Chapitre 15

Bibliothèques de programmes

TI-Nspire travaille sur des *Classeurs* indépendants. Les variables, fonctions ou programmes sont définis dans chaque *Activité* contenue dans ces *Classeurs*, et n'existent qu'à l'intérieur de l'*Activité* où ils ont été créés. Il est naturellement nécessaire de disposer d'un mécanisme permettant de mettre en commun des fonctions ou des programmes susceptibles d'être utilisés dans plusieurs classeurs différents. Ce chapitre va nous permettre de le découvrir en détail.

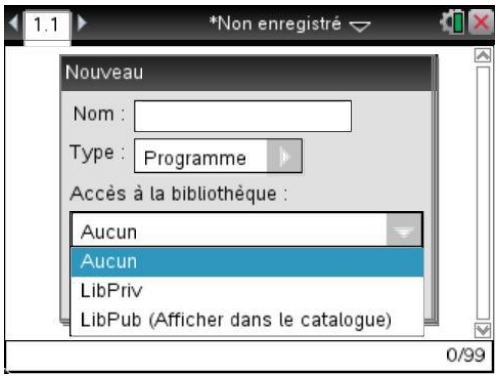
Sommaire

Philippe Fortin (Lycée Louis Barthou – Pau) / Roland Pomès (Lycée René Cassin – Bayonne)

1. Montrer un objet dans le catalogue

1.1 *Création de la fonction*

Lorsque l'on défini un programme ou une fonction, on obtient une boîte de dialogue permettant de choisir différents « *Accès à la bibliothèque* ».



Définissons par exemple une fonction calculant les valeurs du pgcd de deux entiers, et affichant les étapes permettant de l'obtenir. Cela nous permettra d'illustrer ces différents modes.

Nous allons utiliser l'algorithme d'Euclide.

Pour trouver le pgcd de *a* et *b* :

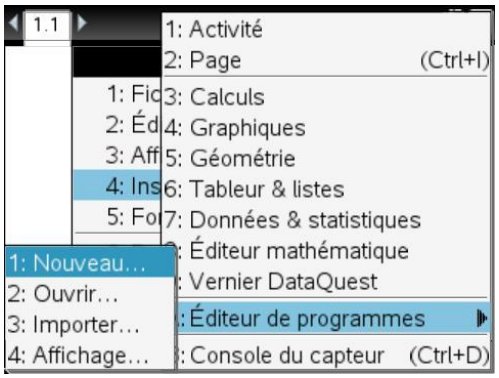
- Si $b=0$, c'est terminé : $\text{pgcd}(a,b)=a$
- Sinon, on calcule le reste *r* de la division de *a* par *b* et on cherche le pgcd de *b* et *r*.

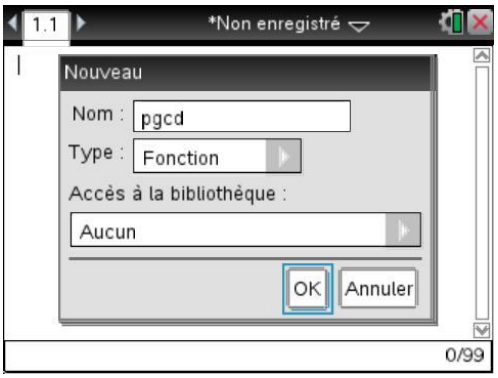
On peut traduire directement cet algorithme en utilisant un appel récursif :

```
Define pgcd(a,b)= Func
Local r
Disp "Calcul du pgcd de ",a," et ",b
if b=0 then
  Disp "Le pgcd est ",a
  a
Else
  r:=mod(a,b)
  Disp "Reste :",r
  pgcd(b,r)
Endif
EndFunc
```

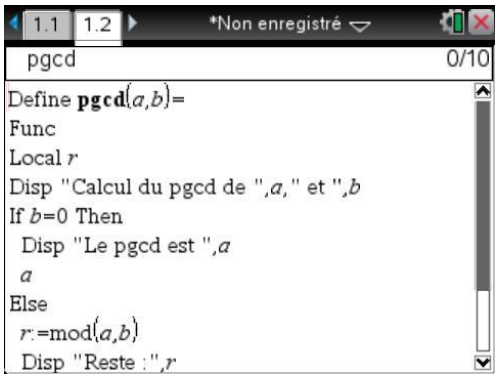
Les trois lignes avec des **Disp** ne sont pas indispensables au bon fonctionnement de l'algorithme, mais elles permettent d'en suivre le déroulement.

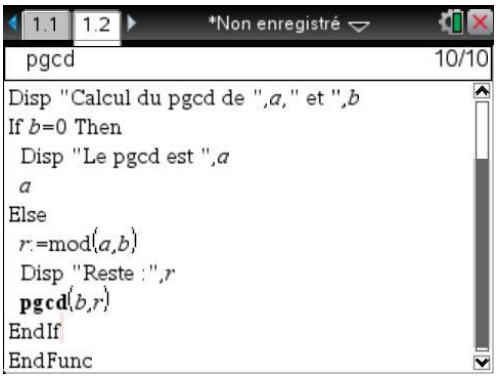
Ouvrir un nouveau classeur, avec une page Calculs. Ouvrir l'éditeur de programme en sélectionnant **Insertion, Éditeur de programmes, Nouveau** accessible par la combinaison de touches doc▾ 4 B 1. ctrl 6 pour un affichage dans une nouvelle page. Entrer ensuite le nom de la fonction, et sélectionner le type correspondant. Pour l'instant, laisser le dernier champ à la valeur Aucun.

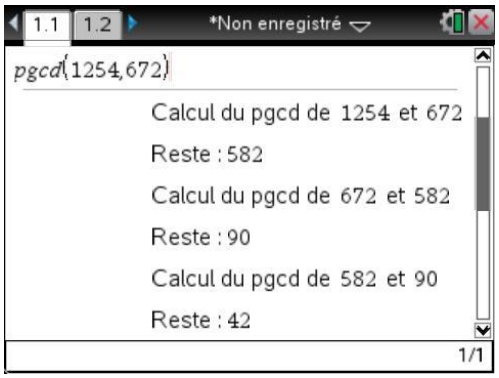


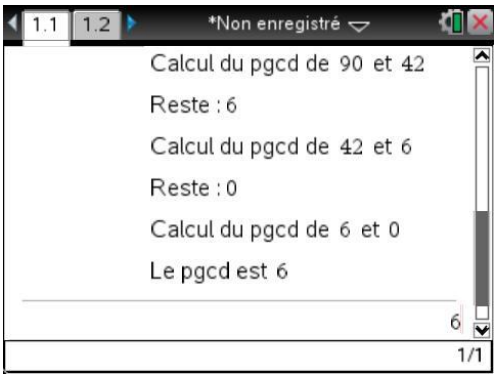


On peut ensuite entrer le texte de la fonction dans l'éditeur de programme, sauver le contenu en utilisant ctrl B, puis faire un premier essai d'utilisation.









Notre fonction pgcd fonctionne bien… De plus c'est une *fonction*, et non un *programme* (revoir la page 16 du **chapitre 14** si la distinction n'est pas claire).

Nous pouvons donc utiliser le résultat qu'elle fourni pour faire un autre calcul.

