SORBONNE UNIVERSITÉ

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Jérémy Marrez

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Représentations adaptées à l'arithmétique modulaire et à la résolution de systèmes flous

Thèse dirigée par Jean-Claude Bajard
et co-encadrée par Lokmane Abbas-Turki

soutenue publiquement le 6 décembre 2019

après avis des **rapporteurs** :

Mme Marine Minier	Professeure, Université de Lorraine
M. Clément Pernet	Maître de conférences HDR, Université Grenoble Alpes

devant le **jury** composé de :

M. Lokmane Abbas-Turki	Maître de conférences, Sorbonne Université
M. Jean-Claude Bajard	Professeur, Sorbonne Université
M. Louis Goubin	Professeur, UVSQ
Mme Marine Minier	Professeure, Université de Lorraine
M. Clément Pernet	Maître de conférences HDR, Université Grenoble Alpes
Mme Annick Valibouze	Professeure, Sorbonne Université

Remerciements

Je tiens à remercier en premier lieu Jean-Claude, mon directeur de thèse, de m'avoir guidé durant ces trois années. Ta bienveillance, tes conseils avisés et ton aptitude à me montrer comment une mer tumultueuse devient une nouvelle zone d'exploration m'ont permis de garder le cap. Cette thèse n'aurait pu se faire sans ta confiance et ta patience. Pour les nombreuses opportunités offertes et toutes les rencontres qui en ont découlées, je te remercie. Je remercie aussi Lokmane de m'avoir fait répéter et de m'avoir fait confiance, en me permettant d'assurer mon premier TD dans son cours de programmation parallèle.

Je remercie chaleureusement les rapporteurs de cette thèse, Marine Minier et Clément Pernet, pour leur considération, le temps consacré à la lecture du présent manuscrit, ainsi que leurs retours. Je tiens aussi à remercier Louis Goubin d'avoir accepté de faire partie de mon jury.

Je remercie également les deux personnes qui m'ont fait découvrir la recherche à la fin de mon master : Annick et Philippe. Je n'aurais pas imaginé que la vie au sein d'un laboratoire puisse être aussi agréable et stimulante avant votre rencontre. Votre enthousiasme et votre sens de la rigueur m'ont beaucoup apporté. Merci Annick d'avoir gardé un oeil sur moi et d'avoir surveillé que tout se passait bien, et surtout merci pour ta bienveillance et ton humour.

Un grand merci à l'équipe ALMASTY et aux personnes que j'y ai rencontrées : Jérôme, Natasha, Anand, Vincent, Lucas, Thomas E., Andrea et Damien. Jérôme merci pour ta disponibilité, nos discussions ont permis d'éclaircir de nombreuses incertitudes, qu'elles soient d'ordre scientifique ou administratif, et ton rire de les rendre moins pesantes. Merci à Anand, Natasha, Lucas, Vincent et Thomas E. pour votre bonne humeur, quand on vit des étapes similaires à plusieurs, les franchir semble plus facile.

Dans la première équipe qui m'a accueilli et que je retrouve cette année, l'équipe APR, je remercie aussi : Racha, Antoine, Maryse, Ghiles, Abdelraouf, Matthieu et Irphane.

Racha, tu as illuminé de nombreux après-midi par ta gentillesse et ton soutien. La curiosité artistique qui accompagne ton attrait pour les sciences a été source d'une grande respiration. Cet intérêt commun a été l'occasion de très bons moments devant les toiles de Dali et Picabia.

Merci aussi à tous ceux rencontrés durant ces recherches. L'équipe de Toulon tout d'abord ; Laurent-Stéphane, Pascal, Yssouf, Nadia, Asma et Fabien. Yssouf, nos pérégrinations dans les rues et les restaurants toulonnais et bretons ont inscrit autant d'agréables souvenirs. Ta bonne humeur et nos conversations ont été fortes stimulantes. Nadia, je te remercie pour tes conseils avisés et ton enthousiasme.

J'en viens maintenant à l'équipe de Lisbonne ; thank you all for your warm welcome and thank you Paulo for your quick rereading of my codes, and for making me travel through Lisbon. These moments remain engraved as an endless afternoon of sloping alleys, medieval castle and ice cream. Enfin, merci à Thomas P., Arnaud et Cédric, trois français rencontrés en Australie, avec une connaissance pointue aussi bien des réseaux que de la faune locale, et dont la compagnie a été plus qu'agréable, jusque dans les falaises de la baie de Wollongong.

Je remercie bien sûr tous ceux qui m'ont permis d'enseigner, une envie qui me démange depuis toujours. Je pense notamment à Valérie, Karine, Mathieu, Antoine G., Antoine M., Béatrice, Lokmane, Emmanuel, Romain et Bihn. Une pensée pour les étudiants qui m'ont amené à considérer de nouvelles façons de penser et de voir des conceps acquis de longue date. Pour le temps consacré à m'écouter répéter mes présentations et à relire mes articles, Binh, je te remercie.

Pour finir, je tiens à remercier ceux qui me soutiennent depuis le début, ma famille. Merci à mes parents, mes cousins, mes oncles et tantes, et mes amis dont la présence m'a accompagné dans les bons moments mais aussi dans les moments plus compliqués, et dont le soutien permanent a été et reste un outil puissant dans cette grande étape.

Table des matières

Introduction

1 Contexte historique et motivations

Garder une information secrète est depuis toujours une préoccupation majeure pour l'Homme. Le jour où un potier babylonien voulut conserver la recette du vernis qui faisait son succès, sans pour autant la révéler, il y a plus de 3000 ans, il entreprit de graver la dîte recette sur une tablette d'argile en y retirant des consonnes, obtenant ainsi une version codée [Kah96]. Il s'agit de la première technique connue de *cryptologie*, c'est-à-dire l'art du secret, érigé au rang de science depuis les années 70 avec l'avènement de l'informatique.

La volonté de rendre secrète une information a rapidement été rejointe par celle de casser le secret pour retrouver l'information, et ce, sans y avoir été invité. Ces deux objectifs ont installé un rapport de force et ont fait émerger deux disciplines aux intérêts opposés ; la *cryptographie* et la *cryptanalyse*, chacune basée sur des techniques qui lui sont propres. En prenant pour comparaison le conflit de la Guerre de Troie, la cryptographie est au Troyens ce que la cryptanalyse est aux Achéens. L'enjeu que représentait la prise d'une cité réside ici dans la découverte d'un message secret.

La cryptographie a pour objectif de protéger une communication entre plusieurs entités. Ces entités peuvent rendre leurs messages confidentiels en les chiffrant par le biais d'un secret ou d'une *clé*. Personne d'autre n'est en mesure de connaître le message sans la clé. La cryptanalyse en revanche est une technique basée sur l'attaque. L'objectif est de retrouver le message clair à partir du chiffré sans posséder la clé de chiffrement. Ces attaques peuvent s'effectuer par force brute, c'est-à-dire en essayant toutes les combinaisons possibles de mots de passe ou de clés, ou bien en usant de ruse comme avec les *attaques par canaux cachés*. Ces dernières profitent de certaines fuites observées lors d'exécutions successives d'un algorithme avec la même clé, indépendantes de la sécurité "mathématique" théorique, comme le temps d'exécution [Koc96], le bruit ou encore la consommation électrique [KJJ99, Gou03]. Ces fuites s'opèrent principalement au niveau des points de mémorisation qui passent de 0 à 1 durant les calculs, et dépendent clairement des représentations des données.

Au fil de l'Histoire, les innovations en matière de chiffrement et de cryptanalyse furent motivées par de nombreux enjeux, sans cesse renouvelés. À la Renaissance, le désir d'atteindre la perfection, morale comme artistique, a poussé plusieurs nobles et intellectuels à se pencher sur la cryptologie. Leon Battista Alberti est considéré comme le père de la cryptographie occidentale car il a fourni la plus ancienne théorie occidentale de cryptanalyse, l'invention de la substitution polyalphabétique, et l'invention du code de chiffrement [Kah96]. Fasciné par les secrets et les mystères, Della Porta a écrit un ouvrage sur les chiffrements des messages et la cryptanalyse [DP$^+$, Kah80].

Dans le Grand Siècle qui a suivi, l'enjeu est surtout diplomatique. Le mathématicien Antoine Rossignol est entré au service de Louis XIV et a conçu un chiffrement par substitution, appelé le Grand Chiffre, pour chiffrer la correspondance du Roi et les archives au contenu sensible [Sin03]. Le chiffrement est également utilisé par les civils au XIXème siècle, comme les frères Joseph et François Blanc qui se tenaient informés avant tout le monde des cours de la rente à la Bourse de Paris en transmettant leur propres codes sur le télégraphe Chappe qu'ils avaient piraté [Hea00].

Lors des deux conflits majeurs du XXème siècle, la cryptologie a joué un rôle tactique prédominant dans une course contre la montre. Durant la Première Guerre mondiale, l'État-major français a ainsi pu décrypter les messages du ministre allemand Zimmermann, ce qui poussa les États-Unis a entrer en guerre et marqua une victoire remarquable pour la cryptanalyse [Sin03]. Exploit renouvelé durant la guerre qui suivit, où les Allemands utilisèrent la machine Enigma inventée par Arthur Scherbiusles pour chiffrer leurs communications. Les Britanniques ripostèrent en construisant des machines spécialisées, appelées « Bombes », réalisées d'après les plans d'Alan Turing, qui les aidèrent à casser ce chiffrement. La cryptanalyse permit de mettre fin à la guerre plus tôt que prévu et la cryptologie fut reconnue par Churchill comme l'une des clés de la victoire.

Cette guerre a poussé Claude Shannon à s'intéresser au chiffrement des messages, et son article « A Mathematical Theory of Communication » publié en 1949 a permis à la cryptographie de passer progressivement du rang d'art à celui de science [Sha48]. Pour protéger les communications, d'autres propriétés sont apparues en plus du chiffrement ; pour garantir *l'intégrité* du message, c'est-à-dire s'assurer que ce dernier n'ait pas été altéré durant l'envoi, et pour *authentifier* son auteur, la signature numérique a été introduite.

Jusque dans les années 70, on parle de cryptographie *symétrique* ; deux personnes souhaitant chiffrer leurs messages utilisent la même clé pour chiffrer et déchiffrer. Le problème majeur réside dans la distribution des clés avant la communication et leur pro-

tection. Cette exigence est d'autant plus difficile à satisfaire lorsqu'un grand nombre de personnes est impliqué. De plus, si un expéditeur souhaite communiquer avec plusieurs destinaires, il doit échanger une clé différente avec chacun d'eux afin d'assurer la confidentialité de chaque communication. La solution est apportée par W. Diffie et M. Hellman qui proposent en 1976 un nouveau concept ; la cryptographie *asymétrique* [DH76].

Chaque participant possède une paire de clés, liées entre elles. Une *clé privée* qu'il est le seul à connaître et qui lui permet de déchiffrer les messages qu'on lui envoie, et une *clé publique*, connue de tous, qui permet aux autres de chiffrer les messages qui lui sont destinés. Dans cette configuration, avec une paire de clés par personne, chacun est capable de communiquer de façon confidentielle avec chaque autre, en distribuant sa clé publique sans problème de confidentialité. Le point important réside dans le lien mathématique entre les deux clés ; si la clé publique est obtenue facilement à partir de la clé privée, retrouver la clé privée à partir de la clé publique doit être difficile au sens calculatoire du terme. Aucun algorithme ne doit pouvoir résoudre ce problème en temps polynomial.

Lors du chiffrement et du déchiffrement, la plupart des cryptosystèmes asymétriques effectuent des exponentiations modulaires ; à partir d'un nombre g et d'une puissance e donnés, l'exponentiation modulaire renvoie le reste obtenu dans la division euclidienne de g^e par une constante connue p, appelée le modulo. Ces exponentiations modulaires sont réalisées grâce à des multiplications modulaires. Pour réaliser cette étape, à partir de deux nombres a et b donnés, la majorité des algorithmes commencent par calculer le produit ab, puis effectuent une *réduction modulaire* pour récupérer le reste r de ab modulo p. On note alors

$$r = ab + qp, \quad \text{avec } r < p.$$

La sécurité de la cryptographie asymétrique repose sur un problème calculatoire facile dans un sens et difficile dans l'autre. Le premier cryptosystème de chiffrement, proposé par Rivest Shamir et Adelman en 1978 et nommé RSA [RSA78], repose sur le problème de la factorisation. Multiplier deux nombres p et q pour obtenir le modulo $n = pq$ est facile, mais retrouver p et q en factorisant leur produit n est difficile. Un attaquant capable de factoriser ce produit peut retrouver le message chiffré par cette méthode. Ce problème majore la sûreté du protocole RSA, bien que les cryptanalystes profitent d'autres propriétées du RSA pour concevoir des attaques sous-exponentielles. De nombreuses attaques sont dues à de mauvais choix d'implémentation ou de protocoles. Par exemple, en choisissant deux clés publiques $(n = pq, e)$ et $(n' = pq', e')$, où e et e' sont les exposants choisis pour l'exponentiation modulaire du chiffrement, le calcul de $\text{pgcd}(n, n') = p$ permet de factoriser n et n', compromettant ainsi les deux clés. Dans un autre cas, lorsque trois personnes reçoivent chacune un chiffré du même message, et que l'exposant e de leur clé

publique est petit (égal à 3), il est aisé de retrouver le message clair en interceptant les trois chiffrés. Pour RSA, le modulo n est fixé par le protocole avec une taille de 1024 bits au moins, obligeant les cryptanalystes à effectuer les 2^{76} opérations recommandées pour considérer un cryptosystème comme sûr. L'exponentiation modulo n doit être rapide.

Les protocoles d'échange des clés Diffie Hellman et de chiffrement ElGamal [DH76, ElG85a], du nom de leurs auteurs, et proposés respectivement en 1976 et 1984, reposent sur le problème du logarithme discret [Odl84]. Ces protocoles effectuent des exponentiations modulo un nombre premier p, laissé libre en théorie, c'est-à-dire sur un groupe fini $(\mathbb{Z}/p\mathbb{Z})^*$. Calculer g^x sur un groupe fini est facile, mais retrouver l'entier x à partir de g^x est difficile sur un groupe générique, i.e. un groupe sur lequel seules les propriétés définissant un groupe sont exploitables. Le problème de Diffie Hellman est aussi difficile que le problème du logarithme discret. L'algorithme rho de Pollard est connu pour résoudre le logarithme discret sur un groupe quelconque avec une complexité exponentielle en $O(\sqrt{p})$ où p est la taille du groupe [Pol78], et a été prouvé optimal pour un groupe générique [Sho97]. Cependant, le groupe $(\mathbb{Z}/p\mathbb{Z})^*$ n'est pas générique et les cryptanalystes ont pu utiliser certaines propriétés du groupe pour attaquer le logarithme discret de façon sous-exponentielle. Pour assurer sa sûreté, les moduli doivent avoir une taille d'au moins 1024 bits. Afin de garantir qu'aucune autre propriété que celles définies sur un groupe générique ne puissent être utilisées, Koblitz propose en 1985 le groupe des points d'une courbe elliptique, introduisant la cryptographie sur les courbes elliptiques, ou Elliptic Curve Cryptography (ECC) [Kob87].

En ECC, les opérations sont effectuées sur les points de la courbe dont les coordonnées se trouvent dans un corps de base qui est soit un corps premier $\mathbb{Z}/p\mathbb{Z}$ avec p un premier, dont le choix est libre en théorie, soit une extension d'un corps premier de degré k. L'avantage est qu'il n'existe pas à ce jour d'attaque générale du logarithme discret sur le groupe des points d'une courbe elliptique et seul l'algorithme rho de Pollard, qui fonctionne sur tous les groupes, garantit le niveau de sécurité sur les courbes elliptiques. Dans le cas d'un corps de base premier, la taille du corps, i.e. le premier p, doit être suffisament grande, pour garantir une sécurité élevée, avec un minimum de 160 bits. L'exponentiation sur $(\mathbb{Z}/p\mathbb{Z})^*$ présente chez Diffie Hellman devient ici une multiplication par un scalaire k d'un point P générateur du groupe des points de la courbe elliptique. La multiplication kP est constituée d'une suite d'additions et de doublements de points. Tous les calculs requis pour l'évaluation des coordonnées des points doivent être implémentés. Il est donc primordial d'avoir une arithmétique modulaire complète efficace, comprenant l'addition, la multiplication et l'inversion modulaire.

Dans la mesure où le modulo est standardisé pour la plupart des applications cryptographiques, le choix de p n'est pas toujours libre dans la pratique. La question s'est donc

rapidement posée de savoir s'il est possible ou non de fournir une arithmétique modulaire efficace pour le plus grand nombre de premiers p possible.

Ce besoin de proposer une arithmétique modulaire efficace pour de nombreux premiers s'accompagne de la nécessité de trouver des méthodes afin de la rendre sûre face aux attaques par canaux cachés. Comme la validité de certaines attaques repose sur l'hypothèse faite par l'attaquant sur la représentation des données [Gou03], on doit éviter d'utiliser la même représentation au cours des calculs. Pour cela, l'arithmétique doit être randomisée, rendant imprévisibles la représentation des données et les transitions d'état, et empêchant ainsi un attaquant de faire des prédictions.

Le besoin d'une arithmétique modulaire efficace pour de nombreux moduli a connu ses premières avancées dans le cas des implantations des protocoles d'ECC, où certains ont profité du fait que les protocoles imposent uniquement la taille du corps, i.e. la taille du modulo, et non le type du modulo. Knuth propose en 1981 une classe de nombres adaptée à l'arithmétique modulaire [Knu81] ; les nombres de Mersenne de la forme $p = 2^n - 1$ avec $n \in \mathbb{N}$, qui simplifient grandement la réduction, en utilisant le fait qu'en représentation binaire, utilisée en informatique, la division par une puissance de 2 est réduite à un simple décalage de bits. Cependant cette classe est insuffisante car les nombres de Mersenne premiers sont rares et il n'existe pas de tels nombres premiers pour les tailles cryptographiques qui nous intéressent.

Pour parer à ce problème, d'autres classes de moduli ont été proposées. Crandall [Cra92] étend cette classe en 1992 avec les nombres Pseudo Mersenne de la forme $p = 2^n - c$ avec $n \in \mathbb{N}$ et $c < 2^{n/2}$, où la taille de c est capitale. Ces moduli sont conseillés pour opérer sur des chiffres, donc sur de petits moduli. En plus de la multiplication initiale obligatoire, plusieurs autres multiplications sont ajoutées avec un surcoût non négligeable.

En réponse à ce surcoût, les nombres de Mersenne Généralisé sont présentés par Solinas en 1999 comme une généralisation des nombre de Mersennes [S+99], où les premiers ont pour forme $p = P(2^k)$ avec P un polynôme unitaire de degré n à coefficients dans $\{-1, 0, 1\}$ et dont le second terme de plus haut degré à un degré inférieur ou égal à $n/2$. La forme polynomiale de ces nombres ne nécessite pas de multiplication supplémentaire, les rendant plus efficaces que les nombres Pseudo Mersenne sur les tailles cryptographiques. Leurs formes les rend donc plus adaptés au calcul sur les grands nombres, souvent représentés sous forme polynomiale. Chung et Hasan [CH03] fusionnent en 2003 les deux classes précédentes. La rapidité de la réduction est directement lié à la taille de la classe. Plusieurs standards proposent des moduli pour différentes tailles cryptographiques [Fip00, SEC00], basées sur la classe des nombres de Mersenne.

Récemment, Bigou et Tisserand ont proposé une approche [BT16] basée sur le système de représentation modulaire, ou Residue Number System (RNS), une représentation nu-

mérique trouvant de nombreuses applications sur les systèmes cryptographiques [SAM16] grâce au fait qu'elle permet des implémentations de l'addition et de la multiplication plus efficaces. Cependant, en raison de sa nature non positionnelle, les réductions modulaires, exigées par exemple par ECC, deviennent plus coûteuses.

Pour cette raison, Bigou et Tisserand ont introduit en 2016 un système hybride ; HPR, offrant un compromis entre l'efficacité de RNS et la flexibilité des représentations des nombres positionnels [BT16]. Les nombres sont représentés dans un système positionnel avec les chiffres représentés en RNS. Cette approche réduit la taille des bases RNS, et ainsi la complexité des extensions de base, tout en bénéficiant de l'indépendance arithmétique des canaux RNS. La complexité des réductions modulo les nombres premiers est atténuée, en s'appuyant sur des extensions de bases plus petites, à condition d'utiliser des nombres premiers de la forme $P = B_1^n - \beta$, où B_1 est l'intervalle dynamique du système RNS, n le nombre de chiffres et β un petit entier. En raison de la nécessité de construire des premiers d'une forme spéciale, cette approche n'est pas directement extensible aux opérations de groupe sur des courbes elliptiques actuellement standardisées.

Le principal inconvénient dont souffrent ces approches est de ne garantir une réduction efficace que sur une classe restreinte de moduli avec des propriétés spécifiques, le choix du modulo n'est donc pas possible.

Par conséquent, lorsque les protocoles imposent certaines propriétés sur le modulo à utiliser, il est important d'utiliser des algorithmes fonctionnant pour tous les moduli ; ces algorithmes sont appelés algorithmes *généralistes*. Certains effectuent une réduction dîte intégrée, c'est-à-dire que les calculs sont réduits au cours de la multiplication. L'algorithme de multiplication modulaire proposé par F. J. Taylor en 1981 [Tay81] permet de décomposer la multiplication sous forme de carrés. Il opère sur la représentation classique des nombres pour des moduli impairs et utilise une table mémoire pour calculer la mise au carrée. G. R. Blakley [Bla83] est le premier à proposer en 1983 un algorithme de multiplication modulaire intégrée à partir d'un "double and add" classique où des réductions sont effectuées après chaque étape de calcul. Il reste néanmoins peu performant. N. Takagi [Tak92] s'en inspire en 1986 et propose un algorithme de multiplication modulaire très efficace en utilisant son propre système d'addition modulaire. Toutes les opérations sont effectuées en utilisant un système redondant. Néanmoins, ces algorithmes demeurent efficaces pour de petits moduli seulement.

C'est pourquoi, pour effectuer des multiplications modulo de grands entiers, comme ceux présents dans les protocoles cryptographiques cités précédemment, les algorithmes généralistes les plus efficaces sont ceux qui effectuent d'abord la multiplication, et qui réduisent ensuite le résultat de cette multiplication. Dans ce cas, l'étape de la multi-

plication, indépendante, peut être optimisée via de nombreuses méthodes classiques de multiplication [KO62, Knu81, SS71].

En 1986, Montgomery [Mon85] propose de remplacer la réduction modulo p par une division par une puissance de la base $\beta = 2^n$, qui dans la pratique, se réduit à un décalage de bits sans coût. Son algorithme utilise une représentation particulière des nombres, nécessitant des conversions. S'il n'est pas pertinent pour une seule multiplication modulaire, pour une exponentiation modulaire effectuant entre n et $2n$ multiplications modulaires, le surcoût lié aux conversions devient négligeable. Paul Barrett [Bar87] propose l'année suivante un algorithme utilisant la représentation classique pour approcher le calcul du quotient de façon suffisament précise afin que le résultat obtenu soit proche du reste exact (à une ou deux soustractions du modulo près). Cet algorithme nécessite de précalculer une division par le modulo. Son coût est très proche de celui de l'algorithme de Montgomery.

Plus récemment, Thomas Plantard a introduit un système avec des algorithmes plus efficaces que les méthodes sans division connues telles que Montgomery et Barrett ; le système de représentation adapté polynomial, ou Polynomial Modular Number System (PMNS) [Pla05]. Ce système de représentation permet l'implémentation d'une arithmétique modulaire efficace impliquant uniquement des additions et des multiplications. Sa motivation principale était d'accélérer l'arithmétique modulo un entier p et de proposer un théorème d'existence du PMNS avec des propriétés spécifiques. Ce système offre à la fois l'avantage d'une arithmétique polynomiale rapide et d'une parallélisation facile pour un p arbitraire. Les éléments sont représentés par des polynômes dont les coefficients sont les chiffres. La construction de tels systèmes repose sur des polynômes creux dont les racines modulo p peuvent être choisies comme bases de ce type de représentation positionnelle. Cependant, le choix de ces polynômes et la recherche de leurs racines n'est pas triviale et les paramètres du théorème d'existence restent très restrictifs. Pour un modulo donné, peu de systèmes peuvent être construits en pratique.

Par ailleurs, pour un modulo p donné, ces systèmes peuvent devenir très redondants quand la taille des chiffres augmente, c'est-à-dire qu'il existe de nombreuses représentations distinctes du même élément modulo p dans le système. Il est fortement soupçonné que cette redondance puisse être exploitée afin de changer aléatoirement de représentation au cours des calculs, au sein d'une même exécution. Un attaquant serait alors dans l'incapacité de prévoir les transitions d'état au niveau des points de mémorisation et de faire des hypothèses utiles sur les représentations.

Enfin, au-delà de la cryptographie, les représentations de nombres jouent également un rôle important dans le domaine de la modélisation de problèmes avec des données incertaines, qui trouve d'importantes applications en ingénierie, économie et sciences sociales [ATT94, Lod07]. Afin de capturer l'incertitude autour d'une valeur donnée, les *nombres*

flous ont été introduits. Pour m un réel fixé, un nombre flou $\widetilde{m}$ est un sous-ensemble de $\mathbb{R}$ défini par sa fonction d'appartenance $\mu_{\widetilde{m}}$ représentant le degré de validité de la proposition "x est égal à m" pour chaque $x \in \mathbb{R}$. Le réel m est appelé le mode. Plus x s'éloigne du mode m, plus le degré de validité $\mu_{\widetilde{m}}(x)$ diminue. La fonction $\mu_{\widetilde{m}}$ à valeur dans $[0, 1]$ est définie sur $\mathbb{R}$, est continue, et satisfait $\mu_{\widetilde{m}}^{-1}(\{1\}) = \{m\}$. Cette fonction d'appartenance est construite à partir de deux fonctions L et R, appelées *fonctions de dispersion*, permettant de représenter l'incertude respectivement à gauche et à droite du mode m. Le nombre flou appartient alors à la famille des nombres flous de type L-R, notée $\mathfrak{F}(L, R)$.

Certains problèmes sont modélisés par un système de fonctions continues à valeurs floues définies sur $\mathbb{R}^n$ (voir [LS03]) ; Liu [Liu02] propose une interpolation polynomiale fournissant une interface entre la résolution de ces problèmes et la résolution de systèmes polynomiaux à coefficients flous. Pour cette raison, plusieurs auteurs se sont intéressés à la recherche de solutions réelles d'équations polynomiales floues.

Les méthodes de résolution se sont d'abord appuyées sur des techniques locales ([AO06], [Rou07], [AOM08]). Puis ces dernières années s'est développée une approche globale faisant appel à des techniques algébriques classiques ([MBR13, BBRV16]). Celles-ci peuvent effectivement être utilisées avec les nombres flous, qui, en dépit d'une dénomination pouvant prêter à confusion, bénéficient d'une définition tout-à-fait formelle.

Pour la résolution des systèmes algébriques, chaque coefficient flou émanant de l'expérimentation est généralement un nombre flou L-R donné sous une certaine représentation, dite *en tuple*. Cette représentation, bien que formelle, est non traitable par des méthodes algébriques usuelles (bases de Gröbner [BW93], décomposition triangulaire [AMM99], représentation univariée rationnelle [Rou99], ...) pour résoudre le système. Néanmoins, la représentation tuple d'un nombre flou L-R est transformable en une autre représentation, dite *paramétrique*, à coefficients non plus flous mais réels.

Jusqu'à présent, étant donné un système flou (S) de s équations et k indéterminées, les méthodes algébriques existantes ont effectué des calculs avec la représentation paramétrique des coefficients, qui implique de calculer l'inverse des fonctions de dispersion, pour obtenir la *forme tranchée* de (S) formée par $4s$ équations réelles. Ces méthodes se limitent néanmoins à une famille particulière de nombres flous appelés les nombres flous *triangulaires*, dont les fonctions de dispersion sont linéaires. Une méthode basée sur la décomposition triangulaire de Wu Wen Tsun [Wu87] a été étudiée dans le cas particulier des nombres flous triangulaires durant un stage de Master au sein du LIP6, et implantée en SageMath pour calculer les solutions positives d'un tel système [Mar19].

Le dernier problème majeur pour obtenir les solutions réelles de ces systèmes réside dans la gestion des signes. En effet, le problème qui se pose lors du calcul avec des variables réelles et des nombres flous est intrinsèque aux nombres flous car leur produit par un scalaire réel est exprimé différemment selon le signe de ce scalaire.

2 But et organisation de la thèse

Le but de cette thèse est de proposer une arithmétique modulaire efficace pour le plus grand nombre de moduli premiers p possible et de la prémunir contre certains types d'attaques comme celles mentionnées précédemment. Nous souhaitons contribuer ainsi à lever une partie des restrictions qui limitent la performance des calculs modulaires lorsque le choix du modulo est imposé par le protocole cryptographique.

Ces travaux s'articulent autour de trois objectifs majeurs :

- fournir de nouvelles bases de systèmes de représentation modulaires, indépendantes de la forme du premier p, en garantissant une arithmétique engendrée efficace,
- exploiter la redondance intrinsèque à ces systèmes afin d'effectuer des changements de représentation des données au cours du calcul, et ainsi empêcher un attaquant d'effectuer des hypothèses utiles sur ces représentations,
- en parallèle de cette recherche en cryptographie, nous nous proposons d'établir de nouvelles méthodes dans le domaine de la modélisation incertaine pour optimiser la résolution réelle des sytèmes à coefficients flous.

La première contribution est présentée dans le chapitre 1. Ce travail est basé sur le système de représentation adapté polynomial (PMNS) introduit en 2004, utilisé pour l'arithmétique modulaire et lié à la caractéristique du corps $\mathbb{Z}/p\mathbb{Z}$.

Nous introduisons dans ce chapitre un théorème général sur l'existence des systèmes PMNS et fournissons des bornes sur la taille des chiffres utilisés pour représenter un entier modulo p. Puis nous présentons des classes de polynômes adaptés pour obtenir des systèmes avec une arithmétique efficace sur les représentations. Enfin pour un p premier, nous évaluons le nombre de racines de polynômes modulo p afin de décrire le nombre minimum de bases PMNS que nous pouvons atteindre.

Par conséquent, pour un premier p donné, il est possible d'obtenir de nombreux PMNS, qui peuvent être utilisés efficacement pour différentes applications basées sur de grands corps finis premiers, tels que celles que l'on trouve en cryptographie, comme RSA, Diffie-Hellmann et ECC.

Dans [BIP05a], les auteurs mentionnent que les PMNS peuvent être très redondants mais n'ont pas vraiment profité de cette possibilité. Dans le second chapitre, nous utilisons, pour la première fois, la redondance du PMNS pour protéger les algorithmes contre les attaques par canaux cachés (SCA). Plus précisément, nous nous concentrons sur la cryptographie sur les courbes elliptiques.

Nous montrons comment randomiser la multiplication modulaire via des algorithmes de type Montgomery et Babaï afin d'être résistant contre les attaques SCA existantes et

nous démontrons la résistance de nos constructions. Nous décrivons la génération d'un PMNS en garantissant, pour tous les éléments de $\mathbb{Z}/p\mathbb{Z}$, le nombre minimum de représentations distinctes que nous voulons. Nous montrons aussi comment accéder à toutes ces représentations.

Le système hybride HPR introduit en 2016 offre un compromis entre l'efficacité de RNS et la flexibilité des représentations des nombres positionnels, mais son utilisation reste limitée à certains moduli premiers d'une forme spéciale. Dans le chapitre 3, dans la veine du système HPR, nous proposons le système hybride HyPoRes, offrant un algorithme de multiplication modulaire généraliste, permettant d'améliorer les réductions modulaires pour tout modulo premier.

Les nombres sont représentés dans un système PMNS avec les coefficients représentés en RNS. Les résultats expérimentaux montrent que la réduction modulaire de HyPoRes, bien qu'au plus 1,4 fois plus lente que HPR pour les nombres premiers conçus pour HPR, est jusqu'à 1,4 fois plus rapide qu'une approche générique RNS pour les nombres premiers standardisés pour ECC.

Enfin, dans le dernier chapitre, nous nous intéressons à un autre type de représentation utilisé pour la résolution réelle de systèmes dont les coefficients sont des nombres flous, dans un travail effectué en parallèle durant cette thèse. Nous revisitons l'approche globale de résolution faisant appel à des techniques algébriques classiques ([MBR13, BBRV16]) et la renforçons.

Jusqu'alors, les méthodes antérieures tant locales que globales se sont focalisées sur la famille des nombres flous dits *triangulaires*.

Les résultats présentés ici s'appliquent plus généralement aux nombres flous *L-R* symétriques, i.e. pour lesquels $L(x) = R(\frac{x}{k})$, $k > 0$, que sont en particulier les nombres flous triangulaires de fonctions de dispersion linéaires, mais aussi, par exemple, les nombres flous quadratiques, de fonctions de dispersion quadratiques.

Jusqu'à présent, étant donné un système flou (S) de s équations et k indéterminées, les méthodes algébriques existantes ont effectué des calculs avec la représentation paramétrique des coefficients pour obtenir la forme tranchée de (S) formée par $4s$ équations réelles. Nous montrons que ces calculs sont superflus et présentons une formule qui définit un système équivalent avec $3s$ équations réelles. Nous l'appelons la transformation réelle $\mathcal{T}(S)$ du système (S). Comme propriété principale, elle possède les mêmes solutions positives que (S). De plus, il est inutile d'effectuer les inversions de fonctions nécessaires pour obtenir la représentation paramétrique puisque cette dernière n'intervient pas dans cette nouvelle méthode, la transformation réelle étant une formule universelle indépendante de la famille du nombre flou.

Nous fournissons une méthode pour obtenir l'ensemble des solutions réelles de (E), et pas seulement les solutions positives. Ces résultats théoriques incluent la gestion des signes des solutions des systèmes flous. Nous montrons, comment dans le cadre de ces nombres flous plus généraux, la résolution d'un système (S) à coefficients flous se ramène à la recherche des solutions positives de 2^k systèmes à coefficients réels. Nous étendons ce résultat aux nombres flous dits *trapézoïdaux*. Après la description d'un algorithme de base nécessitant la résolution de 2^k systèmes, nous proposons l'algorithme optimisé `SolveFuzzySystem` qui réduit le nombre de systèmes à résoudre. Une implémentation dans le package Fuzzy du logiciel d'algèbre informatique SageMath et une version parallèle de l'algorithme sont décrites.

Chapitre I

Bornes et existence des systèmes PMNS sur $\mathbb{Z}/p\mathbb{Z}$

Sommaire

Le système de représentation adapté polynomial, ou Polynomial Modular Number System (PMNS) a été introduit en 2004 [Pla05] comme un système de représentation permettant l'implémentation d'une arithmétique modulaire efficace impliquant uniquement des additions et des multiplications. Les opérations arithmétiques appelées addition, multiplication et réduction modulaires interviennent dans plusieurs algorithmes de cryptographie à clé publique actuels, tels que les célèbres RSA, l'échange de clés Diffie-Hellman et ECC [KMVOV96]. Ces calculs modulo un entier p consistent à ajouter ou multiplier deux entiers, puis à récupérer le reste modulo p. La manière d'effectuer ces opérations varie selon la forme de p (par exemple les nombres Mersenne, de la forme $2^m - 1$, permettent une réduction rapide). Les systèmes PMNS offrent à la fois l'avantage d'une arithmétique polynomiale rapide et d'une parallélisation facile pour un p arbitraire, avec des algorithmes plus efficaces que les méthodes sans division connues telles que Montgomery [Mon85] et Barrett[Bar87].

L'idée principale derrière le PMNS est qu'il s'agit d'un système modulaire, où les entiers modulo un p arbitraire (pas nécessairement premier) sont représentés par des polynômes de degré inférieur à un entier fixé n. Les coefficients des polynômes sont les chiffres et sont bornés par un entier ρ, qui est petit relativement à p ($\rho \approxeq p^{1/n}$). La construction de tels systèmes est basée sur des polynômes creux dont les racines γ sont utilisées comme bases de ce type de représentation positionnelle. L'intérêt de ces polynômes creux réside dans l'efficacité de l'arithmétique modulaire engendrée. Ces opérations sont réalisées en deux étapes. Premièrement, les opérations sont effectuées sur des polynômes modulo un polynôme creux $E(X)$, appelé polynôme de réduction, qui est de degré n, et cette réduction assure que le degré du résultat est inférieur à n. Ensuite, une réduction des coefficients est effectuée impliquant le réseau euclidien associé au système [HPS11, PSZ15, Gal12], garantissant que les coefficients du résultat sont bornés par ρ. Le nombre de systèmes PMNS que nous pouvons générer à partir d'un entier p est directement lié au nombre de racines du polynôme de réduction $E(X)$ dans $\mathbb{Z}/p\mathbb{Z}$.

Une méthode de construction d'un PMNS efficace a été publiée en 2004 [BIP05b]. Le système est construit à partir de polynômes creux avec de bonnes propriétés de réduction (un pour la réduction polynomiale, $E(X)$, un pour la réduction des coefficients), dans le but de déduire l'entier p via le calcul d'un résultant, ainsi qu'une racine γ. Sont conservés uniquement les cas où p est premier, puisque p est le plus souvent premier en pratique dans les applications cryptographiques telles que Diffie-Hellman et ECC. Cependant, avec une telle construction, la sélection de p, requise dans la plupart des protocoles cryptographiques, n'est pas possible. Dans ces protocoles, le modulo p utilisé est souvent fixé ou possède au moins de fortes propriétés mathématiques requises. Par conséquent, dans le but de pouvoir travailler avec un p arbitraire, premier ou non, un premier théorème sur

les systèmes PMNS [BIP05a], donne une construction de PMNS à partir d'un entier p, un nombre de chiffres n et un polynôme entier $E(X)$ sous la forme $E(X) = X^n + aX + b$ satisfaisant certaines hypothèses. Ainsi, ce théorème garantit l'existence du système PMNS $\mathfrak{B} = (p, n, \gamma, \rho)_E$, avec une borne sur ρ. Néanmoins, construire de tels systèmes à partir d'un p donné n'est pas trivial. Pour obtenir un exemple d'un système PMNS à partir d'un p fixé, il faut chercher un polynôme creux $E(X)$ satisfaisant les conditions du théorème, et l'une de ses racines dans $\mathbb{Z}/p\mathbb{Z}$.

Dans ce chapitre, pour un nombre p donné, nous voulons fournir autant de bases PMNS que possible avec des polynômes de réduction efficaces. Par conséquent, nous proposons, dans la Section I.2, un théorème donnant, comme critère d'existence, une borne sur la taille des chiffres en fonction d'un polynôme $E(X)$, quelque soit la forme de ce dernier, et certaines propriétés lorsque $E(X)$ est irréductible. Ensuite, à la Section I.3, nous proposons des classes de polynômes de réduction adaptés qui satisfont les théorèmes précédents, permettent des réductions efficaces, et dont les racines peuvent être clairement identifiées dans un corps premier fini $\mathbb{Z}/p\mathbb{Z}$. Pour terminer, dans la Section I.4, nous donnons le nombre de racines en fonction de p et du polynôme de réduction $E(X)$. Ces racines sont utilisées pour générer le réseau euclidien associé au système, et agissent directement sur la réduction des coefficients, faisant de cette recherche un défi important pour obtenir des représentations efficaces en termes de calcul et de stockage. Ainsi, pour un nombre premier fixé p, il devient possible d'obtenir plusieurs bases PMNS avec leurs propres propriétés de calcul. Cette capacité à fournir plusieurs représentations équivalentes est également un point intéressant en termes de performance si l'on veut masquer les calculs pour protéger une implantation contre des observateurs malveillants.

I.1 Préliminaires

Nous présentons les généralités et notations sur le système PMNS. Pour ce faire, nous commençons par rappeler la définition d'un système de numération positionnelle classique.

Définition I.1.1. Un système de numération positionnel classique est défini à partir d'un entier β supérieur ou égal à 2, appelé la base, tel que tout entier $a < \beta^m$ puisse être représenté par une unique suite d'entiers $(a_i)_{i=0\ldots m-1}$, appelés chiffres, tel que $a = \sum_{i=0}^{m-1} a_i \beta^i$, avec $a_i \in \mathbb{N}$, $0 \leq a_i < \beta$.

Remarque I.1.1. Nous pouvons associer à cette représentation un polynôme $A(X)$ tel que ses coefficients soient les chiffres de a, $A(X) = \sum_{i=0}^{m-1} a_i X^i$, et $A(\beta) = a$.

Nous nous plaçons à présent dans le cadre de l'arithmétique modulaire. Soit $n < m$, soit p un entier tel que $\beta^{n-1} \leq p < \beta^n$ et $\beta^n \equiv \delta \bmod p$. Pour calculer a modulo p, avec

$p \leq a \leq \beta^m$ (i.e. $m \geq n$), une méthode consiste à remplacer β^n par δ dans l'écriture de a et à itérer le processus jusqu'à obtenir une valeur inférieure à β^n :

$$a = \sum_{i=0}^{n-1} a_i \beta^i + \beta^n \sum_{i=n}^{m-1} a_i \beta^{i-n} \equiv \sum_{i=0}^{n-1} a_i \beta^i + \delta \sum_{i=n}^{m-1} a_i \beta^{i-n} \bmod p.$$

Il est clair que le nombre d'itérations dépend de δ ; si $\delta < \beta^{n/2}$ et $m < 2n$ alors la réduction peut être faite en deux itérations au plus. Une ultime soustraction peut ensuite s'avérer nécessaire pour obtenir un résultat inférieur à β^n.

Pour les besoins de notre étude, cette réduction peut être décomposée par une approche polynomiale. Puisque $\beta^n - \delta \equiv 0 \bmod p$, nous pouvons considérer β comme une racine modulo p du polynôme $E(X) = X^n - \Delta(X)$ où $\Delta(\beta) \equiv \delta \bmod p$. Nous désignons par t le degré de $\Delta(X)$.

Ainsi, la réduction modulo p est effectuée avec des itérations divisées en deux étapes :

1. La réduction polynomiale : $C(X) = A(X) \bmod E(X)$.
2. La réduction des coefficients : $C'(\beta) \equiv C(\beta) \bmod p$ avec $C'(X)$ de degré $n-1$ et aux coefficients inférieurs à β.

La réduction polynomiale correspond à une réduction rapide et brute de la taille (du degré), puis la réduction des coefficients réduit les coefficients du polynôme obtenu à des valeurs de chiffres inférieures à β. Généralement, la réduction polynomiale ressemble à ce qui suit :

1. $C(X) = A(X)$
2. tant que le degré de $C(X)$ est supérieur à n, (le degré diminue d'environ $(n-t)$) $C(X) = \Delta(X) \times \sum_{i=n}^{m-1} c_i X^{i-n} + \sum_{i=0}^{n-1} c_i X^i$.

Ainsi, si le degré t de $\Delta(X)$ est inférieur à $n/2$ et $m < 2n$, après la première itération de la boucle de l'étape 2, $\deg C(X) = t + m - n - 1$ et après la seconde $\deg C(X) < n - 1$. Maintenant, si $\Delta(X)$ est un polynôme creux avec de petits coefficients, alors la multiplication par $\Delta(X)$ se réduit à quelques décalages et additions seulement [S+99].

Malheureusement, il n'est pas évident de trouver un couple $(\beta, E(X))$ avec de bonnes propriétés, dans les systèmes de numération positionnelle classiques. Pour obtenir plus d'opportunités d'avoir un tel couple, un nouveau type de représentation a été introduit dans [BIP05a], où la base, pour un p donné, dépend étroitement du choix du polynôme de réduction $E(X)$.

Définition I.1.2. Un *système de représentation adapté polynomial*, ou Polynomial Modular Number System (PMNS), est défini par un quadruplet (p, n, γ, ρ) et un polynôme

$E(X) \in \mathbb{Z}[X]$, appelé *polynôme de réduction* par rapport à p, tels que pour chaque entier x dans $\{0, \ldots, p-1\}$, il existe $(x_0, \ldots, x_{n-1})$ avec $x \equiv \sum_{i=0}^{n-1} x_i \gamma^i \bmod p$, où $x_i \in \mathbb{N}$, $-\rho < x_i < \rho$, $1 < \gamma < p$, et $E(\gamma) \equiv 0 \pmod p$, avec $E(X)$ un polynôme unitaire de degré n.

Exemple I.1.1. Considérons le PMNS défini par $\mathfrak{B} = (p, n, \gamma, \rho)_E$ avec $p = 31$, $n = 3$, $\gamma = 11$ et $\rho = 3$, pour représenter les éléments de $\mathbb{Z}/31\mathbb{Z}$ comme vecteurs avec 3 chiffres à valeur dans {-2, -1, 0, 1, 2}. Notons que $\gamma^3 + 2 \equiv 0 \bmod 31$ (i.e. $E(X) = X^3 + 2$).

0	1	2	3	4	5	6	7
(0, 0, 0)	(-2, 0, -1) (1, 0, 0) (-2, -2, 2)	(-1, 0, -1) (2, 0, 0) (-1, -2, 2)	(0, 0, -1) (0, -2, 2) (-2, 1, 2)	(-2, 0, -2) (1, 0, -1) (-2, -2, 1) (1, -2, 2) (-1, 1, 2)	(-1, 0, -2) (2, 0, -1) (-1, -2, 1) (2, -2, 2) (0, 1, 2)	(0, 0, -2) (0, -2, 1) (-2, 1, 1) (1, 1, 2)	(1, 0, -2) (-2, -2, 0) (1, -2, 1) (-1, 1, 1) (2, 1, 2)
8	9	10	11	12	13	14	15
(2, 0, -2) (-1, -2, 0) (2, -2, 1) (0, 1, 1)	(0, -2, 0) (-2, 1, 0) (1, 1, 1)	(-2, -2, -1) (1, -2, 0) (-1, 1, 0) (2, 1, 1)	(-1, -2, -1) (2, -2, 0) (0, 1, 0)	(0, -2, -1) (-2, 1, -1) (1, 1, 0) (-2, -1, 2)	(-2, -2, -2) (1, -2, -1) (-1, 1, -1) (2, 1, 0) (-1, -1, 2)	(-1, -2, -2) (2, -2, -1) (0, 1, -1) (0, -1, 2) (-2, 2, 2)	(0, -2, -2) (-2, 1, -2) (1, 1, -1) (-2, -1, 1) (1, -1, 2) (-1, 2, 2)
16	17	18	19	20	21	22	23
(1, -2, -2) (-1, 1, -2) (2, 1, -1) (-1, -1, 1) (2, -1, 2) (0, 2, 2)	(2, -2, -2) (0, 1, -2) (0, -1, 1) (-2, 2, 1) (1, 2, 2)	(1, 1, -2) (-2, -1, 0) (1, -1, 1) (-1, 2, 1) (2, 2, 2)	(2, 1, -2) (-1, -1, 0) (2, -1, 1) (0, 2, 1)	(0, -1, 0) (-2, 2, 0) (1, 2, 1)	(-2, -1, -1) (1, -1, 0) (-1, 2, 0) (2, 2, 1)	(-1, -1, -1) (2, -1, 0) (0, 2, 0)	(0, -1, -1) (-2, 2, -1) (1, 2, 0) (-2, 0, 2)
24	25	26	27	28	29	30	
(-2, -1, -2) (1, -1, -1) (-1, 2, -1) (2, 2, 0) (-1, 0, 2)	(-1, -1, -2) (2, -1, -1) (0, 2, -1) (0, 0, 2)	(0, -1, -2) (-2, 2, -2) (1, 2, -1) (-2, 0, 1) (1, 0, 2)	(1, -1, -2) (-1, 2, -2) (2, 2, -1) (-1, 0, 1) (2, 0, 2)	(2, -1, -2) (0, 2, -2) (0, 0, 1)	(1, 2, -2) (-2, 0, 0) (1, 0, 1)	(2, 2, -2) (-1, 0, 0) (2, 0, 1)	

Exemple I.1.2. Considérons les deux systèmes PMNS définis par $\mathfrak{B} = (p, n, \gamma, \rho)_E$. Un premier, avec $p = 23$, $n = 3$, $\gamma = 7$ et $\rho = 2$, pour représenter les éléments de $\mathbb{Z}/23\mathbb{Z}$ comme vecteurs avec 3 chiffres à valeur dans $\{-1, 0, 1\}$. Notons que $\gamma^3 + 2 \equiv 0 \bmod 23$ (i.e. $E(X) = X^3 + 2$).

