Physique Numérique (SMP6)

TP6 : Algèbre linéaire (avec Maple)

But du TP6 :
Nous allons utiliser Maple pour faire de l'algèbre linéaire dans R^n : vecteurs, bases, matrices, systèmes linéaires,...

La plupart des commandes de Maple qui permettent de faire de l'algèbre linéaire se trouvent dans la librairie *LinearAlgebra* :

Remarque : il existe aussi la librairie *linalg*, utilisée dans les versions précédentes de Maple, mais qui est obsolète dans cette version (bien qu'existant toujours).

```
> restart: with(LinearAlgebra);
```

[*&x*, *Add*, *Adjoint*, *BackwardSubstitute*, *BandMatrix*, *Basis*, *BezoutMatrix*, *BidiagonalForm*, **(1)**
BilinearForm, *CARE*, *CharacteristicMatrix*, *CharacteristicPolynomial*, *Column*,
ColumnDimension, *ColumnOperation*, *ColumnSpace*, *CompanionMatrix*,
ConditionNumber, *ConstantMatrix*, *ConstantVector*, *Copy*, *CreatePermutation*,
CrossProduct, *DARE*, *DeleteColumn*, *DeleteRow*, *Determinant*, *Diagonal*, *DiagonalMatrix*,
Dimension, *Dimensions*, *DotProduct*, *EigenConditionNumbers*, *Eigenvalues*, *Eigenvectors*,
Equal, *ForwardSubstitute*, *FrobeniusForm*, *GaussianElimination*, *GenerateEquations*,
GenerateMatrix, *Generic*, *GetResultDataType*, *GetResultShape*, *GivensRotationMatrix*,
GramSchmidt, *HankelMatrix*, *HermiteForm*, *HermitianTranspose*, *HessenbergForm*,
HilbertMatrix, *HouseholderMatrix*, *IdentityMatrix*, *IntersectionBasis*, *IsDefinite*,
IsOrthogonal, *IsSimilar*, *IsUnitary*, *JordanBlockMatrix*, *JordanForm*, *KroneckerProduct*,
LA_Main, *LUDecomposition*, *LeastSquares*, *LinearSolve*, *LyapunovSolve*, *Map*, *Map2*,
MatrixAdd, *MatrixExponential*, *MatrixFunction*, *MatrixInverse*, *MatrixMatrixMultiply*,
MatrixNorm, *MatrixPower*, *MatrixScalarMultiply*, *MatrixVectorMultiply*,
MinimalPolynomial, *Minor*, *Modular*, *Multiply*, *NoUserValue*, *Norm*, *Normalize*,
NullSpace, *OuterProductMatrix*, *Permanent*, *Pivot*, *PopovForm*, *QRDecomposition*,

RandomMatrix, *RandomVector*, *Rank*, *RationalCanonicalForm*, *ReducedRowEchelonForm*, *Row*, *RowDimension*, *RowOperation*, *RowSpace*, *ScalarMatrix*, *ScalarMultiply*, *ScalarVector*, *SchurForm*, *SingularValues*, *SmithForm*, *StronglyConnectedBlocks*, *SubMatrix*, *SubVector*, *SumBasis*, *SylvesterMatrix*, *SylvesterSolve*, *ToeplitzMatrix*, *Trace*, *Transpose*, *TridiagonalForm*, *UnitVector*, *VandermondeMatrix*, *VectorAdd*, *VectorAngle*, *VectorMatrixMultiply*, *VectorNorm*, *VectorScalarMultiply*, *ZeroMatrix*, *ZeroVector*, *Zip*]

Pensez à recharger la librairie après chaque *restart* !

