Texas Instruments

TI-Nspire™ CAS
TI-Nspire™ CX CAS
Guide de référence

Ce manuel fait référence au logiciel TI-Nspire™ version 3.2. Pour obtenir la dernière version de ce document, rendez-vous sur education.ti.com/guides.

Informations importantes

Sauf spécification contraire prévue dans la Licence fournie avec le programme, Texas Instruments n'accorde aucune garantie expresse ou implicite, ce qui inclut sans pour autant s'y limiter les garanties implicites quant à la qualité marchande et au caractère approprié à des fins particulières, liés aux programmes ou aux documents et fournit seulement ces matériels en l'état. En aucun cas, Texas Instruments n'assumera aucune responsabilité envers quiconque en cas de dommages spéciaux, collatéraux, accessoires ou consécutifs, liés ou survenant du fait de l'acquisition ou de l'utilisation de ces matériels. La seule et unique responsabilité incombant à Texas Instruments, indépendamment de la forme d'action, ne doit pas excéder la somme établie dans la licence du programme. En outre, Texas Instruments ne sera pas responsable des plaintes de quelque nature que soit, à l'encontre de l'utilisation de ces matériels, déposées par une quelconque tierce partie.

Licence

Veuillez consulter la licence complète, copiée dans **C:\Program Files\TI Education\<TI-Nspire™ Product Name>\license**.

Table des matières

Informations importantes

Modèles d'expression

Liste alphabétique

A

B

C

D

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Symboles

Éléments vides

Raccourcis de saisie d'expressions mathématiques

Hiérarchie de l'EOS™ (Equation Operating System)

Codes et messages d'erreur

Codes et messages d'avertissement

Informations sur les services et la garantie TI

Guide de référence TI-Nspire™ CAS

Ce guide fournit la liste des modèles, fonctions, commandes et opérateurs disponibles pour le calcul d'expressions mathématiques.

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 147.

Exemple :

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

Modèle Exposant — **Touche** [^]

Exemple :

$$2^3 \qquad 8$$

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 148.

Modèle Racine carrée — **Touches** [ctrl] [x²]

Remarque : Voir aussi **√() (racine carrée)**, page 155.

Exemple :

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

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

Modèle Racine n-ième — **Touches** [ctrl] [^]

Remarque : Voir aussi **root()**, page 105.

Exemple :

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

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

Modèle e Exposant

Touches e^x

$e^{\square}$

La base du logarithme népérien *e* élevée à une puissance

Remarque : Voir aussi **e^()**, page 41.

Exemple :

e^1 → e

$e^{1.}$ → 2.71828182846

Modèle Logarithme

Touches ctrl 10^x

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

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 72.

Exemple :

$\log_4(2.)$ → 0.5

Modèle Fonction définie par morceaux (2 morceaux)

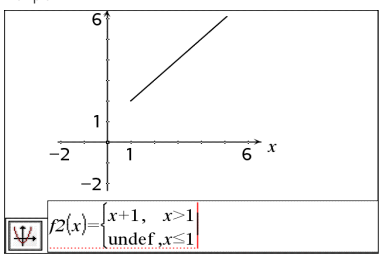
Catalogue >

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

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 90.

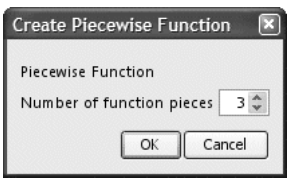
Exemple :



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*.



Remarque : Voir aussi **piecewise()**, page 90.

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

Modèle Système de 2 équations

Catalogue >

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

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 125.

Exemple :

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

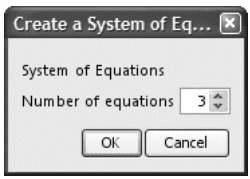
$\text{solve}\left(\begin{cases} y=x^2-2 \\ x+2\cdot y=-1 \end{cases}, x, y\right)$

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

Modèle Système de n équations

Catalogue >

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



Remarque : Voir aussi **system()**, page 125.

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

Modèle Valeur absolue

Catalogue >

$|\square|$ **Remarque :** Voir aussi **abs()**, page 6.

Exemple :

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

Modèle dd°mm'ss.ss''

Catalogue >

$\square°\square'\square''$

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.

Exemple :

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

Modèle Matrice (2 x 2)

Catalogue >

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

Crée une matrice de type 2 x 2.

Exemple :

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

Modèle Matrice (1 x 2)

Catalogue >

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

Exemple :

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

Modèle Matrice (2 x 1)

Catalogue >

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

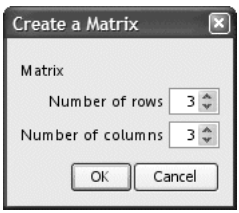
Exemple :

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

Modèle Matrice (m x n)

Catalogue >

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



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

Exemple :

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

Modèle Somme (Σ)

Catalogue >

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

Exemple :

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

Remarque : voir aussi **Σ() (sumSeq)**, page 156.

Modèle Produit (Π)

Catalogue >

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

Exemple :

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

Remarque : Voir aussi **Π() (prodSeq)**, page 156.

Modèle Dérivée première

Catalogue >

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

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

Remarque : voir aussi **d() (dérivée)**, page 154.

Par exemple :

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

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

Modèle Dérivée seconde **Catalogue >**

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

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

Remarque : voir aussi **d() (dérivée)**, page 154.

Par exemple :

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

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

Modèle Dérivée n-ième **Catalogue >**

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

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

Remarque : Voir aussi **d() (dérivée)**, page 154.

Exemple :

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

Modèle Intégrale définie **Catalogue >**

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

Remarque : voir aussi **∫() integral()**, page 154.

Exemple :

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

Modèle Intégrale indéfinie **Catalogue >**

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

Remarque : Voir aussi **∫() integral()**, page 154.

Exemple :

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

Modèle Limite **Catalogue >**

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

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

Remarque : Voir aussi **limit()**, page 64.

Exemple :

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

Liste alphabétique

Les éléments dont le nom n'est pas alphabétique (comme +, !, et >) apparaissent à la fin de cette section, à partir de la page 146. 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(*Expr1***)** ⇒ *expression*
abs(*Liste1***)** ⇒ *liste*
abs(*Matrice1***)** ⇒ *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.

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

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

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

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

amortTbl()

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 135.

- 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.

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 **ΣInt()** et **ΣPrn()**, page 157 et **bal()**, page 13.

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

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

and

Catalogue >

Expr booléenne1 **and** *Expr booléenne2* ⇒ *Expression booléenne*
Liste booléenne1 **et** *Liste booléenne2* ⇒ *Liste booléenne*
Matrice booléenne1 **et** *Matrice booléenne2* ⇒ *Matrice booléenne*

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

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

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

and

Catalogue >

Entier1 **and** *Entier2* ⇒ *entier*

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 Hex :

0h7AC36 and 0h3D5F → 0h2C16

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

En mode base Bin :

0b100101 and 0b100 → 0b100

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()

Catalogue >

angle(*Expr1***)** ⇒ *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 :

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

En mode Angle en grades :

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

En mode Angle en radians :

$\text{angle}(1+i) \to \frac{\pi}{4}$

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

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

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.

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

ANOVA

Catalogue >

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 122.)

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

Variable de sortie	Description
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
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 122.)

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
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 ⇒ *valeur*

Donne le résultat de la dernière expression calculée.

56	56
56+4	60
60+4	64

approx()

Catalogue >

approx(*Expr1***)** ⇒ *expression*

Donne une approximation décimale de l'argument sous forme d'expression, dans la mesure du possible, indépendamment du mode **Auto ou Approché** utilisé.

Ceci est équivalent à la saisie de l'argument suivie d'une pression sur ctrl enter.

$\text{approx}\left(\frac{1}{3}\right)$	0.333333
$\text{approx}\left(\left\{\frac{1}{3},\frac{1}{9}\right\}\right)$	{0.333333,0.111111}
$\text{approx}(\{\sin(\pi),\cos(\pi)\})$	{0.,-1.}
$\text{approx}([\sqrt{2}\quad\sqrt{3}])$	[1.41421 1.73205]
$\text{approx}\left(\begin{bmatrix}\frac{1}{3} & \frac{1}{9}\end{bmatrix}\right)$	[0.333333 0.111111]

approx(*Liste1***)** ⇒ *liste*
approx(*Matrice1***)** ⇒ *matrice*

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

$\text{approx}(\{\sin(\pi),\cos(\pi)\})$	{0.,-1.}
$\text{approx}([\sqrt{2}\quad\sqrt{3}])$	[1.41421 1.73205]

▸approxFraction()

Catalogue >

Expr▸**approxFraction([***tol***])** ⇒ *expression*
Liste▸**approxFraction([***tol***])** ⇒ *liste*
Matrice▸**approxFraction([***tol***])** ⇒ *matrice*

Donne l'entrée sous forme de fraction en utilisant une tolérance *tol*. Si *tol* 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(...)`.

$\frac{1}{2}+\frac{1}{3}+\tan(\pi)$	0.833333
$0.83333333333333\blacktriangleright\text{approxFraction}(5.\text{E}^{-}14)$	$\frac{5}{6}$
$\{\pi,1.5\}\blacktriangleright\text{approxFraction}(5.\text{E}^{-}14)$	$\left\{\frac{5419351}{1725033},\frac{3}{2}\right\}$

approxRational()

Catalogue >

approxRational(*Expr*[**,** *tol*]**)** ⇒ *expression*
approxRational(*Liste*[**,** *tol*]**)** ⇒ *liste*
approxRational(*Matrice*[**,** *tol*]**)** ⇒ *matrice*

Donne l'argument sous forme de fraction en utilisant une tolérance *tol*. Si *tol* est omis, la tolérance 5.E-14 est utilisée.

$\text{approxRational}(0.333,5\cdot10^{-5})$	$\frac{333}{1000}$
$\text{approxRational}(\{0.2,0.33,4.125\},5.\text{E}^{-}14)$	$\left\{\frac{1}{5},\frac{33}{100},\frac{33}{8}\right\}$

arccos()

Voir cos⁻¹(), page 24.

arccosh() Voir $\cosh^{-1}()$, page 24.

arccot() Voir $\cot^{-1}()$, page 25.

arccoth() Voir $\coth^{-1}()$, page 26.

arccsc() Voir $\csc^{-1}()$, page 28.

arccsch() Voir $\operatorname{csch}^{-1}()$, page 28.

arcLen() **Catalogue >**

arcLen(*Expr1***,***Var***,***Début***,***Fin***)** ⇒ *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*.

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

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

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

arcLen(*Liste1***,***Var***,***Début***,***Fin***)** ⇒ *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*.

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

arcsec() Voir $\sec^{-1}()$, page 108.

arcsech() Voir $\operatorname{sech}^{-1}()$, page 109.

arcsin() Voir $\sin^{-1}()$, page 116.

arcsinh() Voir $\sinh^{-1}()$, page 116.

arctan() Voir $\tan^{-1}()$, page 127.

arctanh() Voir $\tanh^{-1}()$, page 128.

augment() **Catalogue >**

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

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

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

augment()

Catalogue >

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.

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

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

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

avgRC()

Catalogue >

avgRC(*Expr1***,** *Var* [=*Valeur*] [**,** *Incrément*]**)** ⇒ *expression*
avgRC(*Expr1***,** *Var* [=*Valeur*] [**,** *Liste1*]**)** ⇒ *liste*
avgRC(*Liste1***,** *Var* [=*Valeur*] [**,** *Incrément*]**)** ⇒ *liste*
avgRC(*Matrice1***,** *Var* [=*Valeur*] [**,** *Incrément*]**)** ⇒ *matrice*

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.

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

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

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

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

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

B

bal()

Catalogue >

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, page 135.

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, page 135.

- 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.

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()**, page 6.

Remarque : voir également **Σint()** et **Σprn()**, page 157.

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

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

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

bal(4,*tbl*) 1674.27

▶Base2

Catalogue >

Entier1 **▶Base2** ⇒ *entier*

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

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

256▶Base2 0b100000000

0h1F▶Base2 0b11111

▸Base2

Catalogue >

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

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
0hFFFFFFFFFFFFFFFF 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

▸Base10

Catalogue >

Entier1 **▸Base10** ⇒ *entier*

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.

0b10011▸Base10	19
0h1F▸Base10	31

▶Base16

Catalogue >

Entier1 **▶Base16** ⇒ *entier*

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

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 13.

256▶Base16	0h100
0b111100001111▶Base16	0hF0F

binomCdf()

Catalogue >

binomCdf(*n***,***p***)** ⇒ *nombre*

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≤X≤*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 ≤ *upBound*), définissez la borne *lowBound*=0

binomPdf()

Catalogue >

binomPdf(*n***,***p***)** ⇒ *nombre*

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.

C

ceiling()

Catalogue >

ceiling(*Expr1***)** ⇒ *entier*

Donne le plus petit entier ≥ à l'argument.

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

Remarque : Voir aussi **floor()**.

ceiling(*Liste1***)** ⇒ *liste*
ceiling(*Matrice1***)** ⇒ *matrice*

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

ceiling(.456)	1.
ceiling({−3.1,1,2.5})	{−3.,1,3.}
ceiling$\begin{bmatrix} 0 & -3.2\cdot i \\ 1.3 & 4 \end{bmatrix}$	$\begin{bmatrix} 0 & -3.\cdot i \\ 2. & 4 \end{bmatrix}$

centralDiff()

Catalogue >

centralDiff(*Expr1*,*Var* [=*Valeur*][,*Pas*]**)** ⇒ *expression*
centralDiff(*Expr1*,*Var* [,*Pas*]**)**|*Var*=*Valeur* ⇒ *expression*
centralDiff(*Expr1*,*Var* [=*Valeur*][,*Liste*]**)** ⇒ *liste*
centralDiff(*Liste1*,*Var* [=*Valeur*][,*Incrément*]**)** ⇒ *liste*
centralDiff(*Matrice1*,*Var* [=*Valeur*][,*Incrément*]**)** ⇒ *matrice*

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()**.

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

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

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

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

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

cFactor()

Catalogue >

cFactor(*Expr1*[,*Var*]**)** ⇒ *expression*
cFactor(*Liste1*[,*Var*]**)** ⇒ *liste*
cFactor(*Matrice1*[,*Var*]**)** ⇒ *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.

$$\text{cFactor}(a^3\cdot x^2+a\cdot x^2+a^3+a) \qquad a\cdot(a+-i)\cdot(a+i)\cdot(x+-i)\cdot(x+i)$$

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

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

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

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.

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.

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

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

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

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()**.

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

$$\text{cFactor}(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3,x) \qquad (x-0.964673)\cdot(x+0.611649)\cdot(x+2.12543)\cdot(x\blacktriangleright$$

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

char()

Catalogue >

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.

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

charPoly()

Catalogue >

charPoly(*matriceCarrée,Var***)** ⇒ *expression polynomiale*
charPoly(*matriceCarrée,Expr***)** ⇒ *expression polynomiale*
charPoly(*matriceCarrée1,matriceCarrée2***)** ⇒ *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

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

où $\boldsymbol{I}$ désigne la matrice identité $n \times n$.

matriceCarrée1 et *matriceCarrée2* doivent avoir les mêmes dimensions.

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

charPoly(m,x)	$-x^3+5\cdot x^2+7\cdot x-35$
charPoly(m,x^2+1)	$-x^6+2\cdot x^4+14\cdot x^2-24$
charPoly(m,m)	0

$\chi^2$2way

Catalogue >

$\chi^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 122.)

Pour plus d'informations concernant les éléments vides dans une matrice, reportez-vous à "Éléments vides", page 165.

Variable de sortie	Description
stat.χ^2	Stats Khi^2 : sum(observée - attendue)^2/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^2Cdf()

Catalogue >

χ^2Cdf(*lowBound*,*upBound*,*df*) ⇒ *nombre* si les bornes *lowBound* et *upBound* sont des nombres, *liste* si les bornes *lowBound* et *upBound* sont des listes

chi2Cdf(*lowBound*,*upBound*,*df*) ⇒ *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 *df* degrés de liberté prenne une valeur entre les bornes *lowBound* et *upBound*.

Pour $P(X \le upBound)$, définissez la borne lowBound=0.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 165.

χ^2GOF

Catalogue >

χ^2GOF *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 122.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 165.

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

χ^2Pdf()

Catalogue >

χ^2Pdf(*ValX*,*df*) ⇒ *nombre* si *ValX* est un nombre, *liste* si *XVal* est une liste
chi2Pdf(*ValX*,*df*) ⇒ *nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

Calcule la probabilité qu'une variable suivant une loi χ^2 à *df* 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 165.

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 137.

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

ClrErr

Catalogue >

ClrErr

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

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 90 et **Try**, page 132.

Remarque pour la saisie des données de l'exemple : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

Pour obtenir un exemple de **ClrErr**, reportez-vous à l'exemple 2 de la commande **Try**, page 132.

colAugment()

Catalogue >

colAugment(*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 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()

Catalogue >

colDim(*Matrice***)** ⇒ *expression*

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

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

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

colNorm()

Catalogue >

colNorm(*Matrice***)** ⇒ *expression*

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

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

$\begin{bmatrix}1 & -2 & 3\\4 & 5 & -6\end{bmatrix} \rightarrow mat$	$\begin{bmatrix}1 & -2 & 3\\4 & 5 & -6\end{bmatrix}$
$\text{colNorm}(mat)$	9

comDenom(*Expr1*[**,***Var*]**)** ⇒ *expression*
comDenom(*Liste1*[**,***Var*]**)** ⇒ *liste*
comDenom(*Matrice1*[**,***Var*]**)** ⇒ *matrice*

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

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

comDenom(*Expr1***,***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.

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

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

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.

$$\text{Define } comden(exprn)=\text{comDenom}(exprn,abc) \qquad Done$$

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

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()**.

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

completeSquare()

Catalogue >

completeSquare(*ExprOuÉqn*, *Var***)** ⇒ *expression ou équation*

completeSquare(*ExprOuÉqn*, *Var*^*Puissance***)** ⇒ *expression ou équation*

completeSquare(*ExprOuÉqn*, *Var1, Var2 [,...]***)** ⇒ *expression ou équation*

completeSquare(*ExprOuÉqn*, Var1, Var2 [,...]**)** ⇒ *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 $a \cdot (x-h)^2=k$.

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* [,…]).

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

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

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

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

$(x+2)^2+(y+3)^2=10$

$\text{completeSquare}(3 \cdot x^2+2 \cdot y+7 \cdot y^2+4 \cdot x=3,\{x,y\})$

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

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

conj()

Catalogue >

conj(*Expr1***)** ⇒ *expression*
conj(*Liste1***)** ⇒ *liste*
conj(*Matrice1***)** ⇒ *matrice*

Donne le conjugué de l'argument.

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

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

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

$\text{conj}(z)$ z

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

constructMat()

Catalogue >

constructMat(*Expr***,***Var1***,***Var2***,***nbreLignes***,***nbreColonnes***)** ⇒ *matrice*

Donne une matrice basée sur les arguments.

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*.

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

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

CopyVar

Catalogue >

CopyVar *Var1*, *Var2*
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.

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.

Define $a(x)=\frac{1}{x}$ — *Done*

Define $b(x)=x^2$ — *Done*

CopyVar *a*,*c*: $c(4)$ — $\frac{1}{4}$

CopyVar *b*,*c*: $c(4)$ — 16

aa.a:=45 — 45

aa.b:=6.78 — 6.78

aa.c:=8.9 — 8.9

getVarInfo() —

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

CopyVar *aa*.,*bb*. — *Done*

getVarInfo() —

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

corrMat()

Catalogue >

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

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

▸cos

Catalogue >

Expr **▸cos**

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(...) 1−cos(...)^2
de sorte que les puissances de cos(...) restantes ont des exposants dans (0, 2). Le résultat ne contient donc pas sin(...) si et seulement si sin(...) 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*.

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

cos()

Touche trig

cos(*Expr1***)** ⇒ *expression*
cos(*Liste1***)** ⇒ *liste*

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

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

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 degrés :

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

$\cos(45)$ → $\frac{\sqrt{2}}{2}$

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

En mode Angle en grades :

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

En mode Angle en radians :

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

$\cos(45°)$ → $\frac{\sqrt{2}}{2}$

cos(*matriceCarrée1***)** ⇒ *matriceCarrée*

Calcule le cosinus de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du cosinus de chaque élément.

Si une fonction scalaire f(A) opère sur *matriceCarrée1* (A), le résultat est calculé par l'algorithme suivant :

Calcul des valeurs propres (λi) et des vecteurs propres (Vi) de A.

matriceCarrée1 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 :

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

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

$\cos^{-1}()$

Touche trig

$\cos^{-1}$(*Expr1*) ⇒ *expression*
$\cos^{-1}$(*Liste1*) ⇒ *liste*

$\cos^{-1}$(*Expr1*) donne l'arc cosinus de *Expr1* et l'affiche sous forme d'expression.

$\cos^{-1}$(*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.

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()**.

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

En mode Angle en degrés :

$\cos^{-1}(1)$ → 0

En mode Angle en grades :

$\cos^{-1}(0)$ → 100

En mode Angle en radians :

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

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

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

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

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

cosh()

Catalogue >

cosh(*Expr1*) ⇒ *expression*
cosh(*Liste1*) ⇒ *liste*

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*

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()**.

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

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

En mode Angle en radians :

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

$\cosh^{-1}()$

Catalogue >

$\cosh^{-1}$(*Expr1*) ⇒ *expression*
$\cosh^{-1}$(*List1*) ⇒ *liste*

$\cosh^{-1}$(*Expr1*) donne l'argument cosinus hyperbolique de l'argument et l'affiche sous forme d'expression.

$^{-1}$cosh^{-1}(*Liste1*) donne la liste des arguments cosinus hyperboliques de chaque élément de *Liste1*.

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

$\cosh^{-1}(1)$ → 0

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

cosh⁻¹()

Catalogue >

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 :

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

$$\begin{bmatrix} 2.52503+1.73485\cdot i & -0.009241-1.4908\text{6} \\ 0.486969-0.725533\cdot i & 1.66262+0.623491 \\ -0.322354-2.08316\cdot i & 1.26707+1.79018 \end{bmatrix}$$

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

cot()

Touche trig

cot(*Expr1***)** ⇒ *expression*
cot(*Liste1***)** ⇒ *liste*

Affiche la cotangente de *Expr1* ou retourne la liste des cotangentes des éléments de *Liste1*.

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 degrés :

$\cot(45)$ → 1

En mode Angle en grades :

$\cot(50)$ → 1

En mode Angle en radians :

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

cot⁻¹()

Touche trig

cot⁻¹(*Expr1***)** ⇒ *expression*
cot⁻¹(*Liste1***)** ⇒ *liste*

Donne l'arc cotangente de *Expr1* ou affiche une liste comportant les arcs cotangentes de chaque élément 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 `arccot(...)`.

En mode Angle en degrés :

$\cot^{-1}(1)$ → 45

En mode Angle en grades :

$\cot^{-1}(1)$ → 50

En mode Angle en radians :

$\cot^{-1}(1)$ → $\frac{\pi}{4}$

coth()

Catalogue >

coth(*Expr1***)** ⇒ *expression*
coth(*Liste1***)** ⇒ *liste*

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

$\coth(1.2)$ → 1.19954

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

coth⁻¹()

Catalogue >

coth⁻¹(*Expr1***)** ⇒ *expression*
coth⁻¹(*Liste1***)** ⇒ *liste*

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(...)**.

$\coth^{-1}(3.5)$ → 0.293893

$\coth^{-1}(\{-2,2.1,6\})$ → $\left\{\frac{-\ln(3)}{2}, 0.518046, \frac{\ln\left(\frac{7}{5}\right)}{2}\right\}$

count()

Catalogue >

count(*Valeur1ouListe1* [,*Valeur2ouListe2*[,...]]**)** ⇒ *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 165.

$\text{count}(2,4,6)$ → 3

$\text{count}(\{2,4,6\})$ → 3

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

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

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 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 165.

Remarque : voir également **sumIf()**, page 125 et **frequency()**, page 52.

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

Compte le nombre d'éléments égaux à 3.

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

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

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

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

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

Compte 1 et 3.

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

Compte 3, 5 et 7.

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

Compte 1, 3, 7 et 9.

cPolyRoots()

Catalogue >

cPolyRoots(*Poly*,*Var***)** ⇒ *liste*
cPolyRoots(*ListeCoeff***)** ⇒ *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 94.

