

## Calcul matriciel

L'objectif de ce chapitre est de présenter l'utilisation de TI-Nspire CAS dans le domaine du calcul matriciel et de montrer comment il est possible d'en étendre les fonctionnalités.

### Sommaire

1. Présentation .....	2
2. Création de matrices .....	4
3. Recherche du rang d'une matrice .....	4
3.1 Étude directe à partir d'une réduite de Gauss .....	4
3.2 Fonction de calcul du rang .....	5
4. La recherche du noyau d'un endomorphisme .....	5
4.1 Recherche du noyau avec la fonction solve .....	6
4.2 Recherche du noyau avec la fonction zeros .....	6
4.3 Fonction de recherche des vecteurs du noyau .....	7
4.4 Fonction de recherche d'une base du noyau .....	8
5. Éléments propres .....	10
5.1 Valeurs propres d'une matrice .....	10
5.2 Vecteurs propres d'une matrice .....	10
6. Diagonalisation .....	11
6.1 Étude directe de la diagonalisation .....	11
6.2 Utilisation d'un programme de diagonalisation .....	12
7. Décomposition DN d'une matrice .....	14
7.1 Résultats utiles .....	14
8. Calcul des puissances symboliques .....	19
8.1 Méthode de calcul .....	19
8.2 Une astuce utile .....	19
8.3 Exemples d'utilisation .....	20
9. Exponentielle d'une matrice .....	20
9.1 Méthode utilisée .....	20
9.2 Exemple d'utilisation .....	21
10. Réduction de Gauss et inversion pas à pas .....	21
10.1 Exemple d'utilisation du programme de réduction .....	22
10.2 Exemple d'utilisation du programme d'inversion .....	23

# 1. Présentation

La TI-Nspire CAS permet d'effectuer directement (sans recours à l'écriture de programme) certains calculs matriciels assez complexes. Il est par exemple possible d'effectuer le calcul exact des puissances d'une matrice, et dans certains cas, le calcul approché des exponentielles.

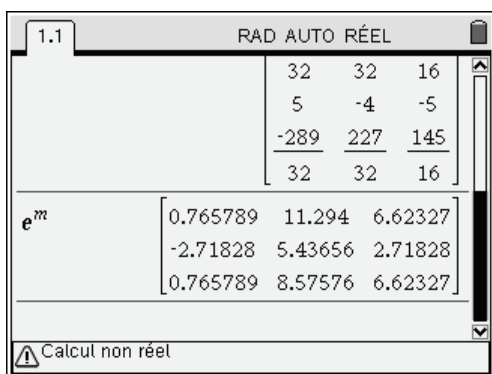
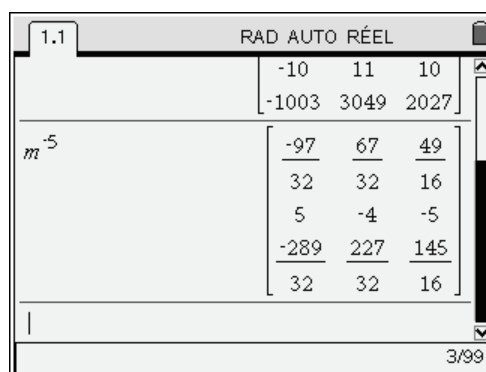
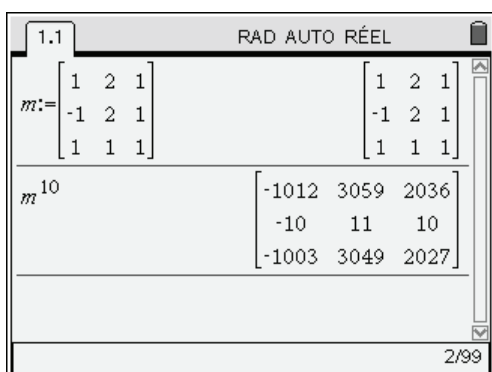
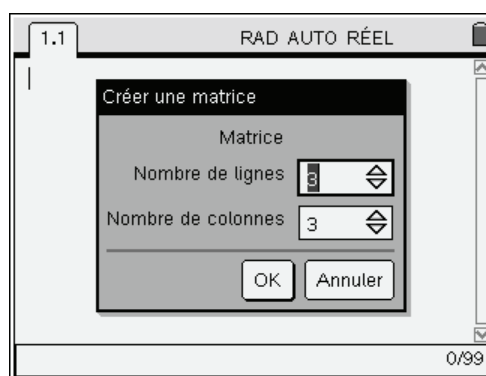
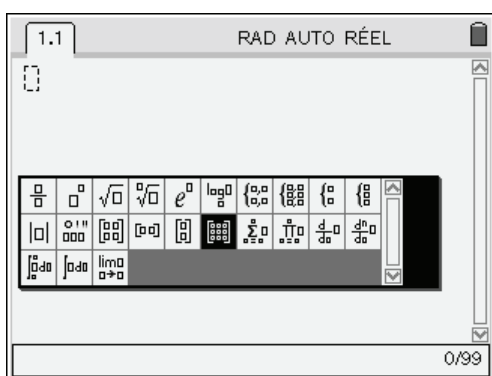
La façon la plus simple pour définir une matrice est d'utiliser l'un des modèles disponibles en appuyant sur  $\text{ctrl} + \left[ \begin{smallmatrix} \square & \square \\ \square & \square \end{smallmatrix} \right]$  :

$\left[ \begin{smallmatrix} \square & \square \\ \square & \square \end{smallmatrix} \right]$  Matrice  $2 \times 2$ .



$\left[ \begin{smallmatrix} \square & \square \\ \square & \square \end{smallmatrix} \right]$  Vecteur ligne de dimension 2 (matrice  $1 \times 2$ ).

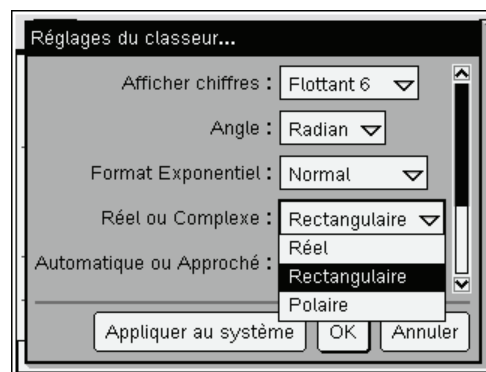
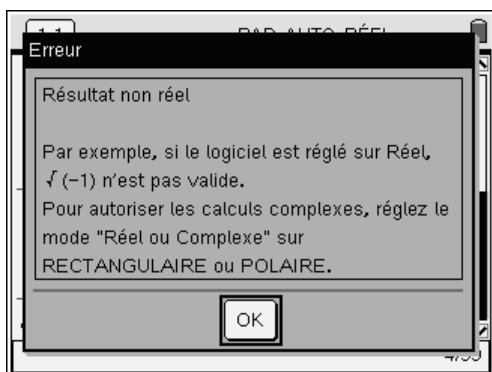
$\left[ \begin{smallmatrix} \square \\ \square \end{smallmatrix} \right]$  Vecteur colonne de dimension 2 (matrice  $2 \times 1$ ).

$\left[ \begin{smallmatrix} \square & \square & \square \\ \square & \square & \square \end{smallmatrix} \right]$  Matrice de taille quelconque. On obtient ensuite une boîte de dialogue permettant de choisir le nombre de lignes et de colonnes.

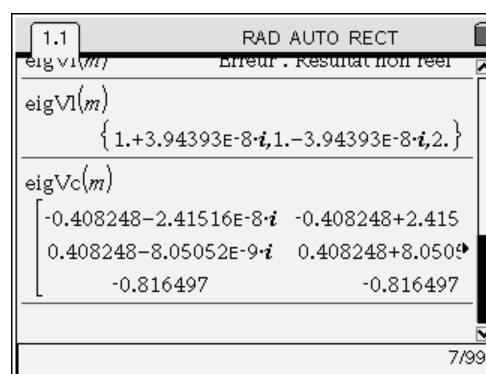
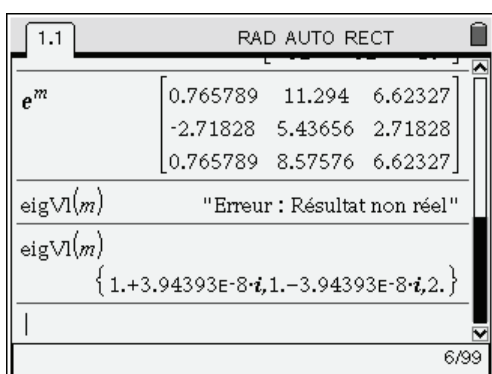


☞ Attention au message en bas de l'écran indiquant que certaines étapes du calcul ont été faites dans l'ensemble des nombres complexes.

On peut aussi obtenir l'expression approchée des valeurs ou vecteurs propres (dans  **7**  ) :



☞ Un calcul en mode réel peut entraîner le message d'erreur ci-dessus, on est convié à passer en mode complexe.



Les fonctions **eigVI** et **eigVc** sont des fonctions utilisant des algorithmes numériques, qui permettent d'obtenir des *valeurs approchées* des valeurs propres et des vecteurs propres. Ces algorithmes travaillent dans  $\mathbb{C}$ . Du fait des erreurs liées aux calculs numériques, il est possible que des valeurs qui devraient être réelles soient obtenues sous la forme de nombres complexes ayant une partie imaginaire très petite. C'est ce qui est arrivé dans l'écran ci-dessus.

Nous allons voir dans ce chapitre qu'il est également possible d'utiliser d'autres fonctions, permettant de résoudre différents problèmes classiques sous forme symbolique :

- Recherche du rang d'une matrice.
- Recherche du noyau d'un endomorphisme.
- Recherche de l'expression symbolique des valeurs et vecteurs propres.
- Diagonalisation d'un endomorphisme.
- Réduction d'un endomorphisme non diagonalisable.
- Calcul des puissances symboliques d'une matrice.
- Calcul de l'exponentielle d'une matrice.

Les fonctions et programmes que nous allons rencontrer et décrire dans ce chapitre font partie de la bibliothèque **linalgcas** téléchargeable sur le site [www.univers-ti-nspire.fr](http://www.univers-ti-nspire.fr).

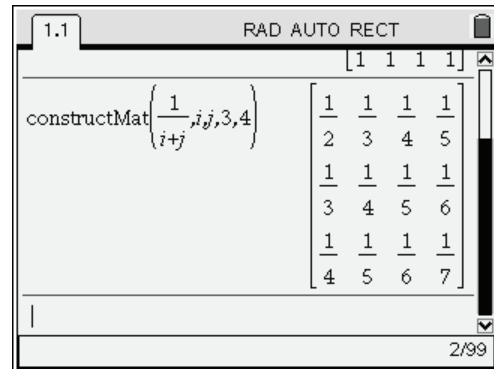
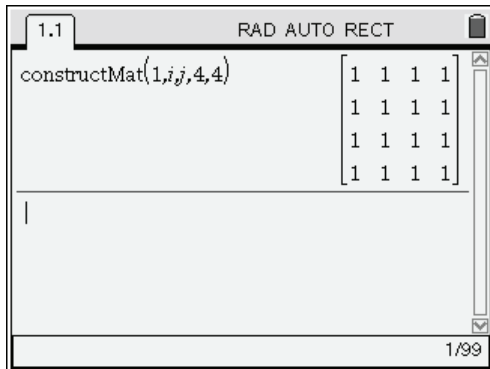


Voir la **démo d'utilisation** de la bibliothèque **linalgcas** sur le site [www.univers-ti-nspire.fr](http://www.univers-ti-nspire.fr).