1. Les vecteurs

1.1 Construction d'un vecteur

La commande *vector* permet de créer un vecteur. Voici quelques exemples :

```
> Vector([1,4,6]);whattype(%);Vector[row]([5,6,7]);
  whattype(%);
```

$$\begin{bmatrix} 1 \\ 4 \\ 6 \end{bmatrix}$$

$$Vector_{column}$$

$$\begin{bmatrix} 5 & 6 & 7 \end{bmatrix}$$

$$Vector_{row} \qquad \textbf{(2)}$$

```
> Vector(5,i->i^2);
```

$$\begin{bmatrix} 1 \\ 4 \\ 9 \\ 16 \\ 25 \end{bmatrix} \qquad \textbf{(3)}$$

```
> RandomVector(4);
```

$$\begin{bmatrix} 44 \\ 92 \\ -31 \\ 67 \end{bmatrix} \qquad \textbf{(4)}$$

Afficher le contenu d'un vecteur :

```
> v:=Vector([7,8,9,10]);
```

(5)

$$v := \begin{bmatrix} 7 \\ 8 \\ 9 \\ 10 \end{bmatrix} \quad (5)$$

```
> v;
```

$$\begin{bmatrix} 7 \\ 8 \\ 9 \\ 10 \end{bmatrix} \quad (6)$$

Extraire un coefficient d'un vecteur :

```
> v[2]; # deuxième coefficient
```

$$8 \quad (7)$$

Modifier le contenu d'un vecteur
Par affectation des coefficients :

```
> v[1]:=0;v;
```

$$v_1 := 0$$

$$\begin{bmatrix} 0 \\ 8 \\ 9 \\ 10 \end{bmatrix} \quad (8)$$

Copier un vecteur
De subtiles règles d'évaluation dans Maple amènent le problème suivant lorsqu'on souhaite copier un vecteur dans une autre variable :

```
> w:=v;
  v[1]:=2;v;w;
```

$$w := \begin{bmatrix} 0 \\ 8 \\ 9 \\ 10 \end{bmatrix}$$

$$v_1 := 2$$

$$\begin{bmatrix} 2 \\ 8 \\ 9 \\ 10 \end{bmatrix}$$

$$\begin{bmatrix} 2 \\ 8 \\ 9 \\ 10 \end{bmatrix} \quad (9)$$

Toute modification de *v* entraine une modification de sa copie *w* ! Pour y remédier, on utilise la commande *copy* :

```
> w:=Copy(v);
  v[1]:=3;v;w;
```

$$w := \begin{bmatrix} 2 \\ 8 \\ 9 \\ 10 \end{bmatrix}$$

$$v_1 := 3$$

$$\begin{bmatrix} 3 \\ 8 \\ 9 \\ 10 \end{bmatrix}$$

$$\begin{bmatrix} 2 \\ 8 \\ 9 \\ 10 \end{bmatrix} \quad (10)$$

1.2 Quelques opérations sur les vecteurs

Somme de deux vecteurs :

```
> u:=Vector([-7,4,7,3]);
```

$$u := \begin{bmatrix} -7 \\ 4 \\ 7 \\ 3 \end{bmatrix} \quad (11)$$

```
> u+v;
```

$$\begin{bmatrix} -4 \\ 12 \\ 16 \\ 13 \end{bmatrix} \quad (12)$$

Multiplication d'un vecteur par un scalaire (réel) :

```
> v*Pi;
```

$$\begin{bmatrix} 3\,\pi \\ 8\,\pi \\ 9\,\pi \\ 10\,\pi \end{bmatrix} \quad (13)$$

Taille d'un vecteur (nombre de composantes) :

```
> Dimension(u);
```

$$4 \quad (14)$$

Tester une égalité de deux vecteurs :

```
> Equal(u,v);
```

$$false \quad (15)$$

Produit scalaire de deux vecteurs :

```
> Vector([1,1]).Vector([2,1]);
```

$$3 \quad (16)$$

Ou bien :

```
> DotProduct(Vector([1,1]),Vector([2,1]));
```

$$3 \quad (17)$$

Produit vectoriel de deux vecteurs :

```
> CrossProduct(Vector([1,1,0]),Vector([1,0,1]));
```

$$\begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix} \quad (18)$$

1.3 Opérations avancées

Trouver une base du sous-espace vectoriel engendré par un *ensemble* de vecteurs :

```
> u:=Vector([1,0,0]);
  v:=Vector([1,1,0]);
  w:=Vector([1,0,1]);
  t:=Vector([0,0,2]);
```

$$u := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$v := \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

$$w := \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$t := \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \quad (19)$$

```
> Basis( { u,v,w,t } );
```

$$\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right\} \quad (20)$$

2. Les matrices

2.1 Construction d'une matrice

La commande *matrix* permet de créer une matrice. Voici quelques exemples :

```
> Matrix([[1,2,3],[4,5,6],[7,8,9]]);
  whattype(%);
```

