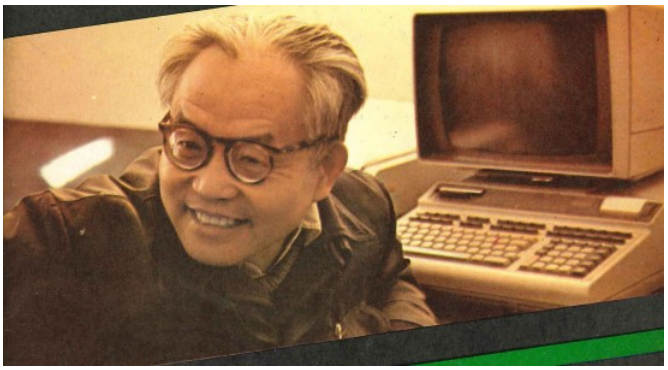
UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

UFR des sciences

Mémoire de stage
Master Algèbre Appliquée

Étude et implantation d'une méthode algébrique pour résoudre des systèmes à coefficients flous



Stage financé par le laboratoire LIP6 de l'UPMC
et effectué au sein de son équipe APR

Jérémy Marrez

Encadrants : Annick Valibouze
Philippe Aubry

Année universitaire 2015/2016

Table des matières

Introduction

Dans de nombreux domaines où l'information est incomplète ou imprécise, la théorie des nombres flous est utilisée pour modéliser des réalités incertaines. Résoudre des systèmes polynomiaux à coefficients flous est l'un des enjeux majeurs dans le champs de la modélisation incertaine car il s'étend à un large spectre d'applications en sciences, comme l'économie, la médecine et l'ingénierie.

Jusque là, deux catégories différentes de méthodes ont été développées pour résoudre des systèmes de polynômes à coefficients flous, l'une reposant sur des calculs approximés, et l'autre sur des calculs exactes. Dans la première catégorie, on retrouve la méthode de Newton et ses extensions comme les réseaux de neurones et d'autres méthodes itératives [1, 2, 5]. Cependant, les résultats de ces approches numériques sont difficiles à évaluer.

Pour faire face à ces problèmes, une autre catégorie de méthodes basées sur du calcul formel a été développée récemment. Contrairement à l'approche numérique, ces techniques algébriques produisent un résultat exacte. Nous nous intéressons ici à une nouvelle approche [4] pour résoudre des systèmes de polynômes du type :

$$AX + B = CX + D \tag{1}$$

où X est un vecteur de variables réelles, et A, B, C, D sont des matrices floues, avec la particularité qu'un nombre flou en général n'a pas d'inverse pour l'opération d'addition.

L'idée principale de cette approche est dans un premier temps de convertir le système (1) en un système paramétrique, c'est-à-dire en faisant intervenir la représentation paramétrique des nombres flous. Cette représentation a été étudiée dans le cas des nombres flous dits triangulaires [12, 4] et nous l'étendons ici à celui des nombres flous dits quadratiques. Pour les nombres flous triangulaires, la conversion en système paramétrique donnera 2 fois plus d'équations et un paramètre r.

Ce nouveau système est converti en un système avec une variable de moins appelé le système tranché collecté (collected crisp system). Ensuite, en utilisant l'algorithme de décomposition triangulaire de Wu Wen Tsun sur ce système d'équations intermédiaire, la variété du système tranché collecté est calculée. Cette variété est composée de toutes les solutions exactes du système (1).

Les coefficients sont donc flous mais leur représentation est formelle. Et c'est en utilisant ces représentations formelles que l'on peut aborder dans certains cas

la forme paramétrique.

Ce document s'organise comme suit. Dans la section 1, la théorie des nombres flous est présentée. Puis, dans la section 2, les notions de bases nécessaires à l'algorithme de décomposition triangulaire de Wu Wen Tsun sont rappelées et cet algorithme de résolution algébrique de systèmes d'équations polynomiales est introduit. La section 3 se concentre sur le passage du flou à l'algébrique pour pouvoir ensuite utiliser la méthode de Wu. La procédure principale de résolution de ces systèmes d'équations est déroulée à la section 4. La section 5 présente notre programme Fuzzy où ont été implantées en SageMath basé sur Python l'arithmétique sur les nombres flous et la méthode algébrique de résolution des systèmes de polynômes à coefficients flous.

1 La théorie des nombres flous

1.1 Ensembles flous

Dans cette partie, plusieurs définitions et notations portant sur la théorie des ensembles flous sont présentées. On nomme X l'ensemble universel.

Dans la théorie des ensembles dite "classique", à tout sous-ensemble B de X, on peut associer sa *fonction caractéristique* définie comme suit :

$$\begin{array}{rccl} \mathbb{1}_B & : & X & \longrightarrow & \{0,1\} \\ & & x & \longmapsto & \begin{cases} 1 & \text{si } x \in B, \\ 0 & \text{si } x \notin B. \end{cases} \end{array}$$

qui exprime l'appartenance ou non à B de chaque élément de X. On parle aussi de *fonction indicatrice*.

On voit clairement que l'appartenance à de tels ensembles est booléenne ; c'est-à-dire qu'un élément appartient à un ensemble ou n'y appartient pas, sans autre alternative.

Dans la théorie des ensembles flous introduite ici, l'appartenance aux ensembles ne sera plus booléenne mais graduelle, et il deviendra possible d'exprimer le degré d'appartenance par une valeur réelle comprise entre 0 et 1.

Définition 1.1. Un *ensemble flou* $\widetilde{E}$ est un sous-ensemble de X défini par sa fonction caractéristique, appelée ici *fonction d'appartenance* $\mu_{\widetilde{E}}$, et définie comme suit :

$$\begin{array}{rccc} \mu_{\widetilde{E}} & : & X & \longrightarrow & [0,1] \\ & & x & \longmapsto & \mu_{\widetilde{E}}(x) \end{array}$$

Pour chaque élément x de X, $\mu_{\widetilde{E}}(x)$ est appelé le *degré d'appartenance* de x à $\widetilde{E}$ et représente le degré de validité de la proposition "x appartient à $\widetilde{E}$". Plus ce degré est proche de 1, plus x appartient à $\widetilde{E}$ de façon certaine. Si $\mu_{\widetilde{E}}(x) = 0$ alors x n'est pas du tout élément de $\widetilde{E}$, et si $\mu_{\widetilde{E}}(x) = 1$ alors x appartient totalement à $\widetilde{E}$.

Note. Le degré d'appartenance est donc un degré de vérité, ce qui le différencie d'une probabilité qui reflète le pourcentage de chance qu'un événement se produise ; on parle alors d'éventualité.
En effet, pour un entier naturel x donné, si on dit que "x est un grand nombre" avec un degré de vérité de $0,9$, on est certain que x sera un nombre plutôt grand. En revanche, si on dit que "x est un grand nombre" avec une probabilité de $0,9$, il y a une probabilité de 10% que x soit petit ou nul.

Exemple 1.1. On considère le poids d'un panier qu'une personne a rempli lors d'une récolte de pommes. On définit l'ensemble flou $\widetilde{E}$ représentant l'ensemble des poids satisfaisants pour qu'un panier soit considéré comme rempli.
Sur la figure ci-dessous sont représentés deux ensembles : l'ensemble classique E avec sa fonction indicatrice, et l'ensemble flou $\widetilde{E}$ avec sa fonction d'appartenance. La courbe associée à $\widetilde{E}$ montre l'évolution du degré d'appartenance : plus le poids augmente, plus il est considérée comme satisfaisant. En revanche, la courbe associée à E illustre une fonction d'appartenance qui caractérise un ensemble "classique" équivalent, impliquant un seuil strict (ici 3 kilos).

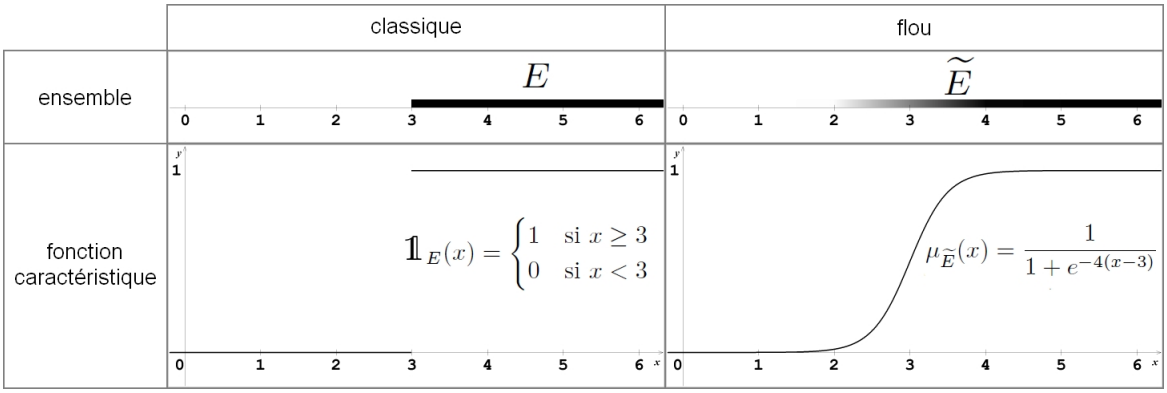


FIGURE 1 – Comparaison d'un ensemble classique et d'un ensemble flou équivalent

Cet exemple rend plus claire la distinction entre l'imprécision d'un ensemble flou et une probabilité. Faisant appel à $\widetilde{E}$, dans l'assertion "Il est certain que le poids de ce panier est à peu près satisfaisant", on constate que l'on a une imprécision (sur le poids exact du panier), autour d'une certitude (sur le fait que le poids de ce panier soit à peu près satisfaisant). Au contraire, dans l'assertion, "Le poids de ce panier est à peu près satisfaisant avec une probabilité de 10%", on retrouve une connaissance floue (toujours sur le poids exact du panier), mais ici, cette connaissance est incertaine (avec une probabilité de véracité de 10%).

Propriétés 1.1. Un sous-ensemble flou $\widetilde{E}$ considéré sur l'ensemble universel X se caractérise par les éléments suivants :

- son noyau $n(\widetilde{E})$ défini par :

$$n(\widetilde{E}) = \{x \in X \mid \mu_{\widetilde{E}}(x) = 1 \},$$

- son support $sp(\widetilde{E})$ défini par :

$$sp(\widetilde{E}) = \{x \in X \mid \mu_{\widetilde{E}}(x) > 0 \},$$

- et sa hauteur $h(\widetilde{E})$ définie par :

$$h(\widetilde{E}) = \sup_{x \in X}(\mu_{\widetilde{E}}(x)).$$

Note. La hauteur définie dans la littérature avec un max ne tient pas compte de la limite sup atteinte à l'infini. Cette pour cette raison que nous préférons la définir avec un sup.

Définition 1.2. On appelle *coupe* r de l'ensemble flou $\widetilde{E}$, que l'on note $\widetilde{E}_r$, l'ensemble défini ci-dessous :

$$\widetilde{E}_r = \{x \in X \mid \mu_{\widetilde{E}}(x) \geq r \}$$

1.2 Nombres flous

Soit $m \in X$ fixé. Un *nombre flou* $\widetilde{m}$ est un ensemble flou dont la fonction d'appartenance représente le degré de validité de la proposition "x est égal à m" pour chaque $x \in X$. Plus x s'éloigne de la valeur m, plus l'appartenance de x à l'ensemble flou correspondant diminue.
L'avantage d'un nombre flou est qu'il permet de capturer l'incertitude autour de la valeur m donnée.

On obtient donc un ensemble flou dont la figure suivante donne une représentation possible :

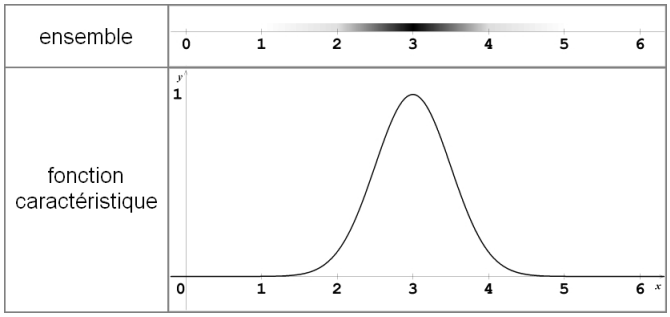


FIGURE 2 – Un exemple de nombre flou

Sur cet exemple, on voit clairement que le nombre flou $\widetilde{3}$ représente le degré de validité de la proposition "x est égal à 3" pour chaque $x \in X$.
De façon plus formelle, nous pouvons définir un nombre flou en réutilisant les propriétés des ensembles flous.

Remarque. Un ensemble flou est dit *convexe* si toutes ses coupes r le sont, $r \in [0, 1]$; i.e. si aucune coupe r n'est union d'au moins deux ensembles-disjoints.

Définition 1.3. Un *nombre flou* $\widetilde{m}$ est un ensemble flou caractérisé par les quatre propriétés suivantes :

i) il est *normal*, i.e. $h(\widetilde{m}) = 1$,

ii) son noyau est de cardinal 1 ; i.e. $\text{card}(n(\widetilde{m})) = 1$,

iii) il est convexe,

iv) sa fonction d'appartenance $\mu_{\widetilde{m}}$ est continue.

Note. L'unique élément du noyau, appelé le mode de $\widetilde{m}$ et noté m, est donc le seul à appartenir totalement à $\widetilde{m}$. C'est lui qui apparaît dans la proposition "x est égal à m" représentée par $\widetilde{m}$.

Note. Si le support d'un nombre flou est réduit à son mode, alors ce nombre est réel et sa valeur est celle de son mode.

Note. Deux nombres flous sont égaux si et seulement s'ils ont les mêmes fonctions d'appartenance.

1.3 Floutage et arithmétique floue

Le principe du floutage évoqué dans [7] consiste, à partir d'une fonction f de la forme :

$$\begin{array}{rccc} f: & \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ & (x_1,\ldots,x_n) & \longmapsto & y=f(x_1,\ldots,x_n) \end{array}$$

à induire une fonction $\widetilde{f}$ de la forme :

$$\begin{array}{rccc} \widetilde{f}: & \mathfrak{B}(\mathbb{R})^n & \longrightarrow & \mathfrak{B}(\mathbb{R}) \\ & (\widetilde{x}_1,\ldots,\widetilde{x}_n) & \longmapsto & \widetilde{y}=\widetilde{f}(\widetilde{x}_1,\ldots,\widetilde{x}_n) \end{array}$$

où $\mathfrak{B}(\mathbb{R})$ est la classe des nombres flous de $\mathbb{R}$.

Le principe du floutage mentionné par Gaines (1976), Negoita (1976) et d'autres [9], permet d'exprimer le résultat $\widetilde{y}$ grâce à sa fonction d'appartenance. Pour cela, on utilise la fonction d'appartenance du n-uplet $(\widetilde{x}_1,\ldots,\widetilde{x}_n)$, notée $\mu_{(\widetilde{x}_1,\ldots,\widetilde{x}_n)}$ définie comme suit :

$$\mu_{(\widetilde{x}_1,\ldots,\widetilde{x}_n)}(s_1,\ldots,s_n)=\min(\mu_{\widetilde{x}_1}(s_1),\ldots,\mu_{\widetilde{x}_n}(s_n))$$

pour tout $(s_1,\ldots,s_n)\in\mathbb{R}^n$; ce qui signifie que le degré d'appartenance d'un n-uplet $(s_1,\ldots,s_n)$ au produit cartésien $\widetilde{x}_1\times\cdots\times\widetilde{x}_n$ (noté ici $(\widetilde{x}_1,\ldots,\widetilde{x}_n)$) est choisi comme le minimum des degrés d'appartenance des s_i à $\widetilde{x}_i$ pour $1\leq i\leq n$.

La fonction d'appartenance $\mu_{\widetilde{y}}$ se définit alors comme suit :

$$\mu_{\widetilde{y}}(t)=\max\{\mu_{(\widetilde{x}_1,\ldots,\widetilde{x}_n)}(s_1,\ldots,s_n)\mid(s_1,\ldots,s_n)\in f^{-1}(t)\}.$$

Posons $\underline{s}=(s_1,\ldots,s_n)$. La définition suivante résume ce principe.

Définition 1.4 (Principe du floutage)**.** Soient n nombres flous $\widetilde{x}_1,\ldots,\widetilde{x}_n$ de $\mathfrak{B}(\mathbb{R})$ de fonctions d'appartenances respectives $\mu_{\widetilde{x}_1},\ldots,\mu_{\widetilde{x}_n}$. La fonction d'appartenance du nombre flou $\widetilde{y}=\widetilde{f}(\widetilde{x}_1,\ldots,\widetilde{x}_n)$ est définie comme suit

$$\forall t\in\mathbb{R},\mu_{\widetilde{y}}(t)=\max_{\underline{s}|f(\underline{s})=t}\min(\mu_{\widetilde{x}_1}(s_1),\ldots,\mu_{\widetilde{x}_n}(s_n)). \tag{2}$$

On constate que le degré d'appartenance $\mu_{\widetilde{y}}(t)$ de t à $\widetilde{y}$ est bien le plus grand des degrés de validité des obtentions de t par $\underline{s}$ à travers f ; le degré de validité d'un n-uplet $\underline{s}$ étant la plus petite valeur parmi $\mu_{\widetilde{x}_1}(s_1),\ldots,\mu_{\widetilde{x}_n}(s_n)$. Un modèle flou ne prend donc pas en compte le nombre de façons dont on peut obtenir t, seulement le fait qu'il peut être obtenu.

Ce principe pose les bases de l'arithmétique floue et nous permet d'introduire les différentes opérations sur les nombres flous. Nous définissons ci-dessous ces opérations avec deux opérandes.