## 2. Création de matrices

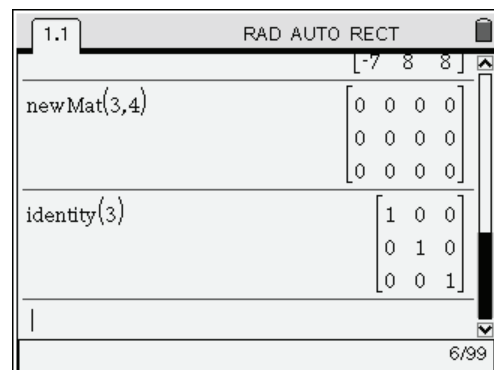
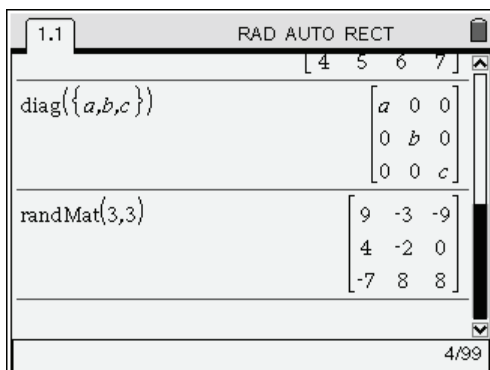
Comme nous l'avons vu dans le paragraphe précédent, on peut utiliser différents modèles prédéfinis. On peut aussi utiliser la syntaxe  $[1, 2; 3, 4]$  le séparateur ; ( $\text{ctrl}(\odot)$ ) indiquant le changement de ligne.

Le sous-menu **Créer** du menu **Matrice & vecteur**, accessible par  $\text{menu}(\odot)7(\odot)6$ , contient différentes fonctions de création de matrices. En particulier, la fonction **constructMat** permet de construire les matrices dont le terme général  $a_{ij}$  est défini par une expression.



Autres fonctions :

- **diag(liste)** : matrice diagonale. La diagonale est définie par une liste (entre { }).
- **randMat(n, p)** : matrice aléatoire, ayant  $n$  lignes et  $p$  colonnes.
- **newMat(n, p)** : matrice nulle, ayant  $n$  lignes et  $p$  colonnes.
- **identity(n)** : matrice unité  $n \times n$ .



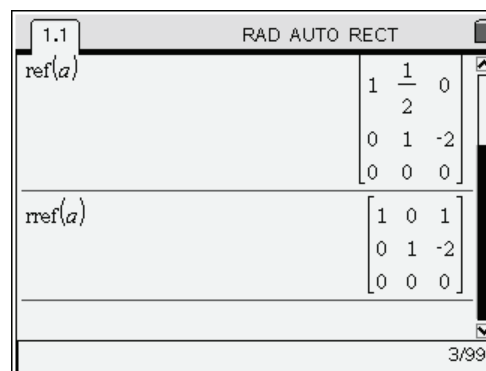
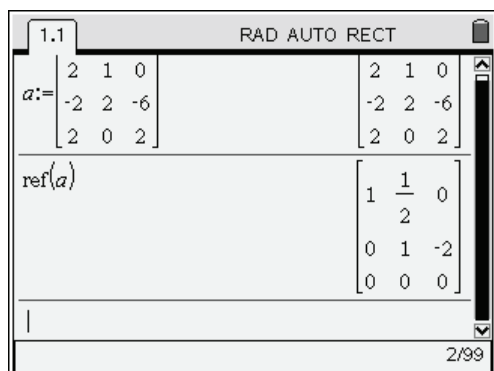
## 3. Recherche du rang d'une matrice

### 3.1 Étude directe à partir d'une réduite de Gauss

Considérons par exemple la matrice  $A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 2 & -6 \\ 2 & 0 & 2 \end{bmatrix}$ .

Pour obtenir le rang de cette matrice, il suffit d'effectuer une réduction de Gauss de cette matrice (**ref**), et de compter le nombre de lignes non nulles.

☞ On peut aussi utiliser la réduite de Gauss-Jordan (**rref**).



Dans cet exemple, on peut voir qu'il y en a deux. Le rang est donc égal à 2.

### 3.2 Fonction de calcul du rang

Il est facile d'écrire une fonction automatisant ce décompte. L'idée est d'ajouter des nombres  $n_i$ , égaux à 1 si la ligne  $n^{\circ}i$  est non nulle, et égaux à 0 dans le cas contraire. Pour tester si la ligne  $n^{\circ}i$  est nulle, il suffit de tester la valeur de sa norme. Ce qui conduit à l'écriture de la fonction suivante.

```

Define LibPub rank(m)=Func
©m: rang de m
Local n
n:= rref(m)
Σ(when(norm(n[i])=0,0,1),i,1,rowDim(n))
EndFunc

```

☞ La fonction **when** s'utilise sous la forme

**when**(condition, résultat-si-vrai, résultat-si-faux, résultat-si-l'on-ne-sait-pas).

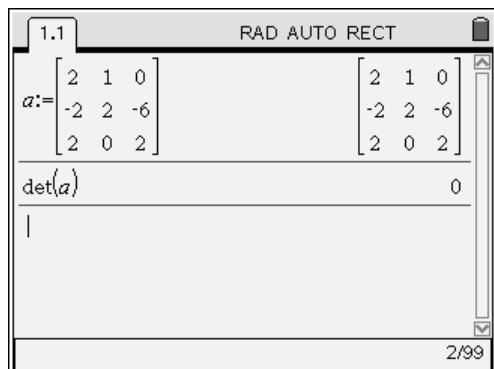
☞ Si **n** est une matrice, **n[i]** correspond à la  $i$ -ème ligne. **norm(n[i])** sera égal à 0 si et seulement si cette ligne est nulle. **rowDim(n)** correspond au nombre de lignes de la matrice.

Reportez-vous au [chapitre 16](#) pour une découverte de la programmation sur TI-Nspire CAS, et au [chapitre 17](#) pour plus d'information sur l'utilisation des commandes **Define LibPub**, **Define LibPriv**.

## 4. La recherche du noyau d'un endomorphisme

Considérons à nouveau la matrice  $A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 2 & -6 \\ 2 & 0 & 2 \end{bmatrix}$ .

Même sans utiliser ce qui précède, on peut vérifier que cette matrice n'est pas inversible. Il suffit de calculer le déterminant de la matrice.

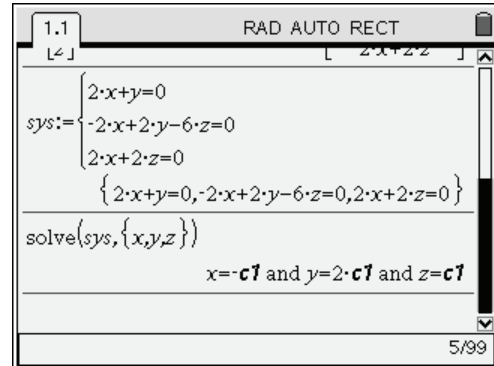
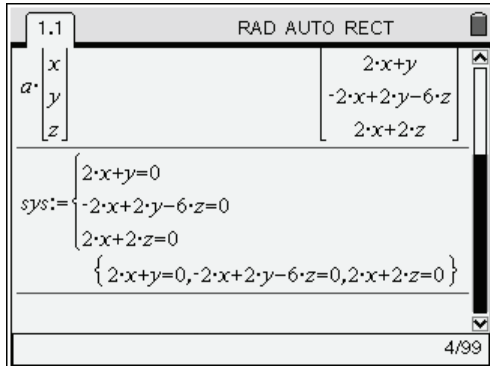


Comment obtenir le noyau de l'application linéaire associée ?

#### 4.1 Recherche du noyau avec la fonction solve

Une première méthode consiste à résoudre le système d'équations définissant le noyau avec la fonction **solve**.

Le système est écrit à l'aide du modèle  $\begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$ , disponible en appuyant sur  $\text{ctrl} \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}$ .

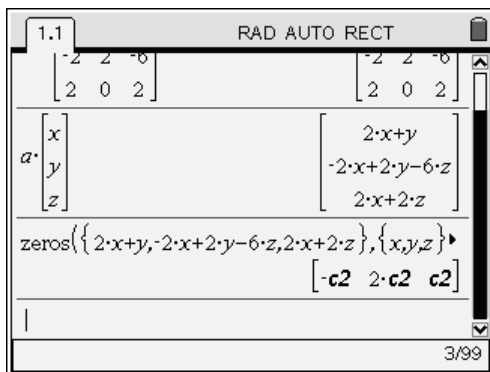


Les solutions sont donc du type  $(-z, 2z, z)$ , avec  $z$  quelconque. Le noyau est donc de dimension 1, et il est engendré par  $(-1, 2, 1)$ .

#### 4.2 Recherche du noyau avec la fonction zeros

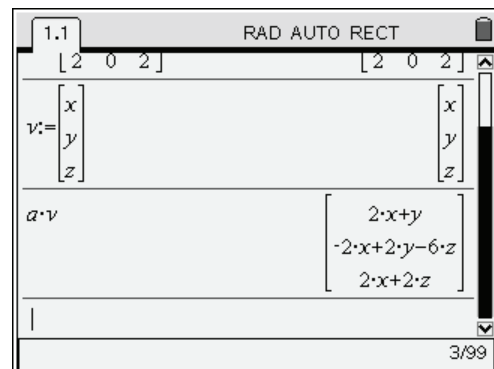
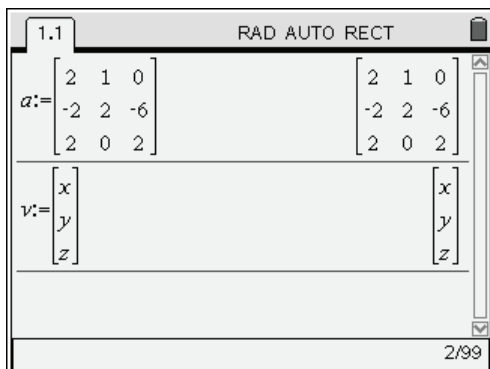
On peut aussi choisir d'utiliser la fonction **zeros**, qui permet d'avoir un résultat plus compact :

$\text{zeros}(\{2x+y, -2x+2y-6z, 2x+2z\}, \{x, y, z\})$

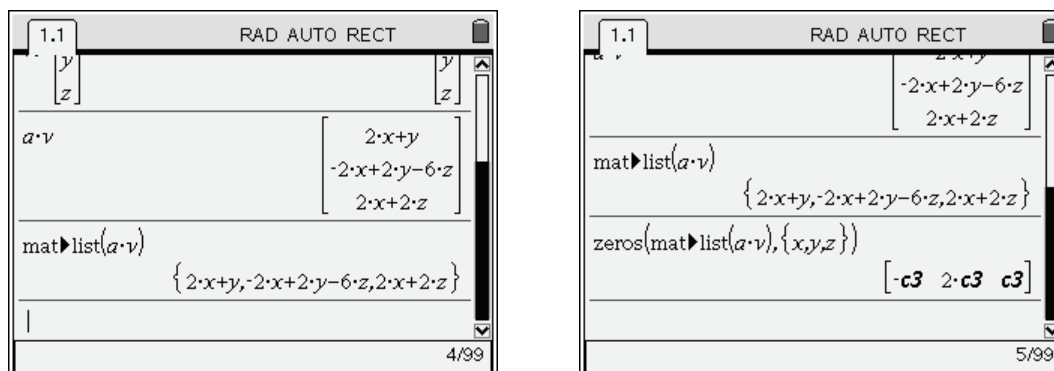


Il est possible de construire automatiquement la liste des équations à résoudre.

