



TI-Nspire™ CX

Guide de référence

Pour en savoir plus sur la technologie TI, consultez l'aide en ligne à l'adresse
education.ti.com/eguide.

Informations importantes

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

© 2024 Texas Instruments Incorporated

Les produits peuvent varier légèrement des images fournies.

Table des matières

Modèles d'expression	1
Liste alphabétique	7
A	7
B	16
C	21
D	38
E	47
F	56
G	64
I	75
L	83
M	100
N	109
O	119
P	122
Q	129
R	132
S	149
T	170
U	183
V	183
W	184
X	187
Z	188
Symboles	194
TI-Nspire™ CX II - Commandes graphiques	219
Programmation en mode graphique	219
Écran de représentation graphique	219
Vue et paramètres par défaut	220
Messages d'erreur de l'écran graphique	221
Commandes non valides dans le mode graphique	221
C	223
D	224
F	228
G	230
P	231
P	233
U	235

Éléments vides	236
Raccourcis de saisie d'expressions mathématiques	238
Hiérarchie de l'EOS™ (Equation Operating System)	240
Fonctions de programmation TI-Basic sur TI-Nspire CX II	242
Auto-indentation dans l'Éditeur de programmes	242
Messages d'erreur améliorés pour TI-Basic	242
Constantes et valeurs	245
Codes et messages d'erreur	246
Codes et messages d'avertissement	255
Informations générales	257
Index	258

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 196.

Exemple :

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

Modèle Exposant

Touche **^**



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

Remarque : Voir aussi ^ (puissance), page 197.

Exemple :

$$2^3 \quad 8$$

Modèle Racine carrée

Touches **ctrl x²**



Remarque : Voir aussi √() (racine carrée), page 207.

Exemple :

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

Modèle Racine n-ième

Touches ctrl ^



Remarque : Voir aussi **root()**, page 145.

Exemple :

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

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

Modèle e Exposant

Touches ex



La base du logarithme népérien e élevée à une puissance

Remarque : Voir aussi **e^()**, page 47.

Exemple :

$$e^1 \quad 2.71828182846$$

Modèle Logarithme

Touches ctrl 10^x



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

Remarque : Voir aussi **log()**, page 95.

Exemple :

$$\log_{\frac{1}{4}}(2.) \quad 0.5$$

Modèle Fonction définie par morceaux (2 morceaux)

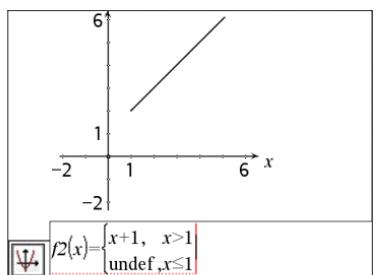
Catalogue > f(x)



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 123.

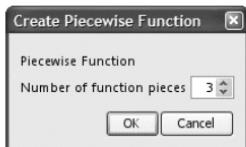
Exemple :



Modèle Fonction définie par morceaux (n morceaux)

Catalogue > 

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



Exemple :

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

Remarque : Voir aussi **piecewise()**, page 123.

Modèle Système de 2 équations

Catalogue > 



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

Remarque : Voir aussi **system()**, page 169.

Exemple :

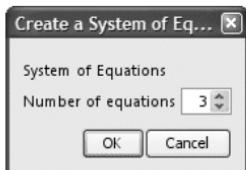
$$\text{solve}\left(\begin{cases} x+y=0 \\ x-y=5 \end{cases}, x, y\right) \quad x=\frac{5}{2} \text{ and } y=-\frac{5}{2}$$

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

Modèle Système de n équations

Catalogue > 

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



Exemple :

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

Remarque : Voir aussi **system()**, page 169.

Modèle Valeur absolue

Catalogue > 



Exemple :

Modèle Valeur absolue**Catalogue >** Remarque : Voir aussi **abs()**, page 7.

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

$$\{ 2, 3, 4, 64 \}$$

Modèle dd°mm'ss.ss"**Catalogue >** 

$$30^{\circ}15'10''$$

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 :

$$30^{\circ}15'10''$$

$$0.528011$$

Modèle Matrice (2 x 2)**Catalogue >** 

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

Crée une matrice de type 2 x 2.

Exemple :

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 5$$

$$\begin{bmatrix} 5 & 10 \\ 15 & 20 \end{bmatrix}$$

Modèle Matrice (1 x 2)**Catalogue >** 

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

Exemple :

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

$$\begin{bmatrix} 0 & 0 & -2 \end{bmatrix}$$

Modèle Matrice (2 x 1)**Catalogue >** 

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

Exemple :

$$\begin{bmatrix} 5 \\ 8 \end{bmatrix} \cdot 0.01$$

$$\begin{bmatrix} 0.05 \\ 0.08 \end{bmatrix}$$

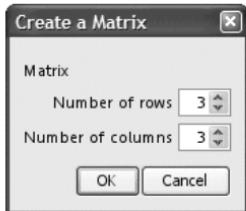
Modèle Matrice (m x n)**Catalogue >** 

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

Exemple :

$$\text{diag} \begin{pmatrix} 4 & 2 & 6 \\ 1 & 2 & 3 \\ 5 & 7 & 9 \end{pmatrix}$$

$$\begin{bmatrix} 4 & 2 & 9 \end{bmatrix}$$



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

Modèle Somme (Σ)

$$\sum_{\square=1}^{\square} (\square)$$

Remarque : voir aussi $\Sigma()$ (sumSeq), page 208.

Exemple :

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

Modèle Produit (Π)

$$\prod_{\square=1}^{\square} (\square)$$

Remarque : Voir aussi $\Pi()$ (prodSeq), page 208.

Exemple :

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

Modèle Dérivée première

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

Par exemple :

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

Modèle Dérivée première

Catalogue > 

Vous pouvez utiliser ce modèle pour calculer la dérivée première numérique en un point, à l'aide de méthodes de différenciation automatique.

Remarque : voir aussi **d()** (dérivée), page 206.

Modèle Dérivée seconde

Catalogue > 

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

Vous pouvez utiliser ce modèle pour calculer la dérivée seconde numérique en un point, à l'aide de méthodes de différenciation automatique.

Remarque : voir aussi **d()** (dérivée), page 206.

Modèle Intégrale définie

Catalogue > 

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

Vous pouvez utiliser ce modèle pour calculer l'intégrale définie numérique, en utilisant la même méthode que **nInt()**.

Remarque : voir aussi **nInt()**, page 113.

Par exemple :

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

18

Exemple :

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

333.333

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 194. 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(Valeur1)⇒valeur	
abs(Liste1)⇒liste	$\left \left[\frac{\pi}{2}, \frac{\pi}{3} \right] \right = \{1.5708, 1.0472\}$
abs(Matrice1)⇒matrice	$ 2^{-3} \cdot i = 3.60555$

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.

amortTbl()	Catalogue >
amortTbl(NPmt,N,I,PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [valArrondi])⇒matrice	amortTbl(12,60,10,5000,,,12,12) $\begin{bmatrix} 0 & 0. & 0. & 5000. \\ 1 & -41.67 & -64.57 & 4935.43 \\ 2 & -41.13 & -65.11 & 4870.32 \\ 3 & -40.59 & -65.65 & 4804.67 \\ 4 & -40.04 & -66.2 & 4738.47 \\ 5 & -39.49 & -66.75 & 4671.72 \\ 6 & -38.93 & -67.31 & 4604.41 \\ 7 & -38.37 & -67.87 & 4536.54 \\ 8 & -37.8 & -68.44 & 4468.1 \\ 9 & -37.23 & -69.01 & 4399.09 \\ 10 & -36.66 & -69.58 & 4329.51 \\ 11 & -36.08 & -70.16 & 4259.35 \\ 12 & -35.49 & -70.75 & 4188.6 \end{bmatrix}$
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 180.	

- 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 $\Sigma\text{Int()}$ et $\Sigma\text{Prn}()$, page 209 et $\text{bal}()$, page 16.

and

Valeur1 and Valeur2⇒*Expression booléenne*

Liste1 and Liste2⇒*Liste booléenne*

Matrice1 and Matrice2⇒*Matrice booléenne*

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

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.

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
--------------------	-------

and**Catalogue > **

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

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

En mode base Dec :

37 and 0b100

4

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

angle()**Catalogue > ****angle(*Valeur1*)⇒*valeur***

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

En mode Angle en degrés :

angle(0+2·i)

90

En mode Angle en grades :

angle(0+3·i)

100

En mode Angle en radians :

angle(1+i)

0.785398

angle({1+2·i,3+0·i,0-4·i})

{1.10715,0.,-1.5708}

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

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

ANOVA**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 164.)

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

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

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 164.)

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

Variable de sortie	Description
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()**approx(Valeur1)⇒valeur**

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**.

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.

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

►approxFraction()**Valeur ►approxFraction([tol])⇒valeur****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
0.8333333333333333 ►approxFraction(5.E-14)	
$\frac{5}{6}$	

approxRational()**approxRational(Valeur[, tol])⇒valeur****approxRational(Liste[, tol])⇒liste****approxRational(Matrice[, tol])⇒matrice**

approxRational(0.333, 5·10 ⁻⁵)	$\frac{333}{1000}$
approxRational({0.2, 0.33, 4.125}, 5.E-14)	$\left\{\frac{1}{5}, \frac{33}{100}, \frac{33}{8}\right\}$

approxRational()Catalogue > 

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

arccos()Voir $\cos^{-1}()$, page 29.**arccosh()**Voir $\cosh^{-1}()$, page 30.**arccot()**Voir $\cot^{-1}()$, page 31.**arccoth()**Voir $\coth^{-1}()$, page 32.**arccsc()**Voir $\csc^{-1}()$, page 35.**arccsch()**Voir $\csch^{-1}()$, page 35.**arcsec()**Voir $\sec^{-1}()$, page 149.**arcsech()**Voir $\sech^{-1}()$, page 150.**arcsin()**Voir $\sin^{-1}()$, page 159.

arcsinh()**Voir $\sinh^{-1}()$, page 160.****arctan()****Voir $\tan^{-1}()$, page 171.****arctanh()****Voir $\tanh^{-1}()$, page 172.****augment()****Catalogue >** **augment(Liste1, Liste2)⇒liste**

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

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

augment(Matrice1, Matrice2)⇒matrice

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

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

avgRC()**Catalogue >** **avgRC(Expr1, Var [=Valeur] [, Incrément])⇒expression**

x:=2 2

avgRC(Expr1, Var [=Valeur] [, Liste1])⇒liste

avgRC(x^2-x+2,x) 3.001

avgRC(Liste1, Var [=Valeur] [, Incrément])⇒liste

avgRC(x^2-x+2,x,,1) 3.1

avgRC(Matrice1, Var [=Valeur] [, Incrément])⇒matrice

avgRC(x^2-x+2,x,3) 6

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

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

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

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

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

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

B

bal()

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

bal(*NPmt,tblAmortissement*) \Rightarrow *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 180.

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 180.

- Si vous omettez *Pmt*, il prend par défaut la valeur **tvmPmt** (*N,I,PV,FV,PpY,CpY,PmtAt*).
- Si vous omettez *FV*, il prend par défaut la valeur *FV=0*.

bal (5,6,5.75,5000,,12,12)	833.11																												
<i>tbl:=amortTbl</i> (6,6,5.75,5000,,12,12)																													
<table border="1"> <tbody> <tr> <td>0</td><td>0.</td><td>0.</td><td>5000.</td></tr> <tr> <td>1</td><td>-23.35</td><td>-825.63</td><td>4174.37</td></tr> <tr> <td>2</td><td>-19.49</td><td>-829.49</td><td>3344.88</td></tr> <tr> <td>3</td><td>-15.62</td><td>-833.36</td><td>2511.52</td></tr> <tr> <td>4</td><td>-11.73</td><td>-837.25</td><td>1674.27</td></tr> <tr> <td>5</td><td>-7.82</td><td>-841.16</td><td>833.11</td></tr> <tr> <td>6</td><td>-3.89</td><td>-845.09</td><td>-11.98</td></tr> </tbody> </table>	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	
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, <i>tbl</i>)	1674.27
-----------------------------	---------

- 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 7.

Remarque : voir également $\Sigma\text{Int}()$ et $\Sigma\text{Prn}()$, page 209.

►Base2

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

0b *nombreBinaire*

0h *nombreHexadécimal*

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

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

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

256►Base2	0b10000000
-----------	------------

0h1F►Base2	0b11111
------------	---------

-1 s'affiche sous la forme

0hFFFFFFFFFFFFFFF 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

0h7FFFFFFFFFFFFF 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.

256►Base16

0h100

0b111100001111►Base16

0hFOF

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 17.

binomCdf()

binomCdf(*n,p*)⇒*liste*

binomCdf

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

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

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

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

binomPdf()

binomPdf(*n,p*)⇒*liste*

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

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

ceiling()**Catalogue >** **ceiling(*Valeur1*)** \Rightarrow *valeur*

ceiling(.456)

1.

Donne le plus petit entier \geq à l'argument.

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

Remarque : Voir aussi **floor()**.**ceiling(*Liste1*)** \Rightarrow *liste*

ceiling({-3.1,1,2.5})

{-3.,1,3.}

ceiling(*Matrice1*) \Rightarrow *matrice*ceiling($\begin{bmatrix} 0 & -3.2 \cdot i \\ 1.3 & 4 \end{bmatrix}$) $\begin{bmatrix} 0 & -3 \cdot i \\ 2. & 4 \end{bmatrix}$

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

centralDiff()**Catalogue >** **centralDiff(*Expr1*,*Var* [=*Valeur*],
[,*Pas*])** \Rightarrow *expression*centralDiff(cos(x),x)|x= $\frac{\pi}{2}$

-1.

**centralDiff(*Expr1*,*Var*
[,*Pas*]) | *Var*=*Valeur*** \Rightarrow *expression***centralDiff(*Expr1*,*Var* [=*Valeur*]
[,*Liste*])** \Rightarrow *liste***centralDiff(*Liste1*,*Var* [=*Valeur*]
[,*Incrément*])** \Rightarrow *liste***centralDiff(*Matrice1*,*Var* [=*Valeur*]
[,*Incrément*])** \Rightarrow *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 *ListeI* ou *MatriceI*, l'opération s'étend aux valeurs de la liste ou aux éléments de la matrice.

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

char()

char(Entier)⇒caractère

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

char(38)

"&"

char(65)

"A"

 χ^2 way

χ^2 way 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 164.)

Pour plus d'informations concernant les éléments vides dans une matrice, reportez-vous à "Éléments vides", page 236.

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

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

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

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

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

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

$\chi^2\text{GOF } \text{ListeObservée}, \text{ListeAttendue}, df$

$\text{chi2GOF } \text{ListeObservée}, \text{ListeAttendue}, df$

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 164.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

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

Variable de sortie	Description
stat.CompList	Contributions statistiques χ^2 élémentaires

$\chi^2\text{Pdf}()$

Catalogue > 

$\chi^2\text{Pdf}(ValX,dl) \Rightarrow \text{nombre}$ si $ValX$ est un nombre,
 liste si $XVal$ est une liste

$\text{chi2Pdf}(ValX,dl) \Rightarrow \text{nombre}$ si $ValX$ est un nombre,
 liste si $ValX$ est une liste

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

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

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 183.

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

ClrErr

Catalogue > 

ClrErr

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

Pour obtenir un exemple de **ClrErr**, reportez-vous à l'exemple 2 de la commande **Try**, page 176.

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 123 et **Try**, page 176.

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

colAugment()

**colAugment(*Matrice1*,
Matrice2)**⇒*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.

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

colDim()

colDim(*Matrice*)⇒*expression*

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

$$\text{colDim} \left[\begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \right] = 3$$

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

colNorm()**Catalogue >** **colNorm(*Matrice*)** \Rightarrow *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()**.

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

conj()**Catalogue >** **conj(*ValeurI*)** \Rightarrow *valeur*

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

conj(*ListeI*) \Rightarrow *liste***conj(*MatriceI*)** \Rightarrow *matrice*

Donne le conjugué de l'argument.

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

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

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

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*.

CopyVar

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.

Catalogue >

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

corrMat()

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

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

Catalogue >

cos()

cos(*Valeur1*) \Rightarrow *valeur*

cos(*Liste1*) \Rightarrow *liste*

Touche

En mode Angle en degrés :

cos()

Touche 

cos(Valeur1) calcule le cosinus de l'argument sous forme de valeur.

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.

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

En mode Angle en grades :

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

En mode Angle en radians :

$\cos\left(\frac{\pi}{4}\right)$	0.707107
$\cos(45^\circ)$	0.707107

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

matriceCarrée1 doit être diagonalisable et ne peut pas présenter de variables symboliques sans valeur affectée.

Formation des matrices :

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

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

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

$$\cos(B) =$$

cos()

Touche

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

Tous les calculs sont exécutés en virgule flottante.

cos⁻¹()

Touche

cos⁻¹(Valeur1)⇒valeur

En mode Angle en degrés :

cos⁻¹(Liste1)⇒liste**cos⁻¹(1)**

0.

cos⁻¹(Valeur1) donne l'arc cosinus de *Valeur1*.

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 grades :

cos⁻¹(0)

100.

En mode Angle en radians :

cos⁻¹({0,0.2,0.5})

{1.5708,1.36944,1.0472}

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

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

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

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

cosh()

Catalogue >

cosh(Valeur1)⇒valeur

En mode Angle en degrés :

cosh(Liste1)⇒liste

cosh()

Catalogue >

cosh(Valeur1) donne le cosinus hyperbolique de l'argument.

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.

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

1.74671e19

cosh⁻¹()

Catalogue >

cosh⁻¹(Valeur1)⇒valeur

cosh⁻¹(Liste1)⇒liste

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

0

$$\cosh^{-1}(\{1,2,1,3\})$$

{0,1.37286,1.76275}

cosh⁻¹(Valeur1) donne l'argument cosinus hyperbolique de l'argument.

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

(matriceCarrée1)⇒matriceCarrée

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

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

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

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

$$\begin{bmatrix} 2.52503+1.73485\cdot i & -0.009241-1.4908i \\ 0.486969-0.725533\cdot i & 1.66262+0.62349i \\ -0.322354-2.08316\cdot i & 1.26707+1.79018i \end{bmatrix}$$

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

cot()**Touche ****cot(Valeur1)⇒valeur****cot(Liste1)⇒liste**

Affiche la cotangente de *Valeur1* 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})

{0.642093,-0.584848,-7.01525}

cot⁻¹()**Touche ****cot⁻¹(Valeur1)⇒valeur****cot⁻¹(Liste1)⇒liste**

Donne l'arc cotangente de *Valeur1* 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)

0.785398

coth()**Catalogue > ****coth(Valeur1)⇒valeur****coth(Liste1)⇒liste**

coth(1.2)

1.19954

coth({1,3.2})

{1.31304,1.00333}

coth()

Catalogue >

Affiche la cotangente hyperbolique de *Valeur1* ou donne la liste des cotangentes hyperboliques des éléments de *Liste1*.

coth⁻¹()

Catalogue >

coth⁻¹(Valeur1)⇒valeur**coth⁻¹(Liste1)⇒liste**

Affiche l'argument cotangente hyperbolique de *Valeur1* ou retourne une liste comportant les arguments cotangente hyperbolique 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})	{-0.549306,0.518046,0.168236}

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 236.

count(2,4,6)	3
count({2,4,6})	3
count(2,{4,6},{8 10},[12 14])	7

countif()

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 236.

Remarque : voir également **sumIf()**, page 168 et **frequency()**, page 61.

Catalogue >

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

2

Compte le nombre d'éléments égaux à 3.

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

1

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

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

2

Compte 1 et 3.

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

3

Compte 3, 5 et 7.

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

4

Compte 1, 3, 7 et 9.

cPolyRoots()

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, dans sa forme développée.
N'utilisez pas les formats non développés comme y^2-y+1 ou $x\cdot x+2\cdot x+1$.

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

Catalogue >

polyRoots(y^3+1,y)

{-1}

cPolyRoots(y^3+1,y)

{-1,0.5-0.866025i,0.5+0.866025i}

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

{-1,-1}

cPolyRoots({1,2,1})

{-1,-1}

Remarque : voir aussi `polyRoots()`, page 125.

crossP()

Catalogue >

crossP(Liste1, Liste2)⇒liste

crossP({0,1,2,2,-5}, {1,-0,5,0})

$$\{-2.5, -5, -2.25\}$$

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.

crossP(Vecteur1, Vecteur2)⇒vecteur

`crossP([[1 2 3][4 5 6]])` $\begin{bmatrix} -3 & 6 & -3 \end{bmatrix}$

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

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.

csc()

Touche

csc(Valeur1)⇒valeur

En mode Angle en degrés :

csc(Liste1) \Rightarrow liste

$$\csc(45) = 1.41421$$

Affiche la cosécante de *Valeur1* ou donne une liste comportant les cosécantes de tous les éléments de *Liste1*.

En mode Angle en grades :

$$\csc(50) \quad 1.41421$$

En mode Angle en radians :

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

csc⁻¹()

Touche

csc⁻¹(Valeur1) ⇒ valeur

En mode Angle en degrés :

csc⁻¹(Liste1) ⇒ listecsc⁻¹(1)

90.

Affiche l'angle dont la cosécante correspond à *Valeur1* ou retourne la liste des arcs cosécante 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 **arccsc (...)**.

csch()

Catalogue >

csch(Valeur1) ⇒ valeur

csch(3) 0.099822

csch(Liste1) ⇒ liste

csch({1,2,1,4}) {0.850918,0.248641,0.036644}

Affiche la cosécante hyperbolique de *Valeur1* ou retourne la liste des cosécanes hyperboliques des éléments de *Liste1*.

csch⁻¹()

Catalogue >

csch⁻¹(Valeur1) ⇒ valeurcsch⁻¹(1) sinh⁻¹(1)**csch⁻¹(Liste1) ⇒ liste**csch⁻¹{1,2,1,3} {sinh⁻¹(1),0.459815,sinh⁻¹(1/3)}

Affiche l'argument cosécante hyperbolique de *Valeur1* ou donne la liste des arguments cosécanes hyperboliques de tous les éléments de *Liste1*.

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

CubicReg

Catalogue >

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

Effectue l'ajustement polynomial de degré $3y = a \cdot x^3 + b \cdot x^2 + c \cdot x + d$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

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

cumulativeSum()

Catalogue >

cumulativeSum(Liste1)⇒liste

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

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

cumulativeSum(Matrice1)⇒matrice

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

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

Un élément vide de *Liste1* ou *Matrice1* génère un élément vide dans la liste ou la matrice résultante. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236

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

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

Define g()=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

[2 2 3] ►Cylind [2.82843 ∠0.785398 3.]

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.

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*

En mode Angle en degrés :

Liste1 ►DD⇒*liste*

(1.5°)►DD 1.5°

Matrice1 ►DD⇒*matrice*

(45°22'14.3")►DD 45.3706°

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

{(45°22'14.3",60°0'0")}►DD

{45.3706°,60°}

En mode Angle en grades :

►DD**Catalogue >**

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).

1►DD

$$\frac{9}{10}$$

En mode Angle en radians :

(1.5)►DD

85.9437°

►Decimal**Catalogue >** *Valeur1* ►Decimal⇒valeur

1

0.333333

3

Liste1 ►Decimal⇒valeur*Matrice1* ►Decimal⇒valeur

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.

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.

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

Done

 $g(1,2)$

-4

 $1 \rightarrow a: 2 \rightarrow b: g(a,b)$

-4

Define $h(x)=\text{when}(x<2, 2 \cdot x - 3, -2 \cdot x + 3)$

Done

 $h(-3)$

-9

 $h(4)$

-5

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

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

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

Func

Bloc

EndFunc

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

Bloc

EndPrgm

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

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

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

Remarque : voir aussi **Define LibPriv**, page 40 et **Define LibPub**, page 41.

Define LibPriv

Define LibPriv Var = Expression

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

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

Define *g(x,y)*=Func

If *x>y* Then

Return *x*

Else

Return *y*

EndIf

EndFunc

Done

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 > 

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 39 et **Define LibPub**, page 41.

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 39 et **Define LibPriv**, page 40.

deltaList()

Voir **ΔList()**, page 91.

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 183.

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.

$2 \rightarrow a$	2
$(a+2)^2$	16
DelVar <i>a</i>	<i>Done</i>
$(a+2)^2$	"Error: Variable is not defined"

delVoid()**Catalogue > ****delVoid(*ListeI*)**⇒*liste*Donne une liste contenant les éléments de *ListeI* sans les éléments vides.

Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

<i>aa.a:=45</i>	45									
<i>aa.b:=5.67</i>	5.67									
<i>aa.c:=78.9</i>	78.9									
getVarInfo()	<table border="1"> <tr> <td><i>aa.a</i></td> <td>"NUM"</td> <td>""</td> </tr> <tr> <td><i>aa.b</i></td> <td>"NUM"</td> <td>""</td> </tr> <tr> <td><i>aa.c</i></td> <td>"NUM"</td> <td>""</td> </tr> </table>	<i>aa.a</i>	"NUM"	"  "	<i>aa.b</i>	"NUM"	"  "	<i>aa.c</i>	"NUM"	"  "
<i>aa.a</i>	"NUM"	"  "								
<i>aa.b</i>	"NUM"	"  "								
<i>aa.c</i>	"NUM"	"  "								
DelVar <i>aa.</i>	<i>Done</i>									
getVarInfo()	"NONE"									

det()**Catalogue > ****det(*matriceCarrée[, Tolérance]*)**⇒*expression*Donne le déterminant de *matriceCarrée*.

$\det\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	-2
$\begin{bmatrix} 1.\text{e}20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow mat1$	$\begin{bmatrix} 1.\text{e}20 & 1 \\ 0 & 1 \end{bmatrix}$
$\det(mat1)$	0
$\det(mat1.,1)$	$1.\text{e}20$

det()**Catalogue >**

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)**

diag()**Catalogue >**

diag(Liste)⇒matrice

diag([2 4 6])

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

diag(matriceLigne)⇒matrice

diag(matriceColonne)⇒matrice

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

diag(matriceCarrée)⇒matriceLigne

$$\begin{array}{|c|c|c|} \hline & [4 & 6 & 8] & [4 & 6 & 8] \\ \hline & [1 & 2 & 3] & [1 & 2 & 3] \\ \hline & [5 & 7 & 9] & [5 & 7 & 9] \\ \hline \text{diag}(Ans) & [4 & 2 & 9] & [4 & 2 & 9] \\ \hline \end{array}$$

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

matriceCarrée doit être une matrice carrée.

dim()**Catalogue >**

dim(Liste)⇒entier

dim({0,1,2})

3

Donne le nombre d'éléments de *Liste*.

dim()**dim(***Matrice*) \Rightarrow *liste*

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

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

dim(*Chaîne*) \Rightarrow *entier*

Donne le nombre de caractères contenus dans *Chaîne*.

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

Disp**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 : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Define <i>chars</i> (<i>start,end</i>)=Prgm For <i>i,start,end</i> Disp <i>i</i> , " ",char(<i>i</i>) EndFor EndPrgm Done	
<i>chars</i> (240,243)	
240	ð
241	ñ
242	ò
243	ó
	Done

DispAt

Catalogue >

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

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

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

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

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

DispAtet Disp peuvent être utilisés au sein du même programme.

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

DispAt

Par exemple :

```
dispat_demo(2/3)
Define dispat_demo()
Prgm
For n,1,5
DispAt n, "Line ",n
EndFor
EndPrgm
```

Line 1
Line 2
Line 3
Line 4
Line 5
Done

```
"dispat_demo" stored si
Define dispat_demo()
Prgm
For n,1,5
DispAt 3, "Line ",n
EndFor
EndPrgm
```

Line 5
Done

Exemples illustratifs :

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

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

Conditions d'erreur :

Message d'erreur	Description
Le numéro de ligne DispAt doit être compris entre 1 et 8	L'expression évalue le numéro de la ligne en dehors de la plage 1 - 8 (inclus)
Nombre insuffisant d'arguments	Il manque un ou plusieurs arguments à la fonction ou commande.
Aucun argument	Identique à la boîte de dialogue « erreur de syntaxe » actuelle
Trop d'arguments	Limiter les arguments. Même erreur que Disp.
Type de données incorrect	Le premier argument doit être un nombre.
Vide : DispAt vide	L'erreur de type de données "Hello World" (Datatype error) est renvoyée pour le vide (si le rappel est défini)

►DMS

Catalogue > ▶DMS

Valeur ►DMS

En mode Angle en degrés :

Liste ►DMS

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

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 213 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.

dotP()

dotP(Liste1, Liste2)⇒expression

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

17

Donne le produit scalaire de deux listes.

dotP(Vecteur1, Vecteur2)⇒expression

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

32

Donne le produit scalaire de deux vecteurs.

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

E**e^()**

e^(Valeur1)⇒valeur

e¹

2.71828

Donne e élevé à la puissance de Valeur1.

e³²

8103.08

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

Remarque : une pression sur pour afficher e^(est différente d'une pression sur le caractère [E] du clavier.

Vous pouvez entrer un nombre complexe sous la forme polaire $r\text{e}^{i\theta}$. 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^A()**Touche****e^A(Liste1)⇒liste**

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

e^A(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^{1,1,0,5}	{2.71828, 2.71828, 1.64872}
------------------------------	-----------------------------

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

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

eff(5.75,12)

5.90398

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 114.

eigVc()**Catalogue** > **eigVc(matriceCarrée)⇒matrice**

En mode Format complexe Rectangulaire :

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

$$\begin{aligned} \text{eigVc}(m1) \\ \begin{bmatrix} -0.800906 & 0.767947 & \\ 0.484029 & 0.573804+0.052258\cdot i & 0.5738 \\ 0.352512 & 0.262687+0.096286\cdot i & 0.2626 \end{bmatrix} \end{aligned}$$

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

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.

eigVI()**eigVI(*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 :

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

eigVI(*m1*)
 $\{ -4.40941, 2.20471 + 0.763006 \cdot i, 2.20471 - 0 \cdot i \}$

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

Else**Voir If, page 75.****ElseIf****If Expr booléenne1 Then***Bloc1***ElseIf Expr booléenne2 Then***Bloc2*

⋮

ElseIf Expr booléenneN Then*BlocN***EndIf**

⋮

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

Define g(x)=Func

If $x \leq 5$ Then

Return 5

ElseIf $x > -5$ and $x < 0$ ThenReturn $-x$ ElseIf $x \geq 0$ and $x \neq 10$ ThenReturn x ElseIf $x = 10$ Then

Return 3

EndIf

EndFunc

Done

EndFor**Voir For, page 59.****EndFunc****Voir Func, page 63.****EndIf****Voir If, page 75.****EndLoop****Voir Loop, page 99.****EndPrgm****Voir Prgm, page 127.****EndTry****Voir Try, page 176.****EndWhile****Voir While, page 186.****euler ()****Catalogue > **

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

Équation différentielle :

$y' = 0.001 * y * (100 - y)$ et $y(0) = 10$

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

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

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

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

Utilise la méthode d'Euler pour résoudre
le système.

Système d'équations :

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

avec $\text{VarD}\ddot{\text{e}}\text{p}(\text{Var}0)=\text{Var}0\text{D}\ddot{\text{e}}\text{p}$ pour l'intervalle $[\text{Var}0,\text{MaxVar}]$. 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 $\text{ListeVarD}\ddot{\text{e}}\text{p}$).

ListeExpr est la liste des côtés droits qui définissent le système des équations différentielles (en fonction de l'ordre des variables dépendantes de la $\text{ListeVarD}\ddot{\text{e}}\text{p}$).

Var est la variable indépendante.

$\text{ListeVarD}\ddot{\text{e}}\text{p}$ est la liste des variables dépendantes.

$\{\text{Var}0, \text{MaxVar}\}$ est une liste à deux éléments qui indique la fonction à intégrer de $\text{Var}0$ à MaxVar .

$\text{ListeVar}0\text{D}\ddot{\text{e}}\text{p}$ est la liste des valeurs initiales pour les variables dépendantes.

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

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

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

$$\text{euler}\left(\begin{cases} y1' = -y1 + 0.1 \cdot y1 \cdot y2 \\ y2' = 3 \cdot y2 - y1 \cdot y2 \end{cases}, t, \{y1, y2\}, \{0.5\}, \{2.5\}, 1 \right)$$

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

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

eval ()**Menu hub**

eval(*Expr*) \Rightarrow chaîne

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

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

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

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

Réinitialisez l'élément bleu sur OFF (ARRÊT).

Send "SET COLOR.BLUE OFF"	<i>Done</i>
---------------------------	-------------

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

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

Programmez pour faire apparaître en fondu l'élément rouge

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

Exécutez le programme.

<i>fadein()</i>	<i>Done</i>
-----------------	-------------

eval ()

Menu hub

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

*iostr.SendAns
iostr.GetAns
iostr.GetStrAns*

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

Remarque : Voir également **Get** (page 65), **GetStr** (page 73) et **Send** (page 150).

Exit

Catalogue >

Exit

Liste des fonctions :

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

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

g() 21

exp()

Touche

exp(Valeur1) \Rightarrow valeur

Donne l'exponentielle de *Valeur1*.

Remarque : voir aussi Modèle e Exposant, page 2.

Vous pouvez entrer un nombre complexe sous la forme polaire $r\text{ei}^{\theta}$. 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.

<i>e¹</i>	2.71828
<i>e³²</i>	8103.08

exp()**Touche****exp(Liste1)⇒liste**

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

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.

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

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

expr()**Catalogue** > **expr(*Chaîne*)⇒expression**

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

"Define cube(x)=x^3" → <i>funcstr</i>	"Define cube(x)=x^3"
expr(<i>funcstr</i>)	Done
<i>cube(2)</i>	8

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

Effectue l'ajustement exponentiel = $a \cdot (b)x$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

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

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.

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

Pour arrêter un calcul manuellement,

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

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

FCdf()**Catalog >**

```
FCdf
(
lowBound
,upBound,dfNumér,dfDénom)⇒nombre si
lowBound et upBound sont des nombres,
liste si lowBound et upBound sont des listes
```

FCdf

Fcdf()

Catalog >

(
lowBound

,upBound,dfNumér,dfDénom)⇒nombre si
lowBound et upBound sont des nombres,
liste si lowBound et upBound sont des listes

Calcule la fonction de répartition de la loi de Fisher F de degrés de liberté dfNumer et dfDenom entre lowBound et upBound.

Pour $P(X \leq upBound)$, utilisez lowBound = 0.

Fill

Catalogue >

Fill Valeur, VarMatrice⇒matrice

Remplace chaque élément de la variable VarMatrice par Valeur.

VarMatrice doit avoir été définie.

Fill Valeur, VarListe⇒liste

Remplace chaque élément de la variable VarListe par Valeur.

