Texas Instruments

TI-nspire™

# L'unité nomade de TI-Nspire™ CAS+

## *Informations importantes*

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

© 2006 Texas Instruments Incorporated

Microsoft®, Windows®, Vernier® EasyTemp, Go!™ Temp et Go!™ Motion sont des marques commerciales de leur propriétaire respectif.

# Table des matières

**Introduction ..... 1**

L'unité nomade de TI-Nspire™ CAS+ ..... 1
Utilisation du manuel ..... 1
Où trouver des informations complémentaires ..... 2

**Présentation de labo de maths TI-Nspire™ ..... 3**

Le modèle de classeur du labo de maths TI-Nspire™ ..... 3
Les composants du labo de maths TI-Nspire™ ..... 4
L'unité nomade de TI-Nspire™ CAS+ ..... 4
Le logiciel pour ordinateur du labo de maths TI-Nspire™ ..... 5
Des applications de base ..... 6
Graphiques & géométrie ..... 6
Tableur & listes ..... 6
Calculs ..... 7
L'éditeur mathématique ..... 8
L'application Acquisition de données ..... 9
Le logiciel d'interface avec l'ordinateur pour TI-Nspire™ ..... 9
Les accessoires ..... 11
La station de connexion TI-Nspire™ ..... 11
Le chargeur de station TI-Nspire™ ..... 11
La tablette de rétroprojection ViewScreen pour TI-Nspire™ ..... 11
Les capteurs se connectant à un port USB ..... 11

**Premiers contacts ..... 13**

Mise en place des piles ..... 13
Pour mettre en place les piles ..... 13
Première mise sous tension de votre unité nomade de TI-Nspire™ CAS+ ..... 13
Réglage du contraste ..... 14
Mise hors tension de l'unité nomade ..... 15
Touches de l'unité nomade de TI-Nspire™ CAS+ ..... 15
Déplacement sur l'écran de l'unité nomade ..... 17
Déplacement du curseur avec le bouton de contrôle de déplacement du curseur ..... 18
Utilisation de la touche de clic ..... 18
Sélection de texte avec la touche de clic ..... 18
Déplacement des éléments avec la touche de sélection ..... 18
Déplacement du curseur avec les touches fléchées ..... 18
Utilisation de la touche Menu ..... 18
Utilisation de la touche Echap ..... 20
Raccourcis clavier ..... 21

Écran de l'unité nomade de TI-Nspire™ CAS+ ..... 22
Utilisation de la barre d'outils et des menus ..... 24
Affichage des menus d'outils ..... 24
Options du menu Classeur de la barre d'outils ..... 24
Options de menu de la barre d'outils de l'application ..... 27
Barre d'outils État du système (System Status) ..... 27
Utilisation du navigateur de classeurs ..... 29
Options des menus du navigateur de classeurs ..... 29
Ouverture du navigateur de classeurs ..... 32
Développement du contenu des dossiers ..... 33
Masquage du contenu des dossiers ..... 34
Création de dossier ..... 35
Conventions de dénomination des fichiers ..... 36
Réglages du système ..... 36
Modification des réglages du système ..... 36
Utilisation du Catalogue (Catalog) pour accéder aux commandes ..... 37
Ouverture du Catalogue ..... 37
Saisie de caractères spéciaux ..... 38
Saisie de caractères linguistiques internationaux ..... 39

**Utilisation des classeurs ..... 41**

Création d'un nouveau classeur ..... 41
Changement de mise en page ..... 42
Ajout d'une activité à un classeur ..... 43
Ajout d'une page à une activité ..... 44
Consultation des différentes pages d'un classeur ..... 44
Utilisation de la trieuse de page ..... 44
Ouverture de la trieuse de pages ..... 45
Options de la barre d'outils de la trieuse de pages ..... 45
Affichage de toutes les pages d'un classeur ..... 47
Réorganisation des pages dans un classeur ..... 48
Modification des réglages du classeur ..... 49

**Annexe : Instructions et fonctions ..... 51**

Modèles d'expression ..... 51
Liste alphabétique des opérations ..... 54
Symboles ..... 152

**Annexe : Informations générales ..... 169**

Informations sur les services et la garantie TI ..... 169
Précautions à prendre lors de l'utilisation des piles ..... 169

**Index ..... 171**

# Introduction

## *L'unité nomade de TI-Nspire™ CAS+*

Ce manuel contient des informations relatives à un puissant outil d'apprentissage conçu comme un labo de maths avancé, développé par Texas Instruments : l'unité nomade de TI-Nspire™ CAS+.

Cette unité nomade est livrée avec une grande variété de logiciels préinstallés dont les fonctions sont particulièrement utiles dans différentes matières et cursus scolaires.

Vous pouvez encore étendre les capacités de votre unité nomade de TI-Nspire™ CAS+ en l'utilisant conjointement aux accessoires, comme versions logiciels pour ordinateur du labo de maths TI-Nspire™, la tablette de rétroprojection ViewScreen™ pour TI-Nspire™, la station de connexion TI-Nspire™ et le logiciel d'interface avec l'ordinateur pour TI-Nspire™.

## *Utilisation du manuel*

Ce manuel d'utilisation est conçu pour fournir des instructions relatives à l'utilisation de base d'unité nomade de TI-Nspire™ CAS+.

Il contient les chapitres suivants :

**Présentation** : fournit un aperçu détaillé des composants et accessoire de labo de maths TI-Nspire™.

**Premiers contacts** : contient des informations de base pour se familiariser avec les fonctions de l'unité nomade et offre aux élèves et aux enseignants un aperçu de l'utilisation d'unité nomade de TI-Nspire™ CAS+.

**Utilisation des classeurs** : fournit des instructions concernant la création et l'utilisation des classeurs.

**Fonctions et instructions** : explique la syntaxe et l'utilisation de chaque fonction et instruction du système d'exploitation et fournit une liste alphabétique des opérations, messages d'erreur et autres informations de référence.

**Assistance et garantie** : fournit des informations relatives à l'assistance, à la garantie et des données permettant de contacter le support technique.

## Où trouver des informations complémentaires

Les informations produit non abordées dans le présent manuel sont fournies sous forme électronique. Le manuel d'utilisation complet d’unité nomade de TI-Nspire™ CAS+ est inclus sur le CD-ROM fourni avec votre unité nomade. Si besoin, ce manuel est accessible en ligne et peut être téléchargé gratuitement à l'adresse education.ti.com/guides.

# Présentation de labo de maths TI-Nspire™

Le labo de maths TI-Nspire™ propose des outils conçus pour permettre aux enseignants de dispenser à leurs élèves un apprentissage dynamique et intéractif. De l'algèbre élémentaire au calcul différentiel ou intégral, TI-Nspire™ donne la possibilité aux élèves de visualiser et d'étudier plusieurs représentations d'un même problème. De cette façon, quelles que soient les méthodes d'apprentissage des élèves, ceux-ci sont en mesure d'analyser et de résoudre efficacement les problèmes.

Grâce à TI-Nspire™, les élèvent peuvent créer et enregistrer leur travail en format électronique, pour le partager afin d'expliquer et de communiquer leurs solutions tout en permettant aux enseignants d'évaluer leur progression.

## *Le modèle de classeur du labo de maths TI-Nspire™*

L'élément central de TI-Nspire™ est le classeur. Tous les travaux réalisés avec le labo de maths TI-Nspire™ sont enregistrés sous forme de classeur. Les classeurs TI-Nspire™ sont transmissibles et peuvent être partagés entre plusieurs utilisateurs d'unités nomades et utilisateurs du logiciel pour ordinateur.

Le modèle de classeur TI-Nspire™ est simple et facile à utiliser dès lors que l'on en comprend la structure :

- tous les travaux effectués sur TI-Nspire™ sont enregistrés dans un **classeur** ;
- chaque classeur comprend une ou plusieurs **activités** ;
- chaque activité peut contenir une ou plusieurs **pages** ;
- chaque page peut être divisée en quatre **espaces de travail** au maximum ;
- chaque espace de travail peut contenir **une application** quelconque de TI-Nspire™.

## *Les composants du labo de maths TI-Nspire™*

Le labo de maths TI-Nspire™ se compose des éléments suivants :

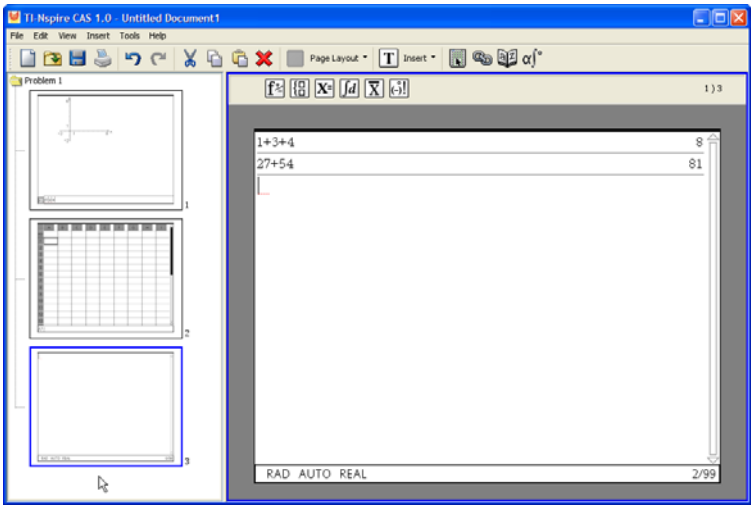
- une unité nomade pour l'élève ou l'enseignant : l'unité nomade de TI-Nspire™ CAS+ ;
- un logiciel pour ordinateur, alternative à l'unité nomade pour travailler en salle informatique et fair des évaluations en ligne : le TI-Nspire CAS ;
- une solution de projection pour un affichage sur grand écran : la tablette de rétroprojection ViewScreen pour TI-Nspire ;
- un logiciel d'interface avec l'ordinateur pour répondre aux besoins de connectivité des unités nomades.

## *L'unité nomade de TI-Nspire™ CAS+*

L'unité nomade de TI-Nspire™ CAS+ permet d'utiliser toutes les capacités du labo de maths TI-Nspire™ sur une plate-forme mobile. Cette unité nomade peut être transportée partout et utilisée instantanément en classe ou à la maison. Avant tout, elle se présente comme un outil de visualisation destiné à la modélisation mathématique. Elle intègre également des outils de visualisation conçus pour offrir aux élèves (comme aux enseignants) une interaction aussi directe que possible avec les représentations mathématiques fournies par le système.

## *Le logiciel pour ordinateur du labo de maths TI-Nspire™*

Le logiciel pour ordinateur du TI-Nspire CAS offre toutes les fonctionnalités de l'unité nomade de TI-Nspire™ CAS+, ce qui permet une utilisation en classe, via un ordinateur.

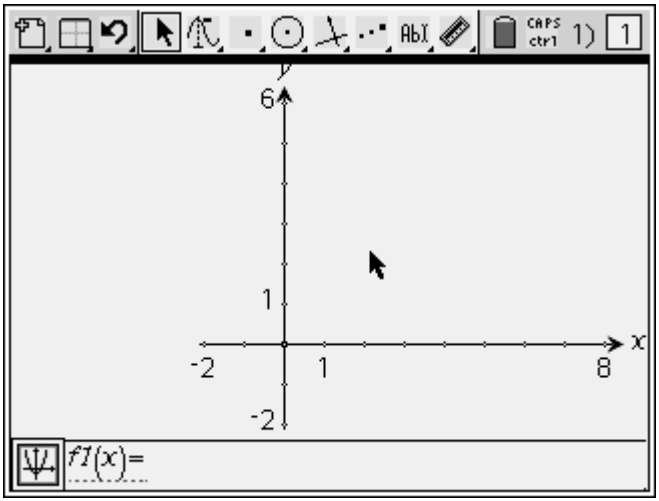


## *Des applications de base*

L'unité nomade et le logiciel pour ordinateur partagent une série d'applications de base. Ces applications constituent le coeur du labo de maths TI-Nspire™.

### Graphiques & géométrie

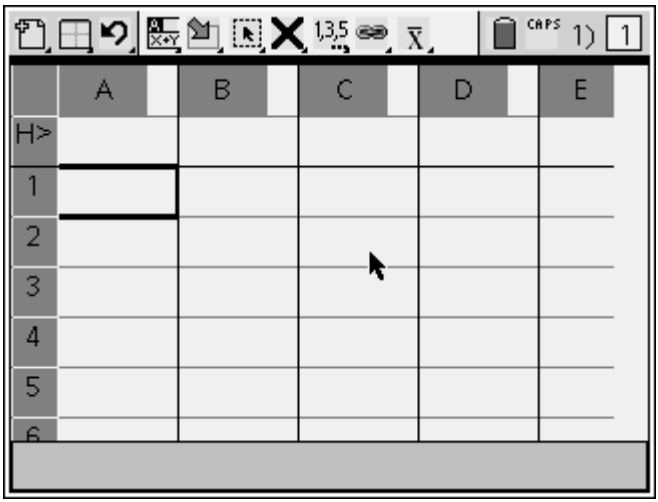
L'application Graphiques & géométrie permet de représenter et d'étudier graphiquement des fonctions, de créer et d'étudier des figures géometriques et d'animer des points sur des objets ou des représentations graphiques et pour en étudier le comportement. Vous pouvez utiliser Graphiques & géométrie pour représenter graphiquement des données collectées via la fonction Acquisition de données, étudier différentes transformations et créer ou analyser des intégrales et des dérivées.




### Tableur & listes

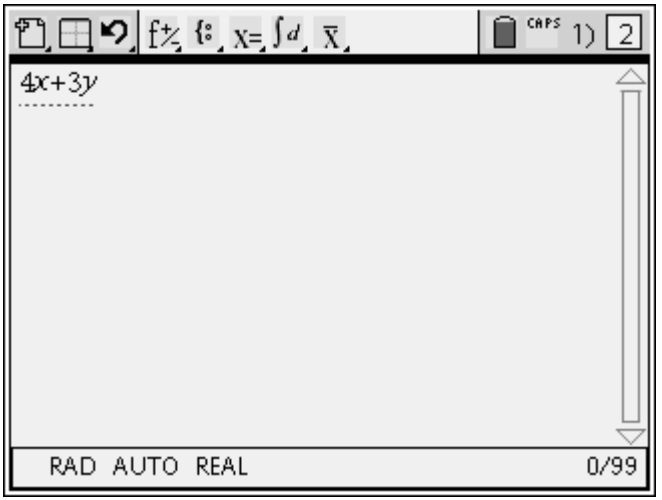
L'application Tableur & listes vous offre l'espace nécessaire pour travailler sur des données présentées sous forme de tableau. Vous pouvez utiliser Tableur & listes pour stocker des données numériques, du texte et des expressions mathématiques. Vous disposez également de la possibilité de définir une cellule du tableau en fonction du contenu d'autres cellules et même, une colonne entière en fonction d'une autre colonne. L'application Tableur & listes autorise la manipulation de variables créées

dans d'autres applications du labo de maths TI-Nspire™, y compris les tableaux de données physiques acquises à partir de capteurs, ainsi que la représentation graphique de ces données à l'aide de Graphiques & géométrie. Vous pouvez également utiliser l'application Tableur & listes pour effectuer des calculs statistiques ou générer des suites.



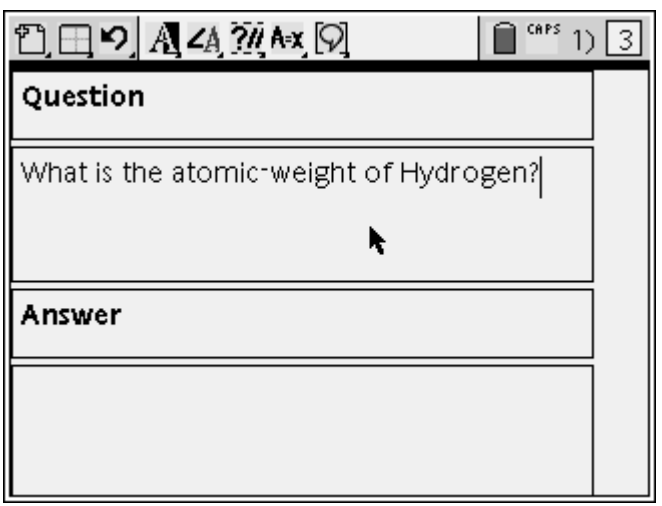

## Calculs

Cette application Calculs peut être utilisée pour effectuer des calculs, utiliser un historique et en faire défiler le contenu afin d'en copier des expressions et des résultats pour les récupérer et les insérer ailleurs. Elle permet également de définir des variables et des fonctions. Après avoir défini ou édité une variable ou une fonction, celle-ci est accessible depuis n'importe quelle autre application du labo de maths TI-Nspire™, telle que Graphiques & géométrie, utilisée dans la même activité.



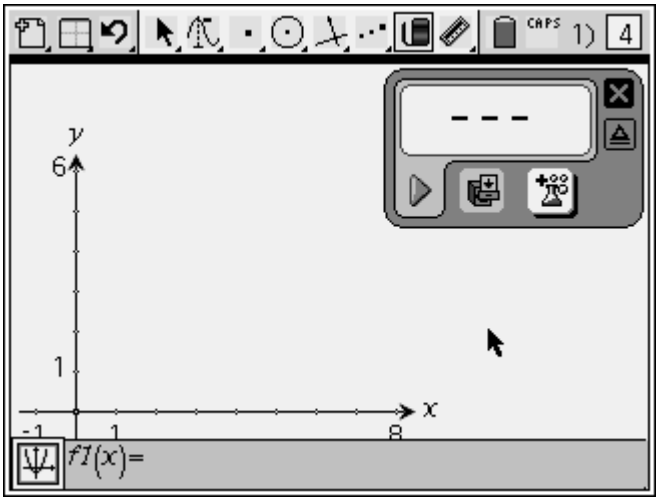

## L’éditeur mathématique

L'application Éditeur mathématique est un éditeur de texte qui peut être utilisé à des fins d'auto-évaluation ou d'interrogation par d'autres personnes. Vous pouvez utiliser cette application pour créer des documents d'aide à votre seule intention ou à partager avec d'autres utilisateurs. L’Éditeur mathématique permet aux utilisateurs d'avoir différents rôles dans les documents, en différenciant bien l'auteur et le relecteur.



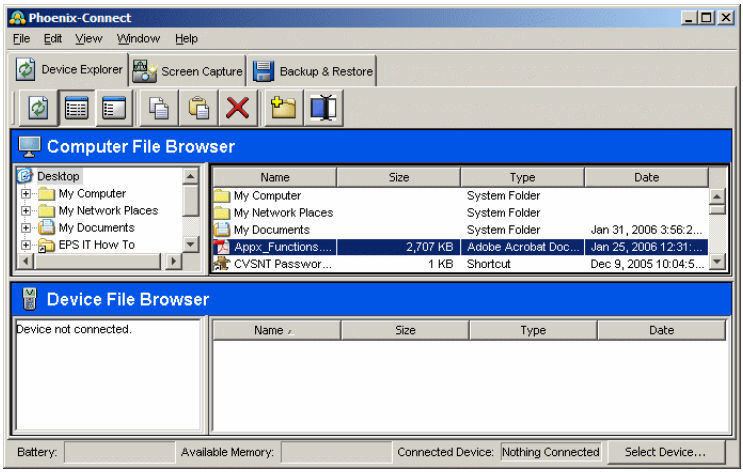
## L’application Acquisition de données

L’application Acquisition de données vous permet de recueillir des données expérimentales à l’aide d'une sonde ou d'un capteur et de les afficher automatiquement dans un tableau et/ou un graphique à des fins d'analyse. Pour cela, l'application Acquisition de données doit être utilisée conjointement aux applications Tableur & listes et Graphiques & géométrie.




## Le logiciel d’interface avec l’ordinateur pour TI-Nspire™

Le application logiciel d’interface avec l’ordinateur pour TI-Nspire™ vous permet de transférer des classeurs entre un ordinateur et une unité nomade de TI-Nspire™ CAS+. Vous pouvez capturer une image de l’écran d’une unité nomade et l’utiliser dans un classeur du logiciel TI-Nspire ™ CAS ou bien dans un fichier de votre ordinateur. Le logiciel d’interface avec l’ordinateur pour TI-Nspire™ vous offre la possibilité de sauvegarder les classeurs d'une unité nomade dans un fichier stocké sur votre ordinateur, puis de restaurer ultérieurement les classeurs. Vous pouvez également utiliser le logiciel d’interface avec l’ordinateur pour TI-Nspire™ pour mettre à jour le système d'exploitation (O.S) d'une unité nomade de TI-Nspire™ CAS+.


Phoenix-Connect
File Edit View Window Help
Device Explorer
Screen Capture
Backup & Restore
Computer File Browser
Desktop
My Computer
My Network Places
My Documents
EPS IT How To
Name
Size
Type
Date
My Computer
System Folder
My Network Places
System Folder
My Documents
System Folder
Jan 31, 2006 3:56:2...
Appx_Functions....
2,707 KB
Adobe Acrobat Doc...
Jan 25, 2006 12:31:...
CVSNT Passwor...
1 KB
Shortcut
Dec 9, 2005 10:04:5...
Device File Browser
Device not connected.
Name
Size
Type
Date
Battery:
Available Memory:
Connected Device:
Nothing Connected
Select Device...

## *Les accessoires*

### La station de connexion TI-Nspire™

La station de connexion TI-Nspire™ facilite l'utilisation de capteurs de sciences se connectant à un port USB.

### Le chargeur de station TI-Nspire™

Le chargeur de station TI-Nspire™ permet de charger jusqu'à quatre stations de connexion simultanément.

### La tablette de rétroprojection ViewScreen pour TI-Nspire™

La tablette de rétroprojection ViewScreen™ pour TI-Nspire™ permet de projeter l'écran d'une unité nomade dans les classes n'utilisant pas le logiciel pour ordinateur. La tablette de rétroprojection ViewScreen™ pour TI-Nspire™ peut être utilisée à la fois par les élèves et l'enseignant car elle ne nécessite pas l'utilisation d'un adaptateur spécial. Ses principales fonctions sont les suivantes :

- La tablette de rétroprojection ViewScreen™ pour TI-Nspire™ peut être connectée à toute unité nomade de TI-Nspire™ CAS+
- La tablette de rétroprojection ViewScreen™ pour TI-Nspire™ affiche l'écran courant de l'unité connectée et le rafraîchit en temps réel

### Les capteurs se connectant à un port USB

L'application Acquisition de données est compatible avec les systémes d'acquisitions de données suivants :

- Vernier® EasyTemp™
- CBR 2™
- Go!™Temp
- Go!™Motion

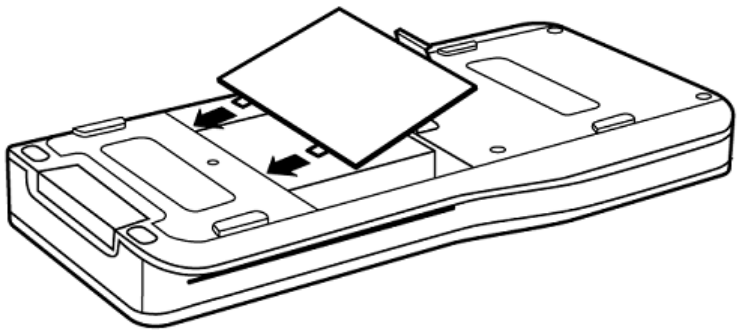
# Premiers contacts

## *Mise en place des piles*

L'unité nomade de TI-Nspire™ CAS+ utilise quatre piles AAA, lesquelles sont fournies avec le produit.

### Pour mettre en place les piles

1. Retirez le couvercle du compartiment des piles, situé au dos de l'unité nomade.
2. Après avoir retiré les quatre piles AAA de leur emballage, insérez-les, une par une, dans le compartiment des piles. Pour ce faire, respectez la polarité (+ et -) indiquée dans le compartiment des piles.



3. Remettez en place le couvercle du compartiment des piles de l'unité nomade. Appuyez sur le couvercle pour le fixer en position.

## *Première mise sous tension de votre unité nomade de TI-Nspire™ CAS+*

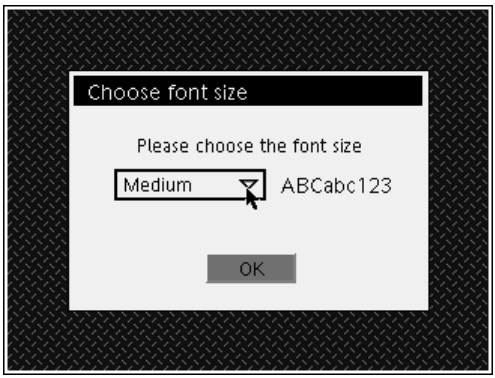
1. Après avoir installé les piles fournies avec l'unité nomade, appuyez sur (off on) pour l'allumer.

   Une barre de progression s'affiche pendant le chargement du système d'exploitation.

   Une fois le système d'exploitation chargé, vous êtes invité à sélectionner la langue à utiliser sur votre unité nomade.

2. Appuyez sur ▼ de façon à mettre la langue voulue en surbrillance, puis appuyez sur enter pour la sélectionner.

   Un message vous invite ensuite à sélectionner la taille de police à utiliser pour afficher le texte sur votre unité nomade.




3. Appuyez sur ▼ de façon à mettre en surbrillance la taille de police voulue, puis appuyez sur enter pour la sélectionner.

   **Remarque :** à tout moment, vous pouvez restaurer la taille de police initiale sur votre unité nomade en modifiant les réglages système.

   Après avoir sélectionné la taille de police, le classeur **Premiers contacts** s'affiche à l'écran.

## *Réglage du contraste*

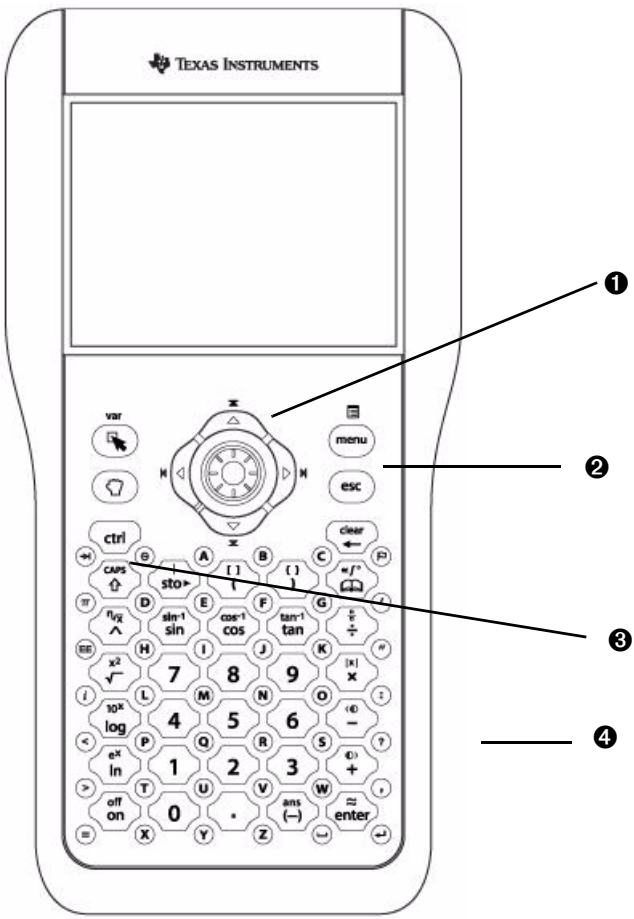
Pour éclaircir l'affichage, maintenez enfoncée la touche ctrl tout en appuyant à plusieurs reprises sur ☼+.

Pour assombrir l'affichage, maintenez enfoncée la touche ctrl tout en appuyant à plusieurs reprises sur ⊕ −.

## *Mise hors tension de l'unité nomade*

Appuyez sur ctrl off on. La prochaine fois que vous mettrez l'unité nomade sous tension, les réglages courants, ainsi que le contenu de la mémoire, seront conservés.

## *Touches de l'unité nomade de TI-Nspire™ CAS+*



❶ **Pavé de navigation** : bouton de contrôle du déplacement du curseur et touches fléchées ◀, ▶, ▲, ▼ permettant de déplacer le curseur sur l'écran.

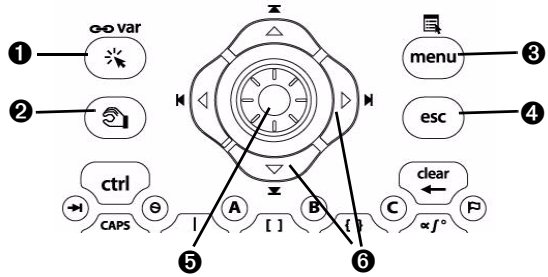
❷ **Touches de contrôle du curseur** ( ) : ces touches déterminent le contrôle du curseur par le bouton de contrôle du curseur ou les touches fléchées.

❸ **Touches modificatrices** ( ) : ces touches accroissent les fonctionnalités de votre unité nomade en augmentant le nombre de commandes de touche.

❹ **Pavé numérique** : permet d'exécuter des fonctions mathématiques et scientifiques. L'utilisation du clavier alphabétique simplifie la saisie de texte.

## *Déplacement sur l'écran de l'unité nomade*

Le pavé de navigation contrôle le déplacement du curseur sur l'écran de l'unité nomade. Les touches fléchées offrent une méthode alternative de déplacement et de navigation familière sur les écrans de l'unité. Le bouton de contrôle de déplacement du curseur offre des fonctions très similaires à celles de la souris d'un ordinateur. Vous pouvez permuter entre ces deux méthodes de déplacement du curseur et les combiner pour naviguer sur l'écran et manipuler les objets, suivant celle qui vous convient le mieux. Les touches situées autour du pavé de navigation déterminent si le bouton de contrôle du déplacement du curseur ou les touches fléchées sont activés. L'exemple ci-dessous fournit un aperçu plus détaillé de la zone du pavé de navigation. Vous trouverez après cet exemple une description de chacune des touches composant le pavé de navigation.




❶ **Touche de clic :** active le bouton de contrôle du déplacement du curseur.

❷ **Touche de sélection :** sélectionne les éléments affichés à l'écran.

❸ **Touche Menu :** active les touches fléchées pour l'affichage des menus dans la barre d'outils et la navigation dans les boîtes de dialogue.

❹ **Touche Echap :** ferme les menus de la barre d'outils, ainsi que les boîtes de dialogue affichés à l'écran.

❺ **Bouton de contrôle de déplacement du curseur :** ce bouton s'active par pression exercée sur les positions haute, basse, gauche et droite.

❻ **Touches fléchées (◀, ▶, ▲, ▼) :** contrôlent le déplacement du curseur à l'écran.

## Déplacement du curseur avec le bouton de contrôle de déplacement du curseur

Les touches (↖) et (☝) s'utilisent conjointement au bouton de contrôle de déplacement du curseur pour déplacer le curseur et sélectionner les éléments affichés sur l'écran.

### Utilisation de la touche de clic

1. Appuyez sur (↖) pour activer le bouton de contrôle de déplacement du curseur.
2. Utilisez le bouton de contrôle de déplacement du curseur pour déplacer le curseur sur le bureau.

   **Remarque :** exercer une pression sur la touche (↖) équivaut à cliquer avec la souris d'un ordinateur. Cette opération sélectionne l'élément vers lequel pointe le curseur.

### Sélection de texte avec la touche de clic

1. Utilisez le bouton de contrôle de déplacement du curseur pour positionner celui-ci au début du texte à sélectionner.
2. Appuyez sur la touche (↖) .
3. Utilisez le bouton de contrôle de déplacement du curseur ou appuyez sur ▶ pour déplacer le curseur à la fin du texte à sélectionner.

### Déplacement des éléments avec la touche de sélection

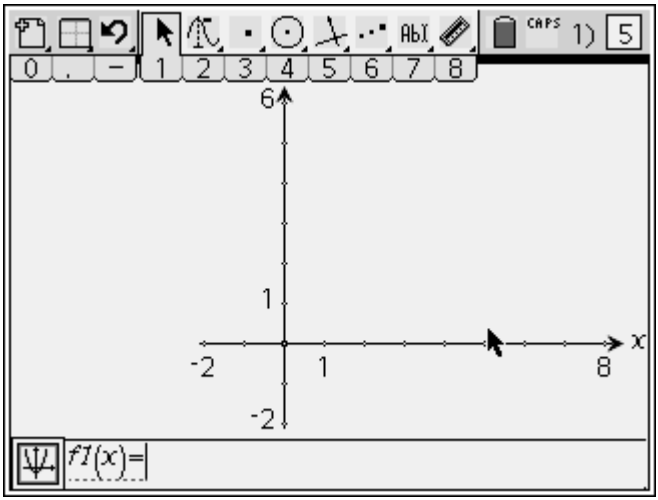
1. Utilisez le bouton de contrôle de déplacement du curseur pour pointer le curseur sur l'objet ou l'élément à déplacer.
2. Appuyez sur (☝) pour sélectionner l'objet. La forme du curseur se transforme en main.
3. Utilisez le bouton de contrôle de déplacement du curseur pour faire glisser l'élément sur l'écran.

## Déplacement du curseur avec les touches fléchées

Les touches (menu) et (esc) s'utilisent conjointement aux touches fléchées (◀, ▶, ▲, ▼) pour afficher les menus et se déplacer à l'intérieur de ces derniers. Ces touches permettent également de déplacer le curseur d'un champ à l'autre dans les boîtes de dialogue.

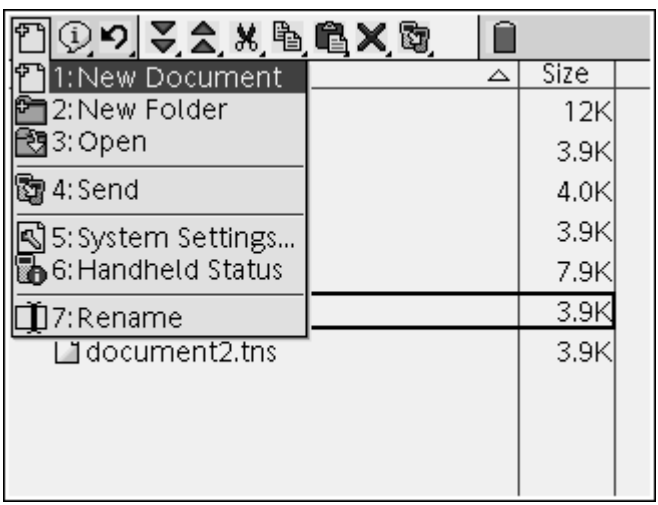
### Utilisation de la touche Menu

1. Appuyez sur (menu) pour afficher le menu des touches rapides.




2. Après vous être familiarisé avec les menus de la barre d'outils et lorsque vous savez quel menu afficher, appuyez sur la touche correspondant à la touche rapide pour afficher le menu associé (au-dessus de la touche).

   Par exemple, appuyez sur (0) pour afficher le menu Classeur ( ).




3. Si vous ne savez pas quel menu afficher et préférez afficher les différents menus avant d'effectuer votre choix, appuyez de nouveau sur (menu) .

Le dernier menu sélectionné s'affiche.

4. Vous pouvez alors appuyer de nouveau sur la touche (menu) ou utiliser les touches ◀ et ▶ pour afficher les menus adjacents de la barre d'outils.
5. Appuyez sur le numéro associé à une option de menu pour la sélectionner.

   Par exemple, appuyez sur (1) à partir du menu des outils système pour ouvrir un nouveau classeur.

## Utilisation de la touche Echap

1. Appuyez sur (esc) pour fermer les menus de la barre d'outils et les boîtes de dialogue affichées à l'écran.

## Touches modificatrices

(ctrl) Accroît les fonctions du clavier.

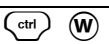
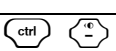
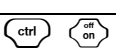
**Remarque :** les fonctions Ctrl sont imprimées au-dessus de la fonction principale de chaque touche, dans la même couleur que celle de la touche (ctrl).

(CAPS ⇧) Permet d'entrer la lettre suivante sur laquelle vous appuyez sous forme de majuscule.

## *Raccourcis clavier*

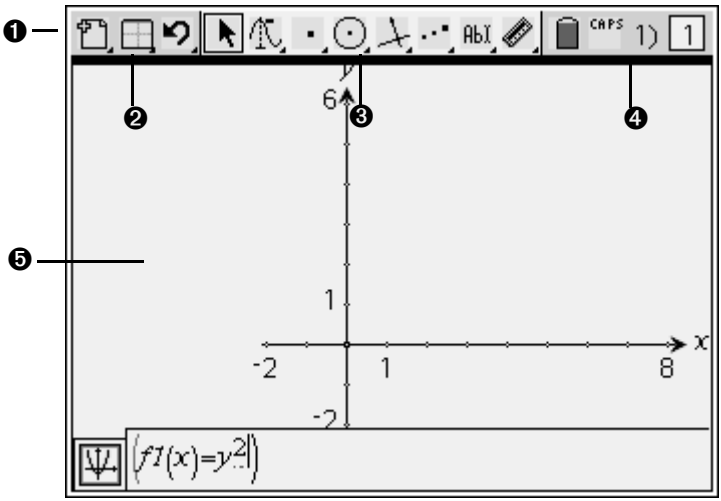
Utilisez les raccourcis clavier suivants pour exécuter des fonctions courantes.

| **Édition de texte** | |
|---|---|
| Couper | ctrl X |
| Copier | ctrl C |
| Coller | ctrl V |
| Annuler | ctrl Z |
| Rétablir | ctrl Y |
| **Insertion de caractères et de symboles** | |
| afficher un caractère ou la palette de modèles | ctrl |
| différent de | ctrl = |
| caractère de soulignement | ctrl ␣ |
| ≥ | ctrl > |
| ≤ | ctrl < |
| point d'exclamation (!) | ctrl ? |
| point-virgule (;) | ctrl : |
| valeur absolue \|x\| | ctrl |
| **Gestion des classeurs** | |
| lancer le navigateur de classeurs | ctrl O |
| créer un classeur | ctrl N |
| insérer une nouvelle page | ctrl I |
| imprimer le classeur courant | ctrl P |
| enregistrer le classeur courant | ctrl S |
| insérer une ligne ou une colonne | ctrl R |
| ∞ | ctrl i |

| **Navigation dans les classeurs** | |
|---|---|
| afficher la page précédente | |
| afficher la page suivante | |
| afficher la trieuse de pages | |
| afficher la dernière page de l'activité courante | |
| **Assistants et modèles** | |
| accéder à un assistant à partir du Catalogue |  |
| ajouter une colonne à une matrice | |
| modèle Intégrale | |
| modèle Dérivée | |
| **Modification de l'affichage** | |
| augmenter le contraste | |
| réduire le contraste |  |
| mise hors tension |  |

## *Écran de l'unité nomade de TI-Nspire™ CAS+*

L'écran de l'unité nomade est divisé en deux principales zones : la barre d'outils et l'espace de travail. L'exemple ci-dessous illustre l'écran de l'unité nomade et ses principales zones. Vous trouverez après cet exemple, une description détaillée de chacune de ces zones.




❶ **Barre d'outils** : contient des menus permettant d'effectuer des opérations sur le système et d'utiliser les classeurs.

❷ **Menus d'outils système** : permettent d'accéder aux outils système (création et enregistrement de classeurs et de pages, etc.).

❸ **Menus des applications** : les outils affichés dans cette zone varient en fonction de l'application active. À chaque application est associé un ensemble d'outils spécifique.

❹ **Icônes d'état** : indiquent l'état d'activation des touches modificatrices, l'état des piles et de la mémoire, le nombre total de pages et la page active.

❺ **Espace de travail des applications** : zone dans laquelle vous utilisez les applications.

## *Utilisation de la barre d'outils et des menus*

La barre d'outils et les menus de l'unité nomade de TI-Nspire™ CAS+ permettent d'accéder aux commandes disponibles à partir des classeurs et des applications. Les barres d'outils peuvent contenir des boutons, des menus ou une combinaison de ces deux éléments. La barre d'outils et les menus disponibles sont décrits ci-dessous.

### Affichage des menus d'outils

1. Appuyez sur (menu) pour afficher les touches rapides d'accès aux menus.

2. Après vous être familiarisé avec les menus de la barre d'outils et lorsque vous savez quel menu afficher, appuyez sur la touche correspondant à la touche rapide d'accès au menu pour afficher le menu associé (au-dessus de la touche).

   Par exemple, appuyez sur (0) pour afficher le menu Classeur ( ).

