

TI-Nspire™ CX CAS

Guide de référence

Informations importantes

Sauf disposition contraire expressément formulée dans la licence qui accompagne un programme, Texas Instruments n'émet aucune garantie expresse ou implicite, y compris sans s'y limiter, toute garantie implicite de valeur marchande et d'adéquation à un usage particulier, concernant les programmes ou la documentation, ceux-ci étant fournis « tels quels » sans autre recours. En aucun cas, Texas Instruments ne saurait être tenue responsable de dommages spéciaux, collatéraux, fortuits ou indirects en relation avec, ou imputables à l'achat ou à l'utilisation de ce matériel. La seule responsabilité exclusive de Texas Instruments, indépendamment de la forme d'action, ne saurait dépasser le prix fixé dans la licence pour ce programme. Par ailleurs, la responsabilité de Texas Instruments ne saurait être engagée pour quelque réclamation que ce soit en rapport avec l'utilisation desdits matériels par toute autre tierce partie.

© 2026 Texas Instruments Incorporated

Les produits peuvent varier légèrement des images fournies.

Sommaire

Modèles d'expression	1
Liste alphabétique	8
A	8
B	18
C	23
D	50
E	65
F	75
G	86
I	98
L	107
M	125
N	134
O	144
P	147
Q	157
R	160
S	176
T	204
U	221
V	221
W	223
X	225
Z	226
Symboles	235
TI-Nspire™ CX II - Commandes graphiques	264
Programmation en mode graphique	264
Écran de représentation graphique	264
Vue et paramètres par défaut	265
Messages d'erreur de l'écran graphique	266
Commandes non valides dans le mode graphique	266
C	268
D	269
F	273
G	275
P	276
P	278
U	280

Éléments vides	281
Raccourcis de saisie d'expressions mathématiques	283
Hiérarchie de l'EOS™ (Equation Operating System)	285
Fonctions de programmation TI-Basic sur TI-Nspire CX II	287
Auto-indentation dans l'Éditeur de programmes	287
Messages d'erreur améliorés pour TI-Basic	287
Constantes et valeurs	290
Codes et messages d'erreur	291
Codes et messages d'avertissement	300
Informations générales	302
Index	303

Modèles d'expression

Les modèles d'expression facilitent la saisie d'expressions mathématiques en notation standard. Lorsque vous utilisez un modèle, celui-ci s'affiche sur la ligne de saisie, les petits carrés correspondants aux éléments que vous pouvez saisir. Un curseur identifie l'élément que vous pouvez saisir.

Utilisez les touches fléchées ou appuyez sur **tab** pour déplacer le curseur sur chaque élément, puis tapez la valeur ou l'expression correspondant à chaque élément. Appuyez sur **enter** ou **ctrl enter** pour calculer l'expression.

Modèle Fraction

Touches **ctrl** **÷**



Remarque : Voir aussi / (division), page 237.

Exemple :

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

Modèle Exposant

Touche **^**



Remarque : Tapez la première valeur, appuyez sur **^**, puis entrez l'exposant. Pour ramener le curseur sur la ligne de base, appuyez sur la flèche droite (►).

Remarque : Voir aussi ^ (puissance), page 238.

Exemple :

$$2^3 \quad 8$$

Modèle Racine carrée

Touches **ctrl** **x²**



Remarque : Voir aussi $\sqrt{\quad}$ (racine carrée), page 249.

Exemple :

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

Modèle Racine n-ième

Touches ctrl ^



Remarque : Voir aussi **root()**, page 172.

Exemple :

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

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

Modèle e Exposant

Touches e^x



La base du logarithme népérien e élevée à une puissance

Remarque : Voir aussi **e^()**, page 65.

Exemple :

$$e^1 \quad e$$

$$e^1. \quad 2.71828182846$$

Modèle Logarithme

Touches ctrl 10^x



Calcule le logarithme selon la base spécifiée. Par défaut la base est 10, dans ce cas ne spécifiez pas de base.

Remarque : Voir aussi **log()**, page 120.

Exemple :

$$\log_{10}(2.) \quad 0.5$$

Modèle Fonction définie par morceaux (2 morceaux)

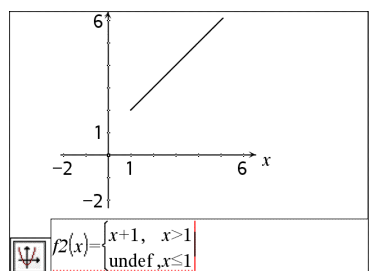
Catalogue > |\alpha|^\alpha



Permet de créer des expressions et des conditions pour une fonction définie par deux morceaux.- Pour ajouter un morceau supplémentaire, cliquez dans le modèle et appliquez-le de nouveau.

Remarque : Voir aussi **piecewise()**, page 148.

Exemple :

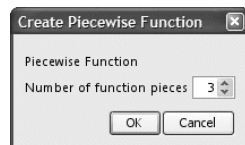


Modèle Fonction définie par morceaux (n morceaux)

Catalogue >



Permet de créer des expressions et des conditions pour une fonction définie par n -morceaux. Le système vous invite à définir n .



Exemple :

Voir l'exemple donné pour le modèle Fonction définie par morceaux (2 morceaux).

Remarque : Voir aussi `piecewise()`, page 148.

Modèle Système de 2 équations

Catalogue >



Crée une système de deux équations . Pour ajouter une nouvelle ligne à un système existant, cliquez dans le modèle et appliquez-le de nouveau.

Remarque : Voir aussi `system()`, page 203.

Exemple :

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

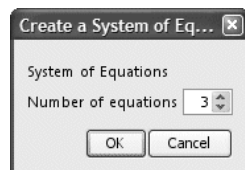
$$\text{solve} \left(\begin{cases} y=x^2-2 \\ x+2 \cdot y=-1 \end{cases}, x, y \right) \quad x = \frac{-3}{2} \text{ and } y = \frac{1}{4} \text{ or } x=1 \text{ and } y=-1$$

Modèle Système de n équations

Catalogue >



Permet de créer un système de N linéaires. Le système vous invite à définir N .



Exemple :

Voir l'exemple donné pour le modèle Système de 2 équations.

Remarque : Voir aussi `system()`, page 203.

Modèle Valeur absolue

Catalogue >



Exemple :

Modèle Valeur absolue

Catalogue > 

Remarque : Voir aussi **abs()**, page 8.

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

Modèle dd°mm'ss.ss''

Catalogue > 



Exemple :

Permet d'entrer des angles en utilisant le format **dd°mm'ss.ss''**, où **dd** correspond au nombre de degrés décimaux, **mm** au nombre de minutes et **ss.ss** au nombre de secondes.

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

Modèle Matrice (2 x 2)

Catalogue > 



Exemple :

Crée une matrice de type 2 x 2.

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

Modèle Matrice (1 x 2)

Catalogue > 



Exemple :

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

Modèle Matrice (2 x 1)

Catalogue > 



Exemple :

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

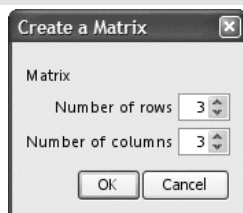
Modèle Matrice (m x n)

Catalogue > 

Le modèle s'affiche après que vous ayez saisi le nombre de lignes et de colonnes.

Exemple :

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



Remarque : si vous créez une matrice dotée de nombreuses lignes et colonnes, son affichage peut prendre quelques minutes.

Modèle Somme (Σ)

$$\sum_{i=0}^{} (i)$$

Exemple :

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

Remarque : voir aussi $\Sigma()$ (**sumSeq**), page 250.

Modèle Produit (Π)

$$\prod_{i=0}^{} (i)$$

Exemple :

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

Remarque : Voir aussi $\Pi()$ (**prodSeq**), page 250.

Modèle Dérivée première

$$\frac{d}{dx} (x)$$

Par exemple :

Vous pouvez utiliser ce modèle pour calculer la dérivée première en un point.

Modèle Dérivée première

Catalogue > 

Remarque : voir aussi **d()** (dérivée), page 246.

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

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

Modèle Dérivée seconde

Catalogue > 

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

Par exemple :

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

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

Vous pouvez utiliser ce modèle pour calculer la dérivée seconde en un point.

Remarque : voir aussi **d()** (dérivée), page 246.

Modèle Dérivée n-ième

Catalogue > 

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

Exemple :

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

Vous pouvez utiliser ce modèle pour calculer la dérivée n -ième.

Remarque : Voir aussi **d()** (dérivée), page 246.

Modèle Intégrale définie

Catalogue > 

$$\int_a^b \square dx$$

Exemple :

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

Remarque : voir aussi **∫()** **integral()**, page 247.

Modèle Intégrale indéfinie

Catalogue > 

$$\int \square dx$$

Exemple :

Modèle Intégrale indéfinie

Catalogue > 

Remarque : Voir aussi `∫()` `integral()`, page 247.

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

Modèle Limite

Catalogue > 

$$\lim_{x \rightarrow 0} \left(\frac{1}{x} \right)$$

Exemple :

$$\lim_{x \rightarrow 5} (2 \cdot x + 3) \quad 13$$

Utilisez `-` ou `(-)` pour définir la limite à gauche et la touche `+` pour la limite à droite.

Remarque : Voir aussi `limit()`, page 109.

Liste alphabétique

Les éléments dont le nom n'est pas alphabétique (comme +, !, et >) apparaissent à la fin de cette section, à partir de la ici. Sauf indication contraire, tous les exemples fournis dans cette section ont été réalisés en mode de réinitialisation par défaut et toutes les variables sont considérées comme indéfinies.

A

abs()

Catalogue >

abs(ExprI)⇒expression

abs(ListeI)⇒liste

abs(MatriceI)⇒matrice

Donne la valeur absolue de l'argument.

Remarque : Voir aussi **Modèle Valeur absolue**, page 3.

Si l'argument est un nombre complexe, donne le module de ce nombre.

Remarque : toutes les variables non affectées sont considérées comme réelles.

$\left \left\{ \frac{\pi}{2}, \frac{\pi}{3} \right\} \right $	$\left\{ \frac{\pi}{2}, \frac{\pi}{3} \right\}$
$ 2-3 \cdot i $	$\sqrt{13}$
$ z $	$ z $
$ x+y \cdot i $	$\sqrt{x^2+y^2}$

amortTbl()

Catalogue >

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

Fonction d'amortissement affichant une matrice représentant un tableau d'amortissement pour un ensemble d'arguments TVM.

NPmt est le nombre de versements à inclure au tableau. Le tableau commence avec le premier versement.

N, I, PV, Pmt, FV, PpY, CpY et PmtAt sont décrits dans le tableau des arguments TVM, page 218.

- Si vous omettez Pmt, il prend par défaut la valeur $Pmt=tvmpmt(N,I,PV,FV,PpY,CpY,PmtAt)$.

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

- Si vous omettez FV , il prend par défaut la valeur $FV=0$.
- Les valeurs par défaut pour PpY , CpY et $PmtAt$ sont les mêmes que pour les fonctions TVM.

$valArrondi$ spécifie le nombre de décimales pour arrondissement. Valeur par défaut=2.

Les colonnes dans la matrice résultante apparaissent dans l'ordre suivant :
Numéro de versement, montant versé pour les intérêts, montant versé pour le capital et solde.

Le solde affiché à la ligne n correspond au solde après le versement n .

Vous pouvez utiliser la matrice de sortie pour insérer les valeurs des autres fonctions d'amortissement $\Sigma Int()$ et $\Sigma Prn()$, page 251 et **bal()**, page 18.

and

$Expr\ booléenne1\ and\ Expr\ booléenne2 \Rightarrow Expression\ booléenne$

$$\begin{array}{ccc} x \geq 3 \text{ and } x \geq 4 & & x \geq 4 \\ \{x \geq 3, x \leq 0\} \text{ and } \{x \geq 4, x \leq 2\} & & \{x \geq 4, x \leq 2\} \end{array}$$

$Liste\ booléenne1\ et\ Liste\ booléenne2 \Rightarrow Liste\ booléenne$

$Matrice\ booléenne1\ and\ Matrice\ booléenne2 \Rightarrow Matrice\ booléenne$

Donne true (vrai) ou false (faux) ou une forme simplifiée de l'entrée initiale.

$Entier1\ and\ Entier2 \Rightarrow entier$

En mode base Hex :

$$0h7AC36 \text{ and } 0h3D5F \quad 0h2C16$$

Important : utilisez le chiffre zéro et pas la lettre O.

En mode base Bin :

$$0b100101 \text{ and } 0b100 \quad 0b100$$

Compare les représentations binaires de deux entiers réels en appliquant un **and** bit à bit. En interne, les deux entiers sont convertis en nombres binaires 64 bits signés. Lorsque les bits comparés correspondent, le résultat est 1 si dans les deux cas il s'agit d'un bit 1 ; dans les autres cas, le résultat est 0. La valeur donnée représente le résultat des bits et elle est affichée selon le mode Base utilisé.

Les entiers de tout type de base sont admis. Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h. Tout entier sans préfixe est considéré comme un nombre en écriture décimale (base 10).

Si vous entrez un nombre dont le codage binaire signé dépasse 64 bits, il est ramené à l'aide d'une congruence dans la plage appropriée.

En mode base Dec :

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

Remarque : une entrée binaire peut comporter jusqu'à 64 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 16 chiffres.

angle()

angle(*Expr1*) $\Rightarrow$ *expression*

Donne l'argument de l'expression passée en paramètre, celle-ci étant interprétée comme un nombre complexe.

Remarque : toutes les variables non affectées sont considérées comme réelles.

En mode Angle en degrés :

angle(0+2 <i>i</i>)	90
----------------------	----

En mode Angle en grades :

angle(0+3 <i>i</i>)	100
----------------------	-----

En mode Angle en radians :

angle(1+i)	$\frac{\pi}{4}$
angle(<i>z</i>)	$\frac{-\pi \cdot (\text{sign}(z) - 1)}{2}$
angle(<i>x</i> + <i>i</i> · <i>y</i>)	$\frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right)$

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

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

angle(Liste1)⇒*liste*

angle(Matrice1)⇒*matrice*

Donne la liste ou la matrice des arguments des éléments de *Liste1* ou *Matrice1*, où chaque élément est interprété comme un nombre complexe représentant un point de coordonnée rectangulaire à deux dimensions.

ANOVA

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

Effectue une analyse unidirectionnelle de variance pour comparer les moyennes de deux à vingt populations. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Indicateur=0 pour Données, *Indicateur*=1 pour Stats

Variable de sortie	Description
stat.F	Valeur de F statistique
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degré de liberté des groupes
stat.SS	Somme des carrés des groupes
stat.MS	Moyenne des carrés des groupes
stat.dfError	Degré de liberté des erreurs
stat.SSError	Somme des carrés des erreurs
stat.MSError	Moyenne des carrés des erreurs
stat.sp	Écart-type du groupe
stat.xbarlist	Moyenne des entrées des listes

Variable de sortie	Description
stat.CLowerList	Limites inférieures des intervalles de confiance de 95 % pour la moyenne de chaque liste d'entrée
stat.CUpperList	Limites supérieures des intervalles de confiance de 95 % pour la moyenne de chaque liste d'entrée

ANOVA2way

Catalogue > 

ANOVA2way *Liste1, Liste2[,...[, Liste10]]*
[, NivLign]

Effectue une analyse de variance à deux facteurs pour comparer les moyennes de deux à dix populations. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

NivLign=0 pour Bloc

NivLign=2,3,..., *Len*-1, pour 2 facteurs, où
Len=length(*Liste1*)=length(*Liste2*) = ... =
length(*Liste10*) et *Len* / *NivLign* ∈ {2,3,...}

Sorties : Bloc

Variable de sortie	Description
stat.F	F statistique du facteur de colonne
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degré de liberté du facteur de colonne
stat.SS	Somme des carrés du facteur de colonne
stat.MS	Moyenne des carrés du facteur de colonne
stat.FBlock	F statistique du facteur
stat.PValBlock	Plus petite probabilité permettant de rejeter l'hypothèse nulle
stat.dfBlock	Degré de liberté du facteur
stat.SSBlock	Somme des carrés du facteur
stat.MSBlock	Moyenne des carrés du facteur
stat.dfError	Degré de liberté des erreurs
stat.SSError	Somme des carrés des erreurs
stat.MSError	Moyenne des carrés des erreurs

Variable de sortie	Description
stat.s	Écart-type de l'erreur

Sorties FACTEUR DE COLONNE

Variable de sortie	Description
stat.Fcol	F statistique du facteur de colonne
stat.PValCol	Valeur de probabilité du facteur de colonne
stat.dfCol	Degré de liberté du facteur de colonne
stat.SSCol	Somme des carrés du facteur de colonne
stat.MSCol	Moyenne des carrés du facteur de colonne

Sorties FACTEUR DE LIGNE

Variable de sortie	Description
stat.Frow	F statistique du facteur de ligne
stat.PValRow	Valeur de probabilité du facteur de ligne
stat.dfRow	Degré de liberté du facteur de ligne
stat.SSRow	Somme des carrés du facteur de ligne
stat.MSRow	Moyenne des carrés du facteur de ligne

Sorties INTERACTION

Variable de sortie	Description
stat.FInteract	F statistique de l'interaction
stat.PValInteract	Valeur de probabilité de l'interaction
stat.dfInteract	Degré de liberté de l'interaction
stat.SSInteract	Somme des carrés de l'interaction
stat.MSInteract	Moyenne des carrés de l'interaction

Sorties ERREUR

Variable de sortie	Description
stat.dfError	Degré de liberté des erreurs
stat.SSError	Somme des carrés des erreurs
stat.MSError	Moyenne des carrés des erreurs
s	Écart-type de l'erreur

Ans	Touches ctrl (-)	
Ans ⇒ <i>valeur</i>	56	56
Donne le résultat de la dernière expression calculée.	56+4	60
	60+4	64

approx()	Catalogue > 	
approx(<i>Expr1</i>) ⇒ <i>expression</i>	$\text{approx}\left(\frac{1}{3}\right)$	0.333333
Donne une approximation décimale de l'argument sous forme d'expression, dans la mesure du possible, indépendamment du mode Auto ou Approché utilisé.	$\text{approx}\left(\left\{\frac{1}{3}, \frac{1}{9}\right\}\right)$	{ 0.333333, 0.111111 }
	$\text{approx}(\{\sin(\pi), \cos(\pi)\})$	{ 0., -1. }
Ceci est équivalent à la saisie de l'argument suivie d'une pression sur ctrl enter .	$\text{approx}([\sqrt{2} \quad \sqrt{3}])$	[1.41421 1.73205]
	$\text{approx}\left(\left[\frac{1}{3} \quad \frac{1}{9}\right]\right)$	[0.333333 0.111111]
approx(<i>Liste1</i>) ⇒ <i>liste</i>	$\text{approx}(\{\sin(\pi), \cos(\pi)\})$	{ 0., -1. }
approx(<i>Matrice1</i>) ⇒ <i>matrice</i>	$\text{approx}([\sqrt{2} \quad \sqrt{3}])$	[1.41421 1.73205]

Donne une liste ou une *matrice* d'éléments pour lesquels une approximation décimale a été calculée, dans la mesure du possible.

►approxFraction()	Catalogue > 
<i>Expr</i> ► <i>approxFraction</i> (<i>[tol]</i>)⇒ <i>expression</i>	$\frac{1}{2} + \frac{1}{3} + \tan(\pi)$ 0.833333
<i>Liste</i> ► <i>approxFraction</i> (<i>[tol]</i>)⇒ <i>liste</i>	0.8333333333333333 ► <i>approxFraction</i> (5.E-14) $\frac{5}{6}$
<i>Matrice</i> ► <i>approxFraction</i> (<i>[tol]</i>)⇒ <i>matrice</i>	$\{\pi, 1.5\}$ ► <i>approxFraction</i> (5.E-14) $\left\{ \frac{5419351}{1725033}, \frac{3}{2} \right\}$
Donne l'entrée sous forme de fraction en utilisant une tolérance <i>tol</i> . Si <i>tol</i> est omis, la tolérance 5.E-14 est utilisée.	
Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant @> approxFraction (...).	

approxRational()	Catalogue > 
approxRational (<i>Expr</i> [, <i>tol</i>])⇒ <i>expression</i>	<i>approxRational</i> (0.333,5·10 ⁻⁵) $\frac{333}{1000}$
approxRational (<i>Liste</i> [, <i>tol</i>])⇒ <i>liste</i>	<i>approxRational</i> ({ 0.2,0.33,4.125 },5.E-14) $\left\{ \frac{1}{5}, \frac{33}{100}, \frac{33}{8} \right\}$
approxRational (<i>Matrice</i> [, <i>tol</i>])⇒ <i>matrice</i>	
Donne l'argument sous forme de fraction en utilisant une tolérance <i>tol</i> . Si <i>tol</i> est omis, la tolérance 5.E-14 est utilisée.	

arccos()	Voir cos ⁻¹ (), page 35.
-----------------	-------------------------------------

arccosh()	Voir cosh ⁻¹ (), page 37.
------------------	--------------------------------------

arccot()	Voir cot ⁻¹ (), page 38.
-----------------	-------------------------------------

arccoth()	Voir coth ⁻¹ (), page 38.
------------------	--------------------------------------

arccsc()Voir $\csc^{-1}()$, page 41.**arccsch()**Voir $\operatorname{csch}^{-1}()$, page 42.**arcLen()**Catalogue > **arcLen**(*Expr1*, *Var*, *Début*, *Fin*)
 $\Rightarrow$ *expression*

Donne la longueur de l'arc de la courbe définie par *Expr1* entre les points d'abscisses *Début* et *Fin* en fonction de la variable *Var*.

$$\begin{array}{l} \text{arcLen}(\cos(x), x, 0, \pi) \\ \text{arcLen}(f(x), x, a, b) \end{array} \quad \begin{array}{l} 3.8202 \\ \int_a^b \sqrt{\left(\frac{d}{dx}(f(x))\right)^2 + 1} dx \end{array}$$

La longueur d'arc est calculée sous forme d'intégrale en supposant la définition du mode fonction.

arcLen(*Liste1*, *Var*, *Début*, *Fin*) $\Rightarrow$ *liste*

Donne la liste des longueurs d'arc de chaque élément de *Liste1* entre les points d'abscisses *Début* et *Fin* en fonction de la variable *Var*.

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

arcsec()Voir $\sec^{-1}()$, page 177.**arcsech()**Voir $\operatorname{sech}^{-1}()$, page 177.**arcsin()**Voir $\sin^{-1}()$, page 188.**arcsinh()**Voir $\sinh^{-1}()$, page 189.

augment()

Catalogue > **augment(Liste1, Liste2)⇒liste** $\text{augment}(\{1, -3, 2\}, \{5, 4\}) \quad \{1, -3, 2, 5, 4\}$

Donne une nouvelle liste obtenue en plaçant les éléments de *Liste2* à la suite de ceux de *Liste1*.

augment(Matrice1, Matrice2)⇒matrice

Donne une nouvelle matrice obtenue en ajoutant les lignes/colonnes de la *Matrice2* à celles de la *Matrice1*. Les matrices doivent avoir le même nombre de lignes et *Matrice2* est ajoutée à *Matrice1* via la création de nouvelles colonnes. *Matrice1* et *Matrice2* ne sont pas modifiées.

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

avgRC()

Catalogue > **avgRC(Expr1, Var [=Valeur] [, Incrément])⇒expression** $\text{avgRC}(f(x), x, h) \quad \frac{f(x+h) - f(x)}{h}$ **avgRC(Expr1, Var [=Valeur] [, Liste1])⇒liste** $\text{avgRC}(\sin(x), x, h) | x=2 \quad \frac{\sin(h+2) - \sin(2)}{h}$ **avgRC(Liste1, Var [=Valeur] [, Incrément])⇒liste** $\text{avgRC}(x^2 - x + 2, x) \quad 2 \cdot (x - 0.4995)$ $\text{avgRC}(x^2 - x + 2, x, 0.1) \quad 2 \cdot (x - 0.45)$ **avgRC(Matrice1, Var [=Valeur] [, Incrément])⇒matrice** $\text{avgRC}(x^2 - x + 2, x, 3) \quad 2 \cdot (x + 1)$

Donne le taux d'accroissement moyen (quotient à différence antérieure) de l'expression.

Expr1 peut être un nom de fonction défini par l'utilisateur (voir **Func**).

Quand la *valeur* est spécifiée, celle-ci prévaut sur toute affectation de variable ou substitution précédente de type « | » pour la variable.

Incrément correspond à la valeur de l'incrément. Si *Incrément* n'est pas spécifié, il est fixé par défaut à 0,001.

Notez que la fonction comparable **nDeriv()** utilise le quotient à différence symétrique.

Notez que la fonction comparable **centralDiff()** utilise le quotient à différence centrée.

B

bal()

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

bal(*NPmt*,*tblAmortissement*)⇒*valeur*

Fonction d'amortissement destinée à calculer le solde après versement d'un montant spécifique.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* et *PmtAt* sont décrits dans le tableau des arguments TVM, xxxx.

NPmt indique le numéro de versement après lequel vous souhaitez que les données soient calculées.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* et *PmtAt* sont décrits dans le tableau des arguments TVM, xxxx.

- Si vous omettez *Pmt*, il prend par défaut la valeur *Pmt=tvmpmt* (*N*,*I*,*PV*,*FV*,*PpY*,*CpY*,*PmtAt*).
- Si vous omettez *FV*, il prend par défaut la valeur *FV=0*.
- Les valeurs par défaut pour *PpY*, *CpY* et *PmtAt* sont les mêmes que pour les fonctions TVM.

bal(5,6,5.75,5000,,12,12)	833.11
---------------------------	--------

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

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

bal(4,tbl)	1674.27
------------	---------

valArrondi spécifie le nombre de décimales pour arrondissement. Valeur par défaut=2.

bal(*NPmt*,*tblAmortissement*) calcule le solde après le numéro de paiement *NPmt*, sur la base du tableau d'amortissement *tblAmortissement*. L'argument *tblAmortissement* doit être une matrice au format décrit à **tblAmortissement()**, xxxx.

Remarque : voir également $\Sigma\text{Int}()$ et $\Sigma\text{Prn}()$, xxxx.

►Base2

Entier1 ►Base2⇒*entier*

256►Base2

0b100000000

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>Base2.

0h1F►Base2

0b11111

Convertit *Entier1* en nombre binaire. Les nombres binaires et les nombres hexadécimaux présentent toujours respectivement un préfixe, 0b ou 0h. Zéro et pas la lettre O, suivi de b ou h.

0b *nombreBinaire*

0h *nombreHexadécimal*

Une entrée binaire peut comporter jusqu'à 64 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 16 chiffres.

Si *Entier1* est entré sans préfixe, il est considéré comme un nombre en écriture décimale (base 10). Le résultat est affiché sous forme binaire, indépendamment du mode Base utilisé.

Les nombres négatifs sont affichés sous forme de complément à deux. Par exemple,

-1 s'affiche sous la forme

0hFFFFFFFFFFFFFF en mode Base Hex

0b111...111 (64 1's) en mode Base
Binaire

-2^{63} s'affiche sous la forme

0h8000000000000000 en mode Base
Hex

0b100...000 (63 zéros) en mode Base
Binaire

Si vous entrez un nombre dont le codage binaire signé est hors de la plage des 64 bits, il est ramené à l'aide d'une congruence dans la plage appropriée. Consultez les exemples suivants de valeurs hors plage.

2^{63} devient -2^{63} et s'affiche sous la forme

0h8000000000000000 en mode Base
Hex

0b100...000 (63 zéros) en mode Base
Binaire

2^{64} devient 0 et s'affiche sous la forme

0h0 en mode Base Hex

0b0 en mode Base Binaire

$-2^{63} - 1$ devient $2^{63} - 1$ et s'affiche sous la forme

0h7FFFFFFFFFFFFFFF en mode Base Hex

0b111...111 (64 1) en mode Base
Binaire

Entier1 ►Base10⇒entier

0b10011►Base10	19
0h1F►Base10	31

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>Base10.

Convertit *Entier1* en un nombre décimal (base 10). Toute entrée binaire ou hexadécimale doit avoir respectivement un préfixe 0b ou 0h.

0b *nombreBinaire*

0h *nombreHexadécimal*

Zéro et pas la lettre O, suivi de b ou h.

Une entrée binaire peut comporter jusqu'à 64 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 8 chiffres.

Sans préfixe, *Entier1* est considéré comme décimal. Le résultat est affiché en base décimale, quel que soit le mode Base en cours d'utilisation.

Entier1 ►Base16⇒*entier*

256►Base16

0h100

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>Base16.

0b111100001111►Base16

0hF0F

Convertit *Entier1* en nombre hexadécimal. Les nombres binaires et les nombres hexadécimaux présentent toujours respectivement un préfixe, 0b ou 0h.

0b *nombreBinaire*

0h *nombreHexadécimal*

Zéro et pas la lettre O, suivi de b ou h.

Une entrée binaire peut comporter jusqu'à 64 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 16 chiffres.

Si *Entier1* est entré sans préfixe, il est considéré comme un nombre en écriture décimale (base 10). Le résultat est affiché sous forme hexadécimal, indépendamment du mode Base utilisé.

Si vous entrez un nombre dont le codage binaire signé dépasse 64 bits, il est ramené à l'aide d'une congruence dans la plage appropriée. Pour de plus amples informations, voir ►Base2, page 19.

binomCdf()

binomCdf(*n,p*)⇒*liste*

binomCdf

(*n,p,lowBound,upBound*)⇒*nombre* si les bornes *lowBound* et *upBound* sont des nombres, *liste* si les bornes *lowBound* et *upBound* sont des listes

binomCdf(*n,p,upBound*) pour $P(0 \leq X \leq upBound)$ ⇒ *nombre* si la borne *upBound* est un nombre, *liste* si la borne *upBound* est une liste

Calcule la probabilité cumulée d'une variable suivant une loi binomiale de paramètres n = nombre d'essais et p = probabilité de réussite à chaque essai.

Pour $P(X \leq upBound)$, définissez la borne *lowBound*=0

binomPdf()

binomPdf(*n,p*)⇒*liste*

binomPdf(*n,p,ValX*)⇒*nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

Calcule la probabilité de *ValX* pour la loi binomiale discrète avec un nombre n d'essais et la probabilité p de réussite pour chaque essai.

ceiling()Catalogue > **ceiling**(*Expr1*) $\Rightarrow$ *entier* $\text{ceiling}(.456)$ 1.Donne le plus petit entier $\geq$ à l'argument.

L'argument peut être un nombre réel ou un nombre complexe.

Remarque : Voir aussi **floor()**.**ceiling**(*Liste1*) $\Rightarrow$ *liste* $\text{ceiling}(\{-3.1, 1, 2.5\})$ $\{-3., 1, 3.\}$ **ceiling**(*Matrice1*) $\Rightarrow$ *matrice* $\text{ceiling}\left(\begin{bmatrix} 0 & -3.2 \cdot i \\ 1.3 & 4 \end{bmatrix}\right)$ $\begin{bmatrix} 0 & -3. \cdot i \\ 2. & 4 \end{bmatrix}$

Donne la liste ou la matrice de plus petites valeurs supérieures ou égales à chaque élément.

centralDiff()Catalogue > **centralDiff**(*Expr1*, *Var* [=Valeur] [, *Pas*]) $\Rightarrow$ *expression* $\text{centralDiff}(\cos(x), x, h)$
$$\frac{-\left(\cos(x-h) - \cos(x+h)\right)}{2 \cdot h}$$
centralDiff(*Expr1*, *Var* [, *Pas*]) | *Var*=Valeur $\Rightarrow$ *expression* $\lim_{h \rightarrow 0} \left(\text{centralDiff}(\cos(x), x, h) \right)$ $-\sin(x)$ **centralDiff**(*Expr1*, *Var* [=Valeur] [, *Liste*]) $\Rightarrow$ *liste* $\text{centralDiff}(x^3, x, 0.01)$
 $3. \cdot (x^2 + 0.000033)$ **centralDiff**(*Liste1*, *Var* [=Valeur] [, *Incrément*]) $\Rightarrow$ *liste* $\text{centralDiff}(\cos(x), x) | x = \frac{\pi}{2}$ -1.**centralDiff**(*Matrice1*, *Var* [=Valeur] [, *Incrément*]) $\Rightarrow$ *matrice* $\text{centralDiff}(x^2, x, \{0.01, 0.1\})$
 $\{2. \cdot x, 2. \cdot x\}$

Affiche la dérivée numérique en utilisant la formule du quotient à différence centrée.

Quand la *valeur* est spécifiée, celle-ci prévaut sur toute affectation de variable ou substitution précédente de type « | » pour la variable.*Incrément* correspond à la valeur de l'incrément. Si *Incrément* n'est pas spécifié, il est fixé par défaut à 0,001.

Si vous utilisez *Liste1* ou *Matrice1*,
l'opération s'étend aux valeurs de la liste
ou aux éléments de la matrice.

Remarque : voir aussi **avgRC()** et **d()**.

cFactor()

cFactor(*Expr1*[, *Var*]) $\Rightarrow$ *expression*

cFactor(*Liste1*[, *Var*]) $\Rightarrow$ *liste*

cFactor(*Matrice1*[, *Var*]) $\Rightarrow$ *matrice*

cFactor(*Expr1*) factorise *Expr1* dans C en
fonction de toutes ses variables et sur un
dénominateur commun.

La factorisation de Expr1 décompose
l'expression en autant de facteurs
rationnels linéaires que possible même si
cela introduit de nouveaux nombres non
réels. Cette alternative peut s'avérer
utile pour factoriser l'expression en
fonction de plusieurs variables.

cFactor(*Expr1*, *Var*) factorise *Expr1* dans
C en fonction de la variable *Var*.

La factorisation de Expr1 décompose
l'expression en autant de facteurs
possible qui sont linéaires dans *Var*, avec
peut-être des constantes non réelles,
même si cela introduit des constantes
irrationnelles ou des sous-expressions
qui sont irrationnelles dans d'autres
variables.

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

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

Les facteurs et leurs termes sont triés, *Var* étant la variable principale. Les mêmes puissances de *Var* sont regroupées dans chaque facteur. Incluez *Var* si la factorisation ne doit s'effectuer que par rapport à cette variable et si vous acceptez les expressions irrationnelles dans les autres variables pour augmenter la factorisation par rapport à *Var*. Une factorisation incidente peut se produire par rapport aux autres variables.

Avec le réglage Auto du mode **Auto ou Approché (Approximate)** l'utilisation de *Var* permet également une approximation avec des coefficients en virgule flottante dans le cadre de laquelle les coefficients irrationnels ne peuvent pas être exprimés explicitement suivant les termes des fonctions intégrées. Même en présence d'une seule variable, l'utilisation de *Var* peut contribuer à une factorisation plus complète.

Remarque : voir aussi **factor()**.

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

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

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

$$(x - 0.964673) \cdot (x + 0.611649) \cdot (x + 2.12543) \cdot (x^2 + 1)$$

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

char()

char(Entier) ⇒ caractère

Donne le caractère dont le code dans le jeu de caractères de l'unité nomade est *Entier*. La plage valide pour *Entier* est comprise entre 0 et 65535.

$\text{char}(38)$	"&"
$\text{char}(65)$	"A"

charPoly()

charPoly
(matriceCarrée, Var) ⇒ expression
polynomiale

charPoly
(matriceCarrée, Expr) ⇒ expression
polynomiale

charPoly

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

(*matriceCarrée1*,*matriceCarrée2*) $\Rightarrow$
expression polynomiale

Donne le polynôme caractéristique de *matriceCarrée*. Le polynôme caractéristique d'une matrice $n \times n$ A , désigné par $p_A(\lambda)$, est le polynôme défini par

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

où I désigne la matrice identité $n \times n$.

matriceCarrée1 et *matriceCarrée2* doivent avoir les mêmes dimensions.

χ^2 2way

χ^2 2way *MatriceObservée*

chi22way *MatriceObservée*

Effectue un test χ^2 d'association sur le tableau 2×2 de valeurs dans la matrice observée *MatriceObservée*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Pour plus d'informations concernant les éléments vides dans une matrice, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat. χ^2	Stats Khi^2 : $\text{sum}(\text{observée} - \text{attendue})^2 / \text{attendue}$
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degré de liberté des statistiques khi^2
stat.ExpMat	Matrice du tableau de valeurs élémentaires attendues, acceptant l'hypothèse nulle
stat.CompMat	Matrice des contributions statistiques khi^2 élémentaires

χ^2 Cdf(*lowBound*,*upBound*,*dl*) $\Rightarrow$ nombre si les bornes *lowBound* et *upBound* sont des nombres, liste si les bornes *lowBound* et *upBound* sont des listes

chi2Cdf(*lowBound*,*upBound*,*dl*) $\Rightarrow$ nombre si les bornes *lowBound* et *upBound* sont des nombres, liste si les bornes *lowBound* et *upBound* sont des listes

Calcule la probabilité qu'une variable suivant une loi χ^2 à *dl* degrés de liberté prenne une valeur entre les bornes *lowBound* et *upBound*.

Pour $P(X \leq \text{upBound})$, définissez la borne *lowBound*=0.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

χ^2 GOF *ListeObservée*,*ListeAttendue*,*df*

chi2GOF *ListeObservée*,*ListeAttendue*,*df*

Effectue un test pour s'assurer que les données des échantillons sont issues d'une population conforme à la loi spécifiée. *ListeObservée* est une liste de comptage qui doit contenir des entiers. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat. χ^2	Stats Khi^2 : $\text{sum}(\text{observée} - \text{attendue})^2 / \text{attendue}$
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degré de liberté des statistiques khi^2

Variable de sortie	Description
stat.ComplList	Contributions statistiques khi ² élémentaires

χ^2 Pdf()

Catalogue > 

χ^2 Pdf(*ValX*,*dl*) $\Rightarrow$ nombre si *ValX* est un nombre, liste si *XVal* est une liste

chi2Pdf(*ValX*,*dl*) $\Rightarrow$ nombre si *ValX* est un nombre, liste si *ValX* est une liste

Calcule la probabilité qu'une variable suivant une loi χ^2 à *dl* degrés de liberté prenne une valeur *ValX* spécifiée.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

ClearAZ

Catalogue > 

ClearAZ

Supprime toutes les variables à une lettre de l'activité courante.

Si une ou plusieurs variables sont verrouillées, cette commande affiche un message d'erreur et ne supprime que les variables non verrouillées. Voir **unLock**, page 221.

$5 \rightarrow b$	5
<i>b</i>	5
ClearAZ.	Done
<i>b</i>	<i>b</i>

ClrErr

Catalogue > 

ClrErr

Efface le statut d'erreur et règle la variable système *errCode* sur zéro.

Pour obtenir un exemple de **ClrErr**, reportez-vous à l'exemple 2 de la commande **Try**, page 214.

L'instruction **Else** du bloc **Try...Else...EndTry** doit utiliser **EffErr** ou **PassErr**. Si vous comptez rectifier ou ignorer l'erreur, sélectionnez **EffErr**. Si vous ne savez pas comment traiter l'erreur, sélectionnez **PassErr** pour la transférer au traitement d'erreurs suivant. S'il n'y a plus d'autre traitement d'erreurs **Try...Else...EndTry**, la boîte de dialogue Erreur s'affiche normalement.

Remarque : voir également **PassErr**, page 148 et **Try**, page 214.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

colAugment()

colAugment(*Matrice1*,
Matrice2) $\Rightarrow$ matrice

Donne une nouvelle matrice obtenue en ajoutant les lignes/colonnes de la *Matrice2* à celles de la *Matrice1*. Les matrices doivent avoir le même nombre de colonnes et *Matrice2* est ajoutée à *Matrice1* via la création de nouvelles lignes. *Matrice1* et *Matrice2* ne sont pas modifiées.

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

colDim()

colDim(*Matrice*) $\Rightarrow$ expression

Donne le nombre de colonnes de la matrice *Matrice*.

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

Remarque : voir aussi **rowDim()**.

colNorm(Matrice) $\Rightarrow$ expression

Donne le maximum des sommes des valeurs absolues des éléments situés dans chaque colonne de la matrice *Matrice*.

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

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

Remarque : les éléments non définis de matrice ne sont pas autorisés. Voir aussi **rowNorm**().

comDenom(Expr1[,Var]) $\Rightarrow$ expression**comDenom**(Liste1[,Var]) $\Rightarrow$ liste**comDenom**(Matrice1[,Var]) $\Rightarrow$ matrice

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

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

comDenom(Expr1) donne le rapport réduit d'un numérateur entièrement développé sur un dénominateur entièrement développement.

comDenom(Expr1,Var) donne le rapport réduit d'un numérateur et d'un dénominateur développé par rapport à *Var*. Les termes et leurs facteurs sont triés, *Var* étant la variable principale. Les mêmes puissances de *Var* sont regroupées. Une factorisation incidente des coefficients regroupés peut se produire. L'utilisation de *Var* permet de gagner du temps, de la mémoire et de l'espace sur l'écran tout en facilitant la lecture de l'expression. Les opérations suivantes basées sur le résultat obtenu sont également plus rapides et moins consommatrices de mémoire.

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

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

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

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

Si *Var* n'intervient pas dans *Expr1*, **comDenom**(*Expr1*,*Var*) donne le rapport réduit d'un numérateur non développé sur un dénominateur non développé. Ce type de résultat offre généralement un gain de temps, de mémoire et d'espace sur l'écran. La factorisation partielle du résultat contribue également à accélérer les opérations suivantes basées sur le résultat et à utiliser moins de mémoire.

Même en l'absence de tout dénominateur, la fonction **comden** permet d'obtenir rapidement une factorisation partielle si la fonction **factor**() est trop lente ou si elle utilise trop de mémoire.

Conseil : entrez cette définition de la fonction **comden**() et utilisez-la régulièrement comme solution alternative à **comDenom**() et à **factor**().

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

Done

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

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

completeSquare ()

completeSquare(*ExprOuÉqn*,
Var) $\Rightarrow$ *expression ou équation*

completeSquare(*ExprOuÉqn*,
Var^{*Puissance*}) $\Rightarrow$ *expression ou équation*

completeSquare(*ExprOuÉqn*, *Var1*,
Var2 [...]) $\Rightarrow$ *expression ou équation*

completeSquare(*ExprOuÉqn*, *Var1*,
Var2 [...]) $\Rightarrow$ *expression ou équation*

Convertit une expression polynomiale du second degré de type $a\cdot x^2+b\cdot x+c$ en $a\cdot (x-h)^2+k$.

- ou -

Convertit une équation du second degré de type $x^2+b\cdot x+c=d$ en $a\cdot (x-h)^2=k$.

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

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

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

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

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

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

Le premier argument doit être une expression ou une équation du second degré en notation standard par rapport au deuxième argument.

Le deuxième argument doit être un terme à une seule variable ou un terme à une seule variable élevé à une puissance rationnelle (par exemple x , y^2 ou $z^{1/3}$).

Le troisième et le quatrième tentent de compléter le carré en fonction des variables $Var1$, $Var2$ [...]).

conj()

conj(*Expr1*) $\Rightarrow$ *expression*

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

conj(*Liste1*) $\Rightarrow$ *liste*

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

conj(*Matrice1*) $\Rightarrow$ *matrice*

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

Donne le conjugué de l'argument.

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

Remarque : toutes les variables non affectées sont considérées comme réelles.

constructMat()

constructMat
(*Expr*, *Var1*, *Var2*, *nbreLignes*,
nbreColonnes) $\Rightarrow$ *matrice*

Donne une matrice basée sur les arguments.

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

Expr est une expression composée de variables *Var1* et *Var2*. Les éléments de la matrice résultante sont formés en évaluant *Expr* pour chaque valeur incrémentée de *Var1* et de *Var2*.

Var1 est incrémentée automatiquement de 1 à *nbreLignes*. Dans chaque ligne, *Var2* est incrémentée de 1 à *nbreColonnes*.

CopyVar *Var1*, *Var2*

Define $a(x) = \frac{1}{x}$	Done
-----------------------------	------

CopyVar *Var1*, *Var2*.

CopyVar *Var1*, *Var2* copie la valeur de la variable *Var1* dans la variable *Var2* et crée *Var2*, si nécessaire. La variable *Var1* doit avoir une valeur.

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

Si *Var1* correspond au nom d'une fonction existante définie par l'utilisateur, copie la définition de cette fonction dans la fonction *Var2*. La fonction *Var1* doit être définie.

Var1 doit être conforme aux règles de dénomination des variables ou correspondre à une expression d'indirection correspondant à un nom de variable conforme à ces règles.

CopyVar *Var1*, *Var2*. copie tous les membres du groupe de variables *Var1*. dans le groupe *Var2* et crée le groupe *Var2*. si nécessaire.

Var1. doit être le nom d'un groupe de variables existant, comme *stat*, *le résultat nn* ou les variables créées à l'aide de la fonction **LibShortcut()**. Si *Var2*. existe déjà, cette commande remplace tous les membres communs aux deux groupes et ajoute ceux qui n'existent pas. Si un ou plusieurs membres de *Var2*. sont verrouillés, tous les membres de *Var2*. restent inchangés.

<i>aa.a</i> :=45	45																
<i>aa.b</i> :=6.78	6.78																
CopyVar <i>aa</i> ., <i>bb</i> ..	Done																
getVarInfo()	<table border="1"> <tr> <td><i>aa.a</i></td> <td>"NUM"</td> <td>"[]"</td> <td>0</td> </tr> <tr> <td><i>aa.b</i></td> <td>"NUM"</td> <td>"[]"</td> <td>0</td> </tr> <tr> <td><i>bb.a</i></td> <td>"NUM"</td> <td>"[]"</td> <td>0</td> </tr> <tr> <td><i>bb.b</i></td> <td>"NUM"</td> <td>"[]"</td> <td>0</td> </tr> </table>	<i>aa.a</i>	"NUM"	"[]"	0	<i>aa.b</i>	"NUM"	"[]"	0	<i>bb.a</i>	"NUM"	"[]"	0	<i>bb.b</i>	"NUM"	"[]"	0
<i>aa.a</i>	"NUM"	"[]"	0														
<i>aa.b</i>	"NUM"	"[]"	0														
<i>bb.a</i>	"NUM"	"[]"	0														
<i>bb.b</i>	"NUM"	"[]"	0														

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

Calcule la matrice de corrélation de la matrice augmentée [*Liste1* *Liste2* ... *List20*].

cos*Expr* **cos**

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

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant `@>cos`.

Exprime *Expr* en cosinus. Il s'agit d'un opérateur de conversion utilisé pour l'affichage. Cet opérateur ne peut être utilisé qu'à la fin d'une ligne.

cos réduit toutes les puissances modulo $\sin(\dots) 1 - \cos(\dots)^2$ de sorte que les puissances de $\cos(\dots)$ restantes ont des exposants dans (0, 2). Le résultat ne contient donc pas $\sin(\dots)$ si et seulement si $\sin(\dots)$ dans l'expression donnée s'applique uniquement aux puissances paires.

Remarque : L'opérateur de conversion n'est pas autorisé en mode Angle Degré ou Grade. Avant de l'utiliser, assurez-vous d'avoir défini le mode Angle sur Radian et de l'absence de références explicites à des angles en degrés ou en grades dans *Expr*.

cos()

Touche 

cos(Expr1)⇒expression

En mode Angle en degrés :

cos(Liste1)⇒liste

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

cos(Expr1) calcule le cosinus de l'argument et l'affiche sous forme d'expression.

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

cos(Liste1) donne la liste des cosinus des éléments de *Liste1*.

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

Remarque : l'argument est interprété comme la mesure d'un angle en degrés, en grades ou en radians, suivant le mode angulaire en cours d'utilisation. Vous pouvez utiliser °, G ou R pour préciser l'unité employée temporairement pour le calcul.

En mode Angle en grades :

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

En mode Angle en radians :

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

cos(matriceCarréeI)⇒matriceCarrée

Calcule le cosinus de la matrice *matriceCarréeI*. Ce calcul est différent du calcul du cosinus de chaque élément.

Si une fonction scalaire f(A) opère sur *matriceCarréeI* (A), le résultat est calculé par l'algorithme suivant :

Calcul des valeurs propres (λ_i) et des vecteurs propres (V_i) de A.

matriceCarréeI doit être diagonalisable et ne peut pas présenter de variables symboliques sans valeur affectée.

Formation des matrices :

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

Alors $A = X B X^{-1}$ et $f(A) = X f(B) X^{-1}$. Par exemple, $\cos(A) = X \cos(B) X^{-1}$ où :

$\cos(B) =$

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

Tous les calculs sont exécutés en virgule flottante.

En mode Angle en radians :

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

cos⁻¹(ExprI)⇒expression

En mode Angle en degrés :

cos⁻¹(ListeI)⇒liste

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

En mode Angle en grades :

cos⁻¹()**Touche** 

cos⁻¹(Expr1) donne l'arc cosinus de Expr1 et l'affiche sous forme d'expression.

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

cos⁻¹(Liste1) donne la liste des arcs cosinus de chaque élément de Liste1.

Remarque : donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

En mode Angle en radians :

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

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arccos (...)**.

cos⁻¹(matriceCarrée1)⇒matriceCarrée

Donne l'arc cosinus de *matriceCarrée1*. Ce calcul est différent du calcul de l'arc cosinus de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

En mode Angle en radians et en mode Format complexe Rectangulaire :

$$\cos^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} 1.73485+0.064606 \cdot i & -1.49086+2.10514 \\ -0.725533+1.51594 \cdot i & 0.623491+0.77836 \cdot i \\ -2.08316+2.63205 \cdot i & 1.79018-1.27182 \cdot i \end{bmatrix}$$

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

cosh()**Catalogue** > 

cosh(Expr1)⇒expression

En mode Angle en degrés :

cosh(Liste1)⇒liste

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

cosh(Expr1) donne le cosinus hyperbolique de l'argument et l'affiche sous forme d'expression.

cosh(Liste1) donne la liste des cosinus hyperboliques de chaque élément de Liste1.

cosh(matriceCarrée1)⇒matriceCarrée

En mode Angle en radians :

Donne le cosinus hyperbolique de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du cosinus hyperbolique de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

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

cosh()Catalogue > 

matriceCarrée1 doit être diagonalisable.
Le résultat contient toujours des chiffres
en virgule flottante.

cosh⁻¹()Catalogue > **cosh⁻¹(Expr1) ⇒ expression**

cosh ⁻¹ (1)	0
------------------------	---

cosh⁻¹(List1) ⇒ liste

cosh ⁻¹ {1,2,1,3}	{0,1.37286,cosh ⁻¹ (3)}
------------------------------	------------------------------------

cosh⁻¹(Expr1) donne l'argument cosinus
hyperbolique de l'argument et l'affiche
sous forme d'expression.

cosh⁻¹(List1) donne la liste des
arguments cosinus hyperboliques de
chaque élément de *List1*.

Remarque : vous pouvez insérer cette
fonction à partir du clavier en entrant
arccosh (...).

cosh⁻¹
(*matriceCarrée1*) ⇒ *matriceCarrée*

Donne l'argument cosinus hyperbolique
de la matrice *matriceCarrée1*. Ce calcul
est différent du calcul de l'argument
cosinus hyperbolique de chaque
élément. Pour plus d'informations sur la
méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable.
Le résultat contient toujours des chiffres
en virgule flottante.

En mode Angle en radians et en mode Format
complexe Rectangulaire :

cosh ⁻¹ $\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$	$\begin{bmatrix} 2.52503+1.73485 \cdot i & -0.009241-1.49086i \\ 0.486969-0.725533 \cdot i & 1.66262+0.623491i \\ -0.322354-2.08316 \cdot i & 1.26707+1.79018i \end{bmatrix}$
---	---

Pour afficher le résultat entier, appuyez sur **▲**,
puis utilisez les touches **◀** et **▶** pour déplacer
le curseur.

cot()Touche **cot(Expr1) ⇒ expression**

En mode Angle en degrés :

cot(List1) ⇒ liste

cot(45)	1
---------	---

Affiche la cotangente de *Expr1* ou
retourne la liste des cotangentes des
éléments de *List1*.

En mode Angle en grades :

cot(50)	1
---------	---

cot()Touche 

Remarque : l'argument est interprété comme la mesure d'un angle en degrés, en grades ou en radians, suivant le mode angulaire en cours d'utilisation. Vous pouvez utiliser °, G ou R pour préciser l'unité employée temporairement pour le calcul.

En mode Angle en radians :

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

cot⁻¹()Touche **cot⁻¹(Expr1) ⇒ expression**

En mode Angle en degrés :

$$\cot^{-1}(1) \quad 45.$$

cot⁻¹(Liste1) ⇒ liste

Donne l'arc cotangente de *Expr1* ou affiche une liste comportant les arcs cotangentes de chaque élément de *Liste1*.

En mode Angle en grades :

$$\cot^{-1}(1) \quad 50.$$

Remarque : donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arccot (...)**.

En mode Angle en radians :

$$\cot^{-1}(1) \quad \frac{\pi}{4}$$

coth()Catalogue > **coth(Expr1) ⇒ expression**

$$\coth(1.2) \quad 1.19954$$

coth(Liste1) ⇒ liste

$$\coth(\{1, 3, 2\}) \quad \left\{ \frac{1}{\tanh(1)}, 1.00333 \right\}$$

Affiche la cotangente hyperbolique de *Expr1* ou donne la liste des cotangentes hyperboliques des éléments de *Liste1*.

coth⁻¹()Catalogue > **coth⁻¹(Expr1) ⇒ expression**

$$\coth^{-1}(3.5) \quad 0.293893$$

coth⁻¹(Liste1) ⇒ liste

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

Affiche l'argument cotangente hyperbolique de *Expr1* ou donne la liste comportant les arguments cotangentes hyperboliques des éléments de *Liste1*.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arccoth (...)**.

count()

Catalogue >

count(*Valeur1* ou *Liste1* [, *Valeur2* ou *Liste2* [, ...]]) ⇒ *valeur*

Affiche le nombre total des éléments dans les arguments qui s'évaluent à des valeurs numériques.

Un argument peut être une expression, une valeur, une liste ou une matrice. Vous pouvez mélanger les types de données et utiliser des arguments de dimensions différentes.

Pour une liste, une matrice ou une plage de cellules, chaque élément est évalué afin de déterminer s'il doit être inclus dans le comptage.

Dans l'application Tableur & listes, vous pouvez utiliser une plage de cellules à la place de n'importe quel argument.

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

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

Dans le dernier exemple, seuls 1/2 et 3+4*i sont comptabilisés. Les autres arguments, dans la mesure où x est indéfini, ne correspondent pas à des valeurs numériques.

countif()

Catalogue >

countif(*Liste*, *Critère*) ⇒ *valeur*

Affiche le nombre total d'éléments dans *Liste* qui répondent au *critère* spécifié.

Le *critère* peut être :

- Une valeur, une expression ou une chaîne. Par exemple, **3** compte uniquement les éléments dans *Liste* qui ont pour valeur 3.
- Une expression booléenne contenant le symbole ? comme paramètre substituable à tout élément. Par

countif({1,3,"abc",undef,3,1},3)	2
----------------------------------	---

Compte le nombre d'éléments égaux à 3.

countif({"abc","def","abc",3},"def")	1
--------------------------------------	---

Compte le nombre d'éléments égaux à "def."

countif({x^2,x^-1,1,x,x^2},x)	1
-------------------------------	---

countif()Catalogue > 

exemple, $?<5$ ne compte que les éléments dans *Liste* qui sont inférieurs à 5.

Dans l'application Tableur & listes, vous pouvez utiliser une plage de cellules à la place de *Liste*.

Les éléments vides de la liste sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

Remarque : voir également **sumif()**, page 202 et **frequency()**, page 84.

Compte le nombre d'éléments égaux à x ; cet exemple part du principe que la variable x est indéfinie.

$$\text{countif}(\{1, 3, 5, 7, 9\}, ? < 5) \quad 2$$

Compte 1 et 3.

$$\text{countif}(\{1, 3, 5, 7, 9\}, 2 < ? < 8) \quad 3$$

Compte 3, 5 et 7.

$$\text{countif}(\{1, 3, 5, 7, 9\}, ? < 4 \text{ or } ? > 6) \quad 4$$

Compte 1, 3, 7 et 9.

cPolyRoots()Catalogue > 

cPolyRoots(*Poly*, *Var*) $\Rightarrow$ *liste*

cPolyRoots(*ListeCoeff*) $\Rightarrow$ *liste*

La première syntaxe, **cPolyRoots**(*Poly*, *Var*), affiche une liste de racines complexes du polynôme *Poly* pour la variable *Var*.

Poly doit être un polynôme d'une seule variable.

La deuxième syntaxe, **cPolyRoots**(*ListeCoeff*), affiche une liste des racines complexes pour les coefficients de la liste *ListeCoeff*.

Remarque : voir aussi **polyRoots()**, page 153.

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

$$\text{cPolyRoots}(y^3 + 1, y) \quad \left\{-1, \frac{1}{2} - \frac{\sqrt{3}}{2}i, \frac{1}{2} + \frac{\sqrt{3}}{2}i\right\}$$

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

$$\text{cPolyRoots}(\{1, 2, 1\}) \quad \{-1, -1\}$$

crossP()Catalogue > 

crossP(*Liste1*, *Liste2*) $\Rightarrow$ *liste*

Donne le produit vectoriel de *Liste1* et de *Liste2* et l'affiche sous forme de liste.

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

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

Liste1 et *Liste2* doivent être de même dimension et cette dimension doit être égale à 2 ou 3.

crossP(*Vecteur1*, *Vecteur2*) ⇒ *vecteur*

Donne le vecteur ligne ou le vecteur colonne (en fonction des arguments) obtenu en calculant le produit vectoriel de *Vecteur1* et *Vecteur2*.

Ces deux vecteurs, *Vecteur1* et *Vecteur2*, doivent être de même type (ligne ou colonne) et de même dimension, cette dimension devant être égale à 2 ou 3.

$$\begin{array}{l} \text{crossP}\left(\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}\right) \quad \begin{bmatrix} -3 & 6 & -3 \end{bmatrix} \\ \text{crossP}\left(\begin{bmatrix} 1 & 2 \end{bmatrix}, \begin{bmatrix} 3 & 4 \end{bmatrix}\right) \quad \begin{bmatrix} 0 & 0 & -2 \end{bmatrix} \end{array}$$

csc()

Touche 

csc(*Expr1*) ⇒ *expression*

En mode Angle en degrés :

csc(*Liste1*) ⇒ *liste*

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

Affiche la cosécante de *Expr1* ou donne une liste comportant les cosécantes de tous les éléments de *Liste1*.

En mode Angle en grades :

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

En mode Angle en radians :

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

csc⁻¹()Touche 

csc⁻¹(*Expr1*) ⇒ *expression*

En mode Angle en degrés :

csc⁻¹(*Liste1*) ⇒ *liste*

$$\text{csc}^{-1}(1) \quad 90.$$

Affiche l'angle dont la cosécante correspond à *Expr1* ou donne la liste des arcs cosécante de chaque élément de *Liste1*.

En mode Angle en grades :

$$\text{csc}^{-1}(1) \quad 100.$$

csc⁻¹()Touche 

Remarque : donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arccsc (...)**.

En mode Angle en radians :

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

csch()Catalogue > 

csch(Expr1) ⇒ *expression*

$$\text{csch}(3) \quad \frac{1}{\sinh(3)}$$

csch(Liste1) ⇒ *liste*

Affiche la cosécante hyperbolique de *Expr1* ou donne la liste des cosécantes hyperboliques de tous les éléments de *Liste1*.

$$\text{csch}\left(\left\{1, 2, 1, 4\right\}\right) \quad \left\{\frac{1}{\sinh(1)}, 0.248641, \frac{1}{\sinh(4)}\right\}$$

csch⁻¹()Catalogue > 

csch⁻¹(Expr1) ⇒ *expression*

$$\text{csch}^{-1}(1) \quad \sinh^{-1}(1)$$

csch⁻¹(Liste1) ⇒ *liste*

Affiche l'argument cosécante hyperbolique de *Expr1* ou donne la liste des arguments cosécantes hyperboliques de tous les éléments de *Liste1*.

$$\text{csch}^{-1}\left(\left\{1, 2, 1, 3\right\}\right) \quad \left\{\sinh^{-1}(1), 0.459815, \sinh^{-1}\left(\frac{1}{3}\right)\right\}$$

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arccsch (...)**.

cSolve()Catalogue > 

cSolve(Équation, Var) ⇒ *Expression booléenne*

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

cSolve(Équation, Var=Init) ⇒ *expression booléenne*

$$\text{solve}(x^3=1, x) \quad x=1$$

cSolve(Inéquation, Var) ⇒ *Expression booléenne*

Résout dans C une équation ou une inéquation pour *Var*. L'objectif est de trouver toutes les solutions réelles et non réelles possibles. Même si *Équation* est à coefficients réels, **cSolve()** autorise les résultats non réels en mode Format complexe : Réel.

cSolve() définit temporairement le domaine sur complexe pendant la résolution, même si le domaine courant est réel. Dans le domaine complexe, les puissances fractionnaires possédant un dénominateur impair utilisent la branche principale plutôt que la branche réelle. Par conséquent, les solutions de **solve()** pour les équations impliquant de telles puissances fractionnaires n'appartiennent pas nécessairement à un sous-ensemble de celles de **cSolve()**.

cSolve() commence la résolution en utilisant les méthodes symboliques exactes. Excepté en mode **Exact**, **cSolve()** utilise aussi une factorisation itérative approchée des polynômes complexes, si nécessaire.

Remarque : voir aussi **cZeros()**, **solve()** et **zeros()**.

cSolve(*Équation1* and *Équation2* [and...],
VarOuInit1, *VarOuInit2* [, ...])
⇒ *expression booléenne*

cSolve(*SystèmeÉqu*, *VarOuInit1*,
VarOuInit2 [, ...])
⇒ *expression booléenne*

Donne les solutions complexes possibles d'un système d'équations algébriques, où chaque *VarOuInit* définit une variable dont vous cherchez la valeur.