VarListe doit avoir été définie.

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

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

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

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

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

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

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 164.)

X est une liste qui contient les données.

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

Catégorie est une liste de codes numériques de catégories pour les valeurs X correspondantes.

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

Tout élément vide dans les listes X , *Fréq* ou *Catégorie* correspond à un élément vide dans l'ensemble des listes résultantes. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

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

floor()

floor(*ValeurI*)⇒entier

floor(-2.14)

-3.

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(*ListeI*)⇒liste

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

{1, 0, -6.}

floor(*MatriceI*)⇒matrice

floor($\begin{bmatrix} 1.2 & 3.4 \\ 2.5 & 4.8 \end{bmatrix}$)

[1. 3.]

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

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

For

Catalogue >

For *Var, Début, Fin [, Incrémentation]*

Bloc

EndFor

Exécute de façon itérative les instructions de *Bloc* pour chaque valeur de *Var*, à partir de *Début* jusqu'à *Fin*, par incrémentations équivalentes à *Incrément*.

Var ne doit pas être une variable système.

Incrément peut être une valeur positive ou négative. La valeur par défaut est 1.

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

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

Define *g()*=Func

Done

Local *tempsum,step,i*

0→*tempsum*

1→*step*

For *i,1,100,step*

tempsum+i→*tempsum*

EndFor

EndFunc

g()

5050

format()

Catalogue >

format(*Valeur[, chaîneFormat]*)⇒*chaîne*

Donne *Valeur* sous la forme d'une chaîne de caractères correspondant au modèle de format spécifié.

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.

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"

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.

fPart()

fPart(*Expr1*)⇒*expression*

fPart(-1.234)	-0.234
---------------	--------

fPart(*Liste1*)⇒*liste*

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

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.

FPdf()

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()**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 236.

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

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

frequency()

frequency*(Liste1,ListeBinaires)⇒liste*

Affiche une liste contenant le nombre total d'éléments dans *Liste1*. Les comptages sont effectués à partir de plages (binaires) définies par l'utilisateur dans *listeBinaires*.

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

frequency{dataList,{2.5,4.5}} {2,4,3}

frequency()

Catalogue >

Si *listeBinaires* est { $b(1), b(2), \dots, b(n)$ }, les plages spécifiées sont { $\leq b(1), b(1) < \leq b(2), \dots, b(n-1) < ? \leq b(n), b(n) > ?$ }. Le résultat comporte un élément de plus que *listeBinaires*.

Chaque élément du résultat correspond au nombre d'éléments dans *Liste1* présents dans la plage. Exprimé en termes de fonction **countIf()**, le résultat est { countIf(liste, $\leq b(1)$), countIf(liste, $b(1) < ? \leq b(2)$), ..., countIf(liste, $b(n-1) < ? \leq b(n)$), countIf(liste, $b(n) > ?$) }.

Les éléments de *Liste1* qui ne sont pas "placés dans une plage" ne sont pas pris en compte. Les éléments vides sont également ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

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

Remarque : voir également **countIf()**, page 33.

Explication du résultat :

2 éléments de *DataList* sont $\leq 2,5$

4 éléments de *DataList* sont $> 2,5$ et $\leq 4,5$

3 éléments de *DataList* sont $> 4,5$

L'élément « hello » est une chaîne et ne peut être placé dans aucune des plages définies.

FTest_2Samp

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 164.)

Pour $H_a : \sigma_1 > \sigma_2$, définissez *Hypoth*>0

Pour $H_a : \sigma_1 \neq \sigma_2$ (par défaut), définissez
 $Hypothesis = 0$

Pour $H_a : \sigma_1 < \sigma_2$, définissez $Hypothesis < 0$

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

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 <i>Liste 1</i> et <i>Liste 2</i> .
stat.x1_bar stat.x2_bar	Moyenne de population d'échantillon des séquences de données dans <i>Liste 1</i> et <i>Liste 2</i> .
stat.n1, stat.n2	Taille des échantillons

Func

Func
Bloc
EndFunc

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

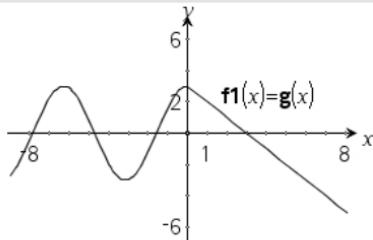
Bloc peut correspondre à une instruction unique ou à une série d'instructions séparées par le caractère ":" ou à une série d'instructions réparties sur plusieurs lignes. La fonction peut utiliser l'instruction **Return** pour donner un résultat spécifique.

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

Définition d'une fonction par morceaux :

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

Résultat de la représentation graphique de g(x)

**G****gcd()****gcd(*Nombre1, Nombre2*)** \Rightarrow expression

gcd(18,33)

3

Donne le plus grand commun diviseur des deux arguments. Le **gcd** de deux fractions correspond au **gcd** de leur numérateur divisé par le **lcm** de leur dénominateur.

En mode Auto ou Approché, le **gcd** de nombre fractionnaires en virgule flottante est égal à 1.

gcd(*Liste1, Liste2*) \Rightarrow liste

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

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

gcd(*Matrice1, Matrice2*) \Rightarrow matrice

gcd([2 4][4 8],[6 8][12 16]) [2 4][6 8]

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

geomCdf()

geomCdf(*p,lowBound,upBound*) \Rightarrow nombre
si les bornes *lowBound* et *upBound* sont des nombres, *liste* si les bornes *lowBound* et *upBound* sont des listes

geomCdf(*p,upBound*) pour $P(1 \leq X \leq upBound)$ \Rightarrow nombre si la borne *upBound* est un nombre, *liste* si la borne *upBound* est

une liste

Calcule la probabilité qu'une variable suivant la loi géométrique prenne une valeur entre les bornes *lowBound* et *upBound* en fonction de la probabilité de réussite *p* spécifiée.

Pour $P(X \leq upBound)$, définissez *lowBound* = 1.

geomPdf(*p, ValX*) \Rightarrow *nombre* si *ValX* est un nombre, *liste* si *ValX* est une liste

Calcule la probabilité que le premier succès intervienne au rang *ValX*, pour la loi géométrique discrète en fonction de la probabilité de réussite *p* spécifiée.

Get

Menu hub

Get[*promptString*,]*var*[, *statusVar*]

Get[*promptString*,] *fonc*{*arg1*, ...*argn*} [, *statusVar*]

Commande de programmation : récupère une valeur d'un hub connecté TI-Innovator™ Hub et affecte cette valeur à la variable *var*.

La valeur doit être demandée :

- À l'avance, par le biais d'une commande **Send "READ ..."** commande.
- ou —
- En incorporant une demande "**READ ...**" comme l'argument facultatif de *promptString*. Cette méthode vous permet d'utiliser une seule commande pour demander la valeur et la récupérer.

Exemple : demander la valeur actuelle du capteur intégré du niveau de lumière du hub. Utilisez **Get** pour récupérer la valeur et l'affecter à la variable *lightval*.

Send "READ BRIGHTNESS"	<i>Done</i>
Get <i>lightval</i>	<i>Done</i>
<i>lightval</i>	0.347922

Incorporez la demande READ dans la commande **Get**.

Get "READ BRIGHTNESS", <i>lightval</i>	<i>Done</i>
<i>lightval</i>	0.378441

Une simplification implicite a lieu. Par exemple, la réception de la chaîne de caractères "123" est interprétée comme étant une valeur numérique. Pour conserver la chaîne de caractères, utilisez **GetStr** au lieu de **Get**.

Si vous incluez l'argument facultatif *statusVar*, une valeur lui sera affectée en fonction de la réussite de l'opération. Une valeur zéro signifie qu'aucune donnée n'a été reçue.

Dans la deuxième syntaxe, l'argument *fonc()* permet à un programme de stocker la chaîne de caractères reçue comme étant la définition d'une fonction. Cette syntaxe équivaut à l'exécution par le programme de la commande suivante :

Define *fonc(arg1, ...argn) = chaîne*
reçue

Le programme peut alors utiliser la fonction définie *fonc()*.

Remarque : vous pouvez utiliser la commande **Get** dans un programme défini par l'utilisateur, mais pas dans une fonction.

Remarque : Voir également **GetStr**, page 73 et **Send**, page 150.

getDenom()

getDenom(*Fraction1*)⇒*valeur*

Transforme l'argument en une expression dotée d'un dénominateur commun réduit, puis en donne le numérateur.

Catalogue >

x:=5; y:=6	6
getDenom($\frac{x+2}{y-3}$)	3
getDenom($\frac{2}{7}$)	7
getDenom($\frac{1+y^2+y}{x-y^2}$)	30

getKey()

getKey([0|1]) ⇒ **returnString**

Description : `getKey()` - permet à un programme TI-Basic de recevoir des entrées de clavier - calculatrice, ordinateur de bureau et émulateur sur ordinateur de bureau.

Par exemple :

- `keypressed := getKey()` retournera une touche ou une chaîne vide si aucune touche n'a été pressée. Cet appel sera immédiatement retourné.
- `keypressed := getKey(1)` attendra l'appui sur une touche. Cet appel mettra en pause l'exécution du programme jusqu'à l'appui sur une touche.

`getKey()`

Par exemple :

```

"getKey_demo" stored sh
Define getKey_demo()h
Prgm
Local key
key:= getKey(1)
While key≠"esc"
key:=getKey(1)
Disp "Key: ",key
EndWhile
EndPrgm

```

Context menu (right-clicked on the code):

- Key: 1
- Key: A
- Key: =
- Key: ^
- Key: square
- Key: var
- Key: esc

Done

Traitement des frappes de touche :

Touche de calculatrice/émulateur	Ordinateur	Valeur de retour
Échap	Échap	« échap »
Pavé tactile - Clic en haut	n/a	« haut »
On	n/a	« accueil »
Scratchapps	n/a	"scratchpad"
Pavé tactile - Clic gauche	n/a	« gauche »
Pavé tactile - Clic au centre	n/a	« centre »
Pavé tactile - Clic droit	n/a	« droite »
Classeur	n/a	« classeur »
Tab	Tab	« tab »
Pavé tactile - Clic en bas	Flèche bas	« bas »
Menu	n/a	« menu »

Touche de calculatrice/émulateur	Ordinateur	Valeur de retour
Ctrl	Ctrl	aucun retour
Maj	Maj	aucun retour
Var	n/a	« var »
Suppr	n/a	« suppr »
=	=	"="
trigonométrie	n/a	« trigonométrie »
0 à 9	0-9	« 0 » ... « 9 »
Modèles	n/a	« modèle »
Catalogue	n/a	« cat »
^	^	"^"
X^2	n/a	« carré »
/ (touche division)	/	"/"
* (touche multiplication)	*	"*"
e^x	n/a	« expr »
10^x	n/a	« puissance de 10 »
+	+	"+"
-	-	"_"
(("("
))	")"
.	.	".."
(-)	n/a	« - » (signe moins)
Entrée	Entrée	« entrée »
ee	n/a	« E » (notation scientifique E)
a - z	a-z	alpha = lettre pressée (minuscule) ("a" - "z")
maj a-z	maj a-z	alpha = lettre pressée « A » - « Z »
		Note : ctrl-maj fonctionne pour le verrouillage des

Touche de calculatrice/émulateur	Ordinateur	Valeur de retour
		majuscules
?!	n/a	"?!"
pi	n/a	« pi »
Marque	n/a	aucun retour
,	,	","
Retour	n/a	Retour
Espace	Espace	« » (espace)
Inaccessible	Touches de caractères spéciaux tels que @, !, ^, etc.	Le caractère est retourné
n/a	Touches de fonction	Aucun caractère retourné
n/a	Touches de commandes spéciales pour ordinateur	Aucun caractère retourné
Inaccessible	Autres touches pour ordinateur non disponibles sur la calculatrice lorsque getKey() est en attente d'une frappe. ({, }, ;, :, ...)	Le même caractère que vous obtenez dans l'Éditeur mathématique (pas dans une boîte mathématique)

Remarque : Il est important de noter que la présence de `getKey()` dans un programme modifie la façon dont certains événements sont traités par le système. Certains sont décrits ci-dessous.

Arrête le programme et traite l'événement - Exactement comme si l'utilisateur quittait le programme en appuyant sur la touche **ON**

« **Support** » ci-dessous signifie - le système fonctionne comme prévu - le programme continue à être exécuté.

Événement	Unité nomade	Ordinateur - TI-Nspire™ Student Software
Questions rapides	Arrête le programme, traite l'événement	Comme avec l'unité nomade (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software- uniquement)
Gestion des fichiers à distance	Arrête le programme, traite l'événement	Comme avec l'unité nomade. (TI-Nspire™ Student

Événement	Unité nomade	Ordinateur - TI-Nspire™ Student Software
(Incl. l'envoi du fichier « Exit Press 2 Test » d'une unité nomade à une autre ou à un ordinateur)		Software, TI-Nspire™ Navigator™ NC Teacher Software-uniquement)
Fermer la classe	Arrête le programme, traite l'événement	Support (TI-Nspire™ Student Software, TI-Nspire™ Navigator™ NC Teacher Software-uniquement)

Événement	Unité nomade	Ordinateur - TI-Nspire™ Toutes les versions
TI-Innovator™ Hub connexion/déconnexion	Support - Peut émettre avec succès des commandes à TI-Innovator™ Hub. Après que vous ayez quitté le programme, le TI-Innovator™ Hub continue de travailler avec l'unité nomade.	Comme avec l'unité nomade

getLangInfo()

Catalogue > 

`getLangInfo()=>chaîne`

getLangInfo()

"en"

Retourne une chaîne qui correspond au nom abrégé de la langue active. Vous pouvez, par exemple, l'utiliser dans un programme ou une fonction afin de déterminer la langue courante.

Anglais = « en »

Danois = « da »

Allemand = « de »

Finlandais = « fi »

Français = « fr »

Italien = « it »

Néerlandais = « nl »

Néerlandais belge = « nl_BE »

getLangInfo()

Catalogue >

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 95 et **unLock**, page 183.

<i>a:=65</i>	65
Lock <i>a</i>	Done
getLockInfo(<i>a</i>)	1
<i>a:=75</i>	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
Unlock <i>a</i>	Done
<i>a:=75</i>	75
DelVar <i>a</i>	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.

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

Si vous enregistrez les réglages avec **getMode(0) → var**, vous pouvez utiliser **setMode(var)** dans une fonction ou un programme pour restaurer temporairement les réglages au sein de l'exécution de la fonction ou du programme uniquement. Voir également **setMode()**, page 153.

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 Complex	4	1 =Réel, 2 =Rectangulaire, 3 =Polaire
Auto ou Approché	5	1 =Auto, 2 =Approché
Format Vecteur	6	1 =Rectangulaire, 2 =Cylindrique, 3 =Sphérique
Base	7	1 =Décimale, 2 =Hexadécimale, 3 =Binaire

getNum(Fraction1)⇒valeur

Transforme l'argument en une expression dotée d'un dénominateur commun réduit, puis en donne le dénominateur.

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

GetStr[*promptString*,] *var*[, *statusVar*]

Par exemple, voir **Get**.

GetStr[*promptString*,] *fonc*{*arg1*, ...*argn*} [, *statusVar*]

Commande de programmation : fonctionne de manière identique à la commande **Get**, sauf que la valeur reçue est toujours interprétée comme étant une chaîne de caractères. En revanche, la commande **Get** interprète la réponse comme une expression, à moins que l'utilisateur ne la saisisse entre guillemets ("").

Remarque : Voir également **Get**, page 65 et **Send**, page 150.

getType()

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".

Catalogue >

{1,2,3}→ <i>temp</i>	{1,2,3}
getType(<i>temp</i>)	"LIST"
3· <i>i</i> → <i>temp</i>	3· <i>i</i>
getType(<i>temp</i>)	"EXPR"
DelVar <i>temp</i>	Done
getType(<i>temp</i>)	"NONE"

getVarInfo()

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).

Catalogue >

getVarInfo()	"NONE"												
Define <i>x</i> =5	Done												
Lock <i>x</i>	Done												
Define LibPriv <i>y</i> ={1,2,3}	Done												
Define LibPub <i>z</i> (<i>x</i>)=3· <i>x</i> ² - <i>x</i>	Done												
getVarInfo()	<table border="1"><tr><td><i>x</i></td><td>"NUM"</td><td>"{}"</td><td>1</td></tr><tr><td><i>y</i></td><td>"LIST"</td><td>"LibPriv"</td><td>0</td></tr><tr><td><i>z</i></td><td>"FUNC"</td><td>"LibPub"</td><td>0</td></tr></table>	<i>x</i>	"NUM"	"{}"	1	<i>y</i>	"LIST"	"LibPriv"	0	<i>z</i>	"FUNC"	"LibPub"	0
<i>x</i>	"NUM"	"{}"	1										
<i>y</i>	"LIST"	"LibPriv"	0										
<i>z</i>	"FUNC"	"LibPub"	0										
getVarInfo(<i>tmp3</i>)	"Error: Argument must be a string"												
getVarInfo("tmp3")	[volcyL2 "NONE" "LibPub" 0]												

getVarInfo

(*chaîneNomBibliothèque*) donne une matrice d'informations pour tous les objets de bibliothèque définis dans la bibliothèque *chaîneNomBibliothèque*. *chaîneNomBibliothèque* doit être une chaîne (texte entre guillemets) ou une variable.

Si la bibliothèque

chaîneNomBibliothèque n'existe pas, une erreur est générée.

Observez l'exemple de gauche dans lequel le résultat de **getVarInfo()** est affecté à la variable *vs*. La tentative d'afficher la ligne 2 ou 3 de *vs* génère un message d'erreur "Liste ou matrice invalide" car pour au moins un des éléments de ces lignes (variable *b*, par exemple) l'évaluation redonne une matrice.

Cette erreur peut également survenir lors de l'utilisation de *Ans* pour réévaluer un résultat de **getVarInfo()**.

Le système génère l'erreur ci-dessus car la version courante du logiciel ne prend pas en charge les structures de matrice généralisées dans lesquelles un élément de matrice peut être une matrice ou une liste.

<i>a:=1</i>	1
<i>b:=[1 2]</i>	[1 2]
<i>c:=[1 3 7]</i>	[1 3 7]
<i>vs:=getVarInfo()</i>	$\begin{bmatrix} a & \text{"NUM"} & "[1 2]" & 0 \\ b & \text{"MAT"} & "[1 3]" & 0 \\ c & \text{"MAT"} & "[1 2]" & 0 \end{bmatrix}$
<i>vs[1]</i>	[1 "NUM" "[1 2]" 0]
<i>vs[1,1]</i>	1
<i>vs[2]</i>	"Error: Invalid list or matrix"
<i>vs[2,1]</i>	[1 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 : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

Define $g() = \text{Func}$ *Done*Local *temp,i* $0 \rightarrow \text{temp}$ $1 \rightarrow i$ Lbl *top* $\text{temp} + i \rightarrow \text{temp}$ If $i < 10$ Then $i + 1 \rightarrow i$ Goto *top*

EndIf

Return *temp*

EndFunc

 $g()$

55

►Grad**Catalogue >** **Expr1 ► Grad⇒expression**

En mode Angle en degrés :

Convertit *Expr1* en une mesure d'angle en grades.

(1.5)►Grad $(1.66667)^g$

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

En mode Angle en radians :

(1.5)►Grad $(95.493)^r$

I

identity()**Catalogue >** **identity(*Entier*) ⇒ matrice**

identity(4)

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

Donne la matrice unité de dimension *Entier*.

Entier doit être un entier positif

If**Catalogue >** **If BooleanExpr
Relevé**Define $g(x) = \text{Func}$ *Done*If $x < 0$ ThenReturn x^2

EndIf

EndFunc

**If BooleanExpr Then
Bloc
EndIf**

 $g(-2)$

4

If

Si *BooleanExpr* est évalué à vrai, exécute l'instruction *Instruction* ou le bloc d'instructions *Bloc* avant de poursuivre l'exécution de la fonction

Si *BooleanExpr* est évalué à faux, poursuit l'exécution en ignorant l'instruction ou le bloc d'instructions

Bloc peut correspondre à une ou plusieurs instructions, séparées par le caractère « : »

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

If BooleanExpr Then

Bloc1

Else

Bloc2

EndIf

Si *BooleanExpr* est évalué à vrai, exécute *Bloc1* et ignore *Bloc2*.

Si *BooleanExpr* est évalué à faux, ignore *Bloc1*, mais exécute *Bloc2*.

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

If BooleanExpr1 Then

Bloc1

ElseIf BooleanExpr2 Then

Bloc2

:

ElseIf BooleanExprN Then

BlocN

EndIf

Permet de traiter les conditions multiples. Si *BooleanExpr1* est évalué à vrai, exécute *Bloc1*. Si *BooleanExpr1* est évalué à faux, évalue *BooleanExpr2*, et ainsi de suite.

Define g(x)=Func	<i>Done</i>
If $x < 0$ Then	
Return $-x$	
Else	
Return x	
EndIf	
EndFunc	
g(12)	12
g(-12)	12

Define g(x)=Func	<i>Done</i>
If $x < -5$ Then	
Return 5	
ElseIf $x > -5$ and $x < 0$ Then	
Return $-x$	
ElseIf $x \geq 0$ and $x \neq 10$ Then	
Return x	
ElseIf $x = 10$ Then	
Return 3	
EndIf	
EndFunc	
g(-4)	4
g(10)	3

ifFn()

Catalogue >

ifFn(exprBooléenne,Valeur_si_Vrai [,Valeur_si_Faux [,Valeur_si_Inconnu]]) \Rightarrow expression, liste ou matrice

Evalue l'expression booléenne *exprBooléenne* (ou chacun des éléments de *exprBooléenne*) et produit un résultat reposant sur les règles suivantes

- *exprBooléenne* peut tester une valeur unique, une liste ou une matrice
- Si un élément de *exprBooléenne* est évalué à vrai, l'élément correspondant de *Valeur_si_Vrai* s'affiche
- Si un élément de *exprBooléenne* est évalué à faux, l'élément correspondant de *Valeur_si_Faux* s'affiche Si vous omettez *Valeur_si_Faux*, *undef* s'affiche.
- Si un élément de *exprBooléenne* n'est ni vrai ni faux, l'élément correspondant de *Valeur_si_Inconnu* s'affiche Si vous omettez *Valeur_si_Inconnu*, *undef* s'affiche
- Si le deuxième, troisième ou quatrième argument de la fonction **ifFn()** est une expression unique, le test booléen est appliqué à toutes les positions dans *exprBooléenne*

Remarque : si l'instruction simplifiée *exprBooléenne* implique une liste ou une matrice, tous les autres arguments de type liste ou matrice doivent avoir la ou les même(s) dimension(s) et le résultat aura la ou les même(s) dimension(s).

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(Valuel) \Rightarrow valeur

imag(1+2·i)

2

Donne la partie imaginaire de l'argument.

imag()**Catalogue >** **imag(ListeI) ⇒ liste**

imag({-3,4-i,i})

{0,-1,1}

Donne la liste des parties imaginaires des éléments.

imag(MatriceI) ⇒ matriceimag $\begin{bmatrix} 1 & 2 \\ i \cdot 3 & i \cdot 4 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 \\ 3 & 4 \end{bmatrix}$

Donne la matrice des parties imaginaires des éléments.

Indirection**Voir #(), page 211.****inString()****Catalogue >** **inString(srcString, subString[, Début])**
⇒ entier

inString("Hello there", "the")

7

inString("ABCEFG", "D")

0

Donne le rang du caractère de la chaîne *chaîneSrce* où commence la première occurrence de *sousChaîne*.*Début*, s'il est utilisé, indique le point de départ de la recherche dans *chaîneSrce*. Par défaut = 1, la recherche commence à partir du (premier caractère de *chaîneSrce*).Si *chaîneSrce* ne contient pas *sousChaîne* ou si *Début* est strictement supérieur à la longueur de *ChaîneSrce*, on obtient zéro**int()****Catalogue >** **int(Value) ⇒ entier**
int(ListI) ⇒ liste
int(MatriceI) ⇒ matrice

int(-2.5)

-3.

int([-1.234 0 0.37])

[-2. 0 0.]

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.

int()**Catalogue >**

Dans le cas d'une liste ou d'une matrice, donne la partie entière de chaque élément.

intDiv()**Catalogue >**

intDiv(Number1, Number2) \Rightarrow entier
intDiv(List1, List2) \Rightarrow liste
intDiv(Matrix1, Matrix2) \Rightarrow matrice

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

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

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

interpoler ()**Catalogue >**

interpoler(Valeursx, Listex, Listey, ListePrincy) \Rightarrow list

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 *Valeursx*. 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 *Valeursx*. Si elle trouve cet intervalle, elle retourne une valeur d'interpolation pour **f**(*Valeursx*), sinon elle donne **undef**.

Listex, *Listey*, et *ListePrincy* doivent être de même dimensions ≥ 2 et contenir des expressions pouvant être évaluées à des nombres.

Valeursx peut être un nombre ou une liste de nombres.

Équation différentielle :

$$y' = -3 \cdot y + 6 \cdot t + 5 \text{ et } y(0) = 5$$

rk=	rk23[-3·y+6·t+5,y,{0,10},5,1)
[0.	1. 2. 3. 4.
5.	3.19499 5.00394 6.99957 9.00593 1c

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

Utilisez la fonction **interpolate()** pour calculer les valeurs de la fonction pour la liste *valeursx* :

xvalueList:=seq(i,i,0,10,0.5)
{0,0.5,1.,1.5,2.,2.5,3.,3.5,4.,4.5,5.,5.5,6.,6.5,
xList:=matList(rk[1])
{0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.}
yList:=matList(rk[2])
{5.,3.19499,5.00394,6.99957,9.00593,10.9978}
yPrimeList:=-3·y+6·t+5 y=yList and t=xList
{-10.,1.41503,1.98819,2.00129,1.98221,2.006}
interpolate(xvalueList,xList,yList,yPrimeList)
{5.,2.67062,3.19499,4.02782,5.00394,6.00011}

invχ²(Aire,df)

invChi2(Aire,df)

Calcule l'inverse de la fonction de répartition de la loi X^2 (Khi2) de degré de liberté df en un point donné *Aire*.

invF(Aire,dfNumer,dfDenom)

invF(Zone,dfNumer,dfDenom)

Calcule l'inverse de la fonction de répartition de la loi F (Fisher) de paramètres spécifiée par *dfNumer* et *dfDenom* en un point donné *Aire*

invBinom

*(CumulativeProb,NumTrials,Prob,
OutputForm)⇒ scalaire ou matrice*

Étant donné le nombre d'essais (*NumTrials*) et la probabilité de réussite de chaque essai (*Prob*), cette fonction renvoie le nombre minimal de réussites, *k*, tel que la probabilité cumulée de *k* réussites soit supérieure ou égale à une probabilité cumulée donnée (*CumulativeProb*).

OutputForm=0, affiche le résultat en tant que scalaire (par défaut).

OutputForm=1, affiche le résultat en tant que matrice.

Par exemple : Mary et Kevin jouent à un jeu de dés. Mary doit deviner le nombre maximal de fois où 6 apparaît dans 30 lancers. Si le nombre 6 apparaît autant de fois ou moins, Mary gagne. Par ailleurs, plus le nombre qu'elle devine est petit, plus ses gains sont élevés. Quel est le plus petit nombre que Mary peut deviner si elle veut que la probabilité du gain soit supérieure à 77 % ?

invBinom(0.77,30,1/6)	6
invBinom(0.77,30,1/6,1)	$\begin{bmatrix} 5 & 0.616447 \\ 6 & 0.776537 \end{bmatrix}$

invBinomN()

Catalogue >

**invBinomN(CumulativeProb,Prob,
NumSuccess,OutputForm)⇒ scalaire
ou matrice**

Étant donné la probabilité de réussite de chaque essai (*Prob*) et le nombre de réussites (*NumSuccess*), cette fonction renvoie le nombre minimal d'essais, *N*, tel que la probabilité cumulée de *x* réussites soit inférieure ou égale à une probabilité cumulée donnée (*CumulativeProb*).

OutputForm=0, affiche le résultat en tant que scalaire (par défaut).

OutputForm=1, affiche le résultat en tant que matrice.

Par exemple : Monique s'entraîne aux tirs au but au volley-ball. Elle sait par expérience que ses chances de marquer un but sont de 70 %. Elle prévoit de s'entraîner jusqu'à ce qu'elle marque 50 buts. Combien de tirs doit-elle tenter pour s'assurer que la probabilité de marquer au moins 50 buts est supérieure à 0,99 ?

invBinomN(0.01,0.7,49)

86

invBinomN(0.01,0.7,49,1)

$$\begin{bmatrix} 85 & 0.010451 \\ 86 & 0.00709 \end{bmatrix}$$

invNorm()

Catalogue >

invNorm(Aire[,μ[σ]])

Calcule l'inverse de la fonction de répartition de la loi normale de paramètres μ et σ en un point donné *Aire*.

invt()

Catalogue >

invt(Aire,df)

Calcule les fractiles d'une loi de Student à *df* degrés de liberté pour une *Aire* donnée.

iPart()

Catalogue >

iPart(Number)⇒ entier

iPart(-1.234)

-1.

iPart(List1)⇒ liste

iPart($\left\{ \frac{3}{2}, -2.3, 7.003 \right\}$)

{ 1, -2., 7. }

iPart(Matrix1)⇒ 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.

irr()

irr(*CF0,CFList [,CFFreq]*) \Rightarrow valeur

Fonction financière permettant de calculer le taux interne de rentabilité d'un investissement.

MTO correspond au mouvement de trésorerie initial à l'heure 0 ; il doit s'agir d'un nombre réel.

Liste MT est une liste des montants de mouvements de trésorerie après le mouvement de trésorerie initial MTO.

FréqMT est une liste 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 104.

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()

isPrime(*Nombre*) \Rightarrow Expression booléenne constante

Donne true ou false selon que *nombre* est ou n'est pas un entier naturel premier ≥ 2 , divisible uniquement par lui-même et 1.

Si *Nombre* dépasse 306 chiffres environ et n'a pas de diviseur ≤ 1021 , **isPrime (*Nombre*)** affiche un message d'erreur.

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

isPrime(5) true

isPrime(6) false

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

<pre>Define nextprim(n)=Func Loop n+1→n If isPrime(n) Return n EndLoop EndFunc</pre>	Done
--	------

nextprim(7) 11

isVoid()

isVoid(Var) \Rightarrow Expression booléenne constante
isVoid(Expr) \Rightarrow Expression booléenne constante
isVoid(Var) \Rightarrow liste d'expressions booléennes constantes

Retourne true ou false pour indiquer si l'argument est un élément de type données vide.

Pour plus d'informations concernant les éléments vides, reportez-vous à . page 236.

<i>a:=_</i>	<i>_</i>
isVoid(<i>a</i>)	true
isVoid({1,_,3})	{false,true,false}

L**Lbl****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 : Pour obtenir des instructions sur la saisie des définitions de fonction ou de programme sur plusieurs lignes, consultez la section relative à la calculatrice dans votre guide de produit.

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

g()	55
-----	----

lcm()

lcm(Nombre1, Nombre2)⇒expression

lcm(Liste1, Liste2)⇒liste

lcm(Matrice1, Matrice2)⇒matrice

$\text{lcm}(6,9)$	18
$\text{lcm}\left(\left\{\frac{1}{3}, -14, 16\right\}, \left\{\frac{2}{15}, 7, 5\right\}\right)$	$\left\{\frac{2}{3}, 14, 80\right\}$

lcm()**Catalogue > **

Donne le plus petit commun multiple des deux arguments. Le **lcm** de deux fractions correspond au **lcm** de leur numérateur divisé par le **gcd** de leur dénominateur. Le **lcm** de nombres fractionnaires en virgule flottante correspond à leur produit.

Pour deux listes ou matrices, donne les plus petits communs multiples des éléments correspondants.

left()**Catalogue > **

left(*chaîneSrce*[, *Nomb*]) \Rightarrow *chaîne*

`left("Hello",2)`

"He"

Donne la chaîne formée par les *Nomb* premiers caractères de la chaîne *chaîneSrce*.

Si *Nomb* est absent, on obtient *chaîneSrce*.

left(*Liste1*[, *Nomb*]) \Rightarrow *liste*

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

{1,3,-2}

Donne la liste formée par les *Nomb* premiers éléments de *Liste1*.

Si *Nomb* est absent, on obtient *Liste1*.

left(*Comparaison*) \Rightarrow *expression*

Donne le membre de gauche d'une équation ou d'une inéquation.

libShortcut()**Catalogue > **

libShortcut(*chaîneNomBibliothèque*,
chaîneNomRaccourci[,
LibPrivFlag]) \Rightarrow *liste de variables*

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

libShortcut()

Catalogue >

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 27. Pour supprimer un groupe de variables, reportez-vous à **DelVar**, page 42.

getVarInfo("linalg2")

<i>clearmat</i>	"FUNC"	"LibPub "
<i>gauss1</i>	"PRGM"	"LibPriv "
<i>gauss2</i>	"FUNC"	"LibPub "

libShortcut("linalg2","la")

{*la.clearmat,la.gauss2*}

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

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

LinRegBx

Catalogue >

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

Effectue l'ajustement linéaire $y = a + b \cdot x$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

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

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

Effectue l'ajustement linéaire $y = m \cdot x + b$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

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

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

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

LinRegIntervals *X, Y[, F[, 1[, Xval[, NivC]]]]*

Réponse. Calcule une valeur y prévue, un intervalle de prévision de niveau C pour une seule observation et un intervalle de confiance de niveau C pour la réponse moyenne.

Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 236.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a+b \cdot x$
stat.a, stat.b	Coefficients d'ajustement
stat.df	Degrés de liberté
stat.r ²	Coefficient de détermination
stat.r	Coefficient de corrélation
stat.Resid	Valeurs résiduelles de l'ajustement

Pour les intervalles de type Slope uniquement

Variable de sortie	Description
[stat.CLower, stat.CUpper]	Intervalle de confiance de pente
stat.ME	Marge d'erreur de l'intervalle de confiance
stat.SESlope	Erreur type de pente

Variable de sortie	Description
stat.s	Erreur type de ligne

Pour les intervalles de type Response uniquement

Variable de sortie	Description
[stat.CLower, stat.CUpper]	Intervalle de confiance pour une réponse moyenne
stat.ME	Marge d'erreur de l'intervalle de confiance
stat.SE	Erreur type de réponse moyenne
[stat.LowerPred, stat.UpperPred]	Intervalle de prévision pour une observation simple
stat.MEPred	Marge d'erreur de l'intervalle de prévision
stat.SEPred	Erreur type de prévision
stat. \hat{y}	$a + b \cdot \text{ValX}$

LinRegtTest

Catalogue > 

LinRegtTest *X,Y[,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 $H_0 : \beta = 0$ (équivalent, $p=0$) 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 ($H_0 : \beta = p = 0$) est testée.

