

# Espaces euclidiens, espaces hermitiens

Nous utiliserons constamment le package `linalg`, il est donc plus simple de la charger une bonne fois pour toute.

```
> with(linalg):
Warning, new definition for norm
Warning, new definition for trace
```

## Formes quadratiques

### Formes bilinéaires et formes sesquilinéaires

Maple ne connaît au départ qu'une seule forme bilinéaire sur le corps des nombres réels; sur le corps des nombres complexes il connaît une forme bilinéaire et une forme sesquilinéaire. Dans tous les cas, la fonction utilisée est la fonction **dotprod**, sans option pour la forme sesquilinéaire, avec l'option `quadratic` pour la forme bilinéaire (pour des vecteurs à coefficients réels, cela ne fait évidemment pas de différence). Attention: contrairement à une convention qui tend à se généraliser, le produit est linéaire à gauche et semi-linéaire à droite.

```
> dotprod([x1,x2,x3],[1,2,I]);
x1 + 2 x2 - I x3
> dotprod([x1,x2,x3],[1,2,I],orthogonal);
x1 + 2 x2 + I x3
```

Remarquons une totale incohérence puisque sur des vecteurs symboliques, Maple utilise la version bilinéaire sur les symboles et sesquilinéaire sur les nombres complexes! Ceci est lié au comportement déficient de la fonction Maple **evalc** (évaluation en complexe) qui considère par défaut que les objets inconnus sont des réels!

```
> dotprod([x1,x2,x3],[t,2,I]);
x1 t + 2 x2 - I x3
```

Nous recommandons donc fortement, dans le cas des nombres complexes, de se passer de cette fonction imprévisible et de la redéfinir chaque fois que cela sera nécessaire.

Bien entendu, vous pouvez définir par vous même d'autres formes bilinéaires ou sesquilinéaires. Voici par exemple un produit scalaire classique sur les fonctions de la variable  $t$  qui sont continues sur  $[0, 1]$  (la présence de la fonction **evalc** est nécessaire, sinon Maple n'évalue pas le conjugué et ne sait donc pas calculer l'intégrale)

```
> fct_scal:=(f,g)-> int(evalc(conjugate(f))*g,t=0..1);
fct_scal := (f, g)  $\int_0^1 \text{evalc}(\bar{f}) g \, dt$ 
```

Cependant le résultat n'est pas toujours correct, car la fonction **evalc** suppose implicitement que les objets dont elle ne connaît pas la nature sont des réels:

```
> fct_scal(f(t),g(t));
 $\int_0^1 f(t) g(t) \, dt$ 
```

### Formes quadratiques

### Bases orthonormées

La fonction prédéfinie **GramSchmidt** n'utilise que le produit scalaire prédéfini **dotprod**. Elle prend en argument une base de  $K^n$  (où  $K$  est soit le corps des nombres réels, soit le corps des nombres complexes), entrée comme liste de vecteurs, et retourne une base orthogonale pour le produit scalaire **dotprod**

```
> GramSchmidt([vector([1,0,1]),vector([-1,1,2]),vector([1,1,3])]);
```

$[1, 0, 1], \frac{-3}{2}, 1, \frac{3}{2}, \frac{1}{11}, \frac{3}{11}, \frac{-1}{11}$

```
> GramSchmidt([vector([1,0,I]),vector([-1,I,2]),vector([1,1,
```

```
3*I]]]);
```

$$\left[1, 0, I\right], -\frac{1}{2} + I, I, 1 + \frac{1}{2}I, -\frac{4}{7} - \frac{1}{7}I, \frac{9}{7} - \frac{2}{7}I, -\frac{1}{7} + \frac{4}{7}I$$

On pourra redéfinir une telle fonction pour des produits scalaires plus *exotiques*, par exemple

```
> GS:=proc(base,scal) local n,i,j,ortho,v;
    n:=nops(base);
    ortho:=[base[1]/sqrt(scal(base[1],base[1]))];
    for i from 2 to n do
        v:=base[i];
        for j from 1 to i-1 do
            v:=v-scal(base[i],ortho[j])*ortho[j]
        od;
        ortho:=[op(ortho),v/sqrt(scal(v,v))];
    od;
    ortho
end;
```

