Université Claude Bernard Lyon 1 - 2007/2008
Licence Sciences et Technologies - UE Mathématiques III, Algèbre

Travaux pratiques - planche 1

L'objectif de ces travaux pratiques est d'illustrer, d'expérimenter et d'appliquer certains résultats théoriques vus en cours ou dans les travaux dirigés sur la réduction des endomorphismes. Nous utiliserons le système de calcul formel Maple et essentiellement son paquetage `LinearAlgebra`, qui définit de nombreuses fonctions permettant de faire du calcul matriciel. Naturellement, le système de calcul Maple permet de résoudre des problèmes issus d'autres domaines des mathématiques. Vous pourrez le constater au détour de votre utilisation de l'aide.

Dans cette première séance, l'objectif est de prendre en main le système de calcul, en particulier sa syntaxe, ses mécanismes d'affectation et d'évaluation.

I. Le système de calcul Maple

1. Généralités

Toutes les commandes se terminent par un point-virgule ou un deux-points :

```
> ma_commande;
```

```
> ma_commandeUn:
  ma_seconde:
  la_derniere;
```

Appuyer sur la touche «Enter» pour exécuter la ligne de commande ou faire la combinaison «Shift + Enter » pour continuer la saisie à la ligne suivante sans lancer l'exécution. Le point-virgule affiche le résultat de l'évaluation, à la différence du deux-points, comme illustré ci-dessous.

```
> x:=1+1;
```

$$x := 2$$

```
> y:=x:
  y;
```

$$2$$

```
> z:=1+2:t:=z+y;u=z+t:
```

$$t := 5$$

L'interprète de commandes est sensible à l'utilisation des majuscules. Pour les commentaires, une fin de ligne après le caractère dièse `#` est ignorée.

Exercice 1.

1. Exécuter les commandes suivantes :

```
diff(x^2+x+sin(x),x)  limit(sqrt(x)/x,x=0) int(sin(x),x)
```

2. Exécuter le bloc de code suivant :

```
x:=1:
y:=x+2:
z:=x+3;
x,y;
```

Si vous avez un doute sur la syntaxe d'une commande, la commande `?nom_commande` retourne le manuel en ligne de la commande.

```
> diff(x,x^2+x,sin(x));
Error, wrong number (or type) of parameters in function diff
> ?diff;
```

La dernière commande ouvre le manuel d'aide sur la page correspondant à la définition de la commande. On peut aussi utiliser les menus déroulants de l'interface graphique et le moteur de recherche pour naviguer dans l'aide. On peut obtenir l'aide de l'aide avec la commande `?`.

La commande `%` (ou `''` selon les versions) permet de rappeler la valeur de la dernière expression évaluée. Il existe aussi les commandes `%%` et `%%%`, dont on imagine le sens.

```
> x:=Pi/3;
```

$$x := \frac{\pi}{3}$$

```
> sin(%);
```

$$\frac{\sqrt{3}}{2}$$

```
> %*% + cos(%%)^2;
```

$$1$$

2. Les variables et les expressions

Une variable est un nom dans lequel une valeur est stockée. Dans ce premier survol du système, nous ne parlerons pas des types des données. L'affection d'une valeur à une variable se fait par l'instruction suivante :

```
> nom_variable := valeur;
```

La commande `restart` permet de briser toutes les affectations déjà réalisées.

Exercice 2.

1. Tester l'enchaînement de commandes suivant :

   `x:=1;  x+x;  y:=x;  x:=x+3;  x:='x';  x:=x;  x+x;`

2. Puis les commandes suivantes :

   `x:=1;  x;  restart; x;`

3. Exécuter la ligne de commandes suivantes :

```
> x:=Pi/2; evalf(x,30);
```

Que réalise la procédure `evalf` ?

Pour changer uniquement l'affectation de la variable `x`, on peut utiliser comme dans l'exemple précédent la syntaxe `x :='x'`. Noter que des chaînes de caractères sont réservées afin de définir des constantes, des noms d'opérateurs ou des mots clés du langage. Par exemple, les chaînes de caractères suivantes sont protégées :

`Pi`, `infinity`, `I`, `D`, `and`, `for`, `if`, `fi`, ...