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

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

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

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

crossP()

Catalogue >

crossP(*Liste1***,** *Liste2***)** ⇒ *liste*

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

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

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

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

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.

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

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

csc()

Touche trig

csc(*Expr1***)** ⇒ *expression*
csc(*Liste1***)** ⇒ *liste*

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

En mode Angle en degrés :

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

En mode Angle en grades :

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

En mode Angle en radians :

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

csc⁻¹()

Touche trig

csc⁻¹(*Expr1***)** ⇒ *expression*
csc⁻¹(*Liste1***)** ⇒ *liste*

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

En mode Angle en degrés :

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

En mode Angle en grades :

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

En mode Angle en radians :

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

csch()

Catalogue >

csch(*Expr1***)** ⇒ *expression*
csch(*Liste1***)** ⇒ *liste*

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

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

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

csch⁻¹()

Catalogue >

csch⁻¹(*Expr1***)** ⇒ *expression*
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*.

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

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

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

cSolve()

Catalogue >

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

Bien que toutes les variables non affectées dont le nom ne se termine pas par (_) soient considérées comme réelles, **cSolve()** permet de résoudre des systèmes d'équations polynomiales en utilisant des solutions complexes.

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()**.

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

$\operatorname{solve}(x^3=-1,x)$ $x=-1$

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

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

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()**.

Remarque : si *Équation* n'est pas polynomiale avec les fonctions comme **abs()**, **angle()**, **conj()**, **real()** ou **imag()**, ajoutez un caractère de soulignement (en appuyant sur ctrl ␣) à la fin de *Var*. Par défaut, les variables sont considérées comme réelles.

Si vous utilisez *var_*, la variable est considérée comme complexe.

Vous pouvez également utiliser *var_* pour toutes les autres variables de *Équation* pouvant avoir des valeurs non réelles. Sinon, vous risquez d'obtenir des solutions inattendues.

En mode Afficher chiffres, Fixe 2 :

$\text{exact}(\text{cSolve}(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3=0,x))$
$x\cdot(x^4+4\cdot x^3+5\cdot x^2-6)=3$

$\text{cSolve}(Ans,x)$
$x=-1.11+1.07\cdot i \text{ or } x=-1.11-1.07\cdot i \text{ or } x=-2.$

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

z est considéré comme réel :

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

z_ est considéré comme complexe :

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

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.

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.

Remarque : les exemples suivants utilisent un caractère de soulignement (obtenu en appuyant sur ctrl ␣) pour que toutes les variables soient considérées comme complexes.

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

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

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

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

cSolve() **Catalogue >**

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 *c*k, 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 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 *VarOuInit*.

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.

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

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

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

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

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

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

CubicReg **Catalogue >**

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

Effectue l'ajustement polynomial de degré 3 $y = a \cdot x^3+b \cdot x^2+c \cdot x+d$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 122.)

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 165.

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

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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

cumulativeSum()

Catalogue >

cumulativeSum(*Liste1***)** ⇒ *liste*

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 élement vide dans la liste ou la matrice résultante. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 165

cumulativeSum({1,2,3,4}) → {1,3,6,10}

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

cumulativeSum(*m1*)

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

Cycle

Catalogue >

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

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

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

g() → 5000

▸Cylind

Catalogue >

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,∠θ, z].

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

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

cZeros(*Expr***,** *Var***)** ⇒ *liste*

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

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

Remarque : si *Expr* n'est pas polynomiale par rapport aux fonctions comme **abs()**, **angle()**, **conj()**, **real()** ou **imag()**, vous pouvez utiliser un caractère de soulignement (obtenu en appuyant sur [ctrl] [␣]) à la fin du nom de *Var*. Par défaut, les variables sont considérées comme réelles. Si vous utilisez *var_*, la variable est considérée comme complexe.

Vous pouvez également utiliser *var_* pour les autres variables de *Expr* pouvant avoir des valeurs non réelles. Sinon, vous risquez d'obtenir des solutions inattendues.

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.

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*].

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.

En mode Afficher chiffres, Fixe 3 :

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

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

z est considéré comme réel :

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

z_ est considéré comme complexe :

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

Remarque : les exemples suivants utilisent un _ (obtenu en appuyant sur [ctrl] [␣]) pour que toutes les variables soient considérées comme complexes.

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

Extraction ligne 2 :

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

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

cZeros() **Catalogue >**

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.

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*.

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

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

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.

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

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

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.

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

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

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.

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

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

D

dbd() **Catalogue >**

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

▸DD

Catalogue >

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 :

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

En mode Angle en grades :

$1\blacktriangleright DD$	$\frac{9}{10}°$

En mode Angle en radians :

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

▸Decimal

Catalogue >

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.

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

Define

Catalogue >

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

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

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*).

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

Define

Catalogue >

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

Remarque : voir aussi **Define LibPriv**, page 35 et **Define LibPub**, page 36.

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

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

Define LibPriv

Catalogue >

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 34 et **Define LibPub**, page 36.

Define LibPub

Catalogue >

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 34 et **Define LibPriv**, page 35.

deltaList()

Voir **ΔList()**, page 69.

deltaTmpCnv()

Voir **ΔtmpCnv()**, page 131.

DelVar

Catalogue >

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

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

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 137.

$2 \rightarrow a$ → 2

$(a+2)^2$ → 16

DelVar a → *Done*

$(a+2)^2$ → $(a+2)^2$

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.

$aa.a := 45$ → 45

$aa.b := 5.67$ → 5.67

$aa.c := 78.9$ → 78.9

getVarInfo() →

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

DelVar *aa.* → *Done*

getVarInfo() → "NONE"

delVoid()

Catalogue >

delVoid(*Liste1***)** ⇒ *liste*

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 165.

delVoid({1,void,3}) → {1,3}

derivative()

Voir *d*(), page 154.

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

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 :

- Utilisez uniquement le symbole « prime » (obtenu en appuyant sur ?1▸) 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***)**
⇒ *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***)** ⇒ *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 :

VarDép (*ValeurIndépendanteInitiale*) = *ValeurInitialeDérivée1*

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

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

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

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

DelVar *temp* *Done*

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

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

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

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

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

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

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

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

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

DelVar *ode*,*soln* *Done*

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

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

$\text{solve}\left(Ans,y\right) \qquad y=\frac{2^{\frac{2}{3}}\cdot\left(3\cdot t\right)^{\frac{4}{3}}}{4} \text{ and } t\geq 0$

deSolve()

Catalogue >

deSolve(*ode2ndOrdre* **and** *condBorne1* **and** *condBorne2***,** *Var***,** *VarDép***)** ⇒ *une solution particulière*

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

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

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

det()

Catalogue >

det(*matriceCarrée***[,** *Tolérance***])** ⇒ *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.

- Si vous utilisez ctrl enter 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 :

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

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

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

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

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

$$\det(mat1) \qquad 0$$

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

diag()

Catalogue >

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

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

$$\text{diag}\left(\begin{bmatrix} 2 & 4 & 6 \end{bmatrix}\right) \qquad \begin{bmatrix} 2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

diag(*matriceCarrée***)** ⇒ *matriceLigne*

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

matriceCarrée doit être une matrice carrée.

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

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

dim()

Catalogue >

dim(*Liste***)** ⇒ *entier*

Donne le nombre d'éléments de *Liste*.

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

dim(Matrice) ⇒ *liste*

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

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

dim(*Chaîne***)** ⇒ *entier*

Donne le nombre de caractères contenus dans *Chaîne*.

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

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

Disp

Catalogue >

Disp [*exprOuChaîne1*] [**,** *exprOuChaîne2*] ...

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define chars(start,end)=Prgm
                        For i,start,end
                        Disp i," ",char(i)
                        EndFor
                        EndPrgm
                                        Done

chars(240,243)
                                        240 ð
                                        241 ñ
                                        242 ò
                                        243 ó
                                        Done
```

▶DMS

Catalogue >

Expr **▶DMS**
Liste **▶DMS**
Matrice **▶DMS**

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 160 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.

En mode Angle en degrés :

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

domain()

Catalogue >

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 **∫()** est une fonction non autorisée.

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

```
domain(x^2,x)                   -∞<x<∞
domain((x+1)/(x^2+2·x),x)       x≠-2 and x≠0
domain((√x)^2,x)                0≤x<∞
domain(1/(x+y),y)               y≠-x
```

dominantTerm()

Catalogue >

dominantTerm(*Expr1*, *Var* [, *Point*]**)** ⇒ *expression*
dominantTerm(*Expr1*, *Var* [, *Point*]**) | *Var*>*Point**
⇒ *expression*
dominantTerm(*Expr1*, *Var* [, *Point*]**) | *Var*<*Point*
⇒ *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 −∞, 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 = ∞ ou −∞.

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 $(-1)^{\text{floor}(\ldots\text{angle}(\ldots)\ldots)}$ 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* → *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 111.

$\text{dominantTerm}(\tan(\sin(x))-\sin(\tan(x)),x)$ → $\frac{x^7}{30}$

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

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

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

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

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

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

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

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

dotP()

Catalogue >

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.

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

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

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

E

e^()

Touche e^x

e^(*Expr1***)** ⇒ *expression*

Donne e élevé à la puissance de *Expr1*.

Remarque : voir aussi Modèle e **Exposant**, page 2.

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^1 → e

$e^{1.}$ → 2.71828

e^{3^2} → e^9

e^(*Liste1***)** ⇒ *liste*

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

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

e^(*matriceCarrée1***)** ⇒ *matriceCarrée*

Donne l'exponentielle de *matriceCarrée1*. 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()**.

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

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

eff()

Catalogue >

eff(*tauxNominal*, *CpY***)** ⇒ *valeur*

Fonction financière permettant de convertir un taux d'intérêt nominal *tauxNominal* en un taux annuel effectif, *CpY* étant le nombre de périodes de calcul par an.

tauxNominal doit être un nombre réel et *CpY* doit être un nombre réel > 0.

Remarque : voir également **nom()**, page 84.

eff(5.75,12) → 5.90398

eigVc()

Catalogue >

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, …, x_n], alors :

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

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 :

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

eigVc(*m1*) →

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

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

eigVl()

Catalogue >

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 :

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

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

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

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

Else

Voir If, page 58.

ElseIf

Catalogue >

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

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

                                    Done
```

EndFor

Voir For, page 50.

EndFunc

Voir Func, page 53.

EndIf

Voir If, page 58.

EndLoop

Voir Loop, page 75.

EndPrgm

Voir Prgm, page 95.

EndTry

Voir Try, page 132.

euler()

Catalogue >

euler(*Expr*, *Var*, *VarDép*, {*Var0*, *MaxVar*}, *Var0Dép*, *IncVar* [, *IncEuler*]**)** ⇒ *matrice*
euler(*SystèmeExpr*, *Var*, *ListeVarDép*, {*Var0*, **Max**/*Var*}, *ListeVar0Dép*, *IncVar* [, *IncEuler*]**)** ⇒ *matrice*
euler(*ListeExpr*, *Var*, *ListeVarDép*, {*Var0*, *MaxVar*}, *ListeVar0Dép*, *IncVar* [, *IncEuler*]**)** ⇒ *matrice*

Utilise la méthode d'Euler pour résoudre le système.

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

avec *VarDép*(*Var0*)=*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.

{*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·*IncVar* pour tout i=0,1,2,… de sorte que *Var0*+i·*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'incréments dans la méthode d'Euler entre deux valeurs de sortie. La taille d'incrément courante utilisée par la méthode d'Euler est *IncVar*/*IncEuler*.

Équation différentielle :
y'=0.001*y*(100-y) et y(0)=10

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

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

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

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

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

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

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

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).

$\text{exact}(0.25) \quad \frac{1}{4}$

$\text{exact}(0.333333) \quad \frac{333333}{1000000}$

$\text{exact}(0.333333,0.001) \quad \frac{1}{3}$

$\text{exact}(3.5\cdot x+y) \quad \frac{7\cdot x}{2}+y$

$\text{exact}(\{0.2,0.33,4.125\}) \quad \left\{\frac{1}{5},\frac{33}{100},\frac{33}{8}\right\}$

Exit

Catalogue >

Exit

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

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

Liste des fonctions :

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

▸exp

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**.

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

$2\cdot\sinh(x)\blacktriangleright\exp$ → $e^{-x}\cdot\left(e^{2\cdot x}-1\right)$

exp()

Touche e^x

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.

e^{1} → e

$e^{1.}$ → 2.71828

e^{3^2} → e^{9}

exp(*Liste1***)** ⇒ *liste*

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

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

exp(*matriceCarrée1***)** ⇒ *matriceCarrée*

Donne l'exponentielle de *matriceCarrée1*. 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()**.

matriceCarrée1 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(...)`.

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

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

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.

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.

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

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

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

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

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.

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

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

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

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

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

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.

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

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

expand()

Catalogue >

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*.

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

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

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

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

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

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

expr()

Catalogue >

expr(*Chaîne***)** ⇒ *expression*

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

$\text{expr}(\text{"1+2+x\^{}2+x"})$ → x^2+x+3

$\text{expr}(\text{"expand((1+x)\^{}2)"})$ → $x^2+2\cdot x+1$

"Define cube(x)=x^3" → *funcstr* → "Define cube(x)=x^3"

$\text{expr}(funcstr)$ → *Done*

$cube(2)$ → 8

ExpReg

Catalogue >

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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a \cdot (b)^x$
stat.a, stat.b	Coefficients d'ajustement

Variable de sortie	Description
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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

F

factor()

Catalogue >

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.

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

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

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

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.

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

$\text{factor}(x^5+4\cdot x^4+5\cdot x^3-6\cdot x-3,x)$ → $(x-0.964673)\cdot(x+0.611649)\cdot(x+2.12543)\cdot(x^$

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.

factor()

Catalogue >

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.

Pour arrêter un calcul manuellement,

- **Windows® :** maintenez enfoncé la touche **F12** et appuyez plusieurs fois sur **Entrée**.
- **Macintosh® :** maintenez enfoncé la touche **F6** et appuyez plusieurs fois sur **Entrée**.
- **Unité :** maintenez enfoncé la touche [on] et appuyez plusieurs fois sur [enter].

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.

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

FCdf()

Catalog >

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($X \le$ *upBound*), utilisez *lowBound* = 0.

Fill

Catalogue >

Fill *Expr, VarMatrice* ⇒ *matrice*

Remplace chaque élément de la variable *VarMatrice* par *Expr*.

VarMatrice doit avoir été définie.

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

Fill *Expr, VarListe* ⇒ *liste*

Remplace chaque élément de la variable *VarListe* par *Expr*.

VarListe doit avoir été définie.

{1,2,3,4,5} → *alist*	{1,2,3,4,5}
Fill 1.01,*alist*	*Done*
alist	{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 122.)

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 a un élément vide dans l'ensemble des listes résultantes. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 165.

Variable de sortie	Description
stat.MinX	Minimum des valeurs de x
stat.Q_1X	1er quartile de x
stat.MedianX	Médiane de x
stat.Q_3X	3ème quartile de x
stat.MaxX	Maximum des valeurs de x

floor()

Catalogue >

floor(*Expr1***)** ⇒ *entier*

Donne le plus grand entier ≤ à l'argument (partie entière). Cette fonction est comparable à **int()**.

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

floor(*Liste1***)** ⇒ *liste*
floor(*Matrice1***)** ⇒ *matrice*

Donne la liste ou la matrice de la partie entière de chaque élément.

Remarque : voi aussi **ceiling()** et **int()**.

$\text{floor}(-2.14)$ → $-3.$

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

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

fMax()

Catalogue >

fMax(*Expr, Var***)** ⇒ *Expression booléenne*
fMax(*Expr***,** *Var***,***LimitInf***)**
fMax(*Expr***,** *Var***,***LimitInf***,***LimitSup***)**
fMax(*Expr***,** *Var***) |** *LimitInf*≤*Var*≤*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.

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

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

fMax()

Catalogue >

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()**.

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

fMin()

Catalogue >

fMin(*Expr, Var***)** $\Rightarrow$ *Expression booléenne*
fMin(*Expr***,** *Var***,***LimitInf***)**
fMin(*Expr***,** *Var***,***LimitInf***,***LimitSup***)**
fMin(*Expr***,** *Var***) |** *LimitInf*≤*Var*≤*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.

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()**.

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

$\text{fMin}(0.5\cdot x^3-x-2,x)|x\ge 1$ $\quad x=1$

For

Catalogue >

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define g()=Func                    Done
    Local tempsum,step,i
    0→tempsum
    1→step
    For i,1,100,step
    tempsum+i→tempsum
    EndFor
    EndFunc
g()                                5050
```

format()

Catalogue >

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()

Catalogue >

fPart(*Expr1***)** ⇒ *expression*
fPart(*Liste1***)** ⇒ *liste*
fPart(*Matrice1***)** ⇒ *matrice*

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.

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

FPdf()

Catalogue >

FPdf(*ValX***,***dfNumér***,***dfDénom***)** ⇒ *nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste
FPdf(*ValX***,***dfNumér***,***dfDénom***)** ⇒ *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()

Catalogue >

freqTable▸list(*Liste1*,*listeEntFréq*) ⇒ *liste*

Donne la liste comprenant les éléments de *Liste1* développés en fonction des fréquences contenues dans *listEntFréq*. 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.

listEntFréq 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*.

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 165.

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

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

frequency()

Catalogue >

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 {**?**≤b(1), b(1)<**?**≤b(2),…,b(n-1)<**?**≤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, **?**≤b(1)), countIf(liste, b(1)<**?**≤b(2)), …, countIf(liste, b(n-1)<**?**≤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 165.

Dans l'application Tableur & listes, vous pouvez utiliser une plage de cellules à la place des deux arguments.

Remarque : voir également **countIf()**, page 26.

datalist:={1,2,**e**,3,π,4,5,6,"hello",7}
{1,2,2.71828,3,3.14159,4,5,6,"hello",7}

frequency(*datalist*,{2.5,4.5}) {2,4,3}

Explication du résultat :
2 éléments de *Datalist* sont ≤2,5
4 éléments de *Datalist* sont >2,5 et ≤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

Catalogue >

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 122.)

Pour H_a : σ1 > σ2, définissez *Hypoth*>0
Pour H_a : σ1 ≠ σ2 (par défaut), définissez *Hypoth* =0
Pour H_a : σ1 < σ2, définissez *Hypoth*<0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 165.

Variable de sortie	Description
stat.F	Statistique F 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 = n1-1
stat.dfDenom	Dénominateur degrés de liberté = n2-1.
stat.sx1, stat.sx2	Écarts types de population d'échantillon des séquences de données dans *Liste 1* et *Liste 2*.
stat.x1_bar stat.x2_bar	Moyenne de population d'échantillon des séquences de données dans *Liste 1* et *Liste 2*.
stat.n1, stat.n2	Taille des échantillons

Func

Catalogue >

Func
Bloc
EndFunc

Modèle de création d'une fonction définie par l'utilisateur.

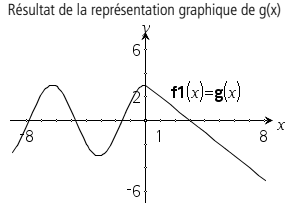
Bloc peut correspondre à une instruction unique ou à une série d'instructions séparées par le caractère ":" ou à une série d'instructions réparties sur plusieurs lignes. La fonction peut utiliser l'instruction **Return** pour donner un résultat spécifique.

Remarque pour la saisie des données de l'exemple : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

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()

Catalogue >

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.

$\gcd(18,33)$ → 3

gcd(*Liste1, Liste2***)** ⇒ *liste*

Donne la liste des plus grands communs diviseurs des éléments correspondants de *Liste1* et *Liste2*.

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

gcd(*Matrice1, Matrice2***)** ⇒ *matrice*

Donne la matrice des plus grands communs diviseurs des éléments correspondants de *Matrice1* et *Matrice2*.

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

geomCdf()

Catalogue >

geomCdf(*p*,*lowBound*,*upBound*) ⇒ *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≤X≤*upBound*) ⇒ *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 ≤ *upBound*), définissez *lowBound* = 1.

geomPdf()

Catalogue >

geomPdf(*p*,*ValX*) ⇒ *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.

getDenom()

Catalogue >

getDenom(*Expr1***)** ⇒ *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

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

getLangInfo()

Catalogue >

getLangInfo() ⇒ *chaîne*

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.

getLangInfo() → "en"

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()

Catalogue >

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 71 et **unLock**, page 137.

$a:=65$	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()

Catalogue >

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.

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 112.

getMode(0)	$\{1,1,2,1,3,1,4,1,5,1,6,1,7,1,8,1\}$
getMode(1)	1
getMode(8)	1

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
Système d'unités	**8**	**1**=SI, **2**=Ang/US

getNum()

Catalogue >

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
$\text{getNum}\left(\frac{1}{x}+\frac{1}{y}\right)$	$x+y$

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".

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

getVarInfo()

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).

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.

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

getVarInfo()

Catalogue >

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.

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

Goto

Catalogue >

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

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

Catalogue >

Expr1 ▸ **Grad** ⇒ *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)▸Grad	$(1.66667)^g$

En mode Angle en radians :

(1.5)▸Grad	$(95.493)^g$

identity()

Catalogue >

identity(*Entier***)** ⇒ *matrice*

Donne la matrice identité (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

Catalogue >

If *Expr booléenne*
Instruction

If *Expr booléenne* **Then**
Bloc
EndIf

Si *Expr booléenne* passe le test de condition, exécute l'instruction *Instruction* ou le bloc d'instructions *Bloc* avant de poursuivre l'exécution de la fonction.

Si *Expr booléenne* ne passe pas le test de condition, poursuit l'exécution en ignorant l'instruction ou le bloc d'instructions.

Bloc peut correspondre à une ou plusieurs instructions, séparées par un « : ».

Remarque pour la saisie des données de l'exemple : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

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

If *Expr booléenne* **Then**
Bloc1
Else
Bloc2
EndIf

Si *Expr booléenne* passe le test de condition, exécute *Bloc1* et ignore *Bloc2*.

Si *Expr booléenne* ne passe pas le texte de condition, ignore *Bloc1*, mais exécute *Bloc2*.

Bloc1 et *Bloc2* peuvent correspondre à une seule instruction.

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

If

Catalogue >

If *Expr booléenne1* **Then**
Bloc1
ElseIf *Expr booléenne2* **Then**
Bloc2
⋮
ElseIf *Expr booléenneN* **Then**
BlocN
EndIf

Permet de traiter les conditions multiples. Si *Expr booléenne1* passe le test de condition, exécute *Bloc1*. Si *Expr booléenne1* ne passe pas le test de condition, calcule *Expr booléenne2*, et ainsi de suite.

```
Define g(x)=Func
              If x<-5 Then
              Return 5
              ElseIf x>-5 and x<0 Then
              Return -x
              ElseIf x≥0 and x≠10 Then
              Return x
              ElseIf x=10 Then
              Return 3
              EndIf
              EndFunc
                                        Done
g(-4)                                    4
g(10)                                    3
```

ifFn()

Catalogue >

ifFn(*exprBooléenne***,***Valeur_si_Vrai* **[,***Valeur_si_Faux* **[,***Valeur_si_Inconnu***]]) ⇒** *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 vrai, l'élément correspondant de *Valeur_si_Vrai* s'affiche.
- Si un élément de *exprBooléenne* est 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).

```
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.

```
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.

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

Valeur_si_Faux n'est pas spécifié. Undef est utilisé.

```
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()

Catalogue >

imag(*Expr1***) ⇒** *expression*

Donne la partie imaginaire de l'argument.

Remarque : toutes les variables non affectées sont considérées comme réelles. Voir aussi **real()**, page 101

```
imag(1+2·i)                              2
imag(z)                                  0
imag(x+i·y)                              y
```

imag()

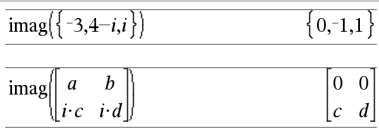
Catalogue >

imag(*Liste1***)** ⇒ *liste*

Donne la liste des parties imaginaires des éléments.

imag(*Matrice1***)** ⇒ *matrice*

Donne la matrice des parties imaginaires des éléments.



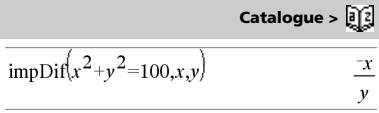
impDif()

Catalogue >

impDif(*Équation***,** *Var***,** *VarDép***[,***Ordre***])** ⇒ *expression*

où la valeur par défaut de l'argument *Ordre* est 1.