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

En mode Afficher chiffres, Fixe 2 :

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

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

Vous pouvez également spécifier une condition initiale pour les variables. Chaque *VarOuInit* doit utiliser le format suivant :

variable

– ou –

variable = nombre réel ou non réel

Par exemple, x est autorisé, de même que $x=3+i$.

Si toutes les équations sont polynomiales et si vous NE spécifiez PAS de condition initiale, **cSolve()** utilise la méthode d'élimination lexicale Gröbner/Buchberger pour tenter de trouver **toutes** les solutions complexes.

Les solutions complexes peuvent combiner des solutions réelles et des solutions non réelles, comme illustré dans l'exemple ci-contre.

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

Pour afficher le résultat entier, appuyez sur $\blacktriangle$, puis utilisez les touches $\blacktriangleleft$ et $\blacktriangleright$ pour déplacer le curseur.

Les systèmes d'équations polynomiales peuvent comporter des variables supplémentaires auxquelles aucune valeur n'est affectée, mais qui représentent des valeurs numériques données pouvant s'y substituer par la suite.

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

Vous pouvez également utiliser des variables qui n'apparaissent pas dans les équations. Ces solutions montrent comment des solutions peuvent dépendre de paramètres arbitraires de type ck , où k est un suffixe entier compris entre 1 et 255.

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

Pour les systèmes d'équations polynomiales, le temps de calcul et l'utilisation de la mémoire peuvent considérablement varier en fonction de l'ordre dans lequel les variables inconnues sont spécifiées. Si votre choix initial ne vous satisfait pas pour ces raisons, vous pouvez modifier l'ordre des variables dans les équations et/ou la liste des variables *VarOulnit*.

Si vous choisissez de ne pas spécifier de condition et s'il l'une des équations n'est pas polynomiale en l'une des variables, mais que toutes les équations sont linéaires par rapport à toutes les variables de solution inconnues, **cSolve()** utilise l'élimination gaussienne pour tenter de trouver toutes les solutions.

Si un système d'équations n'est pas polynomial par rapport à toutes ses variables ni linéaire par rapport aux inconnues, **cSolve()** cherche au moins une solution en utilisant la méthode itérative approchée. Pour cela, le nombre d'inconnues doit être égal au nombre d'équations et toutes les autres variables contenues dans les équations doivent pouvoir être évaluées à des nombres.

Une condition non réelle est souvent nécessaire pour la détermination d'une solution non réelle. Pour assurer une convergence correcte, la valeur utilisée doit être relativement proche de la solution.

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

$$u=\frac{e^w+i}{2} \text{ and } v=\frac{e^w-i}{2}$$

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

$$w=0.494866 \text{ and } z=0.703467$$

$$\text{cSolve}\left(e^z=w \text{ and } w=z^2, \{w, z=1+i\}\right)$$

$$w=0.149606+4.8919 \cdot i \text{ and } z=1.58805+1.5402 \cdot i$$

Pour afficher le résultat entier, appuyez sur , puis utilisez les touches  et  pour déplacer le curseur.

CubicReg

CubicReg $X, Y[, [Fréq] [, Catégorie, Inclure]]$

Effectue l'ajustement polynomial de degré $3y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$ sur les listes X et Y en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et Y sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple X et Y . Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples X et Y correspondants.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a \cdot x^3 + b \cdot x^2 + c \cdot x + d$
stat.a, stat.b, stat.c, stat.d	Coefficients d'ajustement
stat.R ²	Coefficient de détermination
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

cumulativeSum()

cumulativeSum(ListeI) $\Rightarrow$ liste

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

Donne la liste des sommes cumulées des éléments de *Liste1*, en commençant par le premier élément (élément 1).

cumulativeSum(*Matrice1*)⇒*matrice*

Donne la matrice des sommes cumulées des éléments de *Matrice1*. Chaque élément correspond à la somme cumulée de tous les éléments situés au-dessus, dans la colonne correspondante.

Un élément vide de *Liste1* ou *Matrice1* génère un élément vide dans la liste ou la matrice résultante. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
cumulativeSum(<i>m1</i>)	$\begin{bmatrix} 1 & 2 \\ 4 & 6 \\ 9 & 12 \end{bmatrix}$

Cycle

Cycle

Procède au passage immédiat à l'itération suivante de la boucle courante (**For**, **While** ou **Loop**).

La fonction Cycle ne peut pas s'utiliser indépendamment de l'une des trois structures de boucle (**For**, **While** ou **Loop**).

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Liste de fonctions qui additionne les entiers compris entre 1 et 100, en sautant 50.

Define g() Local temp,i 0→temp For i,1,100,1 If i=50 Cycle temp+i→temp EndFor Return temp EndFunc	Done
g()	5000

►Cylind

Vecteur ►Cylind

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>**Cylind**.

Affiche le vecteur ligne ou colonne en coordonnées cylindriques $[r, \angle \theta, z]$.

$\begin{bmatrix} 2 & 2 & 3 \end{bmatrix} \blacktriangleright \text{Cylind}$	$\begin{bmatrix} 2\sqrt{2} & \angle \frac{\pi}{4} & 3 \end{bmatrix}$
---	--

Vecteur doit être un vecteur à trois éléments. Il peut s'agir d'un vecteur ligne ou colonne.

cZeros()

cZeros(*Expr*, *Var*)⇒*liste*

Donne la liste des valeurs réelles et non réelles possibles de *Var* qui annulent *Expr*. Pour y parvenir, **cZeros()** calcule **explist(cSolve(Expr=0,Var),Var)**. Pour le reste, **cZeros()** est comparable à **zeros()**.

Remarque : voir aussi **cSolve()**, **solve()** et **zeros()**.

cZeros({*Expr1*, *Expr2* [, ...] },

{*VarOuInit1*, *VarOuInit2* [, ...]
})⇒*matrice*

Donne les valeurs possibles auxquelles les expressions s'annulent simultanément. Chaque *VarOuInit* définit une inconnue dont vous recherchez la valeur.

Vous pouvez également spécifier une condition initiale pour les variables. Chaque *VarOuInit* doit utiliser le format suivant :

variable

– ou –

variable = *nombre réel ou non réel*

Par exemple, *x* est autorisé, de même que *x=3+i*.

Si toutes les expressions sont polynomiales et si vous NE spécifiez PAS de condition initiale, **cZeros()** utilise la méthode d'élimination lexicale Gröbner/Buchberger pour tenter de trouver **tous** les zéros complexes.

En mode Afficher chiffres, Fixe 3 :

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

$$\{-1.114 + 1.073 \cdot i; -1.114 - 1.073 \cdot i; -2.125, -0.612, 0\}$$

Pour afficher le résultat entier, appuyez sur , puis utilisez les touches  et  pour déplacer le curseur.

Les zéros complexes peuvent combiner des zéros réels et des zéros non réels, comme illustré dans l'exemple ci-contre.

Chaque ligne de la matrice résultante représente un `n_uplet`, l'ordre des composants étant identique à celui de la liste `VarOuInit`. Pour extraire une ligne, indexez la matrice par `[ligne]`.

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

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

Extraction ligne 2 :

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

Les systèmes d'équations polynomiales peuvent comporter des variables supplémentaires auxquelles aucune valeur n'est affectée, mais qui représentent des valeurs numériques données pouvant s'y substituer par la suite.

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

$$\begin{bmatrix} 0 & 0 \\ -(c-1)^2 & -(c-1) \end{bmatrix}$$

Vous pouvez également utiliser des inconnues qui n'apparaissent pas dans les expressions. Ces exemples montrent comment des ensembles de zéros peuvent dépendre de constantes arbitraires de type `ck`, où `k` est un suffixe entier compris entre 1 et 255.

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

$$\text{cZero}\left(\left\{u \cdot (v-1) - v, w + v^2\right\}, \{u, v, w\}\right)$$

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

Pour les systèmes d'équations polynomiales, le temps de calcul et l'utilisation de la mémoire peuvent considérablement varier en fonction de l'ordre dans lequel les inconnues sont spécifiées. Si votre choix initial ne vous satisfait pas pour ces raisons, vous pouvez modifier l'ordre des variables dans les expressions et/ou la liste `VarOuInit`.

Si vous choisissez de ne pas spécifier de condition et s'il l'une des expressions n'est pas polynomiale en l'une des variables, mais que toutes les expressions sont linéaires par rapport à toutes les inconnues, **cZeros()** utilise l'élimination gaussienne pour tenter de trouver tous les zéros.

$$\text{cZeros}\left(\left\{u + v - e^{1w}, u - v - i\right\}, \{u, v\}\right)$$

$$\begin{bmatrix} \frac{e^{1w} + i}{2} & \frac{e^{1w} - i}{2} \end{bmatrix}$$

Si un système d'équations n'est pas polynomial en toutes ses variables ni linéaire par rapport à ses inconnues, **cZeros()** cherche au moins un zéro en utilisant une méthode itérative approchée. Pour cela, le nombre d'inconnues doit être égal au nombre d'expressions et toutes les autres variables contenues dans les expressions doivent pouvoir être évaluées à des nombres.

$$\left| \begin{array}{l} \text{cZeros}\left(\left\{e^z - w, w - z^2\right\}, \left\{w, z\right\}\right) \\ \left[0.494866 \quad -0.703467\right] \end{array} \right|$$

Une condition non réelle est souvent nécessaire pour la détermination d'un zéro non réel. Pour assurer une convergence correcte, la valeur utilisée doit être relativement proche d'un zéro.

$$\left| \begin{array}{l} \text{cZeros}\left(\left\{e^{-z} - w, w - z^2\right\}, \left\{w, z = 1 + i\right\}\right) \\ \left[0.149606 + 4.8919 \cdot i \quad 1.58805 + 1.54022 \cdot i\right] \end{array} \right|$$

D

dbd()

dbd(date1, date2) ⇒ valeur

Calcule le nombre de jours entre *date1* et *date2* à l'aide de la méthode de calcul des jours.

date1 et *date2* peuvent être des chiffres ou des listes de chiffres compris dans une plage de dates d'un calendrier normal. Si *date1* et *date2* sont toutes deux des listes, elles doivent être de la même longueur.

date1 et *date2* doivent être comprises entre 1950 et 2049.

Vous pouvez saisir les dates à l'un des deux formats. L'emplacement de la décimale permet de distinguer les deux formats.

MM.JJAA (format communément utilisé aux Etats-Unis)

JJMM.AA (format communément utilisé en Europe)

dbd(12.3103,1.0104)	1
dbd(1.0107,6.0107)	151
dbd(3112.03,101.04)	1
dbd(101.07,106.07)	151

Valeur ►DD⇒valeur

Liste1 ►DD⇒liste

Matrice1 ►DD⇒matrice

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>DD.

Donne l'équivalent décimal de l'argument exprimé en degrés. L'argument est un nombre, une liste ou une matrice interprété suivant le mode Angle utilisé (grades, radians ou degrés).

En mode Angle en degrés :

(1.5°) ►DD	1.5°
$(45^\circ 22' 14.3'')$ ►DD	45.3706°
$(\{45^\circ 22' 14.3'', 60^\circ 0' 0''\})$ ►DD	$\{45.3706^\circ, 60^\circ\}$

En mode Angle en grades :

1►DD	$\frac{9}{10}^\circ$
------	----------------------

En mode Angle en radians :

(1.5) ►DD	85.9437°
-------------	----------

Expr1 ►Decimal⇒expression

Liste1 ►Decimal⇒expression

Matrice1 ►Decimal⇒expression

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>Decimal.

Affiche l'argument sous forme décimale. Cet opérateur ne peut être utilisé qu'à la fin d'une ligne.

$\frac{1}{3}$ ►Decimal	0.333333
------------------------	----------

Define Var = Expression

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

Définit la variable Var ou la fonction définie par l'utilisateur Fonction.

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

Les paramètres, tels que *Param1*, sont des paramètres substituables utilisés pour transmettre les arguments à la fonction. Lors de l'appel d'une fonction définie par l'utilisateur, des arguments (par exemple, les valeurs ou variables) qui correspondent aux paramètres doivent être fournis. La fonction évalue ensuite *Expression* en utilisant les arguments fournis.

Var et *Fonction* ne peuvent pas être le nom d'une variable système ni celui d'une fonction ou d'une commande prédéfinie.

Remarque : cette utilisation de **Define** est équivalente à celle de l'instruction :
expression → *Fonction*
(*Param1*,*Param2*).

Define *Fonction*(*Param1*, *Param2*, ...) =
Func
Bloc
EndFunc

Define *Programme*(*Param1*, *Param2*, ...) = **Prgm**
Bloc
EndPrgm

Dans ce cas, la fonction définie par l'utilisateur ou le programme permet d'exécuter plusieurs instructions (bloc).

Bloc peut correspondre à une instruction unique ou à une série d'instructions réparties sur plusieurs lignes. *Bloc* peut également contenir des expressions et des instructions (comme **If**, **Then**, **Else** et **For**).

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

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

Remarque : voir aussi **Define LibPriv**, page 53 et **Define LibPub**, page 53.

Define LibPriv

Define LibPriv *Var = Expression*

Define LibPriv *Fonction(Param1, Param2, ...)* = *Expression*

Define LibPriv *Fonction(Param1, Param2, ...)* = **Func**
Bloc
EndFunc

Define LibPriv *Programme(Param1, Param2, ...)* = **Prgm**
Bloc
EndPrgm

S'utilise comme **Define**, mais permet de définir des objets (variables, fonctions, programmes) dans la bibliothèque privée. Les fonctions et programmes privés ne s'affichent pas dans le Catalogue.

Remarque : voir aussi **Define**, page 51 et **Define LibPub**, page 53.

Define LibPub

Define LibPub *Var = Expression*

Define LibPub *Fonction(Param1, Param2, ...)* = *Expression*

Define LibPub *Fonction(Param1, Param2, ...)* = **Func**
Bloc
EndFunc

Define LibPub *Programme(Param1, Param2, ...)* = **Prgm**
Bloc
EndPrgm

S'utilise comme **Define**, mais permet de définir des objets (variables, fonctions, programmes) dans la bibliothèque publique. Les fonctions et programmes publics s'affichent dans le Catalogue après l'enregistrement et le rafraîchissement de la bibliothèque.

Remarque : voir aussi **Define**, page 51 et **Define LibPriv**, page 53.

deltaList()

Voir Δ List(), page 115.

deltaTmpCnv()

Voir Δ tmpCnv(), page 212.

DelVar

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

$2 \rightarrow a$	2
-------------------	---

DelVar *Var*.

$(a+2)^2$	16
-----------	----

Supprime de la mémoire la variable ou le groupe de variables spécifié.

DelVar <i>a</i>	Done
-----------------	------

$(a+2)^2$	$(a+2)^2$
-----------	-----------

Si une ou plusieurs variables sont verrouillées, cette commande affiche un message d'erreur et ne supprime que les variables non verrouillées. Voir **unLock**, page 221.

DelVar *Var*. supprime tous les membres du groupe de variables *Var*, comme les variables statistiques du groupe *stat*, le résultat *nn* ou les variables créées à l'aide de la fonction **LibShortcut()**. Le point (.) dans cette utilisation de la commande **DelVar** limite la suppression au groupe de variables ; la variable simple *Var* n'est pas supprimée.

<i>aa.a</i> :=45	45									
<i>aa.b</i> :=5.67	5.67									
<i>aa.c</i> :=78.9	78.9									
getVarInfo()	<table><tr><td><i>aa.a</i></td><td>"NUM"</td><td>"[]"</td></tr><tr><td><i>aa.b</i></td><td>"NUM"</td><td>"[]"</td></tr><tr><td><i>aa.c</i></td><td>"NUM"</td><td>"[]"</td></tr></table>	<i>aa.a</i>	"NUM"	"[]"	<i>aa.b</i>	"NUM"	"[]"	<i>aa.c</i>	"NUM"	"[]"
<i>aa.a</i>	"NUM"	"[]"								
<i>aa.b</i>	"NUM"	"[]"								
<i>aa.c</i>	"NUM"	"[]"								
DelVar <i>aa</i> .	<i>Done</i>									
getVarInfo()	"NONE"									

delVoid(Liste1) ⇒ liste

delVoid({1,void,3})

{1,3}

Donne une liste contenant les éléments de *Liste1* sans les éléments vides.

Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

deSolve(ode1erOu2ndOrdre, Var, VarDép) ⇒ une solution générale

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

Donne une équation qui définit explicitement ou implicitement la solution générale de l'équation différentielle du 1er ou du 2ème ordre. Dans l'équation différentielle :

right(Ans) → temp $(c3\cdot x+c4)\cdot e^{-x}+x^2-4\cdot x+6$ $\frac{d^2}{dx^2}(temp)+2\cdot \frac{d}{dx}(temp)+temp-x^2$ 0

DelVar temp

Done

- Utilisez uniquement le symbole « prime » (obtenu en appuyant sur ) pour indiquer la dérivée première de la fonction (variable dépendante) par rapport à la variable (variable indépendante).
- Utilisez deux symboles « prime » pour indiquer la dérivée seconde correspondante.

Le symbole « prime » s'utilise pour les dérivées uniquement dans deSolve(). Dans tous les autres cas, utilisez *d()*.

La solution générale d'une équation du 1er ordre comporte une constante arbitraire de type ck, où k est un suffixe entier compris entre 1 et 255. La solution générale d'une équation de 2ème ordre contient deux constantes de ce type.

Appliquez **solve()** à une solution implicite si vous voulez tenter de la convertir en une ou plusieurs solutions explicites équivalente déterminées explicitement.

Si vous comparez vos résultats avec ceux de vos manuels de cours ou ceux obtenus manuellement, sachez que certaines méthodes introduisent des constantes arbitraires en plusieurs endroits du calcul, ce qui peut induire des solutions générales différentes.

deSolve(ode1erOrdre and condInit, Var, VarDép) $\Rightarrow$ une solution particulière

Donne une solution particulière qui satisfait à la fois *ode1erOrdre* et *condInit*. Ceci est généralement plus simple que de déterminer une solution générale car on substitue les valeurs initiales, calcule la constante arbitraire, puis substitue cette valeur dans la solution générale.

codInit est une équation de type :

VarDép (*valeurIndépendanteInitiale*) = *valeurDépendanteInitiale*

valeurIndépendanteInitiale et *valeurDépendanteInitiale* peuvent être des variables comme *x0* et *y0* non affectées. La différentiation implicite peut aider à vérifier les solutions implicites.

deSolve

(ode2ndOrdre and condInit1 and condInit2, Var, VarDép) $\Rightarrow$ une solution particulière

Donne une solution particulière qui satisfait *ode2ndOrdre* et qui a une valeur spécifique de la variable dépendante et sa dérivée première en un point.

Pour *condInit1*, utilisez :

VarDép (*valeurIndépendanteInitiale*) = *valeurDépendanteInitiale*

Pour *condInit2*, utilisez :

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

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

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

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

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

$$soln | x = 0 \text{ and } y = 0 \quad true$$

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

$$\text{DelVar } ode, soln \quad Done$$

$$\text{deSolve}(y'' = y^{-\frac{1}{2}} \text{ and } y(0) = 0 \text{ and } y'(0) = 0, t, y) \quad \frac{\frac{3}{2} \cdot y^{\frac{4}{3}}}{3} = t$$

$$\text{solve}\left(\frac{\frac{3}{2} \cdot y^{\frac{4}{3}}}{3} = t, y\right)$$

$$y = \frac{3 \cdot 3^{\frac{1}{3}} \cdot 2^{\frac{2}{3}} \cdot t^{\frac{3}{4}}}{4} \text{ and } t \geq 0$$

*VarDép (ValeurIndépendanteInitiale) =
ValeurInitialeDérivée1*

deSolve
(*ode2ndOrdre* and *condBorne1* and
condBorne2, *Var*, *VarDép*) $\Rightarrow$ une solution
particulière

Donne une solution particulière qui satisfait
ode2ndOrdre et qui a des valeurs
spécifiques en deux points différents.

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

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

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

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

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

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

det()

det(*matriceCarrée* [,
Tolérance]) $\Rightarrow$ expression

Donne le déterminant de
matriceCarrée.

L'argument facultatif *Tolérance* permet
de considérer comme nul tout élément
de la matrice dont la valeur absolue est
inférieure à *Tolérance*. Cet argument
n'est utilisé que si la matrice contient
des nombres en virgule flottante et ne
contient pas de variables symboliques
sans valeur affectée. Dans le cas
contraire, *Tolérance* est ignoré.

$$\det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) \quad a \cdot d - b \cdot c$$

$$\det\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right) \quad -2$$

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

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

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

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

- Si vous utilisez   ou définissez
le mode **Auto ou Approché** sur
Approché, les calculs sont effectués en
virgule flottante.
- Si *Tolérance* est omis ou inutilisé, la
tolérance par défaut est calculée
comme suit :

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

diag()Catalogue > **diag(Liste)**⇒matrice

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

diag(matriceLigne)⇒matrice**diag(matriceColonne)**⇒matrice

Donne une matrice diagonale, ayant sur sa diagonale principale les éléments de la liste passée en argument.

diag(matriceCarrée)⇒matriceLigne

Donne une matrice ligne contenant les éléments de la diagonale principale de *matriceCarrée*.

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

matriceCarrée doit être une matrice carrée.

dim()Catalogue > **dim(Liste)**⇒entier

dim({0,1,2})	3
--------------	---

Donne le nombre d'éléments de *Liste*.

dim(Matrice)⇒liste

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

Donne les dimensions de la matrice sous la forme d'une liste à deux éléments {lignes, colonnes}.

dim(Chaîne)⇒entier

dim("Hello")	5
dim("Hello "&"there")	11

Donne le nombre de caractères contenus dans *Chaîne*.

Disp *exprOuChaine1* [, *exprOuChaine2*] ...

Affiche les arguments dans l'historique de *Calculator*. Les arguments apparaissent les uns après les autres, séparés par des espaces fines.

Très utile dans les programmes et fonctions pour l'affichage de calculs intermédiaires.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

```
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

DispAt *int,expr1 [,expr2 ...] ...*

DispAt vous permet de spécifier la ligne où l'expression ou la chaîne de caractère spécifiée s'affichera à l'écran.

Le numéro de ligne peut être spécifié sous forme d'expression.

Veuillez noter que le numéro de ligne n'est pas destiné à l'ensemble de l'écran, mais uniquement à la zone suivant immédiatement la commande/le programme.

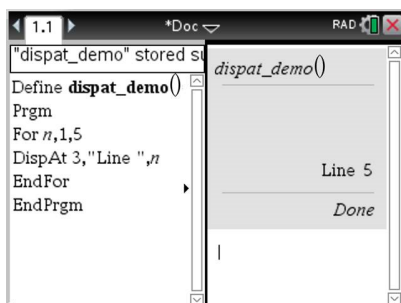
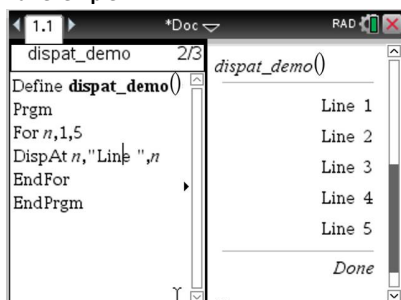
Cette commande permet des sorties de type tableau de bord de programmes où la valeur d'une expression ou d'une lecture de capteur est mise à jour sur la même ligne.

DispAt et **Disp** peuvent être utilisés au sein du même programme.

Remarque : Le nombre maximum est défini sur 8, du fait que cela correspond à un écran entier de lignes sur l'écran d'une calculatrice - du moment que les lignes ne contiennent pas d'expressions mathématiques 2D. Le nombre exact de lignes dépend du contenu des informations affichées.

DispAt

Par exemple :



Exemples illustratifs :

Define z()=	Output
Prgm	z()
For n,1,3	Itération 1 :
DispAt 1,\"N : \",n	Ligne 1 : N :1
Disp \"Bonjour\"	Ligne 2 : Bonjour
EndFor	
EndPrgm	Itération 2 :
	Ligne 1 : N :2
	Ligne 2 : Bonjour
	Ligne 3 : Bonjour
	Itération 3 :
	Ligne 1 : N :3

	Ligne 2 : Bonjour Ligne 3 : Bonjour Ligne 4 : Bonjour
Define z1() Prgm For n,1,3 DispAt 1,"N : ",n EndFor For n,1,4 Disp "Bonjour" EndFor EndPrgm	z1() Ligne 1 : N :3 Ligne 2 : Bonjour Ligne 3 : Bonjour Ligne 4 : Bonjour Ligne 5 : Bonjour

Conditions d'erreur :

Message d'erreur	Description
Le numéro de ligne DispAt doit être compris entre 1 et 8	L'expression évalue le numéro de la ligne en dehors de la plage 1 - 8 (inclus)
Nombre insuffisant d'arguments	Il manque un ou plusieurs arguments à la fonction ou commande.
Aucun argument	Identique à la boîte de dialogue « erreur de syntaxe » actuelle
Trop d'arguments	Limiter les arguments. Même erreur que Disp.
Type de données incorrect	Le premier argument doit être un nombre.
Vide : DispAt vide	L'erreur de type de données "Hello World" (Datatype error) est renvoyée pour le vide (si le rappel est défini)
Opérateur de conversion : DispAt 2_ft @> _m, "Hello World"	CAS : Une erreur de type de données (Datatype Error) est renvoyée (si le rappel est défini) Numérique : La conversion sera évaluée et si le résultat est un argument valide, DispAt imprimera la chaîne de caractère sur la ligne de résultat.

Liste ►DMS

{45.371}►DMS 45°22'15.6"

Matrice ►DMS

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

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>DMS.

Interprète l'argument comme un angle et affiche le nombre DMS équivalent (DDDDDD°MM'SS.ss"). Voir °, ', "page 255 pour le détail du format DMS (degrés, minutes, secondes).

Remarque : ►DMS convertit les radians en degrés lorsque l'instruction est utilisée en mode radians. Si l'entrée est suivie du symbole des degrés °, aucune conversion n'est effectuée. Vous ne pouvez utiliser ►DMS qu'à la fin d'une ligne.

domain()

domain(Expr1, Var)⇒expression

Renvoie le domaine de définition de Expr1 par rapport à Var.

domain() peut être utilisé pour déterminer le domaine de définition d'une fonction. Il est limité au domaine réel et fini.

Cette fonction est limitée, en raison des lacunes en termes de simplification du calcul formel et des algorithmes de résolution.

Certaines fonctions ne peuvent pas être utilisées comme arguments pour **domain()**, indépendamment du fait qu'elles apparaissent de manière explicite ou au sein de variables et de fonctions définies par l'utilisateur. Dans l'exemple suivant, l'expression ne peut pas être simplifiée car $\int()$ est une fonction non autorisée.

$$\text{domain}\left(\frac{1}{x+y}, y\right) \quad -\infty < y < -x \text{ or } -x < y < \infty$$

$$\text{domain}\left(\frac{x+1}{x^2+2 \cdot x}, x\right) \quad x \neq -2 \text{ and } x \neq 0$$

$$\text{domain}\left(\left(\sqrt{x}\right)^2, x\right) \quad 0 \leq x < \infty$$

$$\text{domain}\left(\frac{1}{x+y}, y\right) \quad -\infty < y < -x \text{ or } -x < y < \infty$$

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

dominantTerm()

dominantTerm(Expr1, Var [, Point]) $\Rightarrow$ expression

dominantTerm(Expr1, Var [, Point]) | Var > Point $\Leftrightarrow$ expression

dominantTerm(Expr1, Var [, Point]) | Var < Point $\Rightarrow$ expression

Donne le terme dominant du développement en série généralisé de Expr1 au Point. Le terme dominant est celui dont le module croît le plus rapidement en Var = Point. La puissance de (Var - Point) peut avoir un exposant négatif et/ou fractionnaire. Le coefficient de cette puissance peut inclure des logarithmes de (Var - Point) et d'autres fonctions de Var dominés par toutes les puissances de (Var - Point) ayant le même signe d'exposant.

La valeur par défaut de Point est 0. Point peut être ∞ ou $-\infty$, auxquels cas le terme dominant est celui qui a l'exposant de Var le plus grand au lieu de celui qui l'exposant de Var le plus petit.

dominantTerm(...) donne "**dominantTerm(...)**" s'il ne parvient pas à déterminer la représentation, comme pour les singularités essentielles de type $\sin(1/z)$ en $z=0$, $e^{-1/z}$ en $z=0$ ou e^z en $z = \infty$ ou $-\infty$.

$$\text{dominantTerm}(\tan(\sin(x)) - \sin(\tan(x)), x)$$

$$\frac{x^7}{30}$$

$$\text{dominantTerm}\left(\frac{1 - \cos(x-1)}{(x-1)^3}, x, 1\right) \quad \frac{1}{2 \cdot (x-1)}$$

$$\text{dominantTerm}\left(x^{-2} \cdot \tan\left(x^{\frac{1}{3}}\right), x\right) \quad \frac{1}{x^{\frac{5}{3}}}$$

$$\text{dominantTerm}(\ln(x^x - 1) \cdot x^{-2}, x) \quad \frac{\ln(x \cdot \ln(x))}{x^2}$$

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

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

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

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

Si la série ou une de ses dérivées présente une discontinuité en *Point*, le résultat peut contenir des sous-expressions de type `sign(...)` ou `abs(...)` pour une variable réelle ou `(-1)floor` (`...angle(...)`) pour une variable complexe, qui se termine par « `_` ». Si vous voulez utiliser le terme dominant uniquement pour des valeurs supérieures ou inférieures à *Point*, vous devez ajouter à **dominantTerm(...)** l'élément approprié « `| Var > Point` », « `| Var < Point` », « `|` » « `Var ≥ Point` » ou « `Var ≤ Point` » pour obtenir un résultat simplifié.

dominantTerm() est appliqué à chaque élément d'une liste ou d'une matrice passée en 1er argument.

dominantTerm() est utile pour connaître l'expression la plus simple correspondant à l'expression asymptotique d'un équivalent d'une expression quand $Var \rightarrow Point$. **dominantTerm()** peut également être utilisé lorsqu'il n'est pas évident de déterminer le degré du premier terme non nul d'une série et que vous ne souhaitez pas tester les hypothèses de manière interactive ou via une boucle.

Remarque : voir aussi **series()**, page 180.

dotP()

dotP(Liste1, Liste2) ⇒ expression

Donne le produit scalaire de deux listes.

$\text{dotP}(\{a, b, c\}, \{d, e, f\})$	$a \cdot d + b \cdot e + c \cdot f$
$\text{dotP}(\{1, 2\}, \{5, 6\})$	17

dotP(Vecteur1, Vecteur2) ⇒ expression

Donne le produit scalaire de deux vecteurs.

$\text{dotP}\left(\begin{bmatrix} a & b & c \end{bmatrix}, \begin{bmatrix} d & e & f \end{bmatrix}\right)$	$a \cdot d + b \cdot e + c \cdot f$
$\text{dotP}\left(\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 4 & 5 & 6 \end{bmatrix}\right)$	32

Les deux vecteurs doivent être de même type (ligne ou colonne).

e^()		Touche e^x	
e^(Expr1)⇒expression		e^1	e
Donne e élevé à la puissance de <i>Expr1</i> .		$e^1.$	2.71828
Remarque : voir aussi Modèle e Exposant , page 2.		e^{3^2}	e^9
Remarque : une pression sur e^x pour afficher $e^()$ est différente d'une pression sur le caractère E du clavier.			
Vous pouvez entrer un nombre complexe sous la forme polaire $re^{i\theta}$. N'utilisez toutefois cette forme qu'en mode Angle en radians ; elle provoque une erreur de domaine en mode Angle en degrés ou en grades.			
e^(Liste1)⇒liste		$e^{\{1,1.,0.5\}}$	$\{e,2.71828,1.64872\}$
Donne une liste constituée des exponentielles des éléments de <i>Liste1</i> .			
e(matriceCarrée1)⇒matriceCarrée		$e^{\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}}$	$\begin{bmatrix} 782.209 & 559.617 & 456.509 \\ 680.546 & 488.795 & 396.521 \\ 524.929 & 371.222 & 307.879 \end{bmatrix}$
Donne l'exponentielle de <i>matriceCarrée1</i> . Le résultat est différent de la matrice obtenue en prenant l'exponentielle de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à cos() .			
<i>matriceCarrée1</i> doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.			

eff()		Catalogue	> 
eff(tauxNominal,CpY)⇒valeur		$eff(5.75,12)$	5.90398
Fonction financière permettant de convertir un taux d'intérêt nominal <i>tauxNominal</i> en un taux annuel effectif, <i>CpY</i> étant le nombre de périodes de calcul par an.			
<i>tauxNominal</i> doit être un nombre réel et <i>CpY</i> doit être un nombre réel > 0.			

Remarque : voir également **nom()**, page 138.

eigVc()

eigVc(matriceCarrée)⇒matrice

Donne une matrice contenant les vecteurs propres d'une *matriceCarrée* réelle ou complexe, chaque colonne du résultat correspond à une valeur propre. Notez qu'il n'y a pas unicité des vecteurs propres. Ils peuvent être multipliés par n'importe quel facteur constant. Les vecteurs propres sont normés, ce qui signifie que si $V = [x_1, x_2, \dots, x_n]$, alors :

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

matriceCarrée est d'abord transformée en une matrice semblable dont la norme par rapport aux lignes soit le plus proche de celle par rapport aux colonnes. La *matriceCarrée* est ensuite réduite à la forme de Hessenberg supérieure et les vecteurs propres calculés via une factorisation de Schur.

En mode Format complexe Rectangulaire :

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

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

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

Pour afficher le résultat entier, appuyez sur $\blacktriangle$, puis utilisez les touches $\blacktriangleleft$ et $\blacktriangleright$ pour déplacer le curseur.

eigVl()

eigVl(matriceCarrée)⇒liste

Donne la liste des valeurs propres d'une *matriceCarrée* réelle ou complexe.

matriceCarrée est d'abord transformée en une matrice semblable dont la norme par rapport aux lignes soit le plus proche de celle par rapport aux colonnes. La *matriceCarrée* est ensuite réduite à la forme de Hessenberg supérieure et les valeurs propres calculées à partir de la matrice de Hessenberg supérieure.

En mode Format complexe Rectangulaire :

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

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

$$\{-4.40941, 2.20471+0.763006 \cdot i, 2.20471-0.763006 \cdot i\}$$

Pour afficher le résultat entier, appuyez sur $\blacktriangle$, puis utilisez les touches $\blacktriangleleft$ et $\blacktriangleright$ pour déplacer le curseur.

Else

Voir If, page 98.

If *Expr booléenne1* **Then**

Bloc1

ElseIf *Expr booléenne2* **Then**

Bloc2

⋮

ElseIf *Expr booléenneN* **Then**

BlocN

EndIf

⋮

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions

sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

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

EndFor

Voir For, page 81.

EndFunc

Voir Func, page 86.

EndIf

Voir If, page 98.

EndLoop

Voir Loop, page 124.

EndPrgm

Voir Prgm, page 154.

EndTry

Voir Try, page 214.

euler ()

Catalogue > 

euler(*Expr*, *Var*, *VarDép*, {*Var0*, *MaxVar*}, *Var0Dép*, *IncVar* [, *IncEuler*]) $\Rightarrow$ *matrice*

euler(*SystèmeExpr*, *Var*, *ListeVarDép*, {*Var0*, *MaxVar*}, *ListeVar0Dép*, *IncVar* [, *IncEuler*]) $\Rightarrow$ *matrice*

euler(*ListeExpr*, *Var*, *ListeVarDép*, {*Var0*, *MaxVar*}, *ListeVar0Dép*, *IncVar* [, *IncEuler*]) $\Rightarrow$ *matrice*

Utilise la méthode d'Euler pour résoudre le système.

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

avec $\text{VarDép}(\text{Var0}) = \text{Var0Dép}$ pour l'intervalle [*Var0*, *MaxVar*]. Retourne une matrice dont la première ligne définit les valeurs de sortie de *Var* et la deuxième ligne la valeur du premier composant de la solution pour les valeurs correspondantes de *Var*, etc.

Expr représente la partie droite qui définit l'équation différentielle.

SystèmeExpr correspond aux côtés droits qui définissent le système des équations différentielles (en fonction de l'ordre des variables dépendantes de la *ListeVarDép*).

ListeExpr est la liste des côtés droits qui définissent le système des équations différentielles (en fonction de l'ordre des variables dépendantes de la *ListeVarDép*).

Var est la variable indépendante.

ListeVarDép est la liste des variables dépendantes.

Équation différentielle :

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

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

0.	1.	2.	3.	4.
10.	10.9	11.8712	12.9174	14.042

Pour afficher le résultat entier, appuyez sur $\blacktriangle$, puis utilisez les touches $\blacktriangleleft$ et $\blacktriangleright$ pour déplacer le curseur.

Comparez le résultat ci-dessus avec la solution exacte CAS obtenue en utilisant `deSolve()` et `seqGen()` :

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

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

Système d'équations :

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

avec $y1(0) = 2$ et $y2(0) = 5$

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

0.	1.	2.	3.	4.	5.
2.	1.	1.	3.	27.	243.
5.	10.	30.	90.	90.	-2070.

$\{Var0, MaxVar\}$ est une liste à deux éléments qui indique la fonction à intégrer de $Var0$ à $MaxVar$.

$ListeVar0Dép$ est la liste des valeurs initiales pour les variables dépendantes.

$IncVar$ est un nombre différent de zéro, défini par $sign(IncVar) = sign(MaxVar - Var0)$ et les solutions sont retournées pour $Var0 + i \cdot IncVar$ pour tout $i=0,1,2,\dots$ de sorte que $Var0 + i \cdot IncVar$ soit dans $[var0, MaxVar]$ (il est possible qu'il n'existe pas de solution en $MaxVar$).

$IncEuler$ est un entier positif (valeur par défaut : 1) qui définit le nombre d'incrémentations dans la méthode d'Euler entre deux valeurs de sortie. La taille d'incrément courante utilisée par la méthode d'Euler est $IncVar / IncEuler$.

eval ()

Menu hub

eval(*Expr*) $\Rightarrow$ chaîne

eval() n'est valable que dans TI-Innovator™ Hub l'argument de commande des commandes de programmation **Get**, **GetStr** et **Send**. Le logiciel évalue l'expression *Expr* et remplace l'instruction **eval()** par le résultat sous la forme d'une chaîne de caractères.

L'argument *Expr* doit pouvoir être simplifié en un nombre réel.

Définissez l'élément bleu de la DEL RGB en demi-intensité.

<i>lum</i> :=127	127
Send "SET COLOR.BLUE eval(<i>lum</i>)"	Done

Réinitialisez l'élément bleu sur OFF (ARRÊT).

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

L'argument de eval() doit pouvoir être simplifié en un nombre réel.

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

Programmez pour faire apparaître en fondu l'élément rouge

