Cycle Ingénieur

Mathématiques générales

CI-SST81E6

1re année

Nicolas Bur
n.bur@estia.fr

Version 1.1 du samedi 15 avril 2023

Notes de versions

v 1.0 du 2023-3-22
- Remise en forme du cours de Jean Esterle
- remplacement des codes Mupad par des codes en python

v 1.1 du 2023-4-15 Correction de coquilles

Résumé

Ce document est le support de cours pour le module CI-SST81E6 – Mathématiques générales, dont la présentation est donnée p. ix.

Sommaire

Table des figures

Liste des tableaux

Liste des définitions

Liste des théorèmes

Présentation

Code CI-SST81E6
Libellé Mathématiques générales
Cycle Ingénieur
Année 1
Semestre 6
Domaine d'enseignement SST - Socle scientifique et technique
Responsable Bur Nicolas (n.bur@estia.fr)
Intervenants Bur Nicolas, Arama Adama, Bakni Michel et André Cyrille
Crédit 1
Heures allouées

Type	en présence (h)	à distance (h)	Total (h)
Cours	4		4
Travaux Dirigés (TD)	8		8
Travaux Pratiques (TP)			
Travail Tutoré (TT)			
Projet			
Total	12		12

Description Dans ce module, une alternative à la diagonalisation matricielle est proposée, permettant d'obtenir une puissance de matrice par une décomposition de celle-ci. Les polynômes sont ensuite utilisés dans la construction des courbes de Bézier ou encore dans le cadre de l'interpolation.

Objectifs Ce module vise à

- fournir les méthodes et outils permettant de décomposer une matrice (projecteurs spectraux, Jordan-Chevalley) pour en déterminer la puissance ou l'exponentielle
- fournir les méthodes et outils permettant de construire un polynôme d'interpolation d'un ensemble de points
- fournir les méthodes et outils permettant de construire une courbe de Bézier associée à un ensemble de points

Modalités d'évaluation

Session 1 Un examen sur table évaluant les aspects théoriques et/ou pratiques.

Session 2 Les étudiants ayant obtenu F lors de la session 1 pourront se présenter à une seconde session dont les modalités seront déterminées par le responsable de module.

Lien aux compétences génériques de la formation Ingénieur [1]

- Compétences liées à l'individu : CI
- Compétences liées à l'entreprise : CE
- Compétences scientifiques et techniques : CST

Acquis d'apprentissage visés À l'issue de ce module, l'étudiant doit être capable

- d'utiliser les projecteurs spectraux
- d'utiliser l'algorithme de Newton pour calculer la décomposition de Jordan-Chevalley d'une matrice carrée complexe sans connaître ses valeurs propres
- d'utiliser la décomposition de Jordan-Chevalley pour calculer les puissances d'une matrice et les exponentielles de matrice
- de résoudre l'équation de Bézout en utilisant l'algorithme d'Euclide étendu pour des entiers ou des polynômes premiers entre eux
- de résoudre des équations de congruence concernant les entiers ou les polynômes (théorème chinois)
- d'utiliser les polynômes d'interpolation de Lagrange et de Hermite

1. Se reporter au *Référentiel de compétences* pour les définitions des compétences

— d'utiliser les courbes de Bézier, l'algorithme de de Casteljau

Contenu Les thèmes abordés dans ce module sont

— arithmétique
— décomposition matricielle : projecteurs spectraux et décomposition de Jordan-Chevalley
— courbes de Bézier
— interpolation polynomiale

Prérequis Compétences du module CI-SST81E5 – Compléments de mathématiques

Bibliographie

Ben Rhouma, Alaeddine (2013), « Autour de la décomposition de Dunford réelle ou complexe. Théorie spectrale et méthodes effectives », https://hal.science/hal-00844141.

Chapitre 1

Topologie algébrique

Sommaire

1 Groupes

Définition 1.1 : groupe

Un *groupe* est un ensemble non vide G, muni d'une *loi de composition interne* $\circ : x, y \mapsto x \circ y$ possédant les propriétés suivantes :

1. pour tout triplet $(x\,;y\,;z)$ d'éléments de G, $x \circ (y \circ z) = (x \circ y) \circ z$ (associativité) ;
2. il existe un *élément neutre* 1 de G tel que pour tout $x \in G$, $x \circ 1 = 1 \circ x = x$;
3. pour tout $x \in G$, il existe un *inverse* $y \in G$ tel que $x \circ y = y \circ x = 1$. Cet élément est noté $y = x^{-1}$.

Un groupe est dit *commutatif* ou *abélien* si la loi de composition interne vérifie en plus que pour toute paire $(x, y) \in G^2$, $x \circ y = y \circ x$.

Propriété 1.1

Soit $(G\,;\circ)$ un groupe et soit $(x, y) \in G^2$.

1. $(x^{-1})^{-1} = x$
2. $(x \circ y)^{-1} = y^{-1} \circ x^{-1}$

La loi de composition interne d'un groupe abélien G sera souvent notée « + ». Dans ce cas, l'élément neutre sera noté « 0 » et l'inverse d'un élément x de G sera noté $-x$ et sera appelé « opposé de x ».

Il est aussi possible d'utiliser la notation multiplicative pour certains groupes (abéliens ou non). On note alors, pour $(x, y) \in G^2$, $x \times y$ ou $x \cdot y$ (ou même xy si aucune confusion n'est à craindre) au lieu de $x \circ y$. Dans ce cas l'élément neutre est noté 1_G (ou plus simplement 1 ou Id s'il s'agit de matrices).

Exemple 1.1

Soient $\mathbb{Z}$ l'ensemble des entiers relatifs, $\mathbb{Q}$ l'ensemble de rationnels, $\mathbb{R}$ l'ensemble des réels et $\mathbb{C}$ l'ensemble des nombres complexes. Alors $(\mathbb{Z}\,;+)$, $(\mathbb{Q}\,;+)$, $(\mathbb{R}\,;+)$, $(\mathbb{C}\,;+)$ sont des groupes abéliens.
$(\mathbb{Q}^\star\,;\times)$, $(\mathbb{R}^\star\,;\times)$, $(\mathbb{C}^\star\,;\times)$ sont également des groupes abéliens pour le produit usuel.

Définition 1.2 : sous-groupe

Soit $(G\,;\circ)$ un groupe. On dit qu'une partie H de G est un *sous-groupe de* G si les deux conditions suivantes sont vérifiées :

1. H est non vide et pour toute paire $(x, y) \in H^2$, $x \circ y \in H$;
2. $(H\,;\circ)$ est un groupe.

La propriété 1.2 est utile pour éviter des vérifications fastidieuses.

Propriété 1.2

Soit $(G\,;\circ)$ un groupe et soit $H \subset G$.
Les deux conditions suivantes sont équivalentes

1. H est un sous-groupe de G.
2. H est non vide, et pour toute paire $(a\,;b)$ d'éléments de H, $a \circ b^{-1} \in H$.

Démonstration 1.1

Il est clair que tout sous-groupe de G *vérifie 1.2-2. Réciproquement soit* $H \subset G$ *vérifiant 1.2-2, et soit* $a \in H$. *On a* $e = a \circ a^{-1} \in H$, *donc* H *contient l'élément neutre* e *de* G. *On a* $b^{-1} = e \circ b^{-1} \in H$ *pour tout* $b \in H$. *Enfin, d'après la propriété 1.1 p. 1 on a* $a \circ b = a \circ (b^{-1})^{-1} \in H$ *pour toute paire* $(a\,;b)$ *d'éléments de* H. □

2 Anneaux

Nous nous intéressons ici aux ensembles munis de deux lois de composition interne.

Définition 1.3 : anneau

Soit $(A\,;+\,;\times)$ un ensemble non vide possédant au moins deux éléments et muni de deux lois de composition internes. On dit que $(A\,;+\,;\times)$ est un *anneau* si les conditions suivantes sont vérifiées :

1. $(A\,;+)$ est un groupe abélien.
2. $x \times (y+z) = x \times y + x \times z$ et $(y+z) \times x = y \times x + z \times x$ pour tout triplet $(x\,;y\,;z)$ d'éléments de A.
3. $x \times (y \times z) = (x \times y) \times z$ pour tout triplet $(x\,;y\,;z)$ d'éléments de A.
4. il existe un élément 1 de A appelé *unité de l'anneau* A tel que $x \times 1 = 1 \times x = x$ pour tout $x \in A$.

Un anneau $(A\,;+\,;\times)$ est commutatif si on a de plus que pour toute paire $(x, y) \in A^2$, $x \times y = y \times x$.
Un élément x de A est *inversible* s'il existe $y \in A$ tel que $x \times y = y \times x = 1$. Cet élément y, appelé inverse de x, est alors noté x^{-1}.

Pour éviter d'alourdir les notations on écrira « l'anneau A » au lieu de « l'anneau $(A\,;+\,;\times)$ » quand aucune confusion n'est à craindre. De même on écrira souvent xy au lieu de $x \times y$.

☛ **Exemple 1.2**

$(\mathbb{Z}\,;+\,;\times)$, $(\mathbb{Q}\,;+\,;\times)$, $(\mathbb{R}\,;+\,;\times)$, $(\mathbb{C}\,;+\,;\times)$ sont des anneaux commutatifs.
$\left(\mathscr{M}_n(\mathbb{R})\,;+\,;\times\right)$ est un anneau non commutatif, où $\mathscr{M}_n(\mathbb{R})$ est l'ensemble des matrices carrées de taille n à coefficients dans $\mathbb{R}$ muni de l'addition et du produit matriciels.

3 Corps

Définition 1.4 : corps

Soit K un ensemble muni de deux lois de composition interne.
On dit que $(K\,;+\,;\times)$ est un *corps* si $(K\,;+\,;\times)$ est un anneau commutatif dans lequel tout élément non nul possède un inverse.

Dans les corps, on peut effectuer des opérations analogues à celles usuelles dans $\mathbb{R}$. Par exemple, on peut noter $\frac{1}{a}$ l'inverse d'un élément non nul a d'un corps quelconque, et l'équation $ax = b$ admet pour unique solution $x = \frac{b}{a}$, si $(a\,;b)$ est une paire d'éléments d'un corps. De même la règle « pour qu'un produit de facteurs soit nul il faut et il suffit que l'un des facteurs soit nul » est valable dans un corps quelconque (mais pas dans un anneau quelconque).

☛ **Exemple 1.3**

$(\mathbb{Q}\,;+\,;\times)$, $(\mathbb{R}\,;+\,;\times)$, $(\mathbb{C}\,;+\,;\times)$ sont des corps.
Par contre $\mathbb{Z}$ n'est pas un corps puisque 1 et -1 sont les seuls éléments inversibles de cet anneau.

4 Anneau $\mathbb{Z}/n\mathbb{Z}$

Définition 1.5 : congruence

Soit un entier $n \geq 2$. Alors tout entier a peut s'écrire $a = \overline{a} + k \times n$, avec $\overline{a} \in \mathbb{N}$ le reste de la division entière de a par n tel que $0 \leq \overline{a} \leq n-1$. Ceci revient à dire que a est *congru* à $\overline{a}$ modulo n et se note $a \equiv \overline{a}\,[n]$. Il importe peu de connaître la valeur de k : seul compte le reste $\overline{a}$.

On note $\mathbb{Z}/n\mathbb{Z}$ l'ensemble de tous les restes (entiers) possibles obtenus par la division des entiers par n. Alors $\mathbb{Z}/n\mathbb{Z} = [k \in \mathbb{N}](0 \leq k \leq n-1)$. De plus, cet ensemble muni des lois de compositions internes d'addition et de multiplication forme un anneau commutatif.

Pour fixer les idées, les tableaux 1.1 et 1.2 donnent les tables d'addition et de multiplication pour les anneaux $\mathbb{Z}/n\mathbb{Z}$, pour $2 \leq n \leq 4$.

Tableau 1.1 – Tables d'addition de $\mathbb{Z}/2\mathbb{Z}$, $\mathbb{Z}/3\mathbb{Z}$ et $\mathbb{Z}/4\mathbb{Z}$

$+$	$\overline{0}$	$\overline{1}$
$\overline{0}$	$\overline{0}$	$\overline{1}$
$\overline{1}$	$\overline{1}$	$\overline{0}$

$+$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{0}$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{1}$	$\overline{1}$	$\overline{2}$	$\overline{0}$
$\overline{2}$	$\overline{2}$	$\overline{0}$	$\overline{1}$

$+$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{0}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{1}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{0}$
$\overline{2}$	$\overline{2}$	$\overline{3}$	$\overline{0}$	$\overline{1}$
$\overline{3}$	$\overline{3}$	$\overline{0}$	$\overline{1}$	$\overline{2}$

Tableau 1.2 – Tables de multiplication de $\mathbb{Z}/2\mathbb{Z}$, $\mathbb{Z}/3\mathbb{Z}$ et $\mathbb{Z}/4\mathbb{Z}$

$\times$	$\overline{0}$	$\overline{1}$
$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$

$\times$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{2}$	$\overline{0}$	$\overline{2}$	$\overline{1}$

$\times$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{2}$	$\overline{0}$	$\overline{2}$	$\overline{0}$	$\overline{2}$
$\overline{3}$	$\overline{0}$	$\overline{3}$	$\overline{2}$	$\overline{1}$

Le tableau 1.2 permet de montrer que $\mathbb{Z}/2\mathbb{Z}$ et $\mathbb{Z}/3\mathbb{Z}$ sont des corps, mais pas $\mathbb{Z}/4\mathbb{Z}$. En effet, les deux premières tables montrent que tous les éléments possèdent un inverse. Ceci n'est pas vrai dans la troisième puisque $\overline{2}$ n'est pas inversible.

La librairie sympy permet d'effectuer des calculs dans ces anneaux à l'aide de *Python*.

Code 1.1 : Calculs dans $\mathbb{Z}/n\mathbb{Z}$ avec *Python*

```
>>> import sympy as sp # import de la librairie

>>> sp.Mod(3*3, 4) # calcul de 3×3 dans Z/4Z
1
>>> sp.mod_inverse(3, 4) # inverse de 3 dans (Z/4Z)
3
>>> sp.mod_inverse(2, 4) # inverse de 2 dans (Z/4Z)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/nbur/.local/lib/python3.7/site-packages/sympy/core/numbers.py",
   ↪  line 551, in mod_inverse
    raise ValueError('inverse of %s (mod %s) does not exist' % (a, m))
ValueError: inverse of 2 (mod 4) does not exist
```

Théorème 1.1

L'anneau commutatif $(\mathbb{Z}/n\mathbb{Z}\,;+\,;\times)$ est un corps si et seulement si n est premier.

Et nous avons besoin d'un peu d'arithmétique pour poursuivre.

Chapitre 2

Arithmétique

Sommaire

Quelques résultats classiques d'arithmétique sont ici présentés, sans démonstration. Ils seront repris par extension sur les polynômes.

1 Division euclidienne

Théorème 2.1 : division euclidienne dans $\mathbb{Z}$

Soit $a \in \mathbb{Z}$ et soit $b \in \mathbb{N}^\star$. Alors il existe une unique paire $(q, r) \in \mathbb{Z}^2$ telle que
1. $a = bq + r$
2. $0 \leq r \leq b - 1$

Ce résultat, souvent appelé « division euclidienne dans $\mathbb{Z}$ », a été vu en primaire pour $a > 0$ et l'extension aux entiers négatifs est évidente. L'unicité provient du fait que si $r - r' = b(q' - q)$, avec $q \neq q'$, alors $|r - r'| \geq b$, tandis que $|r - r'| \leq b - 1$ si r et r' vérifient le point 2.1-2 du théorème 2.1.

Exemple 2.1

Par exemple, avec $a = 132$ et $b = 9$, on a $q = 14$ et $r = 6$.

```
>>> divmod(132, 9)
(14, 6)
```

Définition 2.1 : diviseur et multiple

Soient m et n deux entiers relatifs. On dit que m *divise* n ou que m est *un diviseur* de n s'il existe $k \in \mathbb{Z}$ tel que $n = km$. Dans ce cas on dit également que n est *un multiple* de m.

Théorème 2.2

Soit $n \in \mathbb{N}$ et soit $(a_1, \ldots, a_n)$ une famille d'entiers naturels non tous nuls.
Alors il existe un unique entier positif d vérifiant pour $i \in [\![1\,;n]\!]$
1. d divise a_i ;
2. si un entier $\tilde{d}$ divise a_i, alors $\tilde{d}$ divise aussi d.

Cet entier d est appelé *plus grand diviseur commun* (noté dans la suite pgdc) de la famille $(a_1, \ldots, a_n)$.

Soit $n \in \mathbb{N}$. Pour calculer le pgdc d'une famille $(a_1, \ldots, a_n)$ on peut procéder par récurrence finie : si pour $k \in \mathbb{N}$, $k < n$, on note b_k le pgdc de $(a_1, \ldots, a_k)$ alors b_{k+1} est le pgdc de a_{k+1} et b_k.
On peut aussi remarquer que le pgdc de $(a_1, \ldots, a_n)$ est égal à celui de $(|a_1|, \ldots, |a_n|)$.

Il suffit donc de savoir calculer le pgdc de deux entiers positifs a et b, ce qui se fait par l'*algorithme d'Euclide*. Celui-ci consiste à faire des divisions successives.

Soient a et b deux entiers positifs, avec $a \geq b$. Leur pgdc d s'obtient à l'aide de l'algorithme 2.1.

Algorithme 2.1 : d'Euclide pour le calcul du pgdc de deux entiers

Entrée : $(a, b) \in \mathbb{N}^2$, $a \geq b$
Sortie : $d = \text{pgdc}(a\,;b)$

$k = 0$
$a_k = a$ et $b_k = b$
r_k est le reste **de** la division euclidienne **de** a_k par b_k : $a_k = b_k q_k + r_k$
tant que $r_k > 0$, **faire**
  $a_{k+1} = b_k$
  $b_{k+1} = r_k$
  $k = k + 1$
  r_k est le reste **de** la division euclidienne **de** a_k par b_k
fin tant que
retourner $d = b_k$ (tel **que** $r_k = 0$)

Autrement dit, « le pgdc est égal au dernier reste non nul dans l'algorithme d'Euclide. » Comme $r_k > r_{k+1}$ pour tout k, avec les notations de l'algorithme 2.1, cet algorithme s'arrête avec pour une valeur de $k \leq b - 1$. Le fait que le pgdc de a et b est bien égal au dernier reste non nul provient du fait que si u et v sont deux entiers positifs, alors le pgdc de u et de v est égal au pgdc de v et du reste de la division de u par v. On a donc $\text{pgdc}(a\,;b) = \text{pgdc}(b\,;r_0) = \text{pgdc}(r_0\,;r_1) = \ldots = \text{pgdc}(r_{k-1}\,;0) = r_k$.

Exemple 2.2

Déterminons le pgdc de 55 et de 132.
L'algorithme d'Euclide permet de construire le tableau suivant.

k	$a_k = b_k \times q_k + r_k$
0	$132 = 55 \times 2 + 22$
1	$55 = 22 \times 2 + 11$
2	$22 = 11 \times 2 + 0$

Ainsi, $\text{pgdc}(132\,;55) = 11$.

Théorème 2.3 : de Bézout

Soit $(a_1, \ldots, a_n)$ une famille finie d'entiers naturels non tous nuls et soit d le pgdc de cette famille. Alors il existe une famille $(u_1, \ldots, u_n)$ d'éléments de $\mathbb{Z}$ vérifiant $a_1u_1 + \ldots + a_nu_n = d$.
Plus généralement l'équation $a_1v_1 + \ldots + a_nv_n = p$ admet une solution $(v_1, \ldots, v_n)$ dans $\mathbb{Z}^n$ si et seulement si p est un multiple de d.

Soient deux entiers a et b. On dispose d'une méthode effective pour calculer deux entiers u et v tels que $au + bv = d$, d désignant le pgdc de a et de b. Il suffit de « remonter l'algorithme d'Euclide ». On peut en effet utiliser l'avant dernière ligne (donc pour l'indice $k - 1$) de l'algorithme pour exprimer $d = r_k$ en fonction de r_{k-1} et de r_{k-2}. En utilisant la ligne $k - 2$, on exprime r_{k-1} en fonction de r_{k-2} et r_{k-3} et en substituant on exprime d en fonction de r_{k-2} et de r_{k-3}. En itérant ce procédé ligne par ligne vers le haut on obtient les coefficients u et v cherchés.

Il existe en fait une méthode rapide pour calculer u et v, en faisant des calculs intermédiaires pendant le déroulement de l'algorithme d'Euclide. L'idée est la suivante : si $r_k = au_k + bv_k$, en remontant l'algorithme on obtient $r_k = r_{k-2} - r_{k-1}q_k$. Donc on a $r_k = (au_{k-2} + bv_{k-2}) - (au_{k-1} + bv_{k-1})q_k$, soit $r_k = a(u_{k-2} - u_{k-1})q_k + b(v_{k-2} - v_{k-1})q_k$. Par identification, on obtient les relations de récurrence

$$\begin{cases} u_k = u_{k-2} - u_{k-1}q_k & (2.1) \\ v_k = v_{k-2} - v_{k-1}q_k & (2.2) \end{cases}$$

Avec les initialisations $u_0 = 1, v_0 = 0$ et $u_1 = 0, v_1 = 1$, il est alors aisé d'étendre l'algorithme d'Euclide pour trouver les valeurs de u et de v.

Algorithme 2.2 : d'Euclide étendu

Entrée : $(a,b) \in \mathbb{N}^2$, $a \geq b$
Sortie : $d = \text{pgdc}(a\,;b)$, $(u,v) \in \mathbb{Z}^2$ tel que $au + bv = d$

$k = 0$
$a_k = a$ et $b_k = b$
$u_{k-2} = 1$ et $u_{k-1} = 0$
$v_{k-2} = 0$ et $v_{k-1} = 1$
$(q_k\,;r_k)$ tel que $a_k = b_k q_k + r_k$
tant que $r_k > 0$, **faire**
  $a_{k+1} = b_k$
  $b_{k+1} = r_k$
  $u_k = -q_k u_{k-1} + u_{k-2}$
  $v_k = -q_k v_{k-1} + v_{k-2}$
  $k = k + 1$
  $(q_k\,;r_k)$ tel que $a_k = b_k q_k + r_k$
fin tant que
retourner $d = b_k$, $u = u_{k-1}$ et $v = v_{k-1}$ (avec k tel que $r_k = 0$)

Exemple 2.3

Déterminons u et v tels que $132u + 55v = \text{pgdc}(132\,;55)$.
L'algorithme d'Euclide étendu permet de construire le tableau suivant.

k	$a_k = b_k \times q_k + r_k$	u_k	v_k
		1	0
		0	1
0	$132 = 55 \times 2 + 22$	1	-2
1	$55 = 22 \times 2 + 11$	-2	5
2	$22 = 11 \times 2 + 0$		

Ainsi, $\text{pgdc}(132\,;55) = 11$ et $132 \times (-2) + 55 \times 5 = 11$.

2 Applications

Définition 2.2 : nombre premier

Soit un entier $p \geq 2$.
On dit que p est *premier* si ses seuls diviseurs positifs sont 1 et p.

Pour dresser la liste des nombres premiers on peut utiliser le *crible d'Ératosthène* que nous mettons en œuvre pour déterminer les nombres premiers inférieurs ou égaux à 30.

On écrit la liste : 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
- On garde 2 et on supprime tous ses multiples : 2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29.
- On garde 3 et on supprime tous ses multiples : 2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29.
- On garde 5 et on supprime tous ses multiples : 2, 3, 5, 7, 11, 13, 17, 19, 23, 29.

La librairie sympy permet d'effectuer des calculs dans ces anneaux à l'aide de *Python*.

Code 2.1 : Calculs dans $\mathbb{Z}/n\mathbb{Z}$ avec *Python*

```
>> a = 5; b = 24; c = 19
>> print(f"{a} est-il premier ? {sp.isprime(a)}")
5 est-il premier ? True
>> print(f"{b} est-il premier ? {sp.isprime(b)}")
24 est-il premier ? False
>> print(f"Le {a}e nombre premier est {sp.prime(a)}")
Le 5e nombre premier est 11
>> print(f"Les nombres premiers inférieurs à {c} sont {list(sp.primerange(c))}")
```

```
Les nombres premiers inférieurs à 19 sont [2, 3, 5, 7, 11, 13, 17]
>>> print(f"Les nombres premiers jusqu'au {a}e sont
 ↪ {list(sp.primerange(sp.prime(a) + 1))}")
Les nombres premiers jusqu'au 5e sont [2, 3, 5, 7, 11]
```

Définition 2.3 : nombres premiers entre eux

Soient a et b deux entiers relatifs.
On dit que a et b sont *premiers entre eux* si leur pgdc est égal à 1.

Il est clair que si p est premier, et si q n'est pas un multiple de p, alors p et q sont premiers entre eux.
Donnons maintenant deux conséquences importantes du théorème 2.3 (de Bézout) p. 6.

Théorème 2.4 : de Gauss

Soient a, b, et c trois entiers relatifs non nuls.
Si a divise bc, et si a est premier avec b, alors a divise c.

Démonstration 2.1

Comme a et b sont premiers entre eux, il existe $(u,v) \in \mathbb{Z}^2$ tel que $au+bv=1$. Donc $c = c(au+bv)$. Comme a divise b, a divise aussi bc et il existe $w \in \mathbb{Z}$ tel que $bc = aw$. Alors $c = a(uc+vw)$, ce qui montre que a divise c.

□

Corollaire 2.1

Soient a et b deux entiers relatifs non nuls et soit $d = \text{pgdc}(a\,;b)$. Soit $\mathcal{S} = \left\{(u,v) \in \mathbb{Z}^2 \mid au+bv=0\right\}$.
Alors $\mathcal{S} = \left\{\left(-n\tilde{b}, n\tilde{a}\right) \in \mathbb{Z}^2 \mid n \in \mathbb{Z}\right\}$, avec $\tilde{a} = \frac{a}{d}$ et $\tilde{b} = \frac{b}{d}$.

Démonstration 2.2

Il résulte du théorème 2.3 (de Bézout) p. 6 que $\tilde{a}$ et $\tilde{b}$ sont premiers entre eux, et que $\mathcal{S} = \left\{(u,v) \in \mathbb{Z}^2 \mid \tilde{a}u+\tilde{b}v=0\right\}$. Il est clair que si $u = -n\tilde{b}$ et si $v = n\tilde{a}$ alors $(u,v) \in \mathcal{S}^2$. Réciproquement si $\tilde{a}u+\tilde{b}v=0$ alors $\tilde{a}$ divise $\tilde{b}v$, donc $\tilde{a}$ divise v d'après le théorème 2.4 (de Gauss). Donc il existe $n \in \mathbb{Z}$ tel que $v = n\tilde{a}$. On a alors $\tilde{a}u = -\tilde{b}n\tilde{a}$ et par conséquent $u = -n\tilde{b}$.

□

On a alors le résultat suivant concernant l'équation de Bézout.

Corollaire 2.2 : équation de Bézout

Soient a et b deux entiers positifs premiers entre eux.
Il existe alors une unique paire $(u_0, v_0) \in \mathbb{Z}^2$ vérifiant les deux conditions suivantes

1. $au_0 + bv_0 = 1$;
2. $0 < u_0 < b$.

De plus dans ce cas on a $v_0 < 0$ et $|v| < a$.
D'autre part les solutions entières de l'équation $au+bv=1$ sont données par les couples de la forme $u = u_0 - nb$ et $v = v_0 + na$, avec $n \in \mathbb{Z}$.

Ces résultats ont diverses applications pratiques. On peut par exemple les utiliser pour déterminer les points à coordonnées entières d'une droite dont les coefficients de l'équation sont entiers.

Exemple 2.4

Soit la droite $(\mathscr{D})$ d'équation $55x + 132y = 13$.
Pour que des points à coordonnées entières existent sur cette droite, il faudrait que 13 soit un multiple du pgdc de 55 et 132. Or pgdc (55 ; 132) = 11. Donc cette droite n'a pas de points à coordonnées entières.
Soit $\left(\mathscr{D}'\right)$ d'équation $55x + 132y = 22$.
Ici 22 est un multiple de 11, donc l'équation $55x + 132y = 22$ a des solutions entières.

Considérons dans un premier temps l'équation homogène associée : $55x + 132y = 0$. D'après le corollaire 2.1 p. 8, l'ensemble des solutions est $\mathcal{S}_H = \left\{\left(-n\frac{132}{11}, n\frac{55}{11}\right) \in \mathbb{Z}^2 \mid n \in \mathbb{Z}\right\} = \left\{(-12n, 5n) \in \mathbb{Z}^2 \mid n \in \mathbb{Z}\right\}$.
En utilisant l'algorithme d'Euclide étendu, on a $55 \times 5 - 2 \times 132 = 11$. Alors une solution particulière entière est $(x_0\,;y_0) = (10\,;-4)$.
Ainsi tous les points à coordonnées entières de $\left(\mathscr{D}'\right)$ sont tels que $(x\,;y) = (10 - 12n\,;-4 + 5n)$.
On obtient un résultat similaire en utilisant le corollaire 2.2 (équation de Bézout). Comme $\text{pgdc}\,(55\,;132) = 11$, l'équation de $\left(\mathscr{D}'\right)$ s'écrit aussi $5x + 12y = 2$, avec 5 et 12 premiers entre eux. Il existe donc un couple $(u_0, v_0) \in \mathbb{Z}^2$ tel que $55u_0 + 132v_0 = 1$. Par l'algorithme d'Euclide étendu, on obtient $(u_0\,;v_0)\,(5\,;-2)$. En multipliant par 2 pour retrouver l'équation de $\left(\mathscr{D}'\right)$, on a donc une solution particulière $(x_0\,;y_0) = (10\,;-4)$. L'ensemble des solutions entières de l'équation de $\left(\mathscr{D}'\right)$ est alors donné pour tout $n \in \mathbb{Z}$ par $(x\,;y) = \left(x_0 - n \times 12\,;y_0 + n \times 5\right) = (10 - 12n\,;-4 + 5n)$. Comme $n \in \mathbb{Z}$, ce résultat est cohérent avec celui obtenu précédemment.

```
>> a = 132; b = 55
>> u, v, pgdc = sp.gcdex(a, b)
>> print(f"\nUne solution particulière de l'équation de Bézout {a} × u + {b} × v =
 ↪  {pgdc} est\nu={sp.pretty(u)}, v={sp.pretty(v)}")

Une solution particulière de l'équation de Bézout 132 × u + 55 × v = 11 est
u=-2, v=5
```

Voici une variante du théorème de Gauss.

Théorème 2.5 : de Gauss

Soit $(a,\ b_1,\ \ldots,\ b_n)$ une famille d'entiers non nuls.
Si pour $k \in [\![1\,;n]\!]$, a est premier avec b_k, alors a est premier avec le produit $\prod_{k=1}^{n} b_k = b_1 \ldots b_k$.

Corollaire 2.3

Soit $(a_1,\ \ldots,\ a_n)$ une famille d'entiers premiers entre eux deux à deux et soit $x \in \mathbb{Z}$.
Si pour tout $k \in [\![1\,;n]\!]$, x est divisible par a_k, alors x est divisible par le produit $\prod_{k=1}^{n} a_k$.

Démonstration 2.3

Soit $n \in \mathbb{N}$, soit une famille $(a_1,\ \ldots,\ a_n)$ d'entiers premiers entre eux deux à deux et soit $x \in \mathbb{Z}$. Procédons par récurrence.

- *Si $n = 1$, il n'y a rien à démontrer.*
- *Supposons maintenant que le résultat est vrai pour un certain rang $n - 1 \geq 2$. On a par hypothèse de récurrence que x est divisible par le produit $\prod_{k=1}^{n-1} a_k$, donc il existe $y \in \mathbb{Z}$ tel que $x = a_1 \ldots a_{n-1} y$. Il résulte du théorème 2.5 ci-dessus que a_n est premier avec $\prod_{k=1}^{n-1} a_k$, et on déduit alors du théorème 2.4 que a_n divise y. Donc x est divisible par $\prod_{k=1}^{n} a_k$, et la propriété est vraie au rang n.*
- *Le résultat est donc démontré par récurrence.*

□

3 Théorème chinois

La définition 1.5 (congruence) p. 3 peut être considérée du point de vue de la divisibilité. Soit un entier $n \geq 2$. On dira que deux entiers relatifs a et b sont *congrus modulo n*, et on écrira $a \equiv b\,[n]$ ou $b \equiv a\,[n]$ quand $a - b$ est divisible par n. On vérifie facilement que si $a \equiv a'\,[n]$ et $b \equiv b'\,[n]$, alors $a + b \equiv a' + b'\,[n]$ et $ab \equiv a'b'\,[n]$.

Le théorème suivant est dû à un mathématicien chinois anonyme.

Théorème 2.6 : chinois

Soit $(m_1, \ldots, m_n)$ une famille d'entiers positifs et premiers deux à deux.
Alors pour toute famille $(q_1, \ldots, q_n)$ dans $\mathbb{Z}^n$, le système d'équations de congruence

$$\begin{cases} x \equiv q_1 \,[m_1] \\ \ldots \\ x \equiv q_n \,[m_n] \end{cases} \tag{2.3}$$

possède des solutions dans $\mathbb{Z}$.
De plus si x_0 est une solution particulière, alors la solution générale du système est donnée par la formule

$$x_k = x_0 + k \prod_{i=1}^{n} m_n, k \in \mathbb{Z}. \tag{2.4}$$

Démonstration 2.4

Notons que si x_0 est une solution particulière de (2.3), alors $x \in \mathbb{Z}$ est solution du système si et seulement si $x - x_0 \equiv 0 \,[m_i]$ pour tout $i \in [\![1\,;n]\!]$.

- *Supposons que $x = x_0 + k \prod_{i=1}^{n} m_i$, avec $k \in \mathbb{Z}$. Alors nécessairement x est solution du système.*
- *Réciproquement, si x est solution du système, alors pour tout $i \in [\![1\,;n]\!]$, $x - x_0$ est divisible par m_i qui sont premiers entre eux deux à deux. Donc, par le corollaire 2.3 p. 9, $x - x_0$ est divisible par le produit $\prod_{i=1}^{n} m_i$ et on a $x = x_0 + k \prod_{i=1}^{n} m_i$, avec $n \in \mathbb{Z}$.*

□

Pour démontrer l'existence d'une solution on procède par récurrence sur n et on est ramené à chaque étape à résoudre dans $\mathbb{Z}^2$ une équation du type $ax + by = c$, avec $(a, b, c) \in \mathbb{Z}^3$, a et b étant premiers entre eux. Ceci donne un moyen effectif de trouver des solutions pour ce type de système d'équations de congruence, que nous décrivons dans l'exemple suivant.

Exemple 2.5

Trouvons $x \in \mathbb{Z}$ vérifiant $\begin{cases} x \equiv 1\,[2] \\ x \equiv -1\,[3] \\ x \equiv 2\,[5] \end{cases}$

Par définition, les solutions de la première équation sont de la forme $x = 1 + 2m$, avec $m \in \mathbb{Z}$, celle de la seconde $x = -1 + 3n$, avec $n \in \mathbb{Z}$. D'où $1 + 2m = -1 + 3n$ qui donne $3n - 2m = 2$. Comme 2 est un multiple de pgdc $(2\,;3) = 1$, cette équation admet des solutions. On trouve aisément la solution $m = n = 2$, ce qui fait que, d'après la première congruence, $x = 1 + 4 = 5$ qui est donc une solution du système formé par les deux premières équations de congruence.

Autrement dit, les solutions du système intermédiaire $\begin{cases} x \equiv 1\,[2] \\ x \equiv -1\,[3] \end{cases}$ sont de la forme $x = 5 + 6p$, avec $p \in \mathbb{Z}$.

On résout à présent le nouveau système $\begin{cases} x \equiv 5\,[6] \\ x \equiv 2\,[5] \end{cases}$, de la même manière que précédemment. Il existe donc $(p, q) \in \mathbb{Z}^2$, $x = 5 + 6p = 2 + 5q$. En reformulant, on a $6p - 5q = -3$. Comme 5 et 6 sont premiers entre eux, on cherche $(u, v) \in \mathbb{Z}^2$ tel que $6u - 5v = 1$. Une solution évidente est $u = v = 1$. Donc on peut prendre $p = q = -3$, ce qui donne une solution particulière du nouveau système $x_1 = -13$.
Alors les solutions du système initial sont données par $x = -13 + 2 \times 3 \times 5 \times n = -13 + 30n$, pour tout $n \in \mathbb{Z}$.

```
>> couples_r_m = [(1, 2), (-1, 3), (2, 5)]
>> sol = sp.ntheory.modular.solve_congruence(*couples_r_m)
>> # on peut aussi coder
>> # sol = sp.ntheory.modular.crt([2, 3, 5], [1, -1, 2])
>> print(f"Soit le système de congruence\n{f'{chr(10)}'.join([f'x congru à {_r}
 ↪  modulo {_m}' for _r, _m in couples_r_m])}\nLes solutions sont alors de la forme
 ↪  x congru à {sol[0]} modulo {sol[1]}")
Soit le système de congruence
x congru à 1 modulo 2
x congru à 2 modulo 3
x congru à 5 modulo 6
Les solutions sont alors de la forme x congru à 5 modulo 30
```

```
x congru à -1 modulo 3
x congru à 2 modulo 5
Les solutions sont alors de la forme x congru à 17 modulo 30
```

La méthode utilisée ci-dessus est valable pour tous les systèmes d'équations de congruence vérifiant les hypothèses du théorème chinois, mais il faut en général utiliser l'algorithme d'Euclide étendu (voir le algorithme 2.2 (d'Euclide étendu) p. 7) pour résoudre les équations du type Bézout rencontrées à chaque étape des calculs.

4 Décomposition en produit de nombres premiers

La notion de pgdc a pour pendant celle de plus petit multiple commun (noté dans la suite ppmc). Nous nous limiterons au cas de deux entiers.

Théorème 2.7 : ppmc

Soient a et b deux entiers relatifs.
Alors il existe un unique entier positif m, appelé ppmc de a et b, possédant les deux propriétés suivantes
1. m est un multiple commun à a et b ;
2. si n est un multiple commun à a et b, alors n est un multiple de m.