Par exemple, on pourrait définir une autre fonction, calculant le plus petit multiple commun de deux nombres, en utilisant l'égalité $\text{ppcm}(a,b)=\dfrac{a\times b}{\text{pgcd}(a,b)}$.

La définition de cette fonction **ppcm** peut se faire en une seule ligne dans l'application Calculs :

Define ppcm(a,b)=a·b/pgcd(a,b) enter

Son utilisation provoquera celle de la fonction **pgcd**, avec affichage des étapes, puis le résultat obtenu sera utilisé pour le calcul du quotient.

Pour l'instant, ces deux fonctions n'existent que dans l'activité où elles ont été créées. Elles ne sont pas utilisables dans une autre activité du même classeur, ni bien sûr dans un classeur différent.

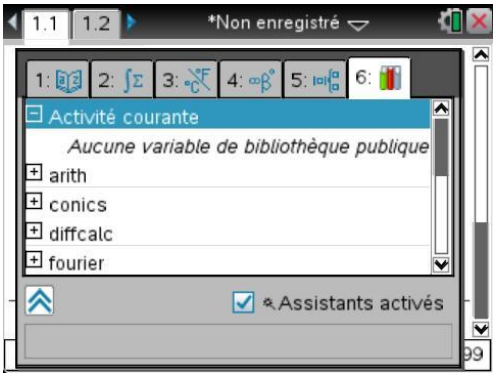
On ne peut pas non plus y accéder depuis le catalogue, ce qui permettrait d'en faciliter l'utilisation. Nous allons commencer par régler ce problème.

1.2 *Choix de l'attribut LibPub*

Le catalogue comporte plusieurs onglets : classement alphabétique, classement thématique, unités et constantes, caractères spéciaux, modèles mathématiques…

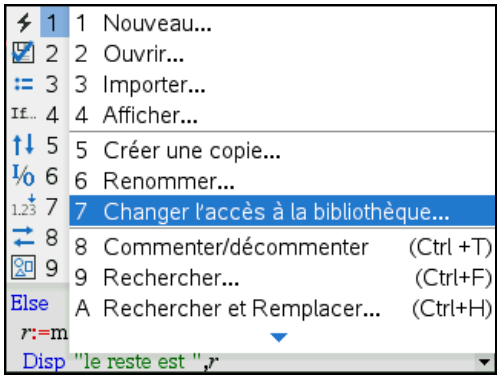
Le dernier onglet permet d'accéder aux fonctions ou programmes définis par l'utilisateur, sous réserve que l'affichage de ces derniers ait été demandé.

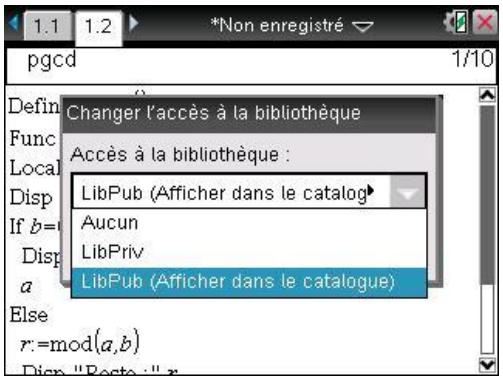
Pour l'instant, ce n'est pas encore le cas, et on peut lire « *Aucune variable de bibliothèque publique* » dans l'entrée correspondant à l'activité courante (nous verrons plus loin à quoi correspondent les autres lignes visibles dans cette copie d'écran).



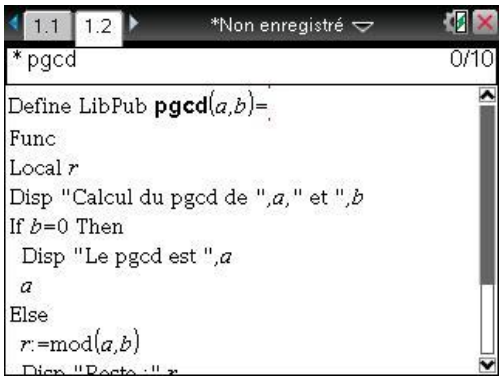
Pour que la fonction **pgcd** devienne visible ici, nous devons la déclarer avec le type **LibPub**. Pour cela, on doit revenir sur la page avec l'éditeur de programme où se trouve le texte de cette fonction.

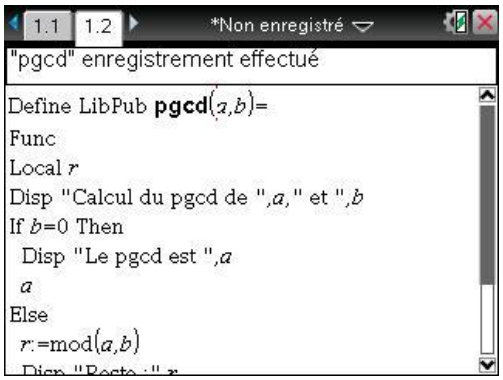
Appuyer sur la touche [menu] et sélectionner **Action** puis **Changer l'accès à la bibliothèque**. Choisir ensuite l'option **LibPub**.



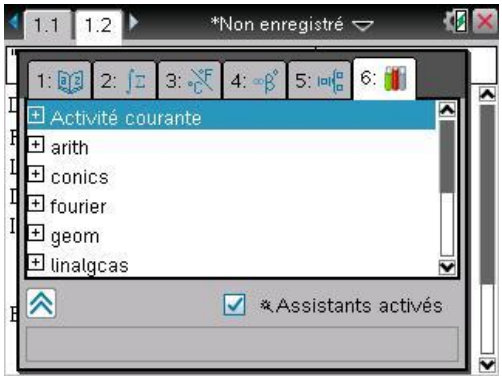


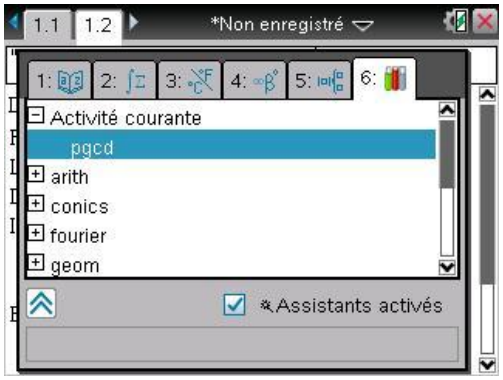
À ce stade, rien n'est encore fait ! Le symbole * précédant le nom de la fonction (écran de gauche) indique que la modification n'a pas été sauvée. Appuyer sur [ctrl] [B] pour enregistrer la nouvelle situation. Un message confirme la sauvegarde de la nouvelle version (écran de droite).





Vérifions l'effet de cette modification dans le catalogue (pour développer la branche Activité courante, et obtenir l'écran de droite, il suffit d'appuyer sur ▸) :





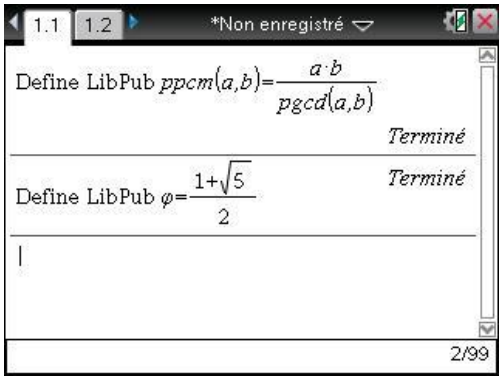
Il est également possible de faire en sorte de définir un objet directement dans l'application Calculs, sans passer par l'éditeur de programmes, en le rendant visible dans le catalogue.