Fixons $\widetilde{m}$ et $\tilde{n}$ deux nombres flous.

La somme :

La fonction d'appartenance $\mu_{\widetilde{m}\oplus\widetilde{n}}$ de la somme $\widetilde{m} \oplus \tilde{n}$ est définie comme suit :

$$\mu_{\widetilde{m}\oplus\widetilde{n}}(z) = \max_{z=x+y} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{n}}(y))$$

avec $(x, y, z) \in \mathbb{R}^3$. Cette formule peut s'exprimer également sous la forme :

$$\mu_{\widetilde{m}\oplus\widetilde{n}}(z) = \max_{x} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{n}}(z-x))$$

On constate que cette opération est une convolution.

Note. L'addition des nombres flous est associative et commutative. L'élément neutre pour $\oplus$ a pour mode 0 et son support est réduit à ce mode. Il s'agit donc du nombre réel 0.

L'opposé :

La fonction d'appartenance $\mu_{\widetilde{-m}}$ de l'opposé de $\widetilde{m}$, noté $\widetilde{-m}$ est définie comme suit :

$$\mu_{\widetilde{-m}}(z) = \max_{z=-x} \min(\mu_{\widetilde{m}}(x)) = \mu_{\widetilde{m}}(-z)$$

Il s'agit de la fonction symétrique de $\mu_{\widetilde{m}}$ par rapport à l'axe des ordonnées.

Remarque. Attention, pour un nombre flou $\widetilde{m}$ dont le support n'est pas réduit à son mode, $\widetilde{-m} \neq -\widetilde{m}$, car dans ce cas $\widetilde{m}$ n'a pas d'élément symétrique pour la loi $\oplus$.

L'élément neutre pour la loi $\oplus$ est le nombre réel 0 de fonction caractéristique :

$$\begin{array}{rccl} \mathbb{1}_0 & : & X & \longrightarrow & \{0,1\} \\ & & x & \longmapsto & \begin{cases} 1 & \text{si } x = 0, \\ 0 & \text{sinon.} \end{cases} \end{array}$$

On souhaite prouver qu'il existe un nombre flou $\widetilde{m}$ tel que $\widetilde{m} \oplus \widetilde{-m} \neq 0$, ou encore $\mu_{\widetilde{m}\oplus\widetilde{-m}} \neq \mathbb{1}_0$; i.e. il existe un $z \in \mathbb{R}, z \neq 0$ tel que $\mu_{\widetilde{m}\oplus\widetilde{-m}}(z) \neq 0$.

Choisissons le nombre flou $\widetilde{m}$ de mode 1, de support $[0,2]$ et de fonction d'appartenance $\mu_{\widetilde{m}}(x) = \max(0, 1 - |x-1|)$. Alors on a

$$\begin{aligned}
\mu_{\widetilde{m}\oplus\widetilde{-m}}(z) &= \max_{z=x+y} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{-m}}(y)) \\
&= \max_{z=x+y} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{m}}(-y)) \\
&= \max_{x} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{m}}(x-z)) \\
&= \max_{x} \min(\max(0, 1-|x-1|), \max(0, 1-|x-z-1|))
\end{aligned}$$

Pour $z = \frac{1}{2}$, on a :

$$\mu_{\widetilde{m}\oplus\widetilde{-m}}(\frac{1}{2}) = \max_{x} \min(\max(0, 1-|x-1|), \max(0, 1-|x-\frac{3}{2}|))$$

La courbe de la fonction $\max(0, 1-|x-\frac{3}{2}|$ est donc la courbe de $\mu_{\widetilde{m}}$ translaté de $\frac{1}{2}$ dans la direction des abscisses. Par la convexité des nombres flous, avant et après l'unique point d'intersection de ces courbes en $x = x_0$, $\min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{m}}(x-\frac{1}{2}))$ sera inférieur à la valeur en x_0 des deux courbes.

Le maximum des minima est donc obtenu au point d'intersection des deux fonctions qui se situe dans l'intervalle $[1, \frac{3}{2}]$.

Le valeur $x_0 = \frac{5}{4}$ est donnée par l'identité :

$$1 - (x-1) = 1 - (\frac{3}{2} - x)$$

Pour $x_0 = \frac{5}{4}$, nous avons :

$$\min(\max(0, 1-|\frac{5}{4}-1|), \max(0, 1-|\frac{5}{4}-\frac{3}{2}|)) = \min(\frac{3}{4}, \frac{3}{4}) = \frac{3}{4}$$

D'où $\mu_{\widetilde{m}\oplus\widetilde{-m}}(\frac{1}{2}) = \frac{3}{4} \neq 0$.

Par ailleurs, on remarque que l'on a bien

$$\mu_{\widetilde{m}\oplus\widetilde{-m}}(0) = \max_{x} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{m}}(x)) = 1.$$

Donc le mode de $\widetilde{m} \oplus \widetilde{-m}$ est bien 0 mais son support n'est pas réduit à ce seul élément.

L'ensemble des nombres flous est donc un semi-groupe commutatif pour la loi $\oplus$.

La soustraction :

On peut alors obtenir la fonction d'appartenance de $\widetilde{m} \ominus \widetilde{n}$ car la soustraction se définit naturellement à partir des deux opérations précédentes comme $\widetilde{m} \ominus \widetilde{n} = \widetilde{m} \oplus \widetilde{-n}$.

Remarque. Dans l'arithmétique floue, la soustraction n'est pas l'inverse de l'addition.

En effet, si on pose $\widetilde{res} = \widetilde{m} \oplus \widetilde{n}$, alors il faut bien faire la distinction entre :

- la soustraction $\widetilde{res} \ominus \widetilde{n}$ qui calcule le nombre flou obtenu par la soustraction définie ci-dessus,
- et l'opération inverse de l'addition par $\widetilde{n}$ qui calcule le nombre flou $\widetilde{j}$ à l'incertitude maximale, i.e. au support le plus large, tel que $\widetilde{j} \oplus \widetilde{n} = \widetilde{res}$.

La multiplication :

La fonction d'appartenance $\mu_{\widetilde{m} \odot \widetilde{n}}$ du produit $\widetilde{m} \odot \widetilde{n}$ est définie comme suit :

$$\mu_{\widetilde{m} \odot \widetilde{n}}(z) = \max_{z=xy} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{n}}(y))$$

L'inverse :

La fonction d'appartenance $\mu_{\widetilde{1/m}}$ de l'inverse de $\widetilde{m}$, noté $\widetilde{1/m}$ est définie comme suit :

$$\mu_{\widetilde{1/m}}(z) = \max_{z=1/x} \min(\mu_{\widetilde{m}}(x)) = \mu_{\widetilde{m}}(1/z)$$

Note. Il ne s'agit pas de l'inversion usuelle au sens du symétrique de $\widetilde{m}$ pour la loi $\odot$.

La division :

On peut alors obtenir la fonction d'appartenance de $\widetilde{m}/\widetilde{n}$ car la division se définie naturellement à partir des deux opérations précédentes comme $\widetilde{m}/\widetilde{n} = \widetilde{m} \odot \widetilde{1/n}$.

Note. Comme pour la soustraction avec l'addition, la division n'est pas l'inverse de la multiplication dans l'arithmétique floue.

Le maximum :

La fonction d'appartenance $\mu_{\widetilde{\max}(\widetilde{m},\widetilde{n})}$ du maximum de $\widetilde{m}$ et $\widetilde{n}$, noté $\widetilde{\max}(\widetilde{m}, \widetilde{n})$ est définie comme suit :

$$\mu_{\widetilde{\max}(\widetilde{m},\widetilde{n})}(z) = \max_{\max(x,y)=z} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{n}}(y))$$

Le minimum :

De la même façon, la fonction d'appartenance $\mu_{\widetilde{\min}(\widetilde{m},\widetilde{n})}$ du minimum de $\widetilde{m}$ et $\widetilde{n}$, noté $\widetilde{\min}(\widetilde{m},\widetilde{n})$ est définie comme suit :

$$\mu_{\widetilde{\min}(\widetilde{m},\widetilde{n})}(z) = \max_{\min(x,y)=z} \min(\mu_{\widetilde{m}}(x), \mu_{\widetilde{n}}(y))$$

1.4 La représentation en tuple

Afin de simplifier les calculs sur les nombres flous, différentes représentations de ces nombres ont été développées. En particulier en ce qui concerne les nombres flous dits gauche-droite, deux représentations font référence :

- la représentation en tuple,
- et la représentation paramétrique.

Nous verrons ici comment se construit la première représentation des nombres flous gauche-droite. C'est dans cette représentation que sont donnés les coefficients flous du système de polynômes que l'ou souhaite résoudre. La représentation paramétrique sera abordée plus tard à la section 3.

1.4.1 Familles gauche-droite

La représentation en tuple des nombres flous est l'une des premières à avoir été proposée par Dubois et Prade en 1977 dans [7] et repose sur les résultats donnés ci-après.

Il s'agit de représenter un nombre flou $\widetilde{n}$ à l'aide des éléments suivants :

i) son mode n.

ii) la forme générale de sa fonction d'appartenance. Elle sera donnée par les *types de ses restrictions* de chaque côté du mode n ; par exemple pour les plus usuels ; linéaire, quadratique, exponentiel ou gaussien.

iii) si son support est fini, il est représenté à l'aide de deux valeurs, α et β, appelées respectivement "left spread" (la dispersion à gauche) et "right spread" (la dispertion à droite) de $\widetilde{n}$. Elles représentent la longueur du support de $\widetilde{n}$, respectivement à gauche et à droite du mode n.

Lorsque le support est fini, on aura donc une représentation de $\widetilde{n}$ par un triplet (n, α, β) pour exprimer i) et iii).

Soit $\widetilde{n}$ un nombre flou de fonction d'appartenance $\mu_{\widetilde{n}}$. On appelle *restriction gauche* (resp. *restriction droite*) et on note $\mu_{\widetilde{n}-}$ (resp. $\mu_{\widetilde{n}+}$) la restriction de $\mu_{\widetilde{n}}$ à gauche (resp. à droite) de son mode n. La figure ci-dessous illustre ces restrictions pour des nombres flous à support fini ou non.

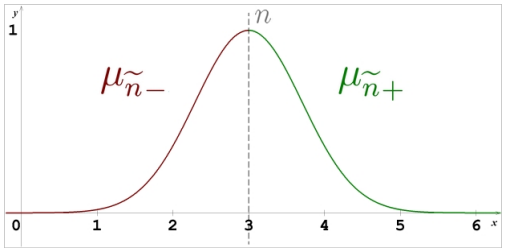


FIGURE 3 – Restrictions $\mu_{\widetilde{n}-}$ et $\mu_{\widetilde{n}+}$ de part et d'autre du mode $n = 3$ d'un nombre flou gaussien à support infini.

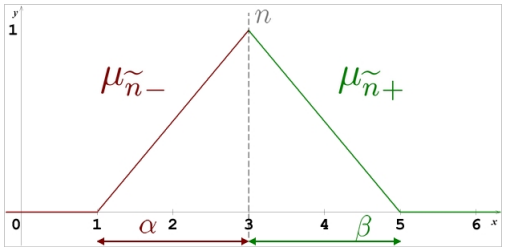


FIGURE 4 – Restrictions $\mu_{\widetilde{n}-}$ et $\mu_{\widetilde{n}+}$ de part et d'autre du mode $n = 3$ d'un nombre flou triangulaire à support fini.

Sur les figures 3 et 4, on a des cas particuliers de nombre flou. En effet, on constate que les restrictions de $\mu_{\widetilde{n}}$ à gauche et à droite du mode sont du même type, et qu'elles sont même la symétrique l'une de l'autre. Or, rien ne l'impose. En effet, les restrictions gauche et droite d'une fonction d'appartenance peuvent être du même type sans être symétriques (figure 5), ce qui impose que $\alpha \neq \beta$. Un nombre flou peut aussi avoir une fonction d'appartenance dont les types des restrictions gauche et droite diffèrent (figure 6).

Les restrictions à gauche et à droite sont usuellement des fonctions de type linéaire, gaussien, exponentiel ou encore quadratique. À partir du type de ces restrictions, on en déduit des *familles de nombres flous gauche-droite* en combinant

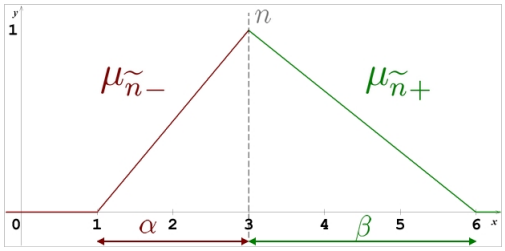


FIGURE 5 – Restrictions $\mu_{\widetilde{n}-}$ et $\mu_{\widetilde{n}+}$ de part et d'autre du mode $n = 3$ d'un nombre flou triangulaire avec $\alpha \neq \beta$.

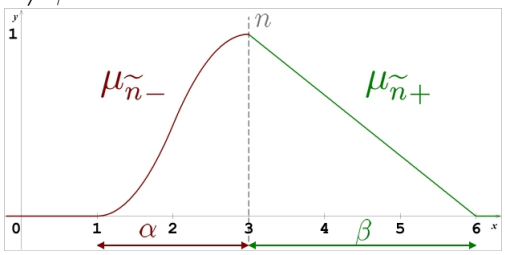


FIGURE 6 – Restrictions $\mu_{\widetilde{n}-}$ et $\mu_{\widetilde{n}+}$ de part et d'autre du mode $n = 3$ d'un nombre flou quadratique-triangulaire avec $\alpha \neq \beta$.

ces types. Voici ci-dessous quelques exemples de combinaisons :

type restriction gauche	type restriction droite	famille nombre flou
linéaire	quadratique	triangulaire - quadratique
exponentiel	quadratique	exponentiel - quadratique
gaussien	linéaire	gaussien - triangulaire

On obtient alors des familles dîtes triangulaire-quadratique, exponentielle-triangulaire, gaussienne-quadratique, ou encore simplement triangulaire pour triangulaire-triangulaire et similairement quadratique, exponentielle ou gaussienne. Ces quatre dernières familles dites *simples* sont les principales utilisées dans les applications.

Pour les nombres flous à support fini appartenant à une famille simple, l'égalité $\alpha = \beta$ implique une symétrie de la fonction d'appartenance de part et d'autre du mode.

Ainsi, dans le cas des familles simples, l'information portant sur la forme générale de la fonction d'appartenance du nombre flou est conservée en imposant que les calculs soient effectués au sein de la même famille. On considère donc que deux familles distinctes sont incompatibles entre elles.

Remarque. Soit $\mathfrak{F}$ une famille de nombres flous. Alors $(\mathfrak{F}, \oplus)$ est un semi-groupe (voir exemple page 8).

– Si $\mathfrak{F}$ est une famille simple, $\widetilde{-m}$ n'est pas le symétrique de $\widetilde{-m}$ pour la loi $\oplus$.
– Si $\mathfrak{F}$ est non simple, $\widetilde{-m}$ n'est pas une opération interne à $\mathfrak{F}$, car le résultat se trouve alors dans la famille symétrique de $\mathfrak{F}$. Par exemple l'opposé d'un nombre flou triangulaire-quadratique est un nombre flou quadratique-triangulaire et inversement.

Il en va de même pour le semi-groupe $(\mathfrak{F}, \odot)$.

Bibliothèque Fuzzy. La représentation en tuple est implantée dans la classe `NombreFlouRed`.

1.4.2 L'arithmétique sur les tuples

Dans toute la suite, on considère des nombres flous à support fini.

Les résultats suivants expliquent le passage de l'arithmétique floue aux calculs sur la représentation en tuple pour le cas de l'addition [7].

Soient $\widetilde{m}$ et $\widetilde{n}$ deux nombre flous à support fini de somme $\widetilde{m} \oplus \widetilde{n}$.

Lemme 1.2. *Soit* $\omega \in]0,1[$. *Alors* $\exists\, p, r \in \mathbb{R}$ *tels que*

$$\omega = \mu_{\widetilde{m}-}(p) = \mu_{\widetilde{n}-}(r) = \mu_{\widetilde{m}\oplus\widetilde{n}-}(z)$$

avec $z = p + r$.

Corollaire 1.1. *Soit* $\omega \in]0,1[$. *Alors* $\exists\, p, r \in \mathbb{R}$ *tels que*

$$\omega = \mu_{\widetilde{m}+}(p) = \mu_{\widetilde{n}+}(r) = \mu_{\widetilde{m}\oplus\widetilde{n}+}(z)$$

avec $z = p + r$.

Soient G et D deux fonctions d'appartenance paires représentant des nombres flous de mode nul et de support $[-1, 1]$ (voir Définition 1.3). On notre G_- la restriction gauche de G et D_+ la restriction droite de D. Supposons que $\widetilde{m}$ et $\widetilde{n}$ appartiennent à une même famille ; i.e. les restrictions $\mu_{\widetilde{m}-}$ (resp. $\mu_{\widetilde{m}+}$) et $\mu_{\widetilde{n}-}$ (resp. $\mu_{\widetilde{n}+}$) sont du même type.

Les nombres flous $\widetilde{m}$ et $\widetilde{n}$ ont des fonctions d'appartenance $\mu_{\widetilde{m}}$ (resp. $\mu_{\widetilde{n}}$) telles que :

$$\mu_{\widetilde{m}-}(x) = G_-\left(\frac{x-m}{\alpha}\right), \quad \mu_{\widetilde{m}+}(x) = D_+\left(\frac{x-m}{\beta}\right),$$

$$\mu_{\widetilde{n}-}(x) = G_-\left(\frac{x-n}{\gamma}\right), \quad \mu_{\widetilde{n}+}(x) = D_+\left(\frac{x-n}{\delta}\right),$$

avec $\alpha, \beta, \gamma, \delta > 0$.