De plus si on note pgdc $(a\,;b) = a \wedge b$ et ppmc $(b) = a \vee b$, on a la relation $(a \wedge b)(a \vee b) = ab$ (avec la convention $0 \wedge 0 = 0$).

Exemple 2.6

On a vu que pgdc $(55\,;132) = 11$. Alors ppmc $(55\,;132) = \frac{55\times132}{11} = 660$.

```
>> a = 55; b = 132
>> pgdc = sp.gcd(a, b)
>> ppmc = sp.lcm(a, b)
>> print(f"Le pgdc de {a} et de {b} est {pgdc}")
Le pgdc de 55 et de 132 est 11
>> print(f"Le ppmc de {a} et de {b} est {ppmc}")
Le ppmc de 55 et de 132 est 660
>> print(f"On vérifie pgdc(a, b) × ppmc(a, b) = a × b : {a*b==ppmc*pgdc}")
On vérifie pgdc(a, b) × ppmc(a, b) = a × b : True
```

Théorème 2.8 : décomposition en facteurs premiers

Soit un entier $a \geq 2$.
Alors il existe une unique suite finie croissante $(p_1,\ \ldots,\ p_k)$ de nombres premiers et une unique suite $(n_1,\ \ldots,\ n_k)$ d'entiers positifs telles que $a = \prod_{i=1}^{k} p_i^{n_i}$.
Cette formule est appelée *décomposition en facteurs premiers* de a.

En utilisant la décomposition en facteurs premiers de deux entiers, on obtient le résultat suivant.

Propriété 2.1

Soient deux entiers $a \geq 2$ et $b \geq 2$.
- Le pgdc de a et de b est égal au produit des diviseurs premiers communs à a et b, affectés du plus petit de leurs exposants dans les décompositions en facteurs premiers de a et de b.
- Le ppmc de a et de b est égal au produit des diviseurs premiers de a ou de b, affectés du plus grand de leurs exposants dans les décompositions en facteurs premiers de a et b.

Exemple 2.7

Les décompositions en facteurs premiers de 132 et 110 sont $132 = 2^2 \times 3 \times 11$ et $110 = 2 \times 5 \times 11$.
Les seuls diviseurs premiers communs à ces deux nombres sont 2 (exposant 2 pour 132 et 1 pour 110) et 11 (exposant 1 pour 132 et 1 pour 110) . Donc $110 \wedge 132 = 2 \times 11 = 22$, et $110 \vee 132 = 2^2 \times 3 \times 5 \times 11 = 660$.

```
>> import sympy as sp
```

```
>>> a = 132; b = 110
>>> sp.factorint(a)
{2: 2, 3: 1, 11: 1}
>>> sp.factorint(b)
{2: 1, 5: 1, 11: 1}
```

Cette deuxième méthode de calcul du pgdc est à première vue plus simple que l'algorithme d'Euclide, mais pour les grands nombres le coût du calcul de la décomposition en facteurs premiers est élevé, et les logiciels de calcul utilisent des variantes de l'algorithme d'Euclide.

Concluons ce chapitre par la démonstration du théorème 1.1 p. 3

Démonstration 2.5 : du théorème 1.1 p. 3

Si p n'est pas premier il existe un diviseur d de p tel que $1 < d < p$. On a alors $p = dq$ avec $1 < q < p$. Donc $\overline{p} = \overline{0} = \overline{d}\,\overline{q}$, avec $\overline{d} \neq 0$, $\overline{q} \neq 0$, et $\mathbb{Z}/p\mathbb{Z}$ n'est pas un corps.
Considérons maintenant que p est premier. Soit $\overline{a}$ un élément non nul de $\mathbb{Z}/p\mathbb{Z}$. On a $1 \leq \overline{a} < p$. Donc $\overline{a}$ et p sont premiers entre eux. D'après le théorème 2.3 (de Bézout) p. 6, il existe $(u, v) \in \mathbb{Z}^2$ tel que $\overline{a}u + pv = 1$ et $0 \leq \overline{a} \leq p - 1$ et le reste de la division de $\overline{a}u$ par p est égal à 1. Donc $\overline{\overline{a}u} = \overline{1}$ et tout élément non nul de $\mathbb{Z}/p\mathbb{Z}$ est inversible, ce qui prouve que $\mathbb{Z}/p\mathbb{Z}$ est un corps.

□

Chapitre 3

Polynômes

Sommaire

1 Polynômes sur un corps $\mathbb{K}[x]$

Commençons par définir l'anneau $\mathbb{K}[x]$ des polynômes sur un corps $\mathbb{K}$.

Définition 3.1 : anneau des polynômes

Soit $\mathbb{K}$ un corps. Un *polynôme* à coefficients dans $\mathbb{K}$ est une *suite* $(a_k)_{k\leq 0}$ d'éléments de $\mathbb{K}$, nulle à partir d'un certain rang. L'ensemble des polynômes à coefficients dans $\mathbb{K}$ est noté $\mathbb{K}[x]$. On définit la somme et le produit de deux polynômes $(a_k)_{k\leq 0}$ et $(b_k)_{k\leq 0}$ ainsi que le produit d'un scalaire $\lambda \in \mathbb{K}$ par un polynôme $(a_k)_{k\leq 0}$ par les formules

- $(a_k)_{k\leq 0} + (b_k)_{k\leq 0} = (a_k + b_k)_{k\leq 0}$;
- $(a_k)_{k\leq 0} \times (b_k)_{k\leq 0} = \left(\sum_{j=0}^{k} a_j b_{k-j}\right)_{k\leq 0}$;
- $\lambda \times (a_k)_{k\leq 0} = (\lambda a_k)_{k\leq 0}$.

Pour tout $x \in \mathbb{K}$, on a $P(x) \in \mathbb{K}$, avec $P(x) = a_0 + a_1 x + \ldots + a_k x^k + \ldots$.

⚠ Il ne faut pas confondre le polynôme $P \in \mathbb{K}[x]$ avec son évaluation en un point $x \in \mathbb{K}$ qui est $P(x) \in \mathbb{K}$.

Des vérifications de routine montrent que $(\mathbb{K}[x]\ ;+\ ;\times)$ est un anneau commutatif. L'élément neutre de l'addition est le polynôme nul $0_{\mathbb{K}[x]} = (0,\ 0,\ 0,\ \ldots)$ et l'élément neutre de la multiplication est le polynôme $1_{\mathbb{K}[x]} = (1,\ 0,\ 0,\ \ldots)$.

Définition 3.2 : degré

Soit $P = (a_k)_{k\leq 0}$ un polynôme non nul.
On appelle *degré* de P, et on note $\deg(P)$, le plus grand entier $n \geq 0$ tel que $a_n \neq 0$.
On pose par convention $\deg\left(0_{\mathbb{K}[x]}\right) = -\infty$.
On appelle *terme dominant* le terme $a_n x^n$ de ce polynôme P. Le coefficient a_n est appelé *coefficient dominant*.

! On notera $\mathbb{K}_n[x]$ l'ensemble de polynôme à coefficients dans l'anneau $\mathbb{K}$ et de degré *au plus* $n \in \mathbb{N}$.

Soit $P = (a_k)_{k\geq 0} \in \mathbb{K}[x]$, et soit $n \in \mathbb{N}$ tel que $n \geq \deg(P)$. Par un abus d'écriture on écrira 0 au lieu de $0_{\mathbb{K}[x]}$ et 1 au lieu de $1_{\mathbb{K}[x]}$. On obtient, avec la convention $x^0 = 1$, l'écriture usuelle $P(x) = a_0 + \ldots + a_n x^n = \sum_{k=0}^{n} a_k x^k$ pour $P \in \mathbb{K}[x]$, $n \geq \deg(P)$.

Propriété 3.1

Soit $(P, Q, R) \in \mathbb{K}[x]^3$ une famille de polynômes non nuls.

1. $\deg(P+Q) \leq \max(\deg(P), \deg(Q))$, et $\deg(P+Q) = \max(\deg(P), \deg(Q))$ si $\deg(P) \neq \deg(Q)$.
2. $\deg(PQ) = \deg(P) + \deg(Q)$
3. Si $PQ = PR$, alors $Q = R$.

```
>>> import sympy as sp # import de la librairie
>>> sp.var("x") # déclaration de la variable symbolique x
x

>>> # définition de deux polynômes
>>> P = sp.Poly(x**3 + 1)
>>> Q = sp.Poly(x**2 + 2*x + 1) # attention à ne pas oublier les opérateurs

>>> p = P.degree() # obtention du degré
>>> q = Q.degree()
>>> print(f"Le degré du polynômes P\n{sp.pretty(P.as_expr())}\nvaut {p}")
Le degré du polynômes P
 3
x  + 1
vaut 3
>>> print(f"Le degré du polynômes Q\n{sp.pretty(Q.as_expr())}\nvaut {q}")
Le degré du polynômes Q
 2
x  + 2*x + 1
vaut 2

>>> print("\n** Opérations entre polynômes**")

** Opérations entre polynômes**
>>> S = P + Q # somme
>>> T = P * Q # produit
>>> U = P**3 # puissance
>>> print(f"Le degré du polynômes P+Q\n{sp.pretty(S.as_expr())}\nvaut {S.degree()}")
Le degré du polynômes P+Q
 3    2
x  + x  + 2*x + 2
vaut 3
>>> assert S.degree() <= max(p, q), "deg(P + Q) > max(deg(P), deg(Q))" # vérification
 ↪  sur le degré
>>> print(f"Le degré du polynômes P*Q\n{sp.pretty(T.as_expr())}\nvaut {T.degree()}")
Le degré du polynômes P*Q
 5      4    3    2
x  + 2*x  + x  + x  + 2*x + 1
vaut 5
>>> assert T.degree() == p+q, "deg(PQ) != deg(P) + deg(Q)" # vérification sur le degré
>>> print(f"Le degré du polynômes P**3\n{sp.pretty(U.as_expr())}\nvaut {U.degree()}")
Le degré du polynômes P**3
 9      6      3
x  + 3*x  + 3*x  + 1
vaut 9
>>> assert U.degree() == 3*p, "deg(P**n) != n*deg(P)" # vérification sur le degré
```

2 Division euclidienne de polynômes

Le théorème suivant est une transposition du théorème 2.1 (division euclidienne dans $\mathbb{Z}$) p. 5 pour des polynômes.

Théorème 3.1 : division euclidienne dans $\mathbb{K}[x]$

Soient $P \in \mathbb{K}[x]$ et $B \in \mathbb{K}[x]$, avec $B \neq 0$.
Il existe alors une unique paire de polynômes $(Q\,;R)$ vérifiant les deux relations suivantes

1. $P = BQ + R$
2. $\deg(R) < \deg(B)$.

Démonstration 3.1

Soit $B \in \mathbb{K}[x]$ tel que $b \neq 0$. On note $m = \deg(B)$. Soit $P \in \mathbb{K}[x]$. On note $n = \deg(P) \in \mathbb{N} + \{+\infty\}$.

- *Il est clair que si $m = 0$, B est un polynôme constant et on a $Q = \frac{P}{B}$ et $R = 0$.*
- *Supposons maintenant $m > 0$. Alors $B(x) = \sum_{i=0}^{n} b_i x^i$, avec $b_n \neq 0$, et procédons par récurrence sur n.*
  - *Si $n = 0$, alors on prend $Q = 0$ et $R = P$: on a bien l'existence d'une paire $(Q\,;R)$ telle que $P = BQ + R$ avec $\deg(R) < n$.*
  - *Soit maintenant $n \in \mathbb{N}^\star$. On suppose que pour tout $k \in [\![0\,;n]\!]$, il existe une paire $(Q_k\,;R_k)$ telle que le polynôme P_k de degré k s'écrit $P_k = BQ_k + R_k$, avec $\deg(R_k) < n$. Soit $P_{n+1} \in \mathbb{K}[x]$ de degré $n+1$: $P_{n+1}(x) = \sum_{i=0}^{n+1} a_i x^i$, avec $a_{n+1} \neq 0$. Il est clair que $\deg\left(P_{n+1}(x) - a_{n+1}x^{n+1} \times \frac{B}{b_m x^m}\right) \leq n$, puisque de la sorte on annule le terme de degré $n+1$. Ce nouveau polynôme étant de degré inférieur ou égal à n, par hypothèse, il existe une paire $(Q\,;R)$ telle que $P_{n+1}(x) - a_{n+1}x^{n+1} \times \frac{B}{b_m x^m} = B(x)Q(x) + R(x)$, avec $\deg(R) < m$. Par suite $P_{n+1}(x) = B(x)\left(Q(x) + \frac{a_{n+1}x^{n+1-m}}{b_m}\right) + R(x)$ est la décomposition cherchée.*
  - *Donc tout polynôme P se décompose en $BQ + R$, avec $\deg(R) < m$.*

Supposons à présent que la décomposition n'est pas unique : Soit $P \in \mathbb{K}[x]$ et soient $(Q\,;R)$ et $(\tilde{Q}\,;\tilde{R})$ deux paires de polynômes vérifiant (3.1-1) et (3.1-2). On a donc $R - \tilde{R} = B(\tilde{Q} - Q)$. Si $\tilde{Q} \neq Q$ on a $R - \tilde{R} \neq 0$ et $\deg(B) > \max(\deg(R), \deg(\tilde{R})) \geq \deg(R - \tilde{R}) = \deg(B) + \deg(\tilde{Q} - Q) \geq \deg(B)$, ce qui est absurde. Donc $Q = \tilde{Q}$ et $R = \tilde{R}$.

□

Notons que la démonstration formelle ci-dessus donne un algorithme explicite pour calculer Q et R.

Exemple 3.1

Nous illustrons cet algorithme dans le cas où $P = x^3 + 1$ et $B = x^2 + 2x + 1$.

$$\begin{array}{r|l} x^3 + 1 & x^2 + 2x + 1 \\ \hline -x^3 - 2x^2 - x & x - 2 \\ -2x^2 - x + 1 & \\ 2x^2 + 4x + 2 & \\ \hline 3x + 3 & \end{array}$$

On obtient donc la décomposition $x^3 + 1 = (x^2 + 2x + 1)(x - 2) + 3x + 3$. Dans ce cas $Q(x) = x - 2$ et $R(x) = 3x + 3$.

```
>>> import sympy as sp # import de la librairie
>>> sp.var("x") # déclaration de la variable symbolique x
x

>>> # définition de deux polynômes
>>> P = sp.Poly(x**3 + 1)
>>> Q = sp.Poly(x**2 + 2*x + 1) # attention à ne pas oublier les opérateurs
>>> B, R = P.div(Q) # division euclidienne : P = BQ + R
>>> print(f"Les polynômes B et R tels que P = QB + R sont\nB(x) = {B.as_expr()} et
 ↪  R(x) = {R.as_expr()}")
```

```
Les polynômes B et R tels que P = QB + R sont
B(x) = x - 2 et R(x) = 3*x + 3
```

Propriété 3.2

Soit $(P, Q) \in \mathbb{K}[x]^2$ et soit $\lambda \in \mathbb{K}$.

1. $(P + Q)(\lambda) = P(\lambda) + Q(\lambda)$.
2. $(PQ)(\lambda) = P(\lambda)Q(\lambda)$.

Définition 3.3 : racine

Soit $P \in \mathbb{K}[x]$, et soit $\lambda \in \mathbb{K}$.
On dit que λ est une *racine* de P quand $P(\lambda) = 0$.

De même que pour les entiers relatifs on introduit la notion de *diviseur* d'un polynôme.

Définition 3.4 : diviseur d'un polynôme

Soit $(P, Q) \in \mathbb{K}[x]^2$.
On dit que P est un *diviseur* de Q s'il existe $U \in \mathbb{K}[x]$ tel que $Q = PU$. On dit alors que Q est un *multiple* de P.

On a alors le résultat suivant, déjà vu en Terminale pour les polynômes à coefficients réels.

Corollaire 3.1

Soit $P \in \mathbb{K}[x]$ et soit $\lambda \in \mathbb{K}$.
Alors λ est une racine de P si et seulement si P est divisible par $x - \lambda$.

Démonstration 3.2

Soit $P \in \mathbb{K}[x]$ *et soit* $\lambda \in \mathbb{K}$. *Soit* $B \in \mathbb{K}[x]$ *tel que* $B(x) = x - \lambda$. *Alors d'après le théorème 3.1, il existe une unique paire de polynômes* $(Q ; R)$ *telle que* $P(x) = (x-\lambda)Q(x) + R(x)$, *avec* $\deg(R) < \deg(B) = 1$. *Donc il existe* $a \in \mathbb{K}$ *tel que* $R = a$ *et ainsi* $P(\lambda) = (\lambda-\lambda)\times Q(\lambda) + a = a$. *Ceci montre que* $P(x) = (x-\lambda)Q(x) + P(\lambda)$, *et on voit que* $x - \lambda$ *est un diviseur de* P *si et seulement si* $P(\lambda) = 0$.

□

Définition 3.5 : multiplicité d'une racine

Soit un polynôme P de $\mathbb{K}[x]$ non nul et soit $a \in \mathbb{K}$ une racine de P.
On définit l'*ordre de multiplicité* de a comme le plus grand entier k tel que $(x - a)^k$ divise P.

3 Idéaux de l'anneau des polynômes

On va maintenant introduire la notion d'*idéal*, qui joue un rôle central dans la théorie des anneaux.

Définition 3.6 : idéal

On dit que $I \subset \mathbb{K}[x]$ est un *idéal* de $\mathbb{K}[x]$ quand les deux conditions suivantes sont vérifiées

1. I est un sous-groupe de $(\mathbb{K}[x] ; +)$;
2. $\forall P \in I, \forall Q \in \mathbb{K}[x], PQ \in I$.

Pour $P \in \mathbb{K}[x]$, on pose $P\mathbb{K}[x] = \{PQ \mid Q \in \mathbb{K}[x]\}$. Il est clair que $P\mathbb{K}[x]$ est un idéal de $\mathbb{K}[x]$. Réciproquement on a le résultat suivant, qui est la base de « l'arithmétique des polynômes ».

Théorème 3.2

Pour tout idéal I de $\mathbb{K}[x]$, il existe $B \in \mathbb{K}[x]$ tel que $I = B\mathbb{K}[x]$.

Démonstration 3.3

Si $I = \{0\}$, *il suffit de prendre* $B = 0$.
Si $I \neq \{0\}$, *on note* $I^\star$ *l'ensemble des éléments non nuls de* I. *Il existe* $B \in I^\star$ *tel que pour tout* $P \in I^\star$, $\deg(B) \leq \deg(P)$. *Alors, puisque* I *est un idéal, pour tout* $Q \in \mathbb{K}[x]$, *on a* $BQ \in I$, *donc* $B\mathbb{K}[x] \subset I$.
Soit maintenant $P \in I$. *Il existe* $(Q, R) \in \mathbb{K}[x]^2$, *avec* $\deg(R) < \deg(B)$, *tel que* $P = BQ + R$. *Comme* $B \in I$, $BQ \in I$, *et comme* $P \in I$, $R = P - BQ \in I$. *Si* $R \neq 0$, *on aurait* $R \in I^\star$ *et* $\deg(R) \geq \deg(B)$, *ce qui n'est pas le cas. Donc* $R = 0$, *ce qui prouve que* $I = B\mathbb{K}[x]$.

□

4 Notion de pgdc de polynômes

Définition 3.7 : polynôme unitaire

On dit qu'un polynôme $P \in \mathbb{K}[x]$ est *unitaire* s'il existe $n \in \mathbb{N}$ tel que $P(x) = \sum_{k<n} a_k x^k + x^n$. On a alors $\deg(P) = n$.
Dit autrement, un polynôme unitaire est tel que son *coefficient dominant* est égal à 1.

Théorème 3.3

Soit $(A_1, \ldots, A_n)$ une famille d'éléments de $\mathbb{K}[x]$ non tous nuls.
Il existe un unique polynôme unitaire $D \in \mathbb{K}[x]$, appelé le pgdc de la famille $(A_1, \ldots, A_n)$, possédant les propriétés suivantes

1. pour $i \in [\![1\,;n]\!]$, D est un diviseur de A_i ;
2. s'il existe $\tilde{D} \in \mathbb{K}[x]$ tel que, pour $i \in [\![1\,;n]\!]$, $\tilde{D}$ est aussi un diviseur de A_i, alors $\tilde{D}$ est un diviseur de D ;
3. il existe une famille $(U_1, \ldots, U_n)$ d'éléments de $\mathbb{K}[x]$ vérifiant $\sum_{i=1}^n A_i U_i = D$ (identité de Bézout).

De plus, si $P \in \mathbb{K}[x]$, alors l'équation $\sum_{i=0}^n A_i V_i = P$ possède des solutions $(V_1, \ldots, V_n)$ dans $\mathbb{K}[x]^n$ si et seulement si P est un multiple de D.

Définition 3.8 : symbole de Kronecker

Soient m et n sont deux entiers naturels.
On définit le *symbole de Kronecker* $\delta_{m,n}$ noté aussi δ_m^n par la formule

$$\delta_{m,n} = \begin{cases} 0 & \text{si } m \neq n \\ 1 & \text{si } m = n \end{cases}$$

Démonstration 3.4

Soit I *l'ensemble des polynômes de la forme* $P = \sum_{i=1}^n A_i Q_i$, *où, pour* $i \in [\![1\,;n]\!]$, $Q_i \in \mathbb{K}[x]$. *Soit* $(i,j) \in [\![1\,;n]\!]^2$. *En posant* $Q_j = \delta_i^j$, *on voit que* $A_i \in I$ *et donc* $I \neq \{0\}$.
Soient $P = \sum_{i=1}^n A_i Q_i$ *et* $\tilde{P} = \sum_{i=1}^n A_i \tilde{Q}_i$ *deux éléments de* I. *On a donc* $P - \tilde{P} = \sum_{i=1}^n A_i(Q_i - \tilde{Q}_i) \in I$, *et par conséquent* I *est un sous-groupe de* $(\mathbb{K}[x]\,;+)$.
Soit maintenant $P = \sum_{i=1}^n A_i Q_i \in I$, *et soit* $Q \in \mathbb{K}[x]$. *On a alors* $PQ = \sum_{i=1}^n A_i Q_i Q \in I$, *et on voit que* I *est un idéal de* $\mathbb{K}[x]$. *D'après le théorème 3.2 p. 16, il existe donc* $D \in \mathbb{K}[x]$ *tel que* $I = D\mathbb{K}[x]$. *Comme* $I \neq \{0\}$, $D \neq 0$, *et, quitte à multiplier* D *par un élément non nul de* $\mathbb{K}$, *on peut supposer que* D *est unitaire : on a l'existence d'un polynôme unitaire de* I.
Pour tout $i \in [\![1\,;n]\!]$, $A_i \in I$, *et, par construction de* I, *il existe* $Q_i \in \mathbb{K}[x]$ *tel que* $A_i = DQ_i$, *ce qui montre que* D *est un diviseur de* A_i. *Le point 3.3-1 est vérifié.*
Comme $D \in I$, *ce polynôme peut se décomposer sous la forme* $D = \sum_{i=1}^n A_i U_i$, *où, pour* $i \in [\![1\,;n]\!]$, $U_i \in \mathbb{K}[x]$. *Le point 3.3-3 est vérifié.*
Soit maintenant $\Delta \in \mathbb{K}[x]$ *tel que pour tout* $i \in [\![1\,;n]\!]$, Δ *divise* A_i. *Alors il existe un famille d'éléments de* $\mathbb{K}[x]$ $(V_1, \ldots, V_n)$ *telle que* $A_i = \Delta V_i$. *Par conséquent,* $D = \sum_{i=1}^n A_i U_i = \sum_{i=1}^n (\Delta V_i) U_i$. *Donc* Δ *divise* D *et* D *vérifie aussi 3.3-2.*

Supposons maintenant l'existence de deux polynômes D_1 *et* D_2 *vérifiant les trois points du théorème. Alors* D_1 *divise* D_2 *et* D_2 *divise* D_1. *Donc il existe* $(B_1, B_2) \in \mathbb{K}[x]^2$ *tel que* $D_1 = D_2B_1$ *et* $D_2 = D_1B_2$ *et ainsi* $D_1 = D_2B_1B_2$. *Comme* $D_1 \neq 0$ *(c'est un polynôme unitaire),* $B_1B_2 = 1$ *et* $\deg(B_1) + \deg(B_2) = 0$. *Il faut nécessairement* $\deg(B_1) = \deg(B_2) = 0$ *: il existe* $b \in \mathbb{K}^\star$ *tel que* $D_1 = bD_2$. *Comme* D_1 *et* D_2 *sont unitaire, pour avoir l'égalité sur les coefficients dominants et que* $b \neq 0$, *on a* $b = 1$, *soit* $D_1 = D_2$, *ce qui prouve l'unicité du* pgdc.
Enfin, considérons $P \in \mathbb{K}[x]$. *Il résulte de la définition de* D *que* P *est un multiple de* D *si et seulement si* $P \in I$. *Et par construction de* I, *comme le point 3.3-3 est vérifié, on a bien que l'équation* $\sum_{i=0}^{n} A_iV_i = P$ *possède des solutions* $(V_1, \ldots, V_n)$ *dans* $\mathbb{K}[x]^n$ *si et seulement si* P *est un multiple de* D.

□

Il est clair que le pgdc d'une famille $(A_1, \ldots, A_n)$ de polynômes ne change pas si on retire de cette famille des polynômes nuls. D'autre part, si pour $k \in \mathbb{N}$ on note D_k le pgdc de la famille $(A_1, \ldots, A_k)$, alors le pgdc de la famille $(A_1, \ldots, A_{k+1})$ est égal au pgdc de D_k et de A_{k+1}. Donc pour calculer le pgdc d'une famille finie quelconque de polynômes il suffit de savoir calculer le pgdc de deux polynômes non nuls.

On peut pour cela utiliser l'algorithme d'Euclide (voir l'algorithme 2.1 p. 6), de même que dans le cas du pgdc de deux entiers.

Définition 3.9 : polynômes équivalents

On dira que deux polynômes U et V à coefficients dans $\mathbb{K}$ sont *équivalents*, et on écrira $U \equiv V$, s'il existe $\lambda \in \mathbb{K}^\star$ tel que $U = \lambda V$.

Propriété 3.3

le pgdc de deux polynômes ne change pas si on remplace ces deux polynômes par des polynômes équivalents.

Comme la procédure a déjà été vue pour le pgdc de deux entiers, considérons un exemple.

Exemple 3.2

Déterminons le pgdc de $x^3 + 1$ et de $x^4 + x^2 + 3x + 1$.
L'algorithme d'Euclide permet de construire le tableau suivant.

k	$A_k = B_k \times Q_k + R_k$
0	$x^4 + x^2 + 3x + 1 = (x^3 + 1) \times x + x^2 + 2x + 1$
1	$x^3 + 1 = (x^2 + 2x + 1) \times (x - 2) + 3x + 3$
2	$x^2 + 2x + 1 = (3x + 3) \times \left(\frac{x}{3} + \frac{1}{3}\right) + 0$

Comme le coefficient dominant du dernier reste non nul vaut 3, on a pgdc $(x^3 + 1\,;x^4 + x^2 + 3x + 1) = x + 1$.

```
>> import sympy as sp
>> sp.var("x")
x
>> P = x**3+1
>> Q = x**4+x**2+3*x+1
>> PGDC = sp.gcd(P, Q)
>> print(f"Le PGDC de P et de Q vaut\n{sp.pretty(PGDC.as_expr())}")
Le PGDC de P et de Q vaut
x + 1
```

Pour ce qui est de résoudre une équation de type Bézout mettant en œuvre des polynômes, on adapte l'algorithme d'Euclide étendu (voir l'algorithme 2.2 p. 7). Considérons là aussi un simple exemple.

Exemple 3.3

Soient deux polynômes P et Q définis par $P(x) = x^3 + 1$ et $Q(x) = x^4 + x^2 + 3x + 1$ Déterminons U et V tels que $PU + Qv = \text{pgdc}(P\,;Q)$.
L'algorithme d'Euclide étendu permet de construire le tableau suivant.

k	$A_k = B_k \times Q_k + R_k$	U_k	V_k
		1	0
		0	1
0	$x^4 + x^2 + 3x + 1 = (x^3 + 1) \times x + (x^2 + 2x + 1)$	1	$-x$
1	$x^3 + 1 = (x^2 + 2x + 1) \times (x - 2) + (3x + 3)$	$2 - x$	$x^2 - 2x + 1$
2	$x^2 + 2x + 1 = (3x + 3) \times \left(\frac{x}{3} + \frac{1}{3}\right) + 0$		

Comme le coefficient dominant du dernier reste non nul vaut 3, on a pgdc $(P ; Q) = x + 1$ et des solutions de l'équation de Bézout sont $U(x) = \frac{(x-1)^2}{3}$ et $V(x) = -\frac{x-2}{3}$.

```
>>> import sympy as sp
>>> sp.var("x")
x
>>> P = x**3+1
>>> Q = x**4+x**2+3*x+1
>>> U, V, PGDC = sp.gcdex(P, Q)
>>> print(f"On cherche U et V tels que PU + QV = Pgdc(P, Q).\nOn peut prendre
 ↪  U =\n{sp.pretty(U.as_expr())}\net V =\n{sp.pretty(V.as_expr())}")
On cherche U et V tels que PU + QV = Pgdc(P, Q).
On peut prendre U =
 2
x    2*x   1
- - - + -
3     3    3
et V =
2   x
- - -
3   3
>>> print(f"On vérifie l'égalité : {sp.simplify(P*U+Q*V) == PGDC}")
On vérifie l'égalité : True
```

5 Applications du théorème de Bézout

Définition 3.10 : polynômes premiers entre eux

On dit que deux polynômes non nuls A et B sont *premiers entre eux* quand leur pgdc est égal à 1.

Théorème 3.4 : de Gauss

Soient A, B et C trois polynômes non nuls.
Si A divise BC et si A est premier avec B, alors A divise C.

Démonstration 3.5

Par définition, comme A *et* B *sont premiers entre eux, leur* pgdc *est égal à 1. De plus, d'après le théorème 3.3 p. 17, il existe* $(U, V) \in \mathbb{K}[x]^2$ *tel que* $AU + BV = 1$. *Donc* $C = AUC + BVC$. *Comme* A *divise* BC, *il existe* $W \in \mathbb{K}[x]$ *tel que* $BC = AW$. *Donc* $C = A(UC + VW)$, *ce qui montre que* A *divise* C. □

Corollaire 3.2

Soient A et B deux polynômes non nuls et soit D le pgdc de A et de B. Soit $\mathcal{S} = \{(U, V) \in \mathbb{K}[x]^2 \mid AU + BV = 0\}$.
On a alors $\mathcal{S} = \{(-P\tilde{B}, P\tilde{A}) \in \mathbb{K}[x]^2 \mid P \in \mathbb{K}[x]\}$, où $\tilde{A} = \frac{A}{D}$ et $\tilde{B} = \frac{B}{D}$.

Démonstration 3.6

Il résulte du théorème de Bézout que $\tilde{A}$ *et* $\tilde{B}$ *sont premiers entre eux, et* $\mathcal{S} = \left\{(U,V) \in \mathbb{K}[x]^2 \mid \tilde{A}U + \tilde{B}V = 0\right\}$. *Il est clair que si* $U = -P\tilde{B}$ *et si* $V = P\tilde{A}$, *alors* $(U,V) \in \mathcal{S}^2$. *Réciproquement, si* $\tilde{A}U + \tilde{B}V = 0$, *alors* $\tilde{A}$ *divise* $\tilde{B}V$, *donc* $\tilde{A}$ *divise* V, *d'après le théorème de Gauss. Donc il existe* $P \in \mathbb{K}[x]$ *tel que* $V = P\tilde{A}$. *On a alors* $U\tilde{A} = -\tilde{B}P\tilde{A}$, *donc* $U = -P\tilde{B}$.

□

Corollaire 3.3

Soient A et B deux polynômes non nuls premiers entre eux, et soit P un polynôme.
Il existe un unique couple (R ; T) de polynômes vérifiant les deux conditions suivantes

1. $AR + BT = P$;
2. $\deg(R) < \deg(B)$.

De plus $AU + BV = P$ si et seulement si il existe un polynôme Q tel que $U = R + BQ$ et $V = T - AQ$.

Démonstration 3.7

On sait que l'équation de Bézout $AU + BV = 1$ *possède une solution* $(U_1, V_1) \in \mathbb{K}[x]^2$. *Soit* R *le reste de la division de* U_1P *par* B : *il existe* $Q \in \mathbb{K}[x]$ *tel que* $U_1P = BQ + R$. *Posons* $T = V_1P + AQ$. *On a alors* $AR + BT = A(U_1P - BQ) + B(V_1P + AQ) = AU_1 + BV_1 = P$ *et* $\deg(R) < \deg(B)$.
On a $AU + BV - P = A(U - R) + B(V - T)$. *Donc* $AU + BV = P$ *si et seulement si il existe un polynôme* Q *tel que* $U = R + BQ$ *et* $V = T - AQ$.
Soit $(\tilde{R} ; \tilde{T})$ *un autre couple de polynômes vérifiant les deux conditions du corollaire. Alors il existe* $Q \in \mathbb{K}[x]$ *tel que* $\tilde{R} = BQ + R$. *D'après l'unicité de la division euclidienne des polynômes on a alors* $Q = 0$ *et* $\tilde{R} = R$. *Comme* $B(\tilde{T} - T) = 0$, *on a alors* $\tilde{T} = T$.

□

On a la variante suivante du théorème de Gauss.

Théorème 3.5

Soit $n \in \mathbb{N}^\star$. Soit $(B_1, \ldots, B_n)$ une famille de polynômes non nuls et soit A un polynôme non nul.
Si pour tout $i \in [\![1 ; n]\!]$ A est premier avec B_i, alors A est premier avec le produit $\prod_{i=1}^{n} B_i$.

Démonstration 3.8

Supposons que A *est premier avec* B_1 *et* B_2 : *il existe* $(U_1, V_1, U_2, V_2) \in \mathbb{K}[x]^4$ *tel que* $AU_1 + B_1V_1 = AU_2 + B_2V_2 = 1$. *En multipliant membre à membre on obtient* $(AU_1 + B_1V_1)(AU_2 + B_2V_2) = 1$ *soit* $A(AU_1U_2 + U_1B_2V_2 + B_1V_1U_2) + B_1B_2(V_1V_2) = 1$. *D'après le théorème 3.3 p. 17, le* pgdc D *de* A *et de* B_1B_2 *est donc un diviseur de* 1, *donc* $D = 1$. *Ainsi* A *est premier avec* B_1B_2 *et la propriété est vraie pour* $n = 2$.
Supposons la propriété vraie pour un certain $n \in \mathbb{N}$, *avec* $n \geq 2$. *Soit* $(B_1, \ldots, B_{n+1})$ *une famille de polynômes non nuls tels que pour tout* $i \in [\![1 ; n+1]\!]$ A *est premier avec* B_i. *D'après l'hypothèse de récurrence, pour tout* $i \in [\![1 ; n]\!]$, A *est premier avec* $\prod_{i=1}^{n} B_i$. *En utilisant la même démonstration que pour la vérification de l'initialisation, on a alors que* A *est premier avec* $\left(\prod_{i=1}^{n} B_i\right) B_{n+1} = \prod_{i=1}^{n+1} B_i$.
On voit donc par récurrence que le théorème est valable pour tout $n \in \mathbb{N}$, $n \geq 2$.

□

Corollaire 3.4

Soit $n \in \mathbb{N}^\star$. Soit $(P_1, \ldots, P_n)$ une famille de polynômes premiers entre eux deux à deux.
S'il existe $U \in \mathbb{K}[x]$ tel que pour tout $i \in [\![1 ; n]\!]$ U est divisible par P_i, alors U est divisible par le produit $\prod_{i=1}^{n} P_i$.

Démonstration 3.9

Procédons par récurrence sur $n \in \mathbb{N}^\star$.

- *Si* $n = 1$, *il n'y a rien à démontrer.*
- *Soit* $n \in \mathbb{N}$, $n \geq 2$ *et soit* $(P_1, \ldots, P_n)$ *une famille de polynômes premiers entre eux deux à deux. Au rang* $n - 1$ *on suppose que s'il existe* $U \in \mathbb{K}[x]$ *tel que pour tout* $i \in [\![1 ; n-1]\!]$ U *est divisible par* P_i, *alors* U *est divisible par le produit* $\prod_{i=1}^{n-1} P_i$.

Soit donc $U \in \mathbb{K}[x]$ *tel que pour tout* $i \in [\![1\,;n]\!]$ U *est divisible par* P_i. *Alors par hypothèse de récurrence,* U *est divisible par le produit* $\prod_{i=1}^{n-1} P_i$ *: il existe* $V \in \mathbb{K}[x]$ *tel que* $U = \prod_{i=1}^{n-1} P_i V$. *Il résulte du théorème 3.5 p. 20 que* P_n *est premier avec* $\prod_{i=1}^{n-1} P_i$ *et on déduit alors du théorème de Gauss que* P_n *divise* V. *Donc* U *est divisible par le produit* $\prod_{i=1}^{n} P_i$ *et la propriété est vraie pour n.*

— *Le résultat est donc démontré par récurrence.*

□

6 Le théorème chinois pour les polynômes

Définition 3.11 : congruence de polynômes

Soit $(A, B, P) \in \mathbb{K}[x]^3$. On dit que A est *congru à* B *modulo* P et on écrit $A \equiv B\ [P]$, quand $A - B$ est divisible par P.

Propriété 3.4

Soit $(A, A_2, B_1, B_2, P) \in \mathbb{K}[x]^5$.
Si $A_1 \equiv A_2\ [P]$ et si $B_1 \equiv B_2\ [P]$, alors $A_1 + B_1 \equiv A_2 + B_2\ [P]$ et $A_1 B_1 \equiv A_2 B_2\ [P]$.

Donnons maintenant une version du théorème 2.6 (chinois) p. 10 valable pour les polynômes.