```

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

```

Exécutez le programme.

<i>fadein()</i>	Done
<i>n:=0.25</i>	0.25
<i>m:=8</i>	8
<i>n · m</i>	2.
Send "SET COLOR.BLUE ON TIME eval(n · m)"	Done
<i>iostr.SendAns</i>	"SET COLOR.BLUE ON TIME 2"

Même si **eval()** n'affiche pas son résultat, vous pouvez afficher la chaîne de commande Hub qui en découle après avoir exécuté la commande en inspectant l'une des variables spéciales suivantes.

iostr.SendAns
iostr.GetAns
iostr.GetStrAns

Remarque : Voir également **Get** (page 87), **GetStr** (page 95) et **Send** (page 177).

exact()

Catalogue >

exact(*Expr1* [, *Tolérance*]) ⇒ *expression*

exact(*Liste1* [, *Tolérance*]) ⇒ *liste*

exact(*Matrice1* [, *Tolérance*]) ⇒ *matrice*

Utilise le mode Exact pour donner, si possible, la valeur formelle de l'argument.

Tolérance fixe la tolérance admise pour cette approximation. Par défaut, cet argument est égal à 0 (zéro).

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

Exit

Catalogue >

Exit

Liste des fonctions :

Permet de sortir de la boucle **For**, **While** ou **Loop** courante.

Exit

Catalogue >

Exit ne peut pas s'utiliser indépendamment de l'une des trois structures de boucle (**For**, **While** ou **Loop**).

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Define $g()$ = Func	Done
Local $temp, i$	
$0 \rightarrow temp$	
For $i, 1, 100, 1$	
$temp + i \rightarrow temp$	
If $temp > 20$ Then	
Exit	
EndIf	
EndFor	
EndFunc	
$g()$	21

►exp

Catalogue >

Expr ►exp

Exprime *Expr* en base du logarithme népérien *e*. Il s'agit d'un opérateur de conversion utilisé pour l'affichage. Cet opérateur ne peut être utilisé qu'à la fin d'une ligne.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant **@>exp**.

$\frac{d}{dx} (e^x + e^{-x})$	$2 \cdot \sinh(x)$
$2 \cdot \sinh(x) \blacktriangleright \text{exp}$	$e^x - e^{-x}$

exp()

Touche

exp(*Expr1*) ⇒ *expression*

Donne l'exponentielle de *Expr1*.

Remarque : voir aussi Modèle e
Exposant, page 2.

Vous pouvez entrer un nombre complexe sous la forme polaire $re^{i\theta}$. N'utilisez toutefois cette forme qu'en mode Angle en radians ; elle provoque une erreur de domaine en mode Angle en degrés ou en grades.

exp(*Liste1*) ⇒ *liste*

Donne une liste constituée des exponentielles des éléments *Liste1*.

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

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

exp()Touche **exp(matriceCarréeI)** ⇒ *matriceCarrée*

Donne l'exponentielle de *matriceCarréeI*. Le résultat est différent de la matrice obtenue en prenant l'exponentielle de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarréeI doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

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

exp►list()Catalogue > **exp►list(Expr, Var)** ⇒ *liste*

Recherche dans *Expr* les équations séparées par le mot « or » et retourne une liste des membres de droite des équations du type *Var=Expr*. Cela permet en particulier de récupérer facilement sous forme de liste les résultats fournis par les fonctions **solve()**, **cSolve()**, **fMin()** et **fMax()**.

Remarque : **exp►list()** n'est pas nécessaire avec les fonctions **zeros** et **cZeros()** étant donné que celles-ci donnent directement une liste de solutions.

vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **exp@>list(...)**.

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

expand()Catalogue > **expand(Expr1 [, Var])** ⇒ *expression***expand(Liste1 [, Var])** ⇒ *liste***expand(Matrice1 [, Var])** ⇒ *matrice*

expand(Expr1) développe *Expr1* en fonction de toutes ses variables. C'est un développement polynomial pour les expressions polynomiales et une décomposition en éléments simples pour les expressions rationnelles.

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

L'objectif de **expand()** est de transformer *Expr1* en une somme et/ou une différence de termes simples. Par opposition, l'objectif de **factor()** est de transformer *Expr1* en un produit et/ou un quotient de facteurs simples.

expand(Expr1, Var) développe *Expr1* en fonction de *Var*. Les mêmes puissances de *Var* sont regroupées. Les termes et leurs facteurs sont triés, *Var* étant la variable principale. Une factorisation ou un développement incident des coefficients regroupés peut se produire. L'utilisation de *Var* permet de gagner du temps, de la mémoire et de l'espace sur l'écran tout en facilitant la lecture de l'expression.

Même en présence d'une seule variable, l'utilisation de *Var* peut contribuer à une factorisation du dénominateur, utilisée pour une décomposition en éléments simples, plus complète.

Conseil : Pour les expressions rationnelles, **propFrac()** est une alternative plus rapide mais moins extrême à **expand()**.

Remarque : voir aussi **comDenom()** pour un numérateur développé sur un dénominateur développé.

expand(Expr1, [Var]) « distribue » également des logarithmes et des puissances fractionnaires indépendamment de *Var*. Pour un plus grand développement des logarithmes et des puissances fractionnaires, l'utilisation de contraintes peut s'avérer nécessaire pour s'assurer que certains facteurs ne sont pas négatifs.

expand(Expr1, [Var]) « distribue » également des valeurs absolues, **sign()**, et des exponentielles, indépendamment de *Var*.

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

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

$$\begin{array}{l} \frac{\ln(2\cdot x\cdot y)+\sqrt{2\cdot x\cdot y}}{\text{expand}(Ans)} \quad \frac{\ln(2\cdot x\cdot y)+\sqrt{2\cdot x\cdot y}}{\ln(x\cdot y)+\sqrt{2\cdot x\cdot y}+\ln(2)} \\ \text{expand}(Ans), y \geq 0 \quad \frac{\ln(x)+\sqrt{2\cdot x\cdot y}+\ln(y)+\ln(2)}{\text{sign}(x\cdot y)+|x\cdot y|+e^{2\cdot x+y}} \\ \quad \frac{e^{2\cdot x+y}+\text{sign}(x\cdot y)+|x\cdot y|}{\text{expand}(Ans)} \\ \quad \frac{\text{sign}(x)\cdot \text{sign}(y)+|x|\cdot |y|+(e^x)^2\cdot e^y}{\text{expand}(Ans)} \end{array}$$

Remarque : voir aussi **tExpand()** pour le développement contenant des sommes et des multiples d'angles.

expr()

expr(*Chaîne*) ⇒ *expression*

Convertit la chaîne de caractères contenue dans *Chaîne* en une expression. L'expression obtenue est immédiatement évaluée.

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

ExpReg

ExpReg *X*, *Y* [, [*Fréq*][, *Catégorie*, *Inclure*]]

Effectue l'ajustement exponentiel $y = a \cdot (b)^x$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et *Y* sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple *X* et *Y*. Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples *X* et *Y* correspondants.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à “Éléments vides”, page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a \cdot (b)^x$
stat.a, stat.b	Coefficients d'ajustement
stat.r ²	Coefficient de détermination linéaire pour les données transformées
stat.r	Coefficient de corrélation pour les données transformées (x , $\ln(y)$)
stat.Resid	Valeurs résiduelles associées au modèle exponentiel
stat.ResidTrans	Valeurs résiduelles associées à l'ajustement linéaire des données transformées
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

F

factor()

factor(Expr1[, Var]) ⇒ *expression*

factor(Liste1[, Var]) ⇒ *liste*

factor(Matrice1[, Var]) ⇒ *matrice*

factor(Expr1) factorise *Expr1* en fonction de l'ensemble des variables associées sur un dénominateur commun.

La factorisation *Expr1* décompose l'expression en autant de facteurs rationnels linéaires que possible sans introduire de nouvelles sous-expressions non réelles. Cette alternative peut s'avérer utile pour factoriser l'expression en fonction de plusieurs variables.

$\text{factor}(a^3 \cdot x^2 - a \cdot x^2 - a^3 + a)$	$a \cdot (a-1) \cdot (a+1) \cdot (x-1) \cdot (x+1)$
$\text{factor}(x^2+1)$	x^2+1
$\text{factor}(x^2-4)$	$(x-2) \cdot (x+2)$
$\text{factor}(x^2-3)$	x^2-3
$\text{factor}(x^2-a)$	x^2-a

factor(Expr1, Var) factorise *Expr1* en fonction de la variable *Var*.

La factorisation de *Expr1* décompose l'expression en autant de facteurs réels possible linéaires par rapport à *Var*, même si cela introduit des constantes irrationnelles ou des sous-expressions qui sont irrationnelles dans d'autres variables.

Les facteurs et leurs termes sont triés, *Var* étant la variable principale. Les mêmes puissances de *Var* sont regroupées dans chaque facteur. Utilisez *Var* si la factorisation ne doit s'effectuer que par rapport à cette variable et si vous acceptez les expressions irrationnelles dans les autres variables pour augmenter la factorisation par rapport à *Var*. Une factorisation incidente peut se produire par rapport aux autres variables.

Avec le réglage Auto du mode **Auto ou Approché (Approximate)**, l'utilisation de *Var* permet également une approximation des coefficients en virgule flottante dans le cas où les coefficients irrationnels ne peuvent pas être exprimés explicitement en termes de fonctions usuelles. Même en présence d'une seule variable, l'utilisation de *Var* peut contribuer à une factorisation plus complète.

Remarque : voir aussi **comDenom()** pour obtenir rapidement une factorisation partielle si la fonction **factor()** est trop lente ou si elle utilise trop de mémoire.

Remarque : voir aussi **cFactor()** pour une factorisation à coefficients complexes visant à chercher des facteurs linéaires.

$$\begin{array}{l} \text{factor}(a^3 \cdot x^2 - a \cdot x^2 - a^3 + a \cdot x) \\ \hline a \cdot (a^2 - 1) \cdot (x - 1) \cdot (x + 1) \\ \text{factor}(x^2 - 3 \cdot x) \\ \hline (x + \sqrt{3}) \cdot (x - \sqrt{3}) \\ \text{factor}(x^2 - a \cdot x) \\ \hline (x + \sqrt{a}) \cdot (x - \sqrt{a}) \end{array}$$

$$\begin{array}{l} \text{factor}(x^5 + 4 \cdot x^4 + 5 \cdot x^3 - 6 \cdot x - 3) \\ \hline x^5 + 4 \cdot x^4 + 5 \cdot x^3 - 6 \cdot x - 3 \\ \text{factor}(x^5 + 4 \cdot x^4 + 5 \cdot x^3 - 6 \cdot x - 3, x) \\ \hline (x - 0.964673) \cdot (x + 0.611649) \cdot (x + 2.12543) \cdot (x^2 + \dots) \end{array}$$

factor(nombreRationnel) factorise le nombre rationnel en facteurs premiers. Pour les nombres composites, le temps de calcul augmente de façon exponentielle avec le nombre de chiffres du deuxième facteur le plus grand. Par exemple, la factorisation d'un entier composé de 30 chiffres peut prendre plus d'une journée et celle d'un nombre à 100 chiffres, plus d'un siècle.

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

Pour arrêter un calcul manuellement,

- **Calculatrice:** Maintenez la touche  enfoncée et appuyez plusieurs fois sur .
- **Windows® :** Maintenez la touche **F12** enfoncée et appuyez plusieurs fois sur **Entrée**.
- **Macintosh® :** Maintenez la touche **F5** enfoncée et appuyez plusieurs fois sur **Entrée**.
- **iPad® :** L'application affiche une invite. Vous pouvez continuer à patienter ou annuler.

Si vous souhaitez uniquement déterminer si un nombre est un nombre premier, utilisez **isPrime()**. Cette méthode est plus rapide, en particulier si *nombreRationnel* n'est pas un nombre premier et si le deuxième facteur le plus grand comporte plus de cinq chiffres.

FCdf

(lowBound,upBound,dfNumér,dfDénom)⇒
nombre si *lowBound* et *upBound* sont des nombres, *liste* si *lowBound* et *upBound* sont des listes

FCdf

(lowBound,upBound,dfNumér,dfDénom)⇒
nombre si *lowBound* et *upBound* sont des nombres, *liste* si *lowBound* et *upBound* sont des listes

Calcule la fonction de répartition de la loi de Fisher F de degrés de liberté $dfNumer$ et $dfDenom$ entre $lowBound$ et $upBound$.

Pour $P(X \leq upBound)$, utilisez $lowBound = 0$.

Fill

Catalogue >

Fill *Expr*, *VarMatrice* $\Rightarrow$ *matrice*

Remplace chaque élément de la variable *VarMatrice* par *Expr*.

VarMatrice doit avoir été définie.

1 2	$\rightarrow$ <i>amatrix</i>	1 2
3 4		3 4

Fill 1.01, <i>amatrix</i>	Done
---------------------------	------

<i>amatrix</i>	1.01 1.01
	1.01 1.01

Fill *Expr*, *VarListe* $\Rightarrow$ *liste*

Remplace chaque élément de la variable *VarListe* par *Expr*.

VarListe doit avoir été définie.

{1,2,3,4,5}	$\rightarrow$ <i>alist</i>	{1,2,3,4,5}
-------------	----------------------------	-------------

Fill 1.01, <i>alist</i>	Done
-------------------------	------

<i>alist</i>	{1.01,1.01,1.01,1.01,1.01}
--------------	----------------------------

FiveNumSummary

Catalogue >

FiveNumSummary *X*[, [*Fréq*]
[, *Catégorie*, *Inclure*]]

Donne la version abrégée des statistiques à une variable pour la liste *X*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

X est une liste qui contient les données.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque valeur *X* correspondante. Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes numériques de catégories pour les valeurs *X* correspondantes.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Tout élément vide dans les listes X , $Fréq$ ou $Catégorie$ correspond à un élément vide dans l'ensemble des listes résultantes. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

Variable de sortie	Description
stat.MinX	Minimum des valeurs de x
stat.Q1X	1er quartile de x
stat.MedianX	Médiane de x
stat.Q3X	3ème quartile de x
stat.MaxX	Maximum des valeurs de x

floor()

floor(Expr1) $\Rightarrow$ entier

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

Donne le plus grand entier $\leq$ à l'argument (partie entière). Cette fonction est comparable à **int()**.

L'argument peut être un nombre réel ou un nombre complexe.

floor(Liste1) $\Rightarrow$ liste

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

floor(Matrice1) $\Rightarrow$ matrice

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

Donne la liste ou la matrice de la partie entière de chaque élément.

Remarque : voir aussi **ceiling()** et **int()**.

fMax()

fMax(Expr, Var) $\Rightarrow$ Expression booléenne

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

fMax(Expr, Var, LimitInf)

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

fMax(Expr, Var, LimitInf, LimitSup)

fMax(Expr, Var) | LimitInf $\leq$ Var $\leq$ LimitSup

Donne une expression booléenne spécifiant les valeurs possibles de *Var* pour laquelle *Expr* est à son maximum ou détermine au moins sa limite supérieure.

Vous pouvez utiliser l'opérateur "sachant que" (« | ») pour restreindre l'intervalle de recherche et/ou spécifier d'autres contraintes.

Avec le réglage Approché (Approximate) du mode **Auto ou Approché (Approximate)**, **fMax()** permet de rechercher de façon itérative un maximum local approché. C'est souvent plus rapide, surtout si vous utilisez l'opérateur « | » pour limiter la recherche à un intervalle relativement réduit qui contient exactement un maximum local.

Remarque : voir aussi **fMin()** et **max()**.

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

fMin(*Expr*, *Var*) $\Rightarrow$ Expression booléenne

fMin(*Expr*, *Var*, *LimitInf*)

fMin(*Expr*, *Var*, *LimitInf*, *LimitSup*)

fMin(*Expr*, *Var*) | *LimitInf* $\leq$ *Var* $\leq$ *LimitSup*

Donne une expression booléenne spécifiant les valeurs possibles de *Var* pour laquelle *Expr* est à son minimum ou détermine au moins sa limite inférieure.

Vous pouvez utiliser l'opérateur "sachant que" (« | ») pour restreindre l'intervalle de recherche et/ou spécifier d'autres contraintes.

$\text{fMin}\left(1 - (x-a)^2 - (x-b)^2, x\right)$	$x = -\infty$ or $x = \infty$
$\text{fMin}\left(0.5 \cdot x^3 - x - 2, x\right) x \geq 1$	$x = 1.$

Avec le réglage Approché (Approximate) du mode **Auto ou Approché (Approximate)**, **fMin()** permet de rechercher de façon itérative un minimum local approché. C'est souvent plus rapide, surtout si vous utilisez l'opérateur « | » pour limiter la recherche à un intervalle relativement réduit qui contient exactement un minimum local.

Remarque : voir aussi **fMax()** et **min()**.

For

For *Var, Début, Fin* [, *Incrément*]
Bloc
EndFor

Exécute de façon itérative les instructions de *Bloc* pour chaque valeur de *Var*, à partir de *Début* jusqu'à *Fin*, par incréments équivalents à *Incrément*.

Var ne doit pas être une variable système.

Incrément peut être une valeur positive ou négative. La valeur par défaut est 1.

Bloc peut correspondre à une ou plusieurs instructions, séparées par un « : ».

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

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

format(*Expr*[, *chaîneFormat*]) ⇒ *chaîne*

Donne *Expr* sous la forme d'une chaîne de caractères correspondant au modèle de format spécifié.

Expr doit avoir une valeur numérique.

chaîneFormat doit être une chaîne du type : « F[n] », « S[n] », « E[n] », « G[n][c] », où [] identifie les parties facultatives.

F[n] : format Fixe. n correspond au nombre de chiffres à afficher après le séparateur décimal.

S[n] : format Scientifique. n correspond au nombre de chiffres à afficher après le séparateur décimal.

E[n] : format Ingénieur. n correspond au nombre de chiffres après le premier chiffre significatif. L'exposant est ramené à un multiple de trois et le séparateur décimal est décalé vers la droite de zéro, un ou deux chiffres.

G[n][c] : identique au format Fixe, mais sépare également les chiffres à gauche de la base par groupes de trois. c spécifie le caractère séparateur des groupes et a pour valeur par défaut la virgule. Si c est un point, la base s'affiche sous forme de virgule.

[Rc] : tous les formats ci-dessus peuvent se voir ajouter en suffixe l'indicateur de base Rc, où c correspond à un caractère unique spécifiant le caractère à substituer au point de la base.

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(*Expr*) ⇒ *expression*

fPart(*Liste*) ⇒ *liste*

fPart(*Matrice*) ⇒ *matrice*

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

Donne la partie fractionnaire de l'argument.

Dans le cas d'une liste ou d'une matrice, donne les parties fractionnaires des éléments.

L'argument peut être un nombre réel ou un nombre complexe.

Fpdf()

Fpdf(*ValX*,*dfNumér*,*dfDénom*) $\Rightarrow$ *nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

FPdf(*ValX*,*dfNumér*,*dfDénom*) $\Rightarrow$ *nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

Calcule la densité de la loi **F** (Fisher) de degrés de liberté *dfNumér* et *dfDénom* en *ValX*.

freqTable►list()

freqTable►list

(*Liste1*,*listeEntFréq*) $\Rightarrow$ *liste*

Donne la liste comprenant les éléments de *Liste1* développés en fonction des fréquences contenues dans *listeEntFréq*. Cette fonction peut être utilisée pour créer une table de fréquences destinée à être utilisée avec l'application Données & statistiques.

Liste1 peut être n'importe quel type de liste valide.

listeEntFréq doit avoir le même nombre de lignes que *Liste1* et contenir uniquement des éléments entiers non négatifs. Chaque élément indique la fréquence à laquelle l'élément correspondant de *Liste1* doit être répété dans la liste des résultats. La valeur zéro (0) exclut l'élément correspond de *Liste1*.

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}

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **freqTable@>list(...)**.

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

frequency()

frequency(Liste1, ListeBinaires)⇒liste

Affiche une liste contenant le nombre total d'éléments dans *Liste1*. Les comptages sont effectués à partir de plages (binaires) définies par l'utilisateur dans *listeBinaires*.

Si *listeBinaires* est {b(1), b(2), ..., b(n)}, les plages spécifiées sont { $? \leq b(1)$, $b(1) < ? \leq b(2)$, ..., $b(n-1) < ? \leq b(n)$, $b(n) > ?$ }. Le résultat comporte un élément de plus que *listeBinaires*.

Chaque élément du résultat correspond au nombre d'éléments dans *Liste1* présents dans la plage. Exprimé en termes de fonction **countIf()**, le résultat est { countIf(liste, $? \leq b(1)$), countIf(liste, $b(1) < ? \leq b(2)$), ..., countIf(liste, $b(n-1) < ? \leq b(n)$), countIf(liste, $b(n) > ?$) }.

Les éléments de *Liste1* qui ne sont pas "placés dans une plage" ne sont pas pris en compte. Les éléments vides sont également ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

Dans l'application Tableur & listes, vous pouvez utiliser une plage de cellules à la place des deux arguments.

Remarque : voir également **countIf()**, page 39.

<i>datalist</i> := { 1, 2, e , 3, π , 4, 5, 6, "hello", 7 }
{ 1, 2, 2.71828, 3, 3.14159, 4, 5, 6, "hello", 7 }
frequency(<i>datalist</i> , { 2.5, 4.5 }) { 2, 4, 3 }

Explication du résultat :

2 éléments de *Datalist* sont $\leq 2,5$

4 éléments de *Datalist* sont $> 2,5$ et $\leq 4,5$

3 éléments de *Datalist* sont $> 4,5$

L'élément « hello » est une chaîne et ne peut être placé dans aucune des plages définies.

FTest_2Samp *Liste1, Liste2[,Fréq1[,Fréq2[,Hypoth]]]*

FTest_2Samp *Liste1, Liste2[,Fréq1[,Fréq2[,Hypoth]]]*

(Entrée de liste de données)

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

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

(Récapitulatif des statistiques fournies en entrée)

Effectue un test F sur deux échantillons. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Pour $H_a : \sigma_1 > \sigma_2$, définissez *Hypoth*>0

Pour $H_a : \sigma_1 \neq \sigma_2$ (par défaut), définissez *Hypoth*=0

Pour $H_a : \sigma_1 < \sigma_2$, définissez *Hypoth*<0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.F	Statistique $\hat{U}$ estimée pour la séquence de données
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.dfNumer	Numérateur degrés de liberté = $n_1 - 1$
stat.dfDenom	Dénominateur degrés de liberté = $n_2 - 1$.
stat.sx1, stat.sx2	Écarts types de population d'échantillon des séquences de données dans <i>Liste 1</i> et <i>Liste 2</i> .
stat.x1_bar stat.x2_bar	Moyenne de population d'échantillon des séquences de données dans <i>Liste 1</i> et <i>Liste 2</i> .
stat.n1, stat.n2	Taille des échantillons

Func*Bloc***EndFunc**

Modèle de création d'une fonction définie par l'utilisateur.

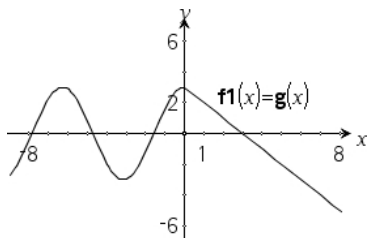
Bloc peut correspondre à une instruction unique ou à une série d'instructions séparées par le caractère “.” ou à une série d'instructions réparties sur plusieurs lignes. La fonction peut utiliser l'instruction **Return** pour donner un résultat spécifique.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Définition d'une fonction par morceaux :

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

Résultat de la représentation graphique de $g(x)$

**G****gcd()**

gcd(Nombre1, Nombre2)⇒expression

Donne le plus grand commun diviseur des deux arguments. Le **gcd** de deux fractions correspond au **gcd** de leur numérateur divisé par le **lcm** de leur dénominateur.

En mode Auto ou Approché, le **gcd** de nombre fractionnaires en virgule flottante est égal à 1.

gcd(Liste1, Liste2)⇒liste

Donne la liste des plus grands communs diviseurs des éléments correspondants de *Liste1* et *Liste2*.

gcd(Matrice1, Matrice2)⇒matrice

Donne la matrice des plus grands communs diviseurs des éléments correspondants de *Matrice1* et *Matrice2*.

gcd(18,33)	3
------------	---

gcd({12,14,16},{9,7,5})	{3,7,1}
-------------------------	---------

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

geomCdf(*p*,*lowBound*,*upBound*) $\Rightarrow$ *nombre* si les bornes *lowBound* et *upBound* sont des nombres, *liste* si les bornes *lowBound* et *upBound* sont des listes

geomCdf(*p*,*upBound*) pour $P(1 \leq X \leq upBound) \Rightarrow$ *nombre* si la borne *upBound* est un nombre, *liste* si la borne *upBound* est une liste

Calcule la probabilité qu'une variable suivant la loi géométrique prenne une valeur entre les bornes *lowBound* et *upBound* en fonction de la probabilité de réussite *p* spécifiée.

Pour $P(X \leq upBound)$, définissez *lowBound* = 1.

geomPdf()

geomPdf(*p*,*ValX*) $\Rightarrow$ *nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

Calcule la probabilité que le premier succès intervienne au rang *ValX*, pour la loi géométrique discrète en fonction de la probabilité de réussite *p* spécifiée.

Get

Menu hub

Get[*promptString*,]*var*[, *statusVar*]

Get[*promptString*,] *fonc*(*arg1*, ...*argn*)
[, *statusVar*]

Commande de programmation : récupère une valeur d'un hub connecté TI-Innovator™ Hub et affecte cette valeur à la variable *var*.

La valeur doit être demandée :

- À l'avance, par le biais d'une commande **Send "READ ..."** commande.
- ou —

Exemple : demander la valeur actuelle du capteur intégré du niveau de lumière du hub. Utilisez **Get** pour récupérer la valeur et l'affecter à la variable *lightval*.

Send "READ BRIGHTNESS"	Done
Get <i>lightval</i>	Done
<i>lightval</i>	0.347922

Incorporez la demande READ dans la commande **Get**.

Get "READ BRIGHTNESS" <i>lightval</i>	Done
<i>lightval</i>	0.378441

- En incorporant une demande "READ ..." comme l'argument facultatif de *promptString*. Cette méthode vous permet d'utiliser une seule commande pour demander la valeur et la récupérer.

Une simplification implicite a lieu. Par exemple, la réception de la chaîne de caractères "123" est interprétée comme étant une valeur numérique. Pour conserver la chaîne de caractères, utilisez **GetStr** au lieu de **Get**.

Si vous incluez l'argument facultatif *statusVar*, une valeur lui sera affectée en fonction de la réussite de l'opération. Une valeur zéro signifie qu'aucune donnée n'a été reçue.

Dans la deuxième syntaxe, l'argument *func()* permet à un programme de stocker la chaîne de caractères reçue comme étant la définition d'une fonction. Cette syntaxe équivaut à l'exécution par le programme de la commande suivante :

Define *func*(arg1, ...argn) = chaîne reçue

Le programme peut alors utiliser la fonction définie *func()*.

Remarque : vous pouvez utiliser la commande **Get** dans un programme défini par l'utilisateur, mais pas dans une fonction.

Remarque : Voir également **GetStr**, page 95 et **Send**, page 177.

getDenom(*Expr1*) $\Rightarrow$ *expression*

Transforme l'argument en une expression dotée d'un dénominateur commun réduit, puis en donne le numérateur.

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

getKey([0|1]) ⇒ returnString

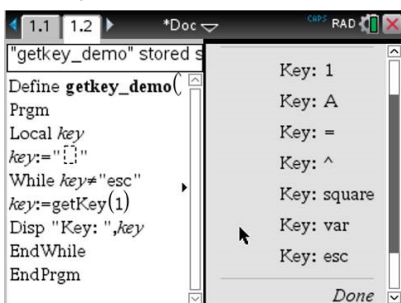
Description : **getKey()** - permet à un programme TI-Basic de recevoir des entrées de clavier - calculatrice, ordinateur de bureau et émulateur sur ordinateur de bureau.

Par exemple :

- `keypressed := getKey()` retournera une touche ou une chaîne vide si aucune touche n'a été pressée. Cet appel sera immédiatement retourné.
- `keypressed := getKey(1)` attendra l'appui sur une touche. Cet appel mettra en pause l'exécution du programme jusqu'à l'appui sur une touche.

getKey()

Par exemple :

**Traitement des frappes de touche :**

Touche de calculatrice/émulateur	Ordinateur	Valeur de retour
Échap	Échap	« échap »
Pavé tactile - Clic en haut	n/a	« haut »
On	n/a	« accueil »
Scratchpads	n/a	"scratchpad"
Pavé tactile - Clic gauche	n/a	« gauche »
Pavé tactile - Clic au centre	n/a	« centre »
Pavé tactile - Clic droit	n/a	« droite »
Classeur	n/a	« classeur »
Tab	Tab	« tab »
Pavé tactile - Clic en bas	Flèche bas	« bas »
Menu	n/a	« menu »

Touche de calculatrice/émulateur	Ordinateur	Valeur de retour
Ctrl	Ctrl	aucun retour
Maj	Maj	aucun retour
Var	n/a	« var »
Suppr	n/a	« suppr »
=	=	"="
trigonométrie	n/a	« trigonométrie »
0 à 9	0-9	« 0 » ... « 9 »
Modèles	n/a	« modèle »
Catalogue	n/a	« cat »
^	^	"^"
X^2	n/a	« carré »
/ (touche division)	/	"/"
* (touche multiplication)	*	"*"
e^x	n/a	« expr »
10^x	n/a	« puissance de 10 »
+	+	"+"
-	-	"_"
(	(	"("
)	)	")"
.	.	". "
(-)	n/a	« - » (signe moins)
Entrée	Entrée	« entrée »
ee	n/a	« E » (notation scientifique E)
a - z	a-z	alpha = lettre pressée (minuscule) ("a" - "z")
maj a-z	maj a-z	alpha = lettre pressée « A » - « Z »
		Note : ctrl-maj fonctionne pour le verrouillage des

Touche de calculatrice/émulateur	Ordinateur	Valeur de retour
		majuscules
?!	n/a	"?!"
pi	n/a	« pi »
Marque	n/a	aucun retour
,	,	" , "
Retour	n/a	Retour
Espace	Espace	« » (espace)
Inaccessible	Touches de caractères spéciaux tels que @,!,^, etc.	Le caractère est retourné
n/a	Touches de fonction	Aucun caractère retourné
n/a	Touches de commandes spéciales pour ordinateur	Aucun caractère retourné
Inaccessible	Autres touches pour ordinateur non disponibles sur la calculatrice lorsque getKey() est en attente d'une frappe. {, },,, :, ...)	Le même caractère que vous obtenez dans l'Éditeur mathématique (pas dans une boîte mathématique)

Remarque : Il est important de noter que la présence de **getKey()** dans un programme modifie la façon dont certains événements sont traités par le système. Certains sont décrits ci-dessous.

Arrête le programme et traite l'événement - Exactement comme si l'utilisateur quittait le programme en appuyant sur la touche **ON**

« **Support** » ci-dessous signifie - le système fonctionne comme prévu - le programme continue à être exécuté.

Événement	Unité nomade	Ordinateur - TI-Nspire™ Student Software
Questions rapides	Arrête le programme, traite l'événement	Comme avec l'unité nomade (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software-uniquelement)
Gestion des fichiers à distance	Arrête le programme, traite l'événement	Comme avec l'unité nomade. (TI-Nspire™ Student

Événement	Unité nomade	Ordinateur - TI-Nspire™ Student Software
(Incl. l'envoi du fichier « Exit Press 2 Test » d'une unité nomade à une autre ou à un ordinateur)		Software, TI-Nspire™ Navigator™ NC Teacher Software-uniquement)
Fermer la classe	Arrête le programme, traite l'événement	Support (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software-uniquement)

Événement	Unité nomade	Ordinateur - TI-Nspire™ Toutes les versions
TI-Innovator™ Hub connexion/déconnexion	Support - Peut émettre avec succès des commandes à TI- Innovator™ Hub. Après que vous ayez quitté le programme, le TI- Innovator™ Hub continue de travailler avec l'unité nomade.	Comme avec l'unité nomade

getLangInfo()

Catalogue > 

getLangInfo() ⇒ chaîne

getLangInfo()

"en"

Retourne une chaîne qui correspond au nom abrégé de la langue active. Vous pouvez, par exemple, l'utiliser dans un programme ou une fonction afin de déterminer la langue courante.

Anglais = « en »

Danois = « da »

Allemand = « de »

Finlandais = « fi »

Français = « fr »

Italien = « it »

Néerlandais = « nl »

Néerlandais belge = « nl_BE »

Norvégien = « no »
Portugais = « pt »
Espagnol = « es »
Suédois = « sv »

getLockInfo()

getLockInfo(Var)⇒valeur

Donne l'état de verrouillage/déverrouillage de la variable Var.

valeur =0 : Var est déverrouillée ou n'existe pas.

valeur =1 : Var est verrouillée et ne peut être ni modifiée ni supprimée.

Voir Lock, page 119 et unlock, page 221.

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()

getMode(EntierNomMode)⇒valeur

getMode(0)⇒liste

getMode(EntierNomMode) affiche une valeur représentant le réglage actuel du mode EntierNomMode.

getMode(0) affiche une liste contenant des paires de chiffres. Chaque paire consiste en un entier correspondant au mode et un entier correspondant au réglage.

Pour obtenir une liste des modes et de leurs réglages, reportez-vous au tableau ci-dessous.

getMode(0)	{ 1,7,2,1,3,1,4,1,5,1,6,1,7,1,8,1 }
getMode(1)	7
getMode(8)	1

Si vous enregistrez les réglages avec **getMode(0)** → *var*, vous pouvez utiliser **setMode(*var*)** dans une fonction ou un programme pour restaurer temporairement les réglages au sein de l'exécution de la fonction ou du programme uniquement. Voir également **setMode()**, page 182.

Nom du mode	Entier du mode	Entiers de réglage
Afficher chiffres	1	1=Flottant, 2=Flottant 1, 3=Flottant 2, 4=Flottant 3, 5=Flottant 4, 6=Flottant 5, 7=Flottant 6, 8=Flottant 7, 9=Flottant 8, 10=Flottant 9, 11=Flottant 10, 12=Flottant 11, 13=Flottant 12, 14=Fixe 0, 15=Fixe 1, 16=Fixe 2, 17=Fixe 3, 18=Fixe 4, 19=Fixe 5, 20=Fixe 6, 21=Fixe 7, 22=Fixe 8, 23=Fixe 9, 24=Fixe 10, 25=Fixe 11, 26=Fixe 12
Angle	2	1=Radian, 2=Degré, 3=Grade
Format Exponentiel	3	1=Normal, 2=Scientifique, 3=Ingénieur
Réel ou Complexe	4	1=Réel, 2=Rectangulaire, 3=Polaire
Auto ou Approché	5	1=Auto, 2=Approché, 3=Exact
Format Vecteur	6	1=Rectangulaire, 2=Cylindrique, 3=Sphérique
Base	7	1=Décimale, 2=Hexadécimale, 3=Binaire

getNum()

getNum(*Expr1*) ⇒ *expression*

Transforme l'argument en une expression dotée d'un dénominateur commun réduit, puis en donne le dénominateur.

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

GetStr

GetStr[*promptString*,] *var*[, *statusVar*]

Par exemple, voir **Get**.

GetStr*[promptString,] func(arg1, ...argn)*
[, statusVar]

Commande de programmation : fonctionne de manière identique à la commande **Get**, sauf que la valeur reçue est toujours interprétée comme étant une chaîne de caractères. En revanche, la commande **Get** interprète la réponse comme une expression, à moins que l'utilisateur ne la saisisse entre guillemets ("").

Remarque : Voir également **Get**, page 87 et **Send**, page 177.

getType()

Catalogue > 

getType(*var*)⇒chaîne de caractères

Retourne une chaîne de caractère qui indique le type de données de la variable *var*.

Si *var* n'a pas été définie, retourne la chaîne "AUCUNE".

$\{1,2,3\} \rightarrow temp$	$\{1,2,3\}$
<code>getType(temp)</code>	"LIST"
$3 \cdot i \rightarrow temp$	$3 \cdot i$
<code>getType(temp)</code>	"EXPR"
<code>DelVar temp</code>	Done
<code>getType(temp)</code>	"NONE"

getVarInfo()

Catalogue > 

getVarInfo()⇒matrice ou chaîne

getVarInfo

(*chaîneNomBibliothèque*)⇒matrice ou chaîne

getVarInfo() donne une matrice d'informations (nom et type de la variable, accès à la bibliothèque et état de verrouillage/déverrouillage) pour toutes les variables et objets de la bibliothèque définis dans l'activité courante.

Si aucune variable n'est définie, **getVarInfo**() donne la chaîne « NONE » (AUCUNE).

<code>getVarInfo()</code>	"NONE"												
Define $x=5$	Done												
Lock x	Done												
Define LibPriv $y=\{1,2,3\}$	Done												
Define LibPub $z(x)=3 \cdot x^2-x$	Done												
<code>getVarInfo()</code>	<table><tr><td>x</td><td>"NUM"</td><td>"{"</td><td>1</td></tr><tr><td>y</td><td>"LIST"</td><td>"LibPriv "</td><td>0</td></tr><tr><td>z</td><td>"FUNC"</td><td>"LibPub "</td><td>0</td></tr></table>	x	"NUM"	"{"	1	y	"LIST"	"LibPriv "	0	z	"FUNC"	"LibPub "	0
x	"NUM"	"{"	1										
y	"LIST"	"LibPriv "	0										
z	"FUNC"	"LibPub "	0										
<code>getVarInfo(<i>tmp3</i>)</code>	"Error: Argument must be a string"												
<code>getVarInfo("tmp3")</code>	<table><tr><td><i>volcyl2</i></td><td>"NONE"</td><td>"LibPub "</td><td>0</td></tr></table>	<i>volcyl2</i>	"NONE"	"LibPub "	0								
<i>volcyl2</i>	"NONE"	"LibPub "	0										

getVarInfo
(*chaîneNomBibliothèque*) donne une matrice d'informations pour tous les objets de bibliothèque définis dans la bibliothèque *chaîneNomBibliothèque*. *chaîneNomBibliothèque* doit être une chaîne (texte entre guillemets) ou une variable.

Si la bibliothèque *chaîneNomBibliothèque* n'existe pas, une erreur est générée.

Observez l'exemple de gauche dans lequel le résultat de **getVarInfo()** est affecté à la variable *vs*. La tentative d'afficher la ligne 2 ou 3 de *vs* génère un message d'erreur "Liste ou matrice invalide" car pour au moins un des éléments de ces lignes (variable *b*, par exemple) l'évaluation redonne une matrice.

Cette erreur peut également survenir lors de l'utilisation de *Ans* pour réévaluer un résultat de **getVarInfo()**.

Le système génère l'erreur ci-dessus car la version courante du logiciel ne prend pas en charge les structures de matrice généralisées dans lesquelles un élément de matrice peut être une matrice ou une liste.

<i>a</i> :=1	1												
<i>b</i> :=[1 2]	[1 2]												
<i>c</i> :=[1 3 7]	[1 3 7]												
<i>vs</i> :=getVarInfo()	<table><tr><td><i>a</i></td><td>"NUM"</td><td>"[]"</td><td>0</td></tr><tr><td><i>b</i></td><td>"MAT"</td><td>"[]"</td><td>0</td></tr><tr><td><i>c</i></td><td>"MAT"</td><td>"[]"</td><td>0</td></tr></table>	<i>a</i>	"NUM"	"[]"	0	<i>b</i>	"MAT"	"[]"	0	<i>c</i>	"MAT"	"[]"	0
<i>a</i>	"NUM"	"[]"	0										
<i>b</i>	"MAT"	"[]"	0										
<i>c</i>	"MAT"	"[]"	0										
<i>vs</i> [1]	[1 "NUM" "[]" 0]												
<i>vs</i> [1,1]	1												
<i>vs</i> [2]	"Error: Invalid list or matrix"												
<i>vs</i> [2,1]	[1 2]												

Goto *nomÉtiquette*

Transfère le contrôle du programme à l'étiquette *nomÉtiquette*.

nomÉtiquette doit être défini dans la même fonction à l'aide de l'instruction **Lbl**.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Define $g()$ =Func	Done
Local $temp,i$	
$0 \rightarrow temp$	
$1 \rightarrow i$	
Lbl top	
$temp+i \rightarrow temp$	
If $i < 10$ Then	
$i+1 \rightarrow i$	
Goto top	
EndIf	
Return $temp$	
EndFunc	
$g()$	55

►Grad

Expr1 ► Grad $\Rightarrow$ *expression*

Convertit *Expr1* en une mesure d'angle en grades.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant **@>Grad**.

En mode Angle en degrés :

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

En mode Angle en radians :

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

/

identity()**identity(Entier)** $\Rightarrow$ *matrice*

Donne la matrice unité de dimension *Entier*.

Entier doit être un entier positif

identity(4)	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
-------------	--

If**If** *BooleanExpr*
*Relevé***If** *BooleanExpr* **Then**
*Bloc***EndIf**

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

Si *BooleanExpr* est évalué à vrai, exécute l'instruction *Instruction* ou le bloc d'instructions *Bloc* avant de poursuivre l'exécution de la fonction

Si *BooleanExpr* est évalué à faux, poursuit l'exécution en ignorant l'instruction ou le bloc d'instructions

Bloc peut correspondre à une ou plusieurs instructions, séparées par le caractère « : »

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

If *BooleanExpr* **Then**
 Bloc1

Else
 Bloc2

Endif

Si *BooleanExpr* est évalué à vrai, exécute *Bloc1* et ignore *Bloc2*.

Si *BooleanExpr* est évalué à faux, ignore *Bloc1*, mais exécute *Bloc2*.

Bloc1 et *Bloc2* peuvent correspondre à une seule instruction.

If *BooleanExpr1* **Then**
 Bloc1

Elseif *BooleanExpr2* **Then**
 Bloc2

⋮

Elseif *BooleanExprN* **Then**
 BlocN

Endif

Permet de traiter les conditions multiples. Si *BooleanExpr1* est évalué à vrai, exécute *Bloc1* Si *BooleanExpr1* est évalué à faux, évalue *BooleanExpr2*, et ainsi de suite.

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

$g(12)$	12
$g(-12)$	12

Define $g(x)$ =Func	
If $x < 5$ Then	
Return 5	
ElseIf $x > 5$ and $x < 0$ Then	
Return $-x$	
ElseIf $x \geq 0$ and $x \neq 10$ Then	
Return x	
ElseIf $x = 10$ Then	
Return 3	
EndIf	
EndFunc	

	Done
$g(-4)$	4
$g(10)$	3

ifFn(*exprBooléenne*, *Valeur_si_Vrai* [, *Valeur_si_Faux* [, *Valeur_si_Inconnu*]]) $\Rightarrow$ *expression*, *liste* ou *matrice*

Evalue l'expression booléenne *exprBooléenne* (ou chacun des éléments de *exprBooléenne*) et produit un résultat reposant sur les règles suivantes

- *exprBooléenne* peut tester une valeur unique, une liste ou une matrice
- Si un élément de *exprBooléenne* est évalué à vrai, l'élément correspondant de *Valeur_si_Vrai* s'affiche
- Si un élément de *exprBooléenne* est évalué à faux, l'élément correspondant de *Valeur_si_Faux* s'affiche Si vous omettez *Valeur_si_Faux*, undef s'affiche.
- Si un élément de *exprBooléenne* n'est ni vrai ni faux, l'élément correspondant de *Valeur_si_Inconnu* s'affiche Si vous omettez *Valeur_si_Inconnu*, undef s'affiche
- Si le deuxième, troisième ou quatrième argument de la fonction **ifFn()** est une expression unique, le test booléen est appliqué à toutes les positions dans *exprBooléenne*

Remarque : si l'instruction simplifiée *exprBooléenne* implique une liste ou une matrice, tous les autres arguments de type liste ou matrice doivent avoir la ou les même(s) dimension(s) et le résultat aura la ou les même(s) dimension(s).

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

La valeur d'essai **1** est inférieure à 2,5, ainsi l'élément correspondant dans

Valeur_si_Vrai **5** est copié dans la liste de résultats.

La valeur d'essai **2** est inférieure à 2,5, ainsi l'élément correspondant dans

Valeur_si_Vrai **6** est copié dans la liste de résultats.

La valeur d'essai **3** n'est pas inférieure à 2,5, ainsi l'élément correspondant dans *Valeur_si_Faux* **10** est copié dans la liste de résultats

$\text{ifFn}(\{1,2,3\} < 2.5, 4, \{8,9,10\})$
 $\{4,4,10\}$

Valeur_si_Vrai est une valeur unique et correspond à n'importe quelle position sélectionnée

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

Valeur_si_Faux n'est pas spécifié Undef est utilisé.

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

Un élément sélectionné à partir de *Valeur_si_Vrai*. Un élément sélectionné à partir de *Valeur_si_Inconnu*.

imag(*ExprI*) $\Rightarrow$ *expression*

Donne la partie imaginaire de l'argument.

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

$\text{imag}(z)$
0

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

Remarque : Toutes les variables non affectées sont considérées comme réelles. Voir aussi `real()`, page 164

imag(ListeI) $\Rightarrow$ *liste*

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

Donne la liste des parties imaginaires des éléments.

imag(MatriceI) $\Rightarrow$ *matrice*

$$\text{imag}\left(\begin{bmatrix} a & b \\ i \cdot c & i \cdot d \end{bmatrix}\right) \quad \begin{bmatrix} 0 & 0 \\ c & d \end{bmatrix}$$

Donne la matrice des parties imaginaires des éléments.

impDif()

impDif(Équation, Var, dependVar [,Ord]) $\Rightarrow$ *expression*

$$\text{impDif}(x^2 + y^2 = 100, x, y) \quad \begin{array}{c} -x \\ y \end{array}$$

où la valeur par défaut de l'argument *Ord* est 1.

Calcule la dérivée implicite d'une équation dans laquelle une variable est définie implicitement par rapport à une autre.

Indirection
inString()

inString(srcString, subString[, Début])
 $\Rightarrow$ *entier*

$$\begin{array}{ll} \text{inString}(\text{"Hello there"}, \text{"the"}) & 7 \\ \text{inString}(\text{"ABCEFG"}, \text{"D"}) & 0 \end{array}$$

Donne le rang du caractère de la chaîne *chaîneSrce* où commence la première occurrence de *sousChaîne*.

Début, s'il est utilisé, indique le point de départ de la recherche dans *chaîneSrce*. Par défaut = 1, la recherche commence à partir du (premier caractère de *chaîneSrce*).

Si *chaîneSrce* ne contient pas *sousChaîne* ou si *Début* est strictement supérieur à la longueur de *ChaîneSrce*, on obtient zéro

int()

int(Expr) $\Rightarrow$ entier

int(List1) $\Rightarrow$ liste

int(Matrix1) $\Rightarrow$ matrice

$\text{int}(-2.5)$	-3.
$\text{int}(\begin{bmatrix} -1.234 & 0 & 0.37 \end{bmatrix})$	$\begin{bmatrix} -2. & 0 & 0. \end{bmatrix}$

Donne le plus grand entier inférieur ou égal à l'argument. Cette fonction est identique à **floor()** (partie entière).

L'argument peut être un nombre réel ou un nombre complexe.

Dans le cas d'une liste ou d'une matrice, donne la partie entière de chaque élément.

intDiv()

intDiv(Number1, Number2) $\Rightarrow$ entier

intDiv(List1, List2) $\Rightarrow$ liste

intDiv(Matrix1, Matrix2) $\Rightarrow$ matrice

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

Donne le quotient dans la division euclidienne de (*Nombre1* $\div$ *Nombre2*).

Dans le cas d'une liste ou d'une matrice, donne le quotient de (argument 1 $\div$ argument 2) pour chaque paire d'éléments.

intégrale

interpoler ()

interpoler(Valeurx, Listex, Listey, ListePrincy) $\Rightarrow$ list

Équation différentielle :
 $y' = -3 \cdot y + 6 \cdot t + 5$ et $y(0) = 5$

Cette fonction effectue l'opération suivante :

interpoler ()

Catalogue > 

Étant donné *Listex*, *Listey*=*f(Listex)* et *ListePrincy*=*f'(Listex)* pour une fonction *f* inconnue, une interpolation par une spline cubique est utilisée pour donner une approximation de la fonction *f* en *Valeurx*. On suppose que *Listex* est une liste croissante ou décroissante de nombres, cette fonction pouvant retourner une valeur même si ce n'est pas le cas. Elle examine la *Listex* et recherche un intervalle [*Listex*[*i*], *Listex*[*i*+1]] qui contient *Valeurx*. Si elle trouve cet intervalle, elle retourne une valeur d'interpolation pour *f(Valeurx)*, sinon elle donne **undef**.

Listex, *Listey*, et *ListePrincy* doivent être de même dimensions ≥ 2 et contenir des expressions pouvant être évaluées à des nombres.

Valeurx peut être une variable indéfinie, un nombre ou une liste de nombres.

```
rk:=rk23(-3*y+6*t+5,I,y,{0,10},5,1)
[0. 1. 2. 3. 4.
5. 3.19499 5.00394 6.99957 9.00593 10.]
```

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

Utilisez la fonction *interpolate()* pour calculer les valeurs de la fonction pour la liste *valeurx* :

```
xvaluelist:=seq(i,i,0,10,0.5)
{0,0.5,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,5.5,6.,6.5,7.,7.5,8.,8.5,9.,9.5,10.}
xlist:=mat▶list(rk[1])
{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.}
ylist:=mat▶list(rk[2])
{5.,3.19499,5.00394,6.99957,9.00593,10.99706,12.99901,15.00099,17.00298,19.00497,21.00696}
yprimelist:=-3*y+6*t+5=ylist and t=xlist
{-10.,1.41503,1.98819,2.00129,1.98221,2.00606,2.01094,2.01586,2.02078,2.0257,2.03062}
interpolate(xvaluelist,xlist,ylist,yprimelist)
{5.,2.67062,3.19499,4.02782,5.00394,6.00011,7.00001,8.00001,9.00001,10.00001,11.00001}
```

invχ²()

Catalogue > 

invχ²(Aire,df)

invChi2(Aire,df)

Calcule l'inverse de la fonction de répartition de la loi χ^2 (Chi2) de degré de liberté *df* en un point donné *Aire*.

invF()

Catalogue > 

invF(Aire,dfNumer,dfDenom)

invF(Zone,dfNumer,dfDenom)

Calcule l'inverse de la fonction de répartition de la loi F (Fisher) de paramètres spécifiée par *dfNumer* et *dfDenom* en un point donné *Aire*

invBinom

(CumulativeProb, NumTrials, Prob, OutputForm) $\Rightarrow$ scalaire ou matrice

Étant donné le nombre d'essais (NumTrials) et la probabilité de réussite de chaque essai (Prob), cette fonction renvoie le nombre minimal de réussites, k , tel que la probabilité cumulée de k réussites soit supérieure ou égale à une probabilité cumulée donnée (CumulativeProb).

OutputForm=0, affiche le résultat en tant que scalaire (par défaut).

OutputForm=1, affiche le résultat en tant que matrice.

Par exemple : Mary et Kevin jouent à un jeu de dés. Mary doit deviner le nombre maximal de fois où 6 apparaît dans 30 lancers. Si le nombre 6 apparaît autant de fois ou moins, Mary gagne. Par ailleurs, plus le nombre qu'elle devine est petit, plus ses gains sont élevés. Quel est le plus petit nombre que Mary peut deviner si elle veut que la probabilité du gain soit supérieure à 77 % ?

$\text{invBinom}\left(0.77, 30, \frac{1}{6}\right)$	6
$\text{invBinom}\left(0.77, 30, \frac{1}{6}, 1\right)$	$\begin{bmatrix} 5 & 0.616447 \\ 6 & 0.776537 \end{bmatrix}$

invBinomN()

invBinomN(CumulativeProb, Prob, NumSuccess, OutputForm) $\Rightarrow$ scalaire ou matrice

Étant donné la probabilité de réussite de chaque essai (Prob) et le nombre de réussites (NumSuccess), cette fonction renvoie le nombre minimal d'essais, N , tel que la probabilité cumulée de x réussites soit inférieure ou égale à une probabilité cumulée donnée (CumulativeProb).

OutputForm=0, affiche le résultat en tant que scalaire (par défaut).

OutputForm=1, affiche le résultat en tant que matrice.

Par exemple : Monique s'entraîne aux tirs au but au volley-ball. Elle sait par expérience que ses chances de marquer un but sont de 70 %. Elle prévoit de s'entraîner jusqu'à ce qu'elle marque 50 buts. Combien de tirs doit-elle tenter pour s'assurer que la probabilité de marquer au moins 50 buts est supérieure à 0,99 ?

$\text{invBinomN}(0.01, 0.7, 49)$	86
$\text{invBinomN}(0.01, 0.7, 49, 1)$	$\begin{bmatrix} 85 & 0.010451 \\ 86 & 0.00709 \end{bmatrix}$

invNorm()

invNorm(Aire, μ , σ)

Calcule l'inverse de la fonction de répartition de la loi normale de paramètres μ et σ en un point donné Aire.

invf(Aire,df)

Calcule les fractiles d'une loi de Student à df degrés de liberté pour une *Aire* donnée.

iPart()

iPart(Number) $\Rightarrow$ entier

iPart(List1) $\Rightarrow$ liste

iPart(Matrix1) $\Rightarrow$ matrice

Donne l'argument moins sa partie fractionnaire.

Dans le cas d'une liste ou d'une matrice, applique la fonction à chaque élément.

L'argument peut être un nombre réel ou un nombre complexe.

iPart (-1.234)	-1.
iPart $\left(\left\{\frac{3}{2}, -2.3, 7.003\right\}\right)$	$\{1, -2., 7.\}$

irr()

irr(CF0,CFList [,CFFreq]) $\Rightarrow$ valeur

Fonction financière permettant de calculer le taux interne de rentabilité d'un investissement.

MT0 correspond au mouvement de trésorerie initial à l'heure 0 ; il doit s'agir d'un nombre réel.

Liste MT est une liste des montants de mouvements de trésorerie après le mouvement de trésorerie initial MT0.

FréqMT est une liste facultative dans laquelle chaque élément indique la fréquence d'occurrence d'un montant de mouvement de trésorerie groupé (consécutif), correspondant à l'élément de *ListeMT*. La valeur par défaut est 1 ; si vous saisissez des valeurs, elles doivent être des entiers positifs < 10 000

Remarque : Voir également **mirr()**, page 129.

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

isPrime(Nombre) ⇒ *Expression booléenne constante*

Donne true ou false selon que *nombre* est ou n'est pas un entier naturel premier ≥ 2 , divisible uniquement par lui-même et 1.

Si *Nombre* dépasse 306 chiffres environ et n'a pas de diviseur ≤ 1021 , **isPrime(Nombre)** affiche un message d'erreur.

Si vous souhaitez uniquement déterminer si *Nombre* est un nombre premier, utilisez **isPrime()** et non **factor()**. Cette méthode est plus rapide, en particulier si *Nombre* n'est pas un nombre premier et si le deuxième facteur le plus grand comporte plus de cinq chiffres.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

isPrime(5)	true
isPrime(6)	false

Fonction permettant de trouver le nombre premier suivant un nombre spécifié :

Define nextprim(<i>n</i>)=Func	Done
Loop	
$n+1 \rightarrow n$	
If isPrime(<i>n</i>)	
Return <i>n</i>	
EndLoop	
EndFunc	
nextprim(7)	11

isVoid(Var) ⇒ *Expression booléenne constante*

isVoid(Expr) ⇒ *Expression booléenne constante*

isVoid(Var) ⇒ *liste d'expressions booléennes constantes*

Retourne true ou false pour indiquer si l'argument est un élément de type données vide.

Pour plus d'informations concernant les éléments vides, reportez-vous à . page 281.

$a := _$	$_$
isVoid(<i>a</i>)	true
isVoid({ 1,_,3 })	{ false,true,false }

Lbl		Catalogue > 
Lbl <i>nomÉtiquette</i>	Define $g()$ =Func	Done
Définit une étiquette en lui attribuant le nom <i>nomÉtiquette</i> dans une fonction.	Local <i>temp,i</i>	
	$0 \rightarrow temp$	
	$1 \rightarrow i$	
	Lbl <i>top</i>	
	$temp+i \rightarrow temp$	
	If $i < 10$ Then	
	$i+1 \rightarrow i$	
	Goto <i>top</i>	
	EndIf	
	Return <i>temp</i>	
	EndFunc	
	$g()$	55
Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.		

lcm()		Catalogue > 
lcm (<i>Nombre1, Nombre2</i>) $\Rightarrow$ <i>expression</i>	$lcm(6,9)$	18
lcm (<i>Liste1, Liste2</i>) $\Rightarrow$ <i>liste</i>	$lcm\left(\left\{\frac{1}{3}, -14, 16\right\}, \left\{\frac{2}{15}, 7, 5\right\}\right)$	$\left\{\frac{2}{3}, 14, 80\right\}$
lcm (<i>Matrice1, Matrice2</i>) $\Rightarrow$ <i>matrice</i>		
Donne le plus petit commun multiple des deux arguments. Le lcm de deux fractions correspond au lcm de leur numérateur divisé par le gcd de leur dénominateur. Le lcm de nombres fractionnaires en virgule flottante correspond à leur produit.		
Pour deux listes ou matrices, donne les plus petits communs multiples des éléments correspondants.		

left()		Catalogue > 
left (<i>chaîneSrc</i> [, <i>Nomb</i>]) $\Rightarrow$ <i>chaîne</i>	$left("Hello", 2)$	"He"

Donne la chaîne formée par les *Nomb* premiers caractères de la chaîne *chaîneSrce*.

Si *Nomb* est absent, on obtient *chaîneSrce*.

left(*Liste1* [, *Nomb*]) ⇒ *liste*

Donne la liste formée par les *Nomb* premiers éléments de *Liste1*.

Si *Nomb* est absent, on obtient *Liste1*.

left(*Comparaison*) ⇒ *expression*

Donne le membre de gauche d'une équation ou d'une inéquation.

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

left(x<3)	x
-----------	---

libShortcut()

libShortcut(*chaîneNomBibliothèque*, *chaîneNomRaccourci* [, *LibPrivFlag*]) ⇒ *liste de variables*

Crée un groupe de variables dans l'activité courante qui contient des références à tous les objets du classeur de bibliothèque spécifié *chaîneNomBibliothèque*. Ajoute également les membres du groupe au menu Variables. Vous pouvez ensuite faire référence à chaque objet en utilisant la *chaîneNomRaccourci* correspondante.

Définissez *LibPrivFlag*=0 pour exclure des objets de la bibliothèque privée (par défaut) et *LibPrivFlag*=1 pour inclure des objets de bibliothèque privée.

Pour copier un groupe de variables, reportez-vous à **CopyVar**, page 33. Pour supprimer un groupe de variables, reportez-vous à **DelVar**, page 54.

Cet exemple utilise un classeur de bibliothèque enregistré et rafraîchi **linalg2** qui contient les objets définis comme *clearmat*, *gauss1* et *gauss2*.

getVarInfo("linalg2")			
	<i>clearmat</i>	"FUNC"	"LibPub "
	<i>gauss1</i>	"PRGM"	"LibPriv "
	<i>gauss2</i>	"FUNC"	"LibPub "

libShortcut("linalg2", "la")	{ <i>la.clearmat</i> , <i>la.gauss2</i> }
------------------------------	---

libShortcut("linalg2", "la", 1)	{ <i>la.clearmat</i> , <i>la.gauss1</i> , <i>la.gauss2</i> }
---------------------------------	--

limit(*Expr1*, *Var*, *Point* [, *Direction*]) ⇒ *expression*

$$\lim_{x \rightarrow 5} (2 \cdot x + 3) \quad 13$$

limit(*Liste1*, *Var*, *Point* [, *Direction*]) ⇒ *liste*

$$\lim_{x \rightarrow 0^+} \left(\frac{1}{x} \right) \quad \infty$$

limit(*Matrice1*, *Var*, *Point* [, *Direction*]) ⇒ *matrice*

$$\lim_{x \rightarrow 0} \left(\frac{\sin(x)}{x} \right) \quad 1$$

Donne la limite recherchée.

$$\lim_{h \rightarrow 0} \left(\frac{\sin(x+h) - \sin(x)}{h} \right) \quad \cos(x)$$

Remarque : voir aussi **Modèle Limite**, page 7.

$$\lim_{n \rightarrow \infty} \left(\left(1 + \frac{1}{n} \right)^n \right) \quad e$$

Direction : négative=limite à gauche, positive=limite à droite, sinon=gauche et droite. (Si *Direction* est absent, la valeur par défaut est gauche et droite.)

Les limites en $+\infty$ et en $-\infty$ sont toujours converties en limites unilatérales.

Dans certains cas, **limit()** retourne lui-même ou undef (non défini) si aucune limite ne peut être déterminée. Cela ne signifie pas pour autant qu'aucune limite n'existe. undef signifie que le résultat est soit un nombre inconnu fini ou infini soit l'ensemble complet de ces nombres.

limit() utilisant des méthodes comme la règle de L'Hôpital, il existe des limites uniques que cette fonction ne permet pas de déterminer. Si *Expr1* contient des variables non définies autres que *Var*, il peut s'avérer nécessaire de les contraindre pour obtenir un résultat plus précis.

$$\lim_{x \rightarrow \infty} (a^x) \quad \text{undef}$$

$$\lim_{x \rightarrow \infty} (a^x) | a > 1 \quad \infty$$

$$\lim_{x \rightarrow \infty} (a^x) | a > 0 \text{ and } a < 1 \quad 0$$

Les limites peuvent être affectées par les erreurs d'arrondi. Dans la mesure du possible, n'utilisez pas le réglage Approché (Approximate) du mode **Auto** ou **Approché (Approximate)** ni des nombres approchés lors du calcul de limites. Sinon, les limites normalement nulles ou infinies risquent de ne pas l'être.

LinRegBx $X, Y, [Fr\acute{e}q], [Cat\acute{e}gorie, Inclure]$

Effectue l'ajustement linéaire $y = a + b \cdot x$ sur les listes X et Y en utilisant la fréquence $Fr\acute{e}q$. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et Y sont des listes de variables indépendantes et dépendantes.

$Fr\acute{e}q$ est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans $Fr\acute{e}q$ correspond à une fréquence d'occurrence pour chaque couple X et Y . Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples X et Y correspondants.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a + b \cdot x$
stat.a, stat.b	Coefficients d'ajustement
stat.r ²	Coefficient de détermination
stat.r	Coefficient de corrélation
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>

Variable de sortie	Description
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

LinRegMx

Catalogue > 

LinRegMx *X*, *Y*, [*Fréq*], [*Catégorie*, *Inclure*]

Effectue l'ajustement linéaire $y = m \cdot x + b$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et *Y* sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple *X* et *Y*. Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples *X* et *Y* correspondants.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $m \cdot x + b$
stat.m, stat.b	Coefficients d'ajustement
stat.r ²	Coefficient de détermination

Variable de sortie	Description
stat.r	Coefficient de corrélation
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

LinRegIntervals

Catalogue > 

LinRegIntervals $X, Y, F, [0, NivC]$

Pente. Calcule un intervalle de confiance de niveau C pour la pente.

LinRegIntervals $X, Y, F, [1, Xval, NivC]$

Réponse. Calcule une valeur y prévue, un intervalle de prévision de niveau C pour une seule observation et un intervalle de confiance de niveau C pour la réponse moyenne.

Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes.

X et Y sont des listes de variables indépendantes et dépendantes.

F est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans F spécifie la fréquence d'occurrence pour chaque couple X et Y . Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a+b \cdot x$
stat.a, stat.b	Coefficients d'ajustement
stat.df	Degrés de liberté
stat.r ²	Coefficient de détermination
stat.r	Coefficient de corrélation
stat.Resid	Valeurs résiduelles de l'ajustement

Pour les intervalles de type Slope uniquement

Variable de sortie	Description
[stat.CLower, stat.CUpper]	Intervalle de confiance de pente
stat.ME	Marge d'erreur de l'intervalle de confiance
stat.SESlope	Erreur type de pente
stat.s	Erreur type de ligne

Pour les intervalles de type Response uniquement

Variable de sortie	Description
[stat.CLower, stat.CUpper]	Intervalle de confiance pour une réponse moyenne
stat.ME	Marge d'erreur de l'intervalle de confiance
stat.SE	Erreur type de réponse moyenne
[stat.LowerPred, stat.UpperPred]	Intervalle de prévision pour une observation simple
stat.MEPred	Marge d'erreur de l'intervalle de prévision
stat.SEPred	Erreur type de prévision
stat. $\hat{y}$	$a + b \cdot \text{ValX}$

LinRegtTest

Catalogue > 

LinRegtTest $X, Y[, \text{Fréq}[, \text{Hypoth}]]$

Effectue l'ajustement linéaire sur les listes X et Y et un t -test sur la valeur de la pente β et le coefficient de corrélation ρ pour l'équation $y=\alpha+\beta x$. Il teste l'hypothèse nulle $H_0: \beta=0$ (équivalent, $\rho=0$) par rapport à l'une des trois hypothèses.

Toutes les listes doivent comporter le même nombre de lignes.

X et Y sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple X et Y . Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Hypoth est une valeur facultative qui spécifie une des trois hypothèses par rapport à laquelle l'hypothèse nulle ($H_0: \beta=\rho=0$) est testée.

Pour $H_a: \beta \neq 0$ et $\rho \neq 0$ (par défaut), définissez *Hypoth*=0

Pour $H_a: \beta < 0$ et $\rho < 0$, définissez *Hypoth*<0

Pour $H_a: \beta > 0$ et $\rho > 0$, définissez *Hypoth*>0

Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a + b \cdot x$
stat.t	t -Statistique pour le test de signification
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degrés de liberté
stat.a, stat.b	Coefficients d'ajustement

Variable de sortie	Description
stat.s	Erreur type de ligne
stat.SESlope	Erreur type de pente
stat.r ²	Coefficient de détermination
stat.r	Coefficient de corrélation
stat.Resid	Valeurs résiduelles de l'ajustement

linSolve()

Catalogue > 

linSolve(*SystèmeÉqLin*, *Var1*, *Var2*, ...)⇒*liste*

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

linSolve(*ÉqLin1* and *ÉqLin2* and ..., *Var1*, *Var2*, ...)⇒*liste*

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

linSolve(*{ÉqLin1, ÉqLin2, ...}*, *Var1*, *Var2*, ...)⇒*liste*

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

linSolve(*SystèmeÉqLin*, *{Var1, Var2, ...}*)⇒*liste*

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

linSolve(*ÉqLin1* and *ÉqLin2* and ..., *{Var1, Var2, ...}*)⇒*liste*

linSolve(*{ÉqLin1, ÉqLin2, ...}*, *{Var1, Var2, ...}*)⇒*liste*

Affiche une liste de solutions pour les variables *Var1*, *Var2*, etc.

Le premier argument doit être évalué à un système d'équations linéaires ou à une seule équation linéaire. Si tel n'est pas le cas, une erreur d'argument se produit.

Par exemple, le calcul de linSolve(x=1 et x=2,x) génère le résultat "Erreur d'argument".

ΔList()

Catalogue > 

ΔList(*Liste1*)⇒*liste*

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

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant `deltaList(...)`.

Donne la liste des différences entre les éléments consécutifs de *Liste1*. Chaque élément de *Liste1* est soustrait de l'élément suivant de *Liste1*. Le résultat comporte toujours un élément de moins que la liste *Liste1* initiale.

list►mat()

`list►mat(Liste [, élémentsParLigne])` ⇒ matrice

Donne une matrice construite ligne par ligne à partir des éléments de *Liste*.

Si *élémentsParLigne* est spécifié, donne le nombre d'éléments par ligne. La valeur par défaut correspond au nombre d'éléments de *Liste* (une ligne).

Si *Liste* ne comporte pas assez d'éléments pour la matrice, on complète par zéros.

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant `list@>mat(...)`.

<code>list►mat({ 1,2,3 })</code>	<table><tr><td>1</td><td>2</td><td>3</td></tr></table>	1	2	3			
1	2	3					
<code>list►mat({ 1,2,3,4,5 },2)</code>	<table><tr><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td></tr><tr><td>5</td><td>0</td></tr></table>	1	2	3	4	5	0
1	2						
3	4						
5	0						

►ln

`Expr ►ln` ⇒ expression

Convertit *Expr* en une expression contenant uniquement des logarithmes népériens (ln).

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

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant `@>ln`.

ln(Expr1)⇒expression $\ln(2.)$ 0.693147**ln(Liste1)⇒liste**

Donne le logarithme népérien de l'argument.

Dans le cas d'une liste, donne les logarithmes népériens de tous les éléments de celle-ci.

En mode Format complexe Réel :

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

En mode Format complexe Rectangulaire :

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

En mode Angle en radians et en mode Format complexe Rectangulaire :

$$\ln \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$$

1.83145+1.73485 <i>i</i>	0.009193-1.49086
0.448761-0.725533 <i>i</i>	1.06491+0.623491 <i>i</i>
-0.266891-2.08316 <i>i</i>	1.12436+1.79018 <i>i</i>

ln(matriceCarrée1)⇒matriceCarrée

Donne le logarithme népérien de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du logarithme népérien de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

LnReg**Catalogue** > **LnReg X, Y[, [Fréq] [, Catégorie, Inclure]]**

Effectue l'ajustement logarithmique $y = a + b \cdot \ln(x)$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et *Y* sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple X et Y . Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples X et Y correspondants.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a+b \cdot \ln(x)$
stat.a, stat.b	Coefficients d'ajustement
stat.r ²	Coefficient de détermination linéaire pour les données transformées
stat.r	Coefficient de corrélation pour les données transformées ($\ln(x)$, y)
stat.Resid	Valeurs résiduelles associées au modèle logarithmique
stat.ResidTrans	Valeurs résiduelles associées à l'ajustement linéaire des données transformées
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

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

Déclare les variables *vars* spécifiées comme variables locales. Ces variables existent seulement lors du calcul d'une fonction et sont supprimées une fois l'exécution de la fonction terminée.

Remarque : les variables locales contribuent à libérer de la mémoire dans la mesure où leur existence est temporaire. De même, elle n'interfère en rien avec les valeurs de variables globales existantes. Les variables locales s'utilisent dans les boucles **For** et pour enregistrer temporairement des valeurs dans les fonctions de plusieurs lignes dans la mesure où les modifications sur les variables globales ne sont pas autorisées dans une fonction.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

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
<i>rollcount</i> ()	16
<i>rollcount</i> ()	3

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

Verrouille les variables ou les groupes de variables spécifiés. Les variables verrouillées ne peuvent être ni modifiées ni supprimées.

Vous ne pouvez pas verrouiller ou déverrouiller la variable système *Ans*, de même que vous ne pouvez pas verrouiller les groupes de variables système *stat.* ou *tvm*.

Remarque : La commande **Verrouiller (Lock)** efface le contenu de l'historique Annuler/Rétablir lorsqu'elle est appliquée à des variables non verrouillées.

<i>a</i> :=65	65
Lock <i>a</i>	Done
getLockInfo(<i>a</i>)	1
<i>a</i> :=75	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
Unlock <i>a</i>	Done
<i>a</i> :=75	75
DelVar <i>a</i>	Done

Voir **unLock**, page 221 et **getLockInfo()**, page 94.

log()Touches ctrl 10^x

log(*Expr1* [, *Expr2*]) ⇒ *expression*

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

log(*Liste1* [, *Expr2*]) ⇒ *liste*

$$\log_4 (2.) \quad 0.5$$

Donne le logarithme de base *Expr2* de l'argument.

$$\log_3 (10) - \log_3 (5) \quad \log_3 (2)$$

Remarque : voir aussi **Modèle Logarithme**, page 2.

Dans le cas d'une liste, donne le logarithme de base *Expr2* des éléments.

En mode Format complexe Réel :

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

Si *Expr2* est omis, la valeur de base 10 par défaut est utilisée.

En mode Format complexe Rectangulaire :

$$\log_{10} (\{-3, 1.2, 5\}) \\ \left\{ \log_{10} (3) + 1.36438 - i, 0.079181, \log_{10} (5) \right\}$$

log(*matriceCarrée1* [, *Expr*]) ⇒ *matriceCarrée*

En mode Angle en radians et en mode Format complexe Rectangulaire :

$$\log_{10} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \\ \begin{bmatrix} 0.795387 + 0.753438 \cdot i & 0.003993 - 0.6474 \cdot i \\ 0.194895 - 0.315095 \cdot i & 0.462485 + 0.2707 \cdot i \\ -0.115909 - 0.904706 \cdot i & 0.488304 + 0.7774 \cdot i \end{bmatrix}$$

Donne le logarithme de base *Expr* de *matriceCarrée1*. Ce calcul est différent du calcul du logarithme de base *Expr* de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

Pour afficher le résultat entier, appuyez sur **↵**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

Si l'argument de base est omis, la valeur de base 10 par défaut est utilisée.

Expr1 ►logbase(*Expr2*)⇒*expression*

Provoque la simplification de l'expression entrée en une expression utilisant uniquement des logarithmes de base *Expr2*.

$$\log_3(10) - \log_5(5) \text{ ► logbase}(5) \quad \frac{\log_5\left(\frac{10}{3}\right)}{\log_5(3)}$$

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>logbase (...).

Logistic

Logistic *X*, *Y*, [*Fréq*] [, *Catégorie*, *Inclure*]]

Effectue l'ajustement logistique = $(c / (1 + a \cdot e^{-bx}))$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et *Y* sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple *X* et *Y*. Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples *X* et *Y* correspondants.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $c/(1+a \cdot e^{-bx})$
stat.a, stat.b, stat.c	Coefficients d'ajustement
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

LogisticD

Catalogue > 

LogisticD *X*, *Y* [, [*Itérations*], [*Fréq*] [, *Catégorie*, *Inclure*]]

Effectue l'ajustement logistique $y = (c/(1+a \cdot e^{-bx})+d)$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq* et un nombre spécifique d'*Itérations*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et *Y* sont des listes de variables indépendantes et dépendantes.

L'argument facultatif Itérations spécifie le nombre maximum d'itérations utilisées lors de ce calcul. Si *Itérations* est omis, la valeur par défaut 64 est utilisée. On obtient généralement une meilleure précision en choisissant une valeur élevée, mais cela augmente également le temps de calcul, et vice versa.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple X et Y . Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples X et Y correspondants.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $c/(1+a \cdot e^{-bx})+d$
stat.a, stat.b, stat.c, stat.d	Coefficients d'ajustement
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

Loop

Bloc

EndLoop

Exécute de façon itérative les instructions de *Bloc*. Notez que la boucle se répète indéfiniment, jusqu'à l'exécution d'une instruction **Goto** ou **Exit** à l'intérieur du *Bloc*.

Bloc correspond à une série d'instructions, séparées par un « : ».

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