0	1	2	3	4	5	6	7
(0, 0, 0)	(1, 0, 0)	(-1, 0, 1)	(-1, 1, -1) (0, 0, 1)	(0, 1, -1) (1, 0, 1)	(1, 1, -1)	(-1, 1, 0)	(0, 1, 0)
8	9	10	11	12	13	14	15
(1, 1, 0)	(-1, 1, 1)	(0, 1, 1)	(1, 1, 1)	(-1, -1, -1)	(0, -1, -1)	(1, -1, -1)	(-1, -1, 0)
16	17	18	19	20	21	22	
(0, -1, 0)	(1, -1, 0)	(-1, -1, 1)	(-1, 0, -1) (0, -1, 1)	(0, 0, -1) (1, -1, 1)	(1, 0, -1)	(-1, 0, 0)	

À présent, avec $p = 31$, $n = 4$, $\gamma = 15$ et $\rho = 2$, pour représenter les éléments de $\mathbb{Z}/31\mathbb{Z}$ comme vecteurs avec 4 chiffres à valeur dans $\{-1, 0, 1\}$. Notons que $\gamma^4 - 2 \equiv 0 \bmod 31$ (i.e. $E(X) = X^4 - 2$).

0	1	2	3	4	5
(0, 0, 0,0)	(1, 0, 0, 0)	(-1, 1, -1, 1)	(-1, -1, -1, 1) (-1, 0, 0, -1) (-1, 0, 1, 1) (0, 1, -1, 1)	(0, -1, -1, 1) (0, 0, 0, -1) (0, 0, 1, 1) (1, 1, -1, 1)	(1, -1, -1, 1) (1, 0, 0, -1) (1, 0, 1, 1)
6	7	8	9	10	11
(-1, 1, -1, 0)	(-1, -1, -1, 0) (-1, 0, 1, 0) (0, 1, -1, 0)	(0, -1, -1, 0) (0, 0, 1, 0) (1, 1, -1, 0)	(1, -1, -1, 0) (1, 0, 1, 0)	(-1, 1, -1, -1) (-1, 1, 0, 1)	(-1, -1, -1, -1) (-1, -1, 0, 1) (-1, 0, 1, -1) (0, 1, -1, -1) (0, 1, 0, 1)
12	13	14	15	16	17
(0, -1, -1, -1) (0, -1, 0, 1) (0, 0, 1, -1) (1, 1, -1, -1) (1, 1, 0, 1)	(1, -1, -1, -1) (1, -1, 0, 1) (1, 0, 1, -1)	(-1, 1, 0, 0)	(-1, -1, 0, 0) (0, 1, 0, 0)	(0, -1, 0, 0) (1, 1, 0, 0)	(1, -1, 0, 0)
18	19	20	21	22	23
(-1, 0, -1, 1) (-1, 1, 0, -1) (-1, 1, 1, 1)	(-1, -1, 0, -1) (-1, -1, 1, 1) (0, 0, -1, 1) (0, 1, 0, -1) (0, 1, 1, 1)	(0, -1, 0, -1) (0, -1, 1, 1) (1, 0, -1, 1) (1, 1, 0, -1) (1, 1, 1, 1)	(1, -1, 0, -1) (1, -1, 1, 1)	(-1, 0, -1, 0) (-1, 1, 1, 0)	(-1, -1, 1, 0) (0, 0, -1, 0) (0, 1, 1, 0)
24	25	26	27	28	29
(0, -1, 1, 0) (1, 0, -1, 0) (1, 1, 1, 0)	(1, -1, 1, 0)	(-1, 0, -1, -1) (-1, 0, 0, 1) (-1, 1, 1, -1)	(-1, -1, 1, -1) (0, 0, -1, -1) (0, 0, 0, 1) (0, 1, 1, -1)	(0, -1, 1, -1) (1, 0, -1, -1) (1, 0, 0, 1) (1, 1, 1, -1)	(1, -1, 1, -1)
30					
(-1, 0, 0, 0)					

Nous pouvons remarquer que la redondance dépend du nombre de chiffres $2\rho - 1$, du degré n et du modulo p. La redondance n'est pas équidistribuée mais nous pouvons observer une symétrie due au signe de la valeur modulo p.

Proposition I.1.1. *Si $\mathfrak{B} = (p, n, \gamma, \rho)_E$ est un PMNS, alors $p \leq (2\rho - 1)^n$.*

Preuve. Le nombre de représentations dans $\mathfrak{B}$ est $(2\rho-1)^n$, ce nombre doit être au moins supérieur à p, c'est-à-dire le nombre de valeurs $0 \leq x < p$. □

Remarque I.1.2.

1. Le système PMNS ressemble à un système de position, mais $(\gamma^i \bmod p) < (\gamma^{i+1} \bmod p)$ n'est plus toujours vrai.

2. Pour chaque quadruplet (p, n, γ, ρ), il existe un polynôme $E(X) \in \mathbb{Z}[X]$ satisfaisant $E(\gamma) \equiv 0 \bmod p$ et $\deg E(X) = n$: par exemple $E(X) = X^n - (\gamma^n \bmod p)$.
3. Si $p < (2\rho-1)^n$, alors le système est redondant (c.-à-d. que certaines valeurs modulo p peuvent avoir plus d'une représentation).
4. Si $\mathfrak{B} = (p, n, \gamma, \rho)_E$ est un PMNS, alors $\mathfrak{B}' = (p, n, \gamma, \rho + 1)_E$ en est un aussi.

Remarque I.1.3. Pour p, n, γ fixés, afin de minimiser la redondance du système, il est judicieux de prendre pour ρ le plus petit entier tel que (p, n, γ, ρ) est un PMNS. Nous notons ρ_{min} cet entier. Cette valeur ρ_{min} peut être déterminée avec une réduction de réseau [Pla05], et donne la taille minimale des représentations pour un n donné.

La question est de savoir, pour p, n donnés, quels polynômes $E(X)$ offrent une bonne réduction modulaire, ont un grand nombre de racines γ dans $\mathbb{Z}/p\mathbb{Z}$, et permettent d'avoir ρ aussi petit que possible, pour assurer plusieurs systèmes PMNS, à la fois compacts et possédant une arithmétique efficace sur les représentations.

Note. Dans ce qui suit, nous notons $A(X) = a_0 + a_1X + \cdots + a_{n-1}X^{n-1}$ le polynôme et $A = (a_0, a_1, \ldots, a_{n-1})$ le vecteur correspondant. Nous utiliserons ces différentes notations en fonction du contexte.

I.2 Théorème sur les bornes et l'existence d'un système PMNS

Dans cette section, nous donnons nos conditions pour assurer l'existence d'un PMNS $\mathfrak{B} = (p, n, \gamma, \rho)_E$.

I.2.1 Réseau associé à un PMNS

Nous considérons le réseau $\mathfrak{L}$ des vecteurs de $\mathbb{Z}^n$ correspondant aux polynômes de degré au plus $n-1$, pour lesquels γ est une racine modulo p (voir [MG02] pour les bases de la théorie des réseaux).

Nous définissons ce réseau en donnant $\mathbf{A}$, l'une de ses bases, dont les éléments sont $A_0 = (p, 0, \ldots, 0)$ (i.e., $A_0(X) = p$), et $A_i = (0, \ldots, -\gamma, 1_i, \ldots, 0)$ (i.e. $A_i(X) = X^i - \gamma X^{i-1}$), pour $1 \leq i \leq n-1$. Ainsi, avec A_0, tous les multiples de p ont une représentation dans ce réseau, et les A_i pour $0 \leq i \leq n-1$ sont linéairement indépendants. Le volume fondamental de $\mathfrak{L}$ est $\det \mathbf{A} = p$.

Remarque I.2.1. Il est possible de considérer $A'_i(X) = X^i - \gamma^i$, pour $1 \leq i \leq n-1$, qui représente le même réseau $\mathfrak{L}$. Nous avons $A_1(X) = A'_1(X) = X - \gamma$ et $A_i(X) = A'_i(X) - \gamma A'_{i-1}(X)$.

$$\mathbf{A} := \begin{pmatrix} p & 0 & \dots & \dots & 0 & 0 \\ -\gamma & 1 & \dots & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \dots & -\gamma & 1 & \dots & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & \dots & -\gamma & 1 \end{pmatrix}, \quad \mathbf{A'} := \begin{pmatrix} p & 0 & \dots & \dots & 0 & 0 \\ -\gamma & 1 & \dots & \dots & 0 & 0 \\ \vdots & & \ddots & & & \vdots \\ -\gamma^i & \dots & 0 & 1 & \dots & 0 \\ \vdots & & & \ddots & \ddots & \vdots \\ -\gamma^{n-1} & 0 & \dots & \dots & 0 & 1 \end{pmatrix} \tag{I.1}$$

Theorème I.2.1. *Soit p un nombre premier, $n > 1$ un entier, $E(X)$ un polynôme de degré n dans $\mathbb{Z}[X]$ et γ une racine de $E(X)$ dans $\mathbb{Z}/p\mathbb{Z}$.*
Soit r le rayon de recouvrement du réseau $\mathfrak{L}$, si $\rho > r$, alors $\mathfrak{B} = (p, n, \gamma, \rho)_E$ est un système de représentation adapté polynomial.

Preuve. Le rayon de recouvrement r de $\mathfrak{L}$ est le plus petit nombre, tel que les boules $\mathcal{B}_V = \{T \in \mathbb{R}^n, \|T - V\|_2 \leq r\}$ centrées sur un point $V \in \mathfrak{L}$, couvrent l'espace $\mathbb{R}^n$. En d'autres termes, pour tout $T \in \mathbb{R}^n$, il existe $V \in \mathfrak{L}$ tel que $\|T-V\|_\infty \leq \|T-V\|_2 \leq r$. Ainsi, pour tout $T \in \mathbb{R}^n$, il existe $V \in \mathfrak{L}$, tel que $T - V \in \mathcal{C}_O$, avec $\mathcal{C}_O = \{T \in \mathbb{R}^n, \|T\|_\infty \leq r\}$.

Par définition, $\mathfrak{B} = (p, n, \gamma, \rho)_E$ est un PMNS si chaque $a \in \mathbb{N}$ possède une représentation dans $\mathfrak{B}$ modulo p.

Soit $a \in \mathbb{N}$ et soit $F_a(X)$ un polynôme tel que $F_a(\gamma) \equiv a \bmod p$, alors $T_a(X) = F_a(X) \bmod E(X)$, satisfait $T_a(\gamma) \equiv a \bmod p$ avec $\deg T_a < n$ (étape de la réduction polynomiale).

Ensuite, il existe $V \in \mathfrak{L}$ tel que, $\|T_a - V\|_\infty \leq r$, et $(T_a - V)(\gamma) \equiv T_a(\gamma) - V(\gamma) \equiv a \bmod p$. Ainsi, si $\rho > r$, $\|T_a - V\|_\infty < \rho$ et $T_a - V$ est un représentant de a dans $\mathfrak{B}$. Par conséquent, tout $a \in \mathbb{N}$ peut être représenté dans $\mathfrak{B}$ modulo p. □

Remarque I.2.2. Soit λ_n le plus petit entier tel que $\mathfrak{L}$ contient au plus n vecteurs linéairement indépendants de longueur inférieure ou égale à λ_n pour la norme euclidienne. Un résultat classique de la théorie des réseaux indique que le rayon de recouvrement r, est tel que, $\frac{1}{2}\lambda_n \leq r \leq \sqrt{n}\lambda_n$ [MG02].

I.2.2 Bases du réseau et PMNS

Actuellement, à notre connaissance, il n'existe pas d'algorithme efficace pour calculer le rayon de recouvrement d'un réseau. Dans cette section, nous fournissons une borne sur ρ, qui peut être calculée à partir d'une base du réseau $\mathfrak{L}$ défini par la matrice $\mathbf{A}$.

Nous considérons $\mathrm{B} = \{B_0, \dots, B_{n-1}\}$ une base de $\mathfrak{L}$, et $\mathbf{B}$ la matrice associée telle que, B_i représente la i-ième ligne, avec $B_i = (b_{i,0}, \dots, b_{i,n-1})$, où $b_{i,j}$ représente le coefficient

de la i-ième ligne, j-ième colonne (les indices commençant à 0).

Theorème I.2.2. *Si* $\rho \geq \frac{1}{2}\|\mathbf{B}\|_1$, $(\|\mathbf{B}\|_1 = \max_j \left\{\sum_{i=0}^{n-1} |b_{i,j}|\right\})$, *alors* $\mathfrak{B} = (p, n, \gamma, \rho)_E$ *est un système de représentation adapté polynomial.*

Preuve. Soit $S \in \mathbb{R}^n$, nous définissons :

— $\lfloor S \rceil$ comme le vecteur dont les coordonnées sont des entiers égaux à l'arrondi au plus proche de ceux de S,

— frac(S) comme le vecteur frac$(S) = S - \lfloor S \rceil$, remarquons que $\|\text{frac}(S)\|_\infty \leq \frac{1}{2}$.

Soit $S \in \mathbb{R}^n$, nous recherchons un vecteur $T \in \mathfrak{L}$ proche de S, en utilisant une approche « round-off » de Babaï [Bab86]. Nous avons, $T = \mathbf{B}^T.\lfloor(\mathbf{B}^T)^{-1}.S\rceil$.

$$S = \mathbf{B}^T.(\mathbf{B}^T)^{-1}.S = T + \mathbf{B}^T.\text{frac}\left((\mathbf{B}^T)^{-1}.S\right) \text{ avec } \left\|\text{frac}\left((\mathbf{B}^T)^{-1}.S\right)\right\|_\infty \leq \frac{1}{2}.$$

Alors

$$\|S - T\|_\infty = \left\|\mathbf{B}^T.\text{frac}\left((\mathbf{B}^T)^{-1}.S\right)\right\|_\infty \leq \frac{1}{2}\left\|\mathbf{B}^T\right\|_\infty = \frac{1}{2}\|\mathbf{B}\|_1 .$$

□

Afin de minimiser $\|\mathbf{B}\|_1$, une première stratégie générale consiste à calculer une base réduite $\mathbf{B}$ de $\mathfrak{L}$ définie par $\mathbf{A}$ en utilisant des algorithmes comme LLL, BKZ ou HKZ [LvdPdW12].

Les stratégies suivantes peuvent être appliquées lorsque le polynôme $E(X)$ est irréductible.

I.2.3 Polynômes irréductibles et PMNS

Soit $E(X) = X^n + a_{n-1}X^{n-1} + \cdots + a_1X + a_0$, et soit C la matrice compagnon de $E(X)$:

$$\mathbf{C} := \begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-2} & -a_{n-1} \end{pmatrix}.$$

Soit $V = (v_0, \dots, v_{n-1})$ le vecteur représentant les coefficients du polynôme $V(X) = \sum_{i=0}^{n-1} v_iX^i$, alors $V.\mathbf{C}$ est un vecteur dont les coordonnées sont les coefficients du polynôme $X.V(X) \bmod E(X)$.

Proposition I.2.3. *Soit V un vecteur non nul de $\mathfrak{L}$, le réseau de rang n défini par $\mathbf{A}$, Equ.* (I.1)*. Soit $B_i = V.\mathbf{C}^i$, le vecteur ligne dont les coordonnées sont les coefficients du polynôme $B_i(X) = X^i.V(X) \bmod E(X)$. Soit $\mathbf{B}$ la matrice $n \times n$ dont la i-ième ligne est le vecteur B_i.*

Si $V(X)$ est inversible modulo $E(X)$ alors :

- *la matrice $\mathbf{B}$ définit un sous-réseau $\mathfrak{L}' \subseteq \mathfrak{L}$ de rang n (i.e. $B = (B_0, \ldots, B_{n-1})$ est une base de $\mathfrak{L}'$),*
- *et $V \in \mathfrak{L}'$.*

Preuve. Les B_i sont linéairement indépendants. En effet, supposons qu'il existe un vecteur non nul $(t_0, t_1, \ldots, t_{n-1}) \in \mathbb{Z}^n$ tel que $\sum_{i=0}^{n-1} t_i\, B_i = 0$. Cela signifie que $\sum_{i=0}^{n-1} t_i\, X^i\, V(X) = 0 \bmod E(X)$, ou de façon équivalente $T(X)\, V(X) = 0 \bmod E(X)$, avec $T(X) = \sum_{i=0}^{n-1} t_i\, X^i$. Alors $T(X)\, V(X)\, V^{-1}(X) \bmod E(X) = T(X) = 0$, puisque $V(X)$ est inversible modulo $E(X)$ et le degré de $T(X)$ est au plus $n-1$. Ainsi les lignes de $\mathbf{B}$ forment une base d'un sous-réseau $\mathfrak{L}' \subseteq \mathfrak{L}$ de rang n, et $V \in \mathfrak{L}'$. □

Corollaire I.2.1. *Soit V un vecteur non nul de $\mathfrak{L}$, le réseau de rang n défini par $\mathbf{A}$, Equ.* (I.1)*.*

Si $E(X)$ est irréductible alors :

- *$\mathbf{B}$ est une base d'un sous-réseau $\mathfrak{L}' \subseteq \mathfrak{L}$ de rang n,*
- *$V \in \mathfrak{L}'$.*

Preuve. Si $E(X)$ est irréductible, alors $V(X)$ est inversible et la Proposition I.2.3 garantit que $B = (B_0, \ldots, B_{n-1})$ est une base de $\mathfrak{L}'$, $\mathfrak{L}' \subseteq \mathfrak{L}$ de rang n, et $V \in \mathfrak{L}'$. □

Ainsi, la deuxième stratégie consiste à prendre un vecteur court $V \in \mathfrak{L}$, satisfaisant la borne de Minkowski, $\|V\|_\infty < \alpha\,(p)^{1/n}$ avec $\alpha \in]0, 1[$. Ce vecteur peut être utilisé comme une heuristique pour minimiser $\|\mathbf{B}\|_1$ sans toutefois assurer sa minimalité. Nous avons alors $\|\mathbf{B}\|_1 \leq \|V\|_1 \|(\mathbf{C}^0|\mathbf{C}^1|\cdots|\mathbf{C}^{n-1})\|_1$, puisque la norme matricielle subordonnée à la norme vectorielle $\|\cdot\|_1$ est consistante. Ainsi $\|\mathbf{B}\|_1 \leq n\,\alpha\,(p)^{1/n} \|(\mathbf{C}^0|\mathbf{C}^1|\cdots|\mathbf{C}^{n-1})\|_1$, où la norme subordonnée de $(\mathbf{C}^0|\mathbf{C}^1|\cdots|\mathbf{C}^{n-1})$ est le max des normes 1 de ses colonnes.

En prenant tout vecteur non nul V de $\mathfrak{L}$, nous pouvons calculer une base B d'un sous-réseau $\mathfrak{L}'$ et le Théorème I.2.2 indique que si $\rho > \frac{1}{2}\|B\|_1$ alors $\mathfrak{B} = (p, n, \gamma, \rho)_E$ est un PMNS.

Dans la dernière stratégie nous proposons un autre moyen de calculer la base B de $\mathfrak{L}'$.

Corollaire I.2.2. *Soit $\mathfrak{L}$ le réseau de rang n engendré par $\mathbf{A}$ (Equ.* (I.1)*), et soit $\mathfrak{L}_D$ le réseau de rang n dans $\mathbb{Z}^{n^2}$ défini par $\mathbf{D} = (\mathbf{A}|\mathbf{A}.\mathbf{C}^1|\cdots|\mathbf{A}.\mathbf{C}^{n-1})$, la matrice obtenue par concaténation horizontale des matrices $\mathbf{A}.\mathbf{C}^i$, $0 \leq i \leq n-1$. Alors pour tout $\overline{V} = (V_0|V_1|\cdots|V_{n-1}) \in \mathfrak{L}_D$ tel que $\overline{V} \neq (0)^{n^2}$:*

Si $E(X)$ est irréductible alors :

1. $V_0 \in \mathfrak{L}$,
2. $(V_0, V_1, ..., V_{n-1})$ *est une base de* $\mathfrak{L}' \subseteq \mathfrak{L}$.

Preuve. V_0 est une combinaison linéaire des lignes de A, donc il appartient à $\mathfrak{L}$. Ensuite, puisque $V_i = V_0.C^i$, pour tout $i \geq 1$, alors, grâce au Corollaire I.2.1, la famille de vecteurs $(V_0, V_1, ..., V_{n-1})$ est une base d'un sous-réseau $\mathfrak{L}' \subseteq \mathfrak{L}$. □

Ainsi, la dernière stratégie consiste à choisir un vecteur court $(V_0|V_1|\cdots|V_{n-1})$ de $\mathfrak{L}_D$ et à construire la base B de $\mathfrak{L}$ à partir de ce dernier.

I.2.4 Quelques exemples

Dans ces exemples, nous donnons la valeur de $\|\mathbf{B}\|_1$ pour chaque approche de base réduite : réduction LLL ou BKZ ou HKZ de $\mathbf{A}$, ou celle du Corollaire I.2.1 ou du Corollaire I.2.2. Nous remarquons que les deux dernières approches offrent les meilleurs résultats pour des polynômes $E(X)$ avec de petits coefficients. Dans la Section I.4.4 et à l'Annexe A, nous donnons des résultats expérimentaux avec des recherches exhaustives.

Exemple I.2.1.

$p = 112848483075082590657416923680536930196574208889254960005437791530871071177777$

$n = 8$, $E(X) = X^8 + X^2 + X + 1$,

$\gamma = 14916364465236885841418726559687117741451144740538386254842986662265545588774$

LLL : $\|\mathbf{B}\|_1 = 16940155314$ BKZ : $\|\mathbf{B}\|_1 = 15289909984$

HKZ : $\|\mathbf{B}\|_1 = 15289909984$

Cor. I.2.1 : $\|\mathbf{B}\|_1 = 13881325101$ Cor. I.2.2 : $\|\mathbf{B}\|_1 = 12883199915$

Exemple I.2.2.

$p = 96777329138546418411606037850670691916278980249035796845487391462163262877831$

$n = 8$, $E(X) = X^8 - X^4 - 1$,

$\gamma = 663781196091410433177282902170533852564491454075567270041323732701464555754 61$

LLL : $\|\mathbf{B}\|_1 = 17955608045$ BKZ : $\|\mathbf{B}\|_1 = 17955608045$

HKZ : $\|\mathbf{B}\|_1 = 17955608045$

Cor. I.2.1 : $\|\mathbf{B}\|_1 = 11628752571$ Cor. I.2.2 : $\|\mathbf{B}\|_1 = 10489321362$

Exemple I.2.3.

$p = 942340893781791483036613393513425006589105952996805455006024534248829782903 51$

$n = 8$, $E(X) = X^8 + X^4 - X^3 + 1$,

$\gamma = 558574895772927518550090985515008520396183509258372756203761663983256785251 51$

LLL : $\|\mathbf{B}\|_1 = 12305954812$ BKZ : $\|\mathbf{B}\|_1 = 12305954812$

HKZ : $\|\mathbf{B}\|_1 = 12305954812$

Cor. I.2.1 : $\|\mathbf{B}\|_1 = 15570303402$ Cor. I.2.2 : $\|\mathbf{B}\|_1 = 14857375293$

Exemple I.2.4.

$p = 967773291385464184116060378506706919162789802490357968454873914621632628778 31$

$n = 8$, $E(X) = X^8 + 6$,

$\gamma = 553827465432951480218172661890659023793629523755366606254280807067648457 2674$

LLL :	$\|\mathbf{B}\|_1 = 12509178620$	BKZ :	$\|\mathbf{B}\|_1 = 12509178620$
HKZ :	$\|\mathbf{B}\|_1 = 12509178620$		
Cor. I.2.1 :	$\|\mathbf{B}\|_1 = 47611052126$	Cor. I.2.2 :	$\|\mathbf{B}\|_1 = 40733847267$

I.3 Des polynômes de réduction adaptés pour les PMNS

Dans le Théorème I.2.1, nous montrons que si $E(X)$ est un polynôme irréductible, alors nous pouvons définir un PMNS $\mathfrak{B} = (p, n, \gamma, \rho)_E$ dépendant de $E(X)$. À présent, dans le but d'optimiser l'algorithme de réduction modulo $E(X)$ par rapport à la taille des chiffres dans le système $\mathfrak{B} = (p, n, \gamma, \rho)_E$, $E(X)$ doit satisfaire certains critères. Nous définissons alors ce que doit être un *polynôme de réduction irréductible adapté au PMNS*.

I.3.1 Polynômes de réduction adaptés au PMNS

Définition I.3.1. Un polynôme $E(X)$ est un polynôme de réduction adapté au PMNS, si :

1. $E(X)$ est irréductible dans $\mathbb{Z}[X]$,
2. $E(X) = X^n + a_k X^k + \cdots + a_1 X + a_0 \in \mathbb{Z}[X]$, avec $n \geq 2$ et $k \leq \frac{n}{2}$,
3. la plupart des coefficients a_i sont nuls, et les autres sont très petits (si possible égaux à ± 1) par rapport à $p^{1/n}$.

Le deuxième point garantit que la réduction polynomiale modulo $E(X)$ d'un polynôme $T(X)$ de degré inférieur à $2n$ est réalisée en deux étapes, i. e. $T(X) = T_1(X)\, X^n + T_0(X)$ avec $T_1(X)$ et $T_0(X)$ de degré inférieur à n, et $X^n \bmod E(X) = -(\sum_{i=0}^{k} a_i\, X^i) \bmod E(X)$.

Le troisième point permet de donner une borne sur les coefficients de $T(X) \bmod E(X)$. Réduire $T(X)$ modulo $E(X)$ revient à multiplier le vecteur des coefficients de $T(X)$ par la matrice $\mathbf{S}$ de taille $(2\,n - 1) \times n$, dont les lignes représentent les coefficients de chaque puissance X^i modulo $E(X)$, pour $0 \leq i \leq 2\,n - 2$. Ces puissances sont définies par $X^i \bmod E(X) = -(\sum_{l=i-n}^{k+i-n} a_{l-i+n} X^l)$ pour $n \leq i < 2\,n - k$ et $X^i \bmod E(X) = -(\sum_{l=i-n}^{n-1} a_{l+n-i} X^l + \sum_{l=0}^{2k-2n+i} (\sum_{j=0}^{l} a_j\, a_{l+2\,n-2\,j}) X^l)$ pour $2\,n - k \leq i \leq 2\,n - 2$. Nous notons $s_{i,j}$ le coefficient de la i-ième ligne, j-ième colonne de $\mathbf{S}$. En posant $s = \|\mathbf{S}\|_1$, $(\|\mathbf{S}\|_1 = \max_j \left\{ \sum_{i=0}^{n-1} |s_{i,j}| \right\})$, on obtient $\|T(X) \bmod E(X)\|_\infty < s\|T(X)\|_\infty$. Par conséquent, si $G(X)$ et $F(X)$ sont deux éléments du PMNS, i.e. $\|F(X)\|_\infty < \rho$ et $\|G(X)\|_\infty < \rho$ alors $\|F(X) \times G(X)\|_\infty < n\rho^2$ et $\|F(X) \times G(X) \bmod E(X)\|_\infty < sn\rho^2$. Comme s est une

somme de produits de coefficients de E, ces derniers doivent être très petits (si possible égaux à ± 1) par rapport à $p^{1/n}$ ($p^{1/n} \simeq \rho$ quand le système n'est pas redondant).

Pour le premier point, nous devons trouver un polynôme irréductible. Ensuite, pour être éligible au rôle de polynôme de réduction adapté au PMNS, le polynôme $E(X)$ doit satisfaire les deux autres conditions. Dans la suite, nous adaptons des critères classiques d'irréductibilité et donnons des exemples de polynômes irréductibles avec peu de coefficients non nuls.

I.3.2 Critères classiques d'irréductibilité des polynômes

Pour vérifier le premier point, nous pouvons utiliser des critères généraux comme ceux de Schönemann-Eisenstein ou Dumas [Dum06] ou la généralisation donnée par N. C. Bonciocat dans [Bon15], que nous adaptons à notre objectif : un polynôme unitaire $E(X) = X^n + a_k X^k + \cdots + a_1 X + a_0 \in \mathbb{Z}[X]$.

Proposition I.3.1 (du critère de Dumas [Dum06])**.** *Nous supposons que s'il existe un premier μ et un entier α tel que, $\mu^\alpha \mid a_0$, $\mu^{\alpha+1} \nmid a_0$ et $\mu^{\lceil \alpha(n-i)/n \rceil} \mid a_i$ pour $1 \leq i \leq n$, et* $\mathrm{pgcd}(\alpha, n) = 1$*, alors $E(X) = X^n + a_k X^k + \cdots + a_1 X + a_0$ est irréductible dans $\mathbb{Z}[X]$.*

Par exemple, $E(X) = X^n + \mu X^k + \mu$ est prouvé irréductible par ce critère. Si $k < n/2$ et $\mu << p^{1/n}$, alors $E(X)$ est un polynôme de réduction adapté au PMNS.

Proposition I.3.2 (du Corollaire 1.2 [Bon15])**.** *Soit $E(X) = X^n + a_k X^k + \cdots + a_1 X + a_0$, $a_0 \neq 0$, soit $t \geq 2$ et soient $\mu_1, \ldots, \mu_t$ des nombres deux-à-deux distincts et $\alpha_1, \ldots, \alpha_t$ des entiers positifs. Si, pour $j = 1, \ldots, t$, et $i = 0, \ldots, k$, $\mu_j^{\alpha_j} \mid a_i$ et $\mu_j^{\alpha_j+1} \nmid a_0$ et* $\mathrm{pgcd}(\alpha_1, \ldots, \alpha_t, n) = 1$ *alors $E(X)$ est irréductible dans $\mathbb{Z}[X]$.*

Par exemple, $E(X) = X^n + \mu_1^{\alpha_1}\mu_2^{\alpha_2} X^k + \mu_1^{\alpha_1}\mu_2^{\alpha_2}$ avec $\mathrm{pgcd}(\alpha_1, \alpha_2, n) = 1$, est prouvé irrédutible par ce critère. Si $k < n/2$ et $\mu_1^{\alpha_1}\mu_2^{\alpha_2} << p^{1/n}$, alors $E(X)$ est un polynôme de réduction adapté au PMNS.

I.3.3 Polynômes cyclotomiques de réduction adaptés au PMNS

Un ensemble bien connu de polynômes irréductibles dans $\mathbb{Z}[X]$ est l'ensemble des polynômes cyclotomiques. Nous désignons par `ClassCyclo(n)` la classe des polynômes cyclotomiques adaptés au PMNS, de degré n.

Proposition I.3.3. `ClassCyclo(n)` $\neq \emptyset$ *si et seulement si $n = 2^i 3^j$ avec $i \geq 0$, $j \geq 0$.*

Preuve. Rappelons d'abord quelques propriétés classiques sur les polynômes cyclotomiques :

(a) Soit $m \in \mathbb{N}^*$, le m-ième polynôme cyclotomique est défini comme :

$$\Phi_m(X) = \prod_{\substack{k=1 \\ \text{pgcd}(k,m)=1}}^{m} (X - \zeta^k).$$

avec ζ une racine primitive m-ième de l'unité, $\deg \Phi_m(X) = \varphi(m)$ (φ est l'indicatrice d'Euler) et $\Phi_m(X)$ est un polynôme unitaire, i.e. $a_{\varphi(m)} = 1$.

(b) Soient $m > 1$ et $n = \varphi(m)$, $\Phi_m(X)$ est réciproque, $\Phi_m(X) = X^n \Phi_m(\frac{1}{X})$ (nous pouvons prouver que ces deux polynômes ont les même racines), i.e. $a_i = a_{n-i}$.

(c) Soient $m = \prod_{i=1}^{r} p_i^{e_i}$ et $m_0 = \prod_{i=1}^{r} p_i$, alors $\Phi_m(X) = \Phi_{m_0}(X^{m/m_0})$.

(d) Soit $m = 2t$ avec t impair et $t \geq 3$ alors $\Phi_m(X) = \Phi_t(-X)$.

(e) Soit μ un entier et p un premier ne divisant pas μ, alors $\Phi_{p\mu}(X) = \frac{\Phi_\mu(X^p)}{\Phi_\mu(X)}$.

Soit n tel que `ClassCyclo(n)` $\neq \emptyset$, alors il existe un entier m tel que $n = \varphi(m)$ et $\Phi_m(X)$ est un polynôme cyclotomique de réduction adapté. D'après le point (b), cela signifie que $\Phi_m(X)$ est soit $X^n + 1$, soit $X^n + a_{n/2}X^{n/2} + 1$. Donc $\Phi_m(X)$ a 2 ou 3 coefficients.

Soient $m = \prod_{i=1}^{r} p_i^{e_i}$, $e_i \geq 0$, $p_1 = 2$, $p_2 = 3$ ... et $m_0 = \prod_{\substack{i=1 \\ e_i \neq 0}}^{r} p_i$. D'après le point (c), si $e_1 = 0$ alors $\Phi_m(X) = \Phi_{m_0}(X^{m/m_0})$. Si $e_1 \neq 0$, d'après (c) et (d), $\Phi_m(X) = \Phi_{m_0/2}(-X^{m/m_0})$. Ainsi, dans tous les cas, $\Phi_m(X)$ a le même nombre de coefficients que $\Phi_{m'}(X)$, où $m' = \prod_{\substack{i=2 \\ e_i \neq 0}}^{r} p_i$.

Supposons que $r \geq 3$ alors il existe $j \geq 3$ tel que $m' = p_j\mu$ et $\text{pgcd}(p_j, \mu) = 1$ (en d'autres termes m' est divisible par un premier autre que 3). D'après (e), $\Phi_{p_j\mu}(X) = \frac{\Phi_\mu(X^{p_j})}{\Phi_\mu(X)}$.

Soit $\Phi_\mu(X) = X^t + a_s X^s + \cdots + 1$, $(t > 0)$, alors $\frac{\Phi_\mu(X^{p_j})}{\Phi_\mu(X)} = X^{(p_j-1)t} - a_s X^{(p_j-2)t+s} + \cdots = \Phi_{m'}(X)$. Si $\Phi_{m'}(X)$ a seulement deux coefficients alors $(p_j - 2)t = -s$, puisque $t > 0$. Cela implique $p_j = 2$ ce qui est impossible puisque $j \geq 3$.

Si $\Phi_{m'}(X)$ a seulement trois coefficients, d'après (b), cela implique que $(p_j - 2)t + s = (p_j - 1)t/2$, qui donne $2s = (3 - p_j)t$. Maintenant, $2s \geq 0$ et $(3 - p_j)t < 0$ puisque $p_j \geq 5$.

D'où $r \leq 2$ et alors $m = 2^{e_1}3^{e_2}$. D'après la définition des polynômes cyclotomiques, nous déduisons que si $\Phi_m(X)$ a deux coefficients alors $m = 2^{e_1}$ avec $e_1 \geq 1$, et si $\Phi_m(X)$ a trois coefficient alors $m = 2^{e_1}3^{e_2}$ avec $e_1 \geq 0$ et $e_2 \geq 1$.

Ainsi les polynôme cyclotomiques adaptés sont :

— $\Phi_{2^i}(X) = X^{2^{i-1}} + 1$, d'où $n = 2^{i-1}$ avec $i \in \mathbb{N}^*$,

— $\Phi_{3^j}(X) = X^{2.3^{j-1}} + X^{3^{j-1}} + 1$, d'où $n = 2.3^{j-1}$ avec $j \in \mathbb{N}^*$,

— $\Phi_{2^i.3^j}(X) = X^{2^i.3^{j-1}} - X^{2^{i-1}.3^{j-1}} + 1$, d'où $n = 2^i.3^{j-1}$ pour $i, j \in \mathbb{N}^*$.

Réciproquement, il est clair que les polynômes ci-dessus sont par construction des polynômes cyclotomiques adaptés.

□

I.3.4 Quadrinômes de réduction adaptés au PMNS à coefficients dans $\{-1, 1\}$

Dans [FJ06], Finch and Jones donnent des critères d'irréductibilité pour les polynômes $X^a+\beta X^b+\gamma X^c+\delta$ avec $\beta, \gamma, \delta \in \{-1, 1\}$ et $a > b > c > 0$. Ils supposent que $\text{pgcd}(a, b, c) = 2^t m$ avec m impair et ils notent $a' = a/2^t$, $b' = b/2^t$ et $c' = c/2^t$. Ils définissent $\overline{a} = \text{pgcd}(a', b' - c')$, $\overline{b} = \text{pgcd}(b', a' - c')$ et $\overline{c} = \text{pgcd}(c', a' - b')$.

Proposition I.3.4 (Théorème 2 dans [FJ06])**.** *Le quadrinôme $X^a + \beta X^b + \gamma X^c + \delta$ est irréductible dans $\mathbb{Z}[X]$, si et seulement si il satisfait une des conditions suivantes :*

1. $(\beta, \gamma, \delta) = (1, 1, 1)$ *et* $\overline{a}\overline{b}\overline{c} \equiv 1 \bmod 2$
2. $(\beta, \gamma, \delta) = (-1, 1, 1)$, $b' - c' \not\equiv 0 \bmod 2\overline{a}$, $b' \not\equiv 0 \bmod 2\overline{b}$ *et* $a' - b' \not\equiv 0 \bmod 2\overline{c}$
3. $(\beta, \gamma, \delta) = (1, -1, 1)$, $b' - c' \not\equiv 0 \bmod 2\overline{a}$, $a' - c' \not\equiv 0 \bmod 2\overline{b}$ *et* $c' \not\equiv 0 \bmod 2\overline{c}$
4. $(\beta, \gamma, \delta) = (1, 1, -1)$, $a' \not\equiv 0 \bmod 2\overline{a}$, $b' \not\equiv 0 \bmod 2\overline{b}$ *et* $c' \not\equiv 0 \bmod 2\overline{c}$
5. $(\beta, \gamma, \delta) = (-1, -1, -1)$, $a' \not\equiv 0 \bmod 2\overline{a}$, $a' - c' \not\equiv 0 \bmod 2\overline{b}$ *et* $a' - b' \not\equiv 0 \bmod 2\overline{c}$

Nous appelons cette classe de quadrinômes de réduction adaptés `QuadrinomialClass`*, et* `QuadrinomialClass(n)` *est le sous-ensemble de ces quadrinômes de degré n.*

Par exemple, $E(X) = X^{2^t 7m} + X^{2^t 5m} + X^{2^t 3m} + 1$ est un quadrinôme de réduction adapté au PMNS.

I.3.5 Trinômes de réduction adaptés au PMNS à coefficients dans $\{-1, 1\}$

Dans cette partie, nous nous référons à un article de W.H. Mills [Mil85] et un autre de W. Ljunggren [Lju60]. Le premier donnant un critère sur les quadrinômes et les racines de l'unité, le second une application aux trinômes.

Proposition I.3.5. *Nous notons* $\text{pgcd}(n, m) = d$ *et* $n = n_1 d$, $m = m_1 d$. *Si* $n_1 + m_1 \not\equiv 0 \bmod 3$ *alors le polynôme* $X^n+\beta X^m+\delta$ *avec* $\delta, \beta \in \{-1, 1\}$ *et* $n \geq 2\,m > 0$ *est irréductible dans* $\mathbb{Z}[X]$.

La classe des trinômes de réduction adaptés vérifiant ces critères est nommée `TrinomialClass`*, et* `TrinomialClass(n)` *est le sous-ensemble de ces tririnômes de degré n.*

Preuve. Dans [Mil85], Millers donne un résultat sur la factorisation des quadrinômes lorsqu'un facteur irréductible apparaît, et que le second facteur a seulement quelques racines de l'unité pour racines.

Nous transformons comme dans [Lju60], $E(X) = X^n + \beta X^m + \delta$ en quadrinôme :

$$(X^n + \beta X^m + \delta)\,(X^n - \delta) = X^{2n} + \beta X^{n+m} - \beta\delta X^m - 1 = F(X).$$

Le Théorème 2 de [Mil85] stipule que si $F(X) = A(X)\,E(X)$, où chaque racine de $A(X)$ et aucune racine de $E(X)$ ne sont racines de l'unité, alors $E(X)$ est irréductible, sauf s'il existe r tel que :

— $(2n, n+m, m) = (8r, 7r, r)$ et $(\beta, \delta) = (1, -1)$, ou $(-1, -1)$,

— ou $(2n, n+m, m) = (8r, 4r, 2r)$ et $(\beta, \delta) = (1, -1)$,

— ou $(2n, n+m, m) = (8r, 6r, 4r)$ et $(\beta, \delta) = (-1, -1)$.

En examinant ces exceptions, $2n = 8r$ implique $n = 4r$, et $n+m = 4r+m$ revient soit à $m = 3r$ pour $n+m = 7r$, soit à $m = 0$ pour $n+m = 4r$, soit à $m = 2r$ pour $n+m = 6r$. Comme aucun des trois cas décrits ci-dessus ne satisfait l'égalité sur le dernier exposant, cela est impossible.

Puisqu'il n'existe pas d'entier r satisfaisant l'une de ces trois contraintes, nous avons donc seulement à vérifier qu'aucune racine de $E(X)$ n'est une racine de l'unité. Remarquons d'abord que, comme $n = dn_1$ et $m = dm_1$ avec $\mathrm{pgcd}(n_1, m_1) = 1$, si γ est une racine de $E(X)$ alors γ^d est racine de $X^{n_1} + \beta X^{m_1} + \delta$. Par conséquent, si les racines de $X^{n_1} + \beta X^{m_1} + \delta$ ne sont pas des racines de l'unité, alors aucune racine de $E(X) = X^n + \beta X^m + \delta$ n'est racine de l'unité.

Supposons que γ est une racine de $X^{n_1} + \beta X^{m_1} + \delta$, qui est aussi une racine d'unité, alors il existe $t > 1$ et k avec $\mathrm{pgcd}(k, t) = 1$, tels que :

$$\lambda = e^{\frac{2ik\pi}{t}} = \cos(\frac{2k\pi}{t}) + i\sin(\frac{2k\pi}{t}).$$

Nous avons donc
$$\begin{cases} \cos(\frac{2n_1k\pi}{t}) + \beta\cos(\frac{2m_1k\pi}{t}) = -\delta \\ \sin(\frac{2n_1k\pi}{t}) + \beta\sin(\frac{2m_1k\pi}{t}) = 0 \end{cases}$$

Supposons que $\beta = 1$, alors

$$\begin{cases} \cos(\frac{2n_1k\pi}{t}) + \cos(\frac{2m_1k\pi}{t}) = 2\cos(\frac{k\pi(n_1+m_1)}{t})\cos(\frac{k\pi(n_1-m_1)}{t}) = -\delta \\ \sin(\frac{2n_1k\pi}{t}) + \sin(\frac{2m_1k\pi}{t}) = 2\sin(\frac{k\pi(n_1+m_1)}{t})\cos(\frac{k\pi(n_1-m_1)}{t}) = 0 \end{cases}$$

La dernière égalité implique que $\sin(\frac{k\pi(n_1+m_1)}{t}) = 0$ ou $\cos(\frac{k\pi(n_1-m_1)}{t}) = 0$. Puisque $\delta \neq 0$, la première équation implique que $\cos(\frac{k\pi(n_1-m_1)}{t}) \neq 0$, ainsi $\frac{k(n_1+m_1)}{t}$ est un entier.

Puisque $\text{pgcd}(k,t) = 1$, $t \mid (n_1+m_1)$. Ce dernier résultat implique que la première équation peut se réduire à

$$\cos(\frac{k\pi(n_1-m_1)}{t}) = \pm\frac{1}{2}$$

car $\delta = \pm 1$.

Cela signifie que

$$\frac{k\pi(n_1-m_1)}{t} = j\frac{\pi}{3}, \quad j = 1,2,4,5 \bmod 6.$$

Ainsi, $t \mid 3(n_1-m_1)$, puisque $\text{pgcd}(k,t) = 1$.
Supposons maintenant que $\beta = -1$, alors le système devient

$$\begin{cases} \cos(\frac{2n_1k\pi}{t}) - \cos(\frac{2m_1k\pi}{t}) = -2\sin(\frac{k\pi(n_1+m_1)}{t})\sin(\frac{k\pi(n_1-m_1)}{t}) = -\delta \\ \sin(\frac{2n_1k\pi}{t}) - \sin(\frac{2m_1k\pi}{t}) = 2\cos(\frac{k\pi(n_1+m_1)}{t})\sin(\frac{k\pi(n_1-m_1)}{t}) = 0 \end{cases}$$

La première équation implique que $\sin(\frac{k\pi(n_1-m_1)}{t}) \neq 0$, ainsi la seconde équation donne $\frac{k\pi(n_1+m_1)}{t} = j\frac{\pi}{2}$ pour j impair, qui implique $t \mid 2(n_1+m_1)$. Puisque $\frac{k\pi(n_1+m_1)}{t} = j\frac{\pi}{2}$ pour j impair, alors la première équation peut se réduire à

$$\sin(\frac{k\pi(n_1-m_1)}{t}) = \pm\frac{1}{2}$$

ce qui revient à

$$\frac{k\pi(n_1-m_1)}{t} = j\frac{\pi}{6}, \quad j = 1,5,7,11 \bmod 12.$$

Ainsi $t \mid 6(n_1-m_1)$.

Pour résumer, si γ est une racine t-ième de l'unité de $X^{n_1} + \beta X^{m_1} + \delta$ avec $\beta, \delta \in \{-1,1\}$ alors :

(a) si $\beta = 1$, $t \mid (n_1+m_1)$ et $t \mid 3(n_1-m_1)$,

(b) si $\beta = -1$, $t \mid 2(n_1+m_1)$ et $t \mid 6(n_1-m_1)$.

Le cas (a) implique que si $3 \nmid t$, alors $t \mid (n_1-m_1)$, d'où $t \mid 2n_1$ et $t \mid 2m_1$, comme $\text{pgcd}(n_1,m_1) = 1$ alors $t = 2$ et $\gamma = 1$ ou -1 est une racine de $E(X)$ ce qui est impossible.

Le cas (b) implique que si $3 \nmid t$, alors $t \mid 2(n_1-m_1)$, d'où $t = 4$ et $\gamma = i, -i, 1$ ou -1 est une racine de $E(X)$ ce qui est impossible.

Par conséquent, si une racine de $E(X)$ est une racine de l'unité alors 3 divise t, et donc $n_1 + m_1 \equiv 0 \bmod 3$.

En conclusion, si $\text{pgcd}(n_1,m_1) = 1$ et $n_1 + m_1 \not\equiv 0 \bmod 3$ alors $X^{n_1} + \beta X^{m_1} + \delta$ et $X^n + \beta X^m + \delta$ sont irréductibles.

□

Remarque I.3.1. Modification du Théorème de Ljunggren dans le cas $n_1 + m_1 \equiv 0 \bmod 3$.