Pour $H_a : \beta \neq 0$ et $p \neq 0$ (par défaut), définissez $Hypoth=0$

Pour $H_a : \beta < 0$ et $p < 0$, définissez $Hypoth<0$

Pour $H_a : \beta > 0$ et $p > 0$, définissez $Hypoth>0$

Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a + b \cdot x$
stat.t	<i>t</i> -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(Système $\acute{E}qLin$, *Var1*, *Var2*, ...) \Rightarrow liste

linSolve($\acute{E}qLin_1$ and $\acute{E}qLin_2$ and ..., *Var1*, *Var2*, ...) \Rightarrow liste

linSolve({ $\acute{E}qLin_1$, $\acute{E}qLin_2$, ...}, *Var1*, *Var2*, ...) \Rightarrow liste

linSolve(Système $\acute{E}qLin$, {*Var1*, *Var2*, ...}) \Rightarrow liste

linSolve($\acute{E}qLin_1$ and $\acute{E}qLin_2$ and ..., {*Var1*, *Var2*, ...}) \Rightarrow liste

linSolve({ $\acute{E}qLin_1$, $\acute{E}qLin_2$, ...}, {*Var1*, *Var2*, ...}) \Rightarrow 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".

linSolve $\left(\begin{cases} 2 \cdot x + 4 \cdot y = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{cases}, \{x,y\}\right)$	$\left\{\frac{37}{26}, \frac{1}{26}\right\}$
linSolve $\left(\begin{cases} 2 \cdot x = 3 \\ 5 \cdot x - 3 \cdot y = 7 \end{cases}, \{x,y\}\right)$	$\left\{\frac{3}{2}, \frac{1}{6}\right\}$
linSolve $\left(\begin{cases} apple + 4 \cdot pear = 23 \\ 5 \cdot apple - pear = 17 \end{cases}, \{apple,pear\}\right)$	$\left\{\frac{13}{3}, \frac{14}{3}\right\}$
linSolve $\left(\begin{cases} apple + 4 + \frac{pear}{3} = 14 \\ -apple + pear = 6 \end{cases}, \{apple,pear\}\right)$	$\left\{\frac{36}{13}, \frac{114}{13}\right\}$

Δ list()

Catalogue >

Δ list(*Liste1*) \Rightarrow liste

Δ List({20,30,45,70})

{10,15,25}

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

Donne la liste des différences entre les éléments consécutifs de *Liste1*. Chaque élément de *Liste1* est soustrait de l'élément suivant de *Liste1*. Le résultat comporte toujours un élément de moins que la liste *Liste1* initiale.

list►mat()**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** (...).

list►mat({1,2,3})

[1 2 3]

list►mat({1,2,3,4,5},2)

1	2
3	4
5	0

ln()**Touches**

ln(Valeur1)⇒valeur

ln(2.)

0.693147

ln(Liste1)⇒liste

Donne le logarithme népérien de l'argument.

Dans le cas d'une liste, donne les logarithmes népériens de tous les éléments de celle-ci.

En mode Format complexe Réel :

ln({-3,1,2,5})

"Error: Non-real calculation"

En mode Format complexe Rectangulaire :

ln({-3,1,2,5})

{1.09861+3.14159·i,0.182322,1.60944}

ln(matriceCarrée1)⇒matriceCarrée

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

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

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

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

{1.83145+1.73485·i 0.009193-1.49086
0.448761-0.725533·i 1.06491+0.623491·i
-0.266891-2.08316·i 1.12436+1.79018·i}

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

LnReg**Catalogue >** 

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

Effectue l'ajustement logarithmique $y = a+b \cdot \ln(x)$ sur les listes X et Y en utilisant la fréquence $Fréq$. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $a+b \cdot \ln(x)$
stat.a, stat.b	Coefficients d'ajustement
stat.r ²	Coefficient de détermination linéaire pour les données transformées
stat.r	Coefficient de corrélation pour les données transformées ($\ln(x)$, y)

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

Local

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

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

Done

<i>rollcount()</i>	16
<i>rollcount()</i>	3

LockVar1 [, Var2] [, Var3] ...

LockVar.

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 183 et **getLockInfo()**, page 71.

<i>a:=65</i>	65
Lock a	Done
getLockInfo(a)	1
<i>a:=75</i>	"Error: Variable is locked."
DelVar a	"Error: Variable is locked."
Unlock a	Done
<i>a:=75</i>	75
DelVar a	Done

log()

Touches  

log(Valeur1[, Valeur2])⇒valeur

log(Liste1[, Valeur2])⇒liste

Donne le logarithme de base *Valeur2* de l'argument.

$\log_{10}(2.)$	0.30103
$\log_4(2.)$	0.5
$\log_3(10) - \log_3(5)$	$\log_3(2)$

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

Dans le cas d'une liste, donne le logarithme de base *Valeur2* des éléments.

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

En mode Format complexe Réel :

$\log_{10}(\{-3,1,2,5\})$	"Error: Non-real calculation"
---------------------------	-------------------------------

En mode Format complexe Rectangulaire :

$\log_{10}(\{-3,1,2,5\})$	$\{0.477121+1.36438\cdot i, 0.079181, 0.69897\}$
---------------------------	--

**log(*matriceCarrée1*,
[*Valeur*])** \Rightarrow *matriceCarrée*

Donne le logarithme de base *Valeur* de *matriceCarrée1*. Ce calcul est différent du calcul du logarithme de base *Valeur* 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 :

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

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

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

Logistic

Catalogue >

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

Effectue l'ajustement logistiquey = (c/
(1+a · 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 164.)

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 numériques ou alphanumériques 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 236.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $c/(1+a \cdot e^{-bx})$
stat.a, stat.b, stat.c	Coefficients d'ajustement
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

LogisticD

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

Effectue l'ajustement logistique $y = (c/(1+a \cdot e^{-bx})+d)$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq* et un nombre spécifique d'*Itérations*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $c/(1+a \cdot e^{-bx})+d)$
stat.a, stat.b, stat.c, stat.d	Coefficients d'ajustement
stat.Resid	Valeurs résiduelles de l'ajustement
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Fréq</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

Loop

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

Define *rollcount()*=Func

Local *i*

$1 \rightarrow i$

Loop

If randInt(1,6)=randInt(1,6)

Goto *end*

$i+1 \rightarrow i$

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*.

$$lMatrice \cdot uMatrice = pMatrice \cdot \text{matrice}$$

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

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

LU *m1,lower,upper,perm* Done

<i>lower</i>	$\begin{bmatrix} 1 & 0 & 0 \\ \frac{5}{6} & 1 & 0 \\ \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$
--------------	---

<i>upper</i>	$\begin{bmatrix} 6 & 12 & 18 \\ 0 & 4 & 16 \\ 0 & 0 & 1 \end{bmatrix}$
--------------	--

<i>perm</i>	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
-------------	---

le mode **Auto ou Approxé** (**Approximate**) sur Approché (Approximate), les calculs sont exécutés en virgule flottante.

- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit : $5E-14 \cdot \max(\dim(\text{Matrice})) \cdot \text{rowNorm}(\text{Matrice})$

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

M**mat@list()**

mat@list(*Matrice*)⇒*liste*

Donne la liste obtenue en copiant les éléments de *Matrice* ligne par ligne.

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

mat@list([1 2 3])	{1,2,3}
$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$
mat@list(m1)	{1,2,3,4,5,6}

max()

max(*Valeur1, Valeur2*)⇒*expression*

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

max(*Liste1, Liste2*)⇒*liste*

max(*Matrice1, Matrice2*)⇒*matrice*

Donne le maximum des deux arguments. Si les arguments sont deux listes ou matrices, donne la liste ou la matrice formée de la valeur maximale de chaque paire d'éléments correspondante.

max(*Liste*)⇒*expression*

Donne l'élément maximal de *liste*.

max({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([1 -3 7; -4 0 0.3])	[1 0 7]
--------------------------------	---------

max()

Catalogue >

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

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

mean()

Catalogue >

mean(Liste[, listeFréq])⇒expression

Donne la moyenne des éléments de *Liste*.

Chaque élément de la liste *listeFréq* totalise le nombre d'occurrences de l'élément correspondant de *Liste*.

mean(MatriceI[, matriceFréq])⇒matrice

Donne un vecteur ligne des moyennes de toutes les colonnes de *MatriceI*.

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

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

mean({0.2,0.1,-0.3,0.4})	0.26
mean({1,2,3},{3,2,1})	$\frac{5}{3}$

En mode Format Vecteur Rectangulaire :

mean	$\begin{bmatrix} 0.2 & 0 \\ -1 & 3 \\ 0.4 & -0.5 \end{bmatrix}$	$\begin{bmatrix} -0.133333 & 0.833333 \end{bmatrix}$
mean	$\begin{bmatrix} \frac{1}{5} & 0 \\ -1 & 3 \\ \frac{2}{5} & -\frac{1}{2} \\ 5 & 2 \end{bmatrix}$	$\begin{bmatrix} -\frac{2}{15} & \frac{5}{6} \end{bmatrix}$
mean	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \begin{bmatrix} 5 & 3 \\ 4 & 1 \\ 6 & 2 \end{bmatrix}$	$\begin{bmatrix} \frac{47}{15} & \frac{11}{3} \end{bmatrix}$

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(MatriceI[, matriceFréq])⇒matrice

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

median({0.2,0.1,-0.3,0.4})	0.2
----------------------------	-----

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

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

Remarques :

- tous les éléments de la liste ou de la matrice doivent correspondre à des valeurs numériques.
- Les éléments vides de la liste ou de la matrice sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

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

Calcule la ligne Med-Medy = $(m \cdot x + b)$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

Variable de sortie	Description
stat.RegEqn	Équation de ligne Med-Med : $m \cdot x + b$
stat.m, stat.b	Coefficient de modèle
stat.Resid	Valeurs résiduelles de la ligne Med-Med
stat.XReg	Liste des points de données de la liste <i>Liste X</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Frég</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.YReg	Liste des points de données de la liste <i>Liste Y</i> modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de <i>Frég</i> , <i>Liste de catégories</i> et <i>Inclure les catégories</i>
stat.FreqReg	Liste des fréquences correspondant à <i>stat.XReg</i> et <i>stat.YReg</i>

mid()

[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("Hello there",2)	"ello there"
mid("Hello there",7,3)	"the"
mid("Hello there",1,5)	"Hello"
mid("Hello there",1,0)	"[]"

mid({9,8,7,6},3)	{7,6}
mid({9,8,7,6},2,2)	{8,7}
mid({9,8,7,6},1,2)	{9,8}
mid({9,8,7,6},1,0)	{[]}

mid()**Catalogue > **

**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({ "A", "B", "C", "D" }, 2, 2)	{ "B", "C" }
-----------------------------------	--------------

min()**Catalogue > **

min(*Valeur1*, *Valeur2*)⇒*expression*

min(2.3,1.4)	1.4
--------------	-----

min(*Liste1*, *Liste2*)⇒*liste*

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

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*

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

Donne l'élément minimal de *Liste*.

min(*Matrice1*)⇒*matrice*

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

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

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

mirr()**Catalogue > **

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

list1:= { 6000, -8000, 2000, -3000 } { 6000, -8000, 2000, -3000 }	
list2:= { 2,2,2,1 } { 2,2,2,1 }	
mirr(4.65,12,5000,list1,list2)	13.41608607

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 82.

mod()

mod(Valeur1, Valeur2)⇒expression

mod(Liste1, List2)⇒liste

mod(Matrice1, Matrice2)⇒matrice

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

$$\text{mod}(x, 0) = x$$

$$\text{mod}(x, y) = x - \lfloor y \rfloor \text{ floor}(x/y)$$

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.

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

mod()**Catalogue > **

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 139

mRow()**Catalogue > **

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

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

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

mRowAdd()**Catalogue > **

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

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

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

$$Valeur \times \text{ligne } Index1 + \text{ligne } Index2$$

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 164.)

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 236.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $b_0+b_1 \cdot x_1+b_2 \cdot x_2+\dots$

Variable de sortie	Description
stat.b0, stat.b1, ...	Coefficients d'ajustement
stat.R ²	Coefficient de détermination multiple
stat.ŷListe	$\hat{y}_{\text{Liste}} = b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Valeurs résiduelles de l'ajustement

MultRegIntervals

Catalogue > 

MultRegIntervals $Y, X1[,X2[,X3,\dots,[X10]]], listeValX[, CLevel]$

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 164.)

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 236.

Variable de sortie	Description
stat.RegEqn	Équation d'ajustement : $b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 + \dots$
stat.ŷ	Prévision d'un point : $\hat{y} = b_0 + b_1 \cdot x_1 + \dots$ pour <i>listeValX</i>
stat.dfError	Degrés de liberté des erreurs
stat.CLower, stat.CUpper	Intervalle de confiance pour une réponse moyenne
stat.ME	Marge d'erreur de l'intervalle de confiance
stat.SE	Erreur type de réponse moyenne
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

Variable de sortie	Description
stat.bList	Liste de coefficients de régression, {b0,b1,b2,...}
stat.Resid	Valeurs résiduelles de l'ajustement

MultRegTests

Catalogue > 

MultRegTests $Y, X1[,X2[,X3,...[,X10]]]$

Le test de régression linéaire multiple calcule une régression linéaire multiple sur les données et donne les statistiques du F -test et du t -test globaux pour les coefficients.

Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

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 coefficients

Variable de sortie	Description
stat.tList	Liste des statistiques t pour chaque coefficient dans la liste bList
stat.PList	Liste des valeurs p pour chaque statistique t
stat.SEList	Liste des erreurs type des coefficients de la liste bList
stat.ŷListe	$\hat{y}\text{Liste} = b_0 + b_1 \cdot x_1 + \dots$
stat.Resid	Valeurs résiduelles de l'ajustement
stat.sResid	Valeurs résiduelles normalisées ; valeur obtenue en divisant une valeur résiduelle par son écart-type
stat.CookDist	Distance de Cook ; Mesure de l'influence d'une observation basée sur la valeur résiduelle et le levier
stat.Leverage	Mesure de la distance séparant les valeurs de la variable indépendante de leurs valeurs moyennes

N

nand

touches

BooleanExpr1 nand BooleanExpr2
renvoie expression booléenne

BooleanList1 nand BooleanList2
renvoie liste booléenne

BooleanMatrix1 nand BooleanMatrix2
renvoie matrice booléenne

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

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}

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

Les entiers peuvent être entrés dans tout type de base. Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe `Ob` ou `Oh`. Tout entier sans préfixe est considéré comme un nombre en écriture décimale (base 10).

nCr()**Catalogue > **

nCr(Valeur1, Valeur2)⇒expression

Pour les entiers *Valeur1* et *Valeur2* avec $Valeur1 \geq Valeur2 \geq 0$, **nCr()** donne le nombre de combinaisons de *Valeur1* éléments pris parmi *Valeur2* éléments. (Appelé aussi « coefficient binomial ».)

nCr(Valeur, 0)⇒1

nCr(Valeur, entierNég)⇒0

nCr(Valeur, entierPos)⇒Valeur · (Valeur-1)... · (Valeur-entierPos+1)/ entierPos!

nCr(Valeur, nonEntier)⇒expression!/((Valeur-nonEntier)! · nonEntier!)

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.

nCr(z,3) z=5	10
---------------------	----

nCr(z,3) z=6	20
---------------------	----

nCr({5,4,3},{2,4,2})	{10,1,3}
-----------------------------	----------

nCr()**Catalogue >** **nCr(*Matrice1*, *Matrice2*) \Rightarrow *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.

$$\text{nCr}\left[\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}\right] = \begin{bmatrix} 15 & 10 \\ 6 & 3 \end{bmatrix}$$

nDerivative()**Catalogue >** **nDerivative(*Expr1*, *Var*=*Valeur* [, *Ordre*]) \Rightarrow *valeur*****nDerivative(*Expr1*, *Var*[, *Ordre*]) | *Var*=*Valeur* \Rightarrow *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.

Si la variable *Var* ne contient pas de valeur numérique, *Valeur* doit être spécifiée.

L'*ordre* de la dérivée doit être **1** ou **2**.

Remarque : l'algorithme nDerivative() présente une limitation : il fonctionne de manière récursive à l'intérieur de l'expression non simplifiée et calcule la valeur de la dérivée première (et seconde, si cela est possible), puis évalue chacune des sous-expressions, ce qui peut générer un résultat inattendu.

nDerivative(x , x=1)	1
nDerivative(x , x=0)	undef
nDerivative(sqrt(x-1), x) x=1	undef

nDerivative(x*(x^2+x)^(1/3), x, 1) x=0	undef
centralDiff(x*(x^2+x)^(1/3), x) x=0	0.000033

nDerivative()

Catalogue >

Observez l'exemple ci-contre. La dérivée première de $x \cdot (x^2+x)^{(1/3)}$ en $x=0$ est égale à 0. Toutefois, comme la dérivée première de la sous-expression $(x^2+x)^{(1/3)}$ n'est pas définie pour $x=0$ et que cette valeur est utilisée pour calculer la dérivée de l'expression complète, **nDerivative()** signale que le résultat n'est pas défini et affiche un message d'erreur.

Si vous rencontrez ce problème, vérifiez la solution en utilisant une représentation graphique. Vous pouvez également tenter d'utiliser **centralDiff()**.

newList()

Catalogue >

newList(*nbreEléments*) \Rightarrow liste

newList(4)

{0,0,0,0}

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

newMat()

Catalogue >

**newMat(*nbreLignes*,
nbreColonnes)** \Rightarrow matrice

newMat(2,3)

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

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

nfMax()

Catalogue >

nfMax(*Expr*, *Var*) \Rightarrow valeurnfMax($-x^2 - 2 \cdot x - 1, x$)

-1.

nfMax(*Expr*, *Var*, *LimitInf*) \Rightarrow valeurnfMax($0.5 \cdot x^3 - x - 2, x, -5, 5$)

5.

**nfMax(*Expr*, *Var*, *LimitInf*,
LimitSup)** \Rightarrow valeur**nfMax(*Expr*, *Var* | *LimitInf* \leq *Var*
 \leq *LimitSup*)** \Rightarrow valeur

Donne la valeur numérique possible de la variable *Var* au point où le maximum local de *Expr* survient.

nfMax()**Catalogue >**

Si *LimitInf* et *LimitSup* sont spécifiés, la fonction recherche le maximum local dans l'intervalle fermé [*LimitInf*,*LimitSup*].

nfMin()**Catalogue >****nfMin(*Expr*, *Var*)** \Rightarrow valeur

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

nfMin(*Expr*, *Var*, *LimitInf*) \Rightarrow valeur

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

**nfMin(*Expr*, *Var*, *LimitInf*,
LimitSup)** \Rightarrow valeur

**nfMin(*Expr*, *Var*) | *LimitInf* \leq *Var*
*LimitSup*** \Rightarrow valeur

Donne la valeur numérique possible de la variable *Var* au point où le minimum local de *Expr* survient.

Si *LimitInf* et *LimitSup* sont spécifiés, la fonction recherche le minimum local dans l'intervalle fermé [*LimitInf*,*LimitSup*].

nInt()**Catalogue >****nInt(*Expr1*, *Var*, *Borne1*,
Borne2)** \Rightarrow expression

$$\text{nInt}(e^{-x^2}, x, -1, 1) = 1.49365$$

Si l'intégrande *Expr1* ne contient pas d'autre variable que *Var* et si *Borne1* et *Borne2* sont des constantes, en $+\infty$ ou en $-\infty$, alors **nInt()** donne le calcul approché de $\int(\text{Expr1}, \text{Var}, \text{Borne1}, \text{Borne2})$. Cette approximation correspond à une moyenne pondérée de certaines valeurs d'échantillon de l'intégrande dans l'intervalle *Borne1* $<$ *Var* $<$ *Borne2*.

L'objectif est d'atteindre une précision de six chiffres significatifs. L'algorithme s'adaptant, met un terme au calcul lorsqu'il semble avoir atteint cet objectif ou lorsqu'il paraît improbable que des échantillons supplémentaires produiront une amélioration notable.

$$\text{nInt}(\cos(x), x, \pi, \pi + 1 \cdot 10^{-12}) = 1.04144 \cdot 10^{-12}$$

Le message « Précision incertaine » s'affiche lorsque cet objectif ne semble pas atteint.

Il est possible de calculer une intégrale multiple en imbriquant plusieurs appels **nInt()**. Les bornes d'intégration peuvent dépendre des variables d'intégration les plus extérieures.

$$\text{nInt}\left(\text{nInt}\left(\frac{e^{-x \cdot y}}{\sqrt{x^2 - y^2}}, y, -x, x\right), x, 0, 1\right) \quad 3.30423$$

nom()

nom(tauxEffectif,CpY)⇒valeur

nom(5.90398,12)

5.75

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

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

Remarque : voir également **eff()**, page 48.

nor

touches

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.

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} or {3,2,1}	{3,2,3}
{1,2,3} nor {3,2,1}	{-4,-3,-4}

norm()

Catalogue >

norm(Matrice)⇒expression

norm(Vecteur)⇒expression

Donne la norme de Frobenius.

norm($\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$)	5.47723
norm($\begin{bmatrix} 1 & 2 \end{bmatrix}$)	2.23607
norm($\begin{bmatrix} 1 \\ 2 \end{bmatrix}$)	2.23607

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 $P(X \leq upBound)$, définissez *lowBound* = -9E999.

normPdf($ValX$, μ , σ) \Rightarrow nombre si $ValX$ est un nombre, liste si $ValX$ est une liste

Calcule la densité de probabilité de la loi normale à la valeur $ValX$ spécifiée pour les paramètres μ et σ .

not

Catalogue > 

not Valeur1 \Rightarrow nombre

not (2>3)	true
not 0hB0►Base16	0hFFFFFFFFFFFFFFF4F
not not 2	2

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

not *Entier* \Rightarrow *entier*

En mode base Hex :

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.

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

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).

not 0h7AC36 0hFFFFFFFFFFFF853C9

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 17.

En mode base Bin :

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(Valeur1, Valeur2)⇒expression

$nPr(z, 3) z=5$	60
$nPr(z, 3) z=6$	120
$nPr(\{5, 4, 3\}, \{2, 4, 2\})$	$\{20, 24, 6\}$
$nPr(\begin{bmatrix} 6 & 5 \\ 4 & 3 \end{bmatrix}, \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix})$	$\begin{bmatrix} 30 & 20 \\ 12 & 6 \end{bmatrix}$

Pour les entiers *Valeur1* et *Valeur2* avec $Valeur1 \geq Valeur2 \geq 0$, **nPr()** donne le nombre de permutations de *Valeur1* éléments pris parmi *Valeur2* éléments.

nPr(Valeur, 0)⇒1

nPr()**Catalogue > **

**nPr(Valeur, entierNég) \Rightarrow 1 /
 $((Valeur+1) \cdot (Valeur+2) \cdots$
 $(Valeur-entierNég))$**

**nPr(Valeur, entierPos) \Rightarrow Valeur ·
 $(Valeur-1) \cdots (Valeur-entierPos+1)$**

**nPr(Valeur, nonEntier) \Rightarrow Valeur! /
 $(Valeur-nonEntier)!$**

nPr(Liste1, Liste2) \Rightarrow 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.

nPr(Matrice1, Matrice2) \Rightarrow 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.

nPr({5,4,3},{2,4,2}) {20,24,6}

nPr[[6 5],[4 3],[2 2]] [30 20]
[12 6]

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.

MTO correspond au mouvement de trésorerie initial à l'heure 0 ; il doit s'agir d'un nombre réel.

**list1:= {6000,-8000,2000,-3000}
{6000,-8000,2000,-3000}**

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

npv(10,5000,list1,list2) 4769.91

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.

nSolve()Catalogue > 

nSolve(Équation,Var[=Condition])⇒ chaîne_nombre ou erreur

nSolve($x^2+5 \cdot x - 25 = 9, x$)	3.84429
nSolve($x^2=4, x=-1$)	-2.
nSolve($x^2=4, x=1$)	2.

nSolve(Équation,Var[=Condition],LimitInf,LimitSup)⇒ chaîne_nombre ou erreur

Remarque : si plusieurs solutions sont possibles, vous pouvez utiliser une condition pour mieux déterminer une solution particulière.

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()**Catalogue > **

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.

$\text{nSolve}(x^2+5 \cdot x-25=9, x)$	-8.84429
$\text{nSolve}\left(\frac{(1+r)^{24}-1}{r}=26, r\right)$	$r > 0$ and $r < 0.25$ 0.006886
$\text{nSolve}(x^2=-1, x)$	"No solution found"

O**OneVar****Catalogue > **

OneVar [*1, X1, 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 164.)

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 à un élément vide dans l'ensemble des listes résultantes. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

Variable de sortie	Description
stat. \bar{x}	Moyenne des valeurs x
stat. Σx	Somme des valeurs x
stat. Σx^2	Somme des valeurs x^2 .
stat.sx	Écart-type de l'échantillon de x
stat. x	Écart-type de la population de x
stat.n	Nombre de points de données
stat.MinX	Minimum des valeurs de x
stat.Q ₁ X	1er quartile de x
stat.MedianX	Médiane de x
stat.Q ₃ X	3ème quartile de x
stat.MaxX	Maximum des valeurs de x
stat.SSX	Somme des carrés des écarts par rapport à la moyenne de x

or

BooleanExpr1 or *BooleanExpr2* renvoie expression booléenne

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.

Define $g(x)=\text{Func}$	<i>Done</i>
If $x \leq 0$ or $x \geq 5$	
Goto <i>end</i>	
Return $x \cdot 3$	
Lbl <i>end</i>	
EndFunc	

$g(3)$	9
$g(0)$	A function did not return a value

Donne true si la simplification de l'une des deux ou des deux expressions est vraie. Donne false uniquement si la simplification des deux expressions est fausse.

Remarque : voir xor.

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

Entier1 or *Entier2* => *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 17.

Remarque : voir xor.

En mode base Hex :

0h7AC36 or 0h3D5F	0h7BD7F
-------------------	---------

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

En mode base Bin :

0b100101 or 0b100	0b100101
-------------------	----------

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

ord()**ord(Chaîne)⇒entier****ord(ListeL)⇒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()****P▶Rx(ExprR, θExpr)⇒expression****P▶Rx(ListeR, θListe)⇒liste****P▶Rx(MatriceR, θMatrice)⇒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 $^{\circ}$, $^{\text{G}}$ ou $^{\text{r}}$ 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 :

P▶Rx(4,60°)	2.
P▶Rx({ -3,10,1.3 }, { $\frac{\pi}{3}, \frac{\pi}{4}, 0 $ })	{ -1.5,7.07107,1.3 }

P▶Ry()**P▶Ry(ValeurR, θValeur)⇒valeur****P▶Ry(ListeR, θListe)⇒liste****P▶Ry(MatriceR, θMatrice)⇒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.

En mode Angle en radians :

P▶Ry(4,60°)	3.4641
P▶Ry({ -3,10,1.3 }, { $\frac{\pi}{3}, \frac{\pi}{4}, 0 $ })	{ -2.59808,-7.07107,0 }

Remarque : vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **P@>Ry (...)**.

PassErr

Catalogue >

PassErr

Passe une erreur au niveau suivant.

Pour obtenir un exemple de **PassErr**, reportez-vous à l'exemple 2 de la commande **Try**, page 176.

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 24 et **Try**, page 176.

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

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.

Define $p(x) = \begin{cases} x, & x > 0 \\ \text{undef}, & x \leq 0 \end{cases}$	Done
$p(1)$	1
$p(-1)$	undef

Remarque : voir aussi **Modèle Fonction définie par morceaux**, page 3.

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
 \leq *upBound*)⇒nombre si la borne *upBound*
 est un nombre, liste si la borne *upBound* est
 une liste

Calcule la probabilité cumulée d'une variable
 suivant une loi 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

[1 3.]	►Polar	[3.16228 ∠ 71.5651]
--------	--------	---------------------

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

Affiche *vecteur* sous forme polaire
 $[r\angle \theta]$. Le vecteur doit être un vecteur ligne ou colonne et de dimension 2.

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 136.

valeurComplexe ►Polar

Affiche *valeurComplexe* sous forme polaire.

- Le mode Angle en degrés affiche $(r\angle \theta)$.

En mode Angle en radians :

$(3+4\cdot i)$	►Polar	$e^{0.927295\cdot i\cdot 5}$
----------------	--------	------------------------------

$\left(4 \angle \frac{\pi}{3}\right)$	►Polar	$e^{1.0472\cdot i\cdot 4}$
---------------------------------------	--------	----------------------------

- Le mode Angle en radians affiche $r\text{e}^{i\theta}$.

valeurComplexe peut prendre n'importe quelle forme complexe. Toutefois, une entrée $r\text{e}^{i\theta}$ génère une erreur en mode Angle en degrés.

Remarque : vous devez utiliser les parenthèses pour les entrées polaires $(r\angle \theta)$.

En mode Angle en grades :

$(4\angle i)\blacktriangleright\text{Polar}$

$(4\angle 100.)$

En mode Angle en degrés :

$(3+4\cdot i)\blacktriangleright\text{Polar}$

$(5\angle 53.1301)$

polyEval()

polyEval(Liste1, Expr1)⇒expression

$\text{polyEval}(\{1,2,3,4\},2)$ 26

polyEval(Liste1, Liste2)⇒expression

$\text{polyEval}(\{1,2,3,4\},\{2,-7\})$ {26,-262}

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.

polyRoots()

polyRoots(Poly,Var)⇒liste

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

polyRoots(ListeCoeff)⇒liste

$\text{cPolyRoots}(y^3+1,y)$ {-1,0.5-0.866025•i,0.5+0.866025•i}

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 : {}.

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

Poly doit être un polynôme d'une seule variable, dans sa forme développée. N'utilisez pas les formats non développés comme $y^2\cdot y+1$ ou $x\cdot x+2\cdot x+1$.

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

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 33.

PowerReg X,Y[, Fréq] [, Catégorie, Inclure]]

Effectue l'ajustement exponentiel = $(a \cdot (x)^b)$ sur les listes *X* et *Y* en utilisant la fréquence *Fréq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

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

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

Prgm

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

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)

4560	60
60	30
30	0
GCD=30	

Done

prodSeq()

Voir **Π()**, page 208.

Product (Π)

Voir **Π()**, page 208.

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(MatriceI[, Début[, Fin]])⇒matrice

Donne un vecteur ligne contenant les produits des éléments ligne par ligne de *MatriceI*. *Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage de colonnes.

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

product({1,2,3,4})	24
product({4,5,8,9},2,3)	40

product($\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$)	[28 80 162]
--	-------------

product($\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, 1,2$)	[4 10 18]
---	-----------

propFrac()**Catalogue >**

propFrac(ValeurI[, Var])⇒valeur

propFrac(nombre_rationnel) décompose *nombre_rationnel* sous la forme de la somme d'un entier et d'une fraction de même signe et dont le dénominateur est supérieur au numérateur (fraction propre).

propFrac(expression_rationnelle,Var) donne la somme des fractions propres et d'un polynôme par rapport à *Var*. Le degré de *Var* dans le dénominateur est supérieur au degré de *Var* dans le numérateur pour chaque fraction propre. Les mêmes puissances de *Var* sont regroupées. Les termes et leurs facteurs sont triés, *Var* étant la variable principale.

propFrac($\frac{4}{3}$)	$1\frac{1}{3}$
propFrac($\frac{-4}{3}$)	$-1\frac{1}{3}$

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.

Vous pouvez utiliser la fonction **propFrac()** pour représenter des fractions mixtes et démontrer l'addition et la soustraction de fractions mixtes.

$\text{propFrac}\left(\frac{11}{7}\right)$	$1\frac{4}{7}$
$\text{propFrac}\left(3+\frac{1}{11}+5+\frac{3}{4}\right)$	$8\frac{37}{44}$
$\text{propFrac}\left(3+\frac{1}{11}-\left(5+\frac{3}{4}\right)\right)$	$-2\frac{29}{44}$

Q**QR**

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 Approxé (Approximate)** sur Approxé (Approximate), les calculs sont exécutés en virgule flottante.
- Si Tol est omis ou inutilisé, la tolérance par défaut est calculée comme suit : $5E-14 \cdot \max(\dim(\text{Matrice})) \cdot \text{rowNorm}(\text{Matrice})$

Le nombre en virgule flottante (9.) dans m1 fait que les résultats seront tous calculés en virgule flottante.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
QR m1,qm,rm	Done
qm	$\begin{bmatrix} 0.123091 & 0.904534 & 0.408248 \\ 0.492366 & 0.301511 & -0.816497 \\ 0.86164 & -0.301511 & 0.408248 \end{bmatrix}$
rm	$\begin{bmatrix} 8.12404 & 9.60114 & 11.0782 \\ 0. & 0.904534 & 1.80907 \\ 0. & 0. & 0. \end{bmatrix}$

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*.

QuadReg

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 164.)

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 numériques ou alphanumériques 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 236.

Variable de sortie	Description
--------------------	-------------

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

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 164.)

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 numériques ou alphanumériques 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 236.

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

R**R►Pθ()**

Catalogue >

R►Pθ (valeursx, valeury) ⇒ valeur

En mode Angle en degrés :

R►Pθ (listex, listey) ⇒ liste

R►Pθ(2,2)

45.

R►Pθ (matricex, matricey) ⇒ matrice

Donne la valeur de l'ordonnée θ - du point de coordonnées rectangulaires (x,y).

En mode Angle en grades :

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

R►Pθ(2,2)

50.

Remarque : Vous pouvez insérer cette fonction à partir du clavier de l'ordinateur en entrant **R@>Ptheta (...)**.

