TP Maple 3 | Algèbre linéaire

Après quelques rappels sur l'algèbre linéaire de première année au moyen des librairies **LinearAlgebra** *et* **Linalg**, *nous exposerons les outils de réduction des endomorphismes. Nous évoquerons pour finir la décomposition de Dunford et détaillerons un algorithme efficace permettant de la calculer.*

1. La librairie LinearAlgebra

Maple met à disposition de l'utilisateur deux bibliothèques d'Algèbre linéaire : LINALG et LINEARALGEBRA. Cette dernière a été introduite depuis la version 7 du logiciel et vise à supplanter la précédente. Nous ne détaillerons ici que les commandes offertes par LINEARALGEBRA.

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

[‘& x‘, Add, Adjoint, BackwardSubstitute, BandMatrix, Basis, BezoutMatrix, BidiagonalForm, BilinearForm, CharacteristicMatrix, CharacteristicPolynomial, Column, ColumnDimension, ColumnOperation, ColumnSpace, CompanionMatrix, ConditionNumber, ConstantMatrix, ConstantVector, Copy, CreatePermutation, CrossProduct, 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, LA Main, LUDecomposition, LeastSquares, LinearSolve, 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, ToeplitzMatrix, Trace, Transpose, TridiagonalForm, UnitVector, VandermondeMatrix, VectorAdd, VectorAngle, VectorMatrixMultiply, VectorNorm, VectorScalarMultiply, ZeroMatrix, ZeroVector, Zip]

2. Calcul matriciel

Nous commencerons par le cas des vecteurs colonne ou ligne.

2.1. Calcul vectoriel

Le logiciel est capable d'effectuer du calcul vectoriel (en lignes ou en colonnes).

Définitions possibles d'un vecteur

Vecteurs colonnes : V := **Vector**($[v_1,\ldots,v_n]$) Vecteurs lignes : V := $<v_1|v_2|\ldots|v_n>$

On pourra alors utiliser les commandes suivantes aussi bien sur des vecteurs colonnes que sur des vecteurs lignes (mais pas les deux à la fois !) :

```
> W1:=Vector([1,-2,1]): W2:=Vector([1,1,-2]): W3:=Vector([-2,1,1]):
  Basis([W1,W2,W3]);
```

$$\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}$$

```
> Dimension(W1);
```

$$3$$

```
> W1+W2, 3*W3;
```

$$\begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -6 \\ 3 \\ 3 \end{bmatrix}$$

Manipulation des vecteurs

- **Dimension** : nombre de composante(s) d'un vecteur.
- $W_1 + W_2$: somme des vecteurs W_1 et W_2.
- $\lambda * W$: action d'un scalaire sur un vecteur.
- **Basis**($[W_1, W_2, \ldots, W_n]$) : retourne une base de vect$(W_1,\ldots,W_n)$.

2.2. Matrices de taille quelconque

Les matrices sont définies par la donnée de leurs lignes.

Définition d'une matrice ou d'un vecteur

► *Définition par la donnée des lignes :*

M := **Matrix(*L*)**

où L est la liste des lignes (données sous forme de listes) de la matrice M.

► *Définition par la donnée des coefficients :*

M := **Matrix(n,p,f)**

où f est une fonction de deux variables $(i,j) \mapsto f(i,j)$, n et p les dimensions de la matrice M de coefficients $f(i,j)$.

On trouvera ci-dessous cette syntaxe ainsi qu'un mode de définition plus ancien mais toujours valable.

```
> A:=Matrix([[0,1,2],[1,1,1]]): B:= <<1,2,3>|<4,5,6>|<7,8,9>>: A,B;
```

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}, \quad \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> f:=(i,j)->i+j: Matrix(4,4,f);
```

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

On pourra aussi définir des matrices au moyen de vecteur-colonnes ou de vecteur-lignes [1].

```
> V:= Vector([1,20,1]);
```

$$\begin{bmatrix} 1 \\ 20 \\ 1 \end{bmatrix}$$

L'utilisateur a accès en lecture et en enregistrement aux coefficients de la matrice.

1. Ce point est important car il assure la validité des opérations décrites ci-dessous.

```
> A[1,2]:= 9: A[1,2],A;
```

$$9, \begin{bmatrix} 0 & 9 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

2.3. Opérations sur les matrices

Voici un petit panorama des opérations matricielles sous Maple. Les commandes essentielles et leur syntaxe sont consignées dans un tableau en fin de paragraphe.