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

$$Matrix \quad (21)$$

```
> Matrix(3,3,[1,2,3,4,5,6,7,8,9]);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad (22)$$

```
> Matrix(5,3,(i,j)->i+j);
```

$$\begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \\ 5 & 6 & 7 \\ 6 & 7 & 8 \end{bmatrix} \quad (23)$$

```
> A:=RandomMatrix(2,3);
```

$$A := \begin{bmatrix} -50 & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix} \quad (24)$$

Il existe beaucoup d'autres fonctions permettant de créer des matrices. Consultez les commandes fournies dans *LinearAlgebra* pour en savoir plus : par exemple *DiagonalMatrix*, *BandMatrix*,...

Afficher le contenu d'une matrice

```
> A;
```

$$\begin{bmatrix} -50 & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix} \quad (25)$$

Extraire un coefficient d'une matrice

```
> A[2,3]; # deuxième ligne, troisième colonne
```

$$-18 \quad (26)$$

Modifier d'une matrice

Par affectation des coefficients :

```
> A[1,1]:=Pi;A;
```

$$A_{1,1} := \pi$$

$$\begin{bmatrix} \pi & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix} \quad (27)$$

Copier une matrice

De subtiles règles d'évaluation dans Maple amènent le problème suivant lorsqu'on souhaite copier une matrice dans une autre variable :

```
> B:=A;
  A[1,1]:=1;A;B;
```

$$B := \begin{bmatrix} \pi & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix}$$

$$A_{1,1} := 1$$

$$\begin{bmatrix} 1 & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix} \quad (28)$$

Toute modification de *A* entraine une modification de sa copie *B* ! Pour y remédier, on utilise la commande C*opy* :

```
> B:=Copy(A);
  A[1,1]:=999;A;B;
```

$$B := \begin{bmatrix} 1 & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix}$$

$$A_{1,1} := 999$$

$$\begin{bmatrix} 999 & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 45 & -38 \\ -22 & -81 & -18 \end{bmatrix} \quad (29)$$

2.2 Quelques opérations sur les matrices

Somme de deux matrices :

```
> B:=Matrix(2,3,[1$6]);
```

$$B := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (30)$$

```
> A+B;
```

$$\begin{bmatrix} 1000 & 46 & -37 \\ -21 & -80 & -17 \end{bmatrix} \quad (31)$$

Multiplication d'une matrice par un scalaire (réel) :

```
> A*x;
```

$$\begin{bmatrix} 999\,x & 45\,x & -38\,x \\ -22\,x & -81\,x & -18\,x \end{bmatrix} \quad (32)$$

Multiplication de deux matrices (produit matriciel) :

```
> M:=Matrix(2,2,[a1,b1,c1,d1]);
  N:=Matrix(2,2,[a2,b2,c2,d2]);
```

$$M := \begin{bmatrix} a1 & b1 \\ c1 & d1 \end{bmatrix}$$

$$N := \begin{bmatrix} a2 & b2 \\ c2 & d2 \end{bmatrix} \quad (33)$$

```
> M.N;
```

$$\begin{bmatrix} a1\,a2 + b1\,c2 & a1\,b2 + b1\,d2 \\ c1\,a2 + d1\,c2 & c1\,b2 + d1\,d2 \end{bmatrix} \quad (34)$$

Multiplication d'une matrice par un vecteur :

```
> A.Vector([1,0,0]);
```

$$\begin{bmatrix} 999 \\ -22 \end{bmatrix} \quad (35)$$

Taille d'une matrice : nombre de lignes, nombres de colonnes