Dans [Lju60], avec le Théorème 3, Ljunggren cherche à prouver l'irréductibilité du trinôme $G(X) = X^n + \delta X^m + \epsilon$, avec $\delta = \pm 1$, $\epsilon = \pm 1$. On note $d = \text{pgcd}(n, m)$, $n = n_1 d$ et $m = m_1 d$.

Le Théorème 1 indique que si un quadrinôme $F(X)$ avec des coefficients prenant les valeurs ± 1 n'a pas de zéros qui sont racines de l'unité, alors $F(X)$ est irréductible. Le Théorème 2 indique que toutes les racines possibles de l'unité de $F(X)$ sont des racines simples que l'on trouve parmi les racines de $X^{dd_i} = \pm 1$, avec $1 \leq i \leq 3$, les d_i étant les pgcd précisés dans le théorème.

En refaisant la preuve du Théorème 3, d'après le Lemme 2 du même papier, une racine possible de l'unité, γ de $G(X)$, satisfait

$$\lambda^{3n} = \epsilon \quad \text{et} \quad \lambda^{3m} = \delta\epsilon.$$

Ces équations se réécrivent

$$\lambda^{3n_1 d} = \epsilon \quad \text{et} \quad \lambda^{3m_1 d} = \delta\epsilon.$$

Dans le cas $n_1 + m_1 \equiv 0 \bmod 3$, on a $d_2 = \text{pgcd}(n_1 + m_1, 2n_1 - m_1) = 3$.

Si $d_2 = 3$, alors trois cas se présentent. Nous ne traitons ici que le 1^{er} cas, les deux autres étant très similaires. Si n_1 et m_1 sont impairs, et $\delta = 1$, alors on obtient

$$\lambda^{3d} = \epsilon.$$

Si $\lambda^d = \pm 1$, alors $\lambda^d = \epsilon$, car $\lambda^{3d} = \epsilon$ et 3 est impair. Or ici, on constate qu'une racine possible de l'unité, λ de $G(X)$ ne peut satisfaire l'équation $\lambda^d = \pm 1$. On calcule donc le polynôme des racines de l'unité recherchées en posant :

$$\frac{X^{3d} - \epsilon}{X^d - \epsilon} = X^{2d} + \epsilon X^d + 1.$$

Pour les deux autres cas, les polynômes trouvés sont respectivement :

— $X^{2d} + \delta X^d + 1$, pour n_1 pair (et donc m_1 impair car $\text{pgcd}(n_1, m_1) = 1$), et $\epsilon = 1$.

— $X^{2d} + \epsilon X^d + 1$ pour m_1 pair (et donc n_1 impair), et $\delta = \epsilon$.

Le polynôme qui apparaît dans la factorisation de $G(X)$ dans ces 3 cas, est donc

$$X^{2d} + \delta^{m_1} \epsilon^{n_1} X^d + 1.$$

On vérifie rapidement en fonction des conditions sur n_1, m_1, δ et ϵ pour chacun des trois

cas précédents que ce polynôme est exact.

Le problème avec le théorème de Ljunggren est que le polynôme présenté comme facteur de $G(X)$ dans ces trois cas est :

$$X^{2d} + \delta^{m_1 d}\epsilon^{n_1 d}X^d + 1 = X^{2d} + (\delta^{m_1}\epsilon^{n_1}X)^d + 1.$$

Or, changer le signe de X^d n'est pas la même chose que changer le signe de X lui-même si d est pair.

Par exemple, lorsque le pgcd $d = 2$, on a pour le 1^{er} cas, $\lambda^{3d} = \lambda^6 = \epsilon$ avec $\delta = 1$. Pour $\epsilon = 1$, on obtient :

$$\frac{X^6 - 1}{X^2 - 1} = X^4 + X^2 + 1.$$

Et pour $\epsilon = -1$, on obtient :

$$\frac{X^6 + 1}{X^2 + 1} = X^4 - X^2 + 1.$$

Le polynôme exhibé dans l'article renverrait quant à lui $X^4 + X^2 + 1$ dans les deux cas, d'où la réécriture des exposants.

Les trinômes satisfaisant l'un des trois cas décrits ci-dessus ont donc pour facteur $X^{2d} + \delta^{m_1}\epsilon^{n_1}X^d + 1$.

I.3.6 Binômes de réduction adaptés au PMNS

Proposition I.3.6. *Nous notons $c = \prod_{j=1}^{k} p_j^{m_j}$, avec p_j des nombres premiers distincts et m_j des entiers positifs.*

Si $\text{pgcd}(m_1, \ldots, m_k, n) = 1$ *alors le polynôme $X^n + c$ avec $c \in \mathbb{Z}$, $|c| \geq 2$, est irréductible dans $\mathbb{Z}[X]$.*

Nous appelons cette classe de polynômes adaptés `BinomialClass`*, et pour n et c satisfaisant cette proposition,* `BinomialClass(n, c)` *est le singleton $\{X^n + c\}$.*

Preuve. Il s'agit d'une application directe du Corollaire 1.2 d'un article de Nicolae Ciprian Bonciocat [Bon15].

□

I.3.7 Polynômes de réduction adaptés au PMNS aux racines complexes bornées

Les deux propositions données dans cette partie sont inspirées du critère d'irréductibilité de Perron, qui est prouvé grâce au théorème de Rouché [Bon10].

Proposition I.3.7. *Soit un entier fixé $n \geq 2$, un premier μ, et $P(X) = X^n + \sum_{i=1}^{n/2} \epsilon_i X^i \pm \mu$ avec $\epsilon_i \in \{-1, 0, 1\}$.*

Si $\mu > 1 + \sum_{i=1}^{n/2} |\epsilon_i|$ alors le polynôme $P(X)$ est irréductible dans $\mathbb{Z}[X]$.

De tels polynômes constituent la cinquième classe de polynômes de réduction adaptés. Nous appelons cette classe `ClassPrimeCst`*, et* $\texttt{ClassPrimeCst}(\texttt{n}, \mu)$ *est le sous-ensemble des polynômes de cette classe pour un $n \geq 2$ et un premier μ fixés.*

Remarque I.3.2. Si $\mu > n/2 + 1$, alors $\texttt{ClassPrimeCst}(\texttt{n}, \mu)$ contient $3^{n/2} \cdot 2$ éléments (pour chaque ϵ_i, trois possibilités), sinon cette classe contient $\sum_{i=0}^{\mu-2} \binom{n/2}{i} 2^{i+1}$ éléments.

Preuve. Puisque $\mu > 1 + \sum_{i=1}^{n/2} |\epsilon_i|$, alors il existe $\delta > 1$ tel que $\mu > \delta^n \left(1 + \sum_{i=1}^{n/2} |\epsilon_i|\right)$.

Considérons $\mathcal{C} = \{z \in \mathbb{C} \ / \ |z| = \delta\}$, $P(X) = X^n + \sum_{i=1}^{n/2} \epsilon_i X^i + \epsilon\mu$ (avec $\epsilon_i \in \{-1, 0, 1\}$ et $\epsilon \in \{-1, 1\}$), $F(X) = \epsilon\mu$ et $G(X) = P(X) - F(X)$.

Sur $\mathcal{C}$ nous avons $|G(z)| \leq \delta^n \left(1 + \sum_{i=1}^{n/2} |\epsilon_i|\right) < \mu = |F(z)|$.

Puisque $F(z)$ and $G(z)$ sont des fonctions holomorphes, le théorème de Rouché stipule que $F(z)$ et $P(z) = F(z) + G(z)$ ont le même nombre de racines à l'intérieur de $\mathcal{C}$. Ainsi $P(z)$ n'a pas de racine dans $\mathcal{C}$ puisque $F(z)$ est constant. En d'autres termes, toute racine α de $P(z)$ satisfait $|\alpha| \geq \delta > 1$.

Supposons à présent que $P(X)$ est réductible dans $\mathbb{Z}[X]$. Alors, $P(X) = H(X)\,Q(X)$ avec $H(X)$ et $Q(X)$ deux polynômes unitaires. Comme $|P(0)| = \mu$ (un nombre premier), on peut supposer que $|H(0)| = \mu$ et $|Q(0)| = 1$. Par ailleurs $\prod |z_i| = 1$, où z_i sont toutes les racines complexes de $Q(X)$. Or les racines de $Q(X)$ sont aussi racines de $P(X)$, ce qui est impossible puisque toute racine α de $P(X)$ vérifie $|\alpha| \geq \delta > 1$.

Par conséquent, $P(X)$ est irréductible dans $\mathbb{Z}[X]$. □

Proposition I.3.8. *Soit $n \geq 2$ fixé, et $P(X) = X^n + \sum_{i=2}^{n/2} \epsilon_i X^i + a_1 X \pm 1$ avec $\epsilon_i \in \{-1, 0, 1\}$ et $a_1 \in \mathbb{Z}^*$.*

Si $|a_1| > 2 + \sum_{i=2}^{n/2} |\epsilon_i|$ alors le polynôme $P(X)$ est irréductible dans $\mathbb{Z}[X]$.

On appelle cette classe `ClassPerron`*, et* $\texttt{ClassPerron}(\texttt{n}, \texttt{a}_1)$ *est le sous-ensemble des polynômes de cette classe pour un $n \geq 2$ et $a_1 \in \mathbb{Z}^*$ fixés.*

Remarque I.3.3. Si $a_1 > n/2+1$, alors $\texttt{ClassPerron}(\texttt{n}, \texttt{a}_1)$ contient $3^{n/2-1} \cdot 2$ éléments, sinon cette classe contient $\sum_{i=0}^{a_1-3} \binom{n/2-1}{i} 2^{i+1}$ éléments.

Preuve. La preuve est similaire à la précédente. À partir de $|a_1| > 2 + \sum_{i=2}^{n/2} |\epsilon_i|$, nous pouvons déduire qu'il existe $\delta > 1$ tel que $|a_1| > \delta^n \left(2 + \sum_{i=2}^{n/2} |\epsilon_i| \right)$. Alors, d'après le théorème de Rouché, $P(z)$ et $F(z) = a_1 z$ ont le même nombre de racines à l'intérieur de $\mathcal{C} = \{z \in \mathbb{C} \ / \ |z| = \delta\}$. Ainsi $P(z)$ possède seulement une racine dont le module est strictement inférieur à δ.

Par ailleurs, si $P(X)$ est réductible dans $\mathbb{Z}[X]$, alors $P(X) = H(X)\,Q(X)$, avec $H(X)$ et $Q(X)$ deux polynômes unitaires et $|H(0)| = |Q(0)| = 1$. Ainsi $H(z)$ possède au moins une racine z_H telle que $|z_H| \leq 1$ et $Q(z)$ possède au moins une racine z_Q telle que $|z_Q| \leq 1$. Cela signifie que $P(z)$ a deux racines au moins à l'intérieur de $\mathcal{C}$, ce qui est impossible.

Par conséquent, $P(X)$ est irréductible dans $\mathbb{Z}[X]$.

□

I.4 Nombre de PMNS à partir des racines du polynôme de réduction modulo p

Dans cette section, nous déterminons pour chaque classe, les polynômes de réduction qui ont une ou plusieurs racines γ dans $\mathbb{Z}/p\mathbb{Z}$. Le nombre de PMNS constructibles à partir d'un tel polynôme correspond au nombre de ses racines dans $\mathbb{Z}/p\mathbb{Z}$, et chaque racine sera utilisée comme base d'un système.

Comme nous devons exposer plusieurs PMNS compacts avec une arithmétique efficace sur les représentations à partir d'un premier p et d'un nombre de chiffres n, trouver des polynômes de réduction de degré n pertinents est crucial. Maintenant que nous avons décrit des classes de polynômes irréductibles avec des propriétés de réduction spécifiques, nous devons identifier pour un premier p ceux qui ont au moins une racine dans $\mathbb{Z}/p\mathbb{Z}$, et si possible, combien.

Nous commençons par une présentation de deux cas particuliers où les polynômes de réduction sont cyclotomiques ou binomiaux, puis nous proposons une méthode dans le cas général, qui fonctionne pour tout polynôme entier irréductible.

I.4.1 Nombre de PMNS avec un polynôme cyclotomique de réduction

Proposition I.4.1. *Soit $p > 2$ un nombre premier, et un entier $m \geq 3$ tel que $m|(p-1)$ alors le polynôme cyclotomique $\Phi_m(X)|(X^{p-1}-1)$ et $\Phi_m(X)$ a $\varphi(m)$ racines dans $\mathbb{Z}/p\mathbb{Z}$,*

avec φ l'indicatrice d'Euler.

Preuve. Nous avons, $(X^{p-1}-1) = \prod_{\xi_i \in (\mathbb{Z}/p\mathbb{Z})^*} (X-\xi_i) = \prod_{d|(p-1)} \Phi_d(X)$.

Alors $\Phi_m(X) \mid \prod_{\xi_i \in (\mathbb{Z}/p\mathbb{Z})^*} (X-\xi_i)$, et $\Phi_m(X)$ a $\varphi(m)$ (son degré) racines dans $\mathbb{Z}/p\mathbb{Z}$ □

Nous appliquons la Proposition I.4.1 aux différents polynômes cyclotomiques de la classe `ClassCyclo(n)` introduit dans la Proposition I.3.3.

Corollaire I.4.1. *Soit p premier, $n \geq 2$ tel que $n = 2^i 3^j$, avec $i, j \in \mathbb{N}$.*

- *Si $i > 0$, $j = 0$, et $2\,n$ divise $p-1$, et $E(X) = \Phi_{2n}(X) = X^n + 1$,*
- *Si $i = 1$, $j \geq 0$, et $3\,n\,/\,2$ divise $p-1$, et $E(X) = \Phi_{\frac{3n}{2}}(X) = X^n + X^{\frac{n}{2}} + 1$,*
- *Si $i \geq 1$, $j \geq 0$, et $3\,n$ divise $p-1$, et $E(X) = \Phi_{3n}(X) = X^n - X^{\frac{n}{2}} + 1$,*

alors, il existe n PMNS $(p, n, \gamma_i, \rho)_{E(X)}$ avec γ_i une des n racines distinctes modulo p de $E(X)$.

Exemple I.4.1. Construction de 8 PMNS avec un polynôme de réduction cyclotomique pour $p = 22777$ et $n = 4$ (Table I.1)

TABLE I.1 – Bornes sur les chiffres des PMNS obtenus à partir des polynômes de `ClassCyclo(4)` pour $p = 22777$, pour chaque approche de base réduite.

$E(X)$	γ	$\rho_{\min}$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^4+1	19189	9	11.5	11.5	17.5	17.5	17.5
	11890	9	11.5	11.5	17.5	17.5	17.5
	10887	9	11.5	11.5	17.5	17.5	17.5
	3588	9	11.5	11.5	17.5	17.5	17.5
X^4-X^2+1	22191	9	13.5	13.5	16.5	16.5	16.5
	17452	9	13.5	13.5	16.5	16.5	16.5
	5325	9	13.5	13.5	16.5	16.5	16.5
	586	9	13.5	13.5	16.5	16.5	16.5

Exemple I.4.2. Construction de 8 PMNS avec un polynôme de réduction cyclotomique pour $p = 40993$ et $n = 4$ (Table I.2)

Exemple I.4.3. Construction de PMNS à partir d'un polynôme de réduction cyclotomique pour $p = 2^{256} \cdot 3^{157} \cdot 115 + 1$ codé sur 512 bits.

— $E(X) = X^8 + 1$, à partir des huit racines, le meilleur ρ est obtenu avec le Corollaire I.2.1 et le Corollaire I.2.2, et a une longueur de 66 bits.

Table I.2 – Bornes sur les chiffres des PMNS obtenus à partir des polynômes de `ClassCyclo(4)` pour $p = 40993$, pour chaque approche de base réduite.

$E(X)$	γ	$\rho_{\min}$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^4+1	39487	10	12.5	12.5	12.5	12.5	12.5
	38516	10	12.5	12.5	12.5	12.5	12.5
	2477	10	12.5	12.5	12.5	12.5	12.5
	1506	10	12.5	12.5	12.5	12.5	12.5
X^4-X^2+1	33712	10	13	13	14.5	14.5	14.5
	20702	10	17	17	14.5	14.5	14.5
	20291	10	17	17	14.5	14.5	14.5
	7281	10	13	13	14.5	14.5	14.5

— $E(X) = X^6 + X^3 + 1$, à partir des six racines, le meilleur ρ est obtenu deux fois avec LLL, sinon avec le Corollaire I.2.1 et le Corollaire I.2.2, et a une longueur de 87 bits.

— $E(X) = X^6 - X^3 + 1$, à partir des six racines, le meilleur ρ est obtenu avec le Corollaire I.2.1 et le Corollaire I.2.2, et a une longueur de 87 bits.

I.4.2 Nombre de PMNS avec un binôme de réduction donné

Proposition I.4.2. *Soit $E(X) = X^n + c$ un élément de* `BinomialClass(n, c)` *(Proposition I.3.6). Soit g un générateur de $(\mathbb{Z}/p\mathbb{Z})^*$ et y tel que $g^y \equiv -c \bmod p$.*

Si $\gcd(n, p-1)$ *divise y, alors, $E(X) = X^n + c$ a* $\gcd(n, p-1)$ *racines distinctes dans $\mathbb{Z}/p\mathbb{Z}$.*

Remarque I.4.1. Si $\gcd(n, p-1) = 1$ alors $E(X) = X^n + c$ est garanti d'avoir une racine modulo p.

Preuve. Posons $\delta = \gcd(n, p-1)$. Supposons que $\delta \mid y$, i.e. il existe m tel que $y = \delta m$. Nous commençons par vérifier qu'il existe une racine de $E(X)$ modulo p. Par l'algorithme d'Euclide étendu, on retrouve les coefficients de Bezout u et v tels que :

$$un + v(p-1) = \delta.$$

En multipliant cette équation par m, on obtient

$$\begin{aligned} unm + v(p-1)m &= \delta m \\ &= y. \end{aligned}$$

On peut donc réécrire

$$\begin{aligned} g^y &\equiv -c \bmod p \\ g^{unm+v(p-1)m} &\equiv -c \bmod p \\ (g^{um})^n (g^{(p-1)})^{vm} &\equiv -c \bmod p. \end{aligned}$$

Or, d'aprés le petit théorème de Fermat, si p est premier, $g^{(p-1)} \equiv 1 \bmod p$. On en déduit

$$(g^{um})^n \equiv -c \bmod p.$$

Donc g^{um} est une racine de $X^n + c \bmod p$. On pose $x_0 = um$.

Nous cherchons à présent les autres racines. Nous considérons pour cela l'équation $x^n \equiv 1 \bmod p$. Dans un groupe cyclique d'ordre $p-1$, le nombre de solutions de cette équation est égal à δ. On sait que g^k est solution de $x \equiv 1 \bmod p$ si et seulement si $(p-1) \mid k$, donc g^k est solution de $x^n \equiv 1 \bmod p$ si et seulement si $(p-1) \mid kn$. Comme $\delta \mid n$, c'est le cas si et seulement si $(p-1)/\delta \mid k$. Toutes les solutions distinctes s'écrivent alors

$$g^{j\,(p-1)/\delta} \bmod p \quad \text{, pour } j \in \{0, \ldots, \delta - 1\}.$$

Pour $j \in \{0, \ldots, \delta - 1\}$, on a donc

$$\begin{aligned} (g^{x_0 + j\,(p-1)/\delta})^n &\equiv (g^{x_0})^n (g^{j\,(p-1)/\delta})^n \bmod p \\ &\equiv -c \bmod p. \end{aligned}$$

Ainsi, l'équation linéaire $x^n \equiv -c \bmod p$ admet δ solutions distinctes si et seulement si δ divise y, chaque solution étant égale à $g^{x_0 + jp'} \bmod p$ où $j \in \{0, \ldots, \delta - 1\}$ et $p' = (p-1)/\delta$.

□

Exemple I.4.4. Pour $p = 22777$, 7 est un générateur de $(\mathbb{Z}/22777)^*$. Soit $n = 4$ et $E(X) = X^4 + c$. Pour $c = 2$, on peut trouver $y = 8820$, tel que $-c = 7^y \bmod p$. Puisque $\text{pgcd}(1, n) = 1$, d'après la Prop I.3.6 $E(X)$ est irréductible dans $\mathbb{Z}[X]$. De plus, $\text{pgcd}(n, p-1) = 4$ divise y, donc quatre PMNS peuvent être générés à partir de $E(X)$. Pour $c' = -2$, on trouve $y' = 20208$ et $\gcd(n, p-1) = 4$ divise y' donnant encore une fois quatre PMNS possibles.

Exemple I.4.5. Pour $p = 40993$, 5 est un générateur de $(\mathbb{Z}/40993)^*$. Soit $n = 4$ et $E(X) = X^4 + c$. Pour $c = 2$, on peut trouver $y = 33788$, tel que $-c = 5^y \bmod p$. Puisque $\text{pgcd}(1, n) = 1$, d'après la Prop I.3.6 $E(X)$ est irréductible dans $\mathbb{Z}[X]$. De plus,

Table I.3 – Bornes sur les chiffres des PMNS obtenus à partir des polynômes de `BinomialClass`$(4, \pm 2)$ pour $p = 22777$, pour chaque approche de base réduite.

$E(X)$	γ	$\rho_{\min}$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^4+2	5620	9	14.5	14.5	14.5	14.5	14.5
	6867	9	11.5	11.5	11.5	11.5	11.5
	15910	9	11.5	11.5	11.5	11.5	11.5
	17157	9	14.5	14.5	14.5	14.5	14.5
X^4-2	5918	8	18.5	18.5	13.5	13.5	13.5
	6915	8	15.5	15.5	16	16	16
	15862	8	15.5	15.5	16	16	16
	16859	8	18.5	18.5	13.5	13.5	13.5

$\mathrm{pgcd}(n, p-1) = 4$ divise y, donc quatre PMNS peuvent être générés à partir de $E(X)$. Pour $c' = -2$, on trouve $y' = 13292$ et $\gcd(n, p-1) = 4$ divise y' donnant encore une fois quatre PMNS possibles.

Table I.4 – Bornes sur les chiffres des PMNS obtenus à partir des polynômes de `BinomialClass`$(4, \pm 2)$ pour $p = 40993$, pour chaque approche de base réduite.

$E(X)$	γ	$\rho_{\min}$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^4+2	12589	10	15.5	15.5	16	16	16
	16177	10	18.5	18.5	19.5	19.5	19.5
	24816	10	18.5	18.5	19.5	19.5	19.5
	28404	10	15.5	15.5	16	16	16
X^4-2	12720	8	12.5	12.5	14	14	14
	20268	11	16.5	16.5	18	18	18
	20725	11	16.5	16.5	18	18	18
	28273	8	12.5	12.5	14	14	14

I.4.3 Nombre de PMNS dans le cas général

Dans cette partie, nous proposons une méthode générale pour compter le nombre minimum de PMNS que l'on peut atteindre à partir d'un premier p et tout polynôme irréductible dans $\mathbb{Z}[X]$.

Nous utilisons le fait que le calcul de $\mathrm{pgcd}(X^p - X, E(X)) \bmod p$ peut être fait en un temps raisonnable, en deux étapes :

1. nous calculons $X^p \bmod E(X) \bmod p$ avec un algorithme d'exponentiation rapide, et nous calculons $F(X) = X^p - X \bmod E(X) \bmod p$,

2. puis, nous évaluons $\text{pgcd}(F(X), E(X)) \bmod p$ avec des polynômes de degrés inférieurs à n.

La première étape représente au plus $\log(p)$ mises au carré et multiplications, et la seconde étape représente au plus n itérations de l'algorithme d'Euclide. Les racines sont trouvées en factorisant le polynôme $\text{gcd}(F(X), E(X)) \bmod p$. Quelques exemples d'algorithmes de factorisation peuvent être trouvés dans [NQ98].

Proposition I.4.3. *Soit p premier, $n > 2$, $E(X)$ un polynôme de degré n et irréductible dans $\mathbb{Z}[X]$, et $D(X) = \text{pgcd}(X^p - X, E(X)) \bmod p$.*

Il existe $\deg(D(X))$ *systèmes PMNS* $(p, n, \gamma_i, \rho)_{E(X)}$.

Preuve. La preuve est triviale si l'on considère, lorsque p est premier, que les racines de $X^p - X \bmod p$ sont les p éléments de $\mathbb{Z}/p\mathbb{Z}$. □

Remarque I.4.2. La Proposition I.4.1 peut être considérée comme un corollaire de la Proposition I.4.3.

Exemple I.4.6. Nous considérons $p = 78264746924694600393874000999992 97$ et $E(X) = X^5 + X^2 + 1$.

$$\begin{aligned}
\text{Alors, } X^p \bmod E(X) = \; & 732212625942009817709398509909462 4\, X^4 \\
& +172782621530124334904222246113526 2\, X^3 \\
& +343884189760812697100452350686441 0\, X^2 \\
& +737295850362666465909672848502029 5\, X \\
& +416728560616853002518029351668087 6
\end{aligned}$$

$$\begin{aligned}
\text{Ainsi, } & \text{pgcd}(X^p \bmod E(X) - X, E(X)) \bmod p \\
& = X^2 + 13058499984190672910003378977052 58\, X \\
& +17930730009542045460341940680988 26 \\
& = (X + 61576990395578092706710688950709 12) \\
& (X + 29746256513307180597166691026336 43)
\end{aligned}$$

D'où, nous obtenons deux racines de $E(X) \bmod p$:

$$\gamma_1 = 16687756529116507687163312049283 85$$
$$\gamma_2 = 48518490411387419796707309973656 54$$

I.4.4 Exemples comptant tous les PMNS possibles pour un p donné

Exemple I.4.7. Cet exemple est obtenu avec des routines en SageMath.

Pour $p = 57896044618658097711785492504343953926634992332820282019728792003956566811073$ un nombre premier de 256 bits, et $n = 9$.

Nous considérons les PMNS $\mathfrak{B} = (p, n, \gamma, \rho)_E$ tels que :

— $E(X) = X^n + a_k X^k + \cdots + a_1 X + a_0 \in \mathbb{Z}[X]$, avec $n \geq 2$ et $k \leq \frac{n}{2}$,

— des coefficients $|a_i| \leq 1$ pour $1 \leq i \leq k$ et $|a_0| \leq 3$,
— $\rho \leq 2^{31}$.

Pour obtenir le nombre de ces PMNS, nous calculons, pour chaque polynôme vérifiant ces critères, le nombre de ses racines dans $\mathbb{Z}/p\mathbb{Z}$, puis nous effectuons la somme totale du nombre de racines pour tous les polynômes. Nous obtenons ainsi un total de 354 PMNS.

La plupart du temps, le meilleur ρ (borne sur les chiffres) est obtenu d'abord par LLL (266 fois) ou BKZ (46 fois), certains sont dus au Corollaire I.2.1 (10) ou au Corollaire I.2.2 (28), ou à la Proposition I.2.3 (4) avec un vecteur court.

Exemple I.4.8. Cet exemple utilise nos classes de polynômes irréductibles présentées à la Section I.3 pour calculer le nombre de PMNS que nous pouvons construire à partir d'un modulo premier p et d'un nombre de chiffres n grâce au Théorème I.4.3. Dans cet exemple, les polynômes apparaissant dans plusieurs classes ne sont compté qu'une seule fois, et $\texttt{ClassPerron}(\texttt{n}, |\texttt{a}_1|)$ contient les polynômes des deux classes $\texttt{ClassPerron}(\texttt{n}, |\texttt{a}_1|)$ et $\texttt{ClassPerron}(\texttt{n}, -|\texttt{a}_1|)$.

On choisit un premier p sur 256 bits et $n = 8$, i.e. $\rho \sim 2^{32}$, avec
$p = 57896044618658097711785492504343953926634992332820282019728792003956566811073$.

Ensuite, nous comptons le nombre de PMNS pour chacune de nos classes de polynômes avec $\|E\|_\infty \leq 7$.

$\texttt{ClassCyclo}(\texttt{n})$: 8 systems	$\texttt{TrinomialClass}(\texttt{n})$: 24 systems
$\texttt{QuadrinomialClass}(\texttt{n})$: no system	$\texttt{BinomialClass}(\texttt{n}, 3)$: no system
$\texttt{BinomialClass}(\texttt{n}, 4)$: no system	$\texttt{BinomialClass}(\texttt{n}, 5)$: no system
$\texttt{BinomialClass}(\texttt{n}, 6)$: 16 systems	$\texttt{BinomialClass}(\texttt{n}, 7)$: no system
$\texttt{ClassPrimeCst}(\texttt{n}, 3)$: 6 systems	$\texttt{ClassPrimeCst}(\texttt{n}, 5)$: 158 systems
$\texttt{ClassPrimeCst}(\texttt{n}, 7)$: 190 systems	$\texttt{ClassPerron}(\texttt{n}, 3)$: 8 systems
$\texttt{ClassPerron}(\texttt{n}, 4)$: 38 systems	$\texttt{ClassPerron}(\texttt{n}, 5)$: 78 systems
$\texttt{ClassPerron}(\texttt{n}, 6)$: 104 systems	$\texttt{ClassPerron}(\texttt{n}, 7)$: 112 systems

Pour ce premier p sur 256 bits et un nombre de chiffres $n = 8$, on trouve 742 systèmes.

Bien que nous nous restreignons ici à des classes particulières de polynômes irréductibles, nous autorisons $\|E\|_\infty \leq 7$, alors que nous imposions $|a_i| \leq 1$ pour $1 \leq i \leq k$ et $|a_0| \leq 3$ dans l'exemple précédent. Le fait de considérer un plus grand nombre de polynômes nous permet d'obtenir plus de systèmes dans cet exemple.

Exemple I.4.9.

Nombre de systèmes PMNS obtenus pour des premiers de 15 bits

Les graphes suivants présentent, pour 100 premiers p sur 15 bits tirés de façon uniforme, le nombre de systèmes PMNS obtenus avec $n = 4$ chiffres, satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})\|_1 < \alpha p^{1/n}$, où $\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})$ est au choix $LLL(A)$, $BKZ(A)$, B, ou D.

Les polynômes de réduction considérés dans cet exemple sont tous les polynômes de degré 4, sous la forme $E(X) = X^4 + a_k X^k + \cdots + a_1 X + a_0 \in \mathbb{Z}[X]$, avec $k \leq 2$, à coefficients dans $\{-1, 0, 1\}$ (au total 27 polynômes).

Chaque système est compté pour une seule des bases possibles pour $\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})$, satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})\|_1 < \alpha p^{1/n}$. Le choix de la base découle des méthodes de construction des bases ; une base dont la construction est plus coûteuse que les autres sera celle retenue seulement si elle fournit une borne inférieure à toutes les autres. Dans cette optique, nous choisissons pour $\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})$, $\arg\min_{X \in \{D,B,BKZ(A),LLL(A)\}}(\|X\|_1)$, si $\|X\|_1 < \alpha p^{1/n}$, i.e.

— D si $\|D\|_1 < \min\{\alpha p^{1/n}, \|LLL(A)\|_1, \|BKZ(A)\|_1, \|B\|_1\}$,

— sinon B si $\|B\|_1 < \min\{\alpha p^{1/n}, \|LLL(A)\|_1, \|BKZ(A)\|_1\}$,

— sinon $BKZ(A)$ si $\|BKZ(A)\|_1 < \min\{\alpha p^{1/n}, \|LLL(A)\|_1\}$,

— sinon $LLL(A)$ si $\|LLL(A)\|_1 < \alpha p^{1/n}$.

Cas 1 : $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 2p^{1/n}$

Pour **E(X) irréductible** : (10 polynômes)

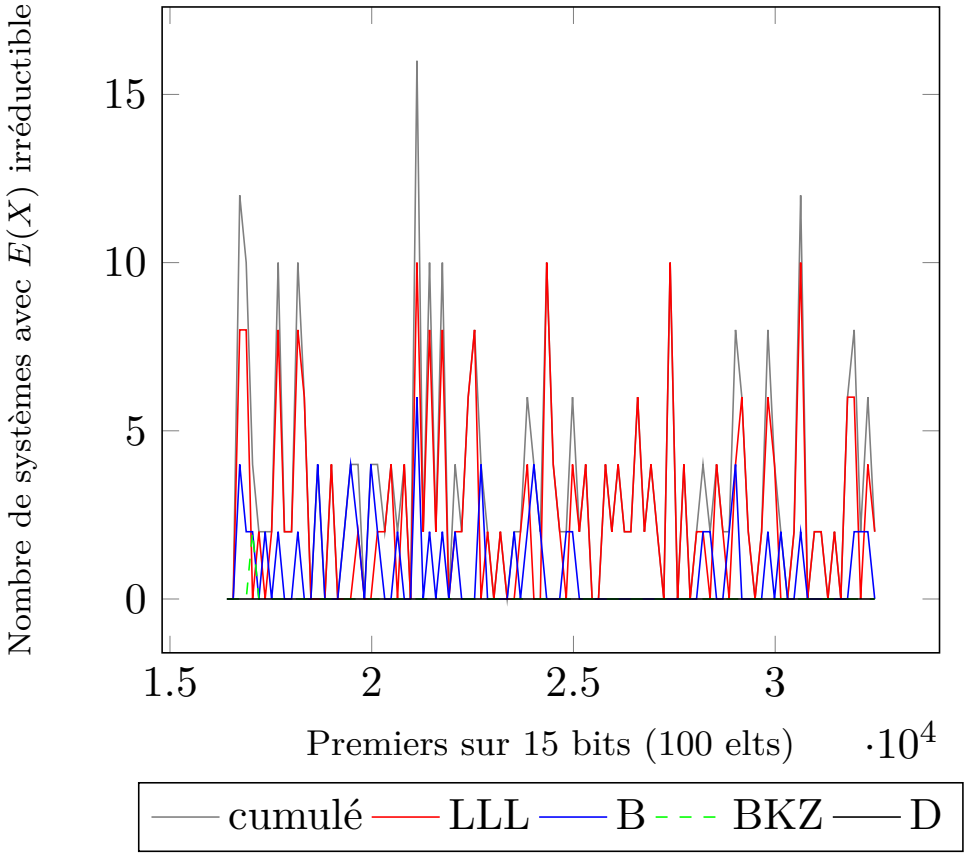


Fig. I.1 *Nombre de systèmes PMNS avec $n = 4$ chiffres, pour 100 premiers p de 15 bits tirés de façon uniforme, tels que $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 2p^{1/n}$, où $\mathbf{base}(\mathfrak{L}_{\mathcal{B}})$ $= LLL(A)$, $BKZ(A)$, B, ou D, pour des polynômes de réduction irréductibles*

▶ Moyennes du nombre de systèmes satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 2p^{1/n}$ pour ces 100 premiers, pour

le cumul pour toutes les bases : 3.48
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = LLL(A)$: 4.06
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = BKZ(A)$: 2.0
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = HKZ(A)$: 2.0
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = B$: 2.53
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = D$: 0

▶ Premiers sans système satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})\|_1 < 2p^{1/n}$ (20 elts) : [16411, 16567, 18493, 18839, 19157, 19819, 20963, 21911, 23027, 23357, 25463, 25621, 27239, 27581, 27893, 29501, 30307, 30803, 31307, 31627]

▶ Premiers avec 10 systèmes ou plus satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})\|_1 < 2p^{1/n}$ (9 elts) : [16729, 16889, 17681, 21121, 21433, 21751, 24337, 27397, 30631]

▶ Nombre de systèmes par classe satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})\|_1 < 2p^{1/n}$ pour les premiers "prolifiques" : (i.e. avec 10 systèmes ou plus satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\boldsymbol{\mathcal{B}}})\|_1 < 2p^{1/n}$)

exemple premier : classe 1 (nb de systèmes), classe 2 (nb de systèmes)...

16729 : cyclo (4), trinomial (8)

16889 : cyclo (4), trinomial (6)

17681 : cyclo (4), trinomial (6)

21121 : cyclo (4), trinomial (12)

21433 : cyclo (4), trinomial (2), quadrinomial (4)

21751 : trinomial (6), quadrinomial (4)

24337 : cyclo (4), trinomial (6)

27397 : trinomial (6), quadrinomial (4)

30631 : trinomial (10), quadrinomial (2)

Cas 2 : $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$

Pour **E(X) irréductible** : (10 polynômes)

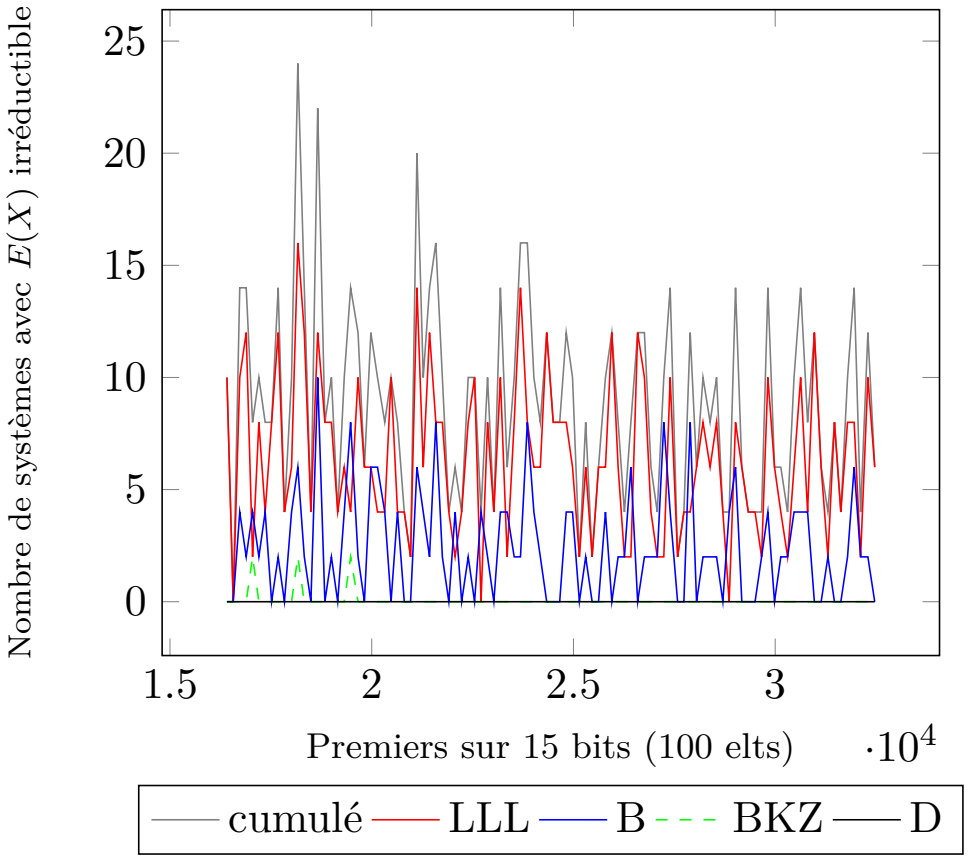


Fig. I.2 *Nombre de systèmes PMNS avec $n = 4$ chiffres, pour 100 premiers p de 15 bits tirés de façon uniforme, tels que $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$, où $\mathbf{base}(\mathfrak{L}_{\mathcal{B}})$ = $LLL(A)$, $BKZ(A)$, B, ou D, pour des polynômes de réduction irréductibles*

Dans ce second cas, des systèmes viennent s'ajouter à ceux comptés dans le **Cas 1** à la Figure I.1. Il s'agit des systèmes pour lesquels il existe une base satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$, mais aucune base telle que $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 2p^{1/n}$.

► Moyennes du nombre de systèmes satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ pour ces 100 premiers, pour

le cumul pour toutes les bases : 8.86
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = LLL(A)$: 6.62

$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = BKZ(A) : 2.0$
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = HKZ(A) : 2.0$
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = B : 3.72$
$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = D : 0$

▶ Premiers sans système satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ (1 elt) : [16567]

▶ Premiers avec 15 systèmes ou plus satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ (6 elts) : [18169, 18661, 21121, 21589, 23687, 23857]

▶ Nombre de systèmes par classe satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ pour les premiers "prolifiques" : (i.e. avec 15 systèmes ou plus satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$)

exemple premier : classe 1 (nb de systèmes), classe 2 (nb de systèmes)...

18169 : cyclo (4), trinomial (16), quadrinomial (4)

18661 : trinomial (20), quadrinomial (2)

21121 : cyclo (4), trinomial (16)

21589 : trinomial (14), quadrinomial (2)

23687 : trinomial (12), quadrinomial (4)

23857 : cyclo (4) trinomial (10), quadrinomial (2)

Dans le cas des polynômes réductibles, le nombre de systèmes satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ est identique à celui obtenu pour $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 2p^{1/n}$.

Nombre de systèmes PMNS obtenus pour des premiers de 250 bits

Les graphes suivants présentent, pour 100 premiers p sur 250 bits tirés de façon uniforme, le nombre de systèmes PMNS obtenus avec $n = 8$ chiffres, satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$, où $\mathbf{base}(\mathfrak{L}_{\mathcal{B}})$ est au choix $LLL(A)$, $BKZ(A)$, B, ou D.

Les polynômes de réduction considérés dans cet exemple sont tous les polynômes de degré 8, sous la forme $E(X) = X^8 + a_k X^k + \cdots + a_1 X + a_0 \in \mathbb{Z}[X]$, avec $k \leq 4$, à coefficients dans $\{-1, 0, 1\}$ (au total 243 polynômes).

La méthode utilisée pour compter chaque système est la même que celle décrite dans l'exemple sur les premiers de 15 bits.

$$\underline{\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}}$$

Pour **E(X) irréductible** : (102 polynômes)

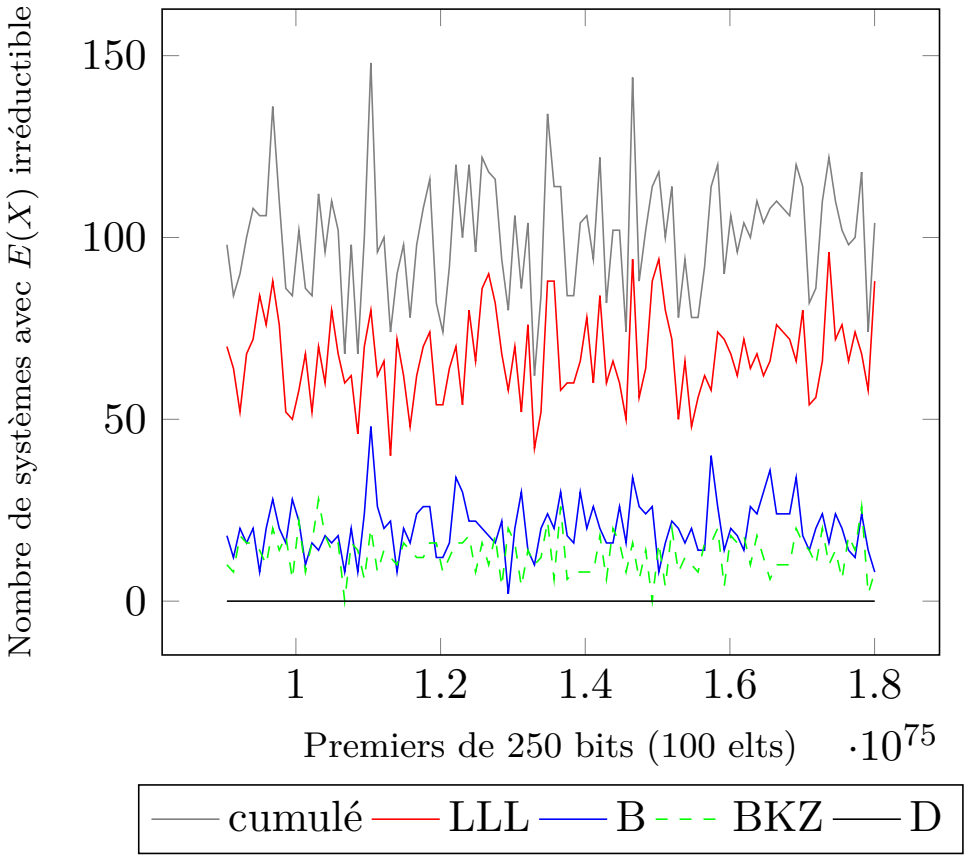


Fig. I.3 *Nombre de systèmes PMNS avec $n = 8$ chiffres, pour 100 premiers p de 250 bits tirés de façon uniforme, tels que $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$, où $\mathbf{base}(\mathfrak{L}_{\mathcal{B}})$ = $LLL(A)$, $BKZ(A)$, B, ou D, pour des polynômes de réduction irréductibles*

▶ Moyennes du nombre de systèmes satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ pour ces 100 premiers, pour

le cumul pour toutes les bases : 100.28

$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = LLL(A)$: 67.16

$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = BKZ(A)$: 13.22

$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = HKZ(A)$: 13.22

$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = B$: 20.16

$\mathbf{base}(\mathfrak{L}_{\mathcal{B}}) = D$: 0

▶ Premiers sans système satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ (aucun) : [ ]

▶ Premiers avec 130 systèmes ou plus satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ (4 elts) : [
967949495968190071118913702807000479710928778064339090017340741316148819089,
110364335054316998763091095086405662172647954134438662600108009757542202049,
134789228877813383735250599736675767735447091524847219077181093884211378261 7,
146549362940978309832957027901620633376794824342451338862438504760015055638 7]

▶ Nombre de systèmes par classe satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$ pour les premiers "prolifiques" :
(i.e. avec 130 systèmes ou plus satisfaisant $\|\mathbf{base}(\mathfrak{L}_{\mathcal{B}})\|_1 < 4p^{1/n}$)

exemple premier : classe 1 (nb de systèmes), classe 2 (nb de systèmes)...

967949495968190071118913702807000479710928778064339090017340741316148819089 :
cyclo (8), trinomial (28), quadrinomial (14)

110364335054316998763091095086405662172647954134438662600108009757542202049 :
cyclo (8), trinomial (38), quadrinomial (28)

134789228877813383735250599736675767735447091524847219077181093884211378261 7 :
trinomial (22), quadrinomial (14)

146549362940978309832957027901620633376794824342451338862438504760015055638 7 :
trinomial (16), quadrinomial (18)

I.5 Conclusion

Dans ce travail, nous avons montré avec le Théorème I.2.1 le lien entre l'existence d'un PMNS et le réseau euclidien généré par son polynôme de réduction et son modulo. Nous avons donc une borne sur la taille des chiffres correspondant au rayon de recouvrement. Ensuite, le Théorème I.2.2 nous donne une borne qui peut être calculée simplement à partir de la norme 1 du réseau. Ce second théorème nous a conduit à considérer les systèmes PMNS définis par un polynôme irréductible. Dans ce cas, il est facile de définir une base du réseau qui peut être associée au PMNS (Proposition I.2.3, et Corollaires I.2.1 et I.2.2). Ces théorèmes, propositions et corollaires nous ont permis de produire des PMNS avec des polynômes de réduction spécifiques permettant des réductions efficaces et dont les racines fournissent les bases de ces systèmes. Maintenant, nous avons la possibilité d'offrir pour un modulo premier donné une grande variété de PMNS.

Chapitre II

Randomisation de l'arithmétique sur les systèmes PMNS

Sommaire

Dans ce chapitre, nous utilisons la redondance du système PMNS pour protéger les algorithmes de calcul modulaire contre les attaques par canaux cachés (SCA). Plus précisément, nous nous concentrons sur la protection de la multiplication scalaire sur les courbes elliptiques (ECSM) contre les attaques SCA [Koc96], en utilisant le PMNS pour représenter les coordonnées de tout point de la courbe. Toutes les opérations effectuées sur les représentations au sein du PMNS utilisent des algorithmes réguliers, de sorte qu'elles sont intrinsèquement immunisées contre les attaques par analyse simple de la consommation, ou Simple Power Analysis (SPA). Ainsi, utiliser un algorithme régulier, comme le Montgomery powering ladder [Mon87], suffit pour protéger une ECSM des attaques SPA.