```
Define rollcount()=Func
    Local i
    1 → i
    Loop
    If randInt(1,6)=randInt(1,6)
    Goto end
    i+1 → i
EndLoop
Lbl end
Return i
EndFunc
```

	Done
rollcount()	16
rollcount()	3

LU

LU *Matrice, lMatrice, uMatrice, pMatrice[,Tol]*

Calcule la décomposition LU (lower-upper) de Doolittle d'une matrice réelle ou complexe. La matrice triangulaire inférieure est stockée dans *lMatrice*, la matrice triangulaire supérieure dans *uMatrice* et la matrice de permutation (qui décrit les échange de lignes exécutés pendant le calcul) dans *pMatrice*.

$lMatrice \cdot uMatrice = pMatrice \cdot matrice$

L'argument facultatif Tol permet de considérer comme nul tout élément de la matrice dont la valeur absolue est inférieure à *Tol*. Cet argument n'est utilisé que si la matrice contient des nombres en virgule flottante et ne contient pas de variables symbolique sans valeur affectée. Dans le cas contraire, *Tol* est ignoré.

- Si vous utilisez   ou définissez

$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 6 & 12 & 18 \\ 5 & 14 & 31 \\ 3 & 8 & 18 \end{bmatrix}$
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	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

le mode **Auto ou Approché (Approximate)** sur Approché (Approximate), les calculs sont exécutés en virgule flottante.

- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit : $5E-14 \cdot \max(\dim(\text{Matrice})) \cdot \text{rowNorm}(\text{Matrice})$

L'algorithme de factorisation **LU** utilise la méthode du Pivot partiel avec échanges de lignes.

$\begin{bmatrix} m & n \\ o & p \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} m & n \\ o & p \end{bmatrix}$
LU m1,lower,upper,perm	Done
lower	$\begin{bmatrix} 1 & 0 \\ m & 1 \\ o & \end{bmatrix}$
upper	$\begin{bmatrix} o & p \\ 0 & n - \frac{m \cdot p}{o} \end{bmatrix}$
perm	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

M

mat►list()

mat►list(Matrice)⇒liste

Donne la liste obtenue en copiant les éléments de *Matrice* ligne par ligne.

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **mat@>list** (...).

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

max()

max(Expr1, Expr2)⇒expression

max(Liste1, Liste2)⇒liste

max(Matrice1, Matrice2)⇒matrice

Donne le maximum des deux arguments. Si les arguments sont deux listes ou matrices, donne la liste ou la matrice formée de la valeur maximale de chaque paire d'éléments correspondante.

max(Liste)⇒expression

Donne l'élément maximal de *liste*.

max(Matrice1)⇒matrice

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

max()

Catalogue > 

Donne un vecteur ligne contenant l'élément maximal de chaque colonne de la matrice *Matrice1*.

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

Remarque : voir aussi **fMax()** et **min()**.

mean()

Catalogue > 

mean(Liste[, listeFréq]) ⇒ *expression*

Donne la moyenne des éléments de *Liste*.

Chaque élément de la liste *listeFréq* totalise le nombre d'occurrences de l'élément correspondant de *Liste*.

mean(Matrice1[, matriceFréq])
⇒ *matrice*

Donne un vecteur ligne des moyennes de toutes les colonnes de *Matrice1*.

Chaque élément de *matriceFréq* totalise le nombre d'occurrences de l'élément correspondant de *Matrice1*.

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

$\text{mean}(\{0.2, 0.1, -0.3, 0.4\})$	0.26
$\text{mean}(\{1, 2, 3\}, \{3, 2, 1\})$	$\frac{5}{3}$

En mode Format Vecteur Rectangulaire :

$\text{mean}\left(\begin{bmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{bmatrix}\right)$	$\begin{bmatrix} -0.133333 & 0.833333 \end{bmatrix}$
$\text{mean}\left(\begin{bmatrix} \frac{1}{5} & 0 \\ -1 & 3 \\ \frac{2}{5} & \frac{-1}{2} \end{bmatrix}\right)$	$\begin{bmatrix} \frac{-2}{15} & \frac{5}{6} \end{bmatrix}$
$\text{mean}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \begin{bmatrix} 5 & 3 \\ 4 & 1 \\ 6 & 2 \end{bmatrix}\right)$	$\begin{bmatrix} \frac{47}{15} & \frac{11}{3} \end{bmatrix}$

median()

Catalogue > 

median(Liste[, listeFréq]) ⇒ *expression*

Donne la médiane des éléments de *Liste*.

Chaque élément de la liste *listeFréq* totalise le nombre d'occurrences de l'élément correspondant de *Liste*.

$\text{median}(\{0.2, 0.1, -0.3, 0.4\})$	0.2
--	-----

median(*MatriceI* [, *matriceFréq*]) ⇒ *matrice*

$$\text{median} \left(\begin{bmatrix} 0.2 & 0 \\ 1 & -0.3 \\ 0.4 & -0.5 \end{bmatrix} \right) = \begin{bmatrix} 0.4 & -0.3 \end{bmatrix}$$

Donne un vecteur ligne contenant les médianes des colonnes de *MatriceI*.

Chaque élément de *matriceFréq* totalise le nombre d'occurrences consécutives de l'élément correspondant de *MatriceI*.

Remarques :

- tous les éléments de la liste ou de la matrice doivent correspondre à des valeurs numériques.
- Les éléments vides de la liste ou de la matrice sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

MedMed

MedMed *X*, *Y* [, *Fréq*] [, *Catégorie*, *Inclure*]

Calcule la ligne Med-Medy = (m · x + b) sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et *Y* sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple *X* et *Y*. Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0.

Catégorie est une liste de codes de catégories pour les couples *X* et *Y* correspondants..

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à “Éléments vides”, page 281.

Variable de sortie	Description
stat.RegEqn	Équation de ligne Med-Med : $m \cdot x + b$
stat.m, stat.b	Coefficient de modèle
stat.Resid	Valeurs résiduelles de la ligne Med-Med
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

mid()

mid(*chaîneSrce*, *Début* [, *Nbre*])⇒*chaîne*

Donne la portion de chaîne de *Nbre* de caractères extraite de la chaîne *chaîneSrce*, en commençant au numéro de caractère *Début*.

Si *Nbre* est omis ou s'il dépasse le nombre de caractères de la chaîne *chaîneSrce*, on obtient tous les caractères de *chaîneSrce*, compris entre le numéro de caractère *Début* et le dernier caractère.

Nbre doit être ≥ 0 . Si *Nbre* = 0, on obtient une chaîne vide.

mid(*listeSource*, *Début* [, *Nbre*])⇒*liste*

Donne la liste de *Nbre* d'éléments extraits de *listeSource*, en commençant à l'élément numéro *Début*.

mid("Hello there",2)	"ello there"
mid("Hello there",7,3)	"the"
mid("Hello there",1,5)	"Hello"
mid("Hello there",1,0)	" "

mid({9,8,7,6},3)	{7,6}
mid({9,8,7,6},2,2)	{8,7}
mid({9,8,7,6},1,2)	{9,8}
mid({9,8,7,6},1,0)	{ }

mid()Catalogue > 

Si *Nbre* est omis ou s'il dépasse le nombre d'éléments de la liste *listeSource*, on obtient tous les éléments de *listeSource*, compris entre l'élément numéro *Début* et le dernier élément.

Nbre doit être ≥ 0 . Si *Nbre* = 0, on obtient une liste vide.

mid(*listeChaînesSource*, *Début*[, *Nbre*]) $\Rightarrow$ *liste*

Donne la liste de *Nbre* de chaînes extraites de la liste *listeChaînesSource*, en commençant par l'élément numéro *Début*.

$\text{mid}(\{ "A", "B", "C", "D" \}, 2, 2)$	$\{ "B", "C" \}$
--	------------------

min()Catalogue > 

min(*Expr1*, *Expr2*) $\Rightarrow$ *expression*

min(*Liste1*, *Liste2*) $\Rightarrow$ *liste*

min(*Matrice1*, *Matrice2*) $\Rightarrow$ *matrice*

Donne le minimum des deux arguments.
Si les arguments sont deux listes ou matrices, donne la liste ou la matrice formée de la valeur minimale de chaque paire d'éléments correspondante.

min(*Liste*) $\Rightarrow$ *expression*

Donne l'élément minimal de *Liste*.

min(*Matrice1*) $\Rightarrow$ *matrice*

Donne un vecteur ligne contenant l'élément minimal de chaque colonne de la matrice *Matrice1*.

Remarque : voir aussi **fMin()** et **max()**.

$\text{min}(2.3, 1.4)$	1.4
$\text{min}(\{1, 2\}, \{-4, 3\})$	$\{-4, 2\}$

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

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

mirr()Catalogue > 

mirr
(*tauxFinancement*,
tauxRéinvestissement, *MT0*, *ListeMT*
[, *FréqMT*]) $\Rightarrow$ *expression*

$\text{list1} := \{6000, -8000, 2000, -3000\}$	$\{6000, -8000, 2000, -3000\}$
$\text{list2} := \{2, 2, 2, 1\}$	$\{2, 2, 2, 1\}$
$\text{mirr}(4.65, 12, 5000, \text{list1}, \text{list2})$	13.41608607

Fonction financière permettant d'obtenir le taux interne de rentabilité modifié d'un investissement.

tauxFinancement correspond au taux d'intérêt que vous payez sur les montants de mouvements de trésorerie.

tauxRéinvestissement est le taux d'intérêt auquel les mouvements de trésorerie sont réinvestis.

MT0 correspond au mouvement de trésorerie initial à l'heure 0 ; il doit s'agir d'un nombre réel.

Liste MT est une liste des montants de mouvements de trésorerie après le mouvement de trésorerie initial *MT0*.

FréqMT est une liste facultative dans laquelle chaque élément indique la fréquence d'occurrence d'un montant de mouvement de trésorerie groupé (consécutif), correspondant à l'élément de *ListeMT*. La valeur par défaut est 1 ; si vous saisissez des valeurs, elles doivent être des entiers positifs < 10 000.

Remarque : voir également *irr()*, page 105.

mod()

mod(Exp1, Exp2)⇒expression

mod(7,0)	7
----------	---

mod(Liste1, List2)⇒liste

mod(7,3)	1
----------	---

mod(Matrice1, Matrice2)⇒matrice

mod(-7,3)	2
-----------	---

Donne le premier argument modulo le deuxième argument, défini par les identités suivantes :

mod(7,-3)	-2
-----------	----

mod(-7,-3)	-1
------------	----

mod({ 12,-14,16 }, { 9,7,-5 })	{ 3,0,-4 }
--------------------------------	------------

$\text{mod}(x,0) = x$

$\text{mod}(x,y) = x - \text{floor}(x/y) \cdot y$

mod()

Catalogue > 

Lorsque le deuxième argument correspond à une valeur non nulle, le résultat est de période dans cet argument. Le résultat est soit zéro soit une valeur de même signe que le deuxième argument.

Si les arguments sont deux listes ou deux matrices, on obtient une liste ou une matrice contenant la congruence de chaque paire d'éléments correspondante.

Remarque : voir aussi **remain()**, page 167

mRow()

Catalogue > 

mRow(*Expr*, *Matrice1*, *Index*) $\Rightarrow$ *matrice*

Donne une copie de *Matrice1* obtenue en multipliant chaque élément de la ligne *Index* de *Matrice1* par *Expr*.

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

mRowAdd()

Catalogue > 

mRowAdd(*Expr*, *Matrice1*, *Index1*, *Index2*) $\Rightarrow$ *matrice*

Donne une copie de *Matrice1* obtenue en remplaçant chaque élément de la ligne *Index2* de *Matrice1* par :

$$\text{Expr} \times \text{ligne Index1} + \text{ligne Index2}$$

$$\begin{array}{l} \text{mRowAdd}\left(-3, \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, 1, 2\right) \quad \begin{bmatrix} 1 & 2 \\ 0 & -2 \end{bmatrix} \\ \text{mRowAdd}\left(n, \begin{bmatrix} a & b \\ c & d \end{bmatrix}, 1, 2\right) \quad \begin{bmatrix} a & b \\ a \cdot n + c & b \cdot n + d \end{bmatrix} \end{array}$$

MultReg

Catalogue > 

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

Calcule la régression linéaire multiple de la liste *Y* sur les listes *X1*, *X2*, ..., *X10*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à “Éléments vides”, page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat.b0, stat.b1, ...	Coefficients d'ajustement
stat.R ²	Coefficient de détermination multiple
stat. $\hat{y}$ Liste	$\hat{y}_{\text{Liste}} = b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Valeurs résiduelles de l'ajustement

MultRegIntervals

MultRegIntervals *Y, X1[,X2[,X3,...[,X10]]],listeValX[,CLevel]*

Calcule une valeur y prévue, un intervalle de prévision de niveau C pour une seule observation et un intervalle de confiance de niveau C pour la réponse moyenne.

Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à “Éléments vides”, page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat. $\hat{y}$	Prévision d'un point : $\hat{y} = b_0 + b_1 \cdot x_1 + \dots$ pour <i>listeValX</i>
stat.dfError	Degrés de liberté des erreurs
stat.CLower, stat.CUpper	Intervalle de confiance pour une réponse moyenne
stat.ME	Marge d'erreur de l'intervalle de confiance
stat.SE	Erreur type de réponse moyenne

Variable de sortie	Description
stat.LowerPred, stat.UpperrPred	Intervalle de prévision pour une observation simple
stat.MEPred	Marge d'erreur de l'intervalle de prévision
stat.SEPred	Erreur type de prévision
stat.bList	Liste de coefficients de régression, {b0,b1,b2,...}
stat.Resid	Valeurs résiduelles de l'ajustement

MultRegTests

Catalogue > 

MultRegTests $Y, X1[,X2[,X3,...[,X10]]]$

Le test de régression linéaire multiple calcule une régression linéaire multiple sur les données et donne les statistiques du F -test et du t -test globaux pour les coefficients.

Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à “Éléments vides”, page 281.

Sorties

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat.F	Statistique du F -test global
stat.PVal	Valeur P associée à l'analyse statistique F globale
stat.R ²	Coefficient de détermination multiple
stat.AdjR ²	Coefficient ajusté de détermination multiple
stat.s	Écart-type de l'erreur
stat.DW	Statistique de Durbin-Watson ; sert à déterminer si la corrélation automatique de premier ordre est présente dans le modèle
stat.dfReg	Degrés de liberté de la régression
stat.SSReg	Somme des carrés de la régression
stat.MSReg	Moyenne des carrés de la régression

Variable de sortie	Description
stat.dfError	Degrés de liberté des erreurs
stat.SSError	Somme des carrés des erreurs
stat.MSError	Moyenne des carrés des erreurs
stat.bList	{b0,b1,...} Liste de coefficients
stat.tList	Liste des statistiques t pour chaque coefficient dans la liste bList
stat.PList	Liste des valeurs p pour chaque statistique t
stat.SEList	Liste des erreurs type des coefficients de la liste bList
stat.ŷListe	$\hat{y}Liste = b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Valeurs résiduelles de l'ajustement
stat.sResid	Valeurs résiduelles normalisées ; valeur obtenue en divisant une valeur résiduelle par son écart-type
stat.CookDist	Distance de Cook ; Mesure de l'influence d'une observation basée sur la valeur résiduelle et le levier
stat.Leverage	Mesure de la distance séparant les valeurs de la variable indépendante de leurs valeurs moyennes

N

nand	touches ctrl =	
<i>BooleanExpr1 nand BooleanExpr2</i> renvoie <i>expression booléenne</i>	$x \geq 3 \text{ and } x \geq 4$	$x \geq 4$
	$x \geq 3 \text{ nand } x \geq 4$	$x < 4$
<i>BooleanList1 nand BooleanList2</i> renvoie <i>liste booléenne</i>		
<i>BooleanMatrix1 nand BooleanMatrix2</i> renvoie <i>matrice booléenne</i>		

Renvoie la négation d'une opération logique **and** sur les deux arguments. Renvoie true (vrai) ou false (faux) ou une forme simplifiée de l'équation.

Pour les listes et matrices, renvoie le résultat des comparaisons, élément par élément.

Integer1 nand Integer2 ⇒ *entier*

Compare les représentations binaires de deux entiers en appliquant une opération **nand**. En interne, les deux entiers sont convertis en nombres binaires 64 bits signés. Lorsque les bits comparés correspondent, le résultat est 0 si dans les deux cas il s'agit d'un bit 1 ; dans les autres cas, le résultat est 1. La valeur donnée représente le résultat des bits et elle est affichée selon le mode de base utilisé.

Les entiers peuvent être entrés dans tout type de base. Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h. Tout entier sans préfixe est considéré comme un nombre en écriture décimale (base 10).

3 and 4	0
3 nand 4	-1
$\{1,2,3\}$ and $\{3,2,1\}$	$\{1,2,1\}$
$\{1,2,3\}$ nand $\{3,2,1\}$	$\{-2,-3,-2\}$

nCr()

Catalogue >

nCr(*Expr1*, *Expr2*) ⇒ *expression*

Pour les expressions *Expr1* et *Expr2* avec *Expr1* ≥ *Expr2* ≥ 0, **nCr()** donne le nombre de combinaisons de *Expr1* éléments pris parmi *Expr2* éléments. (Appelé aussi « coefficient binomial ».) Les deux arguments peuvent être des entiers ou des expressions symboliques.

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

nCr(*Expr*, 0) ⇒ 1

nCr(*Expr*, *entierNég*) ⇒ 0

nCr(*Expr*, *entierPos*) ⇒ *Expr* · (*Expr* - 1) ... (*Expr* - *entierPos* + 1) / *entierPos*!

nCr(*Expr*, *nonEntier*) ⇒ *expression*! / ((*Expr* - *nonEntier*)! · *nonEntier*!)

nCr(*Liste1*, *Liste2*) ⇒ *liste*

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

nCr()Catalogue > 

Donne une liste de combinaisons basées sur les paires d'éléments correspondantes dans les deux listes. Les arguments doivent être des listes comportant le même nombre d'éléments.

nCr(*Matrice1*, *Matrice2*) $\Rightarrow$ *matrice*

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

Donne une matrice de combinaisons basées sur les paires d'éléments correspondantes dans les deux matrices. Les arguments doivent être des matrices comportant le même nombre d'éléments.

nDerivative()Catalogue > 

nDerivative(*Expr1*, *Var*=*Valeur* [, *Ordre*]) $\Rightarrow$ *valeur*

$\text{nDerivative}(x , x=1)$	1
--------------------------------	---

nDerivative(*Expr1*, *Var* [, *Ordre*]) | *Var*=*Valeur* $\Rightarrow$ *valeur*

$\text{nDerivative}(x , x) _{x=0}$	undef
-------------------------------------	-------

$\text{nDerivative}(\sqrt{x-1}, x) _{x=1}$	undef
--	-------

Affiche la dérivée numérique calculée avec les méthodes de différenciation automatique.

Quand la *valeur* est spécifiée, celle-ci prévaut sur toute affectation de variable ou substitution précédente de type « | » pour la variable.

L'ordre de la dérivée doit être **1** ou **2**.

newList()Catalogue > 

newList(*nbreÉléments*) $\Rightarrow$ *liste*

$\text{newList}(4)$	$\{0, 0, 0, 0\}$
---------------------	------------------

Donne une liste de dimension *nbreÉléments*. Tous les éléments sont nuls.

newMat()Catalogue > 

newMat(*nbreLignes*, *nbreColonnes*) $\Rightarrow$ *matrice*

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

Donne une matrice nulle de dimensions
nbreLignes, *nbreColonnes*.

nfMax()

nfMax(*Expr*, *Var*) $\Rightarrow$ valeur

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

nfMax(*Expr*, *Var*, *LimitInf*) $\Rightarrow$ valeur

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

nfMax(*Expr*, *Var*, *LimitInf*,
LimitSup) $\Rightarrow$ valeur

nfMax(*Expr*, *Var*) | *LimitInf* $\leq$ *Var*
 $\leq$ *LimitSup* $\Rightarrow$ valeur

Donne la valeur numérique possible de
la variable *Var* au point où le maximum
local de *Expr* survient.

Si *LimitInf* et *LimitSup* sont spécifiés, la
fonction recherche le maximum local
dans l'intervalle fermé
[*LimitInf*, *LimitSup*].

Remarque : voir aussi **fMax()** et **d()**.

nfMin()

nfMin(*Expr*, *Var*) $\Rightarrow$ valeur

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

nfMin(*Expr*, *Var*, *LimitInf*) $\Rightarrow$ valeur

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

nfMin(*Expr*, *Var*, *LimitInf*,
LimitSup) $\Rightarrow$ valeur

nfMin(*Expr*, *Var*) | *LimitInf* $\leq$ *Var*
 $\leq$ *LimitSup* $\Rightarrow$ valeur

Donne la valeur numérique possible de
la variable *Var* au point où le minimum
local de *Expr* survient.

Si *LimitInf* et *LimitSup* sont spécifiés, la
fonction recherche le minimum local
dans l'intervalle fermé
[*LimitInf*, *LimitSup*].

Remarque : voir aussi **fMin()** et **d()**.

nInt(*Expr1*, *Var*, *Borne1*,
Borne2) $\Rightarrow$ expression

$$\text{nInt}\left(e^{-x^2}, x, -1, 1\right) \quad 1.49365$$

Si l'intégrande *Expr1* ne contient pas d'autre variable que *Var* et si *Borne1* et *Borne2* sont des constantes, en $+\infty$ ou en $-\infty$, alors **nInt()** donne le calcul approché de $\int(\text{Expr1}, \text{Var}, \text{Borne1}, \text{Borne2})$. Cette approximation correspond à une moyenne pondérée de certaines valeurs d'échantillon de l'intégrande dans l'intervalle $\text{Borne1} < \text{Var} < \text{Borne2}$.

L'objectif est d'atteindre une précision de six chiffres significatifs. L'algorithme s'adaptant, met un terme au calcul lorsqu'il semble avoir atteint cet objectif ou lorsqu'il paraît improbable que des échantillons supplémentaires produiront une amélioration notable.

Le message « Précision incertaine » s'affiche lorsque cet objectif ne semble pas atteint.

Il est possible de calculer une intégrale multiple en imbriquant plusieurs appels **nInt()**. Les bornes d'intégration peuvent dépendre des variables d'intégration les plus extérieures.

Remarque : voir aussi $\int()$, page 247.

$$\text{nInt}\left(\cos(x), x, \pi, \pi + 1.E-12\right) \quad -1.04144E-12$$

$$\int_{-\pi}^{\pi + 10^{-12}} \cos(x) dx \quad -\sin\left(\frac{1}{1000000000000}\right)$$

$$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x \cdot y}}{\sqrt{x^2 - y^2}}, y, -x, x\right), x, 0, 1\right) \quad 3.30423$$

nom(*tauxEffectif*, *CpY*) $\Rightarrow$ valeur

$$\text{nom}(5.90398, 12) \quad 5.75$$

Fonction financière permettant de convertir le taux d'intérêt effectif *tauxEffectif* à un taux annuel nominal, *CpY* étant le nombre de périodes de calcul par an.

tauxEffectif doit être un nombre réel et *CpY* doit être un nombre réel > 0 .

Remarque : voir également **eff()**, page 65.

BooleanExpr1 **nor** *BooleanExpr2*
renvoie *expression booléenne*

$x \geq 3$ or $x \geq 4$	$x \geq 3$
$x \geq 3$ nor $x \geq 4$	$x < 3$

BooleanList1 **nor** *BooleanList2* renvoie
liste booléenne

BooleanMatrix1 **nor** *BooleanMatrix2*
renvoie *matrice booléenne*

Renvoie la négation d'une opération logique **or** sur les deux arguments. Renvoie true (vrai) ou false (faux) ou une forme simplifiée de l'équation.

Pour les listes et matrices, renvoie le résultat des comparaisons, élément par élément.

Integer1 **nor** *Integer2* $\Rightarrow$ entier

Compare les représentations binaires de deux entiers en appliquant une opération **nor**. En interne, les deux entiers sont convertis en nombres binaires 64 bits signés. Lorsque les bits comparés correspondent, le résultat est 1 si dans les deux cas il s'agit d'un bit 1 ; dans les autres cas, le résultat est 0. La valeur donnée représente le résultat des bits et elle est affichée selon le mode de base utilisé.

3 or 4	7
3 nor 4	-8
$\{1,2,3\}$ or $\{3,2,1\}$	$\{3,2,3\}$
$\{1,2,3\}$ nor $\{3,2,1\}$	$\{-4,-3,-4\}$

Les entiers peuvent être entrés dans tout type de base. Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h. Tout entier sans préfixe est considéré comme un nombre en écriture décimale (base 10).

norm()Catalogue > **norm**(*Matrice*) $\Rightarrow$ *expression***norm**(*Vecteur*) $\Rightarrow$ *expression*

Donne la norme de Frobenius.

$\text{norm}\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right)$	$\sqrt{a^2+b^2+c^2+d^2}$
$\text{norm}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\right)$	$\sqrt{30}$
$\text{norm}\left(\begin{bmatrix} 1 & 2 \end{bmatrix}\right)$	$\sqrt{5}$
$\text{norm}\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$	$\sqrt{5}$

normalLine()Catalogue > **normalLine****(Expr1,Var,Point)** $\Rightarrow$ *expression***normalLine****(Expr1,Var=Point)** $\Rightarrow$ *expression*

Donne la normale à la courbe représentée par *Expr1* au point spécifié par *Var=Point*.

$\text{normalLine}(x^2,x,1)$	$\frac{3}{2} \frac{x}{2}$
$\text{normalLine}((x-3)^2-4,x,3)$	$x=3$
$\text{normalLine}\left(\frac{1}{x^3},x=0\right)$	0
$\text{normalLine}(\sqrt{ x },x=0)$	undef

Assurez-vous de ne pas avoir affecté une valeur à la variable indépendante. Par exemple, si $f1(x):=5$ et $x:=3$, alors **normalLine**($f1(x),x,2$) retourne « faux ».

normCdf()Catalogue > **normCdf**(*lowBound,upBound* [, μ

[, σ]) $\Rightarrow$ *nombre* si *lowBound* et *upBound* sont des nombres, *liste* si *lowBound* et *upBound* sont des listes

Calcule la probabilité qu'une variable suivant la loi normale de moyenne (*m*, valeur par défaut =0) et d'écart-type (*sigma*, valeur par défaut = 1) prenne des valeurs entre les bornes *lowBound* et *upBound*.

Pour $P(X \leq \text{upBound})$, définissez *lowBound* = $-\infty$.

normPdf()Catalogue > 

normPdf(*ValX* [, μ [, σ]]) $\Rightarrow$ *nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

Calcule la densité de probabilité de la loi normale à la valeur *ValX* spécifiée pour les paramètres μ et σ .

notCatalogue >

not *Expr booléenne* $\Rightarrow$ *Expression booléenne*

Donne true (vrai) ou false (faux) ou une forme simplifiée de l'argument.

not *Entier* $\Rightarrow$ *entier*

Donne le complément à 1 d'un entier. En interne, *Entier*1 est converti en nombre binaire 64 bits signé. La valeur de chaque bit est inversée (0 devient 1, et vice versa) pour le complément à 1. Le résultat est affiché en fonction du mode Base utilisé.

Les entiers de tout type de base sont admis. Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h. Tout entier sans préfixe est considéré comme un nombre en écriture décimale (base 10).

Si vous entrez un nombre dont le codage binaire signé dépasse 64 bits, il est ramené à l'aide d'une congruence dans la plage appropriée. Pour de plus amples informations, voir **Base2**, page 19.

$\text{not}(2 \geq 3)$	true
$\text{not}(x < 2)$	$x \geq 2$
not not innocent	<i>innocent</i>

En mode base Hex :

Important : utilisez le chiffre zéro et pas la lettre O.

$\text{not } 0\text{h}7\text{AC}36$	$0\text{hFFFFFFF}853\text{C}9$
-------------------------------------	--------------------------------

En mode base Bin :

$0\text{b}100101 \blacktriangleright \text{Base}10$	37
$\text{not } 0\text{b}100101$	$0\text{b}11111111111111111111111111111111 \blacktriangleright$
$\text{not } 0\text{b}100101 \blacktriangleright \text{Base}10$	-38

Pour afficher le résultat entier, appuyez sur $\blacktriangle$, puis utilisez les touches $\blacktriangleleft$ et $\blacktriangleright$ pour déplacer le curseur.

Remarque : une entrée binaire peut comporter jusqu'à 64 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 16 chiffres.

nPr()Catalogue >

nPr(*Expr1*, *Expr2*) $\Rightarrow$ *expression*

Pour les expressions *Expr1* et *Expr2* avec $\text{Expr1} \geq \text{Expr2} \geq 0$, **nPr()** donne le nombre de permutations de *Expr1* éléments pris parmi *Expr2* éléments. Les deux arguments peuvent être des entiers ou des expressions symboliques.

nPr(*Expr*, 0) $\Rightarrow$ 1

$\text{nPr}(z, 3)$	$z \cdot (z-2) \cdot (z-1)$
$\text{Ans} z=5$	60
$\text{nPr}(z, -3)$	$\frac{1}{(z+1) \cdot (z+2) \cdot (z+3)}$
$\text{nPr}(z, c)$	$\frac{z!}{(z-c)!}$
$\text{Ans} \cdot \text{nPr}(z-c, -c)$	1

$\text{nPr}(\text{Expr}, \text{entierNég}) \Rightarrow 1 / ((\text{Expr}+1) \cdot (\text{Expr}+2) \dots (\text{expression} - \text{entierNég}))$

$\text{nPr}(\text{Expr}, \text{entierPos}) \Rightarrow \text{Expr} \cdot (\text{Expr}-1) \dots (\text{Expr} - \text{entierPos} + 1)$

$\text{nPr}(\text{Expr}, \text{nonEntier}) \Rightarrow \text{Expr}! / (\text{Expr} - \text{nonEntier})!$

$\text{nPr}(\text{Liste1}, \text{Liste2}) \Rightarrow \text{liste}$

$\text{nPr}(\{5, 4, 3\}, \{2, 4, 2\}) \quad \{20, 24, 6\}$

Donne une liste de permutations basées sur les paires d'éléments correspondantes dans les deux listes. Les arguments doivent être des listes comportant le même nombre d'éléments.

$\text{nPr}(\text{Matrice1}, \text{Matrice2}) \Rightarrow \text{matrice}$

$\text{nPr}\left(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right) \quad \begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$

Donne une matrice de permutations basées sur les paires d'éléments correspondantes dans les deux matrices. Les arguments doivent être des matrices comportant le même nombre d'éléments.

$\text{npv}(\text{tauxIntérêt}, \text{MTO}, \text{ListeMT} [, \text{FréqMT}])$

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

Fonction financière permettant de calculer la valeur actuelle nette ; la somme des valeurs actuelles des mouvements d'entrée et de sortie de fonds. Un résultat positif pour NPV indique un investissement rentable.

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

$\text{npv}(10, 5000, \text{list1}, \text{list2}) \quad 4769.91$

tauxIntérêt est le taux à appliquer pour l'escompte des mouvements de trésorerie (taux de l'argent) sur une période donnée.

MTO correspond au mouvement de trésorerie initial à l'heure 0 ; il doit s'agir d'un nombre réel.

Liste MT est une liste des montants de mouvements de trésorerie après le mouvement de trésorerie initial *MTO*.

FréqMT est une liste dans laquelle chaque élément indique la fréquence d'occurrence d'un montant de mouvement de trésorerie groupé (consécutif), correspondant à l'élément de *ListeMT*. La valeur par défaut est 1 ; si vous saisissez des valeurs, elles doivent être des entiers positifs < 10 000.

nSolve()

Catalogue >

nSolve(*Équation*, *Var*[=*Condition*]) ⇒ *chaîne_nombre* ou *erreur*

nSolve(*Équation*, *Var* [*=Condition*], *LimitInf*) ⇒ *chaîne_nombre* ou *erreur*

nSolve(*Équation*, *Var* [*=Condition*], *LimitInf*, *LimitSup*) ⇒ *chaîne_nombre* ou *erreur*

nSolve(*Équation*, *Var*[=*Condition*]) | *LimitInf* ≤ *Var* ≤ *LimitSup* ⇒ *chaîne_nombre* ou *erreur*

Recherche de façon itérative une solution numérique réelle approchée pour *Équation* en fonction de sa variable. Spécifiez la variable comme suit :

variable

– ou –

variable = *nombre réel*

Par exemple, x est autorisé, de même que x=3.

nSolve() est souvent plus rapide que **solve()** ou **zeros()**, notamment si l'opérateur « | » est utilisé pour limiter la recherche à un intervalle réduit qui contient exactement une seule solution.

$\text{nSolve}(x^2 + 5 \cdot x - 25 = 9, x)$	3.84429
$\text{nSolve}(x^2 = 4, x = -1)$	-2.
$\text{nSolve}(x^2 = 4, x = 1)$	2.

Remarque : si plusieurs solutions sont possibles, vous pouvez utiliser une condition pour mieux déterminer une solution particulière.

$\text{nSolve}(x^2 + 5 \cdot x - 25 = 9, x) x < 0$	-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"

nSolve() tente de déterminer un point où la valeur résiduelle est zéro ou deux points relativement rapprochés où la valeur résiduelle a un signe négatif et où son ordre de grandeur n'est pas excessif. S'il n'y parvient pas en utilisant un nombre réduit de points d'échantillon, la chaîne « Aucune solution n'a été trouvée » s'affiche.

Remarque : voir aussi **cSolve()**, **cZeros()**, **solve()**, et **zeros()**.

O

OneVar [**1**,]*X*[,*Fréq*][,*Catégorie*,*Inclure*]]

OneVar [*n*,]*X1*,*X2*[*X3*[...[,*X20*]]]

Effectue le calcul de statistiques à une variable sur un maximum de 20 listes. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

Les arguments X sont des listes de données.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque valeur *X* correspondante. Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes numériques de catégories pour les valeurs *X* correspondantes.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Tout élément vide dans les listes *X*, *Fréq* ou *Catégorie* a un élément vide correspondant dans l'ensemble des listes résultantes. Tout élément vide dans les listes *X1* à *X20* correspond a un élément vide dans l'ensemble des listes résultantes. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

Variable de sortie	Description
stat. $\bar{x}$	Moyenne des valeurs x
stat. Σx	Somme des valeurs x
stat. Σx^2	Somme des valeurs x^2 .
stat.sx	Écart-type de l'échantillon de x
stat. x	Écart-type de la population de x
stat.n	Nombre de points de données
stat.MinX	Minimum des valeurs de x
stat.Q ₁ X	1er quartile de x
stat.MedianX	Médiane de x
stat.Q ₃ X	3ème quartile de x
stat.MaxX	Maximum des valeurs de x
stat.SSX	Somme des carrés des écarts par rapport à la moyenne de x

or

BooleanExpr1 **or** *BooleanExpr2* renvoie *expression booléenne*

$x \geq 3$ or $x \geq 4$ $x \geq 3$

BooleanList1 **or** *BooleanList2* renvoie *liste booléenne*

Define $g(x)$ = Func Done

BooleanMatrix1 **or** *BooleanMatrix2* renvoie *matrice booléenne*

If $x \leq 0$ or $x \geq 5$
Goto end
Return $x \cdot 3$
Lbl end
EndFunc

Donne true (vrai) ou false (faux) ou une forme simplifiée de l'entrée initiale.

$g(3)$ 9
 $g(0)$ A function did not return a value

Donne true si la simplification de l'une des deux ou des deux expressions est vraie. Donne false uniquement si la simplification des deux expressions est fausse.

Remarque : voir **xor**.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Entier1 or Entier2 $\Rightarrow$ entier

Compare les représentations binaires de deux entiers réels en appliquant un or bit par bit. En interne, les deux entiers sont convertis en nombres binaires 64 bits signés. Lorsque les bits comparés correspondent, le résultat est 1 si dans les deux cas il s'agit d'un bit 1 ; le résultat est 0 si, dans les deux cas, il s'agit d'un bit 0. La valeur donnée représente le résultat des bits et elle est affichée selon le mode Base utilisé.

Les entiers de tout type de base sont admis. Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h. Tout entier sans préfixe est considéré comme un nombre en écriture décimale (base 10).

Si vous entrez un nombre dont le codage binaire signé dépasse 64 bits, il est ramené à l'aide d'une congruence dans la plage appropriée. Pour de plus amples informations, voir ►**Base2**, page 19.

Remarque : voir **xor**.

En mode base Hex :

0h7AC36 or 0h3D5F	0h7BD7F
-------------------	---------

Important : utilisez le chiffre zéro et pas la lettre O.

En mode base Bin :

0b100101 or 0b100	0b100101
-------------------	----------

Remarque : une entrée binaire peut comporter jusqu'à 64 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 16 chiffres.

ord()Catalogue > **ord**(*Chaîne*) $\Rightarrow$ *entier***ord**(*ListeI*) $\Rightarrow$ *liste*

Donne le code numérique du premier caractère de la chaîne de caractères *Chaîne* ou une liste des premiers caractères de tous les éléments de la liste.

ord ("hello")	104
char (104)	"h"
ord (char (24))	24
ord ({"alpha", "beta"})	{ 97, 98 }

P**P>Rx()**Catalogue > **P>Rx**(*ExprR*, θ *Expr*) $\Rightarrow$ *expression***P>Rx**(*ListeR*, θ *Liste*) $\Rightarrow$ *liste***P>Rx**(*MatriceR*, θ *Matrice*) $\Rightarrow$ *matrice*

Donne la valeur de l'abscisse du point de coordonnées polaires (r , θ).

Remarque : l'argument θ est interprété comme une mesure en degrés, en grades ou en radians, suivant le mode Angle utilisé. Si l'argument est une expression, vous pouvez utiliser °, G ou Γ pour ignorer temporairement le mode Angle sélectionné.

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **P@>Rx** (...).

En mode Angle en radians :

P>Rx (r, θ)	$\cos(\theta) \cdot r$
P>Rx (4, 60°)	2
P>Rx ($\{-3, 10, 1.3\}, \{\frac{\pi}{3}, \frac{\pi}{4}, 0\}$)	$\left\{ \frac{-3}{2}, 5\sqrt{2}, 1.3 \right\}$

P>Ry()Catalogue > **P>Ry**(*ExprR*, θ *Expr*) $\Rightarrow$ *expression***P>Ry**(*ListeR*, θ *Liste*) $\Rightarrow$ *liste***P>Ry**(*MatriceR*, θ *Matrice*) $\Rightarrow$ *matrice*

Donne la valeur de l'ordonnée du point de coordonnées polaires (r , θ).

En mode Angle en radians :

P>Ry (r, θ)	$\sin(\theta) \cdot r$
P>Ry (4, 60°)	$2 \cdot \sqrt{3}$
P>Ry ($\{-3, 10, 1.3\}, \{\frac{\pi}{3}, \frac{\pi}{4}, 0\}$)	$\left\{ \frac{-3\sqrt{3}}{2}, -5\sqrt{2}, 0 \right\}$

Remarque : l'argument θ est interprété comme une mesure en degrés, en grades ou en radians, suivant le mode Angle utilisé. Si l'argument est une expression, vous pouvez utiliser °, G ou r pour ignorer temporairement le mode Angle sélectionné.

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **P@Ry (...)**.

PassErr

PassErr

Passe une erreur au niveau suivant.

Si la variable système *errCode* est zéro, **PassErr** ne fait rien.

L'instruction **Else** du bloc **Try...Else...EndTry** doit utiliser **EffErr** ou **PassErr**. Si vous comptez rectifier ou ignorer l'erreur, sélectionnez **EffErr**. Si vous ne savez pas comment traiter l'erreur, sélectionnez **PassErr** pour la transférer au niveau suivant. S'il n'y a plus d'autre programme de traitement des erreurs **Try...Else...EndTry**, la boîte de dialogue Erreur s'affiche normalement.

Remarque : Voir aussi **ClrErr**, page 28 et **Try**, page 214.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Pour obtenir un exemple de **PassErr**, reportez-vous à l'exemple 2 de la commande **Try**, page 214.

piecewise()

piecewise(*Expr1* [, *Condition1* [, *Expr2* [, *Condition2* [, ...]]]])

Define $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, & x \leq 0 \end{cases}$ Done

$p(1)$ 1

$p(-1)$ undef

piecewise()

Catalogue > 

Permet de créer des fonctions définies par morceaux sous forme de liste. Il est également possible de créer des fonctions définies par morceaux en utilisant un modèle.

Remarque : voir aussi **Modèle Fonction définie par morceaux**, page 3.

poissCdf()

Catalogue > 

poissCdf(λ , *lowBound*, *upBound*) $\Rightarrow$ *nombre* si *lowBound* et *upBound* sont des nombres, *liste* si *lowBound* et *upBound* sont des listes

poissCdf(λ , *upBound*) (pour $P(0 \leq X \leq upBound)$) $\Rightarrow$ *nombre* si la borne *upBound* est un nombre, *liste* si la borne *upBound* est une liste

Calcule la probabilité cumulée d'une variable suivant une loi de Poisson de moyenne λ .

Pour $P(X \leq upBound)$, définissez la borne *lowBound*=0

poissPdf()

Catalogue > 

poissPdf(λ , *ValX*) $\Rightarrow$ *nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

Calcule la probabilité de *ValX* pour la loi de Poisson de moyenne λ spécifiée.

►Polar

Catalogue > 

Vecteur ►Polar

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>Polar.

Affiche *vecteur* sous forme polaire [$r \angle \theta$]. Le vecteur doit être un vecteur ligne ou colonne et de dimension 2.

$\begin{bmatrix} 1 & 3. \end{bmatrix}$ ►Polar	$\begin{bmatrix} 3.16228 & \angle 1.24905 \end{bmatrix}$
$\begin{bmatrix} x & y \end{bmatrix}$ ►Polar	$\begin{bmatrix} \sqrt{x^2+y^2} & \angle \frac{\pi \cdot \text{sign}(y)}{2} \tan^{-1}\left(\frac{x}{y}\right) \end{bmatrix}$

Remarque : ►Polar est uniquement une instruction d'affichage et non une fonction de conversion. On ne peut l'utiliser qu'à la fin d'une ligne et elle ne modifie pas le contenu du registre *ans*.

Remarque : voir aussi ►Rect, page 164.

valeurComplexe ►Polar

Affiche *valeurComplexe* sous forme polaire.

- Le mode Angle en degrés affiche ($r \angle \theta$).
- Le mode Angle en radians affiche $re^{i\theta}$.

valeurComplexe peut prendre n'importe quelle forme complexe. Toutefois, une entrée $re^{i\theta}$ génère une erreur en mode Angle en degrés.

Remarque : vous devez utiliser les parenthèses pour les entrées polaires ($r \angle \theta$).

En mode Angle en radians :

$$\begin{array}{l} (3+4i) \text{ ►Polar} \quad e^{i \cdot \left(\frac{\pi}{2} - \tan^{-1} \left(\frac{3}{4} \right) \right)} \cdot 5 \\ \left(4 \angle \frac{\pi}{3} \right) \text{ ►Polar} \quad e^{\frac{i \cdot \pi}{3}} \cdot 4 \end{array}$$

En mode Angle en grades :

$$(4i) \text{ ►Polar} \quad (4 \angle 100.)$$

En mode Angle en degrés :

$$(3+4i) \text{ ►Polar} \quad \left(5 \angle 90 - \tan^{-1} \left(\frac{3}{4} \right) \right)$$

polyCoeffs()

polyCoeffs(*Poly* [, *Var*]) ⇒ *liste*

Affiche une liste des coefficients du polynôme *Poly* pour la variable *Var*.

Poly doit être une expression polynomiale de *Var*. Nous conseillons de ne pas omettre *Var* à moins que *Poly* ne soit une expression dans une variable unique.

$$\text{polyCoeffs}(4x^2 - 3x + 2, x) \quad \{4, -3, 2\}$$

$$\text{polyCoeffs}((x-1)^2 \cdot (x+2)^3) \quad \{1, 4, 1, -10, -4, 8\}$$

Étend le polynôme et sélectionne *x* pour la variable omise *Var*.

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

polyDegree()

polyDegree(*Poly* [, *Var*]) $\Rightarrow$ valeur

Affiche le degré de l'expression polynomiale *Poly* pour la variable *Var*. Si vous omettez *Var*, la fonction **polyDegree()** sélectionne une variable par défaut parmi les variables contenues dans le polynôme *Poly*.

Poly doit être une expression polynomiale de *Var*. Nous conseillons de ne pas omettre *Var* à moins que *Poly* ne soit une expression dans une variable unique.

$\text{polyDegree}(5)$	0
$\text{polyDegree}(\ln(2) + \pi, x)$	0

Polynômes constants

$\text{polyDegree}(4 \cdot x^2 - 3 \cdot x + 2, x)$	2
$\text{polyDegree}((x-1)^2 \cdot (x+2)^3)$	5
$\text{polyDegree}\left(\left(x+y^2+z^3\right)^2, x\right)$	2
$\text{polyDegree}\left(\left(x+y^2+z^3\right)^2, y\right)$	4
$\text{polyDegree}\left((x-1)^{10000}, x\right)$	10000

Il est possible d'extraire le degré, même si cela n'est pas possible pour les coefficients. Cela s'explique par le fait qu'un degré peut être extrait sans développer le polynôme.

polyEval()

polyEval(*Liste1*, *Expr1*) $\Rightarrow$ expression

polyEval(*Liste1*, *Liste2*) $\Rightarrow$ expression

$\text{polyEval}(\{a, b, c\}, x)$	$a \cdot x^2 + b \cdot x + c$
$\text{polyEval}(\{1, 2, 3, 4\}, 2)$	26
$\text{polyEval}(\{1, 2, 3, 4\}, \{2, -7\})$	$\{26, -262\}$

polyEval()Catalogue > 

Interprète le premier argument comme les coefficients d'un polynôme ordonné suivant les puissances décroissantes et calcule la valeur de ce polynôme au point indiqué par le deuxième argument.

polyGcd()Catalogue > 

polyGcd(*Expr1*,*Expr2*) $\Rightarrow$ *expression*

Donne le plus grand commun diviseur des deux arguments.

Expr1 et *Expr2* doivent être des expressions polynomiales.

Les listes, matrices et arguments booléens ne sont pas autorisés.

$\text{polyGcd}(100,30)$	10
$\text{polyGcd}(x^2-1, x-1)$	$x-1$
$\text{polyGcd}(x^3-6 \cdot x^2+11 \cdot x-6, x^2-6 \cdot x+8)$	$x-2$

polyQuotient()Catalogue > 

polyQuotient(*Poly1*,*Poly2*
[,*Var*]) $\Rightarrow$ *expression*

Affiche le quotient de polynôme *Poly1* divisé par le polynôme *Poly2* par rapport à la variable spécifiée *Var*.

Poly1 et *Poly2* doivent être des expressions polynomiales de *Var*. Nous conseillons de ne pas omettre *Var* à moins que *Poly1* et *Poly2* ne soient des expressions dans une même variable unique.

$\text{polyQuotient}(x-1, x-3)$	1
$\text{polyQuotient}(x-1, x^2-1)$	0
$\text{polyQuotient}(x^2-1, x-1)$	$x+1$
$\text{polyQuotient}(x^3-6 \cdot x^2+11 \cdot x-6, x^2-6 \cdot x+8)$	x
$\text{polyQuotient}((x-y) \cdot (y-z), x+y+z, x)$	$y-z$
$\text{polyQuotient}((x-y) \cdot (y-z), x+y+z, y)$	$2 \cdot x-y+2 \cdot z$
$\text{polyQuotient}((x-y) \cdot (y-z), x+y+z, z)$	$-(x-y)$

polyRemainder()Catalogue > 

polyRemainder(*Poly1*,*Poly2*
[,*Var*]) $\Rightarrow$ *expression*

$\text{polyRemainder}(x-1, x-3)$	2
$\text{polyRemainder}(x-1, x^2-1)$	$x-1$
$\text{polyRemainder}(x^2-1, x-1)$	0

polyRemainder()

Catalogue >

Affiche le reste du polynôme *Poly1*
divisé par le polynôme *Poly2* par rapport
à la variable spécifiée *Var*.

Poly1 et *Poly2* doivent être des expressions polynomiales de *Var*. Nous conseillons de ne pas omettre *Var* à moins que *Poly1* et *Poly2* ne soient des expressions dans une même variable unique.

polyRemainder($(x-y) \cdot (y-z), x+y+z, x$)	$(y-z) \cdot (2 \cdot y+z)$
polyRemainder($(x-y) \cdot (y-z), x+y+z, y$)	$-2 \cdot x^2 - 5 \cdot x \cdot z - 2 \cdot z^2$
polyRemainder($(x-y) \cdot (y-z), x+y+z, z$)	$(x-y) \cdot (x+2 \cdot y)$

polyRoots()

Catalogue > 

polyRoots(*Poly*, *Var*) $\Rightarrow$ *liste*

polyRoots(*ListeCoeff*) $\Rightarrow$ *liste*

La première syntaxe, **polyRoots** (*Poly*, *Var*), affiche une liste des racines réelles du polynôme *Poly* pour la variable *Var*. S'il n'existe pas de racine réelle, une liste vide est affichée : {}.

Poly doit être un polynôme d'une seule variable.

La deuxième syntaxe, **polyRoots** (*ListeCoeff*), affiche une liste de racines réelles du polynôme dont les coefficients sont donnés par la liste *ListeCoeff*.

Remarque : voir aussi `cPolyRoots()`, page 40.

$\text{polyRoots}(y^{3+1}, y)$	$\{-1\}$
$\text{cPolyRoots}(y^{3+1}, y)$	$\left\{-1, \frac{1}{2} - \frac{\sqrt{3}}{2}i, \frac{1}{2} + \frac{\sqrt{3}}{2}i\right\}$
$\text{polyRoots}(x^{2+2x+1}, x)$	$\{-1, -1\}$
$\text{polyRoots}\{1, 2, 1\}$	$\{-1, -1\}$

PowerReg

Catalogue > **PowerReg** *X,Y [, Fréq] [, Catégorie, Inclure]*

Effectue l'ajustement exponentiel $y = (a \cdot (x)^b)$ sur les listes X et Y en utilisant la fréquence *Fréq.* Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et Y sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple X et Y . Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples X et Y correspondants.

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a \cdot (x)^b$
stat.a, stat.b	Coefficients d'ajustement
stat.r ²	Coefficient de détermination linéaire pour les données transformées
stat.r	Coefficient de corrélation pour les données transformées ($\ln(x)$, $\ln(y)$)
stat.Resid	Valeurs résiduelles associées au modèle exponentiel
stat.ResidTrans	Valeurs résiduelles associées à l'ajustement linéaire des données transformées
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

Prgm Bloc

Calcule le plus grand commun diviseur et affiche les résultats intermédiaires.

EndPrgm

Modèle de création d'un programme défini par l'utilisateur. À utiliser avec la commande **Define**, **Define LibPub**, ou **Define LibPriv**.

Bloc peut correspondre à une instruction unique ou à une série d'instructions séparées par le caractère ":" ou à une série d'instructions réparties sur plusieurs lignes.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

```
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()Voir $\Pi()$, page 250.**Product (PI)**Voir $\Pi()$, page 250.**product()**Catalogue > 

product(*Liste*[, *Début*[, *Fin*]]) $\Rightarrow$ *expression*

Donne le produit des éléments de *Liste*. *Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage d'éléments.

product(*MatriceI*[, *Début*[, *Fin*]]) $\Rightarrow$ *matrice*

Donne un vecteur ligne contenant les produits des éléments ligne par ligne de *MatriceI*. *Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage de colonnes.

product({ 1,2,3,4 })	24
----------------------	----

product({ 2,x,y })	$2 \cdot x \cdot y$
--------------------	---------------------

product({ 4,5,8,9 },2,3)	40
--------------------------	----

product $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	$\begin{bmatrix} 28 & 80 & 162 \end{bmatrix}$
---	---

product $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, 1,2$	$\begin{bmatrix} 4 & 10 & 18 \end{bmatrix}$
--	---

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

propFrac()

propFrac(*Expr1* [, *Var*]) $\Rightarrow$ *expression*

propFrac(*nombre_rationnel*) décompose *nombre_rationnel* sous la forme de la somme d'un entier et d'une fraction de même signe et dont le dénominateur est supérieur au numérateur (fraction propre).

propFrac(*expression_rationnelle*, *Var*) donne la somme des fractions propres et d'un polynôme par rapport à *Var*. Le degré de *Var* dans le dénominateur est supérieur au degré de *Var* dans le numérateur pour chaque fraction propre. Les mêmes puissances de *Var* sont regroupées. Les termes et leurs facteurs sont triés, *Var* étant la variable principale.

Si *Var* est omis, le développement des fractions propres s'effectue par rapport à la variable la plus importante. Les coefficients de la partie polynomiale sont ensuite ramenés à leur forme propre par rapport à leur variable la plus importante, et ainsi de suite.

Pour les expressions rationnelles, **propFrac()** est une alternative plus rapide mais moins extrême à **expand()**.

Vous pouvez utiliser la fonction **propFrac()** pour représenter des fractions mixtes et démontrer l'addition et la soustraction de fractions mixtes.

$$\text{propFrac}\left(\frac{4}{3}\right) \quad 1 + \frac{1}{3}$$

$$\text{propFrac}\left(\frac{-4}{3}\right) \quad -1 - \frac{1}{3}$$

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

$$\text{propFrac}(\text{Ans}) \quad \frac{1}{x+1} + x + \frac{1}{y+1} + y$$

$$\text{propFrac}\left(\frac{11}{7}\right) \quad 1 + \frac{4}{7}$$

$$\text{propFrac}\left(3 + \frac{1}{11} + 5 + \frac{3}{4}\right) \quad 8 + \frac{37}{44}$$

$$\text{propFrac}\left(3 + \frac{1}{11} - \left(5 + \frac{3}{4}\right)\right) \quad -2 - \frac{29}{44}$$

QR

QR *Matrice*, *qMatrice*, *rMatrice* [,*Tol*]

Calcule la factorisation QR Householder d'une matrice réelle ou complexe. Les matrices Q et R obtenues sont stockées dans les NomsMat *spécifiés*. La matrice Q est unitaire. La matrice R est triangulaire supérieure.

L'argument facultatif Tol permet de considérer comme nul tout élément de la matrice dont la valeur absolue est inférieure à *Tol*. Cet argument n'est utilisé que si la matrice contient des nombres en virgule flottante et ne contient pas de variables symbolique sans valeur affectée. Dans le cas contraire, *Tol* est ignoré.

- Si vous utilisez   ou définissez le mode **Auto ou Approché (Approximate)** sur Approché (Approximate), les calculs sont exécutés en virgule flottante.
- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit : $5E-14 \cdot \max(\dim(\text{Matrice})) \cdot \text{rowNorm}(\text{Matrice})$

La factorisation QR sous forme numérique est calculée en utilisant la transformation de Householder. La factorisation symbolique est calculée en utilisant la méthode de Gram-Schmidt. Les colonnes de *NomMatq* sont les vecteurs de base orthonormaux de l'espace vectoriel engendré par les vecteurs colonnes de *matrice*.

Le nombre en virgule flottante (9.) dans m1 fait que les résultats seront tous calculés en virgule flottante.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9. \end{bmatrix}$
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	$\begin{bmatrix} 8.12404 & 9.60114 & 11.0782 \\ 0. & 0.904534 & 1.80907 \\ 0. & 0. & 0. \end{bmatrix}$

$\begin{bmatrix} m & n \\ o & p \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} m & n \\ o & p \end{bmatrix}$
QR m1,qm,rm	Done
qm	$\begin{bmatrix} \frac{m}{\sqrt{m^2+o^2}} & \frac{-\text{sign}(m \cdot p - n \cdot o) \cdot o}{\sqrt{m^2+o^2}} \\ \frac{o}{\sqrt{m^2+o^2}} & \frac{m \cdot \text{sign}(m \cdot p - n \cdot o)}{\sqrt{m^2+o^2}} \end{bmatrix}$
rm	$\begin{bmatrix} \sqrt{m^2+o^2} & \frac{m \cdot n + o \cdot p}{\sqrt{m^2+o^2}} \\ 0 & \frac{ m \cdot p - n \cdot o }{\sqrt{m^2+o^2}} \end{bmatrix}$

QuadReg X, Y [, $Fr\acute{e}q$] [, $Cat\acute{e}gorie$, $Inclure$]]

Effectue l'ajustement polynomial de degré 2 $y = a \cdot x^2 + b \cdot x + c$ sur les listes X et Y en utilisant la fréquence $Fr\acute{e}q$. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et Y sont des listes de variables indépendantes et dépendantes.

$Fr\acute{e}q$ est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans $Fr\acute{e}q$ correspond à une fréquence d'occurrence pour chaque couple X et Y . Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

$Cat\acute{e}gorie$ est une liste de codes de catégories pour les couples X et Y correspondants..

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a \cdot x^2 + b \cdot x + c$
stat.a, stat.b, stat.c	Coefficients d'ajustement
stat.R ²	Coefficient de détermination
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fr\acute{e}q</i> , <i>Liste de cat\acute{e}gories</i> et <i>Inclure les cat\acute{e}gories</i>

stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

QuartReg

Catalogue > 

QuartReg *X,Y [, Fréq] [, Catégorie, Inclure]*

Effectue l'ajustement polynomial de degré 4

$y = a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même nombre de lignes, à l'exception de *Inclure*.

X et *Y* sont des listes de variables indépendantes et dépendantes.

Fréq est une liste facultative de valeurs qui indiquent la fréquence. Chaque élément dans *Fréq* correspond à une fréquence d'occurrence pour chaque couple *X* et *Y*. Par défaut, cette valeur est égale à 1. Tous les éléments doivent être des entiers ≥ 0 .

Catégorie est une liste de codes de catégories pour les couples *X* et *Y* correspondants..

Inclure est une liste d'un ou plusieurs codes de catégories. Seuls les éléments dont le code de catégorie figure dans cette liste sont inclus dans le calcul.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a \cdot x^4 + b \cdot x^3 + c \cdot x^2 + d \cdot x + e$
stat.a, stat.b, stat.c, stat.d, stat.e	Coefficients d'ajustement

Variable de sortie	Description
stat.R ²	Coefficient de détermination
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

R

R ► Pθ()

Catalogue > 

R ► Pθ (*xExpr*, *yExpr*) ⇒ *expression*

R ► Pθ (*listex*, *listey*) ⇒ *liste*

R ► Pθ (*matricex*, *matricey*) ⇒ *matrice*

Donne la valeur de l'ordonnée θ - du point de coordonnées rectangulaires (x,y).

Remarque : Donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

Remarque : Vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **R@>Ptheta** (...).

En mode Angle en degrés :

$$\text{R ► P}\theta(x,y) \quad 90 \cdot \text{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right)$$

En mode Angle en grades :

$$\text{R ► P}\theta(x,y) \quad 100 \cdot \text{sign}(y) - \tan^{-1}\left(\frac{x}{y}\right)$$

En mode Angle en radians et en mode Auto :

$$\begin{aligned} \text{R ► P}\theta(3,2) & \quad \tan^{-1}\left(\frac{2}{3}\right) \\ \text{R ► P}\theta\left([3 \ -4 \ 2], \left[0 \ \frac{\pi}{4} \ 1.5\right]\right) & \quad \left[0 \ \tan^{-1}\left(\frac{16}{\pi}\right) + \frac{\pi}{2} \ 0.643501\right] \end{aligned}$$

R ► Pr()

Catalogue > 

R ► Pr (*xExpr*, *yExpr*) ⇒ *expression*

R ► Pr (*listex*, *listey*) ⇒ *liste*

R ► Pr (*matricex*, *matricey*) ⇒ *matrice*

Donne la coordonnée r d'un point de coordonnées rectangulaires (x,y)

En mode Angle en radians et en mode Auto :

R►Pr()

Catalogue >

Remarque : Vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **R@>Pr** (...).

$$\begin{array}{l} R \blacktriangleright \text{Pr}(3,2) \qquad \qquad \qquad \sqrt{13} \\ R \blacktriangleright \text{Pr}(x,y) \qquad \qquad \qquad \sqrt{x^2+y^2} \\ R \blacktriangleright \text{Pr}\left(\left[\begin{array}{ccc} 3 & -4 & 2 \end{array} \right], \left[\begin{array}{c} 0 \\ \frac{\pi}{4} \\ 1.5 \end{array} \right] \right) \\ \qquad \qquad \qquad \left[\begin{array}{cc} 3 & \frac{\sqrt{\pi^2+256}}{4} \\ & 2.5 \end{array} \right] \end{array}$$

►Rad

Catalogue >

Expr1 ►Rad ⇒ expression

En mode Angle en degrés :

Convertit l'argument en mesure d'angle en radians.

$$(1.5) \blacktriangleright \text{Rad} \qquad \qquad \qquad (0.02618)^r$$

Remarque : Vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant **@>Rad**.

En mode Angle en grades :

$$(1.5) \blacktriangleright \text{Rad} \qquad \qquad \qquad (0.023562)^r$$

rand()

Catalogue >

rand() ⇒ *expression*

rand(#Trials) ⇒ *liste*

Réinitialise le générateur de nombres aléatoires.

rand() donne un nombre aléatoire compris entre 0 et 1.

$$\begin{array}{l} \text{RandSeed } 1147 \qquad \qquad \qquad \text{Done} \\ \text{rand}(2) \qquad \qquad \qquad \{0.158206, 0.717917\} \end{array}$$

rand(nbreEssais) donne une liste de nombres aléatoires compris entre 0 et 1 pour le nombre d'essais *nbreEssais*

randBin()

Catalogue >

randBin(n, p) ⇒ *expression*

randBin(n, p, #Trials) ⇒ *liste*

$$\begin{array}{l} \text{randBin}(80, 0.5) \qquad \qquad \qquad 42 \\ \text{randBin}(80, 0.5, 3) \qquad \qquad \qquad \{41, 32, 39\} \end{array}$$

randBin(n, p) donne un nombre aléatoire tiré d'une distribution binomiale spécifiée

randBin(n, p, nbreEssais) donne une liste de nombres aléatoires tirés d'une distribution binomiale spécifiée pour un nombre d'essais *nbreEssais*.

randInt()Catalogue > 

randInt
(lowBound,upBound)
 $\Rightarrow$ *expression*
randInt
(LimiteInf,LimiteSup
,NbrEssais) $\Rightarrow$ *liste*

randInt (3,10)	5
randInt (3,10,4)	{ 9,7,5,8 }

randInt
(LimiteInf,LimiteSup
 $)$ donne un entier
aléatoire pris entre
les limites entières
LimiteInf et
LimiteSup

randInt
(LimiteInf,LimiteSup
,nbreEssais) donne
une liste d'entiers
aléatoires pris entre
les limites spécifiées
pour un nombre
d'essais *nbreEssais*.

randMat()Catalogue > 

randMat(*nbreLignes, nbreColonnes*)
 $\Rightarrow$ *matrice*

Donne une matrice d'entiers compris
entre -9 et 9 de la dimension spécifiée.

Les deux arguments doivent pouvoir être
simplifiés en entiers.

RandSeed 1147	Done		
randMat(3,3)	8	-3	6
	-2	3	-6
	0	4	-6

Remarque : Les valeurs de cette matrice
changent chaque fois que l'on appuie sur

**randNorm()**Catalogue > 

randNorm(μ, σ) $\Rightarrow$ *expression*
randNorm($\mu, \sigma, nbreessais$) $\Rightarrow$ *liste*

randNorm(μ, σ) Donne un nombre
décimal issu de la loi normale spécifiée. Il
peut s'agir de tout nombre réel, mais le
résultat obtenu sera essentiellement
compris dans l'intervalle $[\mu-3\cdot\sigma, \mu+3\cdot\sigma]$.

RandSeed 1147	Done
randNorm (0,1)	0.492541
randNorm (3,4.5)	-3.54356

randNorm()

Catalogue > 

randNorm(μ , σ , *nbreEssais*) donne une liste de nombres décimaux tirés d'une distribution normale spécifiée pour un nombre d'essais *nbreEssais*.

randPoly()

Catalogue > 

randPoly(*Var*, *Order*) $\Rightarrow$ *expression*

Donne un polynôme aléatoire de la variable *Var* de degré *Order* spécifié. Les coefficients sont des entiers aléatoires situés dans la plage -9 à 9. Le coefficient du terme de plus au degré (*Order*) sera non nul.

Order doit être un entier compris entre 0 et 99

RandSeed 1147	Done
randPoly(<i>x</i> ,5)	$-2 \cdot x^5 + 3 \cdot x^4 - 6 \cdot x^3 + 4 \cdot x - 6$

randSamp()

Catalogue > 

randSamp(*List*, *#Trials*, [*noRepl*]) $\Rightarrow$ *liste*

Donne une liste contenant un échantillon aléatoire de *nbreEssais* éléments choisis dans *Liste* avec option de remise (*sansRem*=0) ou sans option de remise (*sansRem*=1). L'option par défaut est avec remise.

Define <i>list3</i> ={1,2,3,4,5}	Done
Define <i>list4</i> =randSamp(<i>list3</i> ,6)	Done
<i>list4</i>	{2,3,4,3,1,2}

RandSeed

Catalogue > 

RandSeed *Nombre*

Si *Nombre* = 0, réinitialise le générateur de nombres aléatoires. Si *Nombre* $\neq$ 0, il sert à générer deux germes qui sont stockés dans les variables système seed1 et seed2.

RandSeed 1147	Done
rand()	0.158206

real()

Catalogue >

real(Expr1) ⇒ expression

Donne la partie réelle de l'argument.

Remarque : Toutes les variables non affectées sont considérées comme réelles. Voir également **imag()**, page 100.

real(List1) ⇒ liste

Donne les parties réelles de tous les éléments.

real(Matrix1) ⇒ matrice

Donne les parties réelles de tous les éléments.

$\text{real}(2+3 \cdot i)$	2
$\text{real}(z)$	z
$\text{real}(x+i \cdot y)$	x

$$\text{real}(\{a+i \cdot b, 3, i\}) \quad \{a, 3, 0\}$$

$$\text{real}\left(\begin{bmatrix} a+i \cdot b & 3 \\ c & i \end{bmatrix}\right) \quad \begin{bmatrix} a & 3 \\ c & 0 \end{bmatrix}$$

► Rect

Catalogue >

Vecteur ► Rect

Remarque : Vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant **@>Rect**

Affiche **Vecteur** en coordonnées rectangulaires [x, y, z]. Le vecteur doit être un vecteur ligne ou colonne de dimension 2 ou 3.

Remarque : ► Rect est uniquement une instruction d'affichage et non une fonction de conversion. On ne peut l'utiliser qu'à la fin d'une ligne et elle ne modifie pas le contenu du registre *ans*.

Remarque : Voir également ► Polar, page 149.

complexValue ► Rect

Affiche **valeurComplexe** sous forme rectangulaire (a+bi) La **valeurComplexe** peut prendre n'importe quelle forme rectangulaire Toutefois, une entrée $re^{i\theta}$ génère une erreur en mode Angle en degrés

Remarque : Vous devez utiliser des parenthèses pour les entrées en polaire ($r \angle \theta$).

$$\left(3 \angle \frac{\pi}{4} \angle \frac{\pi}{6}\right) \text{► Rect} \quad \begin{bmatrix} \frac{3 \cdot \sqrt{2}}{4} & \frac{3 \cdot \sqrt{2}}{4} & \frac{3 \cdot \sqrt{3}}{2} \end{bmatrix}$$

$$\begin{bmatrix} a & \angle b & \angle c \end{bmatrix} \quad \begin{bmatrix} a \cdot \cos(b) \cdot \sin(c) & a \cdot \sin(b) \cdot \sin(c) & a \cdot \cos(c) \end{bmatrix}$$

En mode Angle en radians et en modes Auto :

$$\left(4 \cdot e^{\frac{\pi}{3}}\right) \text{► Rect} \quad 4 \cdot e^{\frac{\pi}{3}}$$

$$\left(4 \angle \frac{\pi}{3}\right) \text{► Rect} \quad 2+2 \cdot \sqrt{3} \cdot i$$

En mode Angle en grades :

$$\left((1 \angle 100)\right) \text{► Rect} \quad i$$

En mode Angle en degrés :

$$\left((4 \angle 60) \right) \blacktriangleright \text{Rect} \quad 2+2 \cdot \sqrt{3} \cdot i$$

Remarque : Pour taper $\angle$ à partir du clavier, sélectionnez-le dans la liste des symboles du Catalogue.

ref()

Catalogue >

ref(MatriceI[, Tol]) $\Rightarrow$ *matrice*

Donne une réduite de Gauss de la matrice *MatriceI*.

L'argument facultatif Tol permet de considérer comme nul tout élément de la matrice dont la valeur absolue est inférieure à *Tol*. Cet argument n'est utilisé que si la matrice contient des nombres en virgule flottante et ne contient pas de variables symboliques sans valeur affectée. Dans le cas contraire, *Tol* est ignoré

- Si vous utilisez              Auto ou Approché **sur** Approché, les calculs sont exécutés en virgule flottante
- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit : $5E-14 \cdot \max(\dim(\text{MatriceI})) \cdot \text{rowNorm}(\text{MatriceI})$

N'utilisez pas d'éléments non définis dans *MatriceI*. L'utilisation d'éléments non définis peut générer des résultats inattendus.

Par exemple, si *a* est un élément non défini dans l'expression suivante, un message d'avertissement s'affiche et le résultat affiché est le suivant :

$$\text{ref} \left(\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix} \right) \quad \begin{bmatrix} 1 & -\frac{2}{5} & -\frac{4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow mI \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

ref(*mI*)  $\begin{bmatrix} 1 & \frac{d}{c} \\ 0 & 1 \end{bmatrix}$

$$\text{ref} \left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \quad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Un message d'avertissement est affiché car l'élément $1/a$ n'est pas valide pour $a=0$.

Pour éviter ce problème, vous pouvez stocker préalablement une valeur dans a ou utiliser l'opérateur "sachant que" (« | ») pour substituer une valeur, comme illustré dans l'exemple suivant.

$$\text{ref} \left(\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mid a=0 \right) \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Remarque : Voir également **rref()**, page 176.

RefreshProbeVars

RefreshProbeVars

Vous permet d'accéder aux données de capteur à partir de toutes les sondes de capteur connectées à l'aide de votre programme TI-Basic.

Valeur	État
StatusVar	
<i>statusVar</i> =0	Normal (Poursuivez le programme) L'application Vernier DataQuest™ est en mode Acquisition de données.
<i>statusVar</i> =1	Remarque : L'application Vernier DataQuest™ doit être en mode compteur pour que cette commande fonctionne.



Par exemple

```

Define temp()=
Prgm
© Vérifier si le système est prêt
RefreshProbeVars status
Si le statut=0 alors
Disp "prêt"
For n,1,50
RefreshProbeVars status
température:=compteur.température
Disp "Température: ",température
Si la température>30 alors
Disp "Trop chaude"
EndIf

```

Valeur StatusVar	État	
$statusVar = 2$	L'application Vernier DataQuest™ n'est pas lancée.	© Attendre pendant 1 seconde entre les échantillons Wait 1 EndFor
$statusVar = 3$	L'application Vernier DataQuest™ est lancée, mais vous n'avez pas encore connecté de sonde.	Else Disp "Pas prêt. Réessayer plus tard" EndIf EndPrgm

Remarque : Ceci peut également être utilisé avec le TI-Innovator™ Hub.

remain()

remain(Expr1, Expr2) ⇒ expression
remain(Liste1, Liste2) ⇒ liste
remain(Matrice1, Matrice2) ⇒ matrice

Donne le reste de la division euclidienne du premier argument par le deuxième argument, défini par les identités suivantes :

$\text{remain}(x, 0) = x$
 $\text{remain}(x, y) = x - y \cdot \text{iPart}(x/y)$

Par conséquent, remarquez que **remain** $(-x, y) = -\text{remain}(x, y)$. Le résultat peut soit être égal à zéro, soit être du même signe que le premier argument.

Remarque : Voir aussi **mod()**, page 130.

remain(7,0)	7
remain(7,3)	1
remain(-7,3)	-1
remain(7,-3)	1
remain(-7,-3)	-1
remain({12,-14,16},{9,7,-5})	{3,0,1}

remain($\begin{bmatrix} 9 & -7 \\ 6 & 4 \end{bmatrix}, \begin{bmatrix} 4 & 3 \\ 4 & -3 \end{bmatrix}$)	$\begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$
--	---

Request

Request *promptString*, *var* [, *DispFlag* [, *statusVar*]]

Request *promptString*, *func*(*arg1*, ...*argn*) [, *DispFlag* [, *statusVar*]]

Définissez un programme :

```
Define request_demo()=Prgm
  Request "Rayon : ",r
  Disp "Area = ",pi*r^2
EndPrgm
```

Exécutez le programme et saisissez une réponse :

Commande de programmation : Marque une pause dans l'exécution du programme et affiche une boîte de dialogue contenant le message *chaîneinvite*, ainsi qu'une zone de saisie destinée à la réponse que doit fournir l'utilisateur.