```
> A:=Matrix([[0,1,2],[1,1,1]]): B:=Matrix([[5,4,0],[2,5,-1]]):
  C:=Matrix([[1,-1,2],[1,0,1],[1,1,2]]): V:= Vector([1,20,1]):
> A,B,C,V;
```

$$\begin{bmatrix} 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 5 & 4 & 0 \\ 2 & 5 & -1 \end{bmatrix}, \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 20 \\ 1 \end{bmatrix}$$

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

$$2,3,2,3$$

```
> A+B, A.C, A.V, Transpose(A), C^2, C^(-1);
```

$$\begin{bmatrix} 5 & 5 & 2 \\ 3 & 6 & 0 \end{bmatrix}, \begin{bmatrix} 3 & 2 & 5 \\ 3 & 0 & 3 \end{bmatrix}, \begin{bmatrix} 22 \\ 22 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 2 & 1 \end{bmatrix}, \begin{bmatrix} 2 & 1 & 5 \\ 2 & 0 & 4 \\ 4 & 1 & 7 \end{bmatrix}, \begin{bmatrix} -1/2 & 2 & -1/2 \\ -1/2 & 0 & 1/2 \\ 1/2 & -1 & 1/2 \end{bmatrix}$$

Opérations sur les matrices

- **A+B** : calcule $A+B$.
- **A.B** : calcule AB.
- **Transpose(A)** : calcule ${}^tA$.
- **A^n** : calcule A^n ($n \in \mathbb{Z}$).
- **Trace** : calcule la trace.
- **Determinant** : calcule le déterminant.
- **Equal(A,B)** : teste si $A = B$.
- **A := Matrix(L)** : crée une matrice A dont la liste des lignes est L.
- **A[i,j]** : calcule le coefficient $A_{i,j}$ de la matrice A.
- **HermitianTranspose** : calcule la transconjuguée d'une matrice.

Opérations sur les matrices (suite)

- **DiagonalMatrix(d_1,...,d_n)** : matrice diagonale de coefficients $d_1, \ldots, d_n$.
- **Dimension** : calcule les dimensions d'une matrice (lignes, colonnes).
- **RowDimension** : calcule le nombre de ligne(s) d'une matrice.
- **ColumnDimension** : calcule le nombre de colonne(s) d'une matrice.
- **DeleteColumn** : supprime des colonnes.
- **DeleteRow** : supprime des lignes.
- **SubMatrix(A,R,C)** : sous-matrice de A obtenue en extrayant les colonnes C, les lignes R.

Il est également important de savoir *former* l'ensemble des coefficients d'une matrice donnée. C'est la commande **convert** qui permet ce petit miracle. On s'en souviendra lors de la résolution d'équations d'inconnue matricielle telle que les calculs de commutants (et même certaines équations non-linéaires).

L'ensemble des coefficients d'une matrice

convert(A,set) renvoie l'ensemble dont les éléments sont les coefficients de la matrice A.

2.4. Systèmes linéaires

Deux pistes sont envisageables pour la résolution des systèmes linéaires : l'utilisation de **solve** (qui ne fait pas partie de la bibliothèque LINEARALGEBRA ou de la commande **LinearSolve** de LINEARALGEBRA. On recommande l'utilisation de cette dernière pour ces nombreuses options (voir l'aide en ligne).

La commande **solve** prend en arguments les lignes du système :

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

$$x = 1, y = 0, z = -1$$

La commande **LinearSolve** prend en arguments la matrice et le second membre du système :

```
> A:=Matrix([[1,1,1],[2,3,1],[1,1,-1]]):b:=Vector([0,1,2]):
  LinearSolve(A,b);
```

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

Dans le cas d'un système de Cramer $AX = b$, on peut aussi calculer $A^{-1}.b$ mais cela est souvent maladroit car beaucoup plus coûteux en calculs.

Résolution d'un système linéaire

- **Solve(*lignes du système séparées par des virgules*)** : résout le système linéaire.
- **LinearSolve(A,b)** : résout le système linéaire $AX = b$.

Remarquons que la commande **LinearSolve** accepte des inconnues matricielles.