Pour l'évaluation, le système parcourt en sens inverse toutes les affectations. Par exemple :

```
> x:=y:
  y:=z:
  z:=1:
  x;
```

$$1$$

La procédure `eval` permet de spécifier le niveau d'évaluation. Par exemple :

```
> restart;
  x:=y:
  y:=z:
  z:=1:
  eval(x),eval(x,1),eval(x,2);
```

$$1, y, z$$

D'autres types d'objets peuvent être stockés par des variables ; comme par exemple les expressions algébriques :

```
> restart;
  a:=x^2+x+1+sin(x)+sqrt(y);
  b:=3*x+1=0;
```

$$a := x^2 + x + 1 + \sin(x) + \sqrt{y}$$

$$b := 3x + 1 = 0$$

De nombreuses opérations peuvent être appliquées aux expressions. On peut avoir besoin de les simplifier :

```
> c:=sin(x)^2+cos(x)^2:
  simplify(c);
```

$$1$$

On pourra aussi utiliser l'instruction `subs(x=a,expr)`, pour remplacer chaque occurence d'une sous-expression `x` dans l'expression `expr` par `a`.

```
> p:=cos(x)+2;
  eval(subs(x=Pi,p));
```

$$1$$

Dans la suite, nous aurons parfois besoin de manipuler des polynômes. On peut définir un polynôme en utilisant une instruction de la forme suivante :

```
> p:=x^2+3*x+1;
```

La fonction `divide` teste si un polynôme est diviseur :

```
> p:=3*(x+1)*(x^2+1);
  q:=x+1;
  divide(p,q);
```

$$\text{true}$$

Les fonctions `quo` et `rem` retournent le quotient et le reste de la division euclidienne :

```
> quo(p,q,x);
```

$$3x^2 + 3$$

```
> rem(p,q,x);
```

$$0$$

Pour développer et factoriser un polynôme :

```
> q:=expand(p);
> r:=sort(expand(p));
```

$$q := 3x^3 + 3x + 3x^2 + 3$$

$$q := 3x^3 + 3x^2 + 3x + 3$$

```
> factor(q);
```

$$3(x+1)(x^2+1)$$

```
> factor(q,complex);
```

$$3.(x+1.)(x+1.I)(x-1.I)$$

Pour rassembler des termes de même degré en une variable `x` :

```
> p:=a*x^2+b^2*x^2+y^2*x+c*x+y;
```

$$p := ax^2 + b^2x^2 + y^2x + cx + y$$

```
> collect(p,x);
```

$$(b^2+a)x^2 + (y^2+c)x + y$$

Nous utiliserons la procédure `gcdex` qui est une version étendue de l'algorithme d'Euclide aux polynômes. Nous l'utiliserons pour appliquer le théorème de Bézout : si P et Q sont deux polynômes premiers entre eux, alors on peut trouver des polynômes U et V tels que

$$PU + QV = 1.$$

Avec la procédure `gcdex` :

```
> p:=(x+1)*(x-3);
  q:=(x+2);
  gcdex(p,q,x,'u','v'):
  u;
  v;
```

$$\frac{1}{5}$$

$$\frac{4}{5} - \frac{x}{5}$$

II. Le paquetage `LinearAlgebra`

Le paquetage `LinearAlgebra` est une bibliothèque qui rassemble un ensemble de procédures dédiées au calcul matriciel. La syntaxe de ces procédures peut paraître parfois complexe et fastidieuse, mais l'emploi d'un tel paquetage s'avère très utile pour une pratique avancée du calcul matriciel.

Pour charger ce paquetage, on utilise la commande

```
> with(LinearAlgebra);
```

Les noms des procédures chargées sont alors affichés. Pour eviter cet affichage, il suffit de remplacer le symbole ; par :.

Les procédures principales de ce paquetage sont :

- pour construire des matrices ou des vecteurs :
  `BandMatrix`, `ConstantMatrix`, `DiagonalMatrix`, `IdentityMatrix`, `RandomMatrix`, `RandomVector`, `ScalarMatrix`, `ScalarVector`, `UnitVector`, `VandermondeMatrix`, `ZeroMatrix`, `ZeroVector`
- pour accéder à des éléments d'une matrice ou d'un vecteur :
  `Column, Diagonal, SubMatrix, SubVector`
- pour les opérations élémentaires sur les matrices :
  `Determinant, MatrixAdd, MatrixExponential, MatrixInverse, MatrixMatrixMultiply, MatrixPower, MatrixScalarMultiply, MatrixVectorMultiply, Multiply, NullSpace, Rank, ScalarMultiply, Trace, Transpose, VectorAdd, VectorScalarMultiply`

  On renvoie à l'aide de Maple pour la liste complète des procédures de ce paquetage.