Lorsque l'utilisateur saisit une réponse et clique sur **OK**, le contenu de la zone de saisie est affecté à la variable *var*.

Si l'utilisateur clique sur **Annuler**, le programme continue sans accepter aucune entrée. Le programme utilise la valeur précédente de la variable *var* si *var* était déjà définie.

L'argument optionnel *IndicAff* peut correspondre à toute expression.

- Si *IndicAff* est omis ou a pour valeur **1**, le message d'invite et la réponse de l'utilisateur sont affichés dans l'historique de l'application Calculs.
- Si *IndicAff* a pour valeur **0**, le message d'invite et la réponse de l'utilisateur ne sont pas affichés dans l'historique.

L'argument optionnel *VarÉtat* indique au programme comment déterminer si l'utilisateur a fermé la boîte de dialogue. Notez que *VarÉtat* nécessite la saisie de l'argument *IndicAff*.

- Si l'utilisateur a cliqué sur **OK**, ou a appuyé sur **Entrée** ou sur **Ctrl+Entrée**, la variable *VarÉtat* prend la valeur **1**.
- Sinon, la variable *StatusVar* prend la valeur **0**.

L'argument de *func()* permet à un programme de stocker la réponse de l'utilisateur sous la forme d'une définition de fonction. Cette syntaxe équivaut à l'exécution par l'utilisateur de la commande suivante :

```
request_demo()
```



Après avoir sélectionné **OK**, le résultat suivant s'affiche :

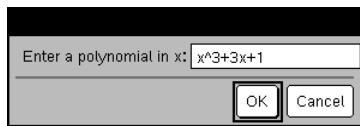
```
Demi-droite : 6/2
Area= 28.2743
```

Définissez un programme :

```
Define polynomial()=Prgm
  Request "Saisissez un polynôme en x
: ",p(x)
  Disp "Les racines réelles sont
: ",polyRoots(p(x),x)
EndPrgm
```

Exécutez le programme et saisissez une réponse :

```
polynomial()
```



Résultat après avoir saisi x^3+3x+1 et sélectionné **OK** :

Definir *func*(arg1, ...argn) = réponse de l'utilisateur

Les racines réelles sont : {-0.322185}

Le programme peut alors utiliser la fonction définie *func*(). La *chaîneinvite* doit guider l'utilisateur pour la saisie d'une *réponse* appropriée qui complète la définition de la fonction.

Remarque : Vous pouvez utiliser l Request commande dans un programme créé par l'utilisateur, mais pas dans une fonction.

Pour arrêter un programme qui contient une commande **Request** dans une boucle infinie :

- **Calculatrice**: Maintenez la touche  enfoncée et appuyez plusieurs fois sur .
- **Windows®** : Maintenez la touche **F12** enfoncée et appuyez plusieurs fois sur **Entrée**.
- **Macintosh®** : Maintenez la touche **F5** enfoncée et appuyez plusieurs fois sur **Entrée**.
- **iPad®** : L'application affiche une invite. Vous pouvez continuer à patienter ou annuler.

Remarque : Voir également **RequestStr**, page 169.

RequestStr

RequestStr chaîneinvite, var[, IndicAff]

Commande de programmation :
Fonctionne de façon similaire à la première syntaxe de la commande **Request**, excepté que la réponse de l'utilisateur est toujours interprétée comme une chaîne de caractères. Par contre, la commande **Request** interprète la réponse comme une expression, à moins que l'utilisateur ne la saisisse entre guillemets ("").

Définissez un programme :

```
Define requestStr_demo()=Prgm
  RequestStr "Votre nom :",name,0
  Disp "La réponse comporte ",dim
(name)," caractères."
EndPrgm
```

Exécutez le programme et saisissez une réponse :

```
requestStr_demo()
```

Remarque : Vous pouvez utiliser la commande **RequestStr** dans un programme créé par l'utilisateur, mais pas dans une fonction.

Pour arrêter un programme qui contient une commande **RequestStr** dans une boucle infinie :

- **Calculatrice**: Maintenez la touche  enfoncée et appuyez plusieurs fois sur .
- **Windows®** : Maintenez la touche **F12** enfoncée et appuyez plusieurs fois sur **Entrée**.
- **Macintosh®** : Maintenez la touche **F5** enfoncée et appuyez plusieurs fois sur **Entrée**.
- **iPad®** : L'application affiche une invite. Vous pouvez continuer à patienter ou annuler.

Remarque : Voir également **Request**, page 167.



Après avoir sélectionné **OK**, le résultat affiché est le suivant (notez que si l'argument *Indic.Aff* a pour valeur **0**, le message d'invite et la réponse de l'utilisateur ne s'affichent pas dans l'historique) :

```
requestStr_demo()
```

La réponse comporte 5 caractères.

Return

Return [*Expr*]

Donne *Expr* comme résultat de la fonction S'utilise dans les blocs **Func...EndFunc**.

Remarque : Vous pouvez utiliser **Return** sans argument dans un bloc **Prgm...EndPrgm** pour quitter un programme

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

```
Define factorial (nn)=
Func
Local answer,counter
1 → answer
For counter,1,nn
answer· counter → answer
EndFor
Return answer
EndFunc

factorial (3)
```

6

right()Catalogue > **right(Liste1[, Num])** ⇒ *liste* $\text{right}(\{1, 3, -2, 4\}, 3)$ $\{3, -2, 4\}$

Donne les *Nomb* éléments les plus à droite de la liste *Liste1*.

Si *Nomb* est absent, on obtient *Liste1*.

right(chaineSrce[, Nomb]) ⇒ *chaîne* $\text{right}(\text{"Hello"}, 2)$ "lo"

Donne la chaîne formée par les *Nomb* caractères les plus à droite de la chaîne de caractères *chaineSrce*.

Si *Nomb* est absent, on obtient *chaineSrce*.

right(Comparaison) ⇒ *expression* $\text{right}(x < 3)$ 3

Donne le membre de droite d'une équation ou d'une inéquation.

rk23 ()Catalogue > 

rk23(Expr, Var, depVar, {Var0, VarMax}, depVar0, VarStep [, diftol])
⇒ *matrice*

Équation différentielle :

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

rk23(SystemOfExpr, Var, ListOfDepVars, {Var0, VarMax}, ListOfDepVars0, VarStep[, diftol]) ⇒ *matrice*

$$\text{rk23}(0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1)$$

0.	1.	2.	3.	4.
10.	10.9367	11.9493	13.042	14.2

Pour afficher le résultat entier, appuyez sur $\blacktriangle$, puis utilisez les touches $\blacktriangleleft$ et $\blacktriangleright$ pour déplacer le curseur.

Même équation avec *TolErr* définie à $1.E-6$

Utilise la méthode de Runge-Kutta pour résoudre le système d'équations.

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

with *depVar*(*Var0*)=*depVar0* pour l'intervalle [*Var0*, *VarMax*]. Retourne une matrice dont la première ligne définit les valeurs de sortie de *Var*, définies à partir de *IncVar*. La deuxième ligne définit la valeur du premier composant de la solution aux valeurs *Var* correspondantes, etc.

$$\text{rk23}(0.001 \cdot y \cdot (100 - y), t, y, \{0, 100\}, 10, 1, 1.E-6)$$

0.	1.	2.	3.	4.
10.	10.9367	11.9495	13.0423	14.2189

Comparez le résultat ci-dessus avec la solution exacte CAS obtenue en utilisant *deSolve()* et *seqGen()* :

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

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

Expr représente la partie droite qui définit l'équation différentielle.

SystèmeExpr correspond aux côtés droits qui définissent le système des équations différentielles (en fonction de l'ordre des variables dépendantes de la *ListeVarDép*).

ListeExpr est la liste des côtés droits qui définissent le système des équations différentielles (en fonction de l'ordre des variables dépendantes de la *ListeVarDép*).

Var est la variable indépendante.

ListeVarDép est la liste des variables dépendantes.

{Var0, MaxVar} est une liste à deux éléments qui indique la fonction à intégrer, comprise entre *Var0* et *MaxVar*.

ListeVar0Dép est la liste des valeurs initiales pour les variables dépendantes.

Si *IncVar* est un nombre différent de zéro, $\text{signe}(\text{IncVar}) = \text{signe}(\text{MaxVar} - \text{Var0})$ et les solutions sont retournées pour $\text{Var0} + i * \text{IncVar}$ pour tout $i=0,1,2,\dots$ tel que $\text{Var0} + i * \text{IncVar}$ soit dans $[\text{var0}, \text{MaxVar}]$ (il est possible qu'il n'existe pas de solution en *MaxVar*).

si *IncVar* est un nombre égal à zéro, les solutions sont retournées aux valeurs *Var* "Runge-Kutta".

tolErr correspond à la tolérance d'erreur (valeur par défaut 0,001).

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

$$\{10., 10.9367, 11.9494, 13.0423, 14.2189, 15.49\}$$

Système d'équations :

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

avec $y1(0) = 2$ et $y2(0) = 5$

$$\text{rk23} \left(\begin{cases} -y1 + 0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{cases}, t, \{y1, y2\}, \{0, 5\}, \{2, 5\}, 1 \right)$$

0.	1.	2.	3.	4.
2.	1.94103	4.78694	3.25253	1.82848
5.	16.8311	12.3133	3.51112	6.27245

root()

root(Expr) $\Rightarrow$ racine

root(Expr1, Expr2) $\Rightarrow$ racine

root(Expr) renvoie la racine carrée de *Expr*.

$\sqrt[3]{8}$	2
$\sqrt[3]{3}$	$\frac{1}{3^3}$
$\sqrt[3]{3.}$	1.44225

root()

root(*Expr1*, *Expr2*) renvoie la racine *Expr2*-ième de *Expr1*. *Expr1* peut être un nombre réel ou complexe en virgule flottante, un entier ou une constante rationnelle complexe, ou une expression symbolique générale

Remarque : Voir aussi **Modèle Racine n-
ième**, page 2.

rotate()

$$\text{rotate}(\text{Entier } l[, \text{NbreRotations}]) \Rightarrow \text{entier}$$

En mode base Bin :

Permute les bits de la représentation binaire d'un entier. Vous pouvez saisir *Entier1* dans un système de numération quelconque ; il est converti automatiquement en une forme binaire 64 bits signée. Si *Entier1* est trop important pour être codé, il est ramené à l'aide d'une congruence dans la plage appropriée. Pour plus d'informations, consultez la section ► **Base2**, page 19.

Si *nbreRotations* est positif, la permutation circulaire s'effectue vers la gauche Si *nbreRotations* est négatif, la permutation circulaire s'effectue vers la droite La valeur par défaut est -1 (permutation circulation de un bit vers la droite)

Par exemple, dans une permutation circulaire vers la droite :

Chaque bit est permuté vers la droite.

0b00000000000001111010110000110101

Le bit le plus à droite passe à la position la plus à gauche.

donne :

0b100000000000000111101011000011010

Le résultat s'affiche suivant le mode Base utilisé.

$$\text{rotate}(\text{Liste1}, \text{NbreRotations}) \Rightarrow \text{liste}$$
[illegible]

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

En mode base Hex :

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h80000000000001E3
rotate(0h78E,2)	0h1E38

Important : Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h (zéro, pas la lettre O).

En mode base Dec :

rotate()Catalogue > 

Donne une copie de *Liste1* dont les éléments ont été permutés circulairement vers la gauche ou vers la droite de *nbreRotations* éléments. Ne modifie en rien *Liste1*.

Si *nbreRotations* est positif, la permutation circulaire s'effectue vers la gauche. Si *nbreRotations* est négatif, la permutation circulaire s'effectue vers la droite. La valeur par défaut est -1 (permutation circulaire de un bit vers la droite).

rotate(*Chaîne1*,*nbreRotations*) $\Rightarrow$ *chaîne*

Donne une copie de *Chaîne1* dont les caractères ont été permutés circulairement vers la gauche ou vers la droite de *nbreRotations* caractères. Ne modifie en rien *Chaîne1*.

Si *nbreRotations* est positif, la permutation circulaire s'effectue vers la gauche. Si *nbreRotations* est négatif, la permutation circulaire s'effectue vers la droite. La valeur par défaut est -1 (permutation vers la droite d'un caractère).

$\text{rotate}(\{1,2,3,4\})$	$\{4,1,2,3\}$
$\text{rotate}(\{1,2,3,4\}, -2)$	$\{3,4,1,2\}$
$\text{rotate}(\{1,2,3,4\}, 1)$	$\{2,3,4,1\}$

$\text{rotate}(\text{"abcd"})$	"dabc"
$\text{rotate}(\text{"abcd"}, -2)$	"cdab"
$\text{rotate}(\text{"abcd"}, 1)$	"bcda"

round()Catalogue > 

round(*Expr1*,*chiffres*) $\Rightarrow$ *expression*

Arrondit l'argument au nombre de chiffres *n* spécifié après la virgule.

chiffres doit être un entier compris dans la plage 0–12. Si *chiffres* est absent, affiche l'argument arrondi à 12 chiffres significatifs.

Remarque : Le mode d'affichage des chiffres peut affecter le résultat affiché.

round(*List1*,*chiffres*) $\Rightarrow$ *liste*

Donne la liste des éléments arrondis au nombre de chiffres spécifié.

$\text{round}(1.234567, 3)$	1.235
-----------------------------	-------

$\text{round}(\{\pi, \sqrt{2}, \ln(2)\}, 4)$	$\{3.1416, 1.4142, 0.6931\}$
--	------------------------------

round()Catalogue > **round**(*Matrice1*, *chiffres*) ⇒ *matrice*

Donne une matrice des éléments arrondis au nombre de chiffres *n* spécifié..

$$\text{round}\left(\begin{bmatrix} \ln(5) & \ln(3) \\ \pi & e^1 \end{bmatrix}, 1\right) \quad \begin{bmatrix} 1.6 & 1.1 \\ 3.1 & 2.7 \end{bmatrix}$$

rowAdd()Catalogue > **rowAdd**(*Matrice1*, *rIndex1*, *rIndex2*) ⇒ *matrice*

Donne une copie de *Matrice1* obtenue en remplaçant dans la matrice la ligne *IndexL2* par la somme des lignes *IndexL1* et *IndexL2*

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

$$\text{rowAdd}\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}, 1, 2\right) \quad \begin{bmatrix} a & b \\ a+c & b+d \end{bmatrix}$$

rowDim()Catalogue > **rowDim**(*Matrix*) ⇒ *expression*

Donne le nombre de lignes de *Matrice*.

Remarque : Voir aussi **colDim()**, page 29.

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

$$\text{rowDim}(m1) \quad 3$$

normeLigCatalogue > **rowNorm**(*Matrice*) ⇒ *expression*

Donne le maximum des sommes des valeurs absolues des éléments de chaque ligne de *Matrice*.

Remarque : La matrice utilisée ne doit contenir que des éléments numériques. Voir aussi **colNorm()** page 30.

$$\text{rowNorm}\left(\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}\right) \quad 25$$

rowSwap()Catalogue > **rowSwap**(*Matrice1*, *IndexL1*, *IndexL2*) ⇒ *matrice*

Donne la matrice *Matrice1* obtenue en échangeant les lignes *IndexL1* et *IndexL2*.

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

$$\text{rowSwap}(mat, 1, 3) \quad \begin{bmatrix} 5 & 6 \\ 3 & 4 \\ 1 & 2 \end{bmatrix}$$

rref()

Catalogue >

rref(*Matrice1* [, *Tol*]) $\Rightarrow$ *matrice*

Donne la réduite de Gauss-Jordan de *Matrice1*.

$$\text{rref}\left(\begin{pmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 & 0 & \frac{66}{71} \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & \frac{-62}{71} \end{pmatrix}$$

L'argument facultatif *Tol* permet de considérer comme nul tout élément de la matrice dont la valeur absolue est inférieure à *Tol*. Cet argument n'est utilisé que si la matrice contient des nombres en virgule flottante et ne contient pas de variables symbolique sans valeur affectée. Dans le cas contraire, *Tol* est ignoré

$$\text{rref}\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- Si vous utilisez Auto ou Approché **sur** Approché, les calculs sont exécutés en virgule flottante
- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit : $5E-14 \cdot \max(\dim(\text{Matrice1})) \cdot \text{rowNorm}(\text{Matrice1})$

Remarque : Voir aussi **ref()** page 165.

S

sec()

Touche

sec(*Expr1*) $\Rightarrow$ *expression*

En mode Angle en degrés :

sec(*Liste1*) $\Rightarrow$ *liste*

Affiche la sécante de *Expr1* ou retourne la liste des sécantes des éléments de *Liste1*.

$$\text{sec}(45) = \sqrt{2}$$

$$\text{sec}(\{1, 2, 3, 4\}) = \left\{ \frac{1}{\cos(1)}, 1.00081, \frac{1}{\cos(4)} \right\}$$

Remarque : l'argument est interprété comme la mesure d'un angle en degrés, en grades ou en radians, suivant le mode angulaire en cours d'utilisation. Vous pouvez utiliser $^{\circ}$, $^{\text{G}}$ ou $^{\text{r}}$ pour préciser l'unité employée temporairement pour le calcul.

sec⁻¹()Touche **sec⁻¹(Expr1)** ⇒ *expression***sec⁻¹(Liste1)** ⇒ *liste*

Affiche l'angle dont la sécante correspond à *Expr1* ou retourne la liste des arcs sécantes des éléments de *Liste1*.

Remarque : donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arcsec (...)**.

En mode Angle en degrés :

$$\text{sec}^{-1}(1) \quad 0$$

En mode Angle en grades :

$$\text{sec}^{-1}(\sqrt{2}) \quad 50$$

En mode Angle en radians :

$$\text{sec}^{-1}(\{1, 2, 5\}) \quad \left\{ 0, \frac{\pi}{3}, \cos^{-1}\left(\frac{1}{5}\right) \right\}$$

sech()Catalogue > **sech(Expr1)** ⇒ *expression***sech(Liste1)** ⇒ *liste*

Affiche la sécante hyperbolique de *Expr1* ou retourne la liste des sécantes hyperboliques des éléments de *liste1*.

$$\text{sech}(3) \quad \frac{1}{\cosh(3)}$$

$$\text{sech}(\{1, 2, 3, 4\}) \quad \left\{ \frac{1}{\cosh(1)}, 0.198522, \frac{1}{\cosh(4)} \right\}$$

sech⁻¹()Catalogue > **sech⁻¹(Expr1)** ⇒ *expression***sech⁻¹(Liste1)** ⇒ *liste*

Donne l'argument sécante hyperbolique de *Expr1* ou retourne la liste des arguments sécantes hyperboliques des éléments de *Liste1*.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arcsech (...)**.

En mode Angle en radians et en mode Format complexe Rectangulaire :

$$\text{sech}^{-1}(1) \quad 0$$

$$\text{sech}^{-1}(\{1, 2, 2, 1\}) \quad \left\{ 0, \frac{2 \cdot \pi}{3} \cdot i, 8 \cdot 10^{-15} + 1.07448 \cdot i \right\}$$

Send

Menu hub

Send*exprOrString1* [, *exprOrString2*] ...

Exemple : allumer l'élément bleu de la DEL RGB intégrée pendant 0,5 seconde.

Commande de programmation : envoie une ou plusieurs TI-Innovator™ Hub commandes à un hub connecté.

exprOrString doit être une commande TI-Innovator™ Hub valide. En général, *exprOrString* contient une commande "SET ..." pour contrôler un appareil ou une commande "READ ..." pour demander des données.

Les arguments sont envoyés au hub les uns après les autres.

Remarque : vous pouvez utiliser la commande **Send** dans un programme défini par l'utilisateur, mais pas dans une fonction.

Remarque : voir également **Get** (page 87), **GetStr** (page 95) et **eval()** (page 69).

Send "SET COLOR.BLUE ON TIME .5"

Done

Exemple : demander la valeur actuelle du capteur intégré du niveau de lumière du hub. Une commande **Get** récupère la valeur et l'affecte à la variable *lightval*.

Send "READ BRIGHTNESS"

Done

Get *lightval*

Done

lightval

0.347922

Exemple : envoyer une fréquence calculée au haut-parleur intégré du hub. Utilisez la variable spéciale *iostr.SendAns* pour afficher la commande du hub avec l'expression évaluée.

n:=50

50

m:=4

4

Send "SET SOUND eval(*m* · *n*)"

Done

iostr.SendAns

"SET SOUND 200"

seq()

Catalogue >

seq(*Expr*, *Var*, *Début*, *Fin*, *Incrément*) ⇒ liste

Incrémente la valeur de *Var* comprise entre *Début* et *Fin* en fonction de l'incrément (*Inc*) spécifié et affiche le résultat sous forme de liste. Le contenu initial de *Var* est conservé après l'application de **seq()**.

La valeur par défaut de *Inc* = 1.

$\text{seq}\left(n^2, n, 1, 6\right)$	$\{1, 4, 9, 16, 25, 36\}$
$\text{seq}\left(\frac{1}{n}, n, 1, 10, 2\right)$	$\left\{1, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}\right\}$
$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)$	$\frac{1968329}{1270080}$

Remarque: Pour afficher un résultat approximatif,

Unité : Appuyez sur  .

Windows® : Appuyez sur **Ctrl+Entrée**.

Macintosh® : Appuyez sur **⌘+Entrée**.

iPad® : Maintenez la touche **Entrée** enfoncée et sélectionnez .

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)$	1.54977
--	---------

seqGen(*Expr*, *Var*, *VarDép*, {*Var0*, *MaxVar*}, [*ListeValeursInit*], *IncVar* [, *ValeurMax*])) ⇒ *liste*

Génère une liste de valeurs pour la suite *VarDép*(*Var*)=*Expr* comme suit :
Incrémente la valeur de la variable indépendante *Var* de *Var0* à *MaxVar* par pas de *IncVar*, calcule *VarDép*(*Var*) pour les valeurs correspondantes de *Var* en utilisant *Expr* et *ListeValeursInit*, puis retourne le résultat sous forme de liste.

seqGen(*ListeOuSystèmeExpr*, *Var*, *ListeVarDép*, {*Var0*, *MaxVar*}, [*MatriceValeursInit*], *IncVar* [, *ValeurMax*])) ⇒ *matrice*

Génère une matrice de valeurs pour un système (ou une liste) de suites *ListeVarDép* (*Var*)=*ListeOuSystèmeExpr* comme suit :
Incrémente la valeur de la variable indépendante *Var* de *Var0* à *MaxVar* par pas de *IncVar*, calcule *ListeVarDép* (*Var*) pour les valeurs correspondantes de *Var* en utilisant *ListeOuSystèmeExpr* et *MatriceValeursInit*, puis retourne le résultat sous forme de matrice.

Le contenu initial de *Var* est conservé après l'application de **seqGen**().

La valeur par défaut de *IncVar* est 1.

Génère les cinq premières valeurs de la suite $u(n) = u(n-1)^2/2$, avec $u(1)=2$ et $IncVar=1$.

$$\text{seqGen}\left(\frac{(u(n-1))^2}{n}, n, u, \{1, 5\}, \{2\}\right)$$

$$\left\{2, 2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\}$$

Exemple avec *Var0*=2 :

$$\text{seqGen}\left(\frac{u(n-1)+1}{n}, n, u, \{2, 5\}, \{3\}\right)$$

$$\left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\}$$

Exemple dans lequel la valeur initiale est symbolique :

$$\text{seqGen}\{u(n-1)+2, n, u, \{1, 5\}, \{a\}\}$$

$$\{a, a+2, a+4, a+6, a+8\}$$

Système de deux suites :

$$\text{seqGen}\left\{\left\{\frac{1}{n}, \frac{u2(n-1)}{2} + u1(n-1)\right\}, n, \{u1, u2\}, \{1, 5\}, \left[-\frac{1}{2}\right]\right\}$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

Remarque : L'élément vide () dans la matrice de valeurs initiales ci-dessus est utilisé pour indiquer que la valeur initiale de $u1(n)$ est calculée en utilisant la suite explicite $u1(n)=1/n$.

seqn(*Expr*{*u*, *n* [, *ListeValeursInit*], *nMax* [, *ValeurMax*]})) ⇒ *liste*

Génère les cinq premières valeurs de la suite $u(n) = u(n-1)/2$, avec $u(1)=2$.

Génère une liste de valeurs pour la suite $u(n)=Expr(u, n)$ comme suit : Incrémente n de 1 à $nMax$ par incrément de 1, calcule $u(n)$ pour les valeurs correspondantes de n en utilisant $Expr(u, n)$ et *ListeValeursInit*, puis retourne le résultat sous forme de liste.

seqn(*Expr*(n [, $nMax$ [, *ValeurMax*]]) $\Rightarrow$ *liste*

Génère une liste de valeurs pour la suite $u(n)=Expr(n)$ comme suit : Incrémente n de 1 à $nMax$ par incrément de 1, calcule $u(n)$ pour les valeurs correspondantes de n en utilisant $Expr(n)$, puis retourne le résultat sous forme de liste.

Si $nMax$ n'a pas été défini, il prend la valeur 2500.

Si $nMax=0$ n'a pas été défini, $nMax$ prend la valeur 2500.

Remarque : **seqn()** appelé **seqGen()** avec $n0=1$ et $Inc n=1$

$$\text{seqn}\left(\frac{u(n-1)}{n}, \{2\}, 6\right) = \left\{2, 1, \frac{1}{3}, \frac{1}{12}, \frac{1}{60}, \frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2}, 6\right) = \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}\right\}$$

series()

series(*Expr1*, *Var*, *Ordre* [, *Point*]) $\Rightarrow$ *expression*

series(*Expr1*, *Var*, *Ordre* [, *Point*]) | *Var*>*Point* $\Rightarrow$ *expression*

series(*Expr1*, *Var*, *Ordre* [, *Point*]) | *Var*<*Point* $\Rightarrow$ *expression*

Donne un développement en série généralisé, tronqué, de *Expr1* en *Point* jusqu'au degré *Ordre*. *Ordre* peut être un nombre rationnel quelconque. Les puissances de (*Var* – *Point*) peuvent avoir des exposants négatifs et/ou fractionnaires. Les coefficients de ces puissances peuvent inclure les logarithmes de (*Var* – *Point*) et d'autres fonctions de *Var* dominés par toutes les puissances de (*Var* – *Point*) ayant le même signe d'exposant.

$$\text{series}\left(\frac{1-\cos(x-1)}{(x-1)^2}, x, 4, 1\right) = \frac{1}{2} - \frac{(x-1)^2}{24} + \frac{(x-1)^4}{720}$$

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

$$\text{series}\left(\left(1 + \frac{1}{n}\right)^n, n, 2, \infty\right) = e - \frac{e}{2 \cdot n} + \frac{11 \cdot e}{24 \cdot n^2}$$

$$\text{series}\left(\tan^{-1}\left(\frac{1}{x}\right), x, 5\right) | x > 0 = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5}$$

$$\text{series}\left(\int \frac{\sin(x)}{x} dx, x, 6\right) = x - \frac{x^3}{18} + \frac{x^5}{600}$$

$$\text{series}\left(\int_0^x \sin(x \cdot \sin(t)) dt, x, 7\right) = \frac{x^3}{2} - \frac{x^5}{24} + \frac{29 \cdot x^7}{720}$$

La valeur par défaut de *Point* est 0.
Point peut être ∞ ou $-\infty$, auxquels cas le développement s'effectue jusqu'au degré *Ordre* en $1/(Var - Point)$.

$$\text{series}\left(\left(1+e^x\right)^2, x, 2, 1\right)$$

$$(e+1)^2+2 \cdot e \cdot (e+1) \cdot (x-1)+e \cdot (2 \cdot e+1) \cdot (x-1)^2$$

series(...) donne "**series(...)**" s'il ne parvient pas à déterminer la représentation, comme pour les singularités essentielles **sin(1/z)** en $z=0$, $e^{-1/z}$ en $z=0$ ou e^z en $z = \infty$ ou $-\infty$.

Si la série ou une de ses dérivées présente une discontinuité en *Point*, le résultat peut contenir des sous-expressions de type **sign(...)** ou **abs(...)** pour une variable réelle ou $(-1)^{\text{floor}}(\dots \text{angle}(\dots) \dots)$ pour une variable complexe, qui se termine par « _ ». Si vous voulez utiliser la série uniquement pour des valeurs supérieures ou inférieures à *Point*, vous devez ajouter l'élément approprié « | *Var* > *Point* », « | *Var* < *Point* », « | » « *Var* ≥ *Point* » ou « *Var* ≤ *Point* » pour obtenir un résultat simplifié.

series() peut donner des approximations symboliques pour des intégrales indéfinies et définies pour lesquelles autrement, il n'est pas possible d'obtenir des solutions symboliques.

series() est appliqué à chaque élément d'une liste ou d'une matrice passée en 1er argument.

series() est une version généralisée de **taylor()**.

Comme illustré dans l'exemple ci-contre, le développement des routines de calcul du résultat donnée par **series(...)** peut réorganiser l'ordre des termes de sorte que le terme dominant ne soit pas le terme le plus à gauche.

Remarque : voir aussi **dominantTerm()**, page 63.

setMode(EntierNomMode, EntierRéglage) $\Rightarrow$ entier

setMode(liste) $\Rightarrow$ liste des entiers

Accessible uniquement dans une fonction ou un programme.

setMode(EntierNomMode, EntierRéglage) règle provisoirement le mode *EntierNomMode* sur le nouveau réglage *EntierRéglage* et affiche un entier correspondant au réglage d'origine de ce mode. Le changement est limité à la durée d'exécution du programme/de la fonction.

EntierNomMode indique le mode que vous souhaitez régler. Il doit s'agir d'un des entiers du mode du tableau ci-dessous.

EntierRéglage indique le nouveau réglage pour ce mode. Il doit s'agir de l'un des entiers de réglage indiqués ci-dessous pour le mode spécifique que vous configurez.

setMode(liste) permet de modifier plusieurs réglages. *liste* contient les paires d'entiers de mode et d'entiers de réglage. **setMode**(liste) affiche une liste dont les paires d'entiers représentent les modes et réglages d'origine.

Si vous avez enregistré tous les réglages du mode avec **getMode**(0) $\rightarrow$ var, **setMode**(var) permet de restaurer ces réglages jusqu'à fermeture du programme ou de la fonction. Voir **getMode**(), page 94.

Remarque : Les réglages de mode actuels sont transférés dans les sous-programmes appelés. Si un sous-programme change un quelconque réglage du mode, le changement sera perdu dès le retour au programme appelant.

Affiche la valeur approchée de π à l'aide du réglage par défaut de Afficher chiffres, puis affiche π avec le réglage Fixe 2. Vérifiez que la valeur par défaut est bien restaurée après l'exécution du programme.

Define <i>prog1</i> ()=Prgm	Done
Disp approx(π)	
setMode(1,16)	
Disp approx(π)	
EndPrgm	
<i>prog1</i> ()	
	3.14159
	3.14
	Done

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Nom du mode	Entier du mode	Entiers de réglage
Afficher chiffres	1	1=Flottant, 2=Flottant 1, 3=Flottant 2, 4=Flottant 3, 5=Flottant 4, 6=Flottant 5, 7=Flottant 6, 8=Flottant 7, 9=Flottant 8, 10=Flottant 9, 11=Flottant 10, 12=Flottant 11, 13=Flottant 12, 14=Fixe 0, 15=Fixe 1, 16=Fixe 2, 17=Fixe 3, 18=Fixe 4, 19=Fixe 5, 20=Fixe 6, 21=Fixe 7, 22=Fixe 8, 23=Fixe 9, 24=Fixe 10, 25=Fixe 11, 26=Fixe 12
Angle	2	1=Radian, 2=Degré, 3=Grade
Format Exponentiel	3	1=Normal, 2=Scientifique, 3=Ingénieur
Réel ou Complexe	4	1=Réel, 2=Rectangulaire, 3=Polaire
Auto ou Approché	5	1=Auto, 2=Approché, 3=Exact
Format Vecteur	6	1=Rectangulaire, 2=Cylindrique, 3=Sphérique
Base	7	1=Décimale, 2=Hexadécimale, 3=Binaire

shift()

shift(Entier1[,nbreDécal])⇒entier

Décale les bits de la représentation binaire d'un entier. *Entier1* peut être un entier de n'importe quelle base ; il est automatiquement converti sous forme binaire (64 bits) signée. Si *Entier1* est trop important pour être codé sur 32 bits, il est ramené à l'aide d'une congruence dans la plage appropriée. Pour de plus amples informations, voir ►Base2, page 19.

En mode base Bin :

```
shift(0b1111010110000110101)
                                0b111101011000011010
shift(256,1)                    0b1000000000
```

En mode base Hex :

```
shift(0h78E)                    0h3C7
shift(0h78E,-2)                 0h1E3
shift(0h78E,2)                  0h1E38
```

Si *nbreDécal* est positif, le décalage s'effectue vers la gauche. Si *nbreDécal* est négatif, le décalage s'effectue vers la droite. La valeur par défaut est -1 (décalage d'un bit vers la droite).

Dans un décalage vers la droite, le dernier bit est éliminé et 0 ou 1 est inséré à gauche selon le premier bit. Dans un décalage vers la gauche, le premier bit est éliminé et 0 est inséré comme dernier bit.

Par exemple, dans un décalage vers la droite :

Tous les bits sont décalés vers la droite.

0b0000000000000111101011000011010

Insère 0 si le premier bit est un 0

ou 1 si ce bit est un 1.

donne :

0b000000000000000111101011000011010

Le résultat est affiché selon le mode Base utilisé. Les zéros de tête ne sont pas affichés.

shift(ListeI [,nbreDécal])⇒*liste*

Donne une copie de *ListeI* dont les éléments ont été décalés vers la gauche ou vers la droite de *nbreDécal* éléments. Ne modifie en rien *ListeI*.

Si *nbreDécal* est positif, le décalage s'effectue vers la gauche. Si *nbreDécal* est négatif, le décalage s'effectue vers la droite. La valeur par défaut est -1 (décalage d'un élément vers la droite).

Les éléments introduits au début ou à la fin de *liste* par l'opération de décalage sont remplacés par undef (non défini).

shift(ChaîneI [,nbreDécal])⇒*chaîne*

Important : pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h (zéro, pas la lettre O).

En mode 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("abcd")	" abc"
shift("abcd",-2)	" ab"
shift("abcd",1)	"bcd "

Donne une copie de *Chaîne1* dont les caractères ont été décalés vers la gauche ou vers la droite de *nbreDécal* caractères. Ne modifie en rien *Chaîne1*.

Si *nbreDécal* est positif, le décalage s'effectue vers la gauche. Si *nbreDécal* est négatif, le décalage s'effectue vers la droite. La valeur par défaut est -1 (décalage d'un caractère vers la droite).

Les caractères introduits au début ou à la fin de *Chaîne* par l'opération de décalage sont remplacés par un espace.

sign()

sign(*Expr1*) ⇒ *expression*

$\text{sign}(-3.2)$	-1.
---------------------	-----

sign(*Liste1*) ⇒ *liste*

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

sign(*Matrice1*) ⇒ *matrice*

$\text{sign}(1 + x)$	1
------------------------	---

Pour une *Expr1* réelle ou complexe, donne *Expr1*/**abs**(*Expr1*) si *Expr1* ≠ 0.

En mode Format complexe Réel :

Donne 1 si l'expression *Expression1* est positive.

$\text{sign}\left(\begin{bmatrix} -3 & 0 & 3 \end{bmatrix}\right)$	$\begin{bmatrix} -1 & \pm 1 & 1 \end{bmatrix}$
--	--

Donne -1 si l'expression *Expr1* est négative.

sign(0) donne -1 en mode Format complexe Réel ; sinon, donne lui-même.

sign(0) représente le cercle d'unité dans le domaine complexe.

Dans le cas d'une liste ou d'une matrice, donne les signes de tous les éléments.

simult()

simult(*matriceCoeff*, *vecteurConst*[, *Tol*]) ⇒ *matrice*

Résolution de x et y :

$$x + 2y = 1$$

Donne un vecteur colonne contenant les solutions d'un système d'équations.

$$3x + 4y = -1$$

Remarque : voir aussi **linSolve()**, page 115.

matriceCoeff doit être une matrice carrée qui contient les coefficients des équations.

vecteurConst doit avoir le même nombre de lignes (même dimension) que *matriceCoeff* et contenir le second membre.

L'argument facultatif *Tol* permet de considérer comme nul tout élément de la matrice dont la valeur absolue est inférieure à *Tol*. Cet argument n'est utilisé que si la matrice contient des nombres en virgule flottante et ne contient pas de variables symbolique sans valeur affectée. Dans le cas contraire, *Tol* est ignoré.

- Si vous réglez le mode **Auto ou Approché (Approximate)** sur Approché (Approximate), les calculs sont exécutés en virgule flottante.
- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit :
 $5E-14 \cdot \max(\dim(\text{matriceCoeff})) \cdot \text{rowNorm}(\text{matriceCoeff})$

simult(matriceCoeff, matriceConst[, Tol]) ⇒ *matrice*

Permet de résoudre plusieurs systèmes d'équations, ayant les mêmes coefficients mais des seconds membres différents.

Chaque colonne de *matriceConst* représente le second membre d'un système d'équations. Chaque colonne de la matrice obtenue contient la solution du système correspondant.

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

La solution est $x=-3$ et $y=2$.

Résolution :

$$ax + by = 1$$

$$cx + dy = 2$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \rightarrow \text{matx1} \quad \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\text{simult}\left(\text{matx1}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) \quad \begin{bmatrix} -(2 \cdot b - d) \\ a \cdot d - b \cdot c \\ 2 \cdot a - c \\ a \cdot d - b \cdot c \end{bmatrix}$$

Résolution :

$$x + 2y = 1$$

$$3x + 4y = -1$$

$$x + 2y = 2$$

$$3x + 4y = -3$$

$$\text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 & 2 \\ -1 & -3 \end{bmatrix}\right) \quad \begin{bmatrix} -3 & -7 \\ 2 & \frac{9}{2} \end{bmatrix}$$

Pour le premier système, $x=-3$ et $y=2$. Pour le deuxième système, $x=-7$ et $y=9/2$.

Expr ▶sin

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>sin.

Exprime *Expr* en sinus. Il s'agit d'un opérateur de conversion utilisé pour l'affichage. Cet opérateur ne peut être utilisé qu'à la fin d'une ligne.

▶sin réduit toutes les puissances modulo $\sin(\dots) 1 - \sin(\dots)^2$ de sorte que les puissances de $\sin(\dots)$ restantes ont des exposants dans (0, 2). Le résultat ne contient donc pas $\cos(\dots)$ si et seulement si $\cos(\dots)$ dans l'expression donnée s'applique uniquement aux puissances paires.

Remarque : L'opérateur de conversion n'est pas autorisé en mode Angle Degré ou Grade. Avant de l'utiliser, assurez-vous d'avoir défini le mode Angle Radian et de l'absence de références explicites à des angles en degrés ou en grades dans *Expr*.

$$\frac{(\cos(x))^2 \blacktriangleright \sin}{1 - (\sin(x))^2}$$

sin()

Touche

sin(*Expr1*) ⇒ *expression*

sin(*Liste1*) ⇒ *liste*

sin(*Expr1*) donne le sinus de l'argument sous forme d'expression.

sin(*Liste1*) donne la liste des sinus des éléments de *Liste1*.

Remarque : l'argument est interprété comme mesure d'angle en degrés, en grades ou en radians, suivant le mode angulaire sélectionné. Vous pouvez utiliser °, G ou r pour ignorer temporairement le mode angulaire sélectionné.

En mode Angle en degrés :

$$\frac{\sin\left(\frac{\pi}{4}\right)}{\sin(45)} = \frac{\sqrt{2}}{2}$$

$$\frac{\sin(\{0,60,90\})}{\left\{0,\frac{\sqrt{3}}{2},1\right\}}$$

En mode Angle en grades :

$$\frac{\sin(50)}{2} = \frac{\sqrt{2}}{2}$$

En mode Angle en radians :

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

$$\sin(45^\circ) = \frac{\sqrt{2}}{2}$$

sin(matriceCarrée1) ⇒ matriceCarrée

Donne le sinus de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du sinus de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

En mode Angle en radians :

$$\sin\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) = \begin{bmatrix} 0.9424 & -0.04542 & -0.031999 \\ -0.045492 & 0.949254 & -0.020274 \\ -0.048739 & -0.00523 & 0.961051 \end{bmatrix}$$

sin⁻¹(Expr1) ⇒ expression

sin⁻¹(Liste1) ⇒ liste

sin⁻¹(Expr1) donne l'arc sinus de *Expr1* sous forme d'expression.

sin⁻¹(List1) donne la liste des arcs sinus des éléments de *Liste1*.

Remarque : donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arcsin(...)**.

sin⁻¹(matriceCarrée1) ⇒ matriceCarrée

Donne l'argument arc sinus de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de l'argument arc sinus de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

En mode Angle en degrés :

$$\sin^{-1}(1) = 90$$

En mode Angle en grades :

$$\sin^{-1}(1) = 100$$

En mode Angle en radians :

$$\sin^{-1}(\{0, 0.2, 0.5\}) = \{0, 0.201358, 0.523599\}$$

En mode Angle en radians et en mode Format complexe Rectangulaire :

$$\sin^{-1}\left(\begin{bmatrix} 1 & 5 \\ 4 & 2 \end{bmatrix}\right) = \begin{bmatrix} -0.174533 - 0.12198 \cdot i & 1.74533 - 2.35591 \cdot i \\ 1.39626 - 1.88473 \cdot i & 0.174533 - 0.593162 \cdot i \end{bmatrix}$$

sinh(*Expr1*) $\Rightarrow$ *expression* $\sinh(1.2)$ 1.50946**sinh**(*Liste1*) $\Rightarrow$ *liste* $\sinh(\{0,1.2,3.\})$ $\{0,1.50946,10.0179\}$ **sinh** (*Expr1*) donne le sinus hyperbolique de l'argument sous forme d'expression.**sinh** (**Liste1**) donne la liste des sinus hyperboliques des éléments de *Liste1*.**sinh**(*matriceCarrée1*) $\Rightarrow$ *matriceCarrée*

Donne le sinus hyperbolique de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du sinus hyperbolique de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

En mode Angle en radians :

$\sinh\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$	$\begin{bmatrix} 360.954 & 305.708 & 239.604 \\ 352.912 & 233.495 & 193.564 \\ 298.632 & 154.599 & 140.251 \end{bmatrix}$

sinh⁻¹(*Expr1*) $\Rightarrow$ *expression* $\sinh^{-1}(0)$ 0**sinh⁻¹**(*Liste1*) $\Rightarrow$ *liste* $\sinh^{-1}(\{0,2.1,3\})$ $\{0,1.48748,\sinh^{-1}(3)\}$ **sinh⁻¹**(*Expr1*) donne l'argument sinus hyperbolique de l'argument sous forme d'expression.**sinh⁻¹**(*Liste1*) donne la liste des arguments sinus hyperboliques des éléments de *Liste1*.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arcsinh** (...).

sinh⁻¹(*matriceCarrée1*) $\Rightarrow$ *matriceCarrée*

Donne l'argument sinus hyperbolique de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de l'argument sinus hyperbolique de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

En mode Angle en radians :

$\sinh^{-1}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$	$\begin{bmatrix} 0.041751 & 2.15557 & 1.1582 \\ 1.46382 & 0.926568 & 0.112557 \\ 2.75079 & -1.5283 & 0.57268 \end{bmatrix}$

matrice Carrée *l* doit être diagonalisable.
Le résultat contient toujours des chiffres
en virgule flottante.

SinReg

SinReg *X*, *Y* [, [*Itérations*],[*Période*] [,
Catégorie, *Inclure*]]

Effectue l'ajustement sinusoïdal sur les listes
X et *Y*. Un récapitulatif du résultat est stocké
dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le même
nombre de lignes, à l'exception de *Inclure*.

X et *Y* sont des listes de variables
indépendantes et dépendantes.

Itérations spécifie le nombre maximum
d'itérations (1 à 16) utilisées lors de ce
calcul. S'il est omis, la valeur par défaut est
8. On obtient généralement une meilleure
précision en choisissant une valeur élevée,
mais cela augmente également le temps de
calcul, et vice versa.

Période spécifie une période estimée. S'il est
omis, la différence entre les valeurs de *X*
doit être égale et en ordre séquentiel. Si
vous spécifiez la *Période*, les différences
entre les valeurs de *x* peuvent être inégales.

Catégorie est une liste de codes de
catégories pour les couples *X* et *Y*
correspondants..

Inclure est une liste d'un ou plusieurs codes
de catégories. Seuls les éléments dont le
code de catégorie figure dans cette liste sont
inclus dans le calcul.

Le résultat obtenu avec **SinReg** est toujours
exprimé en radians, indépendamment du
mode Angle sélectionné.

Pour plus d'informations concernant les
éléments vides dans une liste, reportez-vous
à "Éléments vides", page 281.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a \cdot \sin(bx+c)+d$
stat.a, stat.b, stat.c, stat.d	Coefficients d'ajustement
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

solve()

Catalogue > 

solve(*Équation*, *Var*) $\Rightarrow$ *expression booléenne*

solve(*Équation*, *Var*=*Init*) $\Rightarrow$ *expression booléenne*

solve(*Inéquation*, *Var*) $\Rightarrow$ *expression booléenne*

Résout dans R une équation ou une inéquation en *Var*. L'objectif est de trouver toutes les solutions possibles. Toutefois, il peut arriver avec certaines équations ou inéquations que le nombre de solutions soit infini.

Les solutions peuvent ne pas être des solutions réelles finies pour certaines valeurs des paramètres.

Avec le réglage Auto du mode **Auto ou Approché (Approximate)**, l'objectif est de trouver des solutions exactes quand elles sont concises et de compléter l'opération par des recherches itératives de calcul approché lorsque des solutions exactes ne peuvent pas être trouvées.

$$\text{solve}\left(a \cdot x^2 + b \cdot x + c = 0, x\right)$$

$$x = \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a} \text{ or } x = \frac{-\left(\sqrt{b^2 - 4 \cdot a \cdot c} + b\right)}{2 \cdot a}$$

$$\text{Ans}|a=1 \text{ and } b=1 \text{ and } c=1$$

$$x = \frac{-1}{2} + \frac{\sqrt{3}}{2} \cdot i \text{ or } x = \frac{-1}{2} - \frac{\sqrt{3}}{2} \cdot i$$

$$\text{solve}\left((x-a) \cdot e^x = x \cdot (x-a), x\right)$$

$$x=a \text{ or } x=-0.567143$$

En raison de l'annulation par défaut du plus grand commun diviseur du numérateur et du dénominateur des rapports, les solutions trouvées peuvent ne pas être valides.

$$(x+1) \cdot \frac{x-1}{x-1} + x - 3 \quad 2 \cdot x - 2$$

Pour les inéquations de type $\geq$, $\leq$, $<$ ou $>$, il est peut probable de trouver des solutions explicites, sauf si l'inéquation est linéaire et ne contient que Var .

$$\text{solve}(5 \cdot x - 2 \geq 2 \cdot x, x) \quad x \geq \frac{2}{3}$$

Avec le réglage Exact du mode **Auto ou Approché (Approximate)**, les portions qui ne peuvent pas être résolues sont données sous forme d'équation ou d'inéquation implicite.

$$\text{exact}(\text{solve}((x-a) \cdot e^x = x \cdot (x-a), x)) \quad e^x + x = 0 \text{ or } x = a$$

Utilisez l'opérateur "sachant que" (« | ») pour restreindre l'intervalle de la solution et/ou des autres variables rencontrées dans l'équation ou l'inéquation. Lorsqu'une solution est trouvée dans un intervalle, vous pouvez utiliser les opérateurs d'inéquation pour exclure cet intervalle des recherches suivantes.

En mode Angle en radians :

$$\text{solve}\left(\tan(x) = \frac{1}{x}, x\right) | x > 0 \text{ and } x < 1 \quad x = 0.860334$$

false est affiché si aucune solution réelle n'est trouvée. true est affiché si **solve()** parvient à déterminer que tout réel est solution de l'équation ou de l'inéquation.

$$\begin{array}{ll} \text{solve}(x = x + 1, x) & \text{false} \\ \text{solve}(x = x, x) & \text{true} \end{array}$$

Dans la mesure où **solve()** donne toujours un résultat booléen, vous pouvez utiliser « and », « or » et « not » pour combiner les résultats de **solve()** entre eux ou avec d'autres expressions booléennes.

$$2 \cdot x - 1 \leq 1 \text{ and solve}(x^2 \neq 9, x) \quad x \neq -3 \text{ and } x \leq 1$$

Les solutions peuvent contenir une nouvelle constante non définie de type nj , où j correspond à un entier compris entre 1 et 255. Ces variables désignent un entier arbitraire.

En mode Angle en radians :

$$\text{solve}(\sin(x) = 0, x) \quad x = n \cdot \pi$$

En mode Réel, les puissances fractionnaires possédant un dénominateur impair font uniquement référence à la branche principale. Sinon, les expressions à plusieurs branches, telles que les puissances fractionnaires, les logarithmes et les fonctions trigonométriques inverses font uniquement référence à la branche principale. Par conséquent, **solve()** donne uniquement des solutions correspondant à cette branche réelle ou principale.

Remarque : voir aussi **cSolve()**, **cZeros()**, **nSolve()** et **zeros()**.

solve(*Éqn1* and *Éqn2* [*and*...],
VarOunit1, *VarOunit2* [, ...
) ⇒ *expression booléenne*

solve(*SystèmeÉq*, *VarOunit1*,
VarOunit2 [, ...
) ⇒ *expression booléenne*

solve(*{Eqn1, Eqn2 [...]}* *{VarOunit1*,
VarOunit2 [, ...
) ⇒ *expression booléenne*

Donne les solutions réelles possibles d'un système d'équations algébriques, où chaque *VarOunit* définit une variable du système à résoudre.

Vous pouvez séparer les équations par l'opérateur **and** ou entrer un système d'équations *SystèmeÉq* en utilisant un modèle du Catalogue. Le nombre d'arguments *VarOunit* doit correspondre au nombre d'équations. Vous pouvez également spécifier une condition initiale pour les variables. Chaque *VarOunit* doit utiliser le format suivant :

variable

– ou –

$\text{solve}\left(x^{\frac{1}{3}}=1, x\right)$	$x=1$
$\text{solve}(\sqrt{x}=2, x)$	false
$\text{solve}(\sqrt{x}=2, x)$	$x=4$

$\text{solve}(y=x^2-2 \text{ and } x+2 \cdot y=1, \{x, y\})$
$x=-\frac{3}{2} \text{ and } y=\frac{1}{4} \text{ or } x=1 \text{ and } y=-1$

variable = nombre réel ou non réel

Par exemple, x est autorisé, de même que $x=3$.

Si toutes les équations sont polynomiales et si vous NE spécifiez PAS de condition initiale, **solve()** utilise la méthode d'élimination lexicale Gröbner/Buchberger pour tenter de trouver toutes les solutions réelles.

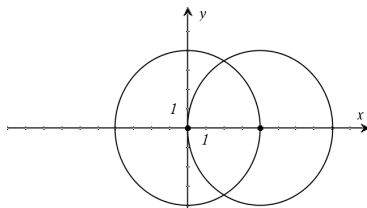
Par exemple, si vous avez un cercle de rayon r centré à l'origine et un autre cercle de rayon r centré, au point où le premier cercle coupe l'axe des x positifs. Utilisez **solve()** pour trouver les intersections.

Comme l'illustre r dans l'exemple ci-contre, les systèmes d'équations polynomiales peuvent avoir des variables auxquelles on peut affecter par la suite des valeurs numériques.

Vous pouvez également utiliser des variables qui n'apparaissent pas dans les équations. Par exemple, vous pouvez utiliser z comme variable pour développer l'exemple précédent et avoir deux cylindres parallèles sécants de rayon r .

La résolution du problème montre comment les solutions peuvent contenir des constantes arbitraires de type ck , où k est un suffixe entier compris entre 1 et 255.

Pour les systèmes d'équations polynomiales, le temps de calcul et l'utilisation de la mémoire peuvent considérablement varier en fonction de l'ordre dans lequel les inconnues sont spécifiées. Si votre choix initial ne vous satisfait pas pour ces raisons, vous pouvez modifier l'ordre des variables dans les équations et/ou la liste des variables *VarOulnit*.



$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y\}\right)$$

$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ or } x=\frac{r}{2} \text{ and } y=-\frac{\sqrt{3}\cdot r}{2}$$

$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2, \{x,y,z\}\right)$$

$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ and } z=c1 \text{ or } x=\frac{r}{2} \text{ and } y=-\frac{\sqrt{3}\cdot r}{2} \text{ and } z=c1$$

Pour afficher le résultat entier, appuyez sur $\blacktriangle$, puis utilisez les touches $\blacktriangleleft$ et $\blacktriangleright$ pour déplacer le curseur.

Si vous choisissez de ne pas spécifier de condition et s'il l'une des équations n'est pas polynomiale dans l'une des variables, mais que toutes les équations sont linéaires par rapport à toutes les variables, **solve()** utilise l'élimination gaussienne pour tenter de trouver toutes les solutions réelles.

Si un système d'équations n'est ni polynomial par rapport à toutes ses variables ni linéaire par rapport aux inconnues, **solve()** cherche au moins une solution en utilisant une méthode itérative approchée. Pour cela, le nombre d'inconnues doit être égal au nombre d'équations et toutes les autres variables contenues dans les équations doivent pouvoir être évaluées à des nombres.

Chaque variable du système commence à sa valeur supposée, si elle existe ; sinon, la valeur de départ est 0.0.

Utilisez des valeurs initiales pour rechercher des solutions supplémentaires, une par une. Pour assurer une convergence correcte, une valeur initiale doit être relativement proche de la solution.

$$\text{solve}\left(x+e^z \cdot y=1 \text{ and } x-y=\sin(z), \{x, y\}\right)$$

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

$$\text{solve}\left(e^z \cdot y=1 \text{ and } -y=\sin(z), \{y, z\}\right)$$

$$y=2.812\text{E-}10 \text{ and } z=21.9911 \text{ or } y=0.001871$$

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

$$\text{solve}\left(e^z \cdot y=1 \text{ and } -y=\sin(z), \{y, z=2 \cdot \pi\}\right)$$

$$y=0.001871 \text{ and } z=6.28131$$

SortA

SortA *Liste1* [, *Liste2*] [, *Liste3*] ...

SortA *Vecteur1* [, *Vecteur2*] [, *Vecteur3*] ...

Trie les éléments du premier argument en ordre croissant.

Si d'autres arguments sont présents, trie les éléments de chacun d'entre eux de sorte que leur nouvelle position corresponde aux nouvelles positions des éléments dans le premier argument.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
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	$\{4,3,2,1\}$

Tous les arguments doivent être des noms de listes ou de vecteurs et tous doivent être de même dimension.

Les éléments vides compris dans le premier argument ont été déplacés au bas de la liste. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

SortD

SortD *Liste1* [, *Liste2*] [, *Liste3*] ...

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
---------------------------------	---------------

SortD *Vecteur1* [, *Vecteur2*] [, *Vecteur3*]
...

$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
---------------------------------	---------------

SortD list1,list2	Done
-------------------	------

Identique à **SortA**, mais **SortD** trie les éléments en ordre décroissant.

list1	$\{4,3,2,1\}$
-------	---------------

list2	$\{3,4,1,2\}$
-------	---------------

Les éléments vides compris dans le premier argument ont été déplacés au bas de la liste. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

►Sphere

Vecteur ►**Sphere**

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>**Sphere**.

Affiche le vecteur ligne ou colonne en coordonnées sphériques [ρ $\angle\theta$ $\angle\phi$].

Vecteur doit être un vecteur ligne ou colonne de dimension 3.

Remarque : ►**Sphere** est uniquement une instruction d'affichage et non une fonction de conversion. On ne peut l'utiliser qu'à la fin d'une ligne.

Remarque: Pour afficher un résultat approximatif,

Unité : Appuyez sur  .

Windows® : Appuyez sur **Ctrl+Entrée**.

Macintosh® : Appuyez sur +**Entrée**.

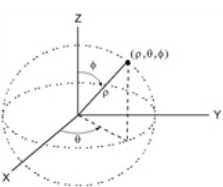
iPad® : Maintenez la touche **Entrée** enfoncée et sélectionnez .

$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$ ►Sphere
$\begin{bmatrix} 3.74166 & \angle 1.10715 & \angle 0.640522 \end{bmatrix}$

$\begin{pmatrix} 2 & \angle \frac{\pi}{4} & 3 \end{pmatrix}$ ►Sphere
$\begin{bmatrix} 3.60555 & \angle 0.785398 & \angle 0.588003 \end{bmatrix}$

Appuyez sur .

$$\left(\begin{matrix} 2 & \angle \frac{\pi}{4} & 3 \end{matrix} \right) \text{►Sphere}$$
$$\left[\sqrt{13} \quad \angle \frac{\pi}{4} \quad \angle \sin^{-1} \left(\frac{2 \cdot \sqrt{13}}{13} \right) \right]$$



sqrt()

sqrt(*Expr1*)⇒*expression*

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

sqrt(*Liste1*)⇒*liste*

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

Donne la racine carrée de l'argument.

Dans le cas d'une liste, donne la liste des racines carrées des éléments de *Liste1*.

Remarque : voir aussi **Modèle Racine carrée**, page 1.

stat.results

Affiche le résultat d'un calcul statistique.

Les résultats sont affichés sous forme d'ensemble de paires nom-valeur. Les noms spécifiques affichés varient suivant la fonction ou commande statistique la plus récemment calculée ou exécutée.

Vous pouvez copier un nom ou une valeur et la coller à d'autres emplacements.

Remarque : ne définissez pas de variables dont le nom est identique à celles utilisées dans le cadre de l'analyse statistique. Dans certains cas, cela peut générer une erreur. Les noms de variables utilisés pour l'analyse statistique sont répertoriés dans le tableau ci-dessous.

 $xlist = \{1, 2, 3, 4, 5\}$
 $\{1, 2, 3, 4, 5\}$
 $ylist = \{4, 8, 11, 14, 17\}$
 $\{4, 8, 11, 14, 17\}$
LinRegMx $xlist, ylist, 1$: stat.results

"Title"	"Linear Regression (mx+b)"
"RegEqn"	"m*x+b"
"m"	3.2
"b"	1.2
"r ² "	0.996109
"r"	0.998053
"Resid"	"{...}"

stat.values	"Linear Regression (mx+b)"
	"m*x+b"
	3.2
	1.2
	0.996109
	0.998053
	"{-0.4, 0.4, 0.2, 0., -0.2}"

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR ²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r ²	stat.SSY
stat.b1	stat.dfInteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.ox	stat.SSRow
stat.b5	stat.DW	stat.MSInteract	stat.oy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.ox1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.ox2	stat.UpperVal
stat.b8	stat.F	stat.n	stat.Σx	stat.Σ̄
stat.b9	stat.FBlock	stat.ŷ	stat.Σx ²	stat.Σ̄1
stat.b10	stat.Fcol	stat.ŷ1	stat.Σxy	stat.Σ̄2
stat.bList	stat.FInteract	stat.ŷ2	stat.Σy	stat.Σ̄Diff
stat.χ ²	stat.FreqReg	stat.ŷDiff	stat.Σy ²	stat.Σ̄List

stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.CompList	stat.LowerVal	stat.PValCol	stat.SEPred	stat. $\bar{y}$
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat. $\hat{y}$
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat. $\hat{y}$ List
stat.CUpper	stat.MaxY	stat.Q1X	stat.sp	stat.YReg
stat.CUpperList	stat.ME	stat.Q1Y	stat.SS	
stat.d	stat.MedianX			

Remarque : Chaque fois que l'application Tableur & listes calcule des résultats statistiques, les variables du groupe « stat. » sont copiées dans un groupe « stat#. », où # est un nombre qui est incrémenté automatiquement. Cela vous permet de conserver les résultats précédents tout en effectuant plusieurs calculs.

stat.values

Catalogue > 

stat.values

Voir l'exemple donné pour
stat.results.

Affiche une matrice des valeurs calculées pour la fonction ou commande statistique la plus récemment calculée ou exécutée.

Contrairement à **stat.results**, **stat.values** omet les noms associés aux valeurs.

Vous pouvez copier une valeur et la coller à d'autres emplacements.

stDevPop()

Catalogue > 

stDevPop(*Liste*[,
listeFréq]) $\Rightarrow$ expression

Donne l'écart-type de population des éléments de *Liste*.

Chaque élément de la liste *listeFréq* totalise le nombre d'occurrences de l'élément correspondant de *Liste*.

En mode Angle en radians et en modes Auto :

$$\text{stDevPop}\left(\left\{a, b, c\right\}\right) = \frac{\sqrt{2 \cdot \left(a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2\right)}}{3}$$

$$\text{stDevPop}\left(\left\{1, 2, 5, -6, 3, -2\right\}\right) = \frac{\sqrt{465}}{6}$$

$$\text{stDevPop}\left(\left\{1.3, 2.5, -6.4\right\}, \left\{3, 2, 5\right\}\right) = 4.11107$$

Remarque : *Liste* doit contenir au moins deux éléments. Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

stDevPop(*Matrice1*[,
matriceFréq]) $\Rightarrow$ *matrice*

Donne un vecteur ligne des écarts-types de population des colonnes de *Matrice1*.

Chaque élément de *matriceFréq* totalise le nombre d'occurrences de l'élément correspondant de *Matrice1*.

Remarque : *Matrice1* doit contenir au moins deux lignes. Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

$$\text{stDevPop}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right) \left[\frac{4 \cdot \sqrt{6}}{3} \quad \frac{\sqrt{78}}{3} \quad \frac{2 \cdot \sqrt{6}}{3} \right]$$

$$\text{stDevPop}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) [2.52608 \quad 5.21506]$$

stDevSamp()

stDevSamp(*Liste*[,
listeFréq]) $\Rightarrow$ *expression*

Donne l'écart-type d'échantillon des éléments de *Liste*.

