Tutoriel PanaMaths

Calcul matriciel sous Xcas

Cette fiche destinée aux élèves des classes de Terminale et de CPGE requiert un premier niveau de connaissance du logiciel Xcas.

Définition d'un vecteur ou d'une matrice

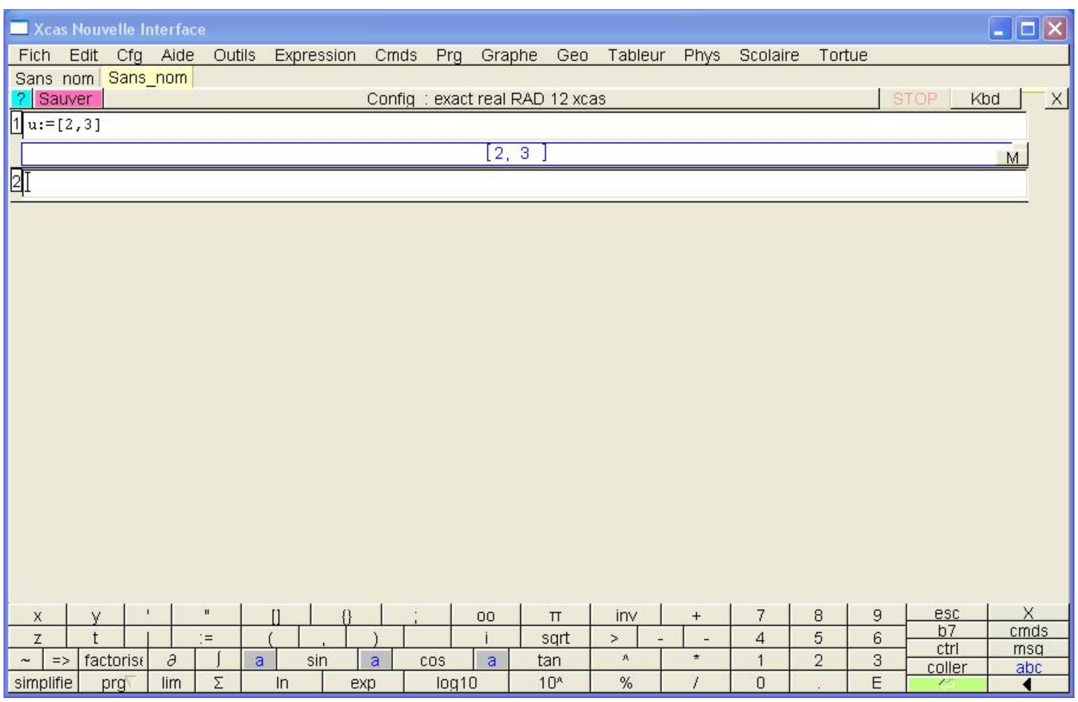
Définir un vecteur

Un vecteur sous Xcas peut-être indifféremment vu comme une matrice ligne ou colonne suivant le contexte (nous y reviendrons plus loin).

Définition d'un vecteur à l'aide de ses coordonnées

On utilise dans ce cas des crochets et on sépare les éléments (fournis en ligne) par des virgules. Par exemple, le vecteur u de coordonnées 2 et 3 sera défini comme suit :

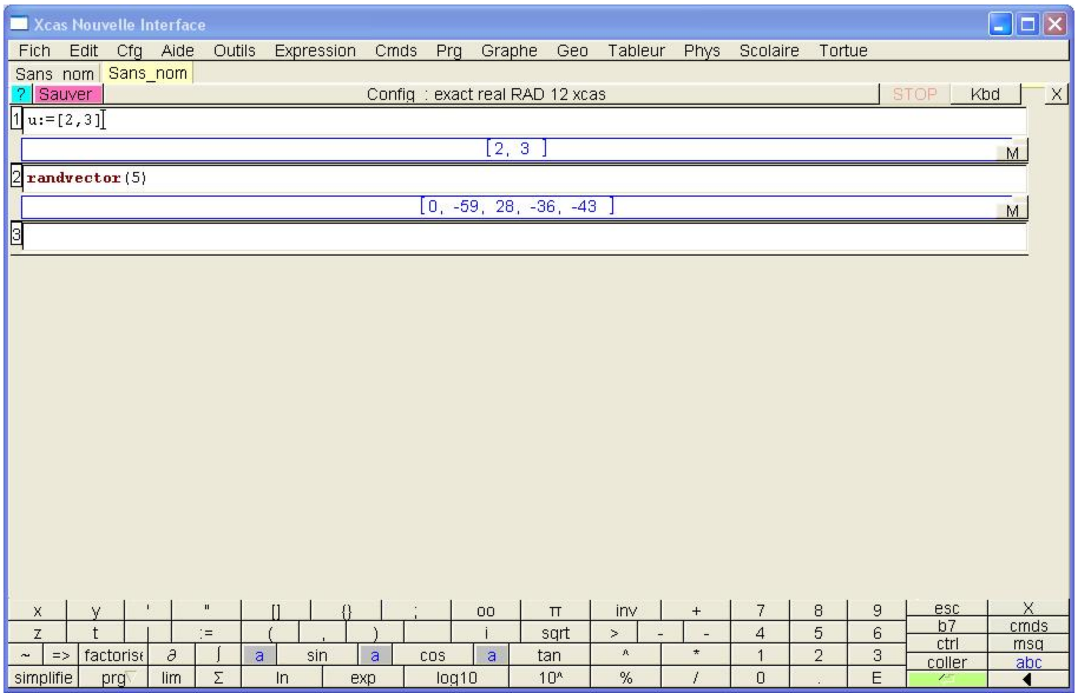
```
u:=[2,3]
```



Définition d'un vecteur aléatoire

On utilise la fonction `randvector` (on peut aussi utiliser la fonction `ranm`, voir plus loin).

Le premier argument de la fonction correspond aux nombre de coordonnées du vecteur. Il est obligatoire. Si on ne précise aucun autre argument, la fonction `randvector` renvoie des coordonnées entières uniformément distribuées dans l'intervalle $[\![-99\,;99]\!]$.

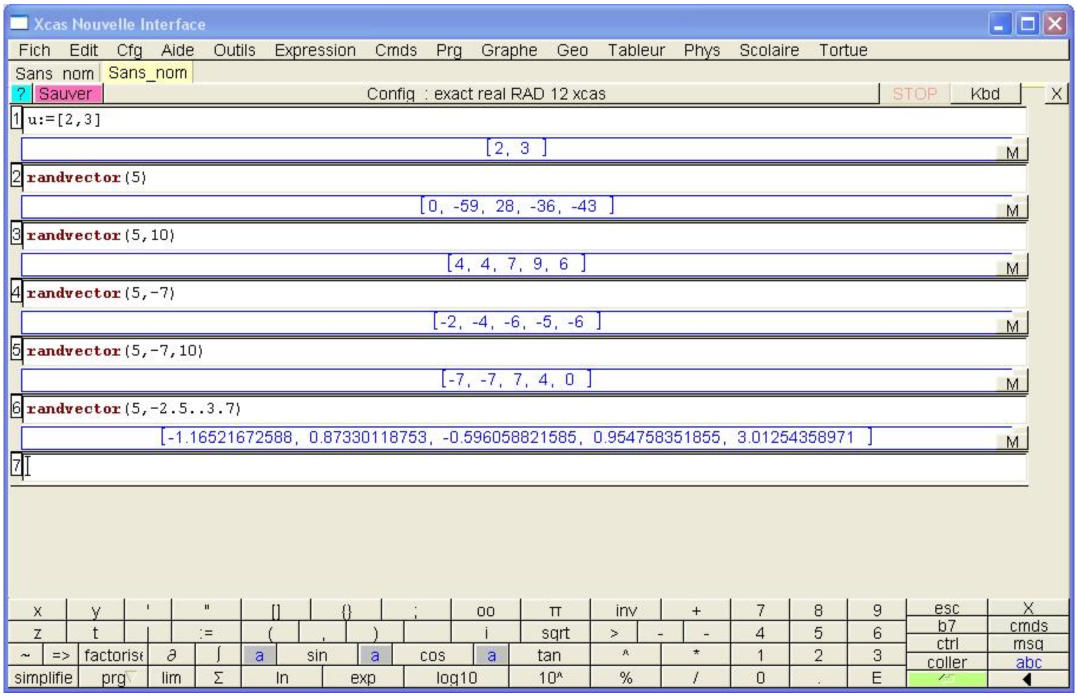


Si on fournit un deuxième argument entier, *k*, la fonction renverra cette fois des coordonnées entières uniformément distribuées dans l'intervalle $[\![0\,;k-1]\!]$ (si *k* est strictement positif) ou dans l'intervalle $[\![k+1\,;0]\!]$ (si *k* est strictement négatif). Si enfin, on fournit un deuxième et un troisième argument, k_1 et k_2, tous deux entiers, on obtient des coordonnées entières uniformément distribuées dans l'intervalle $[\![k_1\,;k_2]\!]$.