Pour cela, on définit un vecteur quelconque, et on fait le produit de la matrice par ce vecteur :



La fonction de conversion **mat►list** permet ensuite de récupérer la liste des composantes, et il est ensuite possible d'utiliser la fonction **zeros** :



### 4.3 Fonction de recherche des vecteurs du noyau

Il est possible d'automatiser ce qui vient d'être fait en utilisant deux fonctions.

Attention, le texte de ces deux fonctions utilise des méthodes un peu sophistiquées, vous pouvez les ignorer dans un premier temps.

```

Define LibPriv vect(n)=Func
©n: vecteur de dim n
Local i,v
v:=newMat(n,1)
For i,1,n
v[i,1]:=expr("θ"&string(i))
EndFor
v
EndFunc

```

On commence par construire une matrice colonne de  $n$  lignes, ne contenant que des 0. Ensuite, une boucle construit les variables  $\theta_1, \theta_2, \dots$  qui sont placées dans ce vecteur. On utilise pour cela plusieurs fonctions de manipulation des chaînes de caractères.

1. La fonction **string** est utilisée pour convertir le contenu de la variable  $i : 1, 2, \dots$  en chaîne de caractères : "1", "2", ...
2. En utilisant l'opérateur de concaténation **&**, on colle cette chaîne à "**θ**" pour obtenir "**θ1**", "**θ2**", ...
3. La fonction **expr** permet de fabriquer les noms de variables  $\theta_1, \theta_2, \dots$  à partir de ces dernières chaînes de caractères.

Voici à présent le texte de la fonction permettant d'obtenir les vecteurs du noyau par la méthode décrite dans le paragraphe précédent :

```

Define LibPub vker(m)=Func
© mat: vect. de ker(mat)
Local i,n,v,leq,linc
n:=colDim(m)
v:=vect(n)
leq:=mat►list(m*v)
linc:= mat►list (v)
expr("zeros("&string(leq)&","&string(linc)&")")
EndFunc

```

La fonction **vect** construit un vecteur quelconque de dimension donnée. La fonction **vker** utilise ce vecteur pour déterminer le noyau en utilisant la méthode précédente.

☞ **vect** doit être définie dans la même activité que **vker** pour que cette dernière puisse fonctionner (voir le **chapitre 17** à ce sujet).

La dernière ligne de cette fonction est pour le moins ésotérique. Elle permet de résoudre un problème qui n'a rien d'évident.

Si une variable, par exemple **leq**, contient la liste des équations, par exemple  $\{2x+y, x-y\}$  et si une autre variable, **linc**, contient la liste des inconnues, par exemple  $\{x, y\}$ , alors

**zeros(leq,linc)**

n'est pas équivalent à

**zeros({2x+y,x-y},{x,y})**

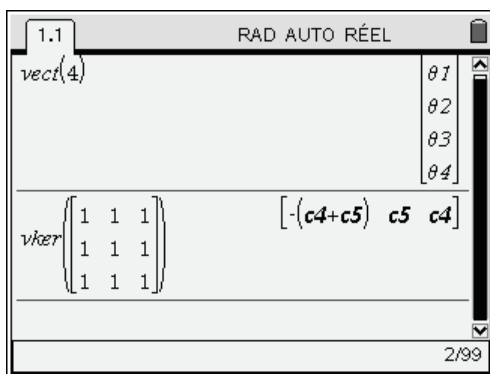
Ceci provient d'un mécanisme d'évaluation un peu particulier de la fonction **zeros**. Comme nous l'avons vu dans l'exemple du paragraphe précédent, il est possible de mémoriser une liste d'équations dans une variable, par contre, il n'est pas possible de faire de même avec la liste des inconnues.

Le seul moyen de résoudre le problème consiste à construire la chaîne de caractères :

**"zeros({2x+y,x-y},{x,y})"**

à partir du contenu des variables **leq** et **linc**, puis à exécuter l'instruction contenue dans cette chaîne à l'aide de la fonction **expr**. C'est exactement ce que fait notre dernière ligne...

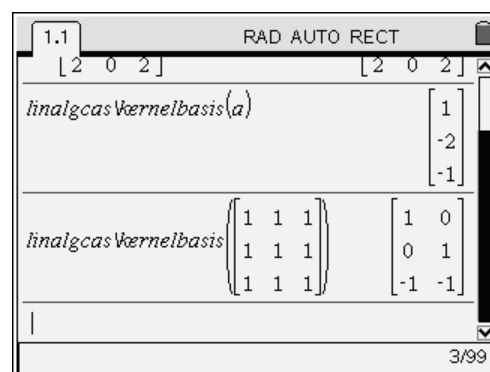
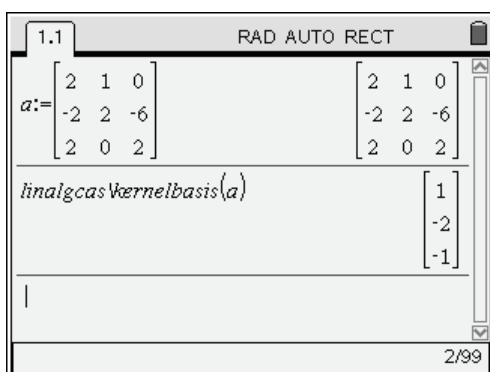
Voici un exemple d'utilisation de ces deux fonctions :



☞ La fonction **vker** correspond à la fonction **kernelvectors** de la bibliothèque **linalgcas**, téléchargeable sur le site [www.univers-ti-nspire.fr](http://www.univers-ti-nspire.fr).

#### 4.4 Fonction de recherche d'une base du noyau

Nous allons maintenant écrire une fonction qui sera capable de déterminer directement une base du noyau d'une application linéaire définie par sa matrice. Voici deux exemples illustrant l'utilisation de cette fonction **kernelbasis** :





#### 4.5 Texte de la fonction kernelbasis

```

Define LibPub kernelbasis(m)=Func
©mat: noyau de mat nxn
Local a
If det(m)≠0 Then
[0]
Else
a:=rref(augment(m†,1+0*m))
(subMat(a, rank(m)+1,rowDim(m)+1))†
EndIf
EndFunc

```

#### 4.6 Quelques explications sur la fonction kernelbasis

Les explications qui suivent nécessitent quelques connaissances de base sur l'interprétation des opérations sur les lignes d'une matrice en terme de produits matriciels, et sur le calcul de produit par blocs. Si vous ne connaissez pas encore ces notions, n'hésitez pas à simplement laisser de côté la suite de ce paragraphe.

Dans ce qui suit on parle du "noyau d'une matrice", il s'agit d'un raccourci pour désigner le noyau de l'endomorphisme associé à cette matrice.

Pour chercher la réduite de Gauss d'une matrice, on effectue des opérations élémentaires sur les lignes de la matrice. Cela revient à faire un produit, à gauche, par une matrice inversible.

Soit  $A$  une matrice  $n \times n$ , et  $I_n$  la matrice identité de même dimension.

Si l'on fait une réduction de ce type sur une matrice du type  $M = \begin{bmatrix} A & I_n \end{bmatrix}$ , matrice  $n \times 2n$ , on obtient  $G = T \cdot \begin{bmatrix} A & I_n \end{bmatrix} = \begin{bmatrix} T \cdot A & T \cdot I_n \end{bmatrix} = \begin{bmatrix} T \cdot A & T \end{bmatrix}$ .

Par ailleurs, en raison de la nature d'une réduite de Gauss, la matrice  $G$  sera du type  $G = \begin{bmatrix} P & Q \\ O & R \end{bmatrix}$ , avec  $P$  et  $Q$  matrices triangulaires  $r \times n$ ,  $O$  matrice nulle  $(n-r) \times n$ ,  $R$  matrice  $(n-r) \times n$ .

En identifiant avec ce qui précède, on a

$$T = \begin{bmatrix} Q \\ R \end{bmatrix} \text{ et } G = \begin{bmatrix} Q \\ R \end{bmatrix} \cdot \begin{bmatrix} A & I_n \end{bmatrix} = \begin{bmatrix} Q \cdot A & Q \\ R \cdot A & R \end{bmatrix} = \begin{bmatrix} P & Q \\ O & R \end{bmatrix}.$$

En particulier  $R \cdot A = 0$  et donc, en transposant,  ${}^t A \cdot {}^t R = 0$ .

La matrice  ${}^t R$  est une matrice  $n \times (n-r)$ , elle est extraite de  ${}^t T$  qui est inversible. Les  $n-r$  vecteurs colonnes de cette matrice forment donc une famille libre.

De plus,  ${}^t A \cdot {}^t R = 0$ , et donc pour tout vecteur colonne  $V_j$  de  ${}^t R$ ,  ${}^t A \cdot V_j = 0$ .

On a ainsi trouvé  $n-r$  vecteurs  $V_j$ , formant une famille libre, et tous dans le noyau de  ${}^t A$  qui est précisément de dimension  $n-r$ .

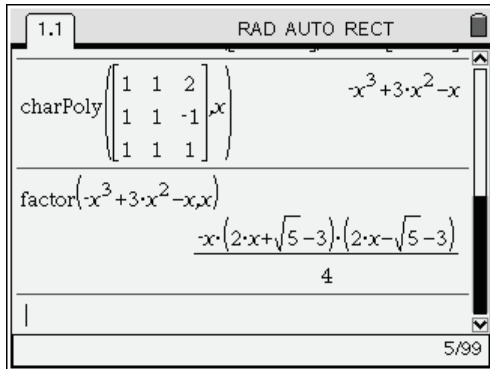
En conclusion, la réduction de Gauss de la matrice  $M = \begin{bmatrix} A & I_n \end{bmatrix}$  sous la forme  $G = \begin{bmatrix} P & Q \\ O & R \end{bmatrix}$  permet de trouver une base du noyau de  ${}^t A$  : il suffit d'extraire  $R$  et de transposer cette matrice.

Pour obtenir une base du noyau de  $A$ , il suffit de partir de  $M = \begin{bmatrix} {}^t A & I_n \end{bmatrix}$  comme on le fait dans la fonction **kernelbasis**.

## 5. Éléments propres

### 5.1 Valeurs propres d'une matrice

Pour calculer les valeurs exactes des valeurs propres d'une matrice, on doit "simplement" rechercher les racines du polynôme caractéristique. Ce polynôme s'obtient à l'aide de la fonction **charPoly** qui correspond au calcul de  $\det(M - xI_n)$ .



Voici maintenant deux courtes fonctions permettant de résoudre le problème de la recherche des valeurs propres.

La fonction **valp** recherche les valeurs propres réelles :

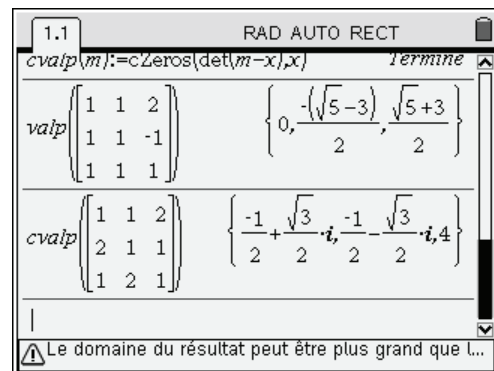
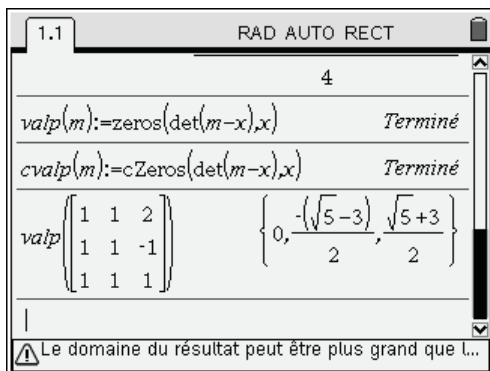
Define LibPub valp(m)=zeros(det(m-x),x)

La fonction **cvalp** traite également les valeurs propres complexes :

Define LibPub cvalp(m)=cZeros(det(m-x),x)

☞ Lors de l'addition entre une matrice de  $M_n(\mathbb{K})$  et un scalaire  $\lambda$ , le scalaire est automatiquement identifié à la matrice  $\lambda I_n$ .