Chaque élément de la liste *listeFréq* totalise le nombre d'occurrences de l'élément correspondant de *Liste*.

Remarque : *Liste* doit contenir au moins deux éléments. Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

stDevSamp(*Matrice1*[,
matriceFréq]) $\Rightarrow$ *matrice*

Donne un vecteur ligne des écarts-types de population des colonnes de *Matrice1*.

Chaque élément de *matriceFréq* totalise le nombre d'occurrences de l'élément correspondant de *Matrice1*.

$$\text{stDevSamp}(\{a, b, c\})$$

$$\frac{\sqrt{3 \cdot (a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2)}}{3}$$

$$\text{stDevSamp}(\{1, 2, 5, -6, 3, -2\}) \quad \frac{\sqrt{62}}{2}$$

$$\text{stDevSamp}(\{1.3, 2.5, -6.4\}, \{3, 2, 5\})$$

$$4.33345$$

$$\text{stDevSamp}\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{bmatrix}\right) [4 \quad \sqrt{13} \quad 2]$$

$$\text{stDevSamp}\left(\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}\right) [2.7005 \quad 5.44695]$$

Remarque : *MatriceI* doit contenir au moins deux lignes. Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

Stop

Catalogue >

Stop

Commande de programmation : Ferme le programme.

Stop n'est pas autorisé dans les fonctions.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

$i:=0$	0
Define $progI()$ =Prgm	Done
For $i,1,10,1$	
If $i=5$	
Stop	
EndFor	
EndPrgm	
$progI()$	Done
i	5

Store

Voir → (store), page 261.

string()

Catalogue >

string(Expr)⇒chaîne

Simplifie *Expr* et donne le résultat sous forme de chaîne de caractères.

$\text{string}(1.2345)$	"1.2345"
$\text{string}(1+2)$	"3"
$\text{string}(\cos(x)+\sqrt{3})$	"cos(x)+√(3)"

subMat()

Catalogue >

subMat(MatriceI[, colDébut] [, colDébut] [, ligneFin] [, colFin])
⇒matrice

Donne la matrice spécifiée, extraite de *MatriceI*.

Valeurs par défaut : *ligneDébut*=1, *colDébut*=1, *ligneFin*=dernière ligne, *colFin*=dernière colonne.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
$\text{subMat}(m1,2,1,3,2)$	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
$\text{subMat}(m1,2,2)$	$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

sum()

Catalogue > **sum(Liste[, Début[, Fin]])** ⇒ *expression*Donne la somme des éléments de *Liste*.*Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage d'éléments.

$\text{sum}\left(\{1,2,3,4,5\}\right)$	15
$\text{sum}\left(\{a,2\cdot a,3\cdot a\}\right)$	$6\cdot a$
$\text{sum}(\text{seq}(n,n,1,10))$	55
$\text{sum}\left(\{1,3,5,7,9\},3\right)$	21

Tout argument vide génère un résultat vide. Les éléments vides de *Liste* sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

sum(Matrice1[, Début[, Fin]]) ⇒ *matrice*Donne un vecteur ligne contenant les sommes des éléments de chaque colonne de *Matrice1*.*Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage de colonnes.

$\text{sum}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}\right)$	$[5 \ 7 \ 9]$
$\text{sum}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}\right)$	$[12 \ 15 \ 18]$
$\text{sum}\left(\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix},2,3\right)$	$[11 \ 13 \ 15]$

Tout argument vide génère un résultat vide. Les éléments vides de *Matrice1* sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

sumIf()

Catalogue > **sumIf(Liste,Critère[, ListeSommes])** ⇒ *valeur*Affiche la somme cumulée de tous les éléments dans *Liste* qui répondent au *critère* spécifié. Vous pouvez aussi spécifier une autre liste, *ListeSommes*, pour fournir les éléments à cumuler.*Liste* peut être une expression, une liste ou une matrice. *ListeSommes*, si spécifiée, doit avoir la/les même(s) dimension (s) que *Liste*.

$\text{sumIf}(\{1,2,e,3,\pi,4,5,6\},2.5<?<4.5)$	$e+\pi+7$
$\text{sumIf}(\{1,2,3,4\},2<?<5,\{10,20,30,40\})$	70

Le critère peut être :

- Une valeur, une expression ou une chaîne. Par exemple, **34** cumule uniquement les éléments dans *Liste* qui donnent la valeur 34.
- Une expression booléenne contenant le symbole ? comme paramètre substituable à tout élément. Par exemple, **?<10** cumule uniquement les éléments de *Liste* qui sont inférieurs à 10.

Lorsqu'un élément de *Liste* répond au critère, il est ajouté à la somme cumulée. Si vous incluez *ListeSommes*, c'est l'élément correspondant dans *ListeSommes* qui est ajouté à la somme.

Dans l'application Tableur & listes, vous pouvez utiliser une plage de cellules à la place de *Liste* et *ListeSommes*.

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

Remarque : voir également **countif()**, page 39.

sumSeq()

Voir $\Sigma()$, page 250.

system()

Catalogue > 

system(Eqn1 [, Eqn2 [, Eqn3 [, ...]])

system(Expr1 [, Expr2 [, Expr3 [, ...]])

$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=8 \end{cases}, x, y\right)$	$x=4 \text{ and } y=-4$
---	-------------------------

Donne un système d'équations, présenté sous forme de liste. Vous pouvez également créer un système d'équation en utilisant un modèle.

Remarque : voir aussi **Système d'équations**, page 3.

T (transposée)

Catalogue > *MatrixI* $\Rightarrow$ *matrice*

Donne la transposée de la conjuguée de *MatriceI*.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @t.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T$	$\begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$
$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^T$	$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$
$\begin{bmatrix} 1+i & 2+i \\ 3+i & 4+i \end{bmatrix}^T$	$\begin{bmatrix} 1-i & 3-i \\ 2-i & 4-i \end{bmatrix}$

tan()

Touche *tan(ExprI)* $\Rightarrow$ *expression*

En mode Angle en degrés :

tan(ListeI) $\Rightarrow$ *liste*

tan(ExprI) donne la tangente de l'argument.

tan(ListI) donne la liste des tangentes des éléments de *ListeI*.

Remarque : l'argument est interprété comme mesure d'angle en degrés, en grades ou en radians, suivant le mode angulaire sélectionné. Vous pouvez utiliser °, G ou r pour ignorer temporairement le mode Angle sélectionné.

En mode Angle en grades :

$\tan\left(\frac{\pi}{4}\right)$	1
$\tan(45)$	1
$\tan(\{0,60,90\})$	$\{0,\sqrt{3},\text{undef}\}$

En mode Angle en radians :

$\tan\left(\frac{\pi}{4}\right)$	1
$\tan(45^\circ)$	1
$\tan\left(\left\{\pi,\frac{\pi}{3},\pi,\frac{\pi}{4}\right\}\right)$	$\{0,\sqrt{3},0,1\}$

tan(matriceMatriceI) $\Rightarrow$ *matriceCarrée*

En mode Angle en radians :

Donne la tangente de la matrice *matriceCarréeI*. Ce calcul est différent du calcul de la tangente de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

tan()Touche 

matriceCarrée1 doit être diagonalisable.
Le résultat contient toujours des chiffres
en virgule flottante.

$$\tan \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$$

tan⁻¹()Touche 

tan⁻¹(Expr1)⇒expression

En mode Angle en degrés :

tan⁻¹(Liste1)⇒liste

$$\tan^{-1}(1) \quad 45$$

tan⁻¹(Expr1) donne l'arc tangente de
Expr1.

En mode Angle en grades :

tan⁻¹(List1) donne la liste des arcs
tangentes des éléments de *Liste1*.

$$\tan^{-1}(1) \quad 50$$

Remarque : donne le résultat en degrés,
en grades ou en radians, suivant le mode
angulaire utilisé.

En mode Angle en radians :

Remarque : vous pouvez insérer cette
fonction à partir du clavier en entrant
arctan (...).

$$\tan^{-1}(\{0,0,2,0,5\}) \quad \{0,0,197396,0,463648\}$$

tan⁻¹(matriceCarrée1)⇒matriceCarrée

En mode Angle en radians :

Donne l'arc tangente de la matrice
matriceCarrée1. Ce calcul est différent
du calcul de l'arc tangente de chaque
élément. Pour plus d'informations sur la
méthode de calcul, reportez-vous à **cos()**.

$$\tan^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} \begin{bmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{bmatrix}$$

matriceCarrée1 doit être diagonalisable.
Le résultat contient toujours des chiffres
en virgule flottante.

tangentLine()Catalogue > **tangentLine** $(Expr1, Var, Point) \Rightarrow expression$ **tangentLine** $(Expr1, Var=Point) \Rightarrow expression$

Donne la tangente de la courbe représentée par *Expr1* au point spécifié par *Var=Point*.

Assurez-vous de ne pas avoir affecté une valeur à la variable indépendante. Par exemple, si $f1(x):=5$ et $x:=3$, alors **tangentLine**($f1(x), x, 2$) donne « faux ».

$\text{tangentLine}(x^2, x, 1)$	$2 \cdot x - 1$
$\text{tangentLine}((x-3)^2 - 4, x=3)$	-4
$\text{tangentLine}\left(\frac{1}{x^3}, x=0\right)$	$x=0$
$\text{tangentLine}(\sqrt{x^2-4}, x=2)$	undef
$x:=3; \text{tangentLine}(x^2, x, 1)$	5

tanh()Catalogue > **tanh**(*Expr1*) $\Rightarrow expression$ **tanh**(*Liste1*) $\Rightarrow liste$

tanh(*Expr1*) donne la tangente hyperbolique de l'argument.

tanh(*Liste1*) donne la liste des tangentes hyperboliques des éléments de *Liste1*.

tanh(*matriceCarrée1*) $\Rightarrow matriceCarrée$

Donne la tangente hyperbolique de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de la tangente hyperbolique de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

$\tanh(1.2)$	0.833655
$\tanh(\{0, 1\})$	$\{0, \tanh(1)\}$

En mode Angle en radians :

$\tanh\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$	$\begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$		
---	---	--	--

tanh⁻¹()Catalogue > **tanh⁻¹**(*Expr1*) $\Rightarrow expression$ **tanh⁻¹**(*Liste1*) $\Rightarrow liste$

tanh⁻¹(*Expr1*) donne l'argument tangente hyperbolique de l'argument sous forme d'expression.

En mode Format complexe Rectangulaire :

$\tanh^{-1}(0)$	0
$\tanh^{-1}(\{1, 2, 1, 3\})$	$\left\{ \text{undef}, 0.518046 - 1.5708 \cdot i, \frac{\ln(2)}{2} - \frac{\pi}{2} \cdot i \right\}$

tanh⁻¹()

Catalogue > 

tanh⁻¹(Liste1) donne la liste des arguments tangentes hyperboliques des éléments de *Liste1*.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **arctanh (...)**.

tanh⁻¹

(matriceCarrée1)⇒matriceCarrée

Donne l'argument tangente hyperbolique de *matriceCarrée1*. Ce calcul est différent du calcul de l'argument tangente hyperbolique de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

En mode Angle en radians et en mode Format complexe Rectangulaire :

$$\begin{array}{l} \text{tanh}^{-1}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right) \\ \left[\begin{array}{ll} -0.099353+0.164058 \cdot i & 0.267834-1.4908 \\ -0.087596-0.725533 \cdot i & 0.479679-0.94730 \\ 0.511463-2.08316 \cdot i & -0.878563+1.7901 \end{array} \right] \end{array}$$

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

taylor()

Catalogue > 

taylor(Expr1, Var, Ordre[, Point])⇒expression

Donne le polynôme de Taylor demandé. Le polynôme comprend des termes non nuls de degrés entiers compris entre zéro et *Ordre* dans (*Var* moins *Point*). **taylor()** donne lui-même en l'absence de développement limité de cet ordre ou si l'opération exige l'utilisation d'exposants négatifs ou fractionnaires. Utilisez des opérations de substitution et/ou de multiplication temporaire par une puissance de (*Var* moins *Point*) pour déterminer un développement généralisé.

Par défaut, la valeur de *Point* est égale à zéro et il s'agit du point de développement.

$$\begin{array}{l} \text{taylor}(e^{\sqrt{x}}, x, 2) \qquad \text{taylor}(e^{\sqrt{x}}, x, 2, 0) \\ \text{taylor}(e^{t, t, 4})/t = \sqrt{x} \qquad \frac{3}{24} + \frac{x^2}{6} + \frac{x}{2} + \sqrt{x} + 1 \\ \text{taylor}\left(\frac{1}{x \cdot (x-1)}, x, 3\right) \qquad \text{taylor}\left(\frac{1}{x \cdot (x-1)}, x, 3, 0\right) \\ \text{expand}\left(\frac{\text{taylor}\left(\frac{x}{x \cdot (x-1)}, x, 4\right)}{x}, x\right) \\ -x^3 - x^2 - x - \frac{1}{x} - 1 \end{array}$$

tCdf()

Catalogue > 

tCdf(LimitInf, LimitSup, df)⇒nombre si

LimitInf et *LimitSup* sont des nombres, *liste* si *LimitInf* et *LimitSup* sont des listes

Calcule la fonction de répartition de la loi de Student-*t* à *df* degrés de liberté entre *LimitInf* et *LimitSup*.

Pour $P(X \leq upBound)$, définissez *lowBound* = $-\infty$.

tCollect()

tCollect(*Expr1*) $\Rightarrow$ *expression*

Donne une expression dans laquelle les produits et les puissances entières des sinus et des cosinus sont convertis en une combinaison linéaire de sinus et de cosinus de multiples d'angles, de sommes d'angles et de différences d'angles. La transformation convertit les polynômes trigonométriques en une combinaison linéaire de leurs harmoniques.

Quelquefois, **tCollect()** permet d'atteindre vos objectifs lorsque la simplification trigonométrique n'y parvient pas. **tCollect()** fait l'inverse des transformations effectuées par **tExpand()**. Parfois, l'application de **tExpand()** à un résultat de **tCollect()**, ou vice versa, permet en deux étapes de simplifier une expression.

$tCollect(\cos(\alpha)^2)$	$\frac{\cos(2 \cdot \alpha) + 1}{2}$
$tCollect(\sin(\alpha) \cdot \cos(\beta))$	$\frac{\sin(\alpha - \beta) + \sin(\alpha + \beta)}{2}$

tExpand()

tExpand(*Expr1*) $\Rightarrow$ *expression*

$tExpand(\sin(3 \cdot \phi))$	$4 \cdot \sin(\phi) \cdot \cos(\phi)^2 - \sin(\phi)$
$tExpand(\cos(\alpha - \beta))$	$\cos(\alpha) \cdot \cos(\beta) + \sin(\alpha) \cdot \sin(\beta)$

Donne une expression dans laquelle les sinus et les cosinus de multiples entiers d'angles, de sommes d'angles et de différences d'angles sont développés. En raison de la présence de l'identité $(\sin(x))^2 + (\cos(x))^2 = 1$, il existe plusieurs résultats équivalents possibles. Par conséquent, un résultat peut différer d'un autre résultat affiché dans d'autres publications.

Quelquefois, **tExpand()** permet d'atteindre vos objectifs lorsque le développement trigonométrique n'y parvient pas. **tExpand()** tend à faire l'inverse des transformations effectuées par **tCollect()**. Parfois, l'application de **tCollect()** à un résultat de **tExpand()**, ou vice versa, permet en deux étapes de simplifier une expression.

Remarque : la conversion en degrés par $\pi/180$ peut interférer avec la capacité de **tExpand()** de reconnaître les formes pouvant être développées. Pour de meilleurs résultats, **tExpand()** doit être utilisé en mode Angle en radians.

Text

Text*chaîneinvite*[, *IndicAff*]

Commande de programmation : Marque une pause dans l'exécution du programme et affiche la chaîne de caractères *chaîneinvite* dans une boîte de dialogue.

Lorsque l'utilisation sélectionne **OK**, l'exécution du programme se poursuit.

L'argument optionnel *IndicAff* peut correspondre à n'importe quelle expression.

- Si *IndicAff* est omis ou a pour valeur **1**, le message est ajouté à l'historique de l'application Calculs.
- Si *IndicAff* a pour valeur **0**, le message n'est pas ajouté à l'historique.

Définissez un programme qui marque une pause afin d'afficher cinq nombres aléatoires dans une boîte de dialogue.

Dans le modèle `Prgm...EndPrgm`, validez chaque ligne en appuyant sur  à la place de . Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée**.

```
Define text_demo()=Prgm
  For i,1,5
    strinfo:="Random number " &
    string(rand(i))
    Text strinfo
  EndFor
```

Si le programme nécessite une réponse saisie par l'utilisateur, voir **Request**, page 167 ou **RequestStr**, page 169.

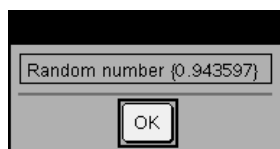
Remarque : vous pouvez utiliser cette commande dans un programme créé par l'utilisateur, mais pas dans une fonction.

EndPrgm

Exécutez le programme :

text_demo()

Exemple de boîte de dialogue :



tInterval *Liste*[,*Fréq*[,*CLevel*]]

(Entrée de liste de données)

tInterval $\bar{x}$,*sx*,*n*[,*CLevel*]

(Récapitulatif des statistiques fournies en entrée)

Calcule un intervalle de confiance *t*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance pour une moyenne inconnue de population
stat. $\bar{x}$	Moyenne d'échantillon de la série de données suivant la loi normale aléatoire
stat.ME	Marge d'erreur

Variable de sortie	Description
stat.df	Degrés de liberté
stat.σx	Écart-type d'échantillon
stat.n	Taille de la série de données avec la moyenne d'échantillon

tInterval_2Samp

Catalogue > 

tInterval_2Samp *Liste1, Liste2[, Fréq1
[, Fréq2[, CLevel[, Group]]]]*

(Entrée de liste de données)

tInterval_2Samp $\bar{x}1, sx1, n1, \bar{x}2, sx2, n2$
[, CLevel[, Group]]

(Récapitulatif des statistiques fournies en entrée)

Calcule un intervalle de confiance t sur 2 échantillons. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Group=1 met en commun les variances ;
Groupe=0 ne met pas en commun les variances.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance contenant la probabilité du niveau de confiance de la loi
stat. $\bar{x}1 - \bar{x}2$	Moyennes d'échantillon des séries de données suivant la loi normale aléatoire
stat.ME	Marge d'erreur
stat.df	Degrés de liberté
stat. $\bar{x}1$, stat. $\bar{x}2$	Moyennes d'échantillon des séries de données suivant la loi normale aléatoire
stat.σx1, stat.σx2	Écarts-types d'échantillon pour <i>Liste 1</i> et <i>Liste 2</i>

Variable de sortie	Description
stat.n1, stat.n2	Nombre d'échantillons dans les séries de données
stat.sp	Écart-type du groupe. Calculé lorsque Group = YES.

tmpCnv()

Catalogue > 

tmpCnv(*Expr*_°*unitéTemp1*, _
°*unitéTemp2*) ⇒ *expression* _
°*unitéTemp2*

tmpCnv(100_°C_°F)	212_°F
tmpCnv(32_°F_°C)	0_°C
tmpCnv(0_°C_°K)	273.15_°K
tmpCnv(0_°F_°R)	459.67_°R

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **deltaTmpCnv** (...).

Remarque : vous pouvez utiliser le Catalogue pour sélectionner des unités de température.

Convertit un écart de température (la différence entre deux valeurs de température) spécifié par *Expr* d'une unité à une autre. Les unités de température utilisables sont :

_°CCelsius

_°FFahrenheit

_°KKelvin

_°RRankine

Pour taper °, sélectionnez ce symbole dans le Jeu de symboles ou entrez @d.

Pour taper _, appuyez sur  .

Par exemple, 100_°C donne 212_°F.

Pour convertir un écart de température, utilisez **ΔtmpCnv()**.

ΔtmpCnv()

Catalogue > 

ΔtmpCnv(*Expr*_°*unitéTemp1*, _
°*unitéTemp2*) ⇒ *expression* _
°*unitéTemp2*

Pour taper Δ, sélectionnez-le dans les symboles du Catalogue.

Δ tmpCnv()

Catalogue > 

Convertit un écart de température (la différence entre deux valeurs de température) spécifié par *Expr* d'une unité à une autre. Les unités de température utilisables sont :

_°CCelsius

_°FFahrenheit

_°KKelvin

_°RRankine

Pour taper °, sélectionnez-le dans les symboles du Catalogue.

Pour taper _, appuyez sur  .

Des écarts de 1_°C et 1_°K représentent la même grandeur, de même que 1_°F et 1_°R. Par contre, un écart de 1_°C correspond au 9/5 d'un écart de 1_°F.

Par exemple, un écart de 100_°C (de 0_°C à 100_°C) est équivalent à un écart de 180_°F.

Pour convertir une valeur de température particulière au lieu d'un écart, utilisez la fonction **tmpCnv()**.

Δ tmpCnv(100·_°C,_°F)	180·_°F
Δ tmpCnv(180·_°F,_°C)	100·_°C
Δ tmpCnv(100·_°C,_°K)	100·_°K
Δ tmpCnv(100·_°F,_°R)	100·_°R
Δ tmpCnv(1·_°C,_°F)	1.8·_°F

Remarque : vous pouvez utiliser le Catalogue pour sélectionner des unités de température.

tPdf()

Catalogue > 

tPdf(ValX,df)⇒*nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

Calcule la densité de probabilité (pdf) de la loi de Student-*t* à *df* degrés de liberté en *ValX*.

trace()

Catalogue > 

trace(matriceCarrée)⇒*expression*

Donne la trace (somme de tous les éléments de la diagonale principale) de *matriceCarrée*.

$\text{trace}\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)$	15
$\text{trace}\left(\begin{bmatrix} a & 0 \\ 1 & a \end{bmatrix}\right)$	2· <i>a</i>

```
Try
  bloc1
Else
  bloc2
EndTry
```

Exécute *bloc1*, à moins qu'une erreur ne se produise. L'exécution du programme est transférée au *bloc2* si une erreur se produit au *bloc1*. La variable système *errCode* contient le numéro d'erreur pour permettre au programme de procéder à une reprise sur erreur. Pour obtenir la liste des codes d'erreur, voir la section « Codes et messages d'erreur », page 291.

bloc1 et *bloc2* peuvent correspondre à une instruction unique ou à une série d'instructions séparées par le caractère “.”.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Pour voir fonctionner les commandes **Try**, **ClrErr** et **PassErr**, saisissez le programme `eigenvals()` décrit à droite. Exécutez le programme en exécutant chacune des expressions suivantes.

$$\text{eigenvals}\left(\begin{bmatrix} -3 \\ -41 \\ 5 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix}\right)$$

$$\text{eigenvals}\left(\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$$

Remarque : voir aussi **ClrErr**, page 28 et **PassErr**, page 148.

```
Define progI()=Prgm
  Try
    z:=z+1
    Disp "z incremented."
  Else
    Disp "Sorry, z undefined."
  EndTry
EndPrgm
```

Done

```
z:=1:progI()
z incremented.
```

Done

```
DelVar z:progI()
Sorry, z undefined.
```

Done

Définition du programme `eigenvals(a,b)=Prgm`

© Le programme `eigenvals(A,B)` présente les valeurs propres A·B

Try

Disp "A= ",a

Disp "B= ",b

Disp " "

Disp "Eigenvalues of A·B are:",eigVl(a*b)

Else

If errCode=230 Then

Disp "Error: Product of A·B must be a square matrix"

ClrErr

Else

PassErr

EndIf

EndTry

EndPrgm

tTest**tTest** μ_0 ,*Liste*[,*Fréq*[,*Hypoth*]]

(Entrée de liste de données)

tTest μ_0 , $\bar{x}$,*sx*,*n*,[*Hypoth*]

(Récapitulatif des statistiques fournies en entrée)

Teste une hypothèse pour une moyenne inconnue de population μ quand l'écart-type de population σ est inconnu. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Test de $H_0 : \mu = \mu_0$, en considérant que :

Pour $H_a : \mu < \mu_0$, définissez *Hypoth*<0

Pour $H_a : \mu \neq \mu_0$ (par défaut), définissez *Hypoth*=0

Pour $H_a : \mu > \mu_0$, définissez *Hypoth*>0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.t	$(\bar{x} - \mu_0) / (\text{stdev} / \sqrt{n})$
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degrés de liberté
stat. $\bar{x}$	Moyenne d'échantillon de la série de données dans <i>Liste</i>

Variable de sortie	Description
stat.sx	Écart-type d'échantillon de la série de données
stat.n	Taille de l'échantillon

tTest_2Samp

Catalogue > 

tTest_2Samp *Liste1, Liste2[, Fréq1[, Fréq2
[, Hypoth[, Group]]]]*

(Entrée de liste de données)

tTest_2Samp $\bar{x}1, sx1, n1, \bar{x}2, sx2, n2[, Hypoth
[, Group]]$

(Récapitulatif des statistiques fournies en entrée)

Effectue un test t sur deux échantillons. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Test de $H_0 : \mu_1 = \mu_2$, en considérant que :

Pour $H_a : \mu_1 < \mu_2$, définissez *Hypoth*<0

Pour $H_a : \mu_1 \neq \mu_2$ (par défaut), définissez *Hypoth*=0

Pour $H_a : \mu_1 > \mu_2$, définissez *Hypoth*>0

Group=1 met en commun les variances

Group=0 ne met pas en commun les variances

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.t	Valeur normale type calculée pour la différence des moyennes
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degrés de liberté des statistiques t
stat. $\bar{x}1$, stat. $\bar{x}2$	Moyennes d'échantillon des séquences de données dans <i>Liste 1</i> et <i>Liste 2</i>

Variable de sortie	Description
stat.sx1, stat.sx2	Écarts-types d'échantillon des séries de données dans <i>Liste 1</i> et <i>Liste 2</i>
stat.n1, stat.n2	Taille des échantillons
stat.sp	Écart-type du groupe. Calculé lorsque <i>Group</i> =1.

tvmFV()

Catalogue > 

tvmFV(*N,I,PV,Pmt,[PpY],[CpY],[PmtAt]*)⇒*valeur*

tvmFV(120,5,0,-500,12,12) 77641.1

Fonction financière permettant de calculer la valeur acquise de l'argent.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 218. Voir également **amortTbl()**, page 8.

tvmI()

Catalogue > 

tvmI(*N,PV,Pmt,FV,[PpY],[CpY],[PmtAt]*)⇒*valeur*

tvmI(240,100000,-1000,0,12,12) 10.5241

Fonction financière permettant de calculer le taux d'intérêt annuel.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 218. Voir également **amortTbl()**, page 8.

tvmN()

Catalogue > 

tvmN(*I,PV,Pmt,FV,[PpY],[CpY],[PmtAt]*)⇒*valeur*

tvmN(5,0,-500,77641,12,12) 120.

Fonction financière permettant de calculer le nombre de périodes de versement.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 218. Voir également **amortTbl()**, page 8.

tvmPmt()Catalogue > 

tvmPmt(*N,I,PV,FV,[PpY],[CpY],[PmtAt]*)⇒*valeur*

tvmPmt(60,4,30000,0,12,12) -552.496

Fonction financière permettant de calculer le montant de chaque versement.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 218. Voir également **amortTbl()**, page 8.

tvmPV()Catalogue > 

tvmPV(*N,I,Pmt,FV,[PpY],[CpY],[PmtAt]*)⇒*valeur*

tvmPV(48,4,-500,30000,12,12) -3426.7

Fonction financière permettant de calculer la valeur actuelle.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 218. Voir également **amortTbl()**, page 8.

Argument TVM*	Description	Type de données
N	Nombre de périodes de versement	nombre réel
I	Taux d'intérêt annuel	nombre réel
PV	Valeur actuelle	nombre réel
Pmt	Montant des versements	nombre réel
FV	Valeur acquise	nombre réel
PpY	Versements par an, par défaut=1	Entier> 0
CpY	Nombre de périodes de calcul par an, par défaut=1	Entier> 0
PmtAt	Versement dû à la fin ou au début de chaque période, par défaut=fin	entier (0=fin, 1=début)

* Ces arguments de valeur temporelle de l'argent sont similaires aux noms des variables TVM (comme **tvm.pv** et **tvm.pmt**) utilisées par le solveur finance de l'application *Calculator*. Cependant, les fonctions financières n'enregistrent pas leurs valeurs ou résultats dans les variables TVM.

TwoVar $X, Y, [Fr\acute{e}q] [, Cat\acute{e}gorie, Inclure]$

Calcule des statistiques pour deux variables.
Un r  capitulatif du r  sultat est stock   dans la variable *stat.results*. (Voir page 198.)

Toutes les listes doivent comporter le m  me nombre de lignes,    l'exception de *Inclure*.

X et Y sont des listes de variables ind  pendantes et d  pendantes.

Fr  q est une liste facultative de valeurs qui indiquent la fr  quence. Chaque   l  ment dans *Fr  q* correspond    une fr  quence d'occurrence pour chaque couple X et Y . Par d  faut, cette valeur est   gale    1. Tous les   l  ments doivent   tre des entiers ≥ 0 .

Cat  gorie est une liste de codes de cat  gories pour les couples X et Y correspondants.

Inclure est une liste d'un ou plusieurs codes de cat  gories. Seuls les   l  ments dont le code de cat  gorie figure dans cette liste sont inclus dans le calcul.

Tout   l  ment vide dans les listes X , *Fr  q* ou *Cat  gorie* a un   l  ment vide correspondant dans l'ensemble des listes r  sultantes. Tout   l  ment vide dans les listes $X1$    $X20$ a un   l  ment vide correspondant dans l'ensemble des listes r  sultantes. Pour plus d'informations concernant les   l  ments vides, reportez-vous    la page 281.

Variable de sortie	Description
stat. $\bar{x}$	Moyenne des valeurs x
stat. x	Somme des valeurs x
stat. x^2	Somme des valeurs x^2
stat.sx	��cart-type de l'��chantillon de x
stat. x	��cart-type de la population de x
stat.n	Nombre de points de donn��es

Variable de sortie	Description
stat. $\bar{y}$	Moyenne des valeurs y
stat. y	Somme des valeurs y
stat. y^2	Somme des valeurs y^2
stat.sy	Écart-type de y dans l'échantillon
stat. y	Écart-type de population des valeurs de y
stat. xy	Somme des valeurs $x \cdot y$
stat.r	Coefficient de corrélation
stat.MinX	Minimum des valeurs de x
stat.Q ₁ X	1er quartile de x
stat.MedianX	Médiane de x
stat.Q ₃ X	3ème quartile de x
stat.MaxX	Maximum des valeurs de x
stat.MinY	Minimum des valeurs de y
stat.Q ₁ Y	1er quartile de y
stat.MedY	Médiane de y
stat.Q ₃ Y	3ème quartile de y
stat.MaxY	Maximum des valeurs y
stat. $(x -)^2$	Somme des carrés des écarts par rapport à la moyenne de x
stat. $(y -)^2$	Somme des carrés des écarts par rapport à la moyenne de y

unitV()

Catalogue > 

unitV(*Vecteur1*)⇒*vecteur*

Donne un vecteur unitaire ligne ou colonne, en fonction de la nature de *Vecteur1*.

Vecteur1 doit être une matrice d'une seule ligne ou colonne.

$\text{unitV}\left(\begin{bmatrix} a & b & c \end{bmatrix}\right)$	$\left[\frac{a}{\sqrt{a^2+b^2+c^2}} \quad \frac{b}{\sqrt{a^2+b^2+c^2}} \quad \frac{c}{\sqrt{a^2+b^2+c^2}} \right]$
$\text{unitV}\left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}\right)$	$\left[\frac{\sqrt{6}}{6} \quad \frac{\sqrt{6}}{3} \quad \frac{\sqrt{6}}{6} \right]$
$\text{unitV}\left(\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}\right)$	$\begin{bmatrix} \frac{\sqrt{14}}{14} \\ \frac{14}{\sqrt{14}} \\ 7 \\ 3 \cdot \frac{\sqrt{14}}{14} \\ 14 \end{bmatrix}$

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

unLock

Catalogue > 

unLockVar1 [, *Var2*] [, *Var3*] ...

unLockVar.

Déverrouille les variables ou les groupes de variables spécifiés. Les variables verrouillées ne peuvent être ni modifiées ni supprimées.

Voir **Lock**, page 119 et **getLockInfo()**, page 94.

<i>a</i> :=65	65
Lock <i>a</i>	Done
getLockInfo(<i>a</i>)	1
<i>a</i> :=75	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
Unlock <i>a</i>	Done
<i>a</i> :=75	75
DelVar <i>a</i>	Done

varPop()

Catalogue > 

varPop(*Liste*[, *listeFréq*])⇒*expression*

Donne la variance de population de *Liste*.

Chaque élément de la liste *listeFréq* totalise le nombre d'occurrences de l'élément correspondant de *Liste*.

$\text{varPop}(\{5,10,15,20,25,30\})$	$\frac{875}{12}$
Ans*1.	72.9167

Remarque : *Liste* doit contenir au moins deux éléments.

Si un élément des listes est vide, il est ignoré et l'élément correspondant dans l'autre liste l'est également. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

varSamp()

varSamp(*Liste*[, *listeFréq*])⇒*expression*

Donne la variance d'échantillon de *Liste*.

Chaque élément de la liste *listeFréq* totalise le nombre d'occurrences de l'élément correspondant de *Liste*.

Remarque : *Liste* doit contenir au moins deux éléments.

Si un élément des listes est vide, il est ignoré et l'élément correspondant dans l'autre liste l'est également. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

varSamp(*MatriceI*[, *matriceFréq*])⇒*matrice*

Donne un vecteur ligne contenant la variance d'échantillon de chaque colonne de *MatriceI*.

Chaque élément de *matriceFréq* totalise le nombre d'occurrences de l'élément correspondant de *MatriceI*.

Remarque : *MatriceI* doit contenir au moins deux lignes.

Si un élément des matrices est vide, il est ignoré et l'élément correspondant dans l'autre matrice l'est également. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 281.

varSamp($\{a, b, c\}$)

$$\frac{a^2 - a \cdot (b+c) + b^2 - b \cdot c + c^2}{3}$$

varSamp($\{1, 2, 5, -6, 3, -2\}$)

$\frac{31}{2}$

varSamp($\{1, 3, 5\}, \{4, 6, 2\}$)

$\frac{68}{33}$

varSamp $\left(\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}\right)$ $[4.75 \quad 1.03 \quad 4]$

varSamp $\left(\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix}, \begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}\right)$ $[3.91731 \quad 2.08411]$

Wait

Catalogue > **Wait** *tempsEnSecondes*

Suspend l'exécution pendant une durée de *tempsEnSecondes* secondes.

La commande **Wait** est particulièrement utile dans un programme qui a besoin de quelques secondes pour permettre aux données demandées d'être accessibles.

L'argument *tempsEnSecondes* doit être une expression qui s'évalue en une valeur décimale comprise entre 0 et 100. La commande arrondit cette valeur à 0,1 seconde près.

Pour annuler un **Wait** qui est en cours,

- **Calculatrice**: Maintenez la touche  enfoncée et appuyez plusieurs fois sur .
- **Windows®** : Maintenez la touche **F12** enfoncée et appuyez plusieurs fois sur **Entrée**.
- **Macintosh®** : Maintenez la touche **F5** enfoncée et appuyez plusieurs fois sur **Entrée**.
- **iPad®** : L'application affiche une invite. Vous pouvez continuer à patienter ou annuler.

Remarque : Vous pouvez utiliser la commande **Wait** dans un programme créé par l'utilisateur, mais pas dans une fonction.

Pour définir un délai d'attente de 4 secondes :

Wait 4

Pour définir un délai d'attente d'une 1/2 seconde :

Wait 0.5

Pour définir un délai d'attente de 1,3 seconde à l'aide de la variable *seccompt* :

seccompt:=1.3

Wait seccompt

Cet exemple allume une DEL verte pendant 0,5 seconde puis l'éteint.

Send "SET GREEN 1 ON"

Wait 0.5

Send "SET GREEN 1 OFF"

warnCodes ()Catalogue > 

warnCodes(*Expr1*,
VarÉtat)⇒*expression*

 **warnCodes** $\left(\text{solve}\left(\sin\left(10 \cdot x\right)=\frac{x^2}{x}, x\right), \text{warn}\right)$
 $x=-0.84232$ or $x=0.706817$ or $x=-0.2852$
warn { 10007, 10009 }

Évalue l'expression *Expr1*, donne le résultat et stocke les codes de tous les avertissements générés dans la variable de liste *VarÉtat*. Si aucun avertissement n'est généré, cette fonction affecte une liste vide à *VarÉtat*.

Expr1 peut être toute expression mathématique TI-Nspire™ ou TI-Nspire™ CAS valide. *Expr1* ne peut pas être une commande ou une affectation.

VarÉtat doit être un nom de variable valide.

Pour la liste des codes d'avertissement et les messages associés, voir page 300.

Pour afficher le résultat entier, appuyez sur **▲**, puis utilisez les touches **◀** et **▶** pour déplacer le curseur.

when()

when(*Condition*, *résultSiOui* [, *résultSiNon*][, *résultSiInconnu*])⇒*expression*

Donne *résultSiOui*, *résultSiNon* ou *résultSiInconnu*, suivant que la *Condition* est vraie, fausse ou indéterminée. Donne l'entrée si le nombre d'argument est insuffisant pour spécifier le résultat approprié.

Ne spécifiez pas *résultSiNon* ni *résultSiInconnu* pour obtenir une expression définie uniquement dans la région où *Condition* est vraie.

Utilisez **undef** *résultSiNon* pour définir une expression représentée graphiquement sur un seul intervalle.

when() est utile dans le cadre de la définition de fonctions récursives.

$\text{when}(x < 0, x + 3) x = 5$	undef
-------------------------------------	-------

$\text{when}(n > 0, n \cdot \text{factorial}(n-1), 1) \rightarrow \text{factorial}(n)$	Done
$\text{factorial}(3)$	6
3!	6

While *Condition*
Bloc
EndWhile

Exécute les instructions contenues dans *Bloc* si *Condition* est vraie.

Bloc peut correspondre à une ou plusieurs instructions, séparées par un « : ».

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Define $sum_of_recip(n)$ =Func	
	Local $i,tempsum$
	$1 \rightarrow i$
	$0 \rightarrow tempsum$
	While $i \leq n$
	$tempsum + \frac{1}{i} \rightarrow tempsum$
	$i + 1 \rightarrow i$
	EndWhile
	Return $tempsum$
	EndFunc
	Done
$sum_of_recip(3)$	$\frac{11}{6}$

X

*BooleanExpr1***xor***BooleanExpr2*
renvoie *expression booléenne*

*BooleanList1***xor***BooleanList2* renvoie
liste booléenne

*BooleanMatrix1***xor***BooleanMatrix2*
renvoie *matrice booléenne*

Donne true si *Expr booléenne1* est vraie et si *Expr booléenne2* est fausse, ou vice versa.

Donne false si les deux arguments sont tous les deux vrais ou faux. Donne une expression booléenne simplifiée si l'un des deux arguments ne peut être résolu vrai ou faux.

Remarque : voir or, xxxx.

Entier1 **xor** *Entier2* $\Rightarrow$ *entier*

true xor true	false
5>3 xor 3>5	true

En mode base Hex :	
Important : utilisez le chiffre zéro et pas la lettre O.	
0h7AC36 xor 0h3D5F	0h79169

Compare les représentations binaires de deux entiers, en appliquant un **xor** bit par bit. En interne, les deux entiers sont convertis en nombres binaires 64 bits signés. Lorsque les bits comparés correspondent, le résultat est 1 si dans l'un des deux cas (pas dans les deux) il s'agit d'un bit 1 ; le résultat est 0 si, dans les deux cas, il s'agit d'un bit 0 ou 1. La valeur donnée représente le résultat des bits et elle est affichée selon le mode Base utilisé.

Les entiers de tout type de base sont admis. Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h. Tout entier sans préfixe est considéré comme un nombre en écriture décimale (base 10).

Si vous entrez un nombre dont le codage binaire signé dépasse 64 bits, il est ramené à l'aide d'une congruence dans la plage appropriée. Pour de plus amples informations, voir ►**Base2**, page 19.

Remarque : voir **or**, **xxxx**.

Z

zeros()

zeros(Expr, Var)⇒liste

zeros(Expr, Var=Init)⇒liste

Donne la liste des valeurs réelles possibles de *Var* avec lesquelles *Expr*=0. Pour y parvenir, **zeros()** calcule **exp►list** (**solve(Expr=0, Var), Var**).

Dans certains cas, la nature du résultat de **zeros()** est plus satisfaisante que celle de **solve()**. Toutefois, la nature du résultat de **zeros()** ne permet pas d'exprimer des solutions implicites, des solutions nécessitant des inéquations ou des solutions qui n'impliquent pas *Var*.

En mode base Bin :

0b100101 xor 0b100	0b100001
--------------------	----------

Remarque : une entrée binaire peut comporter jusqu'à 64 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 16 chiffres.

$$\text{zeros}\left(a \cdot x^2 + b \cdot x + c, x\right) = \left\{ \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a}, \frac{-\left(\sqrt{b^2 - 4 \cdot a \cdot c}\right) - b}{2 \cdot a} \right\}$$

$$a \cdot x^2 + b \cdot x + c | x = \text{Ans}[2] \quad 0$$

$$\text{exact}\left(\text{zeros}\left(a \cdot \left(e^x + x\right) \cdot \left(\text{sign}(x) - 1\right), x\right)\right) \quad \{\emptyset\}$$

$$\text{exact}\left(\text{solve}\left(a \cdot \left(e^x + x\right) \cdot \left(\text{sign}(x) - 1\right) = 0, x\right)\right)$$

$$e^x + x = 0 \text{ or } x > 0 \text{ or } a = 0$$

Remarque : voir aussi **cSolve()**, **cZeros()** et **solve()**.

zeros({Expr1, Expr2}, {VarOutil1, VarOutil2 [, ...]}) ⇒ matrice

Donne les zéros réels possibles du système d'expressions algébriques, où chaque *VarOutil* spécifie une inconnue dont vous recherchez la valeur.

Vous pouvez également spécifier une condition initiale pour les variables. Chaque *VarOutil* doit utiliser le format suivant :

variable

– ou –

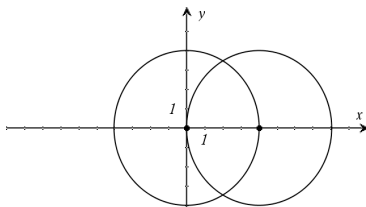
variable = nombre réel ou non réel

Par exemple, x est autorisé, de même que x=3.

Si toutes les expressions sont polynomiales et si vous NE spécifiez PAS de condition initiale, **zeros()** utilise la méthode d'élimination lexicale Gröbner/Buchberger pour tenter de trouver tous les zéros réels.

Par exemple, si vous avez un cercle de rayon r centré à l'origine et un autre cercle de rayon r centré, au point où le premier cercle coupe l'axe des x positifs. Utilisez **zeros()** pour trouver les intersections.

Comme l'illustre r dans l'exemple ci-contre, des expressions polynomiales simultanées peuvent avoir des variables supplémentaires sans valeur assignée, mais représenter des valeurs auxquelles on peut affecter par la suite des valeurs numériques.



$$\text{zeros}\left(\left\{\left\{x^2+y^2-r^2, (x-r)^2+y^2-r^2\right\}, \left\{x,y\right\}\right\}\right)$$

$$\begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} \\ \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} \end{bmatrix}$$

Extraction ligne 2 :

Chaque ligne de la matrice résultante représente un n-uplet, l'ordre des composants étant identique à celui de la liste *VarOuInit*. Pour extraire une ligne, indexez la matrice par [*ligne*].

Vous pouvez également utiliser des inconnues qui n'apparaissent pas dans les expressions. Par exemple, vous pouvez utiliser *z* comme inconnue pour développer l'exemple précédent et avoir deux cylindres parallèles sécants de rayon *r*. La solution des cylindres montre comment des groupes de zéros peuvent contenir des constantes arbitraires de type *ck*, où *k* est un suffixe entier compris entre 1 et 255.

Pour les systèmes d'équations polynomiaux, le temps de calcul et l'utilisation de la mémoire peuvent considérablement varier en fonction de l'ordre dans lequel les inconnues sont spécifiées. Si votre choix initial ne vous satisfait pas pour ces raisons, vous pouvez modifier l'ordre des variables dans les expressions et/ou la liste *VarOuInit*.

Si vous choisissez de ne pas spécifier de condition et s'il l'une des expressions n'est pas polynomiale dans l'une des variables, mais que toutes les expressions sont linéaires par rapport à toutes les inconnues, **zeros()** utilise l'élimination gaussienne pour tenter de trouver tous les zéros réels.

Si un système d'équations n'est pas polynomial dans toutes ses variables ni linéaire par rapport à ses inconnues, **zeros()** cherche au moins un zéro en utilisant une méthode itérative approchée. Pour cela, le nombre d'inconnues doit être égal au nombre d'expressions et toutes les autres variables contenues dans les expressions doivent pouvoir être évaluées à des nombres.

$$\text{Ans}[2] \quad \begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3} \cdot r}{2} \end{bmatrix}$$

$$\text{zeros}\left(\left\{x^2+y^2-r^2, (x-r)^2+y^2-r^2\right\}, \{x, y, z\}\right) \quad \begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3} \cdot r}{2} & c1 \\ \frac{r}{2} & \frac{\sqrt{3} \cdot r}{2} & c1 \end{bmatrix}$$

$$\text{zeros}\left(\left\{x+e^z \cdot y-1, x-y-\sin(z)\right\}, \{x, y\}\right) \quad \begin{bmatrix} \frac{e^z \cdot \sin(z)+1}{e^z+1} & \frac{-\sin(z)-1}{e^z+1} \end{bmatrix}$$

$$\text{zeros}\left(\left\{e^z \cdot y-1, y-\sin(z)\right\}, \{y, z\}\right) \quad \begin{bmatrix} 0.041458 & 3.18306 \\ 0.001871 & 6.28131 \\ 4.76\text{E-}11 & 1796.99 \\ 2.\text{E-}13 & 254.469 \end{bmatrix}$$

Chaque inconnue commence à sa valeur supposée, si elle existe ; sinon, la valeur de départ est 0.0.

Utilisez des valeurs initiales pour rechercher des zéros supplémentaires, un par un. Pour assurer une convergence correcte, une valeur initiale doit être relativement proche d'un zéro.

$$\text{zeros}\left(\left\{\left\{e^z \cdot y - 1, y - \sin(z)\right\}, \left\{y, z = 2 \cdot \pi\right\}\right\}\right)$$

$$\left[0.001871 \quad 6.28131\right]$$

zInterval

zInterval $\sigma, \text{Liste}, \text{Fréq}, \text{CLevel}]$

(Entrée de liste de données)

zInterval $\sigma, \bar{x}, n [, \text{CLevel}]$

(Récapitulatif des statistiques fournies en entrée)

Calcule un intervalle de confiance z . Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance pour une moyenne inconnue de population
stat. $\bar{x}$	Moyenne d'échantillon de la série de données suivant la loi normale aléatoire
stat.ME	Marge d'erreur
stat.sx	Écart-type d'échantillon
stat.n	Taille de la série de données avec la moyenne d'échantillon
stat. σ	Écart-type connu de population pour la série de données <i>Liste</i>

zInterval_1Prop

zInterval_1Prop $x, n [, \text{CLevel}]$

Calcule un intervalle de confiance z pour une proportion. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

x est un entier non négatif.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à “Éléments vides”, page 281.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance contenant la probabilité du niveau de confiance de la loi
stat. $\hat{p}$	Proportion calculée de réussite
stat.ME	Marge d'erreur
stat.n	Nombre d'échantillons dans la série de données

zInterval_2Prop $x1, n1, x2, n2[, CLevel]$

Calcule un intervalle de confiance z pour deux proportions. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

$x1$ et $x2$ sont des entiers non négatifs.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à “Éléments vides”, page 281.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance contenant la probabilité du niveau de confiance de la loi
stat. $\hat{p}$ Diff	Différence calculée entre les proportions
stat.ME	Marge d'erreur
stat. $\hat{p}1$	Proportion calculée sur le premier échantillon
stat. $\hat{p}2$	Proportion calculée sur le deuxième échantillon

Variable de sortie	Description
stat.n1	Taille de l'échantillon dans la première série de données
stat.n2	Taille de l'échantillon dans la deuxième série de données

zInterval_2Samp

Catalogue > 

zInterval_2Samp $\sigma_1, \sigma_2, \text{Liste1}, \text{Liste2}$
 $[\text{Fréq1}, [\text{Fréq2}, [\text{CLevel}]]]$

(Entrée de liste de données)

zInterval_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2$
 $[\text{CLevel}]$

(Récapitulatif des statistiques fournies en entrée)

Calcule un intervalle de confiance z sur deux échantillons. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance contenant la probabilité du niveau de confiance de la loi
stat. $\bar{x}1 - \bar{x}2$	Moyennes d'échantillon des séries de données suivant la loi normale aléatoire
stat.ME	Marge d'erreur
stat. $\bar{x}1$, stat. $\bar{x}2$	Moyennes d'échantillon des séries de données suivant la loi normale aléatoire
stat. $\sigma x1$, stat. $\sigma x2$	Écarts-types d'échantillon pour <i>Liste 1</i> et <i>Liste 2</i>
stat.n1, stat.n2	Nombre d'échantillons dans les séries de données
stat.r1, stat.r2	Écart-type connu de population pour la série de données <i>Liste 1</i> et <i>Liste 2</i>

zTest

Catalogue > 

zTest $\mu0, \sigma, \text{Liste}, [\text{Fréq}, \text{Hypoth}]$

(Entrée de liste de données)

zTest $\mu_0, \sigma, \bar{x}, n[, Hypoth]$

(Récapitulatif des statistiques fournies en entrée)

Effectue un test z en utilisant la fréquence *listeFréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Test de $H_0 : \mu = \mu_0$, en considérant que :

Pour $H_a : \mu < \mu_0$, définissez *Hypoth*<0

Pour $H_a : \mu \neq \mu_0$ (par défaut), définissez *Hypoth*=0

Pour $H_a : \mu > \mu_0$, définissez *Hypoth*>0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.z	$(\bar{x} - \mu_0) / (\sigma / \sqrt{n})$
stat.P Value	Plus petite probabilité permettant de rejeter l'hypothèse nulle
stat. $\bar{x}$	Moyenne d'échantillon de la série de données dans <i>Liste</i>
stat.sx	Écart-type d'échantillon de la série de données Uniquement donné pour l'entrée <i>Data</i> .
stat.n	Taille de l'échantillon

zTest_1Prop

zTest_1Prop $p_0, x, n[, Hypoth]$

Effectue un test z pour une proportion unique. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

x est un entier non négatif.

Test de $H_0 : p = p_0$, en considérant que :

Pour $H_a : p > p_0$, définissez *Hypoth*>0

Pour $H_a : p \neq p_0$ (par défaut), définissez *Hypoth*=0

Pour $H_a : p < p_0$, définissez *Hypoth*<0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.p0	Proportion de population hypothétique
stat.z	Valeur normale type calculée pour la proportion
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat. $\hat{p}$	Proportion calculée sur l'échantillon
stat.n	Taille de l'échantillon

zTest_2Prop *x1, n1, x2, n2, [Hypoth]*

Calcule un test *z* pour deux proportions. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

x1 et *x2* sont des entiers non négatifs.

Test de $H_0 : p1 = p2$, en considérant que :

Pour $H_a : p1 > p2$, définissez *Hypoth*>0

Pour $H_a : p1 \neq p2$ (par défaut), définissez *Hypoth*=0

Pour $H_a : p < p_0$, définissez *Hypoth*<0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.z	Valeur normale type calculée pour la différence des proportions

Variable de sortie	Description
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat. $\hat{p}1$	Proportion calculée sur le premier échantillon
stat. $\hat{p}2$	Proportion calculée sur le deuxième échantillon
stat. $\hat{p}$	Proportion calculée de l'échantillon mis en commun
stat.n1, stat.n2	Nombre d'échantillons pris lors des essais 1 et 2

zTest_2Samp

Catalogue > 

zTest_2Samp $\sigma_1, \sigma_2, Liste1, Liste2[, Fréq1[, Fréq2[, Hypoth]]]$

(Entrée de liste de données)

zTest_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2[, Hypoth]$

(Récapitulatif des statistiques fournies en entrée)

Calcule un test z sur deux échantillons. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 198.)

Test de $H_0 : \mu_1 = \mu_2$, en considérant que :

Pour $H_a : \mu_1 < \mu_2$, définissez *Hypoth*<0

Pour $H_a : \mu_1 \neq \mu_2$ (par défaut), définissez *Hypoth*=0

Pour $H_a : \mu_1 > \mu_2$, *Hypoth*>0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 281.

Variable de sortie	Description
stat.z	Valeur normale type calculée pour la différence des moyennes
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat. $\bar{x}1$, stat. $\bar{x}2$	Moyennes d'échantillon des séquences de données dans <i>Liste 1</i> et <i>Liste 2</i>
stat.sx1, stat.sx2	Écarts-types d'échantillon des séries de données dans <i>Liste 1</i> et <i>Liste 2</i>
stat.n1, stat.n2	Taille des échantillons

Symboles

+ (somme)		Touche 
$Expr1 + Expr2 \Rightarrow expression$	56	56
Donne la somme des deux arguments.	56+4	60
	60+4	64
	64+4	68
	68+4	72
$Liste1 + Liste2 \Rightarrow liste$	$\left\{ 22, \pi, \frac{\pi}{2} \right\} \rightarrow I1$	$\left\{ 22, \pi, \frac{\pi}{2} \right\}$
$Matrice1 + Matrice2 \Rightarrow matrice$	$\left\{ 10, 5, \frac{\pi}{2} \right\} \rightarrow I2$	$\left\{ 10, 5, \frac{\pi}{2} \right\}$
Donne la liste (ou la matrice) contenant les sommes des éléments correspondants de <i>Liste1</i> et <i>Liste2</i> (ou <i>Matrice1</i> et <i>Matrice2</i>).	$I1+I2$	$\{ 32, \pi+5, \pi \}$
	$Ans+\{ \pi, -5, \pi \}$	$\{ \pi+32, \pi, 0 \}$
Les arguments doivent être de même dimension.	$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} a+1 & b \\ c & d+1 \end{bmatrix}$
$Expr + Liste1 \Rightarrow liste$	$15+\{ 10, 15, 20 \}$	$\{ 25, 30, 35 \}$
$Liste1 + Expr \Rightarrow liste$	$\{ 10, 15, 20 \}+15$	$\{ 25, 30, 35 \}$
Donne la liste contenant les sommes de <i>Expr</i> et de chaque élément de <i>Liste1</i> .		
$Expr + Matrice1 \Rightarrow matrice$	$20+\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	$\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$
$Matrice1 + Expr \Rightarrow matrice$		
Donne la matrice obtenue en ajoutant <i>Expr</i> à chaque élément de la diagonale de <i>Matrice1</i> . <i>Matrice1</i> doit être carrée.		
Remarque : utilisez .+ pour ajouter une expression à chaque élément de la matrice.		

- (soustraction)		Touche 
$Expr1 - Expr2 \Rightarrow expression$	6-2	4
Donne la différence de <i>Expr1</i> et de <i>Expr2</i> .	$\pi - \frac{\pi}{6}$	$\frac{5 \cdot \pi}{6}$

-(soustraction)

Touche $\boxed{-}$

Liste1 - *Liste2* $\Rightarrow$ *liste*

$$\left\{22, \pi, \frac{\pi}{2}\right\} - \left\{10, 5, \frac{\pi}{2}\right\} = \left\{12, \pi - 5, 0\right\}$$

Matrice1 - *Matrice2* $\Rightarrow$ *matrice*

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

Soustrait chaque élément de *Liste2* (ou *Matrice2*) de l'élément correspondant de *Liste1* (ou *Matrice1*) et donne le résultat obtenu.

Les arguments doivent être de même dimension.

Expr - *Liste1* $\Rightarrow$ *liste*

$$15 - \{10, 15, 20\} = \{5, 0, -5\}$$

Liste1 - *Expr* $\Rightarrow$ *liste*

$$\{10, 15, 20\} - 15 = \{-5, 0, 5\}$$

Soustrait chaque élément de *Liste1* de *Expr* ou soustrait *Expr* de chaque élément de *Liste1* et donne la liste de résultats obtenue.

Expr - *Matrice1* $\Rightarrow$ *matrice*

$$20 - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$$

Matrice1 - *Expr* $\Rightarrow$ *matrice*

Expr - *Matrice1* donne la matrice *Expr* fois la matrice d'identité moins *Matrice1*. *Matrice1* doit être carrée.

Matrice1 - *Expr* donne la matrice obtenue en soustrayant *Expr* à chaque élément de la diagonale de *Matrice1*. *Matrice1* doit être carrée.

Remarque : Utilisez $\cdot-$ pour soustraire une expression à chaque élément de la matrice.

·(multiplication)

Touche $\boxed{\times}$

Expr1 · *Expr2* $\Rightarrow$ *expression*

$$2 \cdot 3.45 = 6.9$$

Donne le produit des deux arguments.

$$x \cdot y \cdot x = x^2 \cdot y$$

Liste1 · *Liste2* $\Rightarrow$ *liste*

$$\{1., 2, 3\} \cdot \{4, 5, 6\} = \{4., 10, 18\}$$

Donne la liste contenant les produits des éléments correspondants de *Liste1* et *Liste2*.

$$\left\{\frac{2}{a}, \frac{3}{2}\right\} \cdot \left\{a^2, \frac{b}{3}\right\} = \left\{2 \cdot a, \frac{b}{2}\right\}$$

Les listes doivent être de même dimension.

·(multiplication)

Touche ×

Matrice1 · *Matrice2* ⇒ *matrice*

Donne le produit des matrices *Matrice1* et *Matrice2*.

Le nombre de colonnes de *Matrice1* doit être égal au nombre de lignes de *Matrice2*.

Expr · *Liste1* ⇒ *liste*

Liste1 · *Expr* ⇒ *liste*

Donne la liste des produits de *Expr* et de chaque élément de *Liste1*.

Expr · *Matrice1* ⇒ *matrice*

Matrice1 · *Expr* ⇒ *matrice*

Donne la matrice contenant les produits de *Expr* et de chaque élément de *Matrice1*.

Remarque : Utilisez · pour multiplier une expression par chaque élément.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix} = \begin{bmatrix} a+2 \cdot b+3 \cdot c & d+2 \cdot e+3 \cdot f \\ 4 \cdot a+5 \cdot b+6 \cdot c & 4 \cdot d+5 \cdot e+6 \cdot f \end{bmatrix}$$

$$\pi \cdot \{4, 5, 6\} = \{4 \cdot \pi, 5 \cdot \pi, 6 \cdot \pi\}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01 = \begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \end{bmatrix}$$

$$\lambda \cdot \text{identity}(3) = \begin{bmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix}$$

/ (division)

Touche ÷

Expr1 / *Expr2* ⇒ *expression*

Donne le quotient de *Expr1* par *Expr2*.

Remarque : voir aussi **Modèle Fraction**, page 1.

Liste1 / *Liste2* ⇒ *liste*

Donne la liste contenant les quotients de *Liste1* par *Liste2*.

Les listes doivent être de même dimension.

Expr / *Liste1* ⇒ *liste*

Liste1 / *Expr* ⇒ *liste*

Donne la liste contenant les quotients de *Expr* par *Liste1* ou de *Liste1* par *Expr*.

Matrice1 / *Expr* ⇒ *matrice*

$$\frac{2}{3.45} = 0.57971$$

$$\frac{x^3}{x} = x^2$$

$$\frac{\{1, 2, 3\}}{\{4, 5, 6\}} = \left\{0.25, \frac{2}{5}, \frac{1}{2}\right\}$$

$$\frac{a}{\{3, a, \sqrt{a}\}} = \left\{\frac{a}{3}, 1, \sqrt{a}\right\}$$

$$\frac{\{a, b, c\}}{a \cdot b \cdot c} = \left\{\frac{1}{b \cdot c}, \frac{1}{a \cdot c}, \frac{1}{a \cdot b}\right\}$$

$$\frac{\begin{bmatrix} a & b & c \end{bmatrix}}{a \cdot b \cdot c} = \begin{bmatrix} \frac{1}{b \cdot c} & \frac{1}{a \cdot c} & \frac{1}{a \cdot b} \end{bmatrix}$$

Donne la matrice contenant les quotients des éléments de *Matrice1* / *Expression*.

Remarque : Utilisez $\div$ pour diviser une expression par chaque élément.

^ (puissance)

Touche $\wedge$

Expr1 $\wedge$ *Expr2* $\Rightarrow$ *expression*

$$4^2 \qquad 16$$

Liste1 $\wedge$ *Liste2* $\Rightarrow$ *liste*

$$\{a, 2, c\}^{\{1, b, 3\}} \qquad \{a, 2^b, c^3\}$$

Donne le premier argument élevé à la puissance du deuxième argument.

Remarque : voir aussi **Modèle Exposant**, page 1.

Dans le cas d'une liste, donne la liste des éléments de *Liste1* élevés à la puissance des éléments correspondants de *Liste2*.

Dans le domaine réel, les puissances fractionnaires possédant des exposants réduits avec des dénominateurs impairs utilise la branche réelle, tandis que le mode complexe utilise la branche principale.

Expr $\wedge$ *Liste1* $\Rightarrow$ *liste*

$$p^{\{a, 2, -3\}} \qquad \left\{ p^a, p^2, \frac{1}{p^3} \right\}$$

Donne *Expr* élevé à la puissance des éléments de *Liste1*.

List1 $\wedge$ *Expr* $\Rightarrow$ *liste*

$$\{1, 2, 3, 4\}^{-2} \qquad \left\{ 1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16} \right\}$$

Donne les éléments de *Liste1* élevés à la puissance de l'*expression*.

matriceCarrée1 $\wedge$ *entier* $\Rightarrow$ *matrice*

Donne *matriceCarrée1* élevée à la puissance de la valeur de l'*entier*.

matriceCarrée1 doit être une matrice carrée.