Calcule la dérivée implicite d'une équation dans laquelle une variable est définie implicitement par rapport à une autre.



Indirection

Voir #(), page 158.

inString()

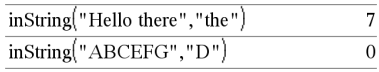
Catalogue >

inString(*chaîneSrce***,** *sousChaîne***[,** *Début***])** ⇒ *entier*

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, la recherche commence à partir du premier caractère de *chaîneSrce*.

Si *chaîneSrce* ne contient pas *sousChaîne* ou si *Début* est > à la longueur de *ChaîneSrce*, on obtient zéro.



int()

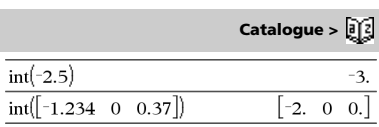
Catalogue >

int(*Expr***)** ⇒ *entier*
int(*Liste1***)** ⇒ *liste*
int(*Matrice1***)** ⇒ *matrice*

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.



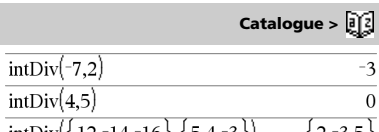
intDiv()

Catalogue >

intDiv(*Nombre1***,** *Nombre2***)** ⇒ *entier*
intDiv(*Liste1***,** *Liste2***)** ⇒ *liste*
intDiv(*Matrice1***,** *Matrice2***)** ⇒ *matrice*

Donne le quotient dans la division euclidienne de (*Nombre1* ÷ *Nombre2*).

Dans le cas d'une liste ou d'une matrice, donne le quotient de (argument 1 ÷ argument 2) pour chaque paire d'éléments.



integral

Voir ∫(), page 154.

interpolate()

Catalogue >

interpolate(*Valeurx***, *Listex*, *Listey*, *ListePrincy*)** ⇒ *liste*

Cette fonction effectue l'opération suivante :

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.

Équation différentielle :
$y'=-3\cdot y+6\cdot t+5$ et $y(0)=5$

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

$\begin{bmatrix}0. & 1. & 2. & 3. & 4. \\ 5. & 3.19499 & 5.00394 & 6.99957 & 9.00593 & 10\end{bmatrix}$

Pour afficher le résultat en 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 listevaleursx :

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

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

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

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

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

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

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

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

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

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

invχ²()

Catalogue >

invχ²(*Zone*,*df***)**
invChi2(*Zone*,*df***)**

Calcule l'inverse de la fonction de répartition de la loi χ^2 (Khi^2) de degré de liberté *df* en un point donné (*Zone*).

invF()

Catalogue >

invF(*Zone*,*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 (*Zone*).

invNorm()

Catalogue >

invNorm(*Zone*[,μ[,σ]]**)**

Calcule l'inverse de la fonction de répartition de la loi normale de paramètres mu et sigma (*m* et σ) en un point donné (*Zone*).

invt()

Catalogue >

invt(*Zone*,*df***)**

Calcule l'inverse de la fonction de répartition de la loi student-t de degré de liberté *df* en un point donné (*Zone*).

iPart()

Catalogue >

iPart(*Nombre***)** ⇒ *entier*
iPart(*Liste1***)** ⇒ *liste*
iPart(*Matrice1***)** ⇒ *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({3/2,−2.3,7.003})	{1,−2.,7.}

irr()

Catalogue >

irr(*MT0,ListeMT* [,*FréqMT*]**)** ⇒ *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 79.

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()

Catalogue >

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 inférieur à ≤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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

isPrime(5)	true
isPrime(6)	false

Fonction permettant de trouver le nombre premier suivant un nombre spécifié :

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

isVoid()

Catalogue >

isVoid(*Var***)** ⇒ *expression booléenne constante*
isVoid(*Expr***)** ⇒ *expression booléenne constante*
isVoid(*Liste***)** ⇒ *liste des 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 la page 165.

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

L

Lbl

Catalogue >

Lbl *nomÉtiquette*

Définit une étiquette en lui attribuant le nom *nomÉtiquette* dans une fonction.

Vous pouvez utiliser l'instruction **Goto** *nomÉtiquette* pour transférer le contrôle du programme à l'instruction suivant immédiatement l'étiquette.

nomÉtiquette doit être conforme aux mêmes règles de dénomination que celles applicables aux noms de variables.

Remarque pour la saisie des données de l'exemple : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define g()=Func                    Done
    Local temp,i
    0→temp
    1→i
    Lbl top
    temp+i→temp
    If i<10 Then
    i+1→i
    Goto top
    EndIf
    Return temp
    EndFunc
g()                                55
```

lcm()

Catalogue >

lcm(*Nombre1***,** *Nombre2***)** ⇒ *expression*
lcm(*Liste1***,** *Liste2***)** ⇒ *liste*
lcm(*Matrice1***,** *Matrice2***)** ⇒ *matrice*

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.

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

left()

Catalogue >

left(*chaîneSrce***[,** *Nomb***])** ⇒ *chaîne*

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("Hello",2)                  "He"
left({1,3,-2,4},3)               {1,3,-2}
left(x<3)                        x
```

libShortcut()

Catalogue >

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 22.
Pour supprimer un groupe de variables, reportez-vous à **DelVar**, page 36.

Cet exemple utilise un classeur de bibliothèque enregistré et rafraîchi **linalg2** qui contient les objets définis comme *clearmat, gauss1* et *gauss2*.

```
getVarInfo("linalg2")
```

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

```
libShortcut("linalg2","la")
```

{*la.clearmat*,*la.gauss2*}

```
libShortcut("linalg2","la",1)
```

{*la.clearmat*,*la.gauss1*,*la.gauss2*}

limit() ou lim()

Catalogue >

limit(*Expr1***,** *Var***,** *Point* [**,***Direction*]**)** ⇒ *expression*
limit(*Liste1***,** *Var***,** *Point* [**,** *Direction*]**)** ⇒ *liste*
limit(*Matrice1***,** *Var***,** *Point* [**,** *Direction*]**)** ⇒ *matrice*

Donne la limite recherchée.

Remarque : voir aussi **Modèle Limite**, page 5.

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.)

$\lim_{x\to 5}(2\cdot x+3)$ → 13

$\lim_{x\to 0^+}\left(\frac{1}{x}\right)$ → ∞

$\lim_{x\to 0}\left(\frac{\sin(x)}{x}\right)$ → 1

$\lim_{h\to 0}\left(\frac{\sin(x+h)-\sin(x)}{h}\right)$ → $\cos(x)$

$\lim_{n\to\infty}\left(\left(1+\frac{1}{n}\right)^n\right)$ → *e*

Les limites en +∞ et en -∞ 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.

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.

$\lim_{x\to\infty}(a^x)$ → undef

$\lim_{x\to\infty}(a^x)\,|\,a>1$ → ∞

$\lim_{x\to\infty}(a^x)\,|\,a>0 \text{ and } a<1$ → 0

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

Effectue l'ajustement linéaire y = a+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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : a+b • 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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

LinRegMx

Catalogue >

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

Effectue l'ajustement linéaire y = 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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : m · x+b
stat.m, 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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

LinRegtIntervals *X*, *Y*[, *F*[, **0**[, *NivC*]]]

Pente. Calcule un intervalle de confiance de niveau C pour la pente.

LinRegtIntervals *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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : a+b · 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

Variable de sortie	Description
[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 · ValX

LinRegtTest

Catalogue >

LinRegtTest *X*,*Y*[,*Fréq*[,*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 $_H0$:$\beta=0$ (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 : $\beta\neq0$ et $\rho\neq0$ (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 122.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides" , page 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : a + b · 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
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(*SystèmÉqLin*, *Var1*, *Var2*, ...**)** ⇒ *liste*
linSolve(*ÉqLin1* **and** *ÉqLin2* **and** ..., *Var1*, *Var2*, ...**)** ⇒ *liste*
linSolve({*ÉqLin1*, *ÉqLin2*, ...**}**, *Var1*, *Var2*, ...**)** ⇒ *liste*
linSolve(*SystèmÉqLin*, **{***Var1*, *Var2*, ...**})** ⇒ *liste*
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".

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

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

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

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

Δlist()

Catalogue >

Δlist(*Liste1***)** ⇒ *liste*

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.

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

list▸mat()

Catalogue >

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(...)`.

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

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

▸ln

Catalogue >

Expr **▸ln** ⇒ *expression*

Convertit *Expr* en une expression contenant uniquement des logarithmes népériens (ln).

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

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

ln()

Touches ctrl e^x

ln(*Expr1***)** ⇒ *expression*
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.

$\ln(2.)$ → 0.693147

En mode Format complexe Réel :

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

En mode Format complexe Rectangulaire :

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

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, reportezvous 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 :

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

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

LnReg

Catalogue >

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

Effectue l'ajustement logarithmique y = a+b • 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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : a+b • 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)

Variable de sortie	Description
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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

Local

Catalogue >

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 des 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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define rollcount()=Func
                    Local i
                    1→i
                    Loop
                    If randInt(1,6)=randInt(1,6)
                    Goto end
                    i+1→i
                    EndLoop
                    Lbl end
                    Return i
                    EndFunc
                                                Done
rollcount()                                     16
rollcount()                                     3
```

Lock

Catalogue >

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.

Voir **unLock**, page 137 et **getLockInfo()**, page 55.

```
a:=65                                           65
Lock a                                          Done
getLockInfo(a)                                  1
a:=75                          "Error: Variable is locked."
DelVar a                       "Error: Variable is locked."
Unlock a                                        Done
a:=75                                           75
DelVar a                                        Done
```

log()

Touches ctrl 10^x

log(*Expr1*[**,***Expr2*]**)** ⇒ *expression*
log(*Liste1*[**,***Expr2*]**)** ⇒ *liste*

Donne le logarithme de base *Expr2* de l'argument.

Remarque : voir aussi **Modèle Logarithme**, page 2.

Dans le cas d'une liste, donne le logarithme de base *Expr2* des éléments.

Si *Expr2* est omis, la valeur de base 10 par défaut est utilisée.

$\log_{10}(2.)$ → 0.30103

$\log_{4}(2.)$ → 0.5

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

En mode Format complexe Réel :

$\log_{10}(\{-3,1.2,5\})$ → *Non*−real *result*

En mode Format complexe Rectangulaire :

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

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

log(*matriceCarrée1*[**,***Expr*]**)** ⇒ *matriceCarrée*

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.

Si l'argument de base est omis, la valeur de base 10 par défaut est utilisée.

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

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

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

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

▶logbase

Catalogue >

Expr1 **▶logbase(***Expr2***)** ⇒ *expression*

Provoque la simplification de l'expression entrée en une expression utilisant uniquement des logarithmes de base *Expr2*.

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

$\log_{3}(10)-\log_{5}(5)\blacktriangleright\text{logbase}(5)$

$-\left(\log_{5}(3)-\log_{5}(2)-1\right)$ / $\log_{5}(3)$

Logistic

Catalogue >

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

Effectue l'ajustement logistique $y = (c/(1+a \cdot e^{-bx}))$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 122.)

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 165.

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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

LogisticD *X*, *Y* [, [*Itérations*], [*Fréq*] [, *Catégorie*, *Inclure*]]

Effectue l'ajustement logistique y = (c/(1+a • 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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : c/(1+a • 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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

Loop

Catalogue >

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define rollcount()=Func
                   Local i
                   1→i
                   Loop
                   If randInt(1,6)=randInt(1,6)
                   Goto end
                   i+1→i
                   EndLoop
                   Lbl end
                   Return i
                   EndFunc
```

Done

rollcount()	16
rollcount()	3

LU

Catalogue >

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 · *uMatrice* = *pMatrice* · *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 ctrl enter 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 · max(dim(*Matrice*)) · rowNorm(*Matrice*)

L'algorithme de factorisation **LU** utilise la méthode du Pivot partiel avec échanges de lignes.

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

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

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

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

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

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

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

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

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

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

mat▸list()

Catalogue >

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(...)`.

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

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \to m1$ → $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$

$\text{mat}\blacktriangleright\text{list}(m1)$ → $\{1,2,3,4,5,6\}$

max()

Catalogue >

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.

$\max(2.3,1.4)$ → 2.3

$\max(\{1,2\},\{-4,3\})$ → $\{1,3\}$

max(*Liste***)** ⇒ *expression*

Donne l'élément maximal de *liste*.

$\max(\{0,1,-7,1.3,0.5\})$ → 1.3

max(*Matrice1***)** ⇒ *matrice*

Donne un vecteur ligne contenant l'élément maximal de chaque colonne de la matrice *Matrice1*.

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

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 165.

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*.

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

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

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 165.

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*.

median(*Matrice1*[**,** *matriceFréq*]**)** ⇒ *matrice*

Donne un vecteur ligne contenant les médianes des colonnes de *Matrice1*.

Chaque élément de *matriceFréq* totalise le nombre d'occurrences consécutives de l'élément correspondant de *Matrice1*.

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 165.

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

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

MedMed

Catalogue >

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

Calcule la ligne Med-Med y = (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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation de ligne Med-Med : m • 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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

mid()

Catalogue >

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*.

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*]**)** ⇒ *liste*

Donne la liste de *Nbre* de chaînes extraites de la liste *listeChaînesSource*, en commençant par 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({"A","B","C","D"},2,2)              {"B","C"}
```

min()

Catalogue >

min(*Expr1*, *Expr2***)** ⇒ *expression*
min(*Liste1*, *Liste2***)** ⇒ *liste*
min(*Matrice1*, *Matrice2***)** ⇒ *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***)** ⇒ *expression*

Donne l'élément minimal de *Liste*.

min(*Matrice1***)** ⇒ *matrice*

Donne un vecteur ligne contenant l'élément minimal de chaque colonne de la matrice *Matrice1*.

Remarque : voir aussi **fMin()** et **max()**.

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

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

min([ 1  -3  7 ])                       [-4  -3  0.3]
   ([-4   0  0.3])
```

mirr()

Catalogue >

mirr(*tauxFinancement*,*tauxRéinvestissement*,*MT0*,*ListeMT* [,*FréqMT*]**)** ⇒ *expression*

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 62.

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

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

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

mod()

Catalogue >

mod(*Exp1*, *Exp2***)** ⇒ *expression*
mod(*Liste1*, *List2***)** ⇒ *liste*
mod(*Matrice1*, *Matrice2***)** ⇒ *matrice*

Donne le premier argument modulo le deuxième argument, défini par les identités suivantes :

mod(x,0) = x
mod(x,y) = x -ÿ floor(x/y)

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 102

$\text{mod}(7,0)$ 7

$\text{mod}(7,3)$ 1

$\text{mod}(-7,3)$ 2

$\text{mod}(7,-3)$ -2

$\text{mod}(-7,-3)$ -1

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

mRow()

Catalogue >

mRow(*Expr*, *Matrice1*, *Index***)** ⇒ *matrice*

Donne une copie de *Matrice1* obtenue en multipliant chaque élément de la ligne *Index* de *Matrice1* par *Expr*.

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

mRowAdd()

Catalogue >

mRowAdd(*Expr*, *Matrice1*, *Index1*, *Index2***)** ⇒ *matrice*

Donne une copie de *Matrice1* obtenue en remplaçant chaque élément de la ligne *Index2* de *Matrice1* par :

Expr × ligne *Index1* + ligne *Index2*

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

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

MultReg

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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : b0+b1 · x1+b2 · x2+ ...
stat.b0, stat.b1, ...	Coefficients d'ajustement
stat.R^2	Coefficient de détermination multiple
stat.$\hat{\mathbf{y}}$Liste	$\hat{\mathbf{y}}$Liste = b0+b1 · x1+ ...
stat.Resid	Valeurs résiduelles de l'ajustement

MultRegIntervals

Catalogue >

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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : b0+b1 · x1+b2 · x2+ ...
stat.$\hat{\mathbf{y}}$	Prévision d'un point : $\hat{\mathbf{y}}$ = b0 + b1 · xl + ... pour *listeValX*
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
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,...}

Variable de sortie	Description
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 122.)

Pour d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides" , page 165.

Sorties

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : b0+b1 · x1+b2 · x2+ ...
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
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 coefficents
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.$\hat{y}$Liste	$\hat{y}$Liste = b0+b1 · x1+ . . .
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

nand

touches ctrl =

BooleanExpr1 **nand** *BooleanExpr2* renvoie *expression booléenne*
BooleanList1 **nand** *BooleanList2* renvoie *liste booléenne*
BooleanMatrix1 **nand** *BooleanMatrix2* renvoie *matrice booléenne*

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

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 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.

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(*Expr***, 0)** ⇒ **1**

nCr(*Expr***,** *entierNég***)** ⇒ **0**

nCr(*Expr***,** *entierPos***)** ⇒ *Expr* · **(***Expr*−1**)...**
(*Expr*−*entierPos***+1)/** *entierPos***!**

nCr(*Expr***,** *nonEntier***)** ⇒ *expression***!/**
((*Expr*−*nonEntier***)!** · *nonEntier***!)**

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

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

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.

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

nCr(*Matrice1***,** *Matrice2***)** ⇒ *matrice*

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.

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

nDerivative()

Catalogue >

nDerivative(*Expr1***,***Var*=*Valeur*[,*Ordre*]) ⇒ *valeur***
nDerivative(*Expr1***,***Var*[,*Ordre*]) | *Var*=*Valeur* ⇒ *valeur***

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**.

$\text{nDerivative}(|x|,x=1)$ → 1

$\text{nDerivative}(|x|,x)|x=0$ → undef

$\text{nDerivative}(\sqrt{x-1},x)|x=1$ → undef

newList()

Catalogue >

newList(*nbreÉléments***) ⇒ *liste***

Donne une liste de dimension *nbreÉléments*. Tous les éléments sont nuls.

$\text{newList}(4)$ → $\{0,0,0,0\}$

newMat()

Catalogue >

newMat(*nbreLignes***, *nbreColonnes***) ⇒ *matrice***

Donne une matrice nulle de dimensions *nbreLignes*, *nbreColonnes*.

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

nfMax()

Catalogue >

nfMax(*Expr***, *Var***) ⇒ *valeur***
nfMax(*Expr***, *Var*, *LimitInf***) ⇒ *valeur***
nfMax(*Expr***, *Var*, *LimitInf*, *LimitSup***) ⇒ *valeur***
nfMax(*Expr*, *Var***) | *LimitInf*≤*Var*≤*LimitSup* ⇒ *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()**.

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

$\text{nfMax}(0.5\cdot x^3-x-2,x,-5,5)$ → -0.816497

nfMin()

Catalogue >

nfMin(*Expr***, *Var***) ⇒ *valeur***
nfMin(*Expr***, *Var*, *LimitInf***) ⇒ *valeur***
nfMin(*Expr***, *Var*, *LimitInf*, *LimitSup***) ⇒ *valeur***
nfMin(*Expr*, *Var***) | *LimitInf*≤*Var*≤*LimitSup* ⇒ *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()**.

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

$\text{nfMin}(0.5\cdot x^3-x-2,x,-5,5)$ → 0.816497

nInt()

Catalogue >

nInt(*Expr1, Var, Borne1, Borne2***) ⇒ *expression***

Si l'intégrande *Expr1* ne contient pas d'autre variable que *Var* et si *Borne1* et *Borne2* sont des constantes, en +∞ ou en -∞, alors **nInt()** donne le calcul approché de ∫(*Expr1, Var, Borne1, Borne2*). Cette approximation correspond à une moyenne pondérée de certaines valeurs d'échantillon de l'intégrande dans l'intervalle *Borne1*<*Var*<*Borne2*.

$\text{nInt}(e^{-x^2},x,-1,1)$ → 1.49365

nInt()

Catalogue >

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.

$\text{nInt}(\cos(x),x,-\pi,\pi+1.\text{E}-12)$ $-1.04144\text{E}-12$

$\int_{-\pi}^{\pi+10^{-12}} \cos(x)\,dx$ $-\sin\left(\frac{1}{1000000000000}\right)$

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.

$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x\cdot y}}{\sqrt{x^2-y^2}},y,-x,x\right),x,0,1\right)$ 3.30423

Remarque : voir aussi ∫(), page 154.

nom()

Catalogue >

nom(*tauxEffectif*,*CpY***)** ⇒ *valeur*

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 41.

$\text{nom}(5.90398,12)$ 5.75

nor

touches ctrl =

BooleanExpr1 **nor** *BooleanExpr2* renvoie *expression booléenne*
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.

$x\geq 3 \text{ or } x\geq 4$	$x\geq 3$
$x\geq 3 \text{ nor } x\geq 4$	$x<3$

Integer1 **nor** *Integer2* ⇒ *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.

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 or 4	7
3 nor 4	-8
$\{1,2,3\} \text{ or } \{3,2,1\}$	$\{3,2,3\}$
$\{1,2,3\} \text{ nor } \{3,2,1\}$	$\{-4,-3,-4\}$

norm()

Catalogue >

norm(*Matrice***)** ⇒ *expression*
norm(*Vecteur***)** ⇒ *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***)** ⇒ *expression*
normalLine(*Expr1***,***Var***=***Point***)** ⇒ *expression*

Donne la normale à 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 **normalLine(**f1(x),*x*,2**)** retourne « faux».

$\text{normalLine}\left(x^2,x,1\right)$	$\frac{3}{2}-\frac{x}{2}$
$\text{normalLine}\left((x-3)^2-4,x,3\right)$	$x=3$
$\text{normalLine}\left(x^{\frac{1}{3}},x=0\right)$	0
$\text{normalLine}\left(\sqrt{\lvert x\rvert},x=0\right)$	undef

normCdf()

Catalogue >

normCdf(*lowBound***,***upBound***[,*μ*[,*σ*]])** ⇒ *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 ≤ *upBound*), définissez *lowBound* = −∞.

normPdf()

Catalogue >

normPdf(*ValX***[,*μ*[,*σ*]])** ⇒ *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.

not

Catalogue >

not *Expr booléenne1* ⇒ *Expression booléenne*

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

$\text{not}\left(2\geq 3\right)$	true
$\text{not}\left(x<2\right)$	$x\geq 2$
not not *innocent*	*innocent*

not

Catalogue >

not *Entier1* ⇒ *entier*

Donne le complément à 1 d'un entier. En interne, *Entier1* 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 13.

En mode base Hex :

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

not 0h7AC36	0hFFFFFFFFFFF853C9

En mode base Bin :

0b100101▶Base10	37
not 0b100101	
0b111111111111111111111111111111111111▶	
not 0b100101▶Base10	−38

Pour afficher le résultat entier, appuyez sur ▲, puis utilisez les touches ◀ et ▶ 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*) ⇒ *expression*

Pour les expressions *Expr1* et *Expr2* avec *Expr1* ≥ *Expr2* ≥ 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**) ⇒ **1**

nPr(*Expr*, *entierNég*) ⇒ **1/((***Expr***+1) · (***Expr***+2)... (***expression*−*entierNég***))**

nPr(*Expr*, *entierPos*) ⇒ *Expr* · (*Expr*−**1)... (***Expr*−*entierPos***+1)**

nPr(*Expr*, *nonEntier*) ⇒ *Expr***! / (***Expr*−*nonEntier***)!**

$\mathrm{nPr}(z,3)$	$z\cdot(z-2)\cdot(z-1)$
$Ans\|z=5$	60
$\mathrm{nPr}(z,-3)$	$\frac{1}{(z+1)\cdot(z+2)\cdot(z+3)}$
$\mathrm{nPr}(z,c)$	$\frac{z!}{(z-c)!}$
$Ans\cdot\mathrm{nPr}(z-c,-c)$	1

nPr(*Liste1*, *Liste2*) ⇒ *liste*

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.

$\mathrm{nPr}(\{5,4,3\},\{2,4,2\})$	$\{20,24,6\}$

nPr(*Matrice1*, *Matrice2*) ⇒ *matrice*

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.

$\mathrm{nPr}\begin{bmatrix}6&5\\4&3\end{bmatrix},\begin{bmatrix}2&2\\2&2\end{bmatrix}$	$\begin{bmatrix}30&20\\12&6\end{bmatrix}$

npv()

Catalogue >

npv(*tauxIntérêt*,*MTO*,*ListeMT*[,*FréqMT*]**)**

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.

tauxIntérêt est le taux à appliquer pour l'escompte des mouvements de trésorerie (taux de l'argent) sur une période donnée.

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 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.

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

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

$\text{npv}(10,5000,list1,list2)$ 4769.91

nSolve()

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.

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()**.