Pour protéger la multiplication scalaire classique $k\mathcal{P}$ contre les attaques par analyse différentielle de la consommation, ou Differential Power Analysis (DPA), nous décrivons d'abord comment le processus de conversion, qui associe un entier à une représentation dans le PMNS, peut facilement être modifié pour randomiser le point de base $\mathcal{P}$. Ensuite, nous montrons comment randomiser toutes les valeurs intermédiaires impliquées dans la multiplication scalaire en introduisant un aléa dans la primitive de multiplication modulaire du PMNS. La randomisation du scalaire k peut être effectuée en utilisant des contre-mesures classiques [FV12]. Nous montrons également pourquoi ces randomisations sont suffisantes pour protéger les calculs des attaques DPA.

À la Section II.2 nous présentons à la fois la randomisation des entrées (Sec. II.2.2) et de la multiplication (Sec. II.2.3). Dans la Section II.2.4, nous donnons les coûts des multiplications modulaires au sein du PMNS et décrivons quelques avantages spécifiques du PMNS par rapport à certaines attaques comme celle de Goubin [Gou03]. Nous concluons à la Section III.7. Dans la suite, les représentations sont parfois considérées comme des polynômes, parfois comme des vecteurs. Dans un souci de lisibilité, nous utilisons des lettres majuscules pour noter polynômes et vecteurs, les deux visions n'étant pas utilisées simultanément.

II.1 Algorithmes pour l'arithmétique sur le système PMNS

Dans un système PMNS, la multiplication $T = VW \bmod E$, avec $V \equiv x_{\boldsymbol{\mathcal{B}}}$ et $W \equiv y_{\boldsymbol{\mathcal{B}}}$, satisfait $T(\gamma) \equiv xy \bmod p$ car $E(\gamma) \equiv 0 \bmod p$. Cependant, même si $\deg(T) < n$, T peut ne pas être une représentation de $xy \bmod p$ dans $\boldsymbol{\mathcal{B}}$, car ses coefficients peuvent être supérieurs ou égaux à ρ. Pour obtenir cette représentation dans $\boldsymbol{\mathcal{B}}$, une primitive spéciale appelée la *réduction interne* doit être appliquée. L'opération qui réduit la taille des coefficients polynomiaux est décrite à la Section II.1.1.

Dans [Pla05, BIP05a], les auteurs montrent comment utiliser le PMNS pour accélérer le calcul modulaire. Les principales primitives du PMNS sont rappelées en Sec-

tion II.1.2, II.1.3 et II.1.4.

II.1.1 Réduction interne

Le but de la réduction interne est de maintenir les coefficients des polynômes suffisamment petits dans le système PMNS durant les calculs. Soit $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$ un système PMNS. Le processus de réduction interne associe à un polynôme $V(X) \in \mathbb{Z}[X]$ un polynôme $\tilde{V}(X) \in \mathbb{Z}[X]$ tel que $\tilde{V}(\gamma) \equiv V(\gamma) \bmod p$, $\|\tilde{V}\|_\infty \leqslant \|V\|_\infty$ et $\deg \tilde{V} = \deg V$. Nous décrivons dans la suite deux façons d'effectuer cette opération.

a) La réduction interne via une méthode de type Montgomery

Dans [NP08], les auteurs donnent une procédure de réduction similaire à l'algorithme de Mongtomery (Alg. 1). À partir de $V(X) \in \mathbb{Z}[X]$, elle calcule un polynôme R' avec $R'(\gamma) \equiv V(\gamma) \bmod p$ tel que la partie basse des coefficients de R' soit nulle. R' peut ainsi être divisé par une puissance de la base r pour obtenir un polynôme R aux coefficients plus petits. L'algorithme renvoie le polynôme R avec $R(\gamma) \equiv V(\gamma)r^{-1} \bmod p$.

Utiliser cette version de la réduction interne nécessite de convertir les éléments de $\mathbb{Z}/p\mathbb{Z}$ dans le domaine de Montgomery au cours de la procédure de conversion vers le PMNS. Un élément $a \in \mathbb{Z}/p\mathbb{Z}$ est représenté par un polynôme $A \in \boldsymbol{\mathcal{B}}$ tel que $A(\gamma) \equiv a.r \bmod p$. De cette façon, nous assurons la cohérence des opérations dans le PMNS tout en utilisant cette méthode pour la réduction des coefficients.

L'algorithme dépend d'un *polynôme de réduction interne* M, d'un entier r et du polynôme $M' = -M^{-1} \bmod (E, r)$. Le polynôme M' existe si et seulement si pgcd(resultant(E, M), r) $= 1$. Le choix du polynôme M pour assurer l'existence de M' est discuté dans [DDV18]. Pour des raisons d'efficacité, r est souvent une puissance de 2 afin que la division exacte par r soit rapide. La Proposition II.1.1, de [NP08], donne une borne sur les coefficients du polynôme retourné par l'Algorithme 1. Ici, $E(X) = X^n - \lambda$, avec $\lambda \in \mathbb{Z}\backslash\{0\}$.

Algorithme 1 RedCoeff - Méthode de type Montgomery[NP08]

Require: $V \in \mathbb{Z}[X]$ tel que $\deg(V) < n$, $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$, $M \in \boldsymbol{\mathcal{B}}$ tel que $M(\gamma) \equiv 0 \bmod p$, un entier r et $M' = -M^{-1} \bmod (E, r)$
Ensure: $R(\gamma) \equiv V(\gamma)r^{-1} \bmod p$
1: $Q \leftarrow V \times M' \bmod (E, r)$
2: $R' \leftarrow V + (Q \times M) \bmod E$
3: $R \leftarrow R'/r$
4: **return** R

Proposition II.1.1. *Soit $m = \|M\|_\infty$. Si V, ρ et m sont tels que : $\|V\|_\infty \leqslant n|\lambda|\rho^2$, $\rho \geqslant 2|\lambda|nm$ et $r \geqslant 2|\lambda|n\rho$ alors l'Algorithme 1 retourne R tel que $\|R\|_\infty < \rho$ (i.e. $R \in \boldsymbol{\mathcal{B}}$).*

Nous généraliserons cette borne dans la Section II.1.2 pour $E(X) = X^n - \Delta(X)$.

b) La réduction interne via une méthode de type Babaï

L'algorithme de type Babaï (Algorithme 2), contrairement à l'algorithme de réduction des coefficients de type Montgomery, ne nécessite pas de maintenir les éléments de $\mathbb{Z}/p\mathbb{Z}$ dans un autre domaine de représentation, et ne cherche pas à calculer le résultat en utilisant un polynôme M avec des propriétés particulières.

Dans cette approche, nous considérons le réseau euclidien $\mathfrak{L}_{\boldsymbol{\mathcal{B}}}$ associé au PMNS $\boldsymbol{\mathcal{B}}$. Ici, $\mathfrak{L}_{\boldsymbol{\mathcal{B}}}$ est l'ensemble des polynômes ayant γ pour racine modulo p et dont le degré est au plus $n-1$:

$$\mathfrak{L}_{\boldsymbol{\mathcal{B}}} = \{A \in \mathbb{Z}[X], \text{ tel que : } \deg(A) < n \text{ et } A(\gamma) \equiv 0 \bmod p\}.$$

Le réseau $\mathfrak{L}_{\boldsymbol{\mathcal{B}}}$ est défini à partir de l'une de ses bases, dont les éléments sont $A_1(X) = p$ et $A_{i+1}(X) = X^i - \gamma^i$, pour $1 \leq i < n$, depuis laquelle nous calculons la base LLL réduite D de $\mathfrak{L}_{\boldsymbol{\mathcal{B}}}$, et la base de Gram-Schmidt $\widetilde{D}$ obtenue à partir de D, dont les éléments sont orthogonaux mais pas nécessairement dans $\mathfrak{L}_{\boldsymbol{\mathcal{B}}}$ [PSZ15]. Dans la suite, nous désignons par $\|\cdot\|$ la norme euclidienne du vecteur correspondant.

Deux polynômes représentant le même élément une fois évalués en γ modulo p sont congruents modulo $\mathcal{L}_{\boldsymbol{\mathcal{B}}}$, c'est-à-dire que leur différence est dans $\mathcal{L}_{\boldsymbol{\mathcal{B}}}$. Pour cette méthode, les représentations du système PMNS sont considérées comme des vecteurs, la i-ième coordonnée correspondant au i-ième coefficient de la forme polynomiale de la représentation.

À partir d'un vecteur $V \in \mathbb{Z}^n$, l'algorithme de type Babai retourne un vecteur U congruent à V modulo $\mathcal{L}_{\boldsymbol{\mathcal{B}}}$, i.e. un vecteur dans $V - \mathfrak{L}_{\boldsymbol{\mathcal{B}}} := \{V - W \mid W \in \mathfrak{L}_{\boldsymbol{\mathcal{B}}}\}$, avec des coefficients inférieurs, en réduisant chaque coordonnée de V une dimension à la fois.

Au cours de la i-ième étape, pour i allant de n à 1, l'algorithme identifie l'hyperplan $span(D_1, \ldots, D_{i-1}) + c\,D_i$, $c \in \mathbb{Z}$, qui est le plus proche de V en dimension i. Il calcule sa distance c depuis l'origine en prenant $\|\widetilde{D}_i\|$ pour unité, puisque $\widetilde{D}_i$ appartient à la base orthogonale $\widetilde{D}$ et sa norme représente la distance d'un hyperplan à l'hyperplan suivant ;

$$c \leftarrow \lfloor < V, \widetilde{D}_i > / \|\widetilde{D}_i\|^2 \rceil.$$

Puisque $\widetilde{D}_i$ et D_i appartiennent au même hyperplan, le vecteur V est mis à jour par une soustraction de la distance c dont l'unité est $\|\widetilde{D}_i\|$, multipliée par D_i, ainsi le résultat est un vecteur de $V - \mathfrak{L}_{\boldsymbol{\mathcal{B}}}$,

$$V \leftarrow V - c \times D_i.$$

Cette méthode garantit que la i-ième coordonnée du résultat est aussi petite que possible dans la base orthogonale $\widetilde{D}$. L'algorithme recherche ensuite les hyperplans les plus proches

de V de rang inférieur. À la fin, la sortie représente le même élément modulo p que l'entrée et appartient au rectangle

$$\left\{\sum a_i \widetilde{D}_i \mid |a_i| \leq \frac{1}{2}\right\}.$$

Algorithme 2 RedCoeff - Méthode de type Babaï

Require: $V \in \mathbb{Z}[X]$ avec $\deg(V) < n$, $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$
Ensure: $R \in \boldsymbol{\mathcal{B}}$ tel que $R(\gamma) = V(\gamma) \bmod p$
Data : $D = \{D_i, 1 \leq i \leq n\}$ la base LLL réduite du réseau associé à $\boldsymbol{\mathcal{B}}$,
$\widetilde{D} = \{\widetilde{D_i}, 1 \leq i \leq n\}$ la base de Gram-Schmidt obtenue à partir de D

```
R ← V
for i = 1 to n do
    c ← ⌊< R, D̃_{n−i+1} > /‖D̃_{n−i+1}‖²⌉
    R ← R − c × D_{n−i+1}
end for
return R
```

Nous décrivons à présent le système auquel appartient la sortie de l'algorithme. À partir de [Gal11] et [PSZ15], nous déduisons une nouvelle proposition :

Proposition II.1.2. *Si ρ est tel que $\rho \geqslant \frac{1}{2}\, 2^{\frac{3n-1}{2}}\, p^{1/n}$, alors l'Algorithme 2 renvoie R tel que $\|R\|_\infty < \rho$ (i.e. $R \in \boldsymbol{\mathcal{B}}$).*

Nous remarquons qu'ici, la borne inférieure ne dépend pas du polynôme E contrairement à la réduction interne de type Montgomery.

II.1.2 Multiplication

Les éléments du PMNS sont des polynômes de degré au plus $n-1$. Leur multiplication s'effectue en utilisant la multiplication polynomiale classique. Cependant, le résultat (de degré au plus $2\,n-2$) doit être réduit modulo le polynôme de réduction externe E. Ensuite, une réduction interne est effectuée pour obtenir un résultat dans $\boldsymbol{\mathcal{B}}$.

Le polynôme E est tel que : $E(X) = X^n - \Delta(X) \in \mathbb{Z}[X]$. Pour réduire un polynôme V modulo E, on procède pas à pas en remplaçant le terme X^n dans V par $\Delta(X)$ jusqu'à satisfaire $\deg(V) < n$. Nous considérons ici les systèmes PMNS avec des polynômes de réduction E de degré $n \geq 2$, avec $\Delta(X) = \delta_k X^k + \cdots + \delta_1 X + \delta_0$ et $k \leq \frac{n}{2}$. Ces systèmes nous assurent une réduction en deux étapes au maximum.

En nous basant sur cette technique, nous calculons des polynômes de degré inférieur à n pour représenter les puissances X^i modulo E, pour $0 \leq i \leq 2\,n-2$, avec $X^i \bmod E = \sum_{l=i-n}^{k+i-n} \delta_{l-i+n} X^l$ quand $n \leq i < 2\,n-k$ et $X^i \bmod E = \sum_{l=i-n}^{n-1} \delta_{l+n-i} X^l + \sum_{l=0}^{2k-2n+i} (\sum_{j=0}^{l} \delta_j\, \delta_{l+2\,n-2\,j}) X^l$ pour $2\,n-k \leq i \leq 2\,n-2$.

Ainsi, tout polynôme V peut être réduit modulo E en multipliant le vecteur des coefficients de V par la matrice $\mathbf{S}$ de taille $(2\,n-1)\times n$ dont les lignes représentent les coefficients de chaque puissance X^i modulo E, pour $0 \leq i \leq 2\,n-2$. Nous posons $s = \|\mathbf{S}\|_1$, ($\|\mathbf{S}\|_1 = \max_j \left\{\sum_{i=0}^{n-1} |s_{i,j}|\right\}$, où $s_{i,j}$ est le coefficient de la i-ième ligne, j-ième colonne de $\mathbf{S}$) et déduisons la proposition suivante.

Proposition II.1.3. *Soit A, $B \in \mathbb{Z}[X]$ deux polynômes, tels que :* $\deg(A) < n$ *et* $\deg(B) < n$. *Nous avons :* $\|A \times B \bmod E\|_\infty < n\,s\,\|A\|_\infty\,\|B\|_\infty$.

À partir de cette proposition, la borne donnée à la Proposition II.1.1 peut être réécrite en remplaçant $|\lambda|$ par s.

II.1.3 Addition

La procédure d'addition est réalisée en utilisant l'addition polynomiale classique dans $\mathbb{Z}[X]$ suivie d'une réduction interne afin d'obtenir un résultat dans $\mathcal{B}$.

Remarque II.1.1. En pratique, l'addition peut être optimisée, car la réduction interne n'est pas toujours nécessaire. Prenons l'exemple de la réduction interne de type Montgomery. À partir de la Proposition II.1.1, si au plus $(n\,s\,\rho - 1)$ additions successives sont effectuées, il n'est pas nécessaire de réaliser une réduction interne après chaque addition polynomiale. Un seul appel à RedCoeff après la dernière addition polynomiale renverra le résultat dans $\mathcal{B}$.

II.1.4 Procédures de conversion

Les procédures de conversion transforment un entier en une représentation dans $\mathcal{B}$ et vice-versa [BIP05a]. Elles dépendent de l'étape de réduction interne qui peut être réalisée en utilisant l'une des deux procédures décrites précédemment.

a) De la représentation binaire vers le PMNS

La conversion de la représentation binaire vers la représentation PMNS requière de précalculer les représentations $P_i(X)$ des puissances de ρ dans $\mathcal{B}$, i. e. $P_i \equiv (\rho^i)_{\mathcal{B}}$, pour $i = 0, \ldots, n-1$. La conversion est alors réalisée en utilisant l'Algorithme 3.

Remarque II.1.2. Si la réduction interne de type Montgomery est utilisée à la ligne 4 de l'Algorithme 3, la ligne 1 doit être décommentée. La sortie sera alors $A \equiv (a\,r)_{\mathcal{B}}$.

Algorithme 3 Conversion vers le système PMNS

Require: $a \in \mathbb{Z}/p\mathbb{Z}$ et $\mathcal{B} = (p, n, \gamma, \rho)_E$

Ensure: $A \equiv a_{\mathcal{B}}$ # ou $A \equiv (a\,r)_{\mathcal{B}}$

Data : $P_i \equiv (\rho^i)_{\mathcal{B}}$, pour $i = 0, \ldots, n-1$ et $r = 2^j$, $j \geq 1$

1: # $a \leftarrow a \times r^2 \bmod p$

2: $b \leftarrow (a_{n-1}, ..., a_0)_\rho$ # décomposition de a en base ρ

3: $U \leftarrow \sum_{i=0}^{n-1} a_i\, P_i$

4: $A \leftarrow \mathbf{RedCoeff}(U)$

5: **return** A

b) Du PMNS vers la représentation binaire

La conversion du PMNS vers la représentation binaire est une simple évaluation polynomiale en γ dans l'arithmétique cible. Elle peut être effectuée en utilisant l'algorithme classique de Horner. Il faut prendre soin de sortir le résultat du domaine de Montgomery si nécessaire. La conversion est alors réalisée en utilisant l'Algorithme 4.

Remarque II.1.3. Si l'entrée de l'Algorithme 4 représente un élément se trouvant dans le domaine de Montgomery, la ligne 1 doit être décommentée. Le résultat sera alors sorti du domaine de Montgomery via un appel à la fonction RedCoeff de type Montgomery.

Algorithme 4 Conversion hors du système PMNS

Require: $A \in \mathcal{B} = (p, n, \gamma, \rho)_E$

Ensure: $a \equiv A(\gamma) \bmod p$

1: #$A \leftarrow \mathbf{RedCoeff}(A)$

2: $a \leftarrow 0$

3: **for** $i = n - 1$ **downto** 0 **do**

4: $\quad a \leftarrow (a\gamma + a_i) \bmod p$ # a_i est le coefficient d'indice i de A

5: **end for**

6: **return** a

II.2 Randomisation de la multiplication scalaire en ECC

Notre but est de présenter des algorithmes qui assureront la résistance de la multiplication scalaire aux attaques SCA existantes. La multiplication scalaire sur les courbes elliptiques utilise un point $\mathcal{P}$ sur une courbe elliptique publique, un entier secret k et calcule $k\mathcal{P}$. Cette opération peut être réalisée en utilisant la méthode classique du double

et de l'addition, mais elle n'est pas résistante aux attaques SCA [Koc96]. L'échelle de Montgomery [Mon87] et sa variante [BJ02, LD99, GJM$^+$11, JY02] sont plus résistantes mais restent attaquables [AVL19]. Une étude récente [AVL19] présente les attaques par canaux cachés et les contre-mesures existantes pour la multiplication scalaire sur les courbes elliptiques. Les contre-mesures existantes reposent soit sur la randomisation du scalaire k [Cor99, CJ01, TB02, CJ03, Che04], soit sur la randomisation du point $\mathcal{P}$. La randomisation du point $\mathcal{P}$ peut être calculée en utilisant la randomisation des coordonnées projectives [Cor99], ou bien en ajoutant un point aléatoire $\mathcal{R}$ [Cor99]. Une seule contre-mesure est conçue sur le corps multiplicatif, faisant intervenir une permutation aléatoire [CFG$^+$10].

Nous fournissons ici deux possibilités pour randomiser la multiplication scalaire :

— La randomisation de chaque coordonnée initiale de $\mathcal{P}$ chaque fois que ce point est utilisé dans l'algorithme de multiplication scalaire. Cette approche permet de contrer les attaques SCA, où l'attaquant tente de découvrir le secret en utilisant la connaissance du point $\mathcal{P}$ [Koc96] et assure la résistance à des attaques de points spécifiques [Gou03, AT03]. Nous utilisons soit l'Algorithme 6, soit l'Algorithme 7 comme procédure de conversion, dépendant de la méthode `RedCoeff`, qui sera utilisée ultérieurement pour la réduction interne ;

— La randomisation de chaque multiplication $A \times B$ pour A et B dans la représentation PMNS afin d'être plus résistant face aux attaques SCA, en utilisant soit l'Algorithme 8 soit l'Algorithme 9.

II.2.1 Génération polynomiale aléatoire

Avant de présenter nos procédures de randomisation, nous expliquons ici l'idée générale de la randomisation dans le système PMNS. Afin de randomiser les données au sein du système, nous générons un polynôme aléatoire $Z \in \mathbb{Z}[X]$ tel que : $\|Z\|_\infty \leq z$ et $\deg Z < n$ avec $z \in \mathbb{N}$. La valeur z est choisie lors du processus de génération du système PMNS. Nous verrons plus loin l'impact de sa valeur sur les paramètres du système. Cet entier z définit le nombre minimum de représentations distinctes dans $\boldsymbol{\mathcal{B}}$ pour chaque élément de $\mathbb{Z}/p\mathbb{Z}$. Une fois z fixé, il existe exactement $(2\,z+1)^n$ polynômes Z. Nous allons montrer comment utiliser Z pour garantir qu'il existe au moins $(2\,z+1)^n$ représentations distinctes pour chaque élément de $\mathbb{Z}/p\mathbb{Z}$.

Ci-après nous utilisons une fonction **randPoly**(z) pour générer des polynômes aléatoires à coefficients dans l'ensemble $\{-z, \ldots, z\}$. Nous considérons cette fonction comme sûre (voir [BSS16] par exemple). L'Algorithme 5 donne un exemple de la génération d'un tel Z. Nous supposons que la fonction **randInt** de cet algorithme est telle que **randInt**(z) génère un entier dans l'ensemble $\{-z, \ldots, z\}$ en utilisant la distribution uniforme.

Algorithme 5 randPoly - Génération polynomiale aléatoire

Require: $z \in \mathbb{N}$
Ensure: $Z \in \mathbb{Z}[X]$, $\|Z\|_\infty \leq z$ et $\deg Z < n$
1: $Z \leftarrow$ **randInt**(z)
2: **for** $i = 1$ **to** $(n-1)$ **do**
3: $\quad t \leftarrow$ **randInt**(z)
4: $\quad Z \leftarrow Z + t\,X^i$
5: **end for**
6: **return** Z

II.2.2 Randomisation des données en entrée

Nous montrons ici comment randomiser la procédure de conversion du binaire vers le PMNS afin d'obtenir des données en entrée randomisées. Soit $\boldsymbol{\mathcal{B}} = (p,\, n,\, \gamma,\, \rho)_E$ un système PMNS et un entier $a \in \mathbb{Z}/p\mathbb{Z}$. Nous souhaitons calculer une représentation randomisée de a dans $\boldsymbol{\mathcal{B}}$.

a) Randomisation de la conversion via la méthode de type Mongtomery

L'Algorithme 6 présenté ci-après est une légère modification de l'algorithme de conversion initial décrit dans [BIP05a]. La méthode de conversion randomisée introduit une multiplication polynomiale (ligne 5) avant la réduction interne. Ce faisant, nous nous assurons qu'avec la même entrée et des polynômes aléatoires différents, cet algorithme produit des représentations différentes.

Algorithme 6 Conversion randomisée vers le PMNS via Montgomery

Require: $a \in \mathbb{Z}/p\mathbb{Z}$ et $\boldsymbol{\mathcal{B}} = (p,\, n,\, \gamma,\, \rho)_E$
Ensure: $A(\gamma) \equiv a\,r \bmod p$
Data : $P_i \equiv (\rho^i)_{\boldsymbol{\mathcal{B}}}$, pour $i = 0, \ldots, n-1$, $z \in \mathbb{N}$, $r = 2^j$, $j \geq 1$ et $M \in \boldsymbol{\mathcal{B}}$ avec $M(\gamma) \equiv 0 \bmod p$ et $\mathrm{pgcd}(r,\, \mathrm{resultant}(E,\, M)) = 1$
1: $Z \leftarrow$ **randPoly**(z)
2: $a' \leftarrow a\,r^2 \bmod p$
3: $b \leftarrow (a'_{n-1}, ..., a'_0)_\rho$ # décomposition de a' en base ρ
4: $U \leftarrow \sum_{i=0}^{n-1} a'_i\,P_i$
5: $V \leftarrow U + ((r+1)Z \times M) \bmod E$
6: $A \leftarrow$ **RedCoeff**(V)
7: **return** A

Theorème II.2.1. *Soit $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$ un PMNS. Soit $a \in \mathbb{Z}/p\mathbb{Z}$ un entier et $r = 2^j$, $j \geq 1$. Soit $M \in \boldsymbol{\mathcal{B}}$ un polynôme tel que : $M(\gamma) \equiv 0 \bmod p$ et* pgcd*(r, resultant(E, M)) = 1. Soit z l'entier donné en entrée de la procédure* `randPoly`*. Nous considérons $\boldsymbol{\mathcal{B}}$, a,*

r, M et z comme entrées et données de l'Algorithme 6. Soit $m = \|M\|_\infty$.
Si ρ et r satisfont $\rho \geq 2\,n\,s\,m\,\left(1 + z + \frac{z}{r}\right)$ et $r \geq 2\,n\,s\,\rho$, alors l'Algorithme 6 génère une représentation aléatoire de loi uniforme sur un ensemble fini, composé de $(2\,z+1)^n$ représentations distinctes de a dans $\boldsymbol{\mathcal{B}}$.

Preuve. Premièrement, nous avons $M(\gamma) \equiv 0 \bmod p$, donc $A(\gamma) \equiv U(\gamma)\,r^{-1} \bmod p$. Ainsi, $A(\gamma) \equiv a'\,r^{-1} \bmod p$.

Ensuite, la sortie A de l'algorithme vérifie :

$$\begin{aligned} A &= r^{-1} \times ((V + Q \times M) \bmod E) \\ &= r^{-1} \times (U + ((r+1)Z \times M + Q \times M) \bmod E) \end{aligned}$$

avec $Q = ((U \times M' \bmod E) - Z) \bmod r$. Puisque nous avons $\|U\|_\infty < n\rho^2$, $\|ZM \bmod E\|_\infty < n\,s\,z\,m$ et $\|QM \bmod E\|_\infty < n\,s\,r\,m$, nous pouvons borner A de la façon suivante :

$$\begin{aligned} \|A\|_\infty &= \|r^{-1} \times (U + ((r+1)Z \times M + Q \times M) \bmod E)\|_\infty \\ &\leqslant (\|U\|_\infty + (r+1)\|Z \times M \bmod E\|_\infty + \|Q \times M \bmod E\|_\infty)/r \\ &< (n\,\rho^2 + (r+1)\,n\,s\,z\,m + n\,s\,r\,m)/r \\ &< \frac{n\,\rho^2}{r} + n\,s\,m\left(1 + z + \frac{z}{r}\right). \end{aligned}$$

Si $r \geq 2\,n\,s\,\rho$, alors $\frac{1}{r} \leq \frac{1}{2\,n\,s\,\rho} \leq \frac{1}{2\,n\,\rho}$. Donc, $\|A\|_\infty < \frac{\rho}{2} + n\,s\,m\,\left(1 + z + \frac{z}{r}\right)$. Afin de garantir $\|A\|_\infty < \rho$, nous devons nous assurer que $\rho \geq 2\,n\,s\,m\,\left(1 + z + \frac{z}{r}\right)$.

À présent, nous démontrons que pour deux polynômes aléatoires distincts et la même entrée $a \in \mathbb{Z}/p\mathbb{Z}$, l'algorithme retourne deux sorties distinctes A_1 et A_2. Nous désignons par p_i le i-ième coefficient d'un polynôme P. Soit Y et Z deux polynômes aléatoires tels que $Y \neq Z$, nous avons :

$$\begin{aligned} A_1 &= r^{-1} \times (U + ((r+1)YM + Q_1 M) \bmod E) \\ A_2 &= r^{-1} \times (U + ((r+1)ZM + Q_2 M) \bmod E). \end{aligned}$$

Si $A_1 = A_2$, alors : $((r+1)Y + Q_1) \times M) \bmod E = ((r+1)Z + Q_2) \times M) \bmod E$. Or resultant$(E,\, M) \neq 0$, donc pgcd$(E,\, M) = 1$ dans $\mathbb{Q}[X]$. Ainsi, $A_1 = A_2$ implique : $(r+1)\,(Y - Z) = Q_2 - Q_1$. Puisque $Y \neq Z$, il existe i tel que $|y_i - z_i| \geq 1$. D'où, $|(r+1)(y_i - z_i)| \geq r + 1 > r$. Mais nous avons nécessairement : $\|Q_2 - Q_1\|_\infty < r$, d'où $(r+1)\,(Y - Z) \neq Q_2 - Q_1$. Par conséquent, $A_1 = A_2$ est impossible, et les collisions sont évitées. Par ailleurs, pour tout $z \in \mathbb{N}$, **randPoly**(z) suit une loi uniforme sur un ensemble

fini de $(2\,z+1)^n$ polynômes. La sortie de l'algorithme suit donc une loi uniforme, sur l'ensemble des $(2\,z+1)^n$ représentations distinctes obtenues pour tous les polynômes possibles retournés par **randPoly**. □

b) Randomisation de la conversion via la méthode de type Babai

L'Algorithme 7 présenté ci-après est une légère modification de l'Algorithme 2 utilisant la méthode de réduction des coefficients de type Babai. Ici, les représentations dans le système PMNS sont vues comme des vecteurs, la i-ième coordonnée correspondant au i-ième coefficient de la forme polynomiale de la représentation. Nous considérons un système PMNS $\boldsymbol{\mathcal{B}}$ et son réseau associé $\mathfrak{L}_{\boldsymbol{\mathcal{B}}}$ avec une base LLL réduite, notée D. La randomisation de la méthode de type Babaï est basée sur deux vecteurs aléatoires :

— Soit V le vecteur *masque*. Il s'agit d'un vecteur du réseau $\mathfrak{L}_{\boldsymbol{\mathcal{B}}}$, i.e. une combinaison linéaire des éléments de D, ajouté à l'entrée au début de l'algorithme pour randomiser les calculs pendant l'exécution. Cet ajout produit une représentation congruente au vecteur d'entrée modulo $\mathfrak{L}_{\boldsymbol{\mathcal{B}}}$ puisque la combinaison linéaire des éléments de D représente 0 modulo p. Le masque n'affecte donc pas la sortie.

— Soit Z le vecteur *translation*. Pour chaque dimension de l'espace euclidien, l'élément dans la base D est multiplié par le coefficient aléatoire correspondant de Z pour obtenir une translation supplémentaire pendant l'algorithme et randomiser le résultat.

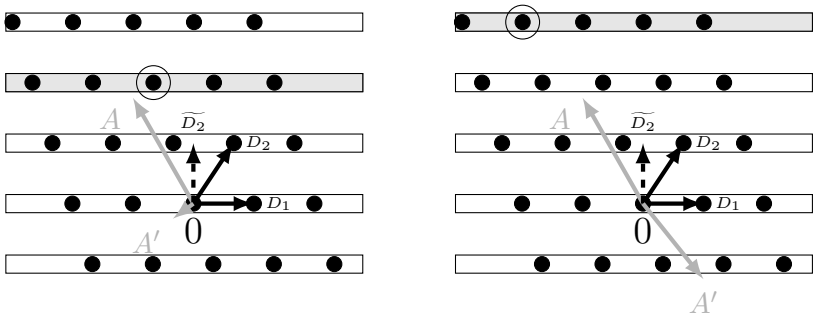


Fig. II.1 *Deux exécutions de l'algorithme de type Babai pour la réduction interne en dimension* 2*, avec* A *en entrée et* A' *en sortie. Non randomisé à gauche, et randomisé à droite avec le décalage* $Z = (-1, 1)$*. Les hyperplans sélectionnés dans* $\mathcal{L}_{\mathcal{B}}$ *sont grisés ou entourés. Dans les deux cas, l'utilisation d'un masque sur l'entrée n'a aucun effet sur la sortie, son rôle est de randomiser les calculs pendant l'exécution de l'algorithme.*

Theorème II.2.2. *Soit* $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$ *un PMNS et* $a \in \mathbb{Z}/p\mathbb{Z}$ *un entier. Soient* v *et* z *les entrées de la procédure* `randPoly` *respectivement pour les polynômes masque et translation. Soit* $M \in \boldsymbol{\mathcal{B}}$ *un polynôme tel que* $M(\gamma) \equiv 0 \bmod p$ *et* $\mathrm{pgcd}(E, M) = 1$ *dans*

Algorithme 7 Conversion randomisée vers le PMNS via Babaï

Require: $a \in \mathbb{Z}/p\mathbb{Z}$ et $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$
Ensure: $A(\gamma) \equiv a \bmod p$
Data : $P_i \equiv (\rho^i)_{\boldsymbol{\mathcal{B}}}$, pour $i = 0, \ldots, n-1$, $D = \{D_i, 1 \leq i \leq n\}$, $\widetilde{D} = \{\widetilde{D_i}, 1 \leq i \leq n\}$, $v \in \mathbb{N}$, $z \in \mathbb{N}$, $M \in \boldsymbol{\mathcal{B}}$ avec $M(\gamma) \equiv 0 \bmod p$ et $\mathrm{pgcd}(E, M) = 1$ dans $\mathbb{Q}[X]$, et $H \in \mathbb{Z}[X]$ tel que $\mathrm{pgcd}(E, H) = 1$ dans $\mathbb{Q}[X]$
1: $V_0 \leftarrow$ **randPoly**(v), $Z_0 \leftarrow$ **randPoly**(z)
2: $V \leftarrow V_0 \times M \bmod E$, $Z \leftarrow Z_0 \times H \bmod E$
3: $b \leftarrow (a_{n-1}, ..., a_0)_\rho$ # décomposition de a en base ρ
4: $T \leftarrow \sum_{i=0}^{n-1} a_i P_i$, $A \leftarrow T + V$
5: **for** $i = 1$ **to** n **do**
6: $\quad c \leftarrow \lfloor < A, \widetilde{D}_{n-i+1} > / \|\widetilde{D}_{n-i+1}\|^2 \rceil + z_{n-i}$
7: $\quad A \leftarrow A - c \times D_{n-i+1}$
8: **end for**
9: **return** A

$\mathbb{Q}[X]$, *et* $H \in \mathbb{Z}[X]$ *tel que* $\mathrm{pgcd}(E, H) = 1$ *dans* $\mathbb{Q}[X]$. *Nous considérons* $\boldsymbol{\mathcal{B}}$, a, v *et* z *comme les entrées et données de l'Algorithme 7. Soit* $h = \|H\|_\infty$.
Si ρ *satisfait* $\rho > \left(\frac{1}{2} + n\,s\,z\,h\right)\left(2^{\frac{3n-1}{2}} p^{1/n}\right)$, *alors l'Algorithme 7 génère une représentation aléatoire de loi uniforme sur un ensemble fini, composé de* $(2\,z+1)^n$ *représentations distinctes de* a *dans* $\boldsymbol{\mathcal{B}}$.

Preuve. Nous prouvons d'abord que l'ajout du vecteur masque V à la ligne 4 ne change pas le résultat de l'algorithme. Comme $M(\gamma) \equiv 0 \bmod p$ et $E(\gamma) \equiv 0 \bmod p$, $V(\gamma) = V_0 \times M(\gamma) + K \times E(\gamma) \equiv 0 \bmod p$, d'où $V \in \mathfrak{L}_{\boldsymbol{\mathcal{B}}}$. En considérant le vecteur $T \in \mathbb{Z}^n$ calculé à la ligne 4, nous montrons alors que remplacer T par $T + V$, pour un vecteur $V \in \mathfrak{L}_{\boldsymbol{\mathcal{B}}}$, avant la boucle de l'algorithme est sans effet sur la sortie. Nous le démontrons en ajoutant à T un vecteur D_i de la base D, $1 \leq i \leq n$. Pour le cas où aucun masque n'est ajouté, nous notons $(A_i)_{0 \leq i \leq n}$ la suite des valeurs de A avant et à chaque itération de la boucle, avec $A_0 = T$ et A_n la sortie de l'algorithme, et la suite correspondante $(c_i)_{0 \leq i \leq n}$. Pour le cas où le masque D_i est ajouté, nous définissons les deux séquences $(A'_i)_{0 \leq i \leq n}$ et $(c'_i)_{0 \leq i \leq n}$. Ici $A_0 = T = T + D_i - D_i = A'_0 - D_i$. Par induction, puisque pour $j \leq n - i$, le produit scalaire $< \widetilde{D}_{n-j+1}, A'_{j-1} - D_i >$ est égal à $< \widetilde{D}_{n-j+1}, A'_{j-1} >$, on a $c_j = c'_j$ et

$$A_j = A_{j-1} - c_j D_{n-j+1} = A'_{j-1} - D_i - c'_j D_{n-j+1} = A'_j - D_i.$$

Puis, à l'étape $j = n - i + 1$, on obtient

$$c_j = \frac{< A_{j-1}, \widetilde{D}_i >}{\|\widetilde{D}_i\|^2} = \frac{< A'_{j-1} - D_i, \widetilde{D}_i >}{\|\widetilde{D}_i\|^2} = \frac{< A'_{j-1}, \widetilde{D}_i >}{\|\widetilde{D}_i\|^2} - 1 = c'_j - 1.$$

Pour ce j, cela implique

$$\begin{aligned} A_j &= A_{j-1} - c_j D_i \\ &= A'_{j-1} - D_i - (c'_j - 1)D_i \\ &= A'_{j-1} - c'_j D_i = A'_j \end{aligned}$$

et la sortie est identique. Par ailleurs, pour deux vecteurs masque V et V', si $V = V'$ alors $V_0 \times M \bmod E = V'_0 \times M \bmod E$. Or $\text{pgcd}(E, M) = 1$ dans $\mathbb{Q}[X]$, donc $V = V'$ implique $V_0 = V'_0$.

Ensuite, nous prouvons que la sortie de l'algorithme appartient au système $\boldsymbol{\mathcal{B}}$. Les vecteurs de Gram-Schmidt sont orthogonaux, ce qui implique que chaque vecteur $u \in \mathbb{R}^n$ peut s'écrire sous la forme $u = \sum \frac{<\widetilde{D}_i, u>}{\|\widetilde{D}_i\|^2} \cdot \widetilde{D}_i$. À la i-ième itération de la boucle, l'algorithme soustrait à A le vecteur $(\lfloor < A, \widetilde{D}_{n-i+1} > / \|\widetilde{D}_{n-i+1}\|^2 \rfloor + z_{n-i}) \cdot D_{n-i+1}$. Puisque $Z = Z_0 \times H \bmod E$, $\|Z\|_\infty < n\,s\,z\,h$. La sortie se trouve alors dans le rectangle

$$\left\{ \sum a_i \widetilde{D}_i \text{ avec } |a_i| \leq \frac{1}{2} + n\,s\,z\,h \right\},$$

d'où

$$\|A\|^2 \leq \left(\frac{1}{2} + n\,s\,z\,h\right)^2 \left(\sum_{i=1}^{n} \|\widetilde{D}_i\|\right)^2.$$

De plus, puisque la base D est une base LLL réduite, pour $1 \leq i \leq n$, $\|\widetilde{D}_i\| \leq 2^{\frac{n-i}{2}} \|\widetilde{D}_n\|$, ce qui implique

$$\begin{aligned} \|A\| &\leq \left(\frac{1}{2} + n\,s\,z\,h\right) \left(\sum_{i=1}^{n} 2^{\frac{n-i}{2}} \|\widetilde{D}_n\|\right), \\ &\leq \left(\frac{1}{2} + n\,s\,z\,h\right) \left(2^{\frac{n-1}{2}} \|\widetilde{D}_n\|\right). \end{aligned}$$

Une borne sur le vecteur $\widetilde{D}_n$ est donnée par le Théorème 1 dans [PSZ15], avec $\|\widetilde{D}_n\| \leq 2^n p^{1/n}$. Ainsi, nous obtenons

$$\|A\|_\infty \leq \|A\| \leq \left(\frac{1}{2} + n\,s\,z\,h\right) \left(2^{\frac{3n-1}{2}} p^{1/n}\right).$$

Afin de garantir $\|A\|_\infty < \rho$, nous devons nous assurer que $\rho > \left(\frac{1}{2} + n\,s\,z\,h\right) \left(2^{\frac{3n-1}{2}} p^{1/n}\right)$.

À présent, nous démontrons que pour deux vecteurs aléatoires distincts Z_0 et Z'_0 et la même entrée $a \in \mathbb{Z}/p\mathbb{Z}$, l'algorithme retourne deux sorties distinctes. Pour deux vec-

teurs de translation Z et Z', si $Z = Z'$ alors $Z_0 \times H \bmod E = Z'_0 \times H \bmod E$. Nous avons $\text{pgcd}(E, H) = 1$ dans $\mathbb{Q}[X]$, d'où $Z = Z'$ implique $Z_0 = Z'_0$. Pour une entrée A donnée et un vecteur translation Z, et pour i de $n-1$ à 0, la i-ième coordonnée z_i de Z représente la translation additionnelle $-z_i\, D_{i+1}$ appliquée à A à la i-ième itération. Puisque l'algorithme calcule la i-ième coordonnée du résultat en soustrayant un point appartenant à un hyperplan de dimension i et à la distance $z_i \|\widetilde{D}_{i+1}\|$ du point $\lfloor < A, \widetilde{D}_{i+1} > / \|\widetilde{D}_{i+1}\|^2 \rceil D_{i+1}$, soustraire un point appartenant à un hyperplan différent entraîne un résultat différent. Ainsi, pour une même entrée et deux vecteurs aléatoires distincts, l'algorithme retourne deux sorties distinctes, et les collisions sont évitées. Par ailleurs, pour tout $z \in \mathbb{N}$, **randPoly**(z) suit une loi uniforme sur un ensemble fini de $(2\,z+1)^n$ polynômes. La sortie de l'algorithme suit donc une loi uniforme, sur l'ensemble des $(2\,z+1)^n$ représentations distinctes obtenues pour tous les polynômes possibles retournés par **randPoly**. □

II.2.3 Randomisation de la multiplication

Dans les systèmes PMNS, la randomisation de la multiplication fournit un niveau supplémentaire de sécurité lorsqu'elle est utilisée avec la conversion randomisée. Dans ce cas, la multiplication de deux entiers modulo un premier au sein du système retourne une représentation aléatoire de loi uniforme sur un ensemble fini, composé de représentations distinctes du produit dans le système. Toutefois, la combinaison de ces deux randomisations entraîne des contraintes plus fortes sur certains paramètres du PMNS.

Dans les deux versions, nous randomisons une entrée afin de randomiser tous les résultats intermédiaires. De plus, pour une même entrée et des polynômes aléatoires différents (vecteurs de translation pour Babaï), les résultats calculés par ces algorithmes sont différentes représentations du même entier dans $\mathbb{Z}/p\mathbb{Z}$.

a) Randomisation de la multiplication via Montgomery

Cet algorithme est une variation de la multiplication modulaire de Montgomery.

Theorème II.2.3. *Soit $\mathcal{B} = (p, n, \gamma, \rho)_E$ un PMNS. Soient $r = 2^j$, $j \geq 1$ et $z \in \mathbb{N}$ la borne sur les polynômes randomisés. Soit $M \in \mathcal{B}$ un polynôme tel que : $M(\gamma) \equiv 0 \bmod p$ et* pgcd*(r, resultant(E, M)) = 1. Soit A et B deux élément de $\mathcal{B}$. Nous considérons $\mathcal{B}$, A, B, M, z et r comme les entrées et données de l'Algorithme 8. Soit $m = \|M\|_\infty$.*
Si ρ et r satisfont $\rho \geq 2\,n\,s\,m\,(2z+1)$ et $r \geq 2\,n\,s\,\rho \times max\left(z, \frac{5}{4}\right)$, alors l'Algorithme 8 génère une représentation aléatoire de loi uniforme sur un ensemble fini, composé de $(2\,z+1)^n$ représentations distinctes de $A(\gamma)B(\gamma)r^{-1} \bmod p$ dans $\mathcal{B}$.

Preuve. Pour commencer, nous avons $M(\gamma) \equiv 0 \bmod p$, donc $J(\gamma) \equiv 0 \bmod p$ et $Q(\gamma)M(\gamma) \equiv 0 \bmod p$. Ainsi, $R(\gamma) \equiv A(\gamma)B(\gamma)r^{-1} \bmod p$.

Algorithme 8 Multiplication PMNS randomisée via Montgomery

Require: $\boldsymbol{\mathcal{B}} = (p,\, n,\, \gamma,\, \rho)_E$ et $A, B \in \boldsymbol{\mathcal{B}}$
Ensure: $R(\gamma) \equiv A(\gamma)B(\gamma)r^{-1} \bmod p$
Data : $r = 2^j$, $j \geq 1$, $z \in \mathbb{N}$, $M \in \boldsymbol{\mathcal{B}}$, tel que : $M(\gamma) \equiv 0 \bmod p$ et pgcd(r, resultant(E, M)) $= 1$, $M' = -M^{-1} \bmod (E, r)$
1: $Z \leftarrow$ **randPoly**(z)
2: $J \leftarrow Z \times M \bmod E$, $B' \leftarrow B + J$
3: $C \leftarrow (A \times B') \bmod E$
4: $Q \leftarrow (C \times M') \bmod (E, r)$
5: $R' \leftarrow C + (Q \times M) \bmod E$
6: $R \leftarrow R'/r + 2 \times J$
7: **return** R

Ensuite, la sortie de l'algorithme peut être écrite sous la forme :

$$\begin{aligned} R &= r^{-1} \times ((C + Q \times M) \bmod E) + 2 \times J, \\ &= r^{-1} \times ((A \times B + A \times J + Q \times M) \bmod E) + 2 \times J, \end{aligned}$$

avec $Q = ((A \times (B + J) \times M') \bmod E) \bmod r$.

Nous avons :

$$\begin{cases} \|J\|_\infty = \|Z \times M \bmod E\|_\infty < ns\|M\|_\infty \|Z\|_\infty \leq nsmz \;\Rightarrow\; \|J\|_\infty < nsmz, \\ \rho \geq 2nsm(2z+1) > 4nsmz \;\Rightarrow\; \rho > 4nsmz \;\Rightarrow\; \|J\|_\infty < \rho/4, \\ \|B'\|_\infty = \|B + J\|_\infty < \|B\|_\infty + \|J\|_\infty \;\Rightarrow\; \|B'\|_\infty < \rho + \rho/4 = \frac{5}{4}\,\rho. \end{cases}$$

D'où,

$$\begin{aligned} \|R\|_\infty &= \|r^{-1} \times ((A \times B' + Q \times M) \bmod E) + 2 \times J\|_\infty, \\ &< r^{-1} n\, s(\|A\|_\infty \|B'\|_\infty + \|Q\|_\infty \|M\|_\infty) + 2\|J\|_\infty, \\ &< n\, s \left(\frac{5}{4}\,\rho^2 + r\, m\right)/r + 2\, n\, s\, m\, z \\ &= \left(\frac{5}{4}\, n\, s\, \rho^2\right)/r + n\, s\, m(2\, z + 1). \end{aligned}$$

Si $r \geq 2\, n\, s\, \rho \times max(z, \frac{5}{4})$, alors $\frac{1}{r} \leq 1/(2\, n\, s\, \rho \times \frac{5}{4})$. Donc, $\|R\|_\infty < \frac{\rho}{2} + n\, s\, m(2\, z + 1)$.

Afin de garantir $\|R\|_\infty < \rho$, ρ doit satisfaire : $\rho \geq 2\, n\, s\, m(2\, z + 1)$.