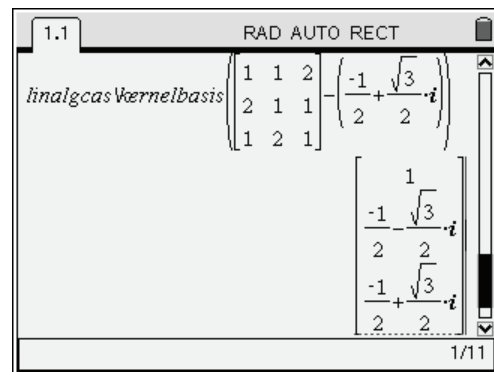
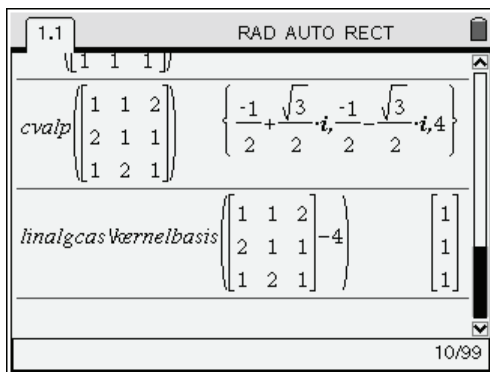
Vous trouverez dans la bibliothèque **linalgcas** les fonctions **eigenvals** et **ceigenvals** ayant les mêmes fonctionnalités que les fonctions ci-dessus.



### 5.2 Vecteurs propres d'une matrice

Pour déterminer les vecteurs propres d'un endomorphisme  $f$  défini par une matrice  $M$ , il suffit d'utiliser la fonction **kernelbasis** pour déterminer une base de chaque  $E_\lambda = \ker(f - \lambda Id)$ .

Voici par exemple une base de deux espaces propres associés à la dernière matrice :



## 6. Diagonalisation

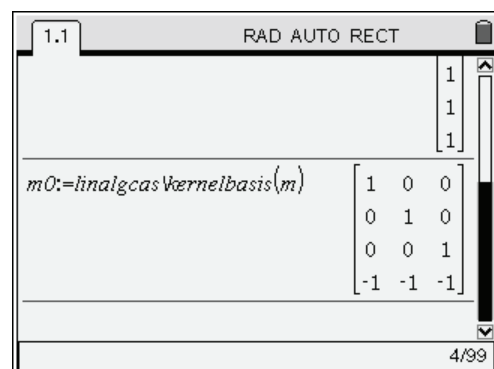
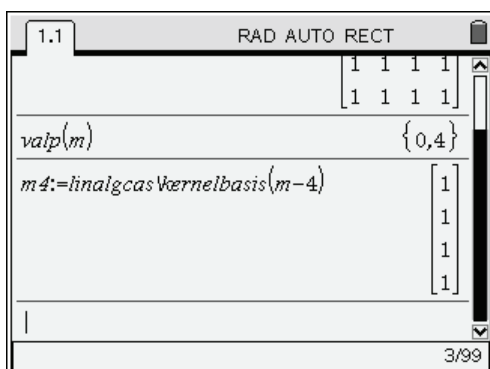
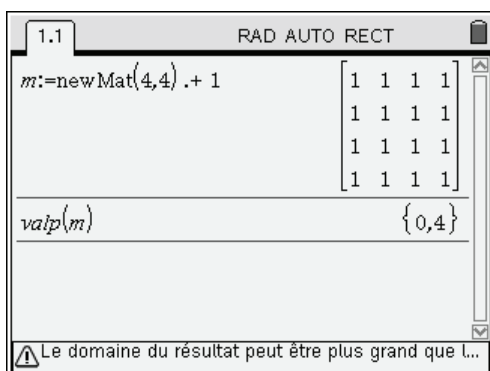
### 6.1 Étude directe de la diagonalisation

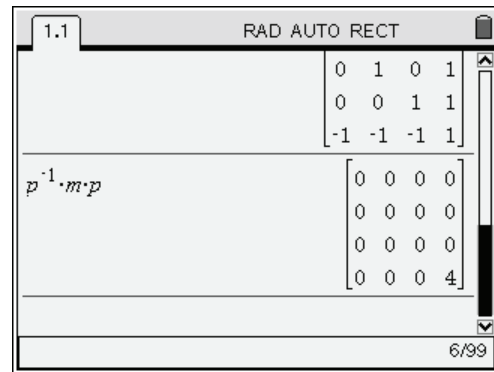
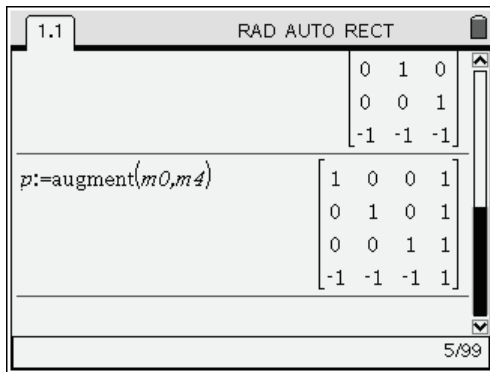
Tout est en place pour résoudre le problème de la diagonalisation des matrices.

Il suffit de rechercher les valeurs propres, puis les espaces propres associés.

Concrètement, on doit “concaténer” les matrices donnant les bases de chaque espace propre pour obtenir la matrice de passage dans le cas où la matrice est effectivement diagonalisable.

En voici un exemple :





- ☞ Sur le premier écran, **newMat** permet de créer une matrice nulle de la taille indiquée, l'opérateur `.*` permet d'ajouter 1 à chaque élément de la matrice, et donc d'obtenir la matrice avec tous les coefficients égaux à 1. On aurait également pu utiliser l'instruction **fill** ou directement **constructMat** pour obtenir ce résultat.
- ☞ Sur le quatrième écran, la fonction **augment** permet de créer une matrice à partir de deux autres matrices ayant un même nombre de lignes. Pour superposer deux matrices ayant le même nombre de colonnes, on utilisera la fonction **colAugment**.

## 6.2 Utilisation d'un programme de diagonalisation

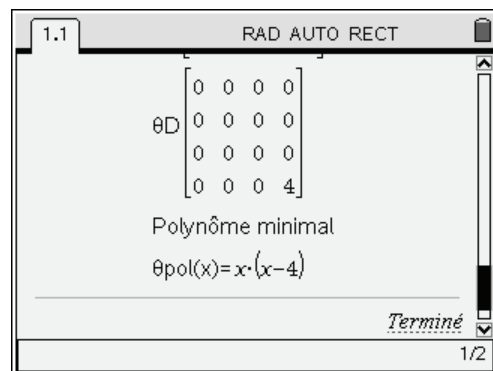
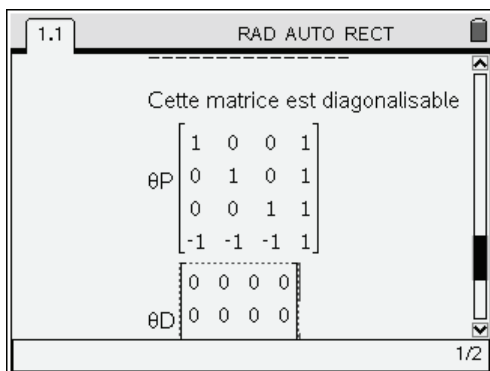
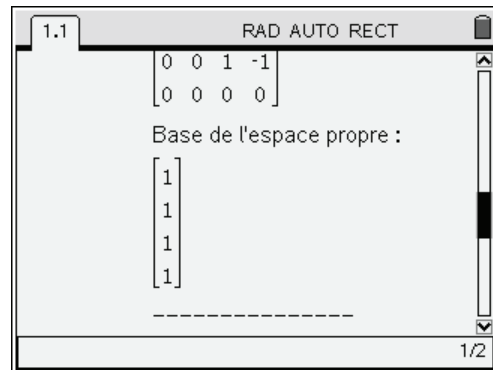
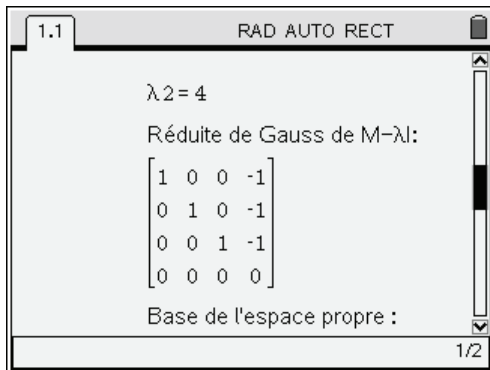
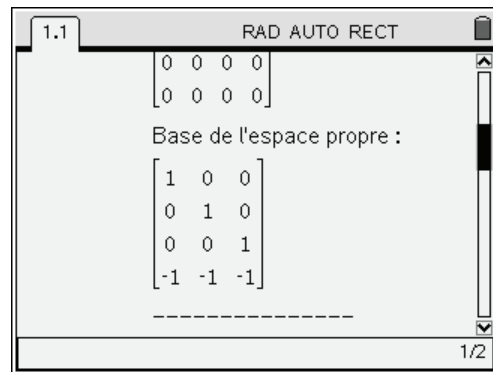
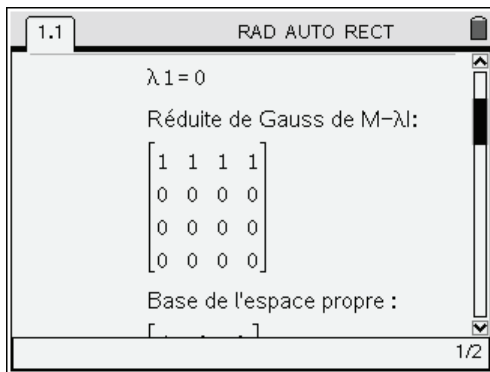
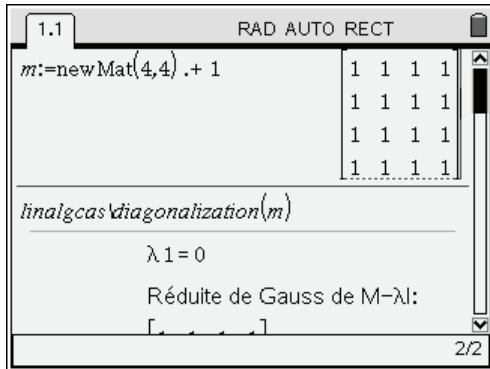
La diagonalisation est effectuée automatiquement par le programme **diagonalization**<sup>1</sup> de la bibliothèque **linalgcas** téléchargeable sur le site [www.univers-ti-nspire.fr](http://www.univers-ti-nspire.fr), voir **chapitre 17**.

1. Ce programme affiche la première valeur propre  $\lambda_1$ .
  2. Puis, il affiche la matrice obtenue en cherchant une réduite de Gauss de  $M - \lambda_1 I_n$ , ce qui est utile lors d'une résolution manuelle pour déterminer le rang de cette matrice, la dimension de l'espace propre associé, et les éléments de cet espace propre.
  3. On obtient ensuite la matrice regroupant les vecteurs formant une base de l'espace propre.
- Les opérations 1, 2, 3 sont répétées pour chaque valeur propre.
4. Le programme affiche le cas échéant un message indiquant que la matrice est diagonalisable.
  5. Dans ce cas il affiche  $P$  et  $D$ , puis le polynôme minimal de la matrice. Ces éléments sont mémorisés dans les variables **θP**, **θD** et **θpol** en vue d'un éventuel usage ultérieur.

