr1* ≤ *Expr2* ⇒ *Expressão booleana*
*Lista1* ≤ *Lista2* ⇒ *Lista booleana*
*Matriz1* ≤ *Matriz2* ⇒ *Matriz booleana*

Devolve verdadeiro se *Expr1* for determinada para igual ou menor que *Expr2*.

Devolve falso se *Expr1* for determinada para ser maior que *Expr2*.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

**Nota:** Pode introduzir este operador através da escrita de **<=** no teclado

Consulte exemplo "=" (igual).

### > (maior que)

**Teclas** ctrl =

*Expr1* **>** *Expr2* ⇒ *Expressão booleana*
*Lista1* **>** *Lista2* ⇒ *Lista booleana*
*Matriz1* **>** *Matriz2* ⇒ *Matriz booleana*

Devolve verdadeiro se *Expr1* for determinada para ser maior que *Expr2*.

Devolve falso se *Expr1* for determinada para ser igual ou menor que *Expr2*.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

Consulte exemplo "=" (igual).

### ≥ (igual ou maior que)

**Teclas** ctrl =

*Expr1* ≥ *Expr2* ⇒ *Expressão booleana*
*Lista1* ≥ *Lista2* ⇒ *Lista booleana*
*Matriz1* ≥ *Matriz2* ⇒ *Matriz booleana*

Devolve verdadeiro se *Expr1* for determinada para ser igual ou maior que *Expr2*.

Devolve falso se *Expr1* for determinada para ser menor que *Expr2*.

Outra coisa qualquer devolve uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

**Nota:** Pode introduzir este operador através da escrita de **>=** no teclado.

Consulte exemplo "=" (igual).

### ⇒ (implicação lógica)

**Teclas** ctrl =

*ExprBooleana1* ⇒ *ExprBooleana2* devolve *expressão booleana*
*ListaBooleana1* ⇒ *ListaBooleana2* devolve *lista booleana*
*MatrizBooleana1* ⇒ *MatrizBooleana2* devolve *matriz booleana*
*NúmeroInteiro1* ⇒ *NúmeroInteiro2* devolve *número inteiro*

Avalia a expressão **not** <argumento1> **or** <argumento2> e devolve falso, verdadeiro ou uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

**Nota:** Pode introduzir este operador ao escrever **=>** com o teclado

| | |
|---|---|
| 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} |

### ⇔ (implicação lógica dupla, XNOR)

**Teclas** ctrl =

*ExprBooleana1* ⇔ *ExprBooleana2* devolve *expressão booleana*
*ListaBooleana1* ⇔ *ListaBooleana2* devolve *lista booleana*
*MatrizBooleana1* ⇔ *MatrizBooleana2* devolve *matriz booleana*
*NúmeroInteiro1* ⇔ *NúmeroInteiro2* devolve *número inteiro*

Devolve a negação de uma operação booleana **XOR** nos dois argumentos. Devolve falso, verdadeiro ou uma forma simplificada da equação.

Para listas e matrizes, devolve comparações elemento por elemento.

**Nota:** Pode introduzir este operador ao escrever **<=>** com o teclado

| | |
|---|---|
| 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} |

### ! (factorial)

**Tecla** ?!

*Valor1*! ⇒ *valor*
*Lista1*! ⇒ *lista*
*Matriz1*! ⇒ *matriz*

Devolve o factorial do argumento.

Para uma lista ou matriz, devolve uma lista ou matriz de factoriais dos elementos.

| | |
|---|---|
| 5! | 120 |
| {{5,4,3}}! | {120,24,6} |
| $\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right)!$ | $\begin{bmatrix}1 & 2\\ 6 & 24\end{bmatrix}$ |

### & (acrescentar)

**Teclas** ctrl

*Cadeia1* & *Cadeia2* ⇒ *cadeia*

Devolve uma cadeia de texto que é *Cadeia2* acrescentada a *Cadeia1*.

| | |
|---|---|
| "Hello "&"Nick" | "Hello Nick" |

### *d*() (derivada)

**Catálogo >**

***d***(*Expr1*, *Var*[, *Ordem*]) | *Var*=*Valor* ⇒ *valor*

***d***(*Expr1*, *Var*[, *Ordem*]) ⇒ *valor*

***d***(*Lista1*, *Var*[, *Ordem*]) ⇒ *lista*

***d***(*Matriz1*, *Var*[, *Ordem*]) ⇒ *matriz*

Excepto quando utilizar a primeira sintaxe, tem de guardar um valor numérico na variável *Var* antes de avaliar ***d*()**. Consulte os exemplos.

***Pode utilizar d*()** para calcular a derivada de primeira e segunda ordem num ponto numericamente com os métodos de diferenciação automáticos.

*Ordem*, se incluída, tem de ser=**1** ou **2**. A predefinição é **1**.

**Nota:** Pode introduzir isto através da escrita de **`derivada(...)`** **`no teclado`**.

$\frac{d}{dx}(|x|)|x=0$ undef

$x:=0:\frac{d}{dx}(|x|)$ undef

$x:=3:\frac{d}{dx}(\{x^2,x^3,x^4\})$ $\{6,27,108\}$

**Nota:** Consulte também **Primeira derivada**, página 5 ou **Segunda derivada**, página 5.

**Nota:** O algoritmo **d()** tem uma limitação: funciona recursivamente através da expressão não simplificada, computação do valor numérico da primeira derivada (e a segunda, se aplicável) e a avaliação de cada subexpressão, que pode conduzir a um resultado imprevisto.

Considere o exemplo da direita. A primeira derivada de x·(x^2+x)^(1/3) em x=0 é igual a 0. No entanto, como a primeira derivada da subexpressão (x^2+x)^(1/3) está indefinida em x=0, e este valor é utilizado para calcular a derivada da expressão total, **d()** reporta o resultado como indefinido e apresenta uma mensagem de aviso.

Se encontrar esta limitação, verifique a solução graficamente. Pode também tentar com **centralDiff()**.

$\frac{d}{dx}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}}\right)|x=0$ undef

$\text{centralDiff}\left(x\cdot\left(x^2+x\right)^{\frac{1}{3}},x\right)|x=0$ 0.000033

### ∫() (integral)

**Catálogo >**

∫(*Expr*, *Var*, *Inferior*, *Superior*) ⇒ *valor*

Devolve o integral de *Expr1* em relação à variável *Var* de *Inferior* a *Superior*. Pode ser utilizada para calcular o integral definido numericamente com o mesmo método de nInt().

**Nota:** Pode introduzir esta função através do teclado, escrevendo **`integral(...)`**.

**Nota:** Consulte também **nInt()**, página 69, e **modelo do integral definido**, página 5.

$\int_0^1 x^2\,dx$ 0.333333

### √() (raiz quadrada)

**Teclas** ctrl x²

√(*Valor1*) ⇒ *valor*

√(*Lista1*) ⇒ *lista*

Devolve a raiz quadrada do argumento.

Para uma lista, devolve as raízes quadradas de todos os elementos em *Lista1*.

**Nota:** Pode introduzir esta função através da escrita de **`sqrt(...)`** no teclado

**Nota:** Consulte também **Modelo de raiz quadrada**, página 1.

$\sqrt{4}$ 2

$\sqrt{\{9,2,4\}}$ $\{3,1.41421,2\}$

## Π() (prodSeq)

**Catálogo >**

Π(*Expr1*, *Var*, *Baixo*, *Alto*) ⇒ *expressão*

**Nota:** Pode introduzir esta função através da escrita de **prodSeq(…)** no teclado.

Avalia *Expr1* para cada valor de *Var* de *Baixo* a *Alto* e devolve o produto dos resultados.

**Nota:** Consulte também **Modelo do produto (Π)**, página 4.

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

$\prod_{n=1}^{5}\left(\left\{\frac{1}{n},n,2\right\}\right)$ → $\left\{\frac{1}{120},120,32\right\}$

Π(*Expr1*, *Var*, *Baixo*, *Baixo* − 1) ⇒ 1

Π(*Expr1*, *Var*, *Baixo*, *Alto*)
⇒ 1/Π(*Expr1*, *Var*, *Alto*+1, *Baixo* − 1) se *Alto* < *Baixo* − 1

$\prod_{k=4}^{3}(k)$ → 1

As fórmulas do produto utilizadas derivam da seguinte referência:

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$\prod_{k=4}^{1}\left(\frac{1}{k}\right)$ → 6

$\prod_{k=4}^{1}\left(\frac{1}{k}\right)\cdot\prod_{k=2}^{4}\left(\frac{1}{k}\right)$ → $\frac{1}{4}$

## Σ() (sumSeq)

**Catálogo >**

Σ(*Expr1*, *Var*, *Baixo*, *Alto*) ⇒ *expressão*

**Nota:** Pode introduzir esta função através da escrita de sum**Seq(…)** no teclado.

Avalia *Expr1* para cada valor de *Var* de *Baixo* a *Alto* e devolve a soma dos resultados.

**Nota:** Consulte também **Modelo da soma**, página 4.

$\sum_{n=1}^{5}\left(\frac{1}{n}\right)$ → $\frac{137}{60}$

Σ(*Expr1*, *Var*, *Baixo*, *Baixo* − 1) ⇒ 0

Σ(*Expr1*, *Var*, *Baixo*, *Alto*)
⇒ −Σ(*Expr1*, *Var*, *Alto*+1, *Baixo* − 1) se *Alto* < *Baixo* − 1

$\sum_{k=4}^{3}(k)$ → 0

As fórmulas da soma utilizadas derivam da seguinte referência :

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$\sum_{k=4}^{1}(k)$ → −5

$\sum_{k=4}^{1}(k)+\sum_{k=2}^{4}(k)$ → 4

### ΣInt()

Catálogo > 

**ΣInt(***NPmt1***,** *NPmt2***,** *N***,** *I***,** *PV***,** [ *Pmt* ]**,** [ *FV* ]**,** [ *PpY* ]**,** [ *CpY* ]**,** [ *PmtAt* ]**,** [ *ValorArredondado* ]**)** ⇒ *valor*

**ΣInt(***NPmt1***,** *NPmt2***,** *TabelaDeDepreciação***)** ⇒ *valor*

Função de amortização que calcula a soma do juro durante um intervalo especificado de pagamentos.

*NPmt1* e *NPmt2* definem os limites iniciais e finais do intervalo de pagamentos.

*N*, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 111.