3. Pour afficher les différents menus avant de faire votre choix, appuyez de nouveau sur (menu) .

   Le dernier menu sélectionné s'affiche.

4. Vous pouvez alors appuyer de nouveau sur la touche (menu) ou utiliser les touches ◀ et ▶ pour afficher les menus adjacents de la barre d'outils.

5. Pour sélectionner une option de menu, appuyez sur le chiffre correspondant, assigné à cet option.

### Options du menu Classeur de la barre d'outils

Le tableau ci-dessous répertorie les options de la barre d'outils disponibles lorsque vous travaillez dans un classeur.

| Menu | Option | Action |
|---|---|---|
| | **Menu Classeur (Document)** | |

| Menu | Option | Action |
|---|---|---|
| | 1: Aller au navigateur de classeurs (Go to Document Browser) | Ferme le classeur courant et affiche le navigateur de classeurs. |
| | 2: Aller à la trieuse de pages (Go to Page Sorter) | Cet écran affiche une miniature représentant chaque page du classeur courant. Vous pouvez sélectionner une page et l'ouvrir. Vous pouvez également réorganiser les pages du classeur. |
| | 3: Nouveau classeur (New document) | Crée un classeur pour lequel la première activité et la première page sont affichées. |
| | 4: Enregistrer (Save) | Enregistre le classeur ouvert. |
| | 5: Enregistrer sous... (Save As...) | Enregistre le classeur ouvert. La boîte de dialogue Enregistrer (Save) s'affiche pour vous permettre de spécifier l'emplacement de stockage et le nom du classeur. |
| | 6: Réglages du classeur (Document Settings) | Affiche les réglages du classeur. |
| | 7: Insérer une activité (Insert Problem) | Ajoute une nouvelle activité au classeur courant. |
| | 8: Insérer une page (Insert Page) | Ajoute une nouvelle page à l'activité courante. |
| | 9: Var | Affiche les variables. |
| | **Menu Application** | |
| | 1: Ajouter Calculs (Add Calculator) | Insère l'application Calculs dans la page courante. |

| Menu | Option | Action |
|---|---|---|
| | 2: Ajouter Graphiques & géométrie (Add Graphs & Geometry) | Insère l'application Graphiques & géométrie dans la page courante. |
| | 3: Ajouter Tableur & listes (Add Lists & Spreadsheet) | Insère l'application Tableur & listes dans la page courante. |
| | 4: Ajouter Éditeur mathématique (Add Notes) | Insère l'application Éditeur mathématique dans la page courante. |
| | 5: Mise en page (Page Layout) | Utilisez la touche ▶ pour afficher les huit modèles prédéfinis de mise en page. |
| | 6: Catalogue (Catalog) | Affiche le Catalogue (Catalog). Utilisez le Catalogue pour insérer des fonctions et des opérations dans vos activités. |
| | 7: Jeu de symboles (Symbol Palette) | Affiche le jeu de symboles. Utilisez le jeu de symboles pour insérer des caractères spéciaux et des symboles dans vos activités. |
| | 8: Sélectionner l'application (Select Application) | Sélectionne l'application courante. |
| | 9: Changer d'application (Swap Applications) | Déplace l'application courante dans l'espace de travail occupé par l'application sélectionnée. |
| | **Menu Édition (Edit)** | |
| | 1: Annuler (Undo) | Annule le résultat de la dernière opération effectuée. |

| Menu | Option | Action |
|---|---|---|
| | 2: Rétablir (Redo) | Rétablit la dernière opération annulée par la commande Annuler (Undo). |
| | 3: Couper (Cut) | Supprime les informations sélectionnées de la page. |
| | 4: Copier (Copy) | Copie dans le Presse-papiers les informations sélectionnées dans la page courante. |
| | 5: Coller (Paste) | Insère les informations, précédemment coupées ou copiées, dans la page courante. |
| | 6: Supprimer (Delete) | Supprime l'application utilisée de la page courante. |
| | 7: Supprimer la page (Delete Page) | Supprime la page courante du classeur. |

## Options de menu de la barre d'outils de l'application

Chaque application est dotée d'une barre d'outils unique, contenant des menus spécifiques à cette application. Consultez le chapitre spécifique à chaque application pour une description de la barre d'outils associée.

## Barre d'outils État du système (System Status)

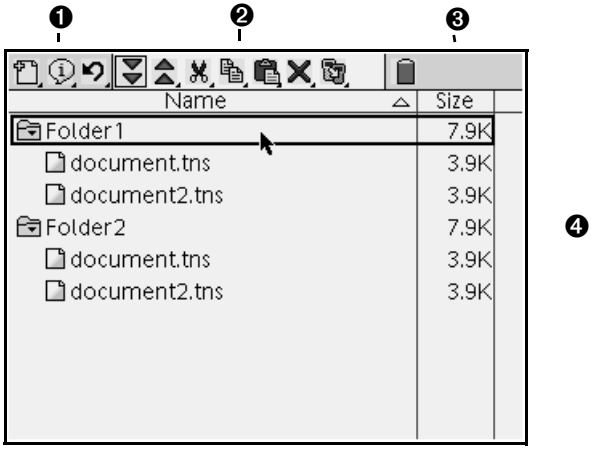
La barre d'outils État du système (System Status) fournit des informations relatives à l'état de fonctionnement de l'unité nomade. Les icônes affichées dans cette zone indiquent l'état d'activation des touches modificatrices et affiche le numéro de la page courante, ainsi que le nombre total de pages. Les indicateurs de piles déchargées et de faible quantité de mémoire disponible sont également affichés dans cette zone.

| Icône | Indicateur | Action |
|---|---|---|
| | Statut de l'unité (Device Status) | Indique si une ou plusieurs unités nomades sont connectées. |

| Icône | Indicateur | Action |
| --- | --- | --- |
| Critique<br> Bon<br> Faible | État des piles (Battery Status) | Indique l'état de charge des piles. |
| | Indicateur de mémoire (Memory Status) | Indique la quantité d'espace mémoire disponible sur l'unité nomade. |

## *Utilisation du navigateur de classeurs*

Utilisez le navigateur de classeurs comme un gestionnaire de fichiers pour stocker et organiser vos classeurs. Les classeurs doivent être stockés dans des dossiers. L'exemple ci-dessous illustre le navigateur de classeurs et ses principales zones. Vous trouverez après cet exemple, une description détaillée de chacune de ces zones.




❶ **Menus des outils Système** : permettent d'accéder aux outils système (création et enregistrement de classeurs et de pages, etc.)

❷ **Boutons de gestion des fichiers** : ces boutons permettent d'accéder rapidement aux outils de gestion des fichiers.

❸ **Icônes d'état** : indiquent l'état d'activation des touches modificatrices, le nombre total de pages et la page active.

❹ **Répertoire** : affiche le nom du dossier et les fichiers stockés sur l'unité nomade, en indiquant leur taille ainsi que la date et l'heure de leur dernier enregistrement.

### Options des menus du navigateur de classeurs

Le navigateur de classeur permet d'accéder à un ensemble d'outils unique qui vous permet d'effectuer des opérations sur les classeurs.

| Menu | Option | Action |
|---|---|---|
| | **Menu des outils Système** | |

| Menu | Option | Action |
|---|---|---|
| | 1: Nouveau classeur (New document) | Crée un nouveau classeur dans le dossier sélectionné. |
| | 2: Nouveau dossier (New Folder) | Crée un nouveau dossier. |
| | 3: Ouvrir (Open) | Ouvre le classeur ou le dossier sélectionné. |
| | 4: Envoyer (Send) | Ouvre une boîte de dialogue à partir de laquelle vous pouvez envoyer le classeur ou le dossier sélectionné à une autre unité nomade. |
| | 5: Réglages du système (System Settings) | Ouvre la boîte de dialogue Réglages du système (System Settings). |
| | 6: État de l'unité (Handheld Status) | Affiche l'état de la mémoire et des piles et indique si d'autres périphériques USB sont connectés. |
| | 7: Renommer (Rename) | Permet de renommer le classeur ou le dossier sélectionné. |
| | **Menu Informations (Information)** | |
| | 1: Jeu de symboles (Symbol Palette) | Ouvre le jeu de symboles. |
| | 2: Envoyer OS (Send OS) | Ouvre une boîte de dialogue à partir de laquelle vous pouvez envoyer l'O.S de votre unité nomade à une autre unité nomade. |
| | 3: À propos (About) | Fournit des informations sur la version courante du système d'exploitation. |

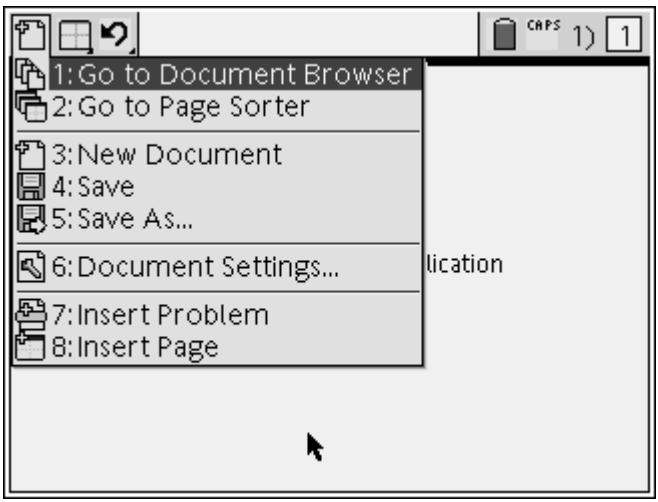
| Menu | Option | Action |
|---|---|---|
| | **Menu Édition (Edit)** | |
| | 1: Annuler (Undo) | Annule la dernière opération. |
| | 2: Rétablir (Redo) | Rétablit l'opération annulée par la commande Annuler (Undo). |
| | 3: Couper (Cut) | Supprime le classeur ou le dossier sélectionné. |
| | 4: Copier (Copy) | Copie le classeur ou le dossier sélectionné. |
| | 5: Coller (Paste) | Insère le classeur ou le dossier sélectionné. |
| | 6: Supprimer (Delete) | Supprime le classeur ou le dossier sélectionné. |
| | **Outils du navigateur de classeur** | |
| | Développer Tout (Expand All) | Affiche le contenu complet du dossier sélectionné. |
| | Réduire Tout (Collapse All) | Masque le contenu du dossier sélectionné. |
| | Couper (Cut) | Supprime le classeur ou le dossier sélectionné. |
| | Copier (Copy) | Copie le classeur ou le dossier sélectionné. |
| | Coller (Paste) | Insère le classeur ou le dossier sélectionné. |
| | Supprimer (Delete) | Supprime le classeur ou le dossier sélectionné. |
| | Envoyer (Envoyer) | Ouvre une boîte de dialogue à partir de laquelle vous pouvez envoyer le classeur ou le dossier sélectionné à une autre unité nomade. |

## Ouverture du navigateur de classeurs

Le navigateur de classeurs affiche tous les classeurs et tous les dossiers de classeurs présents sur l'unité nomade. Pour ouvrir le navigateur de classeurs :

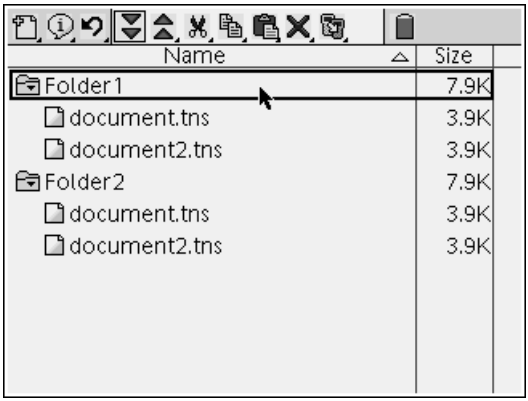
1. Sélectionnez > **1: Aller au navigateur de classeurs (Go to Document Browser)** dans la barre d'outils.

   Appuyez sur menu 0 1.




   Si vous avez apporté des modifications au classeur courant, un message vous invite à l'enregistrer.

2. La fenêtre du navigateur de classeurs s'ouvre et affiche tous les dossiers et fichiers présents sur l'unité nomade.




### *Ouverture des classeurs*

1. Appuyez sur les touches ▲ and ▼ pour mettre en surbrillance le dossier contenant le classeur à ouvrir.
2. Appuyez sur ▶ pour ouvrir le dossier.
3. Appuyez sur les touches ▲ et ▼ pour mettre en surbrillance le nom du classeur.
4. Appuyez sur ▶ pour ouvrir le classeur.

## Développement du contenu des dossiers

Pour développer le contenu d'un seul dossier :

1. Appuyez sur les touches ▲ et ▼ pour mettre en surbrillance le dossier voulu dans le navigateur de classeur.
2. Appuyez sur ▶ pour ouvrir le dossier.
3. L'icône se transforme en icône de dossier ouvert et la liste de tous les classeurs contenus dans le dossier s'affiche.

Pour développer le contenu de tous les dossiers :

- Sélectionnez l'icône **Développer Tout (Expand All)** dans la palette d'outils. La liste des classeurs s'affiche sous les noms de dossiers dans lesquels ils sont stockés.

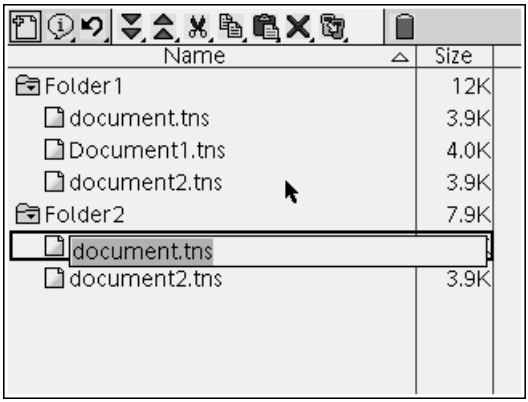
  Appuyez sur (menu) (1).

### *Changement de nom des dossiers ou des classeurs*

1. Appuyez sur les touches ▲ et ▼ pour mettre en surbrillance le classeur ou le dossier à renommer.

2. Sélectionnez > **7: Renommer (Rename)** dans la barre d'outils.

   Appuyez sur (menu) (0) (7).




   Le nom du classeur ou du dossier est mis en surbrillance.

3. Entrez le nouveau nom du dossier ou du classeur et appuyez sur (enter) pour valider le changement.

   **Remarque :** Appuyez sur (esc) pour annuler l'opération.

## Masquage du contenu des dossiers

Pour masquer le contenu d'un seul dossier :

- Appuyez sur ◀ pour masquer le contenu développé d'un dossier.

  L'icône se transforme en icône de dossier fermé et les classeurs stockés dans le dossier ne sont plus affichés à l'écran.

Pour masquer le contenu de tous les dossiers :

- Sélectionnez l'icône **Réduire Tout (Collapse All)** dans la barre d'outils pour masquer le contenu des dossiers et n'afficher que leur nom.

  Appuyez sur (menu) (2).

## Création de dossier

Vous disposez de deux méthodes différentes pour créer un dossier :

- Vous pouvez créer un dossier lorsque vous enregistrez un nouveau classeur : Les options de menu **Enregistrer (Save)** et **Enregistrer sous... (Save As...)** vous permettent de spécifier un nouveau nom de dossier pour le classeur.
- Vous pouvez créer un dossier directement à partir du navigateur de classeurs. Sélectionnez > **Nouveau dossier (New Folder)** dans la barre d'outils. Appuyez sur (enter) pour effacer le contenu du champ de nom. Entrez un nom pour le nouveau dossier et appuyez sur (enter) pour le créer.

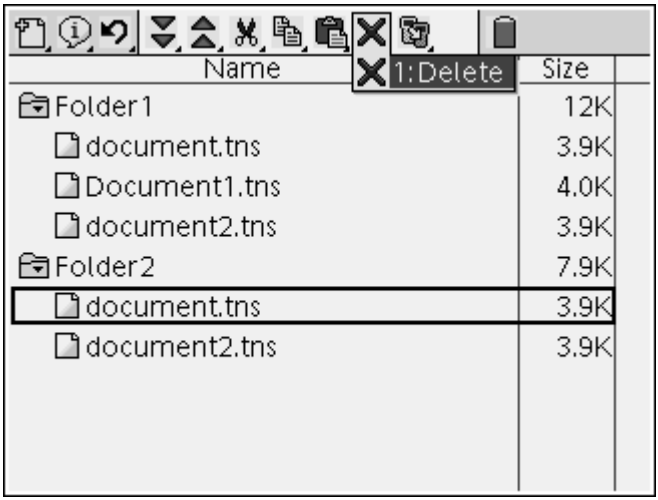
### *Suppression de classeur*

1. Appuyez sur les touches ▲ et ▼ pour mettre en surbrillance le classeur à supprimer.

   **Attention :** la fonction Annuler (Undo) ne peut pas être utilisée dans le cas d'une suppression de classeur. Assurez-vous donc de sélectionner le classeur approprié.

2. Sélectionnez dans la barre d'outils du navigateur de classeur.

   Appuyez sur (menu) (6).




   Le classeur sélectionné est supprimé.

### Conventions de dénomination des fichiers

Les noms de dossiers doivent être uniques et comporter 32 caractères au maximum. Les noms de fichiers peuvent combiner des caractères numériques et alphabétiques, ainsi que le caractère de soulignement (_). La plupart des caractères spéciaux ne sont pas autorisés dans les noms de fichiers.

## *Réglages du système*

Les réglages du système contrôlent la façon dont unité nomade de TI-Nspire™ CAS+ affiche et interprète les informations. Tous les nombres, y compris les éléments de matrices et les listes, sont affichés suivant les réglages de modes courants.

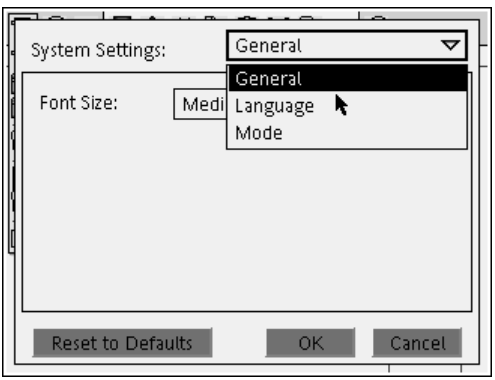
### Modification des réglages du système

1. Sélectionnez > **5: Réglages du système (System Settings)** dans la barre d'outils du navigateur de classeurs.

   Appuyez sur (menu) (0) (5).

   La fenêtre Réglages du système (System Settings) s'affiche avec la catégorie Général (General) sélectionnée.

2. Appuyez sur la touche ▼ pour afficher le menu déroulant des catégories.




3. Appuyez sur la touche ▼ de façon à mettre en surbrillance la catégorie de réglages à modifier.

4. Appuyez sur (enter).

   La fenêtre des réglages sélectionnés s'affiche.

5. Appuyez sur la touche ▶ de façon à mettre en surbrillance le réglage à modifier.
6. Appuyez sur la touche ▼ pour afficher la liste des valeurs possibles.
7. Appuyez sur la touche ▼ de façon à mettre en surbrillance la valeur à sélectionner.
8. Appuyez sur enter pour sélectionner la nouvelle valeur.
9. Une fois les réglages modifiés pour répondre à vos besoins, appuyez sur la touche enter de façon à mettre en surbrillance OK et appuyez sur enter pour appliquer votre changement.

   **Remarque :** cliquez sur **Rétablir les valeurs par défaut (Reset to Default)** pour rétablir les valeurs par défaut de l'unité nomade.

## *Utilisation du Catalogue (Catalog) pour accéder aux commandes*

Utilisez le Catalogue pour insérer des fonctions et des commandes, des unités, des symboles et des modèles d'expression dans vos activités. La fenêtre du Catalogue contient cinq onglets permettant d'accéder aux commandes, aux caractères spéciaux et aux modèles :

contient toutes les fonctions, classées par ordre alphabétique.

contient toutes les fonctions mathématiques.

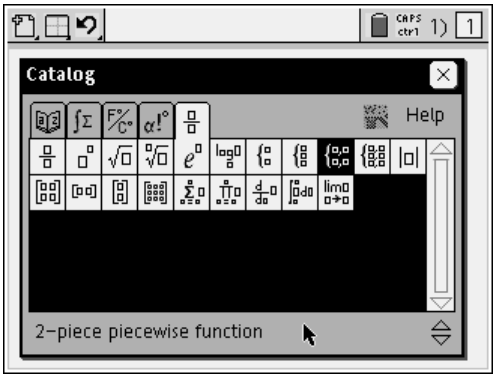
fournit les valeurs pour les unités de mesure standard.

affiche un jeu de symboles permettant l'insertion de caractères spéciaux.

contient des modèles facilitant la création d'expressions en écriture standard, notamment des dérivées partielles, des produits, des sommes, des racines carrées et des intégrales.

### Ouverture du Catalogue

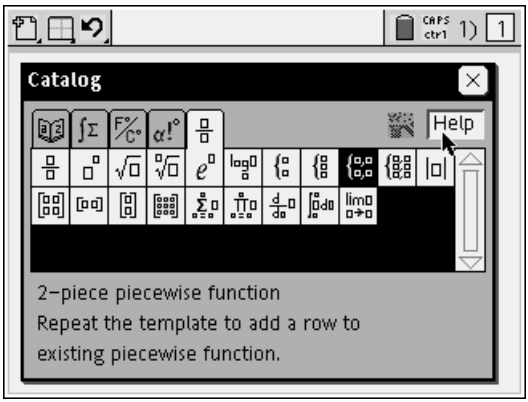
1. Appuyez sur ctrl / pour ouvrir le Catalogue (Catalog).




2. Appuyez sur ⊙ pour sélectionner l'onglet contenant la fonction, le symbole ou l'expression à insérer dans votre activité.

3. Appuyez sur ▼ de façon à mettre en surbrillance l'élément à insérer.

   **Remarque :** vous pouvez cliquer sur le bouton **Aide (Help)** pour afficher la syntaxe de l'élément sélectionné.




4. Appuyez sur enter pour insérer l'élément.

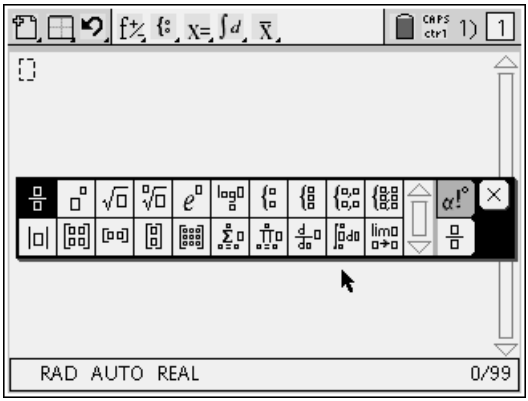
## *Saisie de caractères spéciaux*

Le jeu de symbole comporte un sous-ensemble des objets accessibles via le Catalogue (Catalog). Il contient deux onglets :

α° affiche un jeu de symboles permettant l'insertion de caractères spéciaux.

contient des modèles facilitant la création d'expressions en écriture standard, notamment des dérivées partielles, des produits, des sommes, des racines carrées et des intégrales.

1. Pour afficher le jeu de symboles, appuyez sur ctrl .




2. Appuyez sur ⊕ pour sélectionner l'onglet contenant la fonction, le symbole ou l'expression à insérer dans votre activité.
3. Appuyez sur ▼ de façon à mettre en surbrillance l'élément à insérer.
4. Appuyez sur enter pour insérer l'élément.

## *Saisie de caractères linguistiques internationaux*

La touche ⒻⒻ permet de saisir des caractères accentués ou ponctués spéciaux dans les applications qui autorisent la saisie de texte.

1. Ouvrez une application qui autorise la saisie de texte.
2. Entrez le texte voulu.
3. Mettez en surbrillance la lettre à accentuer, par exemple, « e » en français.
4. Appuyez sur la touche ⒻⒻ du clavier de l'unité nomade : le « e » se transforme en « é ». Maintenez enfoncée la touche ⒻⒻ jusqu'à localisation du « e » à insérer. Appuyez sur ⊖ ou sur la lettre suivante du texte pour accepter le caractère et poursuivre la saisie.

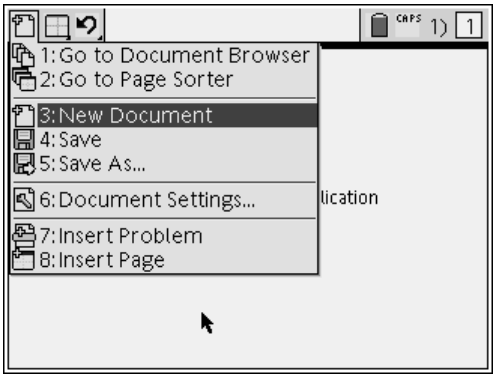
# Utilisation des classeurs

Toutes les opérations exécutées sur le labo de maths TI-Nspire™ sont enregistrées sous forme de classeur. Les classeurs du labo de maths TI-Nspire™ sont totalement portables et peuvent être partagés entre plusieurs utilisateurs d'unités nomades et utilisateurs du logiciel pour ordinateur.

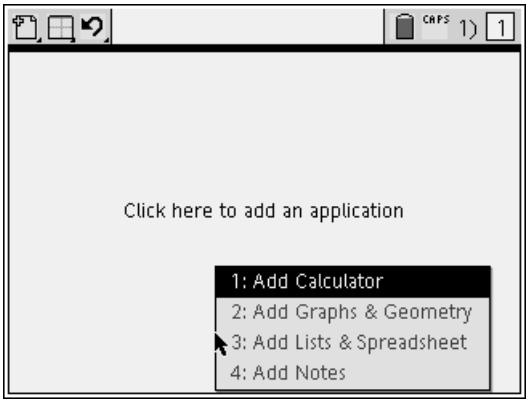
## *Création d'un nouveau classeur*

1. Sélectionnez > **3: Nouveau classeur (New Document)** dans le menu de la barre d'outils.

   Appuyez sur (menu) (0) (3).




   Le nouveau classeur s'ouvre et une page vide s'affiche.

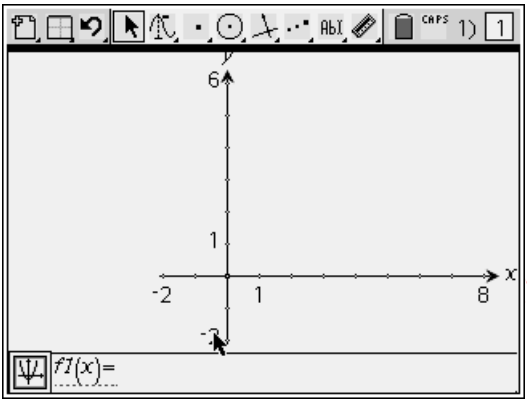


### *Ajout d'une application dans une page*

Lorsque vous ajoutez une nouvelle page à un classeur, un menu s'affiche vous invitant à ajouter une application.

1. Appuyez sur le numéro qui correspond à l'application à ajouter dans la page.

   L'application s'affiche alors dans l'espace de travail.

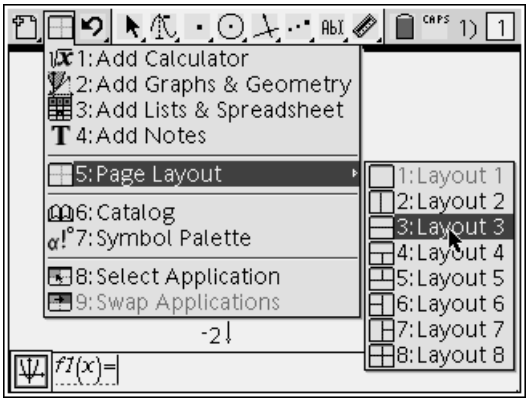



## Changement de mise en page

Par défaut, chaque page contient un espace permettant d'ajouter une application. Si vous souhaitez ajouter plusieurs applications dans une page, vous pouvez changer la mise en page pour accueillir jusqu'à quatre applications.

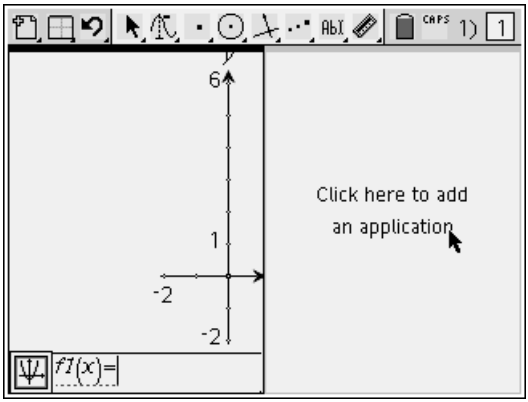
1. Sélectionnez > **5: Changer la mise en page (Change layout)** dans la barre d'outils.

   Appuyez sur (menu) 2 5.




2. Appuyez sur le numéro qui correspond au format à appliquer à la page.

   La mise en page est appliquée.




   **Remarque :** à tout moment, vous pouvez modifier la mise en page afin de pouvoir ajouter ou supprimer des applications.

## Ajout d'une activité à un classeur

Un classeur peut contenir jusqu'à 30 activités. Pour ajouter une nouvelle activité à un classeur :

1. Sélectionnez > **7: Insérer une activité (Insert Problem)** dans le menu de la barre d'outils.

   Appuyez sur menu 0 7.

Une nouvelle activité et une nouvelle page sont alors ajoutées au classeur.

### Ajout d'une page à une activité

Chaque activité peut comporter 50 pages au maximum. Pour ajouter une nouvelle page à une activité :

- Sélectionnez > **8: Insérer une page (Insert Page)** dans le menu de la barre d'outils.

  Appuyez sur (menu) (0) (8) pour ajouter une page vide,

  ou

- Sélectionnez et choisissez l'application à ajouter dans la nouvelle page.

  Appuyez sur (menu) (.) et sur le numéro correspondant à l'application.

Ces deux méthodes permettent d'ajouter une nouvelle page à l'activité courante.

### Consultation des différentes pages d'un classeur

La trieuse de pages vous permet de consulter les différentes pages de vos activités.

Les raccourcis suivants peuvent également être utilisés pour consulter les classeurs.

(ctrl) ◀ affiche la page précédente.

(ctrl) ▶ affiche la page suivante.

(ctrl) ▲ affiche la trieuse de pages.

(ctrl) ▼ affiche la dernière page de l'activité courante.

## *Utilisation de la trieuse de page*

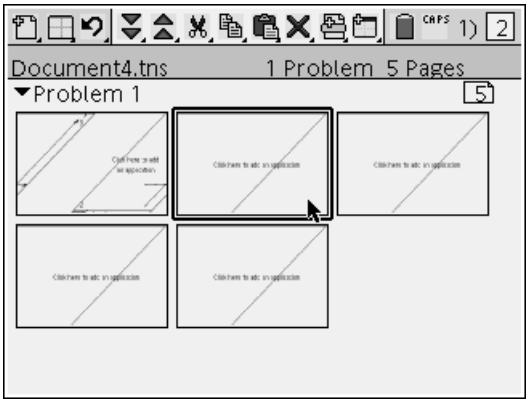
La trieuse de pages affiche toutes les activités associées à un classeur, ainsi que toutes les pages de chaque activité sous forme de miniatures. Utilisez la trieuse de pages pour réorganiser et supprimer des pages, copier une page d'une activité et l'insérer dans une autre activité ou encore pour appliquer des modèles aux pages.

## Ouverture de la trieuse de pages

- Sélectionnez > **2: Aller à la trieuse de page (Go to** Page Sorter) dans la barre d'outils du classeur.

  Appuyez sur (menu) (0) (2).

  La fenêtre de la trieuse de pages affiche alors toutes les activités et les pages du classeur courant.




## Options de la barre d'outils de la trieuse de pages

La trieuse de pages permet d'accéder à un ensemble de menus unique pour effectuer différentes opérations sur les pages.

| Menu | Option | Action |
|---|---|---|
| | **Menu Classeur (Document)** | |
| | 1: Aller au navigateur de classeurs (Go to Document Browser) | Ferme le classeur courant et affiche le navigateur de classeurs. |
| | 2: Afficher la page (View Page) | Affiche la page sélectionnée. |
| | 3: Nouveau classeur (New document) | Crée un classeur pour lequel la première activité et la première page sont affichées. |
| | 4: Enregistrer (Save) | Enregistre le classeur courant. |

| Menu | Option | Action |
|---|---|---|
| | 5: Enregistrer sous... (Save As...) | Enregistre le classeur courant. La boîte de dialogue Enregistrer (Save) s'affiche pour vous permettre de spécifier l'emplacement de stockage et le nom du classeur. |
| | 6: Réglages du classeur (Document Settings) | Affiche les réglages du classeur. |
| | 7: Insérer une activité (Insert Problem) | Ajoute une nouvelle activité au classeur courant. |
| | 8: Insérer une page (Insert Page) | Ajoute une nouvelle page à l'activité courante. |
| | 9: Var | Affiche les variables. |
| | **Menu Gestion des applications (Application Management)** | |
| | 1: Ajouter Calculs (Add Calculator) | Insère l'application Calculs dans la page courante. |
| | 2: Ajouter Graphiques & géométrie (Add Graphs & Geometry) | Insère l'application Graphiques & géométrie dans la page courante. |
| | 3: Ajouter Tableur & listes (Add Lists & Spreadsheet) | Insère l'application Tableur & listes dans la page courante. |
| | 4: Ajouter Éditeur mathématique (Add Notes) | Insère l'application Éditeur mathématique dans la page courante. |
| | 5: Mise en page (Page Layout)<br> | Utilisez la touche ▸ pour afficher les huit modèles prédéfinis de mise en page. |

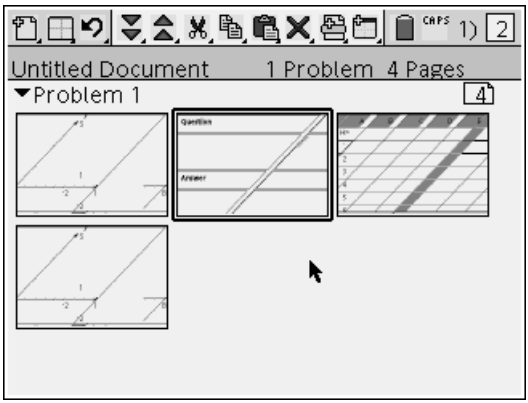
| Menu | Option | Action |
|---|---|---|
| | 6: Catalogue (Catalog) | Affiche le Catalogue (Catalog). |
| | 7: Jeu de symboles (Symbol Palette) | Affiche le jeu de symboles. |
| | 8: Sélectionner l'application (Select Application) | Sélectionne l'application courante. |
| | 9: Changer d'application (Swap Application) | Déplace l'application courante dans l'espace de travail occupé par l'application sélectionnée. |
| | **Menu Édition (Edit)** | |
| | 1: Annuler (Undo) | Annule le résultat de la dernière opération effectuée. |
| | 2: Rétablir (Redo) | Rétablit la dernière opération annulée par la commande Annuler (Undo). |
| | 3: Cut (Couper) | Supprime les informations sélectionnées de la page. |
| | 4: Copy (Copier) | Copie dans le Presse-papiers les informations sélectionnées dans la page courante. |
| | 5: Paste (Coller) | Insère les informations, précédemment coupées ou copiées, dans la page courante. |
| | 6: Delete (Supprimer) | Supprime l'application utilisée de la page courante. |
| | 7: Supprimer la page (Delete Page) | Supprime la page courante du classeur. |

## Affichage de toutes les pages d'un classeur

La trieuse de pages affiche toutes les activités associées à un classeur, ainsi que toutes les pages de chaque activité sous forme de miniatures.

1. Sélectionnez > **2: Aller à la trieuse de pages (Go to** Page Sorter) dans la barre d'outils.

   Appuyez sur (menu) (0) (2).




   L'ordre d'affichage des miniatures est celui des pages dans le classeur. Si le nombre de pages est trop élevé pour être affiché entièrement à l'écran, des barres de défilement apparaissent sur le côté de l'écran de façon à pouvoir accéder à toutes les pages du classeur.

### Réorganisation des pages dans un classeur

Pour déplacer une page dans un classeur avec la trieuse de pages :

1. Utilisez les touches ◀ et ▶ pour sélectionner la page à déplacer.

   Une bordure pleine s'affiche autour de la page sélectionnée.

2. Sélectionnez > **3: Couper (Cut)** dans la barre d'outils.

   Appuyez sur (menu) (3) (1).

3. Utilisez les touches ◀ et ▶ pour positionner le pointeur au nouvel emplacement voulu pour la page.

4. Sélectionnez > **5: Coller (Paste)** dans la barre d'outils.

   Appuyez sur (menu) (5) (1).

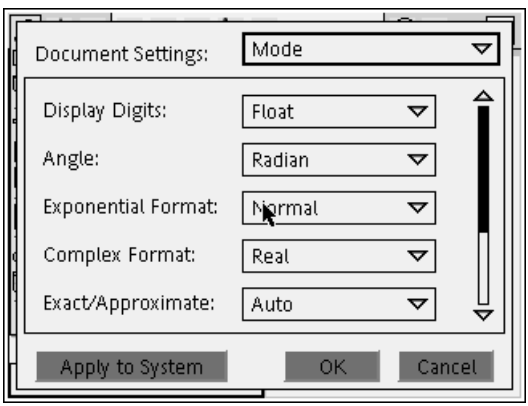
   La page est déplacée à l'emplacement choisi dans le classeur, le compteur de pages étant automatiquement actualisé.

## *Modification des réglages du classeur*

Vous pouvez modifier les réglages de mode d'un classeur si vous envisagez d'utiliser des réglages différents de ceux spécifiés pour l'unité nomade. Par exemple, pour changer le mode Angle ou le format Exponentiel utilisé par le classeur courant sans modifier les réglages des autres classeurs de l'unité nomade, utilisez la fonction Réglages du classeur (Document Settings).

1. Sélectionnez > **Réglages du classeur (Document Settings)** dans la barre d'outils.

   Appuyez sur (menu) (0) (6).




2. Appuyez sur la touche ▶ pour parcourir la liste des réglages vers le bas. (Appuyez sur la touche ◀ revenir en haut de la liste.)
3. Une fois trouvé le réglage à modifier, appuyez sur ▼ pour afficher la liste de sélection.
4. Appuyez sur les touches ▲ et ▼ pour mettre en surbrillance l'option de votre choix, puis appuyez sur (enter) pour la sélectionner.
5. Appuyez sur ▶ jusqu'à ce que le bouton **OK** soit mis en surbrillance.
6. Appuyez sur (enter) pour appliquer les nouveaux réglages du classeur.

   **Remarque :** si vous souhaitez appliquer les nouveaux réglages à tous les classeurs de l'unité nomade, mettez en surbrillance l'option **Appliquer au système (Apply to System)** et appuyez sur (enter).

# Annexe : Instructions et fonctions

## *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 ▶| pour passer d'un élément à l'autre et saisissez la valeur ou l'expression de l'élément affiché. Appuyez ensuite sur enter ou ctrl enter pour calculer l'expression.

**Modèle Fraction** — **Touches** ctrl ÷

Exemple :

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

**Modèle Exposant (Exponent)** — **Touche** $\sqrt[n]{x}$ ^

Exemple :

$$2^3 \qquad 8$$

**Modèle Racine carrée (Square root)** — **Touches** ctrl $\sqrt{\ }$ $x^2$

Exemple :

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

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

**Modèle Racine n-ième** — **Touches** ctrl $\sqrt[n]{x}$ ^

Exemple :

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

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

**Modèle *e* Exposant** — **Touches** ctrl $e^x$ ln

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

Exemple :

$$e^1 \qquad e$$

$$e^{1.} \qquad 2.71828182846$$

## Modèle Logarithme

**Touche** 10ˣ log

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

Calcule le logarithme selon la base spécifiée. Pour une base 10 par défaut, ne spécifiez pas de base.

Exemple :

$\log_4(2.0)$ .5

## Modèle Système de 2 équations

**Catalogue (Catalog) >**

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

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

Exemple :

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

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

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

## Modèle Système de n équations

**Catalogue (Catalog) >**

Permet de créer un système de *n* équations. Le système vous invite à définir *n*.

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

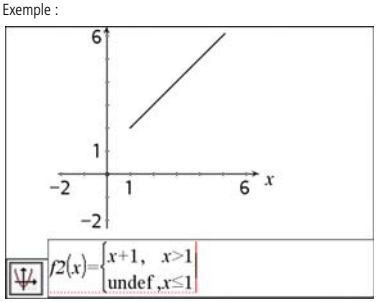
## Modèle Fonction définie par morceaux (2 morceaux)

**Catalogue (Catalog) >**

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

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

Exemple :




## Modèle Fonction définie par morceaux (n morceaux)

**Catalogue (Catalog) >**

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).