$\text{nSolve}(x^2+5\cdot x-25=9,x)$ 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"

O

OneVar

Catalogue >

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 122.)

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 165.

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.σx	Écart-type de la population de x
stat.n	Nombre de points de données
stat.MinX	Minimum des valeurs de x
stat.Q_1X	1er quartile de x
stat.MedianX	Médiane de x
stat.Q_3X	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

Catalogue >

BooleanExpr1 **or** *BooleanExpr2* renvoie *expression booléenne*
BooleanList1 **or** *BooleanList2* renvoie *liste booléenne*
BooleanMatrix1 **or** *BooleanMatrix2* renvoie *matrice booléenne*

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

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

Entier1 **or** *Entier2* ⇒ *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 13.

Remarque : voir **xor**.

$x \geq 3 \text{ or } x \geq 4$	$x \geq 3$

Define $g(x)$=Func	*Done*
If $x \leq 0 \text{ or } x \geq 5$	
Goto *end*	
Return $x \cdot 3$	
Lbl *end*	
EndFunc	
$g(3)$	9
$g(0)$	*A function did* not return *a value*

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***)** ⇒ *entier*
ord(*Liste1***)** ⇒ *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*) ⇒ *expression*
P▶Rx(*ListeR*, θ*Liste*) ⇒ *liste*
P▶Rx(*MatriceR*, θ*Matrice*) ⇒ *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 ʳ 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 :

$\mathrm{P}\blacktriangleright\mathrm{Rx}(r,\theta)$ → $\cos(\theta)\cdot r$

$\mathrm{P}\blacktriangleright\mathrm{Rx}(4,60°)$ → 2

$\mathrm{P}\blacktriangleright\mathrm{Rx}\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right)$ → $\left\{\frac{-3}{2},5\cdot\sqrt{2},1.3\right\}$

P▶Ry()

Catalogue >

P▶Ry(*ExprR*, θ*Expr*) ⇒ *expression*
P▶Ry(*ListeR*, θ*Liste*) ⇒ *liste*
P▶Ry(*MatriceR*, θ*Matrice*) ⇒ *matrice*

Donne la valeur de l'ordonnée 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 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(...)`.

En mode Angle en radians :

$\mathrm{P}\blacktriangleright\mathrm{Ry}(r,\theta)$ → $\sin(\theta)\cdot r$

$\mathrm{P}\blacktriangleright\mathrm{Ry}(4,60°)$ → $2\cdot\sqrt{3}$

$\mathrm{P}\blacktriangleright\mathrm{Ry}\left(\{-3,10,1.3\},\left\{\frac{\pi}{3},\frac{-\pi}{4},0\right\}\right)$ → $\left\{\frac{-3\cdot\sqrt{3}}{2},-5\cdot\sqrt{2},0.\right\}$

PassErr

Catalogue >

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 19 et **Try**, page 132.

Remarque pour la saisie des données de l'exemple : Dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ au lieu de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** et appuyez sur **Entrée**.

Pour obtenir un exemple de **PassErr**, reportez-vous à l'exemple 2 de la commande **Try**, page 132.

piecewise()

Catalogue >

piecewise(*Expr1* [, *Condition1* [, *Expr2* [, *Condition2* [, …]]]])

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 2.

Define $p(x)=\begin{cases}x, & x>0\\ \text{undef}, & x\le 0\end{cases}$ → *Done*

$p(1)$ → 1

$p(-1)$ → undef

poissCdf()

Catalogue >

poissCdf(λ**,***lowBound*,*upBound*) ⇒ *nombre* si *lowBound* et *upBound* sont des nombres, *liste* si *lowBound* et *upBound* sont des listes

poissCdf(λ**,***upBound*) (pour P(0≤X≤*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 de Poisson de moyenne λ.

Pour P(X ≤ *upBound*), définissez la borne lowBound=0

poissPdf()

Catalogue >

poissPdf(λ**,***ValX*) ⇒ *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 ∠θ]. Le vecteur doit être un vecteur ligne ou colonne et de dimension 2.

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 101.

valeurComplexe **▸Polar**

Affiche *valeurComplexe* sous forme polaire.

- Le mode Angle en degrés affiche (r∠θ).
- Le mode Angle en radians affiche reiθ.

valeurComplexe peut prendre n'importe quelle forme complexe. Toutefois, une entrée reiθ génère une erreur en mode Angle en degrés.

Remarque : vous devez utiliser les parenthèses pour les entrées polaires (r∠θ).

$[1 \quad 3.] \blacktriangleright \text{Polar} \qquad [3.16228 \quad \angle 1.24905]$

$[x \quad y] \blacktriangleright \text{Polar} \qquad \left[\sqrt{x^2+y^2} \quad \angle \frac{\pi \cdot \text{sign}(y)}{2} - \tan^{-1}\left(\frac{x}{y}\right)\right]$

En mode Angle en radians :

$(3+4\cdot i) \blacktriangleright \text{Polar} \qquad e^{i\cdot\left(\frac{\pi}{2}-\tan^{-1}\left(\frac{3}{4}\right)\right)\cdot 5}$

$\left(\left(4 \angle \frac{\pi}{3}\right)\right) \blacktriangleright \text{Polar} \qquad e^{\frac{i\cdot\pi}{3}\cdot 4}$

En mode Angle en grades :

$(4\cdot i) \blacktriangleright \text{Polar} \qquad (4 \angle 100)$

En mode Angle en degrés :

$(3+4\cdot i) \blacktriangleright \text{Polar} \qquad \left(5 \angle 90-\tan^{-1}\left(\frac{3}{4}\right)\right)$

polyCoeffs()

Catalogue >

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.

$\text{polyCoeffs}(4\cdot x^2-3\cdot x+2,x)$ $\{4,-3,2\}$

$\text{polyCoeffs}((x-1)^2\cdot(x+2)^3)$ $\{1,4,1,-10,-4,8\}$

Etend le polynôme et sélectionne *x* pour la variable omise *Var*.

$\text{polyCoeffs}((x+y+z)^2,x)$ $\{1,2\cdot(y+z),(y+z)^2\}$

$\text{polyCoeffs}((x+y+z)^2,y)$ $\{1,2\cdot(x+z),(x+z)^2\}$

$\text{polyCoeffs}((x+y+z)^2,z)$ $\{1,2\cdot(x+y),(x+y)^2\}$

polyDegree()

Catalogue >

polyDegree(*Poly* [,*Var*]**)** ⇒ *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

$\text{polyDegree}((x+y^2+z^3)^2,x)$ 2

$\text{polyDegree}((x+y^2+z^3)^2,y)$ 4

$\text{polyDegree}((x-1)^{10000},x)$ 10000

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()

Catalogue >

polyEval(*Liste1***,** *Expr1***)** ⇒ *expression*
polyEval(*Liste1***,** *Liste2***)** ⇒ *expression*

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.

$\text{polyEval}(\{a,b,c\},x)$ $a\cdot x^2+b\cdot x+c$

$\text{polyEval}(\{1,2,3,4\},2)$ 26

$\text{polyEval}(\{1,2,3,4\},\{2,-7\})$ $\{26,-262\}$

polyGcd()

Catalogue >

polyGcd(*Expr1*,*Expr2***)** ⇒ *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*]**)** ⇒ *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*]**)** ⇒ *expression*

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.

$\text{polyRemainder}(x-1,x-3)$ → 2

$\text{polyRemainder}(x-1,x^2-1)$ → $x-1$

$\text{polyRemainder}(x^2-1,x-1)$ → 0

$\text{polyRemainder}((x-y)\cdot(y-z),x+y+z,x)$ → $-(y-z)\cdot(2\cdot y+z)$

$\text{polyRemainder}((x-y)\cdot(y-z),x+y+z,y)$ → $-2\cdot x^2-5\cdot x\cdot z-2\cdot z^2$

$\text{polyRemainder}((x-y)\cdot(y-z),x+y+z,z)$ → $(x-y)\cdot(x+2\cdot y)$

polyRoots()

Catalogue >

polyRoots(*Poly***,***Var***)** ⇒ *liste*

polyRoots(*ListeCoeff***)** ⇒ *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 27.

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

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

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

$\text{polyRoots}(\{1,2,1\})$ $\{-1,-1\}$

PowerReg

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 122.)

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 165.

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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

Prgm

Catalogue >

Prgm
Bloc
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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

Calcule le plus grand commun diviseur et affiche les résultats intermédiaires.

```
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 Π(), page 156.

Product (PI)

Voir Π(), page 156.

product()

Catalogue >

product(*Liste*[**,** *Début*[**,** *Fin*]]**)** ⇒ *expression*

Donne le produit des éléments de *Liste*. *Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage d'éléments.

```
product({1,2,3,4})                  24
product({2,x,y})                    2·x·y
product({4,5,8,9},2,3)              40
```

product(*Matrice1*[**,** *Début*[**,** *Fin*]]**)** ⇒ *matrice*

Donne un vecteur ligne contenant les produits des éléments ligne par ligne de *Matrice1*. *Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage de colonnes.

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 165.

$$\text{product}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix} \qquad \begin{bmatrix}28&80&162\end{bmatrix}$$

$$\text{product}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},1,2 \qquad \begin{bmatrix}4&10&18\end{bmatrix}$$

propFrac()

Catalogue >

propFrac(*Expr1*[**,** *Var*]**)** ⇒ *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).

$\text{propFrac}\left(\frac{4}{3}\right) \quad 1+\frac{1}{3}$

$\text{propFrac}\left(\frac{-4}{3}\right) \quad -1-\frac{1}{3}$

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.

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

$\text{propFrac}(Ans) \quad \frac{1}{x+1}+x+\frac{1}{y+1}+y$

$\text{propFrac}\left(\frac{11}{7}\right) \quad 1+\frac{4}{7}$

$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right) \quad 8+\frac{37}{44}$

$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right) \quad -2-\frac{29}{44}$

Q

QR

Catalogue >

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 ctrl enter 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 · max(dim(*Matrice*)) · rowNorm(*Matrice*)

Le nombre en virgule flottante (9.) dans m1 fait que les résultats seront tous calculés en virgule flottante.

$\begin{bmatrix}1&2&3\\4&5&6\\7&8&9.\end{bmatrix} \rightarrow m1 \quad \begin{bmatrix}1&2&3\\4&5&6\\7&8&9.\end{bmatrix}$

QR *m1*,*qm*,*rm* — *Done*

$qm = \begin{bmatrix}0.123091&0.904534&0.408248\\0.492366&0.301511&-0.816497\\0.86164&-0.301511&0.408248\end{bmatrix}$

$rm = \begin{bmatrix}8.12404&9.60114&11.0782\\0.&0.904534&1.80907\\0.&0.&0.\end{bmatrix}$

ClearAZ — *Done*

QR

Catalogue >

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*.

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

$$\mathrm{QR}\ m1,qm,rm \qquad \textit{Done}$$

$$qm \qquad \begin{bmatrix} \dfrac{m}{\sqrt{m^2+o^2}} & \dfrac{-\mathrm{sign}(m\cdot p-n\cdot o)\cdot o}{\sqrt{m^2+o^2}} \\ \dfrac{o}{\sqrt{m^2+o^2}} & \dfrac{m\cdot \mathrm{sign}(m\cdot p-n\cdot o)}{\sqrt{m^2+o^2}} \end{bmatrix}$$

$$rm \qquad \begin{bmatrix} \sqrt{m^2+o^2} & \dfrac{m\cdot n+o\cdot p}{\sqrt{m^2+o^2}} \\ 0 & \dfrac{|m\cdot p-n\cdot o|}{\sqrt{m^2+o^2}} \end{bmatrix}$$

QuadReg

Catalogue >

QuadReg *X*,*Y* [, *Fréq*] [, *Catégorie*, *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 *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 122.)

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 165.

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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

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 122.)

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 165.

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
stat.R^2	Coefficient de détermination
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

R▸Pθ()

Catalogue >

R▸Pθ (*ExprX*, *ExprY*) ⇒ *expression*
R▸Pθ (*ListeX*, *ListeY*) ⇒ *liste*
R▸Pθ (*MatriceX*, *MatriceY*) ⇒ *matrice*

Donne la coordonnée θ d'un 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 :

$R▸P\theta(x,y)$ $\quad 90\cdot\text{sign}(y)-\tan^{-1}\left(\frac{x}{y}\right)$

En mode Angle en grades :

$R▸P\theta(x,y)$ $\quad 100\cdot\text{sign}(y)-\tan^{-1}\left(\frac{x}{y}\right)$

En mode Angle en radians :

$R▸P\theta(3,2)$ $\quad \tan^{-1}\left(\frac{2}{3}\right)$

$R▸P\theta\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right)$

$\begin{bmatrix}0 & \tan^{-1}\left(\frac{16}{\pi}\right)+\frac{\pi}{2} & 0.643501\end{bmatrix}$

R▸Pr()

Catalogue >

R▸Pr (*ExprX*, *ExprY*) ⇒ *expression*
R▸Pr (*ListeX*, *ListeY*) ⇒ *liste*
R▸Pr (*MatriceX*, *MatriceY*) ⇒ *matrice*

Donne la coordonnée r d'un point de coordonnées rectangulaires (*x*,*y*).

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant `R@>Pr(...)`.

En mode Angle en radians :

$R▸Pr(3,2)$ $\quad \sqrt{13}$

$R▸Pr(x,y)$ $\quad \sqrt{x^2+y^2}$

$R▸Pr\left(\begin{bmatrix}3 & -4 & 2\end{bmatrix},\begin{bmatrix}0 & \frac{\pi}{4} & 1.5\end{bmatrix}\right)$

$\begin{bmatrix}3 & \frac{\sqrt{\pi^2+256}}{4} & 2.5\end{bmatrix}$

▸Rad

Catalogue >

Expr1▸**Rad** ⇒ *expression*

Convertit l'argument en mesure d'angle en radians.

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

En mode Angle en degrés :

$(1.5)▸\text{Rad}$ $\quad (0.02618)^r$

En mode Angle en grades :

$(1.5)▸\text{Rad}$ $\quad (0.023562)^r$

rand()

Catalogue >

rand() ⇒ *expression*
rand(*nmbreEssais***)** ⇒ *liste*

rand() donne un nombre aléatoire compris entre 0 et 1.

rand(*nbreEssais***)** donne une liste de nombres aléatoires compris entre 0 et 1 pour le nombre d'essais *nbreEssais*.

Réinitialise le générateur de nombres aléatoires.

RandSeed 1147 $\quad$ *Done*

rand(2) $\quad \{0.158206, 0.717917\}$

randBin()

Catalogue >

randBin(*n*, *p***)** ⇒ *expression*
randBin(*n*, *p*, *nbreEssais***)** ⇒ *liste*

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*.

randBin(80,.5)	34.
randBin(80,.5,3)	{47.,41.,46.}

randInt()

Catalogue >

randInt(*LimiteInf*,*LimiteSup***)** ⇒ *expression*
randInt(*LimiteInf*,*LimiteSup*,*nbreEssais***)** ⇒ *liste*

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*.

randInt(3,10)	7.
randInt(3,10,4)	{8.,9.,4.,4.}

randMat()

Catalogue >

randMat(*nbreLignes*, *nbreColonnes***)** ⇒ *matrice*

Donne une matrice aléatoire 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)	$\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$

Remarque : Les valeurs de cette matrice changent chaque fois que l'on appuie sur enter.

randNorm()

Catalogue >

randNorm(μ, σ**)** ⇒ *expression*
randNorm(μ, σ, *nbreEssais***)** ⇒ *liste*

Donne un nombre décimal aléatoire issu de la loi normale spécifiée. Il peut s'agir de tout nombre réel, mais le résultat obtenu sera essentiellement compris dans l'intervalle [μ−3 · σ, μ+3 · σ].

randNorm(μ, σ, *nbreEssais***)** donne une liste de nombres décimaux tirés d'une distribution normale spécifiée pour un nombre d'essais *nbreEssais*.

RandSeed 1147	*Done*
randNorm(0,1)	0.492541
randNorm(3,4.5)	−3.54356

randPoly()

Catalogue >

randPoly(*Var*, *Ordre***)** ⇒ *expression*

Donne un polynôme aléatoire de la variable *Var* de degré *Ordre* spécifié. Les coefficients sont des entiers aléatoires compris entre −9 et 9. Le premier coefficient sera non nul.

Ordre doit être un entier compris entre 0 et 99.

RandSeed 1147	*Done*
randPoly(*x*,5)	$-2\cdot x^5+3\cdot x^4-6\cdot x^3+4\cdot x-6$

randSamp()

Catalogue >

randSamp(*Liste*,*nbreEssais*[,*sansRem*]**)** ⇒ *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 *list3*={1,2,3,4,5}	*Done*
Define *list4*=randSamp(*list3*,6)	*Done*
list4	{5.,1.,3.,3.,4.,4.}

RandSeed

Catalogue >

RandSeed *Nombre*

Si *Nombre* = 0, réinitialise le générateur de nombres aléatoires. Si *Nombre* ≠ 0, sert à générer deux nombres initiaux 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 aussi **imag()**, page 59.

real(2+3·*i*)	2
real(*z*)	*z*
real(*x*+*i*·*y*)	*x*

real(*Liste1***)** ⇒ *liste*

Donne la liste des parties réelles de tous les éléments.

real({*a*+*i*·*b*,3,*i*})	{*a*,3,0}

real(*Matrice1***)** ⇒ *matrice*

Donne la matrice des parties réelles de tous les éléments.

real$\begin{bmatrix} a+i\cdot b & 3 \\ c & i \end{bmatrix}$	$\begin{bmatrix} a & 3 \\ c & 0 \end{bmatrix}$

▸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 aussi **▸Polar**, page 91.

valeurComplexe **▸Rect**

Affiche *valeurComplexe* sous forme rectangulaire (a+bi). *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 les parenthèses pour les entrées polaires (r∠θ).

$\begin{bmatrix} 3 & \angle\frac{\pi}{4} & \angle\frac{\pi}{6} \end{bmatrix}$▸Rect

$$\begin{bmatrix} \frac{3\cdot\sqrt{2}}{4} & \frac{3\cdot\sqrt{2}}{4} & \frac{3\cdot\sqrt{3}}{2} \end{bmatrix}$$

$[a \quad \angle b \quad \angle c]$

$$[a\cdot\cos(b)\cdot\sin(c) \quad a\cdot\sin(b)\cdot\sin(c) \quad a\cdot\cos(c)]$$

En mode Angle en radians :

$\left(4\cdot e^{\frac{\pi}{3}}\right)$▸Rect	$4\cdot e^{\frac{\pi}{3}}$
$\left(\left(4 \angle \frac{\pi}{3}\right)\right)$▸Rect	$2+2\cdot\sqrt{3}\cdot i$

En mode Angle en grades :

((1 ∠ 100))▸Rect	*i*

En mode Angle en degrés :

((4 ∠ 60))▸Rect	$2+2\cdot\sqrt{3}\cdot i$

Remarque : pour taper ∠ à partir du clavier, sélectionnez-le dans la liste des symboles du Catalogue.

ref()

Catalogue >

ref(*Matrice1***[, *Tol*]) ⇒ *matrice***

Donne une réduite de Gauss de la matrice *Matrice1*.

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 ctrl enter 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 · max(dim(*Matrice1*)) · rowNorm(*Matrice1*)

N'utilisez pas d'éléments non définis dans *Matrice1*. 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 :

$$\text{ref}\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

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.

$$\text{ref}\begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mid a=0 \quad \Rightarrow \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Remarque : voir aussi **rref()**, page 107.

$$\text{ref}\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix} \quad \begin{bmatrix} 1 & \frac{-2}{5} & \frac{-4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ e & f & g \end{bmatrix} \rightarrow m1 \quad \begin{bmatrix} a & b & c \\ e & f & g \end{bmatrix}$$

$$\text{ref}(m1) \quad \begin{bmatrix} a & b & c \\ e & f & g \end{bmatrix}$$

remain()

Catalogue >

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 :

remain(x,0) x
remain(x,y) x−y · iPart(x/y)

Vous remarquerez que **remain(**−x,y**)** −**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 79.

$$\text{remain}(7,0) \quad 7$$
$$\text{remain}(7,3) \quad 1$$
$$\text{remain}(-7,3) \quad -1$$
$$\text{remain}(7,-3) \quad 1$$
$$\text{remain}(-7,-3) \quad -1$$
$$\text{remain}(\{12,-14,16\},\{9,7,-5\}) \quad \{3,0,1\}$$

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

Request *ChaîneInvite*, *var*[, *IndicAff*[, *VarÉtat*]]
Request *ChaîneInvite*, *fonc*(*arg1*, *...argn*)
[, *IndicAff*[, *VarÉtat*]]

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 pour la réponse de 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 poursuit sans accepter la saisie. Le programme utilise la précédente valeur de *var* si *var* a déjà été 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,** appuyé sur **Entrée** ou sur **Ctrl+Entrée**, la variable *VarÉtat* prend la valeur **1**.
- Sinon, elle prend la valeur **0**.

L'argument *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 :

Define *func*(*arg1*, *...argn*) = *réponse de l'utilisateur*

Le programme peut alors utiliser la fonction définie *func*(). *chaîneinvite* doit guider l'utilisateur pour la saisie d'une réponse de *l'utilisateur appropriée* qui complète la définition de la fonction.

Remarque : vous pouvez utiliser la commande **Request** dans un programme créé par l'utilisateur, mais pas dans une fonction.

Pour arrêter un programme qui contient une **Request** commande dans une boucle infinie :

- **Windows® :** maintenez enfoncé la touche **F12** et appuyez plusieurs fois sur **Entrée**.
- **Macintosh® :** maintenez enfoncé la touche **F6** et appuyez plusieurs fois sur **Entrée**.
- **Unité :** maintenez enfoncé la touche [on] et appuyez plusieurs fois sur [enter].

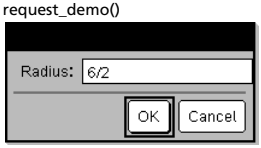
Remarque : voir aussi **RequestStr**, page 104.

Définissez un programme :

```
Define request_demo()=Prgm
  Request "Rayon : ",r
  Disp "Area = ",pi*r^2
EndPrgm
```

Exécutez le programme et saisissez une réponse :

```
request_demo()
```



Après avoir sélectionné **OK**, le résultat suivant s'affiche :

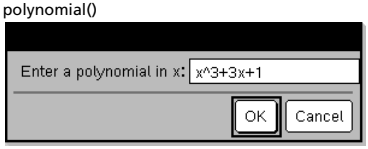
```
Rayon :  6/2
Area= 28.2743
```

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()
```



Après avoir sélectionné **OK**, le résultat suivant s'affiche :

```
Saisissez un polynôme en x : x^3+3x+1
Les racines réelles sont : {-0.322185}
```

RequestStr

Catalogue >

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. La commande **Request** interprète la réponse comme une expression, à moins que l'utilisateur ne la saisisse entre guillemets (" ").

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 **RequestStr** commande dans une boucle infinie :

- **Windows® :** maintenez enfoncé la touche **F12** et appuyez plusieurs fois sur **Entrée**.
- **Macintosh® :** maintenez enfoncé la touche **F6** et appuyez plusieurs fois sur **Entrée**.
- **Unité :** maintenez enfoncé la touche [on] et appuyez plusieurs fois sur [enter].

Remarque : voir aussi **Request**, page 103.

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()
```

Après avoir sélectionné **OK**, le résultat affiché est le suivant (notez que si l'argument *IndicAff* 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

Catalogue >

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur [↵] à la place de [enter] à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define factoral(nn)=Func
                      Local answer,count
                      1→answer
                      For count,1,nn
                      answer·count→answer
                      EndFor
                      Return answer
                      EndFunc
                                        Done
factoral(3)                                6
```

right()

Catalogue >

right(*Liste1*[, *Nomb*]**)** ⇒ *liste*

Donne les *Nomb* éléments les plus à droite de la liste *Liste1*.

Si *Nomb* est absent, on obtient *Liste1*.

right(*chaîneSrce*[, *Nomb*]**)** ⇒ *chaîne*

Donne la chaîne formée par les *Nomb* caractères les plus à droite de la chaîne de caractères *chaîneSrce*.