On constate que l'on a ici des compositions de fonctions. En effet, afin d'obtenir $\mu_{\widetilde{m}-}$, la fonction $x \mapsto \frac{x-m}{\alpha}$ permet à la fois d'effectuer une translation de m dans la direction des abscisses afin que le mode nul de G_- soit bien en m pour $\mu_{\widetilde{m}-}$, et, par la division par α, d'ajuster la longueur du support de $\mu_{\widetilde{m}-}$ à gauche du mode afin de retrouver le bon "leftSpread".

Les nombres $\widetilde{m}$ et $\widetilde{n}$ sont alors décrits comme des "nombres flous $G - D$".

Remarque. Les nombres flous à supports finis imposeront des contraintes aux fonctions G et D afin qu'elles soient uniques pour chaque famille.

Par exemple, pour G, nous aurons les contraintes
$G_-(x) = ax + b$ avec

$$\begin{cases} G_-(\frac{(m-\alpha)-m}{\alpha}) = 0, \\ G_-(\frac{(m-m}{\alpha}) = 1. \end{cases}$$

pour toute la famille des nombres flous triangulaires.
D'où $b = 1$ et $a(\frac{-\alpha}{\alpha}) + 1 = 0$, soit $a = 1$.
Donc $G_-(x) = x + 1$. Similairement, on aura $D_+(x) = -x + 1$.

Calculons $\widetilde{m} \oplus \widetilde{n}$. D'après le lemme 1.2, on sait que pour $\omega \in]0, 1[, \exists\, p, r \in \mathbb{R}$ tels que $\omega = \mu_{\widetilde{m}-}(p) = \mu_{\widetilde{n}-}(r) = \mu_{\widetilde{m}\oplus\widetilde{n}-}(z)$ où $z = p + r$. Cela peut s'exprimer comme suit :

$$\omega = G_- \left(\frac{p-m}{\alpha} \right) = G_- \left(\frac{r-n}{\gamma} \right);$$

ce qui est équivalent à $p = m - \alpha \times G_-^{-1}(\omega)$ et $r = n - \gamma \times G_-^{-1}(\omega)$; cela implique que $z = p + r = m + n - (\alpha + \gamma) \times G_-^{-1}(\omega)$; ce qui est identique à

$$\omega = G_- \left(\frac{z - (m+n)}{\alpha + \gamma} \right) = \mu_{\widetilde{m}\oplus\widetilde{n}-}(z).$$

De la même manière, on peut prouver que

$$\mu_{\widetilde{m}\oplus\widetilde{n}+}(z) = D_+ \left(\frac{z - (m+n)}{\beta + \delta} \right).$$

Une fois les fonctions G et D fixées, on peut écrire les nombres flous gauche-droite $\widetilde{m} = (m, \alpha, \beta)$ et $\widetilde{n} = (n, \gamma, \delta)$. La somme $\widetilde{m} \oplus \widetilde{n}$ est un nombre flou gauche-droite et

$$(m, \alpha, \beta) \oplus (n, \gamma, \delta) = (m + n, \alpha + \gamma, \beta + \delta) \tag{3}$$

Nous savons que l'on peut utiliser des fonctions différentes pour G et D afin d'exprimer une possible asymétrie dans la connaissance de la valeur de m. Une fois la famille fixée, i.e. une fois les fonctions G et D choisies, il est équivalent de connaître (α, β), ou deux degrés d'appartenance, un de chaque côté de m. Plus α et β sont grands, plus le support de la fonction d'appartenance est large et plus le nombre est flou. Un nombre réel ordinaire m sera écrit $(m, 0, 0)$ (les dispersions sont nulles).

Pour des considérations d'implantation, il est essentiel de remarquer que l'équation (3) est indépendante des expressions analytiques de G et D. Si cela s'avère nécessaire, ces expressions peuvent être utilisées à la fin du calcul pour représenter les résultats. En effet, les opérations sont effectuées sur les triplets de façon interne à la famille générique des nombres flous $G - D$ sans que ni G ni D ne soient à priori connues.

La somme des nombres flous est associative et commutative, le triplet (0,0,0) étant l'identité. À moins que ses dispersions soient nulles, un nombre flou n'a pas d'élément symétrique, l'ensemble des nombres flous réels est donc un semi-groupe pour $\oplus$ (voir remarque page 8).

De façon similaire, on retrouve toutes les opérations de base avec cette représentation. Elles sont listées ci-dessous :
Fixons deux fonctions d'appartenance G et D paires :

La somme :
Si $\widetilde{m}$ et $\widetilde{n}$ sont des nombres flous $G-D$, la somme $\widetilde{m} \oplus \widetilde{n}$ est un nombre flou $G-D$ donné par

$$(m,\ \alpha,\ \beta) \oplus (m,\ \gamma,\ \delta) = (m+n,\ \alpha+\gamma,\ \beta+\delta)$$

L'opposé :
Si $\widetilde{m}$ un nombre flou $G - D$, l'opposé $\widetilde{-m}$ est un nombre flou $D - G$ donné par

$$-(m,\ \alpha\ ,\ \beta) = (-m,\ \beta\ ,\ \alpha)$$

La soustraction :
On peut effectuer la soustraction sur deux nombres flous seulement s'ils sont de types opposés ($G-D$ et $D-G$ respectivement), puisque l'opérateur de la différence est défini comme $\widetilde{m} \ominus \widetilde{n} = \widetilde{m} \oplus \widetilde{-n}$. Si $\widetilde{m}$ est un nombre flou $G - D$ et $\widetilde{n}$ est un nombre flou $D - G$, la différence $\widetilde{m} \ominus \widetilde{n}$ est un nombre flou $G - D$ défini comme

$$(m,\ \alpha,\ \beta) \ominus (n,\ \gamma,\ \delta) = (m-n,\ \alpha+\delta,\ \beta+\gamma)$$

En particulier, si $G = D$, la soustraction est facile à effectuer.

La multiplication :
Si $\widetilde{m}$ et $\widetilde{n}$ sont des nombres flous $G-D$, le produit $\widetilde{m} \odot \widetilde{n}$ est un nombre flou $G-D$ et on a l'approximation suivante

$$(m,\ \alpha,\ \beta) \odot (n,\ \gamma,\ \delta) \# (mn,\ m\gamma + n\alpha,\ m\delta + n\beta)$$

À noter ici que le terme $\alpha\gamma\ .\ {G_-}^{-1}(\omega)^2$ a été négligé, ce qui n'est possible qu'à la condition qu'à la fois α et γ soient petits comparés à m et n, et que ω soit dans le voisinage de 1, car alors ${G_-}^{-1}(\omega)$ est proche de 0.

Note. La multiplication approximée des nombres flous positifs est associative et commutative. L'élément neutre pour $\odot$ a pour mode 1 et son support est réduit à ce mode. Il s'agit donc du nombre réel 1. De plus, cette multiplication est distributive pour $\oplus$. L'ensemble des nombres flous positifs est un semi-groupe commutatif pour la loi $\odot$.

L'inverse :
Si $\widetilde{m}$ un nombre flou $G-D$, l'inverse $\widetilde{1/m}$ est approximativement un nombre flou $D-G$, proche de $1/m$ et on l'approximation suivante

$$\widetilde{1/m} = 1/(m,\ \alpha\ ,\ \beta) = (1/m,\ \beta/m^2\ ,\ \alpha/m^2)$$

La division :
La valeur de $\widetilde{m}/\widetilde{n}$ peut maintenant être définie approximativement comme $\widetilde{m}/\widetilde{n} \# \widetilde{m} \odot \widetilde{1/n}$.

Si $\widetilde{m}$ est un nombre flou $G-D$ et $\widetilde{n}$ est un nombre flou $D-G$, le quotient $\widetilde{m}/\widetilde{n}$ est un nombre flou $G-D$ proche de m/n et défini comme suit

$$(m,\ \alpha,\ \beta)/(n,\ \gamma,\ \delta) = (m/n,\ \frac{\delta m + \alpha n}{n^2},\ \frac{\gamma m + \beta n}{n^2})$$

Bibliothèque Fuzzy. Toutes ces opérations sont implantées en redéfinissant les opérations classiques sur les instances de la classe `NombreFlouRed`.

Le maximum :

$$\widetilde{\max}((m,\ \alpha,\ \beta),\ (n,\ \gamma,\ \delta)) = \begin{cases} (n,\ \gamma,\ \delta) & \text{si } n > m, \\ (\max(m,\ n),\ \min(\alpha,\ \gamma),\ \max(\beta,\ \delta)) & \text{si } n = m. \end{cases}$$

Le minimum :

$$\widetilde{\min}((m,\ \alpha,\ \beta),\ (n,\ \gamma,\ \delta)) = \begin{cases} (m,\ \alpha,\ \beta) & \text{si } m < n, \\ (\min(m,\ n),\ \max(\alpha,\ \gamma),\ \min(\beta,\ \delta)) & \text{si } m = n. \end{cases}$$

Ce maximum et ce minimum induisent une relation d'ordre partielle sur les nombres flous.

Bibliothèque Fuzzy. Ces max et min sont implantés dans les fonctions `maxi(self,autre)` et `mini(self,autre)` de la classe `NombreFlou`. Les relations d'ordre partielles qu'ils induisent sont implantées dans les fonctions `plusgrand(self,autre)` et `pluspetit(self,autre)`.

2 Résolution algébrique

Dans cette section sont présentés d'abord les notions nécessaires [3] à la mise en oeuvre de l'algorithme de décomposition triangulaire de Wu Wen Tsun, puis l'algorithme lui même [6]. Dans le but de trouver les solutions exactes du système de polynômes à coefficients flous de départ, cet algorithme sera appliqué sur un système d'équations intermédiaire. Le grand avantage de cet algorithme est qu'il renvoie des ensembles triangulaires de polynômes qui sont faciles à résoudre par substitution, contraiement à d'autres algorithmes de résolution de systèmes algébriques comme ceux utilisant les bases de Gröbner [11].

2.1 Notions de base

Les définitions et notations introduites ici concernent les polynômes multivariés qui seront manipulés ensuite.

Contrairement à la représentation distribuée des polynômes utilisée dans la théorie des bases de Groebner, où ces derniers sont vus comme une somme de monômes, nos méthodes présentées plus loin sont associées à une vision récursives des polynômes. En effet, ici un polynômes p est considéré comme un polynôme univarié en la variable la plus grande apparaissant dans p.

2.1.1 Définitions et notations

Soit $\mathfrak{R} = \mathbb{K}[x_1, x_2, \ldots, x_n]$ l'anneau de polynômes en n variables $x_1, x_2, \ldots, x_n$ sur le corps $\mathbb{K}$ de caractéristique nulle.

On suppose que les variables sont ordonnées selon l'ordre lexicographique, à savoir $x_1 < x_2 < \cdots < x_n$.

En choisissant la variable x_m, alors un polynôme $p \in \mathfrak{R}$ peut s'écrire :

$$p = I_t x_m^t + I_{t-1} x_m^{t-1} + \cdots + I_0$$

avec t le degré de p en x_m, noté $deg_{x_m}(p)$, et

$$I_i \in \mathbb{K}[x_1, x_2, \ldots, x_{m-1}, x_{m+1}, \ldots, x_n]$$

pour $0 \leq i \leq t$.

On appelle coefficient dominant de p en x_m et on note $lc(p, x_m)$ le coefficient I_t de x_m^t.

Définition 2.1. Soit p un polynôme de $\mathfrak{R}$ tel que $p \notin \mathbb{K}$. La classe de p, notée $class(p)$, est le plus grand indice des variables x_i, $i \in \{1, \ldots, n\}$, apparaissant dans p.

Remarque. La classe d'un polynôme constant est définie comme étant 0.

Le fait d'avoir choisi l'ordre lexicographique pour notre anneau signifie que la relation d'ordre sur les classes induit la relation d'ordre sur les variables, i.e. si $i > j$, $x_i > x_j$.

Définition 2.2. Soit p un polynôme non constant de $\mathfrak{R}$ tel que $class(p) = c$. On écrit $p = I_t x_c^t + r$, où $t = deg_{x_c}(p)$, $I_t = lc(p, x_c)$ et r le reste du polynôme.

On définit les notions suivantes :

- le polynôme $lc(p, x_c)$ dans $\mathbb{K}[x_1, x_2, \ldots, x_{c-1}, x_{c+1}, \ldots, x_n]$ est appelé initial de p et noté $init(p)$,
- le polynôme r est appelé queue de p, désigné par $tail(p)$,
- le terme $I_t x_c^d$ est appelé tête de p, et noté $head(p)$.

Exemple 2.1. Soit $n = 4$. Si $p = (x_2 x_3 + x_2^3)x_4 - 2x_1$, alors on a $class(p) = 4$, $init(p) = x_2 x_3 + x_2^3$ et $tail(p) = -2x_1$.

On introduit à présent un ordre partiel sur les polynômes de $\mathfrak{R}$.

Définition 2.3. Ordre de Ritt

Soient $p, q \in \mathfrak{R}$. On dit que p est plus petit que q, et on note $p \prec q$ si l'une des conditions suivantes est vérifiée :

1. $p \in \mathbb{K}$ et $q \notin \mathbb{K}$,
2. $p, q \notin \mathbb{K}$ et $class(p) < class(q)$,

3. $p, q \notin \mathbb{K}$, $class(p) = class(q) = c$, et $deg_{x_c}(p) < deg_{x_c}(q)$.

On dit que p est plus grand que q pour l'ordre de Ritt et on note $p \succ q$ si $q \prec p$.

Si $class(p) = class(q) = c$ et $deg_{x_c}(p) = deg_{x_c}(q)$, ou si les deux polynômes sont constants, alors p et q sont dits équivalents et on note $p \sim q$.

Remarque. Soit $p \in \mathfrak{R}$ tel que $p \notin \mathbb{K}$. On a

$$init(p) \prec p \text{ et } tail(p) \prec p.$$

De plus, il est clair que toute suite de $\mathfrak{R}$ de premier élément p strictement décroissante pour l'ordre de Ritt est finie.

2.1.2 Réduction polynomiale

Définition 2.4. Soient $p, q \in \mathfrak{R}$ tels que $q \notin \mathbb{K}$. On dit que p est réduit par rapport à q si l'une des conditions suivante est vérifiée :

1. $p \prec q$,
2. $p \notin \mathbb{K}$, $class(p) > class(q)$ et $init(p)$ et $tail(p)$ sont tous deux réduits par rapport à q.

La réduction, appelée aussi parfois réduction forte, se caractérise de façon simple.

Proposition 2.1. *Soient $p, q \in \mathfrak{R}$ tels que $q \notin \mathbb{K}$. On dit que p est réduit par rapport à q si et seulement si $deg_{x_c}(p) < deg_{x_c}(q)$ où $c = class(q) \neq 0$.*

Démonstration. L'équivalence est triviale pour les éléments de $\mathbb{K}$ qui sont les éléments minimaux dans $\mathfrak{R}$ pour l'ordre de Ritt. La remarque précédente permet d'obtenir le résultat par une induction sur p.

□

On dit que le polynôme p est réduit par rapport à $Q \subset \mathfrak{R}$ si p est réduit par rapport à chaque $q \in Q$.

2.1.3 Ensembles triangulaires

Dans cette section sont données les définitions les plus générales concernant les ensembles triangulaires. La terminaison de l'algorithme de Wu est basée sur la notion d'ensemble triangulaire réduit.

Définition 2.5. Un ensemble ordonné de polynômes $F = \{f_1, \ldots, f_r\}$ est appelé un ensemble triangulaire si $r = 1$ ou si $class(f_1) < \cdots < class(f_r)$.

Définition 2.6. Un ensemble triangulaire F est dit réduit si chaque f_j est réduit par rapport à chaque f_i, pour $i < j$.

Étendons à présent l'ordre de Ritt sur les polynômes pour fournir un ordre sur les ensembles triangulaires réduits.

Définition 2.7. Soient $F = \{f_1, \ldots, f_r\}$ et $G = \{g_1, \ldots, g_k\}$ deux ensembles triangulaires réduits. On dit que F est plus petit que G pour l'ordre de Ritt et on note $F \prec G$, si l'une des deux conditions suivantes est vérifiée :

1. $\exists\ j \leq min\{r, k\}$ tel que $f_j \prec g_j$ et $f_i \sim g_i$ pour chaque $i < j$,
2. $r > k$ et $f_i \sim g_i\ \forall\ i \leq k$.

On dit que F est plus grand que G et on écrit $F \succ G$ si $G \prec F$. Lorsque ni $F \prec G$ ni $F \succ G$, on dit que F et G sont équivalents, on écrit alors $F \sim G$.

Lemme 2.2. *Une suite strictement décroissante d'ensembles triangulaires réduits est finie.*

D'après le lemme 2.2, il existe un ensemble triangulaire réduit contenu dans $F \subset \mathfrak{R} \setminus \mathbb{K}$ de rang minimal pour l'ordre de Ritt.

Définition 2.8. Soit $F \subset \mathfrak{R} \setminus \mathbb{K}$. On appelle ensemble basique de F un sous-ensemble $B \subset F$ tel que B est un élément de rang minimal pour l'ordre de Ritt parmi la famille d'ensembles triangulaires réduits contenus dans F.

Nous donnons ci-après l'algorithme permettant, à partir d'un ensemble $F \subset \mathfrak{R} \setminus \mathbb{K}$, d'obtenir un ensemble basique B de F.

