LE PACKAGE LinearAlgebra

Il y a deux façons de définir une matrice : soit par la commande *matrix* qui définit en fait un tableau de dimension deux, donc de type *array*, soit par la commande *Matrix* qui renvoie un objet de type *Matrix*.
Deux packages : linalg (déjà disponible sous Maple5) pour le type *matrix* et LinearAlgebra pour le type *Matrix*.
On renvoie au cours de Maple5, chapitre 11 pour l'étude sommaire du type *matrix*. On s'intéresse ici au package LinearAlgebra.

§1 Les principales commandes

1.1 Création d'une matrice ou d'un vecteur

Les commandes propres à l'algèbre linéaire sont regroupées dans le package linalg.
> with(LinearAlgebra);

Une matrice à n ligne et p colonnes est définie par
> A:= Matrix([[$a_{1,1}$, $a_{1,2}$, … , $a_{1,p}$] , [$a_{2,1}$, … , $a_{2,p}$] , … , [$a_{n,1}$, … , $a_{n,p}$]]); # (liste de listes, chaque liste étant une ligne de la matrice).
> A:= << $a_{1,1}$ | $a_{1,2}$ | … | $a_{1,p}$ > , < $a_{2,1}$ | … | $a_{2,p}$> , … , < $a_{n,1}$ | … | $a_{n,p}$ >> ;
> A:= << $a_{1,1}$, $a_{2,1}$, … , $a_{n,1}$ > | < $a_{1,2}$, … , $a_{n,2}$> | … | < $a_{1,p}$, … , $a_{n,p}$ >> ; # | est le séparateur de colonnes ; on l'utilise pour définir une matrice colonne par colonne.

Un vecteur est défini par
> A:= Vector([a_1 , a_2 , … , a_n]); # on obtient un vecteur colonne
> A:= < a_1 , a_2 , … , a_n >; # on obtient un vecteur colonne
> A:= Vector[row]([a_1 , a_2 , … , a_n]); # on obtient un vecteur ligne
> A:= < a_1 | a_2 | … | a_n >; # on obtient un vecteur ligne

1) A:= Matrix ([[1,2,3] , [2,3,1] , [3,1,2]]) ; # équivalent à > A:= <<1|2|3> , <2|3|1> , <3|1|2>>;
2) A1:=<<*a*1,*b*1,*c*1> | <*a*2,*b*2,*c*2>>; whattype(A1);
3) <A1|<*a*3,*b*3,*c*3>>; # rajoute une colonne à A1
4) whattype(<1,2,0>); whattype(<1|2|0>); whattype(Vector[row]([1,5,0]));
5) DiagonalMatrix([1,2,3]); DiagonalMatrix([1,2,3],4,3); DiagonalMatrix(<1,2,3>);
6) DiagonalMatrix([<<0|1>,<1,0>> , <<–1|–1>,<1|1>>]);
7) A:= Matrix (3,3) ; # crée une matrice vide à trois lignes et trois colonnes (sans initialisation des composantes)
8) A:= Matrix(3,3,5); # matrice carrée de taille3 dont toutes les composantes sont initialisées à 5
9) A:= matrix (3 , 3 , (*i* , *j*) –>*a*^*i*/*b*^*j*) ; # ⚠ les indices ,*i*,*j* débutent à 1 et non 0.
10) IdentityMatrix(*n*); # matrice identité de taille *n*.
11)

1.2 Opérations sur les matrices

Pour afficher le résultat d'un calcul matriciel (matrices ou vecteurs), on doit utiliser la commande **evalm** sauf s'il s'agit d'une commande dédiée.

1) Addition : > evalm(*A* + *B*); Add(*A* , *B*) ; # additionne les matrices *A* et *B*
2) Multiplication par un scalaire : > evalm(*a***A*) ; ScalarMultiply(*A* , *a*) ; # multiplie la matrice *A* par le scalaire *a*

3) Produit : > evalm(*A*&∗*B*) ; Multiply(*A*1 , *A*2 , … , *An*) ; produit des matrices *A* et *B* et *A*1, .., *A*n respectivement, pourvu que les dimensions soient compatibles
   ⚠ la multiplication par un scalaire utilise ∗ et non pas &∗
4) Produit d'une matrice *A* par un vecteur colonne *U* : > MatrixVectorMultilply(*A, U*)
   Le résultat est un vecteur colonne
   Produit d'un vecteur ligne *V* par une matrice *A* : > VectorMatrixMultilply(*V, A*)
   Le résultat est un vecteur ligne
5) Puissances : > MatrixPower(A , *n*) ; evalm(*A*^*n*) ; puissance *n*-ième de la matrice *A* , *n* eut-être non entier
6) Inverse : > MatrixInverse(*A*) ; evalm(*A*^(−1))
7) Transposition : > Transpose(*A*);

⚠ Il faut faire attention qu'une matrice, un vecteur ou une liste n'ont pas le même type sous Maple. On peut être amené à faire des conversions de type pour pouvoir effectuer certains calculs :

```
> B1:= Matrix([[1,0,5] , [−3,0,2]]) : B2:= Matrix([[0,−2,3] , [4,0,1]]): B11:= DeleteColumn(B1,−1) ; # supprime la dernière colonne de B1.
> B1 + B2 ; Multiply(B1,Transpose(B2)); evalm(B1 &∗ Transpose(B2));
> ScalarMultiply(B1,3) ; 3∗B1; B11^7; MatrixInverse(B11);
> L1:= Vector([1,2,3]); convert(L1 , Matrix) ; convertit L1 en matrice : le résultat est une matrice unicolonne
> L2:= [1,2,3] ; convert(L2 , Matrix); on obtient une matrice ligne
```

1.3 Transformations sur une matrice

1) Extraction d'une colonne : > Column(*A* , *k*); extrait la *k*-ième colonne de *A*.
   ⚠ Attention le résultat est un vecteur colonne, pas une matrice
2) Extraction d'une ligne : > Row(A , *k*) ; extrait la *k*-ième ligne de *A*. Le résultat est un vecteur.
   On peut extraire aussi plusieurs colonnes ou plusieurs lignes d'un coup; voir l'aide
3) > RowOperation(*A,*[*i,k*],λ); # réalise l'opération élémentaire $L_i \leftarrow L_i + \lambda L_k$
4) > ColumnOperation(*A,*[*i,k*],λ); # réalise l'opération élémentaire $C_i \leftarrow C_i + \lambda C_k$
   avec l'option *inplace* = *true*, la matrice transformée écrase la matrice *A* de départ.
5) Simplification des termes d'une matrice : > map (simplify, *matrice*) ;

§2 Informations sur une matrice

2.1 ColumnDimension(*A*) = nombre de colonnes de la matrice *A*
RowDimension(*A*) = nombre de lignes de la matrice *A*
Dimension(*A*) = nombre de composantes du vecteur *A* ou bien dimension de la matrice *A* selon les cas.
Determinant(*A*) = déterminant de *A* ; Rank(*A*) = rang de la matrice

2.2 Nullspace(*A*) retourne le noyau de la matrice *A*
ColumnSpace(*A*) retourne une base du sev engendré par les colonnes de la matrice *A*
RowSpace(*A*) retourne une base du sev engendré par les lignes de la matrice *A*
LinearSolve(*A,B*) résout le système linéaire *AX* = *B* où *A* est une matrice et *B* un vecteur ou bien une matrice

etc … taper > with(LinearAlgebra); cliquer sur un nom de commande puis F1 pour obtenir la description et des exemples concernant cette commande.