## Modèle Valeur absolue

**Catalogue (Catalog) >**

$|\square|$

Exemple :

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

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

Crée une matrice de type 2 x 2. Pour ajouter une nouvelle ligne et une nouvelle colonne à une matrice existante, cliquez dans le modèle et appliquez-le de nouveau.

Exemple :

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

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

Pour ajouter une colonne à une matrice existante, cliquez dans le modèle et appliquez-le de nouveau.

Exemple :

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

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

Pour ajouter une nouvelle ligne à une matrice existante, cliquez dans le modèle et appliquez-le de nouveau.

Exemple :

$$\begin{bmatrix}5\\ 8\end{bmatrix}\cdot .01 \qquad \begin{bmatrix}.05\\ .08\end{bmatrix}$$

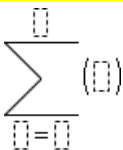
**Modèle Matrice (m x n)** **Catalogue (Catalog) >**

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

Exemple :

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

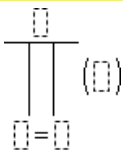
**Modèle Somme (Σ)** **Catalogue (Catalog) >**



Exemple :

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

**Modèle Produit (Π)** **Catalogue (Catalog) >**



Exemple :

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

**Modèle Dérivée ou dérivée n-ième** **Catalogue (Catalog) >**

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

L'utilisation de la séquence de touches ctrl − permet de n'avoir à saisir qu'un seul des paramètres pour le degré supérieur.

Exemple :

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

**Modèle Intégrale définie ou indéfinie** **Catalogue (Catalog) >**

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

L'utilisation de la séquence de touches Maj + avec une intégrale indéfinie permet de ne pas avoir à saisir de limites.

Exemple :

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

**Modèle Limite** **Catalogue (Catalog) >**

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

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

Exemple :

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

# *Liste alphabétique des opérations*

Les opérations dont le nom n'est pas alphabétique (comme +, ! et >) sont regroupées à la fin de cette section, à partir de la page 152. 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. De plus, en raison des contraintes liées au formatage, la plupart des résultats approximatifs ont été arrondis à la troisième décimale (par exemple, 3.14159265359 est affiché sous la forme 3.141).

## *A*

**abs()** **Catalogue > ∫Σ > Nombre**

**abs(***Expression1***)** ⇒ *expression*
**abs(***Liste1***)** ⇒ *list*
**abs(***Matrice1***)** ⇒ *matrice*
Donne la valeur absolue de l'argument.
Si l'argument est un nombre complexe, donne le module de ce nombre.
**Remarque :** toutes les variables non définies sont considérées comme réelles.

abs({π/2,-π/3})

$$\left\{\frac{\pi}{2},\frac{\pi}{3}\right\}$$

abs(2-3i)

$$\sqrt{13}$$

abs(z)

$$|z|$$

abs(x+yi)

$$\sqrt{x^2+y^2}$$

## and

**Catalogue > ∫Σ > Base**

*Expression booléenne1* **and** *Expression booléenne2* ⇒ *Expression booléenne*

*Liste booléenne1* **and** *Liste booléenne2* ⇒ *Liste booléenne*

*Matrice booléenne1* **and** *Matrice booléenne2* ⇒ *Matrice booléenne*

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

x≥3 and x≥4

x≥4

{x≥3,x≤0} and {x≥4,x ≤ -2}

{x≥4,x≤-2}

*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 32 bits signés. Lorsque les bits comparés correspondent, le résultat est 1 si dans les deux cas il s'agit d'un bit 1 ; dans les autres cas, le résultat est 0. La valeur donnée représente le résultat des bits et elle est affichée selon le mode Base utilisé.

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

Si vous entrez un nombre dont le codage binaire signé dépasse 32 bits, il est ramené à l'aide d'une congruence dans la plage appropriée.

En mode base Hex :

0h7AC36 and 0h3D5F

0h2C16

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

En mode base Bin :

0b100101 and 0b100

0b100

En mode base Dec :

37 and 0b100

4

**Remarque :** une entrée binaire peut comporter jusqu'à 32 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 8 chiffres.

## angle()

**Catalogue >**

**angle(***Expression1***)** ⇒ *expression*

Donne l'argument de *Expression1*, où *Expression1* est interprétée comme un nombre complexe.

**Remarque :** toutes les variables non définies sont considérées comme réelles.

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*)

$\frac{\pi}{4}$

angle(z)

angle(x+ *i*y)

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

**angle(***Liste1***)** ⇒ *liste*

**angle(***Matrice1***)** ⇒ *matrice*

Donne une liste ou une 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.

En mode Angle en radians :

angle({1+2*i*,3+0*i*,0−4*i*})

$$\blacksquare\ \text{angle}\left(\{1+2\cdot i\quad 3+0\cdot i\quad 0\}\right)$$
$$\left\{\frac{\pi}{2}-\tan^{-1}(1/2)\quad 0\quad \frac{-\pi}{2}\right\}$$

## ANOVA

**Catalogue > ∫Σ > Statistiques**

**ANOVA** *Liste1*,*Liste2*[,*Liste3*,...,*List20*][,*Indicateur*]
Effectue une analyse unidirectionnelle de variance pour comparer les moyennes de deux à vingt populations. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)
*Indicateur*=0 pour Données, *Indicateur*=1 pour Stats

| Variable de sortie | Description |
|---|---|
| F | Valeur de F statistique. |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **df** | Degré de liberté des groupes |
| **ss** | Somme des carrés des groupes |
| **MS** | Moyenne des carrés des groupes |
| **dfError** | Degré de liberté des erreurs |
| **SSError** | Somme des carrés des erreurs |
| **MSError** | Moyenne des carrés des erreurs |
| **sp** | Écart-type du groupe |
| **xbarlist** | Moyenne des entrées des listes |
| **CLowerList** | Limites inférieures des intervalles de confiance de 95 % pour la moyenne de chaque liste d'entrée |
| **CUpperList** | Limites supérieures des intervalles de confiance de 95 % pour la moyenne de chaque liste d'entrée |

## ANOVA2way

**Catalogue > ∫Σ > Statistiques**

**ANOVA2way** *Liste1*,*Liste2*[,…[,*Liste10*]][,*NivLign*]
Effectue une analyse de variance à deux facteurs pour comparer les moyennes de deux à vingt populations. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)
*NivLign*=0 pour Bloc
*NivLign*=2,3,...,*Long*-1, pour 2 facteurs, où
*Long*=length(*Liste1*)=length(*Liste2*) = … = length(*Liste10*) et
*Long* / *NivLong* ∈ {2,3,…}

Sorties : Bloc

| Variable de sortie | Description |
|---|---|
| F | F statistique du facteur de colonne |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **df** | Degré de liberté du facteur de colonne |
| **ss** | Somme des carrés du facteur de colonne |
| **MS** | Moyenne des carrés du facteur de colonne |
| | |
| F**Block** | F statistique du facteur |

| Variable de sortie | Description |
|---|---|
| **PValBlock** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **dfBlock** | Degré de liberté du facteur |
| **SSBlock** | Somme des carrés du facteur |
| **MSBlock** | Moyenne des carrés du facteur |
| | |
| **dfError** | Degré de liberté des erreurs |
| **SSError** | Somme des carrés des erreurs |
| **MSError** | Moyenne des carrés des erreurs |
| **s** | Écart-type de l'erreur |

Répts ég. 2 facteurs

Sorties FACTEUR DE COLONNE

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

Sorties FACTEUR DE LIGNE

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

Sorties INTERACTION

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

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

## ans

**ans** ⇒ *valeur*
Donne le résultat de la dernière expression calculée.

```
56
                                  56
ans + 4
                                  60
ans + 4
                                  64
ans + 4
                                  68
```

## approx()

**Catalogue > ∫Σ > Algèbre**

**approx(***Expression1***)** ⇒ *expression*
Donne une approximation décimale de *Expression1* sous forme d'expression, dans la mesure du possible, indépendamment du mode Exact/Approché (Exact/Approx) utilisé.
Ceci est équivalent à la saisie de *Expression1* suivie d'une pression sur ctrl ≈ enter.

```
approx(π)
                                  3.141...
```

**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({sin(π),cos(π)})
                                  {0.,-1.}
approx([√(2)  √(3)])
                                  [1.414... 1.732...]
```

## arcLen()

**Catalogue > ∫Σ > Analyse**

**arcLen(***Expression1***,***Var***,***Début***,***Fin***)** ⇒ *expression*
Donne la longueur de l'arc de la courbe définie par *Expression1* entre les points d'abscisses *Début* et *Fin* en fonction de la variable *Var*.
La longueur d'arc est calculée sous forme d'intégrale en supposant la définition du mode fonction.

```
arcLen(cos(x),x,0,π)
                                  3.820...
arcLen(f(x),x,a,b)
                                  b
                                  ∫ √((d/dx(f(x)))²+1) dx
                                  a
```

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

```
arcLen({sin(x),cos(x)},x,0,π)
                                  {3.820...,3.820...}
```

## augment()

**Catalogue > ∫Σ > Matrice**

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

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

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

{1,-3,2,5,4}

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

Donne une nouvelle matrice obtenue en ajoutant les lignes/colonnes de la *Matrice2* à celles de la *Matrice1*. Si le caractère « , » est utilisé, 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(***Expression1*, *Var* [, *H*]**)** ⇒ *expression*

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

*Expression1* peut être un nom de fonction défini par l'utilisateur (voir **Func**).

*H* correspond à la valeur de l'incrément. Si *H* n'est pas précisé, il est fixé par défaut à 0.001.

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

avgRC(f(x),x,h)

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

avgRC(sin(x),x,h)|x=2

$\frac{\sin(h+2) - \sin(2)}{h}$

avgRC($x^2$−x+2,x)

2. • (x − .4995)

avgRC($x^2$−x+2,x,.1)

2. • (x − .45)

avgRC($x^2$−x+2,x,3)

2 • (x+1)

# *B*

## ▸Bin

**Catalogue > ∫Σ > Base**

*Entier1* **▸Bin** ⇒ *entier*

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

256▸Bin

0b100000000

0h1F▸Bin

0b11111

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

0b *nombreBinaire*

0h *nombreHexadécimal*

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.

Si vous entrez un nombre dont le codage binaire signé dépasse 32 bits, il est ramené à l'aide d'une congruence dans la plage appropriée.

Une entrée binaire peut comporter jusqu'à 32 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 8 chiffres.

**binomCdf()** **Catalogue > ∫Σ > Probabilité**

**binomCdf(***n*,*p*[,*lowerBound*[,*upperBound*]]**)**
Calcule la fonction de répartition d'une loi binomiale discrète avec un nombre *n* d'essais et une probabilité *p* de réussite pour chaque essai.

**binomPdf()** **Catalogue > ∫Σ > Probabilité**

**binomPdf(***n*,*p*[,*ValX*]**)**
Calcule la probabilité de *ValX* pour la loi binomiale discrète avec un nombre *n* d'essais et la probabilité *p* de réussite pour chaque essai.

# C

**ceiling()** **Catalogue > ∫Σ > Nombre**

**ceiling(***Expression1***)** ⇒ *entier*
Donne le plus petit entier ≥ à l'argument.
L'argument peut être un nombre réel ou un nombre complexe.
**Remarque :** voir aussi **floor()**.

**ceiling(***Liste1***)** ⇒ *liste*
**ceiling(***Matrice1***)** ⇒ *matrice*
Donne une liste ou une matrice de plus petites valeurs supérieures ou égales à chaque élément.

ceiling(0.456)
1.

ceiling({⁻3.1,1,2.5})
{⁻3.,1,3.}

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

**cFactor()** **Catalogue > ∫Σ > Algèbre**

**cFactor(***Expression1*[,*Var*]**)** ⇒ *expression*
**cFactor(***Liste1*[,*Var*]**)** ⇒ *liste*
**cFactor(***Matrice1*[,*Var*]**)** ⇒ *matrice*

**cFactor(***Expression1***)** factorise *Expression1* dans C en fonction de toutes ses variables et sur un dénominateur commun.
La factorisation de *Expression1* décompose l'expression en autant de facteurs rationnels linéaires que possible même si cela introduit de nouveaux nombres non réels. Cette alternative peut s'avérer utile pour factoriser l'expression en fonction de plusieurs variables.

cFactor($a^3 \cdot x^2+a \cdot x^2+a^3+a$)
$a \cdot (a + ⁻i) \cdot (a + i) \cdot (x + ⁻i) \cdot (x + i)$

cFactor($x^2+4/9$)
$\frac{3 \cdot x + ⁻2 \cdot i \cdot 3 \cdot x + 2 \cdot i}{9}$

cFactor($x^2+3$)
$x^2 + 3$

cFactor($x^2+a$)
$x^2 + a$

## cFactor()

**Catalogue > ∫Σ > Algèbre**

**cFactor(***Expression1***,** *Var***)** factorise *Expression1* dans C en fonction de la variable *Var*.

La factorisation de *Expression1* décompose l'expression en autant de facteurs possible qui sont linéaires dans *Var*, avec peut-être des constantes non réelles, même si cela introduit des constantes irrationnelles ou des sous-expressions qui sont irrationnelles dans d'autres variables.

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

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

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

cFactor($a^3 \cdot x^2+a \cdot x^2+a^3+a,x$)

$a \cdot (a^2 + 1) \cdot (x + -i) \cdot (x + i)$

cFactor($x^2+3,x$)

$(x + \sqrt{3} \cdot i) \cdot (x + -\sqrt{3} \cdot i)$

cFactor($x^2+a,x$)

$(x + \sqrt{a} \cdot -i) \cdot (x + \sqrt{a} \cdot i)$

cFactor($x^5+4x^4+5x^3-6x-3$)

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

cFactor(ans,x)

$(x - .965) \cdot (x + .612) \cdot (x + 2.13) \cdot (x + 1.11 - 1.07 \cdot i) \cdot (x + 1.11 + 1.07 \cdot i)$

## char()

**Catalogue > ∫Σ > Chaîne**

**char(***Entier***)** ⇒ *caractère*

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 255.

char(38)

"&"

char(65)

"A"

## $\chi^2$2way

**Catalogue > ∫Σ > Statistiques**

**$\chi^2$2way** *MatriceObservée*

**chi22way** *MatriceObservée*

Effectue un test $\chi^2$ d'association sur le tableau 2 • 2 de valeurs dans la matrice observée *MatriceObservée*. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| $\chi^2$ | Stats Khi2 : sum(observée - attendue)$^2$/attendue |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **df** | Degré de liberté des statistiques khi2. |
| **ExpMat** | Matrice du tableau de valeurs élémentaires attendues, acceptant l'hypothèse nulle |
| **CompMat** | Matrice des contributions statistiques khi2 élémentaires |

## $\chi^2$Cdf()

**Catalogue > ∫Σ > Probabilité**

**$\chi^2$Cdf(***lowerBound***,***upperBound***,***df***)**
**chi2Cdf(***lowerBound***,***upperBound***,***df***)**
Calcule la fonction de répartition de la loi $\chi^2$ entre *lowerBound* et *upperBound* à *df* degrés de liberté.

## $\chi^2$GOF

**Catalogue > ∫Σ > Statistiques**

**$\chi^2$GOF** *ListeObservée*,*ListeAttendue*,*df*
**chi2GOF** *ListeObservée*,*ListeAttendue*,*df*
Effectue un test pour s'assurer que les données des échantillons sont issues d'une population conforme à la loi spécifiée. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| $\chi^2$ | Stats Khi2 : sum(observée - attendue)$^2$/attendue |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **df** | Degré de liberté des statistiques khi2. |
| **CompList** | Contributions statistiques khi2 élémentaires |

## $\chi^2$Pdf()

**Catalogue > ∫Σ > Probabilité**

**$\chi^2$Pdf(***ValX***,***df***)**
**chi2Pdf(***ValX***,***df***)**
Calcule la densité de probabilité (pdf) de la loi $\chi^2$ à une valeur *ValX* spécifiée et à *df* degrés de liberté.

## clearAZ

**Catalogue >**

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

## colAugment()

**Catalogue > ∫Σ > Matrice**

**colAugment(***Matrice1***,** *Matrice2***)** ⇒ *matrice*
Donne une nouvelle matrice obtenue en ajoutant les lignes 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.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \rightarrow M1$

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

[5 6] → M2

[5 6]

colAugment(M1,M2)

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

## colDim()

Catalogue > ∫Σ > Matrice

**colDim(***Matrice***)** ⇒ *expression*

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

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

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

3

## colNorm()

Catalogue > ∫Σ > Matrice

**colNorm(***Matrice***)** ⇒ *expression*

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

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

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

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

colNorm(mat)

9

## comDenom()

Catalogue > ∫Σ > Algèbre

**comDenom(***Expression1***[,***Var***])** ⇒ *expression*
**comDenom(***Liste1***[,***Var***])** ⇒ *liste*
**comDenom(***Matrice1***[,***Var***])** ⇒ *matrice*

**comDenom(***Expression1***)** donne le rapport réduit d'un numérateur entièrement développé sur un dénominateur entièrement développement.

comDenom($(y^2+y)/(x+1)^2+y^2+y$)

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

**comDenom(***Expression1***,***Var***)** donne le rapport réduit d'un numérateur et d'un dénominateur développé par rapport à *Var*. Les termes et leurs facteurs sont triés, *Var* étant la variable principale. Les mêmes puissances de *Var* sont regroupées. Une factorisation incidente des coefficients regroupés peut se produire. L'utilisation de *Var* permet de gagner du temps, de la mémoire et de l'espace sur l'écran tout en facilitant la lecture de l'expression. Les opérations suivantes basées sur le résultat obtenu sont également plus rapides et moins consommatrices de mémoire.

comDenom($(y^2+y)/(x+1)^2+y^2+y,x$)

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

comDenom($(y^2+y)/(x+1)^2+y^2+y,y$)

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

Si *Var* n'intervient pas dans *Expression1*, **comDenom(***Expression1***,***Var***)** donne le rapport réduit d'un numérateur non développé sur un dénominateur non développé. Ce type de résultat offre généralement un gain de temps, de mémoire et d'espace sur l'écran. La factorisation partielle du résultat contribue également à accélérer les opérations suivantes basées sur le résultat et à utiliser moins de mémoire.

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

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

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

Done

comden($(y^2+y)/(x+1)^2+y^2+y$)

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

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

$$1234\cdot x\cdot(x\cdot y+2)\cdot(y^2-1)$$

## conj()

**Catalogue > ∫Σ > Complexe**

**conj(***Expression1***)** ⇒ *expression*
**conj(***Liste1***)** ⇒ *liste*
**conj(***Matrice1***)** ⇒ *matrice*
Donne le conjugué de l'argument.
**Remarque :** toutes les variables non définies sont considérées comme réelles.

conj(1+2*i*)

$1-2\cdot i$

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

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

conj(z)

z

conj(x+*i*y)

$x-i\cdot y$

## corrMat()

**Catalogue > ∫Σ > Statistiques**

**corrMat(***Liste1***,***Liste2***[,…[,***Liste20***]])**
Calcule la matrice de corrélation de la matrice augmentée [*Liste1 Liste2 . . . Liste20*].

## cos()

**Touche** cos⁻¹ cos

**cos(***Expression1***)** ⇒ *expression*
**cos(***Liste1***)** ⇒ *liste*
**cos(***Expression1***)** calcule le cosinus de l'argument et l'affiche sous forme d'expression.
**cos(***Liste1***)** donne une liste des cosinus de tous les éléments de *Liste1*.
**Remarque :** l'argument est interprété comme la mesure d'un angle en degrés, en grades ou en radians, suivant le mode angulaire en cours d'utilisation. Vous pouvez utiliser °, G ou r pour préciser l'unité employée temporairement pour le calcul.

En mode Angle en degrés :

cos((π/4)ʳ)

$\frac{\sqrt{2}}{2}$

cos(45)

$\frac{\sqrt{2}}{2}$

cos({0,60,90})

{1,1/2 0}

En mode Angle en grades :

cos({0,50,100})

$\{1\ \frac{\sqrt{2}}{2}\ 0\}$

En mode Angle en radians :

cos(π/4)

$\frac{\sqrt{2}}{2}$

cos(45°)

$\frac{\sqrt{2}}{2}$

## cos()

**Touche** cos⁻¹ COS

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

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

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

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

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

Formation des matrices :

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

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

cos (B) =

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

Tous les calculs sont exécutés en virgule flottante.

En mode Angle en radians :

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

$$\begin{bmatrix} .212\dots & .205\dots & .121\dots \\ .160\dots & .259\dots & .037\dots \\ .248\dots & -.090\dots & .218\dots \end{bmatrix}$$

## cos⁻¹()

**Touche** ctrl cos⁻¹ COS

**cos⁻¹(***Expression1***)** ⇒ *expression*
**cos⁻¹(***Liste1***)** ⇒ *liste*

**cos⁻¹(***Expression1***)** donne l'arc cosinus de *Expression1* et l'affiche sous forme d'expression.

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

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

En mode Angle en degrés :

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

0

In Gradian angle mode:

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

100

En mode Angle en radians :

$\cos^{-1}(\{0,.2,.5\})$

$$\left\{\frac{\pi}{2}, 1.37, 1.05\right\}$$

**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 radians et en mode Format complexe Rectangulaire :

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

$$\begin{bmatrix} 1.734\dots+.064\dots\cdot i & -1.490\dots+2.105\dots\cdot i\dots \\ -.725\dots+1.515\dots\cdot i & .623\dots+.778\dots\cdot i\dots \\ -2.083\dots+2.632\dots\cdot i & 1.790\dots-1.271\dots\cdot i\dots \end{bmatrix}$$

## cosh()

**Catalogue > ∫Σ > Hyperbolique**

**cosh(***Expression1***)** ⇒ *expression*
**cosh(***Liste1***)** ⇒ *liste*
**cosh(***Expression1***)** donne le cosinus hyperbolique de l'argument et l'affiche sous forme d'expression.
**cosh(***Liste1*) donne une liste des cosinus hyperboliques de chaque élément de *Liste1*.

cosh(1.2)

1.810...

cosh({0,1.2})

{1,1.810...}

**cosh(***matriceCarrée1***)** ⇒ *matriceCarrée*
Donne le cosinus hyperbolique de la matrice *matriceCarréeMatrix1*. 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.

En mode Angle en radians :

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

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

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

**Catalogue > ∫Σ > Hyperbolique**

**cosh$^{-1}$(***Expression1***)** ⇒ *expression*
**cosh$^{-1}$(***Liste1***)** ⇒ *liste*

**cosh$^{-1}$(***Expression1*) donne l'argument cosinus hyperbolique de l'argument et l'affiche sous forme d'expression.
**cosh$^{-1}$(***Liste1*) donne une liste des arguments cosinus hyperboliques de chaque élément de *Liste1*.

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

0

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

{0,1.372...,$\cosh^{-1}$(3)}

**cosh$^{-1}$(***matriceCarrée1***)** ⇒ *matriceCarrée*
Donne l'argument cosinus hyperbolique de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de l'argument cosinus hyperbolique de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.
*matriceCarrée1* doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

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

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

$$\begin{bmatrix} 2.525\ldots+1.734\ldots\cdot i & -.009\ldots-1.490\ldots\cdot i & \ldots \\ .486\ldots-.725\ldots\cdot i & 1.662\ldots+.623\ldots\cdot i & \ldots \\ -.322\ldots-2.083\ldots\cdot i & 1.267\ldots+1.790\ldots\cdot i & \ldots \end{bmatrix}$$

## cot()

**Catalogue > ∫Σ > Trigonomètrie**

**cot(***Expression1***)** ⇒ *expression*
**cot(***Liste1***)** ⇒ *liste*
Affiche la cotangente de *Expression1* ou retourne la liste des cotangentes des éléments de *Liste1*.
**Remarque :** donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

En mode Angle en degrés :

cot(45)

1

En mode Angle en grades :

cot(50)

1

En mode Angle en radians :

cot({1,2.1,3})

$$\left\{\frac{1}{\tan(1)}, -.585, \frac{1}{\tan(3)}\right\}$$

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

**Catalogue > ∫Σ > Algèbre**

**cot$^{-1}$(***Expression1***)** ⇒ *expression*
**cot$^{-1}$(***Liste1***)** ⇒ *liste*
Donne l'arc cotangente de *Expression1* 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.

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)

## coth()

**Catalogue > ∫Σ > Hyperbolique**

**coth(***Expression1***)** ⇒ *expression*
**cot(***Liste1***)** ⇒ *liste*
Affiche la cotangente hyperbolique de *Expression1* ou donne la liste des cotangentes hyperboliques des éléments de *Liste1*.

coth(1.2)

1.199…

coth({1,3.2})

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

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

**Catalogue > ∫Σ > Hyperbolique**

**coth$^{-1}$(***Expression1***)** ⇒ *expression*
**coth$^{-1}$(***Liste1***)** ⇒ *liste*
Affiche l'argument cotangente hyperbolique de *Expression1* ou donne la liste comportant les arguments cotangentes hyperboliques des éléments de *Liste1*.

coth$^{-1}$(3.5)

.293…

coth$^{-1}$({-2,2.1,6})

$$\left\{\frac{-\ln(3)}{2}, .518046, \frac{\ln\left(\frac{7}{5}\right)}{2}\right\}$$

## crossP()

**Catalogue > ∫Σ > Matrice**

**crossP(***Liste1***,** *Liste2***)** ⇒ *liste*
Donne le produit vectoriel de *Liste1* et de *Liste2* et l'affiche sous forme de liste.

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

crossP({a1,b1},{a2,b2})

{0,0,a1 · b2−a2 · b1}

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

{-2.5,-5.,-2.25}

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

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

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

[-3 6 -3]

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

[0 0 -2]

## csc()

**Catalogue > ∫Σ > Algèbre**

**csc(***Expression1***)** ⇒ *expression*
**csc(***Liste1***)** ⇒ *liste*
Affiche la cosécante de *Expression1* ou donne une liste comportant les cosécantes de tous les éléments de *Liste1*.

En mode Angle en degrés :

csc(45)

$\sqrt{2}$

En mode Angle en grades :

csc(50)

$\sqrt{2}$

En mode Angle en radians :

csc({1,π/2,π/3})

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

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

**Catalogue > ∫Σ > Algèbre**

**csc$^{-1}$(***Expression1***)** ⇒ *expression*
**csc$^{-1}$(***Liste1***)** ⇒ *liste*
Affiche l'angle dont la cosécante correspond à *Expression1*ou donne la liste des arcs cosécante de chaque élément de *Liste1*.

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

En mode Angle en degrés :

csc$^{-1}$(1)

90

En mode Angle en grades :

csc$^{-1}$(1)

100

En mode Angle en radians :

csc$^{-1}$({1,4,6})

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

## csch()

**Catalogue > ∫Σ > Hyperbolique**

**csch(***Expression1***)** ⇒ *expression*
**csch(***Liste1***)** ⇒ *liste*
Affiche la cosécante hyperbolique de *Expression1* ou donne la liste des cosécantes hyperboliques de tous les éléments de *Liste1*.

csch(3)

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

csch({1,2.1,4})

$$\left\{\frac{1}{\sinh(1)},.248\ldots,\frac{1}{\sinh(4)}\right\}$$

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

**Catalogue > ∫Σ > Hyperbolique**

**csch$^{-1}$(***Expression1***)** ⇒ *expression*
**csch$^{-1}$(***Liste1***)** ⇒ *liste*
Affiche l'argument cosécante hyperbolique de *Expression1* ou donne la liste des arguments cosécantes hyperboliques de tous les éléments de *Liste1*.

csch$^{-1}$(1)

sinh$^{-1}$(1)

csch$^{-1}$({1,2.1,3})

{sinh$^{-1}$(1),.459…,sinh$^{-1}$(1/3)}

## cSolve()

**Catalogue > ∫Σ > Algèbre**

**cSolve(***Équation***,** *Var***)** ⇒ *Expression booléenne*
**cSolve(***Inéquation***,** *Var***)** ⇒ *Expression booléenne*

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

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

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

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

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

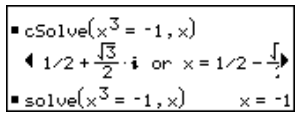
**Remarque :** si *Équation* n'est pas polynomiale avec les fonctions comme **abs()**, **angle()**, **conj()**, **real()** ou **imag()**, ajoutez un caractère de soulignement (en appuyant sur ctrl ␣) à la fin de *Var*. Par défaut, les variables sont considérées comme réelles.

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

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

cSolve($x^3=-1$,x)

solve($x^3=-1$,x)




cSolve($x^{1/3}=-1$,x)

faux

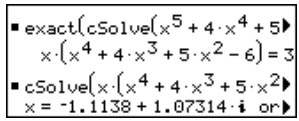
solve($x^{1/3}=-1$,x)

$x=-1$

En mode Afficher chiffres, Fixe 2 :

exact(cSolve($x^5+4x^4+5x^3-6x-3=0$,x))

cSolve(ans,x)




z est considéré comme réel :

cSolve(conj(z)=1+ *i*,z)

z=1+ *i*

z_ est considéré comme complexe :

cSolve(conj(z_)=1+ *i*,z_)

z_=1− *i*

**cSolve(***Équation1* **and** *Équation2* [**and ...** ]**,**
**{***VarOuInit1***,** *VarOuInit2* [**, ...** ]**})** ⇒ *Expression booléenne*

Donne les solutions complexes possibles à des équations algébriques simultanées, où chaque *VarOuInit* définit une variable à résoudre.

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

*variable*

– ou –

*variable = nombre réel ou non réel*

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

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

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

**cSolve()** **Catalogue > $\int\Sigma$ > Algèbre**

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

cSolve(u_ · v_−u_=v_ v_$^2$=−u_,{u_,v_})

$$u\_=1/2+\frac{\sqrt{3}}{2}\cdot i \text{ and } v\_=1/2-\frac{\sqrt{3}}{2}\cdot i$$

$$\text{or } u\_=1/2-\frac{\sqrt{3}}{2}\cdot i \text{ and } v\_=1/2+\frac{\sqrt{3}}{2}\cdot i$$

$$\text{or } u\_=0 \text{ and } v\_=0$$

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

cSolve(u_ · v_−u_=c_ · v_ v_$^2$=−u_,{u_,v_})

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

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

$$\text{or } u\_=0 \text{ and } v\_=0$$

Vous pouvez également utiliser des variables de solution qui n'apparaissent pas dans les équations. Ces solutions montrent comment des groupes de solutions peuvent contenir des constantes arbitraires de type @k, où *k* est un suffixe entier compris entre 1 et 255.

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

cSolve(u_ · v_−u_=v_ and v_$^2$=−u_,{u_,v_,w_})

$$u\_=1/2+\frac{\sqrt{3}}{2}\cdot i \text{ and } v\_=1/2-\frac{\sqrt{3}}{2}\cdot i$$

and w_=**c**1

or

$$u\_=1/2-\frac{\sqrt{3}}{2}\cdot i \text{ and } v\_=1/2+\frac{\sqrt{3}}{2}\cdot i$$

and w_=**c**1

or u_=0 and v_=0 and w_=**c**1

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

cSolve(u_+v_=$e^{w\_}$ and u_−v_= *i*, {u_,v_})

$$u\_=\frac{e^{w\_}}{2}+1/2\cdot i \text{ and } v\_=\frac{e^{w\_}-i}{2}$$

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

cSolve($e^{z\_}$=w_ and w_=z_$^2$, {w_,z_})

w_=.494… and z_=−.703…

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

cSolve($e^{z\_}$=w_ and w_=z_$^2$, {w_,z_=1+ *i*})

w_=.149… + 4.891… · *i* and

z_=1.588… + 1.540… · *i*

**CubicReg** **Catalogue > $\int\Sigma$ > Statistiques**

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

Effectue un ajustement polynomial de degré 3 et met à jour toutes les variables statistiques. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

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

*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| **a,b,c,d** | Coefficients d'ajustement |
| **r²** | Coefficient de détermination |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y - (a • $x^3$+b • $x^2$+c • x+d) |
| **RegEqn** | Équation d'ajustement : a • $x^3$+b • $x^2$+c • x+d |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

## cumSum()

**Catalogue > ∫Σ > Liste**

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

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

**cumSum(***Matrice1***)** ⇒ *matrice*
Donne une 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.

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

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

cumSum(m1)

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

## Cycle

**Barre d'outils Calculs> {□ □} > Structures de contrôle**

**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**).

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

```
:0 → temp
:For i,1,100,1
:If i=50
:Cycle
:temp+i → temp
:EndFor
Return temp
```

Valeur affichée après l'exécution : 5000

## ▸Cylind

**Catalogue > ∫Σ > Matrice**

*Vecteur* **▸Cylind**
Affiche le vecteur ligne ou colonne en coordonnées cylindriques [r∠θ, z].
*Vecteur* doit être un vecteur à trois éléments. Il peut s'agir d'un vecteur ligne ou colonne.

[2,2,3] ▸Cylind

$[2 \cdot \sqrt{2} \angle \frac{\pi}{4} \quad 3]$

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

Donne une liste des valeurs réelles et non réelles possibles de *Var* qui annulent l'expression, soit *Expression*=0. Pour y parvenir, **cZeros()** calcule **expr▸list(cSolve(***Expression*=0,*Var***),***Var***)**. Pour le reste, **cZeros()** est comparable à **zeros()**.

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

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

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

**cZeros({***Expression1***,** *Expression2* **[,** … **] },**
**{***VarOuInit1***,***VarOuInit2* **[,** … **] })** ⇒ *matrice*

Donne les positions possibles auxquelles les expressions sont annulées simultanément. Chaque *VarOuInit* définit une inconnue dont vous recherchez la valeur.

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

*variable*
– ou –
*variable = nombre réel ou non réel*

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

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

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

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

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

En mode Afficher chiffres, Fixe 3 :

cZeros($x^5+4x^4+5x^3-6x-3,x$)

$\{-1.114+1.073\cdot i,-1.114-1.073\cdot i,-2.125,-.612,.965\}$

z est considéré comme réel :

cZeros(conj(z)−1−***i***,z)

{1+***i***}

z_ est considéré comme complexe :

cZeros(conj(z_)−1−***i***,z_)

$\{\{c3,i=c3-1\}$

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

cZeros({u_ • v_−u_−v_,$v\_^2$+u_},{u_,v_})

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

Extraction ligne 2 :

ans[2]

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

cZeros({u_ • v_−u_−(c_ • v_),$v\_^2$+u_},{u_,v_})

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

**cZeros()** **Catalogue > ∫Σ > Algèbre**

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

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

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

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

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

cZeros({u_ · v_ − u_ − v_, v_^2 + u_}, {u_, v_, w_})

$$\begin{bmatrix} 0 & 0 & c9 \\ \frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i & \frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i & c9 \\ \frac{1}{2}+\frac{\sqrt{3}}{2}\cdot i & \frac{1}{2}-\frac{\sqrt{3}}{2}\cdot i & c9 \end{bmatrix}$$

cZeros({u_ + v_ − e^w_, u_ − v_ − *i*}, {u_, v_})

$$\left[\frac{e^{w\_}}{2}+1/2\cdot i \quad \frac{e^{w\_}-i}{2}\right]$$

cZeros({e^z_ − − w_, w_ − z_^2}, {w_, z_})

**[**.494… ⁻.703…**]**

cZeros({e^z_ − − w_, w_ − z_^2}, {w_, z_=1+ *i*})

[.149…+4.89…⁻*i* 1.588…+1.540… • *i*]

# *D*

**▸DD** **Catalogue > ∫Σ > Angle**

*Nombre* **▸DD** ⇒ *valeur*
*Liste1* **▸DD** ⇒ *liste*
*Matrice1* **▸DD** ⇒ *matrice*

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

En mode Angle en degrés :

1.5° ▸DD

1.5°

45° 22'14.3'' ▸DD

45.370…°

{45° 22'14.3'',60° 0'0''} ▸DD

{45.370…,60}°

En mode Angle en grades :

1▸DD

(9/10)°

En mode Angle en radians :

1.5 ▸DD

85.9437°

## ▸Dec

**Catalogue > ∫Σ > Base**

*Entier1* **▸Dec** ⇒ *entier*

Convertit *Entier1* en un nombre décimal (base 10). Toute entrée binaire ou hexadécimaledoit 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 32 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▸Dec
                                19
0h1F▸Dec
                                31
```

## Define

**Catalogue >**

**Define** *nomFonc*(*Arg1*, *Arg2*, ...) = *expression*

Définit *nomFonc* comme une fonction définie par l'utilisateur. Vous pouvez ensuite utiliser *nomFonc*(), comme n'importe quelle autre fonction prédéfinie. La fonction évalue *expression* en utilisant les arguments fournis, puis donne le résultat obtenu.

*nomFonc* ne peut pas être le nom d'une variable système ou d'une fonction prédéfinie.

Les noms des arguments sont des paramètres substituables ; vous ne devez pas utiliser les mêmes noms comme arguments lorsque vous utilisez la fonction.

**Remarque :** cette utilisation de Define est équivalente à celle de l'instruction :

*expression* → *nomFonc*(*Arg1*,*Arg2*).

Cette commande peut également être utilisée pour définir des variables simples ; par exemple, **Define** a=3.

```
Define g(x,y)=2x−3y
                                Done
g(1,2)
                                -4
1 → a : 2 → b : g(a,b)
                                -4
Define h(x)=when(x<2,2x−3, -2x+3)
                                Done
h(-3)
                                -9
h(4)
                                -5
Define eigenvl(a)=
cZeros(det(identity(dim(a)
[1])−x • a),x)
                                Done
eigenvl([-1,2;4,3])
```

$$\left\{\frac{2\cdot\sqrt{3}-1}{11}\ \ \frac{-(2\cdot\sqrt{3}+1)}{11}\right\}$$

**Define** *nomFonc*(*Arg1*, *Arg2*, ...) = **Func**
*Bloc*
**EndFunc**

Cette utilisation de **Define** s'apparente à la précédente, excepté que dans ce cas, la fonction définie par l'utilisateur *nomFonc*() permet d'exécuter plusieurs instructions (bloc).

*Bloc* peut correspondre à une ou plusieurs instructions, séparées par un « : ». *Bloc* peut également contenir des expressions et des instructions (comme **If**, **Then**, **Else** et **For**). Cela permet à la fonction *nomFonc*() d'utiliser l'instruction **Return** pour donner un résultat spécifique.

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

## DelVar

**Catalogue > ∫Σ > Divers**

**DelVar** *Var1*[**,** *Var2*] [**,** *Var3*] ...
Supprime de la mémoire les variables spécifiées.

2 ➞ a

2

$(a+2)^2$

16

DelVar a

Done

$(a+2)^2$

$(a + 2)^2$

## deSolve()

**Catalogue > ∫Σ > Analyse**

**deSolve(***ode1OrdreOu2Ordre***,** *varIndépendante***,** *varDépendante***)**
⇒ *une solution générale*

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

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

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

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

deSolve(y''+2y'+y=$x^2$,x,y)

$$y=(c1\cdot x+c2)\cdot e^{-x}+x^2-4\cdot x+6$$

right(ans) ➞ temp

$$(c1\cdot x+c2)\cdot e^{-x}+x^2-4\cdot x+6$$

d(temp,x,2)+2 · d(temp,x)+temp−$x^2$

0

DelVar temp

Done

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

deSolve(y'=(cos(y))$^2$ · x,x,y)

$$\tan(y)=\frac{x^2}{2}+c3$$

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

solve(ans,y)

$$y=\tan^{-1}\left(\frac{x^2+2\cdot c3}{2}\right)+n1\cdot\pi$$

ans|c3=c−1 and n1=0

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

**deSolve(***ode1Ordre* **and** *ConditionInitiale*, *varIndépendante*, *varDépendante***)**
⇒ *une solution particulière*

Donne une solution particulière qui satisfait à la fois *ode1Ordre* et *ConditionInitiale*. Cette méthode est généralement plus simple que celle qui détermine une solution générale car elle substitue des valeurs initiales, calcule la constante arbitraire, puis substitue cette valeur dans dans la solution générale.