Algorithm 1 Construire un ensemble basique B de F

Require: $F \subset \mathfrak{R} \setminus \mathbb{K}$
Ensure: B, un ensemble basique de F
  $B := \emptyset$
  while $F \neq \emptyset$ **do**
    $B := B \cup \{b\}$ où b est un polynôme de rang minimal dans F
    $F := \{f \in F \mid f$ est réduit par rapport à $b\}$
  end while
  return B.

Lemme 2.3. *Soit B un ensemble basique d'une ensemble polynomial F. Si $g \in \mathfrak{R}$ est réduit par rapport à F, alors l'ensemble basique de $F \cup \{g\}$ est plus petit que B.*

2.1.4 Propriétés de la pseudo-division

Introduisons à présent une division de polynômes multivariés, connue comme la *pseudo-division* et utilisée dans l'algorithme de Wu. Nous présenterons ensuite son algorithme en détail.

Proposition 2.4. *Soient $f, g \in \mathfrak{R}$ et $c = class(f)$. Alors il existe une équation de la forme*

$$I_t^m g = qf + r$$

où $q, r \in \mathfrak{R}$, $I_t = init(f)$, $m \geq 0$ et $r = 0$ ou r est réduit par rapport à f. En particulier, q et r sont uniques si m est minimal.

Les polynômes q et r dans la proposition 2.4 sont respectivement le pseudo-quotient et le pseudo-reste $prem(g, f)$ de g dans sa pseudo-division par f (non uniques quand m n'est pas minimal). L'algorithme 2 calcule le pseudo-reste $prem(g, f)$ et le pseudo-quotient :

Algorithm 2 Algorithme de pseudo-division

Require: $g, f \in \mathfrak{R}$
Ensure: r, q pseudo-reste et pseudo-quotient de g dans sa pseudo-division par f
  $r := g, q := 0$
  while $r \neq 0$ et $deg_{x_c}(r) \geq deg_{x_c}(f)$ où $c = class(f)$ **do**
    $r := init(f)r - lc(r, x_c) f x_c^{deg_{x_c}(r) - deg_{x_c}(f)}$
    $q := init(f)q + lc(r, x_c) x_c^{deg_{x_c}(r) - deg_{x_c}(f)}$
  end while
  return r, q.

2.1.5 Ensembles caractéristiques

Soit un ensemble triangulaire réduit $F = \{f_1, \ldots, f_r\}$ et $g \in \mathfrak{R}$. Par la série de pseudo-divisions successives suivante

$$R = prem(\ldots prem(prem(g, f_r) f_{r-1}), \ldots, f_1),$$

on obtient la formule du reste suivant

$$I_1^{s_1} I_2^{s_2} \ldots I_r^{s_r} g = \sum_{i=1}^{r} q_i f_i + R$$

où $I_i = init(f_i)$, $s_i \geq 0$, $q_i \in \mathfrak{R}$ et R est réduit par rapport à tout élément de F. Par la proposition 2.4, si les s_i sont choisis les plus petits possibles, le reste R est unique et noté $prem(g, F)$. Pour un sous-ensemble fini $G \subset \mathfrak{R}$, on pose

$$prem(G, F) = \{prem(g, F) \,|\, g \in G\}.$$

On note par $< F >$ l'idéal généré par $F \subset \mathfrak{R}$. L'ensemble

$$V(F) = \{(a_1, \ldots, a_n) \in \mathbb{K}^n \,|\, f(a_1, \ldots, a_n) = 0, \forall f \in F\}$$

est la variété définie par F.Pour un ensemble polynomial $G \subset \mathfrak{R}$, on défini $V(F/G) = V(F) \backslash V(G)$ comme étant une *variété quasi-algébrique*.

Définition 2.9. Un ensemble triangulaire réduit B dans $\mathfrak{R}$ est appelé *ensemble caractéristique* d'un ensemble polynomial non vide $F \subset \mathfrak{R}$ si $B \subset< F >$ et $prem(F, B) = \{0\}$.

L'algorithme suivant calcule un ensemble caractéristique.

Algorithm 3 Algorithme du calcul d'un ensemble caractéristique

Require: $F \subset \mathfrak{R}$ non vide
Ensure: B, un ensemble caractérististique de F
  $S := F$
  Choisir un ensemble basique B de S
  while $prem(F, B) \neq \{0\}$ **do**
    $S := S \cup prem(F, B) \backslash \{0\}$
    Choisir un ensemble basique B de S
  end while
  return B.

2.2 Décomposition triangulaire de Wu

Les propriétés principales des ensembles caractéristiques sont regroupées dans le théorème suivant.

Théorème 2.5. *(Principe de Wu)*
Soit B un ensemble caractéristique de $F \subset \mathfrak{R}$. Alors

$$V(F) = V(B/I_B) \bigcup \cup_{b \in B} V(F \cup B \cup \{init(b)\})$$

où $I_B = \prod_{b \in B} init(b)$.

Le corollaire suivant sur le calcul des variétés est la clé principale de l'algorithme de Wu.

Corollaire 2.1. *Par la répétition du théorème du Principe de Wu, pour chaque* $F \cup B \cup \{init(b)\}$, $b \in B$, *la procédure s'achèvera en un nombre fini d'étapes. De ce fait,* $V(F)$ *peut être obtenu comme union d'un nombre fini de variétés* $V(B/I_B)$.

Démonstration. D'après les lemmes 2.2 et 2.3, le corollaire est prouvé. □

L'algorithme de Wu a pour but de donner tous les ensembles caractéristiques requis pour le calcul de la variété $V(F)$ par l'application du corollaire 2.1.

Algorithm 4 Algorithme du Wu

Require: $F \subset \mathfrak{R}$ non vide

Ensure: Z, un ensemble d'ensembles caractéristiques tel que $V(F) = \bigcup_{B \in Z} V(B/J)$, où J est le produit des initiaux des polynômes dans l'ensemble B correspondant

```
Z := ∅, D := {F}
while D ≠ ∅ do
    Prendre un élément F' de D
    D := D\{F'}
    Choisir un ensemble caractéristique B de F'
    if B ≠ {1} then
        Z := Z ∪ {B}
        D := D ⋃ ∪_{b∈B} {F' ∪ B ∪ {init(b) | init(b) ≠ 1}}
    end if
end while
return Z
```

L'algorithme de Wu permet d'exprimer la variéré $V(F)$ comme une union de variétés quasi-algébriques d'ensembles caractéristiques. Ainsi, trouver $V(F)$ devient aisé car ces ensembles caractéristiques sont faciles à résoudre.

Exemple 2.2. On applique l'algorihtme de Wu à $F = \{xy + x + y, xy^2 + x + y\}$, avec $y < x$.

Posons $F' = F$, alors $D = \emptyset$.

On commençons par choisir un ensemble caractéristique B de F'.

Un ensemble basique de F' est $B = \{xy + x + y\}$ et $R = y^2 - y^3$ est le pseudo-reste de $xy^2 + x + y$ par B. En ajoutant R à F', le nouvel ensemble basique

$B = \{y^2 - y^3, xy + x + y\}$ obtenu est un ensemble caractéristique.

Alors $Z = \{B\}$. On a $init(xy + x + y) = y + 1$ et $init(y^2 - y^3) = 1$, alors, $D := \{F \cup \{y + 1\}\}$ car B est déjà ajouté à F'.

Posons maintenant $F' := \{xy + x + y, xy^2 + x + y, y + 1\}$. Un ensemble basique de F' est $B = \{y + 1\}$. Le pseudo-reste de $xy + x + y$ par B est 1. Ainsi, $\{1\}$ est un ensemble caractéristique de F' et $D = \emptyset$. Donc, la sortie est $Z = \{\{y^2 - y^3, xy + x + y\}\}$ et

$$V(F) = V(\{y^2 - y^3, xy + x + y\}) \backslash V(y + 1) = \{(x = 0, y = 0), (x = -\frac{1}{2}, y = 1)\}$$

Afin de pouvoir utiliser cet algorithme pour résoudre un système de polynômes à coefficients flous, il va auparavant nous falloir calculer un système d'équations intermédiaire où toutes les données ne seront plus floues mais exprimées formellement. La section suivante présente cette algébrisation de l'information floue.

3 Du flou à l'algébrique

Dans le système à résoudre, les coefficients flous triangulaires sont donnés dans la représentation en tuple, i.e. par un triplet. Avant de pouvoir utiliser l'algorithme de Wu pour résoudre ce système, il nous faut auparavant obtenir un système d'équations intermédiaire où l'information flou des coefficients a été traitée afin d'obtenir des équations algébriques. Pour cela, les coefficients flous dans la forme en tuple présentée à la section 1 doivent être passés dans une nouvelle représentation ; la représentation paramétrique.

3.1 La forme paramétrique

Définition 3.1. Un nombre flou $\widetilde{n}$ dans la *forme paramétrique* est une paire ordonnée $[\underline{n},\ \overline{n}]$ de fonctions $\underline{n}(r)$ et $\overline{n}(r)$, $r \in [0, 1]$, qui satisfont les conditions suivantes :

- $\underline{n}(r)$ est une fonction bornée à gauche, continue et non décroissante sur $[0, 1]$,
- $\overline{n}(r)$ est une fonction bornée à gauche, continue et non croissante sur $[0, 1]$,
- $\underline{n}(r) \leq \overline{n}(r)$ pour chaque $r \in [0, 1]$.

Bibliothèque Fuzzy. Cette représentation est implantée dans la classe `NombreFlouPM`.

La représentation paramétrique s'accompagne elle aussi d'une arithmétique floue [12] décrite ci-dessous. Les quatre opérations arithmétiques et la multiplication par un scalaire $k \in \mathbb{R}$ sont obtenues grâce à l'arithmétique des intervalles pour tout $r \in [0,1]$.

Soient deux nombres flous $\widetilde{a}$ et $\widetilde{b}$ donnés dans la forme paramétrique et un nombre réel k, on a :

- $\widetilde{a} = \widetilde{b}$, si et seulement si $\underline{a}(r) = \underline{b}(r)$ et $\overline{a}(r) = \overline{b}(r)$ pour chaque $r \in [0,1]$,
- $\widetilde{a} + \widetilde{b} = [\underline{a} + \underline{b},\ \overline{a} + \overline{b}]$,
- $k.\widetilde{a} = \begin{cases} [k.\underline{a},\ k.\overline{a}] & \text{si } k \geq 0, \\ [k.\overline{a},\ k.\underline{a}] & \text{si } k < 0. \end{cases}$

où pour deux fonctions f et g, $f+g$ et $k.f$ sont respectivement la somme classique de fonctions et la multiplication de fonctions par un scalaire k.

- $\widetilde{a} - \widetilde{b} = [\underline{a} - \overline{b},\ \overline{a} - \underline{b}]$,
- $\widetilde{a} * \widetilde{b} = [\,\min\{\underline{ab}\,,\ \underline{a}\overline{b}\,,\ \overline{a}\underline{b}\,,\ \overline{ab}\}\,,\ \max\{\underline{ab}\,,\ \underline{a}\overline{b}\,,\ \overline{a}\underline{b}\,,\ \overline{ab}\}\,]$,
- Si $0 \notin [\underline{b}(0), \underline{b}(0)]$:
  $\widetilde{a}/\widetilde{b} = [\,\min\{\frac{\underline{a}}{\underline{b}}\,,\ \frac{\underline{a}}{\overline{b}}\,,\ \frac{\overline{a}}{\underline{b}}\,,\ \frac{\overline{a}}{\overline{b}}\}\,,\ \max\{\frac{\underline{a}}{\underline{b}}\,,\ \frac{\underline{a}}{\overline{b}}\,,\ \frac{\overline{a}}{\underline{b}}\,,\ \frac{\overline{a}}{\overline{b}}\}\,]$

Bibliothèque Fuzzy. Ces opérations sont implantées en redéfinissant les opérations classiques sur les instances de la classe `NombreFlouPM`. Elles permettent d'assurer un polymorphisme avec les instances de la classe `NombreFlou` pour la loi $\oplus$.

Voyons maintenant comment passer de la représentation en tuple des nombres flous à la représentation paramétrique pour la famille des nombres flous triangulaires, puis la famille des quadratiques.

Pour construire cette représentation à partir de la représentation en tuple, il faut revenir à la fonction d'appartenance d'un nombre flou.

On considère de nouveau un nombre flou $\widetilde{n} = (n, \alpha, \beta)$ de fonction d'appartenance $\mu_{\widetilde{n}}$ telle que :

$$\mu_{\widetilde{n}-}(x) = G_-\left(\frac{x-n}{\alpha}\right), \quad \mu_{\widetilde{n}+}(x) = D_+\left(\frac{x-n}{\beta}\right),$$

avec $\alpha, \beta > 0$ et G et D deux fonctions d'appartenance paires représentant des

nombres flous de mode nul et de support $[-1, 1]$.

On suppose que G et D sont des fonctions inversibles. Alors, dans ce cas, les coupes-r $\widetilde{n}_r$ sont données pour tout $r \in [0, 1]$ par

$$\widetilde{n}_r = [\underline{n}(r), \overline{n}(r)]$$

avec

$$\underline{n}(r) = n + \alpha(G_-^{-1}(r) - 1) \quad \text{et} \quad \overline{n}(r) = n + \beta(1 - D_+^{-1}(r))$$

3.2 Le cas triangulaire

Les coefficients du système de polynômes que nous souhaitons résoudre sont des nombres flous triangulaires [4]. Le cas du passage à la forme paramétrique pour les nombres flous triangulaires se révèle donc d'une grande importance.

Soit $\widetilde{n} = (n, \alpha, \beta)$ un nombre flou triangulaire sous forme tuple de fonction d'appartenance $\mu_{\widetilde{n}}$. Ses restrictions gauche et droite sont donc linéaires et de la forme :

$$\mu_{\widetilde{n}-}(x) = a(\frac{x-n}{\alpha}) + b \text{ et } \mu_{\widetilde{n}+}(x) = a(\frac{n-x}{\beta}) + b$$

On sait que en $\mu_{\widetilde{n}-}(n-\alpha) = 0$ et $\mu_{\widetilde{n}-}(n) = 1$, d'où on a :

$$\begin{cases} 0 &= a(\frac{n-\alpha-n}{\alpha}) + b \\ 1 &= a(\frac{n-n}{\alpha}) + b \end{cases} \Leftrightarrow \begin{cases} 0 &= -a + b \\ 1 &= b \end{cases} \Leftrightarrow \Big\{ a = b = 1$$

La fonction d'appartenance s'écrit donc :

$$\mu_{\widetilde{n}}(x) \quad = \quad \begin{cases} \mu_{\widetilde{n}-}(x) = \frac{x-n}{\alpha} + 1 & n - \alpha \leq x \leq n, \\ \mu_{\widetilde{n}+}(x) = \frac{n-x}{\beta} + 1 & n \leq x \leq n + \beta, \\ 0 & \text{sinon.} \end{cases}$$

Cherchons la fonction $\underline{n}(y)$ inverse de $\mu_{\widetilde{n}-}(x)$ pour $x \in [n - \alpha, n]$ et $y \in [0, 1]$. Nous avons :

$$y = \mu_{\widetilde{n}-}(x) = \frac{x-n}{\alpha} + 1$$

D'où $x = \alpha y + n - \alpha = \underline{n}(y)$.
De façon similaire, on obtient :

$$\overline{n}(y) = -\beta y + n + \beta$$

On pose $y = r$ le paramètre, et on obtient finalement :

$$\widetilde{n} = [\underline{n}, \overline{n}]$$

avec $\underline{n}(r) = \alpha r + n - \alpha$ et $\overline{n}(r) = -\beta r + n + \beta$ pour $r \in [0, 1]$.

On considère le nombre flou triangulaire $\widetilde{m} = (3, 2, 3)$. Le graphe des fonctions de sa forme paramétrique est obtenu en effectuant une rotation plane de 90° autour de l'origine du graphe de sa fonction d'appartenance. On effectue ensuite une symétrie verticale par rapport à l'axe $x = 3$ afin de retrouver les axes dans le bon sens.

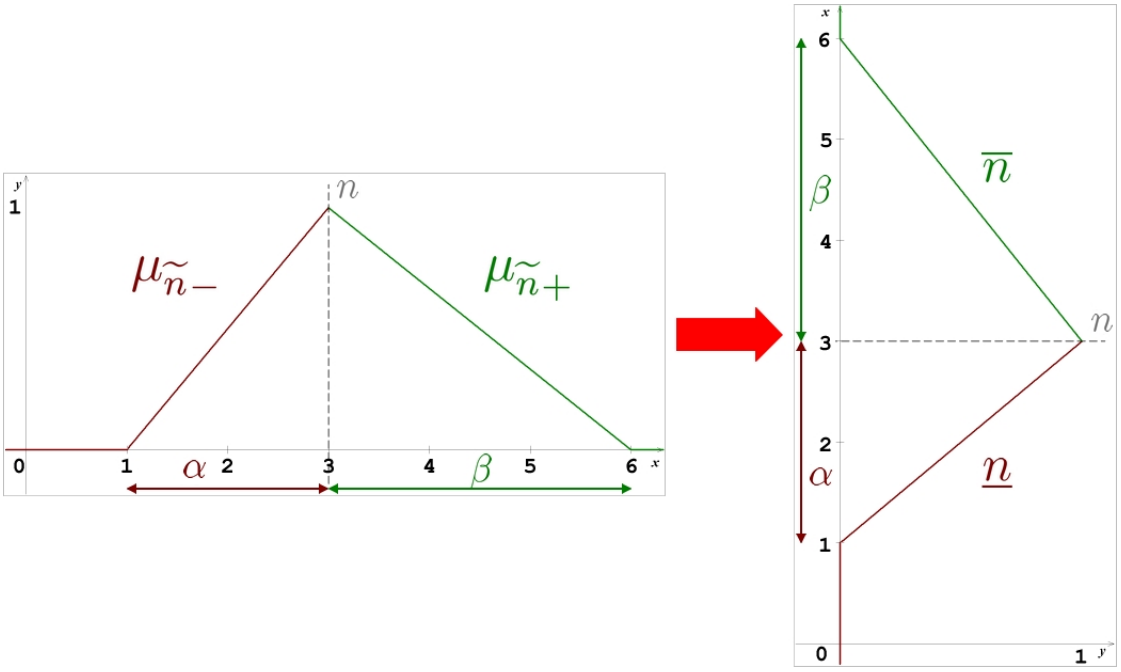


FIGURE 7 – Graphe des fonctions de la forme paramétrique obtenu par rotation plane et symétrie verticale du graphe de la fonction d'appartenance

Bibliothèque Fuzzy. Ce passage d'une représentation à l'autre est assuré par la fonction `forme_parametrique(self)` de la classe `NombreFlou` qui s'applique sur une instance de la classe `NombreFlou` et renvoie une instance équivalente de la classe `NombreFlouPM`.