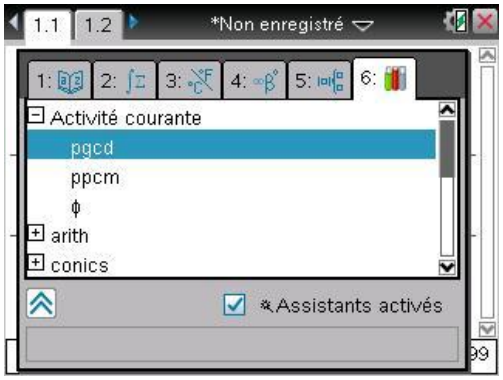
Par exemple, il est possible de saisir la définition la fonction **ppcm** en entrant :

Define LibPub ppcm(a,b)=a·b/pgcd(a,b) enter

Pour entrer cette commande sur l'unité nomade, appuyer sur menu puis sélectionner **Actions**, **Bibliothèque**, **Définir l'accès LibPub**.

Il est également possible de définir des constantes en utilisant cette méthode :





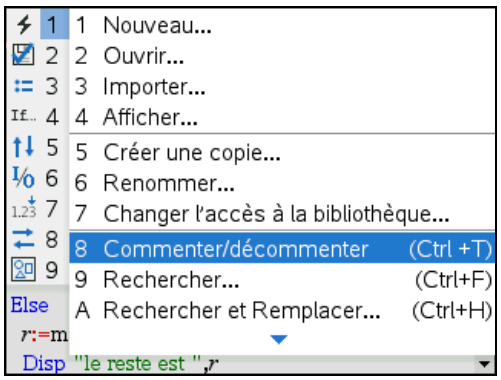
1.3 *Ajout d'un texte d'aide dans le catalogue*

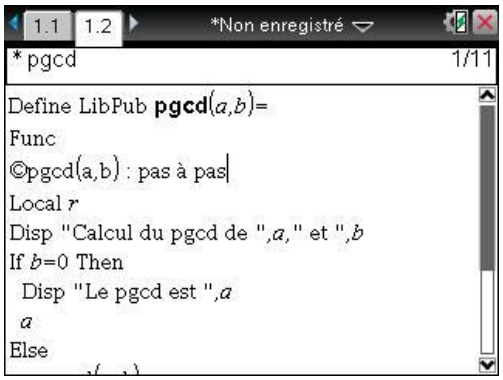
Les fonctions intégrées à TI-Nspire CAS listées dans le catalogue sont accompagnées d'un petit message d'aide en bas de l'écran.

Il est possible de faire de même pour les fonctions ou les programmes créés par l'utilisateur.

Pour cela, il suffit d'insérer une ligne de commentaire, au tout début du texte du programme ou de la fonction (sur la ligne située juste après **Prgm** ou **Func**).

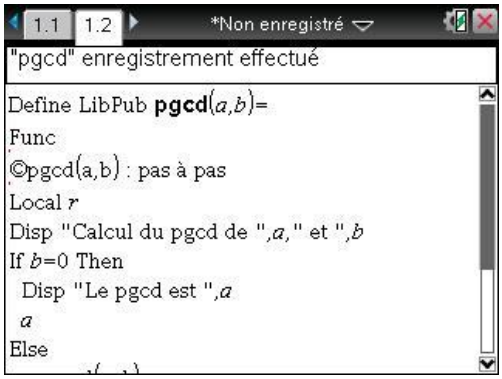
On peut utiliser le menu **Actions**, 8 (ou ctrl T), ce qui insère le caractère © au début de la ligne. Écrire ensuite le texte souhaité.

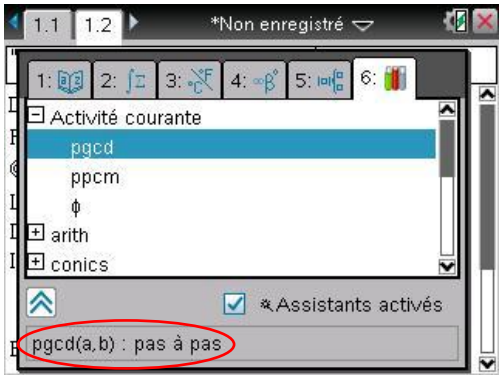




Penser à sauvegarder cette modification du programme par ctrl B.

On peut ensuite vérifier le nouveau contenu du catalogue. L'aide est bien visible en bas de l'écran.





2. Bibliothèques de programmes

Dans la section précédente, nous avons vu comment faire pour que des objets apparaissent dans le catalogue, dans la partie associée à l'*Activité courante*.

Comme son nom l'indique, le contenu de cette partie du catalogue dépend de l'activité actuellement en cours d'utilisation. Si on passe à une nouvelle activité dans le même classeur, ou si on change de classeur, le contenu de cette rubrique change. En particulier, il n'est pas possible d'accéder pour l'instant à la fonction **pgcd** lorsque l'on ouvre un nouveau classeur.

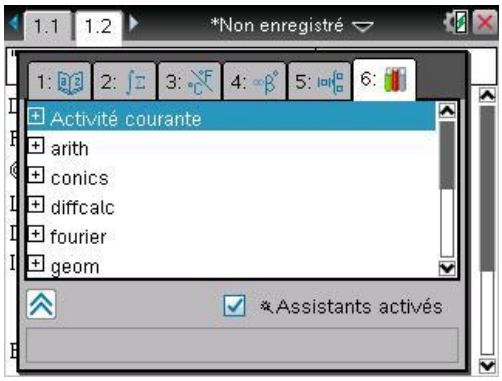
2.1 *Créer une nouvelle bibliothèque*

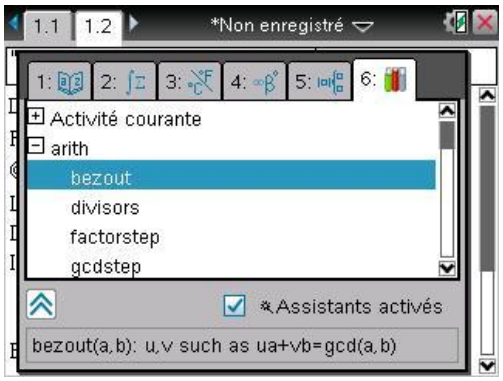
Pour remédier à cette situation, il faut

1. Placer les objets que l'on souhaite partager dans la première activité d'un classeur.
2. S'assurer que ces objets ont bien tous été créés avec l'attribut **LibPub** et que **leur nom est composé de 1 à 15 caractères sans espace, point ou accent et ne débute pas par un tiret**.
3. Enregistrer le classeur dans un dossier spécifique, destiné à accueillir les bibliothèques de programmes. Le nom de ce classeur – **1 à 16 caractères, respectant les règles ci-dessus** – sera visible dans le dernier onglet du catalogue.