Si on souhaite obtenir des coordonnées réelles uniformément distribuées dans l'intervalle $[a\,;b]$, on fournit *a* et *b* sous la forme `a..b` comme deuxième argument. Par exemple, si on souhaite un vecteur aléatoire comportant 5 coordonnées uniformément distribuées dans l'intervalle $[-2,5\,;3,7]$, on appellera la fonction `randvector` comme suit :

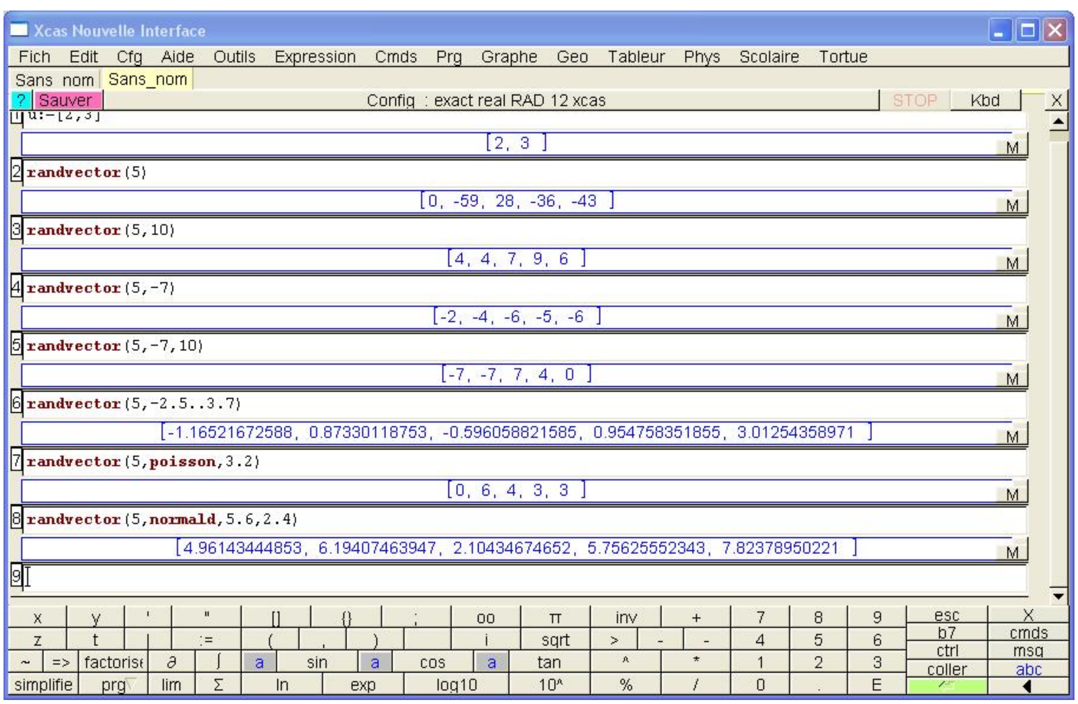
```
randvector(5,-2.5..3.7)
```

La capture d'écran en haut de la page suivante illustre ces différentes situations.



On peut également obtenir des coordonnées comme réalisations d'autres lois de probabilité que la loi uniforme. Par exemple, avec la loi de Poisson de paramètre 3,2 et avec la loi normale de paramètres 5,6 et 2,4 on utilisera respectivement les fonctions `poisson` et `normald` comme arguments de la fonction `randvector` :

```
randvector(5,poisson,3.2)
randvector(5,normald,5.6,2.4)
```



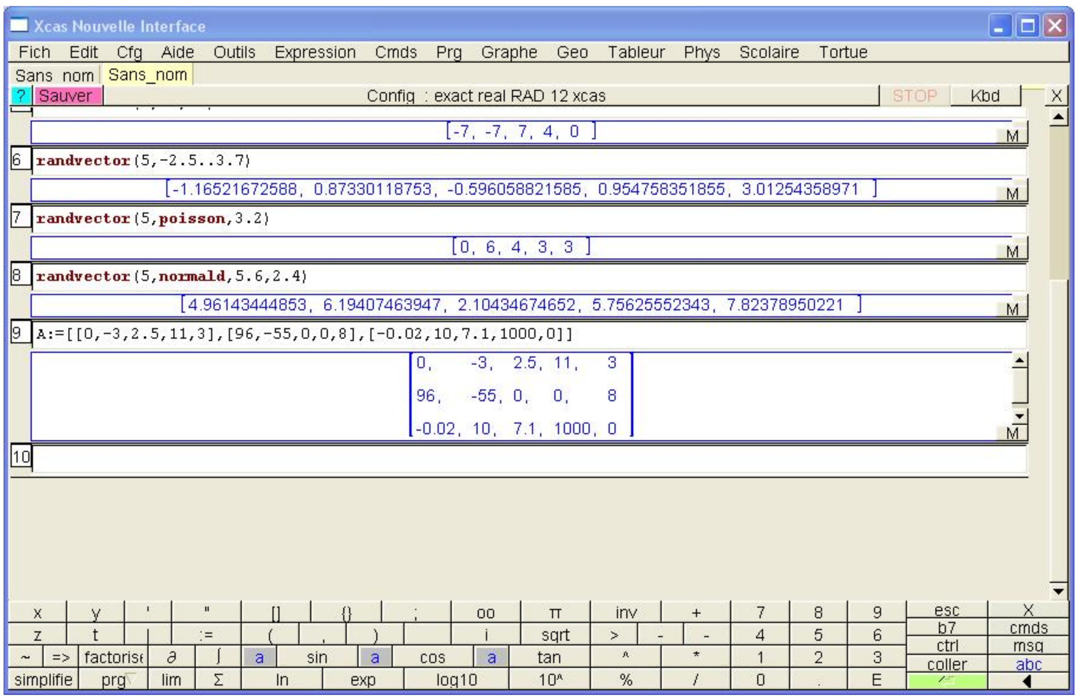
Définir une matrice

Comme pour les vecteurs, on dispose de plusieurs méthodes pour définir des matrices.

Définition d'une matrice à l'aide de ses coefficients

Dans cette méthode, une matrice est vue comme une liste de vecteurs lignes. Ainsi, la matrice $A = \begin{pmatrix} 0 & -3 & 2,5 & 11 & 3 \\ 96 & -55 & 0 & 0 & 8 \\ -0,02 & 10 & 7,1 & 1000 & 0 \end{pmatrix}$ sera saisie comme suit :

```
A:=[[0,-3,2.5,11,3],[96,-55,0,0,8],[-0.02,10,7.1,1000,0]]
```



Matrice dont les coefficients s'expriment en fonction des indices

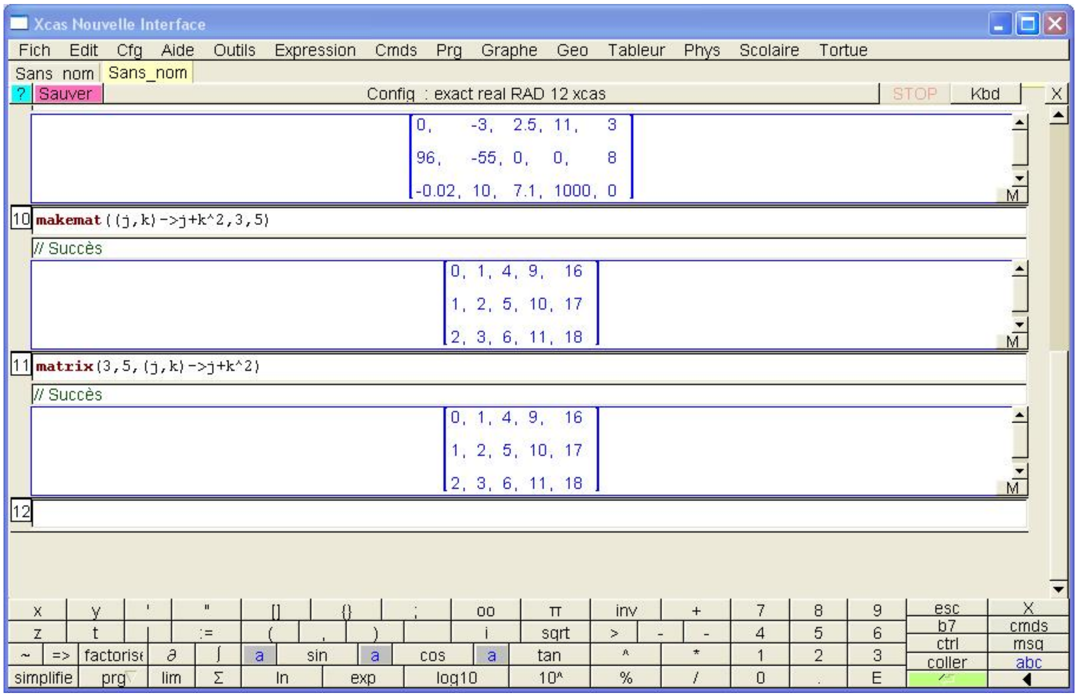
On peut s'intéresse ici à une matrice $A = (a_{ij})$ telle que $a_{ij} = f(i, j)$.

Par exemple : $a_{ij} = i + j^2$.

On dispose de deux fonctions : `makemat` et `matrix`.