- Se omitir *Pmt*, predefine-se para *Pmt* = **tvmPmt(***N, I, PV, FV, PpY, CpY, PmtAt***)**.
- Se omitir *FV*, predefine-se para *FV* =0.
- As predefinições para *PpY*, *CpY* e *PmtAt* são iguais às predefinições para as funções TVM.

*ValorArredondado* especifica o número de casas decimais para arredondamento. Predefinição=2.

**ΣInt(***NPmt1*,*NPmt2*, *TabelaDeDepreciação***)** calcula a soma dos juros com base na tabela de amortização *TabelaDeDepreciação*. O argumento *TabelaDeDepreciação* tem de ser uma matriz na forma descrita em **amortTbl()**, página 6.

**Nota:** Consulte também **ΣPrn()**, abaixo, e **Bal()**, página 12.

ΣInt(1,3,12,4.75,20000,,12,12) −213.48

*tbl*:=amortTbl(12,12,4.75,20000,,12,12)

| | | | |
|---|---|---|---|
| 0 | 0. | 0. | 20000. |
| 1 | −77.49 | −1632.43 | 18367.6 |
| 2 | −71.17 | −1638.75 | 16728.8 |
| 3 | −64.82 | −1645.1 | 15083.7 |
| 4 | −58.44 | −1651.48 | 13432.2 |
| 5 | −52.05 | −1657.87 | 11774.4 |
| 6 | −45.62 | −1664.3 | 10110.1 |
| 7 | −39.17 | −1670.75 | 8439.32 |
| 8 | −32.7 | −1677.22 | 6762.1 |
| 9 | −26.2 | −1683.72 | 5078.38 |
| 10 | −19.68 | −1690.24 | 3388.14 |
| 11 | −13.13 | −1696.79 | 1691.35 |
| 12 | −6.55 | −1703.37 | −12.02 |

ΣInt(1,3,*tbl*) −213.48

### ΣPrn()

Catálogo > 

**ΣPrn(***NPmt1***,** *NPmt2***,** *N***,** *I***,** *PV***,** [ *Pmt* ]**,** [ *FV* ]**,** [ *PpY* ]**,** [ *CpY* ]**,** [ *PmtAt* ]**,** [ *ValorArredondado* ]**)** ⇒ *valor*

**ΣPrn(***NPmt1***,** *NPmt2***,** *TabelaDeDepreciação***)** ⇒ *valor*

Função de amortização que calcula a soma do capital durante um intervalo especificado de pagamentos.

*NPmt1* e *NPmt2* definem os limites iniciais e finais do intervalo de pagamentos.

*N*, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* e *PmtAt* são descritos na tabela de argumentos TVM, página 111.

- Se omitir *Pmt*, predefine-se para *Pmt* = **tvmPmt(***N, I, PV, FV, PpY, CpY, PmtAt***)**.
- Se omitir *FV*, predefine-se para *FV* =0.
- As predefinições para *PpY*, *CpY* e *PmtAt* são iguais às predefinições para as funções TVM.

*ValorArredondado* especifica o número de casas decimais para arredondamento. Predefinição=2.

**ΣPrn(***NPmt1*,*NPmt2*, *TabelaDeDepreciação***)** calcula a soma do capital pago com base na tabela de amortização *TabelaDeDepreciação*. O argumento *TabelaDeDepreciação* tem de ser uma matriz na forma descrita em **amortTbl()**, página 6.

**Nota:** Consulte também **ΣInt()**, acima, e **Bal()**, página 12.

ΣPrn(1,3,12,4.75,20000,,12,12) −4916.28

*tbl*:=amortTbl(12,12,4.75,20000,,12,12)

| | | | |
|---|---|---|---|
| 0 | 0. | 0. | 20000. |
| 1 | −77.49 | −1632.43 | 18367.57 |
| 2 | −71.17 | −1638.75 | 16728.82 |
| 3 | −64.82 | −1645.1 | 15083.72 |
| 4 | −58.44 | −1651.48 | 13432.24 |
| 5 | −52.05 | −1657.87 | 11774.37 |
| 6 | −45.62 | −1664.3 | 10110.07 |
| 7 | −39.17 | −1670.75 | 8439.32 |
| 8 | −32.7 | −1677.22 | 6762.1 |
| 9 | −26.2 | −1683.72 | 5078.38 |
| 10 | −19.68 | −1690.24 | 3388.14 |
| 11 | −13.13 | −1696.79 | 1691.35 |
| 12 | −6.55 | −1703.37 | −12.02 |

ΣPrn(1,3,*tbl*) −4916.28

### # (indirecta)

**Teclas** ctrl 🕮

**#** *CadeiaDeNomeDaVar*

Refere-se à variável cujo nome é *CadeiaDeNomeDaVar*. Permite utilizar cadeias para criar nomes das variáveis a partir de uma função.

| | |
|---|---|
| $xyz:=12$ | 12 |
| $\#("x"\&"y"\&"z")$ | 12 |

Cria ou refere-se à variável xyz.

| | |
|---|---|
| $10\rightarrow r$ | 10 |
| $"r"\rightarrow s1$ | "r" |
| $\#s1$ | 10 |

Devolve o valor da variável (r) cujo nome é guardado na variável s1.

### E (notação científica)

**Tecla** EE

*mantissa* **E** *expoente*

Introduz um número em notação científica. O número é interpretado como *mantissa* × 10 ^expoente.

Sugestão: Se quiser introduzir uma potência de 10 sem resultar num resultado de valor decimal, utilize 10^ *número inteiro*.

**Nota:** Pode introduzir este operador através da escrita de **@E** no teclado do computador. por exemplo, escreva **2.3@E4** para introduzir 2.3**E**4.

| | |
|---|---|
| 23000. | 23000. |
| 2300000000.+4.1E15 | 4.1E15 |
| $3\cdot 10^4$ | 30000 |

### g (gradianos)

**Tecla** π▸

*Expr1* **g** ⇒ *expressão*
*Lista1* **g** ⇒ *lista*
*Matriz1* **g** ⇒ *matriz*

Esta função fornece uma forma para especificar um ângulo de gradianos enquanto está no modo Graus ou Radianos.

No modo de ângulo Radianos, multiplica *Expr1* por π/200.

No modo de ângulo Graus, multiplica *Expr1* por g/100.

No modo Gradianos, devolve *Expr1* inalterada.

**Nota:** Pode introduzir este símbolo através da escrita de **@g** no teclado do computador.

No modo Graus, Gradianos ou Radianos:

| | |
|---|---|
| $\cos(50^g)$ | 0.707107 |
| $\cos(\{0,100^g,200^g\})$ | {1.,0.,-1.} |

### r (radianos)

**Tecla** π▸

*Valor1* **r** ⇒ *valor*
*Lista1* **r** ⇒ *lista*
*Matriz1* **r** ⇒ *matriz*

Esta função fornece uma forma para especificar um ângulo de radianos enquanto está no modo Graus ou Gradianos.

No modo de ângulo Graus, multiplica o argumento por 180/ π.

No modo de ângulo Radianos, devolve o argumento inalterado.

No modo Gradianos, multiplica o argumento por 200/ π.

Sugestão: Utilize **r** se quiser impor os radianos numa definição da função, independentemente do modo que prevalece quando a função é utilizada.

**Nota:** Pode introduzir este símbolo através da escrita de **@r** no teclado.

No modo de ângulo Graus, Gradianos ou Radianos:

| | |
|---|---|
| $\cos\left(\frac{\pi}{4^r}\right)$ | 0.707107 |
| $\cos\left(\left\{0^r,\left(\frac{\pi}{12}\right)^r,-(\pi)^r\right\}\right)$ | {1.,0.965926,-1.} |

### ° (graus)

**Tecla** π

*Valor1* ° ⇒ *valor*
*Lista1* ° ⇒ *lista*
*Matriz1* ° ⇒ *matriz*

Esta função fornece uma forma para especificar um ângulo expresso em graus enquanto está no modo Radianos ou Radianos.

No modo de ângulo Radianos, multiplica o argumento por π/180.

No modo de ângulo Graus, devolve o argumento inalterado.

No modo de ângulo Gradianos, multiplica o argumento por 10/9.

**Nota:** Pode introduzir este símbolo através da escrita de **@d** no teclado do computador.

No modo de ângulo Graus, Gradianos ou Radianos:

$\cos(45°)$ → $0.707107$

No modo de ângulo Radianos:

$\cos\left\{\left\{0,\frac{\pi}{4},90°,30.12°\right\}\right\}$ → $\{1.,0.707107,0.,0.864976\}$

### °, ', '' (grau/minuto/segundo)

**Teclas** ctrl 📖

*gg* °*mm* ' *ss.ss* '' ⇒ *expressão*

*gg* Um número positivo ou negativo

*mm* Um número não negativo

*ss.ss* Um número não negativo

Devolve *gg* +(*mm* /60)+(*ss.ss* /3600).

Este formato de entrada base -60 permite:

- Introduza um ângulo em graus/minutos/segundos sem se preocupar com o modo de ângulo actual.
- Introduza o tempo como horas/minutos/segundos.

**Nota:** Introduza dois apóstrofos a seguir ss.ss (''), não um símbolo de aspas (").

No modo de ângulo Graus:

$25°13'17.5''$ → $25.2215$

$25°30'$ → $\frac{51}{2}$

### ∠ (ângulo)

**Teclas** ctrl 📖

[ *Raio*, ∠θ_*Ângulo* ] ⇒ *vector*
(entrada polar)

[ *Raio*, ∠θ_*Ângulo*, *Z_Coordenada* ] ⇒ *vector*
(entrada cilíndrica)

[ *Raio*, ∠θ_*Ângulo*, ∠θ_*Ângulo* ] ⇒ *vector*
(entrada esférica)

Devolve coordenadas como um vector dependendo da definição do modo Formato do vector: rectangular, cilíndrico ou esférico.

**Nota:** Pode introduzir este símbolo através da escrita de **@<** no teclado do computador.

No modo Radianos e formato do vector definido para: rectangular

$[5 \angle 60° \angle 45°]$ → $[1.76777 \quad 3.06186 \quad 3.53553]$

cilíndrico

$[5 \angle 60° \angle 45°]$ → $[3.53553 \angle 1.0472 \quad 3.53553]$

esférico

$[5 \angle 60° \angle 45°]$ → $[5. \angle 1.0472 \angle 0.785398]$

(*Magnitude* ∠ *Ângulo*) ⇒ *ValorComplexo*
(entrada polar)

Introduz um valor complexo em forma polar (r ∠θ). O *Ângulo* é interpretado de acordo com a definição do modo Ângulo actual.