Si *Nomb* est absent, on obtient *chaîneSrce*.

right(*Comparaison***)** ⇒ *expression*

Donne le membre de droite d'une équation ou d'une inéquation.

```
right({1,3,-2,4},3)                {3,-2,4}

right("Hello",2)                        "lo"

right(x<3)                                3
```

rk23()

Catalogue >

rk23(*Expr*, *Var*, *VarDép*, {*Var0*, *MaxVar*}, *Var0Dép*, *IncVar* [, *TolErr*]**)** ⇒ *matrice*

rk23(*SystèmeExpr*, *Var*, *ListeVarDép*, {*Var0*, **Max***Var*}, *ListeVar0Dép*, *IncVar* [, *TolErr*]**)** ⇒ *matrice*

rk23(*SystèmeExpr*, *Var*, *ListeVarDép*, {*Var0*, **Max***Var*}, *ListeVar0Dép*, *IncVar* [, *TolErr*]**)** ⇒ *matrice*

Utilise la méthode de Runge-Kutta pour résoudre le système d'équations.

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

avec *VarDép*(*Var0*)=*Var0Dép* pour l'intervalle [*Var0*,*MaxVar*]. 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.

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, signe*(IncVar)* = signe(MaxVar-Var0) et les solutions sont retournées pour *Var0*+i*·*IncVar* pour tout i=0,1,2,… tel que *Var0*+i·*IncVar* soit dans [*var0*,*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).

Équation différentielle :
y'=0.001*y*(100-y) et y(0)=10

$$\text{rk23}\left(0.001\cdot y\cdot(100-y),t,y,\{0,100\},10,1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9493 & 13.042 & 14.2 \end{bmatrix}$$

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

Même équation avec *TolErr* définie à 1.E•6

$$\text{rk23}\left(0.001\cdot y\cdot(100-y),t,y,\{0,100\},10,1,1.\text{E}{-6}\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9495 & 13.0423 & 14.2189 \end{bmatrix}$$

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

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

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

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

$$\{10.,10.9367,11.9494,13.0423,14.2189,15.48$$

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

$$\text{rk23}\left(\begin{cases} -y1+0.1\cdot y1\cdot y2 \\ 3\cdot y2-y1\cdot y2 \end{cases},t,\{y1,y2\},\{0,5\},\{2,5\},1\right)$$

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 2. & 1.94103 & 4.78694 & 3.25253 & 1.82848 \\ 5. & 16.8311 & 12.3133 & 3.51112 & 6.27245 \end{bmatrix}$$

root()

Catalogue >

root(*Expr***)** ⇒ *root*

root(*Expr1*, *Expr2***)** ⇒ *root*

root(*Expr***)** affiche la racine carrée de *Expr*.

root(*Expr1*, *Expr2***)** affiche la racine *Expr2* de *Expr1*. *Expr1* peut être un nombre réel ou une constant complexe en virgule flottante, un entier ou une constante rationnelle complexe, ou une expression symbolique générale.

Remarque : voir aussi **Modèle Racine n-ième**, page 1.

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

$$\sqrt[3]{3} \qquad 3^{\frac{1}{3}}$$

$$\sqrt[3]{3.} \qquad 1.44225$$

rotate()

Catalogue >

rotate(*Entier1*[**,***nbreRotations*]**)** ⇒ *entier*

Permute 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 13.

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 :

Tous les bits permutent vers la droite.

0b00000000000000111101011000011010 1

Le bit le plus à droite passe à la position la plus à gauche.

donne :

0b10000000000000111101011000011010

Le résultat est affiché selon le mode Base utilisé.

En mode base Bin :

rotate(0b11111111111111111111111111111111) → 0b10000000000000000000000000000001

rotate(256,1) → 0b1000000000

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).

rotate(*Liste1*[**,***nbreRotations*]**)** ⇒ *liste*

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 circulation de un bit vers la droite).

En mode base Dec :

rotate({1,2,3,4}) → {4,1,2,3}

rotate({1,2,3,4},-2) → {3,4,1,2}

rotate({1,2,3,4},1) → {2,3,4,1}

rotate(*Chaîne1*[**,***nbreRotations*]**)** ⇒ *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 circulaire d'un caractère vers la droite).

rotate("abcd") → "dabc"

rotate("abcd",-2) → "cdab"

rotate("abcd",1) → "bcda"

round()

Catalogue >

round(*Expr1*[**,** *n*]**)** ⇒ *expression*

Arrondit l'argument au nombre de chiffres n spécifié après la virgule.

n doit être un entier compris entre 0 et 12. Si *n* est omis, arrondit l'argument à 12 chiffres significatifs.

Remarque : le mode d'affichage des chiffres peut affecter le résultat affiché.

round(1.234567,3) → 1.235

round(*Liste1*[**,** *n*]**)** ⇒ *liste*

Donne la liste des éléments arrondis au nombre de chiffres n spécifié.

round({π,√2,ln(2)},4) → {3.1416,1.4142,0.6931}

round(*Matrice1*[**,** *n*]**)** ⇒ *matrice*

Donne la matrice des éléments arrondis au nombre de chiffres n spécifié.

round([ln(5) ln(3); π e^1],1) → [1.6 1.1; 3.1 2.7]

rowAdd()

Catalogue >

rowAdd(*Matrice1***,** *IndexL1***,** *IndexL2***)** ⇒ *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(*Matrice***)** ⇒ *expression*

Donne le nombre de lignes de *Matrice*.

Remarque : voir aussi **colDim()**, page 19.

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

$\text{rowDim}(m1)$ $\quad 3$

rowNorm()

Catalogue >

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 19.

$\text{rowNorm}\left(\begin{bmatrix}-5 & 6 & -7\\ 3 & 4 & 9\\ 9 & -9 & -7\end{bmatrix}\right)$ $\quad 25$

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***])** ⇒ *matrice*

Donne la réduite de Gauss-Jordan de *Matrice1*.

$\text{rref}\left(\begin{bmatrix}-2 & -2 & 0 & -6\\ 1 & -1 & 9 & -9\\ -5 & 2 & 4 & -4\end{bmatrix}\right)$ $\quad\begin{bmatrix}1 & 0 & 0 & \frac{66}{71}\\ 0 & 1 & 0 & \frac{147}{71}\\ 0 & 0 & 1 & \frac{-62}{71}\end{bmatrix}$

rref()

Catalogue >

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 ctrl enter 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 · max(dim(*Matrice1*)) · rowNorm(*Matrice1*)

Remarque : Voir aussi **ref()**, page 102.

$$\text{rref}\left(\begin{bmatrix} a & b & x \\ c & d & y \end{bmatrix}\right) \qquad \begin{bmatrix} a & b & x \\ c & d & y \end{bmatrix}$$

S

sec()

Touche

sec(*Expr1***)** ⇒ *expression*
sec(*Liste1***)** ⇒ *liste*

Affiche la sécante de *Expr1* ou retourne la liste des sécantes des éléments de *Liste1*.

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 degrés :

$$\sec(45) \qquad \sqrt{2}$$

$$\sec(\{1,2.3,4\}) \qquad \left\{\frac{1}{\cos(1)}, 1.00081, \frac{1}{\cos(4)}\right\}$$

sec^{-1}()

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 :

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

En mode Angle en grades :

$$\sec^{-1}(\sqrt{2}) \qquad 50$$

En mode Angle en radians :

$$\sec^{-1}(\{1,2,5\}) \qquad \left\{0, \frac{\pi}{3}, \cos^{-1}\left(\frac{1}{5}\right)\right\}$$

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*.

$$\text{sech}(3) \qquad \frac{1}{\cosh(3)}$$

$$\text{sech}(\{1,2.3,4\}) \qquad \left\{\frac{1}{\cosh(1)}, 0.198522, \frac{1}{\cosh(4)}\right\}$$

sech^{-1}()

Catalogue >

sech^{-1}(*Expr1***)** $\Rightarrow$ *expression*

sech^{-1} (*Liste1***)** $\Rightarrow$ *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 :

$\text{sech}^{-1}(1)$ → 0

$\text{sech}^{-1}(\{1,-2,2.1\})$ → $\left\{0,\frac{2\cdot\pi}{3}\cdot i,8.\text{E}^{-}15+1.07448\cdot i\right\}$

seq()

Catalogue >

seq(*Expr***, *Var*, *Début*, *Fin*[, *Incrément*])** $\Rightarrow$ *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.

$\text{seq}(n^2,n,1,6)$ → $\{1,4,9,16,25,36\}$

$\text{seq}\left(\frac{1}{n},n,1,10,2\right)$ → $\left\{1,\frac{1}{3},\frac{1}{5},\frac{1}{7},\frac{1}{9}\right\}$

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$ → $\frac{1968329}{1270080}$

Appuyez sur **Ctrl**+**Entrée** ctrl enter (Macintosh®: ⌘+**Enter**) pour évaluer :

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2},n,1,10,1\right)\right)$ → 1.54977

seqGen()

Catalogue >

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 $u(n) = u(n\text{-}1)^2/2$, avec $u(1)$=**2** et *IncVar*=**1**.

$$\text{seqGen}\left(\frac{(u(n-1))^2}{n}, n, u, \{1,5\}, \{2\}\right) \qquad \left\{2, 2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\}$$

Exemple avec Var0=2 :

$$\text{seqGen}\left(\frac{u(n-1)+1}{n}, n, u, \{2,5\}, \{3\}\right) \qquad \left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\}$$

Exemple dans lequel la valeur initiale est symbolique :

$$\text{seqGen}(u(n-1)+2, n, u, \{1,5\}, \{a\}) \qquad \{a, a+2, a+4, a+6, a+8\}$$

Système de deux suites :

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u2(n-1)}{2}+u1(n-1)\right\}, n, \{u1,u2\}, \{1,5\}, \begin{bmatrix} \_ \\ 2 \end{bmatrix}\right) \qquad \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

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()

Catalogue >

seqn(*Expr*(*u*, *n* [, *ListeValeursInit*[, *nMax* [, *ValeurMax*]]]) ⇒ *liste*

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*]]) ⇒ *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() appel **seqGen( )** avec *n0*=**1** et *Incn* =**1**

Génère les cinq premières valeurs de la suite $u(n) = u(n\text{-}1)/2$, avec $u(1)$=**2**.

$$\text{seqn}\left(\frac{u(n-1)}{n}, \{2\}, 6\right) \qquad \left\{2, 1, \frac{1}{3}, \frac{1}{12}, \frac{1}{60}, \frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2}, 6\right) \qquad \left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}\right\}$$

series(*Expr1***, *Var*, *Ordre* [, *Point*]) ⇒ *expression***
series(*Expr1***, *Var*, *Ordre* [, *Point*]) | *Var*>*Point* ⇒ *expression***
series(*Expr1***, *Var*, *Ordre* [, *Point*]) | *Var*<*Point* ⇒ *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.

La valeur par défaut de Point est 0. *Point* peut être ∞ ou −∞, auxquels cas le développement s'effectue jusqu'au degré *Ordre* en 1/(*Var* − *Point*).

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 = ∞ ou −∞.

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 $(-1)^{\text{floor}(\ldots\text{angle}(\ldots)\ldots)}$ 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 40.

$$\text{series}\left(\frac{1-\cos(x-1)}{(x-1)^2},x,4,1\right)$$
$$\frac{1}{2}-\frac{(x-1)^2}{24}+\frac{(x-1)^4}{720}$$

$$\text{series}\left(\ln\left(x^x-1\right),x,2\right)$$
$$\ln(x\cdot\ln(x))+\frac{x\cdot\ln(x)}{2}+\frac{x^2\cdot(\ln(x))^2}{24}$$

$$\text{series}\left(e^{\frac{-1}{z_-}},z\_,1\right) \qquad \text{series}\left(e^{\frac{-1}{z_-}},z\_,1,0,0\right)$$

$$\text{series}\left(\left(1+\frac{1}{n}\right)^n,n,2,\infty\right) \qquad e-\frac{e}{2\cdot n}+\frac{11\cdot e}{24\cdot n^2}$$

$$\text{series}\left(\tan^{-1}\left(\frac{1}{x}\right),x,5\right)|x>0 \qquad \frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}$$

$$\text{series}\left(\int\frac{\sin(x)}{x}\,dx,x,6\right) \qquad x-\frac{x^3}{18}+\frac{x^5}{600}$$

$$\text{series}\left(\int_0^x \sin(x\cdot\sin(t))\,dt,x,7\right)$$
$$\frac{x^3}{2}-\frac{x^5}{24}-\frac{29\cdot x^7}{720}$$

$$\text{series}\left(\left(1+e^x\right)^2,x,2,1\right)$$
$$e\cdot(2\cdot e+1)\cdot(x-1)^2+(2\cdot e^2+2\cdot e)\cdot(x-1)+(e+1)^2$$

setMode(*EntierNomMode***,** *EntierRéglage***)** ⇒ *entier*
setMode(*liste***)** ⇒ *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)** → *var*, **setMode(***var***)** permet de restaurer ces réglages jusqu'à fermeture du programme ou de la fonction. Voir **getMode()**, page 55.

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.

Remarque pour la saisie des données de l'exemple : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur [↵] à la place de [enter] à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

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 prog1()=Prgm                    Done
               Disp approx(π)
               setMode(1,16)
               Disp approx(π)
               EndPrgm