Théorème 3.6 : chinois pour les polynômes

Soit $n \in \mathbb{N}$ et soit $(M_1, \ldots, M_n)$ une famille de polynômes non nuls premiers deux à deux.
Alors pour toute famille $(Q_1, \ldots, Q_n)$ d'éléments de $\mathbb{K}[x]$, le système d'équations de congruence

$$\begin{cases} P \equiv Q_1\ [M_1] \\ \ldots \\ P \equiv Q_n\ [M_n] \end{cases} \tag{3.1}$$

possède des solutions dans $\mathbb{K}[x]$.
De plus si $\tilde{P} \in \mathbb{K}[x]$ est une solution particulière de (3.1), la solution générale du système est donnée par la formule

$$P = \tilde{P} + U \prod_{i=1}^{n} M_i, \quad \text{avec } U \in \mathbb{K}[x] \text{ un polynôme quelconque.} \tag{3.2}$$

Démonstration 3.10

Soit $\tilde{P} \in \mathbb{K}[x]$ *une solution du système (3.1). Il est clair que si* $P = \tilde{P} + U \prod_{i=1}^{n} M_i$ *avec* $U \in \mathbb{K}[x]$, *alors pour tout* $i \in [\![1\,;n]\!]$, $P \equiv \tilde{P}\ [M_i]$ *et donc* $P \equiv Q_i\ [M_i]$ *: P est solution du système.*
Réciproquement, si P *est solution du système on a pour tout* $i \in [\![1\,;n]\!]$, $P - \tilde{P} \equiv Q_i - Q_i\ [M_i]$, *donc* $P - \tilde{P}$ *est divisible par* M_i. *Il résulte alors du corollaire 3.4 p. 20 que* $P - \tilde{P}$ *est divisible par le produit* $\prod_{i=1}^{n} M_i$ *: il existe* $U \in \mathbb{K}[x]$ *tel que* $P = \tilde{P} + U \prod_{i=1}^{n} M_i$.
Procédons par récurrence pour établir l'existence d'une solution.

— *Le résultat est évident si* $n = 1$ *: il suffit de prendre* $P = Q_1$.

— *Supposons donc que pour un certains* $n \in \mathbb{N}$, $n \geq 1$, *il existe une solution au système (3.1). On ajoute à ce système l'équation* $P \equiv Q_{n+1}\ [M_{n+1}]$ *où* $(Q_{n+1}, M_{n+1}) \in \mathbb{K}[x]^2$ *avec* Q_{n+1} *quelconque et* M_{n+1} *premier avec les* M_i *pour tout* $i \in [\![1\,;n]\!]$. *D'après l'hypothèse de récurrence, il existe* $A \in \mathbb{K}[x]$ *tel que pour tout* $i \in [\![1\,;n]\!]$, $A \equiv Q_i\ [M_i]$. *On a alors que pour tout* $Q \in \mathbb{K}[x]$, *pour tout* $i \in [\![1\,;n]\!]$, $A + \prod_{i=1}^{n} M_i Q \equiv Q_i\ [MP_i]$. *Comme pour tout* $i \in [\![1\,;n]\!]$, M_{n+1} *est premier avec* M_i, *il résulte du théorème 3.5 p. 20* M_{n+1} *est aussi premier avec le produit* $\prod_{i=1}^{n} M_i$. *Il existe donc deux polynômes* U *et* V *tels que* $\left(\prod_{i=1}^{n} M_i\right) U + M_{n+1} V = 1$.

Posons $P = A - \prod_{i=1}^{n} M_i U(A - Q_{n+1})$
$= A - (1 - M_{n+1}V)(A - Q_{n+1})$
$= Q_{n+1} + M_{n+1}V(A - Q_{n+1})$
Alors $P - Q_{n+1}$ *est divisible par* M_{n+1} *et par conséquent, pour tout* $i \in [\![1\,;n+1]\!]$, $P \equiv Q_i \left[M_i\right]$.
— *Le théorème est donc démontré par récurrence.*

□

Notons que cette démonstration fournit une méthode effective pour résoudre ce type d'équations de congruence, en se ramenant à chaque étape à des équations du type équation de Bézout. Nous traitons ci dessous un exemple simple pour fixer les idées.

Exemple 3.4

Soit $(\lambda_1, \lambda_2) \in \mathbb{C}^2$, avec $\lambda_1 \neq \lambda_2$. Trouvez $P \in \mathbb{C}[x]$ tel que $P - \lambda_1$ soit divisible par $x - \lambda_1$ et tel que $P - \lambda_2$ soit divisible par $(x - \lambda_2)^2$.

Dit autrement, on cherche P solution du système $\begin{cases} P \equiv \lambda_1 \left[x - \lambda_1\right] \\ P \equiv \lambda_2 \left[(x - \lambda_2)^2\right] \end{cases}$

Comme $\lambda_1 \neq \lambda_2$, $x - \lambda_1$ est premier avec $x - \lambda_2$. Il résulte du théorème 3.5 p. 20 que $x - \lambda_1$ est premier avec $(x - \lambda_2)^2$ et d'après le théorème 3.6, il existe bien un polynôme P vérifiant les conditions ci-dessus.
On va chercher un polynôme de la forme $P = \lambda_2 + Q(x - \lambda_2)^2$. Un tel polynôme vérifie automatiquement $P \equiv \lambda_2 \left[(x - \lambda_2)^2\right]$.
On va chercher à déterminer Q de façon que $P - \lambda_1$ soit divisible par $x - \lambda_1$. On a $P - \lambda_1 = \lambda_2 - \lambda_1 + Q(x - \lambda_2)^2$.
On peut ici éviter le recours à l'équation de Bézout, car on sait que $P - \lambda_1$ est divisible par $x - \lambda_1$ si et seulement si λ_1 est une racine de $P - \lambda_1$ (voir le corollaire 3.1 p. 16). Donc λ_1 est racine de $P - \lambda_1$ si et seulement si $\lambda_2 - \lambda_1 + (\lambda_1 - \lambda_2)^2 Q(\lambda_1) = 0 \iff Q(\lambda_1) = \frac{1}{\lambda_1 - \lambda_2}$. Le polynôme constant $Q = \frac{1}{\lambda_1 - \lambda_2}$ convient, et le polynôme $P = \lambda_2 + \frac{1}{\lambda_1 - \lambda_2}(x - \lambda_2)^2$ vérifie les conditions voulues.

7 Notion de ppmc de polynômes

Étendons la notion de ppmc vue pour des entiers au cas des polynômes.

Théorème 3.7

Soit $n \in \mathbb{N}^\star$ et soit $(P_1, \ldots, P_n)$ une famille finie de polynômes non nuls.
Il existe alors un unique polynôme unitaire M, appelé le ppmc de la famille $(P_1, \ldots, P_n)$ possédant les deux propriétés suivantes
1. pour tout $i \in [\![1\,;n]\!]$, M est un multiple de P_i ;
2. si pour tout $i \in [\![1\,;n]\!]$ Q est un multiple de P_i, alors Q est un multiple de M.

Démonstration 3.11

Soit $i \in [\![1\,;n]\!]$. *L'ensemble des multiples de* P_i *est égal à* $P_i\mathbb{K}[x]$. *Donc l'ensemble des multiples communs à tous les* P_j, $j \in [\![1\,;n]\!]$ *est l'ensemble* $I = \bigcap_{1\leq j\leq n} P_j\mathbb{K}[x]$. *On vérifie immédiatement que* I *est un idéal de* $\mathbb{K}[x]$, *et* $I \neq \{0\}$ *puisque* $\prod_{j=1}^{n} P_j \in I$. *D'après le théorème 3.2 p. 16, il existe un polynôme unitaire* $M \in \mathbb{K}[x]$ *tel que* $I = M\mathbb{K}[x]$. *Par construction* M *vérifie 3.7-2, et* M *vérifie également 3.7-1 puisque* $M \in I$. *Si* $\tilde{M}$ *est un autre polynôme unitaire vérifiant 3.7-1 et 3.7-2, alors* $\tilde{M}$ *divise* M *et* M *divise* $\tilde{M}$, *ce qui entraine comme on l'a déjà vu que* $M = \tilde{M}$.

□

Soit $i \in \mathbb{N}$. Il est clair que si M_i désigne le ppmc de la famille $(P_1, \ldots, P_i)$, alors le ppmc de la famille $(P_1, \ldots, P_{i+1})$ est égal au ppmc de M_i et de P_{i+1}. Par conséquent, les calculs de ppmc se ramènent à des calculs de ppmc de deux polynômes. Ceci peut se faire en utilisant le résultat suivant.

Théorème 3.8

Soient P et Q deux polynômes unitaires. soit D leur pgdc et soit M leur ppmc.
Alors $PQ = DM$.

Démonstration 3.12

Comme $D = \text{pgdc}(P ; Q)$, *il existe* $(A, B) \in \mathbb{K}[x]^2$ *tel que* $P = DA$ *et* $Q = DB$ *où* A *et* B *sont des polynômes premiers entre eux. Il est clair que* DAB *est un multiple de* P *et de* Q.
Soit maintenant $U \in \mathbb{K}[x]$ *un multiple de* P *et de* Q. *Il existe deux polynômes* V *et* W *tels que* $U = DAV = DBW$. *Donc* $AV = BW$ *et* A *divise* BW. *Comme* A *est premier avec* B, *il résulte du théorème de* Gauss *que* A *divise* W. *Il existe donc un polynôme* S *tel que* $U = DBAS$ *et* U *est donc un multiple de* DAB. *Comme* P, Q *et* D *sont unitaires,* DAB *aussi : on pose* $DAB = M$ *et* $PQ = DM$.

□

Exemple 3.5

Calculez le ppmc de x^3+1 et de x^4+x^2+3x+1.
On a vu que le pgdc de ces deux polynômes est égal à $x+1$. Comme $x^3+1 = (x+1)(x^2-x+1)$, le ppmc de x^3+1 et de x^4+x^2+3x+1 est égal à $(x^2-x+1)(x^4+x^2+3x+1) = x^6-x^5+2x^4+2x^3-x^2+2x+1$.

```
>> import sympy as sp
>> sp.var("x")
x
>> P = x**3+1
>> Q = x**4+x**2+3*x+1
>> PGDC = sp.gcd(P, Q)
>> print(f"Le PGDC de P et de Q vaut\n{sp.pretty(PGDC.as_expr())}")
Le PGDC de P et de Q vaut
x + 1
>> PPMC = sp.lcm(P, Q)
>> print(f"Le PPMC de P et de Q vaut\n{sp.pretty(PPMC.as_expr())}")
Le PPMC de P et de Q vaut
 6    5      4      3    2
x  - x  + 2*x  + 2*x  - x  + 2*x + 1
```

On a également le résultat suivant, qui est une simple re-formulation du corollaire 3.4 p. 20.

Propriété 3.5

Soit $n \in \mathbb{N}$ et soit $(P_1, \ldots, P_n)$ une famille de polynômes non nuls premiers entre eux deux à deux. Alors le ppmc de la famille $(P_1, \ldots, P_n)$ est égal au produit $\prod_{i=1}^{n} P_i$.

8 Polynômes irréductibles

On va maintenant introduire la notion de *polynôme irréductible*, qui est l'analogue pour les polynômes de la notion de *nombre premier*.

Définition 3.12 : polynôme irréductible

Soit $\mathbb{K}$ un corps. On dit que $P \in \mathbb{K}[x]$ est *irréductible* si P est non constant et si les seuls diviseurs de P dans $\mathbb{K}[x]$ sont soit constants, soit de la forme aP avec $a \in \mathbb{K}^\star$.

Contrairement à la notion de pgdc (le pgdc d'une famille de polynômes à coefficients dans $\mathbb{K}$ ne change pas si on remplace le corps $\mathbb{K}$ par un corps plus grand), la notion de polynôme irréductible dépend du corps considéré, comme le montrent les exemples très simples suivants.

Exemple 3.6

Le polynôme x^2-5 est irréductible dans $\mathbb{Q}[x]$, mais pas dans $\mathbb{R}[x]$.
Le polynôme x^2+1 est irréductible dans $\mathbb{R}[x]$, mais pas dans $\mathbb{C}[x]$.
En effet un polynôme non irréductible de degré 2 doit posséder un diviseur de degré 1, qui est de la forme $ax+b$ avec $a \neq 0$, et admet $\frac{-b}{a}$ pour racine.
Comme $\sqrt{5}$ est irrationnel, x^2-5 est irréductible dans $\mathbb{Q}[x]$. Par contre $x^2-5 = (x-\sqrt{5})(x+\sqrt{5})$ n'est pas irréductible dans $\mathbb{R}[x]$.
De même x^2+1 est irréductible dans $\mathbb{R}[x]$, mais $x^2+1 = (x+\imath)(x-\imath)$ n'est pas irréductible dans $\mathbb{C}[x]$.

On a un analogue de la décomposition des nombres entiers en produit de facteurs premiers.

Théorème 3.9

Soit $\mathbb{K}$ un corps. Pour tout polynôme non constant $P \in \mathbb{K}[x]$, il existe un élément non nul $a \in \mathbb{K}$, un entier non nul n, une famille $(P_1, \ldots, P_n)$ de polynômes unitaires irréductibles distincts et une famille $(k_1, \ldots, k_n)$ d'entiers positifs tels que $P = a\prod_{i=1}^{n} P_i^{k_i}$.
De plus cette *décomposition en produit de polynômes irréductibles* est unique à l'ordre des facteurs près.

Démonstration 3.13

Comme tout polynôme $Q \neq 0$ s'écrit de manière unique sous la forme $Q = aP$ avec $a \in \mathbb{K}^\star$ et P unitaire, il suffit de montrer que tout polynôme unitaire P de degré au moins 1 possède une décomposition en produit de polynômes unitaires irréductibles. Procédons par récurrence. On note $\mathscr{P}_n$ la proposition « un polynôme P unitaire de degré n possède une décomposition en produit de polynômes unitaires irréductibles ».

- *C'est évident si $\deg(P) = 1$ car dans ce cas P est irréductible.*
- *Soit un certain $n \in \mathbb{N}^\star$. Supposons cette propriété vraie pour tous les polynômes unitaires tels que $1 \leq \deg(P) \leq n$. Soit $\tilde{P}$ un polynôme unitaire de degré $n+1$.*
  - *Si $\tilde{P}$ est irréductible, on a la décomposition triviale $\tilde{P} = \tilde{P}$.*
  - *Si $\tilde{P}$ n'est pas irréductible il existe deux polynômes non constants Q et R tels que $\tilde{P} = QR$. Comme le produit des coefficients des termes de plus haut degré de Q et R est égal à 1, on peut se ramener au cas où Q et R sont unitaires, en multipliant Q et en divisant R par un élément non nul de $\mathbb{K}$ convenable. On a $\deg(Q) < n+1$ et $\deg(R) < n+1$, et d'après l'hypothèse de récurrence on peut décomposer Q et R en produits de polynômes unitaires irréductible. Donc $\tilde{P} = QR$ se décompose en produit de polynômes irréductibles.*

  La propriété $\mathscr{P}_{n+1}$ est donc vérifiée.
- *L'existence de la décomposition cherchée est établie par récurrence pour tout polynôme non constant P.*

Pour démontrer l'unicité à l'ordre près des facteurs de la décomposition du théorème on peut également se limiter au cas où P est unitaire. Notons que si deux polynômes irréductibles Q et R ne sont pas premiers entre eux alors par définition il existe $a \in \mathbb{K}^\star$ tel que $Q = aR$, ce qui implique que $Q = R$ si Q et R sont unitaires.
Procédons par récurrence. On note $\mathscr{Q}_n$ la proposition « un polynôme P unitaire de degré n possède une décomposition en produit de polynômes unitaires irréductibles unique à l'ordre des facteurs près ».

- *L'unicité de la décomposition est triviale si $\deg(P) = 1$.*
- *Soit un certain $n \in \mathbb{N}^\star$. Supposons que la décomposition est unique à l'ordre près des facteurs pour tout polynôme unitaire P tel que $1 \leq \deg(P) \leq n$. Soit $\tilde{P}$ un polynôme unitaire de degré $n+1$, et soient deux décompositions de P en produit de puissances de polynômes unitaires irréductibles distincts : $P = \prod_{i=1}^{n} P_i^{k_i} = \prod_{j=1}^{n'} Q_j^{k'_j}$.*

  Si pour tout $j \in [\![1\,;n']\!]$, $P_1 \neq Q_j$, alors d'après le théorème 3.5 p. 20, P_1 serait premier avec P, ce qui contredit le fait que P_1 divise P. Quitte à renuméroter les polynômes Q_j on peut alors supposer que $P_1 = Q_1$.

  Si $k_1 > k'_1$ alors en simplifiant par $P_1^{k_1}$ dans les deux décompositions, on obtiendrait que $P_1 = Q_i$ avec $i > 1$, ce qui est absurde. Le même argument montre que l'on ne peut avoir $k'_1 > k_1$. Donc $k_1 = k'_1$.

  Si $n = 1$ on voit en simplifiant par $P_1^{k_1}$ dans les deux décompositions que $n' = 1$. De même $n = 1$ si $n' = 1$. Dans ce cas l'unicité de la décomposition de P est établie.

  Supposons maintenant que $\inf(n, n') > 1$. On a $\prod_{i=2}^{n} P_i^{k_i} = \prod_{j=2}^{n'} Q_j^{k'_j}$ et d'après l'hypothèse de récurrence on a $n = n'$. On peut renuméroter les Q_j de sorte que $Q_j = P_j$ et $k_j = k'_j$ pour $j \in [\![2\,;n]\!]$. L'unicité à l'ordre près des facteurs de la décomposition donnée par le théorème est donc établie par récurrence.

□

Le théorème de D'Alembert (démontré en fait par Gauss) montre que tout polynôme non constant à coefficients complexes possède au moins une racine dans $\mathbb{C}$. On en déduit immédiatement que les seuls polynômes irréductibles dans $\mathbb{C}[x]$ sont les polynômes de degré 1.

Soit maintenant P un polynôme irréductible dans $\mathbb{R}[x]$, et soit λ une racine complexe de P. Si $\lambda \in \mathbb{R}$, P est divisible par $x - \lambda$ dans $\mathbb{R}[x]$, et il existe $a \in \mathbb{C}^\star$ tel que $P = a(x - \lambda)$. Si λ n'est pas réel alors $P(\overline{\lambda}) = \overline{P(\lambda)} = 0$, donc P est aussi divisible par $(x - \overline{\lambda})$ dans $\mathbb{C}[x]$. Mais $(x - \lambda)$ et $(x - \overline{\lambda})$ sont premiers entre eux dans $\mathbb{C}[x]$, donc leur ppmc est $(x - \lambda)(x - \overline{\lambda}) = x^2 + bx + c$, avec $b = -2\Re(\lambda)$ et $c = \lambda\overline{\lambda} = |\lambda|^2$. Comme la division

euclidienne donne les mêmes résultats dans $\mathbb{R}[x]$ et dans $\mathbb{C}[x]$ pour les polynômes à coefficients réels, on voit que P est divisible par x^2+bx+c dans $\mathbb{R}[x]$. On a donc $P = a(x^2+bx+c)$ avec a, b, c réels et $a \neq 0$. De plus $\Delta = b^2 - 4c = 4\Re(\lambda)^2 - |\lambda|^2 < 0$.

Réciproquement il est clair que tout polynôme de la forme ci-dessus est irréductible dans $\mathbb{R}[x]$. Le théorème précédent prend alors la forme concrète suivante pour les polynômes à coefficients réels ou complexes.

Théorème 3.10

1. Pour tout polynôme non constant $P \in \mathbb{C}[x]$, il existe un entier $n \in \mathbb{N}$, une famille $(\lambda_1, \ldots, \lambda_n)$ de nombres complexes distincts, une famille $(k_1, \ldots, k_n)$ d'entiers positifs et un complexe $a \neq 0$ tels que
$$P = a\prod_{i=1}^{n}(x-\lambda_i)^{k_i}.$$
Cette décomposition est unique à l'ordre près des facteurs.
2. Pour tout polynôme non constant $P \in \mathbb{R}[x]$, il existe $(m,n) \in \mathbb{N}^2$, une famille $(\lambda_1, \ldots, \lambda_m)$ de nombres réels distincts, une famille $((b_1, c_1), \ldots, (b_m, c_m))$ de paires de réels, avec pour tout $i \in [\![1\,;m]\!]$, $b_i^2 - 4c_i < 0$, deux familles $(k_1, \ldots, k_m)$ et $(k'_1, \ldots, k'_n)$ d'entiers positifs et un réel $a \neq 0$ tels que
$$P = a\prod_{i=1}^{m}(x-\lambda_i)^{k_i} \times \prod_{j=1}^{n}(x^2+b_jx+c_j)^{k'_j}.$$
On peut avoir $m = 0$ ou $n = 0$ (l'une des familles $(\lambda_1, \ldots, \lambda_m)$ ou $((b_1, c_1), \ldots, (b_m, c_m))$ est vide), et cette décomposition est unique à l'ordre près des facteurs.

Exemple 3.7

La décomposition de x^3+1 dans $\mathbb{R}[x]$ est $x^3+1 = (x+1)(x^2-x+1)$.

La décomposition de x^3+1 dans $\mathbb{C}[x]$ est $x^3+1 = \left(x+1\right)\left(x-\frac{1}{2}+\imath\frac{\sqrt{3}}{2}\right)\left(x-\frac{1}{2}-\imath\frac{\sqrt{3}}{2}\right)$.

En effet x^2-x+1 n'a pas de racines réelles, et ses racines complexes sont $\frac{1}{2} \pm \imath\frac{\sqrt{3}}{2}$.

```
>> import sympy as sp
>> sp.var("x")
x
>> P = (x**3+1).as_poly()
>> décomposition_P = P.factor_list()
>> print("Décomposition P = a × ∏P_i^{k_i} :")
Décomposition P = a × ∏P_i^{k_i} :
>> print(f"a = {décomposition_P[0]}")
a = 1
>> for _pi in décomposition_P[1]:
...    print(f"{sp.pretty(_pi[0].as_expr())}, de multiplicité {_pi[1]}")
```

De même que dans le cas des entiers, on peut utiliser la décomposition en produit de polynômes irréductibles pour calculer le pgdc et le ppmc d'une famille finie de polynômes non constants (les polynômes constants non nuls n'interviennent pas dans le calcul du ppmc et le pgdc d'une famille de polynômes contenant un polynôme constant est égal à 1).

Propriété 3.6

Soit $n \in \mathbb{N}$ et soit $(P_1, \ldots, P_n)$ une famille de polynômes non constants.
Le pgdc de cette famille s'obtient en effectuant le produit des diviseurs unitaires irréductibles communs à chacun des éléments, affectés du plus petits des exposants apparaissant dans les décompositions de ces polynômes.
Le ppmc de cette famille s'obtient en effectuant le produit de tous les diviseurs unitaires irréductibles de chacun des éléments, affectés du plus grand des exposants apparaissant dans les décompositions de ces polynômes.

Exemple 3.8

Calcul du pgdc et du ppmc de x^3+1 et de $(x^4-1)^2$.
On utilise les décompositions dans $\mathbb{R}[x]$. On a vu que la décomposition réelle de x^3+1 est $x^3+1=(x+1)(x^2-x+1)$, et la décomposition de $(x^4-1)^2$ est $(x^4-1)^2=(x+1)^2(x-1)^2(x^2+1)^2$. Le seul diviseur unitaire irréductible commun est $(x+1)$, affecté de l'exposant 1 pour le premier polynôme et de l'exposant 2 pour le second. Alors pgdc $\left(x^3+1\,;(x^4-1)^2\right)=x+1$.
Leur ppmc est égal à $(x+1)^2(x^2-x+1)(x-1)^2(x^2+1)^2=x^{10}-x^9+x^8-2x^6+2x^5-2x^4+x^2-x+1$.

Dans le cas des nombres entiers, on est confronté à des problèmes de temps de calcul pour décomposer de grands nombres en produit de facteurs premiers. En ce qui concerne les polynômes, on sait depuis l'Antiquité trouver les racines d'un polynôme de degré 2. Il a fallu attendre la deuxième moitié du XV$^{\text{e}}$ siècle (formules de Cardan et Tartaglia) pour arriver à résoudre les équations de degré 3, puis assez rapidement celles de degré 4 (ivrognes et paillards, les mathématiciens italiens de la Renaissance gardaient secrètes leurs formules et s'affrontaient moyennant espèces sonnantes et trébuchantes dans des joutes publiques où il s'agissait devant des *tiffosi* passionnés de résoudre des équations concrètes de degré 3 ou 4). C'est d'ailleurs à cette occasion qu'apparait pour la première fois le mystérieux « nombre imaginaire » $\imath$, tel que $\imath^2=-1$. C'est seulement vers 1830 qu'Évariste Galois a démontré [1] qu'*il est impossible de trouver des formules algébriques donnant les racines des polynômes de degré supérieur ou égal à* 5, en inventant au passage la théorie des groupes (ce mathématicien de génie était malheureusement myope comme une taupe. Il est mort à vingt ans en se battant en duel au pistolet pour les beaux yeux d'une jeune femme probablement soudoyée par les sbires de Louis Philippe, en cette période troublée où les polytechniciens n'avaient pas hésité à utiliser les canons de leur prestigieuse école pour pilonner les troupes royales). Le manuscrit laissé par Galois ne sera compris que des années plus tard.

On voit donc qu'il est en général impossible d'expliciter la décomposition exacte en produit de polynômes irréductibles d'un polynôme de degré supérieur ou égal à 5. Même dans le cas du polynôme $x^4+x^2+3x+1=(x+1)(x^3-x^2+2x+1)$, les formules donnant les racines d'un polynôme de degré 3 mènent à des calculs compliqués.

9 Formule de Taylor pour les polynômes

La *dérivée formelle* d'un polynôme $P\in\mathbb{C}[x]$ défini par $P(x)=a_0+a_1x+\ldots+a_nx^n$ est donnée par la formule

$$P'(x)=a_1+2a_2x+\ldots+na_nx^{n-1}. \tag{3.3}$$

On définit alors par récurrence les dérivées successives d'ordre supérieur. Notons que $P^{(k)}=0$ si $k>\deg(P)$.

On a le résultat suivant (évidemment aussi valable pour les polynômes à coefficients réels).

Théorème 3.11 : formule de Taylor pour les polynômes

Soit $P\in\mathbb{C}[x]$, soit $a\in\mathbb{C}$ et soit $n\in\mathbb{N}$.
Si $n\geq\deg(P)$, on a

$$P(x)=P(a)+P'(a)(x-a)+\ldots+\frac{P^{(n)}(a)}{n!}(x-a)^n.$$

Démonstration 3.14

Soit $p\geq 1$, et posons $U=x^p$. On a par récurrence immédiate que $U^{(k)}(a)=\prod_{i=1}^{k}(p-i+1)a^{p-k}=(k!)\binom{p}{k}a^{p-k}$ pour $k\in[\![1\,;p]\!]$, et $U^{(k)}(a)=0$ pour $k>p$.
D'après la formule du binôme de Newton on a pour $n\geq p$

1. Ce résultat a été obtenu indépendamment à la même époque par le mathématicien norvégien Abel, mort de maladie à l'âge de vingt-sept ans

$$\begin{aligned}\mathrm{U}(x) = (x-a+a)^p &= \sum_{i=0}^{p} \binom{p}{i} (x-a)^i a^{p-i} \\ &= a^p + \sum_{i=1}^{p} \binom{p}{i} a^{p-i}(x-a)^i \\ &= \mathrm{U}(a) + \sum_{i=1}^{n} \frac{\mathrm{U}^{(i)}(a)}{i!}(x-a)^i.\end{aligned}$$

Procédons par récurrence sur m le degré du polynôme P.

- *La formule est évidente pour les polynômes constants.*
- *Supposons la formule vraie pour les polynômes de degré inférieur ou égal à m, avec $m \geq 0$. Soit $\mathrm{P} \in \mathbb{C}[x]$ un polynôme de degré $m+1$. On pose* U *tel que $\mathrm{U}(x) = x^{m+1}$. Alors il existe $b \in \mathbb{C}$ et $\mathrm{R} \in \mathbb{C}[x]$ tels que $\mathrm{P} = b\mathrm{U} + \mathrm{R}$, avec $\deg(\mathrm{R}) \leq m$. Par linéarité de la dérivée, il est clair que $\mathrm{P}^{(k)}(a) = b\mathrm{U}^{(k)}(a) + \mathrm{R}^{(k)}(a)$ pour $k \in \mathbb{N}^\star$. En utilisant la remarque préliminaire sur* U *et d'après l'hypothèse de récurrence appliquée à* R *on a donc, pour $n \geq m+1$*

  $$\begin{aligned}\mathrm{P}(x) &= b\left(\mathrm{U}(a) + \sum_{i=1}^{n} \frac{\mathrm{U}^{(i)}(a)}{i!}(x-a)^i\right) + \left(\mathrm{R}(a) + \sum_{i=1}^{n} \frac{\mathrm{R}^{(i)}(a)}{i!}(x-a)^i\right) \\ &= \mathrm{P}(a) + \sum_{i=1}^{n} \frac{\mathrm{P}^{(i)}(a)}{i!}(x-a)^i.\end{aligned}$$

  L'hérédité es donc vérifiée.
- *La formule est donc démontrée par récurrence.*

□

Ce théorème permet d'obtenir le résultat suivant, en complément de la définition 3.5 (multiplicité d'une racine) p. 16.

Corollaire 3.5

Soit $\mathrm{P} \in \mathbb{C}[x]$ un polynôme non nul et soit $a \in \mathbb{C}$ une racine de P.
Alors l'ordre de multiplicité de a est égal au plus petit entier positif k tel que $\mathrm{P}^{(k)}(a) \neq 0$.

Être capable

* Être capable de conduire un schéma de résolution d'un problème mathématique sous forme formelle à l'aide de l'outil informatique * Être capable de mettre en œuvre des méthodes numériques pour résoudre un problème mathématique en maîtrisant l'erreur * Être capable de recourir à des méthodes adéquates pour inverser de grosses matrices et résoudre des systèmes linéaires de grandes tailles * Être capable de mettre en œuvre des schémas numériques implicites ou explicites basés sur la formule de Taylor pour trouver des solutions approchées d'équations aux dérivées partielles classiques comme l'équation des ondes ou l'équation de la chaleur

Chapitre 4

Fractions rationnelles

Sommaire

1 Généralités

Considérons à présent l'ensemble $\tilde{\mathbb{K}}[x]$ des *fractions rationnelles* à coefficients dans un corps $\mathbb{K}$.

Définition 4.1 : fraction rationnelle

On appelle *fraction rationnelle* un quotient de la forme $R = \frac{P}{Q}$, où P et Q sont des polynômes de $\mathbb{K}[x]$ et Q est non identiquement nulle.

On identifie $\frac{P_1}{Q_1}$ et $\frac{P_2}{Q_2}$ quand $P_1Q_2 - P_2Q_1 = 0$. Les règles de calcul (réduction au même dénominateur pour l'addition, produit, simplification par un diviseur commun de P et Q) sont les mêmes que pour les fractions usuelles et on vérifie que l'ensemble des fractions rationnelles, muni de cette addition et de ce produit est un corps.

Propriété 4.1

Soit $R \in \tilde{\mathbb{K}}[x]$ une fraction rationnelle non nulle.

1. R peut s'écrire sous la forme $R = \frac{P_0}{Q_0}$, avec $(P_0, Q_0) \in \mathbb{K}[x]^2$ tel que P_0 et Q_0 sont non nuls et premiers entre eux.
2. Les autres écritures de R sont de la forme $R = \frac{SP_0}{SQ_0}$, avec $S \in \mathbb{K}[x]$ et $S \neq 0$.

Démonstration 4.1

Soit une fraction rationnelle $R \in \tilde{\mathbb{K}}[x]$.

4.1-1 *Soit* $(P_1, Q_1) \in \mathbb{K}[x]^2$ *non nuls tels que* $R = \frac{P_1}{Q_1}$. *Notons* $D = \operatorname{pgdc}(P_1 ; Q_1)$. *Alors il existe* $(P_0, Q_0) \in \mathbb{K}[x]^2$ *tel que* $P_1 = DP_0$ *et* $Q_1 = DQ_0$. *Alors* $R = \frac{P_0}{Q_0}$ *et* $\operatorname{pgdc}(P_0 ; Q_0) = 1$.

4.1-2 *Soit* $R = \frac{P_0}{Q_0}$ *la forme irréductible d'une fraction rationnelle. Soit* $(P, Q) \in \mathbb{K}[x]^2$ *tel que* $R = \frac{P}{Q}$ *une autre écriture de* R *(avec* P *et* Q *non nuls). Alors* $PQ_0 = P_0Q$. *Comme* P_0 *et* Q_0 *sont premiers entre eux, il existe* $S \in \mathbb{K}[x]$ *tel que* $P = SP_0$ *et* $Q = SQ_0$.

□

L'écriture donnée par 4.1-1 est appelée *forme irréductible* de R. D'après la propriété 4.1-2, les autres formes irréductibles de R sont données par $R = \frac{\lambda P_0}{\lambda Q_0}$, avec $\lambda \in \mathbb{R}^\star$.

Définition 4.2 : degré d'une fraction rationnelle

Soit une fraction rationnelle $R \in \tilde{\mathbb{K}}[x]$. On suppose donc qu'il existe $(P, Q) \in \mathbb{K}[x]^2$ tel que $R = \frac{P}{Q}$ avec $Q \neq 0$.

1. Si $R \neq 0$, on appelle *degré* de R l'entier relatif $\deg(R) = \deg(P) - \deg(Q)$.
2. Si $R = 0$, on pose par convention $\deg(R) = -\infty$.

On a donc $\deg(R) \in \mathbb{Z} \bigcup \{+\infty\}$.

Propriété 4.2

Soit une fraction rationnelle $R \in \tilde{\mathbb{K}}[x]$.
R s'écrit de manière unique $R = U + \tilde{R}$ avec $U \in \mathbb{K}[x]$ et $\tilde{R} \in \tilde{\mathbb{K}}[x]$ tel que $\deg\left(\tilde{R}\right) < 0$.
Le polynôme U est appelé *partie entière* de R.

Démonstration 4.2

L'existence et l'unicité de cette écriture provient directement de l'unicité de la division euclidienne. Soit $(P, Q) \in \mathbb{K}[x]^2$ *tel que* $R = \frac{P}{Q}$, *avec* $Q \neq 0$. *Alors il existe une unique paire* $(U, B) \in \mathbb{K}[x]^2$ *avec* $\deg(B) < \deg(Q)$ *telle que* $P = UQ + B$. *Ainsi* $R = U + \frac{B}{Q}$. *En posant* $\tilde{R} = \frac{B}{Q}$, *on a bien* $\deg\left(\tilde{R}\right) < 0$.

□

Définition 4.3 : zéro et pôle

Soit une fraction rationnelle irréductible de la forme $R = \frac{P}{Q}$.

- On appelle *zéro* de R toute racine de P ;
- toute racine de Q est un *pôle* de R.

Si à R on associe la *fonction rationnelle* $R : x \mapsto \frac{P(x)}{Q(x)}$, son domaine de définition est donc $\mathbb{K}$ privé des pôles de R :

$$\mathscr{D}_R = \mathbb{K} \setminus \{x \in \mathbb{K} \mid Q(x) = 0\}.$$

Propriété 4.3

Soient R_1 et R_2 deux fonctions rationnelles. Alors $\forall x \in \mathscr{D}_{R_1} \cap \mathscr{D}_{R_2}$:

- $(R_1 + R_2)(x) = R_1(x) + R_2(x)$;
- $(R_1 R_2)(x) = R_1(x) R_2(x)$.

2 Décomposition en éléments simples d'une fraction rationnelle

On utilisera dans la suite le résultat suivant.

Lemme 4.1 : division suivant les puissances croissantes

Soit P un polynôme et soit Q un polynôme non constant. Soit $n \in \mathbb{N}^\star$.
Il existe une unique famille $(R_0, \ldots, R_n)$ telle que

1. pour tout $i \in [\![0 ; n-1]\!]$, $\deg\left(R_i\right) < \deg(Q)$;
2. $P = \sum_{i=0}^{n} R_i Q^i$.

Démonstration 4.3

Procédons par récurrence sur n.

- *Pour obtenir la décomposition ci-dessus pour* $n = 1$, *il suffit d'effectuer la division euclidienne de* P *par* Q : *il existe une unique paire* $(R_0, R_1) \in \mathbb{K}[x]^2$ *telle que* $P = R_1 Q + R_0$, *avec* $\deg\left(R_0\right) < \deg(Q)$.
- *Supposons la décomposition obtenue de manière unique pour un certain* $n \geq 1$. *En effectuant la division euclidienne de* R_n *par* Q *on obtient une unique paire* $(B, R) \in \mathbb{K}[x]^2$ *telle que* $R_n = BQ + R$, *avec* $\deg(R) < \deg(Q)$. *Par suite, et en utilisant l'hypothèse de récurrence,* $P = \sum_{i=0}^{n-1} \tilde{R}_i Q^i + Q^n (BQ + R)$. *On pose* $R_{n+1} = B$, $R_n = \tilde{R} + R$ *et pour* $i \in [\![0 ; n-1]\!]$, $R_i = \tilde{R}_i$. *Alors la propriété est vraie au rang* $n + 1$.
- *Ainsi l'existence et l'unicité de la décomposition cherchée sont établies par récurrence.*

□

Théorème 4.1

Soit $R = \frac{P}{Q}$ une fraction rationnelle, avec $(P, Q) \in \mathbb{K}[x]^2$, $Q \neq 0$.
Alors il existe un unique $a \in \mathbb{K}^\star$, une unique famille $(Q_1, \ldots, Q_n)$ de n polynômes unitaires distinct non nuls et premiers entre eux deux à deux tels que $Q = a\prod_{i=1}^{n} Q_i$ et une unique famille $(P_0, \ldots, P_n)$ tels que pour tout $i \in [\![1\,;n]\!]$ $\deg(P_i) < \deg(Q_i)$ et

$$R = P_0 + \sum_{i=1}^{n} \frac{P_i}{Q_i} \tag{4.1}$$

Démonstration 4.4

Le théorème 3.9 p. 24 assure que la factorisation de Q *existe. En effet, comme* Q *est un polynôme non constant (dans le cas contraire, la factorisation est immédiate), il existe un élément non nul* $a \in \mathbb{K}$*, un entier non nul* n*, une famille* $(\tilde{Q}_1, \ldots, \tilde{Q}_n)$ *de polynômes unitaires irréductibles distincts et une famille* $(k_1, \ldots, k_n)$ *d'entiers positifs tels que* $Q = a\prod_{i=1}^{n} \tilde{Q}_i^{k_i}$*. De plus cette décomposition est unique (à l'ordre des facteurs près). On pose donc, pour* $i \in [\![1\,;n]\!]$, $Q_i = \tilde{Q}_i^{k_i}$.
Montrons l'existence et l'unicité de la famille $(P_1, \ldots, P_n)$ *par récurrence sur* n*, le nombre de polynômes de la factorisation de* Q.