<sup>1</sup> Il s'agit de l'orthographe anglaise. Les noms des programmes ou fonctions de la bibliothèque de programmes **linalgcas** sont en anglais, comme pour toutes les fonctions intégrées à TI-Nspire CAS. Par contre, la langue des messages affichés lors de l'utilisation de cette bibliothèque s'adapte automatiquement au choix de la langue fait dans le menu **Réglages du système**.

### 6.3 Exemple d'utilisation

On recherche ici la diagonalisation de  $M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$



## 7. Décomposition DN d'une matrice

La diagonalisation d'une matrice est par exemple utile lorsque l'on cherche à calculer une puissance symbolique, une exponentielle, ou encore lors de la résolution d'un système différentiel. Cependant, toutes les matrices ne sont pas diagonalisables.

Pour être en mesure de traiter correctement les questions précédentes, nous allons commencer par mettre en place une méthode de décomposition d'une matrice en somme d'une matrice diagonalisable et d'une matrice nilpotente.

On cherche ici à obtenir une décomposition d'une matrice (à coefficients numériques) sous la forme  $M = P(D + N)P^{-1}$  avec  $D$  diagonale,  $N$  nilpotente,  $D \cdot N = N \cdot D$ .

N.B. Il est ensuite facile de calculer  $D' = P \cdot D \cdot P^{-1}$  et  $N' = P \cdot N \cdot P^{-1}$ .

On obtient ainsi  $M = N' + D'$ ,  $D'$  diagonalisable,  $N'$  nilpotente, et  $D' \cdot N' = N' \cdot D'$ .

Mais nous utiliserons plutôt la forme  $M = P(D + N)P^{-1}$  dans les calculs de ce chapitre.

Dans ce qui suit, on suppose que  $M \in M_n(\mathbb{K})$ , et on note  $u$  l'endomorphisme canoniquement associé à  $M$ .

### 7.1 Résultats utiles

Nous allons utiliser principalement les résultats suivants :

1. Si  $E = E_1 \oplus \dots \oplus E_p$ , et si chaque  $E_i$  est stable par  $u$ , alors la matrice de  $u$  dans une base associée à cette décomposition est diagonale par blocs, c'est-à-dire du type

$$\begin{bmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & B_p \end{bmatrix},$$

avec  $B_i \in M_{n_i}(\mathbb{K})$ ,  $n_i = \dim(E_i)$ .

2. Si  $f$  et  $g$  sont deux endomorphismes tels que  $f \circ g = g \circ f$ , alors le noyau et l'image de  $g$  sont stables par  $f$ .
3. Soit  $P$  le polynôme caractéristique de  $u$ .  
Le théorème de Cayley-Hamilton et le théorème de décomposition des noyaux montrent que si  $P(x) = (-1)^n (x - a_1)^{n_1} \dots (x - a_p)^{n_p}$ , avec les  $a_i$  deux à deux distincts, alors  $E = \ker(P(u)) = \ker(u - a_1 Id)^{n_1} \oplus \dots \oplus \ker(u - a_p Id)^{n_p}$ .  
L'espace est somme directe des sous-espaces caractéristiques.
4. L'utilisation des deux premières propriétés permet de montrer que dans une base associée à cette décomposition, la matrice de  $u$  est diagonale par blocs.  
(Il est clair que  $u$  commute avec  $(u - a_k Id)^{n_k}$ , comme avec tout autre polynôme en  $u$ ).

### 7.2 Description de la méthode utilisée pour la décomposition

Voici comment obtenir la décomposition recherchée à partir de ces propriétés en utilisant la calculatrice.

1. On commence par calculer le polynôme caractéristique de la matrice  $M$  en utilisant la fonction **charPoly**.

2. On factorise le polynôme obtenu sous la forme

$$\det(M - x.I) = (-1)^n (x - a_1)^{n_1} \dots (x - a_p)^{n_p}$$

Ce travail est fait par une fonction **lfactor** qui retourne la matrice

$$\begin{bmatrix} (x - a_1)^{n_1} & \dots & (x - a_p)^{n_p} \\ a_1 & \dots & a_p \\ n_1 & \dots & n_p \end{bmatrix}.$$

3. D'après le théorème de Cayley-Hamilton et le théorème de décomposition des noyaux,

$$E = \ker(P(u)) = \ker(u - a_1 Id)^{n_1} \oplus \dots \oplus \ker(u - a_p Id)^{n_p}.$$

Soit  $E_i = \ker(u - a_i Id)^{n_i}$  l'espace caractéristique associé à la valeur propre  $\lambda = a_i$ .

Il est possible d'obtenir une base de cet espace caractéristique en utilisant la fonction **kernelbasis**, décrite page 9.

Cette fonction retourne une matrice, formée par les composantes des vecteurs d'une base du noyau de la matrice fournie en argument.

En concaténant les matrices obtenues pour chaque espace caractéristique, on obtient la matrice de passage  $P$  vers une base associée à la décomposition en somme directe.

4. On calcule alors  $T = P^{-1} \cdot M \cdot P$  à l'aide de la calculatrice.

C'est la matrice associée à  $u$  dans cette base. Comme nous l'avons vu, cette matrice est toujours diagonale par blocs.

$$T = \begin{bmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & B_p \end{bmatrix}$$

Le  $i$ -ième bloc de la diagonale correspond à la matrice de la restriction de  $u$  à  $E_i$ .

5. On peut aussi construire la matrice diagonale  $D$  formée par les valeurs propres de la matrice  $M$ , chaque valeur  $a_i$  étant répétée  $k_i$  fois.
6. On calcule ensuite  $N = T - D$ .

$$\text{Cette matrice est du type } N = \begin{bmatrix} C_1 & 0 & \dots & 0 \\ 0 & C_2 & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & C_p \end{bmatrix},$$

avec  $C_i = B_i - a_i I_{n_i}$ , matrice de la restriction de  $u - a_i Id$  à  $E_i = \ker(u - a_i Id)^{n_i}$ .

On a donc en particulier  $C_i^{n_i} = 0$ , ce qui entraîne que  $N^{\max(n_1, n_2, \dots, n_p)} = 0$ .

La matrice  $N$  est donc nilpotente.

7. De plus la matrice  $D = \begin{bmatrix} a_1 I_{n_1} & 0 & \dots & 0 \\ 0 & a_2 I_{n_2} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & a_p I_{n_p} \end{bmatrix}$  commute avec  $N$  :

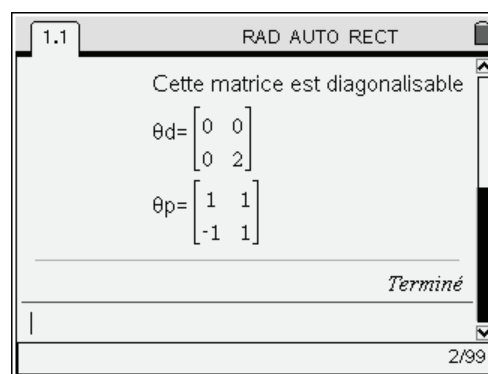
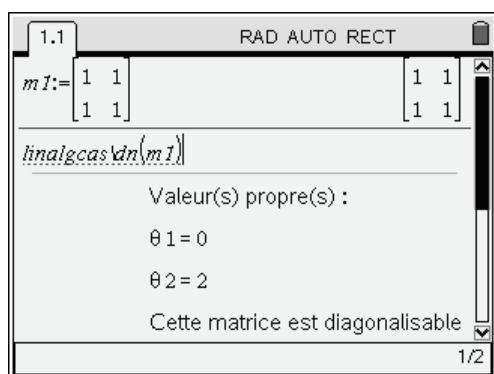
$$D \cdot N = \begin{bmatrix} a_1 I_{n_1} C_1 & 0 & \cdots & 0 \\ 0 & a_2 I_{n_2} C_2 & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & a_p I_{n_p} C_p \end{bmatrix} = \begin{bmatrix} a_1 C_1 & 0 & \cdots & 0 \\ 0 & a_2 C_2 & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & a_p C_p \end{bmatrix} = N \cdot D$$

En conclusion, on a obtenu  $P$ ,  $N$  et  $D$  telles que  $M = P(D + N)P^{-1}$  avec  $D$  diagonale,  $N$  nilpotente, et  $D \cdot N = N \cdot D$ .

### 7.3 Exemples d'utilisations du programme de décomposition

La décomposition DN d'une matrice est effectuée par le programme `dn` de la bibliothèque `linalgcas` téléchargeable sur le site [www.univers-ti-nspire.fr](http://www.univers-ti-nspire.fr), voir [chapitre 17](#). Les matrices  $P$ ,  $D$  et  $N$  sont mémorisées dans les variables `θp`, `θd`, `θn` en vue d'un éventuel usage ultérieur.

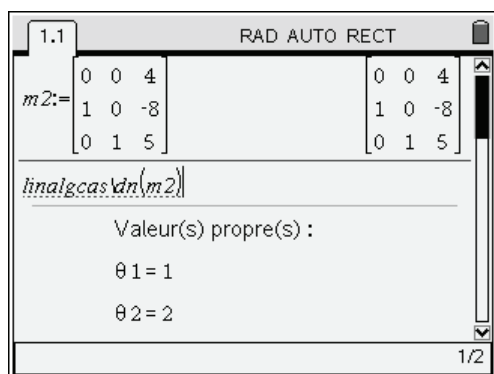
- $M_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$



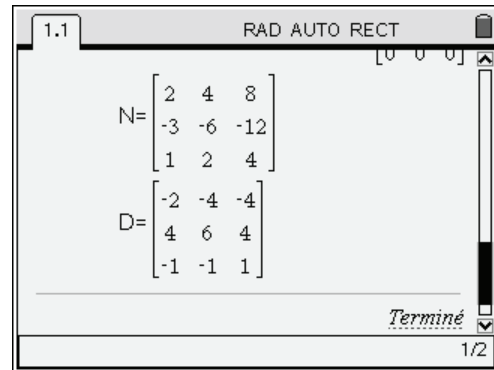
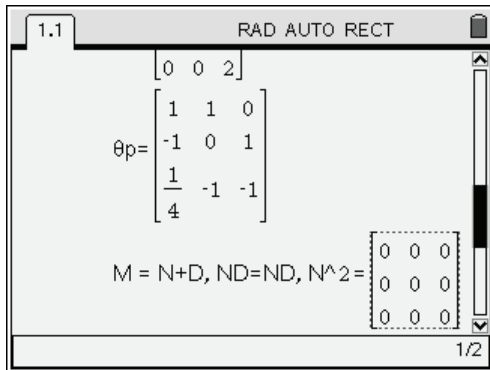
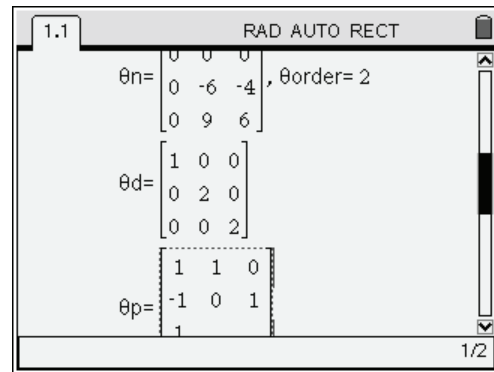
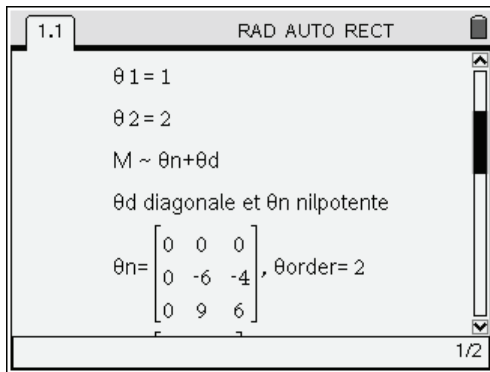
On obtient ici une matrice diagonalisable ( $N = 0$ ).