Avec une application aux polynômes de Legendre (qui forment une base orthonormée pour le produit scalaire  $(f, g) = \int_{-1}^1 f(t) g(t) dt$ )

```
> scal_Legendre:=(f,g)->int(f*g,t=-1..1):
GS([1,t,t^2,t^3,t^4],scal_Legendre);
```

$$\frac{1}{2}\sqrt{2}, \frac{1}{2}t\sqrt{6}, \frac{3}{4}t^2 - \frac{1}{3}, \sqrt{10}, \frac{5}{4}t^3 - \frac{3}{5}t, \sqrt{14}, \frac{105}{16}t^4 + \frac{3}{35} - \frac{6}{7}t^2, \sqrt{2}$$

ou aux polynômes d'Hermite (qui forment une base orthonormée pour le produit scalaire  $(f, g) = \int_{-\infty}^{\infty} f(t) g(t) e^{-t^2} dt$ )

```
> scal_Hermite:=(f,g)->
int(f*g*exp(-t^2),t=-infinity..infinity):
GS([1,t,t^2,t^3,t^4],scal_Hermite);
```

$$\frac{1}{1/4}, \frac{t\sqrt{2}}{1/4}, \frac{t^2 - \frac{1}{2}}{1/4}, \frac{2}{3} \frac{t^3 - \frac{3}{2}t}{1/4}, \frac{1}{3} \frac{t^4 + \frac{3}{4} - 3t^2}{1/4}, \sqrt{6}$$

## Matrices et endomorphismes

### Matrices symétriques

Rappelons que Maple permet de définir directement, à l'aide de la fonction **array**, des matrices symétriques

```
> A:=array(symmetric,1..3,1..3,[(1,1)=a,(1,2)=b,(1,3)=c,(2,2)
)=0,(2,3)=1,(3,3)=2]);
```

$$A := \begin{bmatrix} a & b & c \\ b & 0 & 1 \\ c & 1 & 2 \end{bmatrix}$$

ou antisymétriques

```
> A:=array(antisymmetric,1..3,1..3,[(1,2)=b,(1,3)=c,(2,3)=1]
);
```

$$A := \begin{bmatrix} 0 & b & c \\ -b & 0 & 1 \\ -c & -1 & 0 \end{bmatrix}$$

La fonction **randmatrix** permet de construire *au hasard* des matrices symétriques ou antisymétriques.

```
> A:=randmatrix(4,4,symmetric);
```

```

      4  -59  -55  40
A := -59  62  25  61
     -55  25   9  40
      40  61  40 -78

```

```
> A:=randmatrix(4,4,antisymmetric);
```

```

      0  57  -59  -8
A := -57   0  45 -93
      59 -45   0  92
      8  93 -92   0

```

La fonction Maple définie avec les options `positive_def`, `positive_semidef`, `negative_def` ou `negative_semidef` teste si une matrice est définie positive, positive, définie négative ou négative. Lorsqu'elle ne répond pas à la question, elle renvoie les conditions sur les coefficients pour que la matrice ait la bonne propriété.

```
> A:=matrix([[1,a,a],[a,2,a],[a,a,10]]):
definite(A,positive_def);
```

```
-2 + a^2 < 0 and -20 + 13 a^2 - 2 a^3 < 0
```

## Diagonalisation des matrices symétriques

Les fonctions usuelles permettent de rechercher valeurs propres et sous espaces propres de ces matrices. Rappelons que les matrices symétriques réelles sont diagonalisables sur  $\mathbf{R}$ , alors que les matrices antisymétriques réelles sont diagonalisables sur  $\mathbf{C}$ , les matrices symétriques ou antisymétriques complexes n'étant pas diagonalisables en général. Malheureusement, aucune fonction Maple prédéfinie ne permet la diagonalisation en base orthonormée d'une matrice symétrique réelle. Il est facile d'y remédier en appliquant l'algorithme d'orthonormalisation de Gram-Schmidt à une base de diagonalisation ordonnée suivant les différents sous-espaces propres.

```
> orthonormalise:=proc(base) local n,i,j,ortho,v;
    # version de Gram Schmidt adaptée aux familles de
    "vector"
    n:=nops(base);

    ortho:=[evalm(base[1]/sqrt(dotprod(base[1],base[1])))];
    for i from 2 to n do
        v:=base[i];
        for j from 1 to i-1 do

v:=evalm(v-dotprod(base[i],ortho[j])*ortho[j])
        od;

        ortho:=[op(ortho),evalm(v/sqrt(dotprod(v,v)))]
        od;
    ortho
end:

base_ortho_vp:=proc(A) local base,i;
    # on extrait une base de vecteurs propres
    base:=map(x->op(op(3,x)),[eigenvecs(A)]);
    # et on l'orthonormalise
    orthonormalise(base)
end:
```

Un exemple d'application avec une matrice numérique

```
> A:=map(evalf,randmatrix(4,4,symmetric,entries=rand(-2..2))
);
```

```

      -1. -2. -2.  2.
A := -2.  0  1.  2.
      -2.  1. -2. -2.
      2.  2. -2.  2.