No modo de ângulo Radianos e Formato complexo rectangular:

$5+3\cdot i\angle\left(10 \angle \frac{\pi}{4}\right)$ → $-2.07107-4.07107\cdot i$

### _ (carácter de sublinhado como um elemento vazio)

Consulte "Elementos (nulos) vazios", página 136.

## 10^()

Catálogo >

**10^(***Valor1***)** ⇒ *valor*
**10^(***Lista1***)** ⇒ *lista*

Devolve 10 elevado à potência do argumento.

Para uma lista, devolve 10 elevado à potência dos elementos em *Lista1*.

**10^(***MatrizQuadrada1***)** ⇒ *MatrizQuadrada*

Devolve 10 elevado à potência de *MatrizQuadrada1*. Isto não é o mesmo que calcular 10 elevado à potência de cada elemento. Para mais informações sobre o método de cálculo, consulte **cos()**.

*MatrizQuadrada1* tem de ser diagnolizável. O resultado contém sempre os números de ponto flutuante.

$10^{1.5}$ → 31.6228

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

$$\begin{bmatrix}1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6\\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6\\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6\end{bmatrix}$$

## ^-1 (recíproco)

Catálogo >

*Valor1* **^-1** ⇒ *valor*
*Lista1* **^-1** ⇒ *lista*

Devolve o recíproco do argumento.

Para uma lista, devolve os recíprocos dos elementos em *Lista1*.

*MatrizQuadrada1* **^-1** ⇒ *MatrizQuadrada*

Devolve o inverso de *MatrizQuadrada1*.

*MatrizQuadrada1* tem de ser uma matriz quadrada não singular.

$(3.1)^{-1}$ → 0.322581

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

## | (operador de limite)

Teclas ctrl

*Expr* **|** *ExprBooleana1* [**and** *ExprBooleana2*]...
*Expr* **|** *ExprBooleana1* [**or** *ExprBooleana2*]...

O símbolo de limite ("|") serve como um operador binário. O operando à esquerda de | é uma expressão. O operando à direita de | especifica uma ou mais relações que servem para afetar a simplificação da expressão. Várias relações após | têm de ser reunidas por operadores "**and**" ou "**or**" lógicos.

O operador de limite fornece três tipos de funcionalidades básicas:

- Substituições
- Limites de intervalo
- Exclusões

As substituições estão na forma de uma igualdade, como x=3 ou y=sin(x). Para ser mais eficaz, o membro esquerdo deve ser uma variável simples. *Expr* | *Variável* = *valor* substituem *valor* para todas as ocorrências de *Variável* em *Expr*.

$x+1|x=3$ → 4

$x+55|x=\sin(55)$ → 54.0002

$x^3-2\cdot x+7\rightarrow f(x)$ → *Done*

$f(x)|x=\sqrt{3}$ → 8.73205

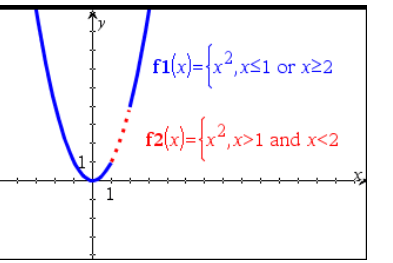
### | (operador de limite)

**Teclas** ctrl

Os limites de intervalos tomam a forma de uma ou mais desigualdades reunidas pelos operadores "**and**" ou "**or**" lógicos. Os limites de intervalos também permitem a simplificação que caso contrário pode ser inválida ou não calculável.

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0, x)$ → 0.

$\text{nSolve}(x^3+2\cdot x^2-15\cdot x=0, x) \mid x>0 \text{ and } x<5$ → 3.




As exclusões utilizam o operador relacional "diferentes" (/= ou ≠) para excluir um valor específico de consideração.

### → (guardar)

**Teclas** ctrl var

*Value* → *Var*
*Lista* → *Var*
*Matriz* → *Var*
*Expr* → *Função*(*Parâm1*,...)
*Lista* → *Função*(*Parâm1*,...)
*Matriz* → *Função*(*Parâm1*,...)

Se a variável *Var* não existir, cria-a e inicia-a para *Valor*, *Lista* ou *Matriz*.

Se a variável *Var* já existir e não estiver bloqueada nem protegida, substitui o conteúdo por *Valor*, *Lista* ou *Matriz*.

**Nota:** Pode introduzir este operador através da escrita de **=: no teclado** como um atalho. Por exemplo, escreva **pi/ 4 =: myvar**.

$\frac{\pi}{4} \rightarrow myvar$ → 0.785398

$2\cdot\cos(x) \rightarrow y1(x)$ → *Done*

$\{1,2,3,4\} \rightarrow lst5$ → $\{1,2,3,4\}$

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \rightarrow matg$ → $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$

"Hello" → *str1* → "Hello"

### := (atribuir)

**Teclas** ctrl

*Var* **:=** *Valor*
*Var* **:=** *Lista*
*Var* **:=** *Matriz*
*Função*(*Parâm1*,...) **:=** *Expr*
*Função*(*Parâm1*,...) **:=** *Lista*
*Função*(*Parâm1*,...) **:=** *Matriz*

Se a variável *Var* não existir, cria *Var* e inicia-a para *Valor*, *Lista* ou *Matriz*.

Se *Var* já existir e não estiver bloqueada nem protegida, substitui o conteúdo por *Valor*, *Lista* ou *Matriz*.

$myvar := \frac{\pi}{4}$ → .785398

$y1(x) := 2\cdot\cos(x)$ → *Done*

$lst5 := \{1,2,3,4\}$ → $\{1,2,3,4\}$

$matg := \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$ → $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$

*str1* := "Hello" → "Hello"

### © (comentário)

**Teclas** ctrl 📖

**©** [ *texto*]

**©** processa *texto* como uma linha de comentário, permitindo anotar as funções e os programas criados.

**©** pode estar no início ou em qualquer parte da linha. Tudo à direita de **©**, no fim da linha, é o comentário.

**Nota para introdução do exemplo:** Na aplicação Calculadora da unidade portátil, pode introduzir definições multilinhas, premindo ↵ em vez de enter no fim de cada linha. No teclado do computador, prima sem soltar **Alt** e prima **Enter**.

```
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

**Teclas** 0 B, **teclas** 0 H

**0b** *NúmeroBinário*
**0h** *NúmeroHexadecimal*

Indica um número binário ou hexadecimal, respectivamente. Para introduzir um número binário ou hexadecimal, utilize sempre o prefixo 0b ou 0h independentemente do modo Base. Sem um prefixo, um número é tratado como decimal (base 10).

Os resultados aparecem de acordo com o modo base.

No modo base Dec:

```
0b10+0hF+10                         27
```

No modo base Bin:

```
0b10+0hF+10                    0b11011
```

No modo base Hex:

```
0b10+0hF+10                       0h1B
```

# *Elementos (nulos) vazios*

Quando analisar dados do mundo real, pode não ter sempre um conjunto de dados completo. A TI-Nspire™ permite elementos de dados, vazios ou nulos, para que possa continuar com os dados quase completos em vez de ter de reiniciar ou eliminar os casos incompletos.

Pode encontrar um exemplo de dados que envolve elementos vazios no capítulo Listas e Folha de cálculo, em "*Representar graficamente os dados da folha de cálculo.*"

A função **delVoid()** permite remover os elementos vazios de uma lista. A função **isVoid()** permite testar um elemento vazio. Para mais informações, consulte **delVoid()**, página 29, e **isVoid()**, página 49.

**Nota:** Para introduzir um elemento vazio manualmente numa expressão de matemática, escreva "_" ou a palavra-chave **void**. A palavra-chave **void** é convertida automaticamente para um símbolo "_" quando a expressão for avaliada. Para escrever "_" na unidade portátil, prima ctrl ␣.

### Cálculos que envolvam elementos nulos

A maioria dos cálculos que envolvam uma entrada nula produz um resultado nulo. Consulte os casos especiais abaixo.

| | |
|---|---|
| $\lvert\_\rvert$ | _ |
| $\gcd(100,\_)$ | _ |
| $3+\_$ | _ |
| $\{5,\_,10\}-\{3,6,9\}$ | $\{2,\_,1\}$ |

### Argumentos da lista que contenham elementos nulos

As seguintes funções e comandos ignoram os elementos nulos encontrados nos argumentos da lista.

**count**, **countIf**, **cumulativeSum**, **freqTable▸list**, **frequency**, **max**, **mean**, **median**, **product**, **stDevPop**, **stDevSamp**, **sum**, **sumIf**, **varPop**, e **varSamp**, assim como cálculos de regressão, **OneVar**, **TwoVar**, e estatística **FiveNumSummary**, intervalos de confiança e testes estatísticos

| | |
|---|---|
| $\text{sum}(\{2,\_,3,5,6.6\})$ | 16.6 |
| $\text{median}(\{1,2,\_,\_,\_,3\})$ | 2 |
| $\text{cumulativeSum}(\{1,2,\_,4,5\})$ | $\{1,3,\_,7,12\}$ |
| $\text{cumulativeSum}\left(\begin{bmatrix}1 & 2\\ 3 & \_\\ 5 & 6\end{bmatrix}\right)$ | $\begin{bmatrix}1 & 2\\ 4 & \_\\ 9 & 8\end{bmatrix}$ |

**SortA** e **SortD** movem todos os elementos nulos no primeiro argumento para a parte inferior.

| | |
|---|---|
| $\{5,4,3,\_,1\}\rightarrow list1$ | $\{5,4,3,\_,1\}$ |
| $\{5,4,3,2,1\}\rightarrow list2$ | $\{5,4,3,2,1\}$ |
| SortA *list1*,*list2* | *Done* |
| *list1* | $\{1,3,4,5,\_\}$ |
| *list2* | $\{1,3,4,5,2\}$ |

| | |
|---|---|
| $\{1,2,3,\_,5\}\rightarrow list1$ | $\{1,2,3,\_,5\}$ |
| $\{1,2,3,4,5\}\rightarrow list2$ | $\{1,2,3,4,5\}$ |
| SortD *list1*,*list2* | *Done* |
| *list1* | $\{5,3,2,1,\_\}$ |
| *list2* | $\{5,3,2,1,4\}$ |

**Argumentos da lista que contenham elementos nulos(continued)**

Nas regressões, um nulo numa lista X ou Y introduz um nulo para o elemento correspondente do resíduo.

| | |
|---|---|
| *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.} |

Uma categoria omitida nas regressões introduz um nulo para o elemento correspondente do residual.

| | |
|---|---|
| *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.} |

Uma frequência de 0 nas regressões introduz um nulo para o elemento correspondente do residuo.

| | |
|---|---|
| *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.} |

# *Atalhos para introduzir expressões matemáticas*

Os atalhos permitem introduzir elementos das expressões matemáticas, escrevendo, em vez da utilização do Catálogo ou da Palete de símbolos. Por exemplo, para introduzir a expressão $\sqrt{6}$, pode escrever **sqrt(6)** na linha de entrada. Quando premir enter, a expressão **sqrt(6)** é alterada para $\sqrt{6}$. Alguns atalhos são úteis na unidade portátil e no teclado do computador. Outros são úteis principalmente no teclado do computador.

## Na unidade portátil ou no teclado do computador

| Para introduzir este: | Escreva este atalho: |
|---|---|
| $\pi$ | **pi** |
| $\theta$ | **theta** |
| $\infty$ | **infinity** |
| $\leq$ | **<=** |
| $\geq$ | **>=** |
| $\neq$ | **/=** |
| $\Rightarrow$ (implicação lógica) | **=>** |
| $\Leftrightarrow$ (implicação lógica dupla, XNOR) | **<=>** |
| **→** (guardar operador) | **=:** |
| \| \| (valor absoluto) | **abs(…)** |
| $\sqrt{}$**()** | **sqrt(…)** |
| $\Sigma$**()** (Modelo da soma) | **sumSeq(…)** |
| $\Pi$**()** (Modelo da produto) | **prodSeq(…)** |
| $\sin^{-1}$**()**, $\cos^{-1}$**()**, … | **arcsin(…)**, **arccos(…)**, … |
| $\Delta$**List()** | **deltaList(…)** |

## No teclado do computador

| Para introduzir este: | Escreva este atalho: |
|---|---|
| ***i*** (constante imaginária) | **@i** |
| e (base logarítmica natural e) | **@e** |
| E (notação científica) | **@E** |
| T (transpor) | **@t** |
| r (radianos) | **@r** |

| Para introduzir este: | Escreva este atalho: |
|---|---|
| ° (graus) | `@d` |
| g (grados) | `@g` |
| ∠ (ângulo) | `@<` |
| ▸ (conversão) | `@>` |
| ▸Decimal, ▸approxFraction(), etc. | `@>Decimal`, `@>approxFraction()`, etc. |

# *Hierarquia do EOS™ (Equation Operating System)*

Esta secção descreve o Equation Operating System (EOS™) utilizado pela tecnologia de aprendizagem de matemática e ciências TI-Nspire™. Os números, as variáveis e as funções são introduzidos numa sequência simples O software EOS™ avalia as expressões e as equações com a associação parentética e de acordo com as prioridades descritas abaixo.

## Ordem de avaliação

| Nível | Operador |
|---|---|
| 1 | Parêntesis curvos ( ), parêntesis rectos [ ], chavetas { } |
| 2 | Indirecta (#) |
| 3 | Chamadas de funções |
| 4 | Pós-operadores: graus-minutos-segundos (°,', "), factorial (!), percentagem (%), radianos ( ʳ ), carácter de sublinhado ([ ]), transpor (ᵀ) |
| 5 | Exponenciação, operador de potência (^) |
| 6 | Negação (¯) |
| 7 | Concatenação de cadeias (&) |
| 8 | Multiplicação (•), divisão (/) |
| 9 | Adição (+), subtracção (-) |
| 10 | Relações de igualdade: igual (=), não igual (≠ ou /=), menor que (<), igual ou menor que (≤ ou <=), maior que (>), igual ou maior que (≥ ou >=) |
| 11 | not **lógico** |
| 12 | and **lógico** |
| 13 | Lógico **or** |
| 14 | **xou**, **nor**, **nand** |
| 15 | Implicação lógica (⇒) |
| 16 | Implicação lógica dupla, XNOR (⇔) |
| 17 | Operador de limite ("\|") |
| 18 | Guardar (→) |

## Parêntesis curvos, parêntesis rectos e chavetas

Todos os cálculos dentro de um par de parêntesis rectos, parêntesis curvos ou chavetas são avaliados primeiro Por exemplo, na expressão 4(1+2), o software EOS™ avalia primeiro a parte da expressão dentro dos parêntesis, 1+2, e, em seguida, multiplica o resultado, 3, por 4.

O número de parêntesis curvos, parêntesis rectos e chavetas de abertura e fecho tem de ser igual numa expressão ou equação. Se não for, aparece uma mensagem de erro que indica o elemento inexistente. Por exemplo, (1+2)/(3+4 mostra a mensagem de erro "Inexistente )."

**Nota:** Como o software TI-Nspire™ permite definir as suas funções próprias, o nome de uma variável seguido por uma expressão entre parêntesis é considerado uma "chamada de função" em vez de uma multiplicação implícita. Por exemplo, *a*(b+c) é a função *a* avaliada por b+c. Para multiplicar a expressão b+c pela variável *a*, utilize a multiplicação explícita: *a*∗(b+c).

## Indirecta

O operador da indirecta (#) converte uma cadeia num nome de função ou variável. Por exemplo, #("x"&"y"&"z") cria o nome de variável xyz. A indirecta permite também a criação e a modificação de variáveis dentro de um programa. Por exemplo, se 10→r e "r"→s1, #s1=10.

## Pós-operadores

Os pós-operadores são operadores que vêm directamente após um argumento, como 5!, 25% ou 60°15' 45. Os argumentos seguidos por um pós-operador são avaliados no quarto nível de prioridade. Por exemplo, na expressão 4^3!, 3! é avaliada primeiro. O resultado, 6, torna-se no expoente de 4 para produzir 4096.

## Exponenciação

A exponenciação (^) e a exponenciação de elemento por elemento (.^) são avaliadas da direita para a esquerda. Por exemplo, a expressão 2^3^2 é avaliada como 2^(3^2) para produzir 512. É diferente de (2^3)^2, que é 64.

## Negação

Para introduzir um número negativo, prima (−) seguida pelo número. As pós-operações e a exponenciação são efectuadas antes da negação. Por exemplo, o resultado de $-x^2$ é um número negativo e $-9^2 = -81$. Utilize os parêntesis para elevar um número negativo ao quadrado $(-9)^2$ para produzir 81.

## Limite ("|")

O argumento a seguir ao operador de limite ("|") fornece um conjunto de limites que afetam a avaliação do argumento antes do operador.

# *Mensagens e códigos de erros*

Quando ocorre um erro, o código é atribuído à variável *errCode*. As funções e os programas definidos pelos utilizadores podem examinar *errCode* para determinar a causa de um erro. Para obter um exemplo da utilização de *errCode*, consulte o Exemplo 2 no comando **Try**, página 108.

**Nota:** Algumas condições de erro aplicam-se apenas aos produtos TI-Nspire™ CAS e algumas aplicam-se apenas aos produtos TI-Nspire™.

| Código de erro | Descrição |
|---|---|
| 10 | Uma função não devolveu um valor |
| 20 | Um teste não resolveu para VERDADEIRO ou FALSO.<br>Geralmente, as variáveis indefinidas não podem ser comparadas. Por exemplo, o teste If a<b provocará este erro se a ou b forem indefinidos quando a afirmação If for executada. |