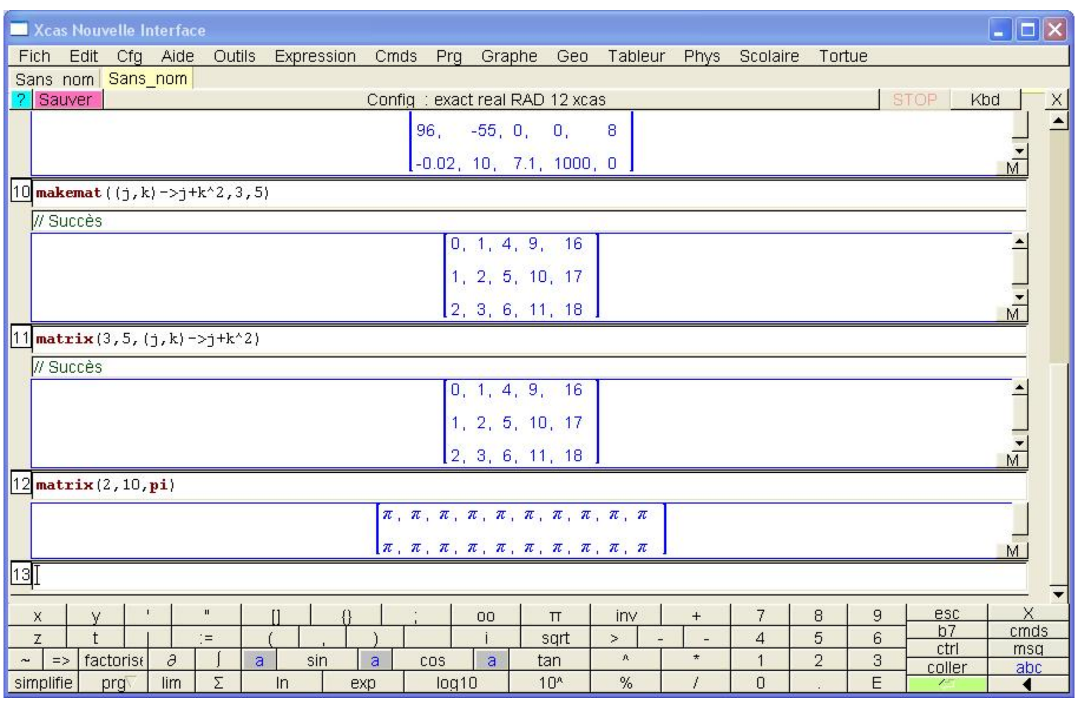
Ces fonctions diffèrent sur l'ordre des arguments : avec `makemat` on précise d'abord la fonction puis les dimensions de la matrice. Avec `matrix`, c'est l'inverse. Dans les deux cas, notons que les indices (des lignes et des colonnes) commencent à 0.

Rappelons qu'avec Xcas la lettre `i` est réservée et ne doit pas être utilisée comme nom de variable dans la définition d'une fonction.



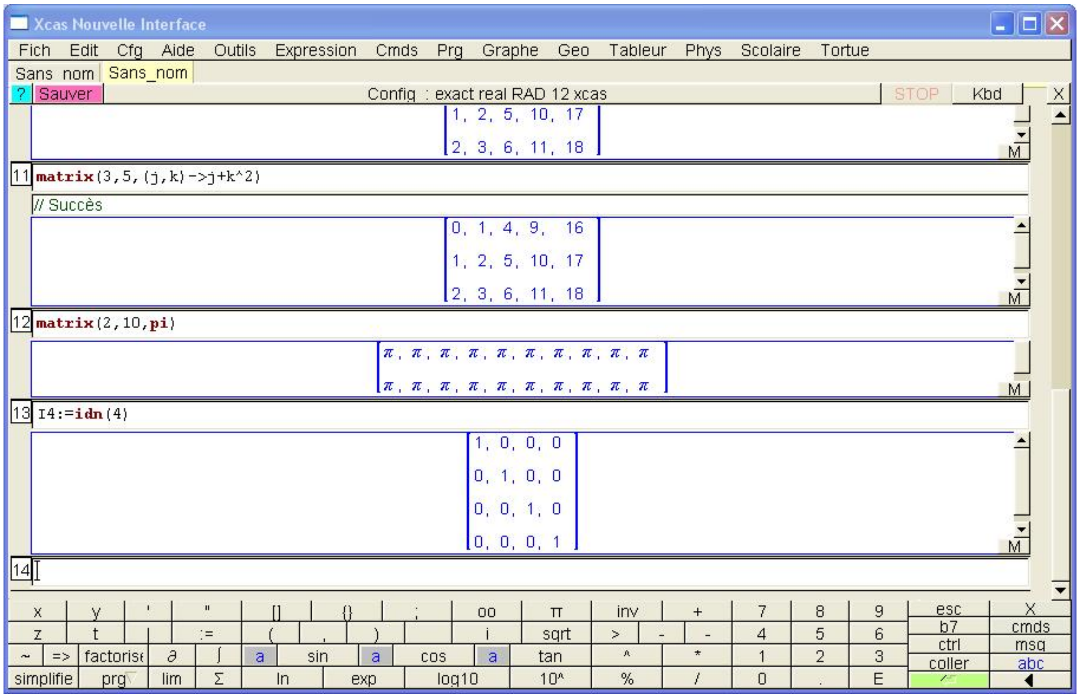
Ces fonctions sont bien sûr très pratiques pour définir des matrices dont tous les coefficients sont égaux. Par exemple, pour construire une matrice de $\mathcal{M}_{2,10}(\mathbb{R})$ dont tous les coefficients sont égaux à π, on utilisera :

```
Matrix(2,10,pi)
```



Matrice identité

On utilise la fonction `idn` qui reçoit comme seul argument l'ordre de la matrice identité souhaitée. Par exemple, I_4 sera obtenue à l'aide de la commande `idn(4)`.
En appelant `I4` cette matrice, on utilise la commande `I4:=idn(4)` et on obtient :



Matrice aléatoire

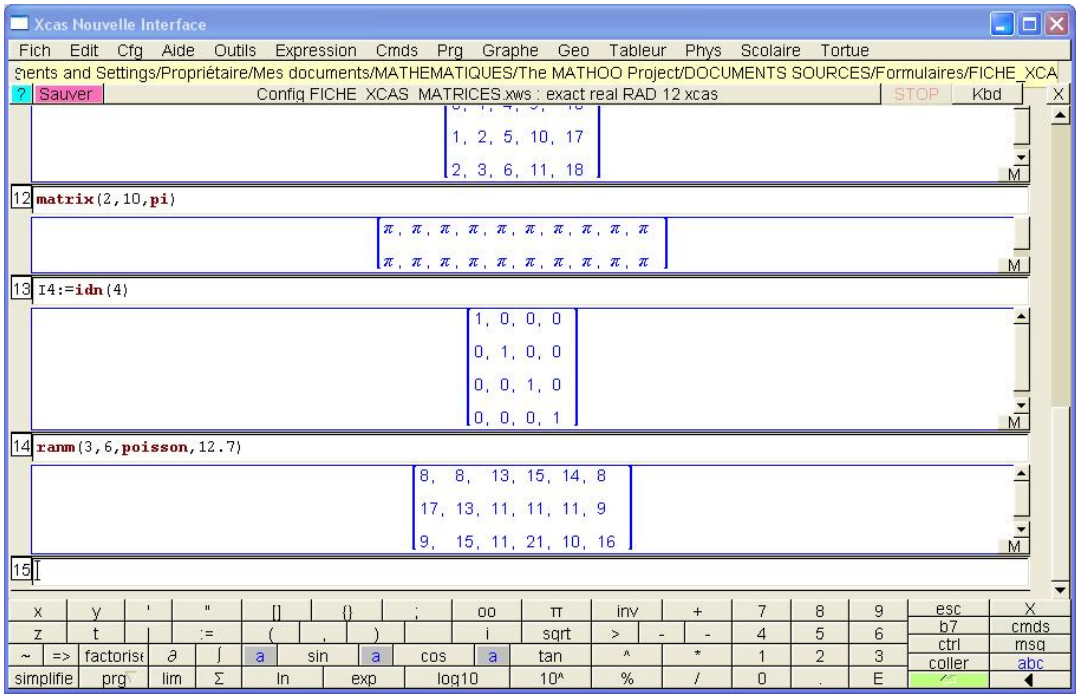
On peut définir des matrices dont les coefficients sont aléatoires à l'aide de la fonction `ranm`. Celle-ci est aux matrices ce que la fonction `randvector` est aux vecteurs. On retrouvera donc les mêmes possibilités relatives aux intervalles et lois de probabilité.

Par exemple, pour définir une matrice de $\mathscr{M}_{3,6}(\mathbb{R})$ dont les coefficients sont des réalisations de la loi de poisson $\mathscr{P}(12,7)$ on saisira dans la ligne de commande (voir la capture d'écran en haut de la page suivante) :

```
ranm(3,6,poisson,12.7)
```

Notons enfin que la fonction `ranm` permet également de générer… des vecteurs aléatoires ! Il suffit, pour cela, de ne fournir qu'une dimension comme premier argument entier de la fonction. Par exemple :

```
ranm(15,poisson,12.7)
```



Matrices par blocs

On peut définir des matrices par blocs grâce à la fonction `blockmatrix`.
Les arguments à fournir sont :

- Deux entiers décrivant la structure des blocs (nombre de lignes et nombre de colonnes).
- La liste (entre crochets) des blocs (i.e. des matrices) à organiser pour construire la nouvelle matrice.