prog1()
                                       3.14159
                                       3.14
                                       Done
```

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
Système d'unités	**8**	**1**=SI, **2**=Ang/US

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 13.

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.

0b00000000000000111101011000011010

Insère 0 si le premier bit est un 0
ou 1 si ce bit est un 1.

donne :

0b00000000000000011110101100011010

Le résultat est affiché selon le mode Base utilisé. Les zéros de tête ne sont pas affichés.

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

Important : pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h (zéro, pas la lettre O).

shift(*Liste1* **[,***nbreDécal***])** ⇒ *liste*

Donne une copie de *Liste1* dont les éléments ont été décalés vers la gauche ou vers la droite de *nbreDécal* éléments. Ne modifie en rien *Liste1*.

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).

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(*Chaîne1* **[,***nbreDécal***])** ⇒ *chaîne*

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.

shift("abcd")	" abc"
shift("abcd",−2)	" ab"
shift("abcd",1)	"bcd "

sign()

Catalogue >

sign(*Expr1***)** ⇒ *expression*
sign(*Liste1***)** ⇒ *liste*
sign(*Matrice1***)** ⇒ *matrice*

Pour une *Expr1* réelle ou complexe, donne *Expr1*/**abs(***Expr1***)** si *Expr1*≠ 0.

Donne 1 si l'expression Expression1 est positive.

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.

$\text{sign}(-3.2)$ → $-1.$

$\text{sign}(\{2,3,4,-5\})$ → $\{1,1,1,-1\}$

$\text{sign}(1+|x|)$ → 1

En mode Format complexe Réel :

$\text{sign}([-3 \quad 0 \quad 3])$ → $[-1 \quad \pm 1 \quad 1]$

simult()

Catalogue >

simult(*matriceCoeff***,** *vecteurConst***[,** *Tol***])** ⇒ *matrice*

Donne un vecteur colonne contenant les solutions d'un système d'équations.

Remarque : voir aussi **linSolve()**, page 69.

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 · max(dim(*matriceCoeff*)) · rowNorm(*matriceCoeff*)

Résolution de x et y :
x + 2y = 1
3x + 4y = −1

$\text{simult}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},\begin{bmatrix}1\\ -1\end{bmatrix}\right)$ → $\begin{bmatrix}-3\\ 2\end{bmatrix}$

La solution est x=−3 et y=2.

Résolution :
ax + by = 1
cx + dy = 2

$\begin{bmatrix}a & b\\ c & d\end{bmatrix} \rightarrow matx1$ → $\begin{bmatrix}a & b\\ c & d\end{bmatrix}$

$\text{simult}\left(matx1,\begin{bmatrix}1\\ 2\end{bmatrix}\right)$ → $\begin{bmatrix}\frac{-(2\cdot b-d)}{a\cdot d-b\cdot c}\\ \frac{2\cdot a-c}{a\cdot d-b\cdot c}\end{bmatrix}$

simult(*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.

Résolution :
x + 2y = 1
3x + 4y = −1

x + 2y = 2
3x + 4y = −3

$\text{simult}\left(\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix},\begin{bmatrix}1 & 2\\ -1 & -3\end{bmatrix}\right)$ → $\begin{bmatrix}-3 & -7\\ 2 & \frac{9}{2}\end{bmatrix}$

Pour le premier système, x=−3 et y=2. Pour le deuxième système, x=−7 et y=9/2.

▸sin

Catalogue >

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(...) 1−sin(...)^2
de sorte que les puissances de sin(...) restantes ont des exposants dans (0, 2). Le résultat ne contient donc pas cos(...) si et seulement si cos(...) 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*.

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

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 :

$\sin\left(\frac{\pi}{4}{}^{r}\right) \qquad \frac{\sqrt{2}}{2}$

$\sin(45) \qquad \frac{\sqrt{2}}{2}$

$\sin(\{0,60,90\}) \qquad \left\{0,\frac{\sqrt{3}}{2},1\right\}$

En mode Angle en grades :

$\sin(50) \qquad \frac{\sqrt{2}}{2}$

En mode Angle en radians :

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

$\sin(45°) \qquad \frac{\sqrt{2}}{2}$

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 :

$$\sin\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} = \begin{bmatrix}0.9424 & -0.04542 & -0.031999\\ -0.045492 & 0.949254 & -0.020274\\ -0.048739 & -0.00523 & 0.961051\end{bmatrix}$$

sin⁻¹()

Touche trig

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(...)`.

En mode Angle en degrés :

$\sin^{-1}(1)$ → 90

En mode Angle en grades :

$\sin^{-1}(1)$ → 100

En mode Angle en radians :

$\sin^{-1}(\{0,0.2,0.5\})$ → $\{0,0.201358,0.523599\}$

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 radians et en mode Format complexe Rectangulaire :

$$\sin^{-1}\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}-0.164058-0.064606\cdot i & 1.49086-2.1051\cdot i\\ 0.725533-1.51594\cdot i & 0.947305-0.7783\cdot i\\ 2.08316-2.63205\cdot i & -1.79018+1.2718\end{bmatrix}$$

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

sinh()

Catalogue >

sinh(*Expr1***)** ⇒ *expression*
sinh(*Liste1***)** ⇒ *liste*

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*.

$\sinh(1.2)$ → 1.50946

$\sinh(\{0,1.2,3.\})$ → $\{0,1.50946,10.0179\}$

sinh(*matriceCarrée1***)** ⇒ *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 :

$$\sinh\begin{bmatrix}1 & 5 & 3\\ 4 & 2 & 1\\ 6 & -2 & 1\end{bmatrix} \rightarrow \begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

sinh⁻¹()

Catalogue >

sinh⁻¹(*Expr1***)** ⇒ *expression*
sinh⁻¹(*Liste1***)** ⇒ *liste*

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(...)`.

$\sinh^{-1}(0)$ → 0

$\sinh^{-1}(\{0,2.1,3\})$ → $\{0,1.48748,\sinh^{-1}(3)\}$

sinh⁻¹()

Catalogue >

sinh⁻¹(*matriceCarrée1***)** ⇒ *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()**.

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

En mode Angle en radians :

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

$$\begin{bmatrix} 0.041751 & 2.15557 & 1.1582 \\ 1.46382 & 0.926568 & 0.112557 \\ 2.75079 & -1.5283 & 0.57268 \end{bmatrix}$$

SinReg

Catalogue >

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 122.)

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 165.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : a • 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 *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.YReg	Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*
stat.FreqReg	Liste des fréquences correspondant à *stat.XReg* et *stat.YReg*

solve(*Équation*, *Var***)** ⇒ *expression booléenne*
solve(*Équation*, *Var*=*Init***)** ⇒ *expression booléenne*
solve(*Inéquation*, *Var***)** ⇒ *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.

$\text{solve}(a\cdot x^2+b\cdot x+c=0,x)$

$x=\frac{\sqrt{b^2-4\cdot a\cdot c}-b}{2\cdot a}$ or $x=\frac{-\left(\sqrt{b^2-4\cdot a\cdot c}+b\right)}{2\cdot a}$

Les solutions peuvent ne pas être des solutions réelles finies pour certaines valeurs des paramètres.

$Ans|a=1$ and $b=1$ and $c=1$

$x=\frac{-1}{2}+\frac{\sqrt{3}}{2}\cdot i$ or $x=\frac{-1}{2}-\frac{\sqrt{3}}{2}\cdot i$

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.

$\text{solve}((x-a)\cdot e^x=-x\cdot(x-a),x)$

$x=a$ 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 ≥, ≤, < 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.

$\text{exact}(\text{solve}((x-a)\cdot e^x=-x\cdot(x-a),x))$

$e^x+x=0$ 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 :

$\text{solve}\left(\tan(x)=\frac{1}{x},x\right)|x>0$ and $x<1$

$x=0.860334$

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.

$\text{solve}(x=x+1,x)$ false

$\text{solve}(x=x,x)$ true

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.

$2\cdot x-1\leq 1$ and $\text{solve}(x^2\neq 9,x)$ $\quad x\neq -3$ 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 :

$\text{solve}(\sin(x)=0,x)$ $\quad x=\boldsymbol{n}1\cdot\pi$

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.

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

$\text{solve}(\sqrt{x}=-2,x)$ false

$\text{solve}(-\sqrt{x}=-2,x)$ $\quad x=4$

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

solve(*Éqn1* **and** *Éqn2* [**and** …], *VarOuInit1*, *VarOuInit2* [, …]**)** ⇒ *expression booléenne*

solve(*SystèmeÉq*, *VarOuInit1*, *VarOuInit2* [, …]**)** ⇒ *expression booléenne*

solve({*Eqn1*, *Eqn2* [,...]} {*VarOuInit1*, *VarOuInit2* [, …]**})** ⇒ *expression booléenne*

Donne les solutions réelles possibles d'un système d'équations algébriques, où chaque *VarOuInit* définit une variable du système à résoudre.

Vous pouvez séparer les équations par l'opérateur **and** ou entrer un système d'équations *SystèmÉq* en utilisant un modèle du Catalogue. Le nombre d'arguments *VarOuInit* doit correspondre au nombre d'équations. Vous pouvez également spécifier une condition initiale pour les variables. Chaque *VarOuInit* doit utiliser le format suivant :

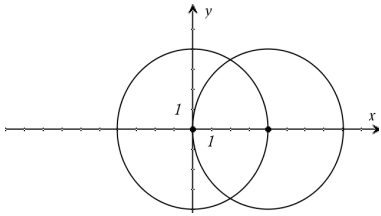
variable
– ou –
variable = nombre réel ou non réel

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

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.

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



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.

$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2,\{x,y\}\right)$$
$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ or } x=\frac{r}{2} \text{ and } y=\frac{-\sqrt{3}\cdot r}{2}$$

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.

$$\text{solve}\left(x^2+y^2=r^2 \text{ and } (x-r)^2+y^2=r^2,\{x,y,z\}\right)$$
$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ and } z=c1 \text{ or } x=\frac{r}{2} \text{ and } y\to$$

La résolution du problème montre comment les solutions peuvent contenir des constantes arbitraires de type *c*k, où k est un suffixe entier compris entre 1 et 255.

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

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 *VarOuInit*.

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.

$$\text{solve}\left(x+e^z\cdot y=1 \text{ and } x-y=\sin(z),\{x,y\}\right)$$
$$x=\frac{e^z\cdot\sin(z)+1}{e^z+1} \text{ and } y=\frac{-(\sin(z)-1)}{e^z+1}$$

solve() **Catalogue >**

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.

$\text{solve}\left(e^z \cdot y=1 \text{ and } -y=\sin(z), \{y,z\}\right)$
$y=2.812\text{E}-10 \text{ and } z=21.9911 \text{ or } y=0.00187$

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

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.

$\text{solve}\left(e^z \cdot y=1 \text{ and } -y=\sin(z), \{y,z=2\cdot\pi\}\right)$
$y=0.001871 \text{ and } z=6.28131$

SortA **Catalogue >**

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.

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 165.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA *list1*	*Done*
list1	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA *list2*,*list1*	*Done*
list2	$\{1,2,3,4\}$
list1	$\{4,3,2,1\}$

SortD **Catalogue >**

SortD *Liste1*[, *Liste2*] [, *Liste3*] ...
SortD *Vecteur1*[, *Vecteur2*] [, *Vecteur3*] ...

Identique à **SortA**, mais **SortD** trie les éléments en ordre décroissant.

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 165.

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD *list1*,*list2*	*Done*
list1	$\{4,3,2,1\}$
list2	$\{3,4,1,2\}$

▸Sphere

Catalogue >

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 ∠θ ∠ϕ].

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.

Appuyez sur **Ctrl**+**Entrée** [ctrl] [enter] (Macintosh®: ⌘+**Enter**) pour évaluer :

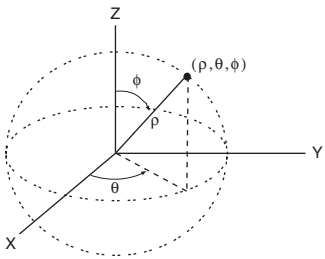
$$\begin{bmatrix}1 & 2 & 3\end{bmatrix}\blacktriangleright\text{Sphere}$$
$$\begin{bmatrix}3.74166 & \angle 1.10715 & \angle 0.640522\end{bmatrix}$$

Appuyez sur **Ctrl**+**Entrée** [ctrl] [enter] (Macintosh®: ⌘+**Enter**) pour évaluer :

$$\left(\begin{bmatrix}2 & \angle\frac{\pi}{4} & 3\end{bmatrix}\right)\blacktriangleright\text{Sphere}$$
$$\begin{bmatrix}3.60555 & \angle 0.785398 & \angle 0.588003\end{bmatrix}$$

Appuyez sur [enter]

$$\left(\begin{bmatrix}2 & \angle\frac{\pi}{4} & 3\end{bmatrix}\right)\blacktriangleright\text{Sphere}$$
$$\begin{bmatrix}\sqrt{13} & \angle\frac{\pi}{4} & \angle\cos^{-1}\left(\frac{3\cdot\sqrt{13}}{13}\right)\end{bmatrix}$$



sqrt()

Catalogue >

sqrt(*Expr1***)** ⇒ *expression*
sqrt(*Liste1***)** ⇒ *liste*

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.

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

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
"r²"	0.996109
"r"	0.998053
"Resid"	"{...}"

stat.values

"Linear Regression (mx+b)"
"m*x+b"
3.2
1.2
0.996109
0.998053
"{⁻0.4,0.4,0.2,0.,⁻0.2}"

stat.a
stat.AdjR²
stat.b
stat.b0
stat.b1
stat.b2
stat.b3
stat.b4
stat.b5
stat.b6
stat.b7
stat.b8
stat.b9
stat.b10
stat.bList
stat.χ^2
stat.c
stat.CLower
stat.CLowerList
stat.CompList
stat.CompMatrix
stat.CookDist
stat.CUpper
stat.CUpperList
stat.d
stat.dfDenom
stat.dfBlock
stat.dfCol
stat.dfError
stat.dfInteract
stat.dfReg
stat.dfNumer
stat.dfRow
stat.DW
stat.e
stat.ExpMatrix
stat.F
stat.FBlock
stat.Fcol
stat.FInteract
stat.FreqReg
stat.Frow
stat.Leverage
stat.LowerPred
stat.LowerVal
stat.m
stat.MaxX
stat.MaxY
stat.ME
stat.MedianX
stat.MedianY
stat.MEPred
stat.MinX
stat.MinY
stat.MS
stat.MSBlock
stat.MSCol
stat.MSError
stat.MSInteract
stat.MSReg
stat.MSRow
stat.n
stat.$\hat{p}$
stat.$\hat{p}$1
stat.$\hat{p}$2
stat.$\hat{p}$Diff
stat.PList
stat.PVal
stat.PValBlock
stat.PValCol
stat.PValInteract
stat.PValRow
stat.Q1X
stat.Q1Y
stat.Q3X
stat.Q3Y
stat.r
stat.r^2
stat.RegEqn
stat.Resid
stat.ResidTrans
stat.σx
stat.σy
stat.σx1
stat.σx2
stat.Σx
stat.Σx^2
stat.Σxy
stat.Σy
stat.Σy^2
stat.s
stat.SE
stat.SEList
stat.SEPred
stat.sResid
stat.SEslope
stat.sp
stat.SS
stat.SSBlock
stat.SSCol
stat.SSX
stat.SSY
stat.SSError
stat.SSInteract
stat.SSReg
stat.SSRow
stat.tList
stat.UpperPred
stat.UpperVal
stat.$\bar{x}$
stat.$\bar{x}$1
stat.$\bar{x}$2
stat.$\bar{x}$Diff
stat.$\bar{x}$List
stat.XReg
stat.XVal
stat.XValList
stat.$\bar{y}$
stat.$\hat{y}$
stat.$\hat{y}$List
stat.YReg

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

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.

Voir l'exemple donné pour **stat.results**.

stDevPop()

Catalogue >

stDevPop(*Liste***[,** *listeFréq***])** ⇒ *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*.

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 165.

En mode Angle en radians et en modes Auto :

$$\text{stDevPop}(\{a,b,c\}) \qquad \frac{\sqrt{2\cdot(a^2-a\cdot(b+c)+b^2-b\cdot c+c^2)}}{3}$$

$$\text{stDevPop}(\{1,2,5,-6,3,-2\}) \qquad \frac{\sqrt{465}}{6}$$

$$\text{stDevPop}(\{1.3,2.5,-6.4\},\{3,2,5\}) \qquad 4.11107$$

stDevPop(*Matrice1***[,** *matriceFréq***])** ⇒ *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 165.

$$\text{stDevPop}\left(\begin{bmatrix}1&2&5\\-3&0&1\\5&7&3\end{bmatrix}\right) \qquad \begin{bmatrix}\frac{4\cdot\sqrt{6}}{3} & \frac{\sqrt{78}}{3} & \frac{2\cdot\sqrt{6}}{3}\end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix}-1.2&5.3\\2.5&7.3\\6&-4\end{bmatrix},\begin{bmatrix}4&2\\3&3\\1&7\end{bmatrix}\right) \qquad \begin{bmatrix}2.52608 & 5.21506\end{bmatrix}$$

stDevSamp()

Catalogue >

stDevSamp(*Liste***[,** *listeFréq***])** ⇒ *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 165.

$$\text{stDevSamp}(\{a,b,c\}) \qquad \frac{\sqrt{3\cdot(a^2-a\cdot(b+c)+b^2-b\cdot c+c^2)}}{3}$$

$$\text{stDevSamp}(\{1,2,5,-6,3,-2\}) \qquad \frac{\sqrt{62}}{2}$$

$$\text{stDevSamp}(\{1.3,2.5,-6.4\},\{3,2,5\}) \qquad 4.33345$$

stDevSamp(*Matrice1***[,** *matriceFréq***])** ⇒ *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 165.

$$\text{stDevPop}\left(\begin{bmatrix}1&2&5\\-3&0&1\\5&7&3\end{bmatrix}\right) \qquad \begin{bmatrix}3.26599 & 2.94392 & 1.63299\end{bmatrix}$$

$$\text{stDevPop}\left(\begin{bmatrix}-1.2&5.3\\2.5&7.3\\6&-4\end{bmatrix},\begin{bmatrix}4&2\\3&3\\1&7\end{bmatrix}\right) \qquad \begin{bmatrix}2.52608 & 5.21506\end{bmatrix}$$

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

$i:=0$	0
Define $prog1()$=Prgm For i,1,10,1 If i=5 Stop EndFor EndPrgm	*Done*
$prog1()$	*Done*
i	5

Store

Voir → **(store)**, page 163.

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(*Matrice1***[, *colDébut*] [, *colDébut*] [, *ligneFin*] [, *colFin*])** ⇒ *matrice*

Donne la matrice spécifiée, extraite de *Matrice1*.

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 (Sigma)

Voir Σ**()**, page 156.

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.

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 165.

$\text{sum}(\{1,2,3,4,5\})$	15
$\text{sum}(\{a,2\cdot a,3\cdot a\})$	$6\cdot a$
$\text{sum}(\text{seq}(n,n,1,10))$	55
$\text{sum}(\{1,3,5,7,9\},3)$	21

sum()

Catalogue >

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.

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 165.

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

$\text{sum}\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix}$ → $\begin{bmatrix}12&15&18\end{bmatrix}$

$\text{sum}\left(\begin{bmatrix}1&2&3\\4&5&6\\7&8&9\end{bmatrix},2,3\right)$ → $\begin{bmatrix}11&13&15\end{bmatrix}$

sumIf()

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*.

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 165.

Remarque : voir également **countIf()**, page 26.

$\text{sumIf}(\{1,2,\boldsymbol{e},3,\pi,4,5,6\},2.5<?<4.5)$ → $\boldsymbol{e}+\pi+7$

$\text{sumIf}(\{1,2,3,4\},2<?<5,\{10,20,30,40\})$ → 70

sumSeq()

Voir Σ(), page 156.

system()

Catalogue >

system(*Eqn1* [**,** *Eqn2* [**,** *Eqn3* [**,** ...]]]**)**
system(*Expr1* [**,** *Expr2* [**,** *Expr3* [**,** ...]]]**)**

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.

$\text{solve}\left(\begin{cases}x+y=0\\x-y=8\end{cases},x,y\right)$ → $x=4$ and $y=-4$

T

T (transposée)

Catalogue >

Matrix1^{T} ⇒ *matrice*

Donne la transposée de la conjuguée de *Matrice1*.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant `@t`.

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

$$\begin{bmatrix}a&b\\c&d\end{bmatrix}^{T} \qquad \begin{bmatrix}a&c\\b&d\end{bmatrix}$$

$$\begin{bmatrix}1+i&2+i\\3+i&4+i\end{bmatrix}^{T} \qquad \begin{bmatrix}1-i&3-i\\2-i&4-i\end{bmatrix}$$

tan()

Touche trig

tan(*Expr1***)** ⇒ *expression*
tan(*Liste1***)** ⇒ *liste*

tan(*Expr1***)** donne la tangente de l'argument.

tan(*List1***)** donne la liste des tangentes 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 Angle sélectionné.

En mode Angle en degrés :

$$\tan\left(\frac{\pi}{4}^{r}\right) \qquad 1$$

$$\tan(45) \qquad 1$$

$$\tan(\{0,60,90\}) \qquad \{0,\sqrt{3},\text{undef}\}$$

En mode Angle en grades :

$$\tan\left(\frac{\pi}{4}^{r}\right) \qquad 1$$

$$\tan(50) \qquad 1$$

$$\tan(\{0,50,100\}) \qquad \{0,1,\text{undef}\}$$

En mode Angle en radians :

$$\tan\left(\frac{\pi}{4}\right) \qquad 1$$

$$\tan(45°) \qquad 1$$

$$\tan\left(\left\{\pi,\frac{\pi}{3},-\pi,\frac{\pi}{4}\right\}\right) \qquad \left\{0,\sqrt{3},0,1\right\}$$

tan(*matriceMatrice1***)** ⇒ *matriceCarrée*

Donne la tangente de la matrice *matriceCarrée1*. 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()**.

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

En mode Angle en radians :

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

$$\begin{bmatrix}-28.2912&26.0887&11.1142\\12.1171&-7.83536&-5.48138\\36.8181&-32.8063&-10.4594\end{bmatrix}$$

tan⁻¹()

Touche trig

tan⁻¹(*Expr1***)** ⇒ *expression*
tan⁻¹(*Liste1***)** ⇒ *liste*

tan⁻¹(*Expr1***)** donne l'arc tangente de *Expr1*.

tan⁻¹(*List1***)** donne la liste des arcs tangentes 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 **arctan(...)**.

En mode Angle en degrés :

$\tan^{-1}(1)$ → 45

En mode Angle en grades :

$\tan^{-1}(1)$ → 50

En mode Angle en radians :

$\tan^{-1}(\{0,0.2,0.5\})$ → $\{0,0.197396,0.463648\}$

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

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()**.

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

En mode Angle en radians :

$$\tan^{-1}\begin{bmatrix}1&5&3\\4&2&1\\6&-2&1\end{bmatrix} \rightarrow \begin{bmatrix}-0.083658&1.26629&0.62263\\0.748539&0.630015&-0.070012\\1.68608&-1.18244&0.455126\end{bmatrix}$$

tangentLine()

Catalogue >

tangentLine(*Expr1***,***Var***,***Point***)** ⇒ *expression*
tangentLine(*Expr1***,***Var***=***Point***)** ⇒ *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

$\text{tangentLine}\left(x^{\frac{1}{3}},x=0\right)$ → $x=0$

$\text{tangentLine}(\sqrt{x^2-4},x=2)$ → undef

$x:=3:\ \text{tangentLine}(x^2,x,1)$ → 5

tanh()

Catalogue >

tanh(*Expr1***)** ⇒ *expression*
tanh(*Liste1***)** ⇒ *liste*

tanh(*Expr1***)** donne la tangente hyperbolique de l'argument.

tanh(*Liste1***)** donne la liste des tangentes hyperboliques des éléments de *Liste1*.

$\tanh(1.2)$ → 0.833655

$\tanh(\{0,1\})$ → $\{0,\tanh(1)\}$

tanh(*matriceCarrée1***)** ⇒ *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.

En mode Angle en radians :

$$\tanh\begin{bmatrix}1&5&3\\4&2&1\\6&-2&1\end{bmatrix} \rightarrow \begin{bmatrix}-0.097966&0.933436&0.425972\\0.488147&0.538881&-0.129382\\1.28295&-1.03425&0.428817\end{bmatrix}$$

tanh⁻¹()

Catalogue >

tanh⁻¹(*Expr1***)** ⇒ *expression*
tanh⁻¹(*Liste1***)** ⇒ *liste*

tanh⁻¹(*Expr1***)** donne l'argument tangente hyperbolique de l'argument sous forme d'expression.

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 Format complexe Rectangulaire :

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

$$\tanh^{-1}(\{1,2.1,3\}) \qquad \left\{\text{undef}, 0.518046-1.5708\cdot i, \frac{\ln(2)}{2}-\frac{\pi}{2}\cdot i\right\}$$

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

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

$$\begin{bmatrix}-0.099353+0.164058\cdot i & 0.267834-1.4908\\ -0.087596-0.725533\cdot i & 0.479679-0.9473\\ 0.511463-2.08316\cdot i & -0.878563+1.7901\end{bmatrix}$$

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.

Comme illustré dans l'exemple ci-contre, le développement des routines de calcul du résultat donnée par taylor(...) peut réorganiser l'ordre des termes de sorte que le terme dominant ne soit pas le terme le plus à gauche.

$$\text{taylor}\left(e^{\sqrt{x}},x,2\right) \qquad \text{taylor}\left(e^{\sqrt{x}},x,2,0\right)$$

$$\text{taylor}\left(e^{t},t,4\right)\Big|t=\sqrt{x} \qquad \frac{x^2}{24}+\frac{x}{6}+\frac{\frac{3}{2}}{2}+\sqrt{x}+1$$

$$\text{taylor}\left(\frac{1}{x\cdot(x-1)},x,3\right) \qquad \text{taylor}\left(\frac{1}{x\cdot(x-1)},x,3,0\right)$$

$$\text{expand}\left(\frac{\text{taylor}\left(\frac{x}{x\cdot(x-1)},x,4\right)}{x},x\right) \qquad -x^3-x^2-x-\frac{1}{x}-1$$

$$\text{taylor}\left((1+e^{x})^2,x,2,1\right)$$
$$e\cdot(2\cdot e+1)\cdot(x-1)^2+(2\cdot e^2+2\cdot e)\cdot(x-1)+(e+1)^2$$

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 ≤ *upBound*), définissez *lowBound* = −∞.

tCollect()

Catalogue >

tCollect(*Expr1***)** ⇒ *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.

$\text{tCollect}(\cos(\alpha)^2)$ $\quad \frac{\cos(2\cdot\alpha)+1}{2}$

$\text{tCollect}(\sin(\alpha)\cdot\cos(\beta))$ $\quad \frac{\sin(\alpha-\beta)+\sin(\alpha+\beta)}{2}$

tExpand()

Catalogue >

tExpand(*Expr1***)** ⇒ *expression*

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 π/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.

$\text{tExpand}(\sin(3\cdot\varphi))$ $\quad 4\cdot\sin(\varphi)\cdot\cos(\varphi)^2-\sin(\varphi)$

$\text{tExpand}(\cos(\alpha-\beta))$ $\quad \cos(\alpha)\cdot\cos(\beta)+\sin(\alpha)\cdot\sin(\beta)$

Text

Catalogue >

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.

Si le programme nécessite une réponse saisie par l'utilisateur, voir **Request**, page 103 ou **RequestStr**, page 104.

Remarque : vous pouvez utiliser cette commande dans un programme créé par l'utilisateur, mais pas dans une fonction.

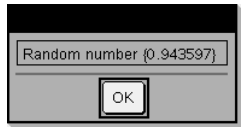
Définissez un programme qui marque une pause afin d'afficher cinq nombres aléatoires dans une boîte de dialogue. Dans le modèle Prgm...EndPrgm, validez chaque ligne en appuyant sur ↵ à la place de **enter**. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée**.

```
Define text_demo()=Prgm
  For i,1,5
    strinfo:="Random number " & string(rand(i))
    Text strinfo
  EndFor
EndPrgm
```

Exécutez le programme :

```
text_demo()
```

Exemple de boîte de dialogue :



Then

Voir **If,** page 58.

tInterval

Catalogue >

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 122.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 165.

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.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*[**,***Freq2*[**,***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 122.)

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 165.

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 *Liste 1* et *Liste 2*
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*

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **deltaTmpCnv(...)**.

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 :

_°C Celsius
_°F Fahrenheit
_°K Kelvin
_°R Rankine

Pour taper °, sélectionnez ce symbole dans le Jeu de symboles ou entrez **@d**.

Pour taper _, appuyez sur ctrl ␣.

Par exemple, 100_°C donne 212_°F.

Pour convertir un écart de température, utilisez **ΔtmpCnv()**.

tmpCnv(100·_°C,_°F)	212.·_°F
tmpCnv(32·_°F,_°C)	0.·_°C
tmpCnv(0·_°C,_°K)	273.15·_°K
tmpCnv(0·_°F,_°R)	459.67·_°R

Remarque : vous pouvez utiliser le Catalogue pour sélectionner des unités de température.

ΔtmpCnv()

Catalogue >

ΔtmpCnv(*Expr*_°*unitéTemp1*, _°*unitéTemp2***)**
⇒ *expression* _°*unitéTemp2*

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 :

_°C Celsius
_°F Fahrenheit
_°K Kelvin
_°R Rankine

Pour taper °, sélectionnez-le dans les symboles du Catalogue.

Pour taper _, appuyez sur ctrl ␣.

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()**.

Pour taper Δ, sélectionnez-le dans les symboles du Catalogue.

ΔtmpCnv(100·_°C,_°F)	180.·_°F
ΔtmpCnv(180·_°F,_°C)	100.·_°C
ΔtmpCnv(100·_°C,_°K)	100.·_°K
ΔtmpCnv(100·_°F,_°R)	100.·_°R
ΔtmpCnv(1·_°C,_°F)	1.8·_°F

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*.

$$\text{trace}\left(\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}\right) \qquad 15$$

$$\text{trace}\left(\begin{bmatrix}a & 0\\ 1 & a\end{bmatrix}\right) \qquad 2\cdot a$$

Try

Catalogue >

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 171.

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define prog1()=Prgm
                Try
                z:=z+1
                Disp "z incremented."
                Else
                Disp "Sorry, z undefined."
                EndTry
                EndPrgm
                                                    Done
z:=1:prog1()
                                          z incremented.
                                                    Done
DelVar z:prog1()
                                          Sorry, z undefined.
                                                    Done
```

Exemple 2

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.

$$eigenvals\left(\begin{bmatrix}-3\\ -41\\ 5\end{bmatrix},\begin{bmatrix}-1 & 2 & -3.1\end{bmatrix}\right)$$

$$eigenvals\left(\begin{bmatrix}1 & 2 & 3\end{bmatrix},\begin{bmatrix}1\\ 2\end{bmatrix}\right)$$

Remarque : voir aussi **ClrErr**, page 19 et **PassErr**, page 90.

```
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

Catalogue >

tTest $\mu 0$,*Liste*[,*Fréq*[,*Hypoth*]]

(Entrée de liste de données)

tTest $\mu 0$,$\bar{x}$,*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 122.)

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 165.

Variable de sortie	Description
stat.t	$(\bar{x} - \mu 0) / (\text{stdev} / \text{sqrt}(n))$
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 *Liste*
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 122.)

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 165.

Variable de sortie	Description
stat.t	Valeur normale type calculée pour la différence des moyennes

Variable de sortie	Description
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 *Liste 1* et *Liste 2*
stat.sx1, stat.sx2	Écarts-types d'échantillon des séries de données dans *Liste 1* et *Liste 2*
stat.n1, stat.n2	Taille des échantillons
stat.sp	Écart-type du groupe. Calculé lorsque *Group*=1.

tvmFV()

Catalogue >

tvmFV(*N*,*I*,*PV*,*Pmt*,[*PpY*],[*CpY*],[*PmtAt*]**)** ⇒ *valeur*

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 135. Voir également **amortTbl()**, page 6.

tvmFV(120,5,0,-500,12,12) 77641.1

tvmI()

Catalogue >

tvmI(*N*,*PV*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)** ⇒ *valeur*

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 135. Voir également **amortTbl()**, page 6.

tvmI(240,100000,-1000,0,12,12) 10.5241

tvmN()

Catalogue >

tvmN(*I*,*PV*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)** ⇒ *valeur*

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 135. Voir également **amortTbl()**, page 6.

tvmN(5,0,-500,77641,12,12) 120.

tvmPmt()

Catalogue >

tvmPmt(*N*,*I*,*PV*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)** ⇒ *valeur*

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 135. Voir également **amortTbl()**, page 6.

tvmPmt(60,4,30000,0,12,12) -552.496

tvmPV()

Catalogue >

tvmPV(*N*,*I*,*Pmt*,*FV*,[*PpY*],[*CpY*],[*PmtAt*]**)** ⇒ *valeur*

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 135. Voir également **amortTbl()**, page 6.

tvmPV(48,4,-500,30000,12,12) -3426.7

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

Catalogue >

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

Calcule des statistiques pour deux variables. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 122.)

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.

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* a un élément vide correspondant dans l'ensemble des listes résultantes. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 165.

Variable de sortie	Description
stat.$\bar{x}$	Moyenne des valeurs x
stat.Σx	Somme des valeurs x
stat.Σx2	Somme des valeurs x2
stat.sx	Écart-type de l'échantillon de x
stat.σx	Écart-type de la population de x
stat.n	Nombre de points de données

Variable de sortie	Description
stat.$\bar{y}$	Moyenne des valeurs y
stat.Σy	Somme des valeurs y
stat.Σy^2	Somme des valeurs y2
stat.sy	Écart-type de y dans l'échantillon
stat.σy	Écart-type de population des valeurs de y
stat.Σxy	Somme des valeurs x • y
stat.r	Coefficient de corrélation
stat.MinX	Minimum des valeurs de x
stat.Q_1X	1er quartile de x
stat.MedianX	Médiane de x
stat.Q_3X	3ème quartile de x
stat.MaxX	Maximum des valeurs de x
stat.MinY	Minimum des valeurs de y
stat.Q_1Y	1er quartile de y
stat.MedY	Médiane de y
stat.Q_3Y	3ème quartile de y
stat.MaxY	Maximum des valeurs y
stat.$\Sigma(x-\bar{x})^2$	Somme des carrés des écarts par rapport à la moyenne de x
stat.$\Sigma(y-\bar{y})^2$	Somme des carrés des écarts par rapport à la moyenne de y

U

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.

$\text{unitV}\left(\begin{bmatrix} a & b & c \end{bmatrix}\right)$

$$\begin{bmatrix} \frac{a}{\sqrt{a^2+b^2+c^2}} & \frac{b}{\sqrt{a^2+b^2+c^2}} & \frac{c}{\sqrt{a^2+b^2+c} \end{bmatrix}$$

$\text{unitV}\left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}\right)$ $\quad \begin{bmatrix} \frac{\sqrt{6}}{6} & \frac{\sqrt{6}}{3} & \frac{\sqrt{6}}{6} \end{bmatrix}$

$\text{unitV}\left(\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}\right)$

$$\begin{bmatrix} \frac{\sqrt{14}}{14} \\ \frac{\sqrt{14}}{7} \\ \frac{3\cdot\sqrt{14}}{14} \end{bmatrix}$$

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

unLock

Catalogue >

unLock *Var1* [**,** *Var2*] [**,** *Var3*] ...
unLock *Var***.**

Déverrouille les variables ou les groupes de variables spécifiés. Les variables verrouillées ne peuvent être ni modifiées ni supprimées.

Voir **Lock**, page 71 et **getLockInfo()**, page 55.

a:=65	65
Lock a	*Done*
getLockInfo(a)	1
a:=75	"Error: Variable is locked."
DelVar a	"Error: Variable is locked."
Unlock a	*Done*
a:=75	75
DelVar a	*Done*

V

varPop()

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*.

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 165.

varPop({5,10,15,20,25,30})	$\frac{875}{12}$
Ans·1.	72.9167

varSamp()

Catalogue >

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 165.

$\text{varSamp}(\{a,b,c\})$

$$\frac{a^2-a\cdot(b+c)+b^2-b\cdot c+c^2}{3}$$

$\text{varSamp}(\{1,2,5,-6,3,-2\})$ $\quad \frac{31}{2}$

$\text{varSamp}(\{1,3,5\},\{4,6,2\})$ $\quad \frac{68}{33}$

varSamp(*Matrice1*[**,** *matriceFréq*]**)** ⇒ *matrice*

Donne un vecteur ligne contenant la variance d'échantillon de chaque colonne 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.

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 165.

$$\text{varSamp}\begin{bmatrix}1 & 2 & 5\\ -3 & 0 & 1\\ .5 & .7 & 3\end{bmatrix} \quad \begin{bmatrix}4.75 & 1.03 & 4\end{bmatrix}$$

$$\text{varSamp}\left(\begin{bmatrix}-1.1 & 2.2\\ 3.4 & 5.1\\ -2.3 & 4.3\end{bmatrix},\begin{bmatrix}6 & 3\\ 2 & 4\\ 5 & 1\end{bmatrix}\right) \quad \begin{bmatrix}3.91731 & 2.08411\end{bmatrix}$$

W

warnCodes()

Catalogue >

warnCodes(*Expr1***,** *VarÉtat***)** ⇒ *expression*

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 176.

$\text{warnCodes}\left(\text{solve}\left(\sin(10\cdot x)=\frac{x^2}{x},x\right),warn\right)$

$x=-0.84232$ or $x=-0.706817$ or $x=-0.285234$ or $x=0$

$warn$ $\quad \{10007,10009\}$

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

when()

Catalogue >

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.

$\text{when}(x<0,x+3)|x=5$ $\quad$ undef

when()

Catalogue >

when() est utile dans le cadre de la définition de fonctions récursives.

$\text{when}(n>0, n\cdot factoral(n-1), 1) \rightarrow factoral(n)$

Done

$factoral(3)$	6
$3!$	6

While

Catalogue >

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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define sum_of_recip(n)=Func
                         Local i,tempsum
                         1→i
                         0→tempsum
                         While i≤n
                         tempsum+1/i→tempsum
                         i+1→i
                         EndWhile
                         Return tempsum
                         EndFunc
```

Done

$sum\_of\_recip(3)$	$\frac{11}{6}$

X

xor

Catalogue >

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**, page 89.

true xor true	false
5>3 xor 3>5	true

xor

Catalogue >

Entier1 **xor** *Entier2* ⇒ *entier*

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 13.