Maintenant, nous démontrons que pour deux polynômes aléatoires distincts et les mêmes entrées A et B, il n'y a pas de collision sur le résultat, c'est-à-dire que l'algorithme retourne deux sorties distinctes. Dans le reste de la démonstration, on désigne par p_i le i-ième coefficient d'un polynôme P. En fixant A, B et M, et Z et Y deux polynômes aléatoires tels que $Z \neq Y$, nous avons :

$$R_1 = r^{-1} \times (A\,B \bmod E + (A\,Z\,M + Q_1\,M + 2\,r\,Z\,M) \bmod E),$$
$$R_2 = r^{-1} \times (A\,B \bmod E + (A\,Y\,M + Q_2\,M + 2\,r\,Y\,M) \bmod E).$$

Nous supposons $R_1 = R_2$, i.e :

$$(A\,Z\,M + Q_1\,M + 2\,r\,Z\,M) \bmod E = (A\,Y\,M + Q_2\,M + 2\,r\,Y\,M) \bmod E.$$

Nous avons resultant$(E,\, M) \neq 0$, donc pgcd$(E,\, M) = 1$. Ainsi, $R_1 = R_2$ implique :

$$(A \times Z \bmod E) + Q_1 + 2\,r\,Z = (A \times Y \bmod E) + Q_2 + 2\,r\,Y.$$

C'est-à-dire : $Q_1 - Q_2 = 2\,r\,(Y - Z) + A\,(Y - Z) \bmod E$.

Soit $T = A\,(Y - Z) \bmod E$. Nous avons : $\|T\|_\infty \leq 2\,n\,s\,\rho\,z \leq r$.

Puisque $Z \neq Y$, il existe i tel que $|z_i - y_i| \geq 1$, alors : $|2\,r\,(z_i - y_i)| \geq 2\,r$. Ainsi, $|2\,r\,(z_i - y_i) + t_i| \geq r$. Mais, nous avons nécessairement : $\|Q_1 - Q_2\|_\infty < r$.
Donc, $2\,r\,(Y - Z) + T \neq Q_1 - Q_2$.
Par conséquent, $R_1 = R_2$ est impossible et les collisions sont donc évitées. Par ailleurs, pour tout $z \in \mathbb{N}$, **randPoly**(z) suit une loi uniforme sur un ensemble fini de $(2\,z + 1)^n$ polynômes. La sortie de l'algorithme suit donc une loi uniforme, sur l'ensemble des $(2\,z+1)^n$ représentations distinctes obtenues pour tous les polynômes possibles retournés par **randPoly**. □

Remarque II.2.1. Plutôt que de considérer les bornes de ρ et r du Théorème II.2.1 pour la conversion vers le PMNS, les bornes du Théorème II.2.3 doivent être utilisées pour la conversion si des multiplications sont randomisées, comme décrit ci-dessus.

b) Randomisation de la multiplication via Babaï

Cette randomisation repose sur les mêmes principes que la conversion randomisée décrite à la Section II.2.2.b). Le caractère aléatoire est introduit dans la multiplication par le vecteur masque V et le vecteur translation Z.

Algorithme 9 Multiplication PMNS randomisée via Babaï

Require: $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$ et $A, B \in \boldsymbol{\mathcal{B}}$
Ensure: $R \in \boldsymbol{\mathcal{B}}$ et $R(\gamma) \equiv A(\gamma)B(\gamma) \bmod p$
Data : $D = \{D_i, 1 \leq i \leq n\}$, $\widetilde{D} = \{\widetilde{D_i}, 1 \leq i \leq n\}$, $v \in \mathbb{N}$, $z \in \mathbb{N}$, $M \in \boldsymbol{\mathcal{B}}$ avec $M(\gamma) \equiv 0 \bmod p$ et $\text{pgcd}(E, M) = 1$ dans $\mathbb{Q}[X]$, et $H \in \mathbb{Z}[X]$ tel que $\text{pgcd}(E, H) = 1$ dans $\mathbb{Q}[X]$
1: $V_0 \leftarrow$ **randPoly**(v), $Z_0 \leftarrow$ **randPoly**(z)
2: $V \leftarrow V_0 \times M \bmod E$, $Z \leftarrow Z_0 \times H \bmod E$
3: $B' \leftarrow B + V$, $R \leftarrow A \times B' \bmod E$
4: **for** $i := 1$ **to** n **do**
5: $\quad c \leftarrow \lfloor < R, \widetilde{D}_{n-i+1} > / \|\widetilde{D}_{n-i+1}\|^2 \rceil + z_{n-i}$
6: $\quad R \leftarrow R - c \times D_{n-i+1}$
7: **end for**
8: **return** R

Le Théorème II.2.2 s'applique également à l'Algorithme 9, qui génère une représentation aléatoire de loi uniforme sur un ensemble fini, composé de $(2\,z\,+\,1)^n$ représentations distinctes de $A(\gamma)B(\gamma) \bmod p$ dans le PMNS $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$, avec $\rho > \left(\frac{1}{2} + n\,s\,z\,h\right)\left(2^{\frac{3n-1}{2}} p^{1/n}\right)$.

II.2.4 Estimation des coûts

Nous décrivons ici l'estimation théorique des coûts des algorithmes de multiplication présentés à la section précédente. Ces derniers sont calculés en fonction du nombre de multiplications, d'additions, de divisions et de décalages de taille w-bit.

Nous supposons que les entrées de ces algorithmes appartiennent à $\boldsymbol{\mathcal{B}} = (p, n, \gamma, \rho)_E$, avec $\rho = 2^w$, $E(X) = X^n - \lambda$ et $\lambda = \pm 2^u$ avec $w, u \in \mathbb{N}$. Il a été démontré dans [DDV18] qu'il est toujours possible de construire de tels PMNS. Par exemple, avec ces notations, l'addition de deux entiers de c bits nécessite $\lceil c/w \rceil$ w-bit additions.

Soient $\mathscr{M}$ et $\mathscr{A}$ désignant respectivement la multiplication et la somme de deux entiers de w bits, $\mathscr{I}$ la division par un entier de $2\,w + \lfloor \log_2(n) \rfloor$ bits (le résultat de n additions d'entiers de $2w$ bits), et $\mathscr{R}$ le coût d'un appel à la fonction **randPoly**. Nous désignons aussi respectivement $\mathscr{S}_l^i$ et $\mathscr{S}_r^i$ un décalage à gauche et un décalage à droite de i bits .

Dans le Tableau II.1, nous comparons le coût de la multiplication randomisée dans le PMNS avec celui de leurs homologues non randomisés. La version non randomisée consiste en une multiplication polynomiale suivie d'une réduction interne de type Babaï (Alg. 2) ou Montgomery (Alg. 1).

Afin de protéger l'ECSM $k\mathcal{P}$ contre les attaques DPA, les contre-mesures classiques randomisent au préalable le scalaire k et le point $\mathcal{P}$ [FV12]. Puis, au cours du processus de calcul, les points intermédiaires calculés sont également randomisés. L'idée de ces

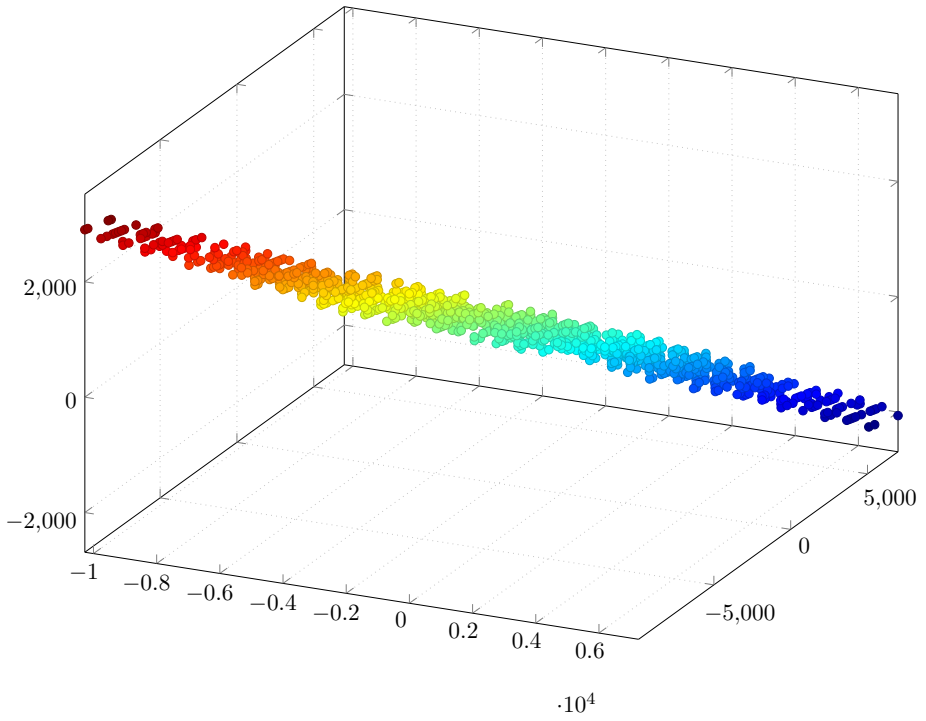


Fig. II.2 *Vecteurs retournés pour 1000 exécutions de la multiplication randomisée via Babaï pour les mêmes entrées* $\mathcal{B} = (p, n, \gamma, \rho)_E$ *et* $A, B \in \mathcal{B}$, *avec* $p = 45667$, $\gamma = 15943$, $E(X) = X^3 - X + 1$, $R(\gamma) = 44981 \bmod p$ *et* $z = 6$.

TABLE II.1 – Coût théoriques des opérations, quand $E(X) = X^n - \lambda$ avec $\lambda = \pm 2^u$, $r = 2^j$.

Méthode de mult.	type Montgomery	type Babaï
Mult. polynomiale	$n^2\mathscr{M} + (3n^2 - 6n + 3)\mathscr{A}$	$n^2\mathscr{M} + (3n^2 - 6n + 3)\mathscr{A}$
Réduction polynomiale	$3(n-1)\mathscr{A} + (n-1)\mathscr{S}_l^u$	$3(n-1)\mathscr{A} + (n-1)\mathscr{S}_l^u$
Réduction interne	$2n^2\mathscr{M} + (4n^2 - n)\mathscr{A} + n\mathscr{S}_r^j$	$2n^2\mathscr{M} + (6n^2 - 3n)\mathscr{A} + n\mathscr{I}$
Total	$3n^2\mathscr{M} + (7n^2 - 4n)\mathscr{A} + (n-1)\mathscr{S}_l^u + n\mathscr{S}_r^j$	$3n^2\mathscr{M} + (9n^2 - 6n)\mathscr{A} + n\mathscr{I} + (n-1)\mathscr{S}_l^u$
Méthode de mult.	Montgomery randomisé (Alg. 8)	Babaï randomisé (Alg. 9)
Mult. polynomiale	$2n^2\mathscr{M} + (4n^2 - 7n + 3)\mathscr{A} + \mathscr{R}$	$3n^2\mathscr{M} + (5n^2 - 8n + 3)\mathscr{A} + 2\mathscr{R}$
Réduction polynomiale	$3(n-1)\mathscr{A} + (n-1)\mathscr{S}_l^u$	$3(n-1)\mathscr{A} + (n-1)\mathscr{S}_l^u$
Réduction interne	$2n^2\mathscr{M} + (4n^2 + n)\mathscr{A} + n(\mathscr{S}_r^j + \mathscr{S}_l^1)$	$2n^2\mathscr{M} + (6n^2 - 2n)\mathscr{A} + n\mathscr{I}$
Total	$4n^2\mathscr{M} + (8n^2 - 3n)\mathscr{A} + (n-1)\mathscr{S}_l^u + n(\mathscr{S}_l^1 + \mathscr{S}_r^j) + \mathscr{R}$	$5n^2\mathscr{M} + (11n^2 - 7n)\mathscr{A} + n\mathscr{I} + (n-1)\mathscr{S}_l^u + 2\mathscr{R}$

approches est d'empêcher un attaquant d'obtenir des informations sur k ou de faire des hypothèses utiles sur les valeurs intermédiaires de l'ECSM.

Dans la représentation binaire classique, ces contre-mesures usuelles semblent inefficaces contre l'attaque de Goubin [Gou03]. Cette attaque peut être effectuée sur des courbes qui ont au moins un point avec une coordonnée égale à zéro. Sur une telle courbe, en utilisant un tel point, l'attaquant peut trouver des informations sur la clé secrète. Il est possible de contrecarrer cette attaque au prix d'une ECSM supplémentaire qui doit être faite en plus des contre-mesures classiques [FV12].

Un premier avantage de nos solutions randomisées basées sur le PMNS est que, quel que soit le type de courbe, cette attaque ne peut être effectuée. En effet, il y a au moins $(2z+1)^n$ représentants distincts de $0 \in \mathbb{Z}/p\mathbb{Z}$ et ces représentants n'ont pas de formes particulières. Ainsi, pour z assez grand, l'attaquant ne devrait pas être capable d'exploiter la moindre information pour effectuer cette attaque.

Par conséquent, la randomisation du processus de conversion utilisant les Algorithmes 6 ou 7 suffit à elle seule à contrer l'attaque de Goubin, même si des multiplications non randomisées sont utilisées ultérieurement.

Un autre avantage de notre approche est qu'elle fonctionne au niveau arithmétique. Elle peut donc être combinée avec d'autres contre-mesures classiques (point blinding, scalar blinding) afin de randomiser $\mathcal{P}$ et les points intermédiaires.

II.3 Conclusion

Ce travail a permis de montrer comment utiliser la redondance du système PMNS pour construire des protections arithmétiques efficaces contre les attaques DPA et a donné lieu à une publication à la conférence du $26^{ème}$ IEEE International Symposium on Computer Arithmetic (ARITH-26) [DDEM+19]. Nous avons décrit comment randomiser les données en entrée lors de la conversion vers le PMNS au moyen de deux méthodes. Nous avons également introduit deux multiplications modulaires randomisées sur le PMNS.

Ces méthodes peuvent être utilisées pour appliquer des contre-mesures classiques sur la multiplication scalaire $k\mathcal{P}$ sur les courbes elliptiques. De plus, nous avons montré que la randomisation du processus de conversion suffit à elle seule à protéger les données contre l'attaque de Goubin. À titre de comparaison, une contre-mesure sûre utilisant la représentation binaire classique, nécessite une ECSM supplémentaire $k\mathcal{R}$, pour un point aléatoire $\mathcal{R}$ [FV12]. Ces résultats constituent une première étape dans l'utilisation de la randomisation des opérations arithmétiques sur le PMNS. Ces travaux ouvrent de nouvelles perspectives dans le domaine des contre-mesures aux attaques SCA, dont l'étude de l'efficacité pratique de ces randomisations, ainsi qu'une comparaison exhaustive avec les contre-mesures existantes afin d'établir quand ces méthodes sont les plus pertinentes.

Chapitre III

Un système de représentation hybride pour ECC : HyPoRes

Sommaire

Le travail présenté dans ce chapitre est consacré à l'introduction d'un nouveau système de représentation hybride, permettant d'améliorer les réductions modulaires pour tous les nombres premiers. Pour ce faire, nous tirons avantage du système modulaire de représentation, ou Residue Number System (RNS), qui trouve de nombreuses applications sur les systèmes cryptographiques [SAM16], grâce au fait qu'il réduit la complexité des additions et des multiplications des longs entiers. Toutefois, en raison de la nature non positionnelle du RNS, les opérations telles que les réductions et les comparaisons modulaires sont plus difficiles à implémenter de manière efficace. La complexité de la plupart des multiplications modulaires basées sur RNS est dominée par les extensions de base RNS [BI04, ABS12]. Dans [BT16], ce problème est atténué par une représentation hybride positionnelle-RNS, appelée Hybrid-Positional Residue Number System (HPR). Les nombres sont exprimés dans une représentation positionnelle, avec les coefficients représentés en RNS. Cette approche réduit la taille des bases RNS, réduisant ainsi la complexité des extensions de base, tout en bénéficiant de l'indépendance arithmétique des canaux RNS.

Bigou et Tisserand ont appliqué [BT16] à la cryptographie sur les courbes elliptiques (ECC) en construisant des nombres premiers sur mesure de la forme $p = B_1^n - \beta$, où B_1 correspond à l'intervalle dynamique d'un RNS, n est le nombre de chiffres positionnels et β un petit entier, permettant ainsi des réductions efficaces modulo p. Après un produit, les chiffres de poids fort, ou Most Significant Digits (MSDs), sont multipliés par β et ajoutés aux chiffres de poids faible, ou Least Significant Digits (LSDs). Ensuite, la taille des coefficients est réduite au moyen de propagations de retenue. Puisque la propagation de retenue nécessite de calculer une division et un arrondi, des extensions de base sont toujours nécessaires, mais avec des bases d'une taille inférieure à celles d'une approche générale RNS.

Bien que les méthodes décrites dans [BT16] constituent une approche efficace pour ECC, nous avons rarement l'occasion de choisir le premier p sous-jacent, puisque ceux-ci sont standardisés pour la plupart des applications cryptographiques. Pour ces raisons, nous proposons dans ce chapitre le nouveau système de représentation hybride polynomial, ou Hybrid-Polynomial Residue Number System (HyPoRes), qui est de nature hybride et permet des réductions efficaces pour tout premier. Nous l'obtenons en découplant la base du système de représentation positionnel de l'intervalle dynamique du RNS. Après un produit, nous pouvons toujours multiplier les MSDs par une petite constante et les ajouter aux LSDs. Cependant, la propagation de retenue n'est plus possible. Par conséquent, pour réduire la taille des coefficients, une réduction de Montgomery, qui exploite une représentation redondante de zéro, est utilisée.

Puisque la réduction de Montgomery est plus coûteuse en calcul que la propagation de retenue, les résultats expérimentaux montrent que HyPoRes est au plus 1,4 fois plus lent que HPR. Néanmoins, alors que HPR ne supporte pas les nombres premiers actuellement

standardisés pour ECC, notre approche est applicable à tout nombre premier, atteignant des accélérations allant jusqu'à 1,4 en comparaison avec les approches générales RNS. Ces résultats montrent un lien entre la solidité des hypothèses sur lesquelles repose une représentation et son efficacité. D'abord, HPR se limite aux nombres premiers avec une forme particulière et obtient les meilleures performances. Ensuite, HyPoRes repose sur le précalcul de valeurs qui nécessitent de connaître la factorisation du module sous-jacent, produisant une bonne performance. Enfin, les approches générales RNS ne font aucune hypothèse sur le module sous-jacent, ce qui se traduit par une performance raisonnable.

Le reste du chapitre est organisé comme suit. La Section III.1 fournit le contexte nécessaire sur l'arithmétique modulaire, le RNS et les réseaux, pour construire le système HyPoRes proposé. Le système hybride est décrit à la Section III.2. La Section III.3 présente l'art associé. L'approche proposée est comparée à l'art associé aux Sections III.4 et III.5, d'un point de vue théorique et pratique, respectivement. Des techniques de représentation permettant d'améliorer la résistance contre les attaques par canaux cachéss (SCAs) sont présentées à la Section III.6. Enfin, la Section III.7 conclut le chapitre.

III.1 Préliminaires

Dans cette section, les résultats fondamentaux sur l'arithmétique modulaire, le RNS et les réseaux sont passés en revue. L'opération modulo correspond au calcul du reste de la division euclidienne. Le théorème de la division euclidienne est le suivant :

À deux entiers $a, p \in \mathbb{Z}$ avec $p > 1$, on associe un unique couple $(q, r) \in \mathbb{Z} \times \mathbb{N}$ tel que

$$a = pq + r \quad \text{et} \quad 0 \leq r < p.$$

Les entiers q et r sont respectivement appelés le quotient et le reste de la division euclidienne de a par p. Afin d'obtenir des restes plus petits, nous utilisons dans la suite une variante de la division euclidienne, avec un reste dit centré [OS12],

$$a = pq + r \quad \text{et} \quad -\frac{1}{2}p \leq r < \frac{1}{2}p.$$

Le couple (q, r) est de nouveau unique. En notant q' et r' le quotient et le reste de la division euclidienne classique, nous retrouvons ceux de la division centrée comme suit :

$$r = \begin{cases} r' & \text{si} \quad 2r' \leq p, \\ r' - p & \text{si} \quad 2r' > p\,; \end{cases} \qquad q = \begin{cases} q' & \text{si} \quad 2r' \leq p, \\ q' - 1 & \text{si} \quad 2r' > p. \end{cases}$$

Dans ce chapitre, $[\cdot]_p$ est utilisé pour désigner le reste centré dans la division euclidienne par p et ppcm le plus petit multiple commun. Notons que les techniques proposées ici

peuvent être légèrement modifiées pour traiter les restes positifs plutôt que centrés.

III.1.1 Arithmétique modulaire

La plupart des branches de la cryptographie utilisent l'arithmétique modulaire. Dans un tel système, deux entiers $a, b \in \mathbb{Z}$ sont dits congruents si leur différence $(a - b)$ est divisible par un module p. Cette relation se note $a \equiv b \bmod p$. En outre, nous désignons par $\mathbb{Z}/p\mathbb{Z}$ l'ensemble des classes d'équivalence modulo p. Après chaque addition $(a + b)$ ou multiplication $(a \times b)$ dans $\mathbb{Z}/p\mathbb{Z}$, le résultat c est réduit modulo p en un entier plus petit, pour diminuer les besoins en mémoire et le coût des opérations à venir.

Cependant, en ECC, p est un grand nombre premier, en général sur des centaines de bits. Il est alors avantageux de faire correspondre les opérations modulo p à d'autres opérations avec des modules plus petits $b_{i,j}$, compatibles avec les architectures informatiques existantes. Pour y parvenir efficacement, le système de représentation proposé ici nécessite de pré-calculer des racines n-ième dans $(\mathbb{Z}/p\mathbb{Z})^*$. Le Lemme III.1.1 précise sous quelles conditions une valeur r a une racine n-ième dans $(\mathbb{Z}/p\mathbb{Z})^*$, c'est-à-dire qu'il existe $x \in (\mathbb{Z}/p\mathbb{Z})^*$ tel que $[x^n]_p = r$. Cette racine peut être calculée avec [Joh99]. Par ailleurs, dans la suite, $a^{-1} \bmod p$ et $1/a \bmod p$ sont tous deux utilisés pour désigner l'inverse modulaire de a modulo p, c'est-à-dire l'entier $-p/2 \leq x < p/2$ tel que $xa \equiv 1 \bmod p$.

Lemme III.1.1. *L'entier $r \in (\mathbb{Z}/p\mathbb{Z})^*$ a une racine n-ième modulo p, avec p un nombre premier, si et seulement si $\left[r^t\right]_p = 1$ pour $\mathrm{ppcm}(n, p-1) = nt$.*

Preuve. Si $\mathrm{ppcm}(n, p-1) = n(p-1)$, alors $\mathrm{pgcd}(n, p-1) = 1$ et $\left[r^t\right]_p = \left[r^{p-1}\right]_p = 1$. Une racine de r peut être calculée via la formule $\left[r^{[1/n]_{p-1}}\right]_p$.

Sinon, $\mathrm{ppcm}(n, p-1) = nt$ et $\mathrm{pgcd}(n, p-1) = (p-1)/t$. En supposant que g soit un générateur de $(\mathbb{Z}/p\mathbb{Z})^*$, nous avons $r \equiv g^y \bmod p$ pour un certain $y \in \mathbb{Z}$. Donc $r^t \equiv g^{ty} \equiv 1 \bmod p$ si et seulement si $p - 1 \mid ty$. C'est le cas si et seulement si $(p-1)/t \mid y$, i.e. il existe m tel que $y = ((p-1)/t)\ m$. Par l'algorithme d'Euclide étendu, on calcule les coefficients de Bezout u et v tels que :

$$un + v(p-1) = (p-1)/t.$$

En multipliant cette équation par m, on obtient

$$\begin{aligned} unm + v(p-1)m &= ((p-1)/t)\ m \\ &= y. \end{aligned}$$

On peut donc réécrire

$$\begin{aligned} g^y &\equiv r \bmod p \\ g^{unm+v(p-1)m} &\equiv r \bmod p \\ (g^{um})^n (g^{(p-1)})^{vm} &\equiv r \bmod p. \end{aligned}$$

Or, d'aprés le petit théoréme de Fermat, si p est premier, $g^{(p-1)} \equiv 1 \bmod p$. On en déduit

$$(g^{um})^n \equiv r \bmod p.$$

Donc $r \in (\mathbb{Z}/p\mathbb{Z})^*$ a une racine n-ième modulo p. □

L'arithmétique modulaire s'étend naturellement aux polynômes. Deux polynômes dans $\mathbb{Z}[X]/f(X)$ sont congruents lorsque leur différence est divisible par $f(X)$; et ils sont également congruents dans $(\mathbb{Z}/p\mathbb{Z}[X])/f(X)$ lorsque leur différence est divisible par $f(X)$.

III.1.2 RNS

Le Théorème des restes chinois (CRT) stipule que pour des nombres entiers premiers entre eux $b_{1,0}, \ldots, b_{1,h_1-1}$ et pour $B_1 = b_{1,0} \times \ldots \times b_{1,h_1-1}$, l'anneau $\mathbb{Z}_{B_1}$ est isomorphe à $\mathbb{Z}_{b_{1,0}} \times \ldots \times \mathbb{Z}_{b_{1,h_1-1}}$ via la fonction suivante :

$$\begin{aligned} g &: \mathbb{Z}_{B_1} \to \mathbb{Z}_{b_{1,0}} \times \ldots \times \mathbb{Z}_{b_{1,h_1-1}} \\ g(a) &= (a_{1,0}, \ldots, a_{1,h_1-1}) \\ &= (a \bmod b_{1,0}, \ldots, a \bmod b_{1,h_1-1}) \end{aligned}$$

et inversement

$$a = \left[\sum_{i=0}^{h_1-1} \xi_{1,i} \frac{B_1}{b_{1,i}} \right]_{B_1} = \sum_{i=0}^{h_1-1} \xi_{1,i} \frac{B_1}{b_{1,i}} - \alpha B_1 \tag{III.1}$$

où $\xi_{1,i} = \left[a_{1,i} \frac{b_{1,i}}{B_1} \right]_{b_{1,i}}$.

Concrètement, RNS fait appel à CRT pour remplacer les additions, soustractions et multiplications sur de grands entiers dans $\mathbb{Z}_{B_1}$ par les additions, soustractions et multiplications sur les plus petits canaux $\mathbb{Z}_{b_{1,0}}, \ldots, \mathbb{Z}_{b_{1,h_1-1}}$. Bien que ces opérations soient plus rapides avec RNS, les opérations telles que les divisions et les réductions modulaires sont plus difficiles à mettre en œuvre. Il faut souvent avoir recours à des extensions de base pour traiter ces opérations. Cette procédure utilise (III.1) pour étendre la représentation d'un nombre en base $\mathcal{B}_1 = \{b_{1,0}, \ldots, b_{1,h_1-1}\}$ à une autre base $\mathcal{B}_2 = \{b_{2,0}, \ldots, b_{2,h_2-1}\}$. Dans les cas où une erreur de αB_1 est tolérée, les extensions peuvent être réalisées avec

FastBConv pour approximer (III.1) de manière efficace [BI04] :

$$a_{2,i} \equiv \text{FastBConv}(a, \mathcal{B}_1) \equiv \sum_{i=0}^{h_1-1} \xi_{1,i} \frac{B_1}{b_{1,i}} \bmod b_{2,i}.$$

Lorsqu'une telle erreur ne peut être tolérée, un résidu supplémentaire $a_{\mathsf{sk}} \equiv a \bmod b_{\mathsf{sk}}$ peut être calculé. Ce résidu permet de calculer α comme suit

$$\alpha = \left[(\text{FastBConv}(a, \mathcal{B}_1) - a_{\mathsf{sk}})B_1^{-1}\right]_{b_{\mathsf{sk}}}$$

avec $|a| < \lambda B_2$, un entier λ et $b_{\mathsf{sk}} \geq 2(h_2+\lambda)$ [BEHZ16, Lemme 6]. Dans ce cas, l'extension de base peut être achevée avec FastBConvSK :

$$a_{2,i} \equiv \text{FastBConvSK}(a, \mathcal{B}_1, \alpha) \equiv \sum_{i=0}^{h_1-1} \xi_{1,i} \frac{B_1}{b_{1,i}} - \alpha B_1 \bmod b_{2,i}.$$

III.1.3 Réseaux

Soit une base donnée

$$R = \begin{bmatrix} r_{0,0} & r_{0,1} & \cdots & r_{0,n-1} \\ r_{1,0} & r_{1,1} & \cdots & r_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m-1,0} & r_{m-1,1} & \cdots & r_{m-1,n-1} \end{bmatrix}$$

le réseau $\mathcal{L}(R)$ généré par les lignes de $R \in \mathbb{R}^{m\times n}$ correspond au sous-groupe discret de $\mathbb{R}^n$ suivant :

$$\mathcal{L}(R) = \{v \,|\, \exists\, z \in \mathbb{Z}^m \text{ avec } zR = v\}.$$

Dans la suite, nous considérerons uniquement des réseaux entiers de rang plein, pour lesquels $m = n$ et $R \in \mathbb{Z}^{n\times n}$. Dans ce cas, le réseau $\mathcal{L}(R)$ induit une relation de congruence sur $\mathbb{Z}^{n\times n}$. Deux vecteurs sont dits congruents lorsque leur différence est dans $\mathcal{L}(R)$:

$$v \equiv u \bmod \mathcal{L}(R) \Leftrightarrow v - u \in \mathcal{L}(R).$$

Réduire un vecteur u modulo une base R équivaut à trouver le vecteur v satisfaisant $v \equiv u \bmod \mathcal{L}(R)$ et

$$v = zR \quad \text{avec } -1/2 \leq z_i < 1/2 \text{ pour } 0 \leq i < n.$$

III.2 Le système HyPoRes

Le système de représentation proposé dans cette section peut être considéré comme une généralisation du système PMNS [BIP05a], et est décrit dans la Définition III.2.1. Un nombre $a \in \mathbb{Z}/p\mathbb{Z}$ est représenté par un polynôme $A(X)$ avec des coefficients de norme strictement inférieure à un entier ρ qui, une fois évalué en γ, vérifie :

$$A(\gamma) \equiv a \bmod p.$$

L'entier γ satisfait $[\gamma^n]_p = \beta$ pour un petit entier β. Par conséquent, opérer sur ces polynômes modulo $X^n - \beta$ équivaut à opérer sur les entiers correspondants modulo p. Dans le cas du système PMNS, il a été prouvé que toutes les classes d'équivalence $a \in \mathbb{Z}/p\mathbb{Z}$ sont représentées dans le système dès lors que $\rho \geq \beta p^{1/n}$ [BIP05a].

Dans la suite, les chiffres sont représentés par rapport à deux bases RNS, $\mathcal{B}_1$ et $\mathcal{B}_2$, et un module b_{sk}. La nécessité de ces modules deviendra évidente en décrivant l'algorithme de multiplication HyPoRes.

Définition III.2.1 (HyPoRes)**.** Un système de représentation hybride polynomial, ou Hybrid-Polynomial Residue Number System (HyPoRes) est un sextuplet $\mathcal{H} = (p, n, \rho, \mathcal{B}_1, \mathcal{B}_2, b_{\mathsf{sk}})$. L'entier β est défini comme étant le plus petit entier qui n'est pas une puissance n-ième sur $\mathbb{Z}$, mais qui a une racine n-ième modulo p (voir Lemme III.1.1). Nous identifions cette racine avec $\gamma^n = \beta \bmod p$.

Un entier positif $0 \leq a < p$ est représenté par un polynôme à n coefficients $(a^{(0)}, \ldots, a^{(n-1)})$, où chaque coefficient $a^{(i)}$ est représenté par rapport aux deux bases RNS $\mathcal{B}_1 = \{b_{1,0}, \ldots, b_{1,h_1-1}\}$ et $\mathcal{B}_2 = \{b_{2,0}, \ldots, b_{2,h_2-1}\}$ et au module b_{sk}, satisfaisant :

$$a \equiv \sum_{i=0}^{n-1} \left[\sum_{j=0}^{h_1-1} \xi_{i,1,j} \frac{B_1}{b_{1,j}} \right]_{B_1} \gamma^i \bmod p$$

avec $\xi_{i,1,j} = \left[a_{i,1,j} \frac{b_{1,j}}{B_j}\right]_{b_{1,j}}$, $|a^{(i)}| < \rho$ et $a_{i,t,j} \equiv a^{(i)} \bmod b_{t,j}$ pour $t \in \{1, 2\}$.

Nous utilisons $a_{i,t}$ pour désigner $a^{(i)} \bmod B_t$, la lettre capitale A pour indiquer une représentation de a dans $\mathcal{H}$, A_1 pour indiquer la représentation de a dans $\mathcal{B}_1$, A_2 la représentation de a dans $\mathcal{B}_2$, a_{sk} la représentation de a modulo b_{sk} et nous définissons la norme $\|A\|_\infty = \max(|a^{(0)}|, \ldots, |a^{(n-1)}|)$.

Algorithme 10 Algorithme de multiplication modulaire

Require: $\|A\|_\infty, \|C\|_\infty < k\rho$

Require: $M' = -M^{-1} \bmod \mathcal{B}_1$

Ensure: $\|R\|_\infty < \rho$ avec $r = acB_1^{-1} \bmod p$

1: $D \leftarrow A \star C \bmod \mathcal{B}_1 \cup \mathcal{B}_2 \cup \{b_{\mathsf{sk}}\}$

2: $Q_1 \leftarrow D \star M' \bmod \mathcal{B}_1$

3: $Q_2 \leftarrow \text{FastBConv}(q, \mathcal{B}_1) \bmod \mathcal{B}_2$ # q correspond à Q_1

4: $q_{\mathsf{sk}} \leftarrow \text{FastBConv}(q, \mathcal{B}_1) \bmod b_{\mathsf{sk}}$

5: $R_2 \leftarrow \frac{D+Q_2 \star M}{B_1} \bmod \mathcal{B}_2$

6: $r_{\mathsf{sk}} \leftarrow \frac{d_{\mathsf{sk}}+q_{\mathsf{sk}} \star M}{B_1} \bmod b_{\mathsf{sk}}$

7: $\alpha \leftarrow \left[(\text{FastBConv}(r, \mathcal{B}_2) - r_{\mathsf{sk}})B_2^{-1}\right]_{b_{\mathsf{sk}}}$ # r correspond à R_2

8: $R_1 \leftarrow \text{FastBConvSK}(r, \mathcal{B}_2, \alpha) \bmod \mathcal{B}_1$

9: $R \leftarrow (R_1, R_2)$

10: **return** R

III.2.1 Algorithme de multiplication modulaire

La multiplication modulaire proposée est décrite dans l'Algorithme 10. Cet algorithme commence par calculer $D = A \star C \cong A \times C \bmod p$. Ensuite, pour réduire la taille des coefficients, un multiple d'une représentation non nulle de zéro M est ajouté à D, rendant le résultat divisible par B_1. Puisque le facteur Q est calculé modulo B_1, il est d'abord produit au sein de $\mathcal{B}_1$ et ensuite étendu à $\mathcal{B}_2$. Pour finir, $R = \frac{D+Q\star M}{B_1}$ est retourné en sortie. La division par B_1 garantit que la norme du résultat est petite. Cependant, comme cette division n'est pas possible dans $\mathcal{B}_1$, cette valeur est d'abord produite dans $\mathcal{B}_2$, puis étendue à $\mathcal{B}_1$.

Dans cet algorithme, l'opération $A \star C$ désigne le produit vecteur-matrice suivant :

$$\begin{aligned} A \star C &= AC \bmod (X^n - \beta) \\ &= \begin{bmatrix} a^{(0)} & a^{(1)} & \dots & a^{(n-1)} \end{bmatrix} \begin{bmatrix} c^{(0)} & c^{(1)} & \dots & c^{(n-1)} \\ \beta c^{(n-1)} & c^{(0)} & \dots & c^{(n-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta c^{(1)} & \beta c^{(2)} & \dots & c^{(0)} \end{bmatrix} \end{aligned} \tag{III.2}$$

où les multiplications sur les coefficients sont effectuées en RNS.

Le vecteur M utilisé dans l'Algorithme 10 correspond à une petite représentation non nulle de zéro dans $\mathcal{H}$. L'existence d'une représentation de M avec une norme inférieure à $p^{1/n}$ est garantie par le Lemme III.2.1.

Lemme III.2.1. *Une représentation non nulle de zéro de norme inférieure à $p^{1/n}$ existe dans $\mathcal{H}$.*

Preuve. Nous commençons par construire le réseau $\mathcal{L}(\Gamma)$ des représentations de zéro dans $\mathcal{H}$ en donnant Γ, l'une de ses bases,

$$\Gamma = \begin{bmatrix} p & 0 & \dots & 0 \\ -\gamma & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -\gamma^n & 0 & \dots & 1 \end{bmatrix}.$$

Chaque ligne de Γ correspond soit à p, congru à 0 modulo p, soit à $-\gamma^i + X^i$ pour $1 \leq i \leq n$, lesquels, une fois évalués en $X = \gamma$, produisent une valeur congruente à 0 modulo p. Le théorème de Minskowski [Mat02] garantit que $\mathcal{L}(\Gamma)$ contient un vecteur non nul de norme au plus $(\det\mathcal{L}(\Gamma))^{1/n} = p^{1/n}$. Ainsi, M peut être obtenu en trouvant le plus court vecteur non nul du réseau $\mathcal{L}(\Gamma)$. Bien que ce problème soit complexe en général, nous nous trouvons ici en présence de réseaux de petite dimension, qui permettent de le résoudre en peu de temps. □

L'Algorithme 10 requiert qu'un inverse de M existe dans l'anneau $\mathbb{Z}_{B_1}[X]/(X^n - \beta)$. Le Lemme III.2.2 garantit que c'est le cas lorsque $\mathcal{B}_1$ est construit à partir de nombres premiers $b_{1,i}$ qui ne divisent pas le résultant de M et $X^n - \beta$ et $M \not\equiv 0 \bmod B_1$.

Lemme III.2.2. *Si $\mathcal{B}_1 = \{b_{1,0}, \dots, b_{1,h_1-1}\}$ est tel que tous les $b_{1,i}$ sont des premiers ne divisant pas le résultant de M et $X^n - \beta$ et $M \not\equiv 0 \bmod B_1$, alors M est inversible dans $\mathbb{Z}_{B_1}[X]/(X^n - \beta)$.*

Preuve. Puisque $X^n - \beta$ est irréductible, M avec $\deg(M) < n$ et $X^n - \beta$ sont premiers entre eux. Donc, il existe $(U, V) \in \mathbb{Z}[X]^2$ tels que

$$UM + V(X^n - \beta) = r$$

où r est le résultant de M et $X^n - \beta$ et $r \neq 0$ [Buh01]. Nous trouvons l'inverse de M modulo B_1 et $X^n - \beta$ en calculant $Ur^{-1} \bmod b_{1,i}$ pour $0 \leq i \leq h_1 - 1$ et en faisant remonter le résultat dans $\mathbb{Z}_{B_1}$ avec le CRT. Le résultant r doit être inversible modulo $b_{1,i}$, c'est-à-dire premier avec $b_{1,i}$. Puisque $b_{1,i}$ est premier, il ne doit pas diviser r. □

Pour finir, le Théorème III.2.3 prouve que l'Algorithme 10 produit le résultat correct. Même si les valeurs représentées dans $\mathcal{H}$ satisfont normalement $\|A\|_\infty < \rho$, leur norme peut augmenter au cours d'additions non réduites. Par conséquent, nous supposons que

les entrées de l'Algorithme 10 ont leur norme bornée par $\|A\|_\infty < k\rho$. Une des principales conditions du Théorème III.2.3 est que $B_1 > \rho/\epsilon$ pour

$$0 < \epsilon < \frac{\rho - \beta n h_1 \|M\|_\infty}{\beta n k^2 \rho}.$$

Puisque B_1 doit être minimisé, il est crucial de choisir un ρ suffisamment grand pour que le deuxième terme de $\epsilon < 1/(\beta n k^2) - h_1\|M\|_\infty/(k^2\rho)$ soit petit, mais pas au point que ρ soit considérablement augmenté puisque $B_1 > \rho/\epsilon$. De façon empirique, nous pouvons choisir $\rho \sim 10000\|M\|_\infty/k^2$, qui fonctionne pour un grand nombre de valeurs. De plus, un choix approprié de λ peut réduire dans certains cas le nombre de modules de $\mathcal{B}_2$ de un par rapport au cas $\lambda = 1$.

Theorème III.2.3. *L'Algorithme 10 est correct (c'est-à-dire qu'il retourne* $R = ACB_1^{-1} \bmod p$ *avec* $\|R\|_\infty < \rho$ *pour* $\|A\|_\infty, \|C\|_\infty < k\rho$*) lorsque* $\mathcal{B}_1$ *est construit à partir de nombres premiers* $b_{1,i}$ *ne divisant pas le résultant de* M *et* $X^n - \beta$*,* $M \not\equiv 0 \bmod B_1$*,* $B_1 > \rho/\epsilon$*,* $B_2 > \rho/\lambda$*,* $\rho > \beta n h_1 \|M\|_\infty$*,* $b_{\mathsf{sk}} \geq 2(h_2 + \lambda)$*, pour un entier* λ*, et* B_1*,* B_2 *et* b_{sk} *deux à deux premiers entre eux, et*

$$0 < \epsilon < \frac{\rho - \beta n h_1 \|M\|_\infty}{\beta n k^2 \rho}.$$

Preuve. Comme $\mathcal{B}_1$ est construit à partir de nombres premiers ne divisant pas le résultant de M et $X^n - \beta$ et $M \not\equiv 0 \bmod B_1$, le Lemme III.2.2 garantit que M est inversible dans $\mathbb{Z}_{B_1}[X]/(X^n - \beta)$. Pour prouver que l'Algorithme 10 produit un résultat correct, nous remarquons que $D + Q \star M \equiv D - (D \star M^{-1} + \alpha B_1) \star M \equiv 0 \bmod B_1$ (pour une erreur polynomiale α qui résulte d'une extension de base inexacte) et donc $D + Q \star M$ est divisible par B_1. De plus, puisque M est une représentation de zéro modulo p, nous avons $R(\gamma) \equiv \frac{D(\gamma)+Q\star M(\gamma)}{B_1} \equiv \frac{D(\gamma)}{B_1} \equiv D(\gamma)B_1^{-1} \bmod p$.

Nous supposons que les entrées de l'Algorithme 10 satisfont $\|A\|_\infty, \|C\|_\infty < k\rho$. En outre, nous supposons que $\rho < \epsilon B_1$. Notons que comme la valeur de Q est étendue de $\mathcal{B}_1$ à $\mathcal{B}_2$ de façon inexacte, sa norme est bornée par $h_1 B_1$. Dans ce cas, la norme de R satisfait

$$\begin{aligned}
\|R\|_\infty &= \left\| \frac{A \star C + Q \star M}{B_1} \right\|_\infty \\
&< \frac{\beta n k^2 \rho^2 + \beta n h_1 B_1 \|M\|_\infty}{B_1} \\
&< \beta n \epsilon k^2 \rho + \beta n h_1 \|M\|_\infty.
\end{aligned}$$

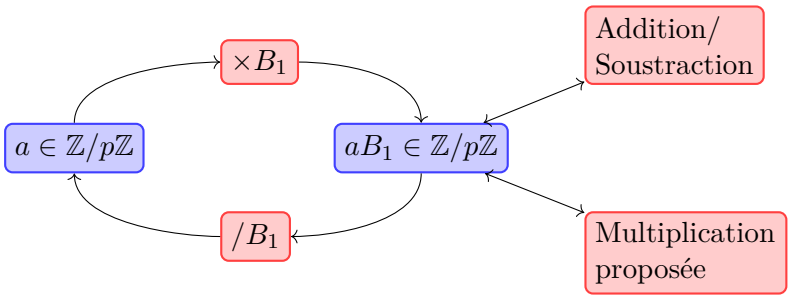


Fig. III.1 *Les valeurs dans $\mathbb{Z}/p\mathbb{Z}$ sont pré-multipliées par B_1 avant que les opérations arithmétiques proposées soient appliquées*

Puisque nous exigeons $\|R\| < \rho$, ϵ doit satisfaire :

$$0 < \epsilon < \frac{\rho - \beta n h_1 \|M\|_\infty}{\beta n k^2 \rho}.$$

Par conséquent, l'Algorithme 10 retourne des valeurs avec la norme voulue tant que $B_1 > \rho/\epsilon$ et $\rho > \beta n h_1 \|M\|_\infty$. De plus, FastBConvSK produit la valeur correcte lorsque $\rho < \lambda B_2$ et $b_{\mathsf{sk}} \geq 2(h_2 + \lambda)$. □

III.2.2 Autres opérations

Puisque les multiplications modulaires dans l'Algorithme 10 sont affectées par un facteur B_1^{-1}, il est judicieux de multiplier les valeurs par $B_1 \bmod p$ avant de les représenter dans le système HyPoRes, comme décrit à la Figure III.1. Les additions et les soustractions sont effectuées sans réduction modulaire. Bien que cela implique une possible augmentation des coefficients, nous en tenons compte lors de la conception de l'algorithme modulaire de multiplication. Dans ce cas, nous supposons que les coefficients des entrées sont bornés par $k\rho$ où ρ est la norme maximale des coefficients de la sortie. Il est clair que le format aB_1 est conservé après additions puisque $aB_1 + bB_1 = (a+b)B_1$. Ce format est également conservé après multiplications modulaires. En particulier, l'Algorithme 10 calcule $(aB_1)(bB_1)B_1^{-1} = abB_1 \bmod p$.

Une première approche pour convertir une valeur a d'une représentation binaire vers le système HyPoRes consiste à considérer le réseau généré par les vecteurs $M \star X^i$ où M est la représentation non nulle de zéro mentionnée au Lemme III.2.1 et $0 \leq i < n$. Ce réseau a pour base

$$R = \begin{bmatrix} m^{(0)} & m^{(1)} & \dots & m^{(n-1)} \\ \beta m^{(n-1)} & m^{(0)} & \dots & m^{(n-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta m^{(1)} & \beta m^{(2)} & \dots & m^{(0)} \end{bmatrix}$$

avec des vecteurs de norme au plus $\beta\|M\|_\infty$. Puisque les vecteurs $M \star X^i$ sont des représentations de 0 dans $\mathcal{H}$, le vecteur U qui résulte de la réduction de $(a, 0, \ldots, 0)$ modulo R représente a dans $\mathcal{H}$. De plus, puisque les vecteurs $M \star X^i$ ont une norme au plus $\beta\|M\|_\infty$, c'est également le cas de U. Une représentation standard HyPoRes est obtenue en réduisant les coefficients de U modulo chaque valeur dans $\mathcal{B}_1$, $\mathcal{B}_2$ et b_{sk}.

