I. Quelques primitives de programmation

Dans cette partie, nous présentons quelques instructions de programmation qui seront utiles pour la suite de ces travaux pratiques.

On appelle procédure un groupe d'instructions placées dans un même bloc selon la syntaxe suivante :

```
> nom_procedure:=proc(arg_1,...,arg_n)
  local var_1,...,var_n;
  global varg_1,...,varg_m;
      [ instructions ]
  end;
```

Lors de l'appel de la procédure, les instructions sont exécutées séquentiellement, i.e., dans l'ordre les unes après les autres. Voici un exemple :

```
> plusun:=proc(n);
  n+1;
  end;
> plusun(2);

                                    3
```

Voici deux structures de contrôle qui vous seront utiles dans la suite. Une instruction conditionnelle s'écrit avec la syntaxe suivante

```
if condition1 then
    suite d'instructions 1
elif condition2 then
    suite d'instructions 2
...
else
    suite d'instructionq 3
fi;
```

Par exemple

```
> exemple:=proc(n);
  if n>0 then
   RETURN(n+1);
  elif n=0 then
   RETURN(n);
  else
   ERROR(‘Entier négatif‘);
  fi;
  end;
> exemple(3);

                                    4
```

```
> exemple(-3);
Error, (in exemple) Entier négatif
```

L'instruction itérative `for` réalise une boucle contrôlée par une variable selon la syntaxe suivante

```
> for var from debut by pas to fin while condition
  do
     suite d'instructions;
  od;
```

Par exemple, tester la procédure suivante

```
> exemple:=proc()
  local N,i;
  N:=0;
  for i to 10 while(N<11)
    do
      N:=N+i;
      print(i,N);
    od;
end;
> exemple();
```

Exercice 1. Écrire une procédure testant si une matrice est nilpotente.

II. Projecteurs spectraux

L'objectif de cet exercice est de construire une procédure `projsSpec(A)`, qui prend en argument une matrice complexe et qui retourne une liste $[[\lambda_1, h_1, \pi_1], \ldots, [\lambda_p, h_p, \pi_p]]$, où λ_1, ..., λ_p sont les valeurs propres dans $\mathbb{C}$ de la matrice $\mathbf{A}$ donnée en argument, h_i est l'ordre de multiplicité algébrique de la valeur propre λ_i et π_i la matrice de la projection sur le sous-espace caractéristique $N_{\lambda_i} = \text{Ker } (\mathbf{A} - \lambda_i \mathbf{1}_n)^{h_i}$ parallèlement au sous-espace $\bigoplus_{\substack{j=1 \\ i \neq j}}^{p} N_{\lambda_j}$.

On pourra utiliser l'algorithme vu en cours. Si Q est un polynôme annulateur de $\mathbf{A} \in \mathcal{M}_n(\mathbb{C})$ (par exemple le polynôme minimal) tel que

$$Q = \prod_{i=1}^{p} (X - \lambda_i)^{\alpha_i},$$

on pose

$$Q_i = \prod_{\substack{j=1 \\ j \neq i}}^{p} (X - \lambda_j)^{\alpha_j}. \tag{1}$$

Les polynômes Q_i, $i = 1, ..., p$ sont premiers entre eux dans leur ensemble. On calcule les polynômes U_i de la décomposition de Bézout, i.e., vérifiant

$$U_1 Q_1 + \ldots + U_p Q_p = 1.$$

Nous avons montré qu'alors le projecteur π_i est donné par $\pi_i = U_i(\mathbf{A}) Q_i(\mathbf{A})$.

Exercice 2.

1. En utilisant la procédure `gcdex`, écrire une procédure `decompositionBezout(liste,x)` qui prend en premier argument une liste de polynômes $[P_1, ..., P_k]$ en la variable spécifiée en le deuxième argument et retourne la liste $[G, [U_1, ..., U_k]]$ où G est le pgcd des polynômes P_1, ..., P_k et les U_i vérifient

$$U_1 P_1 + \ldots + U_k P_k = G.$$

2. Écrire une procédure `decomposePolyMin(A)` qui prend en argument une matrice complexe et retourne une liste $[[\lambda_1, k_1, Q_1], \ldots, [\lambda_p, k_p, Q_p]]$ où les λ_i sont les valeurs propres de A, les k_i leur ordre de multiplicité dans le polynôme minimal et Q_i le polynôme (1) lorsque Q est le polynôme minimal de A. On pourra utiliser la procédure `roots`.

3. En déduire la procédure `projsSpec(A)`.

4. Tester la procédure avec les matrices suivantes, déjà rencontrées à plusieurs reprises,

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

III. Polynôme minimal

On souhaite écrire une procédure permettant de calculer le polynôme minimal d'une matrice. Naturellement on s'interdit l'utilisation de la procédure `MinimalPolynomial` qui réalise ce calcul.

Nous savons que toute matrice $\mathbf{A} \in \mathcal{M}_n(\mathbb{K})$ admet un polynôme minimal. En effet, la famille

$$\mathbf{1}_n, \mathbf{A}, \mathbf{A}^2, \ldots, \mathbf{A}^{n^2},$$

composée de $n^2 + 1$ vecteurs est liée. Il existe donc des scalaires non tous nuls tels que

$$a_0\mathbf{1}_n + a_1\mathbf{A} + a_2\mathbf{A}^2 + \ldots + a_{n^2}\mathbf{A}^{n^2} = 0$$

et le polynôme $Q = a_0 + a_1X + a_2X^2 + \ldots + a_{n^2}X^{n^2}$ est annulateur de $\mathbf{A}$.

Afin d'écrire une procédure permettant de déterminer le polynôme minimal de $\mathbf{A}$, on observe que la première famille de la suite de familles

$$(\mathbf{A}^i)_{0 \leqslant i \leqslant k}, \qquad k \in \{0, \ldots, n^2\},$$

est libre car elle est réduite à l'identité et que la dernière de ces familles est liée. Il existe donc un plus petit indice k_0 tel que la famille $(\mathbf{A}^i)_{0 \leqslant i \leqslant k_0}$ soit liée. Par définition de k_0 la famille $(\mathbf{A}^i)_{0 \leqslant i \leqslant k_0 - 1}$ est libre. Il existe donc une unique famille $(a_i)_{0 \leqslant i \leqslant k_0 - 1}$ de scalaires tels que

$$a_0\mathbf{1}_n + a_1\mathbf{A} + a_2\mathbf{A}^2 + \ldots + a_{k_0-1}\mathbf{A}^{k_0-1}U_k + \mathbf{A}^{k_0} = 0.$$

Pour déterminer le polynôme minimal d'une matrice $\mathbf{A} \in \mathcal{M}_n(\mathbb{K})$, il suffit donc de résoudre successivement le système de n^2 équations en les scalaires a_i, $0 \leqslant i \leqslant k-1$:

$$a_0\mathbf{1}_n + a_1\mathbf{A} + a_2\mathbf{A}^2 + \ldots + a_{k-1}\mathbf{A}^{k-1} + \mathbf{A}^k = 0 \qquad (\mathcal{S}_k)$$

Exercice 3.

1. Montrer que le premier indice pour lequel le système $(\mathcal{S}_k)$ admet une solution est k_0 et que cette solution est unique.

2. Déduire de ce qui précède une procédure permettant de déterminer le poynôme minimal d'une matrice $\mathbf{A}$.

3. Comparer sur des exemples les résultats obtenus avec ceux obtenus avec la procédure `MinimalPolynomial`.