*ConditionInitiale* est une équation de type :

*varDépendante* (*valeurIndépendanteInitiale*) = *ValeurDépendanteInitiale*

*varIndépendanteInitiale* et *varDépendanteInitiale* peuvent être des variables comme x0 et y0, sans valeur enregistrée. La différentiation implicite peut aider à vérifier les solutions implicites.

sin(y)=(y • e^x+cos(y))y' → ode

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

deSolve(ode and y(0)=0,x,y) → soln

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

soln|x=0 and y=0

true

d(right(eq)−left(eq),x)/
(d(left(eq)−right(eq),y))
→ impdif(eq,x,y)

Done

ode|y'=impdif(soln,x,y)

true

DelVar ode,soln

Done

**deSolve(***ode2Ordre* **and** *ConditionInitiale1* **and** *ConditionInitiale2*, *varIndépendante*, *varDépendante***)**
⇒ *une solution particulière*

Donne une solution particulière qui satisfait *ode2Ordre* et qui a une valeur spécifique de la variable dépendante et sa dérivée première en un point.

Pour *ConditionInitiale1*, utilisez :

*varDépendante* (*valeurIndépendanteInitiale*) = *ValeurDépendanteInitiale*

Pour *ConditionInitiale2*, utilisez :

*varDépendante* (*ValeurIndépendanteinitiale*) = *ValeurInitialeDérivée1*

deSolve(y''=y^-1/2 and y(0)=0 and y'(0)=0,t,y)

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

solve(ans,y)

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

**deSolve(***ode2Ordre* **and** *ConditionBorne1* **and** *ConditionBorne2*, *varIndépendante*, *varDépendante***)**
⇒ *une solution particulière*

Donne une solution particulière qui satisfait *ode2Ordre* et qui a des valeurs spécifiques en deux points différents.

deSolve(w''−2w'/x+(9+2/x^2)w=
x • e^x and w(π/6)=0 and w(π/3)=0,x,w)

$$w=\frac{e^{\frac{\pi}{3}} \cdot x \cdot \cos(3 \cdot x)}{10}$$
$$-\frac{e^{\frac{\pi}{6}} \cdot x \cdot \sin(3 \cdot x)}{10}+\frac{x \cdot e^x}{10}$$

## det()

Catalogue > ∫Σ > Matrice

**det(***matriceCarrée***[, *Tol*]) ⇒ *expression***
Donne le déterminant de *matriceCarrée*.
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 Exact/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 • **max(dim(***matriceCarrée***))** • **rowNorm(***matriceCarrée***)**

det($\begin{bmatrix} a & b \\ c & d \end{bmatrix}$)

$a \cdot d - b \cdot c$

det($\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$)

$-2$

det(identity(3) $-x \cdot \begin{bmatrix} 1 & -2 & 3 \\ -2 & 4 & 1 \\ -6 & -2 & 7 \end{bmatrix}$)

$-(98 \cdot x^3 - 55 \cdot x^2 + 12 \cdot x - 1)$

$\begin{bmatrix} 1E20 & 1 \\ 0 & 1 \end{bmatrix} \rightarrow$ mat1

$\begin{bmatrix} 1E20 & 1 \\ 0 & 1 \end{bmatrix}$

det(mat1)

0

det(mat1,.1)

1.E20

## diag()

Catalogue > ∫Σ > Matrice

**diag(***Liste***) ⇒ *matrice***
**diag(***matriceLigne***) ⇒ *matrice***
**diag(***mariceColonne***) ⇒ *matrice***
Donne une matrice diagonale, ayant sur sa diagonale principale les éléments de la liste passée en argument.

**diag(***matriceCarrée***) ⇒ *matriceLigne***
Donne une matrice ligne contenant les éléments de la diagonale principale de *matriceCarrée*.
*matriceCarrée* doit être carrée.

diag({2,4,6})

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

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

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

diag(ans)

[4 2 9]

## dim()

Catalogue > ∫Σ > Matrice

**dim(***Liste***) ⇒ *entier***
Donne le nombre d'éléments de *Liste*.

**dim(***Matrice***) ⇒ *liste***
Donne les dimensions de la matrice sous la forme d'une liste à deux éléments {lignes, colonnes}.

**dim(***Chaîne***) ⇒ *entier***
Donne le nombre de caractères contenus dans *Chaîne*.

dim({0,1,2})

3

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

{3,2}

dim("Hello")

5

dim("Hello"&" there")

11

## ▶DMS

**Catalogue > ∫Σ > Angle**

*Expression* **▶DMS**
*Liste* **▶DMS**
*Matrice* **▶DMS**
Interprète l'argument comme un angle et affiche le nombre DMS équivalent (DDDDDD°MM'SS.ss''). Voir °, ', '' page 163 pour le détail du format DMS (degrés, minutes, secondes).

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

En mode Angle en degrés :

45.371 ▶DMS

45° 22'15.6''

{45.371,60} ▶DMS

{45° 22'15.6'',60°}

## dotP()

**Catalogue > ∫Σ > Matrice**

**dotP(***Liste1***,** *Liste2***)** ⇒ *expression*
Donne le produit scalaire de deux listes.

dotP({a,b,c},{d,e,f})

a•d + b•e + c•f

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

17

**dotP(***Vecteur1***,** *Vecteur2***)** ⇒ *expression*
Donne le produit scalaire de deux vecteurs.
Les deux vecteurs doivent être de même type (ligne ou colonne).

dotP([a b c],[d e f])

a•d + b•e + c•f

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

32

# *E*

## e^()

**Touches** ctrl

**e^(***Expression1***)** ⇒ *expression*
Donne e élevé à la puissance de *Expression1*.

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

$e^1$

e

$e^{1.}$

2.718...

$e^{(3)^2}$

$e^9$

Vous pouvez entrer un nombre complexe sous la forme polaire re^(*i* θ). N'utilisez toutefois cette forme qu'en mode Angle en radians ; elle provoque une erreur de domaine en mode Angle en degrés ou en grades.

**e^(***Liste1***)** ⇒ *liste*
Donne e élevé à la puissance de chaque élément de *Liste1*.

$e^{\{1,1.,0,.5\}}$

{e,2.718...,1,1.648...}

**e^(***matriceCarrée1***)** ⇒ *matriceCarréex*
Donne l'exponentielle de *matriceCarrée1*. Cette utilisation est différente du calcul de e 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.

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

$$\begin{bmatrix} 782.209 & 559.617 & 456.509 \\ 680.546 & 488.795 & 396.521 \\ 524.929 & 371.222 & 307.879 \end{bmatrix}$$

## eigVc()

**Catalogue > ∫Σ > Matrice**

**eigVc(***matriceCarrée***)** ⇒ *matrice*

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

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

*matriceCarrée* subit d'abord de transformations de similarité de sorte que les normes ligne et colonne se rapprochent autant que possible de la même valeur. La *matriceCarrée* est ensuite réduite à la forme de Hessenberg supérieure et les vecteurs propres calculés via une factorisation de Schur.

En mode Format complexe Rectangulaire :

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

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

eigVc(m1)

$$\begin{bmatrix} -.800\ldots & .767\ldots & .767\ldots \\ .484\ldots & .573\ldots+.052\ldots\cdot i & .573\ldots-.052\ldots\cdot i \\ .352\ldots & .262\ldots+.096\ldots\cdot i & .262\ldots-.096\ldots\cdot i \end{bmatrix}$$

## eigVl()

**Catalogue > ∫Σ > Matrice**

**eigVl(***matriceCarrée***)** ⇒ *liste*

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

*matriceCarrée* subit d'abord de transformations de similarité de sorte que les normes ligne et colonne se rapprochent autant que possible de la même valeur. 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 :

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

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

eigVl(m1)

{−4.409…,2.204…+.763…·*i*,2.204…−.763…·*i*}

**Else**

Voir If, page 89.

## ElseIf

**Barre d'outils Calculs > {□ □} > Structures de contrôle**

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

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

**EndFor**

Voir For, page 85.

**EndFunc**

Voir Func, page 87.

**EndIf**

Voir If, page 89.

**EndLoop**

Voir Loop, page 100.

**EndWhile**
Voir While, page 146.

## exact()

**Catalogue > ∫Σ > Nombre**

**exact(***Expression1* **[,** *Tol***]) ⇒** *expression*
**exact(***Liste1* **[,** *Tol***]) ⇒** *liste*
**exact(***Matrice1* **[,** *Tol***]) ⇒** *matrice*

Utilise le mode Exact quel que soit le mode Exact/Approché en cours d'utilisation pour donner, si possible, la valeur formelle de l'argument.

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

exact(.25)

1/4

exact(.333333)

$\frac{333333}{1000000}$

exact(.33333,.001)

1/3

exact(3.5x+y)

$\frac{7 \cdot x}{2} + y$

exact({.2,.33,4.125})

$\{1/5\ \frac{33}{100}\ 33/8\}$

## Exit

**Barre d'outils Calculs > {▯} > Structures de contrôle**

**Exit**

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

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

Liste des fonctions :

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

Done

g()

21

## exp▸liste()

**Catalogue >**

**exp▸list(***Expression***,***Var***) ⇒** *liste*

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

**Remarque : exp4list()** n'est pas nécessaire avec les fonctions **zeros** et **cZeros()** étant donné que celles-ci donnent directement une liste de solutions.

solve($x^2-x-2=0$,x)

x=-1 or x=2

exp▸list(solve($x^2-x-2=0$,x),x)

{-1,2}

**expand(***Expression1* [**,** *Var*]**)** ⇒ *expression*

**expand(***Liste1* [**,***Var*]**)** ⇒ *liste*

**expand(***Matrice1* [**,***Var*]**)** ⇒ *matrice*

**expand(***Expression1***)** développe *Expression1* en fonction de toute ses variables. Le développement est polynomial pour les expressions polynomiales et partiel pour les expressions rationnelles.

L'objectif de **expand()** est de transformer *Expression1* en une somme et/ou une différence de termes simples. Par opposition, l'objectif de **factor()** est de transformer *Expression1* en un produit et/ou un quotient de facteurs simples.

expand((x+y+1)^2)

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

expand((x^2−x+y^2−y)/(x^2 · y^2−x^2 · y−x · y^2+x · y))

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

**expand(***Expression1***,***Var***)** développe *Expression1* en fonction de *Var*. Les mêmes puissances de *Var* sont regroupées. Les termes et leurs facteurs sont triés, *Var* étant la variable principale. Une factorisation ou un développement incident des coefficients regroupés peut se produire. L'utilisation de *Var* permet de gagner du temps, de la mémoire et de l'espace sur l'écran tout en facilitant la lecture de l'expression.

expand((x+y+1)^2,y)

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

expand((x+y+1)^2,x)

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

expand((x^2−x+y^2−y)/(x^2 · y^2−x^2 · y−x · y^2+x · y),y)

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

expand(ans,x)

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

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

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

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

expand((x^3+x^2−2)/(x^2−2))

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

expand(ans,x)

$$\frac{1}{x - \sqrt{2}} + \frac{1}{x + \sqrt{2}} + x + 1$$

**expand(***Expression1***,**[*Var*]**)** distribue également des logarithmes et des puissances fractionnaires indépendamment de *Var*. Pour une plus grande distribution des logarithmes et des puissances fractionnaires, l'utilisation de contraintes d'inéquation peut s'avérer nécessaire pour garantir l'obtention de certains facteurs non négatifs.

ln(2x • y)+√(2x • y)

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

expand(ans)

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

**expand(***Expression1***,** [*Var*]**)** distribue également des valeurs absolues, **sign()** et des exponentielles, indépendamment de *Var*.

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

expand(ans)|y>=0

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

sign(x • y)+abs(x • y)+ e^(2x+y)

$e2 \cdot x+y + \text{sign}(x \cdot y) + |x \cdot y|$

expand(ans)

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

## expr()

**Catalogue > ∫Σ > Chaîne**

**expr(***Chaîne***)** ⇒ *expression*

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

```
expr("1+2+x^2+x")
                                  x^2 + x + 3
expr("expand((1+x)^2)")
                                  x^2 + 2 • x + 1
"Define cube(x)=x^3" → funcstr
                                  "Define cube(x)=x^3"
expr(funcstr)
                                  Done
cube(2)
                                  8
```

## ExpReg

**Catalogue > ∫Σ > Statistiques**

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

Calcule un ajustement exponentiel. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.

*X* représente la liste des valeurs de x.

*Y* représente la liste des valeurs de y.

*Fréq* représente la liste des fréquences.

*Catégorie* représente les numéros de catégories.

*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| **a,b** | Coefficients d'ajustement : $y = a \cdot (b)^x$. |
| **$r^2$** | Coefficient de détermination |
| **r** | Coefficient de corrélation linéaire |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = $y - a \cdot (b)^x$ |
| **ResidTrans** | Valeurs résiduelles associées à l'ajustement linéaire des données transformées |
| **RegEqn** | Équation d'ajustement : $a \cdot (b)^x$. |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

# F

## factor()

**Catalogue > ∫Σ > Algèbre**

**factor(***Expression1***[, *Var*])** ⇒ *expression*
**factor(***Liste1***[, *Var*])** ⇒ *liste*
**factor(***Matrice1***[, *Var*])** ⇒ *matrice*

factor(*Expression1*) factorise *Expression1* en fonction de l'ensemble des variables associées sur un dénominateur commun.

La factorisation de *Expression1* décompose l'expression en autant de facteurs rationnels linéaires que possible sans introduire de nouvelles sous-expressions non réelles. Cette alternative peut s'avérer utile pour factoriser l'expression en fonction de plusieurs variables.

factor(*Expression1*, *Var*) factorise *Expression1* en fonction de la variable *Var*.

La factorisation de *Expression1* décompose l'expression en autant de facteurs réels possible qui sont linéaires dans *Var*, même si cela introduit des constantes irrationnelles ou des sous-expressions qui sont irrationnelles dans d'autres variables.

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

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

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

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

**factor(***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.

**Remarque :** pour arrêter un calcul, appuyez sur .

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

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

factor($x^2+1$)
$x^2+1$

factor($x^2-4$)
$(x-2) \cdot (x+2)$

factor($x^2-3$)
$x^2-3$

factor($x^2-a$)
$x^2-a$

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

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

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

factor($x^5+4x^4+5x^3-6x-3$)
$x^5+4 \cdot x^4+5 \cdot x^3-6 \cdot x-3$

factor(ans,x)
$(x-.964\ldots) \cdot (x+.611\ldots) \cdot (x+2.125\ldots) \cdot (x^2+2.227\ldots \cdot x+2.392\ldots)$

factor(152417172689)
123457 • 1234577

isPrime(152417172689)
faux

## FCdf()

**Catalogue > ∫Σ > Probabilité**

**FCdf(***lowerBound*,*upperBound*,*dfNumer*,*dfDenom***)**
**FCdf(***lowerBound*,*upperBound*,*dfNumer*,*dfDenom***)**
Calcule la fonction de répartition de la loi F entre *lowerBound* et *upperBound* à *dfNumer* (degrés de liberté) et *dfDenom*.

## Fill

**Catalogue > ∫Σ > Matrice**

**Fill** *Expression*, *VarMatrice* ⇒ *matrice*
Remplace chaque élément de la variable *VarMatrice* par *Expression*. *VarMatrice* doit avoir été définie.

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

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

Fill 1.01,amatrx

Done

amatrx

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

**Fill** *Expression*, *VarListe* ⇒ *liste*
Remplace chaque élément de la variable *VarListe* par *Expression*. *VarListe* doit avoir été définie.

{1,2,3,4,5} → alist

{1,2,3,4,5}

Fill 1.01,alist

Done

alist

{1.01,1.01,1.01,1.01,1.01}

## floor()

**Catalogue > ∫Σ > Nombre**

**floor(***Expression1***)** ⇒ *entier*
Donne le plus grand entier ≤ à l'argument (partie entière). Cette fonction est comparable à **int()**.
L'argument peut être un nombre réel ou un nombre complexe.

floor(-2.14)

-3.

**floor(***Liste1***)** ⇒ *liste*
**floor(***Matrice1***)** ⇒ *matrice*
Donne une liste ou une matrice de la partie entière de chaque élément.
**Remarque :** voi aussi **ceiling()** et **int()**.

floor({3/2,0,-5.3})

{1,0,-6.}

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

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

## fMax()

**Catalogue > ∫Σ > Analyse**

**fMax(***Expression*, *Var***)** ⇒ *Expression booléenne*
Donne une expression booléenne spécifiant les valeurs possibles de *Var* pour laquelle *Expression* est à son maximum ou détermine au moins sa limite supérieure.

fMax(1−(x−a)$^2$−(x−b)$^2$,x)

$x = \frac{a+b}{2}$

fMax(.5x$^3$−x−2,x)

x = ∞

## fMax()

**Catalogue > ∫Σ > Analyse**

Utilisez l'opérateur « | » pour préciser l'intervalle de recherche et/ou spécifier le signe des autres variables non définies.

Avec le réglage Approché du mode Exact/Approché, l'utilisation de **fMax()** permet de rechercher de façon itérative un maximum local approché. C'est souvent plus rapide, surtout si vous utilisez l'opérateur « | » pour limiter la recherche à un intervalle relativement réduit qui contient exactement un maximum local.

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

fMax(.5x$^3$−x−2,x)|x≤1

x = ⁻.816...

## fMin()

**Catalogue > ∫Σ > Analyse**

**fMin(***Expression, Var***)** ⇒ *Expression booléenne*

Donne une expression booléenne spécifiant les valeurs possibles de *Var* pour laquelle *Expression* est à son minimum ou déterminer au moins sa limite inférieure.

Utilisez l'opérateur « | » pour préciser l'intervalle de recherche et/ou spécifier le signe des autres variables non définies.

Avec le réglage Approché du mode Exact/Approché, l'utilisation de **fMin()** permet de rechercher de façon itérative un minimum local approché. C'est souvent plus rapide, surtout si vous utilisez l'opérateur « | » pour limiter la recherche à un intervalle relativement réduit qui contient exactement un minimum local.

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

fMin(1−(x−a)$^2$−(x−b)$^2$,x)

x = ∞ ou x = ⁻∞

fMin(.5x$^3$−x−2,x)|x≥1

x = 1

## For

**Barre d'outils Calculs > {▯/▯} > Structures de contrôle**

**For** *Var*, *Début*, *Fin* [, *Incrément*]
*Bloc*
**EndFor**

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

*Var* ne doit pas être une variable système.

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

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

Segment de la fonction :

```
  ⋮
:0 → sommetemp : 1 → incrément
:For i,1,100,incrément
:  sommetemp+i → sommetemp
:EndFor
  ⋮
```

Contenu de sommetemp après exécution : 5050

Contenu de sommetemp lorsque la valeur de incrément est fixée à 2 :
2500

## format()

Catalogue > ∫Σ > Chaîne

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

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

*Expression* doit avoir une valeur numérique. *chaîneFormat* doit être une chaîne du type : « F[n] », « S[n] », « E[n] », « G[n][c] », où [ ] identifie les parties facultatives.

F[n] : format Fixe. n correspond au nombre de chiffres à afficher après le séparateur décimal.

S[n] : format Scientifique. n correspond au nombre de chiffres à afficher après le séparateur décimal.

E[n] : format Ingénieur. n correspond au nombre de chiffres après le premier chiffre significatif. L'exposant est ramené à un multiple de trois et le séparateur décimal est décalé vers la droite de zéro, un ou deux chiffres.

G[n][c] : identique au format Fixe, mais sépare également les chiffres à gauche de la base par groupes de trois. c spécifie le caractère séparateur des groupes et a pour valeur par défaut la virgule. Si c est un point, la base s'affiche sous forme de virgule.

[Rc] : tous les formats ci-dessus peuvent se voir ajouter en suffixe l'indicateur de base Rc, où c correspond à un caractère unique spécifiant le caractère à substituer au point de la base.

```
format(1.234567,"f3")
                    "1.235"
format(1.234567,"s2")
                   "1.23E0"
format(1.234567,"e3")
                  "1.235E0"
format(1.234567,"g3")
                    "1.235"
format(1234.567,"g3")
                  "1,234.567"
format(1.234567,"g3,r:")
                    "1:235"
```

## fPart()

Catalogue > ∫Σ > Nombre

**fPart(***Expression1***)** ⇒ *expression*
**fPart(***Liste1***)** ⇒ *liste*
**fPart(***Matrice1***)** ⇒ *matrice*

Donne la partie fractionnaire de l'argument.

Dans le cas d'une liste ou d'une matrice, donne les parties fractionnaires des éléments.

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

```
fPart(-1.234)
                    -.234
fPart({1, -2.3, 7.003})
                  {0,-.3,.003}
```

## FPdf()

Catalogue > ∫Σ > Probabilité

**FPdf(***ValX*,*dfNumer*,*dfDenom***)**
**FPdf(***ValX*,*dfNumer*,*dfDenom***)**

Calcule la densité de la loi F pour *ValX* à *dfNumer* (degrés de liberté) et *dfDenom*.

## FTest_2Samp

Catalogue > ∫Σ > Statistiques

**FTest_2Samp** [*Liste1*,*Liste2*[,*Fréq1*[,*Fréq2*[,*Hypoth*]]]
**FTest_2Samp** [*Liste1*,*Liste2*[,*Fréq1*[,*Fréq2*[,*Hypoth*]]]
(Entrée de liste de données)

Effectue un F°Test sur deux échantillons. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

*Hypoth* > 0 correspond à Ha : σ1 > σ2

*Hypoth* = 0 correspond à Ha : σ1 ≠ σ2 (par défaut)

*Hypoth* < 0 correspond à Ha : σ1 < σ2

| Variable de sortie | Description |
|---|---|
| F | Statistique [Y-VARS] estimée pour la séquence de données. |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **dfNumer** | numérateur degrés de liberté = n1-1. |
| **dfDenom** | dénominateur degrés de liberté = n2-1. |
| **σx1, σx2** | Écarts types de population d'échantillon des séquences de données dans *Liste 1* et *Liste 2*. |
| **x1_bar**<br>**x2_bar** | Moyenne de population d'échantillon des séquences de données dans *Liste 1* et *Liste 2*. |
| **n1, n2** | Taille des échantillons. |

**Func** **Barre d'outils Calculs > > Structures de contrôle**

**Func**
*Bloc*
**EndFunc**
Requise comme première instruction dans une définition de fonction composée de plusieurs instructions.

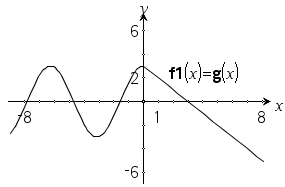
*Bloc* peut correspondre à une ou plusieurs instructions, séparées par un « : ».

Définition d'une fonction par morceaux :

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

Done

Résultat de la représentation graphique de g(x)




# *G*

**gcd()** **Catalogue > > Nombre**

**gcd(***Nombre1*, *Nombre2***)** ⇒ *expression*
Donne le plus grand commun diviseur aux deux arguments. Le **gcd** de deux fractions correspond au **gcd** de leur numérateur divisé par le **lcm** de leur dénominateur.

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

gcd(18,33)

3

**gcd(***Liste1*, *Liste2***)** ⇒ *liste*
Donne la liste des plus grands communs diviseurs des éléments correspondants de *Liste1* et *Liste2*.

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

{3,7,1}

**gcd(***Matrice1*, *Matrice2***)** ⇒ *matrice*
Donne la matrice des plus grands communs diviseurs des éléments correspondants de *Matrice1* et *Matrice2*.

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

$$\begin{bmatrix} 2 & 4 \\ 6 & 8 \end{bmatrix}$$

## geomCdf()

**Catalogue > ∫Σ > Probabilité**

**geomCdf(***p*, *lowerBound*, *upperBound***)**
Calcule la la fonction de répartition de la loi géométrique entre *lowerBound* et *upperBound* en fonction de la probabilité de réussite *p* spécifiée.

## geomPdf()

**Catalogue > ∫Σ > Probabilité**

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

## getDenom()

**Catalogue > ∫Σ > Algèbre**

**getDenom(***Expression1***)** ⇒ *expression*
Transforme *Expression1* en une expression dotée d'un dénominateur commun réduit, puis en donne le dénominateur.

```
getDenom((x+2)/(y−3))
                    y −3
getDenom(2/7)
                      7
getDenom(1/x+(y²+y)/y²)
                    x • y
```

## getNum()

**Catalogue > ∫Σ > Algèbre**

**getNum(***Expression1***)** ⇒ *expression*
Transforme *Expression1* en une expression dotée d'un dénominateur commun réduit, puis en donne le numérateur.

```
getNum((x+2)/(y−3))
                    x + 2
getNum(2/7)
                      2
getNum(1/x+1/y)
                    x + y
```

## Goto

**Barre d'outils Calculs> {□} > Structures de contrôle**

**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**.

```
Segment de la fonction :
  ⋮
:0 → temp
:1 → i
:Lbl TOP
: temp+i → temp
: If i<10 Then
:   i+1 → i
:   Goto TOP
: EndIf
: Return temp
  ⋮
```

### ▶Grad

**Catalogue > ∫Σ > Angle**

*Expression1* ▶ **Grad** ⇒ *expression*

Convertit *Expression1* en une mesure d'angle en grades.

En mode Angle en degrés :

1.5 ▶Grad

1.66667$^{G}$

En mode Angle en radians :

1.5 ▶Grad

95.493$^{G}$

# H

### ▶Hex

**Catalogue > ∫Σ > Base**

*Entier1* ▶**Hex** ⇒ *entier*

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 32 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 8 chiffres.

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

Si vous entrez un nombre dont le codage binaire signé dépasse 32 bits, il est ramené à l'aide d'une congruence dans la plage appropriée.

256 ▶Hex

0h100

0b111100001111 ▶Hex

0hF0F

# I

### identity()

**Catalogue > ∫Σ > Matrice**

**identity(***Entier***)** ⇒ *matrice*

Donne la matrice identité (matrice unité) de dimension *Entier*.

*Entier* doit être un entier positif.

identity(4)

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

### If

**Barre d'outils Calculs> {□ □} > Structures de contrôle**

**If** *Expression booléenne Instruction*

**If** *Expression booléenne* **Then**
*Bloc*
**EndIf**

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

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

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

Segment de la fonction :

```
  ⋮
:If x<0
:  Return x²
:EndIf
  ⋮
```

## If

**Barre d'outils Calculs>** {□} **> Structures de contrôle**

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

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

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

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

Segment de la fonction :

```
  :
:If x<0 Then
:  Return ⁻x
:Else
:  Return x
:EndIf
  :
```

**If** *Expression booléenne1* **Then**
*Bloc1*
**ElseIf** *Expression booléenne2* **Then**
*Bloc2*
:
**ElseIf** *Expression booléenne N* **Then**
*Bloc N*
**EndIf**

Permet de traiter les conditions multiples. Si *Expression booléenne1* passe le test de condition, exécute *Bloc1*. Si *Expression booléenne1* ne passe pas le test de condition, calcule *Expression booléenne2*, etc.

Exemple de fonction :

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

## imag()

**Catalogue >** ∫Σ **> Complexe**

**imag(***Expression1***)** ⇒ *expression*
Donne la partie imaginaire de l'argument.

**Remarque :** toutes les variables non définies sont considérées comme réelles. Voir aussi **real()**.

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

**imag(***Liste1***)** ⇒ *liste*
Donne une liste des parties imaginaires des éléments.

```
imag({⁻3,4⁻i,i})
                {0,⁻1,1}
```

**imag(***Matrice1***)** ⇒ *matrice*
Donne une matrice des parties imaginaires des éléments.

```
imag([a,b;ic,id])
                [0 0]
                [c d]
```

## ImpDif()

**Catalogue >** ∫Σ **> Analyse**

**ImpDif(***Équation***,** *varIndépendante***,** *varDépendante***[,***Ordr* **])** ⇒ *expression*
où la valeur par défaut de l'argument *Ordre* est 1.

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

```
impDif(x²+y²=100,x,y)
                  -x/y
```

**Indirection**

Voir **#()**, page 161.

**inString()** **Catalogue > ∫Σ > Chaîne**

**inString(***chaîneSrce*, *sousChaîne*[, *Début*]**)** ⇒ *entier*
Donne le rang du caractère de la chaîne *chaîneSrce* où commence la première occurrence de *sousChaîne*.
*Début*, s'il est utilisé, indique le point de départ de la recherche dans *chaîneSrce*. Par défaut, la recherche commence à partir du premier caractère de *chaîneSrce*).
Si *chaîneSrce* ne contient pas *sousChaîne* ou si *Début* est > à la longueur de *ChaîneSrce*, on obtient zéro.

```
inString("Hello there","the")
                                7
inString("ABCEFG","D")
                                0
```

**int()** **Catalogue > ∫Σ > Analyse**

**int(***Expression***)** ⇒ *entier*
**int(***Liste1***)** ⇒ *liste*
**int(***Matrice1***)** ⇒ *matrice*
Donne le plus grand entier inférieur ou égal à l'argument. Cette fonction est identique à **floor()** (partie entière).
L'argument peut être un nombre réel ou un nombre complexe.
Dans le cas d'une liste ou d'une matrice, donne la partie entière de chaque élément.

```
int(-2.5)
                               -3.
int([-1.234,0,0.37])
                           [-2. 0 0.]
```

**intDiv()** **Catalogue >**

**intDiv(***Nombre1***,** *Nombre2***)** ⇒ *integer*
**intDiv(***Liste1***,** *Liste2***)** ⇒ *liste*
**intDiv(***Matrice1*, *Matrice2***)** ⇒ *matrice*
Donne le quotient dans la division euclidienne de argument 1 par argument 2.
Dans le cas d'une liste ou d'une matrice, donne le quotient de argument 1 par argument 2 pour chaque paire d'éléments.

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

**integrate**

Voir ∫(), page 160.

**invχ²()** **Catalogue > ∫Σ > Probabilité**

**invχ²(***Area*,*df***)**
**invchi2(***Area*,*df***)**
Calcule l'inverse de la fonction de répartition de la loi $\chi^2$ (Khi2) de degré de liberté *df* en un point donné (*Area*).

**invF** **Catalogue > ∫Σ > Probabilité**

**invF(***Area*,*dfNumer*,*dfDenom***)**
**invF(***Area*,*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 (*Area*).

## invNorm()

Catalogue > ∫Σ > Probabilité

**invNorm(***area*[,μ,σ]**)**

Calcule l'inverse de la fonction de répartition d'une loi normale de paramètres μ et σ en un point donné (area).

## invt()

Catalogue > ∫Σ > Probabilité

**invt(***Area*,*df***)**

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

## iPart()

Catalogue > ∫Σ > Nombre

**iPart(***Nombre***)** ⇒ *entier*
**iPart(***Liste1***)** ⇒ *liste*
**iPart(***Matrice1***)** ⇒ *matrice*

Donne l'argument moins sa partie fractionnaire.

Dans le cas d'une liste ou d'une matrice, applique la fonction à chaque élément.

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

```
iPart(-1.234)
                                  -1.
iPart({3/2,-2.3,7.003})
                                  {1,-2.,7.}
```

## isPrime()

Catalogue > ∫Σ > Test

**isPrime(***Nombre***)** ⇒ *Expression booléenne constante*

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

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

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

```
IsPrime(5)
                                  true
IsPrime(6)
                                  false
```

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

```
Define nextPrim(n)=Func:Loop:
n+1 → n:if isPrime(n):return n:
EndLoop:EndFunc
                                  Done
nextPrim(7)
                                  11
```

# 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.

Segment de la fonction :

```
:Lbl lbl1
:InputStr "Enter password", str1
:If str1≠password
: Goto lbl1
```

## lcm()

Catalogue > ∫Σ > Nombre

**lcm(***Nombre1***, *Nombre2***)** ⇒ *expression*
**lcm(***Liste1***, *Liste2***)** ⇒ *liste*
**lcm(***Matrice1***, *Matrice2***)** ⇒ *matrice*
Donne le plus petit commun multiple aux deux arguments. Le **lcm** de deux fractions correspond au **lcm** de leur numérateur divisé par le **gcd** de leur dénominateur. Le **lcm** de nombres fractionnaires en virgule flottante correspond à leur produit.
Pour deux listes ou matrices, donne les plus petits communs multiples des éléments correspondants.

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

## left()

Catalogue > ∫Σ > Liste

**left(***chaîneSource***[, *Nomb*])** ⇒ *chaîne*
Donne la chaîne formée par les *Nomb* premiers caractères de la chaîne *chaîneSource*.
Si *Nomb* est absent, on obtient *chaîneSource*.

**left(***Liste1***[, *Nomb*])** ⇒ *liste*
Donne la liste formée par les *Nomb* premiers éléments de *Liste1*.
Si *Nomb* est absent, on obtient *Liste1*.

**left(***Comparaison***)** ⇒ *expression*
Donne le membre de gauche d'une équation ou d'une inéquation.

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

## limit()

Catalogue > ∫Σ > Analyse

**limit(***Expression1***, *Var*, *Point* [,*Direction*])** ⇒ *expression*
**limit(***Liste1***, *Var*, *Point* [, *Direction*])** ⇒ *liste*
**limit(***Matrice1***, *Var*, *Point* [, *Direction*])** ⇒ *matrice*
Donne la limite recherchée.
*Direction* : négative=limite à gauche, positive=limite à droite, sinon=gauche et droite. (Si *Direction* est absent, la valeur par défaut est gauche et droite.)

```
limit(2x+3,x,5)
                                  13
limit(1/x,x,0,1)
                                   ∞
limit(sin(x)/x,x,0)
                                   1
limit((sin(x+h)-sin(x))/h,h,0)
                              cos(x)
limit((1+1/n)^n,n,∞)
                                   e
```

Les limites en +∞ et en -∞ négatif sont toujours converties en limites unilatérales.

Dans certains cas, **limit()** retourne lui-même ou undef (non défini) si aucune limite ne peut être déterminée. Cela ne signifie pas pour autant qu'aucune limite n'existe. undef signifie que le résultat est soit un nombre inconnu fini ou infini soit l'ensemble complet de ces nombres.

**limit()** utilisant des méthodes comme la règle de L'Hôpital, il existe des limites uniques que cette fonction ne permet pas de déterminer. Si *Expression1* contient des variables non définies autres que *Var*, il peut s'avérer nécessaire de les contraindre pour obtenir un résultat plus précis.

```
limit(a^x,x,∞)
                              undef
limit(a^x,x,∞)|a>1
                                   ∞
limit(a^x,x,∞)|a>0 and a<1
                                   0
```

Les limites peuvent être affectées par les erreurs d'arrondi. Dans la mesure du possible, n'utilisez pas cette fonction en mode Approché ni des nombres approchés lors du calcul de limites. Sinon, les limites normalement nulles ou infinies risquent de ne pas l'être.

## LinRegMx

**Catalogue > > Statistiques**

**LinRegMx** *X*,*Y*[,*Fréq*[,*Catégorie*,*Inclure*]]
Effectue une régression linéaire de type y=mx+b sur *X* et *Y* en utilisant la fréquence *Fréq*. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **m,b** | Coefficients d'ajustement : y = m • x+b. |
| **r²** | Coefficient de détermination |
| **r** | Coefficient de corrélation linéaire |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes : y - (m • x+b) |
| **RegEqn** | Équation d'ajustement : m • x+b |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

## LinRegBx

**Catalogue > > Statistiques**

**LinRegBx** *X*,*Y*[,*Fréq*[,*Catégorie*,*Inclure*]]
Effectue une régression linéaire de type y=a+bx sur *X* et *Y* en utilisant la fréquence *Fréq*. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **a,b** | Coefficients d'ajustement |
| **r²** | Coefficient de détermination |
| **r** | Coefficient de corrélation linéaire |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes : y - (a+b • x) |
| **RegEqn** | Équation d'ajustement : a+b • x |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

## LinRegtIntervals

**Catalogue > ∫Σ > Statistiques**

**LinRegtIntervals** *X*,*Y*[,*Fréq*[,**0**[,*CLevel*]]]
Pour pente

**LinRegtIntervals** *X*,*Y*[,*Fréq*[,**1**,*ValX*[,*CLevel*]]]
Pour prévision

Calcule l'intervalle t de régression linéaire pour un ajustement de la courbe des paires de points de données, où y(k) = a + b • x(k). Deux types d'intervalle sont disponible : Slope et Predict. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **a,b** | Estimations des paramètres de décalage et de pente pour l'ajustement de courbe |
| **df** | Dégrés de liberté |
| $r^2$ | Coefficient de détermination |
| **r** | Coefficient de corrélation linéaire |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes : y - (a+b • x) |
| **RegEqn** | Équation d'ajustement : a+b • x |

Pour les intervalles de type Slope uniquement

| Variable de sortie | Description |
|---|---|
| **[CLower,CUpper]** | Intervalle de confiance sur la pente contenant le CLevel de dist. |
| **ME** | Marge d'erreur de l'intervalle de confiance b de la pente |
| **SESlope** | SE Slope = s/sqrt(sum(x-xbar)$^2$) |
| **s** | Écart-type de l'erreur d'ajustement pour y - (a + b • x) |

Pour les intervalles de type Predict uniquement

| Variable de sortie | Description |
|---|---|
| **[CLower,CUpper]** | Intervalle de confiance sur la prévision contenant le CLevel de dist. |
| **ME** | Marge d'erreur de l'intervalle de confiance |
| **SE** | Erreur type de l'intervalle de confiance |
| **[LowerPred, UpperPred]** | Intervalle prévu sur la prévision contenant le CLevel de dist. |
| **MEPred** | Marge d'erreur de l'intervalle prévu |
| **SEPred** | Erreur type de l'intervalle prévu |
| $\hat{y}$ | a + b • XVal |

**LinRegtTest** **Catalogue > ∫Σ > Statistiques**

**LinRegtTest** *X*,*Y*[,*Fréq*[,*Hypoth*]]]
Calcule l'ajustement de la courbe de régression linéaire des paires de points de données, où y(k) = a + b • x(k), et teste les hypothèses nulles H0: b = 0 par rapport à l'une des alternatives suivantes :
*Hypoth* > 0 correspond à Ha : σ1 > σ2
*Hypoth* = 0 correspond à Ha : σ1 ≠ σ2 (par défaut)
*Hypoth* < 0 correspond à Ha : σ1 < σ2
Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **a, b** | Estimations des paramètres de décalage et de pente pour l'ajustement de courbe de régression |
| **df** | Dégrés de liberté |
| **s** | Écart-type de l'erreur d'ajustement pour y - (a + b • x) |
| **t** | T-Statistique pour la signification de la pente |
| **PVal** | Probabilité que l'hypothèse alternative est fausse |
| **r** | Coefficient de corrélation de régression linéaire |
| **$r^2$** | Coefficient de détermination |
| **SESlope** | SE Slope = s/sqrt(sum(x-x_bar)$^2$) |
| **Resid** | Valeurs résiduelles de l'ajustement linéaire |
| **RegEqn** | Équation d'ajustement : a + b • x |

**Δlist()** **Catalogue > ∫Σ > Liste**

**Δlist(***Liste1***)** ⇒ *liste*
Donne une 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.

Δlist({20,30,45,70}) {10,15,25}

**list▸mat()** **Catalogue > ∫Σ > Liste**

**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* (ligne une).
Si *Liste* ne comporte pas assez d'éléments pour la matrice, on complète par zéros.

list▸mat({1,2,3}) [1 2 3]

list▸mat({1,2,3,4,5},2)

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 0 \end{bmatrix}$$

## ▸ln

**Catalogue > ∫Σ > Base**

*Expression* **▸ln** ⇒ *expression*

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

Log(x)▸ ln

$$\frac{\ln(x)}{\ln(10)}$$

## ln()

**Touche** $e^x$ ln

**ln(***Expression1***)** ⇒ *expression*

**ln(***Liste1***)** ⇒ *liste*

Donne le logarithme népérien de l'argument.

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

ln(2.0)

.693...

En mode Format complexe Réel :

ln({⁻3,1.2,5})

Error: Non-real result

En mode Format complexe Rectangulaire :

ln({⁻3,1.2,5})

{ln(3) + π • *i*,.182...,ln(5)}

**ln(***matriceCarrée1***)** ⇒ *matriceCarrée*

Donne le logarithme népérien de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du logarithme népérien de chaque élément. Pour plus d'informations sur la méthode de calcul, 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 :

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

$$\begin{bmatrix} 1.831\ldots+1.734\ldots\cdot i & .009\ldots-1.490\ldots\cdot i & \ldots \\ .448\ldots-.725\ldots\cdot i & 1.064\ldots+.623\cdot i & \ldots \\ -.266\ldots-2.083\ldots\cdot i & 1.124\ldots+1.790\ldots\cdot i & \ldots \end{bmatrix}$$

## LnReg

**Catalogue > ∫Σ > Statistiques**

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

Calcule la régression logarithmique. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.