Remarque : voir **or**, page 89.

En mode base Hex :

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

0h7AC36 xor 0h3D5F	0h79169

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.

z

zeros()

Catalogue >

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***)**.

$$\text{zeros}\left(a\cdot x^2+b\cdot x+c,x\right)$$
$$\left\{\frac{\sqrt{b^2-4\cdot a\cdot c}-b}{2\cdot a},\frac{-\left(\sqrt{b^2-4\cdot a\cdot c}+b\right)}{2\cdot a}\right\}$$

$$a\cdot x^2+b\cdot x+c\,|\,x=Ans[2] \qquad 0$$

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*.

$$\text{exact}\left(\text{zeros}\left(a\cdot\left(e^x+x\right)\cdot\left(\text{sign}\left(x\right)-1\right),x\right)\right) \qquad \{\,\}$$
$$\text{exact}\left(\text{solve}\left(a\cdot\left(e^x+x\right)\cdot\left(\text{sign}\left(x\right)-1\right)=0,x\right)\right)$$
$$e^x+x=0 \text{ or } x>0 \text{ or } a=0$$

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

zeros({*Expr1***,** *Expr2***},**
{*VarOuInit1***,** *VarOuInit2* **[,** … **]})** ⇒ *matrice*

Donne les zéros réels possibles du système d'expressions algébriques, où chaque *VarOuInit* spécifie une inconnue dont vous recherchez la valeur.

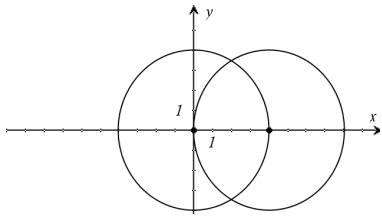
Vous pouvez également spécifier une condition initiale pour les variables. Chaque *VarOuInit* doit utiliser le format suivant :

variable
– ou –
variable = nombre *réel ou nonréel*

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.

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*].

$$\text{zeros}\left(\left\{x^2+y^2-r^2,(x-r)^2+y^2-r^2\right\},\{x,y\}\right)$$
$$\begin{bmatrix} \frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2} \\ \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} \end{bmatrix}$$

Extraction ligne 2 :

$$Ans[2] \qquad \begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} \end{bmatrix}$$

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.

$$\text{zeros}\left(\left\{x^2+y^2-r^2,(x-r)^2+y^2-r^2\right\},\{x,y,z\}\right)$$
$$\begin{bmatrix} \frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2} & c1 \\ \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} & c1 \end{bmatrix}$$

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.

$$\text{zeros}\left(\left\{x+e^z\cdot y-1,x-y-\sin(z)\right\},\{x,y\}\right)$$
$$\begin{bmatrix} \frac{e^z\cdot\sin(z)+1}{e^z+1} & \frac{-(\sin(z)-1)}{e^z+1} \end{bmatrix}$$

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.

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.

$$\text{zeros}\left(\left\{e^z\cdot y-1,-y-\sin(z)\right\},\{y,z\}\right)$$
$$\begin{bmatrix} 0.041458 & 3.18306 \\ 0.001871 & 6.28131 \\ 2.812\text{E}{-10} & 21.9911 \end{bmatrix}$$

$$\text{zeros}\left(\left\{e^z\cdot y-1,-y-\sin(z)\right\},\{y,z=2\cdot\pi\}\right)$$
$$\begin{bmatrix} 0.001871 & 6.28131 \end{bmatrix}$$

zInterval

Catalogue >

zInterval σ,*Liste*[,*Fréq*[,*CLevel*]]

(Entrée de liste de données)

zInterval σ,$\bar{x}$,*n* [,*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 122.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides" , page 165.

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 *Liste*

zInterval_1Prop

Catalogue >

zInterval_1Prop *x*,*n* [,*CLevel*]

Calcule un intervalle de confiance *z* pour une proportion. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 122.)

x est un entier non négatif.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides" , page 165.

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

Catalogue >

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 122.)

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 165.

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
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 σ_1,σ_2 ,*Liste1*,*Liste2*[,*Fréq1*[,*Fréq2*,[*CLevel*]]]

(Entrée de liste de données)

zInterval_2Samp σ_1,σ_2,$\bar{x}1$,*n1*,$\bar{x}2$,*n2*[,*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 122.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides" , page 165.

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.σx1, stat.σx2	Écarts-types d'échantillon pour *Liste 1* et *Liste 2*
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 *Liste 1* et *Liste 2*

zTest

Catalogue >

zTest $\mu 0$,σ,*Liste*,[*Fréq*[,*Hypoth*]]

(Entrée de liste de données)

zTest $\mu 0$,σ,$\bar{x}$,*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 122.)

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 165.

Variable de sortie	Description
stat.z	$(\bar{x} - \mu 0) / (\sigma / \text{sqrt}(n))$
stat.P Value	Plus petite probabilité permettant de rejeter l'hypothèse nulle
stat.$\bar{x}$	Moyenne d'échantillon de la série de données dans *Liste*
stat.sx	Écart-type d'échantillon de la série de données Uniquement donné pour l'entrée *Data*.
stat.n	Taille de l'échantillon

zTest_1Prop

Catalogue >

zTest_1Prop *p0*,*x*,*n*[,*Hypoth*]

Effectue un test *z* pour une proportion unique. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 122.)

x est un entier non négatif.

Test de $H_0 : p = p0$, en considérant que :

Pour $H_a : p > p0$, définissez *Hypoth*>0
Pour $H_a : p \neq p0$ *(par défaut)*, définissez *Hypoth*=0
Pour $H_a : p < p0$, définissez *Hypoth*<0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 165.

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

Catalogue >

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 122.)

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* ≠ *p2 (par défaut)*, définissez *Hypoth*=0
Pour H_a : *p* < *p0*, définissez *Hypoth*<0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides" , page 165.

Variable de sortie	Description
stat.z	Valeur normale type calculée pour la différence des proportions
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 σ_1,σ_2,*Liste1*,*Liste2*[,*Fréq1*[,*Fréq2*[,*Hypoth*]]]

(Entrée de liste de données)

zTest_2Samp σ_1,σ_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 122.)

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 165.

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 *Liste 1* et *Liste 2*
stat.sx1, stat.sx2	Écarts-types d'échantillon des séries de données dans *Liste 1* et *Liste 2*
stat.n1, stat.n2	Taille des échantillons

Symboles

+ (somme) — **Touche** +

Expr1 **+** *Expr2* ⇒ *expression*

Donne la somme des deux arguments.

56 → 56

$56+4$ → 60

$60+4$ → 64

$64+4$ → 68

$68+4$ → 72

Liste1 + *Liste2* ⇒ *liste*
Matrice1 + *Matrice2* ⇒ *matrice*

Donne la liste (ou la matrice) contenant les sommes des éléments correspondants de *Liste1* et *Liste2* (ou *Matrice1* et *Matrice2*).

Les arguments doivent être de même dimension.

$\left\{22,\pi,\frac{\pi}{2}\right\} \rightarrow l1$ → $\left\{22,\pi,\frac{\pi}{2}\right\}$

$\left\{10,5,\frac{\pi}{2}\right\} \rightarrow l2$ → $\left\{10,5,\frac{\pi}{2}\right\}$

$l1+l2$ → $\{32,\pi+5,\pi\}$

$Ans+\{\pi,-5,-\pi\}$ → $\{\pi+32,\pi,0\}$

$\begin{bmatrix} a & b \\ c & d \end{bmatrix}+\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ → $\begin{bmatrix} a+1 & b \\ c & d+1 \end{bmatrix}$

Expr + *Liste1* ⇒ *liste*
Liste1 + *Expr* ⇒ *liste*

Donne la liste contenant les sommes de *Expr* et de chaque élément de *Liste1*.

$15+\{10,15,20\}$ → $\{25,30,35\}$

$\{10,15,20\}+15$ → $\{25,30,35\}$

Expr + *Matrice1* ⇒ *matrice*
Matrice1 + *Expr* ⇒ *matrice*

Donne la matrice obtenue en ajoutant *Expr* à chaque élément de la diagonale de *Matrice1*. *Matrice1* doit être carrée.

Remarque : utilisez **.+** pour ajouter une expression à chaque élément de la matrice.

$20+\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ → $\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$

−(soustraction) — **Touche** −

Expr1 − *Expr2* ⇒ *expression*

Donne la différence de *Expr1* et de *Expr2*.

$6-2$ → 4

$\pi-\frac{\pi}{6}$ → $\frac{5\cdot\pi}{6}$

Liste1 − *Liste2* ⇒ *liste*
Matrice1 − *Matrice2* ⇒ *matrice*

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.

$\left\{22,\pi,\frac{\pi}{2}\right\}-\left\{10,5,\frac{\pi}{2}\right\}$ → $\{12,\pi-5,0\}$

$[3\ \ 4]-[1\ \ 2]$ → $[2\ \ 2]$

Expr − *Liste1* ⇒ *liste*
Liste1 − *Expr* ⇒ *liste*

Soustrait chaque élément de *Liste1* de *Expr* ou soustrait *Expr* de chaque élément de *Liste1* et donne la liste de résultats obtenue.

$15-\{10,15,20\}$ → $\{5,0,-5\}$

$\{10,15,20\}-15$ → $\{-5,0,5\}$

−(soustraction)

Touche −

Expr − *Matrice1* ⇒ *matrice*

Matrice1 − *Expr* ⇒ *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 .− pour soustraire une expression à chaque élément de la matrice.

$20-\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ → $\begin{bmatrix}19 & -2\\ -3 & 16\end{bmatrix}$

· (multiplication)

Touche ×

Expr1 · *Expr2* ⇒ *expression*

Donne le produit des deux arguments.

$2\cdot 3.45$ → 6.9

$x\cdot y\cdot x$ → $x^2\cdot y$

Liste1 · *Liste2* ⇒ *liste*

Donne la liste contenant les produits des éléments correspondants de *Liste1* et *Liste2*.

Les listes doivent être de même dimension.

$\{1.,2,3\}\cdot\{4,5,6\}$ → $\{4.,10,18\}$

$\left\{\frac{2}{a},\frac{3}{2}\right\}\cdot\left\{a^2,\frac{b}{3}\right\}$ → $\left\{2\cdot a,\frac{b}{2}\right\}$

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*.

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}\cdot\begin{bmatrix}a & d\\ b & e\\ c & f\end{bmatrix}$ → $\begin{bmatrix}a+2\cdot b+3\cdot c & d+2\cdot e+3\cdot f\\ 4\cdot a+5\cdot b+6\cdot c & 4\cdot d+5\cdot e+6\cdot f\end{bmatrix}$

Expr · *Liste1* ⇒ *liste*

Liste1 · *Expr* ⇒ *liste*

Donne la liste des produits de *Expr* et de chaque élément de *Liste1*.

$\pi\cdot\{4,5,6\}$ → $\{4\cdot\pi,5\cdot\pi,6\cdot\pi\}$

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.

$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\cdot 0.01$ → $\begin{bmatrix}0.01 & 0.02\\ 0.03 & 0.04\end{bmatrix}$

$\lambda\cdot\text{identity}(3)$ → $\begin{bmatrix}\lambda & 0 & 0\\ 0 & \lambda & 0\\ 0 & 0 & \lambda\end{bmatrix}$

/ (division)

Touche ÷

Expr1 / *Expr2* ⇒ *expression*

Donne le quotient de *Expr1* par *Expr2*.

Remarque : voir aussi **Modèle Fraction**, page 1.

$\frac{2}{3.45}$ → $.57971$

$\frac{x^3}{x}$ → x^2

/ (division)

Touche ÷

Liste1 / *Liste2* ⇒ *liste*

Donne la liste contenant les quotients de Liste1 *par Liste2*.

Les listes doivent être de même dimension.

$$\frac{\{1.,2,3\}}{\{4,5,6\}} \qquad \left\{0.25,\frac{2}{5},\frac{1}{2}\right\}$$

Expr / *Liste1* ⇒ *liste*
Liste1 / *Expr* ⇒ *liste*

Donne la liste contenant les quotients de *Expr* par *Liste1* ou de *Liste1* par *Expr*.

$$\frac{a}{\{3,a,\sqrt{a}\}} \qquad \left\{\frac{a}{3},1,\sqrt{a}\right\}$$

$$\frac{\{a,b,c\}}{a\cdot b\cdot c} \qquad \left\{\frac{1}{b\cdot c},\frac{1}{a\cdot c},\frac{1}{a\cdot b}\right\}$$

Matrice1 / *Expr* ⇒ *matrice*

Donne la matrice contenant les quotients des éléments de *Matrice1*/*Expression*.

Remarque : Utilisez **.** / pour diviser une expression par chaque élément.

$$\frac{\begin{bmatrix}a & b & c\end{bmatrix}}{a\cdot b\cdot c} \qquad \begin{bmatrix}\frac{1}{b\cdot c} & \frac{1}{a\cdot c} & \frac{1}{a\cdot b}\end{bmatrix}$$

^ (puissance)

Touche ^

Expr1 ^ *Expr2* ⇒ *expression*
Liste1 ^ *Liste2* ⇒ *liste*

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.

$$4^2 \qquad 16$$

$$\{a,2,c\}^{\{1,b,3\}} \qquad \{a,2^b,c^3\}$$

Expr ^ *Liste1* ⇒ *liste*

Donne *Expr* élevé à la puissance des éléments de *Liste1*.

$$p^{\{a,2,-3\}} \qquad \left\{p^a,p^2,\frac{1}{p^3}\right\}$$

List1 ^ *Expr* ⇒ *liste*

Donne les éléments de *Liste1* élevés à la puissance de l'*expression*.

$$\{1,2,3,4\}^{-2} \qquad \left\{1,\frac{1}{4},\frac{1}{9},\frac{1}{16}\right\}$$

matriceCarrée1 ^ *entier* ⇒ *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.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^2 \qquad \begin{bmatrix}7 & 10\\ 15 & 22\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-2} \qquad \begin{bmatrix}\frac{11}{2} & \frac{-5}{2}\\ \frac{-15}{4} & \frac{7}{4}\end{bmatrix}$$

x^2 (carré)

Touche x^2

Expr1 $^2 \Rightarrow$ *expression*
Donne le carré de l'argument.

Liste1 $^2 \Rightarrow$ *liste*

Donne la liste comportant les carrés des éléments de *Liste1*.

matriceCarrée1 $^2 \Rightarrow$ *matrice*

Donne le carré de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du carré de chaque élément. Utilisez .^2 pour calculer le carré de chaque élément.

4^2 → 16

$\{2,4,6\}^2$ → $\{4,16,36\}$

$\begin{bmatrix}2&4&6\\3&5&7\\4&6&8\end{bmatrix}^2$ → $\begin{bmatrix}40&64&88\\49&79&109\\58&94&130\end{bmatrix}$

$\begin{bmatrix}2&4&6\\3&5&7\\4&6&8\end{bmatrix}.^\wedge 2$ → $\begin{bmatrix}4&16&36\\9&25&49\\16&36&64\end{bmatrix}$

.+ (addition élément par élément)

Touches . +

Matrice1 .+ *Matrice2* $\Rightarrow$ *matrice*
Expr .+ *Matrice1* $\Rightarrow$ *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*.

$\begin{bmatrix}a&2\\b&3\end{bmatrix}.+\begin{bmatrix}c&4\\5&d\end{bmatrix}$ → $\begin{bmatrix}a+c&6\\b+5&d+3\end{bmatrix}$

$x.+\begin{bmatrix}c&4\\5&d\end{bmatrix}$ → $\begin{bmatrix}x+c&x+4\\x+5&x+d\end{bmatrix}$

.− (soustraction élément par élément)

Touches . −

Matrice1 .− *Matrice2* $\Rightarrow$ *matrice*
Expr .− *Matrice1* $\Rightarrow$ *matrice*

Matrice1 .− *Matrice2* donne la matrice obtenue en calculant la différence entre chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

Expr .− *Matrice1* donne la matrice obtenue en calculant la différence de *Expr* et de chaque élément de *Matrice1*.

$\begin{bmatrix}a&2\\b&3\end{bmatrix}.-\begin{bmatrix}c&4\\d&5\end{bmatrix}$ → $\begin{bmatrix}a-c&-2\\b-d&-2\end{bmatrix}$

$x.-\begin{bmatrix}c&4\\d&5\end{bmatrix}$ → $\begin{bmatrix}x-c&x-4\\x-d&x-5\end{bmatrix}$

.· (multiplication élément par élément)

Touches . ×

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*.

$\begin{bmatrix}a&2\\b&3\end{bmatrix}.\cdot\begin{bmatrix}c&4\\5&d\end{bmatrix}$ → $\begin{bmatrix}a\cdot c&8\\5\cdot b&3\cdot d\end{bmatrix}$

$x.\cdot\begin{bmatrix}a&b\\c&d\end{bmatrix}$ → $\begin{bmatrix}a\cdot x&b\cdot x\\c\cdot x&d\cdot x\end{bmatrix}$

./ (division élément par élément)

Touches . ÷

Matrice1 **./** *Matrice2* ⇒ *matrice*

Expr **./** *Matrice1* ⇒ *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*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} ./ \left\langle \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \right\rangle \qquad \begin{bmatrix} \frac{a}{c} & \frac{1}{2} \\ \frac{b}{5} & \frac{3}{d} \end{bmatrix}$$

$$x ./ \left\langle \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \right\rangle \qquad \begin{bmatrix} \frac{x}{c} & \frac{x}{4} \\ \frac{x}{5} & \frac{x}{d} \end{bmatrix}$$

.^ (puissance élément par élément)

Touches . ^

Matrice1 **.^** *Matrice2* ⇒ *matrice*

Expr **.^** *Matrice1* ⇒ *matrice*

Matrice1 **.^** *Matrice2* donne la matrice obtenue en élevant chaque élément de *Matrice1* à la puissance de l'élément correspondant de *Matrice2*.

Expr **.^** *Matrice1* donne la matrice obtenue en élevant *Expr* à la puissance de chaque élément de *Matrice1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .^ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} a^c & 16 \\ b^5 & 3^d \end{bmatrix}$$

$$x .^ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix} \qquad \begin{bmatrix} x^c & x^4 \\ x^5 & x^d \end{bmatrix}$$

⁻(opposé)

Touche (−)

⁻*Expr1* ⇒ *expression*

⁻*Liste1* ⇒ *liste*

⁻*Matrice1* ⇒ *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
$-\{-1,0.4,1.2\text{E}19\}$	$\{0\text{b}1,-0.4,-1.2\text{E}19\}$
$-a\cdot -b$	$a\cdot b$

En mode base Bin :

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

0b100101▶Dec	37
−0b100101	
0b1111111111111111111111111111111▶Dec	
Ans▶Dec	−37

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

% (pourcentage)

Touches ctrl 🕮

Expr1 **%** ⇒ *expression*
Liste1 **%** ⇒ *liste*
Matrice1 **%** ⇒ *matrice*

Donne $\frac{argument}{100}$

Dans le cas d'une liste ou d'une matrice, donne la liste ou la matrice obtenue en divisant chaque élément par 100.

Appuyez sur **Ctrl**+**Entrée** ctrl enter (Macintosh®: ⌘+**Enter**) pour évaluer :

13%	0.13

Appuyez sur **Ctrl**+**Entrée** ctrl enter (Macintosh®: ⌘+**Enter**) pour évaluer :

$\{\{1,10,100\}\}\%$	$\{0.01,0.1,1.\}$

= (égal à)

Touche =

Expr1 **=** *Expr2* ⇒ *Expression booléenne*
Liste1 **=** *Liste2* ⇒ *Liste booléenne*
Matrice1 **=** *Matrice2* ⇒ *Matrice 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.

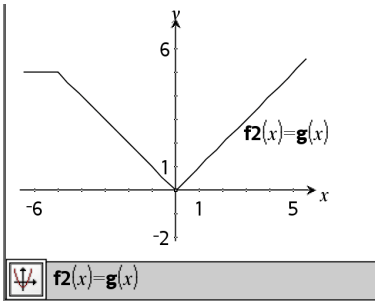
Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Remarque pour la saisie des données de l'exemple : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

Exemple de fonction qui utilise les symboles de test mathématiques : =, ≠, <, ≤, >, ≥

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

Résultat de la représentation graphique de g(x)



≠ (différent de)

Touches ctrl =

Expr1 ≠ *Expr2* ⇒ *Expression booléenne*
Liste1 ≠ *Liste2* ⇒ *Liste booléenne*
Matrice1 ≠ *Matrice2* ⇒ *Matrice 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.

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 **/=**

Voir l'exemple fourni pour « = » (égal à).

< (inférieur à)

Touches ctrl =

Expr1 < *Expr2* ⇒ *Expression booléenne*
Liste1 < *Liste2* ⇒ *Liste booléenne*
Matrice1 < *Matrice2* ⇒ *Matrice 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.

Voir l'exemple fourni pour « = » (égal à).

≤ (inférieur ou égal à)

Touches ctrl =

Expr1 ≤ *Expr2* ⇒ *Expression booléenne*
Liste1 ≤ *Liste2* ⇒ *Liste booléenne*
Matrice1 ≤ *Matrice2* ⇒ *Matrice 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.

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 **<=**

Voir l'exemple fourni pour « = » (égal à).

> (supérieur à)

Touches ctrl =

Expr1 > *Expr2* ⇒ *Expression booléenne*
Liste1 > *Liste2* ⇒ *Liste booléenne*
Matrice1 > *Matrice2* ⇒ *Matrice 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.

Voir l'exemple fourni pour « = » (égal à).