```
> A:=Matrix([[1,1,1],[2,3,1],[1,1,-1]]):
  b:=Matrix([[0,1,1],[0,0,0],[2,1,1]]): LinearSolve(A,b);
```

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

2.5. Noyau, image et rang

Maple dispose de commandes permettant le calcul du rang, du noyau et de l'image d'une matrice donnée. Rappelons que l'image d'une matrice, notée $\mathrm{Im}(A)$, est par définition l'espace engendré par ses vecteurs-colonnes ; le noyau, noté $\mathrm{Ker}(A)$, est l'espace de vecteurs colonnes X tels que $AX = 0$.

Manipulation des matrices

- **Rank** : calcule le rang de la matrice.
- **ColumnSpace** : calcule une base de l'image de la matrice.
- **NullSpace** : calcule une base du noyau.
- **RowSpace** : calcule une base de l'espace engendré par les lignes d'une matrice.

```
> A:=Matrix([[1,1,1],[2,3,1],[1,2,0]]): NullSpace(A), Rank(A);
```

$$\begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}, \ 2$$

```
> ColumnSpace(A), RowSpace(A);
```

$$\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \ \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \ [1,0,2], [0,1,-1]$$

Exercice 1.

Soit

$$A = \begin{pmatrix} 1 & -3 & 3 & -2 \\ 1 & 1 & 1 & 2 \\ 2 & 4 & 1 & 6 \\ 1 & 1 & 1 & 2 \end{pmatrix}.$$

1. Déterminer le rang de A ainsi qu'une base de $\mathrm{Im}(A)$.

2. Déterminer le noyau de A. En déduire des relations de liaisons entre les colonnes de A.

Exercice 2.

Trouver l'ensemble des matrices qui commutent avec

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

Exercice 3.

Soit

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

1. Soit $M \in \mathfrak{M}_3(\mathbb{R})$. Montrer que M commute avec A *si et seulement si* $M \in \mathrm{vect}(\mathbb{I}_3, A, A^2)$.

2. Existe-t-il $B \in \mathfrak{M}_3(\mathbb{R})$ telle que $A = B^2$?

3. Réduction des endomorphismes

Le logiciel permet le calcul des éléments propres d'une matrice.

```
> A := Matrix([[1,-1,-1],[-2,2,3],[2,-2,-3]]):
    A, CharacteristicMatrix(A,lambda),
    factor(CharacteristicPolynomial(A,lambda)),
    Eigenvalues(A); Eigenvectors(A);
```

$$\begin{bmatrix} 1 & -1 & -1 \\ -2 & 2 & 3 \\ 2 & -2 & -3 \end{bmatrix}, \begin{bmatrix} -1+\lambda & 1 & 1 \\ 2 & -2+\lambda & -3 \\ -2 & 2 & 3+\lambda \end{bmatrix}, (\lambda-1)\lambda(\lambda+1), \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

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

Outils de réduction

- **CharacteristicMatrix(A,lambda)** : $\lambda I_n - A$.
- **CharacteristicPolynomial(A,lambda)** : $\det(\lambda I_n - A)$.
- **Eigenvalues** : renvoie les valeurs propres (comptées avec leur multiplicité algébrique, i.e. leur multiplicité en tant que racine du polynôme caractéristique) sous la forme d'un vecteur-colonne.
- **Eigenvectors** : renvoie les valeurs propres sous la forme d'un vecteur-colonne (cf. ci-dessus) *puis*, sous forme d'une matrice de taille n, des familles génératrices des sev propres.
- **MinimalPolynomial(A,lambda)** : calcule le polynôme minimal de A.

Les familles génératrices seront toujours composées d'une base du sous-espace propre correspondant complétée par des vecteurs nuls pour atteindre la multiplicité algébrique lorsque la multiplicité géométrique (i.e. la dimension du sev propre considéré) est strictement inférieure à la multiplicité algébrique.

```
> M:=Matrix([[1,1,0,0],[0,1,0,0],[0,0,2,1],[0,0,0,2]]):
  Eigenvalues(M), Eigenvectors(M);
```

$$\begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Dans le cas d'une matrice diagonalisable, on se souviendra que la matrice renvoyée en deuxième argument par **Eigenvectors** est la matrice de passage de la base canonique de $\mathbb{R}^n$ à la base de vecteurs propres trouvée par le logiciel.

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

$$m := \begin{bmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix}$$

```
> d, p := Eigenvectors(m)[1], Eigenvectors(m)[2];
```

$$d, p := \begin{bmatrix} 0 \\ -3 \\ -3 \end{bmatrix}, \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