En mode Angle en radians et en mode Auto :

R►Pθ(3,2)

0.588003

R►Pθ([3 -4 2], [0 π
4 1.5])

[0. 2.94771 0.643501]

R►Pr()**Catalogue > **

- R►Pr** (*valeursx, valeury*) \Rightarrow *valeur*
R►Pr (*listex, listey*) \Rightarrow *liste*
R►Pr (*matricex, matricey*) \Rightarrow *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 et en mode Auto :

R►Pr(3,2) 3.60555

R►Pr

$$\begin{bmatrix} 3 & -4 & 2 \end{bmatrix}, \begin{bmatrix} 0 & \frac{\pi}{4} & 1.5 \end{bmatrix}$$

$$\begin{bmatrix} 3 & 4.07638 & \frac{5}{2} \end{bmatrix}$$

► Rad**Catalogue > **

Valeur1►Rad \Rightarrow *valeur*

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)►Rad (0.02618)^r

En mode Angle en grades :

(1.5)►Rad (0.023562)^r

rand()**Catalogue > **

rand() \Rightarrow *expression*
rand(#Trials) \Rightarrow *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 Done
rand(2) {0.158206,0.717917}

randBin()**Catalogue > **

randBin(*n, p*) \Rightarrow *expression*
randBin(*n, p, #Trials*) \Rightarrow *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(80,0.5) 46.
randBin(80,0.5,3) {43.,39.,41.}

randInt()**Catalogue > **

```
randInt
(
lowBound,upBound)
⇒ expression
randInt
(
LimiteInf
,
LimiteSup
,NbrEssais) ⇒ 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)	3.
randInt(3,10,4)	{9.,3.,4.,7.}

randMat()**Catalogue > **

```
randMat(nbreLignes, nbreColonnes)
⇒ matrice
```

Donne une matrice d'entiers compris entre -9 et 9 de la dimension spécifiée.

Les deux arguments doivent pouvoir être simplifiés en entiers.

RandSeed 1147	Done
randMat(3,3)	$\begin{bmatrix} 8 & -3 & 6 \\ -2 & 3 & -6 \\ 0 & 4 & -6 \end{bmatrix}$

Remarque : Les valeurs de cette matrice changent chaque fois que l'on appuie sur **del**.

randNorm()

randNorm(μ, σ) \Rightarrow expression
randNorm($\mu, \sigma, nbreEssais$) \Rightarrow liste

randNorm(μ, σ) Donne un nombre décimal issu de la loi normale spécifiée. Il peut s'agir de tout nombre réel, mais le résultat obtenu sera essentiellement compris dans l'intervalle $[\mu - 3\sigma, \mu + 3\sigma]$.

randNorm($\mu, \sigma, nbreEssais$) 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()

randPoly(*Var*, *Order*) \Rightarrow expression

Donne un polynôme aléatoire de la variable *Var* de degré *Ordre* spécifié. Les coefficients sont des entiers aléatoires situés dans la plage -9 à 9. Le coefficient du terme de plus au degré (*Order*) sera non nul.

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()

randSamp(*List*,#*Trials*[,*noRepl*]) \Rightarrow liste

Donne une liste contenant un échantillon aléatoire de *nbreEssais* éléments choisis dans *Liste* avec option de remise (*sansRem*=0) ou sans option de remise (*sansRem*=1). L'option par défaut est avec remise.

Define <i>list3</i> ={1,2,3,4,5}	Done
Define <i>list4</i> =randSamp(<i>list3</i> ,6)	Done
<i>list4</i>	{1.,3.,3.,1.,3.,1.}

RandSeed

RandSeed *Nombre*

RandSeed 1147	Done
rand()	0.158206

Si *Nombre* = 0, réinitialise le générateur de nombres aléatoires Si *Nombre* ≠ 0, il sert à générer deux germes qui sont stockés dans les variables système seed1 et seed2

real()

Catalogue >

real(*Value1*) ⇒ *valeur*

real(2+3·i)

2

Donne la partie réelle de l'argument.

real({1+3·i,3,i})

{1,3,0}

real(*List1*) ⇒ *liste*

Donne les parties réelles de tous les éléments.

real(*Matrix1*) ⇒ *matrice*

Donne les parties réelles de tous les éléments.

real([1+3·i 3
2 i])

[1 3
2 0]

► Rect

Catalogue >

Vecteur ► **Rect**

Remarque : Vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @>Rect

[3 π π
— — —] ► Rect

[1.06066 1.06066 2.59808]

Affiche *Vecteur* en coordonnées rectangulaires [x, y, z]. Le vecteur doit être un vecteur ligne ou colonne de dimension 2 ou 3.

Remarque : ► Rect est uniquement une instruction d'affichage et non une fonction de conversion. On ne peut l'utiliser qu'à la fin d'une ligne et elle ne modifie pas le contenu du registre ans.

Remarque : Voir également ► Polar, page 124.

complexValue ► **Rect**

En mode Angle en radians et en modes Auto :

► Rect

Catalogue >

Affiche *valeurComplexe* sous forme rectangulaire ($a+bi$) La *valeurComplexe* peut prendre n'importe quelle forme rectangulaire Toutefois, une entrée $re^{i\theta}$ génère une erreur en mode Angle en degrés

Remarque : Vous devez utiliser des parenthèses pour les entrées en polaire ($r \angle \theta$).

$\left(4 \cdot e^{\frac{\pi}{3}}\right)$	►Rect	11.3986
$\left(\left(4 \angle \frac{\pi}{3}\right)\right)$	►Rect	$2 + 3.4641 \cdot i$

En mode Angle en grades :

$\left(\left(1 \angle 100\right)\right)$	►Rect	i
--	-------	-----

En mode Angle en degrés :

$\left(\left(4 \angle 60\right)\right)$	►Rect	$2 + 3.4641 \cdot i$
---	-------	----------------------

Remarque : Pour taper \angle à partir du clavier, sélectionnez-le dans la liste des symboles du Catalogue.

ref()

Catalogue >

ref(*Matrix1[, Tol]*) ⇒ matrice

Donne une réduite de Gauss de la matrice *Matrix1*.

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

$$\text{ref} \begin{pmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{pmatrix} \quad \begin{pmatrix} 1 & -\frac{2}{5} & -\frac{4}{5} & \frac{4}{5} \\ 0 & 1 & \frac{4}{7} & \frac{11}{7} \\ 0 & 0 & 1 & -\frac{62}{71} \end{pmatrix}$$

- Si vous utilisez **ctrl** . **eX**
esc ↵ **EE** **EE** **10^X** **10^X** **tab**
^ **tab** **esc** **tab** Auto ou
Approché sur Approché, les calculs sont exécutés en virgule flottante
- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit :
 $5E-14 \cdot \max(\dim(\text{Matrix1})) \cdot \text{rowNorm}(\text{Matrix1})$

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 :

$$\text{ref} \begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & \frac{1}{a} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Un message d'avertissement est affiché car l'élément $1/a$ n'est pas valide pour $a=0$.

Pour éviter ce problème, vous pouvez stocker préalablement une valeur dans *a* ou utiliser l'opérateur "sachant que" ("|") pour substituer une valeur, comme illustré dans l'exemple suivant.

$$\text{ref} \begin{bmatrix} a & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} | a=0 \quad \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Remarque : Voir également **rref()**, page 148.

RefreshProbeVars

Catalogue >

RefreshProbeVars

Vous permet d'accéder aux données de capteur à partir de toutes les sondes de capteur connectées à l'aide de votre programme TI-Basic.

Valeur StatusVar	État
<i>statusVar</i> = 0	Normal (Poursuivez le programme)
<i>statusVar</i>	L'application Vernier

Par exemple

```
Define temp ()=
Prgm
    © Vérifier si le système est prêt
    RefreshProbeVars status
    Si le statut=0 alors
        Disp "prêt"
        For n,1,50
```

Valeur StatusVar	État
=1	DataQuest™ est en mode Acquisition de données. Remarque : L'application Vernier DataQuest™ doit être en mode compteur pour que cette commande fonctionne. 
=2	L'application Vernier DataQuest™ n'est pas lancée.
=3	L'application Vernier DataQuest™ est lancée, mais vous n'avez pas encore connecté de sonde.

```

RefreshProbeVars status
température:=compteur.température
Disp "Température: ",température
Si la température>30 alors
Disp "Trop chaude"
EndIf
© Attendre pendant 1 seconde entre
les échantillons
Wait 1
EndFor
Else
Disp "Pas prêt. Réessayer plus
tard"
EndIf
EndPrgm

```

Remarque : Ceci peut également être utilisé avec le TI-Innovator™ Hub.

remain()

remain(Value1, Value2) ⇒ valeur
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)

Par conséquent, remarquez 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 105.

remain(7,0)	7
remain(7,3)	1
remain(-7,3)	-1
remain(7,-3)	1
remain(-7,-3)	-1
remain({12,-14,16},{9,7,-5})	{3,0,1}

$$\text{remain}\left[\begin{bmatrix} 9 & -7 \\ 6 & 4 \end{bmatrix}, \begin{bmatrix} 4 & 3 \\ 4 & -3 \end{bmatrix}\right] = \begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$

Request *promptString, var[, DispFlag [, statusVar]]*

Request *promptString, func(arg1, ...argn) [, DispFlag [, statusVar]]*

Commande de programmation : Marque une pause dans l'exécution du programme et affiche une boîte de dialogue contenant le message *chaîneinvite*, ainsi qu'une zone de saisie destinée à la réponse que doit fournir l'utilisateur.

Lorsque l'utilisateur saisit une réponse et clique sur **OK**, le contenu de la zone de saisie est affecté à la variable *var*.

Si l'utilisateur clique sur **Annuler**, le programme continue sans accepter aucune entrée. Le programme utilise la valeur précédente de la variable *var* si *var* était déjà définie.

L'argument optionnel *IndicAff* peut correspondre à toute expression.

- Si *IndicAff* est omis ou a pour valeur **1**, le message d'invite et la réponse de l'utilisateur sont affichés dans l'historique de l'application Calculs.
- Si *IndicAff* a pour valeur **0**, le message d'invite et la réponse de l'utilisateur ne sont pas affichés dans l'historique.

L'argument optionnel *VarÉtat* indique au programme comment déterminer si l'utilisateur a fermé la boîte de dialogue. Notez que *VarÉtat* nécessite la saisie de l'argument *IndicAff*.

- Si l'utilisateur a cliqué sur **OK**, ou a appuyé sur **Entrée** ou sur **Ctrl+Entrée**, la variable *VarÉtat* prend la valeur **1**.
- Sinon, la variable *StatusVar* prend la valeur **0**.

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 :

Demi-droite : 6/2
Area= 28.2743

Définissez un programme :

```
Define polynomial()=Prgm
  Request "Saisissez un polynôme en x : ",p(x)
  Disp "Les racines réelles sont : ",polyRoots(p(x),x)
EndPrgm
```

Exécutez le programme et saisissez une réponse :

`polynomial()`

L'argument de *func()* permet à un programme de stocker la réponse de l'utilisateur sous la forme d'une définition de fonction. Cette syntaxe équivaut à l'exécution par l'utilisateur de la commande suivante :

Definir func(arg1, ...argn) = réponse de l'utilisateur

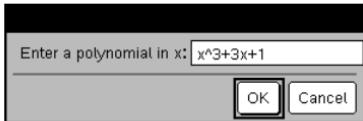
Le programme peut alors utiliser la fonction définie *fond()*. La *chaîneinvite* doit guider l'utilisateur pour la saisie d'une *réponse* appropriée qui complète la définition de la fonction.

Remarque : Vous pouvez utiliser l'**Request** commande dans un programme créé par l'utilisateur, mais pas dans une fonction.

Pour arrêter un programme qui contient une commande **Request** dans une boucle infinie :

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

Remarque : Voir également **RequestStr**, page 141.



Résultat après avoir saisi x^3+3x+1 et sélectionné **OK** :

Les racines réelles sont : $\{-0.322185\}$

RequestStr *chaîneinvite, var[, IndicAff]*

Définissez un programme :

```
Define requestStr_demo()=Prgm
  RequestStr "Votre nom :",name,0
  Disp "La réponse comporte ",dim
  (name)," caractères."
EndPrgm
```

Commande de programmation : Fonctionne de façon similaire à la première syntaxe de la commande **Request**, excepté que la réponse de l'utilisateur est toujours interprétée comme une chaîne de caractères. Par contre, la commande **Request** interprète la réponse comme une expression, à moins que l'utilisateur ne la saisisse entre guillemets ("").

Remarque : Vous pouvez utiliser la commande **RequestStr** dans un programme créé par l'utilisateur, mais pas dans une fonction.

Pour arrêter un programme qui contient une commande **RequestStr** dans une boucle infinie :

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

Remarque : Voir également **Request**, page 140.

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 [Expr]

Donne *Expr* comme résultat de la fonction. S'utilise dans les blocs

Func...EndFunc.

Remarque : Vous pouvez utiliser **Return** sans argument dans un bloc **Prgm...EndPrgm** pour quitter un programme

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

Define **factorial (nn)=**

Func

Local *answer,counter*

1 → *answer*

For *counter,1,nn*

answer· counter → *answer*

EndFor

Return *answer*

EndFunc

factorial (3)

6

right()**right(Liste1[, Nomb]) ⇒ liste**

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

{3,-2,4}

Donne les *Nomb* éléments les plus à droite de la liste *Liste1*.

Si *Nomb* est absent, on obtient *Liste1*.

right(chaîneSrce[,Nomb]) ⇒ chaîne

right("Hello",2)

"lo"

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.

rk23 ()**rk23(Expr, Var, depVar, {Var0, VarMax}, depVar0, VarStep [, diftol]) ⇒ matrice**

Équation différentielle :

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

rk23(0.001·y·(100-y),t,y,{0,100},10,1)

0.	1.	2.	3.	4,
10.	10.9367	11.9493	13.042	14.2

rk23(SystemOfExpr, Var, ListOfDepVars, {Var0, VarMax}, ListOfDepVars0, VarStep[, diftol]) ⇒

matrix

rk23(*ListExpr*, *Var*, *ListDepVars*, {*Var0*, *VarMax*}, *ListDepVars0*, *VarStep*[, *diftol*]) \Rightarrow matrice

Utilise la méthode de Runge-Kutta pour résoudre le système d'équations.

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

with *depVar*(*Var0*)=*depVar0* pour l'intervalle [*Var0*,*VarMax*]. Retourne une matrice dont la première ligne définit les valeurs de sortie de *Var*, définies à partir de *IncVar*. La deuxième ligne définit la valeur du premier composant de la solution aux valeurs *Var* correspondantes, etc.

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.

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

Même équation avec *TolErr* définie à 1.E-6

rk23(0.001,y,{100-y},t,y,{0,100},10,1,1.E-6)

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 10. & 10.9367 & 11.9495 & 13.0423 & 14.2189 \end{bmatrix}$$

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 $y2(0)=5$

rk23($\begin{cases} y1' = 0.1 \cdot y1 \cdot y2 \\ 3 \cdot y2 - y1 \cdot y2 \end{cases}$, t, {y1, y2}, {0, 5}, {2, 5}, 1)

$$\begin{bmatrix} 0. & 1. & 2. & 3. & 4. \\ 2. & 1.94103 & 4.78694 & 3.25253 & 1.82848 \\ 5. & 16.8311 & 12.3133 & 3.51112 & 6.27245 \end{bmatrix}$$

Si $IncVar$ est un nombre différent de zéro, $sign(IncVar) = signe(MaxVar - Var0)$ 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).

root()

root(*Valeur*) \Rightarrow racine

root(Valeur1, Valeur2) \Rightarrow racine

root(*Valeur*) affiche la racine carrée de *Valeur*.

root(*Valeur1*, *Valeur2*) affiche la racine *Valeur2*-ième de *Valeur1*. *Valeur1* peut être un nombre réel ou complexe en virgule flottante, un entier ou une constante rationnelle complexe.

Remarque : Voir aussi Modèle Racine n-ième, page 2.

Catalogue > 

$\sqrt[3]{8}$

?

3

144225

rotate()

Catalogue >

rotate(*EntierI*[, *NbreRotations*]) \Rightarrow entier

En mode hache Bin :

Permute les bits de la représentation binaire d'un entier. Vous pouvez saisir *Entier1* dans un système de numération quelconque ; il est converti automatiquement en une forme binaire 64 bits signée. Si *Entier1* est trop important pour être codé, il est ramené à l'aide d'une congruence dans la plage appropriée Pour plus d'informations, consultez la section ► **Base2**, page 17.

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

rotate()

Si *nbreRotations* est positif, la permutation circulaire s'effectue vers la gauche Si *nbreRotations* est négatif, la permutation circulaire s'effectue vers la droite La valeur par défaut est -1 (permutation circulation de un bit vers la droite)

Par exemple, dans une permutation circulaire vers la droite :

Chaque bit est permuted vers la droite.

0b00000000000001111010110000110101

Le bit le plus à droite passe à la position la plus à gauche.

donne :

0b10000000000000111101011000011010

Le résultat s'affiche suivant le mode Base utilisé.

rotate(Liste1[NbreRotations]) ⇒ liste

Donne une copie de *Liste1* dont les éléments ont été permuted 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)

rotate(Chaîne1[nbreRotations]) ⇒ chaîne

Donne une copie de *Chaîne1* dont les caractères ont été permuted circulairement vers la gauche ou vers la droite de *nbreRotations* caractères. Ne modifie en rien *Chaîne1*

En mode base Hex :

rotate(0h78E)	0h3C7
rotate(0h78E,-2)	0h8000000000000001E3
rotate(0h78E,2)	0h1E38

Important : Pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h (zéro, pas la lettre O).

En mode base Dec :

rotate({1,2,3,4})	{4,1,2,3}
rotate({1,2,3,4},2)	{3,4,1,2}
rotate({1,2,3,4},1)	{2,3,4,1}

rotate("abcd")	"dabc"
rotate("abcd",-2)	"cdab"
rotate("abcd",1)	"bcda"

rotate()

Catalogue >

Si *nbreRotations* est positif, la permutation circulaire s'effectue vers la gauche Si *nbreRotations* est négatif, la permutation circulaire s'effectue vers la droite La valeur par défaut est -1 (permutation vers la droite d'un caractère).

round()

Catalogue >

round(*Valeur1[, chiffres]*) ⇒ *valeur*

round(1.234567,3)

1.235

Arrondit l'argument au nombre de chiffres n spécifié après la virgule.

chiffres doit être un entier compris dans la plage 0–12. Si *chiffres* est absent, affiche l'argument arrondi à 12 chiffres significatifs.

Remarque : Le mode d'affichage des chiffres peut affecter le résultat affiché.

round(*List1[, chiffres]*) ⇒ *liste*

round({π, √2, ln(2)}, 4)

{3.1416, 1.4142, 0.6931}

Donne la liste des éléments arrondis au nombre de chiffres spécifié.

round(*Matrice1[, chiffres]*) ⇒ *matrice*

round([ln(5) ln(3), π, e^1], 1)

[1.6 1.1
3.1 2.7]

Donne une matrice des éléments arrondis au nombre de chiffres n spécifié..

rowAdd()

Catalogue >

rowAdd(*Matrice1, rIndex1, rIndex2*) ⇒ *matrice*

rowAdd([3 4, -3 2], 1, 2)

[3 4
0 2]

Donne une copie de *Matrice1* obtenue en remplaçant dans la matrice la ligne *IndexL2* par la somme des lignes *IndexL1* et *IndexL2*

rowDim()**Catalogue >** **rowDim(*Matrix*)** \Rightarrow *expression*Donne le nombre de lignes de *Matrixe*.**Remarque :** Voir aussi **colDim()**, page 25.

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

normeLig**Catalogue >** **rowNorm(*Matrixe*)** \Rightarrow *expression*Donne le maximum des sommes des valeurs absolues des éléments de chaque ligne de *Matrixe*.

rowNorm	$\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}$	25
---------	---	----

Remarque : La matrice utilisée ne doit contenir que des éléments numériques.
Voir aussi **colNorm()** page 26.**rowSwap()****Catalogue >** **rowSwap(*Matrixe1*, *IndexL1*, *IndexL2*)**
 \Rightarrow *matrice*Donne la matrice *Matrixe1* obtenue en échangeant les lignes *IndexL1* et *IndexL2*.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow mat$	$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$
rowSwap(<i>mat</i> , 1, 3)	$\begin{bmatrix} 5 & 6 \\ 3 & 4 \\ 1 & 2 \end{bmatrix}$

rref()**Catalogue >** **rref(*Matrixe1*[, *Tol*])** \Rightarrow *matrice*Donne la réduite de Gauss-Jordan de *Matrixe1*.

rref	$\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 & \frac{66}{71} \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & \frac{-62}{71} \end{bmatrix}$
------	--	---

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**.

 Auto ou
Approché sur Approché, les calculs sont exécutés en virgule flottante
- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit :
 $5E-14 \cdot \max(\dim(\text{Matrice}I)) \cdot \text{rowNorm}(\text{Matrice}I)$

Remarque : Voir aussi **ref()** page 137.

S

sec()

Touche

sec(ValeurI) ⇒ valeur

En mode Angle en degrés :

sec(ListeI) ⇒ liste

1.41421

Affiche la sécante de *ValeurI* ou retourne la liste des sécantes des éléments de *ListeI*.

{1.00015,1.00081,1.00244}

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

sec⁻¹()

Touche

sec⁻¹(ValeurI) ⇒ valeur

En mode Angle en degrés :

sec⁻¹(ListeI) ⇒ liste

0.

Affiche l'angle dont la sécante correspond à *ValeurI* ou retourne la liste des arcs sécantes des éléments de *ListeI*.

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

En mode Angle en grades :

50.

En mode Angle en radians :

sec⁻¹(*)*

Touche

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

sec⁻¹{1,2,5} {0,1.0472,1.36944}

sech(*)*

Catalogue >

sech(Valeur1) ⇒ valeur

sech(Liste1) ⇒ liste

Affiche la sécante hyperbolique de *Valeur1* ou retourne la liste des sécantes hyperboliques des éléments de *Liste1*.

sech(3)	0.099328
sech({1,2,3,4})	{0.648054,0.198522,0.036619}

sech⁻¹(*)*

Catalogue >

sech⁻¹(Valeur1) ⇒ valeur

sech⁻¹(Liste1) ⇒ liste

Retourne l'argument sécante hyperbolique de *Valeur1* retourne la liste des arguments sécante hyperbolique 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 :

sech ⁻¹ (1)	0
sech ⁻¹ {1,-2,2,1}	{0,2.0944·i,8.e-15+1.07448·i}

Send

Menu hub

SendexprOrString1 [, exprOrString2] ...

Commande de programmation : envoie une ou plusieurs TI-Innovator™ Hub commandes à un hub connecté.

exprOrString doit être une commande TI-Innovator™ Hub valide. En général, *exprOrString* contient une commande "SET ..." pour contrôler un appareil ou une commande "READ ..." pour demander des données.

Les arguments sont envoyés au hub les uns après les autres.

Exemple : allumer l'élément bleu de la DEL RGB intégrée pendant 0,5 seconde.

Send "SET COLOR.BLUE ON TIME .5"

Done

Exemple : demander la valeur actuelle du capteur intégré du niveau de lumière du hub. Une commande **Get** récupère la valeur et l'affecte à la variable *lightval*.

Send "READ BRIGHTNESS"	Done
Get <i>lightval</i>	Done
<i>lightval</i>	0.347922

Send

Menu hub

Remarque : vous pouvez utiliser la commande **Send** dans un programme défini par l'utilisateur, mais pas dans une fonction.

Remarque : voir également **Get** (page 65), **GetStr** (page 73) et **eval()** (page 52).

Exemple : envoyer une fréquence calculée au haut-parleur intégré du hub. Utilisez la variable spéciale `iostrostr.SendAns` pour afficher la commande du hub avec l'expression évaluée.

<code>n:=50</code>	50
<code>m:=4</code>	4
<code>Send "SET SOUND eval(m·n)"</code>	<i>Done</i>
<code>iostrostr.SendAns</code>	"SET SOUND 200"

seq()

Catalogue >

seq(Expr, Var, Début, Fin[, Incrément])⇒liste

Incrémente la valeur de Var comprise entre *Début* et *Fin* en fonction de l'incrément (*Inc*) spécifié et affiche le résultat sous forme de liste. Le contenu initial de *Var* est conservé après l'application de **seq()**.

La valeur par défaut de *Inc* = 1.

$\text{seq}\left(n^2, n, 1, 6\right)$	{1, 4, 9, 16, 25, 36}
$\text{seq}\left(\frac{1}{n}, n, 1, 10, 2\right)$	$\left\{1, \frac{1}{3}, \frac{1}{5}, \frac{1}{7}, \frac{1}{9}\right\}$
$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)$	$\frac{1968329}{1270080}$

Remarque: Pour afficher un résultat approximatif,

Unité : Appuyez sur **ctrl** **enter**.

Windows® : Appuyez sur **Ctrl+Entrée**.

Macintosh® : Appuyez sur **⌘+Entrée**.

iPad® : Maintenez la touche **Entrée** enfoncée et sélectionnez **≈**.

$\text{sum}\left(\text{seq}\left(\frac{1}{n^2}, n, 1, 10, 1\right)\right)$	1.54977
--	---------

seqGen()

Catalogue >

seqGen(Expr, Var, VarDép, {Var0, MaxVar}[, ListeValeursInit [, IncVar [, ValeurMax]]])⇒liste

Génère les cinq premières valeurs de la suite $u(n) = u(n-1)^2/2$, avec $u(1)=2$ et $IncVar=1$.

$\text{seqGen}\left(\frac{(u(n-1))^2}{n}, u, u, \{1, 5\}, \{2\}\right)$	$\left\{2, 2, \frac{4}{3}, \frac{4}{9}, \frac{16}{405}\right\}$
---	---

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.

Exemple avec $Var0=2$:

$$\text{seqGen}\left(\frac{u(n-1)+1}{n}, n, u, \{2, 5\}, \{3\}\right)$$

$$\left\{3, \frac{4}{3}, \frac{7}{12}, \frac{19}{60}\right\}$$

Système de deux suites :

$$\text{seqGen}\left(\left\{\frac{1}{n}, \frac{u_2(n-1)}{2} + u_1(n-1)\right\}, n, \{u_1, u_2\}, \{1, 5\}, \begin{bmatrix} - \\ 2 \end{bmatrix}\right)$$

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ 2 & 2 & \frac{3}{2} & \frac{13}{12} & \frac{19}{24} \end{bmatrix}$$

Remarque : L'élément vide (_) dans la matrice de valeurs initiales ci-dessus est utilisé pour indiquer que la valeur initiale de $u_1(n)$ est calculée en utilisant la suite explicite $u_1(n)=1/n$.

seqn()

```
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 les cinq premières valeurs de la suite $u(n)=u(n-1)/2$, avec $u(1)=2$.

$$\text{seqn}\left(\frac{u(n-1)}{n}, \{2\}, 6\right)$$

$$\left\{2, 1, \frac{1}{3}, \frac{1}{12}, \frac{1}{60}, \frac{1}{360}\right\}$$

$$\text{seqn}\left(\frac{1}{n^2}, 6\right)$$

$$\left\{1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16}, \frac{1}{25}, \frac{1}{36}\right\}$$

Génère une liste de valeurs pour la suite $u(n)=\text{Expr}(n)$ comme suit : Incrémente n de 1 à $nMax$ par incrément de 1, calcule $u(n)$ pour les valeurs correspondantes de n en utilisant $\text{Expr}(n)$, 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() appelle seqGen() avec $n0=1$ et $Incn =1$

setMode()

Catalogue >

**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.

Affiche la valeur approchée de π à l'aide du réglage par défaut de Afficher chiffres, puis affiche π avec le réglage Fixe 2. Vérifiez que la valeur par défaut est bien restaurée après l'exécution du programme.

Define <i>prog1()</i> =Prgm	<i>Done</i>
Disp π	
setMode(1,16)	
Disp π	
EndPrgm	
<i>prog1()</i>	
	3.14159
	3.14
	<i>Done</i>

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 71.

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

Nom du mode	Entier du mode	Entiers de réglage
Afficher chiffres	1	1=Flottant, 2=Flottant 1, 3=Flottant 2, 4=Flottant 3, 5=Flottant 4, 6=Flottant 5, 7=Flottant 6, 8=Flottant 7, 9=Flottant 8, 10=Flottant 9, 11=Flottant 10, 12=Flottant 11, 13=Flottant 12, 14=Fixe 0, 15=Fixe 1, 16=Fixe 2, 17=Fixe 3, 18=Fixe 4, 19=Fixe 5, 20=Fixe 6, 21=Fixe 7, 22=Fixe 8, 23=Fixe 9, 24=Fixe 10, 25=Fixe 11, 26=Fixe 12
Angle	2	1=Radian, 2=Degré, 3=Grade
Format Exponentiel	3	1=Normal, 2=Scientifique, 3=Ingénieur
Réel ou Complex	4	1=Réel, 2=Rectangulaire, 3=Polaire
Auto ou Approché	5	1=Auto, 2=Approché

Nom du mode	Entier du mode	Entiers de réglage
Format Vecteur	6	1 =Rectangulaire, 2 =Cylindrique, 3 =Sphérique
Base	7	1 =Décimale, 2 =Hexadécimale, 3 =Binaire

shift()

Catalogue > 

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 17.

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.

0b000000000000000111101011000011010

Insère 0 si le premier bit est un 0

ou 1 si ce bit est un 1.

donne :

0b000000000000000111101011000011010

Le résultat est affiché selon le mode Base utilisé. Les zéros de tête ne sont pas affichés.

shift(Liste1 [,nbreDécal])⇒liste

En mode base Bin :

shift(0b1111010110000110101)	0b111101011000011010
shift(256,1)	0b10000000000

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).

En mode base Dec :

shift()

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).

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.

sign()

sign(*Valeur1*)⇒*valeur*

sign(*Liste1*)⇒*liste*

sign(*Matrice1*)⇒*matrice*

Pour un *Valeur1* réel ou complexe, donne *Valeur1*/abs(*Valeur1*) si *Valeur1* ≠ 0.

Donne 1 si *Valeur1* est positif.

Donne -1 si *Valeur1* est négatif.

sign(0) donne -1 en mode Format complexe Réel ; sinon, donne lui-même.

Catalogue >

shift({1,2,3,4})	{ undef,1,2,3 }
shift({1,2,3,4}, -2)	{ undef,undef,1,2 }
shift({1,2,3,4}, 2)	{ 3,4,undef,undef }

shift("abcd")	" abc "
shift("abcd", -2)	" ab "
shift("abcd", 1)	"bcd "

Catalogue >

sign(-3.2)	-1
sign({2,3,4,-5})	{1,1,1,-1}

En mode Format complexe Réel :

sign([-3 0 3])	[-1 undef 1]
----------------	--------------

sign(0) représente le cercle d'unité dans le domaine complexe.

Dans le cas d'une liste ou d'une matrice, donne les signes de tous les éléments.

simult()

simult(*matriceCoeff*, *vecteurConst*[, *Tol*]) \Rightarrow *matrice*

Donne un vecteur colonne contenant les solutions d'un système d'équations.

Remarque : voir aussi **linSolve()**, page 91.

matriceCoeff doit être une matrice carrée qui contient les coefficients des équations.

vecteurConst doit avoir le même nombre de lignes (même dimension) que *matriceCoeff* et contenir le second membre.

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

- Si vous réglez le mode **Auto ou Approché (Approximate)** sur **Approché (Approximate)**, les calculs sont exécutés en virgule flottante.
- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit :
 $5E-14 \cdot \max(\dim(\text{matriceCoeff}))$
 $\cdot \text{rowNorm}(\text{matriceCoeff})$

simult(*matriceCoeff*, *matriceConst*[, *Tol*]) \Rightarrow *matrice*

Résolution de x et y :

$$x + 2y = 1$$

$$3x + 4y = -1$$

$$\begin{array}{|c|c|} \hline \text{simult}\left(\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}\right) & \begin{bmatrix} -3 \\ 2 \end{bmatrix} \\ \hline \end{array}$$

La solution est x=-3 et y=2.

Résolution :

$$ax + by = 1$$

$$cx + dy = 2$$

$$\begin{array}{|c|c|} \hline \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow \text{matrx1} & \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \\ \hline \text{simult}\left(\text{matrx1}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) & \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \\ \hline \end{array}$$

Résolution :

$$x + 2y = 1$$

$$3x + 4y = -1$$

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.

$$x + 2y = 2$$

$$3x + 4y = -3$$

<code>simult([1 2; 3 4], [1 2; -1 -3])</code>	$\begin{bmatrix} -3 & -7 \\ 2 & 9 \\ \hline 2 & 2 \end{bmatrix}$
---	--

Pour le premier système, $x=-3$ et $y=2$. Pour le deuxième système, $x=-7$ et $y=9/2$.

sin()

Touche 

`sin(Valeur1)⇒valeur`

`sin(Liste1)⇒liste`

`sin(Valeur1)` donne le sinus de l'argument.

`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 $^{\circ}$, G ou r pour ignorer temporairement le mode angulaire sélectionné.

En mode Angle en degrés :

<code>sin((π/4)r)</code>	0.707107
--------------------------	----------

<code>sin(45)</code>	0.707107
----------------------	----------

<code>sin({0,60,90})</code>	{0.,0.866025,1.}
-----------------------------	------------------

En mode Angle en grades :

<code>sin(50)</code>	0.707107
----------------------	----------

En mode Angle en radians :

<code>sin((π/4))</code>	0.707107
-------------------------	----------

<code>sin(45°)</code>	0.707107
-----------------------	----------

`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 :

<code>sin([1 5 3; 4 2 1; 6 -2 1])</code>	$\begin{bmatrix} 0.9424 & -0.04542 & -0.031999 \\ -0.045492 & 0.949254 & -0.020274 \\ -0.048739 & -0.00523 & 0.961051 \end{bmatrix}$
--	--

sin⁻¹()

Touche

sin⁻¹(Valeur1)⇒valeur

En mode Angle en degrés :

sin⁻¹(Liste1)⇒listesin⁻¹(1)

90.

sin⁻¹(Valeur1) donne l'arc sinus de Valeur1.**sin⁻¹(Liste1) donne la liste des arcs sinus des éléments de Liste1.****Remarque :** donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.**Remarque :** vous pouvez insérer cette fonction à partir du clavier en entrant **arcsin(...)**.**sin⁻¹(matriceCarrée1)⇒matriceCarrée**Donne l'argument arc sinus de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de l'argument arc sinus de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.*matriceCarrée1* doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.**sinh()**

Catalogue >

sinh(Valeur1)⇒valeur

sinh(1.2)

1.50946

sinh(Liste1)⇒liste

sinh({0,1,2,3.})

{0,1.50946,10.0179}

sinh (Valeur1) donne le sinus hyperbolique de l'argument.**sinh (Liste1) donne la liste des sinus hyperboliques des éléments de Liste1.****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()**.

En mode Angle en radians :

sinh $\begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix}$

360.954	305.708	239.604
352.912	233.495	193.564
298.632	154.599	140.251

sinh()**Catalogue > **

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

sinh⁻¹()**Catalogue > ****sinh⁻¹(Valeur1)⇒valeur**sinh⁻¹{0} 0**sinh⁻¹(Liste1)⇒liste**sinh⁻¹{ { 0,2,1,3 } } { 0,1.48748,1.81845 }

sinh⁻¹(Valeur1) donne l'argument sinus hyperbolique de l'argument.

sinh⁻¹(Liste1) donne la liste des arguments sinus hyperboliques des éléments de *Liste1*.

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

sinh⁻¹(matriceCarrée1)⇒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 164.)

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 numériques ou alphanumériques 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 236.