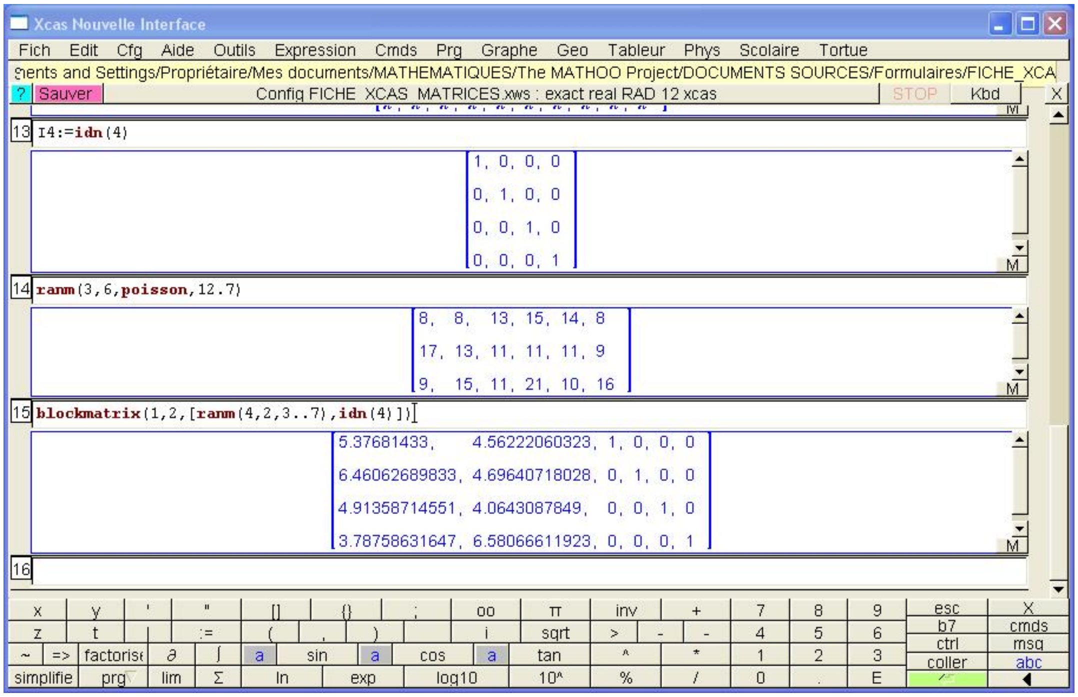
Par exemple :

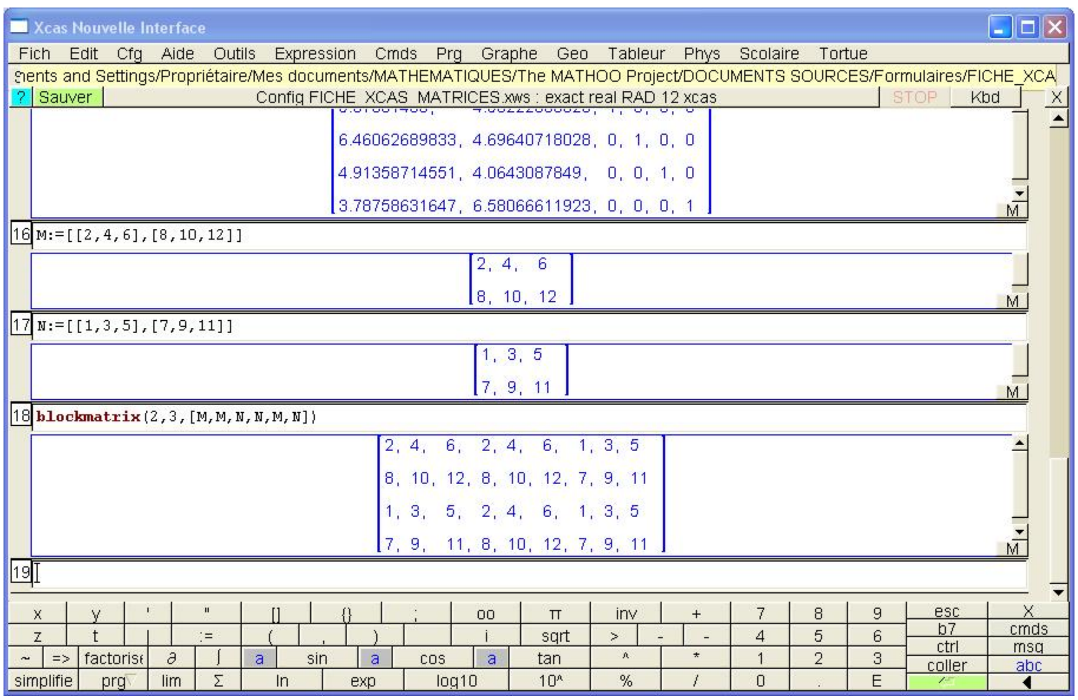
```
blockmatrix(1,2,[ranm(4,2,3..7),idn(4)])
```

Dans cet exemple, les blocs vont être structurés en une ligne et deux colonnes (deux premiers arguments de la fonction). Ces blocs sont respectivement une matrice aléatoire de $\mathscr{M}_{4,2}(\mathbb{R})$ et la matrice identité I_4. On obtient ainsi une matrice de $\mathscr{M}_{4,6}(\mathbb{R})$ (voir la capture d'écran en haut de la page suivante).

Remarque : l'assemblage des blocs se fait ligne par ligne.
On pourra s'intéresser au deuxième exemple suivant :

```
M:=[[2,4,6],[8,10,12]]
N:=[[1,3,5],[7,9,11]]
blockmatrix(2,3,[M,M,N,N,M,N])
```





Accès aux éléments

Rappelons que pour les vecteurs et les matrices, les indices des éléments commencent à 0.

Pour accéder à un élément d'un vecteur (d'une matrice), on utilise le nom du vecteur (de la matrice) suivi de (des) l'indice(s) de l'élément.

Par exemple `M[2,0]` correspond au coefficient de la matrice M situé en troisième ligne et première colonne.
Pour une matrice, on peut aussi fournir entre crochet un seul entier. Dans ce cas, on extraira la ligne correspondante de la matrice. Par exemple, `M[4]` renverra la cinquième ligne de la matrice M.

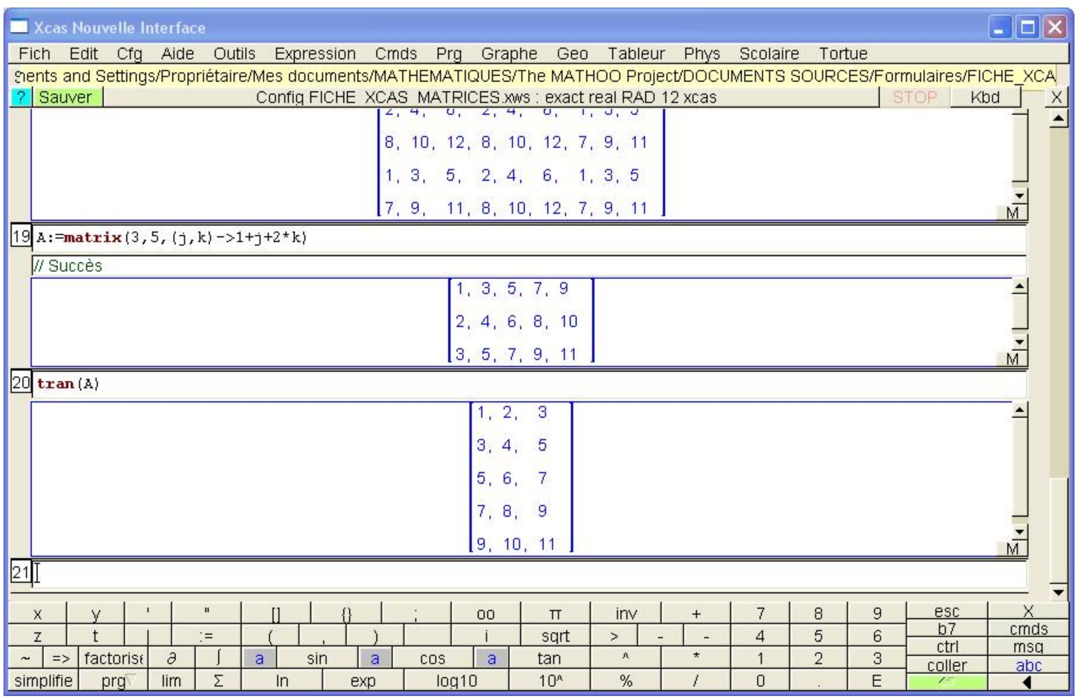
On peut également fournir une(des) plage(s) d'indices afin d'en extraire une sous-matrice. Par exemple, `M[2..4,5..8]` renverra la sous-matrice de M de dimension 3×4 obtenue en ne retenant de M que les coefficients des lignes 2 à 4 et des colonnes 5 à 8.