Dans la section 4, pour algébriser un système polynomial à coefficients flous triangulaires, on remplacera chaque nombre flou triangulaire $\widetilde{n}$ sous sa forme tuple par sa forme paramétrique comprenant ces deux fonctions linéaires. La résolution se ramène alors à celle d'un système sans valeurs floues. L'exemple ci-dessous,

réalisé à l'aide de la bibliothèque Fuzzy, illustre le fondement de cette démarche sur une seule équation.

Exemple 3.1. Soit l'équation suivante :

$$\tilde{n_1}x_1x_2 + \tilde{n_2}x_3^3 = \tilde{n_3}$$

où $\widetilde{n_1}$,$\widetilde{n_2}$ et $\widetilde{n_3}$ sont des nombres flous triangulaires dont les représentations sont

nombre flou	forme tuple	forme paramétrique
$\widetilde{n_1}$	$(2, 1, 3)$	$[r + 1, -3r + 5]$
$\widetilde{n_2}$	$(-1, 0, 1)$	$[-1, -r]$
$\widetilde{n_3}$	$(2, 2, 1)$	$[2r, -r + 3]$

En écrivant chaque nombre flou dans sa forme paramétrique, on a :

$$[r + 1, -3r + 5]x_1x_2 + [-1, -r]x_3^3 = [2r, -r + 3] \tag{4}$$

Par les opérations arithmétiques sur les nombre flous paramétriques décrites plus haut, l'équation (4) est équivalente à l'équation

$$[x_1x_2r + x_1x_2, -3rx_1x_2 + 5x_1x_2] + [-x_3^3, -rx_3^3] = [2r, -r + 3]$$

$$\Leftrightarrow [x_1x_2r - x_3^3 + x_1x_2, (-x_3^3 - 3x_1x_2)r + 5x_1x_2] = [2r, -r + 3]$$

Par identification, on obtient :

$$\begin{cases} x_1x_2r - x_3^3 + x_1x_2 & = 2r \\ (-x_3^3 - 3x_1x_2)r + 5x_1x_2 & = -r + 3 \end{cases}$$

On construit ainsi à partir d'une équation polynomiale un système de 2 équations linéaires en une variable supplémentaire r. Les solutions de l'équation d'origine sont celles pour lesquelles le système obtenu est satisfait pour tout $r \in [0, 1]$.

3.3 Le cas quadratique

Le cas quadratique est également intéressant à étudier car il raffine l'information sur le nombre flou. Cette section se concentre sur les résultats trouvés en poussant l'exploration du passage à la forme paramétrique pour les nombres flous quadratiques [10].

Soit $\widetilde{n} = (n, \alpha, \beta)$ un nombre flou quadratique sous forme tuple de fonction d'appartenance $\mu_{\widetilde{n}}$. Ses restrictions gauche et droite sont donc quadratiques et de la forme :

$$\mu_{\widetilde{n}-}(x) = G_-(\frac{x-n}{\alpha}) \text{ et } \mu_{\widetilde{n}+}(x) = D_+(\frac{n-x}{\beta})$$

où G et D de support [-1,1]. G et D vont être décomposées pour permettre une unicité de représentation pour $\widetilde{n}$ comme dans le cas triangulaire. Afin de respecter les contraintes d'unicité de G et D pour la famille quadratique, les contraintes suivantes sont appliquées à G_- (symétriquement à D_+) :

$$G_-(x) = \begin{cases} G_1(x) & -1 \leq x \leq -\frac{1}{2}, \\ G_2(x) & -\frac{1}{2} \leq x \leq 0, \\ 0 & \text{sinon.} \end{cases}$$

avec

$$\begin{cases} G_1(-1) = 0, \\ G_1'(-1) = 0, \\ G_1(-\frac{1}{2}) = \frac{1}{2}. \end{cases}$$

et

$$\begin{cases} G_2(-\frac{1}{2}) = \frac{1}{2}, \\ G_2(0) = 1, \\ G_2'(0) = 0. \end{cases}$$

Il est facile de vérifier que $G_1(x) = 2(x+1)^2$ et $G_2(x) = 1 - 2x^2$ sont les deux seules fonctions qui satisfont ces contraintes.

De même, on a :

$$D_+(x) = \begin{cases} D_1(x) & 0 \leq x \leq \frac{1}{2}, \\ D_2(x) & \frac{1}{2} \leq x \leq 1, \\ 0 & \text{sinon.} \end{cases}$$

avec $D_1(x) = 1 - 2x^2$ et $D_2(x) = 2(-x+1)^2$.

La fonction d'appartenance d'un nombre flou quadratique s'écrit donc :

$$\mu_{\widetilde{n}}(x) = \begin{cases} \mu_{\widetilde{n}-}(x) = G_-(\frac{x-n}{\alpha}) & n - \alpha \leq x \leq n, \\ \mu_{\widetilde{n}+}(x) = D_+(\frac{n-x}{\beta}) & n \leq x \leq n + \beta, \\ 0 & \text{sinon.} \end{cases}$$

avec

$$G_-(\tfrac{x-n}{\alpha}) = \begin{cases} G_1(\tfrac{x-n}{\alpha}) = g_1(x) = 2\frac{(x-n+\alpha)^2}{\alpha^2} & n-\alpha \le x \le n-\frac{\alpha}{2}, \\ G_2(\tfrac{x-n}{\alpha}) = g_2(x) = 1-2\frac{(x-n)^2}{\alpha^2} & n-\frac{\alpha}{2} \le x \le n, \\ 0 & \text{sinon.} \end{cases}$$

et

$$D_+(\tfrac{n-x}{\beta}) = \begin{cases} D_1(\tfrac{n-x}{\beta}) = d_1(x) = 1-2\frac{(x-n)^2}{\beta^2} & n \le x \le n+\frac{\beta}{2}, \\ D_2(\tfrac{n-x}{\beta}) = d_2(x) = 2\frac{(x-n-\beta)^2}{\beta^2} & n+\frac{\beta}{2} \le x \le n+\beta, \\ 0 & \text{sinon.} \end{cases}$$

Comme nous avons procédé pour le cas triangulaire, cherchons la fonction $b_1(y)$ inverse de $g_1(x)$ pour $x \in [n-\alpha, n-\frac{\alpha}{2}]$ et $y \in [0,1]$.

Nous avons :

$$y = g_1(x) = 2\frac{(x-n+\alpha)^2}{\alpha^2}$$

D'où

$$\begin{aligned} x &= \alpha\sqrt{\frac{y}{2}} + n - \alpha \text{ car } y > 0, \\ &= b_1(y). \end{aligned}$$

De façon similaire, on obtient :

$$\begin{aligned} b_2(y) &= n - \alpha\sqrt{\frac{1-y}{2}} && \text{la fonction inverse de } g_2(x), \\ h_1(y) &= n + \beta\sqrt{\frac{1-y}{2}} && \text{la fonction inverse de } d_1(x), \\ h_2(y) &= -\beta\sqrt{\frac{y}{2}} + n + \beta && \text{la fonction inverse de } d_2(x). \end{aligned}$$

On pose $y = r$ le paramètre, et on obtient :

$$\widetilde{n} = [\underline{n}, \overline{n}]$$

avec

$$\underline{n}(r) = \begin{cases} b_1(r) = \sqrt{\frac{r}{2}} + n - \alpha & 0 \le r \le \frac{1}{2}, \\ b_2(r) = n - \alpha\sqrt{\frac{1-r}{2}} & \frac{1}{2} \le r \le 1, \\ 0 & \text{sinon.} \end{cases}$$

et

$$\overline{n}(r) \quad = \quad \begin{cases} h_1(r) = n + \beta\sqrt{\frac{1-r}{2}} & \frac{1}{2} \leq r \leq 1, \\ h_2(r) = -\beta\sqrt{\frac{r}{2}} + n + \beta & 0 \leq r \leq \frac{1}{2}, \\ 0 & \text{sinon.} \end{cases}$$

On considère le nombre flou quadratique $\widetilde{m} = (3, 2, 3)$. Comme pour le cas triangulaire, le graphe des fonctions de sa forme paramétrique est obtenu en effectuant une rotation plane de 90° autour de l'origine du graphe de sa fonction d'appartenance. On effectue ensuite une symétrie verticale par rapport à l'axe $x = 3$ afin de retrouver les axes dans le bon sens.

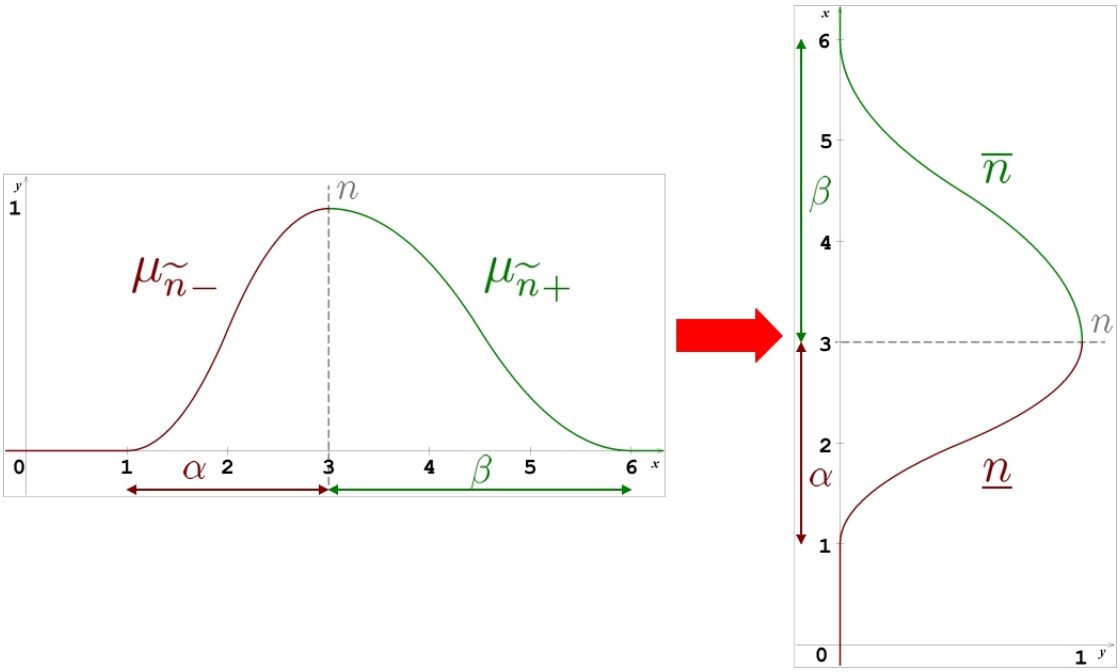


FIGURE 8 – Graphe des fonctions de la forme paramétrique obtenu par rotation plane et symétrie verticale du graphe de la fonction d'appartenance

Comme on ne fait pas de calculs algébriques avec des racines, on procède à deux nouveaux changements de variables.

Posons $u = \sqrt{\frac{r}{2}}$ et $v = \sqrt{\frac{1-r}{2}}$, on obtient finalement :

$$\begin{cases} B_2(v) & = n - \alpha v \\ H_1(v) & = n + \beta v \\ B_1(u) & = \alpha u + n - \alpha \\ H_2(u) & = -\beta u + n + \beta \end{cases}$$

Nous avons alors $u^2 = \frac{r}{2}$ et $v^2 = \frac{1-r}{2}$, d'où $u^2 + v^2 = \frac{1}{2}$.

Ainsi, pour algébriser un système flou quadratique similairement au cas triangulaire, nous remplacerons chaque nombre flou $\widetilde{n}$ par chacune de ses 4 expressions paramétrées $B_2(v), H_1(v), B_1(u), H_2(u)$. Le cas triangulaire produit un système d'équations comportant le double d'équations par rapport au système flou avec la contrainte $r \in [0, 1]$. Dans le traitement d'une seule équation, nous obtiendrons 4 fois plus d'équations que dans le système flou. Les contraintes sont $u^2 + v^2 = \frac{1}{2}$, et les solutions retenues seront celles satisfaites pour tout $u \in [0, \sqrt{\frac{1}{2}}]$ et $v \in [\sqrt{\frac{1}{2}}, 1]$.

4 Procédure de résolution de systèmes de polynômes à coefficients flous triangulaires

Maintenant que l'algébrisation du système de polynômes à coefficients floues et l'algorithme de Wu ont été présentés, nous allons pouvoir dérouler la procédure complète de résolution de ces systèmes.

Nous présentons dans cette section l'algorithme complet pour résoudre un système de polynômes où les coefficients sont des nombres flous triangulaires [4]. Soit le système suivant :

$$AX + B = CX + D \tag{5}$$

où X est un vecteur de n variables réelles, et A, B, C, D sont des matrices de nombres flous triangulaires de s lignes et n colonnes.

4.1 La méthode de résolution

En développant l'expression du système (5), on obtient le système de s polynômes en n variables :

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = \widetilde{b}_1 \\ \quad \vdots \\ f_s(x_1, x_2, \ldots, x_n) = \widetilde{b}_s \end{cases} \tag{6}$$

où $x_1, x_2, \ldots, x_n$ sont des variables réelles et tous les coefficients et les valeurs à droite des égalités sont des nombres flous triangulaires.

Reprenons l'idée ébauchée sur l'exemple 3.1. Pour commencer, nous écrivons tous les nombres flous sous forme paramétrique. Par les opérations arithmétiques sur les nombres flous et la définition de l'égalité de deux nombre flous sous forme paramétrique vu à la section 3, on obtient un système de polynômes réels comme suit :

$$\begin{cases} f_{1,1}(x_1, x_2, \ldots, x_n, r) = \overline{b_1}(r) \\ f_{1,2}(x_1, x_2, \ldots, x_n, r) = \underline{b_1}(r) \\ \quad \vdots \\ f_{s,1}(x_1, x_2, \ldots, x_n, r) = \overline{b_s}(r) \\ f_{s,2}(x_1, x_2, \ldots, x_n, r) = \underline{b_s}(r) \end{cases} \tag{7}$$

avec $2s$ polynômes et $n+1$ variables $x_1, \ldots, x_n, r$ où $r \in [0, 1]$. On apelle ce nouveau système (7) la forme tranchée du système (6).

La proposition suivante exprime la relation entre les solutions d'un système polynomial à coefficients flous et les solutions du système de sa forme tranchée.

Proposition 4.1. *Soit F un système de polynômes à coefficients flous comme le système (6) et F_1 le système de sa forme tranchée. Alors*

$$\{(a_1, \ldots, a_n) \in \mathbb{R}^n \mid (a_1, \ldots, a_n, r) \in V(F_1), \forall r \in [0, 1]\}$$

est l'ensemble des solutions de F où r est le paramètre.

Démonstration. On considère un système de polynômes à coefficients flous F comme le système (6). On commence par écrire tous les nombres flous sous forme paramétrique et on obtient ainsi la forme paramétrique F' du système F avec r comme paramètre. Les systèmes F et F' ont les mêmes solutions quand le paramètre r se trouve compris en 0 et 1 de façon arbitraire. Par les opérations arithmétiques sur les nombres flous et la définition de l'égalité de deux nombres flous sous forme paramétrique, on obtient la forme tranchée F_1 qui est un système à coefficients réels en les variables $x_1, \ldots, x_n, r$. Ensuite, nous obtenons les solutions

de F' à partir des solutions de F_1 avec la variable r comprise arbitrairement entre 0 et 1. Finalement, l'ensemble

$$\{(a_1, \ldots, a_n) \in \mathbb{R}^n \,|\, (a_1, \ldots, a_n, r) \in V(F_1), \forall r \in [0,1]\}$$

est l'ensemble des solutions de F' et par conséquent est aussi celui de F.

□

Quand tous les coefficients sont des nombres flous triangulaires, alors le système (7) est linéaire en r. Par conséquent, ce système peut s'écrire comme suit

$$\begin{cases} h_1(x_1, x_2, \ldots, x_n)r + g_1(x_1, x_2, \ldots, x_n) = 0 \\ h_2(x_1, x_2, \ldots, x_n)r + g_2(x_1, x_2, \ldots, x_n) = 0 \\ \quad\quad \vdots \\ h_{2s}(x_1, x_2, \ldots, x_n)r + g_{2s}(x_1, x_2, \ldots, x_n) = 0 \end{cases} \tag{8}$$

où $h_i, g_i \in \mathbb{K}[x_1, x_2, \ldots, x_n]$. Le système polynomial formé par les polynômes h_i et g_i, $1 \leq i \leq 2s$, est appelé *la forme tranchée collectée* du système flou. Les polynômes h_i et g_i sont appelés respectivement les polynômes de collecte gauche et de collecte droite. Le théorème suivant montre la relation entre les solutions d'un système à coefficients flous et la variété du système de sa forme tranchée collectée.