$$M_1 = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}^{-1}$$

- $M_2 = \begin{bmatrix} 0 & 0 & 4 \\ 1 & 0 & -8 \\ 0 & 1 & 5 \end{bmatrix}$







En conclusion

$$M_2 = P(D + N)P^{-1}$$

avec

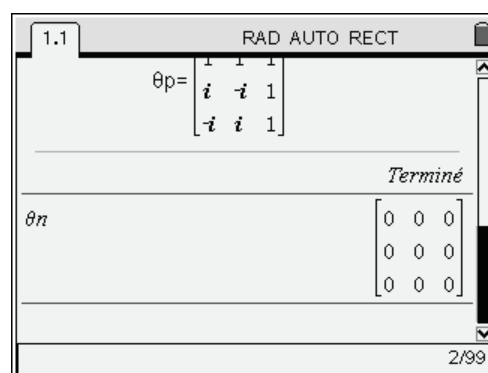
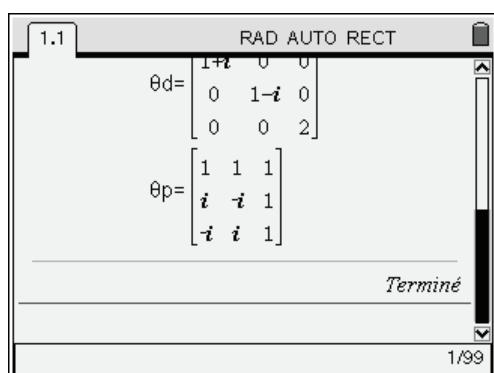
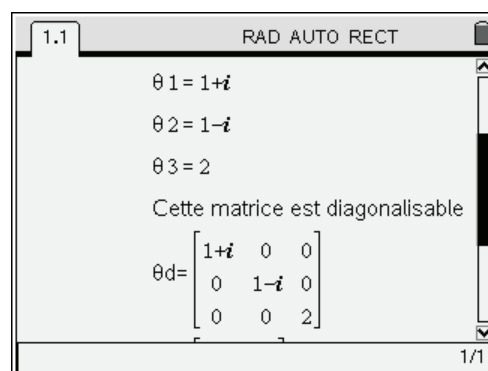
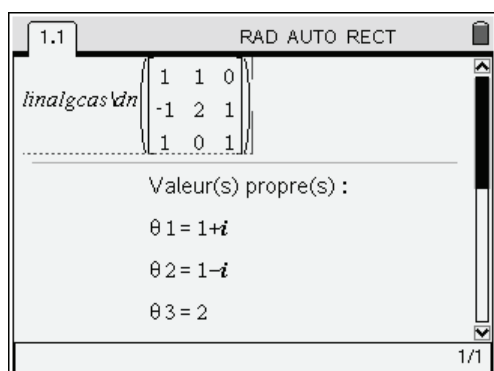
$$P = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 1/4 & -1 & -1 \end{bmatrix}$$

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

$$N = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -6 & -4 \\ 0 & 9 & 6 \end{bmatrix}$$

Cette matrice n'est pas diagonalisable.

$$\bullet M_3 = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 2 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$



On obtient

$$M_3 = P(D + N)P^{-1}$$

avec

$$P = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -i & i \\ 1 & i & -i \end{bmatrix}$$

$$D = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1-i & 0 \\ 0 & 0 & 1+i \end{bmatrix}$$

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

La matrice est donc diagonalisable.

## 8. Calcul des puissances symboliques

Le calcul des puissances d'une matrice est effectué par la fonction `pwrmat` de la bibliothèque `linalgcas` téléchargeable sur le site [www.univers-ti-nspire.fr](http://www.univers-ti-nspire.fr), voir [chapitre 17](#).

### 8.1 Méthode de calcul

On utilise le programme de décomposition `dn`, puis on utilise la formule du binôme de Newton.



Le calcul des puissances symboliques de la matrice diagonale  $D$  ne pose aucun problème.

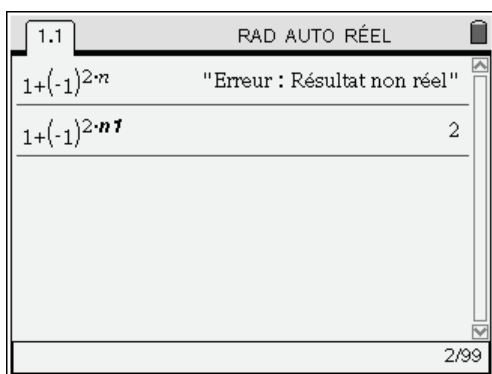
On utilise ici l'opérateur `.^` qui permet d'élever chaque élément d'une matrice à une puissance donnée. Dans le cas d'une matrice diagonale, et dans ce cas seulement,  $d.^n$  est égal à  $d^n$ .

Pour la matrice  $N$ , on effectue seulement le calcul de puissances entières numériques, en se limitant à l'ordre indiqué par la variable `thetaorder`.

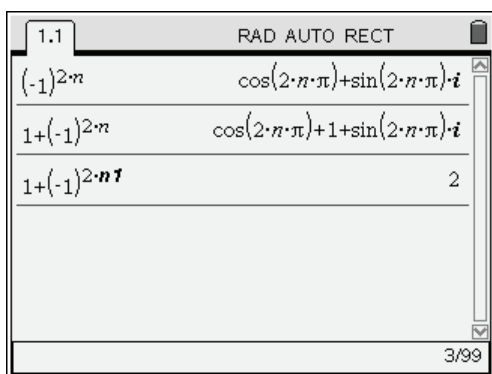
Le résultat du calcul est placé dans la variable globale `theta`.

### 8.2 Une astuce utile

On utilise la variable entière spécifique `n1` (symbole  $n$  accessible dans la palette de caractères  ), qui permet de faire des calculs corrects de puissances symboliques, en particulier avec les puissances d'entiers négatifs comme le montre l'écran ci-dessous :



On comprend mieux le premier résultat si on passe en mode complexe :



- ☞ Le résultat du calcul est "apparemment" complexe.  
 La calculatrice "ne sait pas" que  $n$  est un entier. Elle ne peut donc pas simplifier  $\cos(2n\pi)$  et  $\sin(2n\pi)$ .  
 En mode réel, l'obtention d'un résultat faisant intervenir  $(-1)^n$  provoque l'apparition du message d'erreur obtenu sur l'écran précédent.

### 8.3 Exemples d'utilisation

Voici quelques exemples d'utilisation de notre programme :

1.1 RAD AUTO RECT

$$\text{linalgcas}\backslash\text{pwrmat}\left(\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\right)$$

$$\begin{bmatrix} 2^{n1} & 2^{n1} \\ 2 & 2 \\ 2^{n1} & 2^{n1} \\ 2 & 2 \end{bmatrix}$$

1/99

1.1 RAD AUTO RECT

m2

$$\begin{bmatrix} 0 & 0 & 4 \\ 1 & 0 & -8 \\ 0 & 1 & 5 \end{bmatrix}$$

$$\text{linalgcas}\backslash\text{pwrmat}(m2)$$

$$\begin{bmatrix} (n1-3)\cdot 2^{n1+4} & (2\cdot n1-4)\cdot 2^{n1+4} & (4\cdot n1- \\ (4-\frac{3\cdot n1}{2})\cdot 2^{n1-4} & (5-3\cdot n1)\cdot 2^{n1-4} & (4-6\cdot n1 \\ (n1) & n1 & n1 \end{bmatrix}$$

2/2

1.1 RAD AUTO RECT

$$\begin{bmatrix} 1 & 0 & -8 \\ 0 & 1 & 5 \end{bmatrix}$$

$$\text{linalgcas}\backslash\text{pwrmat}(m2)$$

$$\begin{bmatrix} n1+4 & (2\cdot n1-4)\cdot 2^{n1+4} & (4\cdot n1-4)\cdot 2^{n1+4} \\ 2^{n1-4} & (5-3\cdot n1)\cdot 2^{n1-4} & (4-6\cdot n1)\cdot 2^{n1-4} \\ 2^{n1+1} & (n1-1)\cdot 2^{n1+1} & 2\cdot n1\cdot 2^{n1+1} \end{bmatrix}$$

1/2

1.1 RAD AUTO RECT

m3

$$\begin{bmatrix} 1 & 1 & 0 \\ -1 & 2 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\text{linalgcas}\backslash\text{pwrmat}(m3)$$

$$\begin{bmatrix} \cos\left(\frac{n1\cdot\pi}{4}\right)\cdot 2^{\frac{n1}{2}} & \frac{n1}{2} & \frac{n1}{2} \\ \cos\left(\frac{n1\cdot\pi}{4}\right) & \sin\left(\frac{n1\cdot\pi}{4}\right) & \frac{n1}{2} \\ \cos\left(\frac{n1\cdot\pi}{4}\right) & \sin\left(\frac{n1\cdot\pi}{4}\right) & \frac{n1}{2} \end{bmatrix}$$

2/2

1.1 RAD AUTO RECT

$$\text{linalgcas}\backslash\text{pwrmat}(m3)$$

$$\begin{bmatrix} \frac{n1}{2} & \frac{n1}{2} & \frac{n1}{2} \\ \cos\left(\frac{n1\cdot\pi}{4}\right) & \sin\left(\frac{n1\cdot\pi}{4}\right) & \frac{n1}{2} \\ \cos\left(\frac{n1\cdot\pi}{4}\right) & \sin\left(\frac{n1\cdot\pi}{4}\right) & \frac{n1}{2} \end{bmatrix}$$

1/2

## 9. Exponentielle d'une matrice

Le calcul de  $\exp(tM)$  est effectué par la fonction **expmat** de la bibliothèque **linalgcas** téléchargeable sur le site [www.univers-ti-nspire.fr](http://www.univers-ti-nspire.fr), voir **chapitre 17**.

### 9.1 Méthode utilisée

Soit  $M$  une matrice carrée et  $t$  un scalaire, on se propose de calculer  $\exp(tM)$ .

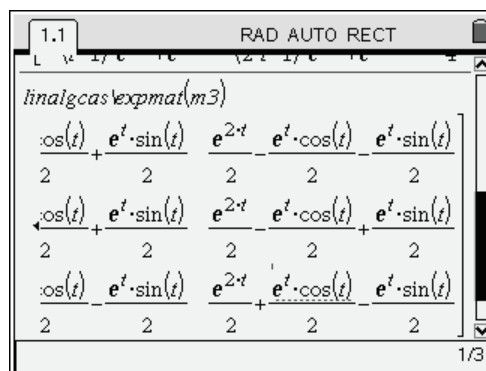
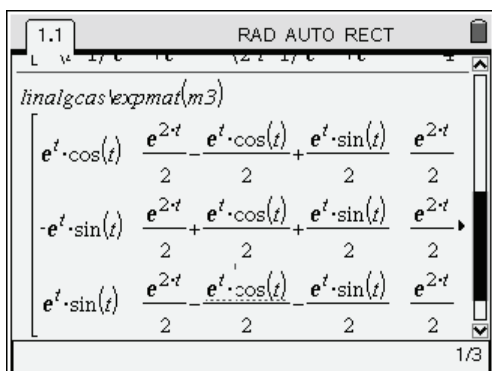
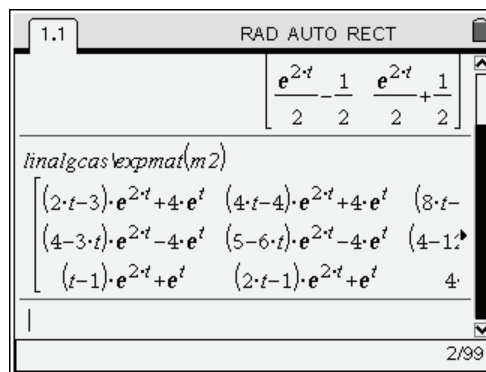
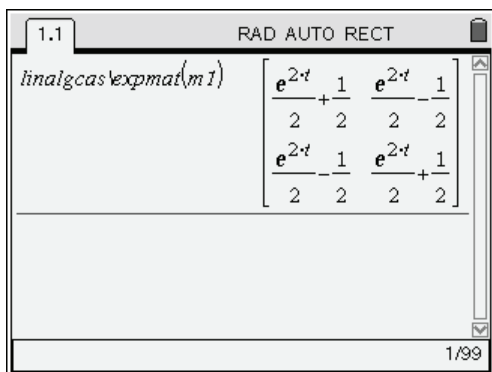
Cela sera particulièrement utile pour étendre les possibilités de la TI-Nspire CAS dans le domaine de la résolution des systèmes différentiels.