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

SortA

SortA *Liste1[, Liste2] [, Liste3] ...*

SortA *Vecteur1[, Vecteur2] [, Vecteur3]*

...

Trie les éléments du premier argument en ordre croissant.

Si d'autres arguments sont présents, trie les éléments de chacun d'entre eux de sorte que leur nouvelle position corresponde aux nouvelles positions des éléments dans le premier argument.

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 236.

SortD

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 236.

►Sphere

Vecteur ►Sphere

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

Affiche le vecteur ligne ou colonne en coordonnées sphériques [$\rho \angle\theta \angle\phi$].

Catalogue >

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
SortA <i>list1</i>	<i>Done</i>
<i>list1</i>	$\{1,2,3,4\}$
$\{4,3,2,1\} \rightarrow list2$	$\{4,3,2,1\}$
SortA <i>list2,list1</i>	<i>Done</i>
<i>list2</i>	$\{1,2,3,4\}$
<i>list1</i>	$\{4,3,2,1\}$

Catalogue >

$\{2,1,4,3\} \rightarrow list1$	$\{2,1,4,3\}$
$\{1,2,3,4\} \rightarrow list2$	$\{1,2,3,4\}$
SortD <i>list1,list2</i>	<i>Done</i>
<i>list1</i>	$\{4,3,2,1\}$
<i>list2</i>	$\{3,4,1,2\}$

Catalogue >

[1 2 3]►Sphere	
[3.74166 $\angle 1.10715$ $\angle 0.640522]$	

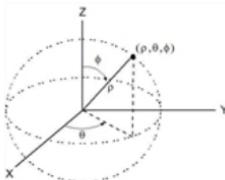
►Sphere

Catalogue >

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.

$\begin{pmatrix} 2 & \angle \frac{\pi}{4} & 3 \\ 3.60555 & \angle 0.785398 & \angle 0.588003 \end{pmatrix}$ ►Sphere



sqrt()

Catalogue >

sqrt(Valeur1)⇒valeur

$\sqrt{4}$ 2

sqrt(Liste1)⇒liste

$\sqrt{\{9,2,4\}}$ {3,1.41421,2}

Donne la racine carrée de l'argument.

Dans le cas d'une liste, donne la liste des racines carrées des éléments de Liste1.

Remarque : voir aussi **Modèle Racine carrée**, page 1.

stat.results

Affiche le résultat d'un calcul statistique.

Les résultats sont affichés sous forme d'ensemble de paires nom-valeur. Les noms spécifiques affichés varient suivant la fonction ou commande statistique la plus récemment calculée ou exécutée.

Vous pouvez copier un nom ou une valeur et la coller à d'autres emplacements.

Remarque : ne définissez pas de variables dont le nom est identique à celles utilisées dans le cadre de l'analyse statistique. Dans certains cas, cela peut générer une erreur. Les noms de variables utilisés pour l'analyse statistique sont répertoriés dans le tableau ci-dessous.

<i>xlist:=</i> {1,2,3,4,5}	{1,2,3,4,5}
----------------------------	-------------

<i>ylist:=</i> {4,8,11,14,17}	{4,8,11,14,17}
-------------------------------	----------------

LinRegMx *xlist,ylist,1:* stat.results

"Title"	"Linear Regression (mx+b)"
"RegEqn"	"m*x+b"
"m"	3.2
"b"	1.2
"r ² "	0.996109
"r"	0.998053
"Resid"	"{...}"

<i>stat.values</i>	"Linear Regression (mx+b)"
	"m*x+b"
	3.2
	1.2
	0.996109
	0.998053
	"{-0.4,0.4,0.2,0.,-0.2}"

stat.a	stat.dfDenom	stat.MedianY	stat.Q3X	stat.SSBlock
stat.AdjR ²	stat.dfBlock	stat.MEPred	stat.Q3Y	stat.SSCol
stat.b	stat.dfCol	stat.MinX	stat.r	stat.SSX
stat.b0	stat.dfError	stat.MinY	stat.r ²	stat.SSY
stat.b1	stat.dflnteract	stat.MS	stat.RegEqn	stat.SSError
stat.b2	stat.dfReg	stat.MSBlock	stat.Resid	stat.SSIInteract
stat.b3	stat.dfNumer	stat.MSCol	stat.ResidTrans	stat.SSReg
stat.b4	stat.dfRow	stat.MSError	stat.ox	stat.SSRow
stat.b5	stat.DW	stat.MSIInteract	stat.oy	stat.tList
stat.b6	stat.e	stat.MSReg	stat.ox1	stat.UpperPred
stat.b7	stat.ExpMatrix	stat.MSRow	stat.ox2	stat.UpperVal
stat.b8	stat.F	stat.n	stat.Sx	stat.X̄
stat.b9	stat.FBlock	stat.ŷ	stat.Sx ²	stat.X̄1
stat.b10	stat.Fcol	stat.ŷ1	stat.Sxy	stat.X̄2
stat.bList	stat.Flnteract	stat.ŷ2	stat.Sy	stat.X̄Diff
stat.χ ²	stat.FreqReg	stat.ŷDiff	stat.Sy ²	stat.X̄List

stat.c	stat.Frow	stat.PList	stat.s	stat.XReg
stat.CLower	stat.Leverage	stat.PVal	stat.SE	stat.XVal
stat.CLowerList	stat.LowerPred	stat.PValBlock	stat.SEList	stat.XValList
stat.ComplList	stat.LowerVal	stat.PValCol	stat.SEPred	stat. \bar{y}
stat.CompMatrix	stat.m	stat.PValInteract	stat.sResid	stat. \hat{y}
stat.CookDist	stat.MaxX	stat.PValRow	stat.SEslope	stat. \hat{y} List
stat.CUpper	stat.MaxY	stat.Q1X	stat.sp	stat.YReg
stat.CUpperList	stat.ME	stat.Q1Y	stat.SS	
stat.d	stat.MedianX			

Remarque : Chaque fois que l'application Tableur & listes calcule des résultats statistiques, les variables du groupe « stat. » sont copiées dans un groupe « stat#. », où # est un nombre qui est incrémenté automatiquement. Cela vous permet de conserver les résultats précédents tout en effectuant plusieurs calculs.

stat.values

Catalogue >

stat.values

Voir l'exemple donné pour
stat.results.

Affiche une matrice des valeurs calculées pour la fonction ou commande statistique la plus récemment calculée ou exécutée.

Contrairement à **stat.results**, **stat.values** omet les noms associés aux valeurs.

Vous pouvez copier une valeur et la coller à d'autres emplacements.

stDevPop()

Catalogue >

stDevPop(Liste[,
listeFréq]) \Rightarrow expression

En mode Angle en radians et en modes Auto :

stDevPop({1,2,5,-6,3,-2})	3.59398
stDevPop({1.3,2.5,-6.4},{3,2,5})	4.11107

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 236.

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 236.

stDevPop	$\begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{pmatrix}$	$[3.26599 \quad 2.94392 \quad 1.63299]$
stDevPop	$\begin{pmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{pmatrix}$	$\begin{pmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{pmatrix}$ $[2.52608 \quad 5.21506]$

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 236.

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*.

stDevSamp({1,2,5,-6,3,-2})	3.937
stDevSamp({1.3,2.5,-6.4},{3,2,5})	4.33345

stDevSamp	$\begin{pmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ 5 & 7 & 3 \end{pmatrix}$	$[4. \quad 3.60555 \quad 2.]$
stDevSamp	$\begin{pmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{pmatrix}$	$\begin{pmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{pmatrix}$ $[2.7005 \quad 5.44695]$

stDevSamp()

Catalogue >

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 236.

Stop

Stop

Commande de programmation : Ferme le programme.

Stop n'est pas autorisé dans les fonctions.

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

Catalogue >

<i>i:=0</i>	0
Define <i>prog1()</i> =Prgm For <i>i</i> ,1,10,1 If <i>i</i> =5 Stop EndFor EndPrgm	Done
<i>prog1()</i>	Done
<i>i</i>	5

Store

Voir → (store), page 217.

string()

string(*Expr*)⇒chaîne

Simplifie *Expr* et donne le résultat sous forme de chaîne de caractères.

Catalogue >

string(1.2345)	"1.2345"
string(1+2)	"3"

subMat()

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.

Catalogue >

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow m1$	$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$
subMat(<i>m1</i> ,2,1,3,2)	$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$
subMat(<i>m1</i> ,2,2)	$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

sum()

sum(*Liste*[, *Début*[, *Fin*]]) \Rightarrow 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 236.

sum(*Matrice1*[, *Début*[, *Fin*]]) \Rightarrow 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 236.

Catalogue >

sum({1,2,3,4,5})	15
sum({a,2·a,3·a})	"Error: Variable is not defined"
sum(seq(n,n,1,10))	55
sum({1,3,5,7,9},3)	21

sum($\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$)	[5 7 9]
sum($\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$)	[12 15 18]
sum($\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$,2,3)	[11 13 15]

sumIf()**Catalogue >**

sumIf(*Liste*,*Critère*[, *ListeSommes*]) \Rightarrow 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*.

sumIf({1,2,e,3,π,4,5,6},2.5<?<4.5)	12.859874482
sumIf({1,2,3,4},2<?<5,{10,20,30,40})	70

Le critère peut être :

- Une valeur, une expression ou une chaîne. Par exemple, **34** cumule uniquement les éléments dans *Liste* qui donnent la valeur 34.
- Une expression booléenne contenant le symbole **?** comme paramètre substituable à tout élément. Par exemple, **?<10** cumule uniquement les éléments de *Liste* qui sont inférieurs à 10.

Lorsqu'un élément de *Liste* répond au critère, il est ajouté à la somme cumulée. Si vous incluez *ListeSommes*, c'est l'élément correspondant dans *ListeSommes* qui est ajouté à la somme.

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

Les éléments vides sont ignorés. Pour plus d'informations concernant les éléments vides, reportez-vous à la page 236.

Remarque : voir également **countIf()**, page 33.

system(*Valeur1 [, Valeur2 [, Valeur3 [, ...]]])*

Donne un système d'équations, présenté sous forme de liste. Vous pouvez également créer un système d'équation en utilisant un modèle.

T (transposée)**Catalogue >** *Matrix1T⇒matrice*Donne la transposée de la conjuguée de *Matrice1*.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^{\text{T}} \quad \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant @t.

tan()**Touche** *tan(Valeur1)⇒valeur**tan(Liste1)⇒liste**tan(Valeur1)* 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 ' pour ignorer temporairement le mode Angle sélectionné.

En mode Angle en degrés :

$\tan\left(\left(\frac{\pi}{4}\right)_r\right)$	1.
$\tan(45)$	1.
$\tan(\{0,60,90\})$	{0.,1.73205,undef}

En mode Angle en grades :

$\tan\left(\left(\frac{\pi}{4}\right)_g\right)$	1.
$\tan(50)$	1.
$\tan(\{0,50,100\})$	{0.,1.,undef}

En mode Angle en radians :

$\tan\left(\frac{\pi}{4}\right)$	1.
$\tan(45^\circ)$	1.
$\tan\left(\left\{\pi, \frac{\pi}{3}, -\pi, \frac{\pi}{4}\right\}\right)$	{0.,1.73205,0.,1.}

En mode Angle en radians :

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()**.

tan()

Touche

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

$$\tan \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} -28.2912 & 26.0887 & 11.1142 \\ 12.1171 & -7.83536 & -5.48138 \\ 36.8181 & -32.8063 & -10.4594 \end{bmatrix}$$

tan⁻¹()

Touche

tan⁻¹(Valeur1)⇒valeur

tan⁻¹(Liste1)⇒liste

tan⁻¹(Valeur1) donne l'arc tangente de *Valeur1*.

tan⁻¹(Liste1) 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(...)**.

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 degrés :

$$\tan^{-1}(1) = 45$$

En mode Angle en grades :

$$\tan^{-1}(1) = 50$$

En mode Angle en radians :

$$\tan^{-1}(\{0,0.2,0.5\}) = \{0,0.197396,0.463648\}$$

En mode Angle en radians :

$$\tan^{-1} \begin{pmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{pmatrix} = \begin{bmatrix} -0.083658 & 1.26629 & 0.62263 \\ 0.748539 & 0.630015 & -0.070012 \\ 1.68608 & -1.18244 & 0.455126 \end{bmatrix}$$

tanh()

Catalogue >

tanh(Valeur1)⇒valeur

tanh(Liste1)⇒liste

tanh(Valeur1) donne la tangente hyperbolique de l'argument.

$$\tanh(1.2) = 0.833655$$

$$\tanh(\{0,1\}) = \{0,0.761594\}$$

tanh(Liste1) donne la liste des tangentes hyperboliques des éléments de *Liste1*.

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 :

$$\tanh \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} \begin{bmatrix} -0.097966 & 0.933436 & 0.425972 \\ 0.488147 & 0.538881 & -0.129382 \\ 1.28295 & -1.03425 & 0.428817 \end{bmatrix}$$

tanh⁻¹()

tanh⁻¹(Valeur1)⇒valeur

tanh⁻¹(Liste1)⇒liste

tanh⁻¹(Valeur1) donne l'argument tangente hyperbolique de l'argument.

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 :

$$\begin{aligned} \tanh^{-1}(0) &= 0. \\ \tanh^{-1}\{1, 2, 1, 3\} &= \{\text{undef}, 0.518046 - 1.5708 \cdot i, 0.346574 - 1.570 \\ &\quad \cdot i, 1.28295 - 1.03425 \cdot i\} \end{aligned}$$

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

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

$$\begin{aligned} \tanh^{-1} \begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix} &= \begin{bmatrix} -0.099353 + 0.164058 \cdot i & 0.267834 - 1.4908 \\ -0.087596 - 0.725533 \cdot i & 0.479679 - 0.9473 \cdot i \\ 0.511463 - 2.08316 \cdot i & -0.878563 + 1.7901 \end{bmatrix} \end{aligned}$$

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

tCdf(*LimitInf*,*LimitSup*,*df*) \Rightarrow nombre si
LimitInf et *LimitSup* sont des nombres, liste
si *LimitInf* et *LimitSup* sont des listes

Calcule la fonction de répartition de la loi de Student-*t* à *df* degrés de liberté entre *LimitInf* et *LimitSup*.

Pour $P(X \leq upBound)$, définissez *lowBound* = -9E999.

Text

Text*chaîneinvite[, IndicAff]*

Commande de programmation : Marque une pause dans l'exécution du programme et affiche la chaîne de caractères *chaîneinvite* dans une boîte de dialogue.

Lorsque l'utilisation sélectionne **OK**, l'exécution du programme se poursuit.

L'argument optionnel *IndicAff* peut correspondre à n'importe quelle expression.

- Si *IndicAff* est omis ou a pour valeur **1**, le message est ajouté à l'historique de l'application Calculs.
- Si *IndicAff* a pour valeur **0**, le message n'est pas ajouté à l'historique.

Si le programme nécessite une réponse saisie par l'utilisateur, voir **Request**, page 140 ou **RequestStr**, page 141.

Remarque : vous pouvez utiliser cette commande dans un programme créé par l'utilisateur, mais pas dans une fonction.

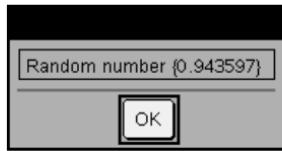
Définissez un programme qui marque une pause afin d'afficher cinq nombres aléatoires dans une boîte de dialogue.

Dans le modèle Prgm...EndPrgm, validez chaque ligne en appuyant sur à la place de . 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 :



tIntervalCatalogue > **tInterval** *Liste[,Fréq[,CLevel]]*

(Entrée de liste de données)

tInterval $\bar{x}, s_x, n[, CLevel]$

(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 164.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

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. s_x	Écart-type d'échantillon
stat.n	Taille de la série de données avec la moyenne d'échantillon

tInterval_2SampCatalogue > **tInterval_2Samp** *Liste1,Liste2[,Fréq1
[,Freq2[,CLevel[,Group]]]]*

(Entrée de liste de données)

tInterval_2Samp $\bar{x}_1, s_{x1}, n_1, \bar{x}_2, s_{x2}, n_2
[, CLevel[, Group]]$

(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 164.)

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 236.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance contenant la probabilité du niveau de confiance de la loi
stat.Ȑx1-Ȑx2	Moyennes d'échantillon des séries de données suivant la loi normale aléatoire
stat.ME	Marge d'erreur
stat.df	Degrés de liberté
stat.Ȑx1, stat.Ȑx2	Moyennes d'échantillon des séries de données suivant la loi normale aléatoire
stat.Ȑx1, stat.Ȑx2	Écarts-types d'échantillon pour <i>Liste 1</i> et <i>Liste 2</i>
stat.n1, stat.n2	Nombre d'échantillons dans les séries de données
stat.sp	Écart-type du groupe. Calculé lorsque Group = YES.

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*)⇒*valeur***

Donne la trace (somme de tous les éléments de la diagonale principale) de *matriceCarrée*.

trace $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$	15
a:=12	12
trace $\begin{bmatrix} a & 0 \\ 1 & a \end{bmatrix}$	24

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 246.

bloc1 et *bloc2* peuvent correspondre à une instruction unique ou à une série d'instructions séparées par le caractère ":".

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

Pour voir fonctionner les commandes **Try**, **ClrErr** et **PassErr**, saisissez le programme *eigenvals()* décrit à droite. Exécutez le programme en exécutant chacune des expressions suivantes.

eigenvals $\begin{bmatrix} -3 \\ -41 \\ 5 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -3.1 \end{bmatrix}$

Define <i>prog1()</i> =Prgm	
Try	
<i>z</i> := <i>z</i> +1	
Disp "z incremented."	
Else	
Disp "Sorry, z undefined."	
EndTry	
EndPrgm	
	<i>Done</i>
<i>z</i> :=1; <i>prog1()</i>	
	z incremented.
	<i>Done</i>
DelVar <i>z</i> : <i>prog1()</i>	
	Sorry, z undefined.
	<i>Done</i>

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 " "
```

Remarque : voir aussi **ClrErr**, page 24 et **PassErr**, page 123.