Transposée, inverse et puissance

Transposée

On utilise la fonction `tran`. Par exemple :

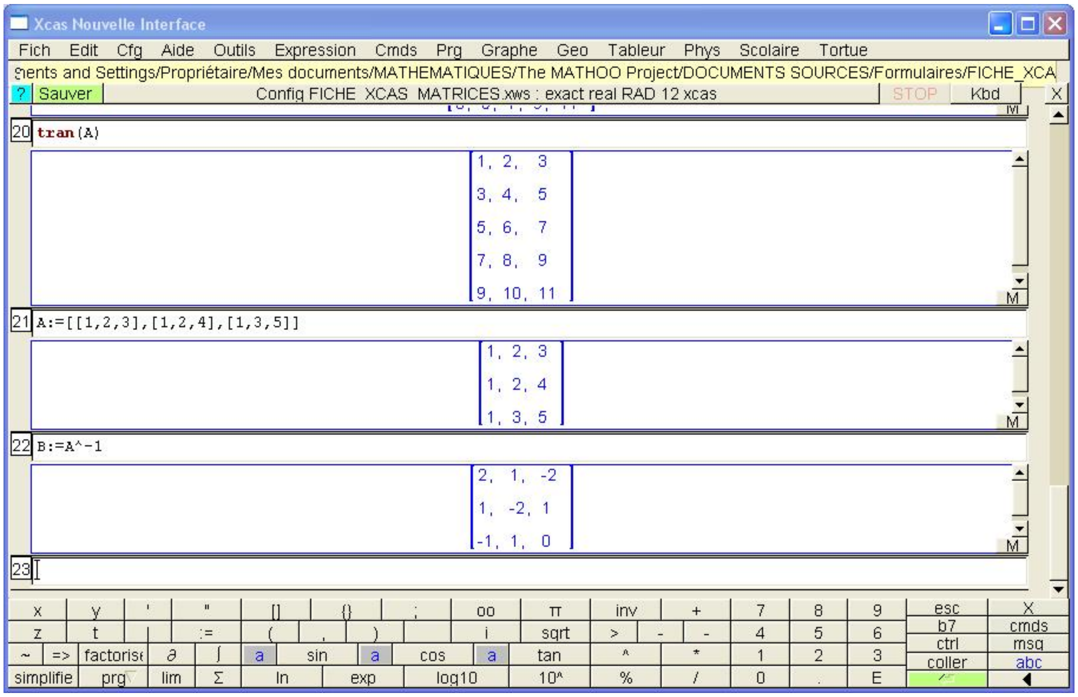
```
A:=matrix(3,5,(j,k)->1+j+2*k)
tran(A)
```



Inverse

Pour obtenir l'inverse d'une matrice carrée inversible, on peut utiliser deux syntaxes que nous illustrons ci-dessous :

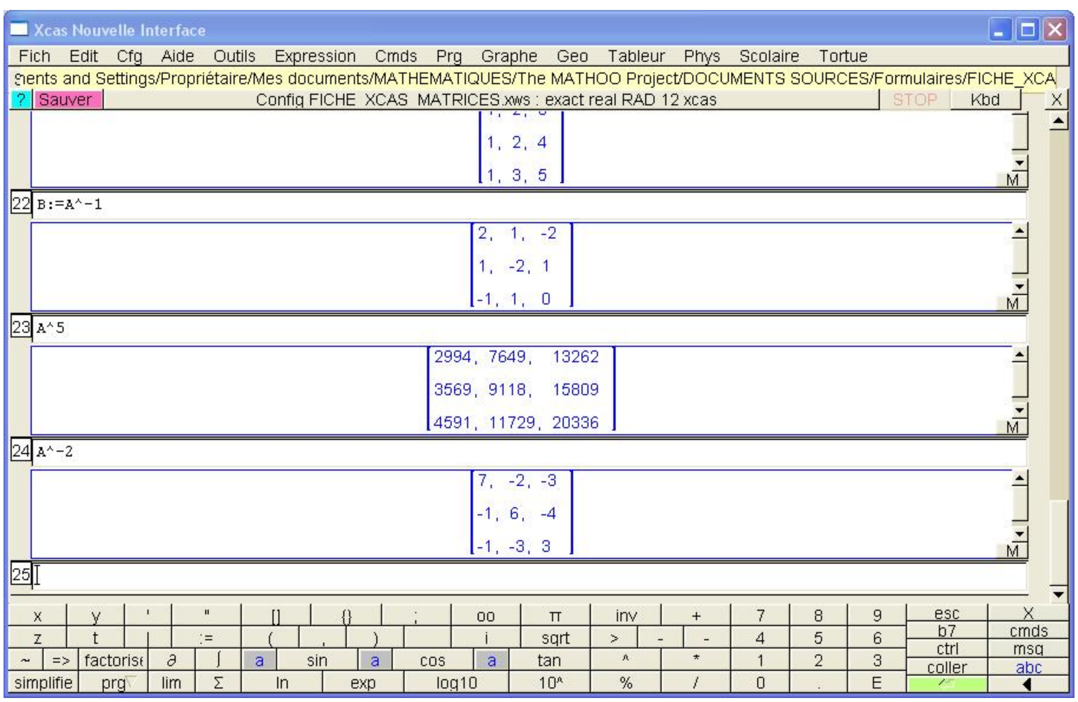
`A:=[[1,2,3],[1,2,4],[1,3,5]]`
`B:=1/A` ou `B:=A^-1`



Puissance

On utilise ici le symbole classique : `^`.
Par exemple, avec la matrice A définie ci-dessus : `A^5`.

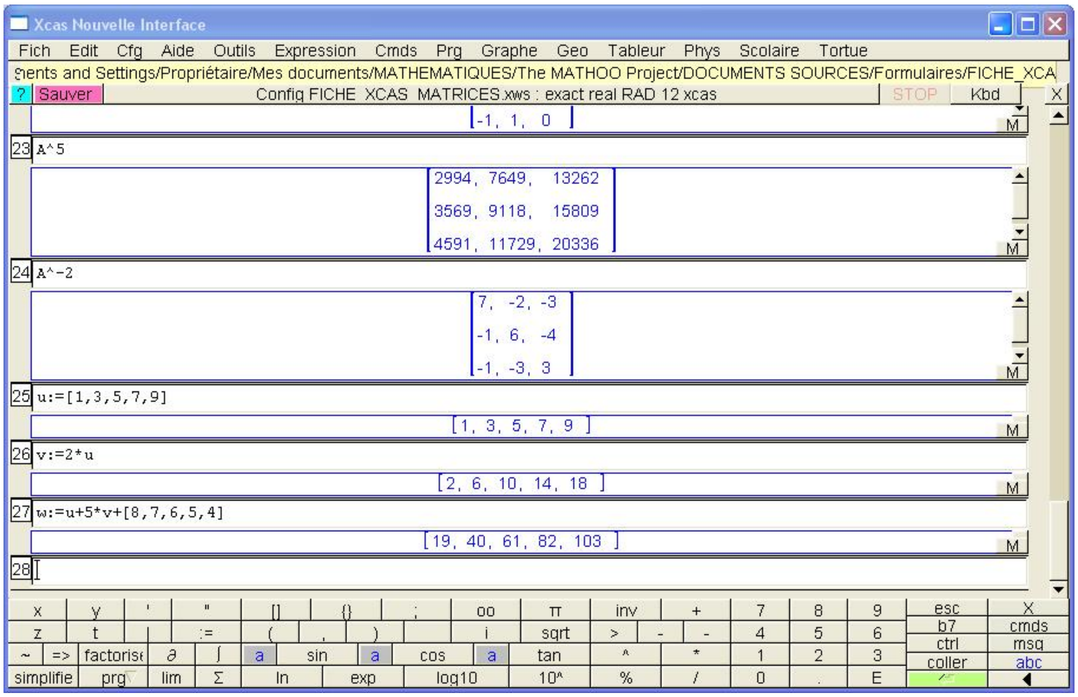
Si la matrice est inversible, on pourra utiliser un exposant négatif : `A^-2`.



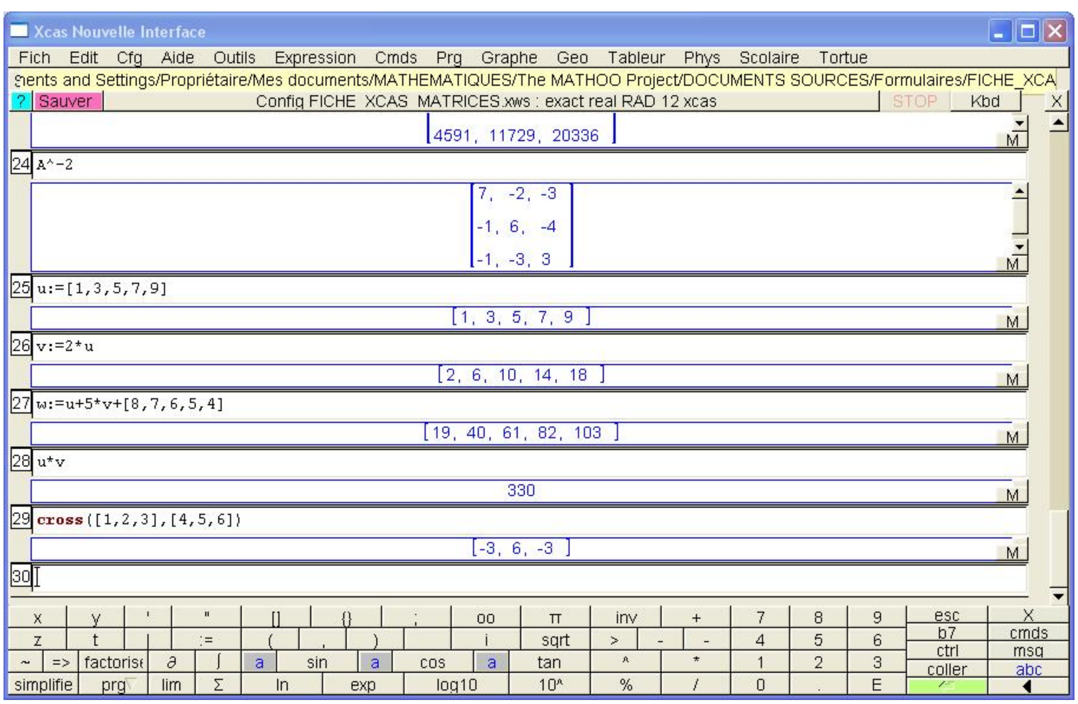
Opérations

Opérations sur les vecteurs

Pour additionner deux vecteurs ou multiplier un vecteur par un scalaire (réel ou complexe) on utilisera les opérateurs classiques : `+` et `*`. Par exemple :