- *Si* $n = 1$*, alors* $Q = aQ_1$ *et* $R = \frac{P}{Q} = \frac{P}{aQ_1}$*. D'après le lemme 4.1 (division suivant les puissances croissantes) p. 30, il existe une unique famille* (R_0, R_1) *telle que* $\deg(R_0) < \deg(Q_1)$ *et* $P = \sum_{i=0}^{1} R_i Q_1^i = R_0 + R_1 Q_1$*. Donc* $R = \frac{1}{a}\left(R_1 + \frac{R_0}{Q_1}\right)$*. On identifie* $P_0 = \frac{R_1}{a}$ *et* $P_1 = \frac{R_0}{a}$*. La propriété est donc vérifiée au rang* 1.
- *Supposons que la décomposition existe et est unique pour toute fraction rationnelle dont le dénominateur se factorise au plus* n *polynômes unitaires distinct non nuls et premiers entre eux deux à deux. Considérons* $R = \frac{P}{Q}$ *avec* $Q = a\prod_{i=1}^{n+1} Q_i$*. Comme les différents polynômes* Q_i *de cette factorisation sont distincts et premiers deux à deux, par le théorème 3.5 p. 20,* Q_{n+1} *est premier avec le produit* $\prod_{i=1}^{n} Q_i$.
  Donc par le corollaire 3.3 p. 20, il existe un unique couple $(U, V) \in \mathbb{K}[x]^2$ *tel que* $UQ_{n+1} + V\prod_{i=1}^{n} Q_i = 1$*, avec* $\deg(V) < \deg(Q_{n+1})$*. Ainsi* $R = \frac{P(UQ_{n+1} + V\prod_{i=1}^{n} Q_i)}{\prod_{i=1}^{n+1} Q_i} = \frac{PU}{\prod_{i=1}^{n+1} Q_i} + \frac{PV}{Q_{n+1}}$*. Les deux termes de cette sommes vérifient les conditions de l'hypothèse de récurrence. On a donc deux familles* $(P_0, \ldots, P_n)$ *et* $(\tilde{P}_0, \tilde{P}_1)$ *telles que* $R = P_0 + \sum_{i=1}^{n} \frac{P_i}{Q_i} + \tilde{P}_0 + \frac{\tilde{P}_1}{Q_{n+1}}$*. Et l'hérédité est établie.*
- *L'existence et l'unicité de la décomposition sont démontrées par récurrence.*

□

Théorème 4.2 : décomposition en éléments simples d'une fraction rationnelle

Soit $\mathbb{K}$ un corps, soit $R = \frac{P}{Q}$ une fraction rationnelle irréductible à coefficients dans K, avec P et Q premiers entre eux. Soit $a \in \mathbb{K}$, soit $(Q_1, \ldots, Q_n)$ de polynômes unitaires irréductibles distincts et soit une famille $(k_1, \ldots, k_n)$ d'entiers positifs tels que $Q = a\prod_{i=1}^{n} Q_i^{k_i}$, la décomposition de Q en produit de polynômes irréductibles unitaires distincts dans $\mathbb{K}[x]$.
Alors il existe $U \in \mathbb{K}[x]$ et pour $i \in [\![1\,;n]\!]$, n familles $\left(A_{i,1}, \ldots, A_{i,k_i}\right)$ de polynômes à coefficients dans $\mathbb{K}$, avec pour $i \in [\![1\,;n]\!]$ et $j \in [\![1\,;k_i]\!]$, $\deg(A_{i,j}) < \deg(Q_i)$ tels que

$$\frac{P}{Q} = U + \sum_{i=1}^{n}\left(\sum_{j=1}^{k_i} \frac{A_{i,j}}{Q_i^j}\right)$$

La décomposition ci-dessus est unique.

Démonstration 4.5

On peut se limiter au cas où Q *est unitaire, ce qui revient à prendre* $a = 1$.
Montrons l'existence et l'unicité de la décomposition par récurrence sur n*, le nombre de polynômes irréductibles unitaires et distincts de la décomposition du dénominateur* Q *sur* $\mathbb{K}[x]$.

— *Supposons que* $Q = Q_1^{k_1}$, *avec* Q_1 *irréductible et* $k_1 \in \mathbb{N}^\star$. *On obtient alors la décomposition en éléments simples en effectuant la division de* P *suivant les puissances croissantes de* Q_1 *à l'ordre* k_1.
En effet, d'après le lemme 4.1 (division suivant les puissances croissantes) p. 30, il existe une unique famille $\left(R_0, \ldots, R_{k_1}\right)$ *telle que pour tout* $i \in [\![0\,;k_1-1]\!]$, $\deg\left(R_i\right) < \deg\left(Q_1\right)$ *et* $P = \sum_{i=0}^{k_1} R_i Q_1^i$.
Donc $R = \frac{\sum_{i=0}^{k_1} R_i Q_1^i}{Q_1^{k_1}} = \frac{R_0}{Q_1^{k_1}} + \frac{R_1}{Q_1^{k_1-1}} + \ldots + \frac{R_{k_1-1}}{Q_1} + R_{k_1}$. *Il suffit alors de poser* $U = R_{k_1}$ *et, pour* $i \in [\![1\,;k_1]\!]$, $A_{1,i} = R_{k_1-i}$, *avec* $\deg\left(A_{1,i}\right) < \deg\left(Q_1\right)$.

— *Supposons que la décomposition en éléments simples d'une fraction rationnelle irréductible existe et est unique quand son dénominateur* Q *possède* n *diviseurs unitaires irréductibles distincts, avec* $n \geq 1$.
Soit $R = \frac{P}{Q}$, *avec* Q *un polynôme unitaire possédant* $n+1$ *diviseurs irréductibles unitaires distincts. On a donc une famille* $\left(k_1, \ldots, k_{n+1}\right)$ *d'entiers positifs et une famille* $\left(Q_1, \ldots, Q_{n+1}\right)$ *de polynômes irréductibles unitaires distincts telles que* $Q = \prod_{i=1}^{n+1} Q_i^{k_i}$. *Pour* $i \in [\![1\,;n+1]\!]$, *on pose* $\tilde{Q}_i = Q_i^{k_i}$.
On a $R = \frac{P}{\tilde{Q}_{n+1}\prod_{i=1}^{n}\tilde{Q}_i} = \frac{1}{\tilde{Q}_{n+1}}\left(U + \sum_{i=1}^{n}\left(\sum_{j=1}^{k_i}\frac{A_{i,j}}{Q_i^j}\right)\right)$, *avec, pour tout* $i \in [\![1\,;n]\!]$, *pour tout* $j \in [\![1\,;k_i]\!]$, $\deg\left(A_{i,j}\right) < \deg\left(Q_i\right)$, *d'après l'hypothèse de récurrence.*
Or, d'après le lemme 4.1 p. 30, il existe

  — *une unique famille* $\left(R_0, \ldots, R_{k_{n+1}}\right)$ *telle que pour tout* $i \in [\![0\,;k_{n+1}-1]\!]$, $\deg\left(R_i\right) < \deg\left(Q_{n+1}\right)$ *et* $U = \sum_{i=0}^{k_{n+1}} R_i Q_{n+1}^i$;
  — *pour tout* $i \in [\![1\,;n]\!]$, *pour tout* $j \in [\![1\,;k_i]\!]$, *une unique famille* $\left(M_{i,j,0}, \ldots, M_{i,j,k_{n+1}}\right)$ *telle que pour tout* $m \in [\![0\,;k_{n+1}-1]\!]$, $\deg\left(M_{i,j,m}\right) < \deg\left(Q_{n+1}\right)$ *et* $A_{i,j} = \sum_{m=0}^{k_{n+1}} M_{i,j,m} Q_{n+1}^m$.

Ainsi $P = \dfrac{R_0}{Q_{n+1}^{k_{n+1}}} + \dfrac{R_1}{Q_{n+1}^{k_{n+1}-1}} + \ldots + \dfrac{R_{k_{n+1}-1}}{Q_{n+1}} + R_{k_{n+1}} +$

$$\sum_{i=1}^{n}\left(\sum_{j=1}^{k_i}\frac{1}{Q_i^j}\left(\frac{M_{i,j,0}}{Q_{n+1}^{k_{n+1}}} + \frac{M_{i,j,1}}{Q_{n+1}^{k_{n+1}-1}} + \ldots + \frac{M_{i,j,k_{n+1}-1}}{Q_{n+1}} + M_{i,j,k_{n+1}}\right)\right) = V + \sum_{i=1}^{n+1}\left(\sum_{j=1}^{k_i}\frac{B_{i,j}}{Q_i^j}\right)$$

Il suffit en effet de poser $V = R_{k_{n+1}}$, *pour* $j \in [\![1\,;k_{n+1}]\!]$, $B_{n+1,j} = R_{k_{n+1}-j}$ *et pour* $i \in [\![1\,;n]\!]$, $j \in [\![1\,;k_i]\!]$, $B_{i,j} = \sum_{m=0}^{k_{n+1}} M_{i,j,m} Q_{n+1}^{m-k_{n+1}}$. *Et cette décomposition est unique.*
Reste à vérifier les degrés des polynômes $B_{i,j}$.

  — *Si* $i = n+1$, *par construction on a bien* $\deg\left(B_{n+1,j}\right) < \deg\left(Q_{n+1}\right)$.
  — *Si* $i \in [\![1\,;n]\!]$, *on se rappelle que, pour tout* $j \in [\![1\,;k_i]\!]$, $\deg\left(A_{i,j}\right) < \deg\left(Q_i\right)$ *et* $A_{i,j} = \sum_{m=0}^{k_{n+1}} M_{i,j,m} Q_{n+1}^m$, *donc* $\deg\left(A_{i,j}\right) < (1+k_{n+1})\deg\left(Q_{n+1}\right)$. *Comme* $B_{i,j}$ *s'écrit* $\frac{A_{i,j}}{Q_{n+1}^{k_{n+1}}}$, *on a alors* $\deg\left(B_{i,j}\right) < \deg\left(A_{i,j}\right) - \deg\left(Q_{n+1}\right)$, *soit* $\deg\left(B_{i,j}\right) < \deg\left(Q_i\right)$.

La décomposition existe donc au rang $n+1$ *et elle est unique.*

— *L'existence et l'unicité de la décomposition en éléments simples sont donc établies par récurrence.*

□

Corollaire 4.1

Soit la fraction rationnelle $R = \frac{P}{Q}$ avec $(P, Q) \in \mathbb{C}[x]^2$ premiers entre eux.
Il existe un unique élément non nul $a \in \mathbb{C}$, un unique entier non nul n, une unique famille $\left(\lambda_1, \ldots, \lambda_n\right)$ de complexes distincts et une unique famille $\left(k_1, \ldots, k_n\right)$ d'entiers positifs tels que $Q = a\prod_{i=1}^{n}(x-\lambda_i)^{k_i}$.
Alors il existe $U \in \mathbb{C}[x]$ et pour $i \in [\![1\,;n]\!]$, n familles $\left(a_{i,1}, \ldots, a_{i,k_i}\right)$ de complexes tels que la décomposition en éléments simples dans $\tilde{\mathbb{C}}[x]$ de R s'écrit

$$R = \frac{P}{Q} = U + \sum_{i=1}^{n}\left(\sum_{j=1}^{k_i}\frac{a_{i,j}}{(x-\lambda_i)^j}\right). \tag{4.2}$$

Cette décomposition est unique.

Démonstration 4.6

L'existence et l'unicité de la décomposition de Q *viennent du théorème 3.9 p. 24, ainsi que du théorème de* d'Alembert.
Il suffit ensuite d'appliquer le théorème 4.2 (décomposition en éléments simples d'une fraction rationnelle) p. 31.

□

Corollaire 4.2

Soit la fraction rationnelle $R = \frac{P}{Q}$ avec $(P, Q) \in \mathbb{R}[x]^2$ premiers entre eux.
Il existe
- un unique élément non nul $a \in \mathbb{R}$,une unique paire $(m, n) \in \mathbb{N}^2$,
- une unique famille $(\lambda_1, \ldots, \lambda_m)$ de réels distincts et une unique famille $(\mu_1, \ldots, \mu_m)$ d'entiers positifs,
- une unique famille $((\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n))$ de paires de réels distinctes et une unique famille $(\nu_1, \ldots, \nu_m)$ d'entiers positifs

tels que $Q(x) = a\prod_{i=1}^{m}(x-\lambda_i)^{\mu_i}\prod_{k=1}^{n}(x^2+\alpha_k x+\beta_k)^{\nu_k}$.
Alors il existe
- $U \in \mathbb{R}[x]$,
- pour $i \in [\![1\,;m]\!]$, m familles $\left(a_{i,1}, \ldots, a_{i,\mu_i}\right)$ de réels,
- pour $j \in [\![1\,;n]\!]$, n familles $\left(\left(b_{j,1}, c_{j,1}\right), \ldots, \left(b_{j,\mu_i}, c_{j,\mu_j}\right)\right)$ de paire de réels

tels que la décomposition en éléments simples dans $\tilde{\mathbb{R}}[x]$ de R s'écrit

$$R = \frac{P}{Q} = U + \sum_{i=1}^{m}\left(\sum_{j=1}^{\mu_i}\frac{a_{i,j}}{(x-\lambda_i)^j}\right) + \sum_{k=1}^{n}\left(\sum_{j=1}^{\nu_k}\frac{b_{k,j}x+c_{k,j}}{(x^2+\alpha_k x+\beta_k)^j}\right). \tag{4.3}$$

Cette décomposition est unique.

Démonstration 4.7

La démonstration est similaire à la précédente.

□

! En pratique, la décomposition en éléments simples d'une fraction rationnelle $R : x \mapsto \frac{P(x)}{Q(x)}$ suit toujours le même schéma :
1. déterminer la factorisation du numérateur P et du dénominateur Q et procéder aux éventuelles simplifications
2. si $\deg(P) \geq \deg(Q)$, opérer la division euclidienne pour obtenir $R = U + \frac{\tilde{P}}{Q}$ (avec $\deg(\tilde{P}) < \deg(Q)$), puis décomposer $\frac{\tilde{P}}{Q}$
3. les pôles de R donnent ensuite la structure de la décomposition, faisant apparaître un certains nombre de paramètres à calculer
4. quatre approches permettent de calculer ces derniers :
   - multiplier par le facteur $(x-x_k)\mu_k$ puis évaluer en x_k, y compris lorsque cette racine est un complexe pur,
   - multiplier par x puis passer à la limite en $+\infty$,
   - évaluer en un point,
   - mettre au même dénominateur et identifier.

☛ **Exemple 4.1**

Décomposition en éléments simples de $R(x) = \frac{x^2}{x^3+2x^2-x-2}$.
Si le numérateur $P(x) = x^2$ est factorisé, il n'en va pas de même pour le dénominateur $Q(x) = x^3 + 2x^2 - x - 2$. Ce polynôme admet 1 et -1 comme racines évidentes. Il peut donc se factoriser par $(x-1)(x+1) = x^2 - 1$, d'après le corollaire 3.1 p. 16. Pour trouver le dernier facteur, il suffit d'opérer une division euclidienne. On obtient ainsi $Q(x) = (x-1)(x+1)(x+2)$.
Par conséquent, comme $\deg(P) < \deg(Q)$, la partie entière de la division euclidienne de P par Q est nulle et la structure de la décomposition en éléments simples est donnée par $R(x) = \frac{\alpha}{x-1} + \frac{\beta}{x+1} + \frac{\gamma}{x+2}$, avec $(\alpha, \beta, \gamma) \in \mathbb{R}^3$.

— En multipliant toute l'égalité précédente par $x-1$, on a $\frac{x^2}{(x+1)(x+2)} = \alpha + (x-1)\left(\frac{\beta}{x+1} + \frac{\gamma}{x+2}\right)$. On évalue ensuite en $x=1$ la racine associée. Il vient alors $\alpha = \frac{1^2}{(1+1)(1+2)} = \frac{1}{6}$.
— En multipliant toute l'égalité par $x+1$, on a $\frac{x^2}{(x-1)(x+2)} = \beta + (x+1)\left(\frac{\alpha}{x-1} + \frac{\gamma}{x+2}\right)$. On évalue ensuite en $x=-1$ la racine associée. Il vient alors $\beta = \frac{(-1)^2}{(-1-1)(-1+2)} = \frac{-1}{2}$.
— En multipliant toute l'égalité par $x+2$, on a $\frac{x^2}{(x-1)(x+1)} = \gamma + (x+2)\left(\frac{\alpha}{x-1} + \frac{\beta}{x+1}\right)$. On évalue ensuite en $x=-2$ la racine associée. Il vient alors $\gamma = \frac{(-2)^2}{(-2-1)(-2+1)} = \frac{4}{3}$.

Alors $R(x) = \frac{1}{6}\frac{1}{x-1} - \frac{1}{2}\frac{1}{x+1} + \frac{4}{3}\frac{1}{x+2}$.

Exemple 4.2

Décomposition en éléments simples de $R(x) = \frac{x^7+2x^5+4x^2-x+1}{x^2(x^2+1)^2}$.
On pose $R = \frac{P}{Q}$. Comme $\deg(P) \geq \deg(Q)$, il faut commencer par une division euclidienne pour extraire la partie entière.

$$\begin{array}{r|l} x^7+2x^5 \quad +4x^2-x+1 & x^6+2x^4+x^2 \\ \hline -x^7\ -2x^5\ -x^3 & x \\ \quad -x^3+4x^2-x+1 & \end{array}$$

Donc $R(x) = x + \frac{-x^3+4x^2-x+1}{x^2(x^2+1)^2} = x + \tilde{R}(x)$, et la nouvelle fraction rationnelle est sous forme irréductible.
Le dénominateur Q est écrit sous sa forme factorisée dans $\mathbb{R}[x]$. La structure de la décomposition en éléments simples est donnée par $\tilde{R}(x) = \frac{\alpha_1}{x} + \frac{\alpha_2}{x^2} + \frac{\beta_1 x+\gamma_1}{x^2+1} + \frac{\beta_2 x+\gamma_2}{(x^2+1)^2}$, avec $(\alpha_1,\alpha_2,\beta_1,\gamma_1,\beta_2,\gamma_2) \in \mathbb{R}^6$.

— En multipliant toute l'égalité précédente par x^2, on a $\frac{-x^3+4x^2-x+1}{(x^2+1)^2} = \alpha_2 + x^2\left(\frac{\alpha_1}{x} + \frac{\beta_1 x+\gamma_1}{x^2+1} + \frac{\beta_2 x+\gamma_2}{(x^2+1)^2}\right)$. On évalue ensuite en $x=0$ la racine associée. Il vient alors $\alpha_2 = 1$.
— En multipliant toute l'égalité précédente par $(x^2+1)^2$, on a $\frac{-x^3+4x^2-x+1}{x^2} = (x^2+1)^2\left(\frac{\alpha_1}{x} + \frac{\alpha_2}{x^2} + \frac{\beta_1 x+\gamma_1}{x^2+1}\right) + \beta_2 x + \gamma_2$. On évalue ensuite en $x = \imath$ une racine associée. Il vient alors $\imath\beta_2 + \gamma_2 = 3$, donc $\beta_2 = 0$ et $\gamma_2 = 3$ (et ce sont bien des réels !).
— On ne peut pas utiliser cette méthode pour trouver les autres paramètres.

On se ramène donc à $\tilde{R}(x) = \frac{\alpha_1}{x} + \frac{1}{x^2} + \frac{\beta_1 x+\gamma_1}{x^2+1} + \frac{3}{(x^2+1)^2}$.
Si l'on évalue cette égalité en $x = 2\imath$, on trouve après simplification que $\alpha_1 + \frac{4\beta_1}{3} - \frac{2\imath\gamma_1}{3} = \frac{1+2\imath}{3}$. Par conséquent, on a $3\alpha_1 + 4\beta_1 = 1$ et $\gamma_1 = -1$.
Si l'on évalue l'égalité en $x=1$, on trouve après simplification que $\alpha_1 + \frac{\beta_1}{2} = \frac{-1}{2}$.
En combinant les deux équations, on obtient facilement que $\alpha_1 = -1$ et $\beta_1 = 1$.
Alors $R(x) = x - \frac{1}{x} + \frac{1}{x^2} + \frac{x-1}{x^2+1} + \frac{3}{(x^2+1)^2}$.
Une autre méthode consiste à appliquer un certain nombre de fois le théorème de Bézout et d'utiliser la division selon les puissances croissantes.
On a ici $(x^2+1)^2 = x^4+2x^2+1 = x^2(x^2+2)+1$. On en déduit que $P(x^2+1)^2 = (x^2(x^2+2)+1)P$, soit $P = (x^2+1)^2 P - x^2(x^2+2)P$. On a donc

$$R = \frac{P}{x^2} - \frac{(x^2+2)P}{(x^2+1)^2}$$

$$= x^5 + 2x^3 + 4 - \frac{1}{x} + \frac{1}{x^2} - \frac{x^9+4x^7+4x^5+4x^4-x^3+9x^2-2x+2}{(x^2+1)^2}$$

On effectue alors la division euclidienne correspondant à la fraction rationnelle de l'équation précédente :

$$
\begin{array}{l|l}
x^9+4x^7+4x^5+4x^4\ -x^3+9x^2-2x+2 & \left(x^2+1\right)^2 \\
\hline
-x^9\ -2x^7\quad -x^5 & x^5+2x^3-x+4 \\
\quad 2x^7\ +3x^5+4x^4\ -x^3+9x^2-2x+2 & \\
\quad -2x^7\ -4x^5\qquad -2x^3 & \\
\qquad -x^5\ +4x^4-3x^3+9x^2-2x+2 & \\
\qquad\ x^5\qquad +2x^3\qquad +x & \\
\qquad\qquad 4x^4\ -x^3+9x^2\ -x+2 & \\
\qquad\qquad -4x^4\qquad -8x^2\qquad -4 & \\
\qquad\qquad\quad -x^3\ +x^2\ -x-2 &
\end{array}
$$

En reportant, on a ainsi

$$
\begin{aligned}
\mathrm{R} &= x^5+2x^3+4-\frac{1}{x}+\frac{1}{x^2}-\left(x^5+2x^3-x+4+\frac{-x^3+x^2-x-2}{\left(x^2+1\right)^2}\right) \\
&= x-\frac{1}{x}+\frac{1}{x^2}+\frac{x^3-x^2+x+2}{\left(x^2+1\right)^2}
\end{aligned}
$$

On effectue alors la division euclidienne correspondant à la fraction rationnelle de l'équation précédente :

$$
\begin{array}{l|l}
x^3-x^2+x+2 & x^2+1 \\
\hline
-x^3\qquad -x & x-1 \\
\quad -x^2\quad +2 & \\
\quad\ x^2\qquad +1 & \\
\qquad\quad 3 &
\end{array}
$$

En reportant, on a ainsi la décomposition cherchée

$\mathrm{R} = x-\frac{1}{x}+\frac{1}{x^2}+\left(\frac{x-1}{x^2+1}+\frac{3}{\left(x^2+1\right)^2}\right)$.

Cette décomposition en éléments simples est utile pour calculer des intégrales. Elle est également utilisée pour déterminer les transformées de Laplace inverses.

Chapitre 5

Application aux matrices

Sommaire

1 Diagonalisation

Soit f une application linéaire sur un espace vectoriel E de dimension finie. On peut associer une matrice à cette application. L'objectif de cette section est de trouver une base $\mathscr{B}$ de E telle que $\mathscr{M}_{u;\mathscr{B}}$ soit diagonale ou triangulaire.

1.1 Spectre d'un endomorphisme

Définitions, exemples

Définition 5.1 : valeur propre, vecteur propre, espace propre

Soit E un $\mathbb{K}$-ev et soit $f \in \mathscr{L}(\mathrm{E})$.
- Soit $\lambda \in \mathbb{K}$. On dit que λ est *valeur propre* de f quand il existe un vecteur x de E non nul tel que $f(x) = \lambda x$.
- Soit $x \in \mathrm{E}$ non nul. On dit que x est *vecteur propre* de f s'il existe $\lambda \in \mathbb{K}$ tel que $f(x) = \lambda x$. Dans ce cas on dit que x est vecteur propre associé à la valeur propre λ.
- Si λ est valeur propre de f, on appelle *espace propre* associé à λ et on note E_λ l'espace vectoriel $\mathrm{E}_\lambda = \{x \in \mathrm{E} \mid f(x) = \lambda x\} = \left\{\text{vecteurs propres associés à } \lambda\right\} \cup \{0\}$.

! Un vecteur nul ne peut être vecteur propre.

☛ **Exemple 5.1**

Soit f un projecteur ($f^2 = f$). On suppose que $f \neq 0$ et que $f \neq \mathrm{id}_\mathrm{E}$. Si λ est valeur propre de f, alors $\exists x \in \mathrm{E} \setminus \{0\}$ tel que $f(x) = \lambda x$.
Comme f est un projecteur, on a donc $f^2(x) = \lambda f(x) = \lambda^2 x$, mais aussi $f^2(x) = f(x) = \lambda x$. D'où il vient que $\lambda^2 - \lambda = 0$, car $x \neq 0$.
Les valeurs propres de f sont donc $\lambda_1 = 0$ et $\lambda_2 = 1$.
Les vecteurs propres de f associés à $\lambda_1 = 0$ sont tous les x de E non nuls tels que $f(x) = 0 \cdot x = 0$: $\mathrm{E}_{\lambda_1} = \mathrm{Ker}\,(f)$, c'est l'espace vectoriel donnant la direction de projection.
Les vecteurs propres de f associés à $\lambda_2 = 1$ sont tous les x de E non nuls tels que $f(x) = 1 \cdot x = x$: $\mathrm{E}_{\lambda_2} = \mathrm{Im}\,(f)$, c'est l'espace vectoriel sur lequel on projette.

Définition 5.2 : spectre

On appelle *spectre* d'un endomorphisme f et on note $\mathrm{Sp}(f)$ l'ensemble des valeurs propres de f.

Caractérisation des valeurs propres

Propriété 5.1

Soit $f \in \mathscr{L}(\mathrm{E})$. Alors $\lambda \in \mathbb{K}$ est valeur propre

1. si et seulement si $\exists x \in \mathrm{E}\backslash\{0\}$ tel que $f(x) = \lambda x$
2. si et seulement si $\exists x \in \mathrm{E}\backslash\{0\}$ tel que $f(x) - \lambda x = 0$
3. si et seulement si $\exists x \in \mathrm{E}\backslash\{0\}$ tel que $x \in \mathrm{Ker}\left(f - \lambda\,\mathrm{id}_{\mathrm{E}}\right)$
4. si et seulement si $\mathrm{Ker}\left(f - \lambda\,\mathrm{id}_{\mathrm{E}}\right) \neq \{0\}$
5. si et seulement si $f - \lambda\,\mathrm{id}_{\mathrm{E}}$ n'est pas injective
6. si et seulement si $\det\left(f - \lambda\,\mathrm{id}_{\mathrm{E}}\right) = 0$ lorsque $\dim \mathrm{E} < \infty$.

Spectre et polynôme annulateur

Définition 5.3 : polynôme annulateur

Soit $f \in \mathscr{L}(\mathrm{E})$. On appelle *polynôme annulateur* de f tout polynôme P tel que $\mathrm{P}(f) = 0$ (endomorphisme nul), avec $\mathrm{P} \neq 0$ (polynôme nul).

Théorème 5.1

Soit $f \in \mathscr{L}(\mathrm{E})$, où E est un $\mathbb{K}$-ev de dimension finie. Alors f possède un polynôme annulateur.

⚠ Il n'y a jamais unicité du polynôme annulateur.

Les polynômes annulateurs permettent de déterminer l'inverse d'une matrice. Si en effet $\mathrm{M} \in \mathscr{M}_n(\mathbb{K})$ et qu'il existe $\mathrm{P} \in \mathbb{K}[x]$ non nul tel que $\mathrm{P}(\mathrm{M}) = 0$, alors M est inversible et M^{-1} s'exprime comme un polynôme en M.

☛ **Exemple 5.2**

Soit $\mathrm{M} \in \mathscr{M}_n(\mathbb{K})$ telle que $\mathrm{M}^3 - 3\mathrm{M}^2 + 7\mathrm{M} + 5\,\mathrm{Id}_n = 0$. Alors $\frac{-1}{5}\left(\mathrm{M}^3 - 3\mathrm{M}^2 + 7\mathrm{M}\right) = \mathrm{Id}_n$ et $\frac{-\mathrm{M}}{5}\left(\mathrm{M}^2 - 3\mathrm{M} + 7\,\mathrm{Id}_n\right) = \mathrm{Id}_n$. D'où $\mathrm{M}^{-1} = \frac{-1}{5}\left(\mathrm{M}^2 - 3\mathrm{M} + 7\,\mathrm{Id}_n\right)$.

Théorème 5.2

Soit $f \in \mathscr{L}(\mathrm{E})$ et soit $\mathrm{P} \in \mathbb{K}[x]$, tel que $\mathrm{P}(f) = 0$. Alors $\mathrm{Sp}(f) \subset \{\text{zéros de P}\}$.

Théorème 5.3

Soient $\lambda_1, \ldots, \lambda_k$ des valeurs propres distinctes d'un endomorphisme f et soient $x_1, \ldots, x_k$ les vecteurs propres associés à ces valeurs propres. Alors $\left(x_1\,;\ldots\,;x_k\right)$ est une famille libre.

Théorème 5.4

Soit $f \in \mathscr{L}(\mathrm{E})$ et soient $\lambda_1, \ldots, \lambda_k$ des valeurs propres distinctes de f. Alors les espaces propres E_{λ_i} sont en somme directe, c'est-à-dire que

- $\forall x \in \mathrm{E}, \exists!\ \left((x_1, \ldots, x_k)\right) \in \mathrm{E}_{\lambda_1} \times \ldots \times \mathrm{E}_{\lambda_k}$ tel que $x = x_1 + \ldots + x_k$;
- ou si $x = x_1 + \ldots + x_k = 0$, avec $x_i \in \mathrm{E}_{\lambda_i}$, alors tous les x_i sont nuls ;
- ou si $\mathscr{B}_i$ est une base de E_{λ_i}, alors $\cup_{1 \leq i \leq k} \mathscr{B}_i$ est une base de E.

Corollaire 5.1

Avec les mêmes hypothèses que dans le théorème 5.4 précédent, on a alors $\dim \mathrm{E} = \dim\left(\mathrm{E}_{\lambda_1} + \ldots + \mathrm{E}_{\lambda_k}\right) = \dim \mathrm{E}_{\lambda_1} + \ldots + \dim \mathrm{E}_{\lambda_k} = \sum_{i=1}^{k} \dim \mathrm{E}_{\lambda_i}$.

1.2 Matrices et endomorphismes diagonalisables

Définitions

Définition 5.4 : endomorphisme diagonalisable

Soit $f \in \mathscr{L}(\mathrm{E})$. On dit que f est *diagonalisable* quand il existe une base $\mathscr{B}$ de E formée des vecteurs propres de f. Dans ce cas, $\mathscr{M}_{f;\mathscr{B}}$ est diagonale.

Définition 5.5 : matrice diagonalisable

Soit $\mathrm{A} \in \mathscr{M}_n(\mathbb{K})$. On dit que A est *diagonalisable* quand A est semblable à une matrice diagonale, c'est-à-dire s'il existe $\mathrm{P} \in \mathscr{M}_n(\mathbb{K})$ inversible et s'il existe $\mathrm{D} \in \mathscr{M}_n(\mathbb{K})$ diagonale telles que $\mathrm{A} = \mathrm{PDP}^{-1}$.

Propriété 5.2

Soit $f \in \mathscr{L}(\mathrm{E})$ et soit $\mathrm{A} \in \mathscr{M}_n(\mathbb{K})$ sa matrice dans une base quelconque.
On suppose qu'il existe $n(\mathbb{N}^\star) \in \mathbb{R}$ tel que $\mathrm{Sp}(\mathrm{A}) = \{\lambda_i\,;i \in [\![1\,;n]\!]\}$.
On note E_{λ_i} le sous-espace propre associé à la valeur propre λ_i.

Alors f est diagonalisable
si et seulement si A est diagonalisable
si et seulement si E est somme directe des E_{λ_i}
si et seulement si $\dim(\mathrm{E}) = \sum_{i=1}^{k} \dim\left(\mathrm{E}_{\lambda_i}\right)$.

Théorème 5.5

Soit E de dimension finie. Soit $f \in \mathscr{L}(\mathrm{E})$ et soit $\mathrm{A} \in \mathscr{M}_n(\mathbb{K})$.

- f est diagonalisable si et seulement s'il existe un polynôme P scindé à racines simples (c'est-à-dire que $\mathrm{P}(x) = \prod_i (x - \alpha_i)$ avec les $\alpha_i \in \mathbb{K}$ distincts deux à deux) tel que $\mathrm{P}(f) = 0$.
- A est diagonalisable si et seulement s'il existe un polynôme P scindé à racines simples tel que $\mathrm{P}(\mathrm{A}) = 0$.

Recherche du spectre

On a déjà vu que $\lambda \in \mathrm{Sp}(f)$ si et seulement si $\det\left(f - \lambda\,\mathrm{id}_\mathrm{E}\right) = 0$ (cf. propriété 5.1-6 p. 38).

Définition 5.6 : polynôme caractéristique

Soit $\mathrm{A} \in \mathscr{M}_n(\mathbb{K})$. On appelle *polynôme caractéristique* de A et on note χ_A le polynôme défini par $\chi_\mathrm{A}(x) = \det\left(x\,\mathrm{Id}_n - \mathrm{A}\right)$.

Propriété 5.3

Soit $\mathrm{A} \in \mathscr{M}_n(\mathbb{K})$. Alors χ_A est un polynôme

1. de degré n,
2. unitaire (le coefficient dominant vaut 1),
3. dont le coefficient en x^{n-1} vaut $-\mathrm{Tr}(\mathrm{A})$,
4. dont le terme de degré 0 vaut $\det \mathrm{A}$.

En utilisant les propriétés du déterminant, on peut aussi définir $\chi_A(x) = \det\left(x\,\mathrm{Id}_n - A\right)$. Ce polynôme diffère du précédent d'un facteur $(-1)^n$. L'avantage de la première formulation est que l'on obtient toujours un polynôme unitaire (le coefficient dominant vaut 1), alors qu'ici ce coefficient dominant vaut $(-1)^n$.

Théorème 5.6

Soit f un endomorphisme de matrice associée A dans une base quelconque. Alors $\lambda \in \mathrm{Sp}(f) = \mathrm{Sp}(A)$ si et seulement si $\chi_A(\lambda) = 0$.

Théorème 5.7 : de Caley-Hamilton

Soit $A \in \mathscr{M}_n(\mathbb{K})$. Alors χ_A est un polynôme annulateur de A.

Définition 5.7 : polynôme minimal

Soit $A \in \mathscr{M}_n(\mathbb{K})$. Le polynôme minimal de A $m \in \mathbb{K}_n[x]$ est le polynôme unitaire, annulateur de A ayant le plus petit degré.

Propriété 5.4 : polynôme minimal

Soit $A \in \mathscr{M}_n(\mathbb{K})$. On note χ_A et m_A les polynômes caractéristique et minimal de A. Alors
- m_A possède les mêmes racines que χ_A,
- m_A divise χ_A,
- m_A est annulateur de A, c'est-à-dire $M_A(A) = 0$.

Cette dernière propriété permet de déterminer le polynôme minimal d'une matrice à partir de son polynôme caractéristique.

☛ **Exemple 5.3**

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

Par définition, $\chi_A(\lambda) = (\lambda - 2)(\lambda(\lambda - 3) + 2)$, ce qui mène à $\chi_A(\lambda) = \lambda^3 - 5\lambda^2 + 8\lambda - 4$. Il est aisé d'obtenir la factorisation de ce polynôme : $\chi_A(\lambda) = (\lambda - 2)^2(\lambda - 1)$.
Donc χ_A est scindé mais pas à racines simples.
Étant données les propriétés du polynôme minimal, les candidats possibles sont $p_1(x) = (x-2)(x-1)$ et $p_2(x) = \chi_A(x)$. Ces deux polynômes ont en effet les mêmes racines que χ_A et divisent χ_A. Bien entendu, d'après le théorème 5.7 (de Caley-Hamilton) p. 40, p_2 est annulateur de A.
On calcule $p_1(A) = \left(A - 2\,\mathrm{Id}_3\right)\left(A - \mathrm{Id}_3\right) = \begin{pmatrix} 0 & 1 & 1 \\ 0 & -2 & -2 \\ 0 & 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 & 1 \\ 0 & -1 & -2 \\ 0 & 1 & 2 \end{pmatrix} = 0_3$.
Donc p_1 est annulateur de A. Comme c'est un polynôme unitaire et que $\deg\left(p_1\right) < \deg\left(p_2\right)$ (c'est le polynôme annulateur de plus petit degré parmi les candidats), on a alors que le polynôme minimal de A $m_A = p_1$.

Théorème 5.8

Soit E de dimension finie n. Soit $f \in \mathscr{L}(E)$ et soit $A \in \mathscr{M}_n(\mathbb{K})$ la matrice associée.
On note χ_A et m_A les polynômes caractéristique et minimal de A.
Alors A est diagonalisable si et seulement si m_A est scindé à racines simples. Dans ce cas, on note λ_i les k valeurs propres distinctes de A, de multiplicités respectives μ_i (donc $\sum_{i=1}^{k} \mu_i = n$).
De plus $A = PDP^{-1}$ avec la matrice diagonale D dont les coefficients sont les λ_i *selon leurs multiplicités.*
La matrice de passage P est donnée par les vecteurs propres *rangés dans le même ordre* que les valeurs propres associées dans D.