*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| **a,b** | Coefficients d'ajustement : y = a+b • ln(x) |
| **r²** | Coefficient de détermination |
| **r** | Coefficient de corrélation linéaire |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y−(a+b • ln(x)) |
| **ResidTrans** | Valeurs résiduelles associées à l'ajustement linéaire des données transformées |
| **RegEqn** | Équation d'ajustement : a+b • ln(x) |

| Variable de sortie | Description |
|---|---|
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

## Local

**Barre d'outils Calculs > {} > Structures de contrôle**

**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.

Liste des fonctions :

```
Define rollcount()=Func
: Local i
: 1 → i
: Loop
:   If rand(6)=6 and rand(6)=6
:     Goto End
:   i+1 → i
: EndLoop
:Lbl End
:Return i
:EndFunc
```

## log()

**Touche** $10^x$ log

**log(***Expression1***[,***Expression2***])** ⇒ *expression*
**log(***Liste1***[,***Expression2***])** ⇒ *liste*

Donne le logarithme de base *Expression2* de l'argument.
Dans le cas d'une liste, donne le logarithme de base *Expression2* des éléments.
Si *Expression2* est omis, la valeur de base 10 par défaut est utilisée.

$\log(2.0)$

.301...

$\text{Log}_4(2.0)$

.5

$\text{Log}_3(10) - \log_3(5)$

$\text{Log}_3(2)$

En mode Format complexe Réel :

log({-3,1.2,5})

Error: Non-real result

En mode Format complexe Rectangulaire :

log({-3,1.2,5})

{log(3)+1.364…• *i*,.079…,log(5)}

**log(***matriceCarrée1***[,***Expression2***])** ⇒ *matriceCarrée*
Donne le logarithme de base *Expression2* de *matriceCarrée1*. Ce calcul est différent du calcul du logarithme de base *Expression2* 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 *Expression2* est omis, la valeur de base 10 par défaut est utilisée.

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

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

$$\begin{bmatrix} .795\ldots+.753\ldots\cdot i & .003\ldots-.647\ldots\cdot i & \ldots \\ .194\ldots-.315\ldots\cdot i & .462\ldots+.270\cdot i & \ldots \\ -.115\ldots-.904\ldots\cdot i & .488\ldots+.777\ldots\cdot i & \ldots \end{bmatrix}$$

## ▸logbase

**Catalogue > ∫Σ > Base**

*Expression* **▸logbase(***Expression1***)** ⇒ *expression*
Provoque la simplification de l'expression entrée en une expression utilisant uniquement des logarithmes de base *Expression1*.

$\text{Log}(10,3) - \text{log}(5,5)$▸logbase(5)

$$\frac{\log_5(30)}{\log_5(3)}$$

## Logistic

**Catalogue > ∫Σ > Statistiques**

**Logistic** *X*, *Y*[, [*Fréq*] [, *Catégorie*, *Inclure*]]
Effectue une régression logistique sur *X* et *Y* en utilisant la fréquence *Fréq*. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **a,b,c** | Coefficients d'ajustement |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y - (c/(1+a • $e^{-bx}$)) |
| **RegEqn** | Équation d'ajustement : c/(1+a • $e^{-bx}$) |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

## LogisticD

**Catalogue > ∫Σ > Statistiques**

**LogisticD** *X*, *Y* [, [*Itérations*], [*Fréq*] [, *Catégorie*, *Inclure*] ]
Calcule la régression logistique. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)
Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.
*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.
*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.

| Variable de sortie | Description |
|---|---|
| **a,b,c,d** | Coefficients d'ajustement |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y - (a/(1+b • $e^{-c \cdot x}$)+d) |
| **RegEqn** | Équation d'ajustement : a/(1+b • $e^{-c \cdot x}$)+d) |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

**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 « : ».

Liste des fonctions :

```
Define rollcount()=Func
:  Local i
:  1 → i
:  Loop
:     If rand(6)=6 and rand(6)=6
:        Goto End
:     i+1 → i
:  EndLoop
:Lbl End
:Return i
:EndFunc
```

**LU** *Matrice*, *NomMatl*, *NomMatu*, *NomMatp*[, *Tol*]
Calcule la décomposition LU (lower-upper) de Doolittle d'une matrice réelle ou complexe. La matrice triangulaire inférieure est stockée dans *NomMatl*, la matrice triangulaire supérieure dans *NomMatu* et la matrice de permutation (qui décrit les échanges de lignes exécutés pendant le calcul) dans *NomMatp*.

*NomMatl* • *NomMatu* = *NomMatp* • *matrice*

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

- Si vous utilisez ctrl ≈ enter ou définissez le mode Exact/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 • max(dim(*Matri*)) • rowNorm(*Matrice*)

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

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

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

LU m1,lower,upper,perm

Done

lower

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

upper

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

perm

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

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

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

LU m1,lower,upper,perm

Done

lower

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

upper

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

perm

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

# M

## mat▸list()

**Catalogue > ∫Σ > Liste**

**mat▸list(***Matrice***)** ⇒ *liste*
Donne la liste obtenue en copiant les éléments de *Matrice* ligne par ligne.

mat▸list([1,2,3])
{1,2,3}

$\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}$ → M1

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

mat▸list(M1)
{1,2,3,4,5,6}

## max()

**Catalogue > ∫Σ > Liste**

**max(***Expression1***,** *Expression2***)** ⇒ *expression*
**max(***Liste1***,** *Liste2***)** ⇒ *liste*
**max(***Matrice1***,** *Matrice2***)** ⇒ *matrice*
Donne le maximum des deux arguments. Si les arguments sont deux listes ou matrices, donne une liste ou une matrice formée de la valeur maximale de chaque paire d'éléments correspondante.

max(2.3,1.4)
2.3

max({1,2},{⁻4,3})
{1,3}

**max(***Liste***)** ⇒ *expression*
Donne l'élément maximal de *liste*.

max({0,1,⁻7,1.3,.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,.3])
[1 0 7]

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

## mean()

**Catalogue > ∫Σ > Statistiques**

**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({.2,0,1,⁻.3,.4})
.26

mean({1,2,3},{3,2,1})
5/3

**mean(***Matrice1***[,** *matriceFréq***])** ⇒ *matrice*
Donne un vecteur ligne des moyennes de touters les colonnes de *Matrice1*.
Chaque élément de *matriceFréq* totalise le nombre d'occurrences de l'élément correspondant de *Matrice1*.

En mode Format Vecteur Rectangulaire :
mean([.2,0;⁻1,3;.4,⁻.5])
[⁻.133... .833...]

mean([1/5,0;⁻1,3;2/5,⁻1/2])
[⁻2/15 5/6]

mean([1,2;3,4;5,6],[5,3;4,1;6,2])
[47/15, 11/3]

## median()

**Catalogue > ∫Σ > Statistiques**

**median(***Liste***)** ⇒ *expression*
Donne la médiane des éléments de *Liste*.

median({.2,0,1,⁻.3,.4})
.2

**median(***Matrice1***)** ⇒ *matrice*
Donne un vecteur ligne contenant les médianes des colonnes de *Matrice1*.

median([.2,0;1,⁻.3;.4,⁻.5])
[.4 ⁻.3]

**Remarque :** tous les éléments de la liste ou de la matrice doivent correspondre à des valeurs numériques.

## MedMed

**Catalogue > ∫Σ > Statistiques**

**MedMed** *X*,*Y* [, *Fréq*] [, *Catégorie*, *Inclure*]]
Calcule la régression linéaire médiane-médiane (MedMed). Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.

*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| **a,b** | Coefficients d'ajustement : y = a • x+b |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y - (a • x+b) |
| **RegEqn** | Équation d'ajustement : a • x+b |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

## mid()

**Catalogue > ∫Σ > Chaîne**

**mid(***chaîneSource*, *Début*[, *Nbre*]**)** ⇒ *chaîne*
Donne la portion de chaîne de *Nbre* de caractères extraite de la chaîne *chaîneSource*, en commençant au numéro de caractère *Début*.

Si *Nbre* est omis ou s'il dépasse le nombre de caractères de la chaîne *chaîneSource*, on obtient tous les caractères de *chaîneSource*, compris entre le numéro de caractère *Début* et le dernier caractère.

*Nbre* doit être ≥ 0. Si *Nbre* = 0, on obtient une chaîne vide.

```
mid("Hello there",2)
                    "ello there"
mid("Hello there",7,3)
                        " the"
mid("Hello there",1,7)
                      "Hello"
mid("Hello there",1,0)
                         ""
```

**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({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(***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 > ∫Σ > Liste**

**min(***Expression1***,** *Expression2***)** ⇒ *expression*
**min(***Liste1*, *Liste2***)** ⇒ *liste*
**min(***Matrice1*, *Matrice2***)** ⇒ *matrice*
Donne le minimum des deux arguments. Si les arguments sont deux listes ou matrices, donne une liste ou une matrice formée de la valeur minimale de chaque paire d'éléments correspondante.

min(2.3,1.4)
1.4

min({1,2},{⁻4,3})
{⁻4,2}

**min(***Liste***)** ⇒ *expression*
Donne l'élément minimal de *Liste*.

min({0,1,⁻7,1.3,.5})
⁻7

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

min($\begin{bmatrix} 1 & -3 & 7 \\ -4 & 0 & .3 \end{bmatrix}$)
[⁻4 ⁻3 .3]

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

## mod()

**Catalogue > ∫Σ > Nombre**

**mod(***Expression1***,** *Expression2***)** ⇒ *expression*
**mod(***Liste1***,** *List2***)** ⇒ *list*
**mod(***Matrice1***,** *Matrice2***)** ⇒ *matrice*
Donne le premier argument modulo le deuxième argument, défini par les identités suivantes :
mod(x,0) = x
mod(x,y) = x − y floor(x/y)
Lorsque le deuxième argument correspond à une valeur non nulle, le résultat est de période de cet argument. Le résultat est soit zéro soit une valeur de même signe que le deuxième argument.
Si les arguments sont deux listes ou deux matrices, on obtient une liste ou une matrice contenant la congruence de chaque paire d'éléments correspondante.
**Remarque :** voir aussi **remain()**.

mod(7,0)
7

mod(7,3)
1

mod(⁻7,3)
2

mod(7,⁻3)
⁻2

mod(⁻7,⁻3)
⁻1

mod({12,⁻14,16},{9,7,⁻5})
{3,0,⁻4}

## mRow()

**Catalogue > ∫Σ > Matrice**

**mRow(***Expression***,** *Matrice1***,** *Index***)** ⇒ *matrice*
Donne une copie de *Matrice1* obtenue en multipliant chaque élément de la ligne *Index* de *Matrice1* par *Expression*.

mRow(⁻1/3,$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$,2)
$\begin{bmatrix} 1 & 2 \\ -1 & -4/3 \end{bmatrix}$

## mRowAdd()

**Catalogue > ∫Σ > Matrice**

**mRowAdd(***Expression***,** *Matrice1***,** *Index1***,** *Index2***)** ⇒ *matrice*
Donne une copie de *Matrice1* obtenue en remplaçant chaque élément de la ligne *Index2* de *Matrice1* par :
*Expression* × ligne *Index1* + ligne *Index2*

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

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

## multReg

**Catalogue > ∫Σ > Statistiques**

**multReg** *Y*, *X1*, *X2*, ..., *X10*

Calcule la régression linéaire multiple de la liste *Y* sur les listes *X1*, *X2*, ..., *X10*. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **bList** | {B0,B1, } Liste des coefficients de l'équation de régression $\hat{y}$ = B0+B1 • x1+ . |
| $r^2$ | Coefficient de détermination multiple |
| $\hat{y}$ | $\hat{y}$ = B0+B1 • x1+ |
| **Resid** | y - list |

## MultRegIntervals

**Catalogue > ∫Σ > Statistiques**

**MultRegIntervals** *Y*,*X1*,*X2*[,...[,*X10*]],*XValList*[,*CLevel*]

Calcule un intervalle de confiance pour la valeur que prendra y en $\hat{y}$ et en $\overline{y}$ en utilisant la régression multiple. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| $\hat{y}$ | Prévision d'un point : $\hat{y}$ = B0 + B1 • xl +... |
| **dfError** | Degrés de liberté des erreurs |
| **CLower, CUpper** | Intervalle de confiance pour une moyenne $\hat{y}$ |
| **ME** | Marge d'erreur de l'intervalle de confiance |
| **se** | Erreur type de l'intervalle de confiance |
| **LowerPred, UpperrPred** | Intervalle de prévision de $\hat{y}$ . |
| **MEPred** | Marge d'erreur de l'intervalle que vous pouvez prédire |
| **SEPred** | Erreur type d'un intervalle que vous pouvez prédire |
| **bList** | Liste de coefficients de régression, {B0,B1,...}. |
| **xvalist** | Valeurs X d'entrée servant à calculer $\hat{y}$ |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes y = B0 + B1 • x1 + B2 • x2 +... |

**MultRegTests** **Catalogue > ∫Σ > Statistiques**

**MultRegTests** *Y*,*X1*,*X2*[,*X3*[,...[,*X10*]]]
Le test *t* de régression linéaire multiple calcule une régression linéaire sur les données et effectue un test F statistique de linéarité. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Sorties

| Variable de sortie | Description |
|---|---|
| F | Statistique du test F global |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **r**$^2$ | Coefficient de détermination multiple |
| **AdjR**$^2$ | Coefficient ajusté de détermination multiple |
| **s** | Écart-type de l'erreur |
| **DW** | Statistique de Durbin-Watson ; sert à déterminer si la corrélation automatique de premier ordre est présente dans le modèle. |
| **dfReg** | Degrés de liberté de la régression |
| **SSReg** | Somme des carrés de la régression |
| **MSReg** | Moyenne des carrés de la régression |
| **dfError** | Degrés de liberté des erreurs |
| **SSError** | Somme des carrés des erreurs |
| **MSError** | Moyenne des carrés des erreurs |
| **bList** | {b0,b1,...} Liste des coefficients de l'équation de régression ? = b0+b1 • x1+... |
| **tList** | Liste des statistiques t pour chaque coefficient dans $\hat{y}$ (liste B) |
| **SEList** | Liste des pentes SE de chaque coefficient dans B |
| **PList** | Liste des valeurs de probabilité pour chaque statistique t |
| **List** | b0 +b1 • X1 + ... |
| **Resid** | y - list |
| **sResid** | Valeurs résiduelles normalisées ; valeur obtenue en divisant une valeur résiduelle par son écart-type. |
| **Leverage** | Mesure de la distance séparant les valeurs de la variable indépendante de leurs valeurs moyennes |
| **CookDist** | Distance de Cook ; Mesure de l'influence d'une observation basée sur la valeur résiduelle et le levier |

# N

## nCr()

**Catalogue > ∫Σ > Probabilité - Aléatoire**

**nCr(***Expression1***,** *Expression2***)** ⇒ *expression*

Pour les expressions *Expression1* et *Expression2* avec *Expression1* ≥ *Expression2* ≥ 0, **nCr()** donne le nombre de combinaisons de *Expression1* éléments pris parmi *Expression2* éléments. (Appelé aussi « coefficient binomial ».) Les deux arguments peuvent être des entiers ou des expressions symboliques.

**nCr(***Expression***, 0)** ⇒ **1**

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

**nCr(***Expression***,** *entierPos***)** ⇒ *Expression* • **(***Expression*−1**)...**
**(***Expression*−*entierPos***+1)/** *entierPos***!**

**nCr(***Expression***,** *nonEntier***)** ⇒ *expression***!/**
**((***Expression*−*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(***Matrice1***,** *Matrice2***)** ⇒ *matrice*

Donne une matrice de combinaisons basées sur les paires d'éléments correspondantes dans les deux matrices. Les arguments doivent être des matrices comportant le même nombre d'éléments.

nCr(z,3)

$$\frac{z\cdot(z-2)\cdot(z-1)}{6}$$

ans|z=5

10

nCr(z,c)

$$\frac{z!}{c!(z-c)!}$$

ans/nPr(z,c)

$$\frac{1}{c!}$$

nCr({5,4,3},{2,4,2})

{10,1,3}

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

$$\begin{bmatrix}15 & 10\\ 6 & 3\end{bmatrix}$$

## nDeriv()

**Catalogue > ∫Σ > Analyse**

**nDeriv(***Expression1***,** *Var***[,***H***])** ⇒ *expression*
**nDeriv(***Expression1***,** *Var***,** *Liste***)** ⇒ *liste*
**nDeriv(***Liste1***,** *Var***[,** *H***])** ⇒ *liste*
**nDeriv(***Matrice1***,** *Var***[,** *H***])** ⇒ *matrice*

Donne la dérivée numérique sous forme d'expression. Utilise la formule de quotient à différence symétrique.

*H* correspond à la valeur de l'incrément. Si *H* n'est pas précisé, il est fixé par défaut à 0.001.

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

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

nDeriv(cos(x),x,h)

$$\frac{-(\cos(x-h)-\cos(x+h))}{2\cdot h}$$

limit(nDeriv(cos(x),x,h),h,0)

−sin(x)

nDeriv($x^3$,x,0.01)

3.•($x^2$+.000033)

nDeriv(cos(x),x)|x=π/2

−1.

nDeriv($x^2$,x,{.01,.1})

{2.•x,2.•x}

## newList()

**Catalogue >**

**newList(***nbreÉléments***)** ⇒ *list*

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

newList(4)

{0,0,0,0}

## newMat()

**Catalogue >**

**newMat(***nbreLignes***,** *nbreColonnes***)** ⇒ *matrice*
Donne une matrice nulle de dimensions *nbreLignes*, *nbreColonnes*.

newMat(2,3)

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

## nInt()

**Catalogue > ∫Σ > Analyse**

**nInt(***Expression1*, *Var*, *Borne1*, *Borne2***)** ⇒ *expression*
Si l'intégrande *Expression1* ne contient pas d'autre variable que *Var* et si *Borne1* et *Borne2* sont des constantes, en +∞ ou en -∞, **nInt()** donne le calcul approché de ∫**(***Expression1*, *Var*, *Borne1*, *Borne2***)**. Cette approximation correspond à une moyenne pondérée de certaines valeurs d'échantillon de l'intégrande dans l'intervalle *Borne1*<*Var*<*Borne2*.

nInt(e ⁻x^2,x, ⁻1,1)

1.493...

L'objectif est d'atteindre une précision de six chiffres significatifs. L'alogrithme adaptif met un terme au calcul lorsqu'il semble avoir atteint cet objectif ou lorsqu'il paraît improbable que des échantillons supplémentaires produiront une amélioration notable.
Le message « Précision incertaine » s'affiche lorsque cet objectif ne semble pas atteint.

nInt(cos(x),x, ⁻π,π+1 E ⁻12)

⁻1.041... E ⁻12

∫(cos(x),x, ⁻π,π+10 ⁻12)

$$-\sin\left(\frac{1}{1000000000000}\right)$$

ans ↵

⁻1. E ⁻12

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

**Remarque :** voir aussi ∫**()**.

nInt(nInt(e ⁻x·y/√(x²−y²), y, ⁻x,x),x,0,1)

3.304...

## norm()

**Catalogue > ∫Σ > Matrice**

**norm(***Matrice***)** ⇒ *expression*
Donne la norme de Frobenius.

norm([a,b;c,d])

$$\sqrt{a^2+b^2+c^2+d^2}$$

norm([1,2;3,4])

$$\sqrt{30}$$

## normCdf()

**Catalogue > ∫Σ > Probabilité**

**normCdf(***lowerBound***,***upperBound***[,μ,σ])**
Calcule la fonction de répartition de la loi normale d'une variable de paramètres μ et σ spécifiés entre les bornes lowerBound et upperBound.

## normPdf()

**Catalogue > ∫Σ > Probabilité**

**normPdf(***ValX***[,μ,σ])**
Calcule la densité de probabilité de la loi normale à la valeur *ValX* spécifiée pour les paramètres μ et σ.

## not

**Catalogue > ∫Σ > Test**

**not** *Expression booléenne1* ⇒ *Expression booléenne*
Donne true (vrai) ou false (faux) ou une forme simplifiée de *Expression booléenne1*.

```
not 2>=3
                                  true
not x<2
                                  x ≥ 2
not not innocent
                                  innocent
```

**not** *Entier1* ⇒ *entier*
Donne le complément à 1 d'un entier. En interne, *Entier1* est converti en nombre binaire 32 bits signé. La valeur de chaque bit est inversée (0 devient 1, et vice versa) pour le complément à 1. Le résultat est affiché en fonction du mode Base utilisé.

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

Si vous entrez un nombre dont le codage binaire signé dépasse 32 bits, il est ramené à l'aide d'une congruence dans la plage appropriée.

En mode base Hex :

```
not 0h7AC36
```

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

```
                                  0hFFF853C9
```

En mode base Bin :

```
0b100101▶dec
                                  37
not 0b100101
          0b11111111111111111111111111011010
ans ▶dec
                                  ⁻38
```

**Remarque :** une entrée binaire peut comporter jusqu'à 32 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 8 chiffres.

## nPr()

**Catalogue > ∫Σ > Probabilité**

**nPr(***Expression1***,** *Expression2***)** ⇒ *expression*

Pour les expressions *Expression1* et *Expression2* entières, où *Expression1* ≥ *Expression2* ≥ 0, **nPr()** donne le nombre de permutations de *Expression1* éléments pris parmi *Expression2* éléments. Les deux arguments peuvent être des entiers ou des expressions symboliques.

**nPr(***Expression***, 0)** ⇒ **1**

**nPr(***Expression***,** *entierNég***)** ⇒
**1/((***Expression***+1) • (***Expression***+2)... (***expression*−*entierNég***))**

**nPr(***Expression***,** *entierPos***)**
⇒ *Expression* **• (***Expression*−**1)... (***Expression*−*entierPos***+1)**

**nPr(***Expression***,** *nonEntier***)**
⇒ *Expression***! / (***Expression*−*nonEntier***)!**

```
nPr(z,3)
                                  z • (z−2) • (z−1)
ans|z=5
                                  60
nPr(z, ⁻3)
                                  1 / ((z+1) • (z+2) • (z+3))
nPr(z,c)
                                  z! / (z− c)!
ans • nPr(z−c, ⁻c)
                                  1
```

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

Donne une liste de permutations basées sur les paires d'éléments correspondantes dans les deux listes. Les arguments doivent être des listes comportant le même nombre d'éléments.

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

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

Donne une matrice de permutations basées sur les paires d'éléments correspondantes dans les deux matrices. Les arguments doivent être des matrices comportant le même nombre d'éléments.

```
nPr([6,5;4,3],[2,2;2,2])
                                  [30 20]
                                  [12  6]
```

## nSolve()

**Catalogue > ∫Σ > Algèbre**

**nSolve(***Équation***,** *VarOuInit***)** ⇒ *chaîne_nombre ou erreur*

Recherche de façon itérative une solution numérique réelle approchée pour *Équation* en fonction de sa variable. Spécifiez *VarOuInit* comme suit :

*variable*

– ou –

*variable = nombre réel*

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

**nSolve()** est souvent plus rapide que **solve()** ou **zeros()**, notamment si l'opérateur « | » est utilisé pour limiter la recherche à un intervalle réduit qui contient exactement une seule solution.

**nSolve()** tente de déterminer un point où la valeur résiduelle est zéro ou deux points relativement rapprochés où la valeur résiduelle a un signe négatif et où son ordre de grandeur n'est pas excessif. S'il n'y parvient pas en utilisant un nombre réduit de points d'échantillon, la chaîne « Aucune solution n'a été trouvée » s'affiche.

Si vous utilisez **nSolve()** dans une fonction, vous pouvez recourir à **getType()** pour vérifier le caractère numérique du résultat avant de l'utiliser dans une expression algébrique.

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

nSolve($x^2$+5x−25=9,x)

3.844...

nSolve($x^2$=4,x=⁻1)

⁻2.

nSolve($x^2$=4,x=1)

2.

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

nSolve($x^2$+5x−25=9,x)|x<0

⁻8.844...

nSolve(((1+r)$^{24}$−1)/r=26,r)

|r>0 and r<.25

.0068...

nSolve($x^2$=⁻1,x)

"no solution found"

# O

## OneVar

**Catalogue > ∫Σ > Statistiques**

**OneVar** [**1,**]*X*[**,**[*Fréq*][**,***Catégorie***,***Inclure*]]

**OneVar** [*n***,**]*X1***,***X2*[*X3*[**,**…[**,***X20*]]]

Effectue le calcul de statistiques à une variable sur un maximum de 20 listes. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.

*X* représente la liste des valeurs de x.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| $\bar{x}$ | Moyenne des valeurs x |
| **Σx** | Somme des valeurs x |
| **Σx**$^2$ | Somme des valeurs x$^2$ |
| **sx_** | Écart-type de l'échantillon de x |
| **σx** | Écart-type de la population de x |
| **n** | Nombre de points de données |
| **MinX** | Minimum des valeurs de x |
| **q$_1$x** | 1er quartile de x. |

| Variable de sortie | Description |
|---|---|
| **MedX** | Médiane de x |
| **$q_3$x** | 3ème quartile de x |
| **MaxX** | Maximum des valeurs de x |
| **SSX** | Somme des carrés des écarts par rapport à la moyenne de x |

## or

**Catalogue > ∫Σ > Test**

*Expression booléenne1* **or** *Expression booléenne2* ⇒ *Expression booléenne*

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

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

**Remarque :** voir **xor**.

*Entier1* **or** *Entier2* ⇒ *entier*

Compare les représentaitons binaires de deux entiers réels en appliquant un or bit par bit. En interne, les deux entiers sont convertis en nombres binaires 32 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 32 bits, il est ramené à l'aide d'une congruence dans la plage appropriée.

**Remarque :** voir **xor**.

```
x≥3 or x≥4
                                  x ≥ 3
Segment de la fonction :
  :
: If x<0 or x≥5
:    Goto END
  :
:Lbl END
  :
```

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 32 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 8 chiffres.

## ord()

**Catalogue > ∫Σ > Chaîne**

**ord(***Chaîne***)** ⇒ *entier*

**ord(***Liste1***)** ⇒ *liste*

Donne le code numérique du premier caractère de la chaîne de caractères *Chaîne* ou une liste des premiers caractères de tous les éléments de la liste.

```
ord("bonjour")
                                  104
char(104)
                                 "b"
ord(char(24))
                                   24
ord({"alpha","beta"})
                               {97,98}
```

# P

## P▶Rx()

**Catalogue > ∫Σ > Angle**

**P▶Rx(***ExpressionR*, θ*Expression*) ⇒ *expression*
**P▶Rx(***ListeR*, θ*Liste*) ⇒ *liste*
**P▶Rx(***MatriceR*, θ*Matrice*) ⇒ *matrice*
Donne la valeur de l'abscisse du point de coordonnées polaires (r, θ).

**Remarque :** l'argument θ est interprété comme une mesure en degrés, en grades ou en radians, suivant le mode Angle utilisé. Si l'argument est une expression, vous pouvez utiliser °, ᴳ ou ʳ pour ignorer temporairement le mode Angle sélectionné.

En mode Angle en radians :

P▶Rx(r,θ)

cos(θ) • r

P▶Rx(4,60°)

2

P▶Rx({⁻3,10,1.3},{π/3, ⁻π/4,0})

{⁻3/2,5 • √2,1.3}

## P▶Ry()

**Catalogue > ∫Σ > Angle**

**P▶Ry(***Expression R*, θ*Expression*) ⇒ *expression*
**P▶Ry(***ListeR*, θ*Liste*) ⇒ *liste*
**P▶Ry(***MatriceR*, θ*Matrice*) ⇒ *matrice*
Donne la valeur de l'ordonnée du point de coordonnées polaires (r, θ).

**Remarque :** l'argument θ est interprété comme une mesure en degrés, en grades ou en radians, suivant le mode Angle utilisé. Si l'argument est une expression, vous pouvez utiliser °, ᴳ ou ʳ pour ignorer temporairement le mode Angle sélectionné.

En mode Angle en radians :

P▶Ry(r,θ)

sin(θ) • r

P▶Ry(4,60°)

2 • √3

P▶Ry({⁻3,10,1.3},{π/3, ⁻π/4,0})

$$\left\{\frac{-3\cdot\sqrt{3}}{2} \quad -5\cdot\sqrt{2} \quad 0.\right\}$$

## piecewise()

**Catalogue > ∫Σ > Probabilité**

**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)=piecewise(x+1,x>1undef,x≤1)

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

## poissCdf()

**Catalogue > ∫Σ > Probabilité**

**poissCdf(**λ**,***lowerBound*[,*upperBound*]**)**
Calcule la fonction de répartition d'une loi de Poisson discrète de moyenne λ spécifiée entre les bornes lowerBound et upperBound.

## poissPdf()

**Catalogue > ∫Σ > Propabilité**

**poissPdf(**λ**,***ValX***)**
Calcule la probabilité de ValX pour la loi de Poisson de moyenne λ spécifiée.

## ▸Polar

**Catalogue > ∫Σ > Matrice**

*Vecteur* **▸Polar**

Affiche *vecteur* sous forme polaire [r ∠θ]. Le vecteur doit être un vecteur ligne ou colonne et de dimension 2.

**Remarque : ▸Polar** est uniquement une instruction d'affichage et non une fonction de conversion. On ne peut l'utiliser qu'à la fin d'une ligne et elle ne modifie pas le contenu du registre *ans*.

**Remarque :** voir aussi **▸Rect**.

*valeurComplexe* **▸Polar**

Affiche *valeurComplexe* sous forme polaire.

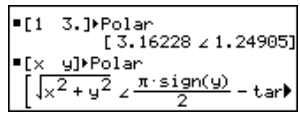
- Le mode Angle en degrés affiche (r∠θ).
- Le mode Angle en radians affiche reiθ.

*valeurComplexe* peut prendre n'importe quelle forme complexe. Toutefois, une entrée reiθ génère une erreur en mode Angle en degrés.

**Remarque :** vous devez utiliser les parenthèses pour les entrées polaires (r∠θ).

[1,3.] ▸Polar

[x,y] ▸Polar




En mode Angle en radians :

3+4i ▸Polar

$e^{i\cdot(\frac{\pi}{2}-\tan^{-1}(3/4))}\cdot 5$

(4∠π/3)▸Polar

$\frac{i\cdot\pi}{e^{3}}\cdot 4$

En mode Angle en grades :

4i ▸Polar

(4∠100)

En mode Angle en degrés :

3+4i ▸Polar

(5∠90−tan⁻¹(3/4))

## polyEval()

**Catalogue > ∫Σ > Liste**

**polyEval(***Liste1***,** *Expression1***)** ⇒ *expression*
**polyEval(***Liste1***,** *Liste2***)** ⇒ *expression*

Interprète le premier argument comme les coefficients d'un polynôme ordonné suivant les puissances décroissantes et calcule la valeur de ce polynôme au point indiqué par le deuxième argument.

polyEval({a,b,c},x)

a • x²+b • x+c

polyEval({1,2,3,4},2)

26

polyEval({1,2,3,4},{2, ⁻7})

{26, ⁻262}

## PowerReg

**Catalogue > ∫Σ > Statistiques**

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

Calcule l'ajustement par une fonction de puissance. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.

*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| **a,b** | Coefficients d'ajustement : y = a • (x)$^b$ |
| **r**$^2$ | Coefficient de détermination |
| **r** | Coefficient de corrélation linéaire |

| Variable de sortie | Description |
|---|---|
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y - a • (x)$^b$ |
| **ResidTrans** | Valeurs résiduelles associées à l'ajustement linéaire des données transformées |
| **RegEqn** | Équation d'ajustement : a • (x)$^b$ |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

**Product (PI)**

Voir **Π()**, page 161.

## product()

**Catalogue > ∫Σ > Liste**

**product(***Liste*[**,** *Début*[**,** *Fin*]]**)** ⇒ *expression*

Donne le produit des éléments de *Liste*. *Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage d'éléments.

```
product({1,2,3,4})
                                  24
product({2,x,y})
                              2 • x • y
product({4,5,8,9},2,3)
                                  40
```

**product(***Matrice1*[**,** *Début*[**,** *Fin*]]**)** ⇒ *matrice*

Donne un vecteur ligne contenant les produits des éléments ligne par line de *Matrice1*. *Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage de colonnes.

```
product([1,2,3;4,5,6;7,8,9])
                            [28 80 162]
product([1,2,3;4,5,6;7,8,9],1,2)
                              [4,10,18]
```

## propFrac()

**Catalogue > ∫Σ > Algèbre**

**propFrac(***Expression1*[**,** *Var*]**)** ⇒ *expression*

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

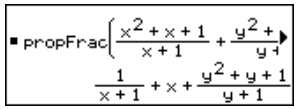
```
propFrac(4/3)
                                1 + 1/3
propFrac(⁻4/3)
                               ⁻1−1/3
```

propFrac(*expression*n_rationnelle,*Var***)** donne la somme des rapports 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 rapport propre. Les mêmes puissances de *Var* sont regroupées. Les termes et leurs facteurs sont triés, *Var* étant la variable principale.

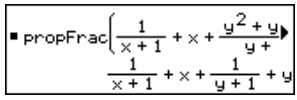
Si *Var* est omis, le développement des fractions propres s'effectue par rapport à la variable la plus importante. Les coefficients de la partie polynomiale sont ensuite ramenés à leur forme propre par rapport à leur variable la plus importante, et ainsi de suite.

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

propFrac((x$^2$+x+1)/(x+1)+ (y$^2$+y+1)/(y+1),x)




propFrac(ans)



# Q

## QR

**Catalogue > ∫Σ > Matrice**

**QR** *Matrice*, *NomMatq*, *NomMatr*[, *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 Exact/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 • max(dim(*Matrice*)) • rowNorm(*Matrice*)

La factorisation QR sous forme numérique est calculée en utilisant la transformation de Householder. La factorisation symbolique est calculée en utilisant la méthode de Gram-Schmidt. Les colonnes de *NomMatq* sont les vecteurs de base orthonormaux qui représentent l'espace défini par *matrice*.

Le nombre en virgule flottante (9.) dans m1 fait que les résultats seront tous calculés en virgule flottante.

[1,2,3;4,5,6;7,8,9.] → m1

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

QR m1,qm,rm

Done

qm

$$\begin{bmatrix} .123\ldots & .904\ldots & .408\ldots \\ .492\ldots & .301\ldots & -.816\ldots \\ .861\ldots & -.301\ldots & .408\ldots \end{bmatrix}$$

rm

$$\begin{bmatrix} 8.124\ldots & 9.601\ldots & 11.078\ldots \\ 0. & .904\ldots & 1.809\ldots \\ 0. & 0. & 0. \end{bmatrix}$$

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

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

QR m1,qm,rm

Done

qm

$$\begin{bmatrix} \frac{m}{\sqrt{m^2+o^2}} & \frac{-\text{sign}(m \cdot p - n \cdot o) \cdot o}{\sqrt{m^2+o^2}} \\ \frac{o}{\sqrt{m^2+o^2}} & \frac{m \cdot \text{sign}(m \cdot p - n \cdot o)}{\sqrt{m^2+o^2}} \end{bmatrix}$$

rm

$$\begin{bmatrix} \sqrt{m^2+o^2} & \frac{m \cdot n + o \cdot p}{\sqrt{m^2+o^2}} \\ 0 & \frac{|m \cdot p - n \cdot o|}{\sqrt{m^2+o^2}} \end{bmatrix}$$

## QuadReg

**Catalogue > ∫Σ > Statistiques**

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

Calcule l'ajustement par un polynôme de degré 2. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.

*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| **a,b,c** | Coefficients d'ajustement |
| **r²** | Coefficient de détermination |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y - (a • x²+b • x+c) |
| **RegEqn** | Équation d'ajustement : a • x²+b • x+c |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

## QuartReg

**Catalogue > ∫Σ > Statistiques**

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

Calcule l'ajustement par un polynôme de degré 4. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.

*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| **a,b,c,d,e** | Coefficients d'ajustement |
| **r²** | Coefficient de détermination |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y - (a • x⁴+b • x³+c • x²+d • x+e). |
| **RegEqn** | Équation d'ajustement : a • x⁴+b • x³+c • x²+d • x+e |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

# R

## R▸Pθ()

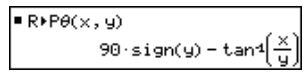
**Catalogue > ∫Σ > Angle**

**R▸Pθ** (*ExpressionX*, *ExpressionY*) ⇒ *expression*
**R▸Pθ** (*ListeX*, *ListeY*) ⇒ *liste*
**R▸Pθ** (*MatriceX*, *MatriceY*) ⇒ *matrice*
Donne la coordonnée θ d'un point de coordonnées rectangulaires (*x*,*y*).

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

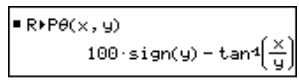
En mode Angle en degrés :

R▸Pθ(x,y)




En mode Angle en grades :

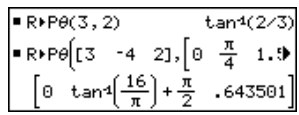
R▸Pθ(x,y)




En mode Angle en radians :

R▸Pθ(3,2)

R▸Pθ([3,-4,2],[0,π/4,1.5])




## R▸Pr()

**Catalogue > ∫Σ > Angle**

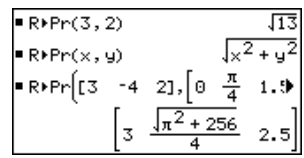
**R▸Pr** (*ExpressionX*, *ExpressionY*) ⇒ *expression*
**R▸Pr** (*ListeX*, *ListeY*) ⇒ *liste*
**R▸Pr** (*MatriceX*, *MatriceY*) ⇒ *matrice*
Donne la coordonnée r d'un point de coordonnées rectangulaires (*x*,*y*).

En mode Angle en radians :

R▸Pr(3,2)

R▸Pr(x,y)

R▸Pr([3,-4,2],[0,π/4,1.5])




## ▸Rad

**Catalogue > ∫Σ > Angle**

*Expression1*▸**Rad** ⇒ *expression*
Convertit *Expression1* en une mesure d'angle en radians.

En mode Angle en degrés :

1.5 ▸Rad

.02618$^{R}$

En mode Angle en grades :

1.5 ▸Rad

.023562$^{R}$

## rand()

Catalogue > ∫Σ > Probabilité

**rand([*nbreEssais*])**
Donne un nombre aléatoire compris entre 0 et 1 pour un nombre d'essais *nbreEssais* spécifié.

```
RandSeed 1147
                Done
rand(2)
                {.158...,.683...}
```

└ Réinitialise le générateur de nombres aléatoires.

## randBin()

Catalogue > ∫Σ > Probabilité

**randBin(*n*, *p* [,*nbreEssais*])**
Génère et affiche un entier aléatoire tiré d'une distribution binomiale spécifiée.

```
randBin(80,.5)
                42.
randbin(80,.5,3)
                {5.,3.,6.}
```

## randInt()

Catalogue > ∫Σ > Probabilité

**randInt(*lowerBound*,*upperBound* [,*nbeEssais*])**
Génère et affiche un entier aléatoire d'une taille délimitée par les limites entières *Lower* et *Upper* pour un nombre d'essais *nbreEssais* spécifié.

```
randInt(3,10)
                9.
randInt(3,10,4)
                {8.,6.,9.,4.}
```

## randMat()

Catalogue > ∫Σ > Probabilité

**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 enter.

## randNorm()

Catalogue > ∫Σ > Probabilité

**randNorm(μ, σ [,*nbreEssais*]) ⇒ *expression***
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 [μ−3 • σ, μ+3 • σ].

```
RandSeed 1147
                Done
randNorm(0,1)
                .492...
randNorm(3,4.5)
                −3.543...
```

## randPoly()

Catalogue > ∫Σ > Probabilité

**randPoly(*Var*, *Ordre*) ⇒ *expression***
Donne un polynôme aléatorie de la variable *Var* et de degré *Ordre* spécifié. Les coefficients sont des entiers aléatoires compris dans la plage −9 à 9. Le premier coefficient sera non nul.
*Ordre* doit être un entier compris entre 0 et 99.

```
RandSeed 1147
                Done
randPoly(x,5)
                −2 • x5+3 • x4−6 • x3+4 • x−6
```

## randSamp()

Catalogue > ∫Σ > Probabilité

**randSamp(***List*,*nbreEssais*[,*sansRem*]**)** ⇒ *liste*

Donne une liste contenant un échantillon aléatoire de *nbreEssais* éléments choisis dans *Liste* avec option de remise (*sansRem*=0) ou sans option de remise (*sansRem*=1). L'option par défaut est avec remise.

```
define list3={1,2,3,4,5}
define list4=randSamp(list3,6)
                                  Done
list4
                          {5.,5.,3.,2.,1.,1.}
```

## RandSeed

Catalogue > ∫Σ > Probabilité

**RandSeed** *Nombre*

Si *Nombre* = 0, réinitialise le générateur de nombres aléatoires. Si *Nombre* ≠ 0, sert à générer deux nombres initiaux qui sont stockés dans les variables système seed1 et seed2.

```
RandSeed 1147
                                  Done
rand()
                                 .158...
```

## real()

Catalogue > ∫Σ > Complexe

**real(***Expression1***)** ⇒ *expression*

Donne la partie réelle de l'argument.