L'opérateur * sert également à effectuer le produit scalaire de deux vecteurs.
Pour le produit vectoriel, on utilisera la fonction `cross`.

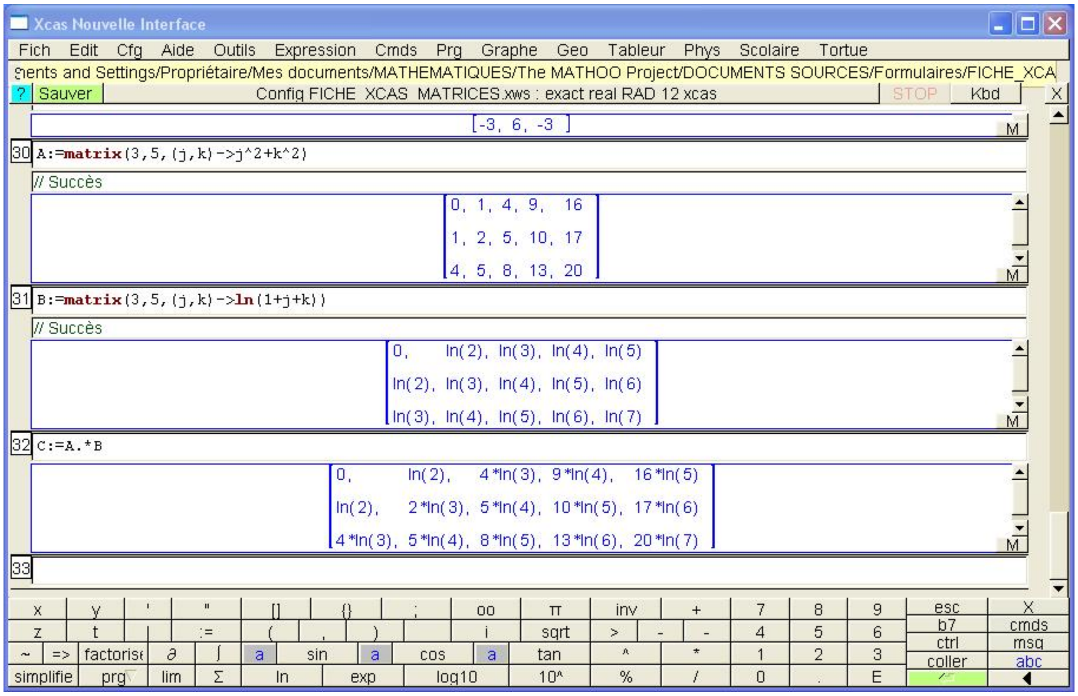


Opérations sur les matrices

Comme pour les vecteurs, on utilisera les opérateurs `+` et `*` pour l'addition des matrices et la multiplication par un scalaire ou une autre matrice.

Le produit d'Hadamard, ou produit terme à terme, est mis en œuvre via la syntaxe `.*` (les matrices doivent bien sûr être de même dimension :

```
A:=matrix(3,5,(j,k)->j^2+k^2)
B:=matrix(3,5,(j,k)->ln(1+j+k))
C:=A.*B
```



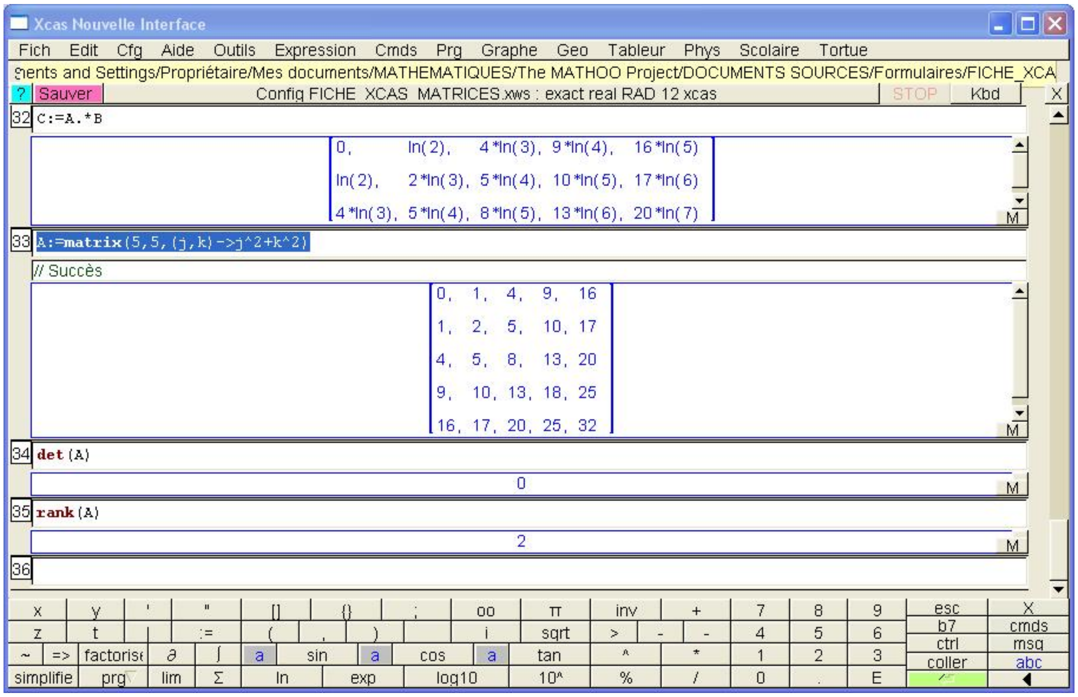
Algèbre linéaire

Déterminant et rang

On utilisera respectivement les fonctions `det` et `rank`. Par exemple :

```
A:=matrix(5,5,(j,k)->j^2+k^2)
det(A)
rank(A)
```

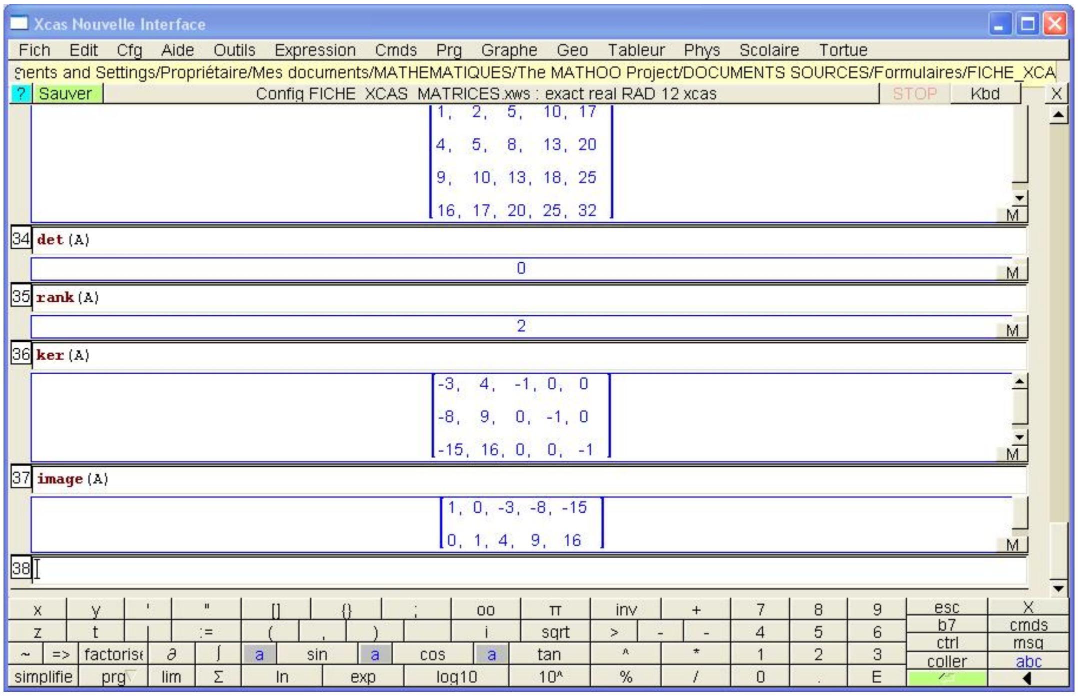
Remarque : dans cet exemple, le déterminant est effectivement nul (n'hésitez pas à le prouver !). Pour autant, on se gardera bien de conclure, lorsque les coefficients comporteront de nombreuses décimales et/ou seront proches de 0, qu'un déterminant numérique ainsi calculé et non nul est exactement égal à 0…