L'Algorithme 11 propose une seconde approche, plus efficace, qui repose sur le précalcul des valeurs $T[i] = [2^{i\lceil \log_2 p \rceil/n} B_1^2]_p$ dans $\mathcal{H}$ en utilisant la conversion mentionnée ci-dessus. Avec cette approche, la multiplication par B_1 est intégrée dans le processus de conversion et n'a pas besoin d'être effectuée dans une étape séparée. La valeur a à convertir est décomposée en n mots de $\lceil \log_2 p \rceil / n$ bits chacun :

$$a = \sum_{i=0}^{n-1} a[i] 2^{i\lceil \log_2 p \rceil/n}.$$

Chaque $a[i]$ peut être représenté directement dans $\mathcal{H}$ en le réduisant modulo chaque valeur dans $\mathcal{B}_1$, $\mathcal{B}_2$ et b_{sk}, et en l'associant avec la première entrée du vecteur $A[i] \in \mathcal{H}$, dont toutes les autres entrées sont égales à 0. Ensuite, $[aB_1]_p$ est calculé comme suit

$$A = \sum_{i=0}^{n-1} \text{HyPoRes-mul}(A[i], T[i]).$$

Algorithme 11 Conversion vers le système HyPoRes

Require: $a \in \mathbb{Z}/p\mathbb{Z}$ et $\mathcal{H} = (p, n, \rho, \mathcal{B}_1, \mathcal{B}_2, b_{\mathsf{sk}})$

Ensure: $A = [aB_1]_p$ dans $\mathcal{H}$

Data : $T[i] = [\beta^i B_1^2]_p$ dans $\mathcal{H}$ avec $\beta = 2^{\lceil \log_2 p \rceil/n}$, pour $i = 0, \ldots, n-1$

1: $B \leftarrow (a_{n-1}, ..., a_0)_\beta$ # décomposition de a en base β
2: $A \leftarrow (0, \ldots, 0)$
3: **for** $i = 0$ **to** $n - 1$ **do**
4: $\quad A[i] \leftarrow (B[i] \bmod \mathcal{B}_1 \cup \mathcal{B}_2 \cup \{b_{\mathsf{sk}}\}, 0, \ldots, 0)$
5: $\quad A \leftarrow A + \text{HyPoRes-mul}(A[i], T[i])$
6: **end for**
7: **return** A

III.3 État de l'art

L'optimisation des réductions modulaires s'est souvent focalisée sur des nombres premiers avec des formes particulières [BIP05b, BT16]. Une première approche [BIP05b]

produit un algorithme de multiplication en temps quadratique. Une valeur a est représentée par un polynôme A dans un anneau $\mathbb{Z}[X]/(X^n - \beta)$ tel que $A(\gamma) \equiv a \bmod p$. Les coefficients des polynômes sont représentés en un seul mot informatique, et p et γ sont choisis de telle sorte que $\gamma^n - \beta \equiv 0 \bmod p$ et 2^k soit représentable par un polynôme M avec de petits coefficients. Après avoir calculé le produit $D = A \star C$, l'amplitude des coefficients de D est réduite de façon récursive en réécrivant D sous la forme $D = D_L + D_H\, 2^k$ avec $\|D_L\| < 2^k$, puis en mettant D à jour avec la valeur $D_L + D_H \star M$, en utilisant seulement des décalages et des additions.

Une seconde approche [BT16] introduit un algorithme de multiplication en temps sous-quadratique. Un entier a est également représenté par un polynôme A dans un anneau $\mathbb{Z}[X]/(X^n - \beta)$ tel que $A(\gamma) = a \bmod p$. Cependant, les coefficients sont représentés en RNS avec deux bases d'intervalle dynamique B_1 et B_2 ; avec $\gamma = B_1$ et $p = B_1^n - \beta$. Après avoir calculé le produit $D = d^{(0)} + d^{(1)}X + \ldots + d^{(n-1)}X^{n-1} = A \star C$, la norme des coefficients de D est réduite via deux phases de propagation de retenue. Les retenues sont calculées de façon approximative avec des extensions de base. En particulier, pour $i \in \{0, \ldots, n-1\}$, la retenue $e_{i,2}$ est calculée en posant $e_{i,2} = \frac{d_{i,2} - \text{FastBConv}(d, \mathcal{B}_1)}{B_1} \bmod \mathcal{B}_2$, puis $e_{i,1}$ est calculée par une extension exacte de $\mathcal{B}_2$ à $\mathcal{B}_1$. Ensuite, $d^{(i+1)}$ est mis à jour avec $d^{(i+1)} + e^{(i)}$ si $i < n-1$, ou $d^{(0)}$ est mis à jour avec $d^{(0)} + \beta e^{(n-1)}$ si $i = n-1$. Une seconde phase de propagation de retenue est nécessaire pour obtenir des coefficients suffisamment petits. Dans ce cas, les retenues sont suffisamment petites pour qu'il suffise de calculer $e_{i,2} = \frac{d_{i,2} - \text{FastBConv}(d, \mathcal{B}_1)}{B_1}$ dans un seul module de $\mathcal{B}_2$, et copier ensuite le reste dans les modules restants de $\mathcal{B}_1$ et $\mathcal{B}_2$.

Les deux méthodes de réduction modulaire décrites ci-dessus ne s'étendent pas aux premiers utilisés actuellement par les standards ECC. Premièrement, la plupart des standards suivent l'approche de [AS99] en choisissant un p optimisé pour les systèmes de représentation binaire, c'est-à-dire $p = f(2)$ pour un polynôme f très creux. Deuxièmement, l'ECC à base d'isogénies [Feo17] est basée sur des nombres premiers de la forme $p = l_A^{e_A} l_B^{e_B} f \pm 1$, pour deux nombres premiers petits distincts l_A et l_B, deux grands exposants e_A et e_B et un petit facteur f. Ces deux cas sont incompatibles avec [BIP05b, BT16]. En revanche, [BIP05a, ABS12] conviennent pour tous les nombres premiers sous-jacents. [BIP05a] construit des systèmes polynomiaux comme dans [BIP05b] mais pour γ racine d'un polynôme E modulo p, avec $E(X) = X^n + \alpha X + \beta$, irréductible dans $\mathbb{Z}[X]$ et p quelconque. Après un produit $D = A \star C$, les coefficients de D sont réduits en réécrivant récursivement D sous la forme $D = D_L + D_H\, 2^{k-1}$ avec $\|D_L\| < 2^{k-1}$, puis en mettant à jour D avec la valeur $D_L + D'_H$, où $D'_H(\gamma) \equiv D_H(\gamma)\, 2^{k-1} \bmod p$ avec $\|D'_H\| < (|\alpha| + |\beta|)p^{1/n}$, précalculé pour chaque $D_H\, 2^{k-1}$ possible. La nature itérative de cette approche, ainsi que la nécessité de consulter des tableaux pré-calculés, la rendent plus appropriée pour les implémentations matérielles. Enfin, [ABS12] peut être vu comme un cas spécifique du

système HyPoRes avec $n = 1$, c'est-à-dire lorsque les réductions polynomiales ne jouent aucun rôle dans l'algorithme.

Bien que [BIP05a, ABS12] puissent supporter n'importe quel p, les deux conduisent à des algorithmes de multiplication modulaire en temps quadratique. En revanche, nous verrons en Section III.4 que HyPoRes atteint une complexité en temps sous-quadratique pour tout premier p. Les comparaisons dans les Sections III.4 et III.5 se concentreront sur [BT16, ABS12] puisque [BT16] atteint également une complexité en temps sous-quadratique mais pour des nombres premiers conçus sur mesure, et [ABS12] fonctionne pour tous les modules, même lorsque leur factorisation est inconnue, nous permettant d'évaluer l'impact des hypothèses que nous pouvons émettre sur les performances des systèmes obtenus.

III.4 Complexité des calculs

L'efficacité de l'Algorithme 10 peut être évaluée en nombre de multiplications modulaires en simple précision, ou Single-precision Modular Multiplications (SMMs). Nous supposons dans (III.2), que les multiplications par β peuvent être calculées soit par des décalages et des additions, puisque β est un petit entier, soit lors de la multiplication par M' et M, où β peut alors être intégré sur les M' et M précalculés. Par conséquent, (III.2) nécessite n^2 multiplications pour chaque module sur lequel il opère. Une partie des constantes nécessaires pour les extensions de base, liées par exemple à la multiplication des résidus de q par $b_{1,j}/B_1$ en préparation de l'extension de q vers $\mathcal{B}_2$, peut également être intégrée dans le pré-calcul de M' et traitée sans coût. De plus, nh_2 multiplications supplémentaires peuvent être évitées en stockant les valeurs $\xi_{i,2,j} = a_{i,2,j}b_{2,j}/B_2 \bmod b_{2,j}$ au lieu de $a_{i,2,j} \bmod b_{2,j}$ en base $\mathcal{B}_2$ pour tout a représenté dans $\mathcal{H}$. On peut conclure que le coût de l'Algorithme 10 en termes de SMMs est :

$$2n^2(h_1 + h_2 + 1) + 2nh_1h_2 + 2n(h_2 + h_1 + 1).$$

Pour parvenir à une comparaison plus juste, l'approche de [BT16] a été adaptée pour utiliser les extensions de base décrites à la Section III.1.2. Dans ce cas, la quantité de SMMs nécessaire pour calculer une multiplication modulo p, incluant les deux phases de propagation de retenue décrites à la Section III.3, est :

$$n^2(h_1 + h_2 + 1) + 2nh_1h_2 + n(4h_1 + 3h_2) + 3n - 2h_1 - h_2 - 1.$$

Sous des hypothèses similaires, une multiplication modulaire RNS classique, telle que décrite dans [ABS12], avec des bases RNS $\mathcal{B}_1$ et $\mathcal{B}_2$ avec $H_1 \sim h_1 n$ et $H_2 \sim h_2 n$ modules

aurait nécessité

$$2H_1H_2 + 4H_1 + 3H_2 + 3$$

SMMs.

Asymptotiquement, en choisissant $n \sim h_1 \sim h_2 \sim \log_2^{1/2} p$, le schéma proposé et [BT16] ont tous deux des complexités en $\mathcal{O}(\log_2^{3/2} p)$ SMMs. En comparaison, avec une approche RNS pure, nous avons $H_1 \sim H_2 \sim \log_2 p$, conduisant à une complexité en $\mathcal{O}(\log_2^2 p)$.

III.5 Résultats expérimentaux

La méthode proposée pour la multiplication modulaire est décrite en C++ [MM19]. Aussi, [BT16, ABS12] ont été implémentés pour comparaison. La multiplication purement basée sur RNS peut être vue comme une simplification de la méthode proposée lorsque $n = 1$, et de ce fait $M = p$ et γ et β ne jouent aucun rôle. Nous avons considéré les nombres premiers p_{383}, p_{448} et p_{521}, et les paramètres HyPoRes décrits dans la Table III.1. Notons que les nombres premiers p_{383}, p_{448} et p_{521} ont été utilisés pour définir respectivement les courbes elliptiques M-383, Ed448-Goldilocks et E-521 [ABPR13, Ham15]. De plus, les nombres premiers conçus pour HPR p_{384}, p'_{448} (différent de p_{448}) et p_{512} de 384, 448 et 512 bits respectivement, ont été considérés pour la mise en œuvre de [BT16]. Les bases $\mathcal{B}_1$ et $\mathcal{B}_2$ de la Table III.1, ainsi que celles choisies pour [BT16, ABS12], sont composées d'entiers de la forme $b_{i,j} = 2^{32} - c_{i,j}$ pour de petits $c_{i,j}$, permettant des réductions rapides [CP01]. En particulier, un nombre a résultant d'un produit est réécrit sous la forme $a = a_0 + 2^{32}a_1$, et la congruence $2^{32} \equiv c \bmod b_{i,j}$ est appliquée itérativement avec $a = a_0 + ca_1$ pour réduire l'amplitude de a.

La Figure III.2 présente le nombre de multiplications élémentaires requis pour l'approche HyPoRes proposée avec les paramètres décrits dans la Table III.1 ; pour une approche purement RNS avec des paramètres équivalents ($h_1 = 13$ pour p_{383} ; $h_1 = 15$ pour p_{448} et $h_1 = 17$ pour p_{521}) ; et pour HPR avec les nombres premiers p_{384} ($n = 2$ et $h_1 = 6$), p'_{448} ($n = 2$ et $h_1 = 7$) et p_{512} ($n = 4$ et $h_1 = 4$). En outre, le code décrit ci-dessus a été compilé avec gcc 4.8.5 avec les options `-Ofast` et `-march=native` et exécuté sur un processeur i7-3770K avec 8GB de mémoire principale exploité par CentOS 7.3. Aucun parallélisme n'a été exploité. Le temps de multiplication modulaire moyen pour les représentations HyPoRes, pure-RNS et HPR est indiqué sur la Figure III.3.

Les Figures III.2 et III.3 suggèrent que, bien qu'une performance similaire soit atteinte pour HyPoRes et une approche purement RNS pour le premier p_{383}, le système HyPoRes a une meilleure évolutivité lorsque la taille des premiers (en nombre de bits) augmente. Ce comportement a été prédit en Section III.4. De plus, alors que les facteurs de second ordre, tels que le nombre d'additions, limitent l'accélération obtenue pour HyPoRes par

Premiers	Paramètres
$p_{383} = 2^{383} - 187$[ABPR13]	$\gamma = 157516578651702607700441165343904620538135723687258491256181187801495721863980985340898259010860851343480102974368 9$ $n = 2$ $\beta = 3$ $m = -991151885317490685877537319994551485852631552512982631751 + 366898661104865555403938150878354627805167553995546070069 1X$ $\mathcal{B}_1 = [4294967291, 4294967279, 4294967231, 4294967197, 4294967161, 4294967111, 4294967087]$ $\mathcal{B}_2 = [4294967295, 4294967293, 4294967287, 4294967281, 4294967273, 4294967269]$ $b_{\mathsf{sk}} = 2^{32}$
$p_{448} = 2^{448} - 2^{224} - 1$[Ham15]	$\gamma = 617037013236874150081232979704524838882978450634686916049074439753925120009974044466434996308660572821900599056417811284834803617543551400$ $n = 3$ $\beta = 2$ $m = -357400072521332008886097893349417451849548435 + 458411098151774579540799965020532854365487420X - 469045874351789005827208204177461134721497131X^2$ $\mathcal{B}_1 = [4294967197, 4294967161, 4294967029, 4294966981, 4294966927, 4294966813]$ $\mathcal{B}_2 = [4294967295, 4294967293, 4294967291, 4294967287]$ $b_{\mathsf{sk}} = 2^{32}$
$p_{521} = 2^{521} - 1$[ABPR13]	$\gamma = 2394524282602951341184917229922358099404279878411878784$ $n = 3$ $\beta = 2$ $m = -1 + 119726214130147567059245861496117904970213993920593 92X^2$ $\mathcal{B}_1 = [4294967197, 4294967161, 4294967029, 4294966981, 4294966927, 4294966813]$ $\mathcal{B}_2 = [4294967295, 4294967293, 4294967291, 4294967287, 4294967281]$ $b_{\mathsf{sk}} = 2^{32}$

TABLE III.1 – Les paramètres HyPoRes utilisés pour évaluer la performance de la méthode proposée pour les nombres premiers p_{383}, p_{448} et p_{521}.

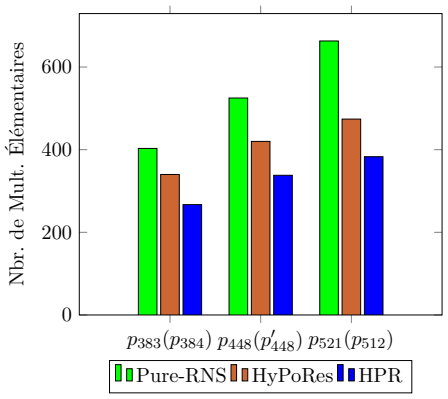


Fig. III.2 *Nombre de multiplications élémentaires requises pour les approches purement RNS et HyPoRes, ainsi que pour HPR avec des nombres premiers conçus sur mesure. Les paramètres $p_{383}, p_{448}, p_{521}$ ont été utilisés pour les multiplications modulaires RNS et HyPoRes, et les paramètres entre parenthèses pour la multiplication modulaire HPR uniquement.*

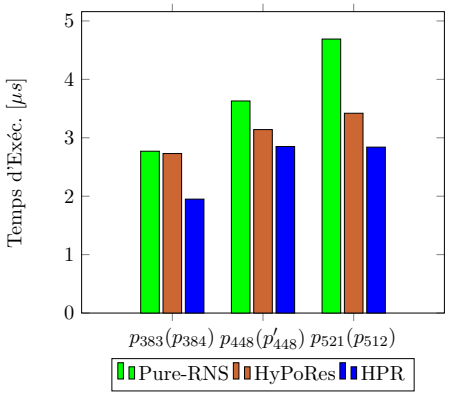


Fig. III.3 *Temps moyen d'exécution pour les approches purement RNS et HyPoRes, ainsi que pour HPR avec des nombres premiers conçus sur mesure. Les paramètres $p_{383}, p_{448}, p_{521}$ ont été utilisés pour les multiplications modulaires RNS et HyPoRes, et les paramètres entre parenthèses pour la multiplication modulaire HPR uniquement.*

rapport à une approche purement RNS pour les petits nombres premiers, en comparant les prévisions théoriques de la Figure III.2 aux résultats expérimentaux de la Figure III.3, une accélération maximale d'environ 1,4 est obtenue dans les deux cas pour p_{521}.

Alors que les Figures III.2 et III.3 montrent que la performance du HyPoRes est légèrement moins bonne que celle du HPR, le système proposé repose sur des hypothèses plus faibles (puisqu'il ne requiert pas l'utilisation de premiers conçus spécialement), le rendant plus souple et applicable dans la pratique. La Figure III.4 souligne la relation entre les hypothèses pure-RNS, HyPoRes et HPR et la performance du système qui en résulte. Puisque HPR s'appuie sur des nombres premiers d'un type particulier, cela le rend difficilement utilisable en pratique, car les nombres premiers des applications cryptographiques ont déjà été standardisés avec une forme différente. En revanche, HyPoRes peut être utilisé chaque fois que la factorisation du module sous-jacent est connue.

Alors que l'applicabilité à ECC a été démontrée ici, HyPoRes peut également être appliquée à ElGamal [ElG85b] et Rivest-Shamir-Adleman (RSA) [RSA78] pour le décryptage et la signature. En outre, bien que HyPoRes soit moins généralisable dans la pratique qu'une approche purement RNS, elle rend viable l'application de petites bases RNS de modules proches d'une puissance de deux [MZMS17], habituellement utilisés en traitement du signal, pour les applications cryptographiques.

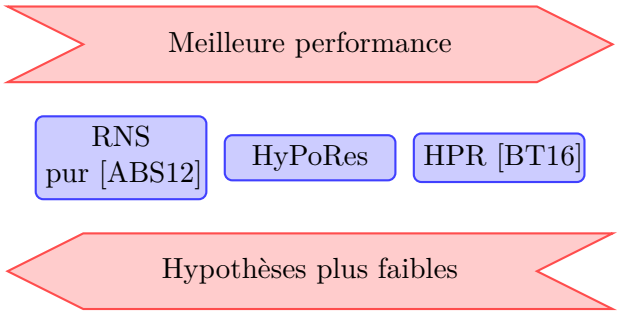


Fig. III.4 *Comparaison qualitative du système HyPoRes avec l'art associé [ABS12, BT16]*

III.6 Au-delà de la performance : Protection contre les attaques SCA

Les attaques SCAs exploitent les faiblesses dans l'implémentation des cryptosystèmes pour obtenir des informations confidentielles à partir de traces de consommation, d'analyses de temps et d'autres sources physiques d'information. Dans le contexte de ECC, les analyses simples de la consommation ou Simple Power Analyses (SPAs) sont des techniques dans lesquelles on tente de distinguer le doublement de l'addition de points sur les courbes elliptiques, en analysant directement les traces de consommation. Ces attaques peuvent être limitées en utilisant des formules où les deux opérations sont réalisées avec les mêmes opérations fondamentales [JQ01].

En revanche, les analyses différentielles de la consommation ou Differential Power Analyses (DPAs) [KJJ99] prédisent la consommation d'énergie sur la base d'une hypothèse sur un sous-ensemble de bits de l'information secrète et la mettent en corrélation avec les mesures de consommation réelle. Le but est alors de déterminer l'hypothèse la plus probable. Ces méthodes peuvent nécessiter un nombre important de traces de consommation pour extraire les données privées. Certaines techniques [Wal01] utilisent des corrélations internes pour réduire le nombre de traces nécessaires à une attaque réussie. Une seule trace peut parfois suffire.

La résistance à ce type d'attaques peut être améliorée par message blinding, où la représentation des valeurs sur lesquelles on opère est randomisée, pour empêcher toute prédiction sur la consommation d'énergie. Premièrement, la résistance aux attaques DPAs [KJJ99] peut être obtenue en randomisant la représentation des valeurs au début de la multiplication de points sur la courbe elliptique. Deuxièmement, si l'on souhaite protéger une implémentation contre [Wal01], les valeurs doivent être randomisées durant la multiplication des points. Ces deux approches sont abordées au Chapitre 2.

III.6.1 Généralisation du polynôme de réduction pour HyPoRes

Dans la suite, la Définition III.2.1 est assouplie pour autoriser plusieurs représentations distinctes de la même valeur, et ainsi permettre la randomisation par un choix arbitraire de l'une d'elles. Au lieu de choisir γ comme racine de $X^n - \beta$ modulo p, γ est défini comme racine de $E(X) = e^{(0)} + \ldots + e^{(n-1)}X^{n-1} + X^n$ modulo p, où E est un polynôme irréductible dans $\mathbb{Z}[X]$, avec de petits coefficients. Dans ce cas, le Lemme III.2.1 est toujours applicable, car indépendant du E sous-jacent, assurant l'existence d'une petite représentation non nulle de zéro M. De plus, la multiplication de A par X est obtenue avec la multiplication vecteur-matrice suivante :

$$A \star X = \begin{bmatrix} a^{(0)} & a^{(1)} & \ldots & a^{(n-1)} \end{bmatrix} \times \underbrace{\begin{bmatrix} 0 & 1 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & 0 \\ -e^{(0)} & -e^{(1)} & -e^{(2)} & \ldots & -e^{(n-2)} & -e^{(n-1)} \end{bmatrix}}_{D}. \tag{III.3}$$

Ainsi, (III.2) peut être généralisé à

$$A \star C = AC \bmod E = \begin{bmatrix} a^{(0)} & a^{(1)} & \ldots & a^{(n-1)} \end{bmatrix} \begin{bmatrix} C \star X^0 \\ C \star X^1 \\ \vdots \\ C \star X^{n-1} \end{bmatrix}$$

où la multiplication par X^i est réalisée en multipliant C par la puissance i-ième de D dans (III.3). Puisque E a de petits coefficients, la multiplication par D^i peut être implémentée uniquement avec des décalages et des additions. Par conséquent, la complexité de l'Algorithme 10 est conservée pour la nouvelle définition de $\star$.

Les paramètres d'une courbe elliptique peuvent être pré-calculés dans plusieurs HyPoRes, chacun associé à un polynôme de réduction E différent. La protection contre DPA peut être assurée en sélectionnant un E aléatoire au début de la multiplication des points. Dans ce cas, une conversion du système de représentation binaire vers un HyPoRes général est nécessaire. Afin d'empêcher des attaques comme [Wal01], des conversions entre HyPoRes ayant chacun un polynôme E différent sont nécessaires tout au long de la multiplication de points sur la courbe elliptique. Nous considérons à présent ces deux types de conversions. La première concerne la conversion d'une valeur binaire vers un HyPoRes

avec un E d'une forme quelconque. Dans ce cas, la stratégie proposée à la Section III.2.2 est toujours applicable en remplaçant R par

$$R = \begin{bmatrix} M \star X^0 \\ M \star X^1 \\ \vdots \\ M \star X^{n-1} \end{bmatrix}.$$

Le second type concerne les conversions d'un HyPoRes associé à un polynôme E, $\mathcal{H}_E$, vers un autre HyPoRes associé à E', $\mathcal{H}_{E'}$. En remarquant que la représentation de $[aB_1]_p$ dans $\mathcal{H}_E$ satisfait

$$A \equiv \sum_{i=0}^{n-1} A[i]\gamma^i \bmod p,$$

la conversion vers $\mathcal{H}_{E'}$ est réalisée comme suit

$$A' = A[0] + \sum_{i=1}^{n-1} \text{HyPoRes-mul}_{E'}(A[i], T_{E\to E'}[i])$$

où $T_{E\to E'}[i]$ est une représentation de $\gamma^i B_1 \bmod p$ dans $\mathcal{H}_{E'}$.

III.7 Conclusion

Alors que le système HPR a rendu les algorithmes de multiplication avec une complexité en temps sous-quadratique viables pour ECC, la nécessité d'utiliser des nombres premiers d'une forme spéciale le rend peu pratique. Le système HyPoRes proposé dans le présent chapitre repose sur un affaiblissement des hypothèses sous-jacentes. Non seulement il atteint une complexité en temps sous-quadratique similaire, mais il supporte également n'importe quelle valeur première, ce qui le rend compatible avec les courbes elliptiques standardisées. Il en résulte un ralentissement d'au plus 1,4 lorsque HyPoRes est comparé à HPR pour les nombres premiers conçus pour HPR, mais une accélération pouvant atteindre 1,4 lorsqu'il est comparé aux approches basées sur RNS pour des nombres premiers des standards ECC. Les implications du système HyPoRes sont multiples. Bien qu'il soit moins applicable que les approches purement RNS, il rend l'application de petites bases RNS de modules proches d'une puissance de deux, habituellement utilisées en traitement du signal, viable pour les applications cryptographiques. De plus, puisqu'il réduit la complexité des extensions de base par rapport à une approche purement RNS, HyPoRes se prête mieux au parallélisme à une plus petite échelle. Enfin, via une généralisation du polynôme de réduction, des représentations redondantes sont introduites, procurant une protection contre les attaques SCA. Ce travail a mené à une publication à la conférence du

$26^{ème}$ IEEE International Symposium on Computer Arithmetic (ARITH-26) [MMBS19].

Chapitre IV

Calculer les solutions réelles des systèmes polynomiaux flous

Sommaire

IV.1 Introduction

Modéliser des problèmes avec des données incertaines trouve d'importantes applications en ingénierie, en économie et en sciences sociales [ATT94, Lod07]. Comme les fonctions floues impliquées dans les équations peuvent être approximées par des polynômes flous [Liu02, AA06], le problème de la résolution des systèmes polynomiaux flous est d'une importance capitale et a motivé de nombreuses études, parmi lesquelles [BQ90, BE97, BFH02, KAV05].

Certains problèmes sont modélisés par un système de fonctions continues à valeurs floues définies sur $\mathbb{R}^n$ (voir [LS03]) ; Liu [Liu02] propose une interpolation polynomiale fournissant une interface entre la résolution de ces problèmes et la résolution de systèmes polynomiaux flous. Pour cette raison, plusieurs auteurs se sont intéressés à la recherche de solutions réelles d'équations polynomiales à coefficients flous. Notons que leur travail, et le nôtre également, est basé exclusivement sur le principe d'extension de Zadeh. Les méthodes de résolution étaient initialement basées sur des techniques locales, d'abord pour un polynôme univarié [AO06, Ami08, Rou07] et par la suite pour des systèmes multivariés [AOM08, AJMAH11]. Abbasbandy et al.[AOM08] étudient en particulier des systèmes de la forme

$$\begin{cases} \widetilde{a}_{1,1}xy + \widetilde{a}_{1,2}x^2y^2 + \cdots + \widetilde{a}_{1,d}x^dy^d &= \widetilde{a}_{1,0} \\ \widetilde{a}_{2,1}xy + \widetilde{a}_{2,2}x^2y^2 + \cdots + \widetilde{a}_{2,d}x^dy^d &= \widetilde{a}_{2,0} \end{cases}$$

et présentent plusieurs systèmes issus d'applications comme la localisation croisée de surfaces quadratiques, ou encore en économie. Dans la Section IV.6.2 nous détaillons notre méthode sur deux d'entre eux en guise d'illustration.

Récemment, une approche globale utilisant des techniques algébriques classiques a été développée [MBR13, FRBM15, BBRV16, FPA19]. En effet, malgré un nom qui peut prêter à confusion, les nombres flous bénéficient d'une définition parfaitement formelle. Nous réexaminons cette approche et nous la renforçons considérablement. Dans le passé, les approches locales et globales se concentraient sur les nombres flous dits *triangulaires*. Les résultats présentés ici considèrent plus généralement les coefficients flous L-R symétriques tels que, par exemple, les nombres flous dits quadratiques.

Dans un système algébrique flou, les coefficients flous (issus des expériences) sont généralement donnés sous une représentation appelée "tuple". Bien que la représentation en tuple soit formelle, elle ne peut pas être gérée par les méthodes algébriques habituelles (bases de Gröbner [BW93], décomposition triangulaire [AMM99], représentation univariée rationnelle [Rou99], ...) pour résoudre le système.

Néanmoins, la représentation tuple de tout nombre flou L-R est transformable en une autre représentation dite "paramétrique", où les coefficients ne sont plus flous mais réels.

Nous donnons l'expression de la représentation paramétrique (voir Proposition IV.2.2).

Dans le présent chapitre, nous considérons un système flou (S) formé de s équations de k variables avec des coefficients flous L-R symétriques. Nous cherchons ses vraies solutions. Obtenir directement l'ensemble des solutions réelles de (S) pose un problème de signes puisque les coefficients sont flous. Nous surmontons ce problème en procédant en deux étapes. Premièrement, sans utiliser l'hypothèse de symétrie sur les coefficients, nous appliquons notre Théorème IV.3.4 pour trouver les solutions réelles positives de (S) qui sont celles de sa *transformation réelle* $\mathcal{T}(S)$, un nouveau système algébrique formé par $3\,s$ équations de k variables avec des coefficients réels. Nous comparons la transformation réelle avec les résultats antérieurs pour des systèmes limités à des coefficients triangulaires flous. Deuxièmement, nous introduisons un outil technique appelé *systèmes induits* ; il permet de calculer l'ensemble des solutions réelles de (S) à partir des solutions positives de 2^k systèmes flous induits via notre Théorème principal IV.4.1. En appliquant ces résultats et en réduisant le nombre de systèmes induits à résoudre, nous proposons un algorithme parallélisable optimisé appelé `SolveFuzzySystem` ou `SFS`.

Notons que dans la Section IV.3.6, par le même raisonnement, nous obtenons la transformation réelle d'un système polynomial avec des coefficients flous L-R plus généraux appelés trapézoïdaux.

La Section IV.2 présente les nombres flous, les nombres flous spéciaux L-R et les transformations requises sur les représentations. La Section IV.3 est consacrée aux transformations réelles et aux équations induites formant les systèmes induits. Nous montrons que, dans le cas flou triangulaire, la transformation réelle est un système équivalent au système appelé forme tranchée collectée telle qu'il est calculé par des méthodes antérieures. L'extension des résultats au cas trapézoïdal clôt la section. La Section IV.4 applique les résultats de la Section IV.3 à un système polynomial flou. Il établit le théorème principal IV.4.1 qui conduit à un algorithme de base appelé `BA-SFS`. Dans la Section IV.5, ce dernier est optimisé dans l'Algorithme `SFS`, puis parallélisé. L'implantation de l'Algorithme `SFS` dans le package `Fuzzy` en `SageMath` (voir [Mar17]) est brièvement présenté dans Section IV.6, avec des exemples illustrant `SFS` et sa parallélisation.

IV.2 Nombres flous

Nous présentons ci-dessous les généralités sur la théorie des ensembles et des nombres flous, la représentation paramétrique de tout nombre flou, puis les nombres flous spéciaux L-R, qui peuvent être représentés par un tuple. Pour aller plus loin, le lecteur pourra s'intéresser à [DKMP00]. Nous donnons les formules exprimant la représentation paramétrique d'un nombre flou L-R à partir de sa représentation en tuple dans le cas où le nombre a un support borné et des fonctions de dispersion bijectives (Proposition IV.2.2).

Ces formules constituent la pierre angulaire de notre méthode algébrique pour déterminer les solutions dans le corps de réels $\mathbb{R}$ de systèmes polynomiaux flous.

IV.2.1 Généralités

Les nombres flous se définissent à partir de la notion d'*ensemble flou*. Dans le cadre classique, l'appartenance à un sous-ensemble E de l'ensemble universel X est booléenne. La fonction caractéristique de E est à valeurs dans $\{0, 1\}$, signifiant si l'élément appartient ou pas à E. La théorie des ensembles flous généralise ce concept classique. Ainsi que l'a décrit Zadeh dans son article original [Zad65], un ensemble flou $\widetilde{E}$ est une classe d'objets avec un continuum de degrés d'appartenance. Un élément x appartient à l'ensemble flou $\widetilde{E}$ avec un certain degré de validité représenté par une fonction à valeurs comprises entre 0 et 1.

Définition IV.2.1. *Un* ensemble flou $\tilde{E}$ *est une paire formée d'un sous-ensemble* E *de* X *et de sa* fonction d'appartenance, $\mu_{\tilde{E}} : X \to [0, 1]$.
Pour chaque élément x *de* X, *la valeur* $\mu_{\widetilde{E}}(x)$ *est appelé le* degré d'appartenance *de* x *à* $\widetilde{E}$.

Le support $\mathrm{Supp}(f)$ d'une fonction f définie sur un ensemble A est l'ensemble des $a \in A$ tels que $f(a) \neq 0$. Le support de $\widetilde{E}$ se définit comme celui de sa fonction d'appartenance : $\mathrm{Supp}(\widetilde{E}) := \mathrm{Supp}(\mu_{\widetilde{E}})$. Les nombres flous sont définis à partir des ensembles flous à l'aide de la notion de r-coupe, qui est aussi sous -jacente à la représentation paramétrique qui sera introduite dans la section IV.2.2.

Définition IV.2.2. *Soit* $\widetilde{E}$ *un ensemble flou et* r *un réel dans* $]0, 1]$. *La* r-coupe *de* $\widetilde{E}$ *est l'ensemble suivant*

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

La 0*-coupe* $\widetilde{E}_0$ *est la fermeture de* $\mathrm{Supp}(\widetilde{E})$. *Un ensemble flou est dit* convexe *si toutes ses* r*-coupes le sont, où* $r \in [0, 1]$.

Quelques exemples de r-coupes apparaissent pour un support fini comme infini dans les figures IV.1 à IV.3. Ils représentent des ensembles flous particuliers de $\mathbb{R}$ définis ci-après :

Définition IV.2.3. *Soit* n *fixé dans* $\mathbb{R}$. *Un* nombre flou $\widetilde{n}$ *est un ensemble flou convexe de réels dont la fonction d'appartenance* $\mu_{\widetilde{n}}$ *de* $\mathbb{R}$ *dans* $[0, 1]$ *est continue et vérifie* $\mu_{\widetilde{n}}^{-1}(\{1\}) = \{n\}$ *; i.e.* n *est le seul réel qui possède* 1 *comme degré d'appartenance. On dit que* n *est le* mode *du nombre flou* $\widetilde{n}$.

D'après la définition d'un nombre flou $\widetilde{n}$, le degré d'appartenance à $\widetilde{n}$ d'un réel x augmente quand x se rapproche du mode n. Deux nombres flous sont égaux s'ils ont les mêmes fonctions d'appartenance (voir [Zad65]).

Remarque IV.2.1. *La littérature fait état de définitions plus larges que celle donnée ci-dessus en intégrant les nombres flous dits* trapézoïdaux*, c'est-à-dire ceux pour lesquels le degré d'appartenance est égal à* 1 *sur un intervalle de X contenant le mode. La définition IV.2.3 les exclut par souci de clarté de notre étude. Notons que la plupart des applications se situent dans le cadre non trapézoïdal. Cependant, une fois établie notre analyse, nous montrerons qu'elle s'étend simplement au cas trapézoïdal (cf section IV.3.6) .*

Fixons $\widetilde{n}$ 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. On distingue suivant la nature des fonctions $\mu_{\widetilde{n}-}$ et $\mu_{\widetilde{n}+}$ certaines familles de nombres flous. Leur dénomination dérive directement de la nature de ces restrictions. La famille la plus populaire est celle des nombres flous dits *triangulaires* pour lesquels, sur $\mathrm{Supp}(\widetilde{n})$, les fonctions $\mu_{\widetilde{n}-}$ et $\mu_{\widetilde{n}+}$ sont linéaires. De façon similaire on parle de nombres flous *gaussiens*, *quadratiques*,... On notera que les deux restrictions ne sont pas forcément de même nature. Ainsi un nombre flou triangulaire-quadratique désigne un nombre flou $\widetilde{n}$ pour lequel $\mu_{\widetilde{n}-}$ est linéaire sur $\mathrm{Supp}(\widetilde{n}) \cap]-\infty, n]$ et $\mu_{\widetilde{n}+}$ est quadratique sur $\mathrm{Supp}(\widetilde{n}) \cap [n, +\infty[$.

La figure IV.1 représente un nombre flou gaussien $\widetilde{3}$, i.e. de mode $n = 3$. Le support est infini et la 0-coupe $\widetilde{E}_0$ de $\widetilde{E} = \widetilde{n}$ est égale à $\mathbb{R}$ tout entier.

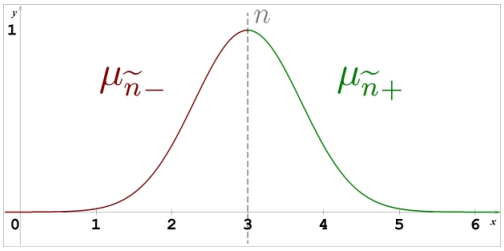


Fig. IV.1 *Support infini : nombre flou gaussien $\tilde{3}$ de mode $n = 3$: $\widetilde{E}_0 = \mathbb{R}$.*

IV.2.2 Représentation paramétrique

La *représentation paramétrique* introduite en 1986 par R. Goetschel et W. Voxman [GV86] leur permet de plonger l'ensemble des nombres flous trapézoïdaux dans un espace vectoriel topologique. La définition suivante est une adaptation pour les nombres flous non trapézoïdaux :

Définition IV.2.4. *La forme paramétrique d'un nombre flou $\widetilde{n}$ est une paire ordonnée $[\underline{n}, \overline{n}]$ de fonctions de l'intervalle réel $[0, 1]$ dans $\mathbb{R}$ qui satisfont les conditions suivantes :*

(i) $\overline{n}$ est une fonction continue bornée à gauche non croissante sur $[0, 1]$,

(ii) $\underline{n}$ est une fonction continue bornée à gauche non décroissante sur $[0, 1]$,

(iii) $\underline{n}(1) = \overline{n}(1) = n$.

Le nombre flou $\widetilde{n}$ défini par les fonctions $\underline{n}$ et $\overline{n}$ a pour fonction d'appartenance $\mu_{\widetilde{n}} : \mathbb{RR} \to [0,1]$ telle que $\mu_{\widetilde{n}}(x) = \sup\{r \mid \underline{n}(r) \leq x \leq \overline{n}(r)\}$; pour $r \in (0,1]$, la r-coupe de $\widetilde{n}$ est $[\underline{n}(r),\ \overline{n}(r)]$.

Une addition et une multiplication scalaire définies par le principe d'extension de Zadeh structurent l'ensemble des nombres flous ; Stefanini et Sorini [SS09] rappellent qu'ils sont exprimés de manière équivalente avec la représentation paramétrique, comme décrit dans le lemme suivant :

Lemme IV.2.1. *Soit $q \in \mathbb{R}$ et $\widetilde{m} = [\underline{m}, \overline{m}]$ et $\widetilde{n} = [\underline{n}, \overline{n}]$ deux nombre flous. Alors*

1. *$\widetilde{m} = \widetilde{n}$ si et seulement si $\underline{m}(r) = \underline{n}(r)$ et $\overline{m}(r) = \overline{n}(r)$ pour tout réel $r \in [0,1]$,*

2. $\widetilde{m} + \widetilde{n} = [\underline{m} + \underline{n},\ \overline{m} + \overline{n}]$,

3. $$q \cdot \widetilde{n} = \begin{cases} [q \cdot \underline{n},\ q \cdot \overline{n}] & \text{si } q \geq 0, \\ [q \cdot \overline{n},\ q \cdot \underline{n}] & \text{si } q \leq 0 \end{cases}$$

où, pour toute fonction f de $\mathbb{R}$ dans $\mathbb{R}$, le produit $g = q \cdot f$ représente la fonction définie par $g(r) = qf(r)$ pour tout $r \in \mathbb{R}$.

IV.2.3 Nombres flous L-R et représentation en tuple

Le célèbre modèle LR a été introduit par D. Dubois et H. Prade en 1977 dans [DP78]. Il considère des familles de nombres flous spéciaux dans lesquels les opérations arithmétiques sont internes et s'expriment très simplement. À l'intérieur d'une famille donnée, les nombres flous sont représentés par une représentation tuple rappelée dans la définition IV.2.5 de cette section et basée sur les *fonctions de dispersion.*

Une fonction H définie sur l'intervalle réel $[0, +\infty[$ à valeurs dans l'intervalle réel fermé $[0,1]$ est appelé *fonction de dispersion* si $H(0) = 1$, $H(1) = 0$ et que H est continue et décroissante sur son domaine de définition.

Définition IV.2.5. *Soient L et R deux fonctions de dispersion. Un nombre flou $\widetilde{n}$ à support borné est dit de* type *L-R s'il existe deux nombres réels positifs α et β tels que la*

fonction d'appartenance $\mu_{\widetilde{n}}$ *de* $\widetilde{n}$ *est de la forme suivante :*

$$\mu_{\widetilde{n}}(x) = \begin{cases} L\left(\frac{n-x}{\alpha}\right) & pour \quad n-\alpha \leq x < n \quad si \quad \alpha \neq 0 \\ 1 & pour \quad x = n \\ R\left(\frac{x-n}{\beta}\right) & pour \quad n < x \leq n+\beta \quad si \quad \beta \neq 0 \\ 0 & pour \quad x \in]-\infty, n-\alpha[\cup]n+\beta, +\infty[. \end{cases}$$

Le triplet (n, α, β) *est appelé la* représentation en tuple *du nombre flou L-R* $\widetilde{n}$*. Les nombres réels* α *et* β *sont respectivement appelés la* dispersion à gauche *et la* dispersion à droite *de* $\widetilde{n}$*.*

Notons que le nombre réel n peut être identifié au nombre flou $\widetilde{n}$ avec $\alpha = \beta = 0$ et $Supp(\widetilde{n}) = \{n\}$.

Nous notons par $\mathfrak{F}(L, R)$ la famille des nombre flous de type L-R. Les fonctions L et R sont respectivement appelées *fonction de dispersion à gauche* et *fonction de dispersion à droite* de la famille $\mathfrak{F}(L, R)$, et par extension les fonctions de dispersion de $\widetilde{n}$ lui-même.

En particulier, pour un nombre flou triangulaire, le support de $\widetilde{n}$ (ou de façon équivalente de $\mu_{\widetilde{n}}$) est donc $]n-\alpha, n+\beta[$ quand $\alpha \neq 0$ et $\beta \neq 0$, $[n, n+\beta[$ quand $\alpha = 0$ et $\beta \neq 0$, $]n-\alpha, n]$ quand $\alpha \neq 0$ et $\beta = 0$, et le singleton $\{n\}$ quand $\alpha = \beta = 0$.

La famille triangulaire, formée des nombres triangulaires flous, est la famille $\mathfrak{F}(L, R)$ où les fonctions de dispersion L et R sont toutes deux linéaires. Dans ce cas, comme $L(0) = R(0) = 1$ et $L(1) = R(1) = 0$, nous avons $L = R$ et cette fonction de dispersion est égale à la fonction F définie par $F(x) = -x + 1$.

Exemple IV.2.1. Soit $\widetilde{3}$ un nombre triangulaire flou dans la représentation en tuple $(3, 2, 2)$. Les figures IV.2 et IV.3 décrivent respectivement la coupe-1/2 $\widetilde{E}_{1/2} = [2, 4]$ et la coupe-0 $\widetilde{E}_0 = [3-\alpha, 3+\beta] = [1, 5]$ où $\widetilde{E} = \widetilde{3}$.

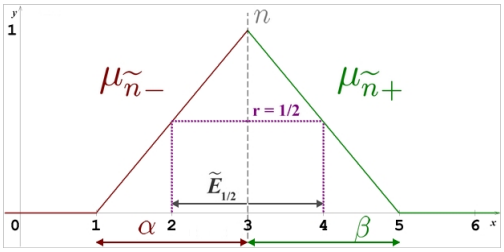


Fig. IV.2 *Support borné* $]1, 5[$ *: coupe-*$1/2$ $\widetilde{E}_{1/2} = [2, 4]$ *du nombre flou triangulaire* $\widetilde{E} = \widetilde{3} = (3, 2, 2)$.

Au sein d'une famille donnée $\mathfrak{F}(L, R)$, un tuple (n, α, β) représente un unique élément $\widetilde{n}$. L'addition de nombres flous est une loi interne à $\mathfrak{F}(L, R)$. La représentation tuple de

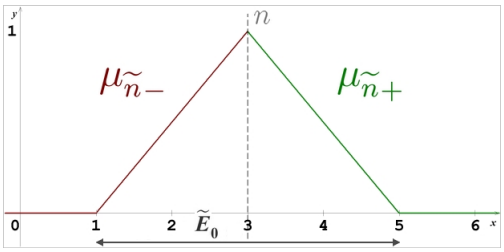


Fig. IV.3 *Support borné* $]1,5[$ *: coupe-0* $\widetilde{E}_0 = [1,5]$ *du nombre flou triangulaire* $\widetilde{E} = \widetilde{3} = (3,2,2)$.

la somme de deux nombres flous Ł-R $\widetilde{n} = (n,\alpha,\beta)$ et $\widetilde{n'} = (n',\alpha',\beta')$ est donnée par :

$$(n,\alpha,\beta) + (n',\alpha',\beta') = (n+n', \alpha+\alpha', \beta+\beta') \ .$$

La famille $(\mathfrak{F}(L,R),+)$ est un monoïde abélien avec le tuple identité $0 = (0,0,0)$.

Le produit approximé "$\cdot$", tel que $(\mathfrak{F}(L,R),\cdot)$ est un monoïde abélien avec le tuple identité $1 = (1,0,0)$, est distributif par rapport à l'addition. Du fait que nous traitons uniquement d'équations à indéterminées réelles, notre étude ne fera intervenir que des produits de la forme $q \cdot \widetilde{n}$, où $q \in \mathbb{R}$. Dans ce cas, le produit décrit par Dubois et Prade devient exact. Pour $\widetilde{n} \in \mathfrak{F}(L,R)$ le produit scalaire $q \cdot \widetilde{n}$ est dans $\mathfrak{F}(L,R)$ si le réel q est positif alors qu'il est dans $\mathfrak{F}(R,L)$ si q est négatif. Sa représentation tuple est donnée par :

$$q \ \cdot \ (n,\alpha,\beta) = \begin{cases} (q\,n, q\,\alpha, q\,\beta) \in \mathfrak{F}(L,R) & \text{si } q \geq 0 \\ (q\,n, -q\,\beta, -q\,\alpha) \in \mathfrak{F}(R,L) & \text{si } q \leq 0 \quad . \end{cases} \tag{IV.1}$$

Le produit scalaire est une loi interne de $\mathfrak{F}(L,R)$ uniquement lorsque $L = R$: en effet, pour avoir $\mathfrak{F}(L,R) = \mathfrak{F}(R,L)$, il est nécessaire que $L(x) = R(\frac{x}{c})$ avec $c > 0$; alors $L = R$ car $0 = L(1) = R(1)$ et R est strictement monotone.

Les éléments de $\mathfrak{F}(L,L)$ seront appelés des nombres flous *symétriques*.

Notons que l'inversion des dispersions les maintient positifs lorsque $q < 0$: $-q\,\beta$ et $-q\,\alpha$ sont respectivement la dispersion à gauche et la dispersion à droite de $q \cdot (n,\alpha,\beta)$. En particulier, nous avons

$$-\widetilde{n} = -1 \cdot (n,\alpha,\beta) = (-n,\beta,\alpha) \quad . \tag{IV.2}$$

Remarque IV.2.2. Lorsque tous les calculs impliquent des produits $q \cdot (n\alpha,\beta)$ où $q \geq 0$, il n'est pas nécessaire d'avoir $L = R$.