```

Disp "Eigenvalues of A·B are:",eigVl(a*b)
Else
    If errCode=230 Then
        Disp "Error: Product of A·B must be a
square matrix"
    ClrErr
    Else
        PassErr
    EndIf
EndTry
EndPrgm

```

tTest

tTest μ_0 ,*Liste*[,*Fréq*[,*Hypoth*]]

(Entrée de liste de données)

tTest μ_0 , \bar{x} ,*sx*,*n*,[*Hypoth*]

(Récapitulatif des statistiques fournies en entrée)

Teste une hypothèse pour une moyenne inconnue de population μ quand l'écart-type de population σ est inconnu. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 236.

Variable de sortie	Description
stat.t	$(\bar{x} - \mu_0) / (\text{stdev} / \sqrt{n})$
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degrés de liberté
stat. \bar{x}	Moyenne d'échantillon de la série de données dans <i>Liste</i>
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 $\bar{x}_1, sx_1, n_1, \bar{x}_2, sx_2, n_2$ [*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 164.)

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 236.

Variable de sortie	Description
stat.t	Valeur normale type calculée pour la différence des moyennes
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat.df	Degrés de liberté des statistiques t
stat. \bar{x} 1, stat. \bar{x} 2	Moyennes d'échantillon des séquences de données dans <i>Liste 1 et Liste 2</i>
stat.sx1, stat.sx2	Écarts-types d'échantillon des séries de données dans <i>Liste 1 et Liste 2</i>
stat.n1, stat.n2	Taille des échantillons
stat.sp	Écart-type du groupe. Calculé lorsque <i>Group=1</i> .

tvmFV()

Catalogue >

tvmFV($N, I, PV, Pmt, [PpY], [CpY], [PmtAt]$)⇒valeur

tvmFV(120,5,0,-500,12,12)

77641.1

Fonction financière permettant de calculer la valeur acquise de l'argent.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 180. Voir également **amortTbl()**, page 7.

tvmI()

Catalogue >

tvmI($N, PV, Pmt, FV, [PpY], [CpY], [PmtAt]$)⇒valeur

tvmI(240,100000,-1000,0,12,12)

10.5241

Fonction financière permettant de calculer le taux d'intérêt annuel.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 180. Voir également **amortTbl()**, page 7.

tvmN()

Catalogue >

tvmN($I, PV, Pmt, FV, [PpY], [CpY], [PmtAt]$)⇒valeur

tvmN(5,0,-500,77641,12,12)

120.

Fonction financière permettant de calculer le nombre de périodes de versement.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 180. Voir également **amortTbl()**, page 7.

tvmPmt()

tvmPmt(*N,I,PV,FV,[PpY],[CpY], [PmtAt]*) \Rightarrow valeur

tvmPmt(60,4,30000,0,12,12)	-552.496
----------------------------	----------

Fonction financière permettant de calculer le montant de chaque versement.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 180. Voir également **amortTbl()**, page 7.

tvmPV()

tvmPV(*N,I,Pmt,FV,[PpY],[CpY], [PmtAt]*) \Rightarrow valeur

tvmPV(48,4,-500,30000,12,12)	-3426.7
------------------------------	---------

Fonction financière permettant de calculer la valeur actuelle.

Remarque : Les arguments utilisés dans les fonctions TVM sont décrits dans le tableau des arguments TVM, page 180. Voir également **amortTbl()**, page 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 $X, Y[, \text{Fréq}][, \text{Catégorie}, \text{Inclure}]$

Calcule des statistiques pour deux variables.
Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 numériques ou alphanumériques 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 236.

Variable de sortie	Description
stat. \bar{x}	Moyenne des valeurs x
stat. x	Somme des valeurs x
stat. x2	Somme des valeurs x2

Variable de sortie	Description
stat.sx	Écart-type de l'échantillon de x
stat.x	Écart-type de la population de x
stat.n	Nombre de points de données
stat. \bar{y}	Moyenne des valeurs y
stat.y	Somme des valeurs y
stat.y ²	Somme des valeurs y ²
stat.sy	Écart-type de y dans l'échantillon
stat.y	Écart-type de population des valeurs de y
stat.xy	Somme des valeurs x · y
stat.r	Coefficient de corrélation
stat.MinX	Minimum des valeurs de x
stat.Q ₁ X	1er quartile de x
stat.MedianX	Médiane de x
stat.Q ₃ X	3ème quartile de x
stat.MaxX	Maximum des valeurs de x
stat.MinY	Minimum des valeurs de y
stat.Q ₁ Y	1er quartile de y
stat.MedY	Médiane de y
stat.Q ₃ Y	3ème quartile de y
stat.MaxY	Maximum des valeurs y
stat.(x-) ²	Somme des carrés des écarts par rapport à la moyenne de x
stat.(y-) ²	Somme des carrés des écarts par rapport à la moyenne de y

U**unitV()****unitV(*Vecteur1*)** \Rightarrow vector

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.

Catalogue >

$\text{unitV}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.408248 & 0.816497 & 0.408248 \end{bmatrix}$
$\text{unitV}\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$	$\begin{bmatrix} 0.267261 \\ 0.534522 \\ 0.801784 \end{bmatrix}$

unLock**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 95 et **getLockInfo()**, page 71.

Catalogue >

<i>a:=65</i>	65
Lock <i>a</i>	Done
getLockInfo(<i>a</i>)	1
<i>a:=75</i>	"Error: Variable is locked."
DelVar <i>a</i>	"Error: Variable is locked."
Unlock <i>a</i>	Done
<i>a:=75</i>	75
DelVar <i>a</i>	Done

V**varPop()****Catalogue >** **varPop(*Liste*[, *listeFréq*])** \Rightarrow expression

$\text{varPop}(\{5,10,15,20,25,30\})$	72.9167
---------------------------------------	---------

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 236.

varSamp()**Catalogue >** **varSamp(Liste[, listeFréq])** \Rightarrow expressionDonne 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 236.

varSamp(Matrice1[, matriceFréq]) \Rightarrow matriceDonne 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 236.

W**Wait****Catalogue >** **Wait tempsEnSecondes**Suspend l'exécution pendant une durée de *tempsEnSecondes* secondes.La commande **Wait** est particulièrement utile dans un programme qui a besoin de quelques secondes pour permettre aux données demandées d'être accessibles.

varSamp({1,2,5,-6,3,-2})	31
	2
varSamp({1,3,5},{4,6,2})	68
	33

varSamp($\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}$)	[4.75 1.03 4]
varSamp($\begin{bmatrix} -1.1 & 2.2 \\ 3.4 & 5.1 \\ -2.3 & 4.3 \end{bmatrix}, \begin{bmatrix} 6 & 3 \\ 2 & 4 \\ 5 & 1 \end{bmatrix}$)	[3.91731 2.08411]

L'argument *tempsEnSecondes* doit être une expression qui s'évalue en une valeur décimale comprise entre 0 et 100. La commande arrondit cette valeur à 0,1 seconde près.

Pour annuler un **Wait** qui est en cours,

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

Remarque : Vous pouvez utiliser la commande **Wait** dans un programme créé par l'utilisateur, mais pas dans une fonction.

warnCodes ()

**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 255.

Pour définir un délai d'attente de 1,3 seconde à l'aide de la variable *seccompt* :
seccompt:=1.3
Wait seccompt

Cet exemple allume une DEL verte pendant 0,5 seconde puis l'éteint.

Send "SET GREEN 1 ON"
Wait 0.5
Send "SET GREEN 1 OFF"

warnCodes(det([1.23456e-999]),warn)	1.23456e-999
warn	{10029}

when(*Condition*, *résultSiOui* [,
résultSiNon][,
résultSiInconnu]) \Rightarrow *expression*

Donne *résultSiOui*, *résultSiNon* ou *résultSiInconnu*, suivant que la *Condition* est vraie, fausse ou indéterminée. Donne l'entrée si le nombre d'argument est insuffisant pour spécifier le résultat approprié.

Ne spécifiez pas *résultSiNon* ni *résultSiInconnu* pour obtenir une expression définie uniquement dans la région où *Condition* est vraie.

Utilisez **undef** *résultSiNon* pour définir une expression représentée graphiquement sur un seul intervalle.

when() est utile dans le cadre de la définition de fonctions récursives.

when($x < 0, x + 3$) $x = 5$	undef
----------------------------------	-------

when($n > 0, n \cdot \text{factoral}(n - 1), 1$) \rightarrow <i>factoral</i> (<i>n</i>)	Done
---	------

<i>factoral</i> (3)	6
---------------------	---

3!	6
----	---

While

While *Condition*

Bloc

EndWhile

Exécute les instructions contenues dans *Bloc* si *Condition* est vraie.

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

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

Define <i>sum_of_recip</i> (<i>n</i>)=Func Local <i>i,tempsum</i> $1 \rightarrow i$ $0 \rightarrow tempsum$ While $i \leq n$ $tempsum + \frac{1}{i} \rightarrow tempsum$ $i + 1 \rightarrow i$ EndWhile Return <i>tempsum</i> EndFunc	Done
--	------

<i>sum_of_recip</i> (3)	11 6
-------------------------	---------

xor

BooleanExpr1 xor BooleanExpr2
renvoie expression booléenne

true xor true	false
5>3 xor 3>5	true

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 120.

Entier1 xor Entier2 \Rightarrow 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).

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.

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 17.

Remarque : voir **or**, page 120.

Z

zInterval

zInterval $\sigma, Liste[, Fréq[, CLevel]]$

(Entrée de liste de données)

zInterval $\sigma, \bar{x}, n [, CLevel]$

(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 164.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance pour une moyenne inconnue de population
stat. \bar{x}	Moyenne d'échantillon de la série de données suivant la loi normale aléatoire
stat.ME	Marge d'erreur
stat.sx	Écart-type d'échantillon
stat.n	Taille de la série de données avec la moyenne d'échantillon
stat. σ	Écart-type connu de population pour la série de données <i>Liste</i>

zInterval_1Prop

zInterval_1Prop $x, n [, CLevel]$

Calcule un intervalle de confiance z pour une proportion. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

x est un entier non négatif.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance contenant la probabilité du niveau de confiance de la loi
stat. \hat{p}	Proportion calculée de réussite
stat.ME	Marge d'erreur
stat.n	Nombre d'échantillons dans la série de données

zInterval_2Prop $x1, n1, x2, n2[, CLevel]$

Calcule un intervalle de confiance z pour deux proportions. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

$x1$ et $x2$ sont des entiers non négatifs.

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

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	Déférence calculée entre les proportions
stat.ME	Marge d'erreur
stat. $\hat{p}1$	Proportion calculée sur le premier échantillon
stat. $\hat{p}2$	Proportion calculée sur le deuxième échantillon

Variable de sortie	Description
stat.n1	Taille de l'échantillon dans la première série de données
stat.n2	Taille de l'échantillon dans la deuxième série de données

zInterval_2Samp

Catalogue >

zInterval_2Samp $\sigma_1, \sigma_2, Liste1, Liste2$
[, Fréq1 [, Fréq2, [CLevel]]]

(Entrée de liste de données)

zInterval_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2$
[, 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 164.)

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

Variable de sortie	Description
stat.CLower, stat.CUpper	Intervalle de confiance contenant la probabilité du niveau de confiance de la loi
stat. $\bar{x}1-\bar{x}2$	Moyennes d'échantillon des séries de données suivant la loi normale aléatoire
stat.ME	Marge d'erreur
stat. $\bar{x}1$, stat. $\bar{x}2$	Moyennes d'échantillon des séries de données suivant la loi normale aléatoire
stat. $\sigma x1$, stat. $\sigma x2$	Écarts-types d'échantillon pour <i>Liste 1</i> et <i>Liste 2</i>
stat.n1, stat.n2	Nombre d'échantillons dans les séries de données
stat.r1, stat.r2	Écart-type connu de population pour la série de données <i>Liste 1</i> et <i>Liste 2</i>

zTest

Catalogue >

zTest $\mu 0, \sigma, Liste, [Fréq], [Hypoth]$

(Entrée de liste de données)

zTest $\mu 0, \sigma, \bar{x}, n, [Hypothesis]$

(Récapitulatif des statistiques fournies en entrée)

Effectue un test z en utilisant la fréquence *listeFreq*. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

Test de $H_0 : \mu = \mu_0$, en considérant que :

Pour $H_a : \mu < \mu_0$, définissez *Hypothesis*<0

Pour $H_a : \mu \neq \mu_0$ (par défaut), définissez *Hypothesis*=0

Pour $H_a : \mu > \mu_0$, définissez *Hypothesis*>0

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

Variable de sortie	Description
stat.z	$(\bar{x} - \mu_0) / (\sigma / \sqrt{n})$
stat.P Value	Plus petite probabilité permettant de rejeter l'hypothèse nulle
stat. \bar{x}	Moyenne d'échantillon de la série de données dans <i>Liste</i>
stat.sx	Écart-type d'échantillon de la série de données Uniquement donné pour l'entrée <i>Data</i> .
stat.n	Taille de l'échantillon

zTest_1Prop $p0, x, n, [Hypothesis]$

Effectue un test z pour une proportion unique. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

x est un entier non négatif.

Test de $H_0 : p = p_0$, en considérant que :

zTest_1Prop

Catalogue > 

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 236.

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 164.)

$x1$ et $x2$ sont des entiers non négatifs.

Test de $H_0 : p1 = p2$, en considérant que :

Pour $H_a : p1 > p2$, définissez *Hypoth>0*

Pour $H_a : p1 \neq p2$ (*par défaut*), définissez
Hypoth=0

Pour $H_a : p1 < p2$, définissez *Hypoth<0*

Pour plus d'informations concernant les éléments vides dans une liste, reportez-vous à "Éléments vides", page 236.

Variable de sortie	Description
stat.z	Valeur normale type calculée pour la différence des proportions

Variable de sortie	Description
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat. $\hat{p}1$	Proportion calculée sur le premier échantillon
stat. $\hat{p}2$	Proportion calculée sur le deuxième échantillon
stat. \hat{p}	Proportion calculée de l'échantillon mis en commun
stat.n1, stat.n2	Nombre d'échantillons pris lors des essais 1 et 2

zTest_2Samp

Catalogue > 

zTest_2Samp $\sigma_1, \sigma_2, Liste1, Liste2[, Fréq1 [, Fréq2[, Hypoth]]]$

(Entrée de liste de données)

zTest_2Samp $\sigma_1, \sigma_2, \bar{x}1, n1, \bar{x}2, n2[, Hypoth]$

(Récapitulatif des statistiques fournies en entrée)

Calcule un test z sur deux échantillons. Un récapitulatif du résultat est stocké dans la variable *stat.results*. (Voir page 164.)

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 236.

Variable de sortie	Description
stat.z	Valeur normale type calculée pour la différence des moyennes
stat.PVal	Plus petit seuil de signification permettant de rejeter l'hypothèse nulle
stat. $\bar{x}1$, stat. $\bar{x}2$	Moyennes d'échantillon des séquences de données dans <i>Liste 1</i> et <i>Liste 2</i>
stat.sx1, stat.sx2	Écart-types d'échantillon des séries de données dans <i>Liste 1</i> et <i>Liste 2</i>
stat.n1, stat.n2	Taille des échantillons

Symboles

+ (somme)

Touche

Valeur1 + Valeur2⇒*valeur*

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.

Valeur + Liste1⇒*liste*

Liste1 + Valeur⇒*liste*

Donne la liste contenant les sommes de *Valeur* et de chaque élément de *Liste1*.

Valeur + Matrice1⇒*matrice*

Matrice1 + Valeur⇒*matrice*

Donne la matrice obtenue en ajoutant *Valeur* à chaque élément de la diagonale de *Matrice1*. *Matrice1* doit être carrée.

Remarque : utilisez *.+* pour ajouter une expression à chaque élément de la matrice.

$\left\{ 22, \pi, \frac{\pi}{2} \right\} \rightarrow l1$	$\{ 22, 3.14159, 1.5708 \}$
$\left\{ 10.5, \frac{\pi}{2} \right\} \rightarrow l2$	$\{ 10.5, 1.5708 \}$
$l1 + l2$	$\{ 32.8, 14159, 3.14159 \}$

$20 + \{ 10, 15, 20 \}$	$\{ 25, 30, 35 \}$
$\{ 10, 15, 20 \} + 15$	$\{ 25, 30, 35 \}$

-(soustraction)

Touche

Valeur1 - Valeur2⇒*valeur*

Donne la différence de *Valeur1* et de *Valeur2*.

$6 - 2$	4
$\pi - \frac{\pi}{6}$	2.61799

-(soustraction)**Touche** *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.

*Valeur - Liste1⇒liste**Liste1 - Valeur⇒liste*

Soustrait chaque élément de *Liste1* de *Valeur* ou soustrait *Valeur* de chaque élément de *Liste1* et donne la liste de résultats obtenue.

*Valeur - Matrice1⇒matrice**Matrice1 - Valeur⇒matrice*

Valeur - Matrice1 donne la matrice *Valeur* fois la matrice d'identité moins *Matrice1*. *Matrice1* doit être carrée.

Matrice1 - Valeur donne la matrice obtenue en soustrayant *Valeur* à chaque élément de la diagonale de *Matrice1*. *Matrice1* doit être carrée.

Remarque : Utilisez -- pour soustraire une expression à chaque élément de la matrice.

$$\begin{array}{c} \left\{ 22, \pi, \frac{\pi}{2} \right\} - \left\{ 10, 5, \frac{\pi}{2} \right\} \\ \hline [3 \ 4] - [1 \ 2] \end{array} \quad \left\{ 12, -1.85841, 0. \right\} \quad [2 \ 2]$$

$$\begin{array}{c} 15 - \{ 10, 15, 20 \} \\ \hline \{ 10, 15, 20 \} - 15 \end{array} \quad \left\{ 5, 0, 5 \right\} \quad \left\{ -5, 0, 5 \right\}$$

$$20 - \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$$

·(multiplication)**Touche** *Valeur1 · Valeur2⇒valeur*

Donne le produit des deux arguments.

Liste1 · Liste2⇒liste

Donne la liste contenant les produits des éléments correspondants de *Liste1* et *Liste2*.

Les listes doivent être de même dimension.

$$2 \cdot 3.45 \quad 6.9$$

$$\{ 1., 2, 3 \} \cdot \{ 4, 5, 6 \} \quad \{ 4, 10, 18 \}$$

·(multiplication)

Touche

Matrice1 · *Matrice2* ⇒ *matrice*

Donne le produit des matrices *Matrice1* et *Matrice2*.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} 7 & 8 \\ 7 & 8 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 42 & 48 \\ 105 & 120 \end{bmatrix}$$

Le nombre de colonnes de *Matrice1* doit être égal au nombre de lignes de *Matrice2*.

Valeur · *Liste1* ⇒ *liste*

$$\pi \cdot \{4,5,6\} = \{12.5664, 15.708, 18.8496\}$$

Liste1 · *Valeur* ⇒ *liste*

Donne la liste des produits de *Valeur* et de chaque élément de *Liste1*.

Valeur · *Matrice1* ⇒ *matrice*

Matrice1 · *Valeur* ⇒ *matrice*

Donne la matrice contenant les produits de *Valeur* et de chaque élément de *Matrice1*.

$$\begin{array}{c} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot 0.01 \\ 6 \cdot \text{identity}(3) \end{array} = \begin{bmatrix} 0.01 & 0.02 \\ 0.03 & 0.04 \\ 6 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 6 \end{bmatrix}$$

Remarque : Utilisez . · pour multiplier une expression par chaque élément.

/ (division)

Touche

Valeur1 / *Valeur2* ⇒ *valeur*

$$\frac{2}{3.45} = 0.57971$$

Donne le quotient de *Valeur1* par *Valeur2*.

Remarque : voir aussi **Modèle Fraction**, page 1.

Liste1 / *Liste2* ⇒ *liste*

Donne la liste contenant les quotients de *Liste1* par *Liste2*.

$$\frac{\{1,2,3\}}{\{4,5,6\}} = \left\{0.25, \frac{2}{5}, \frac{1}{2}\right\}$$

Les listes doivent être de même dimension.

Valeur / *Liste1* ⇒ *liste*

$$\frac{6}{\{3,6,\sqrt{6}\}} = \{2,1,2.44949\}$$

Liste1 / *Valeur* ⇒ *liste*

Donne la liste contenant les quotients de *Valeur* par *Liste1* ou de *Liste1* par *Valeur*.

$$\frac{\{7,9,2\}}{7 \cdot 9 \cdot 2} = \left\{\frac{1}{18}, \frac{1}{14}, \frac{1}{63}\right\}$$

/ (division)

Touche

Matrice1 / Valeur \Rightarrow matrice

Donne la matrice contenant les quotients des éléments de *Matrice1/Valeur*.

$$\frac{[7 \ 9 \ 2]}{7 \cdot 9 \cdot 2} \quad \left[\begin{array}{ccc} \frac{1}{18} & \frac{1}{14} & \frac{1}{63} \end{array} \right]$$

Remarque : Utilisez . / pour diviser une expression par chaque élément.

\wedge (puissance)

Touche

Valeur1 \wedge Valeur2 \Rightarrow valeur

16

Liste1 \wedge Liste2 \Rightarrow liste

$$\frac{4^2}{\{2,4,6\}^{\{1,2,3\}}} \quad \{2,16,216\}$$

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.

Valeur \wedge Liste1 \Rightarrow liste

$$\pi^{\{1,2,-3\}} \quad \{3.14159, 9.8696, 0.032252\}$$

Donne *Valeur* élevé à la puissance des éléments de *Liste1*.

Liste1 \wedge Valeur \Rightarrow liste

$$\{1,2,3,4\}^{-2} \quad \left\{ 1, \frac{1}{4}, \frac{1}{9}, \frac{1}{16} \right\}$$

Donne les éléments de *Liste1* élevés à la puissance de *Valeur*.

\wedge (puissance)

Touche

matriceCarrée1 \wedge *entier* \Rightarrow matrice

Donne *matriceCarrée1* élevée à la puissance de la valeur de *l'entier*.

matriceCarrée1 doit être une matrice carrée.

Si *entier* = -1, calcule la matrice inverse.

Si *entier* < -1, calcule la matrice inverse à une puissance positive appropriée.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^2 \quad \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

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

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-2} \quad \begin{bmatrix} \frac{11}{4} & -\frac{5}{4} \\ 2 & 2 \\ -\frac{15}{4} & \frac{7}{4} \end{bmatrix}$$

x^2 (carré)

Touche

*Valeur1*² \Rightarrow *valeur*

Donne le carré de l'argument.

*Liste1*² \Rightarrow *liste*

Donne la liste comportant les carrés des éléments de *Liste1*.

*matriceCarrée1*² \Rightarrow matrice

$$4^2 \quad 16$$

$$\{2,4,6\}^2 \quad \{4,16,36\}$$

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix}^2 \quad \begin{bmatrix} 40 & 64 & 88 \\ 49 & 79 & 109 \\ 58 & 94 & 130 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & 6 \\ 3 & 5 & 7 \\ 4 & 6 & 8 \end{bmatrix} \cdot^2 \quad \begin{bmatrix} 4 & 16 & 36 \\ 9 & 25 & 49 \\ 16 & 36 & 64 \end{bmatrix}$$

Donne le carré de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du carré de chaque élément. Utilisez \cdot^2 pour calculer le carré de chaque élément.

$\cdot+$ (addition élément par élément)

Touches

Matrice1 $\cdot+$ *Matrice2* \Rightarrow matrice

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \cdot+ \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix} \quad \begin{bmatrix} 11 & 32 \\ 23 & 44 \end{bmatrix}$$

Valeur $\cdot+$ *Matrice1* \Rightarrow matrice

$$5 \cdot+ \begin{bmatrix} 10 & 30 \\ 20 & 40 \end{bmatrix} \quad \begin{bmatrix} 15 & 35 \\ 25 & 45 \end{bmatrix}$$

Matrice1 $\cdot+$ *Matrice2* donne la matrice obtenue en effectuant la somme de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

Valeur $\cdot+$ *Matrice1* donne la matrice obtenue en effectuant la somme de *Valeur* et de chaque élément de *Matrice1*.

. - (soustraction élément par élément)

Touches

Matrice1 . - Matrice2 ⇒ matrice

Valeur . - Matrice1 ⇒ matrice

Matrice1 . - Matrice2 donne la matrice obtenue en calculant la différence entre chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

Valeur . - Matrice1 donne la matrice obtenue en calculant la différence de *Valeur* et de chaque élément de *Matrice1*.

$$\begin{array}{c} \left[\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right] . - \left[\begin{array}{cc} 10 & 20 \\ 30 & 40 \end{array} \right] \\ \hline \left[\begin{array}{cc} 10 & 20 \\ 30 & 40 \end{array} \right] \end{array} \quad \begin{array}{c} \left[\begin{array}{cc} -9 & -18 \\ -27 & -36 \end{array} \right] \\ \hline \left[\begin{array}{cc} -5 & -15 \\ -25 & -35 \end{array} \right] \end{array}$$

. · (multiplication élément par élément)

Touches

Matrice1 . · Matrice2 ⇒ matrice

Valeur . · Matrice1 ⇒ matrice

Matrice1 . · Matrice2 donne la matrice obtenue en calculant le produit de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

Valeur . · Matrice1 donne la matrice contenant les produits de *Valeur* et de chaque élément de *Matrice1*.

$$\begin{array}{c} \left[\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right] . \cdot \left[\begin{array}{cc} 10 & 20 \\ 30 & 40 \end{array} \right] \\ \hline \left[\begin{array}{cc} 10 & 20 \\ 30 & 40 \end{array} \right] \end{array} \quad \begin{array}{c} \left[\begin{array}{cc} 10 & 40 \\ 90 & 160 \end{array} \right] \\ \hline \left[\begin{array}{cc} 50 & 100 \\ 150 & 200 \end{array} \right] \end{array}$$

. / (division élément par élément)

Touches

Matrice1 . / Matrice2 ⇒ matrice

Valeur . / Matrice1 ⇒ matrice

Matrice1 . / Matrice2 donne la matrice obtenue en calculant le quotient de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

Valeur . / Matrice1 donne la matrice obtenue en calculant le quotient de *Valeur* et de chaque élément de *Matrice1*.

$$\begin{array}{c} \left[\begin{array}{cc} 1 & 2 \\ 3 & 4 \end{array} \right] . / \left[\begin{array}{cc} 10 & 20 \\ 30 & 40 \end{array} \right] \\ \hline \left[\begin{array}{cc} 10 & 20 \\ 30 & 40 \end{array} \right] \end{array} \quad \begin{array}{c} \left[\begin{array}{cc} 1 & 1 \\ 10 & 10 \\ 1 & 1 \\ 10 & 10 \end{array} \right] \\ \hline \left[\begin{array}{cc} 1 & 1 \\ 2 & 4 \\ 1 & 1 \\ 6 & 8 \end{array} \right] \end{array}$$

= (égal à)

Touche $=$

$Expr1 = Expr2 \Rightarrow Expression\ booleenne$

$Liste1 = Liste2 \Rightarrow Liste\ booleenne$

$Matrice1 = Matrice2 \Rightarrow Matrice\ booleenne$

Donne true s'il est possible de vérifier que la valeur de $Expr1$ est égale à celle de $Expr2$.

Donne false s'il est possible de déterminer que la valeur de $Expr1$ n'est pas égale à celle de $Expr2$.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

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

Exemple de fonction qui utilise les symboles de test mathématiques : $=$, \neq , $<$, \leq , $>$, \geq

Define $g(x) =$ Func

If $x \leq -5$ Then

Return 5

ElseIf $x > -5$ and $x < 0$ Then

Return $-x$

ElseIf $x \geq 0$ and $x \neq 10$ Then

Return x

ElseIf $x = 10$ Then

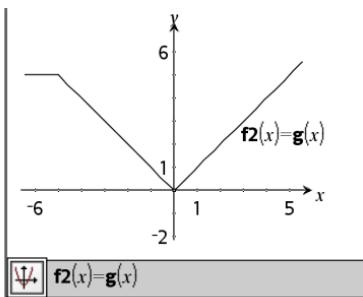
Return 3

EndIf

EndFunc

Done

Résultat de la représentation graphique de $g(x)$



\neq (différent de)

Touches ctrl $=$

$Expr1 \neq Expr2 \Rightarrow Expression\ booleenne$

Voir l'exemple fourni pour « = » (égal à).

$Liste1 \neq Liste2 \Rightarrow Liste\ booleenne$

$Matrice1 \neq Matrice2 \Rightarrow Matrice\ booleenne$

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 /=

< (inférieur à)

$Expr1 < Expr2 \Rightarrow$ Expression booléenne

Voir l'exemple fourni pour « = » (égal à).

$Liste1 < Liste2 \Rightarrow$ Liste booléenne

$Matrice1 < Matrice2 \Rightarrow$ 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.

≤ (inférieur ou égal à)

$Expr1 \leq Expr2 \Rightarrow$ Expression booléenne

Voir l'exemple fourni pour « = » (égal à).

$Liste1 \leq Liste2 \Rightarrow$ Liste booléenne

$Matrice1 \leq Matrice2 \Rightarrow$ 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 `<=`

$>$ (supérieur à)

$Expr1 > Expr2 \Rightarrow$ Expression booléenne

Voir l'exemple fourni pour « = » (égal à).

$Liste1 > Liste2 \Rightarrow$ Liste booléenne

$Matrice1 > Matrice2 \Rightarrow$ 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.

\geq (supérieur ou égal à)

$Expr1 \geq Expr2 \Rightarrow$ Expression booléenne

Voir l'exemple fourni pour « = » (égal à).

$Liste1 \geq Liste2 \Rightarrow$ Liste booléenne

$Matrice1 \geq Matrice2 \Rightarrow$ 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 \geq

\Rightarrow (implication logique)

touches ctrl =

BooleanExpr1 \Rightarrow BooleanExpr2
renvoie expression booléenne

BooleanList1 \Rightarrow BooleanList2 renvoie liste booléenne

BooleanMatrix1 \Rightarrow BooleanMatrix2
renvoie matrice booléenne

Integer1 \Rightarrow Integer2 renvoie entier

$5 > 3 \text{ or } 3 > 5$	true
$5 > 3 \Rightarrow 3 > 5$	false
$3 \text{ or } 4$	7
$3 \Rightarrow 4$	-4
$\{1, 2, 3\} \text{ or } \{3, 2, 1\}$	$\{3, 2, 3\}$
$\{1, 2, 3\} \Rightarrow \{3, 2, 1\}$	$\{-1, -1, -3\}$

Évalue l'expression **not** <argument1> **or** <argument2> et renvoie true (vrai) ou false (faux) ou une forme simplifiée de l'équation.

Pour les listes et matrices, renvoie le résultat des comparaisons, élément par élément.

Remarque : Vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant \Rightarrow

\Leftrightarrow (équivalence logique, XNOR)touches **ctrl** **=***BooleanExpr1 \Leftrightarrow BooleanExpr2*

renvoie expression booléenne

BooleanList1 \Leftrightarrow BooleanList2 renvoie liste booléenne*BooleanMatrix1 \Leftrightarrow BooleanMatrix2*
renvoie matrice booléenne*Integer1 \Leftrightarrow Integer2* renvoie entier

5>3 xor 3>5	true
5>3 \Leftrightarrow 3>5	false
3 xor 4	7
3 \Leftrightarrow 4	-8
{1,2,3} xor {3,2,1}	{2,0,2}
{1,2,3} \Leftrightarrow {3,2,1}	{-3,-1,-3}

Renvoie la négation d'une opération booléenne **XOR** sur les deux arguments.
Renvoie true (vrai) ou false (faux) ou une forme simplifiée de l'équation.

Pour les listes et matrices, renvoie le résultat des comparaisons, élément par élément.

Remarque : Vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant **<=>**

! (factorielle)Touche **?►***Valeur1!* \Rightarrow valeur

5! 120

Liste1! \Rightarrow liste

{ { 5,4,3 } }! { 120,24,6 }

Matrice1! \Rightarrow matrice
$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}!$$

$$\begin{pmatrix} 1 & 2 \\ 6 & 24 \end{pmatrix}$$

Donne la factorielle de l'argument.

Dans le cas d'une liste ou d'une matrice, donne la liste ou la matrice des factorielles de tous les éléments.

& (ajouter)Touches **ctrl** **alt***Chaîne1 & Chaîne2* \Rightarrow chaîne

"Hello "&"Nick" "Hello Nick"

Donne une chaîne de caractères obtenue en ajoutant *Chaîne2* à *Chaîne1*.

d() (dérivée)**d(Expr1, Var[, Ordre])** |

Var=Valeur⇒valeur

d(Expr1, Var[, Ordre])⇒valeur**d(Liste1, Var[, Ordre])⇒liste****d(Matrice1, Var[, Ordre])⇒matrice**

Excepté si vous utilisez la première syntaxe, vous devez stocker une valeur numérique dans la variable *Var* avant de calculer **d()**. Reportez-vous aux exemples.

d() peut être utilisé pour calculer la dérivée première et la dérivée seconde numérique en un point, à l'aide des méthodes de différenciation automatique.

Order, si utilisé, doit avoir la valeur **1** ou **2**. La valeur par défaut est **1**.

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

Remarque : voir aussi **Dérivée première**, page 5 ou **Dérivée seconde**, page 6.

Remarque : l'algorithme **d()** présente une limitation : il fonctionne de manière récursive à l'intérieur de l'expression non simplifiée et calcule la valeur de la dérivée première (et seconde, si cela est possible), puis évalue chacune des sous-expressions, ce qui peut générer un résultat inattendu.

Observez l'exemple ci-contre. La dérivée première de $x \cdot (x^2+x)^{(1/3)}$ en $x=0$ est égale à 0. Toutefois, comme la dérivée première de la sous-expression $(x^2+x)^{(1/3)}$ n'est pas définie à $x=0$ et que cette valeur est utilisée pour calculer la dérivée de l'expression complète, **d()** signale que le résultat n'est pas défini et affiche un message d'avertissement.

$\frac{d}{dx}(x) _{x=0}$	undef
$x:=0: \frac{d}{dx}(x)$	undef
$x:=3: \frac{d}{dx}\left(\left\{x^2, x^3, x^4\right\}\right)$	{6, 27, 108}

$\frac{d}{dx} \left. \left(x \cdot (x^2+x)^{\frac{1}{3}} \right) \right _{x=0}$	undef
$\text{centralDiff} \left. \left(x \cdot (x^2+x)^{\frac{1}{3}}, x \right) \right _{x=0}$	0.000033

d() (dérivée)

Catalogue >

Si vous rencontrez ce problème, vérifiez la solution en utilisant une représentation graphique. Vous pouvez également tenter d'utiliser **centralDiff()**.

ʃ() (intégrale)

Catalogue >

$\int(Expr1, Var, Borne1, Borne2) \Rightarrow$
valeur

Afiche l'intégrale de *Expr1* pour la variable *Var* entre *Borne1* et *Borne2*. Vous pouvez l'utiliser pour calculer l'intégrale définie numérique en utilisant la même méthode qu'avec **nInt()**.

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

Remarque : voir aussi **nInt()**, page 113 et le modèle **Intégrale définie**, page 6.

$$\int_0^1 x^2 \, dx \quad 0.333333$$

√() (racine carrée)

Touches

$\sqrt(Valeur1) \Rightarrow valeur$

$$\sqrt{4} \quad 2$$

$\sqrt(Liste1) \Rightarrow liste$

$$\sqrt{\{9,2,4\}} \quad \{3,1.41421,2\}$$

Donne la racine carrée de l'argument.

Dans le cas d'une liste, donne la liste des racines carrées des éléments de *Liste1*.

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

Remarque : voir aussi **Modèle Racine carrée**, page 1.

$\Pi()$ (prodSeq) $\Pi(Expr1, Var, Début, Fin) \Rightarrow expression$

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.

$$\frac{\prod_{n=1}^5 \left(\frac{1}{n}\right)}{\prod_{n=1}^5 \left(\left\{\frac{1}{n}, n, 2\right\}\right)} = \frac{1}{120}, 120, 32$$

Remarque : voir aussi **Modèle Produit** (Π), page 5.

 $\Pi(Expr1, Var, Début, Début-1) \Rightarrow 1$ $\Pi(Expr1, Var, Début, Fin)$

$\Rightarrow 1/\Pi(Expr1, Var, Fin+1, Début-1)$ if $Début < Fin-1$

$$\prod_{k=4}^3 (k) = 1$$

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.

$$\frac{\prod_{k=4}^1 \left(\frac{1}{k}\right)}{\prod_{k=4}^1 \left(\frac{1}{k}\right) \cdot \prod_{k=2}^4 \left(\frac{1}{k}\right)} = \frac{1}{4}$$

 $\Sigma()$ (sumSeq) $\Sigma(Expr1, Var, Début, Fin) \Rightarrow expression$

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.

$$\sum_{n=1}^5 \left(\frac{1}{n}\right) = \frac{13}{60}$$

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

 $\Sigma(Expr1, Var, Début, Fin-1) \Rightarrow 0$ $\Sigma(Expr1, Var, Début, Fin)$

$\Rightarrow -\Sigma(Expr1, Var, Fin+1, Début-1)$ if $Fin < Début-1$

$$\sum_{k=4}^3 (k) = 0$$

$\Sigma()$ (sumSeq)

Catalogue >

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.

$$\begin{array}{r} \frac{1}{\sum_{k=4}^{\infty} (k)} \\ \hline \frac{1}{\sum_{k=4}^{\infty} (k) + \sum_{k=2}^{\infty} (k)} \end{array}$$

$\Sigma\text{Int}()$

Catalogue >

$\Sigma\text{Int}(NPmt1, NPmt2, N, I, PV, [Pmt], [FV], [PpY], [CpY], [PmtAt], [valArrondi]) \Rightarrow \text{valeur}$

ΣInt

(
NPmt1
,NPmt2,tblAmortissement) \Rightarrow valeur

Fonction d'amortissement permettant de calculer la somme des intérêts au cours d'une plage de versements spécifiée.

NPmt1 et NPmt2 définissent le début et la fin de la plage de versements.

N, I, PV, Pmt, FV, PpY, CpY et PmtAt sont décrits dans le tableau des arguments TVM, page 180.

- 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.

$\Sigma\text{Int}(1,3,12,4.75,20000,,12,12) \quad -218.11$

tbl:=amortTbl(12,12,4.75,20000,,12,12)				
0	0.	0.	20000.	
1	-79.17	-1630.69	18369.3	
2	-72.71	-1637.15	16732.2	
3	-66.23	-1643.63	15088.5	
4	-59.73	-1650.13	13438.4	
5	-53.19	-1656.67	11781.7	
6	-46.64	-1663.22	10118.5	
7	-40.05	-1669.81	8448.7	
8	-33.44	-1676.42	6772.28	
9	-26.81	-1683.05	5089.23	
10	-20.14	-1689.72	3399.51	
11	-13.46	-1696.4	1703.11	
12	-6.74	-1703.12	-0.01	

$\Sigma\text{Int}(1,3,tbl) \quad -218.11$

$\Sigma\text{Int}(NPmt1, NPmt2, tblAmortissement)$
 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 7.

Remarque : voir également $\Sigma\text{Prn}()$ ci
 dessous et **Bal()**, page 16.

 $\Sigma\text{Prn}()$

$\Sigma\text{Prn}(NPmt1, NPmt2, N, I, PV, [Pmt],$
 $[FV], [PpY], [CpY], [PmtAt],$
 $[valArrondi]) \Rightarrow valeur$

 ΣPrn

(
 $NPmt1$
 $,NPmt2, tblAmortissement) \Rightarrow valeur$

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 180.

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

$\Sigma\text{Prn}(1, 3, 12, 4.75, 20000, , , 12, 12)$ -4911.47

$tbl:=\text{amortTbl}(12, 12, 4.75, 20000, , , 12, 12)$				
0	0.	0.	20000.	
1	-79.17	-1630.69	18369.3	
2	-72.71	-1637.15	16732.2	
3	-66.23	-1643.63	15088.5	
4	-59.73	-1650.13	13438.4	
5	-53.19	-1656.67	11781.7	
6	-46.64	-1663.22	10118.5	
7	-40.05	-1669.81	8448.7	
8	-33.44	-1676.42	6772.28	
9	-26.81	-1683.05	5089.23	
10	-20.14	-1689.72	3399.51	
11	-13.46	-1696.4	1703.11	
12	-6.74	-1703.12	-0.01	

$\Sigma\text{Prn}(1, 3, tbl)$ -4911.47

ΣPrn(*NPmt1*,*NPmt2*,tblAmortissement)
 calcule la somme du capital sur la base
 du tableau d'amortissement
tblAmortissement. L'argument
tblAmortissement doit être une matrice
 au format décrit à **tblAmortissement()**,
 page 7.

Remarque : voir également **ΣInt()** ci-
 dessus et **Bal()**, page 16.

(indirection)

ChaîneNomVar

Fait référence à la variable
ChaîneNomVar. Permet d'utiliser des
 chaînes de caractères pour créer des
 noms de variables dans une fonction.

Touches  

<i>xyz:=12</i>	12
<i>#("x" & "y" & "z")</i>	12

Crée ou fait référence à la variable xyz.

<i>10 → r</i>	10
<i>"r" → s1</i>	<i>"r"</i>
<i>#s1</i>	10

Donne la valeur de la variable (r) dont le nom
 est stocké dans la variable s1.

E (notation scientifique)

mantisseEexposant

Saisit un nombre en notation
 scientifique. Ce nombre est interprété
 sous la forme *mantisse* × 10^{exposant}.

Touche 

23000.	23000.
230000000.+4.1e15	4.1e15
3·10 ⁴	30000

Conseil : pour entrer une puissance de
 10 sans passer par un résultat de valeur
 décimale, utilisez 10^{entier}.

Remarque : vous pouvez insérer cet
 opérateur à partir du clavier de
 l'ordinateur en entrant @EE. Par exemple,
 entrez 2 . 3@EE4 pour avoir 2.3E4.

g (grades)

Touche **[1]**

Expr1g \Rightarrow expression

En mode Angle en degrés, grades ou radians :

Liste1g \Rightarrow liste

$$\cos(50^g)$$

0.707107

Matrice1g \Rightarrow matrice

$$\cos(\{0,100^g,200^g\})$$

{1,0.,-1.}

Cette fonction permet d'utiliser un angle en grades en mode Angle en degrés ou en radians.

En mode Angle en radians, multiplie *Expr1* par $\pi/200$.

En mode Angle en degrés, multiplie *Expr1* par $g/100$.

En mode Angle en grades, donne *Expr1* inchangée.

Remarque : vous pouvez insérer ce symbole à partir du clavier de l'ordinateur en entrant @g.

r(radians)

Touche **[1]**

Valeur1r \Rightarrow valeur

En mode Angle en degrés, grades ou radians :

Liste1r \Rightarrow liste

$$\cos\left(\frac{\pi}{4^r}\right)$$

0.707107

Matrice1r \Rightarrow matrice

$$\cos\left(\left\{0^r, \left(\frac{\pi}{12}\right)^r, -(\pi)^r\right\}\right)$$

{1,0.965926,-1.}

Cette fonction permet d'utiliser un angle en radians en mode Angle en degrés ou en grades.

En mode Angle en degrés, multiplie l'argument par $180/\pi$.

En mode Angle en radians, donne l'argument inchangé.

En mode Angle en grades, multiplie l'argument par $200/\pi$.

Conseil : utilisez 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.

° (degré)

Valeurs $1^{\circ} \Rightarrow$ valeur

Liste $1^{\circ} \Rightarrow$ liste

Matrice $1^{\circ} \Rightarrow$ matrice

Cette fonction permet d'utiliser un angle en degrés en mode Angle en grades ou en radians.

En mode Angle en radians, multiplie l'argument par $\pi/180$.

En mode Angle en degrés, donne l'argument inchangé.

En mode Angle en grades, multiplie l'argument par $10/9$.

Remarque : vous pouvez insérer ce symbole à partir du clavier de l'ordinateur en entrant @d.

En mode Angle en degrés, grades ou radians :

$$\cos(45^{\circ}) \quad 0.707107$$

En mode Angle en radians :

$$\cos\left(\left\{0, \frac{\pi}{4}, 90^{\circ}, 30.12^{\circ}\right\}\right) \\ \{1., 0.707107, 0., 0.864976\}$$

°, ', " (degré/minute/seconde)

$dd^{\circ}mm'ss.ss" \Rightarrow$ expression

dd Nombre positif ou négatif

mm Nombre positif ou nul

$ss.ss$ Nombre positif ou nul

Donne $dd + (mm/60) + (ss.ss/3600)$.

Ce format d'entrée en base 60 permet :-

- d'entrer un angle en degrés/minutes/secondes quel que soit le mode angulaire utilisé.
- d'entrer un temps exprimé en heures/minutes/secondes.

En mode Angle en degrés :

$$\begin{array}{r} 25^{\circ}13'17.5'' \\ 25^{\circ}30' \\ \hline & 51 \\ & 2 \end{array} \quad 25.2215$$

Remarque : faites suivre ss.ss de deux apostrophes ("') et non de guillemets ("").

∠ (angle)

[Rayon,∠θ_Angle]⇒vecteur

(entrée polaire)

[Rayon,∠θ_Angle,Valeur_Z]⇒vecteur

(entrée cylindrique)

[Rayon,∠θ_Angle,∠θ_Angle]⇒vecteur

(entrée sphérique)

Donne les coordonnées sous forme de vecteur, suivant le réglage du mode Format Vecteur : rectangulaire, cylindrique ou sphérique.

Remarque : vous pouvez insérer ce symbole à partir du clavier de l'ordinateur en entrant @<.

(Grandeur ∠ Angle)⇒valeurComplexe
(entrée polaire)

Saisit une valeur complexe en coordonnées polaires ($r\angle\theta$). L'Angle est interprété suivant le mode Angle sélectionné.

En mode Angle en radians et avec le Format vecteur réglé sur :

rectangulaire

[5 ∠60° ∠45°]

[1.76777 3.06186 3.53553]

cylindrique

[5 ∠60° ∠45°]

[3.53553 ∠1.0472 3.53553]

sphérique

[5 ∠60° ∠45°]

[5. ∠1.0472 ∠0.785398]

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

$5+3\cdot i\cdot \left(10 \angle \frac{\pi}{4}\right)$ -2.07107-4.07107·i

$5+3\cdot i\cdot \left(10 \angle \frac{\pi}{4}\right)$ -2.07107-4.07107·i

_ (trait bas considéré comme élément vide)

Voir “Éléments vides”, page 236.

10[^]()**Catalogue >** **10[^](Valeur1)⇒valeur**10^{1.5}

31.6228

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[^](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 >** **Valeur1^-1⇒valeur** $(3,1)^{-1}$

0.322581

Liste1^-1⇒liste

Donne l'inverse de l'argument.

Dans le cas d'une liste, donne la liste des inverses des éléments de *Liste1*.

matriceCarrée1^-1⇒matriceCarrée

Donne l'inverse de *matriceCarrée1*.

matriceCarrée1 doit être une matrice carrée non singulière.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \\ 2 & 2 \end{bmatrix}$$

| (opérateur "sachant que")**touches** **Expr | ExprBooléen1** $x+1|x=3$

4

[and]ExprBooléen2|... $x+55|x=\sin(55)$

54.0002

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 :

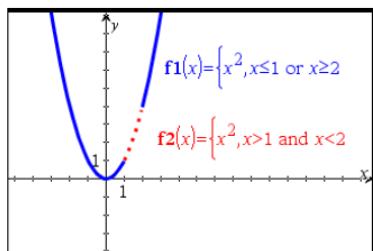
- Substitutions
- Contraintes d'intervalle
- Exclusions

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*.

Les contraintes d'intervalle se présentent sous la forme d'une ou plusieurs inéquations reliées par des opérateurs logiques « **and** » ou « **or** ». Les contraintes d'intervalle permettent également la simplification qui autrement pourrait ne pas être valide ou calculable.

$x^3 - 2 \cdot x + 7 \rightarrow f(x)$	<i>Done</i>
$f(x) x=\sqrt{3}$	8.73205

$nSolve(x^3 + 2 \cdot x^2 - 15 \cdot x = 0, x)$	0.
$nSolve(x^3 + 2 \cdot x^2 - 15 \cdot x = 0, x) x>0 \text{ and } x<5$	3.



Les exclusions utilisent l'opérateur « différent de » (\neq ou \neq) pour exclure une valeur spécifique du calcul.

→ (stocker)Touche **ctrl** **var***Valeur* → *Var*

$$\frac{\pi}{4} \rightarrow myvar \quad 0.785398$$

Liste → *Var*

$$2 \cdot \cos(x) \rightarrow yI(x) \quad Done$$

Matrix → *Var*

$$\{1,2,3,4\} \rightarrow lst5 \quad \{1,2,3,4\}$$

Expr → *Fonction(Param1,...)*

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

Liste → *Fonction(Param1,...)*

$$"Hello" \rightarrow str1 \quad "Hello"$$

Matrice → *Fonction(Param1,...)*

Si la variable *Var* n'existe pas, celle-ci est créée par cette instruction et est initialisée à *Valeur*, *Liste* ou *Matrice*.

Si *Var* existe déjà et n'est pas verrouillée ou protégée, son contenu est remplacé par *Valeur*, *Liste* ou *Matrice*.

Remarque : vous pouvez insérer cet opérateur à partir du clavier de l'ordinateur en entrant `=:` comme un raccourci. Par exemple, tapez `pi/4 =: Mavar`.

:= (assigner)Touches **ctrl** **var***Var* := *Valeur*

$$myvar := \frac{\pi}{4} \quad .785398$$

Var := *Liste*

$$yI(x) := 2 \cdot \cos(x) \quad Done$$

Var := *Matrice*

$$lst5 := \{1,2,3,4\} \quad \{1,2,3,4\}$$

Fonction(Param1,...) := *Expr*

$$matg := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

Fonction(Param1,...) := *Liste*

$$str1 := "Hello" \quad "Hello"$$

Fonction(Param1,...) := *Matrice*

Si la variable *Var* n'existe pas, celle-ci est créée par cette instruction et est initialisée à *Valeur*, *Liste* ou *Matrice*.

Si *Var* existe déjà et n'est pas verrouillée ou protégée, son contenu est remplacé par *Valeur*, *Liste* ou *Matrice*.

© [texte]

© traite *texte* comme une ligne de commentaire, vous permettant d'annoter les fonctions et les programmes que vous créez.

© peut être utilisé au début ou n'importe où dans la ligne. Tous les caractères situés à droite de ©, jusqu'à la fin de la ligne, sont considérés comme partie intégrante du commentaire.

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

Define $g(n) = \text{Func}$

© Declare variables

Local $i, result$

$result := 0$

For $i, 1, n, 1$ ©Loop n times

$result := result + i^2$

EndFor

Return $result$

EndFunc

Done

$g(3)$

14

0b, 0hTouches **0** **B**, touches **0** **H****0b nombreBinaire**

En mode base Dec :

0h nombreHexadécimal

0b10+0hF+10

27

Indique un nombre binaire ou hexadécimal, respectivement. Pour entrer un nombre binaire ou hexadécimal, vous devez utiliser respectivement le préfixe 0b ou 0h, quel que soit le mode Base utilisé. Un nombre sans préfixe est considéré comme décimal (base 10).

Le résultat est affiché en fonction du mode Base utilisé.

En mode base Bin :

0b10+0hF+10

0b11011

En mode base Hex :

0b10+0hF+10

0h1B

TI-Nspire™ CX II - Commandes graphiques

Ceci est un document d'appoint au Guide de référence de la TI-Nspire™ et de la TI-Nspire™ CAS. Toutes les instructions de la TI-Nspire™ CX II seront intégrées et publiées dans la version 5.1 du Guide de référence de la TI-Nspire™ et de la TI-Nspire™ CAS.

Programmation en mode graphique

De nouvelles commandes de programmation graphique ont été ajoutées aux unités TI-Nspire™ CX II et aux applications pour ordinateurs TI-Nspire™.

Les unités TI-Nspire™ CX II basculeront sur ce mode graphique pour exécuter les commandes graphiques avant de revenir au contexte d'exécution du programme initial.

L'écran affiche « En cours d'exécution » sur la barre supérieure pendant que le programme s'exécute. « Terminé » sera affiché à la fin du programme. Il suffit d'appuyer sur une touche quelconque pour faire sortir le système du mode graphique.

- Le passage au mode graphique est automatiquement déclenché lorsqu'une des commandes graphiques est trouvée durant l'exécution d'un programme en TI-Basic.
- Ce passage aura lieu uniquement lors de l'exécution d'un programme dans Calculs, dans un classeur ou dans Calculs dans le scratchpad
- La sortie du mode graphique se produit à la fin de l'exécution du programme.
- Le mode graphique est uniquement disponible sur les unités TI-Nspire™ CX II et dans la vue Unité du logiciel pour ordinateur TI-Nspire™ CX II. Il n'est pas disponible dans la vue Classeur sur PC ou sur Mac.
 - Si une commande graphique est trouvée durant l'exécution d'un programme TI-Basic dans un contexte incorrect, un message d'erreur sera affiché et l'exécution du programme TI-Basic sera interrompue.

Écran de représentation graphique

L'écran de représentation graphique contient une zone d'en-tête dans laquelle les commandes graphiques ne peuvent pas écrire.

La zone de tracé de l'écran graphique sera réinitialisée (couleur = 255,255,255) à son ouverture.

Écran de représentation graphique	Par défaut
Hauteur	212
Largeur	318
Couleur	blanc : 255,255,255

Vue et paramètres par défaut

- Les icônes d'état de la barre supérieure (voyant de batterie, verrouillage examen, indicateur de réseau etc.) ne sont pas visibles durant l'exécution d'un programme graphique.
- Couleur de trait par défaut : Noir (0,0,0)
- Style de stylo par défaut - normal, continu
 - Épaisseur : 1 (fin), 2 (normal), 3 (épais)
 - Style 1 (continu), 2 (tirets), 3 (pointillés)
- Toutes les commandes de tracé utiliseront les paramètres courants pour la couleur et le trait (valeurs par défaut ou définies via des commandes TI-Basic).
- La police du texte ne peut pas être modifiée.
- Toute sortie sur l'écran graphique sera dessinée dans une fenêtre dont la taille correspond à la zone de tracé de l'écran graphique. Toute sortie qui dépasse cette zone de tracé délimitée ne sera pas représentée. Aucun message d'erreur ne sera affiché.
- Les coordonnées (x, y) spécifiées par les commandes de tracé sont définies de façon à ce que (0,0) représente le coin supérieur gauche de la zone de tracé de l'écran graphique.
 - **Exceptions :**
 - Pour l'instruction **DrawText**, les coordonnées indiquées comme paramètres désignent l'angle inférieur gauche de la zone de délimitation du texte.
 - **SetWindow** utilise l'angle inférieur gauche de l'écran.
- Tous les paramètres (arguments) des commandes peuvent être fournis sous forme d'expressions qui sont évaluées à des nombres arrondis à l'entier le plus proche.

Messages d'erreur de l'écran graphique

Un message d'erreur sera affiché si la validation échoue.

Message d'erreur	Description	Afficher
Erreur Syntaxe	Si des erreurs de syntaxe sont détectées, un message d'erreur s'affiche et le curseur est placé, dans la mesure du possible, au niveau de la première erreur pour que vous puissiez la rectifier.	
Erreur Nombre insuffisant d'arguments	Un ou plusieurs arguments de la fonction ou de la commande n'ont pas été spécifiés	Error Too few arguments The function or command is missing one or more arguments. OK
Erreur Trop d'arguments	La fonction ou la commande est impossible à évaluer car elle contient trop d'arguments.	Error Too many arguments The function or command contains an excessive number of arguments and cannot be evaluated. OK
Erreur Type de données incorrect	Le type de donnée de l'un des arguments est incorrect	Error Invalid data type An argument is of the wrong data type. OK

Commandes non valides dans le mode graphique

Certaines commandes ne sont pas autorisées une fois que le programme passe en mode graphique. Si ces commandes sont rencontrées en mode graphique, une erreur sera affichée et l'exécution du programme s'arrêtera.

Commande non autorisée	Message d'erreur
Request	La fonction Request ne peut pas être exécutée en mode graphique
RequestStr	La fonction RequestStr ne peut pas être exécutée en mode graphique
Texte	La fonction Text ne peut pas être exécutée en mode graphique

Les commandes qui affichent du texte dans Calculs - **disp** et **dispAt** - sont prises en charge dans le contexte graphique. Le texte de ces commandes sera envoyé à l'écran Calculs (et non pas sur l'écran graphique) et sera visible à la fin du programme, lorsque le système revient à l'application Calculs.

Supprimer**Catalogue >** 
CXII**Clear *x, y, largeur, hauteur***

Efface tout l'écran si aucun paramètre n'est spécifié.

Si *x, y, largeur, hauteur* sont spécifiés, le rectangle spécifié par ces paramètres sera effacé.

Supprimer

Efface la totalité de l'écran

Clear 10,10,100,50

Efface une zone rectangulaire dont le sommet supérieur gauche a pour coordonnées (10,10), une largeur égale à 100 et une hauteur à 50.

DrawArc

Catalogue >  CXII

DrawArc $x, y, \text{largeur}, \text{hauteur}, \text{startAngle}, \text{arcAngle}$

Trace un arc dans le rectangle spécifié, avec les angles de départ et d'arc fournis.

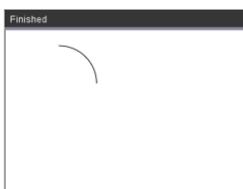
x, y : coordonnées du sommet supérieur gauche du rectangle de délimitation

largeur, hauteur : dimensions du rectangle de délimitation

L'argument « arc angle » définit l'angle de balayage de l'arc.

Ces paramètres (arguments) peuvent être fournis sous forme d'expressions dont le résultat est arrondi à l'entier le plus proche.

DrawArc 20,20,100,100,0,90



DrawArc 50,50,100,100,0,180



Voir également : [FillArc](#)

DrawCircle

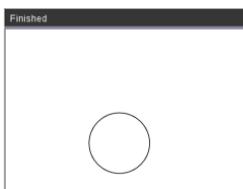
Catalogue >  CXII

DrawCircle x, y, rayon

x, y : coordonnées du centre

rayon : rayon du cercle

DrawCircle 150,150,40



Voir également : [FillCircle](#)

DrawLine

Catalogue > CXII

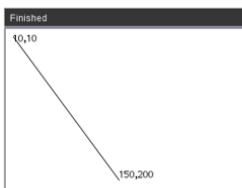
DrawLine x_1, y_1, x_2, y_2

Trace un segment d'extrémités (x_1, y_1) et (x_2, y_2).

Expressions dont le résultat est arrondi à l'entier le plus proche.

Limites de l'écran : Si les coordonnées spécifiées impliquent qu'une partie du segment soit tracée en dehors de l'écran graphique, cette partie sera tronquée sans qu'aucun message d'erreur ne soit affiché.

DrawLine 10,10,150,200



DrawPoly

Catalogue > CXII

Les instructions ont deux variantes :

DrawPoly $xlist, ylist$

ou

DrawPoly $x_1, y_1, x_2, y_2, x_3, y_3 \dots x_n, y_n$

Remarque : DrawPoly $xlist, ylist$
Shape relierax $1, y_1$ à x_2, y_2, x_2, y_2 à x_3, y_3 et ainsi de suite.

Remarque : DrawPoly $x_1, y_1, x_2, y_2, x_3, y_3 \dots x_n, y_n$

x_n, y_n ne seront PAS reliés automatiquement à x_1, y_1 .

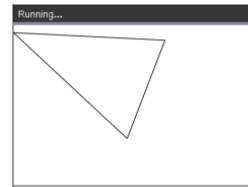
Expressions retournant une liste de nombres réels à virgule flottante
 $xlist, ylist$

Expressions évaluées à un nombre réel à virgule flottante
 $x_1, y_1 \dots x_n, y_n$ = coordonnées des sommets du polygone

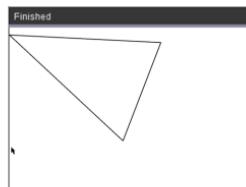
$xlist := \{0, 200, 150, 0\}$

$ylist := \{10, 20, 150, 10\}$

DrawPoly xlist,ylist



DrawPoly 0,10,200,20,150,10,0,10



Remarque : DrawPoly : Permet de spécifier les dimensions (largeur/ hauteur) par rapport aux segments tracés.
Les segments sont tracés dans une zone de délimitation autour des coordonnées spécifiées et dimensionnées de façon à ce que la taille réelle du polygone tracé soit supérieure à la largeur et à la hauteur indiquées.

Voir également : [FillPoly](#)

DrawRect

DrawRect *x, y, largeur, hauteur*

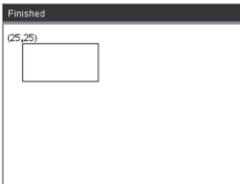
x, y : coordonnées du sommet supérieur gauche du rectangle

hauteur, largeur : hauteur et largeur du rectangle (rectangle tracé vers le bas et vers la droite à partir des coordonnées de départ).

Remarque : Les segments sont tracés dans une zone de délimitation autour des coordonnées spécifiées dont les dimensions font que la taille réelle du rectangle tracé sera supérieure à la largeur et à la hauteur indiquées.

Voir également : [FillRect](#)

DrawRect 25,25,100,50



DrawText

DrawText *x, y, exprOrString1*
[exprOrString2]...

x, y : coordonnées du texte affiché

Trace le texte inclus dans *exprOrString* à l'emplacement spécifié par les coordonnées *x, y* indiquées.

DrawText 50,50,"Hello World"



Les règles pour *exprOrString* sont les mêmes que pour **Disp** – **DrawText** peut avoir plusieurs arguments.

FillArc

Catalogue > CXII

FillArc $x, y, \text{largeur}, \text{hauteur}, \text{startAngle}, \text{arcAngle}$

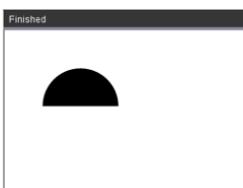
x, y : coordonnées du sommet supérieur gauche du rectangle de délimitation

Trace et remplit un arc dans le rectangle défini, en utilisant l'angle de départ et l'angle de balayage indiqués.

La couleur de remplissage par défaut est le noir. La couleur de remplissage peut être définie via la commande [SetColor](#)

L'argument « arc angle » définit l'angle de balayage de l'arc

FillArc 50,50,100,100,0,180

**FillCircle**

Catalogue > CXII

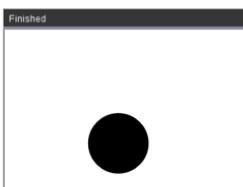
FillCircle x, y, rayon

x, y : coordonnées du centre

Trace et remplit un cercle de centre et de rayon spécifiés.

La couleur de remplissage par défaut est le noir. La couleur de remplissage peut être définie via la commande [SetColor](#).

FillCircle 150,150,40



Ici !

FillPoly

Catalogue > CXII

FillPoly $xlist, ylist$

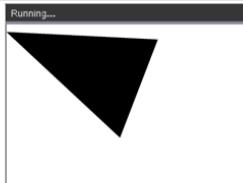
ou

FillPoly $x_1, y_1, x_2, y_2, x_3, y_3, \dots, x_n, y_n$

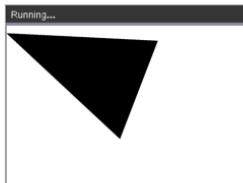
$xlist := \{0, 200, 150, 0\}$

$ylist := \{10, 20, 150, 10\}$

FillPoly $xlist, ylist$



FillPoly `0,10,200,20,150,150,0,10`



FillRect

FillRect *x, y, largeur, hauteur*

x, y : coordonnées du sommet supérieur gauche du rectangle

largeur, hauteur : largeur et hauteur du rectangle

Trace et remplit un rectangle dont le sommet supérieur gauche a pour coordonnées les valeurs spécifiées (*x,y*)

La couleur de remplissage par défaut est le noir. La couleur de remplissage peut être définie via la commande [SetColor](#)

Remarque : Le trait et la couleur sont définis par [SetColor](#) et [SetPen](#)

FillRect `25,25,100,50`



getPlatform()Catalogue > 
CXII**getPlatform()**

getPlatform()

"dt"

Renvoie :

« dt » sur les applications logicielles pour ordinateur
« hh » sur les unités TI-Nspire™ CX
« ios » sur l'appli TI-Nspire™ CX pour iPad®

PaintBuffer

Catalogue >  CXII

PaintBuffer

Dessine le contenu du cache graphique sur l'écran

Cette commande s'utilise en conjonction avec UseBuffer pour augmenter la vitesse d'affichage sur l'écran lorsque le programme génère de multiples objets graphiques.

UseBuffer