⚠
- Si χ_A est scindé à racines simples, alors $\chi_A = m_A$ et A est diagonalisable.
- Si χ_A est scindé à racines multiples et que m_A est à racines simples, A est diagonalisable.
- Si m_A est à racines multiples, A est trigonalisable (A est semblable à une matrice triangulaire).

2 Projecteurs spectraux associés à une matrice diagonalisable

On déduit du fait que le polynôme minimal d'une matrice diagonalisable est à racines simples l'important résultat suivant, qui est très utile pour les calculs de puissances et d'exponentielles de matrices.

Théorème 5.9

Soit $n \in \mathbb{N}$ et soit $\mathrm{D} \in \mathscr{M}_n(\mathbb{K})$ une matrice *diagonalisable*. Soit $m \in \mathbb{N}$ tel que $\mathrm{Sp}(\mathrm{D}) = \{\lambda_i \in \mathbb{K} ; i \in [\![1 ; m]\!]\}$.
Pour $i \in [\![1 ; m]\!]$ il existe $(u_i, v_i) \in \mathbb{C}[x]^2$ tel que $(x-\lambda_i)u_i + \prod_{\substack{j=1 \\ j\neq i}}^{m}(x-\lambda_j)v_i = 1$. On pose $p_i = \prod_{\substack{j=1 \\ j\neq i}}^{m}(x-\lambda_j)v_i$.
On appelle alors *projecteur spectral* P_i associé à la valeur propre λ_i de D le polynôme défini par $\mathrm{P}_i = p_i(\mathrm{D})$.
On a de plus pour $(i,j) \in [\![1 ; m]\!]^2$, $\mathrm{P}_i^2 = \mathrm{P}_i$ et $\mathrm{P}_i\mathrm{P}_j = 0$ si $i \neq j$ et

$$\begin{cases} \mathrm{Id}_n = \sum_{i=1}^m \mathrm{P}_i \\ \mathrm{D} = \sum_{i=1}^m \lambda_i \mathrm{P}_i \end{cases} \tag{5.1}$$

Démonstration 5.1

Comme les valeurs propres sont distinctes, les polynômes $x-\lambda_i$ sont premiers deux à deux. Alors, d'après le théorème 3.3 p. 17, l'existence et l'unicité des paires de polynômes $(u_i ; v_i)$ sont établies.
Le fait que $\mathrm{P}_i\mathrm{P}_j = 0$ pour $i \neq j$ vient du fait que le polynôme minimal de D est égal à $\prod_{i=1}^m (x-\lambda_i)$.
Le fait que $\mathrm{P}_j^2 = \mathrm{P}_j$ provient du fait que $\mathrm{P}_j(\mathrm{D}-\lambda_j \mathrm{Id}_n) \times u_j(\mathrm{D}) = 0$.
Le fait que $\mathrm{Id}_n = \sum_{j=1}^m \mathrm{P}_j$ provient alors du fait que $\prod_{j=1}^m (\mathrm{Id}_n - \mathrm{P}_j) = \prod_{i=1}^m (\mathrm{D}-\lambda_i \mathrm{Id}_n)$.
Le fait que $\mathrm{D} = \sum_{j=1}^m \lambda_j \mathrm{P}_j$ provient du fait que le théorème reste valable si on remplace le polynôme caractéristique de D par le polynôme minimal de D.
L'application $\mathrm{X} \mapsto \mathrm{P}_j\mathrm{X}$ est une projection de $\mathbb{K}^n$ sur le sous-espace propre E_{λ_j} associé à la valeur propre λ_j, ce qui explique le terme de projecteur spectral.

□

La proposition suivante, basée sur une récurrence immédiate, permet de calculer les projecteurs spectraux ainsi que les puissances successives de D.

Propriété 5.5

Soit $n \in \mathbb{N}$ et soit $\mathrm{D} \in \mathscr{M}_n(\mathbb{K})$ une matrice *diagonalisable*. Soit $m \in \mathbb{N}$ tel que $\mathrm{Sp}(\mathrm{D}) = \{\lambda_i \in \mathbb{K} ; i \in [\![1 ; m]\!]\}$. Soit $(\mathrm{P}_1, \ldots, \mathrm{P}_m)$ les projecteurs spectraux associés aux valeurs propres.
Alors on a, pour tout $p \in \mathbb{N}$, avec la convention $\mathrm{D}^0 = \mathrm{I}_n$,

$$\sum_{j=1}^m \lambda_j^p \mathrm{P}_j = \mathrm{D}^p. \tag{5.2}$$

De plus la famille $(\mathrm{P}_1, \ldots, \mathrm{P}_m)$ est l'unique solution du système linéaire de Cramer à inconnues matricielles

$$\begin{cases} \forall p \in [\![0 ; m-1]\!] \\ \sum_{i=1}^m \lambda_i^p \mathrm{P}_i = \mathrm{D}^p \end{cases} \tag{5.3}$$

Exemple 5.4

Soit $\mathrm{A} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$. Comme A est triangulaire, on a aisément que son polynôme caractéristique s'écrit $\chi_{\mathrm{A}}(x) = (x-1)^2(x-2)^2$. Les possibilités pour le polynôme minimal sont $m_{\mathrm{A}}(x) = (x-1)(x-2)$, $m_{\mathrm{A}}(x) = (x-1)^2(x-2)$, $m_{\mathrm{A}}(x) = (x-1)(x-2)^2$ ou $m_{\mathrm{A}} = \chi_{\mathrm{A}}$.

Un rapide calcul matriciel montre que $(A - \mathrm{Id})(A - 2\,\mathrm{Id}) = 0$. Donc $m_A(x) = (x-1)(x-2)$. Comme ce polynôme est scindé à racines simples réelles, A est diagonalisable sur $\mathbb{R}$.
Notons P_1 et P_2 les projecteurs spectraux associés aux valeurs propres 1 et 2. D'après le théorème 5.9, ces matrices sont solutions du système

$$\begin{cases} P_1 + P_2 = \mathrm{Id} \\ P_1 + 2P_2 = A \end{cases}$$

Alors $P_2 = A - \mathrm{Id}$ et $P_1 = \mathrm{Id} - (A - \mathrm{Id}) = 2\,\mathrm{Id} - A$, ce qui donne $P_1 = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ et $P_2 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$.

Par conséquent, pour tout $n \in \mathbb{N}$, $A^n = P_1 + 2^n P_2 = \begin{pmatrix} 1 & 2^n - 1 & 0 & 0 \\ 0 & 2^n & 0 & 0 \\ 0 & 0 & 2^n & 2^n - 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

Le fichier *Projecteurs_spectraux.py* inclus au présent document permet d'effectuer les calculs précédents à l'aide de *Python* et de la librairie sympy. Voir le code A.4 (projecteurs spectraux et *Python*) p. 74.

3 Décomposition de Jordan-Chevalley

3.1 Méthode de la tangente

Il s'agit d'une méthode numérique de résolution d'équations du type $f(x) = 0$ dans laquelle on approche la courbe de f par ses tangentes.

Partant d'un point x_0 (de préférence proche du zéro à trouver), on approche la fonction en la considérant à peu près égale à sa tangente en ce point. On utilise pour cela un développement de Taylor au premier ordre de la fonction :

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0). \tag{5.4}$$

On calcule ensuite l'intersection entre cette tangente et l'axe des abscisses, en résolvant l'équation

$$f(x_0) + f'(x_0)(x - x_0) = 0. \tag{5.5}$$

Cela fournit un point x_1 qui est susceptible d'être plus proche de la racine α de f que le point initial x_0.

Par un procédé itératif, on améliore l'approximation, en utilisant la tangente en x_1 pour déterminer un point x_2, lequel permettra à son tour de calculer un point x_3, etc.

Sous de bonnes hypothèses sur la fonction f, l'algorithme est assuré de converger. De plus, pour un choix convenable de x_0, la convergence est très rapide.

Théorème 5.10 : méthode de Newton

Soit une fonction $f : [a\,;b] \to \mathbb{R}$ de classe $\mathcal{C}^2$, avec $f(a) > 0$ et $f(b) < 0$ (ou inversement), et telle que f' et f'' sont de signes constants sur $[a\,;b]$.
Alors l'équation $f(x) = 0$ possède une unique racine $\tilde{x}$ sur $[a\,;b]$.
De plus, avec $x_0 = \begin{cases} b & \text{si } f'f'' > 0 \\ a & \text{sinon} \end{cases}$ et en posant $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$, la suite (x_n) converge vers $\tilde{x}$, avec l'estimation d'erreur $|x_{n+1} - \tilde{x}| \leq k\,|x_n - \alpha|^2$, où k ne dépend que de f, f' et f''.

Démonstration 5.2

D'après le théorème de Taylor, toute fonction possédant une dérivée seconde peut être approchée par un développement aux alentours d'un points proche d'une racine (notée α) de cette fonction. De la sorte, comme $f \in \mathcal{C}^2$, on a $0 = f(\alpha) = f(x_n) + f'(x_n)(\alpha - x_n) + \frac{1}{2!} f''(\xi_n)(\alpha - x_n)^2$, avec $\xi_n \in \left]x_n\,;\alpha\right[$ ou $\left]\alpha\,;x_n\right[$.

En divisant l'équation précédente par $f'(x_n)$ et en utilisant la définition de x_{n+1}, on se ramène à $\alpha - x_{n+1} = \frac{-f''(\xi_n)}{2f'(x_n)}(\alpha - x_n)^2$.

D'où il vient $|\epsilon_{n+1}| = \frac{|f''(\xi_n)|}{2|f'(x_n)|}\epsilon_n^2$, où ϵ_n est l'erreur entre l'itéré n et la solution exacte α.

Cette dernière équation montre que le taux de convergence est quadratique, si les condition suivantes sont vérifiées :

1. *f' ne s'annule pas dans un voisinage I de α ;*
2. *f'' est continue sur I ;*
3. *$\exists c < \infty, \left|\frac{f''(x_n)}{f'(x_n)}\right| < c\left|\frac{f''(\alpha)}{f'(\alpha)}\right|$;*
4. *$\forall n \in \mathbb{N}, c\left|\frac{f''(\alpha)}{f'(\alpha)}\right|\epsilon_n < 1$.*

Pour retrouver l'estimation d'erreur donnée dans le théorème, il suffit de poser $k = \sup_{x\in I}\left|\frac{f''(x)}{2f'(x)}\right|$.

L'hypothèse $f(a) > 0, f(b) < 0$ permet d'obtenir l'unicité de la racine, en utilisant le théorème des valeurs intermédiaires.

□

Cette méthode peut s'écrire sous la forme de l'algorithme 5.1.

Algorithme 5.1 : méthode de Newton

```
Entrée : f, f', α, ϵ
Sortie : α
---------------------------------------
tant que |f(α)| > ϵ faire
    α = α − f(α)/f'(α)
fin tant que
retourner α
```

La figure 5.1 illustre la construction de la solution α selon les signes des dérivées (sur un intervalle contenant la solution).

En pratique on détermine un intervalle $I = [a\,;b]$ sur lequel

1. f' et f'' sont de signes constants et f' ne s'annule pas ;
2. $f(a)f(b) < 0$;
3. $b - a < \frac{2m}{M}$, où $m = \inf_{x\in I}|f'(x)|$ et $M = \sup_{x\in I}|f''(x)|$.

Notons que si I vérifie les deux premières conditions, on peut toujours trouver, par exemple par dichotomie, un intervalle $\tilde{I} \subset I$ tel que les trois conditions sont vérifiées sur $\tilde{I}$.

Les deux premières conditions garantissent qu'il existe un unique $\alpha \in]a\,;b[$ tel que $f(\alpha) = 0$. Si $f'(x)f''(x) < 0$ sur I, on pose $x_0 = a$ et on définit x_n par l'algorithme de Newton pour $n \in \mathbb{N}^\star$. On vérifie par récurrence que $a \le x_n \le \alpha$ pour $n \in \mathbb{N}$, que la suite $(x_n)_{n\in\mathbb{N}}$ est croissante, et on a

$$0 \le \alpha - x_n \le \left(\frac{M}{2m}\right)^{2^n-1}(b-a)^{2^n}. \tag{5.6}$$

De même, si $f'(x)f''(x) > 0$ sur I, on pose $x_0 = b$ et on définit x_n par l'algorithme de Newton pour $n \in \mathbb{N}^\star$. On vérifie par récurrence que $\alpha \le x_n \le b$ pour $n \in \mathbb{N}$, que la suite $(x_n)_{n\in\mathbb{N}}$ est décroissante, et on a

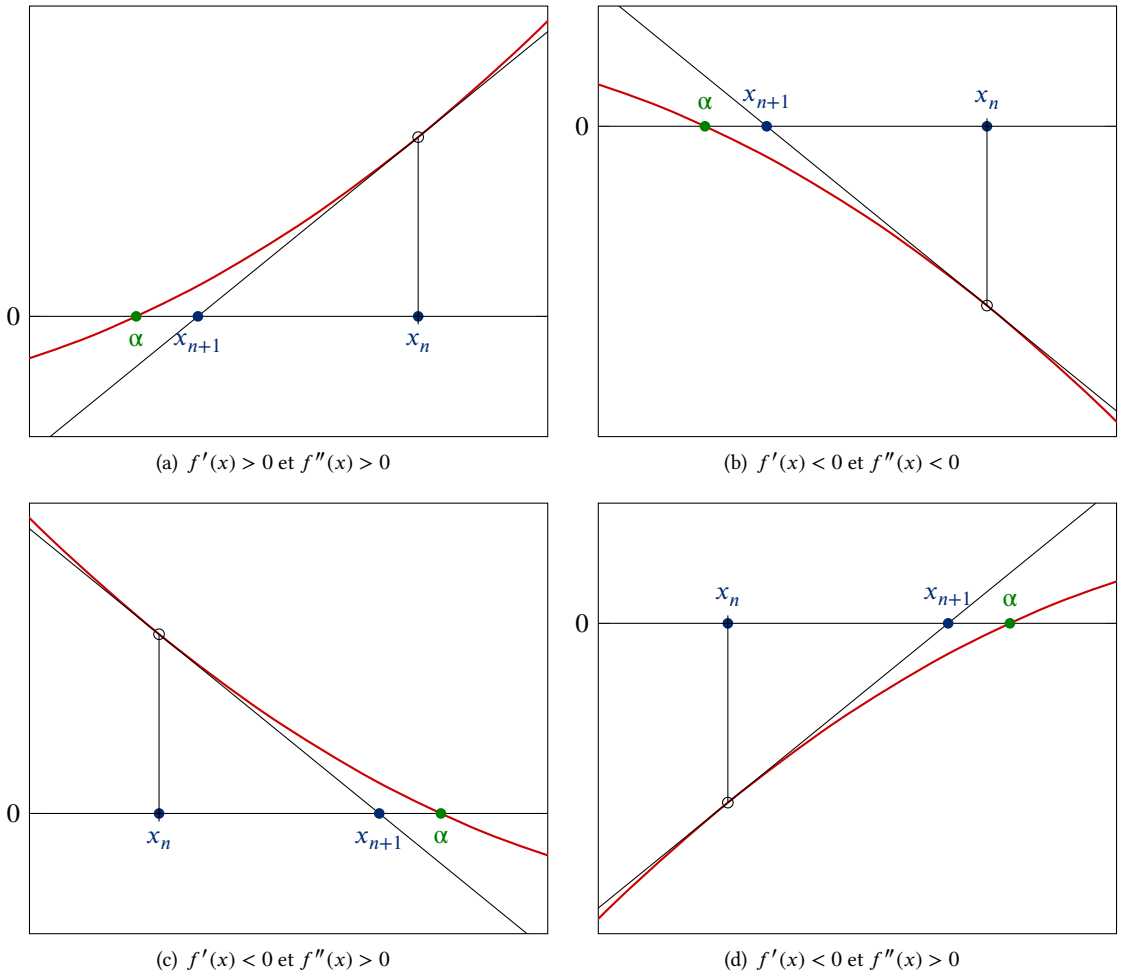
$$0 \le x_n - \alpha \le \left(\frac{M}{2m}\right)^{2^n-1}(b-a)^{2^n}. \tag{5.7}$$

Exemple 5.5

Voici un exemple classique, déjà connu de Héron d'Alexandrie [1]. On pose $f(x) = x^2 - 2$, $I = [a\,;b]$ avec $a = 1{,}4$ et $b = 1{,}5$. Il est clair ici que la solution cherchée est $\alpha = \sqrt{2}$.

On a $f'(x) = 2x$, positif et non nul sur I, et $f''(x) = 2 > 0$ sur I.

La suite $(x_n)_{n\in\mathbb{N}}$ se construit par $\begin{cases} x_0 = b \\ x_{n+1} = x_n - \frac{x_n^2-2}{2x_n} = \frac{x_n}{2} + \frac{1}{x_n} \end{cases}$



(a) $f'(x) > 0$ et $f''(x) > 0$

(b) $f'(x) < 0$ et $f''(x) < 0$

(c) $f'(x) < 0$ et $f''(x) > 0$

(d) $f'(x) < 0$ et $f''(x) > 0$

Figure 5.1 – Illustration de la méthode de Newton

Ici $m = 2 \times a = 2{,}8$ et M $= 2$. Par conséquent on a $\forall n \in \mathbb{N}$, $0 \leq x_n - \alpha \leq \frac{1}{1{,}2^{2^n-1}} \times 0{,}1^{2^n} < 10^{-2^n}$. On a donc, pour $n = 1$, une erreur inférieure à 10^{-2}, pour $n = 2$, une erreur inférieure à 10^{-4} et pour $n = 3$, une erreur inférieure à 10^{-8}.
Réalisons ces calculs avec *Python*.
4 1.41421356237310
On obtient la suite

i	x_i	$f(x_i)$
0	1,5	0,250000000000000
1	1,41666666666667	0,00694444444444464
2	1,41421568627451	0,00000600730488287127
3	1,41421356237469	$4{,}51061410444709 \times 10^{-12}$

Ainsi, en ayant donné la précision à 1×10^{-8}, on obtient bien une valeur approchée après seulement 4 itérations. En comparant en effet la solution obtenue par ce moyen à un calcul direct, on obtient une erreur de 1.59472435257157e-12 , conforme à la prévision. Il suffit aussi de regarder la dernière cellule du tableau précédent.
La figure 5.2 donne une représentation de la fonction étudiée ainsi que des points successifs obtenus par la méthode de Newton.

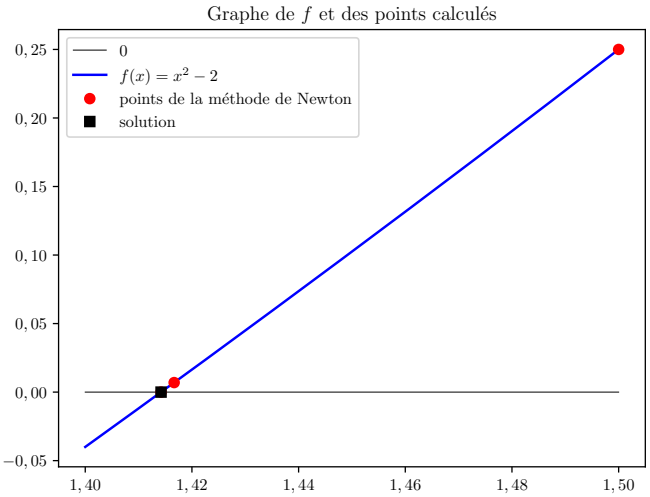


Figure 5.2 – Approximation de $\sqrt{2}$ par la méthode de la tangente

Ces résultats ont été obtenus par le script *Newton.py* inclus au présent document. Voir le code A.5 (méthode de Newton et *Python*) p. 75.

3.2 Décomposition effective de Jordan-Chevalley

Définition 5.8 : matrice nilpotente

Soit $n \in \mathbb{N}^\star$ et soit $A \in \mathscr{M}_n(\mathbb{C})$.
On dit que A est *nilpotente* s'il existe un entier $m \in \mathbb{N}^\star$ tel que pour tout $k \in \mathbb{N}$, $k \geq m$, $A^k = 0$.
L'entier m est appelé l'*indice de nilpotence* de A : c'est donc le plus petit entier tel que la matrice élevée à cette puissance est la matrice nulle.

Le théorème suivant donne la définition de la décomposition de Jordan-Chevalley ainsi qu'un moyen de l'obtenir.

Théorème 5.11 : décomposition de Jordan-Chevalley

Soit $A \in \mathscr{M}_n(\mathbb{C})$.
Alors il existe une unique paire $(D, N) \in \mathscr{M}_n(\mathbb{C})^2$ possédant les propriétés suivantes :

1. $A = D + N$;
2. D est diagonalisable, N est nilpotente, et $DN = ND$.

Soit χ_A le polynôme caractéristique de A et soit $p_A = \frac{\chi_A}{\text{pgdc}(\chi_A ; \chi'_A)}$. Avec $D_0 = A$ et $D_{m+1} = D_m - p_A(D_m)\left(p'_A(D_m)\right)^{-1}$, la suite $\left(D_m\right)_{m\in\mathbb{N}}$ est bien définie et $D_m = D$ pour tout entier $m \in \mathbb{N}^\star$ tel que χ_A divise $p_A^{2^m}$.

L'unicité de la décomposition provient du fait que toute matrice diagonalisable et nilpotente est nulle, et on vérifie que A et D ont le même polynôme caractéristique. On peut faire les remarques suivantes.

1. Soit $k \in \mathbb{N}^\star$ et soit $(\lambda_1, \ldots, \lambda_k)$ la famille des valeurs propres de A de multiplicités respectives $(\mu_1, \ldots, \mu_k)$. On a donc $\chi_A(x) = \prod_{i=1}^k (x - \lambda_i^{\mu_i})$. Par construction, $p_A = \prod_{i=1}^k (x - \lambda_i)$ est à racines simples. Comme le pgdc de χ_A et de χ'_A est donné par l'algorithme d'Euclide, on voit qu'on peut calculer explicitement p_A sans avoir calculé les valeurs propres de A. D'autre part on voit que χ_A divise $p_A^{2^m}$ si et seulement si on a la condition $2^m \geq \max_{i\in[\![1\,;k]\!]} \mu_i$.
2. On notera que l'algorithme ci-dessus n'est autre que la méthode de la tangente, appliquée au polynôme p_A, et initialisée en A. On montre qu'il existe une matrice $U_m \in \mathscr{M}_n(\mathbb{C})$ telle que $D_m - D = U_m(D_1 - A)^{2^m} = U_m \chi'_A(A)^{-2^m} p_A(A)^{2^m}$ et le théorème de Cayley-Hamilton montre que $D_m = D$ si χ_A divise $p(A)^{2^m}$. Comme cet algorithme donne une solution D de l'équation matricielle $p_A(U) = 0$, D est diagonalisable puisque p_A est à racines simples. Le fait que $D - A$ est une matrice nilpotente provient du fait qu'il existe pour tout $m \in \mathbb{N}^\star$ une matrice V_m commutant avec D_m et A telle que $D_m - A = V_m p_A(A)$.

3. Comme p_A est à racines simples, aucune des valeurs propres de A n'est racine de p'_A et pgdc $(\chi_A ; p'_A) = 1$. Il résulte alors de l'identité de Bézout qu'il existe $(u_A, v_A) \in \mathbb{C}[x]^2$ tel que $u_A\chi_A + v_A p'_A = 1$. On vérifie que $\chi_{D_m} = \chi_A$ pour tout $m \in \mathbb{N}$, donc $v_A(D_m)p'_A(D_m) = \mathrm{Id}$, et l'algorithme permettant de calculer D s'écrit sous la forme $D_{m+1} = D_m - p_A(D_m)v_A(D_m)$, où $v_A \in \mathbb{C}[x]$ vérifie $p'_A v_A - 1 \equiv 0 \left[\chi_A\right]$. En particulier la suite $(D_m)_{m\in\mathbb{N}}$ est bien définie.

Exemple 5.6

Soit $A \in \mathscr{M}_2(\mathbb{C})$ admettant une racine double λ. Alors la partie diagonalisable D de A est donnée par la formule $D = \lambda \mathrm{Id}_2$ et $N = A - \lambda \mathrm{Id}_2$.
En effet, dans ce cas on a $\chi_A(x) = (x-\lambda)^2$, $p_A = x - \lambda$ et $p'_A = 1$. Comme $\chi_A = p_A^2$, on obtient $D = D_1 = A - p_A(A) = A - (A - \lambda \mathrm{Id}_2) = \lambda \mathrm{Id}_2$ et $N = A - D = A - \lambda \mathrm{Id}_2$.
Le même argument montre aussi que si $A \in \mathscr{M}_n(\mathbb{C})$ admet une unique valeur propre λ de multiplicité n, alors $D = \lambda \mathrm{Id}_n$ et $N = A - \lambda \mathrm{Id}_n$.

Si on connait les valeurs propres de A, on dispose d'une autre méthode pour calculer la partie diagonalisable D de A. On a en effet le résultat suivant.

Propriété 5.6

Soit $n \in \mathbb{N}^\star$ et soit $A \in \mathscr{M}_n(\mathbb{C})$. Soit $k \in \mathbb{N}^\star$ et soit $(\lambda_1, \ldots, \lambda_k)$ la famille des valeurs propres de A de multiplicités respectives $(\mu_1, \ldots, \mu_k)$. On a donc $\chi_A(x) = \prod_{i=1}^{k}(x - \lambda_i^{\mu_i})$.
Alors la partie diagonalisable D de A est donnée par la formule $D = p(A)$, où $p \in \mathbb{C}[x]$ est une solution quelconque du système de congruence

$$\begin{cases} \forall i \in [\![1\,;k]\!] \\ p \equiv \lambda_i \left[(x-\lambda_i)^{\mu_i}\right] \end{cases}$$

Exemple 5.7

Soit $A \in \mathscr{M}_3(\mathbb{C})$ possédant une valeur propre simple λ_1 et une valeur propre double λ_2. Alors la partie diagonalisable D de A est donnée par la formule $D = \lambda_2 \mathrm{Id}_3 + \frac{(A-\lambda_2 \mathrm{Id}_3)^2}{\lambda_1-\lambda_2}$.
En effet, dans ce cas on a $\chi_A = (x-\lambda_1)(x-\lambda_2)^2$ et on a vu à l'exemple 3.4 p. 22 que le polynôme $p(x) = \lambda_2 + \frac{(x-\lambda_2)^2}{\lambda_1-\lambda_2}$ est solution du système de congruence.

Exemple 5.8

La partie diagonalisable de la matrice $A = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ est égale à $D = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.
Ceci s'obtient par l'une ou l'autre méthode, codée dans le script *Jordan_Chevalley.py* inclus au présent document (voir le code A.6 (décomposition de Jordan-Chevalley et *Python*) p. 79).

On pourra consulter Ben Rhouma 2013 en complément.

En ce qui concerne les matrices réelles, il résulte de l'unicité de la décomposition de Jordan-Chevalley que l'on a le résultat suivant.

Propriété 5.7

Soit $A \in \mathscr{M}_n(\mathbb{R})$.
Alors la partie diagonalisable D et la partie nilpotente N de A sont des matrices réelles, et D est diagonalisable sur $\mathbb{R}$ si et seulement si toutes les valeurs propres de A sont réelles.

D'autre part on verra plus loin que toute matrice symétrique réelle nilpotente est nulle, ce qui implique immédiatement que toute matrice réelle symétrique est diagonalisable.

4 Applications

4.1 Calcul des puissances d'une matrice

Le résultat suivant donne un moyen effectif de calculer les puissances d'une matrice quand on connaît ses valeurs propres.

Propriété 5.8

Soit $n \in \mathbb{N}^\star$ et soit $A \in \mathscr{M}_n(\mathbb{C})$. Soit $A = D + N$, avec D diagonalisable, N nilpotente, et $ND = DN$ la décomposition de Jordan-Chevalley de A. Soit $m \in \mathbb{N}^\star$ l'indice de nilpotence de N.
On a alors, pour tout $p \in \mathbb{N}$,

$$A^p = \sum_{k=0}^{p} \binom{p}{k} D^{p-k} N^k = \sum_{k=0}^{\min(m-1,p)} \binom{p}{k} D^{p-k} N^k, \tag{5.8}$$

où $\binom{p}{k} = \frac{p(p-1)\dots(p-k+1)}{k!} = \frac{p!}{k!(p-k)!}$ pour tout $k \in [\![0\,;p]\!]$.

! Par convention, $0! = 1! = 1$. De plus, pour tout $n \in \mathbb{N}^\star$, pour toute matrice $M \in \mathscr{M}_n(\mathbb{K})$, $M^0 = 0$.

Démonstration 5.3

Comme $ND = DN$, *ceci résulte immédiatement de la formule du binôme de* Newton. *Le fait que l'on peut arrêter la sommation à* $m-1$ *si* $m-1 \leq p$ *provient du fait que* $N^k = 0$ *pour* $k \geq m$.

□

Le calcul de D^p se fait en diagonalisant D. Par construction, D est diagonalisable, donc il existe deux matrices Δ diagonale et P inversible telles que $D = P\Delta P^{-1}$. Une récurrence immédiate montre que l'on a, pour tout $p \in \mathbb{N}$, $D^p = P\Delta^p P^{-1}$. De plus comme $\Delta = (d_{i,j})$ est diagonale, Δ^p aussi et les coefficients de cette dernière matrices sont directement $d_{i,j}^2$, ce qui permet de calculer effectivement D^p.

☛ **Exemple 5.9**

On a, pour tout $p \in \mathbb{N}$,

$$\begin{pmatrix} 1 & -1 \\ 1 & 3 \end{pmatrix}^p = \begin{pmatrix} 2^p - p2^{p-1} & -p2^{p-1} \\ p2^{p-1} & 2^p + p2^{p-1} \end{pmatrix}.$$

Soit en effet $A = \begin{pmatrix} 1 & -1 \\ 1 & 3 \end{pmatrix}^p$. Comme il s'agit d'une matrice de taille 2, on a $\chi_A(x) = x^2 - \operatorname{Tr}(A)\,x + \det(A) = x^2 - 4x + 4 = (x-2)^2$: A possède une valeur propre double égale à 2. En calculant $A - 2\,\mathrm{Id}_2$ qui est non nul, on vérifie que le polynôme minimal est égal au polynôme caractéristique, donc A n'est pas diagonalisable. On est dans la situation de l'exemple 5.6 p. 46 et la décomposition de Jordan $A = D + N$ de A donne $D = 2\,\mathrm{Id}$, $N = A - 2\,\mathrm{Id} = \begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix}$. On vérifie aisément que $N^2 = 0$ et on obtient, pour tout $p \in \mathbb{N}$,

$$A^p = 2^p\,\mathrm{Id} + p2^{p-1}N = \begin{pmatrix} 2^p - p2^{p-1} & -p2^{p-1} \\ p2^{p-1} & 2^p + p2^{p-1} \end{pmatrix}.$$

On peut néanmoins éviter de calculer la diagonalisation. En effet, si D est une matrice diagonalisable, ayant k valeurs propres distinctes $(\lambda_1, \dots, \lambda_k)$ et si $(P_1, \dots, P_k)$ désigne la famille des projecteurs spectraux de D associés aux valeurs propres, on a vu que l'on a, pour tout $p \in \mathbb{N}$, $D^p = \sum_{i=1}^k \lambda_i^p P_i$ (voir l'équation (5.2) dans la propriété 5.5 p. 41).

☛ **Exemple 5.10**

On a vu (exemple 5.8 p. 46) que la décomposition de Jordan-Chevalley de la matrice $A = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ est donnée par $D = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ et $N = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$.

Alors pour tout $p \in \mathbb{N}$, $A^p = \begin{pmatrix} 1 & 0 & 2^p - 1 & p \\ 0 & 2^p & 2^{p-1}p & 0 \\ 0 & 0 & 2^p & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.

On a vu également que A (et donc D) a deux valeurs propres doubles qui sont 1 et 2. D'après le théorème 5.9 p. 41, les projecteurs spectraux de D sont donnés par le système

$$\begin{cases} P_1 + P_2 = \mathrm{Id} \\ P_1 + 2P_2 = D \end{cases}$$

dont la résolution donne $P_2 = D - \mathrm{Id}$ et $P_1 = 2\,\mathrm{Id} - D$.
Un rapide calcul montre que N est nilpotente d'ordre 2. On a donc

$$\begin{aligned} A^p = D^p + pD^{p-1}N &= \left(P_1 + 2^p P_2\right) + p\left(P_1 + 2^{p-1}P_2\right)N \\ &= P_1\left(\mathrm{Id} + pN\right) + P_2\left(2^p\,\mathrm{Id} + 2^{p-1}N\right) \\ &= (2\,\mathrm{Id} - D)\left(\mathrm{Id} + pN\right) + 2^{p-1}\left(D - \mathrm{Id}\right)(2\,\mathrm{Id} + N) \end{aligned}$$

Le dernier calcul s'effectue aisément avec *Python* :

```
... import sympy as sp
  File "<stdin>", line 3
    import sympy as sp
           ^
SyntaxError: invalid syntax
>>> sp.var("p", integer=True, positive=True)
p
>>> D = sp.Matrix([[1, 0, 1, 0], [0, 2, 0, 0], [0, 0, 2, 0], [0, 0, 0, 1]])
>>> N = sp.Matrix([[0, 0, 0, 1], [0, 0, 1, 0], [0, 0, 0, 0], [0, 0, 0, 0]])
>>> I4 = sp.eye(D.shape[0])
>>> sp.pprint(sp.simplify((2*I4-D)*(I4+p*N)+2**(p-1)*(D-I4)*(2*I4+N)))
[        p        ]
[1  0   2  - 1  p ]
[                 ]
[    p   p - 1    ]
[0  2   2        0]
[                 ]
[           p     ]
[0  0      2     0]
[                 ]
[0  0      0     1]
```

On s'intéresse maintenant aux processus itératifs. Soit $n \in \mathbb{N}^\star$. On identifie $\mathbb{C}^n$ à l'espace $\mathscr{M}_{n,1}(\mathbb{C})$ des matrices colonnes à n coefficients complexes On définit une suite $\left(X_m\right)_{m\in\mathbb{N}}$ d'éléments de $\mathbb{C}^n$ par la formule $X_{m+1} = AX_m$ avec $A \in \mathscr{M}_n(\mathbb{C})$. Une récurrence immédiate montre que $X_m = A^m X_0$ pour tout $m \in \mathbb{N}$, ce qui permet de calculer X_m en fonction de X_0 si on a calculé A^m.

Exemple 5.11

Soient $\left(x_m\right)_{m\in\mathbb{N}}$ et $\left(x_m\right)_{m\in\mathbb{N}}$ deux suites définies par les relations de récurrence

$$\begin{cases} x_{m+1} = x_m - y_m \\ y_{m+1} = x_m + 3y_m \\ x_0 = y_0 = 1 \end{cases}$$

Pour obtenir les expressions de x_m et de y_m uniquement en fonction de m, procédons comme suit.
Posons $X_m = \begin{pmatrix} x_m \\ y_m \end{pmatrix}$ et $A = \begin{pmatrix} 1 & -1 \\ 1 & 3 \end{pmatrix}$. On a donc $X_{m+1} = AX_m$.
Par récurrence immédiate, on a alors $X_m = A^m X_0$. Comme on a calculé A^m dans l'exemple 5.9 p. 47, on a donc

$$X_m = \begin{pmatrix} 2^m - m2^{m-1} & -m2^{m-1} \\ m2^{m-1} & 2^m + m2^{m-1} \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} (1-m)2^m \\ (1+m)2^m \end{pmatrix}.$$

Donc pour tout $m \in \mathbb{N}$, $x_m = (1-m)2^m$ et $y_m = (1+m)2^m$.

4.2 Exponentielles de matrices et systèmes différentiels

Soit $n \in \mathbb{N}^\star$. Un système différentiel linéaire du premier ordre à coefficients constants est un système de la forme

$$\begin{cases} y_1'(t) = a_{1,1}y_1(t) + \ldots + a_{1,n}y_n(t) + g_1(t) \\ \ldots \\ y_n'(t) = a_{n,1}y_1(t) + \ldots + a_{n,n}y_n(t) + g_n(t) \end{cases} \tag{5.9}$$

où pour tout $(i,j) \in [\![1\,;n]\!]^2$ les coefficients $a_{i,j}$ sont des nombres réels ou complexes et les fonctions $t \mapsto g_i(t)$ sont définies et continues sur un intervalle ouvert I de $\mathbb{R}$.

Interprétons maintenant un tel système en faisant intervenir des fonctions à valeurs matricielles. Les notions de convergence de suites et de séries de matrices sont celles de la convergence coefficients par coefficients. Soit $(p,q) \in \mathbb{N}^2$. On dit qu'une suite $\left(\mathrm{U}_n\right)_{n\in\mathbb{N}} = \left(\left(u_{i,j,n}\right)_{\substack{1\leq i\leq p\\1\leq j\leq q}}\right)_{n\in\mathbb{N}}$ d'éléments de $\mathscr{M}_{p,q}(\mathbb{C})$ converge vers $\mathrm{U} = \left(u_{i,j}\right)_{\substack{1\leq i\leq p\\1\leq j\leq q}} \in \mathscr{M}_{p,q}(\mathbb{C})$ quand $\lim\limits_{n\to\infty} u_{i,j,n} = u_{i,j}$ pour tout $i \in [\![1\,;p]\!]$ et tout $j \in [\![1\,;q]\!]$. On dit qu'une série $\left(\sum \mathrm{U}_n\right)_{n\in\mathbb{N}} = \left(\sum\left(u_{i,j,n}\right)_{\substack{1\leq i\leq p\\1\leq j\leq q}}\right)_{n\in\mathbb{N}}$ d'éléments de $\mathscr{M}_{p,q}(\mathbb{C})$ est convergente quand pour tout $i \in [\![1\,;p]\!]$ et tout $j \in [\![1\,;q]\!]$, la série $\left(\sum u_{i,j,n}\right)_{n\in\mathbb{N}}$ est convergente. Dans ce cas on pose $\left(\sum \mathrm{U}_n\right)_{n\in\mathbb{N}} = \left(\sum_{n=0}^{\infty} u_{i,j,n}\right)_{\substack{1\leq i\leq p\\1\leq j\leq q}}$.