| 30 | O argumento não pode ser o nome de uma pasta. |
| 40 | Erro do argumento |
| 50 | Argumentos não coincidentes<br>Dois ou mais argumentos têm de ser do mesmo tipo. |
| 60 | O argumento tem de ser uma expressão Booleana ou um número inteiro |
| 70 | O argumento tem de ser um número decimal |
| 90 | O argumento tem de ser uma lista |
| 100 | O argumento tem de ser uma matriz |
| 130 | O argumento tem de ser um conjunto de caracteres alfanuméricos |
| 140 | O argumento tem de ser o nome de uma variável.<br>Certifique-se de que o nome:<br>• não começa por um dígito<br>• não contém espaços ou caracteres especiais<br>• não utiliza o carácter de sublinhado ou um intervalo de forma inválida<br>• não excede as limitações do comprimento<br>Consulte a secção Calculadora para obter mais informações. |
| 160 | O argumento tem de ser uma expressão |
| 165 | Pilhas demasiado fracas para envio ou recepção<br>Instale pilhas novas antes do envio ou da recepção. |
| 170 | Limite<br>O limite inferior tem de ser inferior ao limite superior para definir o intervalo da procura. |
| 180 | Pausa<br>A tecla esc ou on foi premida durante um cálculo longo ou a execução do programa. |
| 190 | Definição circular<br>Esta mensagem aparece para evitar o esgotamento da memória durante a substituição infinita de valores das variáveis durante a simplificação. Por exemplo, a+1->a, em que a é uma variável indefinida, provocará este erro. |
| 200 | Expressão de constrangimento inválida<br>Por exemplo, solve(3x^2-4=0,x) \| x<0 ou x>5 produzirá esta mensagem de erro porque a restrição é separada por "or" em vez de "and." |
| 210 | Tipo de dados inválido<br>Um argumento é do tipo de dados errado. |
| 220 | Limite dependente |

| Código de erro | Descrição |
|---|---|
| 230 | Dimensão<br>Um índice de lista ou matriz não é válido. Por exemplo, se a lista {1,2,3,4} for guardada em L1, L1[5] é um erro de dimensão porque L1 contém apenas quatro elementos. |
| 235 | Erro de dimensão. Elementos insuficientes nas listas. |
| 240 | Erro de dimensão<br>Dois ou mais argumentos têm de ter as mesmas dimensões. Por exemplo, [1,2]+[1,2,3] é uma incorrespondência de dimensões porque as matrizes contêm um número de elementos diferentes. |
| 250 | Dividir por zero |
| 260 | Erro do domínio<br>Um argumento tem de estar num domínio específico. Por exemplo, **rand(0)** não válido. |
| 270 | Nome da variável duplicado |
| 280 | Else e ElseIf inválidas fora do bloco If..EndIf |
| 290 | EndTry não tem a afirmação Else correspondente |
| 295 | Iteração excessiva |
| 300 | Matriz ou lista de 2 ou 3 elementos prevista |
| 310 | O primeiro argumento de **nSolve** tem de ser uma equação de variável individual. Não pode conter uma variável sem valor diferente da variável de interesse. |
| 320 | O primeiro argumento de solve ou cSolve tem de ser uma equação ou desigualdade<br>Por exemplo, solve(3x^2-4,x) não é válido porque o primeiro argumento não é uma equação. |
| 345 | Unidades inconsistentes |
| 350 | Índice fora do intervalo |
| 360 | O nome não é um nome de variável válido |
| 380 | Ans indefinida<br>O cálculo anterior não criou Ans ou nenhum cálculo anterior foi introduzido. |
| 390 | Atribuição inválida |
| 400 | Valor de atribuição inválido |
| 410 | Comando inválido |
| 430 | Inválido para as definições actuais do modo |
| 435 | Tentativa inválida |
| 440 | Multiplicação implícita inválida<br>Por exemplo, x(x+1) não é válida; visto que, x*(x+1) é a sintaxe correcta. Esta serve para evitar confusões entre as chamadas de funções e a multiplicação implícita. |
| 450 | Inválida numa função ou expressão actual<br>Apenas determinados comandos são válidos numa função definida pelo utilizador. |
| 490 | Inválido no bloco Try..EndTry |
| 510 | Matriz ou lista inválida |
| 550 | Programa ou função exterior inválido<br>Vários comandos não são válidos fora de uma função ou de um programa. Por exemplo, **Local** não pode ser utilizado excepto se estiver numa função ou num programa. |
| 560 | Inválido fora dos blocos Loop..EndLoop, For..EndFor ou While..EndWhile<br>Por exemplo, o comando Exit só válido dentro destes blocos circulares. |
| 565 | Programa exterior inválido |

| Código de erro | Descrição |
|---|---|
| 570 | Nome do caminho inválido<br>Por exemplo, \var não é válido. |
| 575 | Complexo polar inválido |
| 580 | Referência de programa inválida<br>Os programas não podem ser referenciados nas funções ou expressões, como, por exemplo, 1+p(x) em que p é um programa. |
| 600 | Tabela inválida |
| 605 | Utilização de unidades inválidas |
| 610 | Nome de variável inválido numa instrução Local |
| 620 | Nome de função ou variável inválido |
| 630 | Referência da variável inválida |
| 640 | Sintaxe de vector inválida |
| 650 | Transmissão da ligação<br>Uma transmissão entre as duas unidades não foi concluída. Verifique se o cabo de ligação foi está ligado correctamente a ambas as extremidades. |
| 665 | Matriz não diagonalizável |
| 670 | Pouca memória<br>1. Eliminar alguns dados deste documento<br>2. Guardar e fechar este documento<br>Se 1 e 2 não resultarem, retirar e reinserir as pilhas |
| 672 | Esgotamento de recursos |
| 673 | Esgotamento de recursos |
| 680 | Falta ( |
| 690 | Falta) |
| 700 | Falta " |
| 710 | Falta ] |
| 720 | Falta } |
| 730 | Falta do início ou do fim da sintaxe do bloco |
| 740 | Falta Then no bloco If..EndIf |
| 750 | Nome não é uma função nem um programa |
| 765 | Nenhuma função seleccionada |
| 780 | Nenhuma solução encontrada |
| 800 | Resultado não real<br>Por exemplo, se o software estiver na definição real, $\sqrt{(-1)}$ não é válido.<br>Para permitir resultados em complexos, altere a definição do modo "Real ou Complexo" para RECTANGULAR ou POLAR. |
| 830 | Excesso |
| 850 | Programa não encontrado<br>Uma referência do programa dentro de outro programa não pode ser encontrada no caminho fornecido durante a execução. |
| 855 | Funções de tipo Rand não permitidas no gráfico |

| Código de erro | Descrição |
|---|---|
| 860 | Recursividade muito profunda |
| 870 | Variável do sistema ou nome reservado |
| 900 | Erro do argumento<br>O modelo mediana-mediana não pode ser aplicado ao conjunto de dados. |
| 910 | Erro de sintaxe |
| 920 | Texto não encontrado |
| 930 | Poucos argumentos<br>A função ou o comando não tem um ou mais argumentos. |
| 940 | Demsiados argumentos<br>A expressão ou equação contém um número excessivo de argumentos e não pode ser avaliada. |
| 950 | Demasiados índices |
| 955 | Demasiadas variáveis indefinidas |
| 960 | Variável indefinida<br>Nenhum valor atribuído à variável. Utilize um dos seguintes comandos:<br>• sto →<br>• **:=**<br>• **Define**<br>para atribuir valores às variáveis. |
| 965 | SO não licenciado |
| 970 | Variável em utilização para que as referências ou as alterações não sejam permitidas |
| 980 | Variável protegida |
| 990 | Nome da variável inválido<br>Certifique-se de que o nome não excede as limitações de comprimento |
| 1000 | Domínio das variáveis da janela |
| 1010 | Zoom |
| 1020 | Erro interno |
| 1030 | Violação da memória protegida |
| 1040 | Função não suportada. Esta função requer o Computer Algebra System. Tente o TI-Nspire™ CAS. |
| 1045 | Operador não suportado. Este operador requer o Computer Algebra System. Tente o TI-Nspire™ CAS. |
| 1050 | Função não suportada. Este operador requer o Computer Algebra System. Tente o TI-Nspire™ CAS. |
| 1060 | O argumento de entrada tem de ser numérico. Apenas entradas com valores numéricos são permitidas. |
| 1070 | Argumento da função Trig demasiado grande para redução precisa |
| 1080 | Utilização não suportada de Ans. Esta aplicação não suporta Ans. |
| 1090 | Função indefinida. Utilize um dos seguintes comandos:<br>• **Define**<br>• **:=**<br>• sto →<br>para definir uma função. |
| 1100 | Cálculo não real<br>Por exemplo, se o software estiver na definição real, $\sqrt{}(-1)$ não é válido.<br>Para permitir resultados em complexos, altere a definição do modo "Real ou Complexo" para RECTANGULAR ou POLAR. |

| Código de erro | Descrição |
|---|---|
| 1110 | Limites inválidos |
| 1120 | Nenhuma alteração de sinal |
| 1130 | O argumento não pode ser uma lista ou matriz |
| 1140 | Erro do argumento<br>O primeiro argumento tem de ser uma expressão polinomial no segundo argumento. Se o segundo argumento for omitido, o software tenta seleccionar uma predefinição. |
| 1150 | Erro do argumento<br>Os primeiros dois argumentos têm de ser uma expressão polinomial no terceiro argumento. Se o terceiro argumento for omitido, o software tenta seleccionar uma predefinição. |
| 1160 | Nome do caminho da biblioteca inválido<br>Um nome do caminho tem de estar no formato *xxx**yyy*, em que:<br>• A parte *xxx* pode ter de 1 a 16 caracteres.<br>• A parte *yyy* pode ter de 1 a 15 caracteres.<br>Consulte a secção Biblioteca na documentação para obter mais informações. |
| 1170 | Utilização inválida do nome do caminho da biblioteca<br>• Não pode atribuir um valor a um nome do caminho com **Define**, **:=**, ou sto **→**.<br>• Não pode declarar o nome de um caminho como uma variável local ou ser utilizada como um parâmetro numa definição de programa ou função. |