```
For n,1,10
x:=randInt(0,300)
y:=randInt(0,200)
radius:=randInt(10,50)
Wait 0,5
DrawCircle x, y, rayon
EndFor
PaintBuffer
```

Ce programme affichera les 10 cercles simultanément.

Si l'instruction « UseBuffer » est retirée, chaque cercle sera affiché lorsqu'il est tracé.

Voir également : [UseBuffer](#)

PlotXY *x, y, forme**x, y* : coordonnées du tracé de la forme*forme* : un nombre compris entre 1 et 13 qui indique la forme

1 - Cercle plein

2 - Cercle vide

3 - Carré plein

4 - Carré vide

5 - Croix

6 - Plus

7 - Fin

8 - point moyen, plein

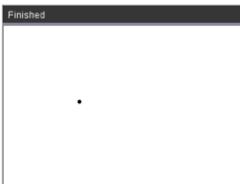
9 - point moyen, vide

10 - point large, plein

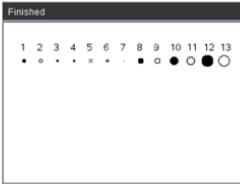
11 - point large, vide

12 - point extra large, plein

13 - point extra large, vide

PlotXY *100,100,1*For *n*,1,13DrawText 1+22*n,40,*n*PlotXY 5+22*n,50,*n*

EndFor



SetColor

Catalogue > CXII

SetColor

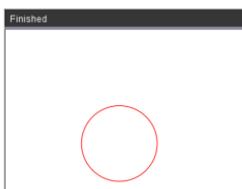
valeur rouge, valeur vert, valeur bleu

Les valeurs valides pour le rouge, le vert et le bleu sont comprises entre 0 et 255

Définit la couleur pour les commandes de tracé suivantes.

`SetColor 255,0,0`

`DrawCircle 150,150,100`

**SetPen**

Catalogue > CXII

SetPen

épaisseur, style

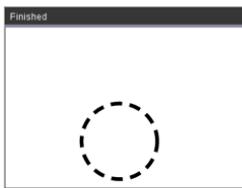
épaisseur : $1 \leqslant$ épaisseur $\leqslant 3$ | 1 est le plus fin, 3 est le plus épais

style : 1 = Continu, 2 = Tirets, 3 = Pointillés

Définit le style du stylo pour les commandes de tracé suivantes

`SetPen 3,3`

`DrawCircle 150,150,50`

**SetWindow**

Catalogue > CXII

SetWindow

xMin, xMax, yMin, yMax

Établit une fenêtre logique qui correspond à la zone de représentation graphique Tous les paramètres sont obligatoires.

Si une partie de l'objet tracé se situe en dehors de la fenêtre, le résultat sera tronqué (non affiché) sans qu'aucun message d'erreur ne soit affiché.

`SetWindow 0,160,0,120`

Définit les coordonnées de l'angle inférieur gauche de la fenêtre de sortie en $0,0$ avec une largeur de 160 et une hauteur de 120

`DrawLine 0,0,100,100`

`SetWindow 0,160,0,120`

`SetPen 3,3`

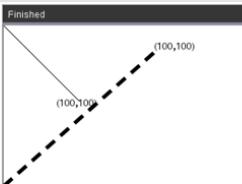
`DrawLine 0,0,100,100`

Si x_{\min} est supérieur ou égal à x_{\max} ou si y_{\min} est supérieur ou égal à y_{\max} , un message d'erreur s'affiche.

Tout objet tracé avant une instruction SetWindow ne sera pas retracé dans la nouvelle configuration.

Pour restaurer les paramètres par défaut de la fenêtre, utilisez :

SetWindow 0,0,0,0



UseBuffer**UseBuffer**

Envoye vers la mémoire tampon de l'écran graphique au lieu d'afficher à l'écran (pour améliorer les performances)

Cette instruction est utilisée avec PaintBuffer pour accélérer l'affichage sur l'écran lorsque le programme génère de multiples objets graphiques.

Avec UseBuffer, tous les graphiques sont affichés uniquement après l'exécution de la commande PaintBuffer suivante.

UseBuffer n'a besoin d'être appelée qu'une seule fois dans le programme : chaque instruction PaintBuffer n'a pas besoin d'avoir une instruction UseBuffer correspondante.

UseBuffer

```
For n,1,10
x:=randInt(0,300)
y:=randInt(0,200)
```

```
radius:=randInt(10,50)
```

```
Wait 0,5
```

```
DrawCircle x, y, rayon
```

```
EndFor
```

```
PaintBuffer
```

Ce programme affichera les 10 cercles simultanément.

Si l'instruction « UseBuffer » est retirée, chaque cercle sera affiché lorsqu'il est tracé.

Voir également : [PaintBuffer](#)

Éléments vides

Lors de l'analyse de données réelles, il est possible que vous ne disposiez pas toujours d'un jeu complet de données. TI-Nspire™ 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 42 et **isVoid()**, page 83.

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.

<code>_</code>	-
<code>gcd(100,_)</code>	-
<code>3+-</code>	-
<code>{5,_..,10}-{3,6,9}</code>	<code>{2,_..,1}</code>

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.

<code>sum({2,_..,3,5,6,6})</code>	16.6
<code>median({1,2,_..,3})</code>	2
<code>cumulativeSum({1,2,_..,4,5})</code>	<code>{1,3,_..,7,12}</code>
<code>cumulativeSum([[1 2], [3 _], [5 6]])</code>	<code>[1 2]</code> <code>[4 _]</code> <code>[9 8]</code>

Arguments de liste contenant des éléments vides

SortA et **SortD** déplacent tous les éléments vides du premier argument au bas de la liste.

$\{5,4,3,_,1\} \rightarrow list1$	$\{5,4,3,_,1\}$
$\{5,4,3,2,1\} \rightarrow list2$	$\{5,4,3,2,1\}$
SortA $list1, list2$	Done
$list1$	$\{1,3,4,5,_\}$
$list2$	$\{1,3,4,5,2\}$

Dans les regressions, la présence d'un élément vide dans la liste X ou Y génère un élément vide correspondant dans le résidu.

$\{1,2,3,_,5\} \rightarrow list1$	$\{1,2,3,_,5\}$
$\{1,2,3,4,5\} \rightarrow list2$	$\{1,2,3,4,5\}$
SortD $list1, list2$	Done
$list1$	$\{5,3,2,1,_\}$
$list2$	$\{5,3,2,1,4\}$

L'omission d'une catégorie dans les calculs de régression 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
$stat.Resid$	$\{0.434286,_, -0.862857, -0.011429, 0.44\}$
$stat.XReg$	$\{1,_,3,4,5,\}$
$stat.YReg$	$\{2,_,3,5,6,6\}$
$stat.FreqReg$	$\{1,_,1,1,1,\}$

Une fréquence 0 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
$stat.Resid$	$\{_,_,0,0,\}$
$stat.XReg$	$\{_,_,4,5,\}$
$stat.YReg$	$\{_,_,5,6,6\}$
$stat.FreqReg$	$\{_,_,1,1,\}$

$l1:=\{1,3,4,5\}; l2:=\{2,3,5,6,6\}$	$\{2,3,5,6,6\}$
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	<code>pi</code>
θ	<code>theta</code>
∞	<code>infinity</code>
\leq	<code><=</code>
\geq	<code>>=</code>
\neq	<code>/=</code>
\Rightarrow (implication logique)	<code>=></code>
\Leftrightarrow (équivalence logique, XNOR)	<code><=></code>
\rightarrow (opérateur de stockage)	<code>:=</code>
$ $ (valeur absolue)	<code>abs(...)</code>
$\sqrt{0}$	<code>sqrt(...)</code>
$\Sigma()$ (Modèle Somme)	<code>sumSeq(...)</code>
$\Pi()$ (Modèle Produit)	<code>prodSeq(...)</code>
$\sin^{-1}(), \cos^{-1}(), \dots$	<code>arcsin(...), arccos(...), ...</code>
$\Delta\text{List}()$	<code>deltaList(...)</code>

Sur le clavier de l'ordinateur

Pour saisir :	Utilisez le raccourci :
i (le nombre complexe)	<code>@i</code>

Pour saisir :	Utilisez le raccourci :
<i>e</i> (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™. 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-seconde (°,'"), 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 (\neq ou $/=$), inférieur à (<), inférieur ou égal à (\leq ou $<=$), supérieur à (>), supérieur ou égal à (\geq ou $>=$)
11	not logique
12	and logique
13	Logique or
14	xor, nor, nand
15	Implication logique (\Rightarrow)
16	Équivalence logique, XNOR (\Leftrightarrow)
17	Opérateur "sachant que" (« »)
18	Stocker (\rightarrow)

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™ vous permet de définir des fonctions personnalisées, un nom de variable suivi d'une expression entre parenthèses est considéré comme un « appel de fonction » et non comme une multiplication implicite. Par exemple, $a(b+c)$ est la fonction a évaluée en $b+c$. Pour multiplier l'expression $b+c$ par la variable a , utilisez la multiplication explicite : $a*(b+c)$.

Indirection

L'opérateur d'indirection (#) convertit une chaîne en une variable ou en un nom de fonction. Par exemple, $\#("x"&"y"&"z")$ crée le nom de variable « xyz ». Cet opérateur permet également de créer et de modifier des variables à partir d'un programme. Par exemple, si $10\rightarrow r$ et $"r"\rightarrow s1$, donc $s1=10$.

Opérateurs en suffixe

Les opérateurs en suffixe sont des opérateurs qui suivent immédiatement un argument, comme $5!$, 25% ou $60^{\circ}15'45''$. Les arguments suivis d'un opérateur en suffixe ont le niveau de priorité 4 dans l'ordre d'évaluation. Par exemple, dans l'expression $4^3!$, $3!$ est évalué en premier. Le résultat, 6, devient l'exposant de 4 pour donner 4096.

Élévation à une puissance

L'élévation à la puissance (^) et l'élévation à la puissance élément par élément (.^) sont évaluées de droite à gauche. Par exemple, l'expression 2^3^2 est évaluée comme $2^{(3^2)}$ 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 élévarions à la puissance postérieures sont évaluées avant la négation. Par exemple, le résultat de $-x^2$ est un nombre négatif et $-9^2 = -81$. Utilisez les parenthèses pour mettre un nombre négatif au carré, comme $(-9)^2$ qui donne 81.

Contrainte (« | »)

L'argument qui suit l'opérateur "sachant que" (« | ») applique une série de contraintes qui affectent l'évaluation de l'argument qui précède l'opérateur.

Fonctions de programmation TI-Basic sur TI-Nspire CX II

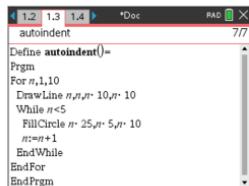
Auto-indentation dans l'Éditeur de programmes

L'Éditeur de programmes de la TI-Nspire™ indente désormais les instructions dans un bloc de commandes.

Les blocs de commandes sont If/EndIf, For/EndFor, While/EndWhile, Loop/EndLoop, Try/EndTry

L'éditeur indente automatiquement les commandes qui se trouvent dans un bloc d'instructions. L'instruction de fin de bloc sera alignée avec l'instruction de début de bloc.

L'exemple ci-dessous illustre l'indentation automatique dans les instructions de bloc imbriquées.