```
> p.DiagonalMatrix(Transpose(d)).p^(-1);
```

$$\begin{bmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{bmatrix}$$

On vérifie bien que $m = p \cdot d \cdot p^{-1}$.

Exercice 4.

On considère n réels deux à deux distincts $a_0, ..., a_n$ et A le polynôme

$$A = \prod_{k=0}^{n} (X - a_k)$$

Soit B un polynôme réel tel que

$$\forall 0 \leqslant k \leqslant n, \ B(a_k) \neq 0.$$

On considère l'application f qui à un polynôme P de $E = \mathbb{R}_n[X]$ associe le reste $R = f(P)$ de la division euclidienne de $B \cdot P$ par A.

1. Justifier qu'on définit ainsi un endomorphisme de $\mathbb{R}_n[X]$.

2. Etude d'un exemple avec le logiciel de calcul formel : on demande de résoudre toutes les questions du **2.** avec le logiciel. On choisit $n = 2$ et

$$A = (X-1)(X-2)(X-3),$$

et $B = X^3$. Ainsi f est ici l'endomorphisme de $E = \mathbb{R}_2[X]$ qui à $P \in E$ associe le reste de la division euclidienne de $X^3 \cdot P$ par le polynôme $(X-1)(X-2)(X-3)$.

2.a. Créer l'application f. On pourra utiliser la commande **rem** qui fournit le reste de la division euclidienne.

2.b. Expliciter alors l'image du polynôme $P = aX^2 + bX + c$.

2.c. Déterminer le noyau de f.

2.d. Suivre le même procédé pour déterminer les éléments propres de f, en annulant les coefficients de $Q = f(P) - \lambda P$.

2.e. Créer la matrice de f dans la base canonique de E, et retrouver ainsi les valeurs propres et les vecteurs propres de f.

3. On revient au cas général. Déterminer le noyau, les éléments propres (valeurs propres, sous-espaces propres) et le déterminant de f. L'endomorphisme f est-il diagonalisable ?

Exercice 5.

Soient a un réel non nul et f l'endomorphisme de $\mathbb{R}^3$ dont la matrice dans la base canonique est

$$A = \begin{pmatrix} -1 & a & a \\ 1 & -1 & 0 \\ -1 & 0 & -1 \end{pmatrix}.$$

1. f est il diagonalisable sur $\mathbb{R}$, trigonalisable sur $\mathbb{R}$?

2. Trouver une base $\mathscr{B}$ de $\mathbb{R}^3$ dans laquelle la matrice de f soit égale à

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

3. On se propose de généraliser cette étude en considérant, pour $n \geqslant 2$, les matrices de la forme

$$A_n = \begin{pmatrix} -1 & a & a & \dots & a \\ 1 & -1 & 0 & \dots & 0 \\ -1 & 0 & -1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ (-1)^n & 0 & 0 & \dots & -1 \end{pmatrix} \in \mathfrak{M}_n(\mathbb{R})$$

avec $a \in \mathbb{R}$.

3.a. Etudier le cas où $a = 0$ (diagonalisabilité ou, à défaut, trigonalisabilité, comment obtenir une matrice de passage).

3.b. Dorénavant $a \neq 0$. Ecrire une fonction qui prend en argument un entier $n \geqslant 3$, un paramètre a, et retourne la matrice A_n donnée ci-dessus. Utiliser alors le logiciel pour émettre quelques conjectures sur la digonalisabilité et la trigonalisabilité de A_n sur $\mathbb{R}$ puis les démontrer.

4. Décomposition de Dunford

Dans ce paragraphe, E désigne un espace vectoriel de dimension finie sur le corps $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$.

4.1. La théorie

Soit $f \in \mathscr{L}(E)$ de polynôme minimal

$$\mu_f = \prod_{\ell=1}^{k} (X - \lambda_\ell)^{\alpha_\ell}$$

avec les λ_ℓ, $1 \leqslant \ell \leqslant k$ deux à deux distincts. Puisque les polynômes $(X-\lambda_\ell)^{\alpha_\ell}$ sont deux à deux premiers, on déduit du théorème chinois dans $\mathbb{K}[X]$ l'existence de polynômes $P_1, P_2, \ldots, P_k$ de $\mathbb{K}[X]$ tels que