```
> RowDimension(A) ; ColumnDimension(A);
```

$$2$$

(36)

$$3 \tag{36}$$

Tester une égalité de deux matrices :

```
> Equal(A,B);
```

$$false \tag{37}$$

```
> A := RandomMatrix(4, 4);DeleteRow(A, 1);
```

$$A := \begin{bmatrix} -61 & 31 & 25 & 50 \\ -48 & -50 & 94 & 10 \\ 77 & -80 & 12 & -16 \\ 9 & 43 & -2 & -9 \end{bmatrix}$$

$$\begin{bmatrix} -48 & -50 & 94 & 10 \\ 77 & -80 & 12 & -16 \\ 9 & 43 & -2 & -9 \end{bmatrix} \tag{38}$$

```
> DeleteColumn(A, 2 .. 3);
```

$$\begin{bmatrix} -61 & 50 \\ -48 & 10 \\ 77 & -16 \\ 9 & -9 \end{bmatrix}$$

```
> ColumnOperation(A, [1, 2]);
```

$$\begin{bmatrix} 31 & -61 & 25 & 50 \\ -50 & -48 & 94 & 10 \\ -80 & 77 & 12 & -16 \\ 43 & 9 & -2 & -9 \end{bmatrix} \tag{39}$$

```
> RowOperation(A, [3, 4]);
```

$$\begin{bmatrix} -61 & 31 & 25 & 50 \\ -48 & -50 & 94 & 10 \\ 9 & 43 & -2 & -9 \\ 77 & -80 & 12 & -16 \end{bmatrix} \tag{40}$$

```
> A := Matrix(3, 3, [1, 2, 3, 4, 5, 6, 7, 8, 9]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \tag{41}$$

```
> RowOperation(A, [1, 2], 1);
```

$$\begin{bmatrix} 5 & 7 & 9 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad (42)$$

```
> RowOperation(A, [1, 2], -2);
```

$$\begin{bmatrix} -7 & -8 & -9 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad (43)$$

```
> ColumnOperation(A, [1, 3], -2);
```

$$\begin{bmatrix} -5 & 2 & 3 \\ -8 & 5 & 6 \\ -11 & 8 & 9 \end{bmatrix} \quad (44)$$

```
> RowOperation(A, 2, -1);
```

$$\begin{bmatrix} 1 & 2 & 3 \\ -4 & -5 & -6 \\ 7 & 8 & 9 \end{bmatrix} \quad (45)$$

```
> ColumnOperation(A, 1, 1/100);
```

$$\begin{bmatrix} \frac{1}{100} & 2 & 3 \\ \frac{1}{25} & 5 & 6 \\ \frac{7}{100} & 8 & 9 \end{bmatrix} \quad (46)$$

2.3 Opérations avancées

Trace d'une matrice (somme des coefficients de la première diagonale) :

```
> Trace(M);
```

$$a1 + d1 \quad (47)$$

Transposée d'une matrice :

```
> Transpose(A);
```

$$\begin{bmatrix} 999 & -76 \\ -72 & -2 \\ -32 & -74 \end{bmatrix} \quad (48)$$

Calcul de l'inverse d'une matrice (carrée inversible) :

```
> A:=Matrix(3,3,[7/10,3/5,1/10,-3/5,11/5,1/5,-3/10,3/5,11/10]):
```

```
MatrixInverse(A);
```

$$\begin{bmatrix} \frac{23}{20} & -\frac{3}{10} & -\frac{1}{20} \\ \frac{3}{10} & \frac{2}{5} & -\frac{1}{10} \\ \frac{3}{20} & -\frac{3}{10} & \frac{19}{20} \end{bmatrix} \quad (49)$$

```
> B := 1/A;
```

$$B := \begin{bmatrix} \frac{23}{20} & -\frac{3}{10} & -\frac{1}{20} \\ \frac{3}{10} & \frac{2}{5} & -\frac{1}{10} \\ \frac{3}{20} & -\frac{3}{10} & \frac{19}{20} \end{bmatrix} \quad (50)$$

Calcul des valeurs propres d'une matrice carrée :

```
> Eigenvalues(A);
```

$$\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix} \quad (51)$$

Calcul des vecteurs propres d'une matrice carrée :

```
> Eigenvectors(A);
```

$$\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 & 2 & \frac{1}{3} \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (52)$$

Maple renvoie ici un premier vecteur constitué des valeurs propres de A, suivi d'une matrice dont les vecteurs colonnes sont les vecteurs propres correspondants.