| 1180 | Nome da variável da biblioteca inválido.<br>Certifique-se de que o nome:<br>• não contém um ponto<br>• não começa com um carácter de sublinhado<br>• não excede 15 caracteres<br>Consulte a secção Biblioteca na documentação para obter mais informações. |
| 1190 | Documento da biblioteca não encontrado:<br>• Verifique se a biblioteca está na pasta MyLib.<br>• Actualizar bibliotecas.<br>Consulte a secção Biblioteca na documentação para obter mais informações. |
| 1200 | Variável da biblioteca não encontrada:<br>• Verifique se a variável da biblioteca existe no primeiro problema da biblioteca.<br>• Certifique-se de que a variável da biblioteca foi definida como BibPub ou BibPriv.<br>• Actualizar bibliotecas.<br>Consulte a secção Biblioteca na documentação para obter mais informações. |
| 1210 | Nome de atalho na biblioteca inválido.<br>Certifique-se de que o nome:<br>• não contém um ponto<br>• não começa com um carácter de sublinhado<br>• não excede 16 caracteres<br>• não é um nome reservado<br>Consulte a secção Biblioteca na documentação para obter mais informações. |
| 1220 | Erro de domínio:<br>As funções RectaTangente e RectaNormal suportam apenas funções reais de variável real. |
| 1230 | Erro de domínio.<br>Os operadores de conversão trigonométrica não são suportados nos modos de ângulos de graus ou grados. |
| 1250 | Erro do argumento<br>Utilize um sistema de equações lineares.<br>Exemplo de um sistema de duas equações lineares com variáveis x e y:<br>3x+7y=5<br>2y-5x=-1 |
| 1260 | Erro do argumento:<br>O primeiro argumento de **nfMin** ou **nfMax** tem de ser uma expressão numa variável individual. Não pode conter uma variável sem valor diferente da variável de interesse. |
| 1270 | Erro do argumento<br>A ordem da derivada tem de ser igual a 1 ou 2. |

| Código de erro | Descrição |
|---|---|
| 1280 | Erro do argumento<br>Utilize um polinómio num formato expandido numa variável. |
| 1290 | Erro do argumento<br>Utilize um polinómio numa variável. |
| 1300 | Erro do argumento<br>Tem de passar os coeficientes do polinómio para valores numéricos. |
| 1310 | Erro do argumento:<br>Uma função não conseguiu avaliar um ou mais argumentos. |
| 1380 | Erro de domínio:<br>Não são permitidas chamadas aninhadas para a função de domínio(). |

# *Códigos de aviso e mensagens*

Pode utilizar a função **warnCodes()** para guardar os códigos de avisos gerados ao avaliar uma expressão. Esta tabela lista todos os códigos de aviso numéricos e as mensagens associadas.

Para um exemplo de guardar códigos de aviso, consulte **warnCodes()**, página 114.

| Código de aviso | Mensagem |
|---|---|
| 10000 | A operação pode introduzir soluções falsas. |
| 10001 | A diferenciação de uma equação pode produzir uma equação falsa. |
| 10002 | Solução questionável |
| 10003 | Precisão questionável |
| 10004 | A operação pode perder as soluções. |
| 10005 | cSolve pode especificar mais zeros. |
| 10006 | Solve pode especificar mais zeros. |
| 10007 | Podem existir mais soluções. Tente especificar limites inferiores e superiores apropriados e/ou uma tentativa. Exemplos que utilizam solve():<br>• solve(Equação, Var=Tentativa)\|LimiteInferior<Var<LimiteSuperior<br>• solve(Equação, Var)\|LimiteInferior<Var<LimiteSuperior<br>• solve(Equação,Var=Tentativa) |
| 10008 | O domínio do resultado pode ser inferior ao domínio da entrada. |
| 10009 | O domínio do resultado pode ser superior ao domínio da entrada. |
| 10012 | Cálculo não real |
| 10013 | ^^0 ou undef^0 substituído por 1 |
| 10014 | undef^0 substituído por 1 |
| 10015 | 1^^ ou 1^undef substituído por 1 |
| 10016 | 1^undef substituído por 1 |
| 10017 | Capacidade excedida substituída por ∞ ou -∞ |
| 10018 | A operação requer e devolve um valor de 64 bits. |
| 10019 | Esgotamento de recursos, a simplificação pode estar incompleta. |

| Código de aviso | Mensagem |
|---|---|
| 10020 | Argumento da função trigonométrica demasiado para redução precisa. |
| 10021 | A entrada contém um parâmetro indefinido.<br>O resultado pode não ser válido para todos os valores de parâmetros possíveis. |
| 10022 | A especificação dos limites superiores e inferiores adequados pode produzir uma solução. |
| 10023 | Escalar foi multiplicado pela matriz de identidade. |
| 10024 | Resultado obtido utilizando aritmético aproximado. |
| 10025 | A equivalência não pode ser verificada no modo EXACTO. |
| 10026 | A restrição pode ser ignorada. Especifique a restrição na forma "\" 'Variable MathTestSymbol Constant' ou uma associação destas formas, por exemplo 'x<3 e x>-12' |

# Assistência e Suporte

## *Apoio técnico, manutenção e garantia dos produtos Texas Instruments*

**Apoio técnico e manutenção**

Para obter apoio técnico relativamente a produtos Texas Instruments, incluíndo informações de uso e/ou manutenção/ assistência técnica, por favor contacte-nos,

E-mail: ti-cares@ti.com

ou visite: education.ti.com

**Garantia do produto**

Para conhecer melhor os termos e a cobertura da garantia desta produto, por favor consulte o Termo de Garantia que o acompanha ou contacte o distribuidor/revendedor Texas Instruments mais próximo.

# Índice remissivo

## Símbolos


^ $^{-1}$, recíproco 133
^, potência 122
:=, atribuir 134
!, factorial 127
.^, ponto potência 123
.*, ponto multiplicação 123
.+, ponto adição 123
.−, ponto subtracção 123
.÷, ponto divisão 123
', notação de minutos 132
", notação de segundos 132
≤, igual ou menor que 126
©, comentário 135
Δlist( ), diferença da lista 55
°, graus/minutos/segundos 132
°, notação de graus 132
〈, raiz quadrada 128
〉, integral 128
 , diferente 125
−, subtrair 120
÷, dividir 121
Π, produto 129
Σ( ), soma 129
⇔, implicação lógica dupla 127
⇒, implicação lógica 127, 138
*, multiplicar 121
→, guardar 134
&, acrescentar 127
#, indirecta 131
#, operador da indirecta 141
%, percentagem 124
+, adicionar 120
<, menor que 126
=, igual 125
>, maior que 126
≥, igual ou maior que 126
|, operador de limite 133


## Numéricos


0b, indicador binário 135
0h, indicador hexadecimal 135
10^( ), potência de dez 133
▸FracçãoAprox( ) 10


## A


a definir
  função ou programa privado 28
  função ou programa público 29
abs( ), valor absoluto 6
acrescentar, & 127
adicionar, + 120
aleatória
  matriz, randMat( ) 83
  norma, randNorm( ) 83
aleatório
  polinómio, randPoly( ) 83
  semente de número, RandSeed 84
amortTbl( ), tabela de amortização 6, 12
amostra aleatória 83
and, Boolean operator 6
angle( ), ângulo 7
ângulo, angle( ) 7
ANOVA, análise de variação de uma via 7
ANOVA2way, análise de variação bidireccional 8
Ans, última resposta 9
apagar
  erro, ClrErr 17
approx( ), aproximado 10
aproximado, approx( ) 10
arccos() 10
arccosh() 10
arccot() 10
arccoth() 11
arccsc() 11
arccsch() 11
arco-coseno, $\cos^{-1}$( ) 20
arco-seno, $\sin^{-1}$( ) 96
arco-tangente, $\tan^{-1}$( ) 104
arcsec() 11
arcsech() 11
arcsin() 11
arcsinh() 11
arctan() 11
arctanh() 11
argumentos em funções TVM 111

Argumentos TVM 111
arredondar, round( ) 89
atalhos do teclado 138
atalhos, teclado 138
augment( ), aumentar/concatenar 11
aumentar/concatenar, augment( ) 11
avaliação, ordem de 140
avaliar polinómio, polyEval( ) 76
avgRC( ), taxa de câmbio média 12

## B

▶Base10, visualizar como número inteiro decimal 13
▶Base16, visualizar como hexadecimal 14
▶Base2, visualizar como binário 12
BibPriv 28
BibPub 29
binário
  indicador, 0b 135
  visualizar, ▶Base2 12
binomCdf( ) 14
binomPdf( ) 14
bloquear variáveis e grupos de variáveis 57
Bloquear, bloquear variável ou grupo de variáveis 57
Boolean operators
  and 6

## C

$\chi^2$ 2way 15
$\chi^2$ Cdf( ) 16
$\chi^2$ GOF 16
$\chi^2$ Pdf( ) 16
cadeia
  comprimento 30
  dimensão, dim( ) 30
cadeia de caracteres, char( ) 15
cadeia do formato, format( ) 38
cadeias
  acrescentar, & 127
  cadeia de caracteres, char( ) 15
  cadeia para expressão, expr( ) 35
  código de carácter, ord( ) 74
  deslocar, shift( ) 94
  direita, right( ) 87
  esquerda, left( ) 50
  expressão para cadeia, string( ) 102
  formatar 38
  formato, format( ) 38
  indirecta, # 131
  mid-string, mid( ) 63
  na, InString 47
  rodar, rotate( ) 88, 89
  utilizar para criar nomes de variáveis 141
caracteres
  cadeia, char( ) 15
  código numérico, ord( ) 74
Cdf( ) 36
ceiling, ceiling( ) 14
ceiling( ), ceiling 14
centralDiff( ) 15
char( ), cadeia de caracteres 15
ciclo, Cycle 26
ciclo, Loop 60
ClearAZ 16
ClrErr, apagar erro 17
códigos de aviso e mensagens 147
colAugment 17
colDim( ), dimensão da coluna da matriz 17
colNorm( ), norma da coluna da matriz 17
com, | 133
Comando Parar 102
Comando Text 106
combinações, nCr( ) 67
comentário, © 135
completeSquare( ), complete square 18
complexo
  conjugado, conj( ) 18
comprimento da cadeia 30
conj( ), conjugado complexo 18
constructMat( ), construir matriz 18
construir matriz, constructMat( ) 18
contar condicionalmente itens numa lista , countif( ) 23
contar dias entre datas, dbd( ) 26
contar itens numa lista, contar( ) 22

converter
▸Grad 45
▸Rad 82
copiar variável ou função, CopyVar 18
corrMat( ), matriz de correlação 19
cos $^{-1}$, arco-coseno 20
cos( ), co-seno 19
co-seno, cos( ) 19
cosh $^{-1}$( ), arco-coseno hiperbólico 21
cosh( ), co-seno hiperbólico 21
cot $^{-1}$( ), arco-cotangente 22
cot( ), co-tangente 21
co-tangente, cot( ) 21
coth $^{-1}$( ), arco-cotangente hiperbólico 22
coth( ), co-tangente hiperbólica 22
count( ), contar itens numa lista 22