Théorème 4.2. *Soit F un système de polynômes à coefficients flous comme le système (6). Alors, l'ensemble des solutions de F est égal à la variété du système de sa forme tranchée collectée.*

Démonstration. D'après la proposition 4.1, l'ensemble des solutions de F est

$$V = \{(a_1, \ldots, a_n) \in \mathbb{R}^n | (a_1, \ldots, a_n, r) \in V(F_1), \forall r \in [0,1]\}$$

où F_1 est le système de la forme tranchée de F et r le paramètre. Soit $V' = V(h_1, \ldots, h_{2s}, g_1, \ldots, g_{2s})$ où $h_1, \ldots, h_{2s}$ et $g_1, \ldots, g_{2s}$ sont respectivement les polynômes de collecte gauche et de collecte droite.

Soit $(a_1, \ldots, a_n) \in V$. Alors $(a_1, \ldots, a_n, 0) \in V(F_1)$. En substituant $x_i = a_i$ et $r = 0$ dans chaque élément du système de la forme tranchée, on obtient $g_i(a_1, \ldots, a_n) = 0$ pour $1 \leq i \leq 2s$. On a également $(a_1, \ldots, a_n, 1) \in V(F_1)$. En substituant $x_i = a_i$ et $r = 1$, on obtient

$$h_i(x_1, x_2, \ldots, x_n) = -g_i(x_1, x_2, \ldots, x_n) = 0$$

pour $1 \leq i \leq 2s$. Par conséquent, $V \subseteq V'$.
À présent, soit $(a_1, \ldots, a_n) \in V'$. Alors $(a_1, \ldots, a_n, r)$ est un élément de la variété de la forme tranchée pour chaque $r \in \mathbb{R}$. Ainsi, $V' \subseteq V$ et la preuve est achevée.

□

À partir de ces méthodes de résolution de systèmes à coefficients flous, on construit l'algorithme principal décrit ci-dessous. On utilise l'algorithme de Wu et on calcule la variété de l'ensemble de polynômes obtenu à partir du système de la forme tranchée collectée dans l'anneau $\mathbb{R}[x_1, x_2, \ldots, x_n]$.

Algorithm 5 Algorithme principal

Require: F un système de polynômes à coefficients flous triangulaires
Ensure: V l'ensemble des solutions de F
1: Calculer la forme paramétrique de F
2: Calculer F', le système de la forme tranchée collectée de F.
3: Calculer un ensemble d'ensembles caractéristiques Z pour F'
4: Calculer la variété V de F' i.e.

$$V(F') = \bigcup_{B \in Z} V(B/I_B)$$

où $I_B = \prod_{b \in B} init(b)$
5: **return** V

Le résultat principal est obtenu par le théorème suivant.

Théorème 4.3. *L'algorithme principal est un algorithme correct et fini.*

Démonstration. Par le corollaire 2.1 et le thèorème 4.2, le théorème est prouvé. □

4.2 Exemple issue d'une application en économie

L'équilibre entre l'offre et la demande détermine le prix du marché d'une marchandise et la quantité de production.

On suppose que la demande et l'offre sont des fonctions polynomiales non linéaires du prix f_d et f_o, telles que $q_d = f_d(p)$ et $q_o = f_o(p)$ avec :

$$\begin{cases} q_d + a = b.p^2, \\ q_o + c = d.p^2 \end{cases}$$

où q_o et q_d sont les quantités d'offre et de demande requise, p est le prix et a, b, c et d sont des coefficients définis par une estimation. Les coefficients a, b, c et d sont représentés par des nombres flous triangulaires et q_o, q_d et p sont des variables réelles.

L'objectif de l'étude est d'arriver à l'égalité de l'offre et de la demande, soit $q_d = q_o$.

Posons $q_d = q_o = x$ et $p = y$. Il s'agit donc de résoudre le système flou non linéaire

$$F : \begin{cases} x + (-1, 1, 1) = (-2, 1, 1)y^2, \\ x + (3, 1, 1) = (2, 1, 1)y^2 \end{cases}$$

On résoud ce système par l'algorithme 4.1 décrit précédemment.

On commence par calculer le système tranché du système F. En passant les équations à la forme paramétrique, on obtient le système suivant :

$$\begin{aligned} &\begin{cases} x + [r - 2, -r] = [r - 3, -r - 1]y^2, \\ x + [r + 2, -r + 4] = [r + 1, -r + 3]y^2 \end{cases} \\ \Leftrightarrow &\begin{cases} [x + r - 2, x - r] = [y^2 r - 3y^2, -y^2 r - y^2], \\ [x + r + 2, x - r + 4] = [y^2 r + y^2, -y^2 r + 3y^2] \end{cases} \\ \Leftrightarrow &\begin{cases} [(1 - y^2)r + x + 3y^2 - 2, (y^2 - 1)r + x + y^2] = [0, 0], \\ [(1 - y^2)r + x - y^2 + 2, (y^2 - 1)r + x - 3y^2 + 4] = [0, 0] \end{cases} \end{aligned}$$

Par identification, on obtient :

$$\begin{cases} (1 - y^2)r + x + 3y^2 - 2 = 0, \\ (y^2 - 1)r + x + y^2 = 0, \\ (1 - y^2)r + x - y^2 + 2 = 0, \\ (y^2 - 1)r + x - 3y^2 + 4 = 0 \end{cases}$$

Ce nouveau système contient le double d'équations par rapport au système F et le paramètre r.

On construit ci-après le système tranché collecté F', vrai $\forall r \in [0, 1]$, en collectant les coefficients des polynômes linéaires en r du système tranché :

$$F' : \begin{cases} (1 - y^2) = 0, \\ (y^2 - 1) = 0, \\ x + 3y^2 - 2 = 0, \\ x + y^2 = 0, \\ x - y^2 + 2 = 0, \\ x - 3y^2 + 4 = 0 \end{cases}$$

En utilisant notre fonction Sage implantant l'algorithme de Wu sur le système F', on obtient l'ensemble caractéristique $Z = [\{x + 1, y^2 - 1\}]$ et la variété solution $V = \{(x = -1, y \pm 1)\}$.

Ainsi, toutes les solutions de F ont été obtenues de manière exacte par la méthode présentée.

5 Description de notre bibliothèque Fuzzy de Sage

Cette bibliothèque permet à la fois de modéliser les nombres flous dans les différentes représentations gauche-droite, et de résoudre des systèmes de polynômes à coefficients flous triangulaires ou réels.

Nous verrons d'abord comment sont structurées les données floues, puis l'implantation de la méthode algébrique de résolution des sytèmes polynomiaux à coefficients flous triangulaires.

5.1 Les classes des nombres flous

5.1.1 La classe NombreFlou

La classe `NombreFlou` permet de modéliser dans la représentation en tuple des nombres flous avec des restrictions gauche et droite linéaires ou quadratiques. Ces nombres flous peuvent être non réduits, i.e. que leur noyau n'est pas forcément réduit à un point.

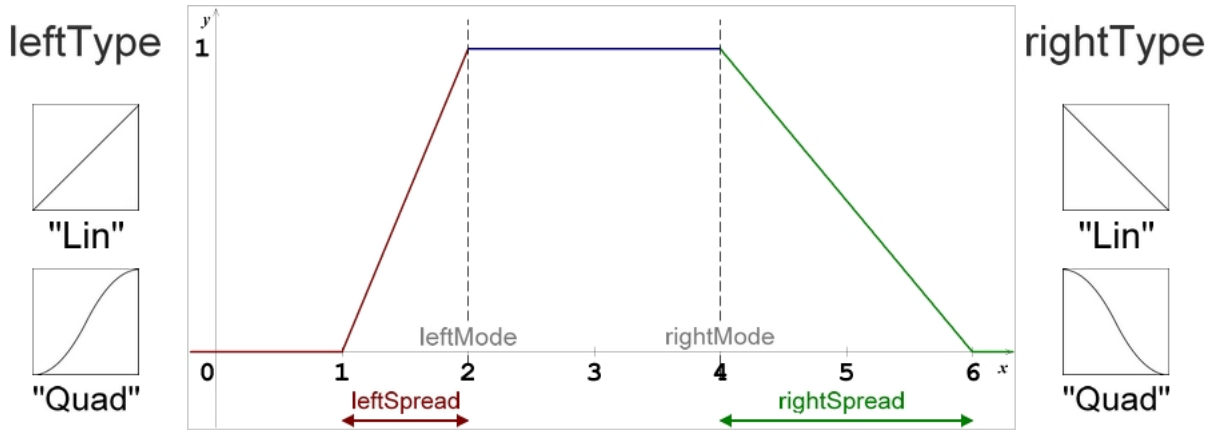
Le constructeur est `NombreFlou(leftMode, rightMode, leftSpread, rightSpread, leftType, rightType)`.

Une instance de cette classe possède les six propriétés suivantes :

- leftMode (un entier) : la borne gauche du noyau du nombre flou
- rightMode (un entier) : la borne droite du noyau du nombre flou
- leftSpread (un entier positif ou nul) : la propagation du nombre flou à gauche du noyau
- rightSpread (un entier positif ou nul) : la propagation du nombre flou à droite du noyau
- leftType (une chaîne de caractères) : le type de la restriction de la fontion d'appartenance à gauche du noyau
- rightType (une chaîne de caractères) : le type de la restriction de la fontion d'appartenance à droite du noyau

Ce ne sont pas toujours, à proprement parler, des nombres flous car leur noyau n'est pas forcément réduit à un seul élément, mais dans la littérature, les nombres flous non réduits aux restrictions linéaires sont souvent référencés par abus de langage comme des nombres flous trapézoïdaux.

Une instance peut être schématisée comme suit



Pour déclarer un nombre flou, on écrit :

```
n1 = NombreFlou(3,4,2,1,"Lin","Quad")
```

Les différentes méthodes de cette classe permettent à la fois d'afficher et de dessiner les graphes de ses instances, mais aussi d'opérer des calculs sur ces instances et de les ordonner.

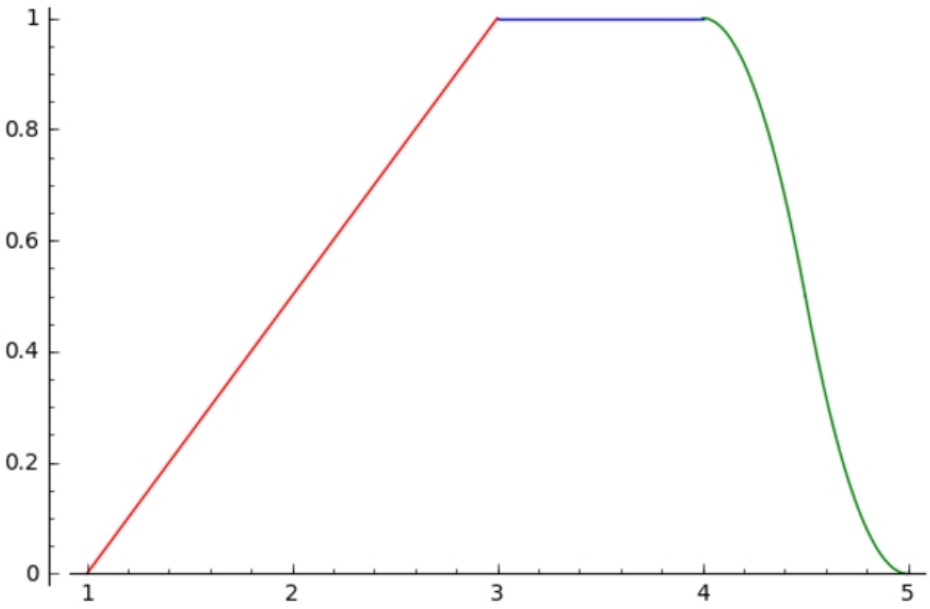
Pour commencer, la fonction `print(nombreFlou)` qui permet d'afficher le nombre flou sous la forme du quadruplet (leftMode, rightMode, leftSpread, rightSpread) et précise sa famille en fonction du type de ses restrictions.

```
print(n1)
```

```
(3,4,2,1), Lin - Quad
```

La fonction `trace(self)` quant à elle, affiche le graphe de la fonction d'appartenance du nombre flou en fonction de son noyau, de son support et de sa famille (self représente l'objet sur lequel la fonction est appelée).

```
n1.trace()
```



Les instances de cette classe dont les restrictions sont linéaires bénéficient de redéfinitions des opérations binaires +, -, *, /, et celles avec des restrictions quadratiques de redéfinitions des opérations binaires +, -. À noter qu'ici, comme le noyau n'est pas toujours réduit à un seul élément, les calculs qui se font normalement sur le mode s'effectuent sur un intervalle. Pour ce faire, nous utilisons l'arithmétique des intervalles explicitée dans [13] pour la multiplication des nombres flous trapézoïdaux.

Chaque méthode commence par vérifier si les familles des deux opérandes sont bien compatibles pour l'opération concernée en appelant les fonctions `famillesEgales(self,autre)` ou `famillesSymetriques(self,autre)`. Deux familles peuvent être égales et symétriques dans le cas d'une même famille simple, simplement égales, simplement symétriques ou incompatibles.

```
n1 = NombreFlou(3,4,2,1,"Lin","Lin")
n2 = NombreFlou(1,5,1,3,"Lin","Lin")
n3 = NombreFlou(-2,3,5,8,"Lin","Quad")
n4 = NombreFlou(5,8,1,6,"Quad","Lin")
```

```
(NombreFlou.famillesEgales(n1,n2),
 NombreFlou.famillesSymetriques(n1,n2))
```

```
(True, True)
```

```
(NombreFlou.famillesEgales(n2,n3),
 NombreFlou.famillesSymetriques(n2,n3))
```

```
(False, False)
```

```
(NombreFlou.famillesEgales(n3,n4),
 NombreFlou.famillesSymetriques(n3,n4))
```

```
(False, True)
```

Les redéfinitions des opérations + et * vérifient que les familles des opérandes sont bien égales, auquel cas elles opèrent les calculs sur les instances de la classe comme ils sont décrits à la section 1.4.2. Si les familles ne sont pas égales, un message d'erreur est renvoyé.

```
n1 = NombreFlou(-3,5,2,6,"Lin","Quad")
n2 = NombreFlou(2,3,4,5,"Lin","Quad")
n3 = NombreFlou(5,8,2,4,"Quad","Lin")
```

```
print(n1 + n2)
```

```
(-1,8,6,11), Lin - Quad
```

```
print(n2 + n3)
```

```
L'opération ne peut aboutir car les familles des opérandes sont différentes
None
```

Les redéfinitions des opérations unaires - et $\sim$ calculent et renvoient respectivement l'opposé et l'inverse des nombres flous passés en paramètre comme décrits à la section 1.4.2.

```
n1 = NombreFlou(3,4,2,5,"Quad","Lin")
print(n1)
```

```
(3,4,2,5), Quad - Lin
```

```
n1_op = -n1
print(n1_op)
```

```
(-4,-3,5,2), Lin - Quad
```

```
n1_op_op = -n1_op
print(n1_op_op)
```

```
(3,4,2,5), Quad - Lin
```

```
n1 = NombreFlou(6,8,3,7,"Quad","Lin")
print(n1)
```

```
(6,8,3,7), Quad - Lin
```

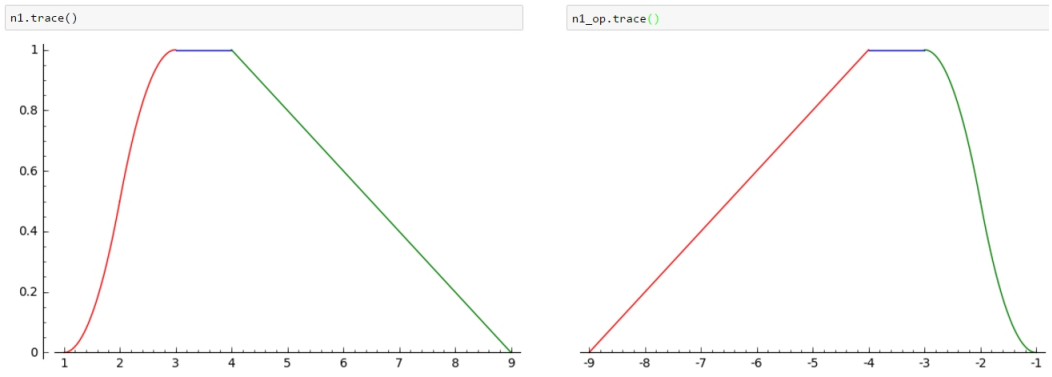
```
n1_inv = ~n1
print(n1_inv)
```

```
(1/8,1/6,7/64,1/12), Lin - Quad
```

```
n1_inv_inv = ~n1_inv
print(n1_inv_inv)
```

```
(6,8,3,7), Quad - Lin
```

Voici les graphes de deux nombres flous opposés :



Enfin, les redéfinition des opérations - et / vérifient que les familles des opérandes sont symétriques, auquel cas elles opèrent les calculs sur les instances de la classe comme ils sont décrits à la section 1.4.2. L'opérateur - calcule l'opposé de l'opérande de droite et l'additionne avec l'opérande de gauche (même principe

avec / et l'inverse). Si les familles ne sont pas symétriques, un message d'erreur est renvoyé.

```
n1 = NombreFlou(-3,5,2,6,"Lin","Quad")
n2 = NombreFlou(2,3,4,5,"Lin","Quad")
n3 = NombreFlou(5,8,2,4,"Quad","Lin")
```

```
print(n1 - n2)
```

```
L'opération ne peut aboutir car les familles des opérandes ne sont pas symétriques
None
```

```
print(n2 - n3)
```

```
(-6,-2,8,7), Lin - Quad
```