Trouver une base du noyau de l'application linéaire associée à une matrice :

```
> NullSpace(A);
```

$$\{\ \} \quad (53)$$

```
> B:=Matrix(2,2,[1,2,1/2,1]);
```

$$B := \begin{bmatrix} 1 & 2 \\ \frac{1}{2} & 1 \end{bmatrix} \quad (54)$$

```
> NullSpace(B);
```

(55)

$$\left\{\left[\begin{array}{c} -2 \\ 1 \end{array}\right]\right\} \quad (55)$$

Que signifient les réponses de Maple ?

Trouver une base de l'image de l'application linéaire associée à une matrice :
(= une base du sous-espace vectoriel engendré par les vecteurs colonnes de la matrice)

```
> ColumnSpace(B);
```

$$\left[\left[\begin{array}{c} 1 \\ \frac{1}{2} \end{array}\right]\right] \quad (56)$$

Rang de la matrice (= dimension de l'image)

```
> Rank(B);
```

$$1 \quad (57)$$

Déterminant d'une matrice carrée (on rappelle qu'une matrice carrée est inversible si et seulement si son déterminant est non nul) :

```
> Determinant(A);
```

$$2 \quad (58)$$

3. Les systèmes linéaires

3.1 Systèmes linéaires donnés par des équations

On utilise *solve*. Par exemple, pour le système :

$$\begin{cases} x+y+z=3 \\ z+2y-z=-1 \\ -x+y+2z=0 \end{cases}$$

```
> solve( {x+y+z=3 , x+2*y-z=-1 , -x+y+2*z=0} , {x,y,z} );
```

$$\left\{x=\frac{16}{7}, y=-\frac{6}{7}, z=\frac{11}{7}\right\} \quad (59)$$

3.2 Systèmes linéaires donnés par une matrice

On utilise *LinearSolve*. Par exemple, le système précédent correspond à A.X=B, où A est la matrice :

```
> A:=Matrix( [ [1,1,1], [1,2,-1], [-1,1,2] ] );
```

$$A := \left[\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 2 & -1 \\ -1 & 1 & 2 \end{array}\right] \quad (60)$$

et B le vecteur colonne :

```
> B:=Vector([3,-1,0]);
```

$$B := \left[\begin{array}{c} 3 \\ -1 \\ 0 \end{array}\right] \quad (61)$$

On résout le système par :

```
> LinearSolve(A,B);
```

$$\begin{bmatrix} \frac{16}{7} \\ -\frac{6}{7} \\ \frac{11}{7} \end{bmatrix} \quad (62)$$

3.3 Le pivot de Gauss

Maple peut effectuer le pivot de Gauss sur une matrice avec la commande *GaussianElimination*. Le résultat est une matrice échelonnée mais non réduite.

```
> C:=Matrix(A);
```

$$C := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & -1 \\ -1 & 1 & 2 \end{bmatrix} \quad (63)$$

```
> C:=GaussianElimination(A);
```

$$C := \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -2 \\ 0 & 0 & 7 \end{bmatrix} \quad (64)$$

Enfin, la commande *BackSubstitution*, appliquée à la matrice échelonnée, permet de trouver le(s) solution(s) :

```
> with(Student[NumericalAnalysis]):BackSubstitution(C,B);
```

$$\begin{bmatrix} 4 \\ -1 \\ 0 \end{bmatrix} \quad (65)$$

On a ici besoin d'une autre librairie servant au calcul numérique matriciel .