- Sur l'unité nomade, le dossier à utiliser est nommé MyLib
- Sur l'ordinateur, par défaut, ce dossier est Mes Documents\TI-Nspire\MyLib

Par exemple la copie d'écran ci-dessous a été obtenue alors que les fichiers arith.tns, conics.tns, diffcalc.tns, fourier.tns… étaient présents dans le dossier MyLib. On peut voir dans l'écran de droite que le fichier arith.tns contient plusieurs fonctions ou programmes enregistrés avec l'attribut **LibPub** : **bezout**, **divisors**, **factorstep**, **gcdstep**…





2.2 *Objets privés*

Lorsque les fonctions ou programmes que l'on souhaite partager (définis avec l'attribut **LibPub**) utilisent d'autres fonctions ou programmes de la bibliothèque, que l'on ne souhaite pas rendre visibles dans le catalogue, ces derniers doivent être définis avec l'attribut **LibPriv** (objets privés de la bibliothèque).

Voici un exemple où cette situation peut se produire. La fonction **pgcd** créée au début de ce chapitre est a priori prévue pour travailler sur des entiers positifs.

Nous avons vu dans le **chapitre 14** une fonction permettant de tester qu'un nombre est bien un entier positif. Le texte de cette fonction est le suivant :

```
Define is_posint (n)= Func
if getType(n)≠"NUM" then
  Return false
Else
  Return (n≥0 and int(n)=n)
Endif
EndFunc
```

Nous pouvons placer cette fonction dans notre classeur, et l'utiliser au début de la fonction **pgcd** qui devient :

```
Define LibPub pgcd(a,b)= Func
Local r
If not (is_posint(a) and is_posint(b)) then
  Return "Erreur, les arguments ne sont pas des entiers positifs"
endif
Disp "Calcul du pgcd de ",a," et ",b
if b=0 then
 Disp "Le pgcd est ",a
 a
Else
 r:=mod(a,b)
 Disp "Reste :",r
 pgcd(b,r)
endif
EndFunc
```

Pour que cela fonctionne correctement, il est indispensable que la fonction **is_posint** soit elle aussi un objet partagé. Si nous ne souhaitons pas qu'elle apparaisse dans le catalogue, nous devons la déclarer avec l'attribut **LibPriv**. Sa définition devient alors :

```
Define LibPriv is_posint (n)= Func
if getType(n)≠"NUM" then
 Return false
Else
 Return (n≥0 and int(n)=n)
Endif
EndFunc
```

Si nous oublions de choisir l'attribut **LibPriv** (ou l'attribut **LibPub**) une erreur se produira lorsque l'on utilisera la fonction **pgcd** depuis un autre classeur. En effet l'exécution de cette fonction provoquera un appel à une fonction **is_posint** locale à ce classeur.
Cette fonction n'existera pas, et on obtiendra un message d'erreur.

☞ *Si on utilise* **LibPub** *au lieu de* **LibPriv**, *tout fonctionnera correctement, mais la fonction* **is_posint** *sera également listée dans le catalogue, ce qui pourra perturber l'utilisateur de cette bibliothèque de programmes.*

2.3 *Documentation, exemples…*

Il est parfaitement possible de sauver un classeur comportant des pages écrites avec l'Éditeur mathématique, ou comportant des exemples exécutés dans l'application Calculs en plus des définitions de programmes ou de fonctions. Cela ne perturbera en rien le bon fonctionnement de la bibliothèque, et permettra de donner des informations sur l'utilisation des objets de cette bibliothèque.

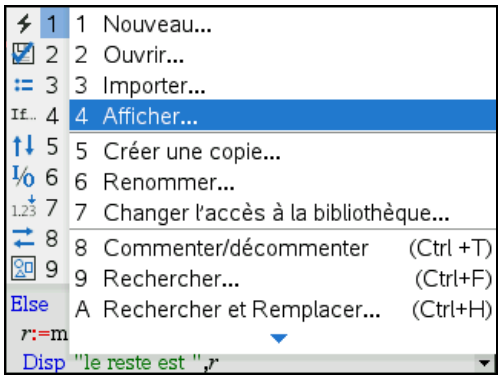
Si des variables sont créées à l'occasion de ces exemples, cela n'aura pas non plus d'incidence, puisque les objets qui n'ont pas été explicitement définis avec l'attribut **LibPub** ou **LibPriv** ne sont pas partagés. Ils sont simplement locaux au classeur où ils ont été définis, et cela n'aura aucune conséquence lors de l'utilisation des fonctions de la bibliothèque depuis un autre classeur.

À l'inverse, on peut parfaitement créer un classeur comportant de nombreuses pages où seront définis les différents objets à partager (avec un attribut **LibPriv** ou **LibPub**), puis supprimer toutes ces pages. Cela n'a aucune conséquence sur l'existence de ces objets dans le classeur.

Un classeur utilisé pour définir une bibliothèque comportant plusieurs dizaines d'objets peut donc se résumer, du moins en apparence, à une activité comportant une seule page avec quelques lignes de texte décrivant rapidement la nature de cette bibliothèque...

Pour en savoir plus sur le contenu réel d'un classeur de ce type, on peut ouvrir l'éditeur de programmes avec l'option **Affichage** pour visualiser les différents objets un à un[1].
Cela permet de visualiser les objets du classeur actuellement ouvert, mais aussi les objets partagés présents dans les différentes bibliothèques de programmes.

Appuyer sur [menu], sélectionner **Actions**, puis **Afficher**…



On a choisi ici de visualiser le contenu d'un objet qui ne fait pas partie du classeur en cours. Dans le cas contraire, la boite de dialogue dans laquelle on peut visualiser le texte du programme ou de la fonction comporte un bouton supplémentaire qui permet de lancer l'éditeur de programmes afin d'apporter des modifications.

3. Utilisation d'une bibliothèque de programmes

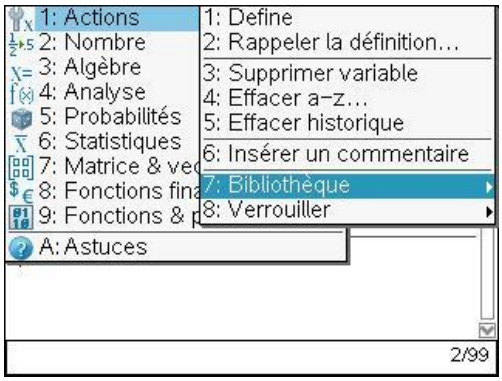
3.1 *Actualisation du contenu des bibliothèques de programmes*

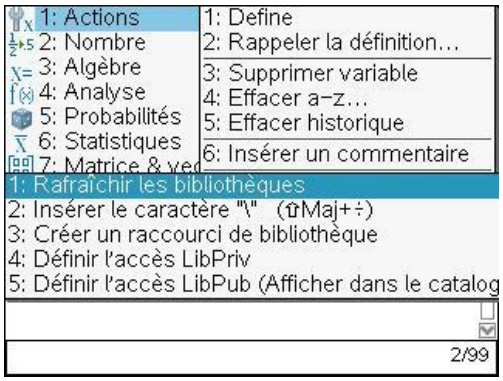
Avant de pouvoir utiliser une bibliothèque de programmes deux opérations sont indispensables

[1] Sur la version 3.2, il n'existe pas de solution permettant de visualiser directement le contenu de l'ensemble de tous les objets créés dans une bibliothèque (ou dans tout autre classeur). On a seulement un accès à la liste de tous les objets créés (par l'intermédiaire de la touche [var] ou du raccourci [ctrl] [L], également utilisable sur la version logicielle), et il est possible d'en visualiser le contenu par l'une des deux méthodes précédentes.