**Remarque :** toutes les variables non définies sont considérées comme réelles. Voir aussi **imag()**.

```
real(2+3i)
                                      2
real(z)
                                      z
real(x+iy)
                                      x
```

**real(***Liste1***)** ⇒ *liste*

Donne les parties réelles de tous les éléments.

```
real({a+i • b,3,i})
                                  {a,3,0}
```

**real(***Matrice1***)** ⇒ *matrice*

Donne les parties réelles de tous les éléments.

```
real([a+i • b,3;c,i])
                                  [a  3]
                                  [c  0]
```

## ▶Rect

Catalogue > ∫Σ > Complexe

*Vecteur* **▶Rect**

Affiche *Vecteur* en coordonnées rectangulaires [x, y, z]. Le vecteur doit être un vecteur ligne ou colonne de dimension 2 ou 3.

**Remarque :** **▶Rect** est uniquement une instruction d'affichage et non une fonction de conversion. On ne peut l'utiliser qu'à la fin d'une ligne et elle ne modifie pas le contenu du registre *ans*.

**Remarque :** voir aussi **▶Polar**.

```
[3 ∠π/4 ∠π/6] ▶Rect
                  [3 • √2/4  3 • √2/4  3 • √3/2]
[a ∠b ∠c]
                                  [a • cos(b) • sin(c)
                                  a • sin(b) • sin(c) a • cos(c)]
```

## ▶Rect

**Catalogue > ∫Σ > Complexe**

*valeurComplexe* **▶Rect**

Affiche *valeurComplexe* sous forme rectangulaire (a+bi). *valeurComplexe* peut prendre n'importe quelle forme rectangulaire. Toutefois, une entrée reiθ génère une erreur en mode Angle en degrés.

**Remarque :** vous devez utiliser les parenthèses pour les entrées polaires (r∠θ).

En mode Angle en radians :

$4e^{\pi/3}$▶Rect

$4 \cdot e^{\frac{\pi}{3}}$

(4∠π/3)▶Rect

$2+2 \cdot \sqrt{3} \cdot i$

En odeAngle en grades :

(1∠100)▶Rect

$i$

En mode Angle en degrés :

(4∠60)▶Rect

$2+2 \cdot \sqrt{3} \cdot i$

**Remarque :** pour taper ∠ à partir du clavier, sélectionnez-le dans la liste des symboles du Catalogue.

## ref()

**Catalogue > ∫Σ > Matrice**

**ref(***Matrice1***[, *Tol*]) ⇒ *matrice***

Donne une réduite de Gauss de la matrice *Matrice1*.

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

- Si vous utilisez ctrl ≈ enter ou définissez le mode Exact/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 • max(dim(*Matrice1*)) • rowNorm(*Matrice1*)

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

$\text{ref}\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}$

$\begin{bmatrix} 1 & -2/5 & -4/5 & 4/5 \\ 0 & 1 & 4/7 & 11/7 \\ 0 & 0 & 1 & -62/71 \end{bmatrix}$

$\begin{bmatrix} a & b & c \\ e & f & g \end{bmatrix} \rightarrow m1$

$\begin{bmatrix} a & b & c \\ e & f & g \end{bmatrix}$

ref(m1)

$\begin{bmatrix} 1 & \frac{f}{e} & \frac{g}{e} \\ 0 & 1 & \frac{a \cdot g - c \cdot e}{a \cdot f - b \cdot e} \end{bmatrix}$

## remain()

**Catalogue > ∫Σ > Nombre**

**remain(***Expression1***, *Expression2***) ⇒ *expression***
**remain(***Liste1***, *Liste2***) ⇒ *list***
**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)

remain(7,0)

7

remain(7,3)

1

remain(⁻7,3)

⁻1

remain(7, ⁻3)

1

remain(⁻7, ⁻3)

⁻1

remain({12, ⁻14,16},{9,7, ⁻5})

{3,0,1}

## remain()

**Catalogue > ∫Σ > Nombre**

Vous remarquerez que **remain(**−x,y**)** ≡ −**remain(**x,y**)**. Le résultat peut soit être égal à zéro , soit être du même signe que le premier argument.

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

remain([9, ⁻7;6,4],[4,3;4, ⁻3])

$$\begin{bmatrix} 1 & -1 \\ 2 & 1 \end{bmatrix}$$

## Return

**Barre d'outils Calculs > {□ □} > Structures de contrôle**

**Return** [*Expression*]

Donne *Expression* comme résultat de la fonction. S'utilise dans les blocs **Func**...**EndFunc**.

**Remarque :** vous pouvez utiliser **Return** sans argument pour quitter une fonction.

**Remarque :** entrez le texte sous forme d'une seule ligne (sans retour à la ligne).

```
Define factoral(nn)=Func
:local answer,count:1 → answer
:For count,1,nn
:answer • count → answer:EndFor
:Return answer:EndFunc
                                Done
factoral(3)
                                6
```

## right()

**Catalogue >**

**right(***Liste1***[, *Nomb*])** ⇒ *liste*

Donne les *Nomb* éléments les plus à droite de la liste *Liste1*.

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

right({1,3, ⁻2,4},3)

{3, ⁻2 4}

**right(***chaîneSource***[, *Nomb*])** ⇒ *chaîne*

Donne la chaîne formée par les *Nomb* caractères les plus à droite de la chaîne de caractères *chaîneSource*.

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

right("Hello",2)

"lo"

**right(***Comparaison***)** ⇒ *expression*

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

right(x<3)

3

## root()

**Catalogue > ∫Σ > Nombre**

**root(***Expression***)** ⇒ *racine*

**root(***Expression1***, *Expression2***)** ⇒ *racine*

Calcule une racine n-ième de x, où x peut être un nombre réel ou complexe en virgule flottante, un entier ou une fraction rationnelle réelle ou complexe, ou encore une expression symbolique générale.

root(8,3)

2

root(3,3)

$3^{1/3}$

root(3.0,3)

1.442249570

## rotate()

**Catalogue > ∫Σ > Base**

**rotate(***Entier1***[,*nbreRotations*])** ⇒ *entier*

Permute les bits de la représentation binaire d'un entier. *Entier1* peut être un entier de n'importe quelle base ; il est automatiquement converti sous forme binaire (32 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.

En mode base Bin :

rotate(0b1111010110000110101)

0b10000000000000111101011000011010

rotate(256,1)

0b1000000000

## rotate()

**Catalogue > ∫Σ > Base**

Si *nbreRotations* est positif, la permutation circulaire s'effectue vers la gauche. Si *nbreRotations* est négatif, la permutation circulaire s'effectue vers la droite. La valeur par défaut est ⁻1 (permutation circulation de un bit vers la droite).

Par exemple, dans une permutation circulaire vers la droite :

Tous les bits permutent vers la droite.

0b00000000000000111101011000011010 1

Le bit le plus à droite passe à la position la plus à gauche.

donne :

0b10000000000000011110101100001101 0

Le résultat est affiché selon le mode Base utilisé.

En mode base Hex :

```
rotate(0h78E)
                                0h3C7
rotate(0h78E, ⁻2)
                            0h800001E3
rotate(0h78E,2)
                                0h1E38
```

**Important :** pour une entrée binaire ou hexadécimale, vous devez utiliser respectivement le préfixe 0b ou 0h (zéro, pas la lettre O).

**rotate(***Liste1*[**,***nbreRotations*]**)** ⇒ *liste*

Donne une copie de *Liste1* dont les éléments ont été permutés circulairement vers la gauche ou vers la droite de *nbreRotations* éléments. Ne modifie en rien *Liste1*.

Si *nbreRotations* est positif, la permutation circulaire s'effectue vers la gauche. Si *nbreRotations* est négatif, la permutation circulaire s'effectue vers la droite. La valeur par défaut est ⁻1 (permutation circulaire d'un élément vers la droite).

En mode base Dec :

```
rotate({1,2,3,4})
                              {4,1,2,3}
rotate({1,2,3,4}, ⁻2)
                              {3,4,1,2}
rotate({1,2,3,4},1)
                              {2,3,4,1}
```

**rotate(***Chaîne1*[**,***nbreRotations*]**)** ⇒ *chaîne*

Donne une copie de *Chaîne1* dont les caractères ont été permutés circulairement vers la gauche ou vers la droite de *nbreRotations* caractères. Ne modifie en rien *Chaîne1*.

Si *nbreRotations* est positif, la permutation circulaire s'effectue vers la gauche. Si *nbreRotations* est négatif, la permutation circulaire s'effectue vers la droite. La valeur par défaut est ⁻1 (permutation circulaire d'un caractère vers la droite).

```
rotate("abcd")
                               "dabc"
rotate("abcd", ⁻2)
                               "cdab"
rotate("abcd",1)
                               "bcda"
```

## round()

**Catalogue > ∫Σ > Nombre**

**round(***Expression1*[**,** *n*]**)** ⇒ *expression*

Arrondit l'argument au nombre de chiffres n spécifié après la virgule. *n* doit être un entier compris entre 0 et 12. Si *n* est omis, arrondit l'argument à 12 chiffres significatifs.

**Remarque :** le mode d'affichage des chiffres peut affecter le résultat affiché.

```
round(1.234567,3)
                                1.235
```

**round(***Liste1*[**,** *n*]**)** ⇒ *liste*

Donne une liste des éléments arrondis au nombre de chiffres n spécifié.

```
round({π,√(2),ln(2)},4)
                          {3.1416,1.4142,.6931}
```

**round(***Matrice1*[**,** *n*]**)** ⇒ *matrice*

Donne une matrice des éléments arrondis au nombre de chiffres n spécifié.

```
round([ln(5),ln(3);π,e^1],1)
                              [1.6  1.1]
                              [3.1  2.7]
```

## rowAdd()

Catalogue > ∫Σ > Matrice

**rowAdd(***Matrice1***,** *IndexL1***,** *IndexL2***)** ⇒ *matrice*

Donne une copie de *Matrice1* obtenue en remplaçant dans la matrice la ligne *IndexL2* par la somme des lignes *IndexL1* et *IndexL2*.

rowAdd([3,4; ⁻3, ⁻2],1,2)

$$\begin{bmatrix} 3 & 4 \\ 0 & 2 \end{bmatrix}$$

rowAdd([a,b;c,d],1,2)

$$\begin{bmatrix} a & b \\ a+c & b+d \end{bmatrix}$$

## rowDim()

Catalogue > ∫Σ > Matrice

**rowDim(***Matrice***)** ⇒ *expression*

Donne le nombre de lignes de *Matrice*.

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

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix} \rightarrow M1$$

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

rowdim(M1)

3

## rowNorm()

Catalogue > ∫Σ > Matrice

**rowNorm(***Matrice***)** ⇒ *expression*

Donne le maximum des sommes des valeurs absolues des éléments de chaque ligne de *Matrice*.

**Remarque :** la matrice utilisée ne doit contenir que des éléments numériques. Voir aussi **colNorm()**.

rowNorm($\begin{bmatrix} -5 & 6 & -7 \\ 3 & 4 & 9 \\ 9 & -9 & -7 \end{bmatrix}$)

25

## rowSwap()

Catalogue > ∫Σ > Matrice

**rowSwap(***Matrice1***,** *IndexL1***,** *IndexL2***)** ⇒ *matrice*

Donne la matrice *Matrice1* obtenue en échangeant les lignes *IndexL1* et *IndexL2*.

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

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

rowSwap(Mat,1,3)

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

**rref()** **Catalogue > ∫Σ > Matrice**

**rref(***Matrice1***[, *Tol*]) ⇒ *matrice***
Donne la réduite de Gauss-Jordan de *Matrice1*.

$\text{rref}\begin{bmatrix} -2 & -2 & 0 & -6 \\ 1 & -1 & 9 & -9 \\ -5 & 2 & 4 & -4 \end{bmatrix}$

$\begin{bmatrix} 1 & 0 & 0 & 66/71 \\ 0 & 1 & 0 & \frac{147}{71} \\ 0 & 0 & 1 & -62/71 \end{bmatrix}$

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 enter ou définissez le mode Exact/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 • max(dim(*Matrice1*)) • rowNorm(*Matrice1*)

$\text{rref}\begin{bmatrix} a & b & x \\ c & d & y \end{bmatrix}$

$\begin{bmatrix} 1 & 0 & \frac{d \cdot x - b \cdot y}{a \cdot d - b \cdot c} \\ 0 & 1 & \frac{-(c \cdot x - a \cdot y)}{a \cdot d - b \cdot c} \end{bmatrix}$

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

# *S*

**sec()** **Catalogue > ∫Σ > Algèbre**

**sec(***Expression1***) ⇒ *expression***
**sec(***Liste1***) ⇒ *liste***
Affiche la sécante de *Expression1* ou retourne la liste des sécantes des éléments de *Liste1*.

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

En mode Angle en degrés :

sec(45)

$\sqrt{(2)}$

sec({1,2.3,4})

$\{\frac{1}{\cos(1)}\ 1.000\ldots\ \frac{1}{\cos(4)}\}$

**sec⁻¹()** **Catalogue > ∫Σ > Algèbre**

**sec⁻¹(***Expression1***) ⇒ *expression***
**sec⁻¹(***Liste1***) ⇒ *liste***
Affiche l'angle dont la sécante correspond à *Expression1* ou retourne la liste des arcs sécantes des éléments de *Liste1*.

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

En mode Angle en degrés :

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

0

En mode Angle en grades :

$\sec^{-1}(\sqrt{2})$

50

En mode Angle en radians :

$\sec^{-1}(\{1,2,5\})$

$\{0\ \frac{\pi}{3}\ \cos^{-1}(1/5)\}$

## sech()

**Catalogue > ∫Σ > Hyperbolique**

**sech(***Expression1***)** ⇒ *expression*
**sech(***Liste1***)** ⇒ *liste*
Affiche la sécante hyperbolique de *Expression1* ou retourne la liste des sécantes hyperboliques des éléments de *liste1*.

sech(3)

$$\frac{1}{\cosh(3)}$$

sech({1,2.3,4})

$$\left\{\frac{1}{\cosh(1)} \quad .198\ldots \quad \frac{1}{\cosh(4)}\right\}$$

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

**Catalogue > ∫Σ > Hyperbolique**

**sech$^{-1}$(***Expression1***)** ⇒ *expression*
**sech$^{-1}$ (***Liste1***)** ⇒ *liste*
Donne l'argument sécante hyperbolique de *Expression1* ou retourne la liste des arguments sécantes hyperboliques des éléments de *Liste1*.

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

sech$^{-1}$(1)

0

sech$^{-1}$({1,−2,2.1})

$$\{0 \quad \left(\frac{2\cdot\pi}{3}\right)\cdot i \quad 1.074\ldots\cdot i\}$$

## seq()

**Catalogue > ∫Σ > Liste**

**seq(***Expression***,** *Var***,** *Début***,** *Fin***[,** *Incrément***])** ⇒ *liste*
Incrémente la valeur de *Var* comprise entre *Début* et *Fin* en fonction de l'incrément *Incrément* spécifié, calcule *Expression* et affiche le résultat sous forme de liste. Le contenu initial de *Var* est conservé après l'application de **seq()**.

*Var* ne peut pas être une variable système.

La valeur par défaut de *Incrément* est 1.

seq($n^2$,n,1,6)

{1,4,9,16,25,36}

seq(1/n,n,1,10,2)

{1,1/3,1/5,1/7,1/9}

sum(seq(1/$n^2$,n,1,10,1))

$$\frac{196\ldots}{127\ldots}$$

ctrl ≈ enter

1.549...

**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 (32 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.

Si *nbreDécal* est positif, le décalage s'effectue vers la gauche. Si *nbreDécal* est négatif, le décalage s'effectue vers la droite. La valeur par défaut est −1 (décalage d'un bit vers la droite).

Dans un décalage vers la droite, le dernier bit est éliminé et 0 ou 1 est inséré à gauche selon le premier bit. Dans un décalage vers la gauche, le premier bit est éliminé et 0 est inséré comme dernier bit.

Par exemple, dans un décalage vers la droite :

Tous les bits sont décalés vers la droite.

0b00000000000000111101011000011010

Insère 0 si le premier bit est un 0
ou 1 si ce bit est un 1.

donne :

0b00000000000000011110101100011010

Le résultat est affiché selon le mode Base utilisé. Les zéros de tête ne sont pas affichés.

En mode base Bin :

shift(0b1111010110000110101)
0b111101011000011010

shift(256,1)
0b1000000000

En mode base Hex :

shift(0h78E)
0h3C7

shift(0h78E, −2)
0h1E3

shift(0h78E,2)
0h1E38

**Important :** pour enter un nombre binaire ou hexadécimal, utilisez toujours le préfixe 0b ou 0h (zéro, pas la lettre O).

**shift(***Liste1* [**,***nbreDécal*]**)** ⇒ *liste*

Donne une copie de *Liste1* dont les éléments ont été décalés vers la gauche ou vers la droite de *nbreDécal* éléments. Ne modifie en rien *Liste1*.

Si *nbreDécal* est positif, le décalage s'effectue vers la gauche. Si *nbreDécal* est négatif, le décalage s'effectue vers la droite. La valeur par défaut est −1 (décalage d'un élément vers la droite).

Les éléments introduits au début ou à la fin de *liste* par l'opération de décalage sont remplacés par undef (non défini).

En mode base Dec :

shift({1,2,3,4})
{undef,1,2,3}

shift({1,2,3,4}, −2)
{undef,undef,1,2}

shift({1,2,3,4},1)
{2,3,4,undef}

**shift(***Chaîne1* [**,***nbreDécal*]**)** ⇒ *chaîne*

Donne une copie de *Chaîne1* dont les caractères ont été décalés vers la gauche ou vers la droite de *nbreDécal* caractères. Ne modifie en rien *Chaîne1*.

Si *nbreDécal* est positif, le décalage s'effectue vers la gauche. Si *nbreDécal* est négatif, le décalage s'effectue vers la droite. La valeur par défaut est −1 (décalage d'un caractère vers la droite).

Les caractères introduits au début ou à la fin de *Chaîne* par l'opération de décalage sont remplacés par un espace.

shift("abcd")
" abc"

shift("abcd", −2)
" ab"

shift("abcd",1)
"bcd "

## sign()

**Catalogue > ∫Σ > Nombre**

**sign(***Expression1***)** ⇒ *expression*
**sign(***Liste1***)** ⇒ *liste*
**sign(***Matrice1***)** ⇒ *matrice*

Pour une *Expression1* réelle ou complexe, donne *Expression1*/**abs(***Expression1***)** si *Expression1*≠ 0.

Donne 1 si l'expression *Expression1* est positive.

Donne ⁻1 si l'expression *Expression1* est négative.

**sign(0)** donne ±1 en mode Format complexe Réel ; sinon, donne lui-même.

**sign(0)** représente le cercle d'unité dans le domaine complexe.

Dans le cas d'une liste ou d'une matrice, donne les signes de tous les éléments.

sign(⁻3.2)

⁻1.

sign({2,3,4,⁻5})

{1,1,1,⁻1}

sign(1+abs(x))

1

En mode Format complexe Réel :

sign([⁻3,0,3])

[⁻1 ±1 1]

## simult()

**Catalogue > ∫Σ > Matrice**

**simult(***matriceCoeff***,** *vecteurConst***[,** *Tol***])** ⇒ *matrice*

Donne un vecteur colonne contenant les solutions d'un système d'équations.

*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 Exact/Approché sur APPROCHÉ, les calculs sont effectués en virgule flottante.
- Si *Tol* est omis ou inutilisé, la tolérance par défaut est calculée comme suit :

  5E⁻14 • max(dim(*matriceCoeff*)) • rowNorm(*matriceCoeff*)

Résolution de x et y :
x + 2y = 1
3x + 4y = ⁻1

simult($\begin{bmatrix}1&2\\3&4\end{bmatrix}$,$\begin{bmatrix}1\\-1\end{bmatrix}$)

$\begin{bmatrix}-3\\2\end{bmatrix}$

La solution est x= ⁻3 et y=2.

Résolution :
ax + by = 1
cx + dy = 2

$\begin{bmatrix}a&b\\c&d\end{bmatrix}$ → matx1

$\begin{bmatrix}a&b\\c&d\end{bmatrix}$

simult(matx1,$\begin{bmatrix}1\\2\end{bmatrix}$)

$\begin{bmatrix}\frac{-(2\cdot b-d)}{a\cdot d-b\cdot c}\\ \frac{2\cdot a-c}{a\cdot d-b\cdot c}\end{bmatrix}$

**simult(***matriceCoeff***,** *matriceConst***[,** *Tol***])** ⇒ *matrice*

Permet de résoudre plusieurs systèmes d'équations, ayant les mêmes coefficients mais des seconds membres différents.

Chaque colonne de *matriceConst* représente le second membre d'un système d'équations. Chaque colonne de la matrice obtenue contient la solution du système correspondant.

Résolution :
x + 2y = 1
3x + 4y = ⁻1

x + 2y = 2
3x + 4y = ⁻3

simult($\begin{bmatrix}1&2\\3&4\end{bmatrix}$,$\begin{bmatrix}1&2\\-1&-3\end{bmatrix}$)

$\begin{bmatrix}-3&-7\\2&9/2\end{bmatrix}$

Pour le premier système, x= ⁻3 et y=2. Pour le deuxième système, x= ⁻7 et y=9/2.

## sin()

**Touche** sin⁻¹ sin

**sin(***Expression1***)** ⇒ *expression*
**sin(***Liste1***)** ⇒ *liste*
**sin(***Expression1***)** donne le sinus de l'argument sous forme d'expression.
**sin(***Liste1***)** donne une liste des sinus de tous les éléments de *Liste1*.
**Remarque :** l'argument est interprété comme mesure d'angle en degrés, en grades ou en radians, suivant le mode angulaire sélectionné. Vous pouvez utiliser °, G ou ʳ pour ignorer temporairement le mode angulaire sélectionné.

En mode Angle en degrés :

sin((π/4)ʳ)

$\frac{\sqrt{2}}{2}$

sin(45)

$\frac{\sqrt{2}}{2}$

sin({0,60,90})

$\{0 \quad \frac{\sqrt{3}}{2} \quad 1\}$

En mode Angle en grades :

sin(50)

$\frac{\sqrt{2}}{2}$

En mode Angle en radians :

sin(π/4)

$\frac{\sqrt{2}}{2}$

sin(45°)

$\frac{\sqrt{2}}{2}$

**sin(***matriceCarrée1***)** ⇒ *matriceCarrée*
Donne le sinus de la matrice *matriceCarrée1*. Ce calcul est différent du calcul du sinus de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.
*matriceCarrée1* doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

En mode Angle en radians :

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

$\begin{bmatrix} .942\ldots & -.045\ldots & -.031\ldots \\ -.045\ldots & .949\ldots & -.020\ldots \\ -.048\ldots & -.005\ldots & .961\ldots \end{bmatrix}$

## sin⁻¹()

**Touche** ctrl sin⁻¹ sin

**sin⁻¹(***Expression1***)** ⇒ *expression*
**sin⁻¹(***Liste1***)** ⇒ *liste*
**sin⁻¹(***Expression1***)** donne l'arc sinus de *Expression1*.
**sin⁻¹(***Liste1***)** donne une liste des arcs sinus de tous les éléments de *Liste1*.
**Remarque :** donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

En mode Angle en degrés :

sin⁻¹(1)

90

En mode Angle en grades :

sin⁻¹(1)

100

En mode Angle en radians :

sin⁻¹({0,.2,.5})

{0,.201...,.523...}

**sin⁻¹(***matriceCarrée1***)** ⇒ *matriceCarrée*
Donne l'argument sinus de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de l'argument sinus de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.
*matriceCarrée1* doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

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

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

$\begin{bmatrix} -.164\ldots-.064\ldots\cdot i & 1.490\ldots-2.105\ldots\cdot i & \ldots \\ .725\ldots-1.515\ldots\cdot i & .947\ldots-.778\ldots\cdot i & \ldots \\ 2.083\ldots-2.632\ldots\cdot i & -1.790\ldots+1.271\ldots\cdot i & \ldots \end{bmatrix}$

## sinh()

**Catalogue > ∫Σ > Hyperbolique**

**sinh(***Expression1***)** ⇒ *expression*
**sinh(***Liste1***)** ⇒ *liste*
**sinh (***Expression1***)** donne le sinus hyperbolique de l'argument sous forme d'expression.
**sinh (***Liste1***)** donne une liste des sinus hyperboliques de tous les 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()**.
*matriceCarrée1* doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

sinh(1.2)
1.509...

sinh({0,1.2,3.})
{0,1.509...,10.017...}

En mode Angle en radians :

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

$$\begin{bmatrix}360.954 & 305.708 & 239.604\\ 352.912 & 233.495 & 193.564\\ 298.632 & 154.599 & 140.251\end{bmatrix}$$

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

**Catalogue > ∫Σ > Hyperbolique**

**sinh$^{-1}$(***Expression1***)** ⇒ *expression*
**sinh$^{-1}$(***Liste1***)** ⇒ *liste*
**sinh$^{-1}$(***Expression1***)** donne l'arc sinus hyperbolique de l'argument sous forme d'expression.
**sinh$^{-1}$(***Liste1***)** donne une liste des arcs sinus hyperboliques de tous les éléments de *Liste1*.

**sinh$^{-1}$(***matriceCarrée1***)** ⇒ *matriceCarrée*
Donne l'arc sinus hyperbolique de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de l'arc 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.

sinh$^{-1}$(0)
0

sinh$^{-1}$({0,2.1,3})
{0,1.487...,sinh$^{-1}$(3)}

En mode Angle en radians :

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

$$\begin{bmatrix}.041\ldots & 2.155\ldots & 1.158\ldots\\ 1.463\ldots & .926\ldots & .112\ldots\\ 2.750\ldots & -1.528\ldots & .572\ldots\end{bmatrix}$$

## SinReg

**Catalogue > ∫Σ > Statistiques**

**SinReg** *X*, *Y* [ , [*Itérations*] ,[ *Période*] [, *Catégorie*, *Inclure*] ]
Calcule une régression sinusoïdale. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)
Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.
*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.
*Itérations* spécifie le nombre maximum d'itérations utilisées (entre 1 et 16) 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.
Le résultat obtenu avec **SinReg** est toujours exprimé en radians, indépendamment du mode angulaire sélectionné.

| Variable de sortie | Description |
|---|---|
| **a,b,c,d** | Coefficients d'ajustement |
| **Resid** | Valeurs résiduelles de l'ajustement des courbes = y N a • sin(bx+c)+d |
| **RegEqn** | Équation d'ajustement : a • sin(bx+c)+d |
| **XOut** | Liste des points de données de la liste *Liste X* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **YOut** | Liste des points de données de la liste *Liste Y* modifiée, actuellement utilisée dans l'ajustement basé sur les restrictions de *Fréq*, *Liste de catégories* et *Inclure les catégories*. |
| **FreqOut** | Liste des fréquences correspondant à *XOut* et *YOut*. |

## solve()

**Catalogue > ∫Σ > Algèbre**

**solve(***Équation***,** *Var***)** ⇒ *Expression booléenne*
**solve(***Inéquation***,** *Var***)** ⇒ *Expression booléenne*

Résout dans R une équation ou une inégalité en *Var*. L'objectif est de trouver toutes les solutions possibles. Toutefois, il peut arriver arriver avec certaines équations ou inéquations que le nombre de solutions soit infini.

solve(a • x²+b • x+c=0,x)

$$x = \frac{\sqrt{b^2 - 4 \cdot a \cdot c} - b}{2 \cdot a}$$

$$\text{or } x = \frac{-(\sqrt{b^2 - 4 \cdot a \cdot c} + b)}{2 \cdot a}$$

Les solutions possibles peuvent ne pas être des solutions réelles finies pour certaines combinaisons de valeurs de variables non définies.

ans | a=1 and b=1 and c=1

Error: Non-real result

Avec le réglage Auto du mode Exact/Approché, l'objectif est de trouver des solutions exactes quand elles sont concises et de compléter l'opération par des recherches itératives de calcul approché lorsque des solutions exactes ne peuvent pas être trouvées.

solve((x-a)e^x= ⁻x • (x-a),x)

x = a or x = ⁻.567...

En raison de l'annulation par défaut du plus grand commun diviseur du numérateur et du dénominateur des rapports, les solutions trouvées peuvent ne pas être valides.

(x+1)(x-1)/(x-1)+x-3

2 • x-2

Pour les inéquations de type ≥, ≤, < ou >, il est peut probalbe de trouver des solutions explicites, sauf si l'inéquation est linéaire et ne contient que *Var*.

solve(5x-2 ≥ 2x,x)

x ≥ 2/3

Avec le réglage Exact du mode Exact/Approché, les portions qui ne peuvent pas être résolues sont données sous forme d'équation ou d'inéquation implicite.

exact(solve((x-a)e^x= ⁻x • (x-a),x))

eˣ + x = 0 or x = a

Utilisez l'opérateur « | » pour limiter l'"intervalle de la solution et/ou les autres variables rencontrées dans l'équation ou l'inéquation. Lorsqu'une solution est trouvée dans un intervalle, vous pouvez utiliser les opérateurs d'inéquation pour exclure cet intervalle des recherches suivantes.

En mode Angle en radians :

solve(tan(x)=1/x,x)|x>0 and x<1

x =.860...

false est affiché si aucune solution réelle n'est trouvée. true est affiché si **solve()** parvient à déterminer qu'une quelconque valeur réelle finie de *Var* satisfait l'équation ou l'inéquation.

solve(x=x+1,x)

false

solve(x=x,x)

true

Dans la mesure où **solve()** donne toujours un résultat booléen, vous pouvez utiliser « and », « or » et « not » pour combiner les résultats de **solve()** entre eux ou avec d'autres expressions booléennes.

2x−1≤1 and solve(x²≠9,x)

x ≤ 1 and x ≠ ⁻3

Les solutions peuvent contenir une nouvelle variable non définie de type ***n*j**, où j correspond à un entier compris dans l'intervalle de 1 à 255. Ces variables désignent un entier arbitraire.

En mode Angle en radians :

solve(sin(x)=0,x)

x=***n***1 • π

En mode Réel, les puissances fractionnaires possédant un dénominateur impair font uniquement référence à la branche principale. Sinon, les expressions à plusieurs branches, telles que les puissances fractionnaires, les logarithmes et les fonctions trigonométriques inverses font uniquement référence à la branche principale. Par conséquent, **solve()** donne uniquement des solutions correspondant à cette branche réelle ou principale.

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

**solve(***Équation1* **and** *Équation2* [**and** … ]**, {***VarOuInit1***,** *VarOuInit2* [**,** … ]**}) ⇒** *Expression booléenne*

**solve(***système d'équations***,** *VarOuInit1***,** *VarOuInit2* [**,** … ]**) ⇒** *Expression booléenne*

Donne les solutions réelles possibles d'un système d'équations algébriques, où chaque *VarOuInit* définit une variable du système à résoudre.

Vous pouvez séparer les équations par l'opérateur **and** ou entrer un *système d'équations* en utilisant un modèle du Catalogue. Le nombre d'arguments *VarOuInit* doit correspondre au nombre d'équations. Vous pouvez également spécifier une supposition initiale pour les variables. Chaque *VarOuInit* doit utiliser le format suivant :

*variable*

– ou –

*variable = nombre réel ou non réel*

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

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

Par exemple, si vous avez un cercle de rayon r centré à l'origine et un autre cercle de rayon r centré, au point où le premier cercle coupe l'axe des x positifs. Utilisez **solve()** pour trouver les intersections.

Comme l'illustre r dans l'exemple ci-contre, des équations polynomiales simultanées peuvent avoir des variables supplémentaires sans valeur assignée, mais représenter des valeurs numériques données pouvant s'y substituer par la suite.

Vous pouvez également utiliser des variables qui n'apparaissent pas dans les équations. Par exemple, vous pouvez utiliser z comme variable pour développer l'exemple précédent et avoir deux cylindres parallèles sécants de rayon r.

La résolution du problème montre comment les solutions peuvent contenir des constantes arbitraires de type @k, où k est un suffixe entier compris entre 1 et 255.

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

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

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

$x=-1$

solve($\sqrt{(x)}=-2$,x)

false

solve($-\sqrt{(x)}=-2$,x)

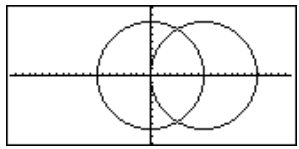
$x=4$

solve($y=x^2-2$ and $x+2y=-1$,{x,y})

$x=1$ and $y=-1$
or $x=-3/2$ and $y=1/4$

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

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



solve($x^2+y^2=r^2$ and $(x-r)^2+y^2=r^2$,{x,y})

$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2}$$
$$\text{or } x=\frac{r}{2} \text{ and } y=\frac{-\sqrt{3}\cdot r}{2}$$

solve($x^2+y^2=r^2$ and $(x-r)^2+y^2=r^2$,{x,y,z})

$$x=\frac{r}{2} \text{ and } y=\frac{\sqrt{3}\cdot r}{2} \text{ and } z=@1$$
$$\text{or } x=\frac{r}{2} \text{ and } y=\frac{-\sqrt{3}\cdot r}{2} \text{ and } z=@1$$

solve($x+e^z\cdot y=1$ and $x-y=\sin(z)$,{x,y})

$$x=\frac{e^z\cdot\sin(z)+1}{e^z+1} \text{ and } y=\frac{-(\sin(z)-1)}{e^z+1}$$

## solve()

**Catalogue > ∫Σ > Algèbre**

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

Chaque variable du système commence à sa valeur supposée, si elle existe ; sinon, la valeur de départ est 0.0.

Utilisez des valeurs initiales pour rechercher des solutions supplémentaires, une par une. Pour assurer une convergence correcte, une valeur initiale doit être relativement proche de la solution.

solve($e^z \cdot y=1$ and $\bar{}y=\sin(z)$,{y,z})

y=.041… and z=3.183…

solve($e^z \cdot y=1$ and $\bar{}y=\sin(z)$,{y,z=2π})

y=.001… and z=6.281…

## SortA

**Catalogue > ∫Σ > Liste**

**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.

{2,1,4,3} → List1

{2,1,4,3}

SortA List1

Done

List1

{1,2,3,4}

{4,3,2,1} → List2

{4,3,2,1}

SortA List2,List1

Done

List2

{1,2,3,4}

List1

{4,3,2,1}

## SortD

**Catalogue > > > ∫Σ > Liste**

**SortD** *Liste1*[**,** *Liste2*] [**,** *Liste3*] …
**SortD** *Vecteur1*[**,***Vecteur2*] [**,***Vecteur3*] …

Identique à **SortA**, mais **SortD** trie les éléments en ordre décroissant.

{2,1,4,3} → List1

{2,1,4,3}

{1,2,3,4} → List2

{1,2,3,4}

SortD List1,List2

Done

List1

{4,3,2,1}

List2

{3,4,1,2}

## ▸Sphere

**Catalogue > ∫Σ > Matrice**

*Vecteur* **▸Sphere**

Affiche le vecteur ligne ou colonne en coordonnées sphériques [ρ ∠θ ∠ϕ].

*Vecteur* doit être un vecteur ligne ou colonne de dimension 3.

**Remarque : ▸Sphere** est uniquement une instruction d'affichage et non une fonction de conversion. On ne peut l'utiliser qu'à la fin d'une ligne.

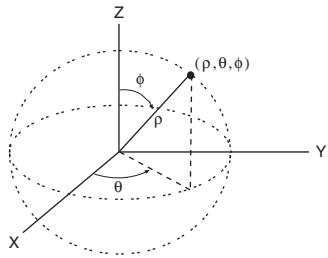
[1 2 3]▸Sphere ↵

[3.741... ∠1.107... ∠.640...]

[2 ∠π/4 3]▸Sphere ↵

[3.605... ∠.785... ∠.588...]

$$[\sqrt{13}\ \angle\frac{\pi}{4}\ \angle\cos^{-1}(\frac{3\cdot\sqrt{13}}{13})]$$




## StatMatrix

**Catalogue >**

**StatMatrix**

Affiche une matrice de résultats d'analyse statistique.

Les résultats sont affichés sous forme d'ensemble de paires nom-valeur. Les noms spécifiques affichés varient suivant la fonction ou commande statistique la plus récemment calculée ou exécutée.

Vous pouvez copier un nom ou une valeur et la coller à d'autres emplacements.

**Remarque :** ne définissez pas de variables dont le nom est identique à celles utilisées dans le cadre de l'analyse statistique. Dans certains cas, cela peut générer une erreur. Les noms de variables utilisés pour l'analyse statistique sont répertoriés dans le tableau ci-dessous.

xlist:={1,2,3,4,5}
ylist:={4,8,11,14,17}
LinRegMx xlist,ylist : StatMatrix

LinRegMx *xlist*,*ylist*,1 : *StatMatrix*

| | |
|---|---|
| "RegEqn" | "m*x+b" |
| "m" | 3. |
| "b" | 2. |
| "r²" | 1. |
| "r" | 1. |
| "Resid" | "{...}" |
| "XOut" | "{...}" |
| "YOut" | "{...}" |
| "FreqOut" | "{...}" |

a
AdjR²
b
bList
$\chi^2$
c
CLower
CLowerList
CompList
CompMatrix
CookDist
CUpper
CUpperList
d
dfDenom
dfBlock
dfCol
dfError
dfInteract
dfReg
dfNumer
dfRow
DW

e
ExpMatrix
F
FBlock
Fcol
FInteract
FreqOut
Frow
Leverage
LowerPred
LowerVal
m
MaxX
MaxY
ME
MedianX
MedianY
MEPred
MinX
MinY
MS
MSBlock

MSCol
MSError
MSInteract
MSReg
MSRow
n
$\hat{p}$
$\hat{p}1$
$\hat{p}2$
$\hat{p}$Diff
PList
PVal
PValBlock
PValCol
PValInteract
PValRow
Q1X
Q1Y
Q3X
Q3Y
r
$r^2$
RegEqn

Resid
ResidTrans
σx
σy
σx1
σx2
Σx
Σx²
Σxy
Σy
Σy²
s
SE
SEList
SEPred
sResid
SEslope
sp
SS
SSBlock
SSCol
SSX

SSY
SSError
SSInteract
SSReg
SSRow
tList
UpperPred
UpperVal
$\bar{x}$
$\bar{x}1$
$\bar{x}2$
$\bar{x}$Diff
$\bar{x}$List
XOut
XVal
XValList
$\bar{y}$
$\hat{y}$
$\hat{y}$ List
YOut

## stdDevPop()

**Catalogue > $\int\Sigma$ > Statistiques**

**stdDevPop(***Liste***[,** *listeFréq***])** ⇒ *expression*

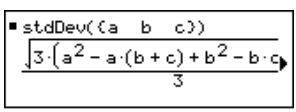
Donne l'écart-type de population des éléments de *Liste*.

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

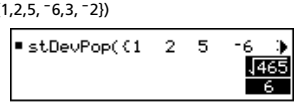
**Remarque :** *Liste* doit contenir au moins deux éléments.

En mode Angle en radians et en modes Auto :

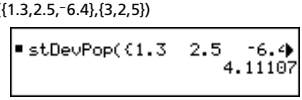
stdevPop({a,b,c})




stdevPop({1,2,5,⁻6,3,⁻2})




stdevPop({1.3,2.5,⁻6.4},{3,2,5})



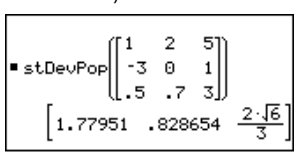

**stdevPop(***Matrice1***[,** *matriceFréq***])** ⇒ *matrice*

Donne un vecteur ligne des écarts-types de population des colonnes de *Matrice1*.