De même on dit qu'une fonction $f : t \mapsto suite(f_{i,j}(t))_{\substack{1\leq i\leq p\\1\leq j\leq q}}$ définie sur un intervalle ouvert I de $\mathbb{R}$ et à valeurs dans $\mathscr{M}_{p,q}(\mathbb{C})$ est *continue* sur I quand pour tout $i \in [\![1\,;p]\!]$ et tout $j \in [\![1\,;q]\!]$, $f_{i,j}$ est continue sur I. Dans ce cas on pose, pour tout $(a,b) \in \mathrm{I}^2$, $\int_a^b \mathrm{F}(t)\,\mathrm{d}t = \left(\int_a^b f_{i,j}(t)\,\mathrm{d}t\right)_{\substack{1\leq i\leq p\\1\leq j\leq q}}$.

Enfin on dit qu'une fonction $\mathrm{F} : t \mapsto \left(f_{i,j}(t)\right)_{\substack{1\leq i\leq p\\1\leq j\leq q}}$ définie sur un intervalle ouvert I de $\mathbb{R}$ et à valeurs dans $\mathscr{M}_{p,q}(\mathbb{C})$ est *dérivable* sur I quand pour tout $i \in [\![1\,;p]\!]$ et tout $j \in [\![1\,;q]\!]$, $f_{i,j}$ est dérivable sur I. Dans ce cas on pose, pour tout $t \in \mathrm{I}$, $\mathrm{F}'(t) = \left(f_{i,j}'(t)\right)_{\substack{1\leq i\leq p\\1\leq j\leq q}}$.

Revenons à notre système différentiel linéaire du premier ordre à coefficients constants de la forme (5.9) et posons $\mathrm{A} = \left(a_{i,j}\right)_{\substack{1\leq i\leq p\\1\leq j\leq q}}$, $\mathrm{Y}(t) = \begin{pmatrix} y_1(t) \\ \vdots \\ y_k(t) \end{pmatrix}$ et $\mathrm{G}(t) = \begin{pmatrix} g_1(t) \\ \vdots \\ g_k(t) \end{pmatrix}$.

Compte tenu de la définition ci-dessus, on a alors $\mathrm{Y}'(t) = \begin{pmatrix} y_1'(t) \\ \vdots \\ y_k'(t) \end{pmatrix}$ et le système prend la forme matricielle

$$\mathrm{Y}'(t) = \mathrm{A}\mathrm{Y}(t) + \mathrm{G}(t). \tag{5.10}$$

Dans le cas où $k = 1$, on obtient une seule équation différentielle linéaire de la forme

$$y'(t) = ay(t) + g(t),$$

qui se résout par la méthode de la « variation de la constante ». Les solutions de l'équation homogène associée sont données par $y_{\mathrm{H}}(t) = \lambda e^{ta}$, pour tout $\lambda \in \mathbb{C}$. On cherche ensuite une solution particulière sous la forme $\tilde{y}(t) = \lambda(t)e^{ta}$ où λ est maintenant une fonction dérivable. En remplaçant dans l'équation différentielle et après simplification on obtient $\lambda'(t) = e^{-ta}g(t)$, d'où il vient que $\lambda(t) = \lambda(t_0) + \int_{t_0}^t e^{-sa}g(s)\,\mathrm{d}s$. Avec la condition initiale $y(t_0)$, on obtient finalement une solution particulière

$$y(t) = e^{a(t-t_0)}y(t_0) + \int_{t_0}^t e^{a(t-s)}g(s)\,\mathrm{d}s.$$

On a un résultat analogue pour les systèmes différentiels, grâce à la notion d'exponentielle de matrices.

Définition 5.9 : exponentielle de matrice

Soit $n \in \mathbb{N}^\star$ et soit $A \in \mathscr{M}_n(\mathbb{C})$. On appelle *exponentielle de matrice* la matrice définie par

$$e^A = \sum_{m=0}^{+\infty} \frac{A^m}{m!} \tag{5.11}$$

On vérifie que si pour tout $t \in \mathbb{R}$ on pose $F(t) = e^{tA}$, alors F est dérivable sur $\mathbb{R}$ et $F'(t) = Ae^{tA} = e^{tA}A = AF(t) = F(t)A$. On obtient alors le résultat suivant.

Théorème 5.12

Soit I un intervalle de $\mathbb{R}$, soit $k \in \mathbb{N}^\star$, soit $A \in \mathscr{M}_k(\mathbb{R})$ et soit $G : I \to \mathbb{R}^k$ une fonction continue. On considère le système différentiel

$$Y'(t) = AY(t) + G(t) \tag{5.12}$$

Alors pour tout $t_0 \in I$ et pour tout $Y_0 \in \mathbb{R}^k$ il existe sur I une unique solution Y du système différentiel vérifiant $Y(t_0) = Y_0$, qui est donnée par la formule

$$Y(t) = e^{(t-t_0)A}Y_0 + \int_{t_0}^{t} e^{(t-s)A}G(s)\,\mathrm{d}s. \tag{5.13}$$

On voit donc que la résolution des systèmes différentiels linéaires à coefficients constants se ramène à un calcul d'exponentielles de matrices.

La propriété suivante est l'extension aux matrices d'un résultat classique.

Propriété 5.9

Soit $n \in \mathbb{N}^\star$ et soit $(u, V) \in \mathscr{M}_n(\mathbb{K})^2$.
Si U et V commutent (c'est-à-dire UV = VU), alors on a

$$e^{U+V} = e^u e^V$$

! Ceci n'est pas vrai si U et V ne commutent pas...

Propriété 5.10

Soit $n \in \mathbb{N}^\star$ et soit $A \in \mathscr{M}_n(\mathbb{C})$. Soit deux matrices D diagonalisable et N nilpotente telles que ND = DN et A = D + N la décomposition de Jordan-Chevalley de A. On a alors, pour tout $t \in \mathbb{R}$

$$e^{tA} = e^{tD}e^{tN}.$$

Pour une matrice nilpotente N, il n'y a pas de difficulté à calculer l'exponentielle. En effet, avec m l'indice de nilpotence de cette matrice, on a

$$e^{tN} = \sum_{i=0}^{\infty} \frac{t^i}{i!}N^i = \sum_{i=0}^{m-1} \frac{t^i}{i!}N^i.$$

Et on se ramène à une somme finie de matrices.

On est donc ramené dans le cas général au calcul de e^{tD}, avec D diagonalisable. Il existe alors P inversible et Δ diagonalisable telles que $D = P\Delta P^{-1}$ et on vérifie que l'on a

$$e^{tD} = Pe^{t\Delta}P^{-1}.$$

De plus comme $\Delta = (d_{i,j})$ est diagonale, $e^{t\Delta}$ aussi et les coefficients de cette dernière matrices sont directement $e^{td_{i,j}}$, ce qui permet de calculer effectivement e^{tD}.

Exemple 5.12

Soit le système différentiel $\begin{cases} x'(t) = x(t) - y(t) \\ y'(t) = x(t) + 3y(t) \end{cases}$ associé aux conditions initiales $x(0) = y(0) = 1$.

Ce système s'écrit donc sous la forme matricielle $\mathrm{Y}'(t) = \mathrm{A}\mathrm{Y}(t)$, avec $\mathrm{Y}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}$ et $\mathrm{A} = \begin{pmatrix} 1 & -1 \\ 1 & 3 \end{pmatrix}$.

On retrouve ainsi la matrice de l'exemple 5.9 p. 47. On a donc $\mathrm{A} = \mathrm{D} + \mathrm{N}$ avec $\mathrm{D} = 2\,\mathrm{Id}$ et $\mathrm{N} = \mathrm{A} - 2\,\mathrm{Id} = \begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix}$, nilpotente d'ordre 2.

On a donc, pour tout $t \in \mathbb{R}$, $e^{t\mathrm{D}} = e^{2t\,\mathrm{Id}} = e^{2t}\,\mathrm{Id}$ et $e^{t\mathrm{N}} = \mathrm{Id} + t\mathrm{N} = \begin{pmatrix} 1-t & -t \\ t & 1+t \end{pmatrix}$.

Par conséquent, pour tout $t \in \mathbb{R}$, $e^{t\mathrm{A}} = e^{t\mathrm{D}}e^{t\mathrm{N}} = e^{2t}e^{t\mathrm{N}} = e^{2t}\begin{pmatrix} 1-t & -t \\ t & 1+t \end{pmatrix}$.

La solution du système proposé est donnée par la formule

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = e^{t\mathrm{A}}\begin{pmatrix} x(0) \\ y(0) \end{pmatrix} = e^{2t}\begin{pmatrix} 1-t & -t \\ t & 1+t \end{pmatrix}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = e^{2t}\begin{pmatrix} 1-2t \\ 1+2t \end{pmatrix}.$$

On peut bien sûr retrouver ces résultats avec *Python*.

```
>> import sympy as sp

>> sp.var("t") # déclaration de la variable t
t
>> sp.var("x y", cls=sp.Function) # déclaration des fonctions inconnues x et y
(x, y)

>> A = sp.Matrix([[1, -1], [1, 3]]) # définition de la matrice
>> e_tA = sp.simplify(sp.exp(t*A)) # calcul de l'exponentielle
>> sp.pprint(e_tA)
[         2*t        2*t   ]
[(1 - t)*e       -t*e      ]
[                          ]
[      2*t              2*t]
[   t*e        (t + 1)*e   ]
>> Y0 = sp.Matrix([[1], [1]]) # conditions initiales
>> sp.pprint(sp.simplify(e_tA * Y0))
[           2*t]
[(1 - 2*t)*e   ]
[              ]
[           2*t]
[(2*t + 1)*e   ]

>> # définition du système différentiel à résoudre
>> sys_diff = [sp.Eq(x(t).diff(t), x(t) - y(t)),
... sp.Eq(y(t).diff(t), x(t) + 3*y(t))
... ]
>> # conditions initiales sous forme d'un dictionnaire
>> condini = {x(0):1, y(0):1}

>> # résolution et récupération de l'expression de la solution
>> solutions = sp.dsolve(sys_diff, ics=condini)
>> print(f"Les solutions du système\n{sp.pretty(sys_diff)}\nassociée aux
 ↪  conditions\n{sp.pretty(condini)}\nsont\n{sp.pretty(solutions)}")
Les solutions du système
 d                      d
[-(x(t)) = x(t) - y(t), -(y(t)) = x(t) + 3*y(t)]
 dt                     dt
associée aux conditions
{x(0): 1, y(0): 1}
```

```
sont
                2*t     2*t                 2*t     2*t
[x(t) = - 2*t*e     + e    , y(t) = 2*t*e     + e    ]
```

Pour calculer e^{D} quand D est diagonalisable, on peut éviter de diagonaliser D quand on connait ses valeurs propres (de toutes façons on ne peut pas diagonaliser D si on ne peut pas calculer ses valeurs propres). Il suffit en effet de calculer les projecteurs spectraux. On a en effet le résultat suivant.

Propriété 5.11

Soit $n \in \mathbb{N}^\star$ et soit $\mathrm{D} \in \mathscr{M}_n(\mathbb{C})$ une matrice diagonalisable. Soit $k \in \mathbb{N}^\star$, soit $(\lambda_1, \ldots, \lambda_k)$ la famille des valeurs propres de D et soit $(\mathrm{P}_1, \ldots, \mathrm{P}_k)$ la famille des projecteurs spectraux associés.
On a alors, pour tout $t \in \mathbb{R}$,

$$e^{t\mathrm{D}} = \sum_{i=1}^{k} e^{t\lambda_i}\mathrm{P}_i.$$

Exemple 5.13

On a vu (exemple 5.10 p. 47) que la décomposition de Jordan-Chevalley de la matrice $\mathrm{A} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ est donnée par $\mathrm{D} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ et $\mathrm{N} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$, nilpotente d'ordre 2.
Les projecteurs spectraux sont $\mathrm{P}_1 = 2\,\mathrm{Id} - \mathrm{D}$ et $\mathrm{P}_2 = \mathrm{D} - \mathrm{Id} =$, associés aux valeurs propres 1 et 2.
Alors pour tout $t \in \mathbb{R}$, $e^{t\mathrm{D}} = e^t\mathrm{P}_1 + e^{2t}\mathrm{P}_2$, $e^{t\mathrm{N}} = \sum_{i=0}^{1} \frac{t^i}{i!}\mathrm{N}^1 = \mathrm{Id} + t\mathrm{N}$ et $e^{t\mathrm{A}} = e^{t\mathrm{D}}e^{t\mathrm{N}}$. Les calculs sont réalisés via les commandes *Python* ci-dessous

```
>> import sympy as sp
>> sp.var("t") # déclaration de la variable t
t
>> A=sp.Matrix([[1, 0, 1, 1], # définition d'une matrice
... [0, 2, 1, 0],
... [0, 0, 2, 0],
... [0, 0, 0, 1]])
>> D=sp.Matrix([[1, 0, 1, 0], # définition d'une matrice
... [0, 2, 0, 0],
... [0, 0, 2, 0],
... [0, 0, 0, 1]])
>> N = A - D
>> P1 = 2*sp.eye(4)-D # projecteur spectral
>> P2 = D-sp.eye(4) # projecteur spectral

>> print(f"Calcul de e^{{tD}}\n- par calcul direct\n{sp.pretty(sp.exp(t*D))}")
Calcul de e^{tD}
- par calcul direct
[ t         2*t    t     ]
[e    0    e    - e    0 ]
[                         ]
[     2*t                 ]
[0   e         0       0 ]
[                         ]
[              2*t        ]
[0    0       e         0 ]
[                         ]
[                        t]
[0    0        0       e ]
```

```
>>> print(f"- par les projecteurs
 ↪  spectraux\n{sp.pretty(sp.exp(t)*P1+sp.exp(2*t)*P2)}")
- par les projecteurs spectraux
[ t         2*t    t    ]
[e    0    e    - e   0 ]
[                        ]
[     2*t                ]
[0   e         0      0 ]
[                        ]
[             2*t        ]
[0    0      e        0 ]
[                        ]
[                       t]
[0    0        0      e ]

>>> print(f"\nCalcul de e^{{tN}}\n- par calcul direct\n{sp.pretty(sp.exp(t*N))}")

Calcul de e^{tN}
- par calcul direct
[1  0  0  t]
[          ]
[0  1  t  0]
[          ]
[0  0  1  0]
[          ]
[0  0  0  1]
>>> print(f"- par la somme\n{sp.pretty(sp.eye(4)+t*N)}")
- par la somme
[1  0  0  t]
[          ]
[0  1  t  0]
[          ]
[0  0  1  0]
[          ]
[0  0  0  1]

>>> print(f"\nCalcul de e^{{tA}}\n- par calcul direct\n{sp.pretty(sp.exp(t*A))}")

Calcul de e^{tA}
- par calcul direct
[ t         2*t    t     t]
[e    0    e    - e   t*e ]
[                          ]
[     2*t      2*t         ]
[0   e      t*e         0  ]
[                          ]
[             2*t          ]
[0    0      e         0  ]
[                          ]
[                       t  ]
[0    0        0       e  ]
>>> print(f"- par la décomposition de
 ↪  Jordan-Chevalley\n{sp.pretty(sp.exp(t*D)*sp.exp(t*N))}")
- par la décomposition de Jordan-Chevalley
[ t         2*t    t     t]
[e    0    e    - e   t*e ]
[                          ]
[     2*t      2*t         ]
[0   e      t*e         0  ]
```

```
[                          ]
[              2*t         ]
[0     0       e        0  ]
[                          ]
[                        t ]
[0     0         0       e  ]
```

Chapitre 6

Polynômes d'interpolation

Sommaire

1 Généralités

Lors de l'acquisition de données via un capteur, on obtient un tableau de valeurs correspondant, par exemple, à la valeur de la température à un instant donné. L'ensemble des instants forme une famille $(t_0,\ \ldots,\ t_n)$ tandis que les températures constituent la famille $(y_0,\ \ldots,\ y_n)$.

La suite $(t_i)_{0\leq i\leq n}$ forme une suite strictement croissante de valeur de la variable t entre le début de l'acquisition t_0 et la fin du relevé t_n. Interpoler le nuage de points $\big((t_i\,;y_i)\big)_{0\leq i\leq n}$ consiste à trouver une méthode pour prédire la valeur de la température pour tout instant de l'intervalle $[t_0\,;t_n]$, pour des valeurs de t distinctes des points de relevé t_i. On cherche donc à construire une fonction $f\ :\ t\mapsto y=f(t)$, définie sur $[t_0\,;t_n]$.

Nous allons dans ce chapitre présenter un certain nombre de méthodes classiques, utilisées en particulier en mécanique. On pose dans la suite $y_i=f(t_i)$ et $d_i=f'(y_i)$.

1. Dans la section 2 (Interpolation linéaire), la fonction f sera construite par morceaux, en reliant graphiquement par une droite deux points successifs $(t_i\,;y_i)$ et $(t_{i+1}\,;y_{i+1})$.
2. La section 3 (Interpolation de Lagrange), indique comment générer f comme l'unique polynôme p de degré inférieur ou égal à n tel que $p(t_i)=y_i$ quelque soit $i\in[\![0\,;n]\!]$.
3. La section 4 (Interpolation de Hermite) p. 59, montre la construction d'un autre polynôme q, unique également, de degré inférieur ou égal à $2n+1$ tel que $q(t_i)=y_i$ et $q'(t_i)=d_i$ pour tout $i\in[\![0\,;n]\!]$.
4. Enfin, la section 6 (Courbes de Bézier) p. 62 présente une autre forme d'interpolation polynomiale, inventée de manière indépendante chez Renault par Pierre Bézier et chez Citroën par Paul de Faget de Casteljau, qui a révolutionné le dessin des formes des automobiles et est toujours largement utilisée en CAO, notamment « morceaux par morceaux » dans leur version « cubique ».

Deux exemples seront traités dans ce chapitre pour illustrer les différentes méthodes :

1. la suite finie $(t_i)_{0\leq i\leq 4}=(0,\ 1,\ 2,\ 3,\ 4)$ associée à la suite $(y_i)_{0\leq i\leq 4}=(3,\ -2,\ 5,\ 7,\ 1)$;
2. la suite finie $(t_i)_{0\leq i\leq 8}$ définie par $t_i=\pi+i\frac{\pi}{8}$ associée aux suites $(y_i)_{0\leq i\leq 8}$ et $(d_i)_{0\leq i\leq 8}$ telles que $y_i=f(t_i)$ et $d_i=f'(t_i)$ où f est la fonction $f\ :\ t\mapsto\sin t+\sin 2t$.

Le script *Python Interpolation.py* inclus au présent document est ici utilisé pour réaliser les calculs et tracer les graphiques.

2 Interpolation linéaire

2.1 Définition

L'interpolation linéaire est la plus simple des interpolations.

Définition 6.1 : interpolation linéaire

Soit $n \in \mathbb{N}^\star$. Soient les suites $(t_i)_{0\leq i\leq n}$ (strictement croissante) et $(y_i)_{0\leq i\leq n}$.
On appelle *interpolation linéaire* des points formés par ces coordonnées, la fonction construite par les morceaux

$$\begin{cases} \forall i \in [\![0\,;n-1]\!], \forall t \in [x_i\,;x_{i+1}] \\ p_i(t) = y_i + \frac{y_{i+1}-y_i}{t_{i+1}-t_i}(t-x_i) \end{cases} \tag{6.1}$$

2.2 Premier exemple

Pour déterminer la fonction d'interpolation sur le premier nuage de points, on utilise les commandes suivantes (si le module `Interpolations` est importé) :

```
points1 = [(0, 3), (1, -2), (2, 5), (3, 7), (4, 1)] # liste des points
I_lin_1 = interpolation_linéaire(points1) # fonction d'interpolation linéaire
print("L'interpolation linéaire sur le premier exemple donne la fonction")
sp.pprint(I_lin_1)
```

L'exécution permet d'obtenir la fonction f définie par morceaux sur $[0\,;4]$ par

$$f(t) = \begin{cases} 3-5x & \text{si } x \geq 0 \text{ et } x \leq 1 \\ 7x-9 & \text{si } x \geq 1 \text{ et } x \leq 2 \\ 2x+1 & \text{si } x \geq 2 \text{ et } x \leq 3 \\ 25-6x & \text{si } x \geq 3 \text{ et } x \leq 4 \end{cases}.$$

Il est également aisé de tracer les points d'interpolation et la fonction précédente sur l'intervalle *ad hoc*, comme montré à la figure 6.1.

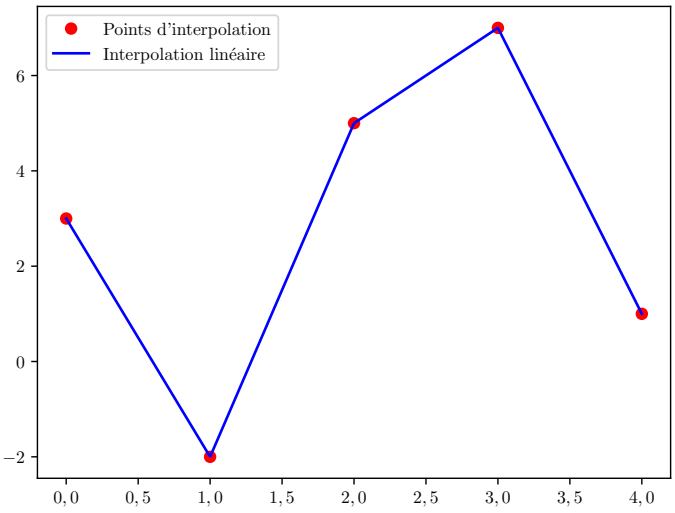


FIGURE 6.1 – Interpolation linéaire pour le premier exemple

2.3 Deuxième exemple

Pour déterminer la fonction d'interpolation sur le second nuage de points, on utilise les commandes suivantes (si le module `Interpolations` est importé) :

```
f = sp.sin(x)+sp.sin(2*x) # la fonction de référence pour l'exemple 2 en symbolique
x2 = [sp.pi*sp.Rational(1+i/8).limit_denominator(1e6) for i in range(9)] # les
  ↪ abscisses de l'exemple 2
y2 = [f.subs(x, _x) for _x in x2] # les y_i de l'exemple 2
points2 = list(zip(x2, y2)) # groupement sous la forme (x_i, y_i)
I_lin_2 = interpolation_linéaire(points2) # fonction d'interpolation linéaire
print("L'interpolation linéaire sur le premier exemple donne la fonction")
sp.pprint(I_lin_2)
```

L'exécution permet d'obtenir la fonction f définie par morceaux sur $[\pi\,;2\pi]$ par

$$f(t) = \begin{cases} \dfrac{8(x-\pi)\left(-\sqrt{\frac{1}{2}-\frac{\sqrt{2}}{4}}+\frac{\sqrt{2}}{2}\right)}{\pi} & \text{si } x \geq \pi \text{ et } x \leq \frac{9\pi}{8} \\ \dfrac{8\left(x-\frac{9\pi}{8}\right)\left(-\sqrt{2}+\sqrt{\frac{1}{2}-\frac{\sqrt{2}}{4}}+1\right)}{\pi} - \sqrt{\frac{1}{2}-\frac{\sqrt{2}}{4}}+\frac{\sqrt{2}}{2} & \text{si } x \geq \frac{9\pi}{8} \text{ et } x \leq \frac{5\pi}{4} \\ \dfrac{8\left(x-\frac{5\pi}{4}\right)\left(-1-\sqrt{\frac{\sqrt{2}}{4}+\frac{1}{2}}+\sqrt{2}\right)}{\pi} - \frac{\sqrt{2}}{2}+1 & \text{si } x \geq \frac{5\pi}{4} \text{ et } x \leq \frac{11\pi}{8} \\ \dfrac{8\left(x-\frac{11\pi}{8}\right)\left(-1-\frac{\sqrt{2}}{2}+\sqrt{\frac{\sqrt{2}}{4}+\frac{1}{2}}\right)}{\pi} - \sqrt{\frac{\sqrt{2}}{4}+\frac{1}{2}}+\frac{\sqrt{2}}{2} & \text{si } x \geq \frac{11\pi}{8} \text{ et } x \leq \frac{3\pi}{2} \\ \dfrac{8\left(x-\frac{3\pi}{2}\right)\left(-\sqrt{\frac{\sqrt{2}}{4}+\frac{1}{2}}-\frac{\sqrt{2}}{2}+1\right)}{\pi} - 1 & \text{si } x \geq \frac{3\pi}{2} \text{ et } x \leq \frac{13\pi}{8} \\ \dfrac{8\left(-1+\sqrt{\frac{\sqrt{2}}{4}+\frac{1}{2}}\right)\left(x-\frac{13\pi}{8}\right)}{\pi} - \sqrt{\frac{\sqrt{2}}{4}+\frac{1}{2}}-\frac{\sqrt{2}}{2} & \text{si } x \geq \frac{13\pi}{8} \text{ et } x \leq \frac{7\pi}{4} \\ \dfrac{8\cdot\left(1-\sqrt{\frac{1}{2}-\frac{\sqrt{2}}{4}}\right)\left(x-\frac{7\pi}{4}\right)}{\pi} - 1 - \frac{\sqrt{2}}{2} & \text{si } x \geq \frac{7\pi}{4} \text{ et } x \leq \frac{15\pi}{8} \\ \dfrac{8\left(x-\frac{15\pi}{8}\right)\left(\sqrt{\frac{1}{2}-\frac{\sqrt{2}}{4}}+\frac{\sqrt{2}}{2}\right)}{\pi} - \frac{\sqrt{2}}{2} - \sqrt{\frac{1}{2}-\frac{\sqrt{2}}{4}} & \text{si } x \geq \frac{15\pi}{8} \text{ et } x \leq 2\pi \end{cases}$$

. Il est également aisé de tracer les points d'interpolation et la fonction précédente sur l'intervalle *ad hoc*, comme montré à la figure 6.2.

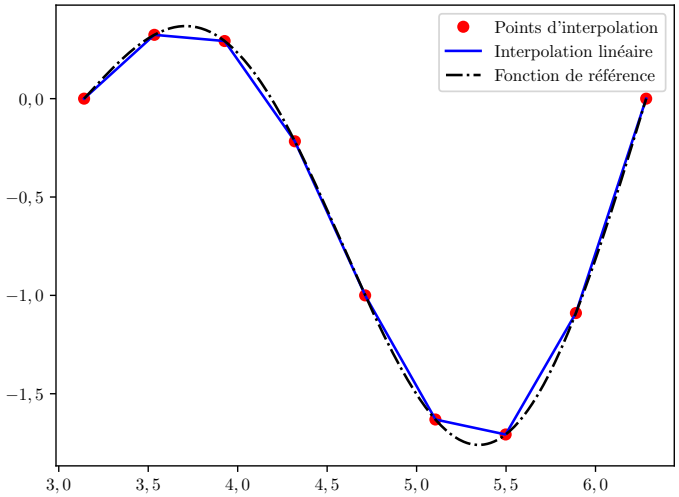


Figure 6.2 – Interpolation linéaire pour le second exemple

3 Interpolation de Lagrange

3.1 Définition

Définition 6.2 : interpolation de Lagrange

Soit $n \in \mathbb{N}^\star$et soient $\boldsymbol{t} = (t_0, \ldots, t_n)$ et $\boldsymbol{y} = (y_0, \ldots, y_n)$ deux familles de $n+1$ éléments tels que pour tout dansi, j($[\![0\,;n]\!]$), $t_i \neq t_j$ si $i \neq j$.
On appelle *polynôme d'interpolation de Lagrange* associé aux familles $\boldsymbol{t}$ et $\boldsymbol{y}$ le polynôme $p_{\boldsymbol{t},\boldsymbol{y}}$ de degré inférieur ou égal à n tel que pour tout $i \in [\![0\,;n]\!]$, $p(t_i) = y_i$.

Ceci est un cas particulier du théorème 3.6 (chinois pour les polynômes) p. 21. En effet, pour tout $(i,j) \in [\![0\,;n]\!]^2$ si $i \neq j$, $(t-t_i)$ est premier avec $(t-t_j)$ et l'équation $p(t_i) = y_i$ équivaut à l'équation $p \equiv y_i \left[t-t_i\right]$. En fait on a dans ce cas une formule explicite qui donne la solution de ce problème d'interpolation.

Propriété 6.1

Soit $n \in \mathbb{N}^\star$et soient $\boldsymbol{t} = (t_0, \ldots, t_n)$ et $\boldsymbol{y} = (y_0, \ldots, y_n)$ deux familles de $n+1$ éléments tels que pour tout dansi, j($[\![0\,;n]\!]$), $t_i \neq t_j$ si $i \neq j$.
Le polynôme de Lagrange $p_{t,y}$ est unique et s'écrit, pour tout $t \in [t_0\,;t_n]$

$$p_{t,y}(t) = \sum_{i=0}^{n} y_i \frac{\prod\limits_{\substack{0\leq j\leq n \\ j\neq i}} (t-t_j)}{\prod\limits_{\substack{0\leq j\leq n \\ j\neq i}} (t_i-t_j)} \tag{6.2}$$

Démonstration 6.1

Pour tout $t \in \mathbb{R}$ et tout $i \in [\![0\,;n]\!]$, posons $l_i(t) = \dfrac{\prod\limits_{\substack{0\leq j\leq n \\ j\neq i}} (t-t_j)}{\prod\limits_{\substack{0\leq j\leq n \\ j\neq i}} (t_i-t_j)}$.

Il est clair que pour tout $j \in [\![0\,;n]\!]$, $l_i(t_i) = 1$ et $l_i(t_j) = 0$ si $j \neq i$. Donc $p_{t,y} = \sum_{i=0}^{n} y_i l_i$ satisfait les conditions voulues.
D'autre part si q est un polynôme distinct de $p_{t,y}$ vérifiant $q(t_i) = y_i$ pour tout $i \in [\![0\,;n]\!]$, alors $q-p$ est non nul et possède $n+1$ racines distinctes. Par conséquent $\deg(q-p) \geq n+1$. Comme $\deg(p) \leq n$, on a $\deg(q) = \deg(q-p) \geq n+1$, ce qui montre que p est unique.

□

Si on cherche des polynômes de degré inférieur à n pour un problème d'interpolation faisant intervenir $n+1$ données on obtiendra seulement une « interpolation approchée ». Par exemple pour $n = 1$ la fonction `polyfit` de la librairie `numpy` permet de construire la « droite des moindres carrés » associée aux points $M_i = (t_i\,;y_i)$ pour $i \in [\![0\,;n]\!]$. Il s'agit de la droite $(\mathscr{D})$ pour laquelle la somme $\sum_{i=0}^{n} d^2\left(M_i, (\mathscr{D})\right)$ est minimale.

3.2 Premier exemple

Pour déterminer la fonction d'interpolation sur le premier nuage de points, on utilise les commandes suivantes (si le module `Interpolations` est importé) :

```
points1 = [(0, 3), (1, -2), (2, 5), (3, 7), (4, 1)] # liste des points
I_lag_1 = interpolation_Lagrange(points1) # polynôme d'interpolation de Lagrange
print("L'interpolation de Lagrange sur le premier exemple donne la fonction")
sp.pprint(I_lag_1)
```

L'exécution permet d'obtenir le polynôme p de degré défini sur $[0\,;4]$ par
$p(t) = \frac{7x^4}{12} - \frac{19x^3}{3} + \frac{251x^2}{12} - \frac{121x}{6} + 3$.
Il est également aisé de tracer les points d'interpolation et la fonction précédente sur l'intervalle *ad hoc*, comme montré à la figure 6.3. La droite des moindres carrés correspond au polynôme de degré un qui approche « au mieux » l'ensemble des points, au sens où elle minimise la somme des distances au carré entre les points et la droite.

3.3 Deuxième exemple

Pour déterminer la fonction d'interpolation sur le second nuage de points, on utilise les commandes suivantes (si le module `Interpolations` est importé) :

```
f = sp.sin(x)+sp.sin(2*x) # la fonction de référence pour l'exemple 2 en symbolique
x2 = [sp.pi*sp.Rational(1+i/8).limit_denominator(1e6) for i in range(9)] # les
  ↪  abscisses de l'exemple 2
y2 = [f.subs(x, _x) for _x in x2] # les y_i de l'exemple 2
points2 = list(zip(x2, y2, y2_prime)) # groupement sous la forme (x_i, y_i)
I_lin_2 = interpolation_linéaire(points2) # fonction d'interpolation linéaire
```

L'exécution permet d'obtenir le polynôme p de degré défini sur $[\pi\,;2\pi]$. L'expression étant ici longue car il y n'y a pas de simplification, nous ne l'afficherons pas.

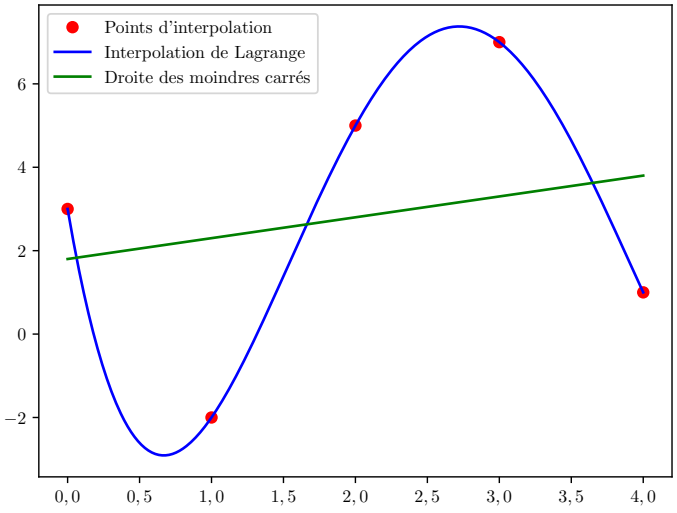


Figure 6.3 – Interpolation de Lagrange pour le premier exemple

Nous traçons les points d'interpolation et la courbe représentative du polynôme sur l'intervalle *ad hoc*, comme montré à la figure 6.4.

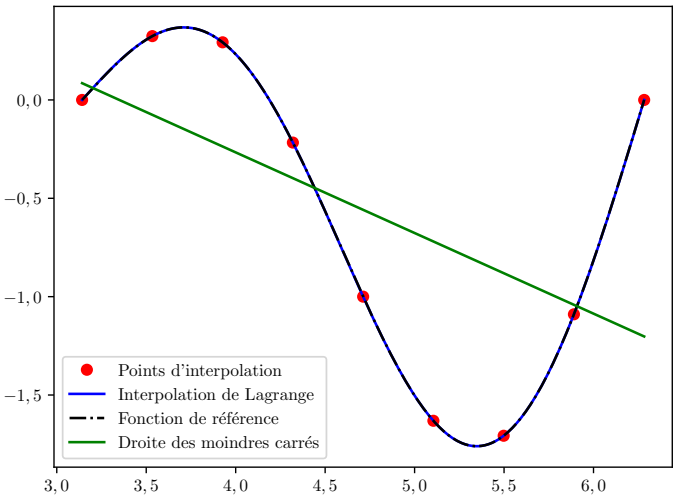


Figure 6.4 – Interpolation Lagrange pour le second exemple

Ici encore nous avons ajouté l'approximation par la droite des moindres carrés.

4 Interpolation de Hermite

4.1 Interpolation de Hermite d'ordre 1

Supposons que les points d'interpolation proviennent d'une fonction f, comme dans l'exemple 2 de ce chapitre. Il peut arriver que les polynômes de Lagrange donnent une approximation médiocre de la fonction f définie et continue sur un intervalle $[a\,;b]$. Si on pose $\boldsymbol{t}_n = \left(a + i\frac{b-a}{n}\right)_{0\leq i\leq n}$ et $\boldsymbol{y}_n = \left(f\left(a + i\frac{b-a}{n}\right)\right)_{0\leq i\leq n}$, le polynôme de Lagrange $p_{\boldsymbol{t}_n,\boldsymbol{y}_n}$ peut ne pas converger uniformément vers f quand n tend vers l'infini. C'est pourquoi Hermite avait proposé une méthode plus subtile faisant intervenir les valeurs de la fonction et celles de ses dérivées successives.

En pratique on se contente des dérivées d'ordre 1.

Définition 6.3 : interpolation de Hermite

Soit $n \in \mathbb{N}^\star$et soient $\boldsymbol{t} = (t_0, \ldots, t_n)$, $\boldsymbol{y} = (y_0, \ldots, y_n)$ et $\boldsymbol{d} = (d_0, \ldots, d_n)$ trois familles de $n+1$ éléments tels que pour tout dansi, j($[\![0\,;n]\!]$), $t_i \neq t_j$ si $i \neq j$.
On appelle *polynôme d'interpolation de Hermite d'ordre un* associé aux familles $\boldsymbol{t}$, $\boldsymbol{y}$ et $\boldsymbol{d}$ le polynôme $h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}$ de degré inférieur ou égal à $2n+$ tel que pour tout $i \in [\![0\,;n]\!]$, $h(t_i) = y_i$ et $h'(t_i) = d_i$.

Propriété 6.2

Soit $n \in \mathbb{N}^\star$et soient $\boldsymbol{t} = (t_0, \ldots, t_n)$, $\boldsymbol{y} = (y_0, \ldots, y_n)$ et $\boldsymbol{d} = (d_0, \ldots, d_n)$ trois familles de $n+1$ éléments tels que pour tout dansi, j($[\![0\,;n]\!]$), $t_i \neq t_j$ si $i \neq j$.
Le polynôme de Hermite $h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}$ est unique et s'écrit, pour tout $t \in [t_0\,;t_n]$

$$h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}(t) = \sum_{i=0}^{n} \frac{q_i(t)}{q_i(t_i)} \left[\left(1 - (t-t_i)\frac{q_i'(t_i)}{q_i(t_i)} \right) y_i + (t-t_i)d_i \right] \tag{6.3}$$

où $q_i(t) = \prod_{\substack{0\leq j\leq n \\ j\neq i}} (t-t_j)^2$.