```

avec la matrice de passage à une base orthonormée de vecteurs propres (il faut transformer

la liste en matrice, puis transposer le tout pour mettre les vecteurs propres en vecteurs colonnes)

```
> B:=transpose(matrix(base_ortho_vp(A)));
```

$$B := \begin{pmatrix} .4746394924 & .6454818850 & .4234878655 & -.4227629544 \\ -.7870677513 & .5969863164 & .09494631441 & .1229507593 \\ -.2735459424 & -.1196603788 & -.3834837744 & -.8739532054 \\ -.2835741930 & -.4611310744 & .8152198598 & -.2058163967 \end{pmatrix}$$

et on vérifie le résultat approché obtenu

```
> evalm(transpose(B) &* A &* B);
```

$$\begin{pmatrix} 2.274230279 & .35 \cdot 10^{-8} & -.10 \cdot 10^{-8} & -.13 \cdot 10^{-8} \\ .34 \cdot 10^{-8} & -3.907771806 & .2 \cdot 10^{-8} & .8 \cdot 10^{-9} \\ -.12 \cdot 10^{-8} & .2 \cdot 10^{-8} & 4.212698682 & -.32 \cdot 10^{-8} \\ .3 \cdot 10^{-9} & .9 \cdot 10^{-9} & -.25 \cdot 10^{-8} & -3.579157156 \end{pmatrix}$$

## Matrices hermitiennes

Aucune fonction spécifique de Maple ne traite des matrices hermitiennes. Le lecteur adaptera sans difficulté la fonction de diagonalisation en base orthonormée des matrices symétriques aux matrices hermitiennes.

## Décomposition de Cholesky

Toute matrice symétrique définie positive peut s'écrire de manière unique comme produit d'une matrice triangulaire supérieure à coefficients diagonaux strictement positifs par sa transposée. La fonction Maple **cholesky** renvoie cette matrice triangulaire.

```
> A:=randmatrix(4,4,entries=rand(-3..3));
B:=evalm(A &* transpose(A));
C:=cholesky(B);
```

$$C := \begin{pmatrix} \sqrt{7} & 0 & 0 & 0 \\ \frac{3}{7}\sqrt{7} & \frac{1}{7}\sqrt{1211} & 0 & 0 \\ \frac{4}{7}\sqrt{7} & \frac{37}{1211}\sqrt{1211} & \frac{22}{173}\sqrt{1211} & 0 \\ -\frac{5}{7}\sqrt{7} & \frac{15}{1211}\sqrt{1211} & \frac{11}{1211}\sqrt{1211} & \frac{8}{7}\sqrt{7} \end{pmatrix}$$

```
> evalm(B) = evalm(C &* transpose(C));
```

$$\begin{pmatrix} 7 & 3 & 4 & -5 \\ 3 & 26 & 7 & 0 \\ 4 & 7 & 23 & -1 \\ -5 & 0 & -1 & 13 \end{pmatrix} = \begin{pmatrix} 7 & 3 & 4 & -5 \\ 3 & 26 & 7 & 0 \\ 4 & 7 & 23 & -1 \\ -5 & 0 & -1 & 13 \end{pmatrix}$$

## Décomposition QR

Toute matrice inversible peut s'écrire de manière unique comme produit d'une matrice orthogonale et d'une matrice triangulaire supérieure à coefficients diagonaux strictement positifs. La fonction Maple **QRdecomp** renvoie la matrice triangulaire, la matrice orthogonale étant éventuellement stockée dans un paramètre optionnel.

```
> A:=randmatrix(4,4,entries=rand(-3..3));
B:=QRdecomp(A,Q='C'):
evalm(A) = evalm(C) &* evalm(B);
```

$$\begin{pmatrix} -2 & -2 & 2 & 1 \\ 3 & 0 & -3 & 3 \\ 1 & -2 & 2 & 1 \\ 0 & 0 & 3 & -2 \end{pmatrix} =$$

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

# Cours de mathématiques

par Denis Monasse

Ed. Vuibert

## Table des matières

- Plan général
- Algèbre générale
- Algèbre linéaire
- Réduction des endomorphismes
- Topologie des espaces métriques
- Espaces vectoriels normés
- Comparaison des fonctions
- Suites et séries numériques
- Fonctions d'une variable réelle
- Intégration
- Suites et séries de fonctions

- Séries entières
- Formes quadratiques
- Formes hermitiennes
- Séries de Fourier
- Calcul différentiel
- Equations différentielles
- Espaces affines
- Courbes
- Surfaces
- Intégrales multiples
- Index