Les méthodes de classe `maxi(nombreFlou1,nombreFlou2)` et `mini(nombreFlou1,nombreFlou2)` renvoient respectivement le max et le min des nombres flous passés en paramètre comme décrit à la section 1.4.2, qui peuvent être un nombre flou différent. Ces fonctions induisent un ordre partiel sur les nombres flous, implantés dans les fonctions `pluspetit(self,autre)` et `plusgrand(self,autre)` qui utilisent respectivement maxi et mini pour ordonner partiellement les nombres flous (tous ne sont pas comparables). La fonction `egale(self, autre)` vérifie que toutes les propriétés des deux opérandes sont identiques afin de savoir si elles sont bien égales.

```
n1 = NombreFlou(-4,7,3,4,"Lin","Lin")
n2 = NombreFlou(2,7,1,0,"Lin","Lin")
```

```
maximum = NombreFlou.maxi(n1,n2)
print(maximum)
```

```
(2,7,1,4), Lin - Lin
```

```
NombreFlou.plusgrand(n1,n2)
```

```
Ces deux arguments ne sont pas comparables pour cette relation d'ordre partiel
```

```
minimum = NombreFlou.mini(n1,n2)
print(minimum)
```

```
(-4,7,3,4), Lin - Lin
```

```
NombreFlou.pluspetit(n1,n2)
```

```
True
```

```
NombreFlou.equivalent(n1,n2)
```

```
False
```

La fonction `forme_parametrique(self,A=None)` prend en entrée un nombre flou sous forme tuple et un anneau de polynômes univariés en une variable fixée r (par défaut l'anneau $\mathbb{Q}[r]$). Elle passe le nombre flou dans sa forme paramétrique comme décrit à la section 3.1 en calculant la fonction inverse en r de chaque restriction de la fonction d'appartenance (deux fonctions pour les restrictions de type quadratique). La fonction renvoie une instance de la classe `NombreFlouPM`, décrite plus loin, avec ces fonctions inverses en r, la famille du nombre flou en chaîne de caractère sous la forme "leftType - rightType", et l'anneau A si précisé.

```
n1 = NombreFlou(1,6,4,2,"Lin","Lin")
```

```
n1PM = n1.forme_parametrique()
print(n1PM)
```

```
[4*r - 3,-2*r + 8] , Lin - Lin
```

```
n2 = NombreFlou(2,4,1,3,"Lin","Quad")
```

```
n2PM = n2.forme_parametrique()
print(n2PM)
```

```
[r + 1,[3*sqrt(-1/2*r + 1/2) + 4, -3*sqrt(1/2)*sqrt(r) + 7]] , Lin - Quad
```

5.1.2 La classe NombreFlouRed

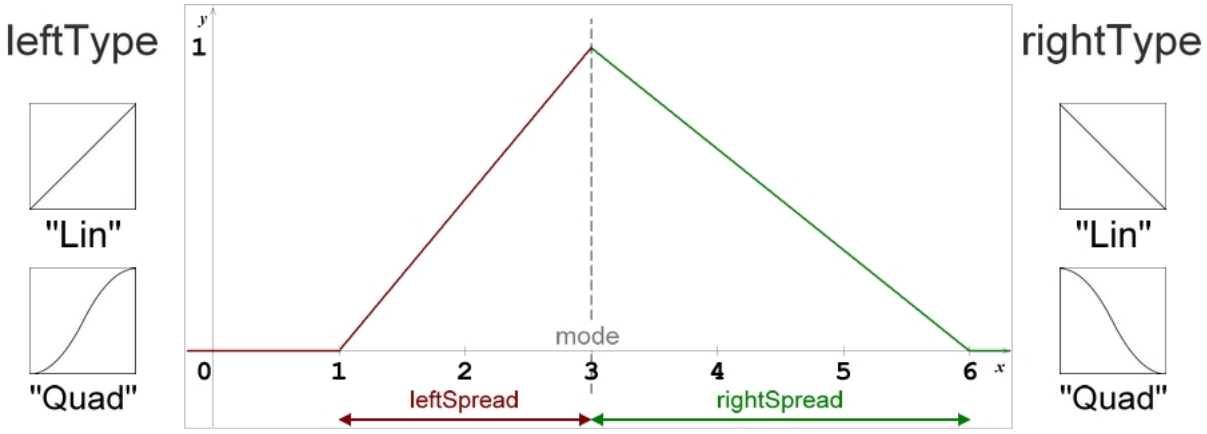
La classe `NombreFlouRed` hérite de la classe `NombreFlou`. Elle permet de modéliser dans la représentation en tuple des nombres flous avec des restrictions gauche et droite linéaires ou quadratiques. Ces nombres flous sont réduits, i.e. que leur noyau est réduit à un point qui est leur mode.

Le constructeur est `NombreFlouRed(mode, leftSpread, rightSpread, leftType, rightType)`.

Une instance de cette classe possède les cinq propriétés suivantes :

- mode (un entier) : le mode du nombre flou
- leftSpread (un entier positif ou nul) : la propagation du nombre flou à gauche du noyau
- rightSpread (un entier positif ou nul) : la propagation du nombre flou à droite du noyau
- leftType (une chaîne de caractères) : le type de la restriction de la fontion d'appartenance à gauche du noyau
- rightType (une chaîne de caractères) : le type de la restriction de la fontion d'appartenance à droite du noyau

On a donc ici leftMode = rightMode = mode.
Une instance peut être schématisée comme suit



Certaines fonctions sont redéfinies dans cette classe. Pour commencer, la fonction d'affichage `print(nombreFlou)` qui affiche le nombre flou sous la forme du triplet (mode, leftSpread, rightSpread) et précise sa famille.

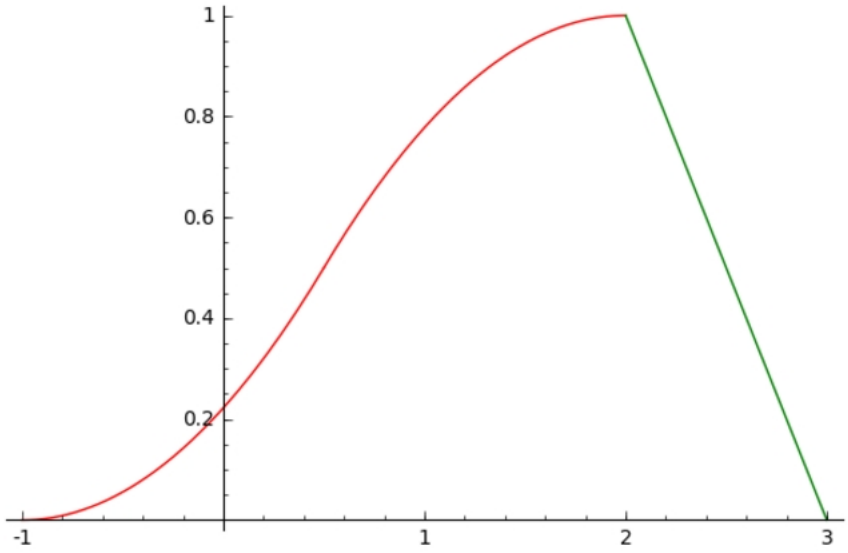
Les redéfinitions des opérations unaires - et $\sim$ sont aussi redéfinies pour prendre en compte le fait que le noyau est unique, et renvoient bien une instance de la classe `NombreFlouRed`.

Il en va de même avec les opérations binaires + et * dont les calculs sont exactement ceux vu en 1.4.2 avec le noyau réduit à un élément.

```
n1 = NombreFlouRed(2,3,1,"Quad","Lin")
print(n1)
```

```
(2,3,1), Quad - Lin
```

```
n1.trace()
```




```
n1_op = -n1
print(n1_op)
```

```
(-2,1,3), Lin - Quad
```

```
n2 = NombreFlouRed(4,3,3,"Quad","Lin")
```

```
res = n1 + n2
print(res)
```

```
(6,6,4), Quad - Lin
```

5.1.3 La classe NombreFlouPM

La classe `NombreFlouPM` permet de modéliser un nombre flou dans la représentation paramétrique.

Le constructeur est `NombreFlouPM(bas, haut, famille, A=None)`.

Une instance de cette classe possède les quatre propriétés suivantes :

- bas : la fonction inverse de la restriction gauche, en la variable para, qui donne pour chaque r dans [0,1] la borne gauche de la coupe-r correspondante
- haut : la fonction inverse de la restriction droite, en la variable para , qui

donne pour chaque r dans [0,1] la borne droite de la coupe-r correspondante
- famille (une chaîne de caractères) : la famille du nombre flou
- para : le générateur de l'anneau donné en paramètre (ou par défaut le générateur de l'anneau de polynômes à une variable sur le corps de rationnels).

Les attributs haut et bas sont chacun soit une fonction en *para* pour le cas triangulaire, soit un tableau de deux fonctions pour le cas quadratique.

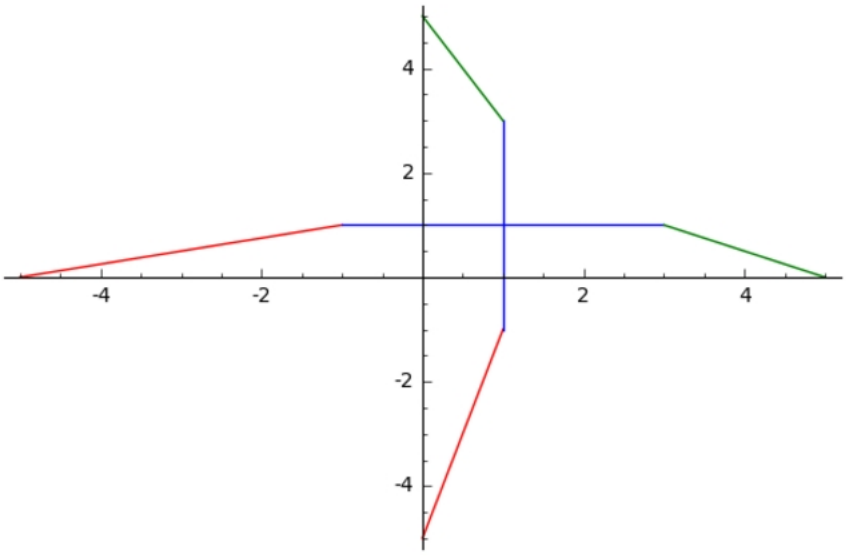
Dans cette classe, les différentes méthodes permettent également à la fois d'afficher et de dessiner les graphes de ses instances, et d'opérer des calculs sur ces instances et de les ordonner.

La fonction `print(nombreFlou)` permet d'afficher le nombre flou sous la forme d'une liste [bas, haut] et précise sa famille. La fonction `trace(self)` affiche la fonction d'appartenance du nombre flou en fonction des fonctions bas et haut. Le graphe obtenu est une rotation plane de 90ř par rapport à l'origine et une symétrie verticale par rapport à l'axe du mode du graphe de la fonction d'appartenance.

On le voit plus clairement en superposant les graphes des deux représentations d'un même nombre flou :

```
n1 = NombreFlou(-1,3,4,2,"Lin","Lin")
n1PM = n1.forme_parametrique()
```

```
n1.trace() + n1PM.trace()
```



La fonction `get_types(self)` renvoie, à partir de la famille du nombre flou paramétrique sous forme de chaîne de caractère "leftType - rightType", la liste de

ses types [leftType, rightType]. Cette fonction permet d'utiliser les fonctions `famillesEgales(self,autre)` ou `famillesSymetriques(self,autre)` exactement de la même façon que dans la classe `NombreFlou` pour vérifier la compatibilité des familles pour chaque opération.

Les redéfinitions des opérations binaires + et * (resp. - et /) peuvent alors vérifier que leurs opérandes ont bien des familles égales (resp. symétriques) avant d'effectuer les calculs de l'arithmétique floue vue en 3.1 pour la forme paramétrique.

On obtient ainsi un polymorphisme des opérations binaires + et - dans les deux représentations.

```
n1 = NombreFlou(1,2,2,1,"Lin","Quad")
n2 = NombreFlou(2,3,1,2,"Quad","Lin")
```

```
n3 = n1 - n2
print(res)
```

```
(-7,4,5,7), Lin - Quad
```

```
p1 = n1.forme_parametrique()
p2 = n2.forme_parametrique()
print(p1)
print(p2)
```

```
Symbolic Ring
[[2*sqrt(1/2)*sqrt(r) - 1, -2*sqrt(-1/2*r + 1/2) + 1],-r + 3] , Quad - Lin
[r + 1,[2*sqrt(-1/2*r + 1/2) + 3, -2*sqrt(1/2)*sqrt(r) + 5]] , Lin - Quad
```

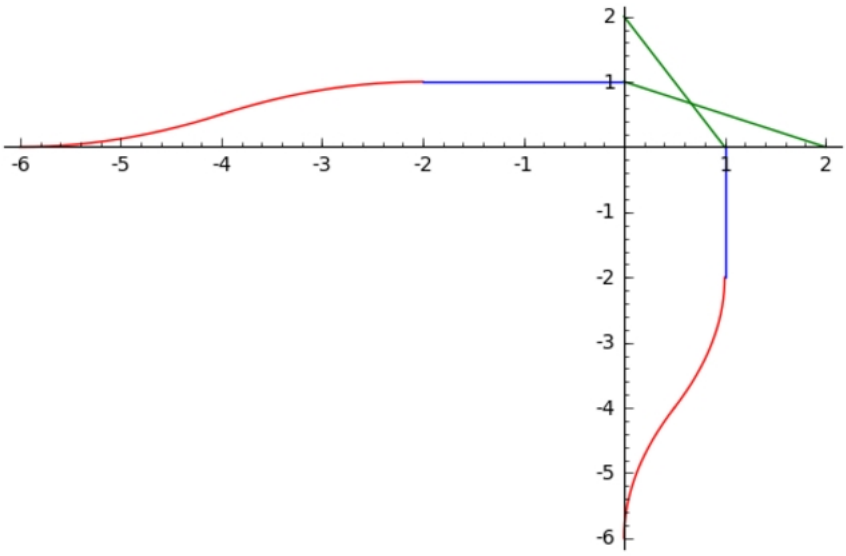
```
p3 = p1 - p2
print(p3)
```

```
[[4*sqrt(1/2)*sqrt(r) - 6, -4*sqrt(-1/2*r + 1/2) - 2],-2*r + 2] , Quad - Lin
```

```
n3PM = n3.forme_parametrique()
print(n3PM)
```

```
Symbolic Ring
[[4*sqrt(1/2)*sqrt(r) - 6, -4*sqrt(-1/2*r + 1/2) - 2],-2*r + 2] , Quad - Lin
```

```
n3.trace() + p3.trace()
```



Les méthodes de classe `maxi(nombreFlou1,nombreFlou2)` et `mini(nombreFlou1,nombreFlou2)` procèdent exactement comme les fonctions maxi et mini de la classe `NombreFlou`, en s'adaptant à la structure des nombres flous paramétriques, et induisent un ordre partiel sur les nombres flous. La fonction `egale(self, autre)` vérifie que les propriétés des instances sont identiques, et les fonctions `plusgrand(self,autre)` et `pluspetit(self,autre)` utilisent respectivement maxi et mini pour ordonner partiellement les nombres flous (tous ne sont pas comparables).

Un ordre total sur les nombres flous dans la forme paramétrique est donné via des calculs d'intégrales dans [8]. Cette méthode a été implantée dans la fonction `R(self, autre)` et est décrite ci-dessous :

Soit un nombre flou u de forme paramétrique $[\underline{u}(r), \overline{u}(r)]$, avec $0 \leq r \leq 1$, on définit :

$$Mag(u) = \frac{1}{2}\sum_{0}^{1}(\underline{u}(r) + \overline{u}(r) + \underline{u}(1) + \overline{u}(1))f(r)dr,$$

$$Momag(u) = \frac{1}{2}\sum_{0}^{1}(\underline{u}(r) - \overline{u}(r) + \underline{u}(1) - \overline{u}(1))dr$$

où la fonction poids $f(\alpha)$ est positive et croissante sur $[0, 1]$ avec $f(0) = 0$ et $\sum_0^1 f(\alpha)d\alpha = \frac{1}{2}$. Ici, on utilise $f(r) = r$.

Ces deux calculs sont effectués par les méthodes `momag(self)` et `momag(self)`.

À partir de 2 nombres flous sous forme paramétrique a et b, la méthode `R(self, autre)` calcule les quantités

$$R(a,\lambda) = Mag(a) + \lambda \cdot Momag(a) \text{ et } R(b,\lambda) = Mag(b) + \lambda \cdot Momag(b),$$

où

$$\lambda = \begin{cases} 0 & \text{si } Mag(a) \neq Mag(b), \\ 1 & \text{si } Mag(a) = Mag(b) \text{ et } z >= 0, \\ -1 & \text{si } Mag(a) = Mag(b) \text{ et } z < 0. \end{cases}$$

avec $z = \frac{a.bas(1)+a.haut(1)}{2}$, et on renvoie ces deux valeurs dans un tuple.

Alors, pour deux nombres flous a et b, on définit le rang de a et b comme suit :

- $R(a,\lambda) > R(b,\lambda) \Leftrightarrow a > b$,
- $R(a,\lambda) < R(b,\lambda) \Leftrightarrow a < b$,
- $R(a,\lambda) = R(b,\lambda) \Leftrightarrow a \sim b$.

5.2 Méthodes de résolution algébrique

Les méthodes listées ci-après sont celles implantées pour la résolution algébrique de systèmes de polynômes réels décrite à la section 2, entrant dans la mise en oeuvre de l'algorithme de Wu.