1. Comme dans le programme précédent, on commence par déterminer les trois matrices  $P$ ,  $D$  et  $N$  en utilisant le programme **dn**.
2. On calcule  $\exp(tD)$ , ce qui ne pose pas de problème.

3. On calcule ensuite  $\exp(tN)$  à partir d'un développement en série entière tronqué à l'ordre  $\theta$ order.
4. On peut ensuite utiliser le fait que  $\exp(A+B) = \exp(A) \cdot \exp(B)$  lorsque  $A \cdot B = B \cdot A$  et  $\exp(P \cdot A \cdot P^{-1}) = P \cdot \exp(A) \cdot P^{-1}$  pour calculer l'exponentielle de  $tM$ .
5. Le résultat du calcul est placé dans la variable globale  $\theta w$ .

### 9.2 Exemples d'utilisation

Voici les résultats obtenus lors du calcul de  $\exp(tM)$  pour les matrices **m1**, **m2** et **m3** définies précédemment :



## 10. Réduction de Gauss et inversion pas à pas

Comme nous l'avons vu, la TI-Nspire CAS dispose des fonctions **ref** et **rref** pour déterminer une réduite de Gauss d'une matrice. Il est également possible de déterminer directement l'inverse d'une matrice.

Vous trouverez également dans la bibliothèque **linalgcas** deux programmes permettant de faire ce type de calcul avec un affichage de toutes les opérations nécessaires.

Voici deux exemples d'utilisation de ces programmes.

### 10.1 Exemple d'utilisation du programme de réduction

La réduction de Gauss par étapes est effectuée par le programme **gausstep** de la bibliothèque **linalgcas**.

1.1 RAD AUTO RECT

```
m:=constructMat(i+j,i,j,3,3)
```

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

```
linalgcas\gausstep(m)
```

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

3·ligne1  
1/2

1.1 RAD AUTO RECT

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

$ligne2 = ligne2 - \frac{3 \cdot ligne1}{2}$

$$\begin{bmatrix} 2 & 3 & 4 \\ 0 & -\frac{1}{2} & -1 \\ 4 & 5 & 6 \end{bmatrix}$$

1/2

1.1 RAD AUTO RECT

$$\begin{bmatrix} 0 & -\frac{1}{2} & -1 \\ 2 & 3 & 4 \\ 4 & 5 & 6 \end{bmatrix}$$

$ligne3 = ligne3 - 2 \cdot ligne1$

$$\begin{bmatrix} 2 & 3 & 4 \\ 0 & -\frac{1}{2} & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

1/2

1.1 RAD AUTO RECT

$$\begin{bmatrix} 0 & -\frac{1}{2} & -1 \\ 2 & 3 & 4 \\ 0 & -1 & -2 \end{bmatrix}$$

$ligne3 \leftrightarrow ligne2$

$$\begin{bmatrix} 2 & 3 & 4 \\ 0 & -1 & -2 \\ 0 & -\frac{1}{2} & -1 \end{bmatrix}$$

1/2

1.1 RAD AUTO RECT

$$\begin{bmatrix} 0 & -1 & -2 \\ 0 & -\frac{1}{2} & -1 \\ 2 & 3 & 4 \end{bmatrix}$$

$ligne3 = ligne3 - \frac{ligne2}{2}$

$$\begin{bmatrix} 2 & 3 & 4 \\ 0 & -1 & -2 \\ 0 & 0 & 0 \end{bmatrix}$$

1/2

1.1 RAD AUTO RECT

$ligne3 = ligne3 - \frac{ligne2}{2}$

$$\begin{bmatrix} 2 & 3 & 4 \\ 0 & -1 & -2 \\ 0 & 0 & 0 \end{bmatrix}$$

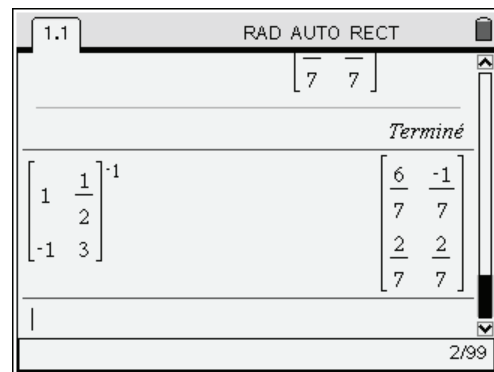
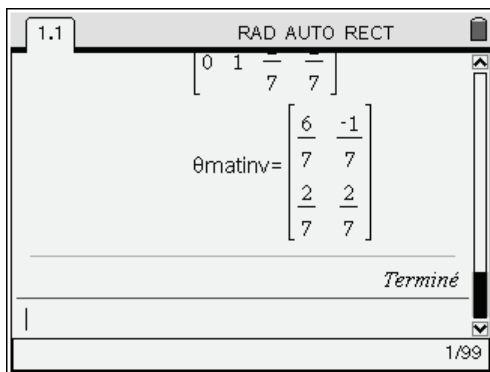
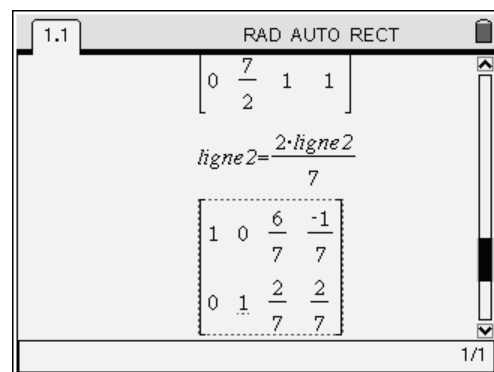
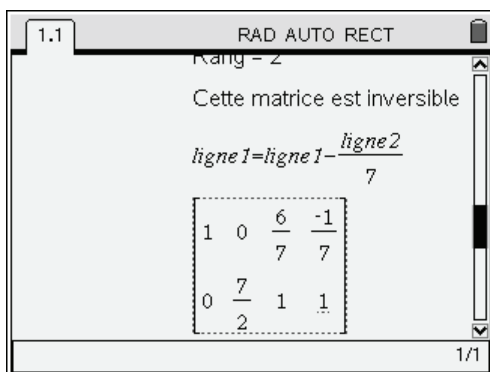
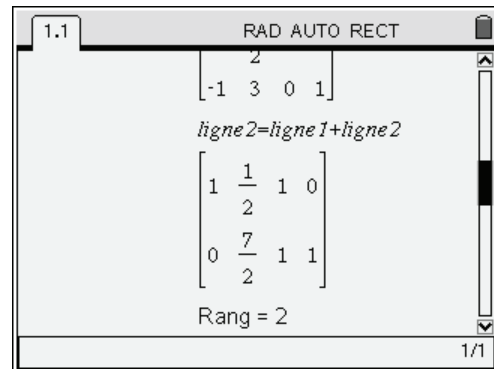
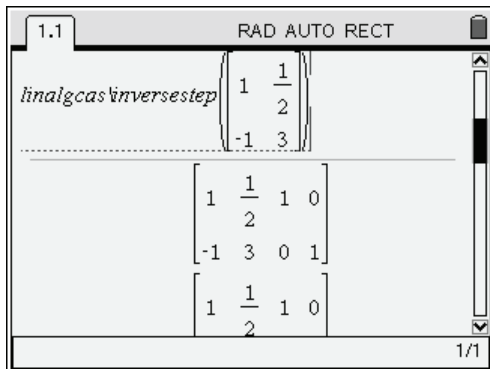
Rang = 2

Terminé

2/99

### 10.2 Exemple d'utilisation du programme d'inversion

L'inversion d'une matrice par la méthode des opérations élémentaires est effectuée par le programme **inversestep** de la bibliothèque **linalgcas**.



Vérification.

## Exercices

### 1 Calcul matriciel et suites récurrentes

Calculer la valeur exacte, sans programmation, mais à l'aide d'un calcul matriciel, du terme  $u_{100}$  de la suite de Fibonacci :  $u_0 = 1$ ,  $u_1 = 1$  et  $\forall n \geq 0$ ,  $u_{n+2} = u_{n+1} + u_n$ .

### 2 Matrices semblables

Les matrices  $A = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 0 & -2 \\ 0 & 1 & 3 \end{bmatrix}$  et  $B = \begin{bmatrix} 0 & 0 & 4 \\ 1 & 0 & -8 \\ 0 & 1 & 5 \end{bmatrix}$  sont-elles semblables ?

### 3 Matrices à paramètre

Pour quelles valeurs du paramètre réel  $a$ , la matrice  $M = \begin{bmatrix} a+1 & 1 & -1 \\ 2a+2 & 2 & -2 \\ 2 & -1 & 1 \end{bmatrix}$  est-elle diagonalisable ?

### 4 Équation matricielle

Trouver les matrices  $M$  telles que  $M^2 = \begin{bmatrix} 11 & -5 & 5 \\ -5 & 3 & -3 \\ 5 & -3 & 3 \end{bmatrix}$ .

*Indication.* On pourra commencer par montrer que si  $A$  et  $B$  sont deux matrices de  $M_n(\mathbb{C})$  telles que  $A$  admet  $n$  valeurs propres distinctes (et est donc diagonalisable) et  $B$  commute avec  $A$ , alors  $B$  est également diagonalisable, dans une même base de diagonalisation que la matrice  $A$ .



## Solutions des exercices

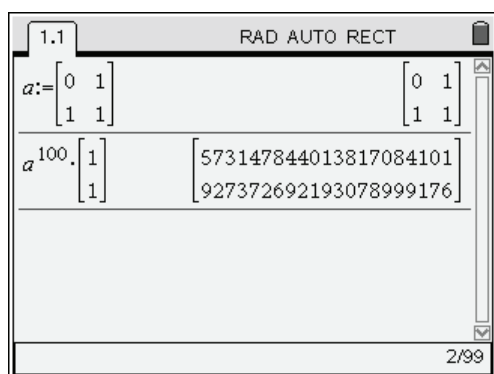
### 1 Calcul matriciel et suites récurrentes

On écrit que  $\begin{bmatrix} u_{n+1} \\ u_{n+2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_n \\ u_{n+1} \end{bmatrix}$ , d'où  $\begin{bmatrix} u_n \\ u_{n+1} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^n \cdot \begin{bmatrix} u_0 \\ u_1 \end{bmatrix}$ .

Il est alors facile de calculer  $u_{100}$ .