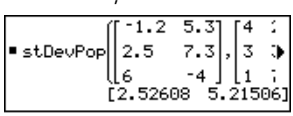
Chaque élément de *matriceFréq* totalise le nombre d'occurrences de l'élément correspondant de *Matrice1*.

**Remarque :** *Matrice1* doit contenir au moins deux lignes.

stdevPop($\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}$)




stdevPop($\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}, \begin{bmatrix} 4 & 2 \\ 3 & 3 \\ 1 & 7 \end{bmatrix}$)



## stDevSamp()

**Catalogue > ∫Σ > Statistiques**

**stDevSamp(***Liste*[**,** *listeFréq*]**)** ⇒ *expression*

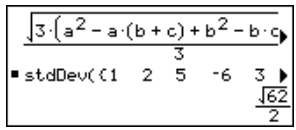
Donne l'écart-type d'échantillon des éléments de *Liste*.

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.

stdevSamp({a,b,c})

stdevSamp({1,2,5, ⁻6,3, ⁻2})




stdevSamp({1.3,2.5,⁻6.4},{3,2,5})

4.33345

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

Donne un vecteur ligne des écarts-types de population des colonnes de *Matrice1*.

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

**Remarque :** *Matrice1* doit contenir au moins deux lignes.

stdevSamp($\begin{bmatrix} 1 & 2 & 5 \\ -3 & 0 & 1 \\ .5 & .7 & 3 \end{bmatrix}$)

[2.179... 1.014... 2]

stdevSamp($\begin{bmatrix} -1.2 & 5.3 \\ 2.5 & 7.3 \\ 6 & -4 \end{bmatrix}$)

[2.7005,5.44695]

## Store

Voir → **(store)**, page 166.

## string()

**Catalogue > ∫Σ > Chaîne**

**string(***Expression***)** ⇒ *chaîne*

Simplifie *Expression* et donne le résultat sous forme de chaîne de caractères.

string(1.2345)

"1.2345"

string(1+2)

"3"

string(cos(x)+√(3))

"cos(x) + √(3)"

## subMat()

**Catalogue > ∫Σ > Matrice**

**subMat(***Matrice1*[**,** *ligneDébut*] [**,** *colDébut*] [**,** *ligneFin*] [**,** *colFin*]**)**
⇒ *matrice*

Donne la matrice spécifiée, extraite de *Matrice1*.

Valeurs par défaut : *ligneDébut*=1, *colDébut*=1, *ligneFin*=dernière ligne, *colFin*=dernière colonne.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ → m1

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

subMat(m1,2,1,3,2)

$\begin{bmatrix} 4 & 5 \\ 7 & 8 \end{bmatrix}$

subMat(m1,2,2)

$\begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix}$

Voir **Σ()**, page 161.

### sum()

**Catalogue > Σ > Liste**

**sum(***Liste*[**,** *Début*[**,** *Fin*]]**)** ⇒ *expression*

Donne la somme des éléments de *Liste*.

*Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage d'éléments.

sum({1,2,3,4,5})

15

sum({a,2a,3a})

6 • a

sum(seq(n,n,1,10))

55

sum({1,3,5,7,9},3)

21

**sum(***Matrice1*[**,** *Début*[**,** *Fin*]]**)** ⇒ *matrice*

Donne un vecteur ligne contenant les sommes des éléments de chaque colonne de *Matrice1*.

*Début* et *Fin* sont facultatifs. Ils permettent de spécifier une plage de lignes.

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]

### system()

**Catalogue >**

**system(***Expr1* [**,** *Expr2* [**,** *Expr3* [**,** ...]]]**)**

**system(***Équ1* [**,** *Équ2* [**,** *Équ3* [**,** ...]]]**)**

Donne un système d'équations, présenté sous forme de liste. Vous pouvez également créer un système d'équation en utilisant un modèle.

solve(system(x+y=0,x-y=8),x,y)

$\text{solve}\left\{\begin{matrix} x+y=0 \\ x-y=8 \end{matrix}, x, y\right\}$ $x=4$ and $y=-4$

## *T*

### T (transposée)

**Catalogue > Σ > Matrice**

*Matrice1*T ⇒ *matrice*

Donne la transposée de la conjuguée de *Matrice1*.

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

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

$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^T$

$\begin{bmatrix} a & c \\ b & d \end{bmatrix}$

$\begin{bmatrix} 1+i & 2+i \\ 3+i & 4+i \end{bmatrix}^T$

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

## tan()

**key** tan-1 tan

**tan(***Expression1***)** ⇒ *expression*
**tan(***Liste1***)** ⇒ *liste*
**tan(***Expression1***)** donne la tangente de l'argument.
**tan(***Liste1***)** donne une liste des tangentes de tous les éléments de *Liste1*.
**Remarque :** l'argument est interprété comme mesure d'angle en degrés, en grades ou en radians, suivant le mode angulaire sélectionné. Vous pouvez utiliser °, G ou r pour ignorer temporairement le mode Angle sélectionné.

En mode Angle en degrés :

tan((π/4)r)

1

tan(45)

1

tan({0,60,90})

{0,√3,undef}

En mode Angle en grades :

tan((π/4)r)

$$\frac{200 \cdot \tan\left(\frac{\pi}{4}\right)}{\pi}$$

tan(50)

1

tan({0,50,100})

{0,1,undef}

En mode Angle en radians :

tan(π/4)

1

tan(45°)

1

tan({π,π/3,-π,π/4})

{0,√3,0,1}

**tan(***matriceCarrée1***)** ⇒ *matriceCarrée*
Donne la tangente de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de la tangente de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**. *matriceCarrée1* doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

En mode Angle en radians :

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

$$\begin{bmatrix} -28.291\ldots & 26.088\ldots & 11.114\ldots \\ 12.117\ldots & -7.835\ldots & -5.481\ldots \\ 36.818\ldots & -32.806\ldots & -10.459\ldots \end{bmatrix}$$

## tan-1()

**Touche** ctrl tan-1 tan

**tan-1(***Expression1***)** ⇒ *expression*
**tan-1(***Liste1***)** ⇒ *liste*
**tan-1(***Expression1***)** donne l'arc tangente de *Expression1*.
**tan-1(***Liste1***)** donne la liste des arcs tangentes de tous les éléments de *Liste1*.
**Remarque :** donne le résultat en degrés, en grades ou en radians, suivant le mode angulaire utilisé.

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,.2,.5})

{0,.197...,.463...}

**tan-1(***matriceCarrée1***)** ⇒ *matriceCarrée*
Donne l'arc tangente de la matrice *matriceCarrée1*. Ce calcul est différent du calcul de l'arc tangente de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**. *matriceCarrée1* doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

En mode Angle en radians :

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

$$\begin{bmatrix} -.083\ldots & 1.266\ldots & .622\ldots \\ .748\ldots & .630\ldots & -.070\ldots \\ 1.686\ldots & -1.182\ldots & .455\ldots \end{bmatrix}$$

## tanh()

**Catalogue > ∫Σ > Hyperbolique**

**tanh(***Expression1***)** ⇒ *expression*
**tanh(***Liste1***)** ⇒ *liste*
**tanh(***Expression1***)** donne la tangente hyperbolique de l'argument.
**tanh(***Liste1***)** donne une liste des tangentes hyperboliques de tous les 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.

tanh(1.2)
.833...

tanh({0,1})
{0,tanh(1)}

En mode Angle en radians :

$$\tanh\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$$
$$\begin{bmatrix} -.097\ldots & .933\ldots & .425\ldots \\ .488\ldots & .538\ldots & -.129\ldots \\ 1.282\ldots & -1.034\ldots & .428\ldots \end{bmatrix}$$

## tanh⁻¹()

**Catalogue > ∫Σ > Hyperbolique**

**tanh$^{-1}$(***Expression1***)** ⇒ *expression*
**tanh$^{-1}$(***Liste1***)** ⇒ *liste*
**tanh$^{-1}$(***Expression1***)** donne l'argument tangente hyperbolique de l'argument.
**tanh$^{-1}$(***Liste1***)** donne une liste des arguments tangentes hyperboliques de tous les éléments de *Liste1*.

**tanh$^{-1}$(***matriceCarrée1***)** ⇒ *matriceCarrée*
Donne l'argument tangente hyperbolique de *matriceCarrée1*. Ce calcul est différent du calcul de l'argument tangente hyperbolique de chaque élément. Pour plus d'informations sur la méthode de calcul, reportez-vous à **cos()**.
*matriceCarrée1* doit être diagonalisable. Le résultat contient toujours des chiffres en virgule flottante.

En mode Format complexe Rectangulaire :

tanh$^{-1}$(0)
0

tanh$^{-1}$({1,2.1,3})
$\{\infty \quad .518\ldots - 1.570\ldots \cdot i \quad \frac{\ln(2)}{2} - \frac{\pi}{2} \cdot i\}$

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

$$\tanh^{-1}\left(\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}\right)$$
$$\begin{bmatrix} -.099\ldots+.164\ldots\cdot i & .267\ldots-1.490\ldots\cdot i & \ldots \\ -.087\ldots-.725\ldots\cdot i & .479\ldots-.947\ldots\cdot i & \ldots \\ .511\ldots-2.083\ldots\cdot i & -.878\ldots+1.790\ldots\cdot i & \ldots \end{bmatrix}$$

## taylor()

**Catalogue > ∫Σ > Analyse**

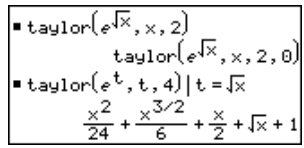
**taylor(***Expression1***,** *Var***,** *Ordre***[,** *Point***])** ⇒ *expression*

Donne le polynôme de Taylor demandé. Le polynôme comprend des termes supérieurs à zéro de degrés entiers compris entre zéro et *Ordre* dans (*Var* moins *Point*). **taylor()** donne lui-même en l'absence de série de puissances tronquées de cet ordre ou si l'opération exigerait l'utilisation d'exposants négatifs ou fractionnaires. Utilisez des opérations de substitution et/ou de multiplication temporaire par une puissance de (*Var* moins *Point*) pour déterminer une série de puissance plus générale.

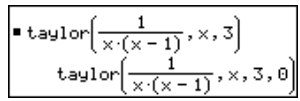
Par défaut, la valeur de *Point* est égale à zéro et il s'agit du point de développement.

taylor($E^{\sqrt{(x)}}$,x,2)

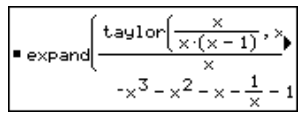
taylor($E^t$,t,4)|t=√(x)




taylor(1/(x • (x−1)),x,3)




expand(taylor(x/(x • (x−1)),
x,4)/x,x)




## tCdf()

**Catalogue > ∫Σ > Probabilité**

**tCdf(***lowerBound***,***upperBound***,***df***)**

Calcule la fonction de répartition de la loi Student-*t* entre *lowerBound* et *upperBound* pour les degrés de liberté *df* spécifiés.

## tCollect()

**Catalogue > ∫Σ > Algèbre**

**tCollect(***Expression1***)** ⇒ *expression*

Donne une expression dans laquelle les produits et les puissances entières des sinus et des cosinus sont convertis en une combinaison linéaire de sinus et de cosinus de multiples d'angles, de sommes d'angles et de différences d'angles. La transformation convertit les polynômes trigonométriques en une combinaison linéaire de leurs harmoniques.

Quelquefois, **tCollect()** permet d'atteindre vos objectifs lorsque la simplification trigonométrique n'y parvient pas. **tCollect()** fait l'inverse des transformations effectuées par **tExpand()**. Parfois, l'application de **tExpand()** à un résultat de **tCollect()**, ou vice versa, permet en deux étapes de simplifier une expression.

tCollect($(\cos(\alpha))^2$)

$$\frac{\cos(2\cdot\alpha)+1}{2}$$

tCollect(sin(α)cos(β))

$$\frac{\sin(\alpha-\beta)+\sin(\alpha+\beta)}{2}$$

## tExpand()

**Catalogue > ∫Σ > Algèbre**

**tExpand(***Expression1***)** ⇒ *expression*

Donne une expression dans laquelle les sinus et les cosinus de multiples entiers d'angles, de sommes d'angles et de différences d'angles sont développés. En raison de la présence de l'identité (sin(x))2+(cos(x))2=1, il existe plusieurs résultats équivalents possibles. Par conséquent, un résultat peut différer d'un autre résultat affiché dans d'autres publications.

Quelquefois, **tExpand()** permet d'atteindre vos objectifs lorsque le développement trigonométrique n'y parvient pas. **tExpand()** tend à faire l'inverse des transformations effectuées par **tCollect()**. Parfois, l'application de **tCollect()** à un résultat de **tExpand()**, ou vice versa, permet en deux étapes de simplifier une expression.

**Remarque :** la conversion en degrés par π/180 peut interférer avec la capacité de **tExpand()** de reconnaître les formes pouvant être développées. Pour de meilleurs résultats, **tExpand()** doit être utilisé en mode Angle en radians.

tExpand(sin(3ϕ))

$4 \cdot \sin(\phi) \cdot (\cos(\phi))^2 - \sin(\phi)$

tExpand(cos(α− β))

$\cos(\alpha) \cdot \cos(\beta) + \sin(\alpha) \cdot \sin(\beta)$

## Then

Voir **If,** page 89.

## TInterval

**Catalogue > ∫Σ > Statistiques**

**TInterval** [*Liste*,*Fréq*,*CLevel*]
(Entrée de liste de données)

**TInterval** $\bar{x}$,*Sx*,*n*[,*CLevel*]
(Récapitulatif des statistiques fournies en entrée)

Calcule un intervalle de confiance *t*. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **CLower, CUpper** | Intervalle de confiance pour une moyenne inconnue de population. |
| $\bar{x}$ | Moyenne d'échantillon de la série de données suivant la loi normale aléatoire |
| **ME** | Marge d'erreur |
| **df** | Degrés de liberté |
| **σx** | Écart-type d'échantillon |
| **n** | Taille de la série de données avec la moyenne d'échantillon |

## TInterval_2Samp

**Catalogue > ∫Σ > Statistiques**

**TInterval_2Samp** [*Liste1*,*Liste2*,*Fréq1*,*Fréq2*,*CLevel*,*Group*]
(Entrée de liste de données)

**TInterval_2Samp** $\bar{x}$*1*,*Sx1*,*n1*,$\bar{x}$*2*,*Sx2*,*n2*[,*CLevel*,*Group*]
(Récapitulatif des statistiques fournies en entrée)

Calcule un intervalle de confiance *t* sur 2 échantillons. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

*Group*=**1** met en commun les variances ; *Groupe*=**0** ne met pas en commun les variances.

| Variable de sortie | Description |
|---|---|
| **CLower, CUpper** | Intervalle de confiance contenant la probabilité du niveau de confiance de la loi |
| $\bar{x}$**1-**$\bar{x}$**2** | Moyennes d'échantillon des séries de données suivant la loi normale aléatoire |
| **ME** | Marge d'erreur |
| **df** | Degrés de liberté |
| $\bar{x}$**1,** $\bar{x}$**2** | Moyennes d'échantillon des séries de données suivant la loi normale aléatoire |
| $\sigma$**x1,** $\sigma$**x2** | Écarts-types d'échantillon pour *Liste 1* et *Liste 2*. |
| **n1, n2** | Nombre d'échantillons dans les séries de données |
| **sp** | Écart-type du groupe. Calculé lorsque *Group* = YES. |

## tmpCnv()

**Catalogue >**

**tmpCnv(***Expression1*_°*unitéTemp1*, _°*unitéTemp2***)**
⇒ *expression* _°*unitéTemp2*

Convertit une valeur de température spécifiée par *Expression1* d'une unité à une autre. Les unités de température utilisables sont :

_°C Celsius

_°F Fahrenheit

_°K Kelvin

_°R Rankine

Pour taper ° à partir du clavier, sélectionnez-le dans les symboles du Catalogue.

Pour taper _ , appuyez sur ctrl ␣.

Par exemple, 100_°C donne 212_°F.

Pour convertir un écart de température, utilisez Δ**tmpCnv()**.

tmpCnv(100_°c,_°f)
212. • _°F

tmpCnv(32_°f,_°c)
0. • _°C

tmpCnv(0_°c,_°k)
273.15 • _°K

tmpCnv(0_°f,_°r)
459.67 • _°R

**Remarque :** vous pouvez utiliser le Catalogue pour sélectionner des unités de température.

## Δ**tmpCnv()**

**Catalogue >**

Δ**tmpCnv(***Expression1*_°*unitéTemp1*, _°*unitéTemp2***)**
⇒ *expression* _°*unitéTemp2*

Convertit un écart de température (la différence entre deux valeurs de température) spécifié par *Expression1* d'une unité à une autre. Les unités de température utilisables sont :

_°C Celsius
_°F Fahrenheit
_°K Kelvin
_°R Rankine

Pour taper ° à partir du clavier, sélectionnez-le dans les symboles du Catalogue.

Pour taper _ , appuyez sur ctrl ␣.

Des écarts de 1_°C et 1_°K représentent la même grandeur, de même que 1_°F et 1_°R. Par contre, un écart de 1_°C correspond au 9/5 d'un écart de 1_°F.

Par exemple, un écart de 100_°C (de 0_°C à 100_°C) est équivalent à un écart de 180_°F.

Pour convertir une valeur de température particulière au lieu d'un écart, utilisez la fonction **tmpCnv()**.

Pour taper Δ, sélectionnez-le dans les symboles du Catalogue.

Δtmpcnv(100_°c,_°f)
180. • _°F

Δtmpcnv(180_°f,_°c)
100. • _°C

Δtmpcnv(100_°c,_°k)
100. • _°K

Δtmpcnv(100_°f,_°r)
100. • _°R

Δtmpcnv(1_°c,_°f)
1.8 • _°F

**Remarque :** vous pouvez utiliser le Catalogue pour sélectionner des unités de température.

## **tPdf()**

**Catalogue > > Probabilité**

**tPdf(***ValX***,***df***)**
Calcule la densité de probabilité (pdf) de la loi Student-*t* en *ValX* à *df* degrés de liberté. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

## **tTest**

**Catalogue > > Statistiques**

**tTest** μ*0*,*Liste*[,*Fréq*[,*Hypoth*]]
(Entrée de liste de données)

**tTest** μ*0*,$\overline{x}$,*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. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Test de H0 : μ = μ0, en considérant que :

*Hypoth* < 0 pour Ha : μ < μ0

*Hypoth* = 0 pour Ha : μ ≠ μ0 (par défaut)

*Hypoth* > 0 pour Ha : μ > μ0

| **Variable de sortie** | **Description** |
|---|---|
| **μ0** | Moyenne de pouplation connue pour la série de données x. |
| **t** | ($\overline{x}$ - μ0)/( stdev/ √(n)) |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **df** | Degrés de liberté |

| Variable de sortie | Description |
|---|---|
| $\bar{x}$ | Moyenne d'échantillon de la série de données dans *Liste* |
| **σx** | Écart-type d'échantillon de la série de données |
| **n** | Taille de l'échantillon |

### tTest_2Samp

**Catalogue > ∫Σ > Statistiques**

**tTest_2Samp** [*Liste1*,*Liste2*,*Fréq1*,*Fréq2*,*Hypoth*,*Group*]
(Entrée de liste de données)

**tTest_2Samp** $\bar{x}$*1*,*Sx1*,*n1*,$\bar{x}$*2*,*Sx2*,*n2*[,*Hypoth*,*Group*]
(Récapitulatif des statistiques fournies en entrée)

Effectue un test *t* sur deux échantillons. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Test de H0 : μ1 = μ2, en considérant que :

*Hypoth* < 0 pour Ha : μ1 < μ2

*Hypoth* = 0 pour Ha : μ1 ≠ μ2 (par défaut)

*Hypoth* > 0 pour Ha : μ1 > μ2

*Group*=**1** met en commun les variances
*Group*=**0** ne met pas en commun les variances

| Variable de sortie | Description |
|---|---|
| **z** | Valeur normale type calculée pour la différence des moyennes |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| $\bar{x}$**1,** $\bar{x}$**2** | Moyennes d'échantillon des séquences de données dans *Liste 1* et *Liste 2*. |
| **σx1, σx2** | Écarts-types d'échantillon des séries de données dans *Liste 1* et *Liste 2*. |
| **n1, n2** | Taille des échantillons |
| **σ1, σ2** | Écarts-types de population de *Liste 1* et *Liste 2*. |

### TwoVar

**Catalogue > ∫Σ > Statistiques**

**TwoVar** *X*, *Y*[, [*Fr2q*] [, *Catégorie*, *Inclure*]]
Calcule des statistiques pour deux variables. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Tous les arguments doivent être de mêmes dimensions, à l'exception de *Inclure*.

*X* représente la liste des valeurs de x.
*Y* représente la liste des valeurs de y.
*Fréq* représente la liste des fréquences.
*Catégorie* représente les numéros de catégories.
*Inclure* représente la liste des catégories à utiliser.

| Variable de sortie | Description |
|---|---|
| $\bar{x}$ | Moyenne des valeurs x |
| **Σx** | Somme des valeurs x |
| **Σx2** | Somme des valeurs x2 |

| Variable de sortie | Description |
|---|---|
| **sx** | Écart-type de l'échantillon de x |
| **σx** | Écart-type de la population de x |
| **n** | Nombre de points de données |
| $\bar{y}$ | Moyenne des valeurs y |
| **Σy** | Somme des valeurs y |
| $\Sigma y^2$ | Somme des valeurs y2 |
| **sy** | Écart-type de y dans l'échantillon |
| **σy** | Écart-type de population des valeurs de y |
| **Σxy** | Somme des valeurs x • y |
| **MinX** | Minimum des valeurs de x |
| $q_1x$ | 1er quartile de x |
| **MedX** | Médiane de x |
| $q_3x$ | 3ème quartile de x |
| **MaxX** | Maximum des valeurs de x |
| **MinY** | Minimum des valeurs de y |
| $q_1y$ | 1er quartile de y |
| **MedY** | Médiane de y |
| $q_3y$ | 3ème quartile de y |
| **MaxY** | Maximum des valeurs y |
| $\Sigma(x-\bar{x})^2$ | Somme des carrés des écarts par rapport à la moyenne de x |
| $\Sigma(y-\bar{y})^2$ | Somme des carrés des écarts par rapport à la moyenne de y |

# U

## unitV()

**Catalogue > ∫Σ > Matrice**

**unitV(***Vecteur1***)** ⇒ *vecteur*

Donne un vecteur unitaire ligne ou colonne, en fonction de la nature de *Vecteur1*.

*Vecteur1* doit être une matrice d'une seule ligne ou colonne.

unitV([a b c])

$$\left[\frac{a}{\sqrt{a^2+b^2+c^2}} \quad \frac{b}{\sqrt{a^2+b^2+c^2}} \quad \frac{c}{\sqrt{a^2+b^2+c^2}}\right]$$

unitV([1 2 1])

$$\left[\frac{\sqrt{6}}{6} \quad \frac{\sqrt{6}}{3} \quad \frac{\sqrt{6}}{6}\right]$$

unitV($\begin{bmatrix}1\\2\\3\end{bmatrix}$)

$$\begin{bmatrix}\frac{\sqrt{14}}{14}\\ \frac{\sqrt{14}}{7}\\ \frac{3\cdot\sqrt{14}}{14}\end{bmatrix}$$

# V

## varPop()

**Catalogue > ∫Σ > Statistiques**

**varPop(***Liste***[,** *listeFréq***])** ⇒ *expression*

Donne la variance de population de *Liste*.

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

**Remarque :** *Liste* doit contenir au moins deux éléments.

varPop({5,10,15,20,25,30})

$$\frac{875}{12}$$

Ans • 1.

72.916...

## varSamp()

**Catalogue ∫Σ Statistiques**

**varSamp(***Liste***[,** *listeFréq***])** ⇒ *expression*

Donne la variance d'échantillon de *Liste*.

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

**Remarque :** *Liste* doit contenir au moins deux éléments.

varSamp({a,b,c})

$$\frac{a^2-a\cdot(b+c)+b^2-b\cdot c+c^2}{3}$$

varSamp({1,2,5, ⁻6,3, ⁻2})

31/2

varSamp({1,3,5},{4,6,2})

68/33

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

Donne un vecteur ligne contenant la variance d'échantillon de chaque colonne de *Matrice1*.

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

**Remarque :** *Matrice1* doit contenir au moins deux lignes.

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]

# W

## when()

**Catalogue >**

**when(***Condition***,** *résultSiOui* **[,** *résultSiNon***][,** *résultSiInconnu***])** ⇒ *expression*

Donne *résultSiOui*, *résultSiNon* ou *résultSiInconnu*, suivant que la *Condition* est vraie, fausse ou indéterminée. Donne l'entrée si le nombre d'argument est insuffisant pour spécifier le résultat approprié.

Ne spécifiez pas *résultSiNon* ni *résultSiInconnu* pour obtenir une expression définie uniquement dans la région où *Condition* est vraie.

Utilisez **undef** *résultSiNon* pour définir une expression représentée graphiquement sur un seul intervalle.

**when()** est utile dans le cadre de la définition de fonctions récursives.

```
when(x<0,x+3)|x=5
                    when(x<0,3+x)
```

```
when(n>0,n • factorial(n−1),1)
→ factorial(n)
                              Done
factorial(3)
                                6
3!
                                6
```

## While

**Barre d'outils Calculs >** **> Structures de contrôle**

**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 « : ».

Exemple de fonction :

```
Define sum_of_recip(n)=Func
:  Local i, tempsum
:  1 → i
:  0 → tempsum
:  While 1 ≤ n
:     sommetemp+1/i → sommetemp
:     i+1 → i
:  EndWhile
:  Return sommetemp
:EndFunc
```

**« With »**

Voir page 165.

# X

## xor

**Catalogue >** **> Test**

*Expression booléenne1* **xor** *Expression booléenne2* ⇒ *Expression booléenne*

Donne true si *Expression booléenne1* est vraie et si *Expression booléenne2* est fausse, ou vice versa.

Donne false si *Expression booléenne1* et *Expression booléenne2* sont toutes les deux vraies ou fausses. Donne une expression booléenne simplifiée si l'une des deux expressions booléennes initiales ne peut être résolues vraie ou fausse.

**Remarque :** voir **or**.

```
true xor true
                              false
(5>3) xor (3>5)
                               true
```

## xor

**Catalogue > ∫Σ > Test**

*Entier1* **xor** *Entier2* ⇒ *entier*

Compare les représentations binaires de deux entiers, en appliquant un **xor** bit par bit. En interne, les deux entiers sont convertis en nombres binaires 32 bits signés. Lorsque les bits comparés correspondent, le résultat est 1 si dans l'un des deux cas (pas dans les deux) il s'agit d'un bit 1 ; le résultat est 0 si, dans les deux cas, il s'agit d'un bit 0 ou 1. La valeur donnée représente le résultat des bits et elle est affichée selon le mode Base utilisé.

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

Si vous entrez un nombre dont le codage binaire signé dépasse 32 bits, il est ramené à l'aide d'une congruence dans la plage appropriée.

**Remarque :** voir **or**.

En mode base Hex :

0h7AC36 xor 0h3D5F

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

0h79169

En mode base Bin :

0b100101 xor 0b100

0b100001

**Remarque :** une entrée binaire peut comporter jusqu'à 32 chiffres (sans compter le préfixe 0b) ; une entrée hexadécimale jusqu'à 8 chiffres.

# z

## zeros()

**Catalogue > ∫Σ > Algèbre**

**zeros(***Expression***,** *Var***)** ⇒ *liste*

Donne une liste des valeurs réelles possibles de *Var* avec lesquelles *Expression*=0. Pour y parvenir, **zeros()** calcule **exp▶list(solve(***Expression*=0,*Var***),***Var***)**.

zeros(a • x²+b • x+c,x)

$$\left\{\frac{-\left(\sqrt{b^2-4\cdot a\cdot c}+b\right)}{2\cdot a},\frac{\sqrt{b^2-4\cdot a\cdot c}-b}{2\cdot a}\right\}$$

a • x²+b • x+c|x=ans[2]

0

Dans certains cas, la nature du résultat de **zeros()** est plus satisfaisante que celle de **solve()**. Toutefois, la nature du résultat de **zeros()** ne permet pas d'exprimer des solutions implicites, des solutions nécessitant des inéquations ou des solutions qui n'impliquent pas *Var*.

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

exact(zeros(a • ($e^x$+x)

(sign (x)−1),x))

{}

exact(solve(a • ($e^x$+x)

(sign (x)−1)=0,x))

ex + x = 0 or x>0 or a = 0

**zeros({***Expression1***,** *Expression2***},**

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

Donne les zéros réels possibles des expressions algébriques simultanées, où chaque *VarOuInit* spécifie une inconnue dont vous recherchez la valeur.

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

*variable*

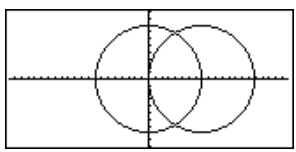
– ou –

*variable = nombre réel ou non réel*

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

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

Par exemple, si vous avez un cercle de rayon r centré à l'origine et un autre cercle de rayon r centré, au point où le premier cercle coupe l'axe des x positifs. Utilisez **zeros()** pour trouver les intersections.



## zeros()

**Catalogue > ∫Σ > Algèbre**

Comme l'illustre r dans l'exemple ci-contre, des expressions polynomiales simultanées peuvent avoir des variables supplémentaires sans valeur assignée, mais représenter des valeurs numériques données pouvant s'y substituer par la suite.

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

zeros({$x^2+y^2-r^2$, $(x-r)^2+y^2-r^2$},{x,y})

$$\begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} \\ \frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2} \end{bmatrix}$$

Extraction ligne 2 :

ans[2]

$$\begin{bmatrix} \frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2} \end{bmatrix}$$

Vous pouvez également utiliser des inconnues qui n'apparaissent pas dans les expressions. Par exemple, vous pouvez utiliser z comme inconnue pour développer l'exemple précédent et avoir deux cylindres parallèles sécants de rayon r. La résolution du problème montre comment des groupes de zéros peuvent contenir des constantes arbitraires de type ck, où k est un suffixe entier compris entre 1 et 255.

zeros({$x^2+y^2-r^2$, $(x-r)^2+y^2-r^2$},{x,y,z})

$$\begin{bmatrix} \frac{r}{2} & \frac{\sqrt{3}\cdot r}{2} & @1 \\ \frac{r}{2} & \frac{-\sqrt{3}\cdot r}{2} & @1 \end{bmatrix}$$

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

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

zeros({$x+e^z\cdot y-1$,$x-y-\sin(z)$},{x,y})

$$\begin{bmatrix} \frac{e^z\cdot\sin(z)+1}{e^z+1} & \frac{-(\sin(z)-1)}{e^z+1} \end{bmatrix}$$

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

Chaque inconnue commence à sa valeur supposée, si elle existe ; sinon, la valeur de départ est 0.0.

zeros({$e^z\cdot y-1$, $-y-\sin(z)$},{y,z})

[.041… 3.183…]

Utilisez des valeurs initiales pour rechercher des zéros supplémentaires, un par un. Pour assurer une convergence correcte, une valeur initiale doit être relativement proche d'un zéro.

zeros({$e^z\cdot y-1$, $-y-\sin(z)$}, {y,z=2π})

[.001… 6.281…]

## zInterval

**Catalogue > ∫Σ > Statistiques**

**zInterval** σ,*Liste*[,*Fréq*[,*CLevel*]]
(Entrée de liste de données)

**zInterval** σ,$\bar{x}$,*n* [,*CLevel*]
(Récapitulatif des statistiques fournies en entrée)

Calcule un intervalle de confiance *z*. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **CLower, CUpper** | Intervalle de confiance pour une moyenne inconnue de population |
| $\bar{x}$ | Moyenne d'échantillon de la série de données suivant la loi normale aléatoire |
| **ME** | Marge d'erreur |

| Variable de sortie | Description |
|---|---|
| **sx** | Écart-type d'échantillon |
| **n** | Taille de la série de données avec la moyenne d'échantillon |
| σ | Écart-type connu de population pour la série de données *Liste*. |

**zInterval_1Prop** **Catalogue > ∫Σ > Stastistiques**

**zInterval_1Prop** *x*,*n* [,*CLevel*]
Calcule un intervalle de confiance *z* pour une proportion. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **CLower, CUpper** | Intervalle de confiance contenant la probabilité du niveau de confiance de la loi |
| **phat** | Proportion calculée de réussite |
| **ME** | Marge d'erreur |
| **n** | Nombre d'échantillons dans la série de données |

**zInterval_2Prop** **Catalogue > ∫Σ > Statistiques**

**zInterval_2Prop** *x1*,*n1*,*x2*,*n2*[,*CLevel*]
Calcule un intervalle de confiance *z* pour deux proportions. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **CLower, CUpper** | Intervalle de confiance contenant la probabilité du niveau de confiance de la loi |
| **phatDiff** | Différence calculée entre les proportions |
| **ME** | Marge d'erreur |
| **phat1** | Proportion calculée sur le premier échantillon |
| **phat2** | Proportion calculée sur le deuxième échantillon |
| **n1** | Taille de l'échantillon dans la première série de données |
| **n2** | Taille de l'échantillon dans la deuxième série de données |

**zInterval_2Samp** **Catalogue > ∫Σ > Statistiques**

**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. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

| Variable de sortie | Description |
|---|---|
| **CLower, CUpper** | Intervalle de confiance contenant la probabilité du niveau de confiance de la loi |
| $\bar{x}$**1-**$\bar{x}$**2** | Moyennes d'échantillon des séries de données suivant la loi normale aléatoire |
| **ME** | Marge d'erreur |
| $\bar{x}$**1,** $\bar{x}$**2** | Moyennes d'échantillon des séries de données suivant la loi normale aléatoire |
| σ**x1,** σ**x2** | Écarts-types d'échantillon pour *Liste 1* et *Liste 2*. |
| **n1, n2** | Nombre d'échantillons dans les séries de données |
| **r1, r2** | Écart-type connu de population pour la série de données *Liste 1* et *Liste 2*. |

## zTest

**Catalogue > ∫Σ > Statistiques**

**zTest** μ*0*,σ,*Liste*[,*Fréq*[,*Hypoth*]]
(Entrée de liste de données)

**zTest** μ*0*,σ,$\bar{x}$,*n*[,*Hypoth*]
(Récapitulatif des statistiques fournies en entrée)

Effectue un test *z* en utilisant la fréquence *listeFréq*. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Test de H0 : μ = μ0, en considérant que :

*Hypoth* < 0 pour Ha : μ < μ0

*Hypoth* = 0 pour Ha : μ ≠ μ0 (par défaut)

*Hypoth* > 0 pour Ha : μ > μ0

| Variable de sortie | Description |
|---|---|
| **z** | ($\bar{x}$ - μ0)/( σ / /(n)) |
| **P Value** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| $\bar{x}$ | Moyenne d'échantillon de la série de données dans *Liste* |
| **sx_** | Écart-type d'échantillon de la série de données. Uniquement donné pour l'entrée *Data*. |
| **n** | Taille de l'échantillon |

## zTest_1Prop

**Catalogue > ∫Σ > Statistiques**

**zTest_1Prop** *p0*,*x*,*n*[,*Hypoth*]
Effectue un test *z* pour une proportion unique. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)

Test de H0 : *p = p0*, en considérant que :

*Hypoth* > 0 pour Ha : *p > p0*

*Hypoth* = 0 pour Ha : *p ≠ p0 (par défaut)*

*Hypoth* < 0 pour Ha : *p < p0*

| Variable de sortie | Description |
|---|---|
| **p0** | Proportion de population hypothétique |
| **z** | Valeur normale type calculée pour la proportion |

| Variable de sortie | Description |
|---|---|
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **p_hat** | Proportion calculée sur l'échantillon |
| **n** | Taille de l'échantillon |

**zTest_2Prop** **Catalogue > ∫Σ > Statistiques**

**zTest_2Prop** *x1*,*n1*,*x2*,*n2*[,*Hypoth*]
Calcule un test *z* pour deux proportions. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)
Test de H0 : $p1 = p2$, en considérant que :
*Hypoth* > 0 pour Ha : $p1 > p2$
*Hypoth* = 0 pour Ha : $p1 \neq p2$ (par défaut)

| Variable de sortie | Description |
|---|---|
| **z** | Valeur normale type calculée pour la différence des proportions |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| **phat1** | Proportion calculée sur le premier échantillon |
| **phat2** | Proportion calculée sur le deuxième échantillon |
| **phat** | Proportion calculée de l'échantillon mis en commun |
| **n1, n2** | Nombre d'échantillons pris lors des essais 1 et 2 |

**zTest_2Samp** **Catalogue > ∫Σ > Statistiques**

**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. Le résultat est stocké dans la variable StatMatrix. (Voir page 133.)
Test de H0 : $\mu1 = \mu2$, en considérant que :
*Hypoth* < 0 pour Ha : $\mu1 < \mu2$
*Hypoth* = 0 pour Ha : $\mu1 \neq \mu2$ (par défaut)
*Hypoth* > 0 pour Ha : $\mu1 > \mu2$

| Variable de sortie | Description |
|---|---|
| **z** | Valeur normale type calculée pour la différence des moyennes |
| **PVal** | Plus petite probabilité permettant de rejeter l'hypothèse nulle |
| $\bar{x}$**1**, $\bar{x}$**2** | Moyennes d'échantillon des séquences de données dans *Liste 1* et *Liste 2*. |
| $\sigma$**x1**, $\sigma$**x2** | Écarts-types d'échantillon des séries de données dans *Liste 1* et *Liste 2*. |
| **n1, n2** | Taille des échantillons |
| $\sigma$**1**, $\sigma$**2** | Écarts-types de population de *Liste 1* et *Liste 2*. |

# Symboles

## + (somme)

Touche

*Expression1* + *Expression2* ⇒ *expression*
Donne la somme de *Expression1* et *Expression2*.

56
56
ans+4
60
ans+4
64
ans+4
68
ans+4
72

*Liste1* + *Liste2* ⇒ *liste*
*Matrice1* + *Matrice2* ⇒ *matrice*
Donne une liste (ou une matrice) contenant les sommes des éléments correspondants de *Liste1* et *Liste2* (ou *Matrice1* et *Matrice2*).
Les arguments doivent être de même dimension.

{22,π,π/2} ➞ L1
{22,π,π/2}
{10,5,π/2} ➞ L2
{10,5,π/2}
L1+L2
{32,π+5,π}
ans+{π, ⁻5, ⁻π}
{π+32,π,0}

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} a+1 & b \\ c & d+1 \end{bmatrix}$$

*Expression* + *Liste1* ⇒ *liste*
*Liste1* + *Expression* ⇒ *liste*
Donne une liste contenant les sommes de *Expression* et de chaque élément de *Liste1*.

15+{10,15,20}
{25,30,35}
{10,15,20}+15
{25,30,35}

*Expression* + *Matrice1* ⇒ *matrice*
*Matrice1* + *Expression* ⇒ *matrice*
Donne une matrice obtenue en ajoutant *Expression* à chaque élément de la diagonale de *Matrice1*. *Matrice1* doit être carrée.
**Remarque :** utilisez **.+** pour ajouter une expression à chaque élément de la matrice.

$$20+\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 21 & 2 \\ 3 & 24 \end{bmatrix}$$

## − (soustraction)

Touche

*Expression1* − *Expression2* ⇒ *expression*
Donne la différence de *Expression1* et de *Expression2*.

6−2
4
π−π/6

$$\frac{5 \cdot \pi}{6}$$

*Liste1* − *Liste2* ⇒ *liste*
*Matrice1* − *Matrice2* ⇒ *matrice*
Soustrait chaque élément de *Liste2* (ou *Matrice2*) de l'élément correspondant de *Liste1* (ou *Matrice1*) et donne le résultat obtenu.
Les arguments doivent être de même dimension.

{22,π,π/2}−{10,5,π/2}
{12,π−5,0}
[3 4]−[1 2]
[2 2]

## −(soustraction)

Touche −

*Expression* − *Liste1* ⇒ *liste*

*Liste1* − *Expression* ⇒ *liste*

Soustrait chaque élément de *Liste1* de *Expression* ou soustrait *Expression* de chaque élément de *Liste1* et donne la liste de résultats obtenue.