Noyau et image

Les fonctions `ker` et `image` permettent d'obtenir respectivement une base du noyau et de l'image de l'application linéaire canoniquement associée à la matrice considérée. Par exemple, avec la matrice ci-dessus, on utilisera :

```
ker(A)
image(A)
```



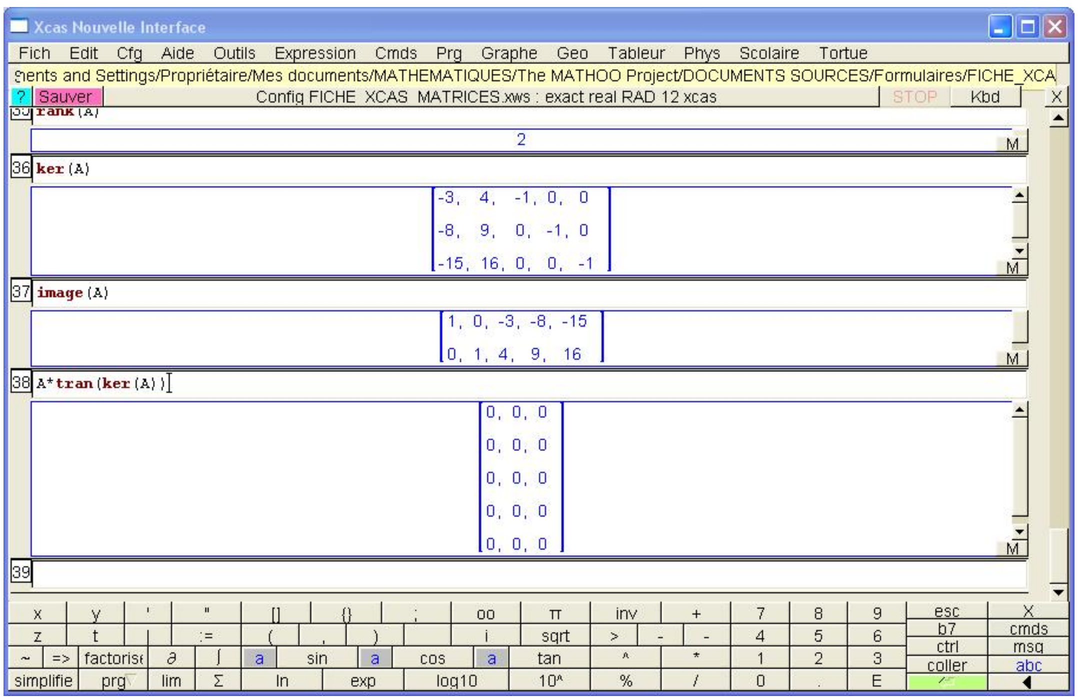
On prend garde au fait que les vecteurs obtenus sont donnés <u>en ligne</u>. Ainsi, les vecteurs suivants forment une base du noyau :

$$\begin{pmatrix} -3 \\ 4 \\ -1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} -8 \\ 9 \\ 0 \\ -1 \\ 0 \end{pmatrix} \text{ et } \begin{pmatrix} -15 \\ 16 \\ 0 \\ 0 \\ -1 \end{pmatrix}$$

On pourra, par exemple, s'en assurer en appliquant successivement A à chacun de ces trois vecteurs grâce aux commandes suivantes :

```
A*ker(A)[0]
A*ker(A)[1]
A*ker(A)[2]
```

Ou, directement, en multipliant A par la transposée de ker(A) :



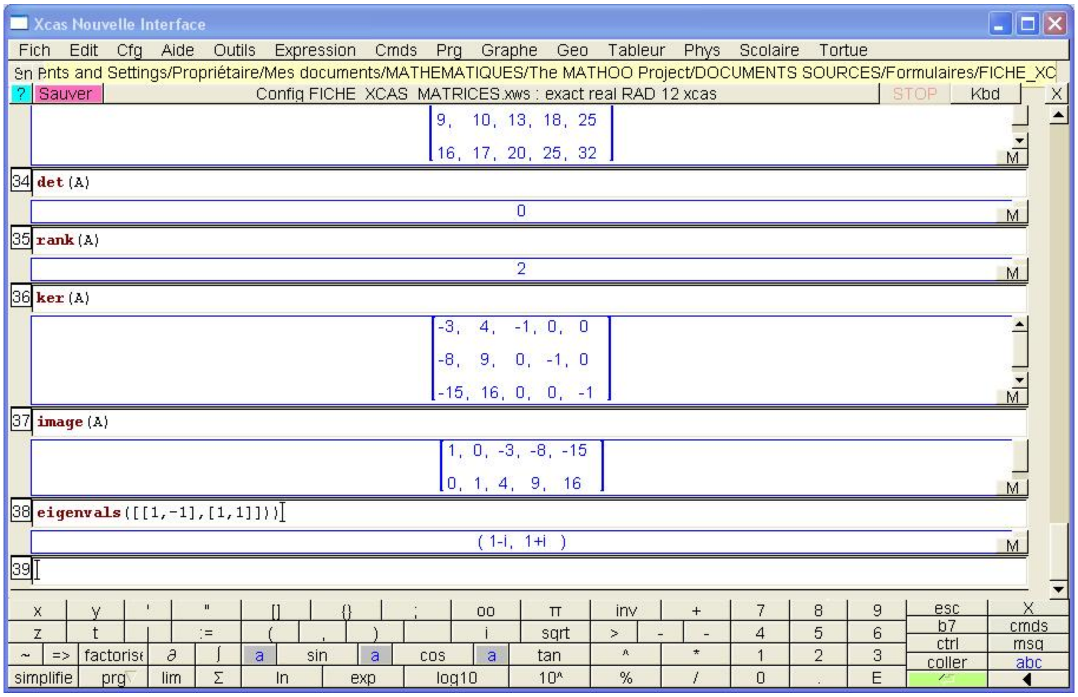
Eléments propres

Les valeurs propres et vecteurs propres associés sont respectivement obtenus à l'aide des fonctions `eigenvals` et `eigenvects`.

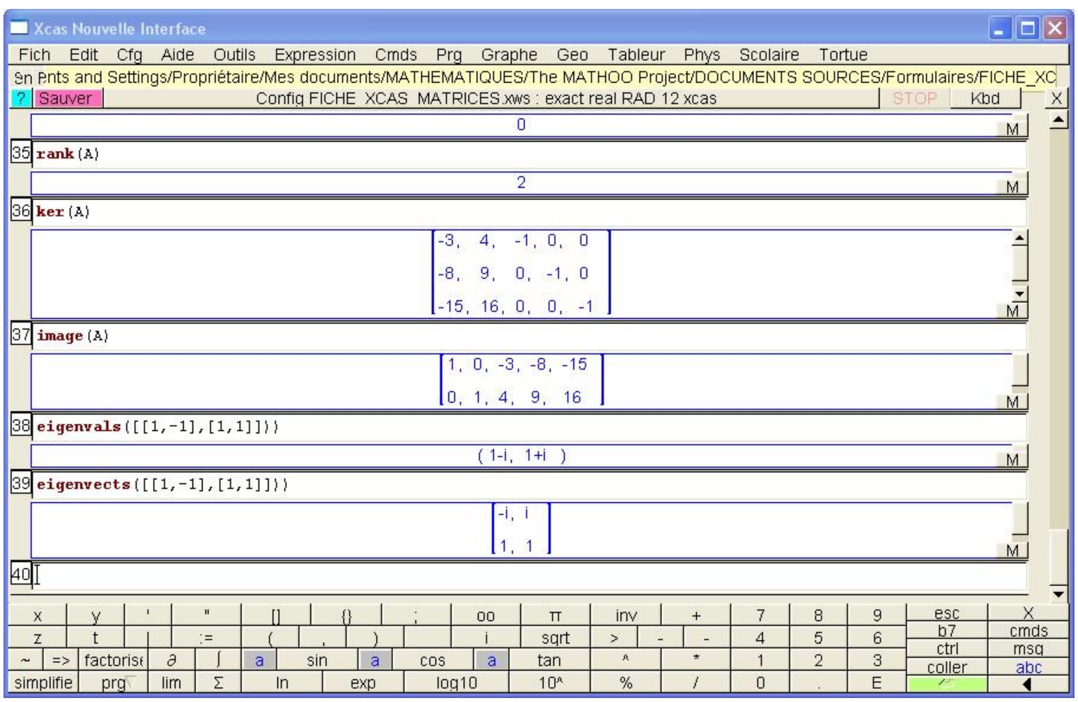
Notons d'emblée que la fonction `eigenvals` renvoie les valeurs propres complexes de la matrices considérée. Par exemple :

```
eigenvals([[1,-1],[1,1]]))
```

Les valeurs sont renvoyées dans une liste.