countif( ), contar condicionalmente itens numa lista 23
cPolyRoots() 23
crossP( ), produto cruzado 23
csc $^{-1}$( ), co-secante inversa 24
csc( ), co-secante 24
csch $^{-1}$( ), co-secante hiperbólica inversa 24
csch( ), co-secante hiperbólica 24
CubicReg, regressão cúbica 25
Cycle, ciclo 26
▸Cylind, visualizar como vector cilíndrico 26

## D

d ( ), primeira derivada 128
dbd( ), dias entre datas 26
▸DD, visualizar como ângulo decimal 27
▸Decimal, visualizar resultado como decimal 27
decimal
visualizar ângulo, ▸DD 27
visualizar número inteiro, ▸Base10 13
definição, Lbl 50
definições do modo, getMode( ) 42
definições, obter actual 42
Definir 27
definir
modo, setMode( ) 93
Definir BibPriv 28
Definir BibPub 29
Definir, definir 27
DelVar, eliminar variável 29
delVoid( ), remover elementos nulos 29
densidade da probabilidade, normPdf( ) 70
densidade de probabilidade student- *t*, tPdf( ) 107
derivada
numérica, nDerivative( ) 68
derivadas
derivada numérica, nDeriv( ) 68, 69
derivada numérica, nDerivative( ) 68
primeira derivada, d ( ) 128
desbloquear variáveis e grupos de variáveis 113
Desbloquear, desbloquear variável ou grupo de variáveis 113
deslocar, shift( ) 94
desvio padrão, stdDev( ) 101, 113
det( ), determinante da matriz 30
diag( ), diagonal da matriz 30
dias entre datas, dbd( ) 26
diferente, 125
dim( ), dimensão 30
dimensão, dim( ) 30
direita, right( ) 87
Disp, visualizar dados 31
distribuição normal acumulada inversa (invNorm( ) 48
dividir, ÷ 121
divisão do número inteiro, intDiv( ) 47
▸DMS, visualizar como grau/minuto/segundo 31
dotP( ), produto do ponto 31

## E

**E**, expoente 131
e para uma potência, e^( ) 32, 35

e^( ), e para uma potência 32
eff ), converter taxa nominal para efectiva 32
eigVc( ), vector eigen 32
eigVl( ), valor próprio 33
elementos (nulos) vazios 136
elementos nulos 136
elementos nulos, remover 29
eliminar
elementos nulos da lista 29
variável, DelVar 29
else if, ElseIf 33
else, Else 45
ElseIf, else if 33
end
for, EndFor 38
função, EndFunc 40
if, EndIf 45
loop, EndLoop 60
programa, EndPrgm 78
end function, EndFunc 40
end if, EndIf 45
end loop, EndLoop 60
EndWhile, terminar enquanto 114
enquanto, While 114
EOS (Equation Operating System) 140
equações simultâneas, simult( ) 95
Equation Operating System (EOS) 140
erro de passagem, PassErr 75
erros e resolução de problemas
apagar erro, ClrErr 17
erro de passagem, PassErr 75
esquerda, left( ) 50
estatística
combinações, nCr( ) 67
desvio padrão, stdDev( ) 101, 113
estatística de uma variável, OneVar 73
factorial, ! 127
média, mean( ) 61
mediana, median( ) 61
norma aleatória, randNorm( ) 83
permutações, nPr( ) 71
resultados de duas variáveis, TwoVar 111
semente de número aleatório, RandSeed 84
variação, variance( ) 113
estatística de uma variável, OneVar 73
euler( ), Euler function 34
exclusão com operador "|" 133
Exit, sair 34
exp( ), e para uma potência 35
*Expoente e*
modelo para 2
expoente, **E** 131
expoentes
modelo para 1
expr( ), cadeia para expressão 35
ExpReg, refrsessão exponencial 35
expressões
cadeia para expressão, expr( ) 35

## F

factor, factor( ) 36
factor( ), factor 36
factorial, ! 127
factorização QR, QR 79
Fill, preencher matriz 37
FiveNumSummary 37
floor, floor( ) 37
floor( ), floor 37
For 38
For, for 38
for, For 38
forma de escalão-linha reduzida, rref( ) 90
forma de escalão-linha, ref( ) 85
format( ), cadeia do formato 38
fpart( ), parte da função 38
fracção própria, propFrac 79
fracções
modelo para 1
propFrac 79
fracções mistas, com propFrac(› com 79
freqTable( ) 39
frequência( ) 39
Func, função 40

Func, função do programa 40
função por ramos (2 ramos)
modelo para 2
função por ramos (N-ramos)
modelo para 2
funções
definidas pelo utilizador 27
função do programa, Func 40
parte, fpart( ) 38
funções de distribuição
binomCdf( ) 14
binomPdf( ) 14
$\chi^2$ 2way( ) 15
$\chi^2$ Cdf( ) 16
$\chi^2$ GOF( ) 16
$\chi^2$ Pdf( ) 16
Inv $\chi^2$ ( ) 48
invNorm( ) 48
invt( ) 48
normCdf( ) 70
normPdf( ) 70
poissCdf( ) 75
poissPdf( ) 75
tCdf( ) 106
tPdf( ) 107
funções definidas pelo utilizador 27
funções e programas definidos pelo utilizador 28, 29
funções e variáveis
a copiar 18
funções financeiras, tvmFV( ) 110
funções financeiras, tvmI( ) 110
funções financeiras, tvmN( ) 110
funções financeiras, tvmPmt( ) 110
funções financeiras, tvmPV( ) 110

## G

$^g$, gradianos 131
gcd( ), máximo divisor comum 41
geomCdf( ) 41
geomPdf( ) 41
getDenom( ), obter denominador 41
getLangInfo( ), obter/apresentar informações do idioma 42
getLockInfo( ), testar o estado de bloqueio da variável ou do grupo de variáveis 42
getMode( ), obter definições do modo 42
getNum( ), obter número 43
getType( ), get type of variable 43
getVarInfo( ), obter/apresentar informações das variáveis 44
Goto, ir para 44
▶, converter para ângulo de gradianos 45
grupos, bloquear e desbloquear 57, 113
grupos, testar estado de bloqueio 42
guardar
símbolo, → 134

## H

hexadecimal
indicador, 0h 135
visualizar, ▶Base16 14
hiperbólica
tangente, tanh( ) 105
hiperbólico
arco-coseno, $\cosh^{-1}$ ( ) 21
arco-seno, $\sinh^{-1}$ ( ) 97
arco-tangente, $\tanh^{-1}$ ( ) 105
co-seno, cosh( ) 21
seno, sinh( ) 97

## I

identity( ), matriz de identidade 45
idioma
obter informações do idioma 42
If, if 45
if, If 45
ifFn( ) 46
igual ou maior que, ≥ 126
igual ou menor que, ≤ 126
igual, = 125
imag( ), parte imaginária 46
implicação lógica dupla, ⇔ 127
implicação lógica, ⇒ 127, 138
indirecta, # 131
inString( ), na cadeia 47

int( ), número inteiro 47
intDiv( ), divisão do número inteiro 47
integral definido
modelo para 5
integral, 〉 128
interpolate( ), interpolate 48
Inv $\chi^2$ ( ) 48
inverso, ^-1 133
invF( ) 48
invNorm( ), distribuição normal acumulada inversa) 48
invt( ) 48
iPart( ), parte do número inteiro 49
ir para, Goto 44
irr( ), taxa de retorno interna
taxa de retorno interna, irr( ) 49
isPrime( ), teste da plica 49
isVoid( ), testar para nulo 49

## L

Lbl, definição 50
lcm, mínimo múltiplo comum 50
left( ), esquerda 50
limite máximo, limite máximo( ) 15, 23
LinRegBx, regressão linear 51
LinRegMx, regressão linear 52
LinRegtIntervals, regressão linear 53
LinRegtTest 54
linSolve() 55
list ▶mat( ), lista para matriz 55
lista para matriz, list ▶mat( ) 55
lista, contar condicionalmente itens numa 23
lista, contar itens em 22
ListaDelta() 29
listas
aumentar/concatenar, aumentar( ) 11
diferença, Δlist( ) 55
diferenças numa lista, Δlist( ) 55
elementos vazios em 136
lista para matriz, list ▶mat( ) 55
matriz para lista, mat ▶lista( ) 60
máximo, max( ) 61
mid-string, mid( ) 63
mínimo, min( ) 63
nova, newList( ) 68
ordenar ascendente, SortA 98
ordenar descendente, SortD 99
produto cruzado, crossP( ) 23
produto do ponto, dotP( ) 31
produto, product( ) 78
soma cumulativa, SomaCumulativa( ) 25
soma, sum( ) 102, 103
ln( ), logaritmo natural 55
LnReg, regressão logarítmica 56
local, Local 57
Local, variável local 57
Log
modelo para 2
logaritmo natural, ln( ) 55
logaritmos 55
LogisticD, regressão logística 59
Loop, ciclo 60
LU, decomposição inferior-superior da matriz 60

## M

maior que, > 126
mat ▶list( ), matriz para lista 60
matriz (1 × 2)
modelo para 3
matriz (2 × 1)
modelo para 4
matriz (2 × 2)
modelo para 3
matriz (m × n)
modelo para 4
matriz de correlação, corrMat( ) 19
matriz de identidade, identity( ) 45
matriz para lista, mat ▶list( ) 60
matrizes
adição da linha, rowAdd( ) 90
adição e multiplicação da linha, mRowAdd( ) 64
aleatória, randMat( ) 83
aumentar/concatenar, aumentar( ) 11
decomposição inferior-superior, LU 60
determinante, det( ) 30

diagonal, diag( ) 30
dimensão da coluna, colDim( ) 17
dimensão da linha, rowDim( ) 90
dimensão, dim( ) 30
factorização QR, QR 79
forma de escalão-linha reduzida, rref( ) 90
forma de escalão-linha, ref( ) 85
identidade, identity( ) 45
lista para matriz, list▶mat( ) 55
matriz para lista, mat▶list( ) 60
máximo, max( ) 61
mínimo, min( ) 63
norma da coluna, colNorm( ) 17
norma da linha, rowNorm( ) 90
nova, newMat( ) 68
operação da linha, mRow( ) 64
ponto adição, .+ 123
ponto divisão, .÷ 123
ponto multiplicação, .* 123
ponto potência, .^ 123
ponto subtracção, .− 123
preencher, Fill 37
produto, product( ) 78
soma cumulativa, SomaCumulativa( ) 25
soma, sum( ) 102, 103
submatriz, subMat( ) 102, 103
transpor, T 103
troca da linha~, rowSwap( ) 90
valor próprio, eigVl( ) 33
vector eigen, eigVc( ) 32
max( ), máximo 61
máximo divisor comum, gcd( ) 41
máximo, max( ) 61
mean( ), média 61
média, mean( ) 61
median( ), mediana 61
mediana, median( ) 61
MedMed, regressão da recta média-média 62
menor que, < 126
mid( ), mid-string 63
mid-string, mid( ) 63
min( ), mínimo 63
mínimo múltiplo comum, lcm 50
mínimo, min( ) 63
mirr( ), taxa de retorno interna modificada 64
mod( ), módulo 64
modelos
expoente 1
*Expoente e* 2
fracção 1
função por ramos (2 ramos) 2
função por ramos (N-ramos) 2
integral definido 5
Log 2
matriz (1 × 2) 3
matriz (2 × 1) 4
matriz (2 × 2) 3
matriz (m × n) 4
primeira derivada 5
produto (Π) 4
raiz de índice N 1
raiz quadrada 1
segunda derivada 5
sistema de equações (2 equações) 3
sistema de equações (N equações) 3
soma (Σ) 4
valor absoluto 3
modos
definir, setMode( ) 93
módulo, mod( ) 64
mRow( ), operação da linha da matriz 64
mRowAdd( ), adição e multiplicação da linha da matriz 64
multiplicar, * 121
MultReg 65
MultRegIntervals( ) 65
MultRegTests( ) 66