15−{10,15,20}

{5,0,-5}

{10,15,20}−15

{-5,0,5}

*Expression* − *Matrice1* ⇒ *matrice*

*Matrice1* − *Expression* ⇒ *matrice*

*Expression* − *Matrice1* donne une matrice de *Expression* fois la matrice d'identité moins *Matrice1*. *Matrice1* doit être carrée.

*Matrice1* − *Expression* donne une matrice obtenue en soustrayant *Expression* à chaque élément de la diagonale de *Matrice1*. *Matrice1* doit être carrée.

20−$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

$\begin{bmatrix} 19 & -2 \\ -3 & 16 \end{bmatrix}$

**Remarque :** utilisez **.**− pour soustraire une expression de chaque élément de la matrice.

## ·(multiplication)

Touche ×

*Expression1* · *Expression2* ⇒ *expression*

Donne le produit de *Expression1* et de *Expression2*.

2·3.45

6.9

x·y·x

$x^2 \cdot y$

*Liste1* · *Liste2* ⇒ *liste*

Donne une liste contenant les produits des éléments correspondants de *Liste1* et *Liste2*.

Les listes doivent être de même dimension.

{1.0,2,3}·{4,5,6}

{4.,10,18}

{2/a,3/2}·{$a^2$,b/3}

$\{2 \cdot a \ \frac{b}{2}\}$

*Matrice1* · *Matrice2* ⇒ *matrice*

Donne le produit des matrices *Matrice1* et *Matrice2*.

Le nombre de lignes de *Matrice1* doit être égal au nombre de colonnes de *Matrice2*.

$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} . \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$

$\blacksquare \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \cdot \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$

$\begin{bmatrix} a+2\cdot b+3\cdot c & d+2\cdot e+3 \\ 4\cdot a+5\cdot b+6\cdot c & 4\cdot d+5\cdot e \end{bmatrix}$

*Expression* · *Liste1* ⇒ *liste*

*Liste1* · *Expression* ⇒ *liste*

Donne une liste des produits de *Expression* et de chaque élément de *Liste1*.

π·{4,5,6}

{4·π,5·π,6·π}

*Expression* · *Matrice1* ⇒ *matrice*

*Matrice1* · *Expression* ⇒ *matrice*

Donne une matrice contenant les produits de *Expression* et de chaque élément de *Matrice1*.

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} . .01$

$\begin{bmatrix} .01 & .02 \\ .03 & .04 \end{bmatrix}$

**Remarque :** utilisez **.**· pour multiplier une expression par chaque élément.

λ·identity(3)

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

## / (division)

**Touche**

*Expression1* / *Expression2* ⇒ *expression*
Donne le quotient de *Expression1* par *Expression2*.

2/3.45

.57971

$x^3/x$

$x^2$

*Liste1* / *Liste2* ⇒ *liste*
Donne une liste contenant les quotients de *Liste1* par *Liste2*.
Les listes doivent être de même dimension.

{1.0,2,3}/{4,5,6}

{.25,2/5,1/2}

*Expression* / *Liste1* ⇒ *liste*
*Liste1* / *Expression* ⇒ *liste*
Donne une liste contenant les quotients de *Expression* par *Liste1* ou *Liste1* par *Expression*.

a/{3,a,√(a)}

$\left\{\frac{a}{3}\ 1\ \frac{\sqrt{a}}{a}\right\}$

{a,b,c}/(a • b • c)

$\left\{\frac{1}{b\cdot c}\ \frac{1}{a\cdot c}\ \frac{1}{a\cdot b}\right\}$

*Expression* / *Matrice1* ⇒ *matrice*
*Matrice1* / *Expression* ⇒ *matrice*
Donne une matrice contenant les quotients des éléments de *Matrice1* παρ *Expression*.
**Remarque :** utilisez **. /** pour diviser une expression par chaque élément.

[a b c]/(a • b • c)

$\begin{bmatrix}\frac{1}{b\cdot c} & \frac{1}{a\cdot c} & \frac{1}{a\cdot b}\end{bmatrix}$

## ^ (puissance)

**Touche**

*Expression1* ^ *Expression2* ⇒ *expression*
*Liste1* ^ *Liste2* ⇒ *liste*
Donne le premier argument élevé à la puissance du deuxième argument.
Dans le cas d'une liste, donne les éléments de *Liste1* élevés à la puissance des éléments correspondants de *Liste2*.
Dans le domaine réel, les puissances fractionnaires possédant des exposants réduits avec des dénominateurs impairs utilise la branche réelle, tandis que le mode complexe utilise la branche principale.

$4^2$

16

$\{a,2,c\}^{\{1,b,3\}}$

$\{a,2^b,c^3\}$

*Expression* ^ *Liste1* ⇒ *liste*
Donne *Expression* élevé à la puissance des éléments de *Liste1*.

$p^{\{a,2,-3\}}$

$\{p^a\ p^2\ \frac{1}{p^3}\}$

*Liste1* ^ *Expression* ⇒ *liste*
Donne les éléments de *Liste1* élevés à la puissance de l'*expression*.

$\{1,2,3,4\}^{-2}$

{1,1/4,1/9,1/16}

## ^ (puissance)

**Touche**

*matriceCarrée1* **^** *entier* ⇒ *matrice*

Donne *matriceCarrée1* élevée à la puissance de la valeur de l'*entier*. *matriceCarrée1* doit être une matrice carrée.

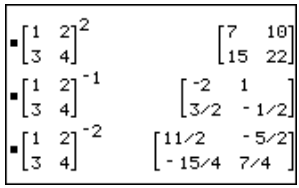
Si *entier* = ⁻1, calcule la matrice inverse.

Si *entier* < ⁻1, calcule la matrice inverse à une puissance positive appropriée.

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

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

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




## .+

**Touches**

*Matrice1* **.+** *Matrice2* ⇒ *matrice*

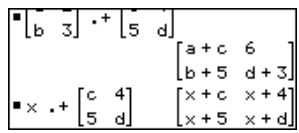
*Expression* **.+** *Matrice1* ⇒ *matrice*

*Matrice1* **.+** *Matrice2* donne une matrice obtenue en effectuant la somme de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

*Expression* **.+** *Matrice1* donne une matrice obtenue en effectuant la somme de *Expression* et de chaque élément de *Matrice1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$$

$$x \ .+ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$$




## .−

**Touches**

*Matrice1* **.−** *Matrice2* ⇒ *matrice*

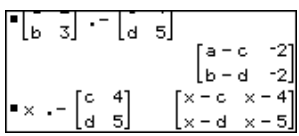
*Expression* **.−***Matrice1* ⇒ *matrice*

*Matrice1* **.−***Matrice2* donne une matrice obtenue en calculant la différence entre chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

*Expression* **.−***Matrice1* donne une matrice obtenue en calculant la différence de *Expression* et de chaque élément de *Matrice1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix}$$

$$x \ .- \begin{bmatrix} c & 4 \\ d & 5 \end{bmatrix}$$



## .·

**Touches** ·, |x|

*Matrice1* **.·** *Matrice2* ⇒ *matrice*

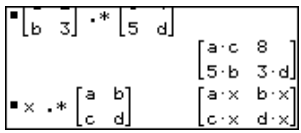
*Expression* **.·** *Matrice1* ⇒ *matrice*

*Matrice1* **.·** *Matrice2* donne une matrice obtenue en calculant le produit de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

*Expression* **.·** *Matrice1* donne une matrice contenant les produits de *Expression* et de chaque élément de *Matrice1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .\cdot \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$$

$$x \,.\cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$



## ./

**Touches** ·, ÷

*Matrice1* **./** *Matrice2* ⇒ *matrice*

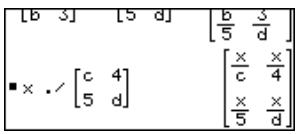
*Expression* **./** *Matrice1* ⇒ *matrice*

*Matrice1* **./** *Matrice2* donne une matrice obtenue en calculant le quotient de chaque paire d'éléments correspondants de *Matrice1* et de *Matrice2*.

*Expression* **./** *Matrice1* donne une matrice obtenue en calculant le quotient de *Expression* et de chaque élément de *Matrice1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} ./ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$$

$$x \,./ \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$$



## .^

**keys** ·, ^

*Matrice1* **.^** *Matrice2* ⇒ *matrice*

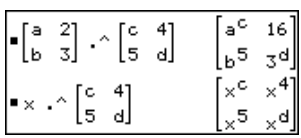
*Expression* **.^** *Matrice1* ⇒ *matrice*

*Matrice1* **.^** *Matrice2* donne une matrice obtenue en appliquant la puissance de chaque élément correspondant de *Matrice1* à chaque élément de *Matrice1*.

*Expression* **.^** *Matrice1* donne une matrice obtenue en appliquant la puissance de *Expression* à chaque élément de *Matrice1*.

$$\begin{bmatrix} a & 2 \\ b & 3 \end{bmatrix} .\wedge \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$$

$$x \,.\wedge \begin{bmatrix} c & 4 \\ 5 & d \end{bmatrix}$$



## ¯

**Touche** (ans (−))

¯*Expression1* ⇒ *expression*
¯*Liste1* ⇒ *liste*
¯*Matrice1* ⇒ *matrice*

Donne l'opposé de l'argument.

Dans le cas d'une liste ou d'une matrice, donne l'opposé de chacun des éléments.

Si *Expression1* est un entier binaire ou hexadécimal, donne le complément à deux.

¯2.43

¯2.43

¯{¯1,0.4,1.2E19}

{1, ¯.4, ¯1.2E19}

¯a • ¯b

a • b

En mode base Bin :

0b100101▸dec

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

37

¯0b100101

0b11111111111111111111111111011011

ans▸dec

¯37

## % (pourcentage)

**Catalogue > ∫Σ > Opérateurs mathématiques**

*Expression1* **%** ⇒ *expression*
*Liste1* **%** ⇒ *liste*
*Matrice1* **%** ⇒ *matrice*

Donne $\frac{argument}{100}$

Dans le cas d'une liste ou d'une matrice, donne une liste ou une matrice obtenue en divisant chaque élément par 100.

13% (ctrl) (≈ enter)

.13

{1, 10, 100}% (ctrl) (≈ enter)

{.01,.1,1.}

## = (égal à)

**Touche** (=)

*Expression1* **=** *Expression2* ⇒ *Expression booléenne*

*Liste1* **=** *Liste2* ⇒ *Liste booléenne*

*Matrice1* **=** *Matrice2* ⇒ *Matrice booléenne*

Donne true s'il est possible de vérifier que la valeur de *Expression1* est égale à celle de *Expression2*.

Donne false s'il est possible de déterminer que la valeur de *Expression1* n'est pas égale à celle de *Expression2*.

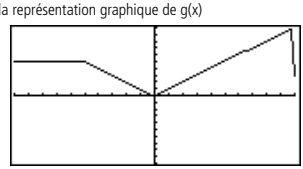
Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Exemple de fonction qui utilise les symboles de test mathématiques : =, ≠, <, ≤, >, ≥

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

Résultat de la représentation graphique de g(x)



### ≠ (différent de)

**Touche** ctrl =

*Expression1* ≠ *Expression2* ⇒ *Expression booléenne*

*Liste1* ≠ *Liste2* ⇒ *Liste booléenne*

*Matrice1* ≠ *Matrice2* ⇒ *Matrice booléenne*

Donne true s'il est possible de déterminer que la valeur de *Expression1* n'est pas égale à celle de *Expression2*.

Donne false s'il est possible de vérifier que la valeur de *Expression1* est égale à celle de *Expression2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Voir l'exemple fourni pour "=" (égal à).

### < (inférieur à)

**Touche** <

*Expression1* < *Expression2* ⇒ *Expression booléenne*

*Liste1* < *Liste2* ⇒ *Liste booléenne*

*Matrice1* < *Matrice2* ⇒ *Matrice booléenne*

Donne true s'il est possible de déterminer que la valeur de *Expression1* est inférieure à celle de *Expression2*.

Donne false s'il est possible de déterminer que la valeur de *Expression1* est supérieure ou égale à celle de *Expression2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Voir l'exemple fourni pour "=" (égal à).

### ≤ (inférieur ou égal à)

**Touche** ctrl <

*Expression1* ≤ *Expression2* ⇒ *Expression booléenne*

*Liste1* ≤ *Liste2* ⇒ *Liste booléenne*

*Matrice1* ≤ *Matrice2* ⇒ *Matrice booléenne*

Donne true s'il est possible de déterminer que la valeur de *Expression1* est inférieure ou égale à celle de *Expression2*.

Donne false s'il est possible de déterminer que la valeur de *Expression1* est supérieure à celle de *Expression2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Voir l'exemple fourni pour "=" (égal à).

### > (supérieur à)

**Touche** >

*Expression1* > *Expression2* ⇒ *Expression booléenne*

*Liste1* > *Liste2* ⇒ *Liste booléenne*

*Matrice1* > *Matrice2* ⇒ *Matrice booléenne*

Donne true s'il est possible de déterminer que la valeur de *Expression1* est supérieure à celle de *Expression2*.

Donne false s'il est possible de déterminer que la valeur de *Expression1* est inférieure ou égale à celle de *Expression2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Voir l'exemple fourni pour "=" (égal à).

### ≥ (supérieur ou égal à)

**Touche** ctrl >

*Expression1* ≥ *Expression2* ⇒ *Expression booléenne*

*Liste1* ≥ *Liste2* ⇒ *Liste booléenne*

*Matrice1* ≥ *Matrice2* ⇒ *Matrice booléenne*

Donne true s'il est possible de déterminer que la valeur de *Expression1* est supérieure ou égale à celle de *Expression2*.

Donne false s'il est possible de déterminer que la valeur de *Expression1* est inférieure à celle de *Expression2*.

Dans les autres cas, donne une forme simplifiée de l'équation.

Dans le cas d'une liste ou d'une matrice, donne le résultat des comparaisons, élément par élément.

Voir l'exemple fourni pour "=" (égal à).

### ! (factorielle)

**Catalogue > ∫Σ > Probabilité**

*Expression1*! ⇒ *expression*

*Liste1*! ⇒ *liste*

*Matrice1*! ⇒ *matrice*

Donne la factorielle de l'argument.

Dans le cas d'une liste ou d'une matrice, donne une liste ou une matrice des factorielles de tous les éléments.

5!

120

{5,4,3}!

{120,24,6}

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

$\begin{bmatrix} 1 & 2 \\ 6 & 24 \end{bmatrix}$

### & (ajouter)

**Catalog > α°**

*Chaîne1* **&** *Chaîne2* ⇒ *chaîne*

Donne une chaîne de caractères obtenue en ajoutant *Chaîne2* à *Chaîne1*.

"Hello " & "Nick"

"Hello Nick"

### d() (dérivée)

**Catalogue > ∫Σ > Analyse**

***d***(*Expression1*, *Var* [,*Ordre*]) ⇒ *expression*

***d***(*Liste1*,*Var* [,*Ordre*]) ⇒ *liste*

***d***(*Matrice1*,*Var* [,*Ordre*]) ⇒ *matrice*

Donne la dérivée première de *Expression1* par rapport à la variable *Var*. *Expression1* peut être une liste ou une matrice.

*Ordre*, si spécifié, doit être un entier. Si l'ordre spécifié est inférieur à zéro, on obtient une primitive.

***d***() n'applique pas la méthode de calcul standard qui consiste à simplifier entièrement ses arguments, puis à appliquer la définition de la fonction aux arguments simplifiés obtenus. Par contre, ***d***() procède de la façon suivante :

Il simplifie le deuxième argument uniquement dans la mesure où cette opération permet d'obtenir une variable.

Il simplifie le premier argument uniquement dans la mesure où cette opération appelle une valeur stockée pour la variable déterminée à l'étape 1.

Il détermine la dérivée symbolique du résultat obtenu à l'étape 2 par rapport à la variable générée à l'étape 1.

Si la variable déterminée à l'étape 1 a une valeur stockée ou une valeur spécifiée par un opérateur « with » (|), cette valeur est substituée dans le résultat obtenu à l'étape 3.

d(f(x) · g(x),x)

$\frac{d}{dx}(f(x)) \cdot g(x) + \frac{d}{dx}(g(x)) \cdot f(x)$

d(d(x² · y³,x),y)

6 · y² · x

d(x²,x,-1)

$\frac{x^3}{3}$

d({x²,x³,x⁴},x)

{2 · x,3 · x²,4 · x³}

## ∫() (intégrale)

**Catalogue > ∫Σ > Analyse**

∫(*Expression1*, *Var*[,*Borne1*] [,*Borne2*]) ⇒ *expression*
∫(*Liste1*,*Var* [,*Ordre*]) ⇒ *liste*
∫(*Matrice1*,*Var* [,*Ordre*]) ⇒ *matrice*

Donne l'intégrale de *Expression1* par rapport à la variable *Var* entre *Borne1* et *Borne2*.

∫(x²,x,a,b)

$$\frac{b^3}{3}-\frac{a^3}{3}$$

Donne une primitive si *Borne1* et *Borne2* sont omises. Une constante d'intégration est omise.

Toutefois, *Borne1* est ajoutée comme constante d'intégration si *Borne2* seulement est omise.

∫(x²,x)

$$\frac{x^3}{3}$$

∫(a • x²,x,c)

$$\frac{a\cdot x^3}{3}+c$$

Les primitives valides peut différer d'une constante numérique. Ce type de constante peut être masqué, notamment lorsqu'une primitive contient des logarithmes ou des fonctions trigonométriques inverses. De plus, des expressions constantes par morceaux sont parfois ajoutées pour assurer la validité d'une primitive sur un intervalle plus grand que celui d'une formule courante.

∫() donne lui-même pour les morceaux de *Expression1* qui ne peuvent pas être déterminés sous forme de combinaison explicite finie de fonctions et d'opérateurs intégrés.

Si *Borne1* et *Borne2* sont toutes les deux présentes, la fonction tente de localiser toute discontinuité ou dérivée discontinue comprise dans l'intervalle de *Borne1* < *Var* < *Borne2* et de subdiviser l'intervalle à ces emplacements.

∫(b • e^(⁻x²)+a/(x²+a²),x)

$$\int\left(b\cdot e^{-x^2}+\frac{a}{x^2+a^2}\right)dx$$
$$b\cdot\int\left(e^{-x^2}\right)dx+\tan^{-1}\left(\frac{x}{a}\right)$$

Avec le réglage Auto du mode Exact/Approché, l'intégration numérique est utilisée, si elle est applicable, chaque fois qu'une primitive ou une limite ne peut pas être déterminée.

Avec le réglage Approché, on procède en premier à une intégration numérique, si elle est applicable. Les primitives ne peuvent être trouvées que dans le cas où cette intégration numérique ne s'applique pas ou échoue.

∫(e^(⁻x²),x, ⁻1,1) ctrl ≈ enter

1.493...

∫() peut être imbriqué pour obtenir des intégrales multiples. Les bornes d'intégration peuvent dépendre des variables d'intégration les plus extérieures.

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

∫(∫(ln(x+y),y,0,x),x,0,a)

$$\int_0^a\int_0^x \ln(x+y)\,dy\,dx$$
$$\frac{a^2\cdot\ln(a)}{2}+a^2\cdot(\ln(2)-3/4)$$

## √() (racine carrée)

**Touche** $\sqrt{x^2}$

√ (*Expression1*) ⇒ *expression*
√ (*Liste1*) ⇒ *liste*

Donne la racine carrée de l'argument.

Dans le cas d'une liste, donne les racines carrées de tous les éléments de *Liste1*.

√(4)

2

√({9,a,4})

{3,√a,2}

## Π() (produit)

**Catalogue > ∫Σ > Liste**

**Π(***Expression1***,** *Var***,** *Début***,** *Fin***)** ⇒ *expression*
Calcule *Expression1* pour chaque valeur de *Var* comprise entre *Début* et *Fin* et donne le produit des résultats obtenus.

Π(1/n,n,1,5)

$$\frac{1}{120}$$

Π(k²,k,1,n)

$$(n!)^2$$

Π({1/n,n,2},n,1,5)

$$\left\{\frac{1}{120}\ 120\ 32\right\}$$

**Π(***Expression1***,** *Var***,** *Début***,** *Fin*−1**)** ⇒ 1

Π(k,k,4,3)

1

**Π(***Expression1***,** *Var***,** *Début***,** *Fin***)**
⇒ **1/Π(***Expression1***,** *Var***,** *Fin*+1**,** *Début*−1**)** if *Fin* < *Début*−1

Π(1/k,k,4,1)

6

Π(1/k,k,4,1) · Π(1/k,k,2,4)

1/4

## Σ() (somme)

**Catalogue > ∫Σ > Liste**

**Σ(***Expression1***,** *Var***,** *Début***,** *Fin***)** ⇒ *expression*
Calcule *Expression1* pour chaque valeur de *Var* comprise entre *Début* et *Fin* et donne la somme des résultats obtenus.

Σ(1/n,n,1,5)

$$\frac{137}{60}$$

Σ(k²,k,1,n)

$$\frac{n\cdot(n+1)\cdot(2\cdot n+1)}{6}$$

Σ(1/n²,n,1,∞)

$$\frac{\pi^2}{6}$$

**Σ(***Expression1***,** *Var***,** *Début***,** *Fin*−1**)** ⇒ 0

Σ(k,k,4,3)

0

**Σ(***Expression1***,** *Var***,** *Début***,** *Fin***)**
⇒ −**Σ(***Expression1***,** *Var***,** *Fin*+1**,** *Début*−1**)** if *Fin* < *Début*−1

Σ(k,k,4,1)

−5

Σ(k,k,4,1)+Σ(k,k,2,4)

4

## # (indirection)

**Catalogue > ∫Σ > Opérateurs mathématiques**

**#** *ChaîneNomVar*
Fait référence à la variable *ChaîneNomVar*. Permet d'utiliser des chaînes de caractères pour créer des noms de variables dans une fonction.

#("x"&"y"&"z")

xyz

Crée ou fait référence à la variable xyz.

10 → r

10

"r" → s1

"r"

#s1

10

Donne la valeur de la variable (r) dont le nom est stocké dans la variable s1.

## E (notation scientifique)

**Touche** EE

*mantisse*E*exposant*

Saisit un nombre en notation scientifique. Ce nombre est interprété sous la forme *mantisse* × $10^{exposant}$.

Conseil : pour entrer une puissance de 10 sans passer par un résultat de valeur décimale, utilisez 10^*entier*.

2.3 E 4

23000.

2.3 E 9+4.1 E 15

4.1 E 15

$3 \cdot 10^4$

30000

## G (grade)

**Catalogue > ∫Σ > Angle**

*Expression1*$^G$ ⇒ *expression*

*Liste1*$^G$ ⇒ *liste*

*Matrice1*$^G$ ⇒ *matrice*

Cette fonction permet d'utiliser un angle en grades en mode Angle en degrés ou en radians.

En mode Angle en radians, multiplie *Expression1* par π/200.

En mode Angle en degrés, multiplie *Expression1* par g/100.

En mode Angle en grades, donne *Expression1* inchangée.

En mode Angle en degrés, grades ou radians :

cos(50$^G$)

$\frac{\sqrt{2}}{2}$

cos({0,100$^G$,200$^G$})

{1,0.-1}

## r (radian)

**Catalogue > ∫Σ > Angle**

*Expression1*$^r$ ⇒ *expression*

*Liste1*$^r$ ⇒ *liste*

*Matrice1*$^r$ ⇒ *matrice*

En mode Angle en degrés, multiplie *Expression1* par 180/π.

En mode Angle en radians, donne *Expression1* inchangée.

En mode Angle en grades, multiplie *Expression1* par 200/π.

Cette fonction permet d'utiliser un angle en radians en mode Angle en degrés ou en grades.

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.

En mode Angle en degrés, grades ou radians :

cos((π/4)$^r$)

$\frac{\sqrt{2}}{2}$

cos({0$^r$,(π/12)$^r$, $^-$π$^r$})

$\{1\ \frac{(\sqrt{3}+1)\cdot\sqrt{2}}{4}\ -1\}$

## ° (degré)

**Catalogue > ∫Σ > Angle**

*Expression1*° ⇒ *expression*

*Liste1*° ⇒ *liste*

*Matrice1*° ⇒ *matrice*

En mode Angle en radians, multiplie *Expression1* par π/180.

En mode Angle en degrés, donne *Expression1* inchangée.

En mode Angle en grades, multiplie *Expression1* par 10/9.

Cette fonction permet d'utiliser un angle en degrés en mode Angle en grades ou en radians.

En mode Angle en degrés, grades ou radians :

cos(45°)

$\frac{\sqrt{2}}{2}$

cos({0,π/4,90°,30.12°}) ctrl ≈ enter

{1,.707...,0,.864...}

## °, ', '' (degré/minute/seconde)

**Catalogue > ∫Σ > opérateurs mathématiques**

*dd*°*mm*'*ss.ss*'' ⇒ *expression*

*dd* Nombre positif ou négatif

*mm* Nombre positif ou nul

*ss.ss* Nombre positif ou nul

Donne *dd*+(*mm*/60)+(*ss.ss*/3600).

Ce format d'entrée en base 60 permet : -

- d'entrer un angle en degrés/minutes/secondes quel que soit le mode angulaire utilisé.
- d'entrer un temps exprimé en heures/minutes/secondes.

**Remarque :** faites suivre ss.ss de deux apostrophes ('') et non de guillemets (").

En mode Angle en degrés :

25°13'17.5''

25.221...

25°30'

51/2

## ∠ (angle)

**Catalogue > ∫Σ > Angle**

[*Rayon*,∠θ_*Angle*] ⇒ *vecteur*
(entrée polaire)

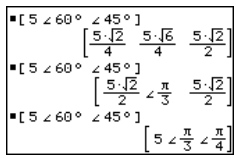
[*Rayon*,∠θ_*Angle*,*Valeur_Z*] ⇒ *vecteur*
(entrée cylindrique)

[*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.

[5 ∠60° ∠45°]

En mode Angle en radians et avec le Format vecteur réglé sur :




(*Grandeur* ∠ *Angle*) ⇒ *valeurComplexe*
(entrée polaire)

Saisit une valeur complexe en coordonnées polaires (r∠θ). L'*Angle* est interprété suivant le mode Angle sélectionné.

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

5+3i−(10∠π/4)

5−5•2+(3−5•2)•*i*

ctrl ≈ enter

⁻2.071...−4.071...•*i*

## ' (prime)

**Touche** /

*variable* '
*variable* ''

Saisit le symbole prime dans une équation différentielle. Ce symbole caractérise une équation différentielle du premier ordre ; deux symboles prime, une équation différentielle du deuxième ordre, etc.

deSolve(y''=y⁻¹/² and y(0)=0 and y'(0)=0,t,y)

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

## _ (soulignement)

**Touches** ctrl ␣

*Expression_Unité*

Indique l'unité d'une *Expression*. Tous les noms d'unités doivent commencer par un trait de soulignement.

Il est possible d'utiliser les unités prédéfinies ou de créer des unités personnalisées. Pour obtenir la liste des unités prédéfinies, reportez-vous au module consacré aux constantes et aux unités de mesure. Vous pouvez sélectionner les noms d'unités dans le Catalogue ou taper directement les noms des unités.

3_m▶_ft

9.842... • _ft

**Remarque :** vous pouvez trouver le symbole de conversion, ▶, dans le Catalogue. Cliquez sur ∫Σ, puis sur **Opérateurs mathématiques**.

## _ (soulignement)

**Touches** ctrl ␣

*Variable_*

Si *Variable* n'a pas de valeur, elle est considérée comme représentant un nombre complexe. Par défaut, sans _ , la variable est considérée comme réelle.

Si *Variable* a une valeur, _ est ignoré et *Variable* conserve son type de données initial.

**Remarque :** vous pouvez stocker un nombre complexe dans une variable sans utiliser _ . Toutefois, pour optimiser les résultats dans des calculs tels que **cSolve()** et **cZeros()**, l'utilisation de _ est recommandée.

En supposant que z est une variable non définie :

real(z)

z

real(z_)

real(z_)

imag(z)

0

imag(z_)

imag(z_)

## ▸ (conversion)

**Catalogue >** ∫Σ **> Opérateurs mathématiques**

*Expression_Unité1* ▸ *_Unité2* ⇒ *Expression_Unité2*

Convertit l'unité d'une expression. Les unités doivent être de la même catégorie.

Le trait de soulignement _ indique les unités. Pour obtenir la liste des unités prédéfinies, reportez-vous au module consacré aux constantes et aux unités de mesure. Vous pouvez sélectionner les noms d'unités dans un menu ou les taper directement.

Pour taper le trait de soulignement lors de la saisie directe des unités, appuyez sur ctrl ␣.

**Remarque :** pour convertir des unités de température, utilisez **tmpCnv()** et **ΔtmpCnv()**. L'opérateur de conversion ▸ ne gère pas les unités de température.

3_m▸_ft

9.842... • _ft

## 10^()

**Catalogue >** ∫Σ **> Opérateurs mathématiques**

**10^ (***Expression1***)** ⇒ *expression*

**10^ (***Liste1***)** ⇒ *liste*

Donne 10 élevé à la puissance de l'argument.

Dans le cas d'une liste, donne 10 élevé à la puissance des éléments de *Liste1*.

$10^{1.5}$

31.622...

$10^{\{0,\ -2,2,a\}}$

$\{1\ \frac{1}{100}\ 100\ 10^{a}\}$

**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.

$10^{\begin{bmatrix} 1 & 5 & 3 \\ 4 & 2 & 1 \\ 6 & -2 & 1 \end{bmatrix}}$

$\begin{bmatrix} 1.143...E7 & 8.171...E6 & 6.675...E6 \\ 9.956...E6 & 7.115...E6 & 5.813...E6 \\ 7.652...E6 & 5.469...E6 & 4.468...E6 \end{bmatrix}$

## $x^{-1}$ (inverse)

**Catalogue >** ∫Σ **> Opérateurs mathématiques**

*Expression1* $\mathbf{x}^{-1}$ ⇒ *expression*

*Liste1* $\mathbf{x}^{-1}$ ⇒ *liste*

Donne l'inverse de l'argument.

Dans le cas d'une liste, donne les inverses des éléments de *Liste1*.

$3.1^{-1}$

.322581

$\{a,4,\ -.1,x-2\}^{-1}$

$\{\frac{1}{a}\ \frac{1}{4}\ -10.\ \frac{1}{x-2}\}$

## $x^{-1}$ (inverse)

**Catalogue > ∫Σ > Opérateurs mathématiques**

*matriceCarrée1* $x^{-1} \Rightarrow$ *matriceCarrée*
Donne l'inverse de *matriceCarrée1*.
*matriceCarrée1* doit être une matrice carrée non singulière.

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

$$\begin{bmatrix} 1 & 2 \\ a & 4 \end{bmatrix}^{-1}$$

$$\blacksquare \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}^{-1} \qquad \begin{bmatrix} -2 & 1 \\ 3/2 & -1/2 \end{bmatrix}$$

$$\blacksquare \begin{bmatrix} 1 & 2 \\ a & 4 \end{bmatrix}^{-1} \qquad \begin{bmatrix} \frac{-2}{a-2} & \frac{1}{a-2} \\ \frac{a}{2\cdot(a-2)} & \frac{-1}{2\cdot(a-2)} \end{bmatrix}$$

## | (sachant que)

**Touche** ctrl sto

*Expression* **|** *Expression booléenne1* [**and** *Expression booléenne2*]...[**and** *Expression booléenneN*]
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'un « and » logique.
Cet opérateur fournit les trois types de fonctionnalités suivants : substitutions, contraintes d'intervalle et exclusions.

Les substitutions se présentent sous forme d'une égalité, telle que x=3 ou y=sin(x). Pour de meilleurs résultats, la partie gauche doit être une variable simple. *Expression* | *Variable* = *valeur* substituera *valeur* à chaque occurrence de *Variable* dans *Expression*.

Les contraintes d'intervalle se présentent sous la forme d'une ou plusieurs inéquations reliées par des opérateurs « and » logiques. Les contraintes d'intervalle permettent également la simplification qui autrement pourrait ne pas être valide ou calculable.

Les exclusions utilisent l'opérateur relationnel « différent de » (/= ou ≠) pour exclure une valeur spécifique du calcul. Elles servent principalement à exclure une solution exacte lors de l'utilisation de **cSolve()**, **cZeros()**, **fMax()**, **fMin()**, **solve()**, **zeros()**, etc.

x+1| x=3
4

x+y| x=sin(y)
sin(y) + y

x+y| sin(y)=x
x + y

$x^3-2x+7 \rightarrow f(x)$
Done

f(x)| $x=\sqrt{(3)}$
$\sqrt{3} + 7$

$(\sin(x))^2+2\sin(x)-6 \mid \sin(x)=d$
$d^2+2d-6$

solve($x^2-1=0$,x)|x>0 and x<2
x = 1

$\sqrt{(x)} \cdot \sqrt{(1/x)}|x>0$
1

$\sqrt{(x)} \cdot \sqrt{(1/x)}$
$\sqrt{\frac{1}{x}} \cdot \sqrt{x}$

solve($x^2-1=0$,x)| x≠1
x = ⁻1

## → (stocker)

**Touche** sto▸

*Expression* → *Var*
*List* → *Var*
*Matrice* → *Var*
*Expression* → *nomFonc*(*Param1*,...)
*List* → *nomFonc*(*Param1*,...)
*Matrice* → *nomFonc*(*Param1*,...)

Si la variable *Var* n'existe pas, celle-ci est créée par cette instruction et reçoit *Expression*, *Liste* ou *Matrice* comme valeur initiale.

Si *Var* existe déjà et n'est pas verrouillée ou protégée, son contenu est remplacé par *Expression*, *Liste* ou *Matrice*.

**Conseil :** si vous envisagez d'effectuer des calculs symboliques en utilisant des variables non définies, ne stockez aucune valeur dans les variables communément utilisées à une lettre, telles que a, b, c, x, y, z, etc.

π/4 → mavar

$\frac{\pi}{4}$

2cos(x) → Y1(x)

Done

{1,2,3,4} → Lst5

{1,2,3,4}

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

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

"Hello" → str1

"Hello"

## := (assigner)

**Touche** sto▸

*Var* := *Expression*
*Var* := *Liste*
*Var* := *Matrice*
*nomFonc*(*Param1*,...) := *Expression*
*nomFonc*(*Param1*,...) := *Liste*
*nomFonc*(*Param1*,...) := *Matrice*

Si la variable *Var* n'existe pas, celle-ci est créée par cette instruction et reçoit *Expression*, *Liste* ou *Matrice* comme valeur initiale.

Si *Var* existe déjà et n'est pas verrouillée ou protégée, son contenu est remplacé par *Expression*, *Liste* ou *Matrice*.

**Conseil :** si vous envisagez d'effectuer des calculs symboliques en utilisant des variables non définies, ne stockez aucune valeur dans les variables communément utilisées à une lettre, telles que a, b, c, x, y, z, etc.

π/4 → mavar

$\frac{\pi}{4}$

2cos(x) → Y1(x)

Done

{1,2,3,4} → Lst5

{1,2,3,4}

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

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

"Hello" → str1

"Hello"

## © (commentaire)

**Barre d'outils Calculs >** {□□} **> Structures de contrôle**

© [*texte*]

© considère *texte* comme une ligne de commentaire qui peut être utilisée pour annoter les instructions des fonctions.

© peut être utilisé au début ou à tout emplacement de la ligne. Tous les caractères situés à droite de ©, jusqu'à la fin de la ligne, sont considérés comme partie intégrante du commentaire.

Segment de la fonction :

```
  ⋮
:© Obtenir 10 points
:For i,1,10 © 10 fois cette boucle
  ⋮
```

**0b, 0h** — Touches 0 B, touches 0 H

**0b** *nombreBinaire*
**0h** *nombreHexadécimal*
Indique un nombre binaire ou hexadécimal, respectivement. Pour entrer un nombre binaire ou hexadécimal, vous devez utiliser respectivement le préfixe 0b ou 0h, quel que soit le mode Base utilisé. Un nombre sans préfixe est considéré comme décimal (base 10). Le résultat est affiché en fonction du mode Base utilisé.

En mode base Dec :

0b10+0hF+10

27

En mode base Bin :

0b10+0hF+10

0b11011

En mode base Hex :

0b10+0hF+10

0h1B

# Annexe : Informations générales

## *Informations sur les services et la garantie TI*

| | |
|---|---|
| **Informations sur les produits et les services TI** | Pour plus d'informations sur les produits et les services TI, contactez TI par e-mail ou consultez la pages du site Internet éducatif de TI.<br>adresse e-mail : ti-cares@ti.com<br>adresse internet : http://education.ti.com/france |
| **Informations sur les services et le contrat de garantie** | Pour plus d'informations sur la durée et les termes du contrat de garantie ou sur les services liés aux produits TI, consultez la garantie fournie avec ce produit ou contactez votre revendeur Texas Instruments habituel. |

## *Précautions à prendre lors de l'utilisation des piles*

Prenez ces précautions lorsque vous changez les piles.

- Ne laissez pas les piles à portée des enfants.
- Ne mélangez pas des piles neuves et usagées. Ne mélangez pas les marques ou divers types de piles d'une même marque.
- Ne mélangez pas des piles rechargeables et des piles qui ne le sont pas.
- Insérez les piles en respectant la polarité (+ et -).
- Ne mettez pas des piles non rechargeables dans un chargeur.
- Jetez correctement et immédiatement les piles usées.
- Ne brûlez ni ne démontez les piles.
- Consultez immédiatement un médecin en cas d'ingestion d'une batterie d'accumulateurs ou d'une pile.

### Précautions supplémentaires pour les piles rechargeables :

- N'utilisez que le chargeur recommandé pour ce type de pile ou celui fourni avec le matériel d'origine.
- Retirez les piles du chargeur ou l'adaptateur AC lorsque vous n'utilisez pas le produit ou lorsqu'il n'est pas en charge.
- L'utilisation des piles dans d'autres appareils peut causer des blessures corporelles, endommager le matériel ou d'autres appareils.

### Pour un bon recyclage des piles usées :

Ne dégradez pas, ne percez pas et ne jetez pas les piles dans un feu. Les piles pourraient éclater ou exploser et émettre des produits chimiques dangereux. Jetez les piles usées conformément aux réglementations locales.

# Index


## A

Accessoires
  présentation *11*
Acquisition de données
  présentation *9*
activité
  ajout dans une page *41*
  ajout de page *42*

## B

barre d'outils
  application *27*
barre d'outils d'application *27*
barre d'outils État du système (System Status) *27*
bouton de déplacement *16*

## C

Calculs
  présentation *7*
Catalogue (Catalog)
  accès *47*
Chargeur de station TI-Nspire™
  description *11*
classeur
  affichage des pages *45*
  création *39*
  ouverture, classeur existant *31*
clavier
  bouton de déplacement *16*
  description *15*
  fonctions des touches *15*
  raccourcis *21*
  touches modificatrices *20*
conventions
  dénomination des fichiers *34*
conventions de dénomination des fichiers *34*
copyright statement *ii*
curseur
  déplacement *17*

## D

déplacement du curseur *17*
dossier
  création *47*
  développement *32*
  fermeture *33*

## E

écran de l'unité nomade
  déplacement *17*
Éditeur mathématique
  présentation *8*

## F

fichier
  conventions de dénomination *34*

## G

Graphiques & géométrie
  présentation *6*

## L

logiciel
  présentation *5*
Logiciel d'interface avec l'ordinateur pour TI-Nspire
  présentation *9*

## M

modèle de classeur *3*

## N

navigateur de classeurs *30*
  accès *28*
  gestion des fichiers et des dossiers *28*
  options des menus *28*

## P

page
  affichage d'un classeur *45*

ajout à une activité *42*
réorganisation dans un classeur *46*

## R

raccourcis
clavier *21*
réglages du classeur
changement *47*
réglages du système
accès *34*
modification *34*
restauration des réglages par défaut *34*
réorganisation des pages dans un classeur *46*

## S

sondes USB à connexion directe
compatibilité *11*
Station de connexion TI-Nspire™
description *11*

## T

Tablette de rétroprojection ViewScreen pour TI-Nspire™
description *11*
Tableur & listes
présentation *6*
technologie
composants *4*
technologie, prÈsentation *3*
touches modificatrices *20*
trieuse de pages
accès *42*
options de la barre d'outils *43*

## U

unité nomade *4*
barre d'outils *24*
barre d'outils d'application *27*
barre d'outils État du système (System Status) *27*
écran *22*
éléments de l'écran *22*