≥ (supérieur ou égal à)

Touches ctrl =

Expr1 ≥ *Expr2* ⇒ *Expression booléenne*
Liste1 ≥ *Liste2* ⇒ *Liste booléenne*
Matrice1 ≥ *Matrice2* ⇒ *Matrice 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.

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 **>=**

Voir l'exemple fourni pour « = » (égal à).

⇒ (implication logique)

touches ctrl =

BooleanExpr1 ⇒ *BooleanExpr2* renvoie *expression booléenne*
BooleanList1 ⇒ *BooleanList2* renvoie *liste booléenne*
BooleanMatrix1 ⇒ *BooleanMatrix2* renvoie *matrice booléenne*
Integer1 ⇒ *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 **=>**

5>3 or 3>5	true
5>3 ⇒ 3>5	false
3 or 4	7
3 ⇒ 4	-4
{1,2,3} or {3,2,1}	{3,2,3}
{1,2,3} ⇒ {3,2,1}	{-1,-1,-3}

⇔ (équivalence logique, XNOR)

touches ctrl =

BooleanExpr1 ⇔ *BooleanExpr2* renvoie *expression booléenne*
BooleanList1 ⇔ *BooleanList2* renvoie *liste booléenne*
BooleanMatrix1 ⇔ *BooleanMatrix2* renvoie *matrice booléenne*
Integer1 ⇔ *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 **<=>**

5>3 xor 3>5	true
5>3 ⇔ 3>5	false
3 xor 4	7
3 ⇔ 4	-8
{1,2,3} xor {3,2,1}	{2,0,2}
{1,2,3} ⇔ {3,2,1}	{-3,-1,-3}

! (factorielle)

Touche ?!

*Expr1***!** ⇒ *expression*
*Liste1***!** ⇒ *liste*
*Matrice1***!** ⇒ *matrice*

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.

5!	120
{{5,4,3}}!	{120,24,6}
$\left\{\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\right\}!$	$\begin{bmatrix}1 & 2\\ 6 & 24\end{bmatrix}$

& (ajouter)

Touches ctrl

Chaîne1 **&** *Chaîne2* ⇒ *chaîne*

Donne une chaîne de caractères obtenue en ajoutant *Chaîne2* à *Chaîne1*.

"Hello "&"Nick"	"Hello Nick"

d() (dérivée)

Catalogue >

d*(Expr1*, *Var*[, *Ordre*]) ⇒ *expression***

d*(Liste1*, *Var*[, *Ordre*]) ⇒ *liste***

d*(Matrice1*, *Var*[, *Ordre*]) ⇒ *matrice***

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 4, **Dériviée seconde**, page 5 ou **Dérivée n-ième**, page 5.

$$\frac{d}{dx}\left(f(x)\cdot g(x)\right) \qquad \frac{d}{dx}\left(f(x)\right)\cdot g(x)+\frac{d}{dx}\left(g(x)\right)\cdot f(x)$$

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

$$\frac{d}{dx}\left(\left\{x^2,x^3,x^4\right\}\right) \qquad \left\{2\cdot x,3\cdot x^2,4\cdot x^3\right\}$$

∫() (intégrale)

Catalogue >

∫(*Expr1*, *Var*[, *Borne1*, *Borne2*]) ⇒ *expression***

∫(*Expr1*, *Var*[, *Constante*]) ⇒ *expression***

Affiche l'intégrale de *Expr1* pour la variable *Var* entre *Borne1* et *Borne2*.

Remarque : voir aussi le modèle **Intégrale définie** ou **indéfinie**, page 5.

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*.

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.

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

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

$$\int\left(a\cdot x^2,x,c\right) \qquad \frac{a\cdot x^3}{3}+c$$

∫() (intégrale)

Catalogue >

∫() 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.

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.

$$\int b\cdot e^{-x^2}+\frac{a}{x^2+a^2}\,dx \quad b\cdot\int e^{-x^2}\,dx+\tan^{-1}\left(\frac{x}{a}\right)$$

Appuyez sur **Ctrl**+**Entrée** ctrl enter (Macintosh®: ⌘+**Enter**) pour évaluer :

$$\int_{-1}^{1} e^{-x^2}\,dx \qquad 1.49365$$

∫() 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 83.

$$\int_0^a\int_0^x \ln(x+y)\,dy\,dx$$

$$\frac{a^2\cdot\ln(a)}{2}+a^2\cdot\left(\ln(2)-\frac{3}{4}\right)$$

√() (racine carrée)

Touches ctrl x²

√ (*Expr1*) ⇒ *expression*
√ (*Liste1*) ⇒ *liste*

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.

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

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

Π() (prodSeq)

Catalogue >

Π(*Expr1*, *Var*, *Début*, *Fin*) ⇒ *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 (Π)**, page 4.

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

$\prod_{k=1}^{n}\left(k^2\right)$ → $(n!)^2$

$\prod_{n=1}^{5}\left(\left\{\frac{1}{n}, n, 2\right\}\right)$ → $\left\{\frac{1}{120}, 120, 32\right\}$

Π(*Expr1*, *Var*, *Début*, *Début*−1) ⇒ 1

Π(*Expr1*, *Var*, *Début*, *Fin*)
⇒ 1/Π(*Expr1*, *Var*, *Fin*+1, *Début*−1) if *Début* < *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.

$\prod_{k=4}^{3}\left(k\right)$ → 1

$\prod_{k=4}^{1}\left(\frac{1}{k}\right)$ → 6

$\prod_{k=4}^{1}\left(\frac{1}{k}\right)\cdot\prod_{k=2}^{4}\left(\frac{1}{k}\right)$ → $\frac{1}{4}$

Σ() (sumSeq)

Catalogue >

Σ(*Expr1*, *Var*, *Début*, *Fin*) ⇒ *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 4.

$\sum_{n=1}^{5}\left(\frac{1}{n}\right)$ → $\frac{137}{60}$

$\sum_{k=1}^{n}\left(k^2\right)$ → $\frac{n\cdot(n+1)\cdot(2\cdot n+1)}{6}$

$\sum_{n=1}^{\infty}\left(\frac{1}{n^2}\right)$ → $\frac{\pi^2}{6}$

Σ() (sumSeq)

Catalogue >

Σ(*Expr1***,** *Var***,** *Début***,** *Fin*−*1***)** ⇒ 0

Σ(*Expr1***,** *Var***,** *Début***,** *Fin***)**
⇒ −**Σ(***Expr1***,** *Var***,** *Fin*+*1***,** *Début*−**1)** if *Fin* < *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.

$\sum_{k=4}^{3}(k)$ 0

$\sum_{k=4}^{1}(k)$ −5

$\sum_{k=4}^{1}(k)+\sum_{k=2}^{4}(k)$ 4

ΣInt()

Catalogue >

ΣInt(*NPmt1***,** *NPmt2***,** *N***,** *I***,** *PV***,**[*Pmt*]**,** [*FV*]**,** [*PpY*]**,** [*CpY*]**,** [*PmtAt*]**,** [*valArrondi*]**)** ⇒ *valeur*

ΣInt(*NPmt1***,***NPmt2***,***tblAmortissement***)** ⇒ *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 135.

- 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.

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 6.

Remarque : voir également **ΣPrn()** ci dessous et **Bal()**, page 13.

ΣInt(1,3,12,4.75,20000,,12,12) −213.48

tbl:=amortTbl(12,12,4.75,20000,,12,12)

0	0.	0.	20000.
1	−77.49	−1632.43	18367.6
2	−71.17	−1638.75	16728.8
3	−64.82	−1645.1	15083.7
4	−58.44	−1651.48	13432.2
5	−52.05	−1657.87	11774.4
6	−45.62	−1664.3	10110.1
7	−39.17	−1670.75	8439.32
8	−32.7	−1677.22	6762.1
9	−26.2	−1683.72	5078.38
10	−19.68	−1690.24	3388.14
11	−13.13	−1696.79	1691.35
12	−6.55	−1703.37	−12.02

ΣInt(1,3,*tbl*) −213.48

Σ**Prn()**

Catalogue >

Σ Prn(*NPmt1***,** *NPmt2***,** *N***,** *I***,** *PV***,** [*Pmt*]**,** [*FV*]**,** [*PpY*]**,** [*CpY*]**,** [*PmtAt*]**,** [*valArrondi*]**)** ⇒ *valeur*

Σ Prn(*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 135.

- 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.

Σ Prn(*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 6.

Remarque : voir également **Σ Int()** ci-dessus et **Bal()**, page 13.

$\Sigma\text{Prn}(1,3,12,4.75,20000,,12,12)$ -4916.28

$tbl := \text{amortTbl}(12,12,4.75,20000,,12,12)$

0	0.	0.	20000.
1	-77.49	-1632.43	18367.57
2	-71.17	-1638.75	16728.82
3	-64.82	-1645.1	15083.72
4	-58.44	-1651.48	13432.24
5	-52.05	-1657.87	11774.37
6	-45.62	-1664.3	10110.07
7	-39.17	-1670.75	8439.32
8	-32.7	-1677.22	6762.1
9	-26.2	-1683.72	5078.38
10	-19.68	-1690.24	3388.14
11	-13.13	-1696.79	1691.35
12	-6.55	-1703.37	-12.02

$\Sigma\text{Prn}(1,3,tbl)$ -4916.28

(indirection)

Touches ctrl

*ChaîneNomVar*

Fait référence à la variable *ChaîneNomVar*. Permet d'utiliser des chaînes de caractères pour créer des noms de variables dans une fonction.

$\#("x"\&"y"\&"z")$ xyz

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 EE

*mantisse***E***exposant*

Saisit un nombre en notation scientifique. Ce nombre est interprété sous la forme *mantisse* × $10^{exposant}$.

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 2.3**E**4.

$23000.$	$23000.$
$2300000000.+4.1\text{E}15$	$4.1\text{E}15$
$3 \cdot 10^4$	30000

ᵍ (grades)

Touche π▸

*Expr1*ᵍ ⇒ *expression*
*Liste1*ᵍ ⇒ *liste*
*Matrice1*ᵍ ⇒ *matrice*

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 π/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**.

En mode Angle en degrés, grades ou radians :

$\cos\left(50^{g}\right)$ → $\frac{\sqrt{2}}{2}$

$\cos\left(\{0,100^{g},200^{g}\}\right)$ → $\{1,0,-1\}$

ʳ (radians)

Touche π▸

*Expr1*ʳ ⇒ *expression*
*Liste1*ʳ ⇒ *liste*
*Matrice1*ʳ ⇒ *matrice*

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/π.

En mode Angle en radians, donne l'argument inchangé.

En mode Angle en grades, multiplie l'argument par 200/π.

Conseil : utilisez 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.

Remarque : vous pouvez insérer ce symbole à partir du clavier de l'ordinateur en entrant **@r**.

En mode Angle en degrés, grades ou radians :

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

$\cos\left(\left\{0^{r},\frac{\pi}{12}^{r},-(\pi)^{r}\right\}\right)$ → $\left\{1,\frac{\left(\sqrt{3}+1\right)\cdot\sqrt{2}}{4},-1\right\}$

° (degré)

Touche π▸

Expr1° ⇒ *expression*
Liste1° ⇒ *liste*
Matrice1° ⇒ *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 π/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 :

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

En mode Angle en radians :

Appuyez sur **Ctrl**+**Entrée** ctrl enter (Macintosh®: ⌘+**Enter**) pour évaluer :

$\cos\left(\left\{0,\frac{\pi}{4},90^{\circ},30.12^{\circ}\right\}\right)$
→ $\{1.,0.707107,0.,0.864976\}$

°, ', '' (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.

Remarque : faites suivre ss.ss de deux apostrophes ('') et non de guillemets (").

En mode Angle en degrés :

$25°13'17.5''$	25.2215
$25°30'$	$\frac{51}{2}$

∠ (angle)

Touches ctrl 📖

[*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 @<.

En mode Angle en radians et avec le Format vecteur réglé sur : rectangulaire

$$[5 \quad \angle 60° \quad \angle 45°] \qquad \left[\frac{5\cdot\sqrt{2}}{4} \quad \frac{5\cdot\sqrt{6}}{4} \quad \frac{5\cdot\sqrt{2}}{2}\right]$$

cylindrique

$$[5 \quad \angle 60° \quad \angle 45°] \qquad \left[\frac{5\cdot\sqrt{2}}{2} \quad \angle\frac{\pi}{3} \quad \frac{5\cdot\sqrt{2}}{2}\right]$$

sphérique

$$[5 \quad \angle 60° \quad \angle 45°] \qquad \left[5 \quad \angle\frac{\pi}{3} \quad \angle\frac{\pi}{4}\right]$$

(*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 en mode Format complexe Rectangulaire :

$$5+3\cdot i-\left(10\angle\frac{\pi}{4}\right) \qquad 5-5\cdot\sqrt{2}+\left(3-5\cdot\sqrt{2}\right)\cdot i$$

Appuyez sur **Ctrl**+**Entrée** ctrl enter (Macintosh®: ⌘+**Enter**) pour évaluer :

$$5+3\cdot i-\left(10\angle\frac{\pi}{4}\right) \qquad -2.07107-4.07107\cdot i$$

' (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 165.

_ (trait bas considéré comme unité) **Touches** ctrl ␣

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.

$3 \cdot \_m \blacktriangleright \_ft$	$9.84252 \cdot \_ft$

Remarque : vous pouvez trouver le symbole de conversion, ▶, dans le Catalogue. Cliquez sur ∫Σ, puis sur **Opérateurs mathématiques**.

*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.

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.

En supposant que z est une variable non définie :

$\text{real}(z)$	z
$\text{real}(z\_)$	$\text{real}(z\_)$
$\text{imag}(z)$	0
$\text{imag}(z\_)$	$\text{imag}(z\_)$

▶ (conversion) **Touches** ctrl 📖

Expr_Unité1 ▶ *_Unité2* ⇒ *Expr_Unité2*

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 ctrl ␣.

$3 \cdot \_m \blacktriangleright \_ft$	$9.84252 \cdot \_ft$

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^ (*Liste1***)** ⇒ *liste*

Donne 10 élevé à la puissance de l'argument.

Dans le cas d'une liste, donne 10 élevé à la puissance des éléments de *Liste1*.

$10^{1.5}$	31.6228
$10^{\{0,-2,2,a\}}$	$\left\{1,\frac{1}{100},100,10^{a}\right\}$

10^()

Catalogue >

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*
Liste1 **^–1** ⇒ *liste*

Donne l'inverse de l'argument.

Dans le cas d'une liste, donne la liste des inverses des éléments de *Liste1*.

$$(3.1)^{-1} \qquad 0.322581$$

$$\{a,4,-0.1,x,-2\}^{-1} \qquad \left\{\frac{1}{a},\frac{1}{4},-10.,\frac{1}{x},\frac{-1}{2}\right\}$$

matriceCarrée1 **^–1** ⇒ *matriceCarrée*

Donne l'inverse de *matriceCarrée1*.

matriceCarrée1 doit être une matrice carrée non singulière.

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}-2 & 1\\ \frac{3}{2} & \frac{-1}{2}\end{bmatrix}$$

$$\begin{bmatrix}1 & 2\\ a & 4\end{bmatrix}^{-1} \qquad \begin{bmatrix}\frac{-2}{a-2} & \frac{1}{a-2}\\ \frac{a}{2\cdot(a-2)} & \frac{-1}{2\cdot(a-2)}\end{bmatrix}$$

| (opérateur "sachant que")

touches ctrl

Expr **|** *ExprBooléen1* [**and** *ExprBooléen2*]...
Expr **|** *ExprBooléen1* [**or** *ExprBooléen2*]...

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*.

$$x+1|x=3 \qquad 4$$

$$x+y|x=\sin(y) \qquad \sin(y)+y$$

$$x+y|\sin(y)=x \qquad x+y$$

$$x^3-2\cdot x+7\rightarrow f(x) \qquad Done$$

$$f(x)|x=\sqrt{3} \qquad \sqrt{3}+7$$

$$(\sin(x))^2+2\cdot\sin(x)-6|\sin(x)=d \qquad d^2+2\cdot d-6$$

| (opérateur "sachant que")

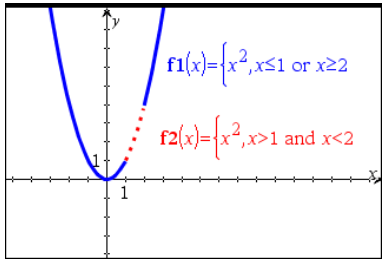
touches ctrl

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.

$\text{solve}(x^2-1=0, x) \mid x>0 \text{ and } x<2$ → $x=1$

$\sqrt{x}\cdot\sqrt{\frac{1}{x}} \mid x>0$ → 1

$\sqrt{x}\cdot\sqrt{\frac{1}{x}}$ → $\sqrt{\frac{1}{x}}\cdot\sqrt{x}$



Les exclusions utilisent l'opérateur « différent de » (/= ou ≠) 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.

$\text{solve}(x^2-1=0, x) \mid x\neq 1$ → $x=-1$

→ (stocker)

Touche ctrl var

Expr → *Var*
Liste → *Var*
Matrice → *Var*
Expr → *Fonction*(*Param1*,...)
Liste → *Fonction*(*Param1*,...)
Matrice → *Fonction*(*Param1*,...)

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`.

$\frac{\pi}{4} \to myvar$ → $\frac{\pi}{4}$

$2\cdot\cos(x) \to y1(x)$ → *Done*

$\{1,2,3,4\} \to lst5$ → $\{1,2,3,4\}$

$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} \to matg$ → $\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$

"Hello" → *str1* → "Hello"

:= (assigner)

Touches ctrl [x]

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*

$lst5 := \{1,2,3,4\}$ → $\{1,2,3,4\}$

$matg := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ → $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

$str1 :=$ "Hello" → "Hello"

© (commentaire)

Touches ctrl [ ]

© [*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 : dans l'application Calculs de l'unité nomade, vous pouvez entrer des définitions sur plusieurs lignes en appuyant sur ↵ à la place de enter à chaque fin de ligne. Sur le clavier de l'ordinateur, maintenez enfoncée la touche **Alt** tout en appuyant sur **Entrée (Enter)**.

```
Define g(n)=Func
    © Declare variables
    Local i,result
    result:=0
    For i,1,n,1 ©Loop n times
    result:=result+i^2
    EndFor
    Return result
    EndFunc
```

Done

$g(3)$ → 14

0b, 0h

Touches 0 B, **touches** 0 H

0b *nombreBinaire*
0h *nombreHexadécimal*

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).

Le résultat est affiché en fonction du mode Base utilisé.

En mode base Dec :

0b10+0hF+10 → 27

En mode base Bin :

0b10+0hF+10 → 0b11011

En mode base Hex :

0b10+0hF+10 → 0h1B

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 Tableur 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 36 et **isVoid()**, page 62.

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 ctrl ␣.

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.

$\lvert \_ \rvert$	_
$\gcd(100,\_)$	_
$3+\_$	_
$\{5,\_,10\}-\{3,6,9\}$	$\{2,\_,1\}$

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
$\text{cumulativeSum}(\{1,2,\_,4,5\})$	$\{1,3,\_,7,12\}$
$\text{cumulativeSum}\left(\begin{bmatrix}1 & 2\\ 3 & \_\\ 5 & 6\end{bmatrix}\right)$	$\begin{bmatrix}1 & 2\\ 4 & \_\\ 9 & 8\end{bmatrix}$

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	$\{5,3,2,1,4\}$

Arguments de liste contenant des éléments vides(continued)

Dans les regressions, la présence d'un élément vide dans la liste X ou Y génère un élément vide correspondant dans le résidu.

l1:={1,2,3,4,5}: *l2*:={2,_,3,5,6.6}	{2,_,3,5,6.6}
LinRegMx *l1*,*l2*	*Done*
stat.Resid	{0.434286,_,-0.862857,-0.011429,0.44}
stat.XReg	{1.,_,3.,4.,5.}
stat.YReg	{2.,_,3.,5.,6.6}
stat.FreqReg	{1.,_,1.,1.,1.}

L'omission d'une catégorie dans les calculs de régression génère un élément vide correspondant dans le résidu.

l1:={1,3,4,5}: *l2*:={2,3,5,6.6}	{2,3,5,6.6}
cat:={"M","M","F","F"}: *incl*:={"F"}	{"F"}
LinRegMx *l1*,*l2*,1,*cat*,*incl*	*Done*
stat.Resid	{_,_,0.,0.}
stat.XReg	{_,_,4.,5.}
stat.YReg	{_,_,5.,6.6}
stat.FreqReg	{_,_,1.,1.}

Une fréquence 0 dans les calculs de régression génère un élement vide correspondant dans le résidu.

l1:={1,3,4,5}: *l2*:={2,3,5,6.6}	{2,3,5,6.6}
LinRegMx *l1*,*l2*,{1,0,1,1}	*Done*
stat.Resid	{0.069231,_,-0.276923,0.207692}
stat.XReg	{1.,_,4.,5.}
stat.YReg	{2.,_,5.,6.6}
stat.FreqReg	{1.,_,1.,1.}

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(...)`, ...
ΔList()	`deltaList(...)`
ΔtmpCnv()	`deltaTmpCnv(...)`

Sur le clavier de l'ordinateur

Pour saisir :	Utilisez le raccourci :
c1, ***c2***, ... (constantes)	`@c1`, `@c2`, ...
n1, ***n2***, ... (constantes entières)	`@n1`, `@n2`, ...

Pour saisir :	Utilisez le raccourci :
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→r et "r"→s1, donc #s1=10.

Opérateurs en suffixe

Les opérateurs en suffixe sont des opérateurs qui suivent immédiatement un argument, comme 5!, 25 % ou 60°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^(3^2) pour donner 512. Ce qui est différent de (2^3)^2, qui donne 64.

Négation

Pour saisir un nombre négatif, appuyez sur (−) 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.

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**, page 132.

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 If a<b génère cette erreur si a ou b n'est pas défini lorsque l'instruction If 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 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 esc ou on 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->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.

Code d'erreur	Description
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
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.
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.

Code d'erreur	Description
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 : • sto → • **:=** • **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.

Code d'erreur	Description
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
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 *xxx\yyy*, où : • La partie *xxx* du nom peut contenir de 1 à 16 caractères, et • la partie *yyy*, 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**, **:=** ou sto **→**. • 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 : • 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

Code d'erreur	Description
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.
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 domain() 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, voir **warnCodes()**, page 138.

Code d'avertissement	Message
10000	L'opération peut donner des solutions fausses.
10001	L'équation générée par dérivation peut être fausse.
10002	Solution incertaine
10003	Précision incertaine
10004	L'opération peut omettre des solutions.
10005	CSolve peut donner plus de zéros.
10006	Solve peut donner plus de zéros.
10007	Autres solutions possibles
10008	Le domaine du résultat peut être plus petit que le domaine de l'entrée.
10009	Le domaine du résultat peut être plus grand que le domaine de l'entrée.
10012	Calcul non réel
10013	∞^0 ou undef^0 remplacés par 1.
10014	undef^0 remplacé par 1.
10015	1^ou 1^undef remplacés par 1
10016	1^undef remplacé par 1

Code d'avertissement	Message
10017	Capacité remplacée par ∞ ou ⁻∞
10018	Requiert et retourne une valeur 64 bits.
10019	Ressources insuffisantes, la simplification peut être incomplète.
10020	L'argument de la fonction trigonométrique est trop grand pour une réduction fiable.
10007	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)
10021	Les données saisies comportent un paramètre non défini. Le résultat peut ne pas être valide pour toutes les valeurs possibles du paramètre.
10022	La spécification des bornes inférieure et supérieure peut donner une solution.
10023	Le scalaire a été multiplié par la matrice d'identité.
10024	Résultat obtenu en utilisant un calcule approché
10025	L'équivalence ne peut pas être vérifiée en mode EXACT.
10026	La contrainte peut être ignorée. Spécifiez la contrainte sous forme de type 'Constante avec symbole de test mathématique variable' "\" ou en combinant ces deux formes (par exemple, par exemple "x<3 et x>-12").

Informations générales

Informations sur les services et la garantie TI

Informations sur les produits et les services TI	Pour plus d'informations sur les produits et les services TI, contactez TI par e-mail ou consultez la pages du site Internet éducatif de TI. adresse e-mail : ti-cares@ti.com adresse internet : education.ti.com
Informations sur les services et le contrat de garantie	Pour plus d'informations sur la durée et les termes du contrat de garantie ou sur les services liés aux produits TI, consultez la garantie fournie avec ce produit ou contactez votre revendeur Texas Instruments habituel.

Index

Symboles

Nombre

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z