Si *entier* = -1, calcule la matrice inverse.

Si *entier* < -1, calcule la matrice inverse à une puissance positive appropriée.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2 \qquad \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

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

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2} \qquad \begin{bmatrix} 11 & -5 \\ 2 & 2 \\ -15 & 7 \\ 4 & 4 \end{bmatrix}$$

x² (carré)Touche $\boxed{x^2}$ **Expr1² ⇒ expression**

Donne le carré de l'argument.

$$4^2 \qquad 16$$

$$\{2,4,6\}^2 \qquad \{4,16,36\}$$

Liste1² ⇒ listeDonne la liste comportant les carrés des éléments de *Liste1*.

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2 \qquad \begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$$

matriceCarrée1² ⇒ matrice

Donne le carré de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du carré de chaque élément. Utilisez $\cdot^2$ pour calculer le carré de chaque élément.

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix} \cdot^2 \qquad \begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$$

.+ (addition élément par élément)Touches $\boxed{.} \boxed{+}$ **Matrice1 .+ Matrice2 ⇒ matrice****Expr .+ Matrice1 ⇒ matrice**

Matrice1 .+ *Matrice2* donne la matrice obtenue en effectuant la somme de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

Expr .+ *Matrice1* donne la matrice obtenue en effectuant la somme de *Expr* et de chaque élément de *Matrice1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} a+c & 6 \\ b+5 & d+3 \end{bmatrix}$$

$$x .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} x+c & x+4 \\ x+5 & x+d \end{bmatrix}$$

.- (soustraction élément par élément)Touches $\boxed{.} \boxed{-}$ **Matrice1 .- Matrice2 ⇒ matrice****Expr .- Matrice1 ⇒ matrice**

Matrice1 .- *Matrice2* donne la matrice obtenue en calculant la différence entre chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \qquad \begin{bmatrix} a-c & -2 \\ b-d & -2 \end{bmatrix}$$

$$x .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix} \qquad \begin{bmatrix} x-c & x-4 \\ x-d & x-5 \end{bmatrix}$$

.- (soustraction élément par élément)

Touches $\boxed{.}$ $\boxed{-}$

Expr .- *Matrice1* donne la matrice obtenue en calculant la différence de *Expr* et de chaque élément de *Matrice1*.

. (multiplication élément par élément)

Touches $\boxed{.}$ $\boxed{\times}$

Matrice1 . *Matrice2* $\Rightarrow$ matrice

Expr . *Matrice1* $\Rightarrow$ matrice

Matrice1 . *Matrice2* donne la matrice obtenue en calculant le produit de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

Expr . *Matrice1* donne la matrice contenant les produits de *Expr* et de chaque élément de *Matrice1*.

$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix}$	$\cdot$	$\begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\Rightarrow$	$\begin{bmatrix} a \cdot c & 8 \\ 5 \cdot b & 3 \cdot d \end{bmatrix}$
$x \cdot$	$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$		$\Rightarrow$	$\begin{bmatrix} a \cdot x & b \cdot x \\ c \cdot x & d \cdot x \end{bmatrix}$

. / (division élément par élément)

Touches $\boxed{.}$ $\boxed{\div}$

Matrice1 . / *Matrice2* $\Rightarrow$ matrice

Expr . / *Matrice1* $\Rightarrow$ matrice

Matrice1 . / *Matrice2* donne la matrice obtenue en calculant le quotient de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

Expr . / *Matrice1* donne la matrice obtenue en calculant le quotient de *Expr* et de chaque élément de *Matrice1*.

$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix}$	$\cdot /$	$\begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\Rightarrow$	$\begin{bmatrix} \frac{a}{c} & \frac{1}{2} \\ \frac{b}{5} & \frac{3}{d} \end{bmatrix}$
$x \cdot /$	$\begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$		$\Rightarrow$	$\begin{bmatrix} \frac{x}{c} & \frac{x}{4} \\ \frac{x}{5} & \frac{x}{d} \end{bmatrix}$

.^ (puissance élément par élément)

Touches $\boxed{.}$ $\boxed{\wedge}$

Matrice1 .^ *Matrice2* $\Rightarrow$ matrice

Expr .^ *Matrice1* $\Rightarrow$ matrice

Matrice1 .^ *Matrice2* donne la matrice obtenue en élevant chaque élément de *Matrice1* à la puissance de l'élément correspondant de *Matrice2*.

$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix}$	$\cdot ^$	$\begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$	$\Rightarrow$	$\begin{bmatrix} a^c & 16 \\ b^5 & 3^d \end{bmatrix}$
$x \cdot ^$	$\begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$		$\Rightarrow$	$\begin{bmatrix} x^c & x^4 \\ x^5 & x^d \end{bmatrix}$

.^ (puissance élément par élément)

Touches  

$Expr.^{Matrice1}$ donne la matrice obtenue en élevant $Expr$ à la puissance de chaque élément de $Matrice1$.

-(opposé)

Touche 

$-Expr1 \Rightarrow expression$

$-Liste1 \Rightarrow liste$

$-Matrice1 \Rightarrow matrice$

Donne l'opposé de l'argument.

Dans le cas d'une liste ou d'une matrice, donne l'opposé de chacun des éléments.

Si l'argument est un entier binaire ou hexadécimal, la négation donne le complément à deux.

-2.43	-2.43
$-\{-1,0.4,1.2E19\}$	$\{1,-0.4,-1.2E19\}$
$-a \cdot b$	$a \cdot b$

En mode base Bin :

Important : utilisez le chiffre zéro et pas la lettre O.

-0b100101
0b11111111111111111111111111111111▶

Pour afficher le résultat entier, appuyez sur , puis utilisez les touches  et  pour déplacer le curseur.

% (pourcentage)

Touches  

$Expr1 \% \Rightarrow expression$

$Liste1 \% \Rightarrow liste$

$Matrice1 \% \Rightarrow matrice$

argument

Donne 100

Dans le cas d'une liste ou d'une matrice, donne la liste ou la matrice obtenue en divisant chaque élément par 100.

Remarque : Pour afficher un résultat approximatif,

Unité : Appuyez sur  .

Windows® : Appuyez sur **Ctrl+Entrée**.

Macintosh® : Appuyez sur **⌘+Entrée**.

iPad® : Maintenez la touche **Entrée** enfoncée et sélectionnez .

13%	0.13
$\{\{1,10,100\}\}\%$	$\{0.01,0.1,1.\}$

= (égal à)

Touche 

$Expr1 = Expr2 \Rightarrow Expression booléenne$

Exemple de fonction qui utilise les symboles de test mathématiques : =, ≠, <, ≤, >, ≥

= (égal à)Touche  $Liste1 = Liste2 \Rightarrow Liste \text{ booléenne}$ $Matrice1 = Matrice2 \Rightarrow Matrice \text{ booléenne}$

Donne true s'il est possible de vérifier que la valeur de *Expr1* est égale à celle de *Expr2*.

Donne false s'il est possible de déterminer que la valeur de *Expr1* n'est pas égale à celle de *Expr2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Define $g(x) = \text{Func}$ If $x \leq -5$ Then

Return 5

ElseIf $x > -5$ and $x < 0$ ThenReturn $-x$ ElseIf $x \geq 0$ and $x \neq 10$ ThenReturn x ElseIf $x = 10$ Then

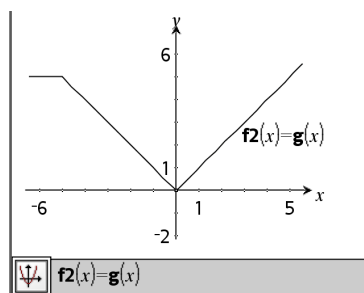
Return 3

EndIf

EndFunc

Done

Résultat de la représentation graphique de $g(x)$

**≠ (différent de)**Touches   $Expr1 \neq Expr2 \Rightarrow Expression \text{ booléenne}$

Voir l'exemple fourni pour « = » (égal à).

 $Liste1 \neq Liste2 \Rightarrow Liste \text{ booléenne}$ $Matrice1 \neq Matrice2 \Rightarrow Matrice \text{ booléenne}$

Donne true s'il est possible de déterminer que la valeur de *Expr1* n'est pas égale à celle de *Expr2*.

Donne false s'il est possible de vérifier que la valeur de *Expr1* est égale à celle de *Expr2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

≠ (différent de)

Touches  

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant /=

< (inférieur à)

Touches  

$Expr1 < Expr2 \Rightarrow Expression \text{ booléenne}$

Voir l'exemple fourni pour « = » (égal à).

$Liste1 < Liste2 \Rightarrow Liste \text{ booléenne}$

$Matrice1 < Matrice2 \Rightarrow Matrice \text{ booléenne}$

Donne true s'il est possible de déterminer que la valeur de *Expr1* est strictement inférieure à celle de *Expr2*.

Donne false s'il est possible de déterminer que la valeur de *Expr1* est strictement supérieure ou égale à celle de *Expr2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

≤ (inférieur ou égal à)

Touches  

$Expr1 \leq Expr2 \Rightarrow Expression \text{ booléenne}$

Voir l'exemple fourni pour « = » (égal à).

$Liste1 \leq Liste2 \Rightarrow Liste \text{ booléenne}$

$Matrice1 \leq Matrice2 \Rightarrow Matrice \text{ booléenne}$

Donne true s'il est possible de déterminer que la valeur de *Expr1* est inférieure ou égale à celle de *Expr2*.

Donne false s'il est possible de déterminer que la valeur de *Expr1* est strictement supérieure à celle de *Expr2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

$\leq$ (inférieur ou égal à)

Touches  

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant $\leq$

$>$ (supérieur à)

Touches  

$Expr1 > Expr2 \Rightarrow Expression \text{ booléenne}$

Voir l'exemple fourni pour « $=$ » (égal à).

$Liste1 > Liste2 \Rightarrow Liste \text{ booléenne}$

$Matrice1 > Matrice2 \Rightarrow Matrice \text{ booléenne}$

Donne true s'il est possible de déterminer que la valeur de $Expr1$ est supérieure à celle de $Expr2$.

Donne false s'il est possible de déterminer que la valeur de $Expr1$ est strictement inférieure ou égale à celle de $Expr2$.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

$\geq$ (supérieur ou égal à)

Touches  

$Expr1 \geq Expr2 \Rightarrow Expression \text{ booléenne}$

Voir l'exemple fourni pour « $=$ » (égal à).

$Liste1 \geq Liste2 \Rightarrow Liste \text{ booléenne}$

$Matrice1 \geq Matrice2 \Rightarrow Matrice \text{ booléenne}$

Donne true s'il est possible de déterminer que la valeur de $Expr1$ est supérieure ou égale à celle de $Expr2$.

Donne false s'il est possible de déterminer que la valeur de $Expr1$ est inférieure ou égale à celle de $Expr2$.

Dans les autres cas, donne une forme simplifiée de l'équation.

$\geq$ (supérieur ou égal à)

Touches **ctrl** **=**

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant $\geq$

$\Rightarrow$ (implication logique)

touches **ctrl** **=**

BooleanExpr1 $\Rightarrow$ BooleanExpr2
renvoie *expression booléenne*

$5 > 3$ or $3 > 5$	true
--------------------	------

$5 > 3 \Rightarrow 3 > 5$	false
---------------------------	-------

BooleanList1 $\Rightarrow$ BooleanList2 renvoie
liste booléenne

3 or 4	7
------------	---

$3 \Rightarrow 4$	-4
-------------------	----

BooleanMatrix1 $\Rightarrow$ BooleanMatrix2
renvoie *matrice booléenne*

$\{1, 2, 3\}$ or $\{3, 2, 1\}$	$\{3, 2, 3\}$
--------------------------------	---------------

$\{1, 2, 3\} \Rightarrow \{3, 2, 1\}$	$\{-1, -1, -3\}$
---------------------------------------	------------------

Integer1 $\Rightarrow$ Integer2 renvoie *entier*

Évalue l'expression **not** <argument1> or <argument2> et renvoie true (vrai) ou false (faux) ou une forme simplifiée de l'équation.

Pour les listes et matrices, renvoie le résultat des comparaisons, élément par élément.

Remarque : Vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant $\Rightarrow$

$\Leftrightarrow$ (équivalence logique, XNOR)

touches **ctrl** **=**

BooleanExpr1 $\Leftrightarrow$ BooleanExpr2
renvoie *expression booléenne*

$5 > 3$ xor $3 > 5$	true
---------------------	------

$5 > 3 \Leftrightarrow 3 > 5$	false
-------------------------------	-------

BooleanList1 $\Leftrightarrow$ BooleanList2 renvoie
liste booléenne

3 xor 4	7
-------------	---

$3 \Leftrightarrow 4$	-8
-----------------------	----

BooleanMatrix1 $\Leftrightarrow$ BooleanMatrix2
renvoie *matrice booléenne*

$\{1, 2, 3\}$ xor $\{3, 2, 1\}$	$\{2, 0, 2\}$
---------------------------------	---------------

$\{1, 2, 3\} \Leftrightarrow \{3, 2, 1\}$	$\{-3, -1, -3\}$
---	------------------

Integer1 $\Leftrightarrow$ Integer2 renvoie *entier*

Renvoie la négation d'une opération booléenne **XOR** sur les deux arguments. Renvoie true (vrai) ou false (faux) ou une forme simplifiée de l'équation.

Pour les listes et matrices, renvoie le résultat des comparaisons, élément par élément.

Remarque : Vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant <=>

! (factorielle)

Touche 

Expr1! ⇒ expression

5! 120

Liste1! ⇒ liste

{ { 5,4,3 } }! { 120,24,6 }

Matrice1! ⇒ matrice

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}!$ $\begin{pmatrix} 1 & 2 \\ 6 & 24 \end{pmatrix}$

Donne la factorielle de l'argument.

Dans le cas d'une liste ou d'une matrice, donne la liste ou la matrice des factorielles de tous les éléments.

& (ajouter)

Touches  

Chaîne1 & Chaîne2 ⇒ chaîne

"Hello "&"Nick" "Hello Nick"

Donne une chaîne de caractères obtenue en ajoutant *Chaîne2* à *Chaîne1*.

d() (dérivée)

Catalogue > 

d(Expr1, Var[, Ordre])⇒expression

$\frac{d}{dx}(f(x) \cdot g(x))$ $\frac{d}{dx}(f(x)) \cdot g(x) + \frac{d}{dx}(g(x)) \cdot f(x)$

d(Liste1, Var[, Ordre])⇒liste

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

d(Matrice1, Var[, Ordre])⇒matrice

$\frac{d}{dx} \left(\begin{pmatrix} x^2 & x^3 & x^4 \end{pmatrix} \right)$ $\begin{pmatrix} 2 \cdot x & 3 \cdot x^2 & 4 \cdot x^3 \end{pmatrix}$

Affiche la dérivée première du premier argument par rapport à la variable *Var*.

Ordre, si spécifié, doit être un entier. Si l'ordre spécifié est inférieur à zéro, on obtient une primitive.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **derivative (...)**.

d() n'applique pas la méthode de calcul standard qui consiste à simplifier entièrement ses arguments, puis à appliquer la définition de la fonction aux arguments simplifiés obtenus. Par contre, **d()** procède de la façon suivante :

1. Il simplifie le deuxième argument uniquement dans la mesure où cette opération permet d'obtenir une variable.
2. Il simplifie le premier argument uniquement dans la mesure où cette opération appelle une valeur stockée pour la variable déterminée à l'étape 1.
3. Il détermine la dérivée symbolique du résultat obtenu à l'étape 2 par rapport à la variable générée à l'étape 1.

Si la variable déterminée à l'étape 1 a une valeur stockée ou une valeur spécifiée par l'opérateur "sachant que" (« | »), cette valeur est substituée dans le résultat obtenu à l'étape 3.

Remarque : voir aussi **Dérivée première**, page 5, **Dérivée seconde**, page 6 ou **Dérivée n-ième**, page 6.

∫() (intégrale)

∫(Expr1, Var[, Borne1, Borne2]) ⇒
expression

∫(Expr1, Var[, Constante]) ⇒
expression

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

Affiche l'intégrale de *Expr1* pour la variable *Var* entre *Borne1* et *Borne2*.

Remarque : voir aussi le modèle **Intégrale définie** ou **indéfinie**, page 6.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **integral (...)**.

Donne une primitive si *Borne1* et *Borne2* sont omises. La constante d'intégration est omise si vous spécifiez l'argument *Constante*.

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

$$\int (a \cdot x^2, x, c) = \frac{a \cdot x^3}{3} + c$$

Les primitives valides peuvent différer d'une constante numérique. Ce type de constante peut être masqué, notamment lorsqu'une primitive contient des logarithmes ou des fonctions trigonométriques inverses. De plus, des expressions constantes par morceaux sont parfois ajoutées pour assurer la validité d'une primitive sur un intervalle plus grand que celui d'une formule courante.

f() retourne les intégrales non évaluées des morceaux de *Expr1* dont les primitives ne peuvent pas être déterminées sous forme de combinaison explicite finie de fonctions usuelles.

$$\int b \cdot e^{-x^2} + \frac{a}{x^2 + a^2} dx \quad b \cdot \int e^{-x^2} dx + \tan^{-1}\left(\frac{x}{a}\right)$$

Si *Borne1* et *Borne2* sont toutes les deux spécifiées, la fonction tente de localiser toute discontinuité ou dérivée discontinue comprise dans l'intervalle $Borne1 < Var < Borne2$ et de subdiviser l'intervalle en ces points.

Avec le réglage Auto du mode **Auto ou Approché (Approximate)**, l'intégration numérique est utilisée, si elle est applicable, chaque fois qu'une primitive ou une limite ne peut pas être déterminée.

Avec le réglage Approché, on procède en premier à une intégration numérique, si elle est applicable. Les primitives ne peuvent être trouvées que dans le cas où cette intégration numérique ne s'applique pas ou échoue.

Remarque: Pour afficher un résultat approximatif,

Unité : Appuyez sur  .

Windows® : Appuyez sur **Ctrl+Entrée**.

Macintosh® : Appuyez sur **⌘+Entrée**.

iPad® : Maintenez la touche **Entrée** enfoncée et sélectionnez .

$$\int_{-1}^1 e^{-x^2} dx \quad 1.49365$$

∫() peut être imbriqué pour obtenir des intégrales multiples. Les bornes d'intégration peuvent dépendre des variables d'intégration les plus extérieures.

Remarque : voir aussi **nInt()**, page 138.

$$\int_0^a \int_0^x \ln(x+y) dy dx \quad \frac{a^2 \cdot \ln(a)}{2} + \frac{a^2 \cdot (4 \cdot \ln(2) - 3)}{4}$$

√() (racine carrée)

Touches  

√(*Expr1*) ⇒ *expression*

$$\sqrt{4} \quad 2$$

√(*Liste1*) ⇒ *liste*

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

Donne la racine carrée de l'argument.

Dans le cas d'une liste, donne la liste des racines carrées des éléments de *Liste1*.

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **sqrt(...)**

Remarque : voir aussi **Modèle Racine carrée**, page 1.

$\prod()$ (prodSeq)

Catalogue > 

$\prod(\text{Expr1}, \text{Var}, \text{Début}, \text{Fin}) \Rightarrow \text{expression}$

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **prodSeq (...)**.

Calcule *Expr1* pour chaque valeur de *Var* comprise entre *Début* et *Fin* et donne le produit des résultats obtenus.

Remarque : voir aussi **Modèle Produit** ($\prod$), page 5.

$\prod(\text{Expr1}, \text{Var}, \text{Début}, \text{Début}-1) \Rightarrow 1$

$\prod(\text{Expr1}, \text{Var}, \text{Début}, \text{Fin})$

$\Rightarrow 1/\prod(\text{Expr1}, \text{Var}, \text{Fin}+1, \text{Début}-1)$ if $\text{Début} < \text{Fin}-1$

Les formules de produit utilisées sont extraites des références ci-dessous :

Ronald L. Graham, Donald E. Knuth et Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Reading, Massachusetts: Addison-Wesley, 1994.

$\prod_{n=1}^5 \left(\frac{1}{n} \right)$	$\frac{1}{120}$
$\prod_{k=1}^n (k^2)$	$(n!)^2$
$\prod_{n=1}^5 \left\{ \left\{ \frac{1}{n}, n, 2 \right\} \right\}$	$\left\{ \frac{1}{120}, 120, 32 \right\}$
$\prod_{k=4}^3 (k)$	1
$\prod_{k=4}^1 \left(\frac{1}{k} \right)$	6
$\prod_{k=4}^1 \left(\frac{1}{k} \right) \cdot \prod_{k=2}^4 \left(\frac{1}{k} \right)$	$\frac{1}{4}$

$\Sigma()$ (sumSeq)

Catalogue > 

$\Sigma(\text{Expr1}, \text{Var}, \text{Début}, \text{Fin}) \Rightarrow \text{expression}$

Remarque : vous pouvez insérer cette fonction à partir du clavier en entrant **sumSeq (...)**.

Calcule *Expr1* pour chaque valeur de *Var* comprise entre *Début* et *Fin* et donne la somme des résultats obtenus.

Remarque : voir aussi **Modèle Somme**, page 5.

$\Sigma(\text{Expr1}, \text{Var}, \text{Début}, \text{Fin}-1) \Rightarrow 0$

$\Sigma(\text{Expr1}, \text{Var}, \text{Début}, \text{Fin})$

$\sum_{n=1}^5 \left(\frac{1}{n} \right)$	$\frac{137}{60}$
$\sum_{k=1}^n (k^2)$	$\frac{n \cdot (n+1) \cdot (2 \cdot n+1)}{6}$
$\sum_{n=1}^{\infty} \left(\frac{1}{n^2} \right)$	$\frac{\pi^2}{6}$
$\sum_{k=4}^3 (k)$	0

$\Rightarrow \Sigma(\text{Expr1}, \text{Var}, \text{Fin}+1, \text{Début}-1)$ if
 $\text{Fin} < \text{Début}-1$

Le formules d'addition utilisées sont
 extraites des références ci-dessous :

Ronald L. Graham, Donald E. Knuth et
 Oren Patashnik. *Concrete Mathematics:
 A Foundation for Computer Science*.
 Reading, Massachusetts: Addison-
 Wesley, 1994.

$$\sum_{k=4}^1 (k) \quad -5$$

$$\sum_{k=4}^1 (k) + \sum_{k=2}^4 (k) \quad 4$$

ΣInt()

$\Sigma\text{Int}(\text{NPmt1}, \text{NPmt2}, \text{N}, \text{I}, \text{PV}, [\text{Pmt}],$
 $[\text{FV}], [\text{PpY}], [\text{CpY}], [\text{PmtAt}],$
 $[\text{valArrondi}]) \Rightarrow \text{valeur}$

ΣInt
 $(\text{NPmt1}, \text{NPmt2}, \text{tblAmortissement}) \Rightarrow$
 valeur

Fonction d'amortissement permettant
 de calculer la somme des intérêts au
 cours d'une plage de versements
 spécifiée.

NPmt1 et *NPmt2* définissent le début et
 la fin de la plage de versements.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* et *PmtAt*
 sont décrits dans le tableau des
 arguments TVM, page 218.

- Si vous omettez *Pmt*, il prend par
 défaut la valeur *Pmt=tvmpmt*
 $(\text{N}, \text{I}, \text{PV}, \text{FV}, \text{PpY}, \text{CpY}, \text{PmtAt})$.
- Si vous omettez *FV*, il prend par
 défaut la valeur *FV=0*.
- Les valeurs par défaut pour *PpY*, *CpY*
 et *PmtAt* sont les mêmes que pour les
 fonctions TVM.

valArrondi spécifie le nombre de
 décimales pour arrondissement. Valeur
 par défaut=2.

$\Sigma\text{Int}(1, 3, 12, 4, 75, 20000, \dots, 12, 12)$ -218.11

<i>tbl:=amortTbl(12,12,4.75,20000,,,12,12)</i>				
0	0.	0.	20000.	
1	-79.17	-1630.69	18369.3	
2	-72.71	-1637.15	16732.2	
3	-66.23	-1643.63	15088.5	
4	-59.73	-1650.13	13438.4	
5	-53.19	-1656.67	11781.7	
6	-46.64	-1663.22	10118.5	
7	-40.05	-1669.81	8448.7	
8	-33.44	-1676.42	6772.28	
9	-26.81	-1683.05	5089.23	
10	-20.14	-1689.72	3399.51	
11	-13.46	-1696.4	1703.11	
12	-6.74	-1703.12	-0.01	

$\Sigma\text{Int}(1, 3, \text{tbl})$ -218.11

ΣInt(*NPmt1*,*NPmt2*,*tblAmortissement*)
 calcule la somme de l'intérêt sur la base
 du tableau d'amortissement
tblAmortissement. L'argument
tblAmortissement doit être une matrice
 au format décrit à **tblAmortissement()**,
 page 8.

Remarque : voir également ΣPrn() ci
 dessous et Bal(), page 18.

ΣPrn()

**ΣPrn(*NPmt1*, *NPmt2*, *N*, *I*, *PV*, [*Pmt*],
 [*FV*], [*PpY*], [*CpY*], [*PmtAt*],
 [*valArrondi*])**⇒*valeur*

ΣPrn(1,3,12,4.75,20000,,,12,12) -4911.47

ΣPrn

(*NPmt1*,*NPmt2*,*tblAmortissement*)⇒
valeur

Fonction d'amortissement permettant
 de calculer la somme du capital au cours
 d'une plage de versements spécifiée.

NPmt1 et *NPmt2* définissent le début et
 la fin de la plage de versements.

N, *I*, *PV*, *Pmt*, *FV*, *PpY*, *CpY* et *PmtAt*
 sont décrits dans le tableau des
 arguments TVM, page 218.

- Si vous omettez *Pmt*, il prend par
 défaut la valeur *Pmt=tvmPmt*
(N,I,PV,FV,PpY,CpY,PmtAt).
- Si vous omettez *FV*, il prend par
 défaut la valeur *FV=0*.
- Les valeurs par défaut pour *PpY*, *CpY*
 et *PmtAt* sont les mêmes que pour les
 fonctions TVM.

valArrondi spécifie le nombre de
 décimales pour arrondissement. Valeur
 par défaut=2.

tbl:=amortTbl(12,12,4.75,20000,,,12,12)			
0	0.	0.	20000.
1	-79.17	-1630.69	18369.3
2	-72.71	-1637.15	16732.2
3	-66.23	-1643.63	15088.5
4	-59.73	-1650.13	13438.4
5	-53.19	-1656.67	11781.7
6	-46.64	-1663.22	10118.5
7	-40.05	-1669.81	8448.7
8	-33.44	-1676.42	6772.28
9	-26.81	-1683.05	5089.23
10	-20.14	-1689.72	3399.51
11	-13.46	-1696.4	1703.11
12	-6.74	-1703.12	-0.01

ΣPrn(1,3,12,12) -4911.47

ΣPrn(NPmt1, NPmt2, tblAmortissement) calcule la somme du capital sur la base du tableau d'amortissement *tblAmortissement*. L'argument *tblAmortissement* doit être une matrice au format décrit à **tblAmortissement()**, page 8.

Remarque : voir également ΣInt() ci-dessus et Bal(), page 18.

(indirection)

Touches  

ChaîneNomVar

#("x"&"y"&"z") xyz

Fait référence à la variable *ChaîneNomVar*. Permet d'utiliser des chaînes de caractères pour créer des noms de variables dans une fonction.

Crée ou fait référence à la variable xyz.

$10 \rightarrow r$	10
"r" $\rightarrow s1$	"r"
#s1	10

Donne la valeur de la variable (r) dont le nom est stocké dans la variable s1.

E (notation scientifique)

Touche 

*mantisse*EE*exposant*

23000.	23000.
--------	--------

Saisit un nombre en notation scientifique. Ce nombre est interprété sous la forme *mantisse* × 10^{*exposant*}.

23000000000.+4.1E15	4.1E15
---------------------	--------

3·10 ⁴	30000
-------------------	-------

Conseil : pour entrer une puissance de 10 sans passer par un résultat de valeur décimale, utilisez 10^{entier}.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @E. Par exemple, entrez 2.3@E4 pour avoir 2.3E4.

g (grades)

Touche 

*Expr1*g $\Rightarrow$ *expression*

En mode Angle en degrés, grades ou radians :

g (grades)**Touche** 1 $ListeI\mathbf{g} \Rightarrow liste$

$$\cos(50^g) \qquad \frac{\sqrt{2}}{2}$$

 $MatriceI\mathbf{g} \Rightarrow matrice$

$$\cos(\{0, 100^g, 200^g\}) \qquad \{1, 0, -1\}$$

Cette fonction permet d'utiliser un angle en grades en mode Angle en degrés ou en radians.

En mode Angle en radians, multiplie $Expr1$ par $\pi/200$.

En mode Angle en degrés, multiplie $Expr1$ par $g/100$.

En mode Angle en grades, donne $Expr1$ inchangée.

Remarque : vous pouvez insérer ce symbole à partir du clavier de l'ordinateur en entrant @g.

r (radians)**Touche** 1 $Expr1\mathbf{r} \Rightarrow expression$

En mode Angle en degrés, grades ou radians :

 $ListeI\mathbf{r} \Rightarrow liste$

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

 $MatriceI\mathbf{r} \Rightarrow matrice$

$$\cos\left(\left\{0^r, \frac{\pi}{12}, \frac{\pi}{r}, -(\pi)^r\right\}\right) \qquad \left\{1, \frac{(\sqrt{3}+1)\sqrt{2}}{4}, -1\right\}$$

Cette fonction permet d'utiliser un angle en radians en mode Angle en degrés ou en grades.

En mode Angle en degrés, multiplie l'argument par $180/\pi$.

En mode Angle en radians, donne l'argument inchangé.

En mode Angle en grades, multiplie l'argument par $200/\pi$.

Conseil : utilisez $\mathbf{r}$ si vous voulez forcer l'utilisation des radians dans une définition de fonction quel que soit le mode dominant lors de l'utilisation de la fonction.

r (radians)

Touche **1**

Remarque : vous pouvez insérer ce symbole à partir du clavier de l'ordinateur en entrant @r.

° (degré)

Touche **1**

*Expr*1°⇒*expression*

*Liste*1°⇒*liste*

*Matrice*1°⇒*matrice*

Cette fonction permet d'utiliser un angle en degrés en mode Angle en grades ou en radians.

En mode Angle en radians, multiplie l'argument par $\pi/180$.

En mode Angle en degrés, donne l'argument inchangé.

En mode Angle en grades, multiplie l'argument par 10/9.

Remarque : vous pouvez insérer ce symbole à partir du clavier de l'ordinateur en entrant @d.

En mode Angle en degrés, grades ou radians :

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

En mode Angle en radians :

Remarque: Pour afficher un résultat approximatif,

Unité : Appuyez sur **ctrl** **enter**.

Windows® : Appuyez sur **Ctrl+Entrée**.

Macintosh® : Appuyez sur **⌘+Entrée**.

iPad® : Maintenez la touche **Entrée** enfoncée et sélectionnez $\approx$.

$$\cos\left(\left\{0, \frac{\pi}{4}, 90^\circ, 30.12^\circ\right\}\right) \quad \{1, 0.707107, 0., 0.864976\}$$

°, ', " (degré/minute/seconde)

Touches **ctrl** 

dd°mm'ss.ss"⇒*expression*

*dd*Nombre positif ou négatif

*mm*Nombre positif ou nul

*ss.ss*Nombre positif ou nul

Donne $dd+(mm/60)+(ss.ss/3600)$.

Ce format d'entrée en base 60 permet :-

- d'entrer un angle en degrés/minutes/secondes quel que soit le mode angulaire utilisé.
- d'entrer un temps exprimé en heures/minutes/secondes.

En mode Angle en degrés :

$$\begin{array}{r} 25^\circ 13' 17.5'' \\ 25^\circ 30' \end{array} \quad \begin{array}{r} 25.2215 \\ \frac{51}{2} \end{array}$$

Remarque : faites suivre ss.ss de deux apostrophes (") et non de guillemets (").

∠ (angle)

Touches  

[Rayon,∠θ_Angle]⇒vecteur

(entrée polaire)

[Rayon,∠θ_Angle,Valeur_Z]⇒vecteur

(entrée cylindrique)

[Rayon,∠θ_Angle,∠θ_Angle]⇒vecteur

(entrée sphérique)

Donne les coordonnées sous forme de vecteur, suivant le réglage du mode Format Vecteur : rectangulaire, cylindrique ou sphérique.

Remarque : vous pouvez insérer ce symbole à partir du clavier de l'ordinateur en entrant @<.

(Grandeur ∠ Angle)⇒valeurComplexe

(entrée polaire)

Saisit une valeur complexe en coordonnées polaires (r∠θ). L'Angle est interprété suivant le mode Angle sélectionné.

En mode Angle en radians et avec le Format vecteur réglé sur :

rectangulaire

$$\left[5 \angle 60^\circ \angle 45^\circ \right] \left[\frac{5 \cdot \sqrt{2}}{4} \quad \frac{5 \cdot \sqrt{6}}{4} \quad \frac{5 \cdot \sqrt{2}}{2} \right]$$

cylindrique

$$\left[5 \angle 60^\circ \angle 45^\circ \right] \left[\frac{5 \cdot \sqrt{2}}{2} \quad \angle \frac{\pi}{3} \quad \frac{5 \cdot \sqrt{2}}{2} \right]$$

sphérique

$$\left[5 \angle 60^\circ \angle 45^\circ \right] \left[5 \quad \angle \frac{\pi}{3} \quad \angle \frac{\pi}{4} \right]$$

En mode Angle en radians et en mode Format complexe Rectangulaire :

$$5+3 \cdot i - \left(10 \angle \frac{\pi}{4} \right) \quad 5-5 \cdot \sqrt{2} + (3-5 \cdot \sqrt{2}) \cdot i$$

Remarque: Pour afficher un résultat approximatif,

Unité : Appuyez sur  .

Windows® : Appuyez sur **Ctrl+Entrée**.

Macintosh® : Appuyez sur **⌘+Entrée**.

iPad® : Maintenez la touche **Entrée** enfoncée et sélectionnez .

$$5+3 \cdot i - \left(10 \angle \frac{\pi}{4} \right) \quad -2.07107-4.07107 \cdot i$$

' (guillemets)

Touche 

variable '

variable "

Saisit le symbole prime dans une équation différentielle. Ce symbole caractérise une équation différentielle du premier ordre ; deux symboles prime, une équation différentielle du deuxième ordre, et ainsi de suite.

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

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

_ (trait bas considéré comme élément vide)

Voir "Éléments vides", page 281.

_ (trait bas considéré comme unité)

Touches  

Expr_Unité

Indique l'unité d'une *Expr*. Tous les noms d'unités doivent commencer par un trait de soulignement.

Il est possible d'utiliser les unités prédéfinies ou de créer des unités personnalisées. Pour obtenir la liste des unités prédéfinies, ouvrez le Catalogue et affichez l'onglet Conversion d'unité. Vous pouvez sélectionner les noms d'unités dans le Catalogue ou les taper directement.

Variable_

Si *Variable* n'a pas de valeur, elle est considérée comme représentant un nombre complexe. Par défaut, sans _, la variable est considérée comme réelle.

Si *Variable* a une valeur, _ est ignoré et *Variable* conserve son type de données initial.

3·_m►_ft 9.84252·_ft

Remarque : vous pouvez trouver le symbole de conversion, ►, dans le Catalogue. Cliquez sur , puis sur **Opérateurs mathématiques**.

En supposant que z est une variable non définie :

$\text{real}(z)$	z
$\text{real}(z_)$	$\text{real}(z_)$
$\text{imag}(z)$	0
$\text{imag}(z_)$	$\text{imag}(z_)$

_ (trait bas considéré comme unité)

Touches  

Remarque : vous pouvez stocker un nombre complexe dans une variable sans utiliser `_`. Toutefois, pour optimiser les résultats dans des calculs tels que `cSolve()` et `cZeros()`, l'utilisation de `_` est recommandée.

► (conversion)

Touches  

Expr_Unité1 ► _Unité2 ⇒ Expr_Unité2

3·_m ► _ft

9.84252·_ft

Convertit l'unité d'une expression.

Le trait bas de soulignement `_` indique les unités. Les unités doivent être de la même catégorie, comme Longueur ou Aire.

Pour obtenir la liste des unités prédéfinies, ouvrez le Catalogue et affichez l'onglet Conversion d'unité :

- Vous pouvez sélectionner un nom d'unité dans la liste.
- Vous pouvez sélectionner l'opérateur de conversion, ►, en haut de la liste.

Il est également possible de saisir manuellement les noms d'unités. Pour saisir « `_` » lors de l'entrée des noms d'unités sur la calculatrice, appuyez sur

 .

Remarque : pour convertir des unités de température, utilisez `tmpCnv()` et `ΔtmpCnv()`. L'opérateur de conversion ► ne gère pas les unités de température.

10^()

Catalogue > 

10^ (*Expr1*) ⇒ *expression*

$10^{1.5}$

31.6228

10^ (*Liste1*) ⇒ *liste*

$10^{\{0, 2.2, a\}}$

$\left\{1, \frac{1}{100}, 100, 10^a\right\}$

Donne 10 élevé à la puissance de l'argument.

10^()

Catalogue > 

Dans le cas d'une liste, donne 10 élevé à la puissance des éléments de *Liste1*.

10^(*matriceCarrée1*) ⇒ *matriceCarrée*

Donne 10 élevé à la puissance de *matriceCarrée1*. Ce calcul est différent du calcul de 10 élevé à la puissance de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.

matriceCarrée1 doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

$$10^{\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}} = \begin{bmatrix} 1.14336\text{E}7 & 8.17155\text{E}6 & 6.67589\text{E}6 \\ 9.95651\text{E}6 & 7.11587\text{E}6 & 5.81342\text{E}6 \\ 7.65298\text{E}6 & 5.46952\text{E}6 & 4.46845\text{E}6 \end{bmatrix}$$

^-1 (inverse)

Catalogue > 

Expr1 ^-1 ⇒ *expression*

$$(3.1)^{-1} = 0.322581$$

Liste1 ^-1 ⇒ *liste*

$$\{a, 4, -0.1, x, -2\}^{-1} = \left\{ \frac{1}{a}, \frac{1}{4}, -10, \frac{1}{x}, \frac{-1}{2} \right\}$$

Donne l'inverse de l'argument.

Dans le cas d'une liste, donne la liste des inverses des éléments de *Liste1*.

matriceCarrée1 ^-1 ⇒ *matriceCarrée*

Donne l'inverse de *matriceCarrée1*.

matriceCarrée1 doit être une matrice carrée non singulière.

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

$$\begin{bmatrix} 1 & 2 \\ a & 4 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{-2}{a-2} & \frac{1}{a-2} \\ \frac{a}{2 \cdot (a-2)} & \frac{-1}{2 \cdot (a-2)} \end{bmatrix}$$

| (opérateur "sachant que")

touches  

Expr | *ExprBooléen1*
[*andExprBooléen2*]...

$$x+1 | x=3 = 4$$

$$x+y | x=\sin(y) = \sin(y)+y$$

Expr | *ExprBooléen1*
[*orExprBooléen2*]...

$$x+y | \sin(y)=x = x+y$$

Le symbole (« | ») est utilisé comme opérateur binaire. L'opérande à gauche du symbole | est une expression. L'opérande à droite du symbole | spécifie une ou plusieurs relations destinées à affecter la simplification de l'expression. Plusieurs relations après le symbole | peuvent être reliées au moyen d'opérateurs logiques « **and** » ou « **or** ».

L'opérateur "sachant que" fournit trois types de fonctionnalités de base :

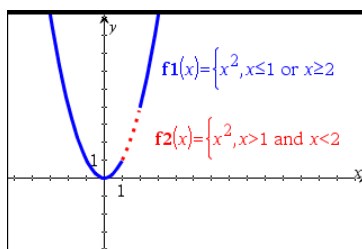
- Substitutions
- Contraintes d'intervalle
- Exclusions

Les substitutions se présentent sous la forme d'une égalité, telle que $x=3$ ou $y=\sin(x)$. Pour de meilleurs résultats, la partie gauche doit être une variable simple. *Expr* | *Variable* = *valeur* substituera une *valeur* à chaque occurrence de *Variable* dans *Expr*.

Les contraintes d'intervalle se présentent sous la forme d'une ou plusieurs inéquations reliées par des opérateurs logiques « **and** » ou « **or** ». Les contraintes d'intervalle permettent également la simplification qui autrement pourrait ne pas être valide ou calculable.

$x^3 - 2 \cdot x + 7 \rightarrow f(x)$	<i>Done</i>
$f(x) _{x=\sqrt{3}}$	$\sqrt{3} + 7$
$(\sin(x))^2 + 2 \cdot \sin(x) - 6 \sin(x) = d$	$d^2 + 2 \cdot d - 6$

$\text{solve}(x^2 - 1 = 0, x) x > 0 \text{ and } x < 2$	$x = 1$
$\sqrt{x} \cdot \sqrt{\frac{1}{x}} x > 0$	1
$\sqrt{x} \cdot \sqrt{\frac{1}{x}}$	$\sqrt{\frac{1}{x}} \cdot \sqrt{x}$



I (opérateur "sachant que")

touches

ctrl



Les exclusions utilisent l'opérateur « différent de » ($\neq$ ou $\neq$) pour exclure une valeur spécifique du calcul. Elles servent principalement à exclure une solution exacte lors de l'utilisation de **cSolve()**, **cZeros()**, **fMax()**, **fMin()**, **solve()**, **zeros()** et ainsi de suite.

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

$$x=-1$$

→ (stocker)

Touche

ctrl

var

Expr → *Var*

Liste → *Var*

Matrice → *Var*

Expr → *Fonction*(*Param1*,...)

Liste → *Fonction*(*Param1*,...)

Matrice → *Fonction*(*Param1*,...)

$$\frac{\pi}{4} \rightarrow myvar$$

$$\frac{\pi}{4}$$

$$2 \cdot \cos(x) \rightarrow yI(x)$$

Done

$$\{1, 2, 3, 4\} \rightarrow lst5$$

$$\{1, 2, 3, 4\}$$

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

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

$$\text{"Hello"} \rightarrow str1$$

$$\text{"Hello"}$$

Si la variable *Var* n'existe pas, celle-ci est créée par cette instruction et est initialisée à *Expr*, *Liste* ou *Matrice*.

Si *Var* existe déjà et n'est pas verrouillée ou protégée, son contenu est remplacé par *Expr*, *Liste* ou *Matrice*.

Conseil : si vous envisagez d'effectuer des calculs symboliques en utilisant des variables non définies, ne stockez aucune valeur dans les variables communément utilisées à une lettre, telles que a, b, c, x, y, z, et ainsi de suite.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant **=:** comme un raccourci. Par exemple, tapez **pi/4 =: Mavar**.

:= (assigner)

Touches  

Var := *Expr*

Var := *Liste*

Var := *Matrice*

Fonction(*Param1*,...) := *Expr*

Fonction(*Param1*,...) := *Liste*

Fonction(*Param1*,...) := *Matrice*

Si la variable *Var* n'existe pas, celle-ci est créée par cette instruction et est initialisée à *Expr*, *Liste* ou *Matrice*.

Si *Var* existe déjà et n'est pas verrouillée ou protégée, son contenu est remplacé par *Expr*, *Liste* ou *Matrice*.

Conseil : si vous envisagez d'effectuer des calculs symboliques en utilisant des variables non définies, ne stockez aucune valeur dans les variables communément utilisées à une lettre, telles que a, b, c, x, y, z, et ainsi de suite.

$myvar := \frac{\pi}{4}$	$\frac{\pi}{4}$
$y1(x) := 2 \cdot \cos(x)$	Done
$lst5 := \{1, 2, 3, 4\}$	$\{1, 2, 3, 4\}$
$matg := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
$str1 := \text{"Hello"}$	"Hello"

© (commentaire)

Touches  

© [*texte*]

© traite *texte* comme une ligne de commentaire, vous permettant d'annoter les fonctions et les programmes que vous créez.

© peut être utilisé au début ou n'importe où dans la ligne. Tous les caractères situés à droite de ©, jusqu'à la fin de la ligne, sont considérés comme partie intégrante du commentaire.

Remarque pour la saisie des données de l'exemple : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Define $g(n) = \text{Func}$

© 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 *nombreBinaire*

En mode base Dec :

0h *nombreHexadécimal*

0b10+0hF+10	27
-------------	----

Indique un nombre binaire ou hexadécimal, respectivement. Pour entrer un nombre binaire ou hexadécimal, vous devez utiliser respectivement le préfixe 0b ou 0h, quel que soit le mode Base utilisé. Un nombre sans préfixe est considéré comme décimal (base 10).

En mode base Bin :

0b10+0hF+10	0b11011
-------------	---------

En mode base Hex :

0b10+0hF+10	0h1B
-------------	------

Le résultat est affiché en fonction du mode Base utilisé.

TI-Nspire™ CX II - Commandes graphiques

Ceci est un document d'appoint au Guide de référence de la TI-Nspire™ et de la TI-Nspire™ CAS. Toutes les instructions de la TI-Nspire™ CX II seront intégrées et publiées dans la version 5.1 du Guide de référence de la TI-Nspire™ et de la TI-Nspire™ CAS.

Programmation en mode graphique

De nouvelles commandes de programmation graphique ont été ajoutées aux unités TI-Nspire™ CX II et aux applications pour ordinateurs TI-Nspire™.

Les unités TI-Nspire™ CX II basculeront sur ce mode graphique pour exécuter les commandes graphiques avant de revenir au contexte d'exécution du programme initial.

L'écran affiche « En cours d'exécution » sur la barre supérieure pendant que le programme s'exécute. « Terminé » sera affiché à la fin du programme. Il suffit d'appuyer sur une touche quelconque pour faire sortir le système du mode graphique.

- Le passage au mode graphique est automatiquement déclenché lorsqu'une des commandes graphiques est trouvée durant l'exécution d'un programme en TI-Basic.
- Ce passage aura lieu uniquement lors de l'exécution d'un programme dans Calculs, dans un classeur ou dans Calculs dans le scratchpad
- La sortie du mode graphique se produit à la fin de l'exécution du programme.
- Le mode graphique est uniquement disponible sur les unités TI-Nspire™ CX II et dans la vue Unité du logiciel pour ordinateur TI-Nspire™ CX II. Il n'est pas disponible dans la vue Classeur sur PC ou sur Mac.
 - Si une commande graphique est trouvée durant l'exécution d'un programme TI-Basic dans un contexte incorrect, un message d'erreur sera affiché et l'exécution du programme TI-Basic sera interrompue.

Écran de représentation graphique

L'écran de représentation graphique contient une zone d'en-tête dans laquelle les commandes graphiques ne peuvent pas écrire.

La zone de tracé de l'écran graphique sera réinitialisée (couleur = 255,255,255) à son ouverture.

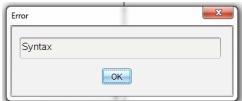
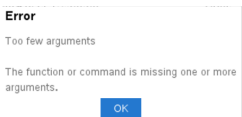
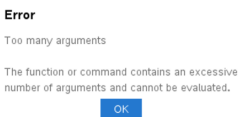
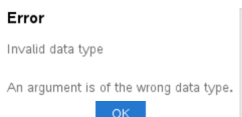
Écran de représentation graphique	Par défaut
Hauteur	212
Largeur	318
Couleur	blanc : 255,255,255

Vue et paramètres par défaut

- Les icônes d'état de la barre supérieure (voyant de batterie, verrouillage examen, indicateur de réseau etc.) ne sont pas visibles durant l'exécution d'un programme graphique.
- Couleur de trait par défaut : Noir (0,0,0)
- Style de stylo par défaut - normal, continu
 - Épaisseur : 1 (fin), 2 (normal), 3 (épais)
 - Style 1 (continu), 2 (tirets), 3 (pointillés)
- Toutes les commandes de tracé utiliseront les paramètres courants pour la couleur et le trait (valeurs par défaut ou définies via des commandes TI-Basic).
- La police du texte ne peut pas être modifiée.
- Toute sortie sur l'écran graphique sera dessinée dans une fenêtre dont la taille correspond à la zone de tracé de l'écran graphique. Toute sortie qui dépasse cette zone de tracé délimitée ne sera pas représentée. Aucun message d'erreur ne sera affiché.
- Les coordonnées (x, y) spécifiées par les commandes de tracé sont définies de façon à ce que (0,0) représente le coin supérieur gauche de la zone de tracé de l'écran graphique.
 - **Exceptions :**
 - Pour l'instruction **DrawText**, les coordonnées indiquées comme paramètres désignent l'angle inférieur gauche de la zone de délimitation du texte.
 - **SetWindow** utilise l'angle inférieur gauche de l'écran.
- Tous les paramètres (arguments) des commandes peuvent être fournis sous forme d'expressions qui sont évaluées à des nombres arrondis à l'entier le plus proche.

Messages d'erreur de l'écran graphique

Un message d'erreur sera affiché si la validation échoue.

Message d'erreur	Description	Afficher
Erreur Syntaxe	Si des erreurs de syntaxe sont détectées, un message d'erreur s'affiche et le curseur est placé, dans la mesure du possible, au niveau de la première erreur pour que vous puissiez la rectifier.	
Erreur Nombre insuffisant d'arguments	Un ou plusieurs arguments de la fonction ou de la commande n'ont pas été spécifiés	
Erreur Trop d'arguments	La fonction ou la commande est impossible à évaluer car elle contient trop d'arguments.	
Erreur Type de données incorrect	Le type de donnée de l'un des arguments est incorrect	

Commandes non valides dans le mode graphique

Certaines commandes ne sont pas autorisées une fois que le programme passe en mode graphique. Si ces commandes sont rencontrées en mode graphique, une erreur sera affichée et l'exécution du programme s'arrêtera.

Commande non autorisée	Message d'erreur
Request	La fonction Request ne peut pas être exécutée en mode graphique
RequestStr	La fonction RequestStr ne peut pas être exécutée en mode graphique
Texte	La fonction Text ne peut pas être exécutée en mode graphique

Les commandes qui affichent du texte dans Calculs - **disp** et **dispAt** - sont prises en charge dans le contexte graphique. Le texte de ces commandes sera envoyé à l'écran Calculs (et non pas sur l'écran graphique) et sera visible à la fin du programme, lorsque le système revient à l'application Calculs.

Supprimer

Catalogue > 
CXII**Clear** *x, y, largeur, hauteur*

Efface tout l'écran si aucun paramètre n'est spécifié.

Si *x, y, largeur, hauteur* sont spécifiés, le rectangle spécifié par ces paramètres sera effacé.

Supprimer

Efface la totalité de l'écran

Clear 10,10,100,50

Efface une zone rectangulaire dont le sommet supérieur gauche a pour coordonnées (10,10), une largeur égale à 100 et une hauteur à 50.

DrawArc

Catalogue > 
CXII**DrawArc** *x, y, largeur, hauteur, startAngle, arcAngle*

Trace un arc dans le rectangle spécifié, avec les angles de départ et d'arc fournis.

x, y : coordonnées du sommet supérieur gauche du rectangle de délimitation

largeur, hauteur : dimensions du rectangle de délimitation

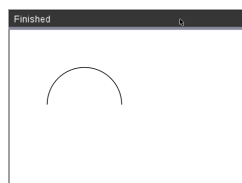
L'argument « arc angle » définit l'angle de balayage de l'arc.

Ces paramètres (arguments) peuvent être fournis sous forme d'expressions dont le résultat est arrondi à l'entier le plus proche.

DrawArc 20,20,100,100,0,90



DrawArc 50,50,100,100,0,180



Voir également : [FillArc](#)

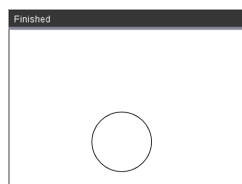
DrawCircle

Catalogue > 
CXII**DrawCircle** *x, y, rayon*

x, y : coordonnées du centre

rayon : rayon du cercle

DrawCircle 150,150,40



Voir également : [FillCircle](#)

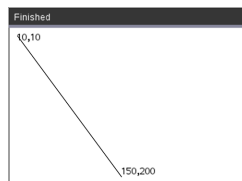
DrawLine $x1, y1, x2, y2$

Trace un segment d'extrémités $(x1, y1)$ et $(x2, y2)$.

Expressions dont le résultat est arrondi à l'entier le plus proche.

Limites de l'écran : Si les coordonnées spécifiées impliquent qu'une partie du segment soit tracée en dehors de l'écran graphique, cette partie sera tronquée sans qu'aucun message d'erreur ne soit affiché.

DrawLine 10,10,150,200


DrawPoly

Les instructions ont deux variantes :

DrawPoly $xlist, ylist$

ou

DrawPoly $x1, y1, x2, y2, x3, y3...xn, yn$

Remarque : DrawPoly $xlist, ylist$
 Shape relierax1, y1 à x2, y2, x2, y2 à x3, y3et ainsi de suite.

Remarque : DrawPoly $x1, y1, x2, y2, x3, y3...xn, yn$
 xn, yn ne seront **PAS** reliés automatiquement à $x1, y1$.

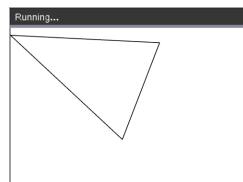
Expressions retournant une liste de nombres réels à virgule flottante
 $xlist, ylist$

Expressions évaluées à un nombre réel à virgule flottante
 $x1, y1...xn, yn$ = coordonnées des sommets du polygone

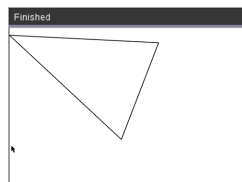
$xlist:=\{0,200,150,0\}$

$ylist:=\{10,20,150,10\}$

DrawPoly $xlist,ylist$



DrawPoly 0,10,200,20,150,150,0,10



Remarque : DrawPoly : Permet de spécifier les dimensions (largeur/ hauteur) par rapport aux segments tracés.
Les segments sont tracés dans une zone de délimitation autour des coordonnées spécifiées et dimensionnés de façon à ce que la taille réelle du polygone tracé soit supérieure à la largeur et à la hauteur indiquées.

Voir également : [FillPoly](#)

DrawRect

DrawRect *x, y, largeur, hauteur*

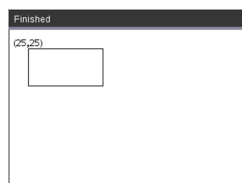
x, y : coordonnées du sommet supérieur gauche du rectangle

hauteur, largeur : hauteur et largeur du rectangle (rectangle tracé vers le bas et vers la droite à partir des coordonnées de départ).

Remarque : Les segments sont tracés dans une zone de délimitation autour des coordonnées spécifiées dont les dimensions font que la taille réelle du rectangle tracé sera supérieure à la largeur et à la hauteur indiquées.

Voir également : [FillRect](#)

DrawRect 25,25,100,50



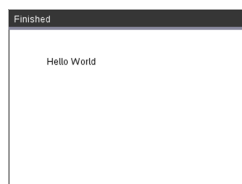
DrawText

DrawText *x, y, exprOrString1 [,exprOrString2]...*

x, y : coordonnées du texte affiché

Trace le texte inclus dans *exprOrString* à l'emplacement spécifié par les coordonnées *x, y* indiquées.

DrawText 50,50,"Hello World"



Les règles pour *exprOrString* sont les mêmes que pour **Disp** – **DrawText** peut avoir plusieurs arguments.

FillArc

Catalogue > 
CXII**FillArc** *x, y, largeur, hauteur, startAngle, arcAngle*

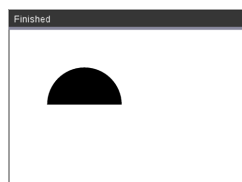
FillArc 50,50,100,100,0,180

x, y : coordonnées du sommet supérieur gauche du rectangle de délimitation

Trace et remplit un arc dans le rectangle défini, en utilisant l'angle de départ et l'angle de balayage indiqués.

La couleur de remplissage par défaut est le noir. La couleur de remplissage peut être définie via la commande [SetColor](#)

L'argument « arc angle » définit l'angle de balayage de l'arc



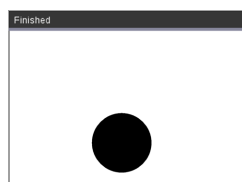
FillCircle

Catalogue > 
CXII**FillCircle** *x, y, rayon*

FillCircle 150,150,40

x, y : coordonnées du centre

Trace et remplit un cercle de centre et de rayon spécifiés.

La couleur de remplissage par défaut est le noir. La couleur de remplissage peut être définie via la commande [SetColor](#).

Ici !

FillPoly

Catalogue > 
CXII**FillPoly** *xlist, ylist*

ou

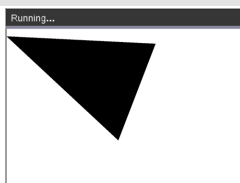
FillPoly *x1, y1, x2, y2, x3, y3...xn, yn*

xlist:={0,200,150,0}

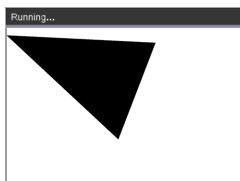
ylist:={10,20,150,10}

FillPoly xlist,ylist

Remarque : Le trait et la couleur sont définis par [SetColor](#) et [SetPen](#)



FillPoly 0,10,200,20,150,150,0,10



FillRect

FillRect *x, y, largeur, hauteur*

x, y : coordonnées du sommet supérieur gauche du rectangle

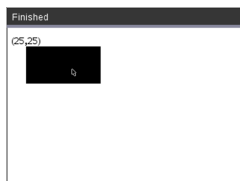
largeur, hauteur : largeur et hauteur du rectangle

Trace et remplit un rectangle dont le sommet supérieur gauche a pour coordonnées les valeurs spécifiées (*x,y*)

La couleur de remplissage par défaut est le noir. La couleur de remplissage peut être définie via la commande [SetColor](#)

Remarque : Le trait et la couleur sont définis par [SetColor](#) et [SetPen](#)

FillRect 25,25,100,50



getPlatform()Catalogue > 
CXII**getPlatform()**`getPlatform()``"dt"`

Renvoie :

« dt » sur les applications logicielles pour ordinateur

« hh » sur les unités TI-Nspire™ CX

« ios » sur l'appli TI-Nspire™ CX pour iPad®

PaintBuffer

Dessine le contenu du cache graphique sur l'écran

Cette commande s'utilise en conjonction avec UseBuffer pour augmenter la vitesse d'affichage sur l'écran lorsque le programme génère de multiples objets graphiques.

UseBuffer

For n,1,10

x:=randInt(0,300)

y:=randInt(0,200)

radius:=randInt(10,50)

Wait 0,5

DrawCircle x, y, rayon

EndFor

PaintBuffer

Ce programme affichera les 10 cercles simultanément.

Si l'instruction « UseBuffer » est retirée, chaque cercle sera affiché lorsqu'il est tracé.

Voir également : [UseBuffer](#)

PlotXY $x, y, forme$

x, y : coordonnées du tracé de la forme

forme : un nombre compris entre 1 et 13 qui indique la forme

- 1 - Cercle plein
- 2 - Cercle vide
- 3 - Carré plein
- 4 - Carré vide
- 5 - Croix
- 6 - Plus
- 7 - Fin
- 8 - point moyen, plein
- 9 - point moyen, vide
- 10 - point large, plein
- 11 - point large, vide
- 12 - point extra large, plein
- 13 - point extra large, vide

PlotXY 100,100,1

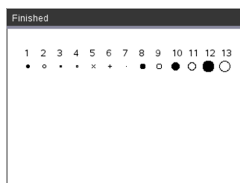


For n,1,13

DrawText 1+22*n,40,n

PlotXY 5+22*n,50,n

EndFor



SetColorCatalogue > 
CXII**SetColor**

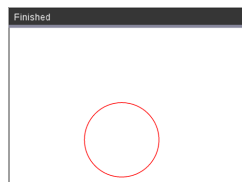
valeur rouge, valeur vert, valeur bleu

Les valeurs valides pour le rouge, le vert et le bleu sont comprises entre 0 et 255

Définit la couleur pour les commandes de tracé suivantes.

SetColor 255,0,0

DrawCircle 150,150,100

**SetPen**Catalogue > 
CXII**SetPen**

épaisseur, style

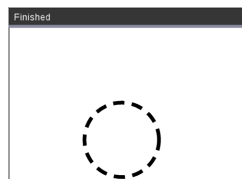
épaisseur : 1 <= épaisseur <= 3 | 1 est le plus fin, 3 est le plus épais

style : 1 = Continu, 2 = Tirets, 3 = Pointillés

Définit le style du stylo pour les commandes de tracé suivantes

SetPen 3,3

DrawCircle 150,150,50

**SetWindow**Catalogue > 
CXII**SetWindow**

xMin, xMax, yMin, yMax

Établit une fenêtre logique qui correspond à la zone de représentation graphique Tous les paramètres sont obligatoires.

Si une partie de l'objet tracé se situe en dehors de la fenêtre, le résultat sera tronqué (non affiché) sans qu'aucun message d'erreur ne soit affiché.

SetWindow 0,160,0,120

Définit les coordonnées de l'angle inférieur gauche de la fenêtre de sortie en 0,0 avec une largeur de 160 et une hauteur de 120

DrawLine 0,0,100,100

SetWindow 0,160,0,120

SetPen 3,3

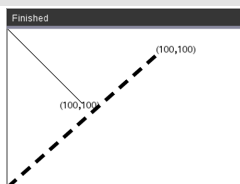
DrawLine 0,0,100,100

Si x_{min} est supérieur ou égal à x_{max} ou si y_{min} est supérieur ou égal à y_{max} , un message d'erreur s'affiche.

Tout objet tracé avant une instruction SetWindow ne sera pas retracé dans la nouvelle configuration.

Pour restaurer les paramètres par défaut de la fenêtre, utilisez :

SetWindow 0,0,0,0



UseBuffer

Catalogue > 
CXII

UseBuffer

Envoie vers la mémoire tampon de l'écran graphique au lieu d'afficher à l'écran (pour améliorer les performances)

Cette instruction est utilisée avec PaintBuffer pour accélérer l'affichage sur l'écran lorsque le programme génère de multiples objets graphiques.