Démonstration 6.2

On a, pour tout $(i,j) \in [\![0\,;n]\!]^2$, $q_i(t_j) = q_i'(t_j) = 0$ *si* $i \neq j$.

$$\text{Alors } h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}(t_j) = \frac{q_j(t_j)}{q_j(t_j)} \left[\left(1 - (t_j - t_j)\frac{q_j'(t_j)}{q_j(t_j)} \right) y_j + (t_j - t_j)d_j \right] = y_j$$

$$h'_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}(t_j) = \frac{q_j'(t_j)}{q_j(t_j)} \left[\left(1 - (t_j - t_j)\frac{q_j'(t_j)}{q_j(t_j)} \right) y_j + (t_j - t_j)d_j \right] + \frac{q_j(t_j)}{q_j(t_j)} \left[-\frac{q_j'(t_j)}{q_j(t_j)} y_j + d_j \right] = d_j.$$

Par construction, il est clair que $\deg\left(h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}\right) \leq 2n+1$. *Il existe donc un polynôme vérifiant les conditions. Soit* $q \in \mathbb{R}[x]$ *un autre polynôme tel que pour tout dansi*($[\![0\,;n]\!]$), $q(t_i) = y_i$ *et* $q'(t_i) = d_i$. *Posons* $r = q - h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}$. *On a alors, par construction,* $r(t_i) = r'(t_i) = 0$, *donc* r *est divisible par* $(t-t_i)^2$. *Comme* $t_i \neq t_j$ *pour* $i \neq j$, r *est divisible par* $\prod_{j=0}^{n}(t-t_j)^2$ *qui est de degré* $2n+2$. *Donc ou bien* $r = 0$, *ou bien* $\deg(r) \geq 2n+2$. *Dans le premier cas* $q = h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}} + r = h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}$, *dans le deuxième cas, comme* $\deg\left(h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}\right) \leq 2n+1$, $\deg(q) = \deg\left(h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}} + r\right) = \max(\deg\left(h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}\right), \deg(r)) \geq 2n+2$.
Par conséquent, $h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}$ *est le seul polynôme de degré inférieur ou égal à* $2n+1$ *vérifiant les conditions de la proposition.*

□

Puisque le degré du polynôme de Hermite d'ordre un associé à des familles de $n+1$ éléments est inférieur ou égal à $2n+1$, il existe une famille $(a_0, \ldots, a_{2n+1})$ telle que $h_{\boldsymbol{t},\boldsymbol{y},\boldsymbol{d}}(t) = \sum_{i=0}^{2n+1} a_i t^i$. Il est clair que ces coefficients sont les solutions du système

$$\begin{cases} \sum_{i=0}^{2n+1} a_i t_0^i = y_0 \\ \ldots \\ \sum_{i=0}^{2n+1} a_i t_n^i = y_n \\ \sum_{i=1}^{2n} i a_i t_0^{i-1} = d_0 \\ \ldots \\ \sum_{i=1}^{2n} i a_i t_n^{i-1} = d_n \end{cases}$$

pour limiter l'ordre du polynôme d'interpolation, il est possible de travailler « par morceaux »

4.2 Premier exemple

Pour déterminer l'interpolation de Hermite sur le premier nuage de points, on utilise les commandes suivantes (si le module `Interpolations` est importé) :

```
points1 = [(0, 3, 0), (1, -2, 0), (2, 5, -1), (3, 7, 2), (4, 1, 0)] # ici (t_i, y_i, d_i)
I_herm_1 = interpolation_Hermite(points1)
print("L'interpolation de Hermite sur le premier exemple donne la fonction")
sp.pprint(I_herm_1)
```

L'exécution permet d'obtenir le polynôme h de degré défini sur $[0\,;4]$ par
$p(t) = -\frac{707x^9}{1728} + \frac{6557x^8}{864} - \frac{50315x^7}{864} + \frac{102859x^6}{432} - \frac{959251x^5}{1728} + \frac{628577x^4}{864} - \frac{208901x^3}{432} + \frac{953x^2}{8} + 3.$
Il est également aisé de tracer les points d'interpolation et la fonction précédente sur l'intervalle *ad hoc*, comme montré à la figure 6.5.

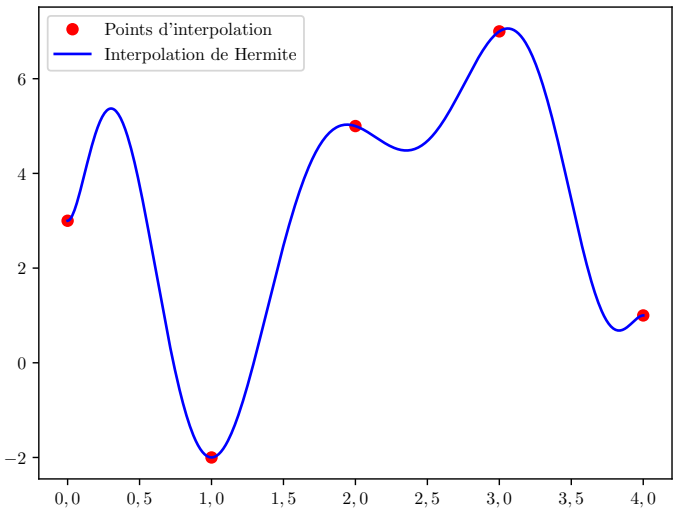


FIGURE 6.5 – Interpolation de HERMITE pour le premier exemple

4.3 Deuxième exemple

Pour déterminer la fonction d'interpolation sur le second nuage de points, on utilise les commandes suivantes (si le module `Interpolations` est importé) :

```
f = sp.sin(x)+sp.sin(2*x) # la fonction de référence pour l'exemple 2 en symbolique
df = f.diff() # sa dérivée
x2 = [sp.pi*sp.Rational(1+i/8).limit_denominator(1e6) for i in range(9)] # les
  ↪ abscisses de l'exemple 2
y2, y2_prime = [f.subs(x, _x) for _x in x2], [df.subs(x, _x) for _x in x2] # les y_i et
  ↪ d_i de l'exemple 2
points2 = list(zip(x2, y2, y2_prime)) # groupement sous la forme (t_i, y_i, d_i)
I_herm_2 = interpolation_Hermite(points2) # fonction d'interpolation de Hermite
```

L'exécution permet d'obtenir le polynôme h de degré défini sur $[\pi\,;2\pi]$. L'expression étant ici longue car il y n'y a pas de simplification, nous ne l'afficherons pas.

Nous traçons les points d'interpolation et la courbe représentative du polynôme sur l'intervalle *ad hoc*, comme montré à la figure 6.6.

5 Application

Un cas d'application de l'interpolation est l'intégration numérique.

Supposons que l'on a une fonction g, intégrable sur un intervalle $[a\,;b]$. On cherche à déterminer une approximation numérique de $\mathrm{I} = \int_a^b g(t)\,\mathrm{d}t$.

Soit $n \in \mathbb{N}^\star$ et $(t_0,\ \ldots,\ t_n)$ une subdivision de l'intervalle $[a\,;b]$. On a donc $a = t_0 < t_1 < \ldots < t_{n-1} < t_n = b$. On pose $h_k = t_{k+1} - t_k$ la longueur de l'intervalle $\left[t_k\,;t_{k+1}\right]$. Alors $\mathrm{I} = \sum_{k=0}^{n-1} \int_{t_k}^{t_{k+1}} g(t)\,\mathrm{d}t$.

Effectuons le changement de variable $t = x\frac{h_k}{2} + \frac{t_k+t_{k+1}}{2} = \frac{t_{k+1}}{2}(x+1) + \frac{t_k}{2}(1-x)$: pour $t \in \left[t_k\,;t_{k+1}\right]$, on a alors $x \in [-1\,;1]$.

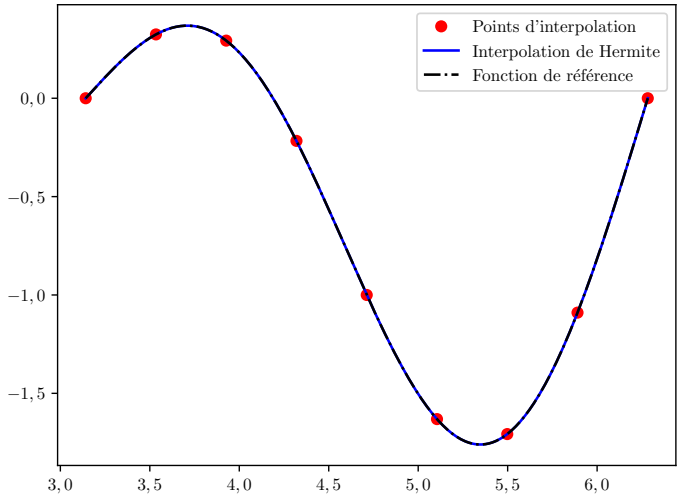


FIGURE 6.6 – Interpolation HERMITE pour le second exemple

Alors $\int_{t_k}^{t_{k+1}} g(t)\,\mathrm{d}t = \int_{-1}^{1} g\left(x\frac{h_k}{2} + \frac{t_k+t_{k+1}}{2}\right)\frac{h_k}{2}\,\mathrm{d}x = \frac{h_k}{2}\int_{-1}^{1} f_k(x)\,\mathrm{d}x$, où f_k est la fonction définie sur $[-1\,;1]$ par $f_k(x) = g\left(x\frac{h_k}{2} + \frac{t_k+t_{k+1}}{2}\right)$.

Considérons une subdivision $(x_0,\ \ldots,\ x_m)$ de $[-1\,;1]$. Soit p_m le polynôme d'interpolation de Lagrange de f_k en ces points : $p_m(x) = \sum_{i=0}^{m} f_k(x_i)\dfrac{\prod\limits_{\substack{0\leq j\leq n\\ j\neq i}}(x-x_j)}{\prod\limits_{\substack{0\leq j\leq n\\ j\neq i}}(x_i-x_j)} = \sum_{i=0}^{n} f_k(x_i)l_i(x)$, où $l_i(x) = \dfrac{\prod\limits_{\substack{0\leq j\leq n\\ j\neq i}}(x-x_j)}{\prod\limits_{\substack{0\leq j\leq n\\ j\neq i}}(x_i-x_j)}$. Ces fonctions l_i sont donc indépendantes de f_i ou de g : elles ne sont construites qu'à partir de la subdivision $(x_0,\ \ldots,\ x_m)$ de $[-1\,;1]$.

De la sorte, $\displaystyle\int_{-1}^{1} f_k(x)\,\mathrm{d}x \approx \int_{-1}^{1} p_m(x)\,\mathrm{d}x$

$$\approx \sum_{i=0}^{m} f_k(x_i)\int_{-1}^{1} l_i(x)\,\mathrm{d}x$$

$$\approx \sum_{i=0}^{m} f_k(x_i)\omega_i, \text{ où les } \omega_i \text{ sont les poids d'intégration numérique.}$$

Comme les fonctions l_i ne dépendent que de la subdivision $(x_0,\ \ldots,\ x_m)$ de $[-1\,;1]$, il suffit de calculer une fois les poids d'intégration.

L'intégrale initiale se ramène ainsi à intégrer des polynômes l_i puis à effectuer des sommes.

6 Courbes de Bézier

6.1 Barycentres

On se place dans l'espace affine de dimension 3 usuel. Un point pondéré est un couple $(\mathrm{A}\,;m)$ formé d'un point A et d'un réel m.

Propriété 6.3

Soit $n \in \mathbb{N}$. Soit $(\mathbf{A}\,;\boldsymbol{m}) = \left\{\left(\mathrm{A}_0\,;m_0\right)\,;\ldots;\left(\mathrm{A}_n\,;m_n\right)\right\}$ une famille de n points pondérés telle que $\mu = \sum_{k=0}^{n} m_k \neq 0$.

Il existe alors un unique point G, appelé le *barycentre* de la famille $(\mathbf{A}\,;\boldsymbol{m})$, vérifiant

$$\sum_{k=0}^{n} m_k\overrightarrow{\mathrm{GA}_k} = \vec{0}. \tag{6.4}$$

De plus on a, pour tout point U du plan,

$$\overrightarrow{\mathrm{UG}} = \frac{1}{\mu}\sum_{k=0}^{n} m_k \overrightarrow{\mathrm{UA}_k}. \tag{6.5}$$

Démonstration 6.3

Soit un point U *de l'espace. Par la relation de* Chasles*, on a pour tout point* M*,*

$$\sum_{k=0}^{n} m_k \overrightarrow{\mathrm{MA}_k} = \sum_{k=0}^{n} m_k \left(\overrightarrow{\mathrm{MU}} + \overrightarrow{\mathrm{UA}_k}\right) = \mu\overrightarrow{\mathrm{UM}} + \sum_{k=0}^{n} m_k \overrightarrow{\mathrm{UA}_k}.$$

Donc G *vérifie la condition* (6.4) *si et seulement si la formule* (6.5) *est vérifiée par* U *et* G*. Il existe donc un unique point* G *vérifiant* (6.4)*, et la formule* (6.5) *est vérifiée par* U *et* G*, pour tout* U*.*

□

Les barycentres possèdent une propriété d'associativité, donnée par le résultat suivant.

Propriété 6.4

Soit $n \in \mathbb{N}$ et soit $(\mathbf{A}\,;\boldsymbol{m}) = \left\{\left(\mathrm{A}_0\,;m_0\right)\,;\ldots;\left(\mathrm{A}_n\,;m_n\right)\right\}$ une famille de points pondérés telle que $\sum_{k=0}^{n} m_k \neq 0$. Soit $\left(\mathrm{I}_0,\ \ldots,\ \mathrm{I}_s\right)$ une famille de sous-ensembles finis disjoints deux à deux de $\{0\,;\ldots;n\}$ vérifiant les propriétés suivantes

1. $\cup_{0\leq i\leq s}\mathrm{I}_i = \{0\,;\ldots;n\}$
2. $\tilde{m}_i = \sum_{j\in\mathrm{I}_i} m_j \neq 0$ pour tout $i \in [\![0\,;s]\!]$.

Soit G le barycentre de la famille $(\mathbf{A}\,;\boldsymbol{m})$. Soit G_i le barycentre de la famille $\left\{\left(\mathrm{A}_j\,;m_j\right)\ \middle|\ j\in\mathrm{I}_i\right\}$. Alors G est égal au barycentre de la famille $\left\{\left(\mathrm{G}_i\,;\tilde{m}_i\right)\ \middle|\ i\in[\![0\,;s]\!]\right\}$.

Démonstration 6.4

Par définition, on a

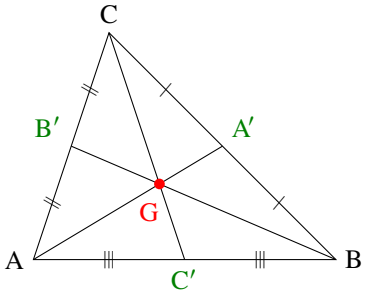
$$\sum_{i=0}^{s}\tilde{m}_i\overrightarrow{\mathrm{GG}_i} = \sum_{i=0}^{s}\left(\sum_{j\in\mathrm{I}_i} m_j\overrightarrow{\mathrm{GA}_j}\right) = \sum_{j=0}^{n} m_i\overrightarrow{\mathrm{GA}_i} = \vec{0}.$$

Donc G *est bien le barycentre de la famille* $\left\{\left(\mathrm{G}_i\,;\tilde{m}_i\right)\ \middle|\ i\in[\![0\,;s]\!]\right\}$*.*

□

Exemple 6.1

Soit ABC un triangle quelconque. Soit A′ le milieu de BC, soit B′ le milieu de CA et soit C′ le milieu de AB. Alors les médianes AA′, BB′ et CC′ sont concourantes en un point G tel que $\overrightarrow{\mathrm{GA}} = -2\overrightarrow{\mathrm{GA'}}$, $\overrightarrow{\mathrm{GB}} = -2\overrightarrow{\mathrm{GB'}}$ et $\overrightarrow{\mathrm{GC}} = -2\overrightarrow{\mathrm{GC'}}$.
En effet, soit G le barycentre de la famille $\{(\mathrm{A}\,;1)\,;(\mathrm{B}\,;1)\,;(\mathrm{C}\,;1)\}$. Comme A′ est le milieu de BC, A′ est le barycentre de la famille $\{(\mathrm{B}\,;1)\,;(\mathrm{C}\,;1)\}$. Donc par associativité on voit que G est le barycentre de la famille $\left\{(\mathrm{A}\,;1)\,;\left(\mathrm{A'}\,;2\right)\right\}$ et $\overrightarrow{\mathrm{GA}} + 2\overrightarrow{\mathrm{GA'}} = \vec{0}$, ce qui montre en particulier que G appartient à la droite $\left(\mathrm{AA'}\right)$. On établit de même que $\overrightarrow{\mathrm{GB}} + 2\overrightarrow{\mathrm{GB'}} = \overrightarrow{\mathrm{GC}} + 2\overrightarrow{\mathrm{GC'}} = \vec{0}$, et G appartient aux trois médianes du triangle.



6.2 Polynômes de Bernstein

Définition 6.4 : polynômes de Bernstein

Soient n et k deux entiers, avec $n \geq 1$ et $0 \leq k \leq n$. Le *polynôme de Bernstein* $\mathrm{B}_{n,k}$ est défini sur $\mathbb{R}$ par la formule

$$\mathrm{B}_{n,k}(x) = \binom{n}{k} x^k (1-x)^{n-k} \tag{6.6}$$

où $\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\ldots(n-k+1)}{k!}$, avec les conventions usuelles $0! = 1! = x^0 = (1-x)^0 = 1$.

Propriété 6.5

Pour tout $n \in \mathbb{N}^\star$, on a les propriétés suivantes :

1. (partition de l'unité) $\sum_{k=0}^{n} \mathrm{B}_{n,k} = 1$;
2. (symétrie) pour tout $k \in [\![0\,;n]\!]$ et pour tout $u \in \mathbb{R}$, $\mathrm{B}_{n,k}(1-u) = \mathrm{B}_{n,n-k}(u)$;
3. (positivité) pour tout $k \in [\![0\,;n]\!]$ et pour tout $u \in [0\,;1]$, $\mathrm{B}_{n,k}(u) \geq 0$;
4. (formules de récurrence) $\begin{cases} \mathrm{B}_{n,0} = (1-x)\mathrm{B}_{n-1,0} & \forall n \geq 2 \\ \mathrm{B}_{n,k} = (1-x)\mathrm{B}_{n-1,k} + x\mathrm{B}_{n-1,k-1} & \forall n \geq 2, \forall k \in [\![1\,;n-1]\!] \\ \mathrm{B}_{n,n} = x\mathrm{B}_{n-1,n-1} & \forall n \geq 2 \end{cases}$.

Démonstration 6.5

La partition de l'unité résulte immédiatement de la formule du binôme de Newton, puisqu'on a

$$\sum_{k=0}^{n} \mathrm{B}_{n,k} = \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k} = (x+1-x)^k = 1.$$

La symétrie, la positivité et le fait que $\mathrm{B}_{n,0} = (1-x)\mathrm{B}_{n-1,0}$ *et* $\mathrm{B}_{n,n} = x\mathrm{B}_{n-1,n-1}$ *pour* $n \geq 2$ *sont évidents. Enfin, pour* $n \geq 2$, *et pour tout* $k \in [\![1\,;n-1]\!]$, *on a* $\binom{n-1}{k} + \binom{n-1}{k-1} = \frac{(n-1)!}{k!(n-k-1)!} + \frac{(n-1)!}{(k-1)!(n-k)!} = \frac{(n-1)!(n-k+k)}{k!(n-k)!} = \frac{n!}{k!(n-k)!} = \binom{n}{k}$. *Par conséquent,*

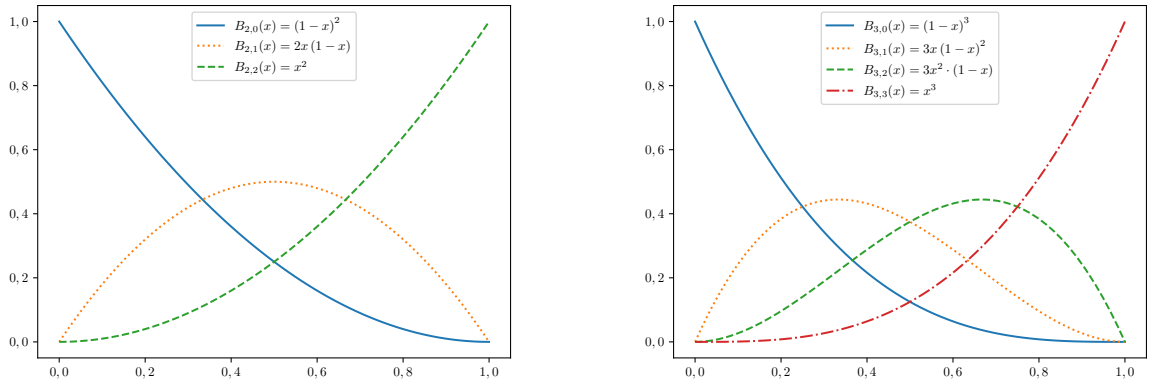
$$\begin{aligned}(1-x)\mathrm{B}_{n-1,k} + x\mathrm{B}_{n-1,k-1} &= \binom{n-1}{k} x^k (1-x)^{n-k} + \binom{n-1}{k-1} x^k (1-x)^{n-k} \\ &= \binom{n}{k} x^k (1-x)^{n-k} \\ &= \mathrm{B}_{n,k}.\end{aligned}$$

□

Pour fixer les idées, on trace à la figure 6.7 avec *Python* les courbes représentatives de ces polynômes pour $n = 2$ (a) et $n = 3$ (b).

```
import sympy as sp # librairie de calcul formel
import matplotlib.pyplot as plt # librairie graphique

sp.var("x") # déclaration de la variable symbolique x

def polynôme_Bernstein(n, k):
  # fonction pour construire le polynôme de Bernstein B_{n,k}
  assert (k<=n), f"Il faut k <= n. On a ici k={k}>n={n}"
  return sp.binomial(n, k)*x**k*(1-x)**(n-k)

style_de_ligne = ["solid", "dotted", "dashed", "dashdot"] # pour différencier les
  ↪ courbes

for n in [2, 3]:
```

```
14  B_nk = [polynôme_Bernstein(n, k) for k in range(n+1)]
15  courbes = sp.plot(*[(_bnk, (x, 0, 1)) for _bnk in B_nk], show=False) # construction
    ↪ des nuages de points
16  plt.close()
17  plt.figure()
18  # ajout des courbes
19  [plt.plot(*_courbe.get_points(), linestyle=style_de_ligne[_i], label=f"$B_{{{n},
    ↪ {_i}}}(x)={sp.latex(B_nk[_i])}$") for _i, _courbe in enumerate(courbes)]
20  plt.title(f"Polynômes de Bernstein pour $n={n}$")
21  plt.legend()
22  plt.show()
```



(a) $n = 2$ (b) $n = 3$

Figure 6.7 – Polynômes de Bernstein

6.3 Courbes de Bézier

Pierre Bézier, est né le 1er septembre 1910 à Paris et mort le 25 novembre 1999 à Bures-sur-Yvette (Essonne) Il intègre en 1927 l'École nationale supérieure des arts et métiers (ENSAM) en étant premier au concours d'entrée. Il est diplômé en 1930. Il obtient également le diplôme d'ingénieur de l'École supérieure d'électricité (Supélec) en 1931. Il soutient un doctorat en mathématiques à l'université de Poitiers en 1962 et un doctorat d'État en physique à l'université Pierre-et-Marie-Curie en 1973.

En 1963, ingénieur chez Renault, il développe les courbes polynômiales suivantes associées à une famille de points, qui portent son nom.

Définition 6.5 : courbe de Bézier

La *courbe de Bézier* associée à une famille de $n+1$ points $(P_0, \ldots, P_n)$ est la courbe paramétrée définie sur $[0\,;1]$ par $G(t)$ le barycentre de la famille $\{(P_0\,;B_{n,0}(t))\,;(P_1\,;B_{n,1}(t))\,;\ldots;(P_n\,;B_{n,n}(t))\}$, où pour tout $j \in [\![0\,;n]\!]$, $B_{n,j}$ est le polynôme de Bernstein.

En d'autre termes, si M est un point du plan, le point $G(t)$ est défini par la formule vectorielle

$$\overrightarrow{MG(t)} = \sum_{j=0}^{n} B_{n,j}(t)\overrightarrow{MP_j}. \tag{6.7}$$

Premier exemple

Pour déterminer la courbe de Bézier associée aux points du premier exemple, utilisons les commandes *Python* suivantes (si le module `Interpolations` est importé) :

```
points1 = [(0, 3), (1, -2), (2, 5), (3, 7), (4, 1)]
Bézier1 = courbe_Bézier(points1)
print("La courbe paramétrée de Bézier sur le premier exemple donne (pour x dans [0,
  ↪ 1])")
sp.pprint(Bézier1)
```

L'exécution permet d'obtenir l'équation paramétrique de la courbe de Bézier définie sur $[0\,;1]$ par

$$\begin{cases} 4x \\ 14x^4 - 68x^3 + 72x^2 - 20x + 3 \end{cases}.$$

Il est également aisé de tracer les points de contrôle et la courbe précédente sur l'intervalle *ad hoc*, comme montré à la figure 6.8.

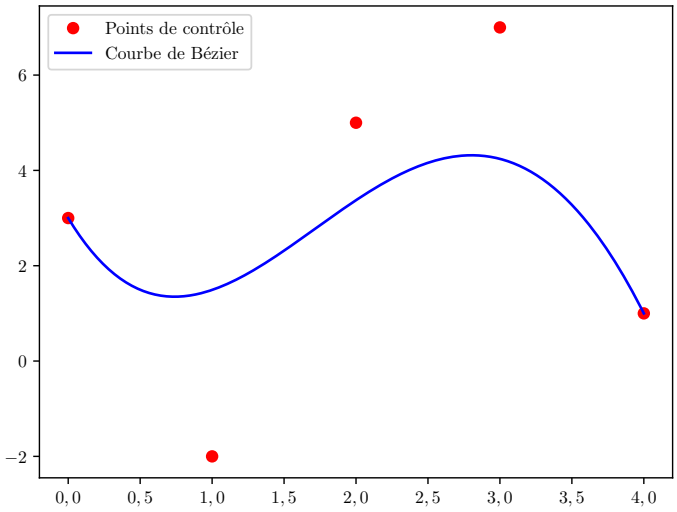


Figure 6.8 – Courbe de Bézier pour le premier exemple

Deuxième exemple

Pour déterminer la courbe de Bézier associée aux points du second exemple, utilisons les commandes *Python* suivantes (si le module `Interpolations` est importé) :

```
f = sp.sin(x)+sp.sin(2*x) # la fonction de référence pour l'exemple 2 en symbolique
x2 = [sp.pi*sp.Rational(1+i/8).limit_denominator(1e6) for i in range(9)] # les abscisses de l'exemple 2
y2 = [f.subs(x, _x) for _x in x2] # les y_i de l'exemple 2
points2 = list(zip(x2, y2)) # groupement sous la forme (t_i, y_i)
Bézier2 = courbe_Bézier(points2)
print("La courbe paramétrée de Bézier sur le second exemple donne (pour x dans [0, 1])")
sp.pprint(Bézier2)
```

L'exécution permet d'obtenir l'équation paramétrique de la courbe de Bézier définie sur $[0\,;1]$. Comme cette expression n'est pas simple, nous choisissons de ne pas l'afficher.

Il est aisé de tracer les points de contrôle et la courbe précédente sur l'intervalle *ad hoc*, comme montré à la figure 6.9.

6.4 L'algorithme de de Casteljau

Paul de Faget de Casteljau, né en 1930, est un ancien élève de l'École normale supérieure. Il est ensuite rentré chez Citroën et a travaillé dans le groupe de recherche « Fraisage Numérique » du département « Service Outillage » qui devait révolutionner la modélisation des formes des automobiles.

Ses travaux théoriques ont été enregistrés à l'INPI (Institut National de la Propriété Industrielle, Paris) et ont fait seulement l'objet d'une diffusion interne à Citroën. Il avait introduit les mêmes courbes que Bézier, appelées « courbes à pôles » par son supérieur, mais ces courbes ont pris le nom de courbes de Bézier car les travaux de Bézier chez Renault ont été rendus publics dès 1965. C'est seulement en 1975 que les travaux de de Casteljau ont été diffusés, avec en particulier un algorithme, connu sous le nom d'algorithme de de Casteljau, que nous allons maintenant décrire. Cet algorithme est basé sur l'observation suivante.

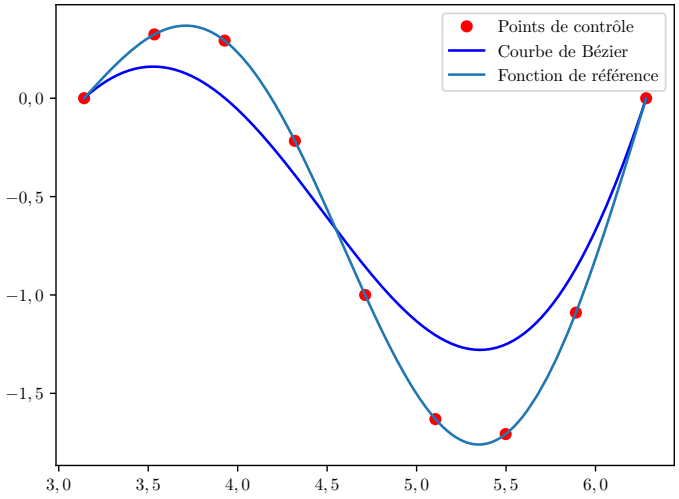


Figure 6.9 – Courbe de Bézier pour le deuxième exemple

Propriété 6.6

Soit $(P_0, \ldots, P_n)$ une famille de $n+1$ points et soit G définie sur $[0\,;1]$ la courbe paramétré de Bézier associée. Pour tout $j \in [\![0\,;n-1]\!]$ et tout $t \in [0\,;1]$, on pose $P_{j,t}$ le barycentre de $\{(P_j\,;1-t)\,;(P_{j+1}\,;t)\}$ et G_1 définie sur $[0\,;1]$ la courbe de Bézier associée à la famille $(P_{0,t}, \ldots, P_{n-1,t})$.
Alors pour tout $t \in [0\,;1]$, $G(t) = G_1(t)$.

Démonstration 6.6

Fixons un point U. *Il résulte des formules de récurrence de la propriété 6.5 que l'on a*

$$\begin{aligned}
\overrightarrow{UG_1}(t) &= \sum_{k=0}^{n-1} B_{n-1,k}(t)\overrightarrow{UQ_k} \\
&= \sum_{k=0}^{n-1}(1-t)B_{n-1,k}(t)\overrightarrow{UP_k} + \sum_{k=0}^{n-1} tB_{n-1,k-1}(t)\overrightarrow{UP_{k+1}} \\
&= \sum_{k=0}^{n-1}(1-t)B_{n-1,k}(t)\overrightarrow{UP_k} + \sum_{k=1}^{n} tB_{n-1,k-1}(t)\overrightarrow{UP_k} \\
&= (1-t)B_{n-1,0}(t) + \sum_{k=1}^{n-1}\left((1-t)B_{n,}(t) + tB_{n-1,k-1}(t)\right)\overrightarrow{UP_k} + tB_{n-1,n-1}(t)\overrightarrow{UP_n} \\
&= \sum_{k=0}^{n} B_{n,k}(t)\overrightarrow{UP_k} = \overrightarrow{UG}(t)\,.
\end{aligned}$$

□

L'algorithme de de Casteljau consiste, étant donné une famille $(P_0, \ldots, P_n)$ de points et un réel $t \in [0\,;1]$, à construire par récurrence finie pour $i \in [\![0\,;n]\!]$ des familles $(Q_{i,j})_{j\in[\![0\,;n-i]\!]}$ en suivant la procédure suivante

- $Q_{0,j} = P_j$ pour tout $j \in [\![0\,;n]\!]$;
- $Q_{i,j} = (1-t)Q_{i-1,j} + tQ_{i-1,j+1}$ pour $i \in [\![1\,;n]\!]$ et $j \in [\![0\,;n-i]\!]$.

Il résulte alors de la proposition que $G(t) = Qn, 0$, où G est la courbe paramétrée de Bézier définie sur $[0\,;1]$, associée à la famille $(P_0, \ldots, P_n)$.

Premier exemple

Pour déterminer la courbe de de Casteljau associée aux points du premier exemple, utilisons les commandes *Python* suivantes (si le module `Interpolations` est importé) :

```
points1 = [(0, 3), (1, -2), (2, 5), (3, 7), (4, 1)]
x1, y1 = [_p[0] for _p in points1], [_p[1] for _p in points1]
deCasteljau1 = courbe_deCasteljau(points1)
print("Équation de la première coordonnée")
```

```
sp.pprint(sp.simplify(sp.expand(deCasteljau1[-1][0][0])))
print("Équation de la deuxième coordonnée")
sp.pprint(sp.simplify(sp.expand(deCasteljau1[-1][0][1])))
```

Par exemple, pour $t = \frac{2}{3}$, on a la construction du point G(t) en étapes. Le lieu géométrique du point G(t), pour $t \in [0\,;1]$ forme la courbe de Bézier associée aux points de contrôle. Ici, l'équation paramétrée est donnée par $\begin{cases} x(t) = 4t \\ y(t) = 14t^4 - 68t^3 + 72t^2 - 20t + 3 \end{cases}$

Ceci est illustré à la figure 6.10.

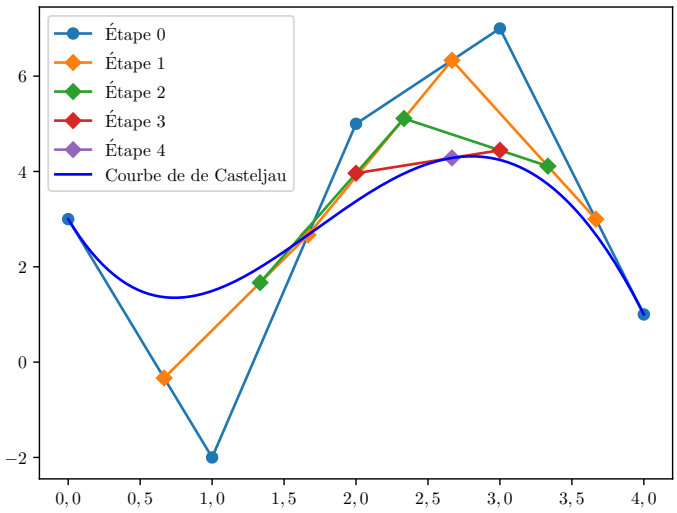


Figure 6.10 – Courbe de de Casteljau pour le premier exemple

Deuxième exemple

Pour déterminer la courbe de Bézier associée aux points du second exemple, utilisons les commandes *Python* suivantes (si le module `Interpolations` est importé) :

```
f = sp.sin(x)+sp.sin(2*x) # la fonction de référence pour l'exemple 2 en symbolique
x2 = [sp.pi*sp.Rational(1+i/8).limit_denominator(1e6) for i in range(9)] # les
  ↪  abscisses de l'exemple 2
y2 = [f.subs(x, _x) for _x in x2] # les y_i de l'exemple 2
points2 = list(zip(x2, y2)) # groupement sous la forme (x_i, y_i)
deCasteljau2 = courbe_deCasteljau(points2)
print("Équation de la première coordonnée")
sp.pprint(sp.simplify(sp.expand(deCasteljau2[-1][0][0])))
print("Équation de la deuxième coordonnée")
sp.pprint(sp.simplify(sp.expand(deCasteljau2[-1][0][1])))
```

Par exemple, pour $t = \frac{2}{3}$, on a la construction du point G(t) en étapes. Le lieu géométrique du point G(t), pour $t \in [0\,;1]$ forme la courbe de Bézier associée aux points de contrôle. Ici, l'équation paramétrée est donnée par une expression trop longue pour être affichée.

La construction est illustrée à la figure 6.11.

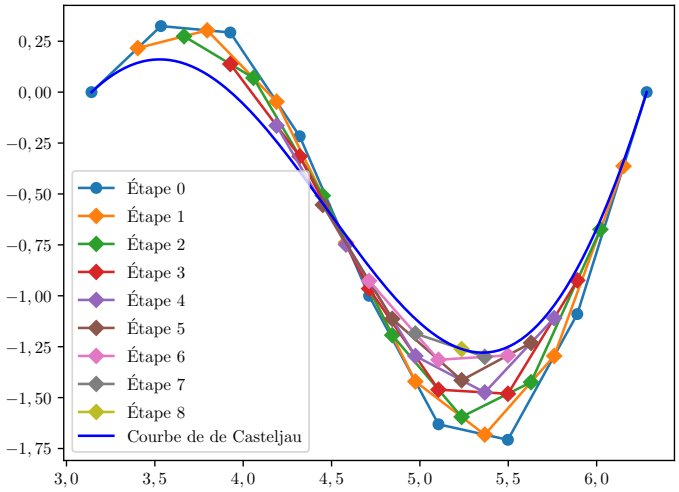


Figure 6.11 – Courbe de de Casteljau pour le second exemple

Annexe A

Avec Python

Sommaire

On donne ici des scripts utilisés dans ce cours pour illustrer les exemples.