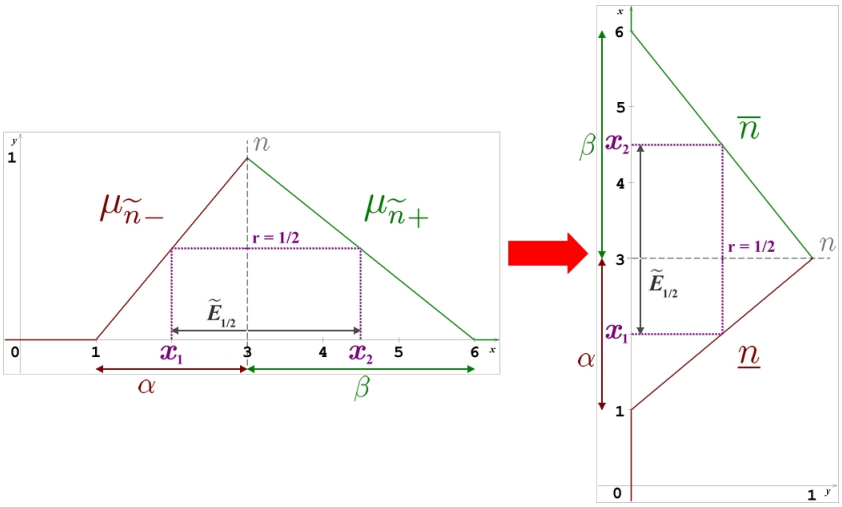


Fig. IV.4 *Graphe des fonctions* $\underline{3}$ *et* $\overline{3}$ *à partir du graphe d'une fonction d'appartenance linéaire.*

IV.2.4 Du tuple à la représentation paramétrique

Notre méthode de résolution s'appliquera à des équations polynomiales dont les coefficients sont des nombres flous d'une même famille $\mathfrak{F}(L,R)$ satisfaisant la condition suffisante que les fonctions de dispersion L et R soient bijectives. Elle implique de réécrire algébriquement chaque coefficient flou $\widetilde{n} \in \mathfrak{F}(L,R)$ de la représentation tuple (n,α,β) dans la représentation paramétrique. Le changement de représentation est donné par les formules de Proposition IV.2.2 ci-dessous.

La représentation paramétrique de $\widetilde{n}$ est fortement liée à ses r-coupes $\widetilde{n}_r$ puisque les fonctions $\underline{n}$ et $\overline{n}$ définies par

$$\underline{n}(r) = \inf_{r\in[0,1]} \widetilde{n}_r \quad \text{et} \quad \overline{n}(r) = \sup_{r\in[0,1]} \widetilde{n}_r$$

satisfont les conditions de la Définition IV.2.4. Cette relation apparaît graphiquement sur la Figure IV.4 où $x_1 = \underline{n}(1/2)$ et $x_2 = \overline{n}(1/2)$ pour un nombre flou triangulaire $\widetilde{3} = (3,2,3)$.

La représentation graphique des fonctions $\underline{n}$ et $\overline{n}$ est obtenue par une rotation plane du graphe de la fonction d'appartenance suivie d'une symétrie verticale. Cette transformation est illustrée dans la figure IV.4 et dans la Figure IV.5, pour un nombre flou quadratique $\widetilde{3} = (3,2,3)$, dans la figure IV.5 . Formellement elle est décrite par les formules suivantes.

Proposition IV.2.2. *Soit* $\widetilde{n} = (n,\alpha,\beta) \in \mathfrak{F}(L,R)$ *où* L *et* R *sont des fonctions de dispersion bijectives. Alors la représentation paramétrique* $[\underline{n},\overline{n}]$ *de* $\widetilde{n}$ *satisfait les formules*

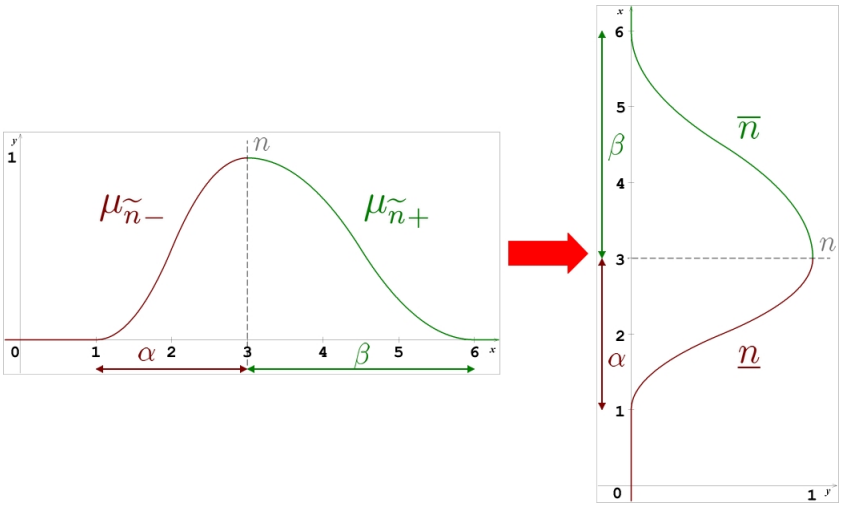


Fig. IV.5 *Graphe des fonctions* $\underline{3}$ *et* $\overline{3}$ *à partir du graphe d'une fonction d'appartenance quadratique.*

suivantes :

$$\begin{cases} \underline{n}(r) = n - \alpha\, L^{-1}(r) \\ \overline{n}(r) = n + \beta\, R^{-1}(r) \end{cases} . \tag{IV.3}$$

En particulier, lorsque le nombre flou $\widetilde{n}$ *est triangulaire, nous avons :*

$$\underline{n}(r) = \alpha\, r + n - \alpha \quad \textit{et} \quad \overline{n}(r) = -\beta\, r + n + \beta \quad . \tag{IV.4}$$

Preuve. Pour le nombre réel $r \in [0,1]$, la Définition IV.2.5 implique $r = L\left(\frac{n-\underline{n}(r)}{\alpha}\right) = R\left(\frac{\overline{n}(r)-n}{\beta}\right)$. Comme L et R sont bijectives, $\underline{n}(r)$ et $\overline{n}(r)$ satisfont l'Identité (IV.3) de la proposition.
Dans le cas triangulaire, on obtient la formule (IV.4) car $L = R = F$ où $F(x) = 1 - x$ est bijective avec $F^{-1} = F$. □

Dans la suite du chapitre, les différents nombres flous qui apparaissent dans une équation ou un système sont considérés comme appartenant à une même famille $\mathfrak{F}(L,R)$.

IV.2.5 Précisions sur l'état de l'art

La question du sens d'une solution d'un système polynomial à coefficients flous s'est rapidement posée. Le principe d'extension introduit par Zadeh en 1975 permet d'étendre une fonction réelle continue à une fonction acceptant comme arguments des ensembles flous. À partir d'une application f d'un ensemble X vers un ensemble Y et d'un sous-ensemble flou A défini sur X, ce principe stipule que l'image par f de A, $f(A)$, est un sous-ensemble flou B de Y dont la fonction d'appartenance est définie par $\mu_B(y) = \sup_{x \mid y=f(x)} \mu_A(x)$. En

1990, Buckley et Qu [BQ90] concluent que trop souvent, l'approche fondée sur le principe d'extension est trop restrictive, et qu'un nouveau concept de solution est nécessaire afin qu'une équation quadratique à une variable ait toujours une solution. Considérer l'inclusion ou l'intersection non vide plutôt que l'égalité des fonctions d'appartenance a conduit à une formulation non unique. La question se pose même pour les systèmes linéaires : Friedman et al. [FMK98] distinguent ainsi les solutions floues fortes et faibles, et différents algorithmes présentés dans des articles ultérieurs peuvent retourner des solutions au sens ambigu. Dans la mesure où les systèmes linéaires avec intervalle peuvent être considérés comme un cas particulier de systèmes linéaires flous, les techniques d'analyse par intervalle ont été exploitées : Muzzioli et Reynaerts [MR06] étudient ainsi l'existence d'un élément homologue dans le cas linéaire flou, pour différentes notions de solutions dans la résolution de systèmes linéaires avec intervalles. Cependant, les méthodes basées sur ces techniques se heurtent à des problèmes dus aux intervalles inadéquats qui peuvent apparaître. En 2015, Lodwick et Dubois [LD15] se penchent sur la sémantique et les liens unissant systèmes linéaires flous et systèmes linéaires à intervalles, dans une approche unifiée qui clarifie les résultats anormaux ou inconsistants des travaux précédents.

Nous réexaminons dans la suite l'approche globale utilisant des techniques algébriques classiques [MBR13, FRBM15, BBRV16, FPA19] et nous la renforçons amplement. Dans le passé, les approches locales et globales se concentraient sur les nombres flous dits *triangulaires*, c'est-à-dire, avec des fonctions de dispersion linéaires. Les résultats présentés ici considèrent plus généralement des coefficients flous L-R symétriques avec un support borné et les mêmes fonctions de dispersion bijectives comme, par exemple, les nombres flous quadratiques, dont les fonctions de dispersion sont quadratiques. Comme dans les précédents travaux sur ce sujet, notre notion de solution repose sur l'égalité des fonctions d'appartenance.

IV.3 La transformation réelle d'une équation floue

Le but est de résoudre dans $\mathbb{R}$ un système d'équations polynomiales dont les coefficients sont des nombres flous symétriques appartenant à une même famille $\mathfrak{F}(L, L)$ où L est bijective. Pour ce faire, nous transformons indépendamment chaque équation afin d'obtenir des systèmes polynomiaux à coefficients réels pour qu'ils puissent être résolus par des méthodes algébriques. Cette section est consacrée à la transformation d'une seule équation. Notons qu'en pratique nous ne rencontrons pas une seule équation multivariée isolée. Pour un système réduit à une unique équation, le nombre de variables est généralement réduit à une seule. Ce cas particulier peut être traité avec notre méthode ou par d'autres, comme [AO06], et plus récemment [FPA19], mais ce n'est pas l'objectif de ce travail.

Nous utilisons la terminologie suivante : une variable est dite *réelle* si elle représente un nombre réel ; une variable réelle est dite *positive* si elle représente un nombre réel positif, c'est-à-dire appartenant à $\mathbb{R}^+$; un k-uplet $(b_1, \ldots, b_k)$ de variables réelles ou de nombres réels est dit positif si chaque composante b_i est positive. Dans cette section, nous considérons une équation algébrique (E) avec des coefficients flous L-R symétriques et k variables réelles $x_1, \ldots, x_k$ aussi appelées les indéterminées.

Le problème lors du calcul avec des variables réelles et des nombres flous vient des nombres flous exprimés comme un produit $q \cdot \widetilde{n}$, car les dispersions dépendent du signe de $q \in \mathbb{R}$ (voir le Lemme IV.2.1). Dans l'équation floue (E), chaque monôme $m = x_1^{d_1} \cdots x_k^{d_k}$ $(d_i \in \mathbb{N})$ dont le signe est inconnu a priori avant évaluation, joue le rôle de q ; notons que si chaque exposant d_i est pair, m est positif. Lorsque le k-uplet $\boldsymbol{x} = (x_1, \ldots, x_k)$ est positif, le monôme m est nécessairement positif.

Ainsi, pour éviter ce problème de signe, dans la Section IV.3.2, nous allons chercher à obtenir les solutions réelles de (E) dans $\mathbb{R}^k$ à partir des solutions *positives* de 2^k équations floues auxiliaires $E(I)$, où $I \in \{-1, 1\}^k$. Nous appellerons ces dernières équations les équations *induites* de E.

Dans les Sections IV.3.3 et IV.3.4, comme nous ne considérons que les variables réelles positives, les coefficients flous ne doivent pas nécessairement être symétriques mais peuvent plus généralement appartenir à une famille unique $\mathfrak{F}(L, R)$ où L et R sont bijectives. Dans la Section IV.3.3, nous construisons une *forme tranchée* de (E) afin de déduire une *forme tranchée collectée* de (E) ; en d'autres termes un système algébrique d'équations à coefficients réels dont les solutions positives sont celles de (E). La *forme tranchée collectée* est formée de quatre équations obtenues par un algorithme qui ne s'applique que dans le cas où les coefficients flous sont triangulaires. De plus, dans la Section IV.3.4, nous établissons une formule qui fournit une forme tranchée particulière de (E) formée par seulement trois équations. Nous l'appelons la *transformation réelle* de (E) et la désignons par $\mathcal{T}(E)$.

Ensuite, dans la Section IV.4, nous montrons que pour obtenir toutes les solutions réelles de (E), il est nécessaire et suffisant de collecter les solutions réelles positives des 2^k transformations réelles $\mathcal{T}(E(I))$, où $E(I)$ est une équation floue induite de (E). En pratique, les équations $E(I)$ ne sont pas distinctes deux à deux et il n'est pas nécessaire de résoudre chacun des 2^k systèmes $\mathcal{T}(E(I))$.

La Section IV.3.5 compare la transformation réelle $\mathcal{T}(E)$ à la forme tranchée collectée usuelle donnée dans la littérature pour le cas triangulaire. La Section IV.3.6 généralise finalement les résultats aux nombres flous trapézoïdaux.

IV.3.1 Préliminaires

Posons $\boldsymbol{d} = (d_1, \ldots, d_k) \in \mathbb{N}^k$, $\boldsymbol{x} = (x_1, \ldots, x_k)$, et $\boldsymbol{x}^{\boldsymbol{d}} = x_1^{d_1} \cdots x_k^{d_k}$ le *monôme* de multidegré $\boldsymbol{d}$ en les variables $x_1, \ldots, x_k$. De même, pour $\boldsymbol{a} = (a_1, \ldots, a_k) \in \mathbb{R}^k$, on désigne par $\boldsymbol{a}^{\boldsymbol{d}}$ le produit $a_1^{d_1} \cdots a_k^{d_k}$. Pour $\boldsymbol{y} = (y_1, \ldots, y_k)$, nous désignons par $\boldsymbol{x} \times \boldsymbol{y}$ le produit classique $(x_1\, y_1, \ldots, x_k\, y_k)$. Notons que $(\boldsymbol{x} \times \boldsymbol{y})^{\boldsymbol{d}} = \boldsymbol{x}^{\boldsymbol{d}} \boldsymbol{y}^{\boldsymbol{d}}$.

Dans cette section, nous considérons l'équation polynomiale suivante

$$(E) : \sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} \widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{x}^{\boldsymbol{d}} = \widetilde{m} \quad , \tag{IV.5}$$

où les coefficients sont donnés dans leurs représentations en tuple respectives : $\widetilde{m} = (m, \alpha, \beta)$ et $\widetilde{n_{\boldsymbol{d}}} = (n_{\boldsymbol{d}}, \alpha_{\boldsymbol{d}}, \beta_{\boldsymbol{d}}) \neq (0, 0, 0)$ pour chaque d dans $\mathrm{Expon}(E)$, un sous-ensemble fini de $\mathbb{N}^k$. Par exemple, lorsque $k = 3$ et (E) est l'équation $\widetilde{3}\, x_1^2 x_2 + \widetilde{1}\, x_3^4 = \widetilde{6}$, $\mathrm{Expon}(E)$ est le sous-ensemble $\{(2, 1, 0), (0, 0, 4)\}$ de $\mathbb{N}^3$.

Lorsque nous cherchons les solutions positives de (E) seulement, dans le cas $q \geq 0$ de l'Identité (IV.1), il n'est pas nécessaire d'imposer $L = R$: $\widetilde{m} \in \mathfrak{F}(L, R)$ et $\widetilde{n_{\boldsymbol{d}}} \in \mathfrak{F}(L, R) \backslash \{(0, 0, 0)\}$, $d \in \mathrm{Expon}(E)$. Nous désignons par $\mathrm{Sol}^+(E)$ l'ensemble des solutions réelles positives de (E) :

$$\mathrm{Sol}^+(E) = \{\, \boldsymbol{a} \in {\mathbb{R}^+}^k \mid \sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} \widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{a}^{\boldsymbol{d}} = \widetilde{m} \,\} \quad .$$

Dans le cas $q \leq 0$ de l'Identité (IV.1), pour les solutions dans l'ensemble des nombres réels, les coefficients sont nécessairement des nombres flous symétriques : $\widetilde{m} \in \mathfrak{F}(L, L)$ et $\widetilde{n_{\boldsymbol{d}}} \in \mathfrak{F}(L, L) \backslash \{(0, 0, 0, 0)\}$, $d \in \mathrm{Expon}(E)$. Nous désignons par $Sol(E)$ l'ensemble des solutions réelles de (E) :

$$\mathrm{Sol}(E) = \{\, \boldsymbol{a} \in \mathbb{R}^k \mid \sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} \widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{a}^{\boldsymbol{d}} = \widetilde{m} \,\} \quad .$$

Nous cherchons $\mathrm{Sol}(E)$ en utilisant les r-coupes de façon à obtenir un système algébrique à coefficients réels qui peut être résolu avec des méthodes classiques de calcul formel.

Cependant, selon l'Identité (3) du Lemme IV.2.1, la r-coupe de $\widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{x}^{\boldsymbol{d}}$ dépend du signe de $\boldsymbol{x}^{\boldsymbol{d}}$ alors que ce signe est autant inconnu que ceux des indéterminés $x_1, \ldots, x_k$.

C'est pourquoi nous devons considérer le cas où les indéterminés $x_1, \ldots, x_k$ sont réelles et positives. Seulement nous voulons obtenir l'ensemble de tous les zéros réels de (E), pas seulement les zéros positifs. C'est la discussion de la Section IV.3.2. Cette section vise à construire $\mathrm{Sol}(E)$ à partir des 2^k ensembles de zéros réels positifs de $\mathrm{Sol}^+(E(I))$ des équations floues induites $E(I)$, où I parcourt les k-uplets de $\{-1, 1\}^k$.

Après la Section IV.3.2, il restera à déterminer comment calculer les 2^k ensembles $\mathrm{Sol}^+(E(I))$. Dans cette optique, les Section IV.3.3 et Section IV.3.4 supposent que chaque variable est réelle et positive. Dans ce contexte, nous établissons la formule de la transformation réelle $\mathcal{T}(E)$ de (E). Les solutions positives du système algébrique réel $\mathcal{T}(E)$ sont exactement les solutions positives de l'équation algébrique floue (E). Dans ces deux sections, l'équation floue (E) jouera le rôle de chacune de ses équations induites $E(I)$.

IV.3.2 Solutions de (E) en fonction des solutions positives de ses équations induites

Soit $\boldsymbol{a} = (a_1, \ldots, a_k) \in \mathbb{R}^k$. Nous posons $|\boldsymbol{a}| = (|a_1|, \ldots |a_k|)$ où $|b|$ désigne la valeur absolue de $b \in \mathbb{R}$. Notons que $|\boldsymbol{a}^{\boldsymbol{d}}| = |\boldsymbol{a}|^{\boldsymbol{d}}$. Le k-uplet des signes des composantes de $\boldsymbol{a}$ est $\boldsymbol{\varepsilon}(\boldsymbol{a}) = (\,\mathrm{sign}(a_1), \ldots, \mathrm{sign}(a_k)\,) \in \{-1, 1\}^k$. La valeur $\boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}} = \prod_{i=1}^k \mathrm{sign}(a_i)^{d_i}$ est le signe 1 ou -1 du réel $\boldsymbol{a}^{\boldsymbol{d}}$ et nous avons

$$\boldsymbol{a}^{\boldsymbol{d}} = \boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}}\, |\boldsymbol{a}^{\boldsymbol{d}}| = \boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}}\, |\boldsymbol{a}|^{\boldsymbol{d}} \quad .$$

L'évaluation en la valeur $\boldsymbol{a}$ d'un terme $\widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{x}^{\boldsymbol{d}}$ consiste à remplacer le monôme $\boldsymbol{x}^{\boldsymbol{d}}$ par sa valeur $\boldsymbol{a}^{\boldsymbol{d}} = \boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}}\, |\boldsymbol{a}|^{\boldsymbol{d}}$. Selon l'Identité (3) du Lemme IV.2.1, quand le signe $\boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}}$ de $\boldsymbol{a}^{\boldsymbol{d}}$ est 1, la r-coupe du nombre flou $\widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{a}^{\boldsymbol{d}}$ est

$$[\ \underline{n_{\boldsymbol{d}}}(r)\ |\boldsymbol{a}|^{\boldsymbol{d}}\ ,\ \overline{n_{\boldsymbol{d}}}(r)\ |\boldsymbol{a}|^{\boldsymbol{d}}\ ]\ ,$$

et si ce signe est -1 alors la r-coupe est

$$[\ -\overline{n_{\boldsymbol{d}}}(r)\ |\boldsymbol{a}|^{\boldsymbol{d}}\ ,\ -\underline{n_{\boldsymbol{d}}}(r)\ |\boldsymbol{a}|^{\boldsymbol{d}}\ ] \quad .$$

Il est donc impossible de transformer $\widetilde{n_{\boldsymbol{d}}} x^{\boldsymbol{d}}$ en sa représentation paramétrique sans connaître le signe de $x^{\boldsymbol{d}}$. Notre idée pour contourner ce problème de signe inconnu de $\boldsymbol{x}^{\boldsymbol{d}}$ est d'introduire un k-uplet artificiel $I \in \{-1, 1\}^k$ pour les signes des indéterminés. Alors le réel $I^{\boldsymbol{d}}$ peut être multiplié par le coefficient $\widetilde{n_{\boldsymbol{d}}}$ et $\boldsymbol{x}$ peut être supposé positif. Cela conduit à construire à partir de (E) 2^k équations floues induites $E(I)$ définies ci-dessous.

Définition IV.3.1. Soit (E) une équation algébrique floue avec k variables et des coefficients flous symétriques donnée par (IV.5). Les *équations induites* de (E) sont les équations suivantes :

$$E(I) : \sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} I^{\boldsymbol{d}}\, \widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{x}^{\boldsymbol{d}} = \widetilde{m} \qquad \text{où} \quad I \in \{-1, 1\}^k. \tag{IV.6}$$

Notons que $(E) = E((1, 1, \ldots, 1))$. Dans $E(I)$, le k-uplet I joue le rôle du k-uplet $\boldsymbol{\varepsilon}(\boldsymbol{a})$

des signes d'une solution possible $\boldsymbol{a}$ de (E). Pour chaque terme $I^{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}}\ \boldsymbol{x}^{\boldsymbol{d}}$ dans la partie gauche de $E(I)$, $I^{\boldsymbol{d}} \in \{-1, 1\}$ joue le rôle du signe $\boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}}$ de $\boldsymbol{a}^{\boldsymbol{d}}$ alors que le signe de $\boldsymbol{x}^{\boldsymbol{d}}$ lui-même est considéré comme positif. Dans $E(I)$, il n'y a pas de problème pour exprimer les r-coupes de $I^{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}}\ \boldsymbol{x}^{\boldsymbol{d}}$ avec $\boldsymbol{x}$ positif alors que c'est impossible dans (E) pour $\widetilde{n_{\boldsymbol{d}}}\ \boldsymbol{x}^{\boldsymbol{d}}$ puisque $\boldsymbol{x}$ n'est pas nécessairement positif.

Pour décrire les relations entre les solutions des équations (E) et $E(I)$, pour $\boldsymbol{q} = (q_1, \ldots, q_k) \in \mathbb{R}^k$ et $\mathcal{S} \subset \mathbb{R}^k$, nous introduisons la notation suivante :

$$\boldsymbol{q} \otimes \mathcal{S} \ = \ \{\ \boldsymbol{q} \times \boldsymbol{a} = (q_1\ a_1, \ldots, q_k\ a_k) \mid \boldsymbol{a} = (a_1, \ldots, a_k) \in \mathcal{S}\ \} \quad .$$

Lemme IV.3.1. *Soit $I \in \{-1, 1\}^k$ et $E(I)$ une équation induite de l'équation (E). Alors*

$$\mathrm{Sol}(E) = I \otimes \mathrm{Sol}(E(I)) \quad \text{et} \quad \mathrm{Sol}(E(I)) = I \otimes \mathrm{Sol}(E)\ .$$

Preuve. Comme $I \times I = (1, \ldots, 1)$, il suffit de prouver la première égalité. Pour $\boldsymbol{b} \in \mathbb{R}^k$ une solution de $E(I)$, nous avons

$$\widetilde{m} = \sum_{\boldsymbol{d}} I^{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}}\ \boldsymbol{b}^{\boldsymbol{d}} = \sum_{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}}\ (I \times \boldsymbol{b})^d\ .$$

Donc $I \times \boldsymbol{b}$ est une solution de (E).
Inversement, soit $\boldsymbol{a} \in \mathrm{Sol}(E)$. Alors $\boldsymbol{b} = I \times a$ est une solution de $E(I)$ puisque

$$\widetilde{m} = \sum_{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}}\ (I \times I \times \boldsymbol{a})^{\boldsymbol{d}} = \sum_{\boldsymbol{d}}\ I^{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}}\ (I \times \boldsymbol{a})^d. \qquad \square$$

Pour chaque $\boldsymbol{a} \in \mathbb{R}^k$, il existe $I \in \{-1, 1\}^k$ tel que $\boldsymbol{\varepsilon}(\boldsymbol{a}) = I$ et pour lequel l'évaluation de $E(I)$ en $|\boldsymbol{a}|$ est identique à celle de (E) en $\boldsymbol{a}$. La question de trouver toutes les solutions de (E) à partir des solutions positives de ses équations floues induites $E(I)$ est résolue par le théorème fondamental suivant :

Theorème IV.3.2. *Considérons une équation algébrique floue (E) avec des coefficients flous symétriques. Alors, l'ensemble des solutions réelles de (E) est formé par les $I \times \boldsymbol{b}$ où I parcourt l'ensemble des k-uplets $\{-1, 1\}^k$ et $\boldsymbol{b}$ parcourt l'ensemble des solutions positives de $E(I)$. En d'autres termes,*

$$\mathrm{Sol}(E) = \bigcup_{I \in \{-1,1\}^k} I \otimes \mathrm{Sol}^+(E(I))\ .$$

Preuve. Il découle du Lemme IV.3.1 que chaque $I \otimes \mathrm{Sol}^+(E(I)) \subset \mathrm{Sol}(E)$. Inversement,

si $\boldsymbol{a} \in \mathrm{Sol}(E)$ alors $\boldsymbol{a}^{\boldsymbol{d}} = \boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}}\ |\boldsymbol{a}^{\boldsymbol{d}}| = \boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}}\ |\boldsymbol{a}|^{\boldsymbol{d}}$. Il s'ensuit que

$$\widetilde{m} = \sum_{\boldsymbol{d}} \boldsymbol{a}^{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}} = \sum_{\boldsymbol{d}}\ \boldsymbol{\varepsilon}(\boldsymbol{a})^{\boldsymbol{d}}\ |\boldsymbol{a}|^{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}}\ .$$

En posant $I = \boldsymbol{\varepsilon}(\boldsymbol{a})$ et $\boldsymbol{b} = |\boldsymbol{a}| \in \mathbb{R}^{+k}$, on obtient $I \times \boldsymbol{b} = \boldsymbol{\varepsilon}(\boldsymbol{a}) \times (\boldsymbol{\varepsilon}(\boldsymbol{a}) \times \boldsymbol{a}) = \boldsymbol{a}$ où $\boldsymbol{b}$ est une solution de $E(I)$. □

Remarque IV.3.1. *Les coefficients flous symétriques de* (E) *étant donnés dans leur représentation en tuple, les équations induites* $E(I)$ *sont en réalité directement dérivées de* (E)*. Pour tout* $\boldsymbol{d}$ *dans Expon*(E) *le coefficient* $I^{\boldsymbol{d}}\ \widetilde{n_{\boldsymbol{d}}}$ *est égal à* $\widetilde{n_{\boldsymbol{d}}}$ *lui-même quand* $I^{\boldsymbol{d}} = 1$*, sinon selon* (IV.2)*, il est égal à* $(-n_d, \beta_d, \alpha_d)$ *quand* $I^{\boldsymbol{d}} = -1$*. Ceci assure que pour trouver l'ensemble de toutes les solutions réelles de* (E)*, il suffit d'être capable de calculer les solutions réelles positives de n'importe quelle équation floue.*

Dans la pratique, les 2^k équations floues $E(I)$ ne sont pas toujours deux-à-deux distinctes. En particulier, dans le cas où chaque composante d_i de tout $\boldsymbol{d} = (d_1, \ldots, d_k) \in \mathrm{Expon}(E)$ est paire, toutes les équations induites $E(I)$ sont identiques à l'équation (E).

Exemple IV.3.1. Soient $\widetilde{3}, \widetilde{1}, \widetilde{6} \in \mathfrak{F}(L, L)$. Prenons $k = 3$ variables x_1, x_2, x_3 et considérons l'équation

$$(E)\ :\quad \widetilde{3}\, x_1^2 x_2 + \widetilde{1}\, x_3^4\ =\ \widetilde{6}\quad .$$

Les $8 = 2^3$ triplets de $\{-1, 1\}^3$ sont : $I_1 = (1,1,1), I_2 = (1,1,-1), I_3 = (-1,1,1), I_4 = (-1,1,-1), I_5 = (1,-1,1), I_6 = (1,-1,-1), I_7 = (-1,-1,1)$ et $I_8 = (-1,-1,-1)$. Il y a seulement deux équations induites distinctes $E(I_j)$. En effet, puisque $\mathrm{Expon}(E) = \{\boldsymbol{d}_1 = (2,1,0), \boldsymbol{d}_2 = (0,0,4)\}$, nous avons $I_j^{\boldsymbol{d}_2} = 1$ pour tout $j = 1, \ldots, 8$. Ainsi, les huit I_j se scindent en les deux groupes suivants : celui pour $j = 1, \ldots, 4$ avec $I_j^{\boldsymbol{d}_1} = 1$ et celui pour $j = 5, \ldots, 8$ avec $I_j^{\boldsymbol{d}_1} = -1$. Donc $(E) = E(I_j)$ pour $j = 1, \ldots, 4$ et $E(I_5) = E(I_j)$ pour $j = 5, \ldots, 8$.

Pour $j = 5, \ldots, 8$, la représentation en tuple des coefficients de $E(I_j)$ est déterminée à l'aide de la Remarque IV.3.1 : avec $\widetilde{3} = (3, \alpha_1, \beta_1)$, $\widetilde{1} = (1, \alpha_2, \beta_2)$ et $\widetilde{6} = (6, \alpha, \beta)$, nous avons $-\widetilde{3} = (-3, \beta_1, \alpha_1)$ et

$$E(I_j) : (-3, \beta_1, \alpha_1)x_1^2 x_2 + (1, \alpha_2, \beta_2)x_3^4 = (6, \alpha, \beta).$$

D'après le Théorème IV.3.2, nous avons

$$\mathrm{Sol}(E) = \bigcup_{j=1}^{4} I_j \otimes \mathrm{Sol}^+(E(I_1)) \qquad \bigcup \qquad \bigcup_{j=5}^{8} I_j \otimes \mathrm{Sol}^+(E(I_5)) \quad .$$

Les sections qui suivent sont consacrées à la recherche des solutions positives de (E)

que nous appliquerons à chaque équation $E(I)$, en tenant compte de la Remarque IV.3.1 illustrée dans l'exemple précédent.

IV.3.3 Forme tranchée de (E) pour trouver $\mathrm{Sol}^+(E)$

Dans cette section et la suivante, nous exprimons les solutions positives de (E). Puisqu'il n'est pas nécessaire de supposer $L = R$, nous considérons plus généralement que les coefficients flous se trouvent tous dans une famille unique $\mathfrak{F}(L, R)$ où L et R sont bijectifs.

Comme expliqué précédemment, nous appliquerons les résultats de cette section à chaque équation induite $E(I)$. La résolution algébrique de l'équation floue (E) est généralement basée sur le passage de la représentation L-R des nombres flous à leur représentation paramétrique. Dans la présentation ci-dessous, nous renforçons de manière significative cette méthode classique pour les coefficients flous triangulaires en l'appliquant à un système générique et en l'étendant à des coefficients flous plus généraux.

D'après le Lemme IV.2.1, l'équation (E) se réécrit en deux égalités sur les r-coupes de ses membres gauche et droite si tous les $\boldsymbol{x}^{\boldsymbol{d}}$, $\boldsymbol{d} \in \mathrm{Expon}(E)$, représentent des réels de même signe. En effet, selon la Règle (3) de ce lemme, la multiplication d'un nombre flou par un scalaire q se divise en deux cas : $q \leq 0$ et $q \geq 0$. Ainsi nous ne cherchons que les solutions $\boldsymbol{a} \in \mathbb{R}^{+k}$ puisque le réel $q := \boldsymbol{a}^{\boldsymbol{d}}$ est alors positif pour chaque $\boldsymbol{d} \in \mathbb{N}^k$.

Selon le Lemme IV.2.1, l'équivalence suivante s'applique à tout $\boldsymbol{a} \in \mathbb{R}^{+k}$:

$$\boldsymbol{a} \in \mathrm{Sol}^+(E) \iff \left[\sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} \underline{n_{\boldsymbol{d}}}(r)\, \boldsymbol{a}^{\boldsymbol{d}}, \sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} \overline{n_{\boldsymbol{d}}}(r)\, \boldsymbol{a}^{\boldsymbol{d}} \right] = [\underline{m}(r), \overline{m}(r)] \quad . \tag{IV.7}$$

Cette équivalence nous amène à considérer le système $\mathcal{C}(E)$ suivant de deux équations à coefficients réels et à $k+1$ variables $x_1, \ldots, x_k, r$, appelé *la forme tranchée* de (E) :

$$\mathcal{C}(E) : \begin{cases} \displaystyle\sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} \underline{n_{\boldsymbol{d}}}(r)\, \boldsymbol{x}^{\boldsymbol{d}} = \underline{m}(r) \\ \displaystyle\sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} \overline{n_{\boldsymbol{d}}}(r)\, \boldsymbol{x}^{\boldsymbol{d}} = \overline{m}(r) \end{cases} .$$

Soit F un ensemble fini d'équations dans $\mathbb{R}[x_1, \ldots, x_k, r]$. Nous posons

$$\mathrm{Sol}_k^+(F) = \{\boldsymbol{a} \in \mathbb{R}^{+k} \mid \forall r \in [0,1]\ (a_1, \ldots, a_k, r) \in \mathrm{Sol}(F)\}$$

où $\mathrm{Sol}(F)$ est l'ensemble des solutions de F dans $\mathbb{R}^{k+1}$. Prenons $\boldsymbol{a} = (a_1, \ldots, a_k) \in \mathbb{R}^{+k}$. D'après l'Équivalence (IV.7), le k-uplet $\boldsymbol{a}$ est une solution de (E) si et seulement si pour tout réel $r \in [0,1]$ le $(k+1)$-uplet $(a_1, \ldots, a_k, r)$ est une solution de la forme tranchée $\mathcal{C}(E)$. En d'autres termes, $\mathrm{Sol}_k^+(\mathcal{C}(E))$ est l'ensemble des solutions positives de l'équation

floue (E) :

$$\mathrm{Sol}^+(E) = \mathrm{Sol}^+_k(\mathcal{C}(E)) \quad . \tag{IV.8}$$

Exemple IV.3.2. Reprenons l'équation floue (E) de l'Exemple IV.3.1 avec des coefficients flous triangulaires. Nous rappelons que leurs r-coupes sont données par les formules (IV.4) et que $\mathrm{Expon}(E(I_1)) = \mathrm{Expon}(E(I_5)) = \{(2,1,0),(0,0,4)\}$. Les représentations paramétriques des coefficients des deux équations induites sont alors écrites comme suit :

- $\widetilde{n_{\boldsymbol{d}_1}} = \widetilde{3} = [\alpha_1 r + 3 - \alpha_1, -\beta_1 r + 3 - \beta_1]$ pour $E(I_1)$;
- $\widetilde{n_{\boldsymbol{d}_1}} = -\widetilde{3} = [\beta_1 r - 3 - \beta_1, -\alpha_1 r - 3 - \alpha_1]$ pour $E(I_5)$;
- $\widetilde{n_{\boldsymbol{d}_2}} = \widetilde{1} = [\alpha_2 r + 1 - \alpha_2, -\beta_2 r + 1 - \beta_2]$ et $\widetilde{6} = [\alpha r + 6 - \alpha, -\beta r + 6 + \beta]$ pour $E(I_1)$ et $E(I_5)$.

Par conséquent, les formes tranchées respectives des deux équations distinctes induites de (E) sont :

$$\mathcal{C}(E(I_1)) : \begin{cases} (\alpha_1 r + 3 - \alpha_1)\, x_1^2 x_2 & + & (\alpha_2 r + 1 - \alpha_2)\, x_3^4 & = \alpha r + 6 - \alpha \\ (-\beta_1 r + 3 - \beta_1)\, x_1^2 x_2 & + & (-\beta_2 r + 1 - \beta_2)\, x_3^4 & = -\beta r + 6 + \beta \end{cases} \quad \text{et}$$

$$\mathcal{C}(E(I_5)) : \begin{cases} (\beta_1 r - 3 - \beta_1)\, x_1^2 x_2 & + & (\alpha_2 r + 1 - \alpha_2)\, x_3^4 & = \alpha r + 6 - \alpha \\ (-\alpha_1 r - 3 - \alpha_1)\, x_1^2 x_2 & + & (-\beta_2 r + 1 - \beta_2)\, x_3^4 & = -\beta r + 6 + \beta \end{cases} \quad .$$

Dans le cas particulier triangulaire, illustré dans l'Example IV.3.2 ci-dessous, la forme tranchée possède deux équations avec dans chaque membre des expressions linéaires en une seule indéterminée r ; c'est une conséquence des formules (IV.4) restreintes au cas particulier triangulaire. Le cas triangulaire est simple car les fonctions de dispersion sont égales à $F : x \mapsto 1 - x$, avec $F^{-1} = F$. Le théorème suivant traite le cas général avec deux interminées, lorsque les fonctions de dispersion sont justes bijectives :

Theorème IV.3.3. *Soient L et R deux fonctions de dispersion et*

$$(E) : \sum_{\boldsymbol{d} \in Expon(E)} \widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{x}^{\boldsymbol{d}} = \widetilde{m} \ ,$$

une équation floue à coefficients dans la famille $\mathfrak{F}(L,R)$ donnés dans leurs représentations en tuple comme suit : $\widetilde{m} = (m, \alpha, \beta)$ et $\widetilde{n_{\boldsymbol{d}}} = (n_{\boldsymbol{d}}, \alpha_{\boldsymbol{d}}, \beta_{\boldsymbol{d}})$ pour $\boldsymbol{d} \in Expon(E)$. Si les fonctions de dispersion L et R sont bijectives alors la forme tranchée de (E) est donnée par :

$$\mathcal{C}(E) : \begin{cases} \sum\limits_{\boldsymbol{d}} n_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} - m + (\alpha - \sum\limits_{\boldsymbol{d}} \alpha_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}})\, u & = & 0 \\ \sum\limits_{\boldsymbol{d}} n_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} - m + (-\beta + \sum\limits_{\boldsymbol{d}} \beta_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}})\, v & = & 0 \ , \end{cases} \tag{IV.9}$$

où $u = L^{-1}(r)$ et $v = R^{-1}(r)$ pour tout $r \in [0,1]$. Pour $\boldsymbol{a} \in \mathbb{R}^{+k}$, nous avons $\boldsymbol{a} \in \mathrm{Sol}^+(E)$ si et seulement si, pour tout $r \in [0,1]$, le système (IV.9) *est satisfait par le $(k+2)$-uplet $(a_1, \ldots, a_k, L^{-1}(r), R^{-1}(r))$.*

Preuve. Par définition, une fonction de dispersion H envoie $[0,1]$ dans lui-même, et si de plus H est bijective, alors son inverse H^{-1} est continue et décroissante avec $H^{-1}(1) = 0$ et $H^{-1}(0) = 1$. Supposons que les fonctions de dispersion L et R soient bijectives. Comme chaque r appartient à $[0,1]$, nous pouvons poser $u = L^{-1}(r)$ et $v = R^{-1}(r)$. Lorsque r parcourt $[0,1]$ dans le sens croissant, les paramètres u et v parcourent le même intervalle $[0,1]$ dans le sens décroissant par définition des fonctions de dispersion. Ainsi, selon les formules (IV.3), la forme paramétrique des coefficients de l'équation (E) est donnée par

$$\begin{array}{rclcrcll} \underline{n_{\boldsymbol{d}}}(r) & = & n_{\boldsymbol{d}} - \alpha_{\boldsymbol{d}}\, u & , & \overline{n_{\boldsymbol{d}}}(r) & = & n_{\boldsymbol{d}} + \beta_{\boldsymbol{d}}\, v & \text{pour } \boldsymbol{d} \in \mathrm{Expon}(E) \\ \underline{m}(r) & = & m - \alpha\, u & , & \overline{m}(r) & = & m + \beta\, v & . \end{array} \tag{IV.10}$$

Alors la forme tranchée $\mathcal{C}(E)$ de (E) donnée dans (IV.8) s'écrit comme un système de deux équations avec $k+2$ variables $x_1, \ldots, x_k, u, v$, où u et v sont dépendantes l'une de l'autre ($L(u) = R(v)$ sur $[0,1]$) :

$$\mathcal{C}(E) : \begin{cases} \sum\limits_{\boldsymbol{d}} n_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} - \alpha_{\boldsymbol{d}}\, u\, \boldsymbol{x}^{\boldsymbol{d}} = m - \alpha\, u \\ \sum\limits_{\boldsymbol{d}} n_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} + \beta_{\boldsymbol{d}}\, v\, \boldsymbol{x}^{\boldsymbol{d}} = m + \beta\, v. \end{cases}$$

En rassemblant tous les termes à gauche des deux égalités, on retrouve la forme tranchée exprimée sous la forme (IV.9) du théorème.
La dernière affirmation du théorème sur $\mathrm{Sol}^+(E)$ découle directement de l'Identité (IV.8).
□

Ce qui est remarquable, c'est que le Théorème IV.3.3 requiert seulement que les restrictions sur $[0,1]$ des deux fonctions de dispersion L et R soient bijectives.

Notre approche permet à la fois d'améliorer et de généraliser les méthodes connues à ce jour. Par exemple, les résultats de [MBR13] et [BBRV16] sont restreints aux nombres flous triangulaires. En effet, la forme tranchée de (E) avec deux paramètres $u = L^{-1}(r)$ et $v = R^{-1}(r)$ donnée dans l'Identité (IV.9) est une généralisation de la forme tranchée connue dans le cas triangulaire avec un seul paramètre r où $r \in [0,1]$.

Dans les articles précités, pour chaque problème à résoudre, l'algorithme calcule le système $\mathcal{C}(E)$ en les variables $x_1, \ldots, x_k, r$, qui est linéaire par rapport à r. Puis il est réécrit en un système équivalent de quatre équations algébriques en $x_1, \ldots, x_k$ à coefficients réels, appelé *forme tranchée collectée* de (E).

Dans la section suivante IV.3.4, nous montrons comment obtenir une forme tranchée collectée particulière réduite à trois équations et non pas quatre, pour toute famille $\mathfrak{F}(L,R)$

avec des fonctions de dispersion L et R bijectives. Il s'agit de la transformation réelle de (E). De plus, nous donnons explicitement sa formulation à partir de (E).

IV.3.4 Transformation réelle et les solutions réelles positives de (E)

Nous définissons ici la transformation réelle d'une équation floue (E) et montrons que ses solutions réelles positives sont aussi celles de (E).

Définition IV.3.2. Soient L et R deux fonctions de dispersion et

$$(E) : \sum_{\boldsymbol{d}\in \mathrm{Expon}(E)} \widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{x}^{\boldsymbol{d}} = \widetilde{m}$$

une équation floue à coefficients symétriques dans la famille $\mathfrak{F}(L,R)$ donnés dans leurs représentations en tuple comme suit : $\widetilde{n_{\boldsymbol{d}}} = (n_{\boldsymbol{d}}, \alpha_{\boldsymbol{d}}, \beta_{\boldsymbol{d}})$ $(\boldsymbol{d} \in \mathrm{Expon}(E))$ et $\widetilde{m} = (m, \alpha, \beta)$. La *transformation réelle* $\mathcal{T}(E)$ de (E) est le système polynomial sur $\mathbb{R}$ suivant :

$$\mathcal{T}(E) : \begin{cases} \sum\limits_{\boldsymbol{d}\in \mathrm{Expon}(E)} n_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} &= m \\ \sum\limits_{\boldsymbol{d}\in \mathrm{Expon}(E)} \alpha_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} &= \alpha \\ \sum\limits_{\boldsymbol{d}\in \mathrm{Expon}(E)} \beta_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} &= \beta\ . \end{cases} \tag{IV.11}$$

Cette définition s'étend naturellement à un système (S) d'équations floues telles que (E). Nous désignons par $\mathcal{T}(S)$ sa transformation réelle, c'est-à-dire le système formé par les transformations réelles des équations de (S).

Theorème IV.3.4. *Selon les données de la Définition IV.3.2, si les deux fonctions de dispersion L et R sont bijectives alors l'ensemble des solutions réelles positives de (E) est celui de sa transformation réelle $\mathcal{T}(E)$; en d'autres termes :*

$$\mathrm{Sol}^+(E) = \mathrm{Sol}^+(\mathcal{T}(E)) \quad .$$

Preuve. Fixons $\boldsymbol{a} \in \mathbb{R}^{+k}$. Comme les fonctions de dispersion L et R sont bijectives, nous pouvons appliquer le Théorème IV.3.3. Selon ce théorème, nous savons que $\boldsymbol{a} \in \mathrm{Sol}^+(E)$ si et seulement si, pour tout $r \in [0,1]$, la forme tranchée de (E) exprimée en (IV.9) est satisfaite par le $(k+2)$-uplet $(a_1, \ldots, a_k, L^{-1}(r), R^{-1}(r))$.

Avec $r = 1$, nous avons $u = L^{-1}(1) = 0$. Alors $\boldsymbol{a} \in \mathrm{Sol}^+(E)$ satisfait l'équation $\sum_d\ n_{\boldsymbol{d}}\boldsymbol{x}^{\boldsymbol{d}} = m$. Notons que lorsque $r = 1$ nous avons également $v = 0$ car $R(0) = 1$, et nous trouvons la même équation, et non une seconde distincte. C'est pourquoi nous obtenons trois équations au lieu de quatre. Ensuite, en prenant $r = 0$ nous avons $u = L^{-1}(0) = 1$ et $v = R^{-1}(0) = 1$. En remplaçant dans (IV.9) l'expression $\sum_d\ n_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} - m$

par 0 et chaque variable u et v par 1, nous déduisons qu'une solution positive de (E) est aussi une solution positive de la transformation réelle $\mathcal{T}(E)$ de (E).

Pour l'inclusion inverse, nous considérons la forme tranchée $\mathcal{C}(E)$ comme un système polynomial en les variables de $\boldsymbol{x}$ et à coefficients dans l'anneau $\mathbb{R}[u, v]$. Toute solution $(a_1, \ldots, a_k) \in \mathbb{R}^k$ de $\mathcal{T}(E)$ est aussi une solution de $\mathcal{C}(E)$ dans $\mathbb{R}^k$ quels que soient les paramètres u et v dans l'intervalle $[0, 1]$. Evidemment cela reste vrai quand ils sont de plus reliés par la contrainte $L^{-1}(u) = R^{-1}(v) \in [0, 1]$. Par conséquent, toute solution réelle positive de la transformation réelle $\mathcal{T}(E)$ est aussi une solution réelle positive de l'équation floue (E) □

Le Théorème IV.3.4 assure que trouver les racines réelles positives de (E) revient à trouver les racines réelles positives de sa transformation réelle $\mathcal{T}(E)$. Il n'est donc pas utile de développer des calculs intermédiaires sur la représentation paramétrique comme les méthodes précédentes l'ont fait dans le cas spécifique triangulaire.