```
Define autoindent()=
Prgm
For n,1,10
DrawLine n,n,n- 10,n- 10
While n<5
FillCircle n- 25,n- 5,n- 10
n:=n+1
EndWhile
EndFor
EndPrgm
```

Les fragments de code qui sont copiés -collés conservent leur indentation originale.

Un programme créé avec une version précédente du logiciel conservera son indentation originale à l'ouverture.

Messages d'erreur améliorés pour TI-Basic

Erreurs

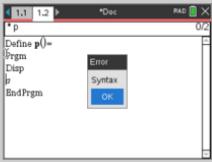
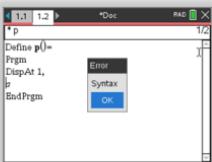
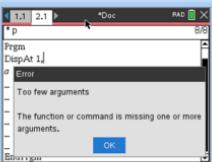
Condition d'erreur	Nouveau message
Erreur dans une instruction conditionnelle (If/While)	L'une des conditions a renvoyé une valeur qui n'était ni VRAI ni FAUX REMARQUE : Le curseur étant désormais placé sur la ligne où se trouve l'erreur, nous n'avons plus besoin d'indiquer si l'erreur se trouvait dans une instruction « If » ou une instruction « While ».
Instruction EndIf manquante	L'instruction de fin devrait être EndIf, mais une instruction de fin différente a été trouvée
Instruction Endfor manquante	L'instruction de fin devrait être EndFor, mais une instruction de fin différente a été trouvée
Instruction EndWhile manquante	L'instruction de fin devrait être EndWhile, mais une instruction de fin différente a été trouvée

Condition d'erreur	Nouveau message
Instruction EndLoop manquante	L'instruction de fin devrait être EndLoop , mais une instruction de fin différente a été trouvée
Instruction EndTry manquante	L'instruction de fin devrait être EndTry , mais une instruction de fin différente a été trouvée
« Then » manquant après If <condition>	Instruction If..Then manquante
« Then » manquant après ElseIf <condition>	Instruction Then manquante dans le bloc : ElseIf
En cas d'instruction « Then », « Else » ou « ElseIf » trouvée en dehors des blocs de contrôle.	Instruction Else invalide en dehors des blocs : If..Then..EndIf ou Try..EndTry
« ElseIf » apparaît en dehors d'un bloc « If..Then..EndIf »	Instruction ElseIf invalide en dehors du bloc : If..Then..EndIf
« Then » apparaît en dehors d'un bloc « If....EndIf »	Instruction Then invalide en dehors du bloc : If..EndIf

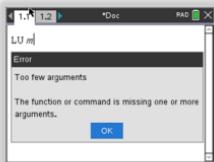
Erreurs de syntaxe

Si des instructions qui attendent un ou plusieurs arguments sont appelées avec un nombre insuffisant d'arguments, une erreur « **Nombre insuffisant d'arguments** » sera générée au lieu d'une « **erreur de syntaxe** »

Comportement actuel	Nouveau comportements de la CX II

Comportement actuel	Nouveau comportements de la CX II
	
	

Remarque : Lorsqu'une liste d'arguments incomplète n'est pas suivie d'une virgule, le message d'erreur est : « Nombre insuffisant d'arguments » Idem que pour les versions précédentes.



Constantes et valeurs

Le tableau suivant liste les constantes ainsi que leurs valeurs qui sont disponibles lors de la réalisation de conversions d'unités. Elles peuvent être saisies manuellement ou sélectionnées depuis la liste **Constantes** dans **Utilitaires > Conversions d'unité** (Unité nomade : Appuyez sur 3).

Constante	Nom	Valeur
_c	Vitesse de la lumière	299792458 _m/_s
_Cc	Constante de Coulomb	8987551792.261 _m/_F
_Fc	Constante de Faraday	96485.33212 _coul/_mol
_g	Accélération de la pesanteur	9.80665 _m/_s ²
_Gc	Constante de gravitation	6.6743E-11 _m ³ /_kg/_s ²
_h	Constante de Planck	6.62607015E-34 _J_s
_k	Constante de Boltzmann	1.380649E-23 _J/_°K
_μ0	Perméabilité du vide	1.25663706212E-6 _N/_A ²
_μb	Magnéton de Bohr	9.274009994E-24 _J_m ² /_Wb
_Me	Masse de l'électron	9.1093837015E-31 _kg
_Mμ	Masse du muon	1.883531627E-28 _kg
_Mn	Masse du neutron	1.67492749804E-27 _kg
_Mp	Masse du proton	1.67262192369E-27 _kg
_Na	Nombre d'Avogadro	6.02214076E23 /_mol
_q	Charge de l'électron	1.602176634E-19 _coul
_Rb	Rayon de Bohr	5.29177210903E-11 _m
_Rc	Constante molaire des gaz	8.314462618 _J/_mol/_°K
_Rdb	Constante de Rydberg	10973731.568160/_m
_Re	Rayon de l'électron	2.8179403262E-15 _m
_u	Masse atomique	1.6605390666E-27 _kg
_Vm	Volume molaire	2.241396954E-2 _m ³ /_mol
_ε0	Permittivité du vide	8.8541878128E-12 _F/_m
_σ	Constante de Stefan-Boltzmann	5.670367E-8 _W/_m ² /_°K ⁴
_Φ0	Quantum de flux magnétique	2.067833831E-15 _Wb

Codes et messages d'erreur

En cas d'erreur, le code correspondant est assigné à la variable *errCode*. Les programmes et fonctions définis par l'utilisateur peuvent être utilisés pour analyser *errCode* et déterminer l'origine de l'erreur. Pour obtenir un exemple d'utilisation de *errCode*, reportez-vous à l'exemple 2 fourni pour la commande Try, page 176.

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 : <ul style="list-style-type: none">• ne commence pas par un chiffre,• ne contienne ni espaces ni caractères spéciaux,• n'utilise pas le tiret de soulignement ou le point de façon incorrecte,• ne dépasse pas les limitations de longueur. Pour plus d'informations à ce sujet, reportez-vous à la section Calculs dans la documentation.
160	L'argument doit être une expression.
165	Piles trop faibles pour envoi/réception Installez des piles neuves avant toute opération d'envoi ou de réception.
170	Bornes

Code d'erreur	Description
	Pour définir l'intervalle de recherche, la limite inférieure doit être inférieure à la limite supérieure.
180	Arrêt de calcul
	Une pression sur la touche [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, <code>solve(3x^2-4=0,x) x<0 or x>5</code> génère ce message d'erreur car "or" est utilisé à la place de "and" pour séparer les contraintes.
210	Type de données incorrect
	Le type de l'un des arguments est incorrect.
220	Limite dépendante
230	Dimension
	Un index de liste ou de matrice n'est pas valide. Par exemple, si la liste {1,2,3,4} est stockée dans L1, L1[5] constitue une erreur de dimension, car L1 ne comporte que quatre éléments.
235	Erreur de dimension. Le nombre d'éléments dans les listes est insuffisant.
240	Dimension inadaptée
	Deux arguments ou plus doivent être de même dimension. Par exemple, [1,2]+[1,2,3] constitue une dimension inadaptée, car les matrices n'ont pas le même nombre d'éléments.
250	Division par zéro
260	Erreur de domaine
	Un argument doit être situé dans un domaine spécifique. Par exemple, <code>rand(0)</code> est incorrect.
270	Nom de variable déjà utilisé
280	Else et Elseif sont invalides hors du bloc If..EndIf.
290	La déclaration Else correspondant à EndTry manque.
295	Nombre excessif d'itérations

Code d'erreur	Description
300	Une liste ou une matrice de dimension 2 ou 3 est requise.
310	Le premier argument de nSolve doit être une équation d'une seule variable. Il ne doit pas contenir d'inconnue autre que la variable considérée.
320	Le premier argument de solve ou cSolve doit être une équation ou une inéquation. Par exemple, solve($3x^2-4,x$) n'est pas correct car le premier argument n'est pas une équation.
345	Unités incompatibles
350	Indice non valide
360	La chaîne d'indirection n'est pas un nom de variable valide.
380	Ans invalide Le calcul précédent n'a pas créé Ans, ou aucun calcul précédent n'a pas été entré.
390	Affectation invalide
400	Valeur d'affectation invalide
410	Commande invalide
430	Invalide pour les réglages du mode en cours
435	Valeur Init invalide
440	Multiplication implicite invalide Par exemple, $x(x+1)$ est incorrect ; en revanche, $x*(x+1)$ est correct. Cette syntaxe permet d'éviter toute confusion entre les multiplications implicites et les appels de fonction.
450	Invalide dans une fonction ou expression courante Seules certaines commandes sont valides à l'intérieur 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	Invalidé hors programme
570	Nom de chemin invalide Par exemple, \var est incorrect.
575	Complexe invalide en polaire
580	Référence de programme invalide Les programmes ne peuvent pas être référencés à l'intérieur de fonctions ou d'expressions, comme par exemple 1+p(x), où p est un programme.
600	Table invalide
605	Utilisation invalide d'unités
610	Nom de variable invalide dans une déclaration locale
620	Nom de variable ou de fonction invalide
630	Référence invalide à une variable
640	Syntaxe vectorielle invalide
650	Transmission La transmission entre deux unités n'a pas pu aboutir. Vérifiez que les deux extrémités du câble sont correctement branchées.
665	Matrice non diagonalisable
670	Mémoire insuffisante 1. Supprimez des données de ce classeur. 2. Enregistrez, puis fermez ce classeur. Si les suggestions 1 & 2 échouent, retirez les piles, puis remettez-les en place.
680	{ manquante
690) manquante
700	" manquant
710] manquant
720	} manquante
730	Manque d'une instruction de début ou de fin de bloc
740	Then manquant dans le bloc If..EndIf
750	Ce nom n'est pas un nom de fonction ou de programme.
765	Aucune fonction n'est sélectionnée.

Code d'erreur	Description
672	Dépassement des ressources
673	Dépassement des ressources
780	Aucune solution n'a été trouvée.
800	Résultat non réel Par exemple, si le logiciel est réglé sur Réel, $\sqrt{(-1)}$ n'est pas valide. Pour autoriser les résultats complexes, réglez le mode "Réel ou Complex" sur "RECTANGULAIRE ou POLAIRE".
830	Capacité
850	Programme introuvable Une référence de programme à l'intérieur d'un autre programme est introuvable au chemin spécifié au cours de l'exécution.
855	Les fonctions aléatoires ne sont pas autorisées en mode graphique.
860	Le nombre d'appels est trop élevé.
870	Nom ou variable système réservé
900	Erreurs 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 : <ul style="list-style-type: none"> • sto → • :=

Code d'erreur	Description
	<ul style="list-style-type: none"> • 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	<p>La fonction n'est pas définie. Utilisez l'une des commandes suivantes :</p> <ul style="list-style-type: none"> • Define • := • sto → <p>pour définir une fonction.</p>
1100	<p>Calcul non réel</p> <p>Par exemple, si le logiciel est réglé sur Réel, $\sqrt{(-1)}$ n'est pas valide.</p> <p>Pour autoriser les résultats complexes, réglez le mode "Réel ou Complex" sur "RECTANGULAIRE ou POLAIRE".</p>
1110	Limites invalides

Code d'erreur	Description
1120	Pas de changement de signe
1130	L'argument ne peut être ni une liste ni une matrice.
1140	Erreurs d'arguments 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	Erreurs d'arguments Les deux premiers arguments doivent être des expressions polynomiales du troisième argument. Si le troisième argument est omis, le logiciel tente de sélectionner une valeur par défaut.
1160	Nom de chemin de bibliothèque invalide Les noms de chemins doivent utiliser le format <code>xxx\yyy</code> , où : <ul style="list-style-type: none">• La partie <code>xxx</code> du nom peut contenir de 1 à 16 caractères, et• la partie <code>yyy</code>, si elle est utilisée, de 1 à 15 caractères. Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.
1170	Utilisation invalide de nom de chemin de bibliothèque <ul style="list-style-type: none">• Une valeur ne peut pas être assignée à un nom de chemin en utilisant la commande Define, <code>:=</code> ou <code>sto →</code>.• Un nom de chemin ne peut pas être déclaré comme variable Local ni être utilisé dans une définition de fonction ou de programme.
1180	Nom de variable de bibliothèque invalide. Assurez-vous que ce nom : <ul style="list-style-type: none">• 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 : <ul style="list-style-type: none">• Vérifiez que la bibliothèque se trouve dans le dossier Ma bibliothèque.• Rafraîchissez les bibliothèques. Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.
1200	Variable de bibliothèque introuvable :

Code d'erreur	Description
	<ul style="list-style-type: none"> 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. <p>Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.</p>
1210	<p>Nom de raccourci de bibliothèque invalide</p> <p>Assurez-vous que ce nom :</p> <ul style="list-style-type: none"> 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. <p>Pour plus d'informations à ce sujet, reportez-vous à la section Bibliothèques dans la documentation.</p>
1220	<p>Erreurs d'argument :</p> <p>Les fonctions tangentLine et normalLine prennent uniquement en charge les fonctions à valeurs réelles.</p>
1230	<p>Erreurs de domaine.</p> <p>Les opérateurs de conversion trigonométrique ne sont pas autorisés en mode Angle Degré ou Grade.</p>
1250	<p>Erreurs d'argument</p> <p>Utilisez un système d'équations linéaires.</p> <p>Exemple de système à deux équations linéaires avec des variables x et y :</p> $3x+7y=5$ $2y-5x=-1$
1260	<p>Erreurs d'argument :</p> <p>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.</p>
1270	<p>Erreurs d'argument</p> <p>La dérivée doit être une dérivée première ou seconde.</p>
1280	<p>Erreurs d'argument</p> <p>Utilisez un polynôme dans sa forme développée dans une seule variable.</p>

Code d'erreur	Description
1290	Erreur d'argument Utilisez un polynôme dans une seule variable.
1300	Erreur d'argument Les coefficients du polynôme doivent s'évaluer à des valeurs numériques.
1310	Erreur d'argument : Une fonction n'a pas pu être évaluée en un ou plusieurs de ses arguments.
1380	Erreur d'argument : Les appels imbriqués de la fonction 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, consultez **warnCodes()**, page 185.

Code d'avertissement	Message
10 000	L'opération peut donner des solutions fausses. Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 001	L'équation générée par dérivation peut être fausse.
10 002	Solution incertaine Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 003	Précision incertaine Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 004	L'opération peut omettre des solutions. Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 005	CSolve peut donner plus de zéros.
10 006	Solve peut donner plus de zéros. Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 007	D'autres solutions sont possibles. Essayez de spécifier des bornes inférieure et supérieure ou une condition initiale. Exemples utilisant la fonction solve() : <ul style="list-style-type: none">• <code>solve(Equation, Var=Guess) lowBound<Var<upBound</code>• <code>solve(Equation, Var) lowBound<Var<upBound</code>• <code>solve(Equation, Var=Guess)</code> Le cas échéant, essayez d'utiliser des méthodes graphiques pour vérifier les résultats.
10 008	Le domaine du résultat peut être plus petit que le domaine de l'entrée.
10 009	Le domaine du résultat peut être plus grand que le domaine de l'entrée.
10 012	Calcul non réel

Code d'avertissement	Message
10 013	∞^0 ou undef^0 remplacé par 1
10 014	undef^0 remplacés par 1
10 015	1^∞ ou 1^{undef} remplacé par 1
10 016	1^{undef} remplacés par 1
10 017	Capacité remplacée par ∞ ou $-\infty$
10 018	Requiert et retourne une valeur 64 bits.
10 019	Ressources insuffisantes, la simplification peut être incomplète.
10 020	L'argument de la fonction trigonométrique est trop grand pour une réduction fiable.
10 021	Les données saisies comportent un paramètre non défini. Le résultat peut ne pas être valide pour toutes les valeurs possibles du paramètre.
10 022	La spécification des bornes inférieure et supérieure peut donner une solution.
10 023	Le scalaire a été multiplié par la matrice unité.
10 024	Résultat obtenu par calcul approché.
10 025	L'équivalence ne peut pas être vérifiée en mode EXACT.
10 026	Il est possible que la contrainte soit ignorée. Spécifiez la contrainte sous la forme "\'Variable MathTestSymbol Constant' ou une combinaison de ces formats, par exemple 'x<3 and x>-12'

Informations générales

Aide en ligne

education.ti.com/eguide

Sélectionnez votre pays pour obtenir d'autres informations relatives aux produits.

Contacter l'assistance technique TI

education.ti.com/ti-cares

Sélectionnez votre pays pour obtenir une assistance technique ou d'autres types de support.

Informations Garantie et Assistance

education.ti.com/warranty

Sélectionnez votre pays pour en savoir plus sur la durée et les termes de la garantie et sur l'assistance pour le produit.

Garantie limitée. Cette garantie n'affecte pas vos droits statutaires.

Texas Instruments Incorporated

12500 TI Blvd.

Dallas, TX 75243

Index

	, opérateur "sachant que"
	215
	+
-	
-, soustraction[*]	194
	+, somme
	194
!	
!, factorielle	205
	/, division[*]
	196
"	
", secondes	213
	=
#	
#, indirection	211
#, opérateur dindirection	241
	≠, différent de[*]
	201
%	
%, pourcentage	200
	=, égal à
	201
&	
&, ajouter	205
*	
*	
*	
*, multiplication	195
,	
,	
, minutes	213
	Π, produit[*]
	208
	Σ
., soustraction élément par élément	199
.*, multiplication élément par élément	199
./, division élément par élément	199
.^, Puissance élément par élément	200
.+, addition élément par élément	198
:	
:=, assigner	217
	≥
^	
^-1, inverse	215
^, puissance	197
	≤
	≥, supérieur ou égal à
	203
	▶
	►, convertir mesure dangle en
	grades[Grad]
	75
	►approxFraction()
	13

►Base10, afficher comme entier décimal[Base10]	19	0h, indicateur hexadécimal	218
►Base16, convertir en nombre hexadécimal[Base16]	19	1	
►Base2, convertir en nombre binaire [Base2]	17	10^(), puissance de 10	215
►Cylind, afficher vecteur en coordonnées cylindriques [Cylind]		A	
►DD, afficher comme angle décimal [DD]	37	abs(), valeur absolue	7
►Decimal, afficher le résultat sous forme décimale[décimal] ..	38	affichage degrés/minutes/secondes, ►DMS	46
►DMS, afficher en degrés/minutes/secondes [DMS]	39	afficher comme angle décimal, ►DD	38
►Polar, afficher vecteur en coordonnées polaires[Polar]	124	afficher données, Disp	44, 150
►Rad, converti la mesure de l'angle en radians	133	afficher vecteur en coordonnées sphériques, ►Sphere ..	162
►Rect, afficher vecteur en coordonnées rectangulaires	136	afficher vecteur en coordonnées cylindriques, ►Cylind	37
►Sphere, afficher vecteur en coordonnées sphériques [Sphere]	162	afficher vecteur en coordonnées rectangulaires	136
⇒		afficher vecteur en coordonnées rectangulaires, ►Rect	136
⇒, implication logique[*]	204, 238	afficher vecteur en coordonnées sphériques, ►Sphere	162
→		afficher/donner dénominateur, getDenom()	66
→, stocker	217	informations sur les variables, getVarInfo()	70, 73
↔		nombre, getNum()	72
↔, équivalence logique[*]	205	ajouter, &	205
©		ajustement	
©, commentaire	218	degré 2, QuadReg	130
◦		degré 4, QuartReg	131
◦, degrés/minutes/secondes[*]	213	exponentiel, ExpReg	54
◦, degrés[*]	213	linéaire MedMed, MedMed	102
0		logarithmique, LnReg	93
0b, indicateur binaire	218	Logistic	96
		logistique, Logistic	97
		MultReg	106
		puissance, PowerReg	126
		régression linéaire, LinRegBx	85, 87
		régression linéaire, LinRegMx	86
		sinusoïdale, SinReg	160
		ajustement de degré 2, QuadReg	130
		ajustement de degré 3, CubicReg	35

ajustement exponentiel, ExpReg	54	indicateur, 0b	218
aléatoire	135	binomCdf()	20, 80-81
matrice, randMat()	134	binomPdf()	20
aléatoires		Boolean operators	
initialisation nombres, Germe ..	135	and	8
amortTbl(), tableau damortissement	8	boucle, Loop	99
and, Boolean operator	7, 16		
angle(), argument	9	C	
ANOVA, analyse unidirectionnelle de			
variance	9	caractère	
ANOVA2way, analyse de variance à		chaîne, char()	22
deux facteurs	10	code de caractère, ord()	122
Ans, dernière réponse	12	Cdf()	56
approché, approx()	13	ceiling(), entier suivant	21
approx(), approché	13	centralDiff()	21
approxRational()	13	chaîne	
arc cosinus, cos ⁻¹ ()	29	ajouter, &	205
arc sinus, sin ⁻¹ ()	159	chaîne de caractères, char() ...	22
arc tangente, tan ⁻¹ ()	171	code de caractère, ord()	122
arccos()	14	convertir chaîne en expression,	
arccosh()	14	expr()	54
arccot()	14	convertir expression en chaîne,	
arccoth()	14	string()	167
arccsc()	14	décalage, shift()	155
arccsch()	14	dimension, dim()	43
arcsec()	14	format, format()	59
arcsech()	14	formatage	59
arcsin()	14	gauche, left()	84
arctan()	15	indirection, #	211
argsh()	15	longueur	43
argth()	15	portion de chaîne, mid()	103
argument, angle()	9	utilisation, création de nom de	
arguments présents dans les		variable	241
fonctions TVM	180	chaîne de caractères, char()	22
arguments TVM	180	chaîne format, format()	59
arrondi, round()	147	chaînes	
augment(), augmenter/concaténer	15	dans la chaîne, inString	78
augmenter/concaténer, augment()	15	char(), chaîne de caractères	22
avec, 	215	ClearAZ	24
avgRC(), taux d'accroissement	15	ClrErr, effacer erreur	24
moyen		codes et messages d'avertissement	255
		codes et messages d'erreur	255
		colAugment	25
		colDim(), nombre de colonnes de la	
		matrice	25
bibliothèque		colNorm(), norme de la matrice	26
créer des raccourcis vers des		combinatoires, nCr()	110
objets	84	Commande Stop	167
binaire		commande Text	173
convertir, ►Base2	17		

B

bibliothèque	
créer des raccourcis vers des	
objets	84
binaire	
convertir, ►Base2	

Commande Wait	184	crossP(), produit vectoriel	34
commentaire, ©	218	csc ⁻¹ (), argument cosécante	35
complexe	26	csc(), cosécante	34
conjugué, conj()	26	csch ⁻¹ (), argument cosécante hyperbolique	35
comptage conditionnel déléments dans une liste, countif()	33	csch(), cosécante hyperbolique	35
comptage du nombre de jours entre deux dates, dbd()	38	CubicReg, ajustement de degré 3	35
compter les éléments dune liste, count()	32	cumulativeSum(), somme cumulée	37
conj(), conjugué complexe	26	cycle, Cycle	37
constructMat(), construire une matrice	26	Cycle, cycle	37
construire une matrice, constructMat()	26	D	
convertir	26	d(), dérivée première	206
4Grad	26	dans la chaîne, inString()	78
binaire, ►Base2	26	dbd(), nombre de jours entre deux dates	38
degrés/minutes/seconde, ►DMS	75	décalage, shift()	155
entier décimal, ►Base10	17	décimal	
hexadécimal, ►Base16	17	afficher angle, ►DD	38
convertir en	17	afficher entier, ►Base10	19
►Rad	46	Define	39
convertir liste en matrice, list►mat()	19	Define LibPriv	40
convertir matrice en liste, mat►list()	19	Define LibPub	41
coordonnée x rectangulaire, P►Rx()	100	Define, définir	39
coordonnée y rectangulaire, P►Ry()	122	définir, Define	39
copier la variable ou fonction, CopyVar	122	définition	
corrMat(), matrice de corrélation	133	fonction ou programme privé	40
cos ⁻¹ , arc cosinus	92	fonction ou programme public	41
cos(), cosinus	100	degrés, -	213
cosh ⁻¹ (), argument cosinus hyperbolique	122	degrés/minutes/seconde	213
cosh(), cosinus hyperbolique	27	deltaList()	41
cosinus, cos()	27	DelVar, suppression variable	42
cot ⁻¹ (), argument cotangente	29	delVoid(), supprimer les éléments vides	42
cot(), cotangente	27	densité de probabilité pour la loi normale, normPdf()	116
cotangente, cot()	30	densité de probabilité pour la loi Student-t, tPdf()	175
coth ⁻¹ (), arc cotangente hyperbolique	29	dérivée	
coth(), cotangente hyperbolique	31	dérivée numérique, nDeriv() ... 112-113	
count(), compter les éléments dune liste	31	dérivée première, d()	206
countif(), comptage conditionnel déléments dans une liste	32	dérivée première	
cPolyRoots()	32	modèle	5
	33	dérivée seconde	6
	33	dérivées	
	33	dérivée numérique, nDerivative()	111

dessiner, construire	224-226	EndWhile	186		
det(), déterminant de matrice	42	entier suivant, ceiling()	21, 33		
déverrouillage des variables et des groupes de variables	183	EOS (Equation Operating System) ..	240		
diag(), matrice diagonale	43	Equation Operating System (EOS) ..	240		
différent de, ≠	201	équivalence logique, \Leftrightarrow	205		
dim(), dimension	43	erreurs et dépannage			
dimension, dim()	43	effacer erreur, ClrErr	24		
Disp, afficher données	44, 150	passer erreur, PassErr	123		
DispAt	45	étiquette, Lbl	83		
division, /	196	euler(), Euler function	50		
dotP(), produit scalaire	47	évaluation, ordre d	240		
droite, right()	79, 143	évaluer le polynôme, polyEval()	125		
E					
e élevé à une puissance, e^()	47, 53	exclusion avec l'opérateur « »	215		
E, exposant	211	Exit	53		
e^(), e élevé à une puissance	47	exp(), e élevé à une puissance	53		
écart-type, stdDev()	165-166, 183	exposant			
échantillon aléatoire	135	modèle	1		
eff), conversion du taux nominal au taux effectif	48	exposant e			
effacer		modèle	2		
erreur, ClrErr	24	exposant, E	211		
égal à, =	201	expr(), convertir chaîne en expression	54		
eigVc(), vecteur propre	48	ExpReg, ajustement exponentiel	54		
eigVi(), valeur propre	49	expression			
élément par élément		convertir chaîne en expression,			
addition, .+	198	expr()	54		
division, .P	199	F			
multiplication, .*	199	F-Test sur 2 échantillons	62		
puissance, .^	200	factor(), factoriser	56		
soustraction, .N	199	factorielle, !	205		
élément vide, tester	83	factorisation QR, QR	129		
éléments vides	236	factoriser, factor()	56		
éléments vides, supprimer	42	Fill, remplir matrice	57		
else, Else	75	fin			
Elself	49	EndFor	59		
end		FiveNumSummary	57		
EndLoop	99	floor(), partie entière	58		
fonction, EndFunc	63	fonction			
if, EndIf	75	définie par l'utilisateur	39		
while, EndWhile	186	fractionnaire, fpart()	60		
end function, EndFunc	63	Func	63		
end while, EndWhile	186	Fonction de répartition de la loi de			
EndIf	75	Student-t, tCdf()	173		
EndLoop	99	fonction définie par morceaux (2			
EndTry, end try	176	morceaux)			
		modèle	2		
		fonction définie par morceaux (n			

morceaux)		
modèle	3	
fonction financière, tvmFV()	179	gcd(), plus grand commun diviseur
fonction financière, tvml()	179	geomCdf()
fonction financière, tvmN()	179	geomPdf()
fonction financière, tvmPmt()	180	Get
fonction financière, tvmPV()	180	getDenom(), afficher/donner dénominateur
fonctions de distribution		getLangInfo(), afficher/donner les informations sur la langue
binomCdf()	20, 80-81	getLockInfo(), teste l'état de verrouillage d'une variable
binomPdf()	20	ou d'un groupe de variables
invNorm()	81	getMode(), réglage des modes
invt()	81	getNum(), afficher/donner nombre
Invχ²()	80	GetStr
normCdf()	115	getType(), get type of variable
normPdf()	116	getVarInfo(), afficher/donner les informations sur les variables
poissCdf()	124	Goto
poissPdf()	124	grades, G
tCdf()	173	groupes, tester l'état de verrouillage
tPdf()	175	groupes, verrouillage et déverrouillage
χ²way()	22	95, 183
χ²Cdf()	23	
χ²GOF()	23	
χ²Pdf()	24	
fonctions définies par l'utilisateur ...	39	
fonctions et programmes définis par l'utilisateur	40-41	H
fonctions et variables		
copie	27	hexadécimal
For	59	convertir, ►Base16
format(), chaîne format	59	indicateur, 0h
forme échelonnée (réduite de Gauss), ref()	137	hyperbolique
forme échelonnée réduite par lignes (réduite de Gauss-Jordan), rref()	148	argument cosinus, cosh⁻¹()
fpart(), partie fractionnaire	60	argument sinus, sinh⁻¹()
fraction		argument tangente, tanh⁻¹()
FracProp	128	cosinus, cosh()
modèle	1	sinus, sinh()
fraction propre, propFrac	128	tangente, tanh()
freqTable()	61	I
frequency()	61	
Func	63	identity(), matrice unité
Func, fonction	63	If
		iffFn()
G		imag(), partie imaginaire
G, grades	212	implication logique, ⇒
gauche, left()	84	204, 238
		indirection, #
		inférieur ou égal à, {
		inString(), dans la chaîne
		int(), partie entière

intDiv(), quotient (division euclidienne)	79	maximum, max()	100	
intégrale définie modèle	6	minimum, min()	104	
intégrale, \int	207	nouvelle, newList()	112	
interpolate(), interpoler	79	portion de chaîne, mid()	103	
inverse, \wedge^{-1}	215	produit scalaire, dotP()	47	
invF()	80	produit vectoriel, crossP()	34	
invNorm((fractiles de la loi normale)	81	produit, product()	128	
invNorm(), inverse fonction de répartition loi normale	81	somme cumulée, cumulativeSum()	37	
invt()	81	somme, sum()	168	
Invx ² ()	81	tri croissant, SortA	162	
iPart(), troncature	80	tri décroissant, SortD	162	
irr(), taux interne de rentabilité taux interne de rentabilité, irr()	82	liste, comptage conditionnel		
isPrime(), test de nombre premier ..	82	déléments dans	33	
isVoid(), tester l'élément vide	83	liste, compter les éléments	32	
L				
langue afficher les informations sur la langue	70	ln(), logarithme népérien	92	
Lbl, étiquette	83	Logistic, régression logistique	96	
lcm, plus petit commun multiple	83	LogisticD, régression logistique	97	
left(), gauche	84	longueur dune chaîne	43	
LibPriv	40	Loop, boucle	99	
LibPub	41	LU, décomposition LU dune matrice	99	
libShortcut(), créer des raccourcis vers des objets de bibliothèque	84	M		
LinRegBx, régression linéaire	85	matList(), convertir matrice en liste	100	
LinRegMx, régression linéaire	86	matrice		
LinRegtIntervals, régression linéaire	87	addition élément par élément, .+	198	
LinRegtTest	89	augmenter(concaténer, augment()	15	
linSolve()	91	convertir liste en matrice, listMat()	92	
listMat(), convertir liste en matrice	92	convertir matrice en liste, matList()	100	
liste augmenter(concaténer, augment()	15	décomposition LU, LU	99	
convertir liste en matrice, listMat()	92	déterminant, det()	42	
convertir matrice en liste, matList()	100	diagonale, diag()	43	
des différences, @list()	91	dimension, dim()	43	
différences dans une liste, @list()	91	division élément par élément, .P	199	
éléments vides	236	factorisation QR, QR	129	
		maximum, max()	100	

minimum, min()	104	ligne, rowNorm()	
multiplication élément par élément, .*	199	unité, identity()	75
multiplication et addition sur ligne de matrice, mRowAdd()	106	max(), maximum	100
nombre de colonnes, colDim()	25	maximum, max()	100
norme (colonnes), colNorm()	26	mean(), moyenne	101
nouvelle, newMat()	112	median(), médiane	101
opération sur ligne de matrice, mRow()	106	médiane, median()	101
produit, product()	128	MedMed, régression linéaire	
Puissance élément par élément, ^	200	MedMed	102
remplir, Fill	57	mid(), portion de chaîne	103
somme cumulée, cumulativeSum()	37	min(), minimum	104
somme, sum()	168	minimum, min()	104
sous-matrice, subMat()	167, 169	minutes,	213
sauf élément par élément, .N	199	mirr(), Taux interne de rentabilité modifié	104
transposée, T	170	mod(), modulo	105
valeur propre, eigVl()	49	modèle	
vecteur propre, eigVc()	48	dérivée première	5
matrice (1×2)		dérivée seconde	6
modèle	4	e exposant	2
matrice (2×1)		exposant	1
modèle	4	fonction définie par morceaux (2 morceaux)	2
matrice (2×2)		fonction définie par morceaux (n morceaux)	3
modèle	4	fraction	1
matrice ($m \times n$)		intégrale définie	6
modèle	4	logarithme	2
matrice de corrélation, corrMat()	27	matrice (1×2)	4
matrice unité, identity()	75	matrice (2×1)	4
matrices		matrice (2×2)	4
ajout ligne, rowAdd()	147	matrice ($m \times n$)	4
aléatoire, randMat()	134	produit (P)	5
échange de deux lignes, rowSwap()	148	racine carrée	1
forme échelonnée (réduite de Gauss), ref()	137	racine n-ième	2
forme échelonnée réduite par lignes (réduite de Gauss-Jordan), rref()	148	somme (G)	5
nombre de lignes, rowDim()	148	système de 2 équations	3
norme (Maximum des sommes des valeurs absolues des termes ligne par ligne, *	148	système de n équations	3
mRow(), opération sur ligne de matrice		Valeur absolue	3-4
mRowAdd(), multiplication et addition sur ligne de matrice		modes	
mRowAdd(), multiplication et addition sur ligne de matrice		définition, setMode()	153
multiplication, *		modulo, mod()	105
norme		moyenne, mean()	101
opération sur ligne de matrice		mRow(), opération sur ligne de matrice	106
mRowAdd(), multiplication et addition sur ligne de matrice		mRow(), opération sur ligne de matrice	106
multiplication, *		mRowAdd(), multiplication et addition sur ligne de matrice	106
norme		multiplication, *	195

MultReg	106	opérateur "sachant que" « »	215
MultRegIntervals()	107	opérateur "sachant que", ordre	
MultRegTests()	108	dévaluation	240
N			
nand, opérateur booléen	109	Opérateurs booléens ⇒	204
nCr(), combinaisons	110	↔	205
nDerivative(), dérivée numérique ..	111	nand	109
négation, saisie de nombres négatifs	241	nor	114
newList(), nouvelle liste	112	not	116
newMat(), nouvelle matrice	112	or	120
nfMax(), maximum de fonction		p	238
numérique	112	xor	187
nfMin(), minimum de fonction		or (booléen), or	120
numérique	113	or, opérateur booléen	120
nInt(), intégrale numérique	113	ord(), code numérique de caractère	122
nom), conversion du taux effectif au		 P	
taux nominal	114		
nombre aléatoire, randNorm()	135	P►Rx(), coordonnée x rectangulaire	122
nombre de jours entre deux dates,		P►Ry(), coordonnée y rectangulaire	122
dbd()	38	partie entière, floor()	58
nombre de permutations, nPr()	116	partie entière, int()	78
nor, opérateur booléen	114	partie imaginaire, imag()	77
norm(), norme de Frobenius	115	passer erreur, PassErr	123
normCdf()	115	PassErr, passer erreur	123
norme de Frobenius, norm()	115	Pdf()	60
normPdf()	116	permutation circulaire, rotate()	145
not, opérateur booléen	116	piecewise()	123
nouvelle		plus grand commun diviseur, gcd() ..	64
liste, newList()	112	plus petit commun multiple, lcm() ..	83
matrice, newMat()	112	poissCdf()	124
nPr(), nombre de permutations	116	poissPdf()	124
npv(), valeur actuelle nette	117	polaire	
nSolve(), solution numérique	118	coordonnée polaire, R►Pr()	133
numérique		coordonnée polaire, R►Pθ()	132
dérivée, nDeriv()	112-113	polar	
dérivée, nDerivative()	111	afficher vecteur, vecteur en	
intégrale, nInt()	113	coordonnées 4Polar ...	124
solution, nSolve()	118	polyEval(), évaluer le polynôme	125
O			
objet		polynôme	
créer des raccourcis vers la		évaluer, polyEval()	125
bibliothèque	84	polynôme, randPoly()	135
OneVar, statistiques à une variable ..	119	polynômes	
opérateur		aléatoire, randPoly()	135
ordre dévaluation	240	PolyRoots()	125
portion de chaîne, mid()		portion de chaîne, mid()	103
pourcentage, %		pourcentage, %	200
PowerReg, puissance		PowerReg, puissance	126

Prgm, définir programme	127	rand(), nombre aléatoire	133
probabilité de loi normale, normCdf()	115	randBin, nombre aléatoire	133
prodSeq()	127	randInt(), entier aléatoire	134
product(), produit	128	randMat(), matrice aléatoire	134
produit (P)		randNorm(), nombre aléatoire	135
modèle	5	randPoly(), polynôme aléatoire	135
produit vectoriel, crossP()	34	randSamp()	135
produit, P()	208	RandSeed, initialisation nombres	
produit, product()	128	aléatoires	135
programmation		réduite de Gauss-Jordan, rref()	148
afficher données, Disp	44, 150	réel, real()	136
définir programme, Prgm	127	ref(), forme échelonnée (réduite de	
passer erreur, PassErr	123	Gauss)	137
programmes		RefreshProbeVars	138
définition d'une bibliothèque		réglage des modes, getMode()	71
privée	40	réglages, mode actuel	71
définition d'une bibliothèque		régression	
publique	41	degré 3, CubicReg	35
programmes et programmation		puissance, PowerReg	125, 173
afficher écran E/S, Disp	44	régression de degré 4, QuartReg	131
afficher l'écran E/S, Disp	150	régression linéaire MedMed,	
effacer erreur, ClrErr	24	MedMed	102
try, Try	176	régression linéaire, LinRegBx	85, 87
propFrac, fraction propre	128	régression linéaire, LinRegMx	86
puissance de 10, 10^()	215	régression logarithmique, LnReg	93
puissance, ^	197	régression logistique, Logistic	96
puissance, PowerReg	125-126, 140-141, 173	régression logistique, LogisticD	97
		régression sinusoïdale, SinReg	160
		regressions	
		Regression puissance, PowerReg	140-141
		remain(), reste (division euclidienne)	139
		réponse (dernière), Ans	12
		Request	140
		RequestStr	141
		résolution simultanée d'équations,	
		simult()	157
		reste (division euclidienne), remain()	139
		résultat, statistiques	164
		Return	143
		Return, renvoi	143
		right(), droite	143
		right, right()	50, 185
		rk23(), fonction de Runge-Kutta	143
		rotate(), permutation circulaire	145
		round(), arrondi	147
		rowAdd(), ajout ligne de matrice	147
		rowDim(), nombre de lignes de la	
		matrice	148
		rowNorm(), norme de la matrice	148

(Maximum des sommes des valeurs absolues des termes ligne par ligne)	165
rowSwap(), échange de deux lignes de la matrice	148
S	
scalaire	
produit, dotP()	47
sec ⁻¹ (), arc sécante	149
sec(), secante	149
sech ⁻¹ (), argument sécante hyperbolique	150
sech(), sécante hyperbolique	150
secondes, "	213
seq(), suite	151
seqGen()	151
seqn()	152
sequence, seq()	151-152
set	
mode, setMode()	153
setMode(), définir mode	153
shift(), décalage	155
sign(), signe	156
signe, sign()	156
simult(), résolution simultanée déquations	157
sin ⁻¹ (), arc sinus	159
sin(), sinus	158
sinh ⁻¹ (), argument sinus hyperbolique	160
sinh(), sinus hyperbolique	159
SinReg, régression sinusoïdale	160
sinus, sin()	158
somme (G)	
modèle	5
somme cumulée, cumulativeSum()	37
somme des intérêts versés	209
somme du capital versé	210
somme, +	194
somme, sum()	168
somme, $\Sigma()$	208
SortA, tri croissant	162
SortD, tri décroissant	162
sous-matrice, subMat()	167, 169
soustraction, -	194
sqrt(), racine carrée	163
stat.results	164
stat.values	165
statistique	
combinaisons, nCr()	110
écart-type, stdDev()	165-166, 183
factorielle, !	205
médiane, median()	101
moyenne, mean()	101
nombre de permutations, nPr()	116
statistiques à deux variables,	
TwoVar	181
statistiques à une variable,	
OneVar	119
variance, variance()	184
statistiques	
initialisation nombres aléatoires, Germe	135
nombre aléatoire, randNorm()	135
statistiques à deux variables,	
TwoVar	181
statistiques à une variable, OneVar	119
stdDevPop(), écart-type de population	165
stdDevSamp(), écart-type déchantillon	166
stockage	
symbole, &	217
string(), convertir expression en chaîne	167
strings	
droite, right()	79, 143
permutation circulaire, rotate()	145
right, right()	50, 185
subMat(), sous-matrice	167, 169
substitution avec lopérateur « »	215
suite, seq()	151
sum(), somme	168
sumIf()	168
sumSeq()	169
supérieur à, >	203
supérieur ou égal à, 	203
suppression	
variable, DelVar	42
supprimer	
éléments vides dune liste	42
Supprimer	223
Syntaxe de	228-229
système de 2 équations	
modèle	3

système de n équations		tvmPV()	180
modèle	3	TwoVar, statistiques à deux variables	181
T			
t-test de régression linéaire multiple	108		
T, transposée	170	unitV(), vecteur unitaire	183
tableau damortissement, amortTbl()	7, 16	unLock, déverrouiller une variable ou un groupe de variables ..	183
tan ⁻¹ (), arc tangente	171		
tan(), tangente	170		
tangente, tan()	170		
tanh ⁻¹ (), argument tangente hyperbolique	172	Valeur absolue	
tanh(), tangente hyperbolique	171	modèle	3-4
taux d'accroissement moyen, avgRC()	15	valeur actuelle nette, npv()	117
taux effectif, eff()	48	valeur propre, eigVI()	49
Taux interne de rentabilité modifié, mirr()	104	valeur temporelle de l'argent, montant des versements ..	180
Taux nominal, nom()	114	valeur temporelle de l'argent, nombre de versements ..	179
tCdf(), fonction de répartition de loi de studentt	173	valeur temporelle de l'argent, taux dintérêt	179
test de nombre premier, isPrime()	82	valeur temporelle de l'argent, valeur acquise	179
test t, tTest	177	valeur temporelle de l'argent, valeur actuelle	180
Test_2S, F-Test sur 2 échantillons	62	valeurs de résultat, statistiques	165
tester l'élément vide, isVoid()	83	variable	
tInterval, intervalle de confiance t	174	locale, Local	94
tInterval_2Samp, intervalle de confiance t sur 2 échantillons	174	nom, création à partir d'une chaîne de caractères ..	241
tPdf(), densité de probabilité pour la loi Studentt	175	suppression, DelVar	42
trace()	176	supprimer toutes les variables à une lettre	24
transposée, T	170	variable locale, Local	94
tri		variables et fonctions	
croissant, SortA	162	copie	27
décroissant, SortD	162	variables, verrouillage et déverrouillage	71, 95, 183
troncature, iPart()	81	variance, variance()	184
Try, commande de gestion des erreurs	176	varPop()	183
try, Try	176	varSamp(), variance déchantillon ..	184
Try, try	176	vecteur	
tTest, test t	177	afficher vecteur en coordonnées cylindriques, ►Cylind ..	37
tTest_2Samp, test t sur deux échantillons	178	produit scalaire, dotP()	47
tvmFV()	179	produit vectoriel, crossP()	34
tvmI()	179	unitaire, unitV()	183
tvmN()	179	vecteur propre, eigVc()	48
tvmPmt()	180		

vecteur unitaire, unitV()	183
verrouillage des variables et des groupes de variables	95

W

warnCodes(), Warning codes	185
when(), when	186
when, when()	186
while, While	186
While, while	186

X

χ^2 , carré	198
XNOR	205
xor, exclusif booléen or	187

Z

zInterval, intervalle de confiance z ..	188
zInterval_1Prop, intervalle de confiance z pour une proportion	188
zInterval_2Prop, intervalle de confiance z pour deux proportions	189
zInterval_2Samp, intervalle de confiance z sur 2 échantillons	190
zTest	190
zTest_1Prop, test z pour une proportion	191
zTest_2Prop, test z pour deux proportions	192
zTest_2Samp, test z sur deux échantillons	193

Δ

Δlist(), liste des différences	91
---------------------------------------	----

X

χ^2 2way	22
χ^2 Cdf()	23
χ^2 GOF	23
χ^2 Pdf()	24