$$\forall 1 \leqslant \ell \leqslant k, \quad \forall 1 \leqslant j \leqslant k, \quad P_\ell \equiv \delta_{\ell,j} \left[\left(X - \lambda_j\right)^{\alpha_j} \right]$$

On vérifie que les endomorphismes $P_\ell(f)$ sont les projections sur $\operatorname{Ker}\left(\left(f - \lambda_\ell id\right)^{\alpha_\ell}\right)$ parallélement à

$$\bigoplus_{j \neq \ell} \operatorname{Ker}\left(\left(f - \lambda_j id\right)^{\alpha_j}\right)$$

On prouve alors facilement qu'en posant

$$\begin{cases} d = \displaystyle\sum_{\ell=1}^{k} \lambda_\ell \cdot P_\ell(f) \\ \\ n = f - d \end{cases}$$

n est nilpotent, d diagonalisable, $d \circ n = n \circ d$ et $f = d + n$.

Nous venons d'établir le théorème suivant :

Théorème 1. (Décomposition de Dunford)
Soient E un espace vectoriel de dimension finie et f un endomorphisme de E dont le polynôme caractéristique est scindé. Alors, il existe un unique couple (d, n) d'endomorphismes de E tels que d est diagonalisable, n est nilpotent, d et n commutent et $f = d + n$. De plus, d et n sont des polynômes en f : $f = d + n$ est la décomposition de Dunford de f.

La démonstration donnée ci-dessus n'est *constructive* que si les valeurs propres de f sont connues. Dans la pratique, c'est loin d'être le cas car on ne sait pas calculer explicitement les racines du polynôme caractéristique. Il reste donc à élaborer une méthode constructive de la décomposition de Dunford.

4.2. Un algorithme efficace

L'algorithme que nous présentons ici est adapté de la célèbre méthode de Newton pour le calcul approché d'une solution de $\phi(x) = 0$ ou ϕ est une fonction dérivable de dérivée ne s'annulant pas. On définit une suite par récurence par la fomule :

$$x_{n+1} = x_n - \frac{\phi(x_n)}{\phi'(x_n)}$$

Dans les bons cas $(x_n)_{n\geqslant 0}$ converge. La limite est alors une solution de l'équation $\phi(x) = 0$.

Revenons à la décomposition de Dunford de $f \in \mathscr{L}(E)$: le point de départ est la constation que les racines du polynôme minimal de d sont simples et sont les valeurs propres de f. Notons $P = \chi_f$ le polynôme caractéristique de f et $Q = \frac{P}{P \wedge P'}$. On a alors $Q(d) = 0$.

Théorème 2. (Décomposition effective de Dunford)
Soient E un $\mathbb{K}$-ev de dimension finie, $f \in \mathscr{L}(E)$, $P = \chi_f$ et $Q = \dfrac{P}{P \wedge P'}$. La suite d'endormorphismes définie par

$$\begin{cases} u_0 &= f \\ u_{n+1} &= u_n - Q(u_n) \cdot Q'(u_n)^{-1} \end{cases}$$

est bien définie et stationnaire. On note d sa limite et on pose $n = f - d$. Alors n est nilpotent, d est diagonalisable et ces deux endomorphismes commutent.

Exercice 6. *Preuve de la décomposition de Dunford effective*

On reprend les notations précédentes.

1. Montrer qu'il existe un couple $(U, V) \in \mathbb{K}[X]$ tel que $UP + VQ' = 1$.

2. On définit une suite $(S_n)_{n\in\mathbb{N}}$ de polynômes de $\mathbb{K}[X]$ en posant :

$$\begin{cases} S_0 &= X \\ S_{n+1} &= S_n - (Q \circ S_n) \cdot (V \circ S_n) \end{cases}$$

2.a. Etablir que

$$\forall n \in \mathbb{N},\ Q \circ S_n \equiv 0 \left[Q^{2^n}\right]$$

2.b. Montrer que

$$\forall n \in \mathbb{N},\ X - S_n \equiv 0\,[Q]$$

2.c. En déduire l'existence d'un entier naturel n_0 tel que

$$\begin{cases} f - S_{n_0}(f) \text{ est nilpotent} \\ Q\left(S_{n_0}(f)\right) = 0 \end{cases}$$

3. En déduire le théorème 2.

Exercice 7. *Calcul de la décomposition de Dunford*

Ecrire une procédure `Dunford(A)` d'argument A, matrice carrée, renvoyant la décomposition de Dunford de A.