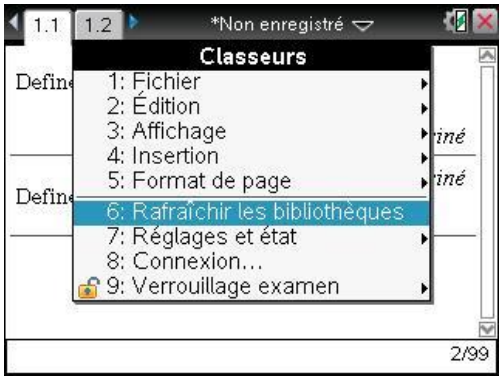
1. Enregistrer (ou transférer) le classeur dans le dossier des bibliothèques de programmes. Sur l'unité nomade, il s'agit du dossier MyLib.

2. Une fois que c'est fait, une autre étape ne doit pas être oubliée !
   Vous devez demander à « Rafraîchir les bibliothèques ».
   Une façon de procéder consiste à passer par le menu général en appuyant sur la touche menu, puis à sélectionner **Actions**, **Bibliothèque**, **Rafraîchir les bibliothèques**.





   Cette opération est suffisamment importante pour qu'un accès plus direct ait été mis en place. Appuyer sur doc, puis sélectionner **Rafraichir les bibliothèques**.



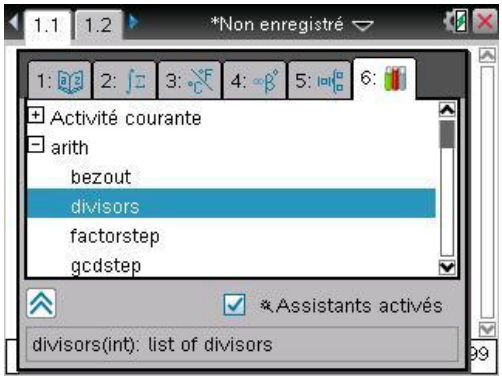
   De la même manière, dans la version logicielle de TI-Nspire CAS, on a un accès direct à **Rafraichir les bibliothèques** dans le menu **Outils** dans la barre des menus.

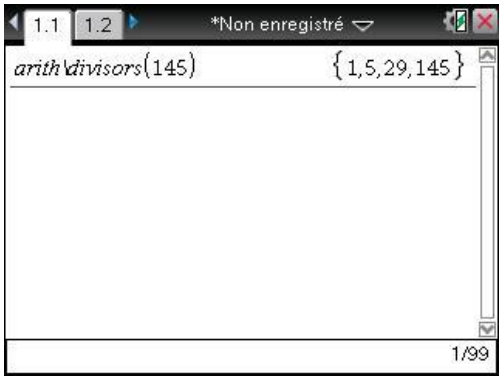
Attention, ce n'est qu'après avoir accompli ces deux étapes (sauvegarde du fichier dans le dossier des bibliothèques, et « rafraichissement ») que les objets de cette nouvelle bibliothèque seront accessibles.

3.2 *Sélection des objets publics dans le catalogue*

Une manière très simple d'utiliser une fonction ou un programme d'une bibliothèque est d'utiliser le dernier onglet du catalogue, et d'y sélectionner l'objet souhaité, comme on le fait pour tout autre objet prédéfini dans TI-Nspire.

Vous observerez que le nom de cet objet est copié avec un préfixe correspondant au nom de la bibliothèque. Il suffit ensuite de compléter la ligne de commande en ajoutant les arguments nécessaires.
La ligne d'aide située en bas du catalogue devrait vous donner les informations nécessaires pour utiliser les différents programmes ou fonctions.





3.3 *Saisie directe du nom d'un objet partagé (public ou privé)*

Il est également possible de saisir directement le nom d'un objet présent dans une bibliothèque. Il suffit de saisir son nom complet, incluant le nom de la bibliothèque, comme par exemple **arith\divisors**.

Le symbole \ s'obtient sur l'unité nomade par le raccourci ⇧shift ÷
(l'utilisation de ÷ seule provoquerait l'insertion du symbole /)

On peut ainsi accéder à tous les objets d'une bibliothèque, y compris à ses objets privés, qui ne sont pas visibles dans le catalogue.

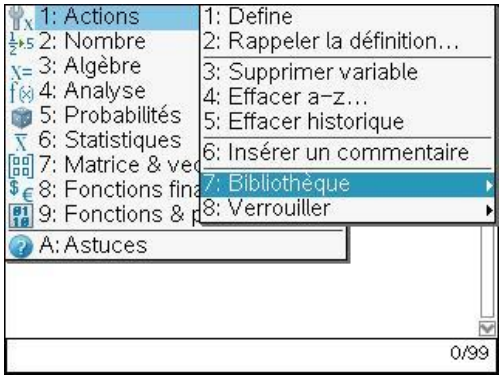
3.4 *Mise en place et utilisation d'un raccourci*

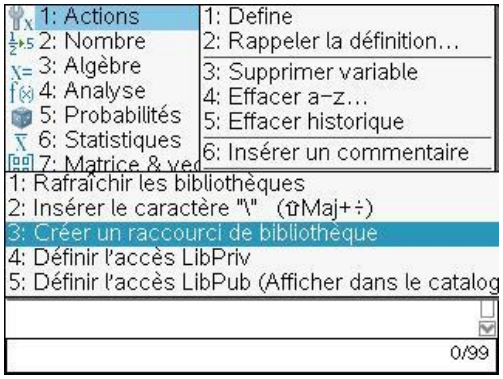
Si on souhaite utiliser plusieurs fois les objets d'une bibliothèque au cours d'une même activité (par exemple lors de la résolution d'un problème faisant appel aux fonctionnalités de la bibliothèque de programmes **linalgcas** téléchargeable avec ce chapitre, on peut créer un raccourci.

On peut par exemple décider d'associer la lettre m à la bibliothèque de calcul matriciel.

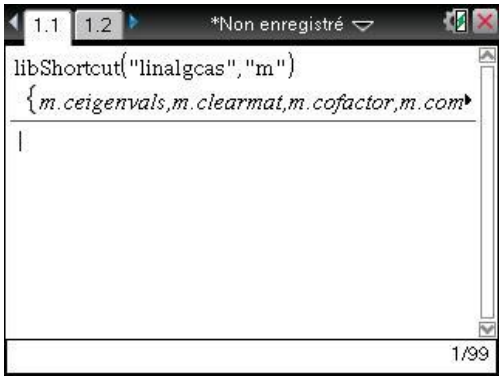
Il suffit pour cela d'utiliser la commande **LibShortcut**(*nom de bibliothèque*, *nom de raccourci*)

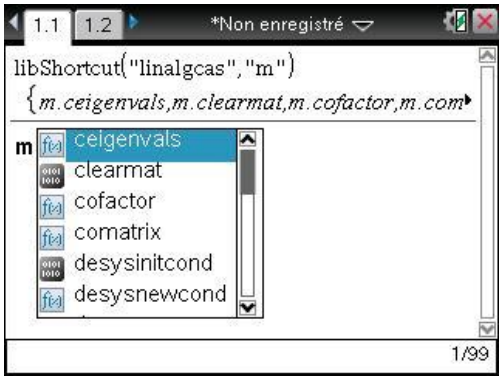
Pour entrer cette commande sur l'unité nomade, appuyez sur menu puis sélectionnez **Actions**, **Bibliothèque**, **Créer un raccourci de bibliothèque**.