La définition d'une matrice et d'un vecteur peut se faire en utilisant les procédures `Matrix` et `Vector`

```
> Matrix(2,3);
```

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

```
> Matrix([[1,1,1,1],[1,0,1,1]]);
```

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

```
> Vector([1,2,3]);
```

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Exercice 3.

1. Écrire les instructions permettant de définir les matrices suivantes. On préférera les procédures les mieux adaptées à la structure de la matrice.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}, \qquad \mathbf{V} = \begin{bmatrix} 1 & a & a^2 & a^3 & a^4 \\ 1 & b & b^2 & b^3 & b^4 \\ 1 & c & c^2 & c^3 & c^4 \\ 1 & d & d^2 & d^3 & d^4 \\ 1 & e & e^2 & e^3 & e^4 \end{bmatrix},$$

$$\mathbf{S} = \begin{bmatrix} 1 & a & b \\ a & 2 & c \\ b & c & 3 \end{bmatrix}, \qquad \mathbf{T} = \begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix}.$$

2. Calculer $\mathbf{A}^{50}$.

3. Montrer que $\mathbf{B}$ est orthogonale, i.e., que $\mathbf{B}\,{}^t\mathbf{B} = {}^t\mathbf{B}\mathbf{B} = \mathbf{1}_2$.

4. Calculer la trace et le rang de la matrice de Vandermonde $\mathbf{V}$, ainsi que son déterminant sous une forme factorisée. Calculer l'inverse de $\mathbf{V}$.

5. Les matrices $\mathbf{S}$ et $\mathbf{T}$ sont-elles inversibles ? Si oui, calculer leur inverse.

6. Pour quelle valeur de a, la matrice suivante

$$\begin{bmatrix} 1 & 1 & a \\ 1 & a & 1 \\ a & 1 & 1 \end{bmatrix}$$

est-elle inversible ?

Pour se donner des exemples de matrices, on peut faire appel à la procédure `RandomMatrix`, permettant de construire des matrices aléatoires.

Exercice 4. Construire une matrice de taille 10×10, dont les coefficients sont des entiers pris aléatoirement dans l'intervalle $[0, \ldots, 10]$.

III. Quelques exercices de diagonalisation

1. Représentation graphique des vecteurs propres

Pour débuter, on va utiliser le sous paquetage `LinearAlgebra` du paquetage `Student`, afin d'obtenir des représentations graphiques des vecteurs propres d'une matrice. La procédure `EigenPlot` illustre l'action d'une matrice sur des vecteurs unitaires. L'option `showunitvectors` permet de visualiser les vecteurs unitaires.

Exercice 5. Exécuter le bloc de code suivant et tester la procédure `EigenPlot` sur différentes matrices.

```
> with(Student[LinearAlgebra]):
  infolevel[Student[LinearAlgebra]] := 1:
  EigenPlot(<<1,1/2>|<1/3,3/5>>);
  EigenPlot(<<1,1/2>|<1/3,3/5>>, showunitvectors);
  EigenPlot(<<1/2,-5/3,0>|<0,4/3,0>|<-3/2,5/3,-1>>, axes=normal);
```

On pourra visiter l'aide pour tester d'autres procédures de visualisation.

2. Généralités

Le paquetage `LinearAlgebra` contient des procédures permettant de calculer le polynôme caractéristique ou le polynôme minimal d'une matrice, ou encore sa réduite de Jordan. On pourra par exemple regarder l'aide des commandes suivantes :

`CharacteristicPolynomial`, `Eigenvalues`, `Eigenvectors`, `JordanForm`, `MinimalPolynomial`.

Exercice 6.

1. Calculer le polynôme caractéristique et le polynôme minimal de la matrice suivante :

$$\mathbf{A} = \begin{bmatrix} -1 & 1 & 3 \\ -2 & 2 & 2 \\ -2 & 1 & 4 \end{bmatrix}$$

2. La matrice $\mathbf{A}$ est-elle diagonalisable ?
3. Déterminer le spectre de $\mathbf{A}$ ainsi qu'une base pour chaque espace propre.

Exercice 7. Les matrices suivantes sont-elles diagonalisables sur $\mathbb{Q}$, sur $\mathbb{R}$, sur $\mathbb{C}$?

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

Exercice 8.