1 Arithmétique

Code A.1 : arithmétique et *Python*

```python
#!/usr/bin/env python

# Auteur : Nicolas BUR
# Date : 2023/03/15
# Version : 1.0

"""
Présentation de quelques commandes sympy relatives à l'arithmétique
"""

import sympy as sp # import de la librairie

a = 5; b = 24; c = 19
print(f"{a} est-il premier ? {sp.isprime(a)}")
print(f"{b} est-il premier ? {sp.isprime(b)}")

print(f"\nLe {a}e nombre premier est {sp.prime(a)}")
print(f"Les nombres premiers inférieurs à {c} sont {list(sp.primerange(c))}")
print(f"Les nombres premiers jusqu'au {a}e sont {list(sp.primerange(sp.prime(a)
    ↪ + 1))}")

print(f"\nLes facteurs premiers de la décomposition de {b} sont
    ↪ {sp.primefactors(24)}")
print(f"La décomposition en facteurs premiers de {b} est {sp.factorint(24)}")
print(f"Les diviseurs de {b} sont {sp.divisors(24)}")

```

```
a = 55; b = 132
pgdc = sp.gcd(a, b)
ppmc = sp.lcm(a, b)
print(f"\nLe pgdc de {a} et de {b} est {pgdc}")
print(f"Le ppmc de {a} et de {b} est {ppmc}")
print(f"On vérifie pgdc(a, b) × ppmc(a, b) = a × b : {a*b==ppmc*pgdc}")

print(f"\nUne solution particulière de l'équation de Bézout {a} × u + {b} × v =
↪ {pgdc} est\n(u, v, pgdc) = {sp.gcdex(a, b)}")

couples_r_m = [(1, 2), (-1, 3), (2, 5)]
sol = sp.ntheory.modular.solve_congruence(*couples_r_m)
# on peut aussi coder
# sol = sp.ntheory.modular.crt([2, 3, 5], [1, -1, 2])
print(f"\nSoit le système de congruence\n{f'{chr(10)}'.join([f'x congru à {_r}
↪ modulo {_m}' for _r, _m in couples_r_m])}\nLes solutions sont alors de la
↪ forme x congru à {sol[0]} modulo {sol[1]}")
```

Le code A.1 est donné dans le fichier *Arithmetique.py* inclus au présent document.

2 Polynômes

Code A.2 : polynômes et *Python*

```
#!/usr/bin/env python

# Auteur : Nicolas BUR
# Date : 2023/03/15
# Version : 1.0

"""
Présentation de quelques commandes sympy relatives aux polynômes
"""

import sympy as sp # import de la librairie
sp.var("x") # déclaration de la variable symbolique x

# définition de deux polynômes
P = sp.Poly(x**3 + 1)
Q = sp.Poly(x**2 + 2*x + 1) # attention à ne pas oublier les opérateurs

p = P.degree() # obtention du degré
q = Q.degree()
print(f"Le degré du polynômes P\n{sp.pretty(P.as_expr())}\nvaut {p}")
print(f"Le degré du polynômes Q\n{sp.pretty(Q.as_expr())}\nvaut {q}")

print("\n** Opérations entre polynômes**")
S = P + Q # somme
T = P * Q # produit
U = P**3 # puissance
print(f"Le degré du polynômes P+Q\n{sp.pretty(S.as_expr())}\nvaut {S.degree()}")
assert S.degree() <= max(p, q), "deg(P + Q) > max(deg(P), deg(Q))" #
↪ vérification sur le degré
print(f"Le degré du polynômes P*Q\n{sp.pretty(T.as_expr())}\nvaut {T.degree()}")
assert T.degree() == p+q, "deg(PQ) != deg(P) + deg(Q)" # vérification sur le
↪ degré
```

```
31 print(f"Le degré du polynômes P**3\n{sp.pretty(U.as_expr())}\nvaut
    ↪ {U.degree()}")
32 assert U.degree() == 3*p, "deg(P**n) != n*deg(P)" # vérification sur le degré
33
34 B, R = P.div(Q) # division euclidienne : P = BQ + R
35 print(f"Les polynômes B et R tels que P = QB + R sont\nB(x) = {B.as_expr()} et
    ↪ R(x) = {R.as_expr()}")
36
37 print("\n** Racines de polynômes**")
38 racines = sp.roots(P)
39 print("Les racines de P sont :")
40 [print(f"{sp.pretty(_r)} de multiplicité {_m}") for _r, _m in racines.items()]
41 racines_réelles = P.real_roots()
42 print("soit en valeurs numériques :")
43 [sp.pprint(_r) for _r in P.nroots()]
44
45 print("Les racines réelles de P sont :")
46 [sp.pprint(_r) for _r in racines_réelles]
47 racines = sp.roots(Q)
48 print("Les racines de Q sont :")
49 [print(f"{sp.pretty(_r)} de multiplicité {_m}") for _r, _m in racines.items()]
50
51 racines_solve = sp.solve(sp.Eq(Q, 0)) # autre manière de faire
52 print(f"Reprenons le polynômes Q.\nSes racines sont\n")
53 [sp.pprint(_r) for _r in racines_solve]
54 print("On utilise les dérivées pour déterminer les multiplicités")
55 for _r in racines_solve:
56     m = 1
57     while True:
58         Q_der = sp.diff(Q, (x, m))
59         if Q_der.subs(x, _r) != 0:
60             break
61         m +=1
62     print(f"{sp.pretty(_r)} est racine de multiplicité {m}")
63
64 print("\n** Décomposition en polynômes irréductibles **")
65
66 décomposition_P = P.factor_list()
67 print("Décomposition P = a × ∏P_i^{k_i} :")
68 print(f"a = {décomposition_P[0]}")
69 for _pi in décomposition_P[1]:
70     print(f"{sp.pretty(_pi[0].as_expr())}, de multiplicité {_pi[1]}")
71
72 print("\n** PGDC, PPMC et équation de Bézout**")
73 PGDC = sp.gcd(P, Q)
74 print(f"Le PGDC de P et de Q vaut\n{sp.pretty(PGDC.as_expr())}")
75
76 PPMC = sp.lcm(P, Q)
77 print(f"Le PPMC de P et de Q vaut\n{sp.pretty(PPMC.as_expr())}")
78
79 U, V, PGDC = sp.gcdex(P, Q)
80 print(f"On cherche U et V tels que PU + QV = Pgdc(P, Q).\nOn peut prendre
    ↪ U =\n{sp.pretty(U.as_expr())}\net V =\n{sp.pretty(V.as_expr())}")
81 print(f"On vérifie l'égalité : {sp.simplify(P*U+Q*V) == PGDC}")
```

Le code A.2 est donné dans le fichier *Polynomes.py* inclus au présent document.

3 Fractions rationnelles

Code A.3 : fractions rationnelles et *Python*

```python
#!/usr/bin/env python

# Auteur : Nicolas BUR
# Date : 2023/03/15
# Version : 1.0

"""
Présentation de quelques commandes sympy relatives aux fractions rationnelles
"""

import sympy as sp # import de la librairie

sp.var("x") # déclaration de la variable symbolique x

# définition des polynômes
P = sp.Poly(x**7 + 2*x**5 + 4*x**2 - x + 1)
Q = sp.Poly(x**2 * (x**2 + 1)**2)
R = P / Q

print(f"Le Pgdc de P est de Q vaut\n{sp.pretty(sp.gcd(P, Q))}")

print(f"Soit la fraction rationnelle R = P / Q :\n{sp.pretty(R)}")

# Décomposition en éléments simples de P / Q
DES = sp.apart(P/Q)
print(f"Sa décomposition en éléments simples s'écrit\n{sp.pretty(DES)}")
```

Le code A.3 est donné dans le fichier *Fractions_rationnelles.py* inclus au présent document.

4 Projecteurs spectraux

Code A.4 : projecteurs spectraux et *Python*

```python
#!/usr/bin/env python

# Auteur : Nicolas BUR
# Date : 2023/03/15
# Version : 1.0

"""
Présentation de quelques commandes sympy
pour la construction des projecteurs spectraux
d'une matrice diagonalisable
"""

import sympy as sp # import de la librairie

sp.var("x") # déclaration de la variable symbolique x

def polynôme_matrice(p, M):
    # fonction qui évalue un polynôme en une matrice
    résultat = sp.zeros(*M.shape) # initialisation par la matrice nulle
```

```
    for _i, _ai in enumerate(p.all_coeffs()[::-1]): # boucle sur les
     ↪ coefficients de p
        résultat += _ai * (M**_i)
    return résultat

A=sp.Matrix([[1, 1, 0, 0], # définition d'une matrice
             [0, 2, 0, 0],
             [0, 0, 2, 1],
             [0, 0, 0, 1]])
print(f"soit la matrice A :\n{sp.pretty(A)}")

chi_A = A.charpoly(x) # construction du polynôme caractéristique de variable x
print(f"Le polynôme caractéristique de
 ↪ A s'écrit\n{sp.pretty(sp.factor(chi_A.as_expr()))}")
_coef, Qi = chi_A.as_poly().factor_list() # décomposition en polynômes
 ↪ irréductibles + coefficient multiplicateur
# construction du polynôme scindé à racines simples à partir de χA
m_A = _coef # initialisation
for _qi in Qi:
    m_A *= _qi[0] # multiplication par les polynômes irréductibles
assert polynôme_matrice(m_A, A).is_zero_matrix, f"Le polynôme {m_A} n'annule pas
 ↪ la matrice.\nDonc m_A n'est pas scindé à racines simples\net A n'est pas
 ↪ diagonalisable"
print(f"Le polynôme minimal de
 ↪ A s'écrit\n{sp.pretty(sp.factor(m_A.as_expr()))}")

val_p = sp.roots(m_A) # calcul des valeurs propres
print("Les valeurs propres de A sont")
[print(f"{sp.pretty(_vp)}") for _vp in val_p]

# construction des projecteurs spectraux
Pi = [ (_vp, polynôme_matrice(sp.quo(m_A, x-_vp)*sp.gcdex(x-_vp, sp.quo(m_A,
 ↪ x-_vp))[1], A)) for _vp in val_p]
[print(f"Le projecteur associé à {_Pi[0]} est\n{sp.pretty(_Pi[1])}") for _Pi in
 ↪ Pi]

# utilisation pour le calcul de la puissance de A
sp.var("n", integer=True, positive=True)
print(f"La puisance n-ième de A vaut :\n{sp.pretty(sum([_Pi[0]**n*_Pi[1] for _Pi
 ↪ in Pi], sp.zeros(*A.shape)))}")
```

Le code A.4 est donné dans le fichier *Projecteurs_spectraux.py* inclus au présent document.

5 Méthode de la tangente

Code A.5 : méthode de Newton et *Python*

```
#!/usr/bin/env python

# Auteur : Nicolas BUR
# Date : 2023/03/15
# Version : 1.0

"""
Traduction en python de la méthode de Newton
"""
```

```
10
11 import sympy as sp # librairie de calcul symbolique
12 import numpy as np # librairie de calcul numérique
13 import matplotlib.pyplot as plt # librairie graphique
14 import locale
15
16 # définit une locale spécifique (pour l'affichage correct des nombres) :
17 locale.setlocale(locale.LC_NUMERIC, "fr_FR.UTF-8")
18 # Différentes options pour les graphes
19 plt.rcParams['axes.formatter.use_locale'] = True
20 plt.rc('text', usetex=True)
21 plt.rc('font', weight='normal', size=10.0, family='serif')
22 plt.rc('legend', fontsize=10.0)
23
24 sp.var("x") # déclaration de la variable symbolique x
25
26 def Newton(f, a, b, epsilon1=1e-6, epsilon2=1e-6, kmax=500, x0=None, sm=0,
    ↪ pdf=None, nom=None, tracé=False, sortie_tex=False):
27     """
28     Cette méthode détermine la racine d'une fonction sur un intervalle donné par
        ↪ la méthode de Newton.
29
30     :param f: la fonction dont on cherche la racine
31     :type f: expression sympy
32     :param a: borne inférieure pour la recherche de la solution (et le tracé du
        ↪ graphe)
33     :type a: float
34     :param b: borne supérieure pour la recherche de la solution (et le tracé du
        ↪ graphe)
35     :type b: float
36     :param epsilon1: précision sur le critère d'arrêt $|(x_{n+1}-x_n)/x_n|$
37     :type epsilon1: float
38     :param epsilon2: précision sur le critère d'arrêt $|f(x)|$
39     :type epsilon2: float
40     :param x0: itéré initial
41     :type x0: None ou float. Si None (défaut), x0 est pris automatiquement en
        ↪ a ou en b selon le signe des dérivées
42     :param sm: second membre pour résoudre f(x)=sm
43     :type sm: expression sympy
44     :param pdf: le nom du fichier où sera enregistré le graphe (sans extension).
        ↪ Si None, pas de sauvegarde
45     :type pdf: None ou str
46     :param nom: nom de la fonction étudiée, pour affichage dans la légende du
        ↪ graphe.
47     :type nom: None ou str
48     :param tracé: pour afficher le graphe
49     :type tracé: bool
50
51     :returns: x la racine de f(x) = sm
52     """
53     _a = min(a, b)
54     _b = max(a, b)
55     I=sp.Interval(_a, _b)
56
57     g = f - sm # fonction intermédiaire permettant de résoudre g(x) = 0 au lieu
        ↪ de f(x) = sm
58
```

```
59      g_der = g.diff() # calcul de la dérivée de g
60      g_der2 = g_der.diff() # calcul de la dérivée seconde de g
61
62      ## vérification des conditions
63      # signe de g'
64      _g_der_min, _g_der_max = sp.minimum(g_der, x, I), sp.maximum(g_der, x, I)
65      assert _g_der_min * _g_der_max > 0, f"La dérivée n'est pas de signe constant
        ↪ ou s'annule"
66      signe_g_der = sp.sign(_g_der_max)
67      # signe de g''
68      _g_der2_min, _g_der2_max = sp.minimum(g_der2, x, I), sp.maximum(g_der2, x,
        ↪ I)
69      assert _g_der2_min * _g_der_max >= 0, f"La dérivée seconde n'est pas de
        ↪ signe constant"
70      signe_g_der2 = sp.sign(_g_der2_max)
71
72      assert g.subs(x, _a) * g.subs(x, _b) < 0, f"Il faut que la fonction s'annule
        ↪ entre {_a} et {_b}"
73
74      ## initialisation
75      if not x0:
76          if signe_g_der * signe_g_der2 < 0:
77              x0 = _a
78          else:
79              x0 = _b
80      gx0 = g.subs(x, x0)
81      if sp.Abs(gx0) <= epsilon2: # vérifie si le point initial n'est pas solution
82          return [x0], [gx0], f"x0 = {x0:e{epsilon1}} convient"
83
84      liste_x0 = [x0] # liste dans laquelle sont stockées les abscisses
85      liste_gx0 = [gx0] # liste dans laquelle sont stockées les ordonnées
86
87      critère = f"Nombre maximal d'itérations {kmax} atteint -> arrêt"
88      for i in range(1, kmax+1): # boucle sur le nombre maximal d'itérations
89          delta = - gx0 / g_der.subs(x, x0) # différence entre deux abscisses
            ↪ successives
90          if sp.Abs(delta / x0) <= epsilon1:
91              critère = f"L'écart relatif entre deux points successif est
                ↪ inférieur au critère {epsilon1:g} -> arrêt"
92              print(i, x0+delta)
93              break
94          if sp.Abs(gx0) <= epsilon2:
95              critère = f"Ce point est solution à {epsilon2:g} près -> arrêt"
96              print(i, x0+delta)
97              break
98          x0 += delta # nouvelle abscisse
99          if x0 in liste_x0:
100             critère = "Ce point est déjà dans la liste -> arrêt"
101             break
102
103         liste_x0.append(x0) # ajout de l'abscisse
104         gx0 = g.subs(x, x0) # nouvelle abscisse
105         if not sortie_tex:
106             print(f"\t{i} :\t{x0:g}\t{gx0:g}")
107         liste_gx0.append(gx0) # ajout de l'ordonnée
108     if not sortie_tex:
109         print(critère)
```

```
110     n_it = i
111     if n_it < kmax:
112         n_it -= 1
113     if not sortie_tex:
114         print(f"Solution trouvée en {n_it} itérations : x = {x0:g}") # affichage
            ↪ de la solution et du nombre d'itérations utilisées
115 
116     if pdf or tracé:
117         domX = np.linspace(a, b, 500, endpoint=True) # tableaux d'abscisses pour
            ↪ le tracé de la fonction
118         plt.close("all") # fermeture des graphes éventuellement ouverts
119         fig = plt.figure() # nouvelle figure
120         if isinstance(sm, sp.Expr):
121             plt.plot(domX, sp.lambdify(x, sm)(domX), "-k", linewidth=.5,
                ↪ label=f"\ensuremath{{{sp.latex(sm)}}}") # tracé de sm
122         else:
123             plt.plot(domX, sm*np.ones((len(domX),)), "-k", linewidth=.5,
                ↪ label=f"\ensuremath{{{sp.latex(sm)}}}") # tracé de sm
124 
125         if nom:
126             f_label = f"\ensuremath{{{nom}}}"
127         else:
128             f_label = f"\ensuremath{{f(x)={sp.latex(f)}}}"
129 
130         plt.plot(domX, sp.lambdify(x, f)(domX), "-b", label=f_label) # tracé de
            ↪ la fonction
131         plt.plot(np.array(liste_x0[:-1]), np.array(liste_gx0[:-1])+sm, "ro",
            ↪ label=r"points de la m\'ethode de Newton") # tracé des points
            ↪ calculés (sauf le dernier)
132         plt.plot(liste_x0[-1], liste_gx0[-1]+sm, "ks", label="solution") # mise
            ↪ en exergue de la solution : le dernier point obtenu
133         plt.legend() # ajoute la légende
134         plt.title(r"Graphe de $f$ et des points calcul\'es") # ajoute le titre
135 
136         if pdf:
137             fig.savefig(pdf+'.pdf', bbox_inches='tight')
138         if tracé:
139             plt.show() # affiche le graphe
140     return liste_x0, liste_gx0, critère # retourne la solution
141 
142 
143 ##########################
144 if __name__ == '__main__':
145     liste_x, liste_y, critère = Newton(x**3-2, 1, 2, x0=3/2, kmax=20,
        ↪ epsilon1=1e-8, epsilon2=1e-15, tracé=True, sortie_tex=False)
146     print(f"{sp.N(liste_x[-1]):.9f}")
147     print(f"{sp.N(sp.sqrt(2)):.9f}")
148     print(f"{liste_x[-1]-sp.N(sp.sqrt(2)):g}")
```

Le code A.5 est donné dans le fichier *Newton.py* inclus au présent document.

6 Décomposition de Jordan-Chevalley

Code A.6 : décomposition de Jordan-Chevalley et *Python*

```python
#!/usr/bin/env python

# Auteur : Nicolas BUR
# Date : 2023/03/15
# Version : 1.0

"""
Présentation de quelques commandes sympy
pour la décomposition de Jordan-Chevalley
"""

import sympy as sp # import de la librairie

sp.var("x") # déclaration de la variable symbolique x

def polynôme_matrice(p, M):
    # fonction qui évalue un polynôme en une matrice
    résultat = sp.zeros(*M.shape) # initialisation par la matrice nulle
    for _i, _ai in enumerate(p.all_coeffs()[::-1]): # boucle sur les
     ↪ coefficients de p
        résultat += _ai * (M**_i)
    return résultat

A = sp.Matrix([
    [1, 0, 1, 1],
    [0, 2, 1, 0],
    [0, 0, 2, 0],
    [0, 0, 0, 1]])

print("** Par la méthode de Newton **")
chi_A = A.charpoly(x) # construction du polynôme caractéristique de variable x
print(f"Le polynôme caractéristique de la matrice
 ↪ s'écrit\n{sp.pretty(sp.factor(chi_A.as_expr()))}")
mu_max = max([_vp for _vp in A.eigenvals()])
m = sp.ceiling(sp.log(mu_max)/sp.log(2))
print(f"Comme la plus grande multiplicité des valeurs propres est {2},\nil
 ↪ faudra {m} itération(s) pour obtenir D")
p_A = sp.simplify(chi_A / sp.gcd(chi_A, chi_A.diff())).as_poly()
p_A_der = p_A.diff()
print(f"Le polynôme utilisé dans la méthode de Newton
 ↪ s'écrit\n{sp.pretty(sp.factor(p_A.as_expr()))}")
print(f"dont la dérivée est\n{sp.pretty(sp.factor(p_A_der.as_expr()))}")
D = A # initialisation
D = D-polynôme_matrice(p_A, D)*(polynôme_matrice(p_A_der, D)).inv()
N = A - D
print(f"La partie diagonalisable de la matrice s'écrit\n{sp.pretty(D)}")
print(f"La partie nilpotente de la matrice s'écrit alors\n{sp.pretty(N)}")

print("\n** Par le théorème chinois **")
chi_A = A.charpoly(x) # construction du polynôme caractéristique de variable x
print(f"Le polynôme caractéristique de la matrice
 ↪ s'écrit\n{sp.pretty(sp.factor(chi_A.as_expr()))}")
val_p_mult = A.eigenvals()
val_p = list(val_p_mult.keys())
```

```
print(f"Les couples (valeur propre, multiplicité)
    ↪ sont :\n{f'{chr(10)}'.join([str(_vp) for _vp in val_p_mult.items()])}")
assert len(val_p) == 2, "Il n'y a pas 2 valeurs propres ici"
facteurs = [(x-_vp)**_mu for _vp, _mu in val_p_mult.items()]
u, v, pgdc = sp.gcdex(*facteurs)
p = val_p[0] + (val_p[1]-val_p[0]) * u * facteurs[0]
print(f"Un polynôme solution du système de congruences est\n{sp.pretty(p)}")
D = polynôme_matrice(p.as_poly(), A)
N = A-D
print(f"La partie diagonalisable de la matrice s'écrit alors
    ↪ p(A) :\n{sp.pretty(D)}")
print(f"La partie nilpotente de la matrice s'écrit alors\n{sp.pretty(N)}")
```

Le code A.6 est donné dans le fichier *Jordan_Chevalley.py* inclus au présent document.

7 Interpolations

Code A.7 : interpolations et *Python*

```
#!/usr/bin/env python

# Auteur : Nicolas BUR
# Date : 2023/03/17
# Version : 2.0

"""
Présentation de quelques commandes sympy
pour la construction des polynômes d'interpolation
"""

import sympy as sp # librairie de calcul formel
import numpy as np # librairie de calcul numérique
import matplotlib.pyplot as plt # librairie graphique
import locale

# définit une locale spécifique (pour l'affichage correct des nombres) :
locale.setlocale(locale.LC_NUMERIC, "fr_FR.UTF-8")
# Différentes options pour les graphes
plt.rcParams['axes.formatter.use_locale'] = True
plt.rc('text', usetex=True)
plt.rc('font', weight='normal', size=10.0, family='serif')
plt.rc('legend', fontsize=10.0)

sp.var("x") # déclaration de la variable symbolique x

def interpolation_linéaire(points):
    """
    Cette méthode code l'interpolation linéaire à partir des points donnés

    :param points: les points par lesquels passe la courbe
    :type points: liste ou tuple de paires de 2 éléments : [($x_0$, $y_0$), …]
    """
    nb_points = len(points) # nombre de points donnés
    _points = sorted(points) # tri des points (selon l'abscisse) pour garantir
    ↪ la croissance de la suite ($x_i$)
```

```
36      assert len(_points) == len(set([_pt[0] for _pt in _points])) # vérifie
        ↪ l'unicité des x_i
37      l_i = [] # initialisation
38      for i in range(nb_points-1):
39          # ajout dans la liste des morceaux de polynôme
40          l_i.append((_points[i][1] + (_points[i+1][1] - _points[i][1]) /
            ↪ (_points[i+1][0] - _points[i][0]) * (x - _points[i][0]),
            ↪ ((x>=_points[i][0]) & (x<=_points[i+1][0]))))
41      return sp.Piecewise(*l_i) # construction de la fonction par morceaux
42
43  def interpolation_Lagrange(points):
44      """
45      Cette méthode code l'interpolation de Lagrange à partir des points donnés
46
47      :param points: les points par lesquels passe la courbe
48      :type points: liste ou tuple de paires de 2 éléments : [($x_0$, $y_0$), …]
49      """
50      nb_points = len(points) # nombre de points donnés
51      _points = sorted(points) # tri des points (selon l'abscisse) pour garantir
        ↪ la croissance de la suite (x_i)
52      assert len(_points) == len(set([_pt[0] for _pt in _points])) # vérifie
        ↪ l'unicité des x_i
53      l = 0 # initialisation
54      for i in range(nb_points):
55          num_i = 1 # numérateur
56          den_i = 1 # dénominateur
57          for j in range(nb_points):
58              if i == j:
59                  continue
60              else:
61                  num_i *= (x - _points[j][0])
62                  den_i *= (_points[i][0] - _points[j][0])
63          l += _points[i][1] * num_i / den_i
64      return sp.simplify(l)
65
66  def interpolation_Hermite(points):
67      """
68      Cette méthode code l'interpolation d'Hermite à partir des points donnés
69
70      :param points: les points par lesquels passe la courbe, avec la valeur de la
        ↪ dérivée en ces points
71      :type points: liste ou tuple de paires de 3 éléments : [($x_0$, $y_0$,
        ↪ $d_0$), …]
72      """
73      nb_points = len(points) # nombre de points donnés
74      _points = sorted(points) # tri des points (selon l'abscisse) pour garantir
        ↪ la croissance de la suite (x_i)
75      assert len(_points) == len(set([_pt[0] for _pt in _points])) # vérifie
        ↪ l'unicité des x_i
76      h = 0 # initialisation
77      for i in range(nb_points):
78          qi = 1
79          for j in range(nb_points):
80              if i == j:
81                  continue
82              else:
83                  qi *= (x - _points[j][0])**2
```

```
84          h += qi / qi.subs(x, _points[i][0]) *
             ↪  ((1-(x-_points[i][0])*qi.diff().subs(x, _points[i][0]) / qi.subs(x,
             ↪  _points[i][0])) * _points[i][1] + (x-_points[i][0])*_points[i][2])
85      return sp.simplify(h)
86
87  def barycentre(points):
88      """
89      Cette méthode code le calcul du barycentre d'une famille de points associés
         ↪  à leur pondération
90
91      :param points: les points dont on cherche le barycentre
92      :type points: liste ou tuple d'éléments : [($x_0$, $y_0$,… , $m_0$), …]
93      """
94      nb_points = len(points)
95      dimension = len(points[0]) - 1 # la dernière coordonnée correspond à la
         ↪  pondération
96
97      _G = sp.symbols(f"_g:{dimension}")
98      return sp.linsolve([sum([points[i][-1] * (points[i][c]-_G[c]) for i in
         ↪  range(nb_points)]) for c in range(dimension)], _G).args[0]
99
100 def polynôme_Bernstein(n, k):
101     """
102     Cette méthode retourne le polynôme de Bernstein $B_{n, k}$
103     """
104     assert (k<=n), "Il faut k <= n. On a ici k={}>n={}".format(k, n)
105     return sp.binomial(n, k)*x**k*(1-x)**(n-k)
106
107 def courbe_Bézier(points):
108     """
109     Cette méthode retourne l'équation de la courbe de Bézier associée aux points
         ↪  donnés
110
111     :param points: les points de contrôle de la courbe
112     :type points: liste ou tuple de paires de 2 éléments : [($x_0$, $y_0$), …]
113     """
114     return barycentre([(*_p, polynôme_Bernstein(len(points)-1, i)) for i, _p in
         ↪  enumerate(points)])
115
116 def courbe_deCasteljau(points):
117     nb_points = len(points)
118     Q = [points]
119     for m in range(nb_points-1):
120         q = []
121         for j in range(nb_points-m-1):
122             q.append(((1-x)*Q[-1][j][0]+x*Q[-1][j+1][0],
                 ↪  (1-x)*Q[-1][j][1]+x*Q[-1][j+1][1]))
123         Q.append(q)
124     return Q
125
126 ############################
127 if __name__ == '__main__':
128
129
130     points1 = [(0, 3), (1, -2), (2, 5), (3, 7), (4, 1)] # ici ($x_i$, $y_i$)
131     points1_Hermite = [(0, 3, 0), (1, -2, 0), (2, 5, -1), (3, 7, 2), (4, 1, 0)]
         ↪  # ici ($x_i$, $y_i$, $d_i$)
```

```
    x1, y1 = [_p[0] for _p in points1], [_p[1] for _p in points1] # séparation des abscisses et des ordonnés pour le graphe

    f = sp.sin(x)+sp.sin(2*x) # la fonction de référence pour l'exemple 2 en symbolique
    df = f.diff() # sa dérivée
    x2 = [sp.pi*sp.Rational(1+i/8).limit_denominator(1e6) for i in range(9)] # les abscisses de l'exemple 2
    y2, y2_prime = [f.subs(x, _x) for _x in x2], [df.subs(x, _x) for _x in x2] # les y_i et d_i de l'exemple 2
    points2 = list(zip(x2, y2)) # groupement sous la forme (x_i, y_i)
    points2_Hermite = list(zip(x2, y2, y2_prime)) # groupement sous la forme (x_i, y_i, d_i)

    #référence = sp.plot((f, (x, min(x2), max(x2))), show=False) # nuage de points pour la fonction de référence

    ## Interpolation linéaire ##
    I_lin_1 = interpolation_linéaire(points1)
    sp.pprint(sp.nsimplify(I_lin_1, rational=True))
    plt.figure()
    plt.plot(x1, y1, "ro", label=r"Points d'interpolation") # tracé des points d'interpolation
    courbe = sp.plot((I_lin_1, (x, min(x1), max(x1))), show=False)
    plt.plot(*courbe[0].get_points(), "-b", label="Interpolation linéaire")
    plt.legend() # ajoute la légende
    plt.title(f"Interpolation linéaire - Exemple 1 ({len(points1)} points)") # ajoute le titre
    plt.show()
    print(sp.latex(I_lin_1).replace("for", "si").replace("\wedge", "\\text{ et }"))

    #I_lin_2 = interpolation_linaire(points2)#fonction d'interpolation linaire
    #sp.pprint(I_lin_2)
    #plt.figure()
    #plt.plot(x2, y2, "ro", label=r"Points d'interpolation") # tracé des points d'interpolation
    #courbe = sp.plot((I_lin_2, (x, min(x2), max(x2))), show = False)
    #plt.plot(*courbe[0].get_points(), "-b", label = "Interpolation linaire")
    #plt.plot(*référence[0].get_points(), "-.k", label = "Fonction de référence")
    #plt.legend() # ajoute la légende
    #plt.title(f"Interpolation linéaire - Exemple 2 (len(points2) points)") # ajoute le titre
    #plt.show()

    ### Interpolation de Lagrange ##
    #I_lag_1 = interpolation_Lagrange(points1)#polynme d'interpolation de Lagrange
    #sp.pprint(I_lag_1)
    #plt.figure()
    #plt.plot(x1, y1, "ro", label=r"Points d'interpolation") # tracé des points d'interpolation
    #courbe = sp.plot((I_lag_1, (x, min(x1), max(x1))), show = False)
    #plt.plot(*courbe[0].get_points(), "-b", label =
        "Interpolation de Lagrange")#tracé de la fonction
    #plt.plot([min(x1), max(x1)], np.poly1d(np.polyfit(x1, y1, 1))([min(x1), max(x1)]), "-g", label=r"Droite des moindres carrés")
    #plt.legend() # ajoute la légende
```

```
    #plt.title(f"Interpolation de Lagrange - Exemple 1 (len(points1) points)") #
     ↪ ajoute le titre
    #plt.show()

    #I_lag_2 = interpolation_Lagrange(points2)#polynmed'interpolationdeLagrange
    #sp.pprint(I_lag_2)
    #plt.figure()
    #plt.plot(x2, y2, "ro", label=r"Points d'interpolation") # tracé des points
     ↪ d'interpolation
    #courbe = sp.plot((I_lag_2, (x, min(x2), max(x2))), show = False)
    #plt.plot(*courbe[0].get_points(), "-b", label =
        "InterpolationdeLagrange")#tracdelafonction
    #plt.plot(*référence[0].get_points(), "-.k", label = "Fonctionderfrence")
    #plt.plot([min(x2), max(x2)],
        np.poly1d(np.polyfit([float(_x2)for_x2inx2], [float(_y2)for_y2iny2], 1))([min(x2), max(x2)]), "-
        g", label = r"Droitedesmoindrescarrs")
    #plt.legend() # ajoute la légende
    #plt.title(f"Interpolation de Lagrange - Exemple 2 (len(points2) points)") #
     ↪ ajoute le titre
    #plt.show()

    ### Interpolation d'Hermite ##
    #I_herm_1 = interpolation_Hermite(points1_Hermite)
    #sp.pprint(I_herm_1)
    #plt.figure()
    #plt.plot(x1, y1, "ro", label=r"Points d'interpolation") # tracé des points
     ↪ d'interpolation
    #courbe = sp.plot((I_herm_1, (x, min(x1), max(x1))), show = False)
    #plt.plot(*courbe[0].get_points(), "-b", label =
        "InterpolationdeHermite")#tracdelafonction
    #plt.legend() # ajoute la légende
    #plt.title(f"Interpolation de Hermite - Exemple 1 (len(points1) points)") #
     ↪ ajoute le titre
    #plt.show()

    #I_herm_2 = interpolation_Hermite(points2_Hermite)#fonctiond'interpolationdeHermite
    #sp.pprint(I_herm_2)
    #plt.figure()
    #plt.plot(x2, y2, "ro", label=r"Points d'interpolation") # tracé des points
     ↪ d'interpolation
    #courbe = sp.plot((I_herm_2, (x, min(x2), max(x2))), show = False)
    #plt.plot(*courbe[0].get_points(), "-b", label =
        "InterpolationdeHermite")#tracdelafonction
    #plt.plot(*référence[0].get_points(), "-.k", label = "Fonctionderfrence")
    #plt.legend() # ajoute la légende
    #plt.title(f"Interpolation de Hermite - Exemple 2 (len(points2) points)") #
     ↪ ajoute le titre
    #plt.show()

    ## Courbe de Bézier
    #Bézier1 = courbe_Bzier(points1)
    #sp.pprint(Bézier1)
    #courbes = sp.plot(*[(_bzier, (x, 0, 1))for_bzierinBzier1], show = False, adaptive =
        False, nb_of_points = 200)#constructiondesnuagesdepoints
    #plt.figure()
    #plt.plot(x1, y1, "ro", label=r"Points de contrôle") # tracé des points de
     ↪ contrôle
```

```
    #plt.plot(courbes[0].get_points()[1], courbes[1].get_points()[1], "-b", label =
        r"Courbe de Bzier")
    #plt.legend() # ajoute la légende
    #plt.title(f"Courbe de Bézier - Exemple 1 (len(points1) points)") # ajoute
    ↪ le titre
    #plt.show()

    #Bézier2 = courbe_Bzier(points2)
    #sp.pprint(Bézier2)
    #courbes = sp.plot(*[(_bzier, (x, 0, 1)) for _bzier in Bzier2], show = False, adaptive =
        False, nb_of_points = 200)#construction des nuages de points
    #plt.figure()
    #plt.plot(x2, y2, "ro", label=r"Points de contrôle") # tracé des points de
    ↪ contrôle
    #plt.plot(courbes[0].get_points()[1], courbes[1].get_points()[1], "-b", label =
        r"Courbe de Bzier")
    #plt.plot(*référence[0].get_points(), "-.k", label = "Fonction derfrence")
    #plt.legend() # ajoute la légende
    #plt.title(f"Courbe de Bézier - Exemple 2 (len(points2) points)") # ajoute
    ↪ le titre
    #plt.show()

    ## Coube de de Casteljau
    #deCasteljau1 = courbe_de Casteljau(points1)
    #print("Équation de la première coordonnée")
    #sp.pprint(sp.simplify(sp.expand(deCasteljau1[-1][0][0])))
    #print("Équation de la deuxième coordonnée")
    #sp.pprint(sp.simplify(sp.expand(deCasteljau1[-1][0][1])))
    #plt.figure()
    #t = sp.Rational(2/3).limit_denominator(1e6)
    #for i, étape in enumerate(deCasteljau1):
        #if i == 0:
            #plt.plot([_p[0] for _p in tape], [_p[1] for _p in tape], marker = "o", label =
                "tape 0")#tracé des points de contrle
        #else:
            #plt.plot([_p[0].subs(x, t) for _p in tape], [_p[1].subs(x, t) for _p in tape], marker =
                "D", label = f"tape i")#tracé des points de contrle
    #courbes =
        sp.plot(*[(_decasteljau, (x, 0, 1)) for _decasteljau in deCasteljau1[-1][0]], show =
        False, adaptive = False, nb_of_points = 200)#construction des nuages de points
    #plt.plot(courbes[0].get_points()[1], courbes[1].get_points()[1], "-b", label =
        f"Courbe de de Casteljau")
    #plt.legend() # ajoute la légende
    #plt.title(f"Algorithme de de Casteljau pour t = sp.latex(t) - Exemple 1
    ↪ (len(points1) points)") # ajoute le titre
    #plt.show()

    #deCasteljau2 = courbe_de Casteljau(points2)
    #print("Équation de la première coordonnée")
    #sp.pprint(sp.simplify(sp.expand(deCasteljau2[-1][0][0])))
    #print("Équation de la deuxième coordonnée")
    #sp.pprint(sp.simplify(sp.expand(deCasteljau2[-1][0][1])))
    #plt.figure()
    #t = sp.Rational(2/3).limit_denominator(1e6)
    #for i, étape in enumerate(deCasteljau2):
        #if i == 0:
```

```
262             #plt.plot([_p[0] for _p in tape], [_p[1] for _p in tape], marker = "o", label =
                    "tape0")#tracdespointsdecontrle
263         #else:
264             #plt.plot([_p[0].subs(x, t) for _p in tape], [_p[1].subs(x, t) for _p in tape], marker =
                    "D", label = f"tape i")#tracdespointsdecontrle
265     #courbes =
            sp.plot(*[(_decasteljau, (x, 0, 1)) for _decasteljau in deCasteljau2[-1][0]], show =
            False, adaptive = False, nb_of_points = 200)#constructiondesnuagesdepoints
266     #plt.plot(courbes[0].get_points()[1], courbes[1].get_points()[1], " - b", label =
            f"Courbe de de Casteljau")
267     #plt.legend() # ajoute la légende
268     #plt.title(f"Algorithme de de Casteljau pour t = sp.latex(t) - Exemple 2
        ↪  (len(points2) points)") # ajoute le titre
269     #plt.show()
```

Le code A.7 est donné dans le fichier *Interpolations.py* inclus au présent document.

Index

Bibliographie

Ben Rhouma, Alaeddine (2013), « Autour de la décomposition de Dunford réelle ou complexe. Théorie spectrale et méthodes effectives », https://hal.science/hal-00844141 (cf. p. 46).

Table des matières