Compléter ensuite l'instruction en indiquant les noms de la bibliothèque et du raccourci souhaité. Une fois que c'est fait, il suffit de taper le nom de ce raccourci, suivi d'un point, pour obtenir l'affichage d'une liste déroulante contenant le nom des objets disponibles.





La définition d'un raccourci est locale à une activité. Ce raccourci n'est plus valable dans une autre activité. Il reste par contre actif si on sauve et ré-ouvre un classeur.

Vous trouverez dans les pages suivantes un descriptif de plusieurs bibliothèques de programmes librement téléchargeables avec ce chapitre.

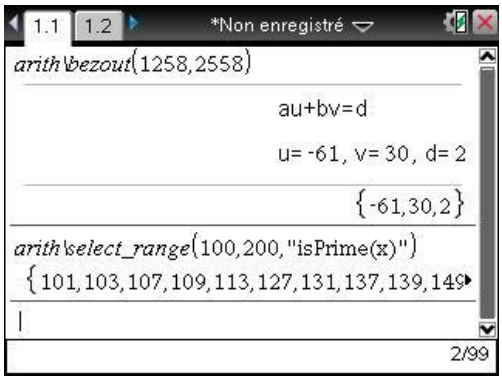
4. Quelques bibliothèques disponibles

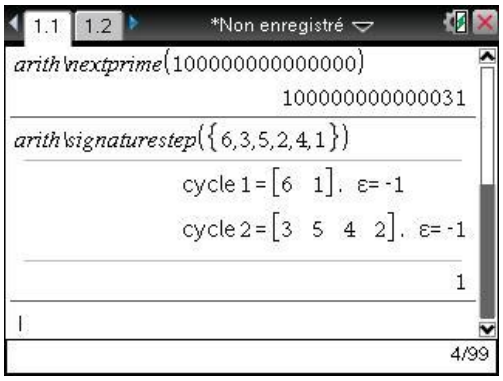
Vous trouverez plusieurs bibliothèques de programmes avec ce chapitre.

Pour vous aider, certaines disposent d'un fichier « démo » ou d'un fichier « help » donnant le contenu et des explications ou des exemples d'utilisation.

4.1 *La bibliothèque* arith

Cette bibliothèque comporte des fonctions destinées à l'étude de problèmes d'arithmétique, ainsi que des outils facilitant les opérations de sélection dans une liste, ou encore l'étude des permutations.





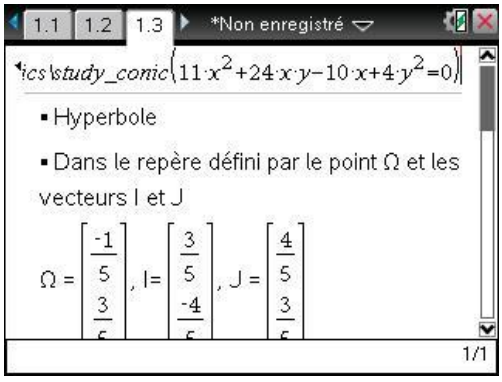
Conseils d'utilisation

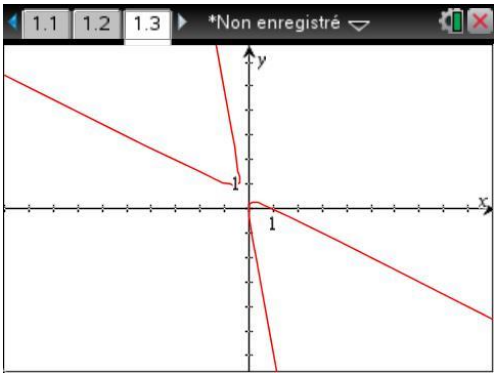
- Utiliser le programme **arith\help** pour obtenir la liste des fonctions et leur syntaxe d'utilisation.
- Utiliser le programme **arith\select_help** pour une information plus détaillée concernant la fonction de sélection dans une liste.

4.2 *La bibliothèque* conics

Cette bibliothèque permet de représenter graphiquement des coniques définies par leur équation cartésienne et d'en déterminer les éléments géométriques.

Inversement, il est possible de demander la recherche de l'équation d'une conique définie par foyer, directrice ou excentricité, ou encore d'une ellipse *E* définie par ses deux foyers et la distance *d* telle que $\forall M \in E, F_1M + F_2M = d$.





Conseils d'utilisation

- La façon la plus simple d'utiliser cette bibliothèque est de le faire depuis le fichier Graph Coniques CAS 2_0.tns que vous trouverez également en téléchargement avec le **chapitre 6**.

- Utiliser le programme **conics\help** pour obtenir la liste des fonctions et leur syntaxe d'utilisation.
- Utiliser le programme **conics\demo_plot** et **conics\demo_study** pour une série d'exemples d'utilisation.

Informations complémentaires

Les programmes de la bibliothèque **conics** agissent sur les définitions des fonctions **x1**, **x2**, **y1**, **y2** utilisées pour représenter des courbes définies par une équation paramétrique. Pour obtenir la construction de la conique il reste simplement à demander la représentation de ces deux courbes.

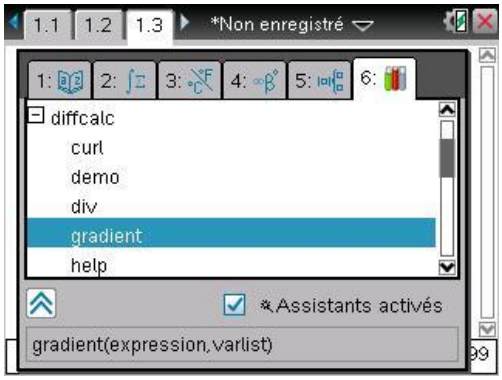
Pour cela, se placer dans l'application Graphiques, appuyer sur la touche menu, choisir **Type de graphique**, **Paramétrique** et valider simplement les lignes contenant les définitions du couple (**x1**,**y1**) et du couple (**x2**,**y2**).

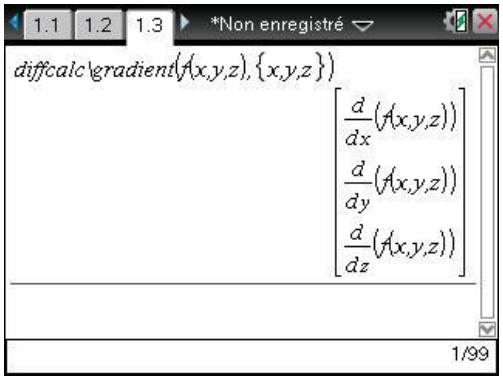
Deux courbes sont utilisées pour gérer le cas où la courbe représentant la conique est en deux parties : couples de droites ou hyperboles. Lorsque la conique est une parabole, un cercle ou une ellipse, on utilise seulement le couple (**x1**,**y1**) pour la représenter, et les fonctions **x2** et **y2** on alors la valeur undef.