## N

na cadeia, inString( ) 47
nand, Operador booleano 67
nCr( ), combinações 67
nDerivative( ), derivada numérica 68
negação, introduzir números negativos 141
newList( ), nova lista 68

newMat( ), nova matriz 68
nfMax( ), função numérica máxima 68
nfMin( ), função numérica mínima 69
nInt( ), integral numérico 69
nom ), converter taxa efectiva para nominal 69
nor, Operador booleano 69
norma Frobenius, norma( ) 70
norma( ), norma Frobenius 70
normCdf( ) 70
normPdf( ) 70
not, Operador booleano 70
notação de gradianos, $^g$ 131
notação de grau/minuto/segundo 132
notação de graus, ° 132
notação de minutos, ' 132
notação de segundos, " 132
nova
  lista, newList( ) 68
  matriz, newMat( ) 68
nPr( ), permutações 71
npv( ), valor líquido actual 72
nSolve( ), solução numérica 72
nulo, testar para 49
numérica
  derivada, nDeriv( ) 68, 69
  solução, nSolve( ) 72
numérico
  integral, nInt( ) 69
número inteiro, int( ) 47

## O

obter
  denominador, getDenom( ) 41
  número, getNum( ) 43
obter/apresentar
  informações das variáveis, getVarInfo( ) 42, 44
OneVar, estatística de uma variável 73
operador da indirecta (#) 141
operador de limite "|" 133
operador de limite, ordem de avaliação 140
operadores
  ordem de avaliação 140
Operadores booleanos
  nand 67
  nor 69
  not 70
  ou 74
  ⇔ 127
  xou 115
  ⇒ 127, 138
ord( ), código de carácter numérico 74
ordenar
  ascendente, SortA 98
  descendente, SortD 99
ou (Booleano), or 74
ou, Operador booleano 74

## P

P▶Rx( ), rectangular x coordenada 74
P▶Ry( ), rectangular y coordenada 75
parte do número inteiro, iPart( ) 49
parte imaginária, imag( ) 46
PassErr, erro de passagem 75
Pdf( ) 39
percentagem, % 124
permutações, nPr( ) 71
piecewise( ) 75
poissCdf( ) 75
poissPdf( ) 75
▶Polar, visualizar como vector polar 76
polar
  coordenada, R▶P θ( ) 82
  coordenada, R▶Pr( ) 82
  visualizar vector, ▶Polar 76
polinómios
  aleatório, randPoly( ) 83
  avaliar, polyEval( ) 76
polyEval( ), avaliar polinómio 76
PolyRoots() 76
ponto
  adição, .+ 123
  divisão,.÷ 123
  multiplicação, .* 123

potência, .^ 123
produto, dotP( ) 31
subtracção, .− 123
potência de dez, 10^( ) 133
potência, ^ 122
PowerReg, regressão de potência 77
Prgm, definir programa 78
primeira derivada
modelo para 5
probabilidade da distribuição
normal, normCdf( ) 70
probabilidade da distribuição
student- *t* , tCdf( ) 106
product( ), produto 78
produto ( $\Pi$)
modelo para 4
produto cruzado, crossP( ) 23
produto, $\Pi$ ( ) 129
produto, product( ) 78
programar
definir programa, Prgm 78
erro de passagem, PassErr 75
visualizar dados, Disp 31
programas
definir biblioteca privada 28
definir biblioteca pública 29
programas e programação
apagar erro, ClrErr 17
terminar programa, EndPrgm 78
visualizar ecrã E/S, Disp 31
propFrac, fracção própria 79

## Q

QR, factorização QR 79
QuadReg, regressão quadrática 80
quando, when( ) 114
QuartReg, regressão quártica 81

## R

$^{r}$ , radianos 131
R▶P $\theta$( ), coordenada polar 82
R▶Pr( ), coordenada polar 82
RacionalAprox( ) 10
▶Rad, converter para ângulo de radianos 82
radianos, $^{r}$ 131
raiz de índice N
modelo para 1
raiz quadrada
modelo para 1
raiz quadrada, $\sqrt{}$( ) 99, 128
rand( ), número aleatório 82
randBin, número aleatório 83
randInt( ), número inteiro aleatório 83
randMat( ), matriz aleatória 83
randNorm( ), norma aleatória 83
randPoly( ), polinómio aleatório 83
randSamp( ) 83
RandSeed, semente de número aleatório 84
real, real( ) 84
real( ), real 84
recíproco, $^{-1}$ 133
▶Rect, visualizar como vector rectangular 84
rectangular x coordenada, P▶Rx( ) 74
rectangular y coordenada, P▶Ry( ) 75
ref( ), forma de escalão-linha 85
regressão cúbica, CubicReg 25
regressão da recta média-média, MedMed 62
regressão de potência, PowerReg 77
regressão exponencial, ExpReg 35
regressão linear, LinRegAx 52
regressão linear, LinRegBx 51, 53
regressão logarítmica, LnReg 56
regressão logística, LogisticD 59
regressão potencial, PowerReg 76, 86, 87, 106
regressão quadrática, QuadReg 80
regressão quártica, QuartReg 81
regressão sinusoidal, SinReg 98
regressões
cúbica, CubicReg 25
exponencial, ExpReg 35
logarítmica, LnReg 56
logística, Logística 59
MultReg 65
quadrática, QuadReg 80
quártica, QuartReg 81
recta média-média, MedMed 62

regressão de potência, PowerReg 77
regressão linear, LinRegAx 52
regressão linear, LinRegBx 51, 53
regressão potencial, PowerReg 76, 86, 87, 106
sinusoidal, SinReg 98
remain( ), resto 85
remover
elementos nulos da lista 29
Request 86
RequestStr 87
resposta (última), Ans 9
resto, remain( ) 85
resultados de duas variáveis, TwoVar 111
resultados, estatística 100
Return, return 87
return, Return 87
right, right( ) 18, 34, 48, 88, 114
right( ), direita 87
rk23( ), Runge Kutta function 88
rodar, rotate( ) 88, 89
rotate( ), rodar 88, 89
round( ), arredondar 89
rowAdd( ), adição da linha da matriz 90
rowDim( ), dimensão da linha da matriz 90
rowNorm( ), norma da linha da matriz 90
rowSwap( ), troca da linha da matriz 90
rref( ), forma de escalão-linha reduzida 90

## S

Σ Int( ) 130
Σ Prn( ) 130
sair, Exit 34
sec $^{-1}$ ( ), secante inversa 91
sec( ), secante 91
sech $^{-1}$ ( ), secante hiperbólica inversa 91
sech( ), secante hiperbólica 91
segunda derivada
modelo para 5
seno, sin( ) 96
seq( ), sequência 92
seqGen( ) 92
seqn( ) 93
SeqProd() 78
SeqSom() 103
sequence, seq( ) 92, 93
sequência, seq( ) 92
setMode( ), definir modo 93
shift( ), deslocar 94
sign( ), sinal 95
simult( ), equações simultâneas 95
sin $^{-1}$ ( ), arco-seno 96
sin( ), seno 96
sinal, sign( ) 95
sinh $^{-1}$ ( ), arco-seno hiperbólico 97
sinh( ), seno hiperbólico 97
SinReg, regressão sinusoidal 98
sistema de equações (2 equações)
modelo para 3
sistema de equações (N equações)
modelo para 3
soma ( Σ)
modelo para 4
soma cumulativa, SomaCumulativa( ) 25
soma de pagamentos principais 130
soma dos pagamentos de juros 130
soma, Σ ( ) 129
soma, sum( ) 102
SomaCumulativa( ), soma cumulativa 25
SortA, ordenar ascendente 98
SortD, ordenar descendente 99
▸Sphere, visualizar como vector esférico 99
sqrt( ), raiz quadrada 99
stat.results 100
stat.values 101
stdDevPop( ), desvio padrão da população 101
stdDevSamp( ), desvio padrão da amostra 101
string( ), expressão para cadeia 102
strings
right, right( ) 18, 34, 48, 88, 114

subMat( ), submatriz 102, 103
submatriz, subMat( ) 102, 103
substituição com operador "|" 133
subtrair, − 120
sum( ), soma 102
sumIf( ) 103

## T

T, transpor 103
tabela de amortização, amortTbl( ) 6, 12
tan $^{-1}$( ), arco-tangente 104
tan( ), tangente 104
tangente, tan( ) 104
tanh $^{-1}$( ), arco-tangente hiperbólico 105
tanh( ), tangente hiperbólica 105
taxa de câmbio média, avgRC( ) 12
taxa de retorno interna modificada, mirr( ) 64
taxa efectiva, eff( ) 32
taxa nominal, nom( ) 69
tCdf( ), probabilidade da distribuição student -*t* 106
terminar
  enquanto, EndWhile 114
terminar enquanto, EndWhile 114
Test_2S, Teste F de 2 amostras 40
testar para nulo, isVoid( ) 49
teste da plica, isPrime( ) 49
Teste F de 2 amostras 40
*teste t* , tTest 109
Teste t de regressões lineares múltiplas 66
tInterval_2Samp, -intervalo de confiança *t* de duas amostras 107
tInterval, *t* intervalo de confiança 106
tPdf( ), densidade de probabilidade student -*t* 107
transpor, T 103
tTest_2Samp, teste *t* de duas amostras 109
tTest, teste *t* 109
tvmFV( ) 110
tvmI( ) 110
tvmN( ) 110
tvmPmt( ) 110
tvmPV( ) 110
TwoVar, resultados de duas variáveis 111

## U

unitV( ), vector da unidade 112

## V

valor absoluto
  modelo para 3
valor líquido actual, npv( ) 72
valor próprio, eigVl( ) 33
valor temporal do dinheiro, juro 110
valor temporal do dinheiro, montante do pagamento 110
valor temporal do dinheiro, número de pagamentos 110
valor temporal do dinheiro, valor actual 110
valor temporal do dinheiro, Valor futuro 110
valores dos resultados, estatística 101
variação, variance( ) 113
variáveis
  apagar todas as letras individuais 16
  eliminar, DelVar 29
  local, Local 57
variáveis, bloquear e desbloquear 42, 57, 113
variável
  criar nome a partir de uma cadeia de caracteres 141
variável e funções
  a copiar 18
variável local, Local 57
varPop( ) 113
varSamp( ), variação da amostra 113
vector eigen, eigVc( ) 32
vector unitário, unitV( ) 112
vectores
  produto cruzado, crossP( ) 23

produto do ponto, dotP( ) 31
unidade, unitV( ) 112
visualizar vector cilíndrico, ▶Cylind 26
visualizar como
ângulo decimal, ▶DD 27
binário, ▶Base2 12
grau/minuto/segundo, ▶DMS 31
hexadecimal, ▶Base16 14
número inteiro decimal, ▶Base10 13
vector , ▶Polar 76
vector cilíndrico, ▶Cylind 26
vector esférico, ▶Sphere 99
vector rectangular, ▶Rect 84
visualizar dados, Disp 31
visualizar grau/minuto/segundo, ▶DMS 31
visualizar vector cilíndrico, ▶Cylind 26
visualizar vector esférico, ▶Sphere 99
visualizar vector rectangular, ▶Rect 84

## W

warnCodes( ), Warning codes 114
when( ), quando 114
While, enquanto 114

## X

x 2 , quadrado 122
XNOR 127
xou, Booleano exclusivo ou 115

## Z

zInterval_1Prop, intervalo de confiança *z* de uma proporção 116
zInterval_2Prop, intervalo de confiança *z* de duas proporções 116
zInterval_2Samp, intervalo de confiança *z* de duas amostras 117
zInterval, *z* intervalo de confiança 115
zTest 117
zTest_1Prop, teste z de uma proporção 118
zTest_2Prop, teste z de duas proporções 118
zTest_2Samp, teste *z* de duas amostras 119