`classe(p)` : Retourne la classe d'un polynôme (le maximum parmi les indices des variables apparaissant dans p)

`degre(p)` : Retourne le degré d'un polynôme (le degré maximum en la variable dont l'indice est la classe de p, i.e. en la variable principale de p)

`init(p)` : Retourne le coefficient dominant en la variable principale de p

`tail(p)` : Renvoie la queue d'un polynôme p, i.e. p - son terme dominant

`estConstant(p,i)` : Renvoie True si p est constant en la variable d'indice i, et False sinon

`ordreRitt(a,b)` : Renvoie 1 si $a > b$, 0 si $a = b$, -1 si $a < b$ pour l'ordre de Ritt

`estReduit(a,b)` : Renvoie True si a est réduit par rapport à b, et False sinon

`basicSet(F)` : Calcule un ensemble basique d'un ensemble de polynômes F

`pseudoDiv(p,f)` : Effectue la pseudo-division de p par f et renvoie la liste $[quotient, reste, init(f), init(f)^j]$

`premEnsTri(p,T)` : Effectue la pseudo-division de p par tous les polynômes de l'ensemble T et renvoie le reste

`PREM(F,B)` : Effectue la pseudo-division de chaque polynôme de F par tous les polynômes de l'ensemble T et renvoie la liste des restes

`CharacSet(F)` : Calcule l'ensemble caractéristique de l'ensemble de polynômes F

`Wu(F)` : Algorithme de Wu : Calcule l'ensemble des ensembles caractéristiques à partir du système F

`prodInit(B)` : Renvoie le produit des initiaux des polynômes de l'ensemble B

`ensInit(Z)` : Renvoie la liste des produits des initiaux pour chaque ensemble de Z

5.3 Algorithme principal

Pour commencer, l'utilisateur crée l'anneau dans lequel se trouvent la ou les variables de son système. Il recupère la liste des générateurs de l'anneau afin de pouvoir nommer ses variables dans Sage et récupérer le nombre de variables.

Il crée ensuite un anneau de polynômes univariés B à coefficient dans le premier anneau.

Il entre les équations de son système dans une liste, chaque équation étant représentée sous la forme d'un couple. L'équation $A == B$ s'écrit donc (A, B).

La fonction `solveFuzzyPolynomialSystem(syst_depart)` prend en entrée ce système de départ et va renvoyer les solutions exactes.

Pour ce faire, elle fait d'abord appel à la fonction `syst_collecte(syst_dep)`, qui calcule à partir du système de départ le système tranché collecté. Pour cela,

on passe les membres de chaque équation sous forme paramétrique dans l'anneau B si ce n'est pas déjà fait pour obtenir les équations du système tranché et on procède à l'identification. Les coefficients en le générateur de B sont récupérées et on obtient ainsi les polynômes du système tranché collecté, renvoyés dans une liste.

On revient dans la fonction principale où cette liste est récupérée puis passée en paramètre de la fonction `Wu(F)`.
La méthode `Wu(F)` est l'algorithme de Wu qui prend en entrée une liste de polynômes réels et renvoie l'ensemble des ensembles caractéristiques sous forme de liste comme l'explique l'algorithme à la section 2.2.

Cette liste de liste est récupéré dans la variable `resultat` et la fonction principale renvoie à l'utilisateur le tuple `(resultat, ensInit(resultat))`.

L'utilisateur n'a plus qu'à passer ces variables en symbolique et utiliser la fonction `solve` pour retrouver les solutions grâce au principe de Wu.

5.4 Tests

Des fonctions `print` ont été appelées dans les différentes méthodes afin de pouvoir suivre correctement le déroulement des algorithmes.

5.4.1 Test 1

On reprend le système de l'exemple 4.2.

```
A = PolynomialRing(QQ, 'x,y',order ='invlex')
B = A['r']

var = list(A.gens())
n = len(var)
x = var[0]
y = var[1]
x<y
```

```
True
```

```
dep=[(NombreFlouRed(1,0,0,"Lin","Lin")*x + NombreFlouRed(-1,1,1,"Lin","Lin"), NombreFlouRed(-2,1,1,"Lin","Lin")*y**2),\
     (NombreFlouRed(1,0,0,"Lin","Lin")*x + NombreFlouRed(3,1,1,"Lin","Lin"), NombreFlouRed(2,1,1,"Lin","Lin")*y**2)]
```

```
results = solveFuzzyPolynomialSystem(dep)
```

```
Le système tranché collecté est composé des polynômes suivants :
[-3*y^2 + x + 4, -y^2 + 1, -y^2 + x + 2, y^2 - 1, y^2 + x, 3*y^2 + x - 2]
Calcul de l'ensemble des ensembles caractéristiques du système tranché collecté :
D =
[[-3*y^2 + x + 4, -y^2 + 1, -y^2 + x + 2, y^2 - 1, y^2 + x, 3*y^2 + x - 2]]
Calcul de l'ensemble caractéristique :
S =
[-3*y^2 + x + 4, -y^2 + 1, -y^2 + x + 2, y^2 - 1, y^2 + x, 3*y^2 + x - 2]
Calcul d'un ensemble basique :
L'élément de rang minimal est -3*y^2 + x + 4
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à y^2 - 1/3*x - 4/3, on obtient l'ensemble []
```

```
Basic Set B =
[y^2 - 1/3*x - 4/3]
PREM(F,B) =
[0, x + 1, x + 1, x + 1, x + 1, x + 1]
S ∪ PREM (F,B)\{0} =
[x + 1, y^2 - 1, -3*y^2 + x + 4, -y^2 + 1, y^2 + x, 3*y^2 + x - 2, -y^2 + x + 2]
Calcul d'un ensemble basique :
L'élément de rang minimal est x + 1
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x + 1, on obtient l'ensemble [y^2 - 1, -y^2 + 1]
L'élément de rang minimal est y^2 - 1
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à y^2 - 1, on obtient l'ensemble []
Basic Set B =
[x + 1, y^2 - 1]
retour dans la fonction CharacSet
Charac Set =
[x + 1, y^2 - 1]
retour dans la fonction Wu
```

```
print(results)
```

```
([[x + 1, y^2 - 1]], [1])
```

```
var('x'), var('y')
solve([x + 1 ==0, y^2 - 1==0],x,y)
```

```
[[x == -1, y == 1], [x == -1, y == -1]]
```

5.4.2 Test 2

Voici un second exemple un peu plus ardu. On part du système

$$dep : \begin{cases} (2,1,1)xy + (3,1,1)x^2y^2 + (2,1,1)x^3y^3 = (7,3,3), \\ (5,1,1)xy + (2,3,1)x^2y^2 + (2,2,1)x^3y^3 = (9,6,3) \end{cases}$$

```
A = PolynomialRing(QQ, 'x,y',order ='invlex')
B = A['r']

var = list(A.gens())
n = len(var)
x = var[0]
y = var[1]
x<y
```

```
True
```

```
dep=[(NombreFlouRed(2,1,1,"Lin","Lin")*x*y + NombreFlouRed(3,1,1,"Lin","Lin")*x**2*y**2 + NombreFlouRed(2,1,1,"Lin","Lin")*x**3*y**3,\
      NombreFlouRed(7,3,3,"Lin","Lin")), (NombreFlouRed(5,1,1,"Lin","Lin")*x*y + NombreFlouRed(2,3,1,"Lin","Lin")*x**2*y**2\
      + NombreFlouRed(2,2,1,"Lin","Lin")*x**3*y**3, NombreFlouRed(9,6,3,"Lin","Lin"))]
```

```
results = solveFuzzyPolynomialSystem(dep)
```

```
Le système tranché collecté est composé des polynômes suivants :
[-x^2*y^2 + 4*x*y - 3, -x^3*y^3 - x^2*y^2 - x*y + 3, x^3*y^3 + x^2*y^2 + x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6, 3*x^3*y^3 + 3*
x^2*y^2 + 6*x*y - 12, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10]
Calcul de l'ensemble des ensembles caractéristiques du système tranché collecté :
D =
[[-x^2*y^2 + 4*x*y - 3, -x^3*y^3 - x^2*y^2 - x*y + 3, x^3*y^3 + x^2*y^2 + x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6, 3*x^3*y^3 + 3
*x^2*y^2 + 6*x*y - 12, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10]]
Calcul de l'ensemble caractéristique :
S =
[-x^2*y^2 + 4*x*y - 3, -x^3*y^3 - x^2*y^2 - x*y + 3, x^3*y^3 + x^2*y^2 + x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6, 3*x^3*y^3 + 3*
x^2*y^2 + 6*x*y - 12, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10]
Calcul d'un ensemble basique :
L'élément de rang minimal est -x^2*y^2 + 4*x*y - 3
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x^2*y^2 - 4*x*y + 3, on obtient l'ensemble []
Basic Set B =
[x^2*y^2 - 4*x*y + 3]



PREM(F,B) =
[0, x^5*y - x^4, x^5*y - x^4, x^5*y - x^4, x^5*y - x^4, x^5*y - x^4, x^5*y - x^4]
S U PREM (F,B)\{0} =
[x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*y^2 + 6*x*
y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]
Calcul d'un ensemble basique :
L'élément de rang minimal est x^5*y - x^4
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x^5*y - x^4, on obtient l'ensemble []
Basic Set B =
[x^5*y - x^4]
retour dans la fonction CharacSet
Charac Set =
[x^5*y - x^4]
retour dans la fonction Wu
D =
[[x^5, x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*y^2
 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]]
Calcul de l'ensemble caractéristique :
S =
[x^5, x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*y^2 +
 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]
Calcul d'un ensemble basique :
L'élément de rang minimal est x^5
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x^5, on obtient l'ensemble [-x^2*y^2 + 4*x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x
^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*y^2 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]
L'élément de rang minimal est -x^2*y^2 + 4*x*y - 3
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x^2*y^2 - 4*x*y + 3, on obtient l'ensemble []
Basic Set B =
[x^5, x^2*y^2 - 4*x*y + 3]
PREM(F,B) =
[0, 0, x^4, x^4, x^4, x^4, x^4, x^4, x^4]
S U PREM (F,B)\{0} =
[x^4, x^5, x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*
y^2 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]
```

```
Calcul d'un ensemble basique :
L'élément de rang minimal est x^4
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x^4, on obtient l'ensemble [-x^2*y^2 + 4*x*y - 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x
^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*y^2 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]
L'élément de rang minimal est -x^2*y^2 + 4*x*y - 3
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x^2*y^2 - 4*x*y + 3, on obtient l'ensemble []
Basic Set B =
[x^4, x^2*y^2 - 4*x*y + 3]
retour dans la fonction CharacSet
Charac Set =
[x^4, x^2*y^2 - 4*x*y + 3]
retour dans la fonction Wu
D =
[[x^2, x^4, x^5, x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^2*y^2 - 4*x*y + 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 +
 x*y - 3, 3*x^3*y^3 + 3*x^2*y^2 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]]
Calcul de l'ensemble caractéristique :
S =
[x^2, x^4, x^5, x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^2*y^2 - 4*x*y + 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 + x
*y - 3, 3*x^3*y^3 + 3*x^2*y^2 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]
Calcul d'un ensemble basique :
L'élément de rang minimal est x^2
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x^2, on obtient l'ensemble []
Basic Set B =
[x^2]
PREM(F,B) =
[0, 0, 0, 0, x*y - 6, x*y - 4, x*y - 10/3, x*y - 3, x*y - 3, x*y - 2, x*y - 3/4, x*y - 3/4]
S ∪ PREM (F,B)\{0} =
[x^2, x^4, x^5, x*y - 2, x*y - 6, x*y - 10/3, x*y - 4, x*y - 3, x*y - 3/4, x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^2*y^2 - 4*x*y + 3, x^3*y^3 + 2*x^2*y^2 + x*y -
 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 + x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*y^2 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x
*y - 6]
Calcul d'un ensemble basique :
L'élément de rang minimal est x^2
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x^2, on obtient l'ensemble [x*y - 2, x*y - 6, x*y - 10/3, x*y - 4, x*y - 3, x*y - 3/
4]
```

```
L'élément de rang minimal est x*y - 2
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x*y - 2, on obtient l'ensemble []
Basic Set B =
[x^2, x*y - 2]
retour dans la fonction CharacSet
Charac Set =
[x^2, x*y - 2]
retour dans la fonction Wu
D =
[[x, x^2, x^4, x^5, x*y - 2, x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^2*y^2 - 4*x*y + 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3
 + x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*y^2 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]]
Calcul de l'ensemble caractéristique :
S =
[x, x^2, x^4, x^5, x*y - 2, x^5*y - x^4, -x^2*y^2 + 4*x*y - 3, x^2*y^2 - 4*x*y + 3, x^3*y^3 + 2*x^2*y^2 + x*y - 4, 3*x^3*y^3 + 4*x^2*y^2 + 3*x*y - 10, x^3*y^3 +
 x^2*y^2 + x*y - 3, 3*x^3*y^3 + 3*x^2*y^2 + 6*x*y - 12, -x^3*y^3 - x^2*y^2 - x*y + 3, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6]
Calcul d'un ensemble basique :
L'élément de rang minimal est x
On ne garde dans l'ensemble que les éléments qui sont réduits par rapport à x, on obtient l'ensemble []
Basic Set B =
[x]
PREM(F,B) =
[0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1]
retour dans la fonction Wu
```

```
print(results)
```

```
([[x^5*y - x^4], [x^4, x^2*y^2 - 4*x*y + 3], [x^2, x*y - 2]], [x^5, x^2, x])
```

```
var('x'), var('y')
solve([x^5*y - x^4==0],x,y)
```

```
([x == (1/y)], [1])
```

L'ensemble des solutions est donc $\{(x = \frac{1}{y}, y) | y \in \mathbb{R}\backslash\{0\}\}$.

6 Conclusion

Ce mémoire a présenté les différentes étapes permettant de trouver toutes les solutions d'un système de polynômes à coefficients flous triangulaires. Les nombres flous et leurs représentations gauche-droite ont été étudiés afin de comprendre comment les algébriser. Pour atteindre cet objectif, nous faisons intervenir la forme paramétrique des nombres flous. Cette représentation a pu être abordée car, si ces nombres sont flous, leur représentation elle, est formelle.

Une fois cette algébrisation effectuée, c'est un algorithme basé sur l'algorithme de résolution algébrique de systèmes d'équations polynomiales de Wu Wen Tsun qui entre en jeu. Cet algorithme nous amène à des systèmes triangulaires qui sont facile à résoudre. Toutes les solutions peuvent être obtenues simultanément. De plus, il n'est pas nécessaire d'isoler les solutions en fonction de la valeur du paramètre r.

La bibliothèque Fuzzy résultant de ce travail a aussi été présentée et des exemples, ainsi que des tests, ont pu être réalisés par son biais.

Nous avons également initié une exploration de la représentation paramétrique des nombres flous quadratiques, et il sera intéressant pour la suite de poursuivre l'étude de ces nombres avec Annick Valibouze et Philippe Aubry dans le but d'écrire un article et d'utiliser cette représentation afin de résoudre des systèmes de polynômes à coefficients flous quadratiques.

Cela impliquera un développement de la bibliothèque Fuzzy qui pourra par la suite être inclue dans Sage.

Références

[1] Saeid Abbasbandy and B Asady. Newton's method for solving fuzzy nonlinear equations. *Applied Mathematics and Computation*, 159(2) :349–356, 2004.

[2] Saeid Abbasbandy and Mahmood Otadi. Numerical solution of fuzzy polynomials by fuzzy neural network. *Applied Mathematics and Computation*, 181(2) :1084–1089, 2006.

[3] Philippe Aubry. *Ensembles triangulaires de polynomes et resolution de systemes algebriques. Implantation en axiom.* 1999.

[4] Marziyeh Boroujeni, Abdolali Basiri, Sajjad Rahmany, and Annick Valibouze. Finding solutions of fuzzy polynomial equations systems by an algebraic method. *Journal of Intelligent & Fuzzy Systems*, 30(2) :791–800, 2016.

[5] James J Buckley, Thomas Feuring, and Yoichi Hayashi. Solving fuzzy equations using evolutionary algorithms and neural nets. *Soft Computing*, 6(2) :116–123, 2002.

[6] Shang-Ching Chou and Xiao-Shan Gao. Ritt-wu's decomposition algorithm and geometry theorem proving. In *International Conference on Automated Deduction*, pages 207–220. Springer, 1990.

[7] Didier Dubois and Henri Prade. Operations on fuzzy numbers. *International Journal of systems science*, 9(6) :613–626, 1978.

[8] R Ezzati, S Khezerloo, and S Ziari. Application of parametric form for ranking of fuzzy numbers. *Iranian Journal of Fuzzy Systems*, 12(1) :59–74, 2015.

[9] Brian R. Gaines and Ladislav J. Kohout. The fuzzy decade : A bibliography of fuzzy systems and closely related topics. *International Journal of Man-Machine Studies*, 9 :1–68, 1977.

[10] Sanjay Jain. Close interval approximation of piecewise quadratic fuzzy numbers for fuzzy fractional program. *Iranian Journal of Operations Research*, 2(1) :77–88, 2010.

[11] Ali Abbasi Molai, Abdolali Basiri, and Sajjad Rahmany. Resolution of a system of fuzzy polynomial equations using the gröbner basis. *Information Sciences*, 220 :541–558, 2013.

[12] Luciano Stefanini and Laerte Sorini. Fuzzy arithmetic with parametric lr fuzzy numbers. In *IFSA/EUSFLAT Conf.*, pages 600–605, 2009.

[13] A Taleshian and S Rezvani. Multiplication operation on trapezoidal fuzzy numbers. 2011.