Si vous utilisez le classeur Graph Coniques CAS 2_0.tns, aucune manipulation n'est nécessaire pour obtenir la représentation graphique. Cela a déjà été fait.

4.3 *La bibliothèque* diffcalc

Cette bibliothèque permet d'utiliser les fonctions usuelles : divergence, gradient, hessien, jacobien, laplacien et rotationnel.



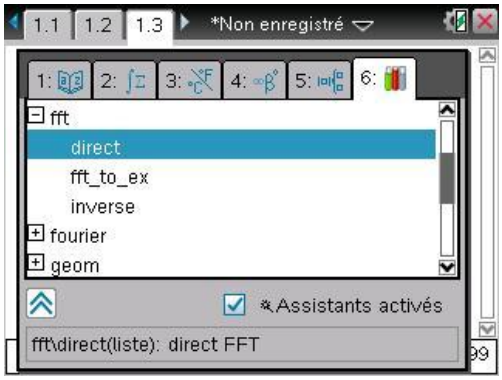


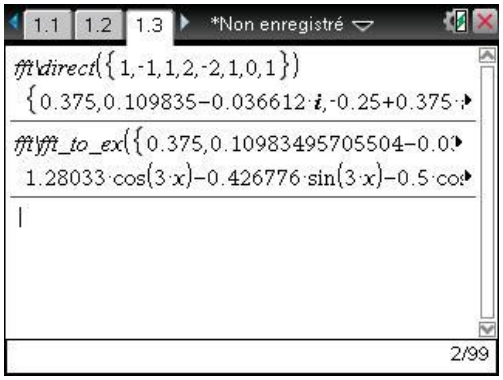
Conseils d'utilisation

- Utiliser le programme **diffcalc\help** pour obtenir la liste des fonctions et leur syntaxe d'utilisation.
- Utiliser le programme **diffcalc\demo** pour une série d'exemples d'utilisation.

4.4 *La bibliothèque* FFT

Cette bibliothèque offre la possibilité d'effectuer une FFT directe ou inverse.



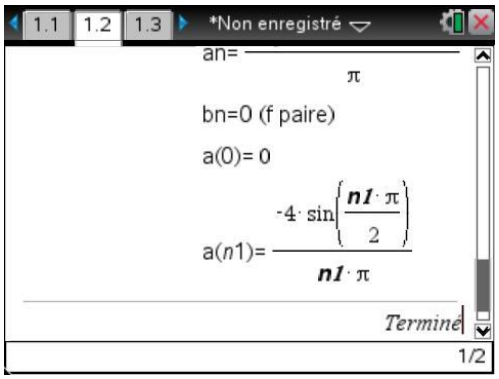


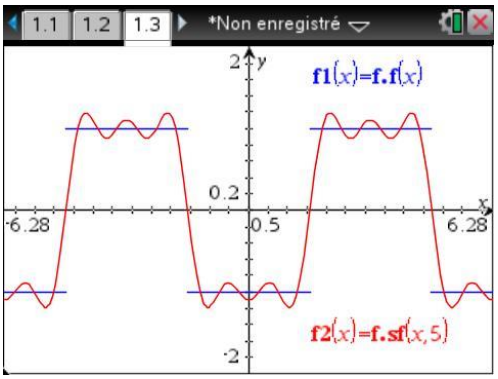
Conseils d'utilisation

- Utiliser le programme **diffcalc\demo** pour voir un exemple d'utilisation.

4.5 *La bibliothèque* fourier

Cette bibliothèque offre la possibilité de déterminer l'expression formelle des coefficients de Fourier et d'obtenir les représentations graphiques des sommes partielles $S_n(f)$ pour $n \in 1,10$.





Conseils d'utilisation

- Utiliser le programme **fourier\help** pour obtenir la liste des fonctions et leur syntaxe d'utilisation.
- Utiliser le programme **fourier\demo** pour une série d'exemples d'utilisation.

Informations complémentaires

Les programmes de la bibliothèque **fourier** agissent sur différentes variables globales, qui sont regroupées dans le conteneur **f.**

f.a Fonction permettant de calculer un coefficient a_n

f.b Fonction permettant de calculer un coefficient b_n

f.f Fonction prolongeant la fonction *f* à partir de sa définition sur un intervalle particulier en utilisant les propriétés de périodicité et de parité éventuelle. Il est ainsi possible de représenter la fonction sur plusieurs périodes, comme cela est fait dans le classeur demo_fourier.tns.

f.la Liste à 11 éléments, formée par $\{a_0, a_1, a_2, \ldots, a_{10}\}$

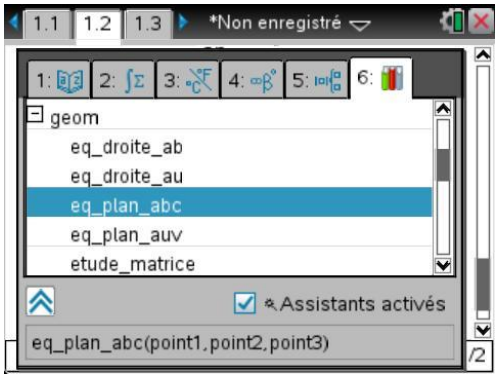
f.lb Liste à 11 éléments, formée par $\{0, b_1, b_2, \ldots, b_{10}\}$

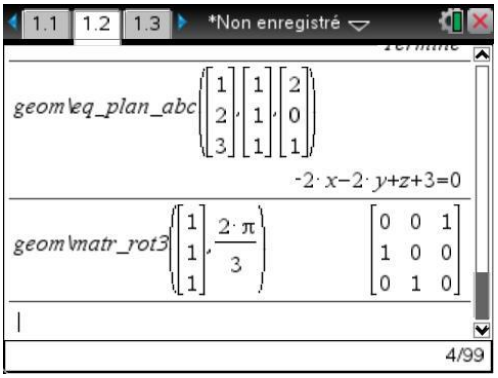
f.sf Fonction de deux variables **f.sf**(*x*, *n*). Permet de construire la somme partielle d'ordre *n*, pour *n* compris entre 1 et 10.

Vous trouverez plus d'information au sujet de cette bibliothèque dans le **chapitre 11**, Séries de Fourier.