Avec UseBuffer, tous les graphiques sont affichés uniquement après l'exécution de la commande PaintBuffer suivante.

UseBuffer n'a besoin d'être appelée qu'une seule fois dans le programme : chaque instruction PaintBuffer n'a pas besoin d'avoir une instruction UseBuffer correspondante.

UseBuffer

For n,1,10

x:=randInt(0,300)

y:=randInt(0,200)

radius:=randInt(10,50)

Wait 0,5

DrawCircle x, y, rayon

EndFor

PaintBuffer

Ce programme affichera les 10 cercles simultanément.

Si l'instruction « UseBuffer » est retirée, chaque cercle sera affiché lorsqu'il est tracé.

Voir également : [PaintBuffer](#)

Éléments vides

Lors de l'analyse de données réelles, il est possible que vous ne disposiez pas toujours d'un jeu complet de données. TI-Nspire™ CAS vous permet d'avoir des éléments de données vides pour vous permettre de disposer de données presque complètes plutôt que d'avoir à tout recommencer ou à supprimer les données incomplètes.

Vous trouverez un exemple de données impliquant des éléments vides dans le chapitre Tableau et listes, sous « Représentation graphique des données de tableau ».

La fonction **delVoid()** vous permet de supprimer les éléments vides d'une liste, tandis que la fonction **isVoid()** vous offre la possibilité de tester si un élément est vide. Pour plus de détails, voir **delVoid()**, page 55 et **isVoid()**, page 106.

Remarque : Pour entrer un élément vide manuellement dans une expression, tapez « _ » ou le mot clé **void**. Le mot clé **void** est automatiquement converti en caractère « _ » lors du calcul de l'expression. Pour saisir le caractère « _ » sur la calculatrice, appuyez sur  .

Calculs impliquant des éléments vides

La plupart des calculs impliquant des éléments vides génère des résultats vides. Reportez-vous à la liste des cas spéciaux ci-dessous.

	—
$\gcd(100, _)$	—
$3+ _$	—
$\{5, _, 10\} - \{3, 6, 9\}$	$\{2, _, 1\}$

Arguments de liste contenant des éléments vides

Les fonctions et commandes suivantes ignorent (passent) les éléments vides rencontrés dans les arguments de liste.

count, **countIf**, **cumulativeSum**, **freqTable**, **list**, **frequency**, **max**, **mean**, **median**, **product**, **stDevPop**, **stDevSamp**, **sum**, **sumIf**, **varPop** et **varSamp**, ainsi que les calculs de régression, **OneVar**, **TwoVar** et les statistiques **FiveNumSummary**, les intervalles de confiance et les tests statistiques.

$\text{sum}(\{2, _, 3, 5, 6, 6\})$	16.6
$\text{median}(\{1, 2, _, _, 3\})$	2
$\text{cumulativeSum}(\{1, 2, _, 4, 5\})$	$\{1, 3, _, 7, 12\}$
$\text{cumulativeSum}\left(\begin{bmatrix} 1 & 2 \\ 3 & - \\ 5 & 6 \end{bmatrix}\right)$	$\begin{bmatrix} 1 & 2 \\ 4 & - \\ 9 & 8 \end{bmatrix}$

Arguments de liste contenant des éléments vides

SortA et **SortD** déplacent tous les éléments vides du premier argument au bas de la liste.

$\{5,4,3,_,1\} \rightarrow list1$	$\{5,4,3,_,1\}$
$\{5,4,3,2,1\} \rightarrow list2$	$\{5,4,3,2,1\}$
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	$\{5,3,2,1,4\}$

Dans les regressions, la présence d'un élément vide dans la liste X ou Y génère un élément vide correspondant dans le résidu.

$I1:=\{1,2,3,4,5\}; I2:=\{2,_,3,5,6,6\}$	$\{2,_,3,5,6,6\}$
LinRegMx I1,I2	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	$\{1,_,1,1,1\}$

L'omission d'une catégorie dans les calculs de régression génère un élément vide correspondant dans le résidu.

$I1:=\{1,3,4,5\}; I2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
cat:={"M","M","F","F"}; incl:={"F"}	$\{"F"\}$
LinRegMx I1,I2,1,cat,incl	Done
stat.Resid	$\{_,_,0,0\}$
stat.XReg	$\{_,_,4,5\}$
stat.YReg	$\{_,_,5,6,6\}$
stat.FreqReg	$\{_,_,1,1\}$

Une fréquence 0 dans les calculs de régression génère un élément vide correspondant dans le résidu.

$I1:=\{1,3,4,5\}; I2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
LinRegMx I1,I2,{1,0,1,1}	Done
stat.Resid	$\{0.069231,_, -0.276923, 0.207692\}$
stat.XReg	$\{1,_,4,5\}$
stat.YReg	$\{2,_,5,6,6\}$
stat.FreqReg	$\{1,_,1,1\}$

Raccourcis de saisie d'expressions mathématiques

Les raccourcis vous permettent de saisir directement des éléments d'expressions mathématiques sans utiliser le Catalogue ni le Jeu de symboles. Par exemple, pour saisir l'expression $\sqrt{6}$, vous pouvez taper **sqrt(6)** dans la ligne de saisie. Lorsque vous appuyez sur **[enter]**, l'expression **sqrt(6)** est remplacée par $\sqrt{6}$. Certains raccourcis peuvent s'avérer très utiles aussi bien sur la calculatrice qu'à partir du clavier de l'ordinateur. Certains sont plus spécifiquement destinés à être utilisés à partir du clavier de l'ordinateur.

Sur la calculatrice ou le clavier de l'ordinateur

Pour saisir :	Utilisez le raccourci :
π	pi
θ	theta
∞	infinity
$\leq$	<=
$\geq$	>=
$\neq$	/=
$\Rightarrow$ (implication logique)	=>
$\Leftrightarrow$ (équivalence logique, XNOR)	<=>
$\rightarrow$ (opérateur de stockage)	:=
$ $ (valeur absolue)	abs (...)
$\sqrt{}$	sqrt (...)
$d()$	derivative (...)
$\int()$	integral (...)
$\Sigma()$ (Modèle Somme)	sumSeq (...)
$\Pi()$ (Modèle Produit)	prodSeq (...)
$\sin^{-1}()$, $\cos^{-1}()$, ...	arcsin (...) , arccos (...) , ...
$\Delta\text{List}()$	deltaList (...)
$\Delta\text{tmpCnv}()$	deltaTmpCnv (...)

Sur le clavier de l'ordinateur

Pour saisir :	Utilisez le raccourci :
c1, c2, ... (constantes)	@ c1 , @ c2 , ...
n1, n2, ... (constantes entières)	@ n1 , @ n2 , ...
i (le nombre complexe)	@ i
e (base du logarithme népérien e)	@ e
E (notation scientifique)	@ E
T (transposée)	@ t
r (radians)	@ r
° (degré)	@ d
g (grades)	@ g
∠ (angle)	@ <
► (conversion)	@ >
►Decimal, ►approxFraction (()), et ainsi de suite.	@ >Decimal , @ >approxFraction() , et ainsi de suite.

Hiérarchie de l'EOS™ (Equation Operating System)

Cette section décrit l'EOS™ (Equation Operating System) qu'utilise le labo de maths TI-Nspire™ CAS. Avec ce système, la saisie des nombres, des variables et des fonctions est simple et directe. Le logiciel EOS™ évalue les expressions et les équations en utilisant les groupements à l'aide de parenthèses et en respectant l'ordre de priorité décrit ci-dessous.

Ordre d'évaluation

Niveau	Opérateur
1	Parenthèses (), crochets [], accolades { }
2	Indirection (#)
3	Appels de fonction
4	Opérateurs en suffixe : degrés-minutes-secondes ([°] , ', "), factoriel (!), pourcentage (%), radian (^r), indice ([]), transposée (^T)
5	Élévation à une puissance, opérateur de puissance (^)
6	Négation (-)
7	Concaténation de chaîne (&)
8	Multiplication (•), division (/)
9	Addition (+), soustraction (-)
10	Relations d'égalité : égal à (=), différent de (≠ ou ≠), inférieur à (<), inférieur ou égal à (≤ ou ≤), supérieur à (>), supérieur ou égal à (≥ ou ≥)
11	not logique
12	and logique
13	Logique or
14	xor , nor , nand
15	Implication logique (⇒)
16	Équivalence logique, XNOR (⇔)
17	Opérateur "sachant que" (« »)
18	Stocker (→)

Parenthèses, crochets et accolades

Toutes les opérations entre parenthèses, crochets ou accolades sont calculées en premier lieu. Par exemple, dans l'expression 4(1+2), l'EOS™ évalue en premier la partie de l'expression entre parenthèses, 1+2, puis multiplie le résultat, 3, par 4.

Le nombre de parenthèses, crochets et accolades ouvrants et fermants doit être identique dans une équation ou une expression. Si tel n'est pas le cas, un message d'erreur s'affiche pour indiquer l'élément manquant. Par exemple, $(1+2)/(3+4$ génère l'affichage du message d'erreur ") manquante".

Remarque : Parce que le logiciel TI-Nspire™ CAS vous permet de définir des fonctions personnalisées, un nom de variable suivi d'une expression entre parenthèses est considéré comme un « appel de fonction » et non comme une multiplication implicite. Par exemple, $a(b+c)$ est la fonction a évaluée en $b+c$. Pour multiplier l'expression $b+c$ par la variable a , utilisez la multiplication explicite : $a*(b+c)$.

Indirection

L'opérateur d'indirection (#) convertit une chaîne en une variable ou en un nom de fonction. Par exemple, $\#("x"&"y"&"z")$ crée le nom de variable « xyz ». Cet opérateur permet également de créer et de modifier des variables à partir d'un programme. Par exemple, si $10 \rightarrow r$ et $"r" \rightarrow s1$, donc $\#s1=10$.

Opérateurs en suffixe

Les opérateurs en suffixe sont des opérateurs qui suivent immédiatement un argument, comme $5!$, 25% ou $60^\circ 15' 45''$. Les arguments suivis d'un opérateur en suffixe ont le niveau de priorité 4 dans l'ordre d'évaluation. Par exemple, dans l'expression $4^3!$, $3!$ est évalué en premier. Le résultat, 6, devient l'exposant de 4 pour donner 4096.

Élévation à une puissance

L'élévation à la puissance (^) et l'élévation à la puissance élément par élément (.^) sont évaluées de droite à gauche. Par exemple, l'expression 2^3^2 est évaluée comme 2^9 (3^2) pour donner 512. Ce qui est différent de $(2^3)^2$, qui donne 64.

Négation

Pour saisir un nombre négatif, appuyez sur $\boxed{-}$ suivi du nombre. Les opérations et élévations à la puissance postérieures sont évaluées avant la négation. Par exemple, le résultat de $-x^2$ est un nombre négatif et $-9^2 = -81$. Utilisez les parenthèses pour mettre un nombre négatif au carré, comme $(-9)^2$ qui donne 81.

Contrainte (« | »)

L'argument qui suit l'opérateur "sachant que" (« | ») applique une série de contraintes qui affectent l'évaluation de l'argument qui précède l'opérateur.

Fonctions de programmation TI-Basic sur TI-Nspire CX II

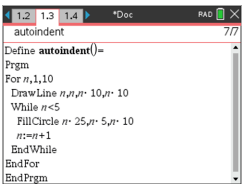
Auto-indentation dans l'Éditeur de programmes

L'Éditeur de programmes de la TI-Nspire™ indente désormais les instructions dans un bloc de commandes.

Les blocs de commandes sont If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry

L'éditeur indente automatiquement les commandes qui se trouvent dans un bloc d'instructions. L'instruction de fin de bloc sera alignée avec l'instruction de début de bloc.

L'exemple ci-dessous illustre l'indentation automatique dans les instructions de bloc imbriquées.



Les fragments de code qui sont copiés -collés conservent leur indentation originale.

Un programme créé avec une version précédente du logiciel conservera son indentation originale à l'ouverture.

Messages d'erreur améliorés pour TI-Basic

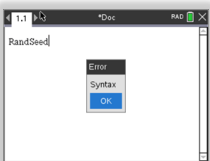
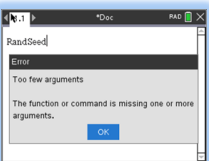
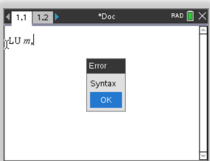
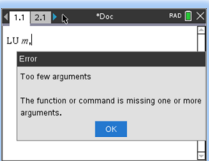
Erreurs

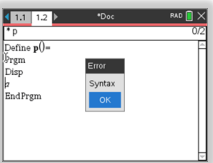
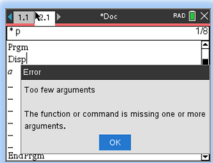
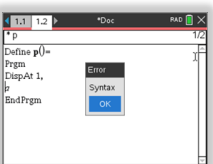
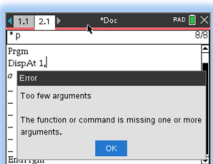
Condition d'erreur	Nouveau message
Erreur dans une instruction conditionnelle (If/While)	L'une des conditions a renvoyé une valeur qui n'était ni VRAI ni FAUX REMARQUE : Le curseur étant désormais placé sur la ligne où se trouve l'erreur, nous n'avons plus besoin d'indiquer si l'erreur se trouvait dans une instruction « If » ou une instruction « While ».
Instruction EndIf manquante	L'instruction de fin devrait être EndIf , mais une instruction de fin différente a été trouvée
Instruction Endfor manquante	L'instruction de fin devrait être EndFor , mais une instruction de fin différente a été trouvée
Instruction EndWhile manquante	L'instruction de fin devrait être EndWhile , mais une instruction de fin différente a été trouvée

Condition d'erreur	Nouveau message
Instruction EndLoop manquante	L'instruction de fin devrait être EndLoop , mais une instruction de fin différente a été trouvée
Instruction EndTry manquante	L'instruction de fin devrait être EndTry , mais une instruction de fin différente a été trouvée
« Then » manquant après If <condition>	Instruction If..Then manquante
« Then » manquant après ElseIf <condition>	Instruction Then manquante dans le bloc : Elseif
En cas d'instruction « Then », « Else » ou « Elseif » trouvée en dehors des blocs de contrôle.	Instruction Else invalide en dehors des blocs : If..Then..EndIf ou Try..EndTry
« Elseif » apparaît en dehors d'un bloc « If..Then..EndIf »	Instruction Elseif invalide en dehors du bloc : If..Then..EndIf
« Then » apparaît en dehors d'un bloc « If....EndIf »	Instruction Then invalide en dehors du bloc : If..EndIf

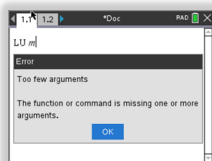
Erreurs de syntaxe

Si des instructions qui attendent un ou plusieurs arguments sont appelées avec un nombre insuffisant d'arguments, une erreur « **Nombre insuffisant d'arguments** » sera générée au lieu d'une « **erreur de syntaxe** »

Comportement actuel	Nouveaux comportements de la CX II
	
	

Comportement actuel	Nouveaux comportements de la CX II
	
	

Remarque : Lorsqu'une liste d'arguments incomplète n'est pas suivie d'une virgule, le message d'erreur est : « Nombre insuffisant d'arguments » Idem que pour les versions précédentes.



Constantes et valeurs

Le tableau suivant liste les constantes ainsi que leurs valeurs qui sont disponibles lors de la réalisation de conversions d'unités (NIST 2018). Elles peuvent être saisies manuellement ou sélectionnées depuis la liste **Constantes** dans **Utilitaires > Conversions d'unité** (Unité nomade : Appuyez sur  3).

Constante	Nom	Valeur
_c	Vitesse de la lumière	299792458 _m/_s
_Cc	Constante de Coulomb	8987551792.261 _m/_F
_Fc	Constante de Faraday	96485.33212 _coul/_mol
_g	Accélération de la pesanteur	9.80665 _m/_s²
_Gc	Constante de gravitation	6.6743E-11 _m³/_kg/_s²
_h	Constante de Planck	6.62607015E-34 _J _s
_k	Constante de Boltzmann	1.380649E-23 _J/_°K
_μ0	Perméabilité du vide	1.25663706127E-6 _N/_A²
_μb	Magnéton de Bohr	9.274009994E-24 _J _m²/_Wb
_Me	Masse de l'électron	9.10938371390E-31 _kg
_Mμ	Masse du muon	1.883531627E-28 _kg
_Mn	Masse du neutron	1.67492750058E-27 _kg
_Mp	Masse du proton	1.67262192595E-27 _kg
_Na	Nombre d'Avogadro	6.02214076E23 /_mol
_q	Charge de l'électron	1.602176634E-19 _coul
_R	Constante de pression pour le potentiel hydrique	0.0831 L bars/mole K
_Rb	Rayon de Bohr	5.29177210903E-11 _m
_Rc	Constante molaire des gaz	8.314462618 _J/_mol/_°K
_Rdb	Constante de Rydberg	10973731.568160/_m
_Re	Rayon de l'électron	2.8179403262E-15 _m
_u	Masse atomique	1.66053906893E-27 _kg
_Vm	Volume molaire	2.241396954E-2 _m³/_mol
_ε0	Permittivité du vide	8.8541878188E-12 _F/_m
_σ	Constante de Stefan-Boltzmann	5.670367E-8 _W/_m²/_°K⁴
_φ0	Quantum de flux magnétique	2.067833831E-15 _Wb

Codes et messages d'erreur

En cas d'erreur, le code correspondant est assigné à la variable *errCode*. Les programmes et fonctions définis par l'utilisateur peuvent être utilisés pour analyser *errCode* et déterminer l'origine de l'erreur. Pour obtenir un exemple d'utilisation de *errCode*, reportez-vous à l'exemple 2 fourni pour la commande **Try**, ici.

Remarque : certaines erreurs ne s'appliquent qu'aux produits TI-Nspire™ CAS, tandis que d'autres ne s'appliquent qu'aux produits TI-Nspire™.

Code d'erreur	Description
10	La fonction n'a pas retourné de valeur.
20	Le test n'a pas donné de résultat VRAI ou FAUX. En général, les variables indéfinies ne peuvent pas être comparées. Par exemple, le test <code>If a<b</code> génère cette erreur si <code>a</code> ou <code>b</code> n'est pas défini lorsque l'instruction <code>If</code> est exécutée.
30	L'argument ne peut pas être un nom de dossier.
40	Erreur d'argument
50	Argument inadapté Deux arguments ou plus doivent être de même type.
60	L'argument doit être une expression booléenne ou un entier.
70	L'argument doit être un nombre décimal.
90	L'argument doit être une liste.
100	L'argument doit être une matrice.
130	L'argument doit être une chaîne de caractères.
140	L'argument doit être un nom de variable. Assurez-vous que ce nom : • ne commence pas par un chiffre, • ne contienne ni espaces ni caractères spéciaux, • n'utilise pas le tiret de soulignement ou le point de façon incorrecte, • ne dépasse pas les limitations de longueur. Pour plus d'informations à ce sujet, reportez-vous à la section Calculs dans la documentation.
160	L'argument doit être une expression.
165	Piles trop faibles pour envoi/réception Installez des piles neuves avant toute opération d'envoi ou de réception.
170	Bornes

Code d'erreur	Description
	Pour définir l'intervalle de recherche, la limite inférieure doit être inférieure à la limite supérieure.
180	Arrêt de calcul Une pression sur la touche  ou  a été détectée au cours d'un long calcul ou lors de l'exécution d'un programme.
190	Définition circulaire Ce message s'affiche lors des opérations de simplification afin d'éviter l'épuisement total de la mémoire lors d'un remplacement infini de valeurs dans une variable en vue d'une simplification. Par exemple, $a+1 \rightarrow a$, où a représente une variable indéfinie, génère cette erreur.
200	Condition invalide Par exemple, solve($3x^2-4=0,x$) $x < 0$ or $x > 5$ génère ce message d'erreur car "or" est utilisé à la place de "and" pour séparer les contraintes.
210	Type de données incorrect Le type de l'un des arguments est incorrect.
220	Limite dépendante
230	Dimension Un index de liste ou de matrice n'est pas valide. Par exemple, si la liste {1,2,3,4} est stockée dans L1, L1[5] constitue une erreur de dimension, car L1 ne comporte que quatre éléments.
235	Erreur de dimension. Le nombre d'éléments dans les listes est insuffisant.
240	Dimension inadaptée Deux arguments ou plus doivent être de même dimension. Par exemple, [1,2]+[1,2,3] constitue une dimension inadaptée, car les matrices n'ont pas le même nombre d'éléments.
250	Division par zéro
260	Erreur de domaine Un argument doit être situé dans un domaine spécifique. Par exemple, rand(0) est incorrect.
270	Nom de variable déjà utilisé
280	Else et Elseif sont invalides hors du bloc If..EndIf.
290	La déclaration Else correspondant à EndTry manque.
295	Nombre excessif d'itérations

Code d'erreur	Description
300	Une liste ou une matrice de dimension 2 ou 3 est requise.
310	Le premier argument de nSolve doit être une équation d'une seule variable. Il ne doit pas contenir d'inconnue autre que la variable considérée.
320	Le premier argument de solve ou cSolve doit être une équation ou une inéquation. Par exemple, solve($3x^2-4$,x) n'est pas correct car le premier argument n'est pas une équation.
345	Unités incompatibles
350	Indice non valide
360	La chaîne d'indirection n'est pas un nom de variable valide.
380	Ans invalide Le calcul précédent n'a pas créé Ans, ou aucun calcul précédent n'a pas été entré.
390	Affectation invalide
400	Valeur d'affectation invalide
410	Commande invalide
430	Invalide pour les réglages du mode en cours
435	Valeur Init invalide
440	Multiplication implicite invalide Par exemple, $x(x+1)$ est incorrect ; en revanche, $x*(x+1)$ est correct. Cette syntaxe permet d'éviter toute confusion entre les multiplications implicites et les appels de fonction.
450	Invalide dans une fonction ou expression courante Seules certaines commandes sont valides à l'intérieure d'une fonction définie par l'utilisateur.
490	Invalide dans un bloc Try..EndTry
510	Liste ou matrice invalide
550	Invalide hors fonction ou programme Un certain nombre de commandes ne sont pas valides hors d'une fonction ou d'un programme. Par exemple, la commande Local ne peut pas être utilisée, excepté dans une fonction ou un programme.
560	Invalide hors des blocs Loop..EndLoop, For..EndFor ou While..EndWhile Par exemple, la commande Exit n'est valide qu'à l'intérieur de ces blocs de boucle.

Code d'erreur	Description
565	Invalide hors programme
570	Nom de chemin invalide Par exemple, \var est incorrect.
575	Complexe invalide en polaire
580	Référence de programme invalide Les programmes ne peuvent pas être référencés à l'intérieur de fonctions ou d'expressions, comme par exemple 1+p(x), où p est un programme.
600	Table invalide
605	Utilisation invalide d'unités
610	Nom de variable invalide dans une déclaration locale
620	Nom de variable ou de fonction invalide
630	Référence invalide à une variable
640	Syntaxe vectorielle invalide
650	Transmission La transmission entre deux unités n'a pas pu aboutir. Vérifiez que les deux extrémités du câble sont correctement branchées.
665	Matrice non diagonalisable
670	Mémoire insuffisante 1. Supprimez des données de ce classeur. 2. Enregistrez, puis fermez ce classeur. Si les suggestions 1 & 2 échouent, retirez les piles, puis remettez-les en place.
680	(manquante
690	) manquante
700	“ manquant
710	] manquant
720	} manquante
730	Manque d'une instruction de début ou de fin de bloc
740	Then manquant dans le bloc If..EndIf
750	Ce nom n'est pas un nom de fonction ou de programme.
765	Aucune fonction n'est sélectionnée.

Code d'erreur	Description
672	Dépassement des ressources
673	Dépassement des ressources
780	Aucune solution n'a été trouvée.
800	Résultat non réel Par exemple, si le logiciel est réglé sur Réel, $\sqrt{-1}$ n'est pas valide. Pour autoriser les résultats complexes, réglez le mode "Réel ou Complexe" sur "RECTANGULAIRE ou POLAIRE".
830	Capacité
850	Programme introuvable Une référence de programme à l'intérieur d'un autre programme est introuvable au chemin spécifié au cours de l'exécution.
855	Les fonctions aléatoires ne sont pas autorisées en mode graphique.
860	Le nombre d'appels est trop élevé.
870	Nom ou variable système réservé
900	Erreur d'argument Le modèle Med-Med n'a pas pu être appliqué à l'ensemble de données.
910	Erreur de syntaxe
920	Texte introuvable
930	Il n'y a pas assez d'arguments. Un ou plusieurs arguments de la fonction ou de la commande n'ont pas été spécifiés.
940	Il y a trop d'arguments. L'expression ou l'équation comporte un trop grand nombre d'arguments et ne peut pas être évaluée.
950	Il y a trop d'indices.
955	Il y a trop de variables indéfinies.
960	La variable n'est pas définie. Aucune valeur n'a été associée à la variable. Utilisez l'une des commandes suivantes : • <code>sto</code> → • <code>:=</code>

Code d'erreur	Description
	• Define pour assigner des valeurs aux variables.
965	O.S sans licence
970	La variable est en cours d'utilisation. Aucune référence ni modification n'est autorisée.
980	Variable protégée
990	Nom de variable invalide
	Assurez-vous que le nom n'excède pas la limite de longueur.
1000	Domaine de variables de fenêtre
1010	Zoom
1020	Erreur interne
1030	Accès illicite à la mémoire
1040	Fonction non prise en charge. Cette fonction requiert CAS (Computer Algebra System). Essayez d'utiliser TI-Nspire™ CAS.
1045	Opérateur non pris en charge. Cet opérateur requiert CAS (Computer Algebra System). Essayez d'utiliser TI-Nspire™ CAS.
1050	Fonction non prise en charge. Cet opérateur requiert CAS (Computer Algebra System). Essayez d'utiliser TI-Nspire™ CAS.
1060	L'argument entré doit être numérique. Seules les entrées comportant des valeurs numériques sont autorisées.
1070	L'argument de la fonction trig est trop grand pour une réduction fiable.
1080	Utilisation de Ans non prise en charge. Cette application n'assure pas la prise en charge de Ans.
1090	La fonction n'est pas définie. Utilisez l'une des commandes suivantes : • Define • := • sto → pour définir une fonction.
1100	Calcul non réel Par exemple, si le logiciel est réglé sur Réel, $\sqrt{-1}$ n'est pas valide. Pour autoriser les résultats complexes, réglez le mode "Réel ou Complexe" sur "RECTANGULAIRE ou POLAIRE".
1110	Limites invalides

Code d'erreur	Description
1120	Pas de changement de signe
1130	L'argument ne peut être ni une liste ni une matrice.
1140	Erreur d'argument Le premier argument doit être une expression polynomiale du second argument. Si le second argument est omis, le logiciel tente de sélectionner une valeur par défaut.
1150	Erreur d'argument Les deux premiers arguments doivent être des expressions polynomiales du troisième argument. Si le troisième argument est omis, le logiciel tente de sélectionner une valeur par défaut.
1160	Nom de chemin de bibliothèque invalide Les noms de chemins doivent utiliser le format <code>xxx\yyy</code>, où : • La partie <code>xxx</code> du nom peut contenir de 1 à 16 caractères, et • la partie <code>yyy</code>, si elle est utilisée, de 1 à 15 caractères. Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.
1170	Utilisation invalide de nom de chemin de bibliothèque • Une valeur ne peut pas être assignée à un nom de chemin en utilisant la commande Define, <code>:=</code> ou <code>sto</code> →. • Un nom de chemin ne peut pas être déclaré comme variable Local ni être utilisé dans une définition de fonction ou de programme.
1180	Nom de variable de bibliothèque invalide. Assurez-vous que ce nom : • ne contienne pas de point, • ne commence pas par un tiret de soulignement, • ne contienne pas plus de 15 caractères. Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.
1190	Classeur de bibliothèque introuvable : • Vérifiez que la bibliothèque se trouve dans le dossier Ma bibliothèque. • Rafraîchissez les bibliothèques. Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.
1200	Variable de bibliothèque introuvable :

Code d'erreur	Description
	• Vérifiez que la variable de bibliothèque existe dans la première activité de la bibliothèque. • Assurez-vous d'avoir défini la variable de bibliothèque comme objet LibPub ou LibPriv. • Rafraîchissez les bibliothèques. Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.
1210	Nom de raccourci de bibliothèque invalide Assurez-vous que ce nom : • ne contienne pas de point, • ne commence pas par un tiret de soulignement, • ne contienne pas plus de 16 caractères, • ne soit pas un nom réservé. Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.
1220	Erreur d'argument : Les fonctions tangentLine et normalline prennent uniquement en charge les fonctions à valeurs réelles.
1230	Erreur de domaine. Les opérateurs de conversion trigonométrique ne sont pas autorisés en mode Angle Degré ou Grade.
1250	Erreur d'argument Utilisez un système d'équations linéaires. Exemple de système à deux équations linéaires avec des variables x et y : $3x+7y=5$ $2y-5x=-1$
1260	Erreur d'argument : Le premier argument de nfMin ou nfMax doit être une expression dans une seule variable. Il ne doit pas contenir d'inconnue autre que la variable considérée.
1270	Erreur d'argument La dérivée doit être une dérivée première ou seconde.
1280	Erreur d'argument Utilisez un polynôme dans sa forme développée dans une seule variable.

Code d'erreur	Description
1290	Erreur d'argument Utilisez un polynôme dans une seule variable.
1300	Erreur d'argument Les coefficients du polynôme doivent s'évaluer à des valeurs numériques.
1310	Erreur d'argument : Une fonction n'a pas pu être évaluée en un ou plusieurs de ses arguments.
1380	Erreur d'argument : Les appels imbriqués de la fonction <code>domain()</code> ne sont pas permis.

Codes et messages d'avertissement

Vous pouvez utiliser la fonction **warnCodes()** pour stocker les codes d'avertissement générés lors du calcul d'une expression. Le tableau ci-dessous présente chaque code d'avertissement et le message associé. Pour un exemple de stockage des codes d'avertissement, consultez **warnCodes()**. [ici](#).

Code d'avertissement	Message
10 000	L'opération peut donner des solutions fausses. Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 001	L'équation générée par dérivation peut être fausse.
10 002	Solution incertaine Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 003	Précision incertaine Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 004	L'opération peut omettre des solutions. Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 005	CSolve peut donner plus de zéros.
10 006	Solve peut donner plus de zéros. Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 007	D'autres solutions sont possibles. Essayez de spécifier des bornes inférieure et supérieure ou une condition initiale. Exemples utilisant la fonction solve() : • solve(Equation, Var=Guess) lowBound<Var<upBound • solve(Equation, Var) lowBound<Var<upBound • solve(Equation, Var=Guess) Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 008	Le domaine du résultat peut être plus petit que le domaine de l'entrée.
10 009	Le domaine du résultat peut être plus grand que le domaine de l'entrée.
10 012	Calcul non réel

Code d'avertissement	Message
10 013	∞^0 ou undef^0 remplacé par 1
10 014	undef^0 remplacés par 1
10 015	1^∞ ou 1^undef remplacé par 1
10 016	1^undef remplacés par 1
10 017	Capacité remplacée par ∞ ou $-\infty$
10 018	Requiert et retourne une valeur 64 bits.
10 019	Ressources insuffisantes, la simplification peut être incomplète.
10 020	L'argument de la fonction trigonométrique est trop grand pour une réduction fiable.
10 021	Les données saisies comportent un paramètre non défini. Le résultat peut ne pas être valide pour toutes les valeurs possibles du paramètre.
10 022	La spécification des bornes inférieure et supérieure peut donner une solution.
10 023	Le scalaire a été multiplié par la matrice unité.
10 024	Résultat obtenu par calcul approché.
10 025	L'équivalence ne peut pas être vérifiée en mode EXACT.
10 026	Il est possible que la contrainte soit ignorée. Spécifiez la contrainte sous la forme " <code>\</code> " 'Variable MathTestSymbol Constant' ou une combinaison de ces formats, par exemple ' <code>x<3 and x>-12</code> '

Informations générales

Aide en ligne

education.ti.com/eguide

Sélectionnez votre pays pour obtenir d'autres informations relatives aux produits.

Contacter l'assistance technique TI

education.ti.com/ti-cares

Sélectionnez votre pays pour obtenir une assistance technique ou d'autres types de support.

Informations Garantie et Assistance

education.ti.com/warranty

Sélectionnez votre pays pour en savoir plus sur la durée et les termes de la garantie et sur l'assistance pour le produit.

Garantie limitée. Cette garantie n'affecte pas vos droits statutaires.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Index

!		—	
!, factorielle	246	_, désignation d'unité	257
"		 	
", secondes	255	, opérateur "sachant que"	259
#		+	
#, indirection	253	+, somme	235
#, opérateur d'indirection	286	/	
%		/, division[*]	237
%, pourcentage	241	=	
&		≠, différent de[*]	242
&, ajouter	246	=, égal à	241
*		>	
*, multiplication	236	>, supérieur à	244
,		Π	
, minutes	255	Π, produit[*]	250
-		Σ	
-, soustraction[*]	235	Σ(), somme[*]	250
.		ΣInt()	251
., multiplication élément par		ΣPrn()	252
élément	240	√	
., soustraction élément par élément	239	√, racine carrée[*]	249
./, division élément par élément	240	∫	
., Puissance élément par élément	240	∫, intégrale[*]	247
., addition élément par élément	239	≤	
:		≤, inférieur ou égal à	243
:=, assigner	262	≥	
^		≥, supérieur ou égal à	244
^, puissance	238		
^-1, inverse	259		

		©	
►, conversion d'unité[*]	258	©, commentaire	262
►, convertir mesure d'angle en		°	
grades[Grad]	98	°, degrés/minutes/secondes[*]	255
►approxFraction()	15	°, degrés[*]	255
►Base2, convertir en nombre binaire		#	
[Base2]	19	0b, indicateur binaire	263
►Base10, afficher comme entier		#	
décimal[Base10]	20	0h, indicateur hexadécimal	263
►Base16, convertir en nombre		#	
hexadécimal[Base16]	21	10^(), puissance de 10	258
►cos, exprimer les valeurs en cosinus		A	
[cos]	33	abs(), valeur absolue	8
►Cylind, afficher vecteur en		affichage degrés/minutes/secondes,	
coordonnées cylindriques		►DMS	61
[Cylind]	47	afficher comme	
►DD, afficher comme angle décimal		angle décimal, ►DD	51
[DD]	51	afficher données, Disp	59, 177
►Decimal, afficher le résultat sous		afficher vecteur	
forme décimale[décimal]	51	en coordonnées cylindriques,	
►DMS, afficher en		4Cylind	47
degrés/minutes/secondes		en coordonnées polaires, ►Polar	149
[DMS]	61	vecteur en coordonnées	
►exp, exprimer les valeurs en e[expr]	71	sphériques, ►Sphere	196
►Polar, afficher vecteur en		afficher vecteur en coordonnées	
coordonnées polaires[Polar]	149	cylindriques, ►Cylind	47
►Rad, converti la mesure de l'angle		afficher vecteur en coordonnées	
en radians	161	rectangulaires	164
►Rect, afficher vecteur en		afficher vecteur en coordonnées	
coordonnées rectangulaires	164	rectangulaires, ►Rect	164
►sin, exprimer les valeurs en sinus		afficher vecteur en coordonnées	
[sin]	187	sphériques, ►Sphere	196
►Sphere, afficher vecteur en		afficher/donner	
coordonnées sphériques		dénominateur, getDenom()	89
[Sphere]	196	informations sur les variables,	
⇒		getVarInfo()	93, 96
⇒, implication logique[*]	245, 283	nombre, getNum()	95
→		ajouter, &	246
→, stocker	261		
↔			
↔, équivalence logique[*]	245		

ajustement		fonctions TVM	218
degré 2, QuadReg	158	arguments TVM	218
degré 4, QuartReg	159	arrondi, round()	174
exponentiel, ExpReg	74	augment(), augmenter/concaténer	17
linéaire MedMed, MedMed	127	augmenter/concaténer, augment()	17
logarithmique, LnReg	117	avec,	259
Logistic	121	avgRC(), taux d'accroissement	
logistique, Logistic	122	moyen	17
MultReg	131		
puissance, PowerReg	153	B	
régression linéaire, LinRegBx	110, 112	bibliothèque	
régression linéaire, LinRegMx	111	créer des raccourcis vers des	
sinusoïdale, SinReg	190	objets	108
ajustement de degré 2, QuadReg	158	binaire	
ajustement de degré 3, CubicReg	45	convertir, ►Base2	19
ajustement exponentiel, ExpReg	74	indicateur, Ob	263
aléatoire	163	binomCdf()	22, 104
matrice, randMat()	162	binomPdf()	22
aléatoires		Boolean operators	
initialisation nombres, Germe	163	and	9
amortTbl(), tableau d'amortissement	8, 18	boucle, Loop	124
and, Boolean operator	9		
angle(), argument	10	C	
ANOVA, analyse unidirectionnelle de		caractère	
variance	11	chaîne, char()	25
ANOVA2way, analyse de variance à		code de caractère, ord()	147
deux facteurs	12	Cdf()	77
Ans, dernière réponse	14	ceiling(), entier suivant	23
approché, approx()	14, 16	centralDiff()	23
approx(), approché	14, 16	cFactor(), facteur complexe	24
approxRational()	15	chaîne	
arc cosinus, $\cos^{-1}()$	35	ajouter, &	246
arc sinus, $\sin^{-1}()$	188	chaîne de caractères, char()	25
arc tangente, $\tan^{-1}()$	205	code de caractère, ord()	147
arccos()	15	convertir chaîne en expression,	
arccosh()	15	expr()	74, 121
arccot()	15	convertir expression en chaîne,	
arcoth()	15	string()	201
arccsc()	16	décalage, shift()	183
arccsch()	16	dimension, dim()	58
arcLen(), longueur d'arc	16	format, format()	82
arcsec()	16	formatage	82
arcsech()	16	gauche, left()	107
arcsin()	16	indirection, #	253
arctan()	17	longueur	58
argsh()	16	portion de chaîne, mid()	128
argth()	17	utilisation, création de nom de	286
argument, angle()	10		
arguments présents dans les	218		

variable		degrés/minutes/secondes,	
chaîne de caractères, char()	25	►DMS	61
chaîne format, format()	82	entier décimal, ►Base10	20
chaînes		hexadécimal, ►Base16	21
dans la chaîne, inString	101	unité	258
char(), chaîne de caractères	25	convertir en	
charPoly()	25	►Rad	161
ClearAZ	28	convertir liste en matrice, list►mat()	116
ClrErr, effacer erreur	28	convertir matrice en liste, mat►list()	125
codes et messages d'erreur	300	coordonnée x rectangulaire, P►Rx()	147
codes et messages d'avertissement	300	coordonnée y rectangulaire, P►Ry()	147
colAugment	29	copier la variable ou fonction,	
colDim(), nombre de colonnes de la		CopyVar	33
matrice	29	corrMat(), matrice de corrélation ..	33
colNorm(), norme de la matrice	30	cos(), cosinus	34
combinaisons, nCr()	135	cos ⁻¹ , arc cosinus	35
comDenom(), dénominateur		cosh(), cosinus hyperbolique	36
commun	30	cosh ⁻¹ (), argument cosinus	
Commande Stop	201	hyperbolique	37
commande Text	209	cosinus	
Commande Wait	223	afficher l'expression en	33
commentaire, ©	262	cosinus, cos()	34
completeSquare(), complete square	31	cot(), cotangente	37
complexe		cot ⁻¹ (), argument cotangente	38
conjugué, conj()	32	cotangente, cot()	37
facteur, cFactor()	24	coth(), cotangente hyperbolique ...	38
résolution, cSolve()	42	coth ⁻¹ (), arc cotangente	
zéros, cZeros()	48	hyperbolique	38
comptage conditionnel déléments		count(), compter les éléments d'une	
dans une liste, countif()	39	liste	39
comptage du nombre de jours entre		countif(), comptage conditionnel ..	
deux dates, dbd()	50	déléments dans une liste ..	39
compter les éléments d'une liste,		cPolyRoots()	40
count()	39	crossP(), produit vectoriel	40
conj(), conjugué complexe	32	csc(), cosécante	41
constante		csc ⁻¹ (), argument cosécante	41
dans solve()	192	csch(), cosécante hyperbolique	42
constantes		csch ⁻¹ (), argument cosécante	
dans cSolve()	44	hyperbolique	42
dans cZeros()	49	cSolve(), résolution complexe	42
dans deSolve()	55	CubicReg, ajustement de degré 3 ...	45
dans solve()	194	cumulativeSum(), somme cumulée ..	46
constructMat(), construire une		Cycle, cycle	47
matrice	32	cycle, Cycle	47
construire une matrice,		cZeros(), zéros complexes	48
constructMat()	32		
convertir			
4Grad	98		
binaire, ►Base2	19		

D

d(), dérivée première	246
------------------------------	-----

dans la chaîne, inString()	101	déverrouillage des variables et des groupes de variables	221
dbd(), nombre de jours entre deux dates	50	diag(), matrice diagonale	58
décalage, shift()	183	différent de, ≠	242
décimal		dim(), dimension	58
afficher angle, ►DD	51	dimension, dim()	58
afficher entier, ►Base10	20	Disp, afficher données	59, 177
Define	51	DispAt	60
Define LibPriv	53	division, /	237
Define LibPub	53	domain(), domaine de définition dune fonction	62
Define, définir	51	domaine de définition dune fonction, domaine()	62
définir, Define	51	dominantTerm(), terme dominant	63
définition		dotP(), produit scalaire	64
fonction ou programme privé	53	droite, right()	102, 171
fonction ou programme public	53		
degrés, -	255	E	
degrés/minutes/secondes	255	e élevé à une puissance, e^()	65, 71
deltaList()	54	e, afficher l'expression en	71
deltaTmpCnv()	54	E, exposant	253
DelVar, suppression variable	54	e^(), e élevé à une puissance	65
delVoid(), supprimer les éléments vides	55	écart-type, stdDev()	199-200, 221
dénominateur	30	échantillon aléatoire	163
dénominateur commun, comDenom ()	30	eff), conversion du taux nominal au taux effectif	65
densité de probabilité pour la loi normale, normPdf()	140	effacer	
densité de probabilité pour la loi Student-t, tPdf()	213	erreur, ClrErr	28
derivative()	55	égal à, =	241
dérivée		eigVc(), vecteur propre	66
dérivée numérique, nDeriv()	137	eigVl(), valeur propre	66
dérivée première, d()	246	élément par élément	
dérivée implicite, Impdif()	101	addition, .+	239
dérivée ou dérivée n-ième		division, .P	240
modèle	6	multiplication, .*	240
dérivée première		puissance, .^	240
modèle	5	soustraction, .N	239
dérivée seconde		élément vide, tester	106
modèle	6	éléments vides	281
dérivées		éléments vides, supprimer	55
dérivée numérique, nDerivative()	136	else, Else	98
deSolve(), solution	55	Elseif	67
dessiner, construire	269-271	end	
det(), déterminant de matrice	57	EndLoop	124
développement trigonométrique, tExpand()	208	fonction, EndFunc	86
développer, expand()	72	if, EndIf	98
		while, EndWhile	225
		end function, EndFunc	86

end while, EndWhile	225	floor(), partie entière	79
EndIf	98	fMax(), maximum de fonction	79
EndLoop	124	fMin(), minimum de fonction	80
EndTry, end try	214	fonction	
EndWhile	225	définie par l'utilisateur	51
entier suivant, ceiling()	23, 40	fractionnaire, fpart()	82
entrée, Input	101	Func	86
EOS (Equation Operating System) ..	285	maximum, fMax()	79
Equation Operating System (EOS) ..	285	minimum, fMin()	80
équivalence logique, $\Leftrightarrow$	245	Fonction de répartition de la loi de	
erreurs et dépannage		Student-t, tCdf()	207
effacer erreur, ClrErr	28	fonction définie par morceaux (2	
passer erreur, PassErr	148	morceaux)	
étiquette, Lbl	107	modèle	2
euler(), Euler function	68	fonction définie par morceaux (n	
évaluation, ordre d	285	morceaux)	
évaluer le polynôme, polyEval()	151	modèle	3
exact(), exact	70	fonction financière, tvnFV()	217
exact, exact()	70	fonction financière, tvnI()	217
exclusion avec l'opérateur « »	259	fonction financière, tvnN()	217
Exit	70	fonction financière, tvnPmt()	218
exp(), e élevé à une puissance	71	fonction financière, tvnPv()	218
exp►hist(), conversion expression		fonctions de distribution	
en liste	72	binomCdf()	22, 104
expand(), développer	72	binomPdf()	22
exposant		invNorm()	104
modèle	1	invT()	105
exposant e		Invx ² ()	103
modèle	2	normCdf()	140
exposant, E	253	normPdf()	140
expr(), convertir chaîne en		poissCdf()	149
expression	74, 121	poissPdf()	149
ExpReg, ajustement exponentiel	74	tCdf()	207
expression		tPdf()	213
conversion expression en liste,		x ² 2way()	26
exp►hist()	72	x ² Cdf()	27
convertir chaîne en expression,		x ² GOF()	27
expr()	74, 121	x ² Pdf()	28
F		fonctions définies par l'utilisateur ...	51
factor(), factoriser	75	fonctions et programmes définis par	
factorielle, !	246	l'utilisateur	53
factorisation QR, QR	157	fonctions et variables	
factoriser, factor()	75	copie	33
Fill, remplir matrice	78	For	81
fin		format(), chaîne format	82
EndFor	81	forme échelonnée (réduite de	
FiveNumSummary	78	Gauss), ref()	165
		forme échelonnée réduite par lignes	
		(réduite de Gauss-Jordan),	176

rref()	82	argument tangente, $\tanh^{-1}()$	206
fpart(), partie fractionnaire	82	cosinus, cosh()	36
fraction		sinus, sinh()	189
FracProp	156	tangente, $\tanh()$	206
modèle	1		
fraction propre, propFrac	156	I	
freqTable()	83	identity(), matrice unité	98
frequency()	84	If	98
F-Test sur 2 échantillons	85	ifFn()	100
Func	86	imag(), partie imaginaire	100
Func, fonction	86	ImpDif(), dérivée implicite	101
		implication logique, $\Rightarrow$	245, 283
G		indirection, #	253
G, grades	253	inférieur ou égal à, {	243
gauche, left()	107	Input, entrée	101
gcd(), plus grand commun diviseur	86	inString(), dans la chaîne	101
geomCdf()	87	int(), partie entière	102
geomPdf()	87	intDiv(), quotient (division	
Get	87, 275	euclidienne)	102
getDenom(), afficher/donner		intégrale définie	
dénominateur	89	modèle	6
getKey()	90	intégrale indéfinie	
getLangInfo(), afficher/donner les		modèle	6
informations sur la langue	93	intégrale, $\int$	247
getLockInfo(), teste l'état de		interpolate(), interpoler	102
verrouillage d'une variable		inverse, $\wedge^{-1}$	259
ou d'un groupe de variables	94	invF()	103
getModel(), réglage des modes	94	invNorm((fractiles de la loi normale)	104
getNum(), afficher/donner nombre	95	invNorm(), inverse fonction de	
GetStr	95	répartition loi normale	104
getType(), get type of variable	96	invT()	105
getVarInfo(), afficher/donner les		Inv $\chi^2()$	103
informations sur les		iPart(), troncature	105
variables	96	irr(), taux interne de rentabilité	
Goto	98	taux interne de rentabilité, irr()	105
grades, G	253	isPrime(), test de nombre premier	106
groupes, tester l'état de verrouillage	94	isVoid(), tester l'élément vide	106
groupes, verrouillage et			
déverrouillage	119, 221	L	
H		langue	
hexadécimal		afficher les informations sur la	
convertir, $\Rightarrow$ Base16	21	langue	93
indicateur, 0h	263	Lbl, étiquette	107
hyperbolique		lcm, plus petit commun multiple	107
argument cosinus, $\cosh^{-1}()$	37	left(), gauche	107
argument sinus, $\sinh^{-1}()$	189	LibPriv	53
		LibPub	53

sous-matrice, subMat()	201, 203	dérivée première	5
soustraction élément par		dérivée seconde	6
élément, .N	239	e exposant	2
transposée, T	204	exposant	1
valeur propre, eigVl()	66	fonction définie par morceaux	
vecteur propre, eigVc()	66	(2 morceaux)	2
matrice (1 × 2)		fonction définie par morceaux	
modèle	4	(n morceaux)	3
matrice (2 × 1)		fraction	1
modèle	4	intégrale définie	6
matrice (2 × 2)		intégrale indéfinie	6
modèle	4	limite	7
matrice (m × n)		logarithme	2
modèle	4	matrice (1 × 2)	4
matrice de corrélation, corrMat()	33	matrice (2 × 1)	4
matrice unité, identity()	98	matrice (2 × 2)	4
matrices		matrice (m × n)	4
ajout ligne, rowAdd()	175	produit (P)	5
aléatoire, randMat()	162	racine carrée	1
échange de deux lignes,		racine n-ième	2
rowSwap()	175	somme (G)	5
forme échelonnée (réduite de		système de 2 équations	3
Gauss), ref()	165	système de n équations	3
forme échelonnée réduite par		Valeur absolue	3-4
lignes (réduite de		modes	
Gauss-Jordan), rref()	176	définition, setMode()	182
nombre de lignes, rowDim()	175	modulo, mod()	130
norme (Maximum des sommes		moyenne, mean()	126
des valeurs absolues		mRow(), opération sur ligne de	
des termes ligne par		matrice	131
ligne, rowNorm()	175	mRowAdd(), multiplication et	
unité, identity()	98	addition sur ligne de	
max(), maximum	125	matrice	131
maximum, max()	125	multiplication, *	236
mean(), moyenne	126	MultReg	131
median(), médiane	126	MultRegIntervals()	132
médiane, median()	126	MultRegTests()	133
MedMed, régression linéaire		N	
MedMed	127	nand, opérateur booléen	134
mid(), portion de chaîne	128	nCr(), combinaisons	135
min(), minimum	129	nDerivative(), dérivée numérique	136
minimum, min()	129	négation, saisie de nombres négatifs	286
minutes,	255	newList(), nouvelle liste	136
mirr(), Taux interne de rentabilité		newMat(), nouvelle matrice	136
modifié	129	nfMax(), maximum de fonction	
mod(), modulo	130	numérique	137
modèle		nfMin(), minimum de fonction	137
dérivée ou dérivée n-ième	6		

numérique		or (booléen), or	145
nInt(), intégrale numérique	138	or, opérateur booléen	145
nom), conversion du taux effectif au		ord(), code numérique de caractère	147
taux nominal	138		
nombre aléatoire, randNorm()	162	P	
nombre de jours entre deux dates,		P►Rx(), coordonnée x rectangulaire	147
dbd()	50	P►Ry(), coordonnée y rectangulaire	147
nombre de permutations, nPr()	141	partie entière, floor()	79
nor, opérateur booléen	139	partie entière, int()	102
norm(), norme de Frobenius	140	partie imaginaire, imag()	100
normale, normalLine()	140	passer erreur, PassErr	148
normalLine()	140	PassErr, passer erreur	148
normCdf()	140	Pdf()	83
norme de Frobenius, norm()	140	permutation circulaire, rotate()	172-173
normPdf()	140	piecewise()	148
not, opérateur booléen	141	plus grand commun diviseur, gcd()	86
nouvelle		plus petit commun multiple, lcm()	107
liste, newList()	136	poissCdf()	149
matrice, newMat()	136	poissPdf()	149
nPr(), nombre de permutations	141	polaire	
npv(), valeur actuelle nette	142	coordonnée polaire, R►Pr()	160
nSolve(), solution numérique	143	coordonnée polaire, R►Pθ()	160
numérique		polar	
dérivée, nDeriv()	137	afficher vecteur, vecteur en	
dérivée, nDerivative()	136	coordonnées 4Polar	149
intégrale, nInt()	138	polyCoef()	150
solution, nSolve()	143	polyDegree()	151
		polyEval(), évaluer le polynôme	151
O		polyGcd()	152
objet		polynôme	
créer des raccourcis vers la		évaluer, polyEval()	151
bibliothèque	108	polynôme de Taylor, taylor()	207
OneVar, statistiques à une variable	144	polynôme, randPoly()	163
opérateur		polynômes	
ordre dévaluation	285	aléatoire, randPoly()	163
opérateur "sachant que" « »	259	PolyRoots()	153
opérateur "sachant que", ordre		portion de chaîne, mid()	128
dévaluation	285	pourcentage, %	241
opérateur d'indirection (#)	286	PowerReg, puissance	153
Opérateurs booléens		Prgm, définir programme	154
⇒	245	probabilité de loi normale, normCdf(	
⇔	245	)	140
nand	134	prodSeq()	155
nor	139	product(), produit	155
not	141	produit (P)	
or	145	modèle	5
p	283	produit vectoriel, crossP()	40
xor	225	produit, P()	250

produit, product()	155	réduite de Gauss-Jordan, rref()	176
programmation		réel, real()	164
afficher données, Disp	59, 177	ref(), forme échelonnée (réduite de Gauss)	165
définir programme, Prgm	154	RefreshProbeVars	166
passer erreur, PassErr	148	réglage des modes, getModel()	94
programmes		réglages, mode actuel	94
définition d'une bibliothèque privée	53	régression	
définition d'une bibliothèque publique	53	degré 3, CubicReg	45
programmes et programmation		puissance, PowerReg	153, 209
afficher écran E/S, Disp	59	régression de degré 4, QuartReg	159
afficher l'écran E/S, Disp	177	régression linéaire MedMed,	
effacer erreur, ClrErr	28	MedMed	127
try, Try	214	régression linéaire, LinRegBx	110, 112
propFrac, fraction propre	156	régression linéaire, LinRegMx	111
puissance de 10, 10^()	258	régression logarithmique, LnReg	117
puissance, ^	238	régression logistique, Logistic	121
puissance, PowerReg	153, 167, 169, 209	régression logistique, LogisticD	122
		régression sinusoïdale, SinReg	190
		regressions	
		Regression puissance, PowerReg	167, 169
		remin(), reste (division euclidienne)	167
		réponse (dernière), Ans	14
		Request	167
		RequestStr	169
		résolution simultanée d'équations,	
		simult()	185
		résolution, solve()	191
		reste (division euclidienne), remin()	167
		résultat	
		exprime les valeurs en e	71
		exprimer les valeurs en cosinus	33
		exprimer les valeurs en sinus	187
		résultat, statistiques	198
		Return	170
		Return, renvoi	170
		right(), droite	171
		right, right()	31, 68, 223
		rk23(), fonction de Runge-Kutta	171
		rotate(), permutation circulaire	172-173
		round(), arrondi	174
		rowAdd(), ajout ligne de matrice	175
		rowDim(), nombre de lignes de la matrice	175
		rowNorm(), norme de la matrice (Maximum des sommes des valeurs absolues des termes ligne par ligne)	175
		rowSwap(), échange de deux lignes	175
Q			
QR, factorisation QR	157		
QuadReg, ajustement de degré 2	158		
QuartReg, régression de degré 4	159		
quotient (division euclidienne), intDiv()	102		
R			
R, radians	254		
R►Pr(), coordonnée polaire	160		
R►Pθ(), coordonnée polaire	160		
raccourcis clavier	283		
raccourcis, clavier	283		
racine carrée			
modèle	1		
racine carrée, √()	197, 249		
racine n-ième			
modèle	2		
radians, R	254		
rand(), nombre aléatoire	161		
randBin, nombre aléatoire	161		
randInt(), entier aléatoire	162		
randMat(), matrice aléatoire	162		
randNorm(), nombre aléatoire	162		
randPoly(), polynôme aléatoire	163		
randSamp()	163		
RandSeed, initialisation nombres aléatoires	163		

de la matrice		soustraction, -	235
		sqrt(), racine carrée	197
		stat.results	198
		stat.values	199
	S	statistique	
scalaire		combinaisons, nCr()	135
produit, dotP()	64	écart-type, stdDev()	199-200, 221
sec(), secante	176	factorielle, !	246
sec ⁻¹ (), arc sécante	177	médiane, median()	126
sech(), sécante hyperbolique	177	moyenne, mean()	126
sech ⁻¹ (), argument sécante		nombre de permutations, nPr()	141
hyperbolique	177	statistiques à deux variables,	
secondes, "	255	TwoVar	219
seq(), suite	178	statistiques à une variable,	
seqGen()	179	OneVar	144
seqn()	179	variance, variance()	222
sequence, seq()	179	statistiques	
série, series()	180	initialisation nombres	
series(), série	180	aléatoires, Germe	163
set		nombre aléatoire, randNorm()	162
mode, setMode()	182	statistiques à deux variables,	
setMode(), définir mode	182	TwoVar	219
shift(), décalage	183	statistiques à une variable, OneVar	144
sign(), signe	185	stdDevPop(), écart-type de	
signe, sign()	185	population	199
simult(), résolution simultanée		stdDevSamp(), écart-type	
déquations	185	déchantillon	200
sin(), sinus	187	stockage	
sin ⁻¹ (), arc sinus	188	symbole, &	261-262
sinh(), sinus hyperbolique	189	string(), convertir expression en	
sinh ⁻¹ (), argument sinus		chaîne	201
hyperbolique	189	strings	
SinReg, régression sinusoidale	190	droite, right()	102, 171
sinus		permutation circulaire, rotate()	172-173
afficher l'expression en	187	right, right()	31, 68, 223
sinus, sin()	187	subMat(), sous-matrice	201, 203
solution, deSolve()	55	substitution avec l'opérateur « » ..	259
solve(), résolution	191	suite, seq()	178
somme (G)		sum(), somme	202
modèle	5	sumif()	202
somme cumulée, cumulativeSum() ..	46	sumSeq()	203
somme des intérêts versés	251	supérieur à, >	244
somme du capital versé	252	supérieur ou égal à, 	244
somme, +	235	suppression	
somme, sum()	202	variable, DelVar	54
somme, Σ()	250	supprimer	
SortA, tri croissant	195	éléments vides d'une liste	55
SortD, tri décroissant	196	Supprimer	268
soulignement, _	257	Syntaxe de	273-274
sous-matrice, subMat()	201, 203		

système de 2 équations modèle	3	Try, commande de gestion des erreurs	214
système de n équations modèle	3	Try, try	214
		try, Try	214
		t-test de régression linéaire multiple	133
		tTest, test t	215
		tTest_2Samp, test t sur deux échantillons	216
		tvmFV()	217
		tvmI()	217
		tvmN()	217
		tvmPmt()	218
		tvmPV()	218
		TwoVar, statistiques à deux variables	219
T			
T, transposée	204		
tableau damortissement, amortTbl()	8, 18		
tan(), tangente	204		
tan ⁻¹ (), arc tangente	205		
tangente, tan()	204		
tangente, tangentLine()	206		
tangentLine()	206		
tanh(), tangente hyperbolique	206		
tanh ⁻¹ (), argument tangente hyperbolique	206		
taux daccroissement moyen, avgRC()	17		
taux effectif, eff)	65		
Taux interne de rentabilité modifié, mirr()	129		
Taux nominal, nom()	138		
taylor(), polynôme de Taylor	207		
tCdf(), fonction de répartition de loi de studentt	207		
tCollect(), linéarisation trigonométrique	208		
terme dominant, dominantTerm()	63		
test de nombre premier, isPrime()	106		
test t, tTest	215		
Test_2S, F-Test sur 2 échantillons	85		
tester l'élément vide, isVoid()	106		
tExpand(), développement trigonométrique	208		
tInterval, intervalle de confiance t	210		
tInterval_2Samp, intervalle de confiance t sur 2 échantillons	211		
tmpCnv()	212		
tPdf(), densité de probabilité pour la loi Studentt	213		
trace()	213		
trait bas, _	257		
transposée, T	204		
tri croissant, SortA	195		
décroissant, SortD	196		
troncature, iPart()	105		
		U	
		unité convertir	258
		unitV(), vecteur unitaire	221
		unLock, déverrouiller une variable ou un groupe de variables	221
V			
		Valeur absolue modèle	3-4
		valeur actuelle nette, npv()	142
		valeur propre, eigVl()	66
		valeur temporelle de l'argent, montant des versements	218
		valeur temporelle de l'argent, nombre de versements	217
		valeur temporelle de l'argent, taux d'intérêt	217
		valeur temporelle de l'argent, valeur acquise	217
		valeur temporelle de l'argent, valeur actuelle	218
		valeurs de résultat, statistiques variable	199
		locale, Local	119
		nom, création à partir d'une chaîne de caractères	286
		suppression, DelVar	54
		supprimer toutes les variables à une lettre	28
		variable locale, Local	119

variables et fonctions		zTest_2Prop, test z pour deux proportions	233
copie	33	zTest_2Samp, test z sur deux échantillons	234
variables, verrouillage et déverrouillage	94, 119, 221		
variance, variance()	222	Δ	
varPop()	221	Δlist(), liste des différences	115
varSamp(), variance déchantillon ..	222	ΔtmpCnv() [tmpCnv]	212
vecteur			
afficher vecteur en coordonnées cylindriques, ►Cylind ..	47	X	
produit scalaire, dotP()	64	χ ² 2way	26
produit vectoriel, crossP()	40	χ ² Cdf()	27
unitaire, unitV()	221	χ ² GOF	27
vecteur propre, eigVc()	66	χ ² Pdf()	28
vecteur unitaire, unitV()	221		
verrouillage des variables et des groupes de variables	119		
W			
warnCodes(), Warning codes	223		
when(), when	224		
when, when()	224		
While, while	225		
while, While	225		
X			
x ² , carré	239		
XNOR	245		
xor, exclusif booléen or	225		
Z			
zeros(), zéros	226		
zéros, zeros()	226		
zInterval, intervalle de confiance z ..	229		
zInterval_1Prop, intervalle de confiance z pour une proportion	229		
zInterval_2Prop, intervalle de confiance z pour deux proportions	230		
zInterval_2Samp, intervalle de confiance z sur 2 échantillons	231		
zTest	231		
zTest_1Prop, test z pour une proportion	232		