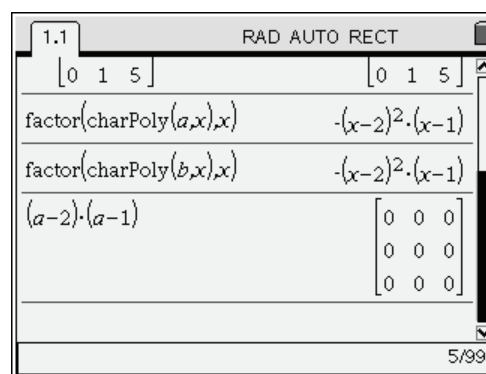
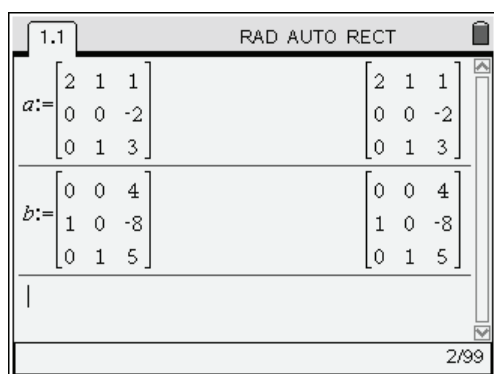
C'est la première composante du vecteur  $\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{100} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^{100} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ .

Un calcul de ce type ne pose aucun problème à une TI-Nspire CAS.



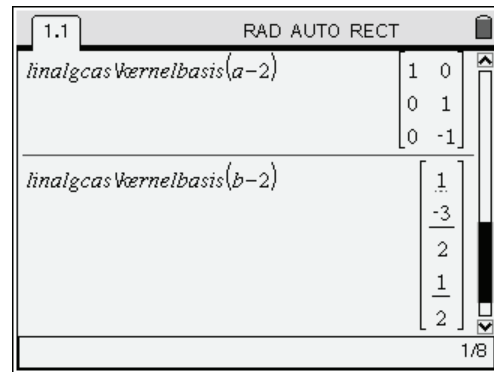
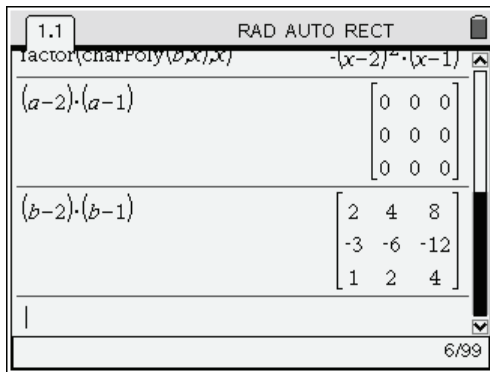
### 2 Matrices semblables

Voici une correction utilisant les fonctions de base de la calculatrice. On peut facilement vérifier que les valeurs propres de  $A$  et  $B$  sont 1 et 2. Le calcul suivant montre que  $A$  est diagonalisable (elle admet un polynôme annulateur scindé à racines simples).



En revanche  $B$  n'est pas diagonalisable. Si elle l'était  $(B - I)(B - 2I)$  serait nul.

On peut aussi le montrer en étudiant directement les espaces propres de  $A$  et  $B$  associés à la valeur propre 2 (voir écran de droite ci-après).



En conclusion,  $A$  et  $B$  ne sont pas semblables.

### 3 Matrices à paramètre

Pour résoudre cet exercice, nous allons utiliser les idées suivantes :

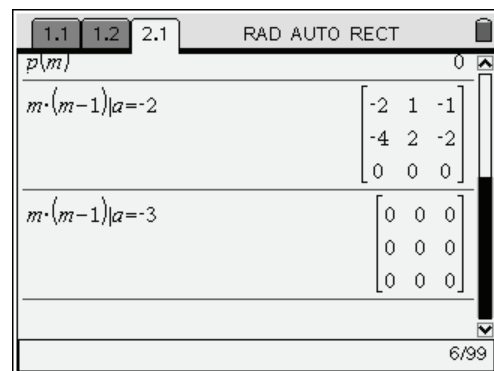
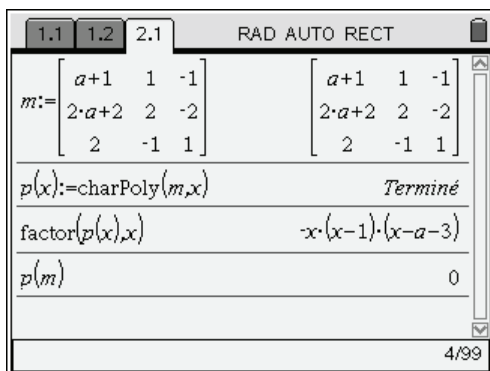
1. Si une matrice  $M$  admet un polynôme annulateur scindé à racines simples, alors elle est diagonalisable (son spectre est alors inclus dans l'ensemble des racines de ce polynôme).
2. Si  $M$  est diagonalisable et si son spectre est  $\{\lambda_1, \lambda_2, \dots, \lambda_p\}$ ,  $(X - \lambda_1)(X - \lambda_2) \cdots (X - \lambda_p)$  est un polynôme annulateur de  $M$ .

En particulier, si le polynôme caractéristique de  $M$  est  $P_M = (X - \lambda_1 I_n)^{n_1} (X - \lambda_2 I_n)^{n_2} \cdots (X - \lambda_p I_n)^{n_p}$ , il suffit de calculer le produit  $(M - \lambda_1 I_n)(M - \lambda_2 I_n) \cdots (M - \lambda_p I_n)$  pour savoir si la matrice est, ou n'est pas, diagonalisable. Si on obtient la matrice nulle, la matrice est diagonalisable. Dans le cas contraire, elle ne l'est pas.

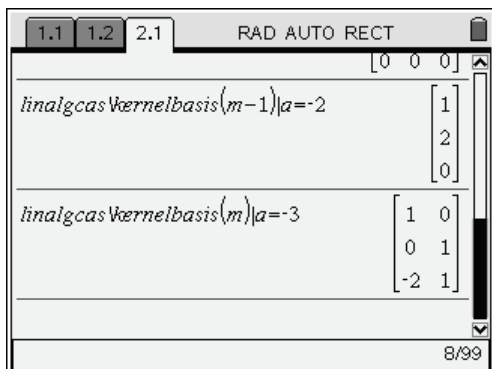
Calculons le polynôme caractéristique de la matrice  $M$  proposée dans cet exercice. On peut vérifier au passage le théorème de Cayley-Hamilton. Voir écran ci-dessous, à gauche.

Ici,  $P_M = -X(X-1)(X-a-3)$ .

- Si  $a+3$  est différent de 0 et de 1, c'est-à-dire  $a \neq -3$  et  $a \neq -2$ ,  $M$  possède trois valeurs propres distinctes et est donc diagonalisable.
- Si  $a = -2$ ,  $P_M = -X(X-1)^2$ . Le premier calcul fait dans l'écran de droite montre que le polynôme  $X(X-1)$  n'est pas annulateur.  $M$  n'est pas diagonalisable.
- Si  $a = -3$ ,  $P_M = -X^2(X-1)$ . Le second calcul montre que le polynôme  $X(X-1)$  est annulateur de  $M$  scindé à racines simples.  $M$  est diagonalisable.



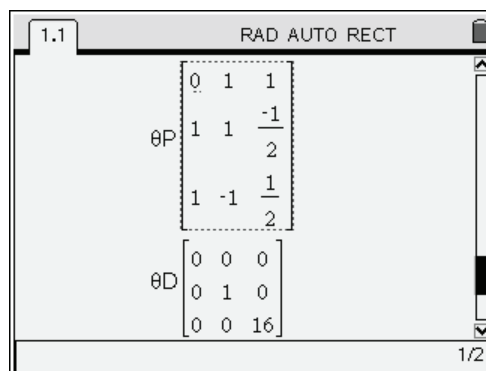
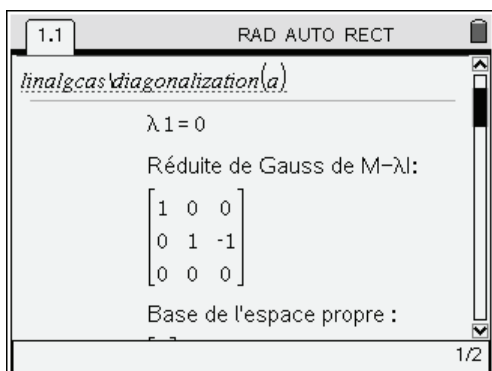
On peut le vérifier en cherchant la dimension des sous-espaces propres correspondants. Pour  $a = -2$  c'est 1 qui est valeur propre double, alors que pour  $a = -3$  c'est 0.



$M$  est diagonalisable pour tout réel  $a \neq -2$ .

#### 4 Équation matricielle

Il est facile, par exemple en utilisant le programme **diagonalization**, de vérifier que la matrice est diagonalisable et que ses valeurs propres sont 0, 1 et 16 (la matrice est symétrique réelle... le fait qu'elle soit diagonalisable n'est donc pas une surprise).



Cette matrice  $A$  ayant trois valeurs propres distinctes en dimension 3, il est facile de montrer que toute matrice qui commute avec  $A$  est elle-même diagonalisable, dans la même base de diagonalisation que  $A$ . (Il suffit de remarquer que chaque espace propre est de dimension 1, et qu'il est stable par toute application linéaire  $g$  commutant avec l'application linéaire  $f$  associée à  $A$ . En particulier, si  $u$  est un vecteur propre associé à  $\lambda$ , alors  $g(u)$  est dans le même espace propre, ce qui entraîne qu'il est colinéaire à  $u$ ... ce qui montre que  $u$  est vecteur propre pour  $g$ ).

Or il est clair que si  $M^2 = A$ , alors  $MA = AM = M^3$ , donc  $M$  commute avec  $A$ , et est donc diagonalisable dans la même base que  $A$ .

En résumé, on a  $A = PDP^{-1}$  et  $M = P\Delta P^{-1}$ . L'équation  $M^2 = A$  est alors équivalente à

$$\Delta^2 = D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 16 \end{bmatrix}.$$

On obtient ainsi  $M_i = P\Delta_i P^{-1}$  avec :

$$\Delta_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{bmatrix}, \Delta_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -4 \end{bmatrix} = -\Delta_1,$$

$$\Delta_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 4 \end{bmatrix}, \Delta_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -4 \end{bmatrix} = -\Delta_3.$$

Il reste à calculer les différentes matrices solutions :

1.1 RAD AUTO RECT Terminé

$m1 := \theta p \cdot \text{diag}(\{0, 1, 4\}) \cdot \theta p^{-1}$

$$\begin{bmatrix} 3 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

$m2 := -m1$

$$\begin{bmatrix} -3 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix}$$

4/99

1.1 RAD AUTO RECT [-1 1 -1]

$m3 := \theta p \cdot \text{diag}(\{0, -1, 4\}) \cdot \theta p^{-1}$

$$\begin{bmatrix} \frac{7}{3} & \frac{-5}{3} & \frac{5}{3} \\ 3 & 3 & 3 \\ -5 & 1 & -1 \\ 3 & 3 & 3 \\ \frac{5}{3} & \frac{-1}{3} & \frac{1}{3} \\ 3 & 3 & 3 \end{bmatrix}$$

5/99

1.1 RAD AUTO RECT [ 3 3 3 ]

$m4 := -m3$

$$\begin{bmatrix} \frac{-7}{3} & \frac{5}{3} & \frac{-5}{3} \\ 3 & 3 & 3 \\ \frac{5}{3} & \frac{-1}{3} & \frac{1}{3} \\ 3 & 3 & 3 \\ \frac{-5}{3} & \frac{1}{3} & \frac{-1}{3} \\ 3 & 3 & 3 \end{bmatrix}$$

6/99