4.6 *La bibliothèque* geom

Cette bibliothèque regroupe différentes fonctions utiles en sup ou en spé : étude d'une matrice 3×3, recherche de la matrice d'une rotation, d'une projection…



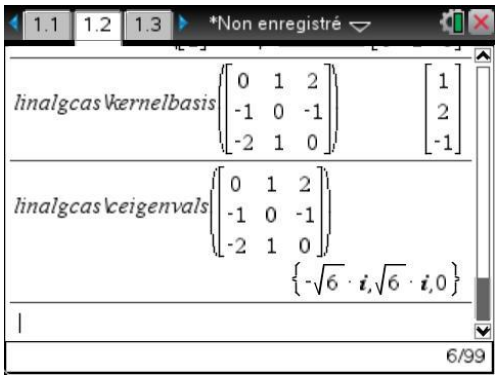


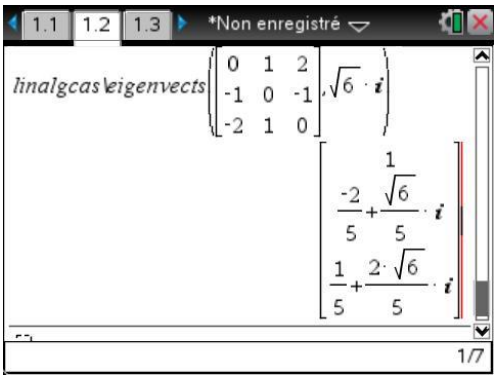
4.7 *La bibliothèque linalgcas*

Cette bibliothèque permet d'étendre les possibilités de TI-Nspire CAS dans le domaine de l'algèbre linéaire, en offrant par exemple des fonctions permettant de rechercher les valeurs ou vecteurs propres sous forme exacte. Voir le fichier **linalgcas\help**().

Les fonctionnalités offertes en algèbre linéaire ont également été utilisées pour définir un programme de résolution de systèmes d'équations différentielles linéaires, du type $X'(t) = A \cdot X(t) + B(t)$.

Vous trouverez plus d'information à ce sujet dans le **chapitre 9** (algèbre linéaire) et dans le **chapitre 10** (utilisation du programme de résolution d'équations différentielles).





Informations complémentaires

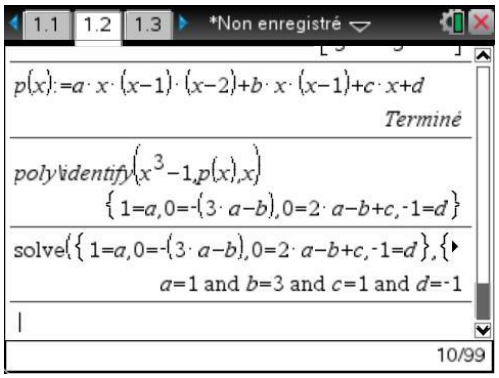
Certains programmes de la bibliothèque **linalg** doivent déterminer plusieurs éléments (matrices). Les résultats sont mémorisés dans différentes variables globales qui restent utilisables après l'exécution de ces programmes en vue d'une éventuelle utilisation ultérieure dans d'autres calculs.

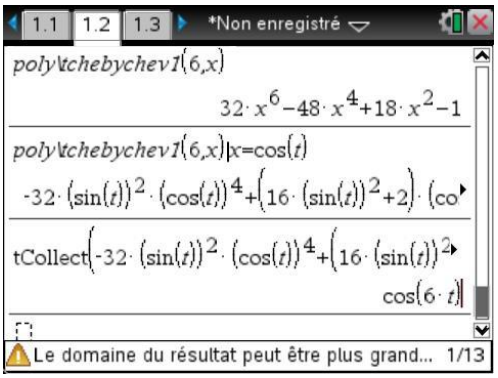
- Le programme de réduction de Gauss pas à pas **gausstep** utilise la variable globale :
  θmatg Réduite de Gauss
- Le programme d'inversion de matrice pas à pas utilise les deux variables globales :
  θmatg Réduite de Gauss
  θmatinv Matrice inverse
- Le programme de diagonalisation **diagonalization** (orthographe anglo-saxonne !) utilise les deux variables globales :
  θp Matrice de passage P
  θd Matrice diagonale D
  La matrice M transmise en argument vérifie l'égalité $M = P \cdot D \cdot P^{-1}$
- Le programme de décomposition DN **dn** utilise les variables globales suivantes :
  θp Matrice de passage P
  θd Matrice diagonale D
  θn Matrice nilpotente N
  θorder Indice de nilpotence de la matrice N
  La matrice M transmise en argument vérifie l'égalité $M = P \cdot (D + N) \cdot P^{-1}, N \cdot D = D \cdot N$
- Les programmes de résolution d'équations différentielles **desysinitcond**, **desysnewcond** et **desystem** utilisent les variables globales suivantes :
  θw Matrice wronskienne, base de solutions de l'équation homogène.
  θse Solution générale de l'équation complète.
  θsh Solution générale de l'équation homogène.
  θspart Solution particulière de l'équation complète.
  θsol Solution de l'équation complète vérifiant les conditions initiales.

Il est possible d'effacer toutes ces variables en utilisant le programme **clearmat.**

4.8 *La bibliothèque* poly

Cette bibliothèque comporte différentes fonctions utiles pour le calcul polynomial : identification, polynômes de Lagrange et d'interpolation, polynômes orthogonaux usuels. D'autres outils plus élaborés (résultant, matrice de Sylvester, algorithme de Gosper…) sont également disponibles (voir le fichier **poly\help**(), ainsi que les exemples de l'utilisation de **gosper_sum** fichier joint gosper3.tns).

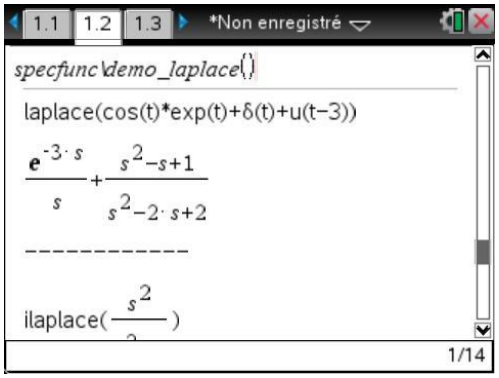


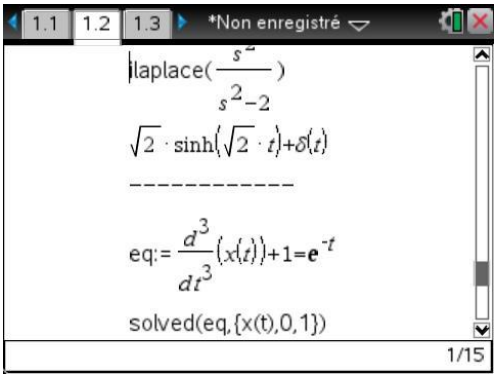


4.9 *La bibliothèque* specfunc

Cette bibliothèque permet d'effectuer une transformation de Laplace directe ou inverse.

Elle comprend également les outils de résolution d'équations différentielles linéaires d'ordre supérieur à 2 ou encore de systèmes d'équations différentielles linéaires.





Informations complémentaires

Il est nécessaire d'être en mode complexe rectangulaire, et en radians pour utiliser ces programmes.

Utiliser le programme **specfunc\demo_laplace** pour une série d'exemples d'utilisation.

Cette bibliothèque est une adaptation directe de l'ensemble de fonctions initialement contenues dans le package "Advanced Laplace 1.4" qui avait été écrit par Lars FREDERICKSEN pour la Voyage 200.

La version originale de ce package, dans sa version Voyage 200, est disponible sur les pages suivantes :

http://www.seg.etsmtl.ca/ti/laplace.html

http://paxm.org/symbulator/download/am.html

La puissance de calcul de la TI-Nspire CAS permet une utilisation beaucoup plus efficace de cet ensemble de fonctions.