Quant aux vecteurs propres (complexes) associés, ils sont cette fois retournés <u>en colonne</u> dans une matrice (voir plus loin le paragraphe « Réduction ») :

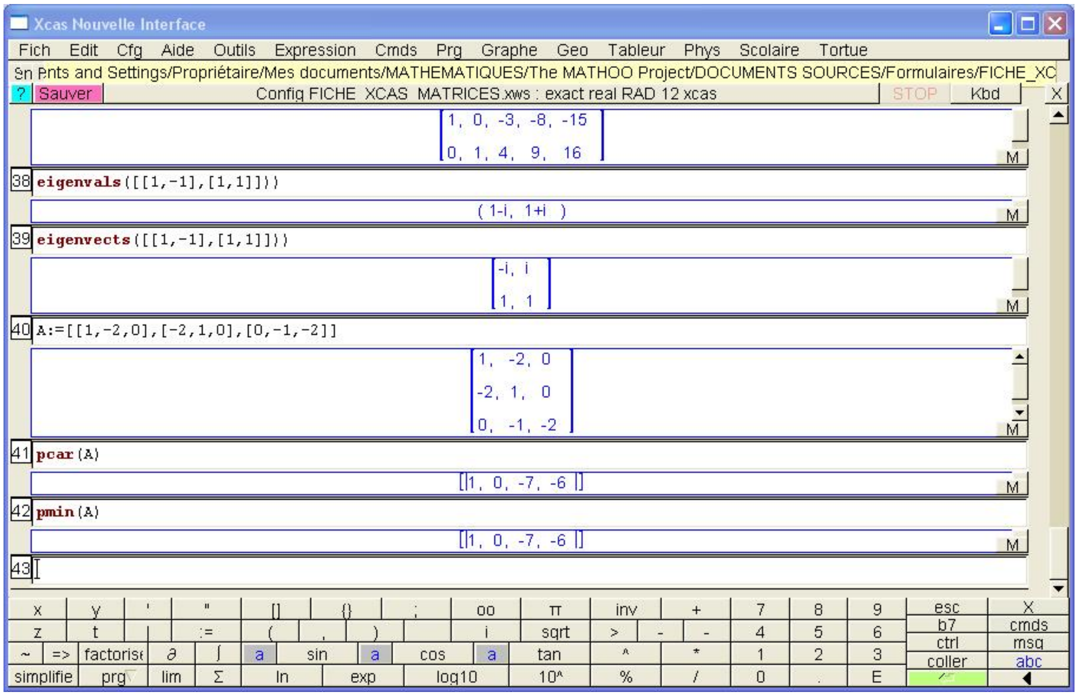


Polynôme caractéristique et polynôme minimal

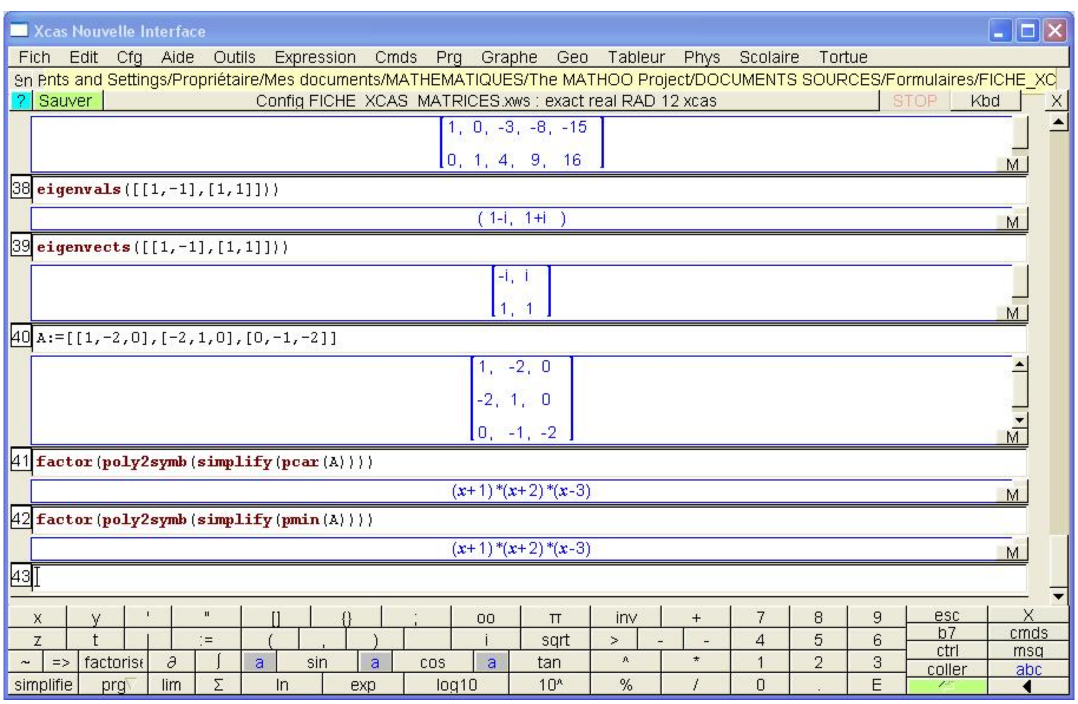
Pour obtenir les polynômes caractéristique et minimal d’une matrice donnée, on utilise les fonctions `pcar` et `pmin`. Ces fonctions renvoient uniquement les coefficients de ces polynômes.

Par exemple :

```
A:=[[1,-2,0],[-2,1,0],[0,-1,-2]]
pcar(A)
pmin(A)
```



Classiquement, si on souhaite un affichage plus lisible (en particulier une forme factorisée, très intéressante dans le cadre de la réduction), on peut utiliser les fonctions `simplify` (ici, vu les coefficients, elle ne joue aucun rôle), `poly2symb` et `factor` :

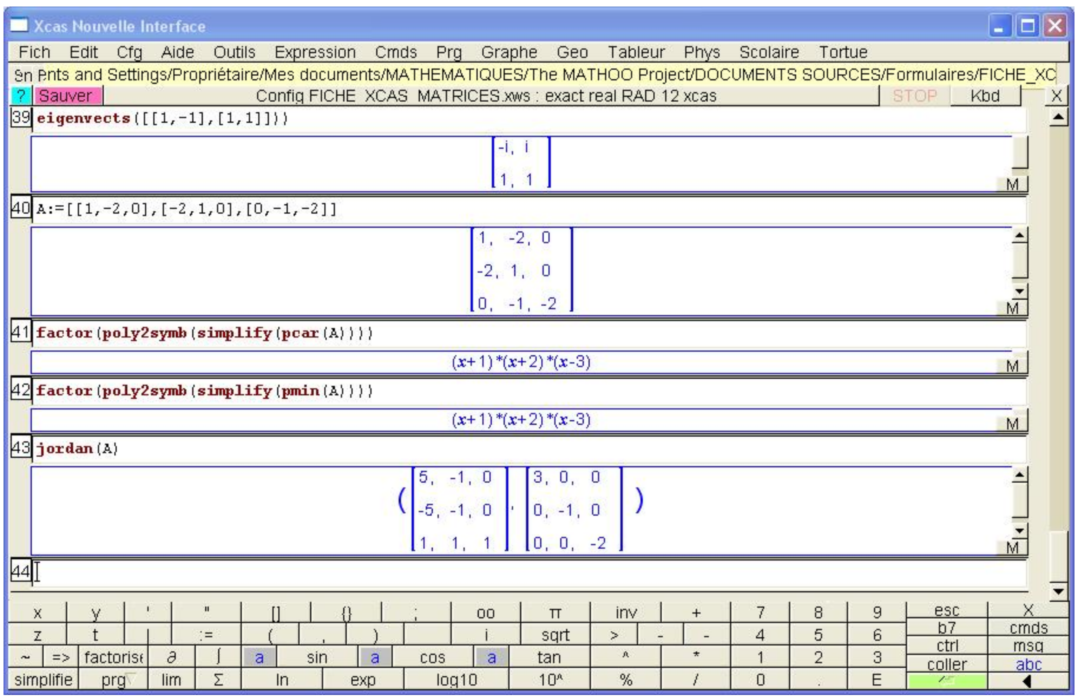


Réduction

Dans la situation précédente, le polynôme caractéristique de la matrice A d'ordre 3 admet 3 racines distinctes. La matrice A est donc diagonalisable. On peut bien sûr, grâce à la fonction `eigenvects` obtenir une base de vecteurs propres. On a vu qu'ils étaient fournis avec leurs coordonnées en colonne. Concrètement, Xcas renvoie donc la matrice de passage P telle que : $A = PDP^{-1}$ où D est matrice diagonale associée aux valeurs propres (dans l'ordre correspondant à celui des colonnes de P).

La fonction `jordan` permet d'obtenir une information complète, en particulier, mais pas seulement, dans le cas des matrices diagonalisables :

```
A:=[[1,-2,0],[-2,1,0],[0,-1,-2]]
jordan(A)
```



On obtient ainsi une liste comportant deux matrices (on peut obtenir chacune d'elles en utilisant `jordan(A)[0]` et `jordan(A)[1]`), la première étant la matrice de passage P et la seconde la matrice diagonale D.