On considère les trois matrices réelles suivantes

$$\mathbf{A} = \begin{bmatrix} a & 1 & 1 & 1 \\ 1 & b & 1 & 1 \\ 1 & 1 & c & 1 \\ d & 1 & 1 & d \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -1 & a & d \\ b & e & -1 \\ c & 1 & f \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} a & 2 & 1 \\ 2 & b & -1 \\ -1 & 1 & c \end{bmatrix}.$$

1. Déterminer les quadruplets $(a, b, c, d) \in \mathbb{R}^4$, pour lesquels la matrice $\mathbf{A}$ admet $\begin{bmatrix} -1 \\ 2 \\ 2 \\ -1 \end{bmatrix}$ pour vecteur propre. On pourra utiliser la procédure `solve`.

2. Déterminer les réels a, b, c, d, e, f, pour lesquels la matrice $\mathbf{B}$ admet $\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$ pour vecteurs propres.

3. Déterminer les triplets $(a, b, c) \in \mathbb{R}^3$, pour lesquels la matrice $\mathbf{C}$ admet 1, a et b comme valeurs propres.

Exercice 9. Déterminer l'expression générale du polynôme minimal d'une matrice antisymétrique de taille 3×3 puis d'une matrice antisymétrique de taille 4×4.

Exercice 10. On considère la matrice réelle suivante

$$\mathbf{A} = \begin{bmatrix} 1 & a^2 & a^2 & a \\ 1 & 1 & a & 1 \\ 1 & a & 1 & 1 \\ a & a^2 & a^2 & 1 \end{bmatrix}.$$

1. Calculer le polynôme minimal et le polynôme caractéristique de $\mathbf{A}$.
2. Pour quelles valeurs de a, la matrice $\mathbf{A}$ est-elle diagonalisable ?
3. Diagonaliser $\mathbf{A}$ en donnant une matrice de passage.

3. Matrices compagnons

Afin de construire des exemples de matrices permettant de tester nos programmes des prochaines séances, on pourra utiliser la procédure `CompanionMatrix` qui retourne la matrice compagnon du polynôme donné en argument :

```
> P := x^4 + x^3 + x^2 + x:
  CompanionMatrix(P,x);
```

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

Un des intérêts de la procédure `CompanionMatrix` est de construire des exemples de matrices dont on connaît par avance le polynôme caractéristique. On pourra aussi les procédures `CharacteristicMatrix`, `JordanBlockMatrix`.

Exercice 11.

1. Construire une matrice ayant pour polynôme caractéristique

$$P = (X^3 - 1)(X^2 + X + 1).$$

2. Cette matrice est-elle diagonalisable sur $\mathbb{R}$ ou sur $\mathbb{C}$?
3. Construire une matrice compagnon de taille 3×3 diagonalisable et une matrice compagnon 3×3 non diagonalisable.

4. Calcul des projecteurs spectraux

Soit u un endomorphisme d'un $\mathbb{K}$-espace vectoriel E. Nous savons, d'après le thèorème de décomposition en noyaux, que si P et Q sont deux polynômes premiers entre eux et tels que leur produit PQ soit annulateur de u, alors on a une décomposition de E en somme directe :

$$E = \text{Ker } P(u) \oplus \text{Ker } Q(u).$$

Nous souhaitons calculer les projections de E sur les sous-espaces Ker $P(u)$ et Ker $Q(u)$.

Exercice 12.

1. Définir deux polynômes à coefficients réels P et Q premiers entre eux.
2. Construire la matrice compagnon $\mathbf{C}$ du polynôme PQ.
3. Vérifier que le polynôme PQ annule la matrice $\mathbf{C}$. On pourra utiliser la procédure `MatrixFunction`.
4. Construire deux polynômes U et V tels que $UP + VQ = 1$. On pourra utiliser la procédure `gcdex`.
5. Calculer les matrices des projecteurs π_1 et π_2 sur les sous-espaces Ker $P(\mathbf{C})$ et Ker $Q(\mathbf{C})$ respectivement.
5. Vérifier que π_1 et π_2 sont bien des projecteurs.

Exercice 13. Calculer les projecteurs spectraux des matrices suivantes rencontrées au cours des travaux dirigés

$$\mathbf{A} = \begin{bmatrix} 5 & 1 & 3 \\ 4 & 3 & 4 \\ -1 & -1 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} -1 & 1 & 3 \\ -2 & 2 & 2 \\ -2 & 1 & 4 \end{bmatrix}.$$