À partir de l'expression de la transformation réelle $\mathcal{T}(E)$ de E, nous pouvons maintenant déduire celles de ses équations induites $E(I)$. En nous référant à la Remarque IV.3.1, nous obtenons immédiatement le corollaire suivant :

Corollaire IV.3.1. *Sous les hypothèses de la Définition IV.3.2, soit $I \in \{-1, 1\}^k$ et $E(I)$ l'équation induite de E définie dans* (IV.6).

Alors la transformation réelle de $E(I)$ est donnée par :

$$\mathcal{T}(E(I)) : \begin{cases} \sum\limits_{\boldsymbol{d}|I^{\boldsymbol{d}}>0} n_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} - \sum\limits_{\boldsymbol{d}|I^{\boldsymbol{d}}<0} n_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} &= m \\ \sum\limits_{\boldsymbol{d}|I^{\boldsymbol{d}}>0} \alpha_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} + \sum\limits_{\boldsymbol{d}|I^{\boldsymbol{d}}<0} \beta_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} &= \alpha \\ \sum\limits_{\boldsymbol{d}|I^{\boldsymbol{d}}>0} \beta_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} + \sum\limits_{\boldsymbol{d}|I^{\boldsymbol{d}}<0} \alpha_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} &= \beta \quad . \end{cases} \tag{IV.12}$$

Si, de plus, les fonctions de dispersion L et R sont toutes deux bijectives alors $\mathrm{Sol}^+(E(I)) = \mathrm{Sol}^+(\mathcal{T}(E(I)))$.

IV.3.5 Comparaison avec les méthodes précédentes dans le cas triangulaire

Considérons un système (S) formé par s équations polynomiales à coefficients flous. Dans le cas particulier de nombres flous triangulaires utilisés comme coefficients, les auteurs de [MBR13] et [BBRV16] calculent une forme tranchée collectée de (S) formée par $4\,s$ équations algébriques réelles. Dans cette partie, nous nous intéressons à la relation entre leur forme tranchée collectée avec $4\,s$ équations et notre système tranché collectée avec $3\,s$ équations, à savoir la transformation réelle de (S). Les ensembles respectifs de solutions réelles positives de ces deux systèmes sont égaux à $\mathrm{Sol}(S)$; c'est le principe de toute forme tranchée collectée de (S).

Considérons ci-dessous le système F_1 de la Section 6 dans [BBRV16] :

$$F_1 : \begin{cases} (2,1,1)xy + (3,1,1)x^2y^2 + (2,1,1)x^3y^3 = (7,3,3) \\ (5,1,1)xy + (2,3,1)x^2y^2 + (2,2,1)x^3y^3 = (9,6,3) \end{cases}.$$

Appliqué à la première équation, l'algorithme proposé dans [BBRV16] produit la forme tranchée collectée suivante :

$$\begin{cases} xy + x^3y^3 - 3 + x^2y^2 = 0, \\ xy + 2\,x^2y^2 - 4 + x^3y^3 = 0 \\ -xy - x^3y^3 + 3 - x^2y^2 = 0 \\ 3\,xy + 4\,x^2y^2 - 10 + 3\,x^3y^3 = 0 \end{cases};$$

et il produit la forme tranchée collectée suivante de la deuxième équation de F_1 :

$$\begin{cases} xy + 3\,x^2y^2 + 2\,x^3y^3 - 6 = 0, \\ 4\,xy - x^2y^2 - 3 = 0, \\ -xy - x^3y^3 + 3 - x^2y^2 = 0, \\ 6\,xy + 3\,x^2y^2 - 12 + 3\,x^3y^3 = 0 \end{cases}.$$

Appelons T_1 le système formé des huit équations précédentes.

Par ailleurs, en appliquant à F_1 notre formule (IV.11) définissant la transformation réelle, on obtient $\mathcal{T}(F_1)$, le système suivant à six équations :

$$\mathcal{T}(F_1) : \begin{cases} 2xy + 3x^2y^2 + 2x^3y^3 = 7 \\ xy + x^2y^2 + x^3y^3 = 3 \\ xy + x^2y^2 + x^3y^3 = 3 \\ 5xy + 2x^2y^2 + 2x^3y^3 = 9 \\ xy + 3x^2y^2 + 2x^3y^3 = 6 \\ xy + x^2y^2 + x^3y^3 = 3 \end{cases}.$$

Un calcul facile montre l'équivalence des systèmes T_1 et $\mathcal{T}(F_1)$ dont l'ensemble des solutions est $\{(x,y) \in \mathbb{R} \mid xy = 1\}$. Ce phénomène d'équivalence entre les deux approches peut s'expliquer de manière très générale comme nous le montrons ci-dessous en considérant le calcul classique de la forme tranchée collectée obtenue par une application de l'algorithme de [BBRV16] sur l'équation générique (IV.5) de (E).

Soient $\widetilde{m} = (n, \alpha, \beta)$ et $\widetilde{n_{\boldsymbol{d}}} = (n_{\boldsymbol{d}}, \alpha_{\boldsymbol{d}}, \beta_{\boldsymbol{d}})$ ($\boldsymbol{d} \in \mathrm{Expon}(E)$) les représentations en tuple respectives des coefficients flous de (E) qui sont supposés triangulaires. Selon les formules

(IV.4), dans le cas triangulaire les r-coupes sont données par

$$\begin{aligned} \widetilde{n_{\boldsymbol{d}}}(r) &= [\alpha_{\boldsymbol{d}} r + n_{\boldsymbol{d}} - \alpha_{\boldsymbol{d}} \, , -\beta_{\boldsymbol{d}} r + n_{\boldsymbol{d}} + \beta_{\boldsymbol{d}}] \quad \text{pour } \boldsymbol{d} \in \mathrm{Expon}(E), \quad \text{et} \\ \widetilde{m}(r) &= [\alpha r + m - \alpha \, , -\beta r + m + \beta] \; . \end{aligned}$$

Pour un nombre flou triangulaire, $L = R = F$, où $F(x) = F^{-1}(x) = 1 - x$ est connue, les méthodes précédentes remplacent directement $L^{-1}(r)$ et $R^{-1}(r)$ par leur expression en la variable r dans les équations. C'est ainsi qu'elles se retrouvent dans la forme tranchée de (E) exprimée ci-dessous comme deux polynômes en la variable r :

$$\mathcal{C}(E) : \begin{cases} (\sum\limits_{\boldsymbol{d}} \alpha_{\boldsymbol{d}} \, \boldsymbol{x}^{\boldsymbol{d}} - \alpha) r + \sum\limits_{\boldsymbol{d}} (n_{\boldsymbol{d}} - \alpha_{\boldsymbol{d}}) \, \boldsymbol{x}^{\boldsymbol{d}} - m + \alpha = 0 \\ (\beta - \sum\limits_{\boldsymbol{d}} \beta_{\boldsymbol{d}} \, \boldsymbol{x}^{\boldsymbol{d}}) r + \sum\limits_{\boldsymbol{d}} (n_{\boldsymbol{d}} + \beta_{\boldsymbol{d}}) \, \boldsymbol{x}^{\boldsymbol{d}} - m - \beta = 0 \; . \end{cases}$$

Un k-uplet $(x_1, \ldots, x_k) \in \mathbb{R}^k$ est une solution de $\mathcal{C}(E)$ pour tout $r \in [0, 1]$ si et seulement si les coefficients constants et ceux de la variable r de ces équations indépendantes sont tous nuls (car un polynôme non nul de degré 1 possède une seule racine). La forme tranchée collectée de (E) s'écrit alors :

$$\begin{cases} \sum\limits_{\boldsymbol{d}} \alpha_{\boldsymbol{d}} \, \boldsymbol{x}^{\boldsymbol{d}} &= \alpha \\ \sum\limits_{\boldsymbol{d}} (n_{\boldsymbol{d}} - \alpha_{\boldsymbol{d}}) \, \boldsymbol{x}^{\boldsymbol{d}} &= m - \alpha \\ \sum\limits_{\boldsymbol{d}} \beta_{\boldsymbol{d}} \, \boldsymbol{x}^{\boldsymbol{d}} &= \beta \\ \sum\limits_{\boldsymbol{d}} (n_{\boldsymbol{d}} + \beta_{\boldsymbol{d}}) \, \boldsymbol{x}^{\boldsymbol{d}} &= m + \beta \; . \end{cases}$$

En appliquant cette transformation à chaque équation du système F_1, nous trouvons la forme tranchée collectée T_1 de notre exemple. Pour l'équation générique (E), en injectant la première équation dans la seconde et en notant que la dernière équation est la somme des trois autres, nous obtenons la transformation réelle $\mathcal{T}(E)$ avec trois équations définies en (IV.11).

IV.3.6 Cas des nombres flous trapézoïdaux

Dans cette section, nous étendons la définition des nombres flous aux nombres flous *trapézoïdaux*, en nous autorisant à prendre pour $\mu_{\widetilde{n}}^{-1}(\{1\})$ un intervalle $[a, b]$ et non plus la seule valeur n. Comme mentionné en Section IV.2.1, nos résultats s'adaptent aux nombres flous trapézoïdaux symétriques à support borné.

Dans ce contexte, un nombre flou $\widetilde{n}$ à support borné est de type L-R si sa fonction

d'appartenance $\mu_{\widetilde{n}}$ a la forme suivante :

$$\mu_{\widetilde{n}}(x) = \begin{cases} L\left(\frac{a-x}{\alpha}\right) & \text{pour} \quad a-\alpha \leq x < a \quad \text{si} \quad \alpha \neq 0 \\ 1 & \text{pour} \quad x \in [a,b] \\ R\left(\frac{x-b}{\beta}\right) & \text{pour} \quad b < x \leq b+\beta \quad \text{si} \quad \beta \neq 0 \\ 0 & \text{pour} \quad x \in]-\infty, a-\alpha[\cup]b+\beta, +\infty[\end{cases} .$$

Alors la représentation en tuple du nombre flou $\widetilde{n}$ est le quadruplet (a, b, α, β).

L'expression de la représentation paramétrique donnée dans la Proposition IV.2.2 prend la forme suivante pour un nombre trapézoïdal de type $L-R$ dont les fonctions de dispersion L et R sont bijectives :

$$\begin{cases} \underline{n}(r) = a - \alpha\, u \\ \overline{n}(r) = b + \beta\, v \qquad \text{où } u = L^{-1}(r) \text{ et } v = R^{-1}(r) \text{ pour } r \in [0,1] \end{cases} .$$

Premièrement, nous cherchons les solutions réelles positives d'une équation $(E) : \sum_{\boldsymbol{d} \in \mathrm{Expon}(E)} \widetilde{n_{\boldsymbol{d}}}\, \boldsymbol{x}^{\boldsymbol{d}} = \widetilde{m}$ avec des coefficients flous trapézoïdaux de type L-R, où $\widetilde{n_{\boldsymbol{d}}} = (a_d, b_d, \alpha_d, \beta_d)$ et $\widetilde{m} = (a, b, \alpha, \beta)$ sont leurs représentations en tuple. Pour ce faire, nous calculons sa transformation réelle $\mathcal{T}(E)$ sans recourir à l'hypothèse de symétrie sur les coefficients. Les formes paramétriques (IV.10) données dans la preuve du Théorème IV.3.3 deviennent

$$\begin{array}{rclcrcl} \underline{n_{\boldsymbol{d}}}(r) & = & a_{\boldsymbol{d}} - \alpha_{\boldsymbol{d}}\, u & , & \overline{n_{\boldsymbol{d}}}(r) & = & b_{\boldsymbol{d}} + \beta_{\boldsymbol{d}}\, v \quad \text{pour} \quad \boldsymbol{d} \in \mathrm{Expon}(E) \quad ; \\ \underline{m}(r) & = & a - \alpha\, u & , & \overline{m}(r) & = & b + \beta\, v \quad . \end{array}$$

En appliquant l'argument de Section IV.3.4 (ici $L(1) = R(1) = 0$ et $L(0) = R(0) = 1$), nous obtenons de la même manière une transformation réelle de (E), mais cette fois avec quatre équations :

$$\mathcal{T}(E) : \begin{cases} \sum\limits_{\boldsymbol{d}} a_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} = a \\ \sum\limits_{\boldsymbol{d}} b_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} = b \\ \sum\limits_{\boldsymbol{d}} \alpha_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} = \alpha \\ \sum\limits_{\boldsymbol{d}} \beta_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} = \beta \; . \end{cases} \tag{IV.13}$$

Deuxièmement, l'utilisation de la transformation réelle pour résoudre les systèmes polynomiaux flous se transpose directement aux systèmes à coefficients flous trapézoïdaux symétriques.

Les algorithmes proposés dans ce chapitre restent valables pour les nombres flous trapézoïdaux. Seule la fonction algorithmique **RealTransform**(S), qui retourne la transformation réelle d'un système flou (S) à s équations, doit être adaptée afin de retourner

la transformation réelle $\mathcal{T}(S)$ avec $4\,s$ au lieu de $3\,s$ équations, en appliquant la formule (IV.13) à chaque équation de (S).

IV.4 Résolution réelle des systèmes polynomiaux flous

La résolution d'un système flou (S) de s équations à coefficients dans $\mathfrak{F}(L, L)$ découle directement des résultats pour une seule équation. Dans la Section IV.3, de l'équation (E) nous déduisons 2^k équations floues induites $E(I)$, où I parcourt les 2^k k-uplets de $\{-1, 1\}^k$. En appliquant le Théorème IV.3.2 les solutions de (E) sont déduites des solutions positives de chaque équation floue induite $E(I)$. Le Corollaire IV.3.1 assure que les solutions positives de chaque $E(I)$ sont les solutions positives de la transformation réelle $\mathcal{T}(E(I))$. Dans cette section, pour le système (S), nous considérerons de la même manière 2^k transformations réelles $\mathcal{T}(S(I))$ de systèmes induits $S(I)$ avec $3\,s$ équations où I parcourt les 2^k k-uplets de $\{-1, 1\}^k$.

Cette section établit le Théorème principal IV.4.1 qui exprime les solutions réelles d'un système à partir des solutions positives des 2^k transformations réelles de ces systèmes induits. Un premier algorithme résulte d'une application directe de ce théorème. Ensuite, nous discutons de la façon de réduire le nombre d'instructions de calcul en déduisant certains des 2^k ensembles de solutions réelles de (S) à partir des solutions réelles positives de certains systèmes $\mathcal{T}(S(I))$ précédemment traitées par l'algorithme. Ces éléments conduisent à l'algorithme optimisé `SolveFuzzySystem` présenté à la Section IV.5.

IV.4.1 Fondations

Soient $(E_1), \ldots, (E_s)$ les s équations floues du système (S) avec des coefficients flous symétriques. Pour chaque $I \in \{-1, 1\}^k$, nous désignons par $S(I)$ le système flou induit formé par les s équations floues induites $E_1(I), \ldots, E_s(I)$ selon la notation de la Section IV.3.2. Pour chaque $I \in \{-1, 1\}^k$ la transformation réelle $\mathcal{T}(S(I))$ du système induit $S(I)$ est le système de $3\,s$ équations formé par les transformations réelles des équations induites $E_1(I), \ldots, E_s(I)$.

Le théorème principal qui suit est une conséquence directe du Théorème IV.3.2 et du Corollaire IV.3.1 :

Theorème IV.4.1. *Soit (S) un système flou à coefficients dans une même famille $\mathfrak{F}(L, L)$. Si la fonction de dispersion L est bijective alors l'ensemble des solutions de (S) est l'union des $I \times \boldsymbol{b}$ pour tous les k-uplets $\boldsymbol{b}$ qui sont des solutions réelles positives de la transformation réelle $\mathcal{T}(S(I))$ où I parcourt $\{-1, 1\}^k$. En d'autres termes,*

$$\mathrm{Sol}(S) = \bigcup_{I \in \{-1,1\}^k} \{\, I \times \boldsymbol{b} \;\mid\; \boldsymbol{b} \in \mathrm{Sol}^+(\mathcal{T}(S(I)))\,\} = \bigcup_{I \in \{-1,1\}^k} I \otimes \mathrm{Sol}^+(\mathcal{T}(S(I))).$$

Un premier algorithme pour la résolution réelle des systèmes flous, appelé BA-SFS, découle naturellement du Théorème IV.4.1 et des résultats précédents. Il est donné ci-dessous. Son entrée est un système polynomial à coefficients flous donnés dans la représentation en tuple et supposés appartenir à une même famille $\mathfrak{F}(L, L)$ dont la fonction de dispersion L est bijective. Il est basé sur les fonctions suivantes :

- **Multisign**(j) est la bijection naturelle entre $\{0, \ldots, 2^k - 1\}$ et $\{-1, 1\}^k$: l'image de l'entier $j = \sum_{i=0}^{2^k-1} b_i 2^i$ est le k-uplet $I = (c_o, \ldots, c_{2^k-1}) \in \{-1, 1\}^k$ avec $c_i = 2b_i - 1$;
- **RealTransform**(S, I) applique la formule (IV.12) pour retourner les $3\,s$ équations de la transformation réelle du système flou $S(I)$ de s équations (il s'agit de $4\,s$ équations lorsque les nombres flous sont trapézoïdaux) ;
- **SolPos**(SR) renvoie les solutions réelles positives d'un système SR d'équations polynomiales à coefficients dans $\mathbb{R}$. Elle peut être basée sur n'importe quelle méthode de calcul formel connue pour résoudre un tel système. Par exemple, dans l'implémentation décrite à la Section IV.6 c'est la méthode de décomposition de Wu en systèmes polynomiaux triangulaires qui est utilisée [Wu87].

Algorithme 12 BA-SFS, un algorithme basique pour résoudre les systèmes flous

Require: S, un système polynomial à coefficients flous dans $\mathfrak{F}(L, L)$ donnés sous la représentation en tuple, k la dimension, i.e. le nombre de variables
Ensure: *sol* , l'ensemble des solutions réelles de (S)
1: $sol := \{\ \}$
2: **for** $j := 0$ **to** $2^k - 1$ **do**
3: $I :=$ **MultiSign**(j)
4: $TRSI :=$ **RealTransform**(S, I)
5: $PR :=$ **SolPos**$(TRSI)$
6: $sol := sol \bigcup I \otimes PR$
7: **end for**
8: **return** *sol*

IV.4.2 Réduction du nombre de systèmes algébriques à résoudre

Par extension de la notation employée pour une équation, nous notons Expon(S) l'ensemble des $\boldsymbol{d}$ dans $\mathbb{N}^k$ tels que $\boldsymbol{x}^{\boldsymbol{d}}$ apparaît dans la partie gauche d'une ou plusieurs équations de (S), avec un coefficient non nul (la constante du côté droit peut être $0 = (0, 0, 0)$).

Dans la boucle for de l'algorithme BA-SFS, le système $S(I)$ peut être identique à l'un de ceux obtenus lors des étapes précédentes. Dans ce cas, un nouveau calcul de PR, l'ensemble des solutions réelles positives de $S(I)$, est redondant et il est souhaitable de l'éviter.

L'algorithme optimisé appelé `SolveFuzzySystem`, ou `SFS`, proposé à la Section IV.5 fournit chaque solution réelle de (S) en évitant les calculs inutiles.

Nous cherchons donc d'abord à identifier, à partir de (S), les systèmes induits $S(I)$ qui sont identiques. L'Exemple IV.3.1, qui traite une seule équation, montre comment cette situation peut se produire à partir des huit équations floues qui se réduisent à seulement deux équations différentes. Lorsque, par exemple, toutes les composantes de chaque k-uplet $\boldsymbol{d}$ de $\mathrm{Expon}(S)$ sont paires, les 2^k systèmes induits $S(I)$ sont identiques. Notre objectif est d'automatiser la reconnaissance de systèmes induits identiques.

Pour ce faire, nous construisons la matrice $M(S)$ des signes où les colonnes sont indicées par les 2^k k-uplets I de $\{-1,1\}^k$ et les lignes par les k-uplets $\boldsymbol{d}$ de $\mathrm{Expon}(S)$. L'élément à la ligne $\boldsymbol{d}$ et à la colonne I est $I^{\boldsymbol{d}} \in \{-1,1\}$ qui remplace le signe de $\boldsymbol{x}^{\boldsymbol{d}}$ dans $S(I)$:

$$M(S) = \begin{array}{c} \\ \boldsymbol{d} \end{array} \begin{array}{c} \begin{array}{c} I \end{array} \\ \begin{pmatrix} \cdots & \vdots \\ & I^{\boldsymbol{d}} \\ & \end{pmatrix} \end{array}$$

Fixons deux k-uplets distincts $I_1, I_2 \in \{-1,1\}^k$ et considérons $C(I_1)$ et $C(I_2)$ les deux colonnes de $M(S)$ indicées respectivement par I_1 et I_2. Si $C(I_1) = C(I_2)$ alors $S(I_1) = S(I_2)$. En effet, selon la définition des équations induites donnée dans (IV.6), pour chaque équation (E) du système (S), le coefficient $I_1^{\boldsymbol{d}}\, \widetilde{n_{\boldsymbol{d}}}$ de chaque monôme $\boldsymbol{x}^{\boldsymbol{d}}$ qui apparait dans la partie gauche de chaque équation $E(I_1)$ est identique au coefficient $I_2^{\boldsymbol{d}}\, \widetilde{n_{\boldsymbol{d}}}$ dans $E(I_2)$.

Le nombre de systèmes distincts $S(I)$ parmi les 2^k induits par (S) est donc au plus égal au nombre de colonnes distinctes dans $M(S)$. La matrice des signes permet d'éviter la résolution de nombreux systèmes induits. D'autres détections rapides de systèmes induits identiques peuvent être implémentées. En effet, deux systèmes $S(I_1)$ et $S(I_2)$ peuvent être identiques à une permutation des équations près alors que $C(I_1) \neq C(I_2)$. On peut aussi trouver des équations de la forme "$a = 0$" et "$-a = 0$" qui possèdent les mêmes solutions.

Remarque IV.4.1. Si deux systèmes sont tels que $S(I_1) = S(I_2)$, un seul système sera résolu, par exemple $S(I_1)$, mais ils produiront au moins deux ensembles de solutions $I_1 \times \mathrm{Sol}$ et $I_2 \times \mathrm{Sol}$ avec $\mathrm{Sol} = \mathrm{Sol}^+(S(I_1))$.

IV.5 Algorithme `SolveFuzzySystem`

Cette section propose un algorithme optimisé `SolveFuzzySystem`, ou `SFS` sous forme abrégée, pour calculer les solutions réelles d'un système polynomial flou. Les coefficients symétriques du système sont des nombres flous donnés dans la représentation en

tuple et sont supposés appartenir à une même famille $\mathfrak{F}(L, L)$ où la fonction de dispersion L est bijective. Après la description de l'algorithme séquentiel `SFS` dans la Section IV.5.1, nous discutons sa parallélisation dans la Section IV.5.2.

IV.5.1 L'algorithme séquentiel `SolveFuzzySystem`

Comme dans le premier algorithme `BA-SFS`, l'algorithme `SFS` consiste à parcourir de façon itérative les colonnes de la matrice des signes $M(S)$ décrite à la Section IV.4.2 (dans l'algorithme `BA-SFS` seuls les indices des colonnes sont utiles et non la matrice des signes). Pour chaque colonne $C(I)$ indexée par un k -uplet I de signes, l'algorithme recherche des solutions réelles du système (S) à partir de $S(I)$ en évitant tout calcul inutile si une colonne précédente le permet.

À cette fin, il utilise les trois vecteurs suivants, indexés de 0 à $2^k - 1$, qui sont nuls au début de l'algorithme :

- *DistinctColumns* contient les colonnes distinctes de la matrice des signes ;
- *DistinctSystems* contient les systèmes flous distincts dont les solutions positives ont été calculées ; ses indices sont liés aux indices de *DistinctColumns* ;
- *lb* contient les solutions réelles positives de $S(I)$ telles que $C(I)$ est dans *DistinctColumns* ; il sera donc naturellement indicé en suivant *DistinctColumns*.

Pour une colonne courante I, comme expliqué à la Section IV.4.2, les calculs redondants peuvent être évités dans les cas suivants :

Cas 1 Si la colonne $C(I)$ est identique à une colonne précédente $C(J)$ dans $M(S)$ alors $S(I) = S(J)$. Dans ce cas, l'algorithme n'ajoute pas la colonne $C(I)$ dans le vecteur *DistinctColumns*. Il utilise les solutions réelles positives de $S(J)$ déjà calculées et enregistrées dans *lb* lors d'une étape précédente (voir Remarque IV.4.1). C'est la première étape de l'algorithme (voir l'instruction **4 :**).

Cas 2 Si $S(I)$ est identique à un système $S(J)$ où $C(J)$ est une colonne qui précède $C(I)$ mais qui en est distincte, alors l'algorithme passe à l'indice suivant. Il ajoute la colonne $C(I)$ au vecteur *DistinctColumns* et les solutions positives de $S(J)$ déjà calculées dans *lb*. Comme il n'est pas pertinent d'ajouter $S(I)$ au vecteur *DistinctSystems*, l'algorithme ne l'ajoute pas à l'indice *cpt* lui correspondant, et sa place reste vide.

Nous pouvons ensuite appliquer les tests 1. et 2. aux colonnes suivantes de $M(S)$ (voir l'instruction **7 :**).

En dehors des situations 1. et 2., les solutions positives du système flou $S(I)$ sont calculées, comme dans l'algorithme de base `BA-SFS`, avec les fonctions **RealTransform** et **SolPos**. À l'indice *cpt* dans les vecteurs *DistinctColumns*, *DistinctSystems* et *lb*,

l'algorithme stocke respectivement $C(I)$, $S(I)$ (dans le cas 2., la place reste vide) et les solutions positives de $S(I)$. Il est possible d'appliquer les tests 1. et 2. aux colonnes suivantes de $M(S)$ (voir les instructions **9 :**).

À chaque itération de la boucle, en d'autres termes, pour chaque k-uplet I de signes, le Théorème IV.4.1 est appliqué pour obtenir les solutions réelles de (S) associées aux solutions réelles positives du système induit $S(I)$.

Les fonctions utilisées par l'algorithme `SolveFuzzySystem` sont celles de l'algorithme basique `BA-SFS` complétées par les fonctions suivantes :

— la fonction **SignColumn**(j, S) retourne la $(j+1)$-ième colonne de la matrice $M(S)$ des signes (j commence à 0 et non 1) ;

— la fonction **IsIn**(e , *Distinct*) renvoie -1 si e n'est pas dans le vecteur *Distinct*, sinon elle renvoie l'indice de la première occurrence de e dans *Distinct*. Cette fonction est appelée indifféremment sur les colonnes des signes $M(S)$ et sur les systèmes polynomiaux. Pour une recherche efficace sur un système polynomial, nous ordonnons les polynômes selon un ordre total sur les monômes en attribuant le signe $+$ au monôme dominant (les monômes d'un même polynôme sont ordonnés et les équations du système le sont également).

IV.5.2 Une version parallèle de l'algorithme `SolveFuzzySystem`

Les solutions positives de chaque système induit $S(I)$ sont calculées indépendamment de celles des autres systèmes induits flous (avec les fonctions **RealTransform** et **SolPos** successivement). Cette partie étant la plus coûteuse de l'algorithme `SFS`, sa parallélisation éventuelle est la bienvenue. Celle-ci nécessite de modifier l'algorithme afin d'identifier les systèmes induits distincts $S(I)$ à résoudre, mais sans effectuer la résolution et en stockant simultanément les informations utiles pour une application ultérieure du Théorème IV.4.1.

Pour atteindre cet objectif, nous modifions le rôle du vecteur *lb*. Il se retrouve indexé de 0 à $2^k - 1$, comme les colonnes de $M(S)$, et $lb[j]$ contient désormais l'indice dans *DistinctSystems* du système correspondant à la colonne **SignColumn**(j). Lorsque la résolution d'un système est interrompue, ses solutions positives sont stockées dans un nouveau vecteur *SPos*, avec les mêmes indices que *DistinctSystems*.

L'algorithme parallèle général se déroule en les trois étapes suivantes :

1. Détecter efficacement les systèmes distincts à résoudre en parallèle. De plus, pour chaque colonne I de $M(S)$, (et de façon équivalente, pour chaque j de 0 à $2^k - 1$), l'indice du premier système précédent égal à la transformation réelle de $S(I)$ est affecté à $lb[j]$ (il peut s'agir de j lui-même si $S(I)$ est nouveau). Les systèmes distincts sont stockés dans le vecteur *DistinctSystems*.

Algorithm 13 `SolveFuzzySystem` ou `SFS`, un algorithme optimisé pour résoudre les systèmes flous

Require: S, un système polynomial à coefficients dans $\mathfrak{F}(L, L)$ en représentation en tuple
k, la dimension, i.e. le nombre de variables
Data : $cpt := -1$, le compteur du nombre de colonnes distinctes de $M(S)$
$DistinctColumns := []$, les colonnes distinctes de $M(S)$ déjà parcourues
$DistinctSystems := []$, les systèmes distincts $S(I)$ déjà rencontrés
$lb := [] ; lb[i]$ est l'ensemble $\mathrm{Sol}^+(S(I))$ lorsque $C(I)$ est $DistinctColumns[i]$.
$sol := \{\ \}$
Ensure: sol, l'ensemble des solutions réelles de (S)

# l'indice j parcourt les colonnes de $M(S)$ indicées par **MultiSign**(j)
for $j := 0$ **to** $2^k - 1$ **do**
  $I :=$ **MultiSign**(j)
  $C :=$ **SignColumn**(j) # C est la colonne courante dans $M(S)$
  # teste si C est égal à une colonne précédente de $M(S)$
  $i :=$ **IsIn**$(\ C\ ,\ DistinctColumns)$
  if $i \neq -1$ **then** # le test 1. est satisfait
    # $lb[i]$ contient les solutions réelles positives de $S(I)$, appliquer le Théorème **IV.4.1** à $lb[i]$ et I
    $sol := sol \bigcup I \otimes lb[i]$
    # passe à la colonne suivante dans $M(S)$ et cpt n'est pas incrémenté
    go to 1 :
  end if
  # ici C est une nouvelle colonne
  $cpt := cpt + 1$
  $DistinctColumns[cpt] := C$
  # teste si S(I) est un nouveau système
  $i :=$ **IsIn**$(\ S(I)\ ,\ DistinctSystems\ )$
  if $i \neq -1$ **then** # le test 2. est satisfait
    # le i-ème système est égal à $S(I)$. Ses solutions réelles positives dans $lb[i]$ sont copiées dans $lb[cpt]$ car C est une "nouvelle" colonne
    $lb[cpt] := lb[i]$
  else
    # stocke le nouveau système à l'indice cpt, calcule ses solutions réelles positives et les stocke dans $lb[cpt]$
    $DistinctSystems[cpt] := S(I)$
    $TRSI :=$ **RealTransform**$(S(I))$
    $lb[cpt] :=$ **SolPos**$(TRSI)$
  end if
  # Appliquer le Théorème **IV.4.1**
  $sol := sol \bigcup I \otimes lb[cpt]$
end for
return sol

Ceci est réalisé dans le cadre de l'algorithme séquentiel SFS modifié de la manière suivante :

- dans **4 :** remplacer l'instruction $sol := sol \bigcup I \otimes lb[i]$ par $lb[j] := lb[i]$,
- dans **7 :** remplacer l'instruction $lb[cpt] := lb[i]$ par $lb[j] := lb[i]$,
- dans **9 :** remplacer les deux intructions $TRSI :=$ **RealTransform**$(S(I))$ et $lb[cpt] :=$ **SPos**$(TRSI)$ par $lb[j] := cpt$.

2. En parallèle, résoudre chaque système distinct du vecteur $DistinctSystems$. Pour chaque système $SI := DistinctSystems[cpt]$, s'il existe (voir Remarque IV.5.1 ci-dessous), on applique les instructions suivantes :
   $TRSI :=$ **RealTransform**(SI)
   $SPos[cpt] :=$ **SolPos**$(TRSI)$
3. Parcourir le vecteur lb pour construire toutes les solutions du système (S) à partir des résultats de l'étape précédente ; Il s'agit principalement d'appliquer le Théorème IV.4.1. Cette dernière étape peut également être effectuée en parallèle. Chaque $lb[j]$ possède la valeur de l'indice cpt des solutions réelles positives de $S(I)$ dans le vecteur $SPos$ où $I =$ **MultiSign**(j). Cette étape est réalisée par la boucle suivante :
   for $j := 0$ **to** $2^k - 1$ **do**
   $sol := sol \bigcup$ **MultiSign**$(j) \otimes SPos[lb[j]]$
   end for

Remarque IV.5.1. Comme dans l'algorithme SFS, quand $C = C(I)$ est une nouvelle colonne mais $S(I)$ n'est pas un nouveau système, rien n'est stocké à l'indice correspondant dans $DistinctSystems$.

Un exemple détaillé de cette version parallèle est donné à la Section IV.6.2.

IV.6 Implémentation de l'algorithme SFS et Exemples

Nous disposons du package Fuzzy écrit sous SageMath (voir [Mar17] et en Annexe B). Il contient une fonction resolution_reelle_systemes_flous qui implémente l'algorithme SFS de la Section IV.5. Rappelons que les nombres flous doivent appartenir à une famille $\mathfrak{F}(L, L)$ où L est bijective.

La Section IV.6.1 décrit la fonction resolution_reelle_systemes_flous. Ensuite, deux exemples complets sont donnés en Section IV.6.2. Le premier détaille les calculs intermédiaires de la fonction résolution_reelle_systemes_flous. Le second illustre sur un autre système flou les calculs effectués par la version parallèle de l'algorithme. Ces exemples initialement proposés dans [AOM08] ont été traités par plusieurs autres auteurs.

IV.6.1 Représentations et fonctions principales implémentées dans `Fuzzy`

La représentation externe des données dans le package `Fuzzy` est décrite ci-dessous :

- une fonction de dispersion H a une représentation rep(H) ; par exemple `"Quad"` si H est quadratique ;
- un nombre flou $\widetilde{n} = (n, a, b)$ avec les fonctions de dispersion L et R est représenté par :
$$\text{rep}(\widetilde{n}) \;=\; \texttt{NombreFlouRed}((n, a, b, \text{rep}(L), \text{rep}(R)) \;;$$
- un terme $\boldsymbol{x^d}$ est représenté classiquement par rep($\boldsymbol{x^d}$) $=$ $x_1 \texttt{**} d_1 \cdots x_k \texttt{**} d_k$;
- un monôme $m = \widetilde{n}\,\boldsymbol{x^d}$ est représenté par la paire rep(m) $=$ (rep($\widetilde{n}$), rep($\boldsymbol{x^d}$)) ;
- un polynôme p avec des coefficients flous est représenté par rep(p), la liste des rep(m) où m est un monôme de p ; c'est-à-dire une représentation creuse ;
- une équation polynomiale $(E) : p = q$ est représenté par la paire rep((E))=(rep(p), rep(q)) ;
- un système (S) formé par s équations $(E_1), \ldots, (E_s)$ est représenté par la liste
rep((S))=[rep((E_1)), ..., rep((E_s))].

Par exemple, avec des nombres flous quadratiques, le système

$$F : \begin{cases} x + (-1, 1, 1) = (-2, 1, 1)y^2, \\ x + (3, 1, 1) = (2, 1, 1)y^2 \end{cases} \tag{IV.14}$$

est représentée par la variable `System` définie comme suit :

```
LeftSide1 = [(NombreFlouRed(1,0,0,"Quad","Quad"),x),
            (NombreFlouRed(-1,1,1,"Quad","Quad"),1)]
RightSide1 = [(NombreFlouRed(-2,1,1,"Quad","Quad"), y**2) ]
LeftSide2 = [(NombreFlouRed(1,0,0,"Quad","Quad"),x),
            (NombreFlouRed(3,1,1,"Quad","Quad"),1)]
RightSide2 = [(NombreFlouRed(2,1,1,"Quad","Quad"),y**2) ]
System = [ (LeftSide1, RightSide1 ), (LeftSide2, RightSide2)]
```

Remarque IV.6.1. Les équations floues du système F n'ont pas exactement la même forme que l'équation générique (E) étudiée dans ce chapitre. La partie droite de (E) n'est pas restreinte à un nombre flou $\widetilde{m}$. Cependant, nos résultats s'étendent clairement à une telle forme d'équations.

La fonction `resolution_reelle_systemes_flous(S,`k`)` implémente l'algorithme séquentiel `SFS` de ce travail. Il prend comme premier paramètre la représentation d'un

système flou (S) à s équations. Le deuxième paramètre k est le nombre de variables. Il retourne l'ensemble des solutions réelles de (S) sous forme de k-uplets . Pour cela, il utilise principalement les fonctions `transformee_reelle()` et `SolPos()` décrites ci-dessous.

La fonction `transformee_reelle(S,rep(`I`))`, où rep(I) est une liste représentant un k-uplet de signes $I \in \{-1,1\}^k$, retourne la transformation réelle du système flou induit $S(I)$, c'est-à-dire le système à $3\,s$ équations à coefficients réels obtenu en transformant chacune de ses équations $E(I)$ comme énoncé dans le Corollaire IV.3.1. Il implémente notre fonction **RealTransform** des algorithmes des Sections IV.4 et IV.5.

La fonction `SolPos(Sr)` retourne l'ensemble des solutions réelles positives d'un système polynomial `Sr` à coefficients réels. La représentation de `Sr` est une liste $[p_1., \ldots, p_r]$ de polynômes tels que $p_i = 0$. Elle appelle d'abord la fonction `Wu(Sr)` qui implémente l'algorithme de Wu [Wu87]. Celle-ci retourne un ensemble Z d'ensembles polynomiaux appelés ensembles *caractéristiques* tels que la variété $V($`Sr`$)$ de zéros de `Sr` admet une décomposition en systèmes polynomiaux triangulaires de la forme $V($`Sr`$) = \bigcup_{B\in Z} V(B) \setminus V(I_B)$ où $I_B = \prod_{b\in B} init(b)$. On peut trouver la définition de l'initial $init(b)$ d'un polynôme b dans [CG90] ou dans [BBRV16], un article où la méthode de Wu est décrite afin de résoudre des systèmes polynomiaux avec des nombres triangulaires flous comme coefficients. À partir de l'ensemble Z, une fonction `get_zeros(`Z`)` retourne les éléments dans $\mathbb{R}^{+k}$ de la variété $V($`Sr`$)$.

IV.6.2 Exemples

Exemple 1 : Nous considérons le système F donné dans (IV.14). Comme dans l'exemple 6.1 de [BBRV16], l'appel `resolution_reelle_systemes_flous(F,2)` retourne la variété $V(F) = \{(x = -1, y \pm 1)\}$. Nous décrivons ici les calculs intermédiaires.

Avec $k = 2$ variables x, y, on a les $2^2 = 4$ multisignes suivants : $I_0 = [-1,-1], I_1 = [-1,1], I_2 = [1,-1], I_3 = [1,1]$. En les prenant dans cet ordre, correspondant respectivement à $j = 0,1,2,3$ dans l'algorithme `SFS`, et en ordonnant les monômes présents dans F comme suit : $x, 1, y^2$, la matrice des signes $M(F)$ est

$$M(F) = \begin{array}{c} \\ x \\ 1 \\ y^2 \end{array} \begin{array}{c} \begin{array}{cccc} I_0 & I_1 & I_2 & I_3 \end{array} \\ \begin{pmatrix} -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Notons que lors de l'exécution de l'algorithme ci-dessous, une colonne de $M(F)$ est représentée par un vecteur ligne.

La variable `Sr` est la transformation réelle du système flou induit $F(I)$ des $s = 2$ équations, avec $I \in \{I_0, \ldots, I_3\}$; en théorie `Sr` est formée par $3s = 6$ équations. Mais dans la pratique, certaines sont identiques. Par exemple, lorsque $I = [-1, -1]$, la transformation réelle de $F(I)$ est formée par les six équations $2y^2 - x - 1 = 0, -y^2 + 1 = 0, -y^2 + 1 = 0, -2y^2 - x + 3 = 0, -y^2 + 1 = 0, -y^2 + 1 = 0$, qui se réduit à `Sr`= $[-y^2 + 1, 2y^2 - x - 1, -2y^2 - x + 3]$ par la suppression des polynômes doublons. C'est le premier système polynomial réel qui est résolu dans le programme. Pour des raisons pratiques évidentes, ce système est celui enregistré dans la variable `DistinctSystems`, et non le système flou $F(I)$ comme mentionné dans l'algorithme `SFS`.

Présentons maintenant la trace de `resolution_reelle_systemes_flous(F,2)` en reliant les paramètres effectifs aux paramètres formels : `S=F` et `k=2`.
Au début, nous avons : `cpt = -1, DistinctColumns =[ ]` et `DistinctSystems =[ ]`.

```
j=0 : I=I_{0}=[-1,-1], C=[-1,1,1]
   Comme la colonne C est nouvelle
      cpt=cpt+1 ; i.e. cpt =0
      ColonnesDistinctes[0] = C
      Sr = transformee_reelle(S,I) donne Sr=[-y^2+1,2*y^2-x-1,-2*y^2-x+3].
      DistinctSystems[0] = Sr
      SolPos(Sr) donne lb[0]=set([(1, 1)]), les solutions réelles positives de Sr
      lb[0]= set([(1, 1)])
         le produit de I avec (1, 1) dans lb[0] est ajouté à sol :
      sol = set([(-1, -1)])

Nous avons
  ColonneDistincte=[[1,-1,1]]
  DistinctSystems=[[y^2-1,2*y^2-x-1,2*y^2+x-3]]

j=1 : I=I_{1}=[-1,1], C=[-1,1,1]
    Comme C est égal à ColonnesDistinctes[0].
         le produit de I avec (1, 1) dans lb[0] est ajouté à sol :
         sol = set([(-1, 1), (-1, -1, -1)])

j=2 : I=I_{2}=[1,-1], C=[1,1,1]
    Comme la colonne C est nouvelle
      cpt=cpt+1 ; i.e. cpt =1
      ColonnesDistinctes[1] = C
      Sr = transformee_reelle(S,I) donne Sr=[2*y^2+x-1,-2*y^2+x+3,-y^2+1].
      DistinctSystems[1] = Sr
      SolPos(Sr) donne lb[1]= set([]), les solutions réelles positives de Sr
```

```
            tout produit d'éléments de lb[1] avec I est ajouté à sol :
      sol = set([(-1, 1), (-1, -1, -1)])

Nous avons
  ColonneDistincte=[[-1,1,1],[1,1,1]]
  DistinctSystems=[[y^2-1,2*y^2-x-1,2*y^2+x-3],[2*y^2+x-1,2*y^2-x-3,y^2-1]]

j=3 : I=I_{3}=[1,1], C=[1,1,1]
   C est égal à ColonnesDistinctes[1]
       tout produit d'éléments de lb[1] avec I est ajouté à sol :
   sol = set([(-1, 1), (-1, -1, -1)])
```

Par conséquent, nous trouvons la variété solution $V(F) = \{(x = -1, y \pm 1)\}$.

Exemple 2 : Exemple détaillé de la version parallèle de l'algorithme SFS sur le système flou suivant F_1 :

$$F_1 : \begin{cases} (2,1,1)xy + (3,1,1)x^2y^2 + (2,1,1)x^3y^3 = (7,3,3), \\ (5,1,1)xy + (2,3,1)x^2y^2 + (2,2,1)x^3y^3 = (9,6,3). \end{cases}$$

La solution retournée par la fonction `resolution_reelle_systemes_flous(`F_1`,2)` du package `Fuzzy` est la variété $V(F_1) = \{(x = \frac{1}{y}, y) \mid y \in \mathbb{R}\backslash\{0\})\}$, comme dans [BBRV16]. Nous remarquons que dans la première équation (E), la dispersion à gauche de chaque coefficient est égale à la dispersion à droite. Ainsi, dans la transformation réelle de (E), les deux équations

$$\sum_{\boldsymbol{d}\in \mathrm{Expon}(E)} \alpha_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} = \alpha \quad \text{et} \quad \sum_{\boldsymbol{d}\in \mathrm{Expon}(E)} \beta_{\boldsymbol{d}}\, \boldsymbol{x}^{\boldsymbol{d}} = \beta$$

sont égales à $x^3y^3y^3 + x^2y^2 + xy = 3$. On retrouve cette équation avec les dispersions à droite de la seconde équation de F_1. C'est pourquoi nous avons toujours 4 et non $6 = 3s$ équations dans la transformation réelle de toute équation floue induite $F_1(I)$. Lors d'une implémentation, il est possible d'en tenir compte afin d'identifier une fois pour toutes les équations identiques dans le système flou initial, puis d'utiliser le résultat sur chacun de ses systèmes induits.

Nous trouvons $V(F_1)$ en appliquant la version parallèle de notre algorithme. Les multi-signes sont les mêmes que pour le système F du premier exemple et les monômes présents

dans F_1 sont $\{x^3y^3, x^2y^2, xy, 1\}$. La matrice des signes de F_1 est la suivante :

$$M(F_1) = \begin{array}{c} \\ x^3y^3 \\ x^2y^2 \\ xy \\ 1 \end{array} \begin{array}{c} \begin{array}{cccc} I_0 & I_1 & I_2 & I_3 \end{array} \\ \begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Nous voyons qu'il suffit de résoudre uniquement les systèmes correspondant aux deux premières colonnes ; c'est-à-dire que nous avons les identités suivantes sur les systèmes flous induits : $F_1(I_0) = F_1(I_3)$ et $F_1(I_1) = F_1(I_2)$. Dans l'exécution de l'algorithme, cette information se retrouve dans le vecteur `lb`. Ainsi, le nombre de systèmes à résoudre est réduit de moitié.

Dans la première étape de l'algorithme parallèle, après les boucles `j= 0` et `j=1`, nous avons :

```
DistinctColumns = [[1, 1, 1, 1], [-1, 1, -1, 1]]
DistinctSystems=
    [[2*x^3*y^3+2*x^2*y^2+5*x*y-9,2*x^3*y^3+3*x^2*y^2+2*x*y-7,
                        2*x^3*y^3+3*x^2*y^2+x*y-6, x^3*y^3+x^2*y^2+x*y-3],
     [2*x^3*y^3-3*x^2*y^2+2*x*y+7,2*x^3*y^3-3*x^2*y^2+x*y+6,
                    x^3*y^3-x^2*y^2+x*y+3, 2*x^3*y^3-2*x^2*y^2+5*x*y+9]]
lb = [0,1]
```

Après les boucles `j=2` et `j=3`, seul le vecteur `lb` est modifié comme suit : `lb=[0,1,1,0]`.

Comme `lb[2]=lb[1]` les solutions réelles positives du troisième système induit $F_1(I_2)$ correspondant à `lb[2]` sont celles du deuxième système induit $F_1(I_1)$ correspondant à `lb[1]` ; après la deuxième étape, ces solutions réelles positives se trouvent dans `SPos[1]`. Il en va de même avec `lb[3]=lb[0]`.

Dans la deuxième étape, la fonction `SolPos` est appelée en parallèle sur chacun des deux systèmes stockés dans `DistinctSystems` afin de calculer leurs solutions réelles positives respectives. Ces solutions sont récupérées et stockées dans le vecteur `SPos`. Nous avons alors :

`SPos=[set([(1/y,'R+')]),set([])]`.

La dernière étape calcule en parallèle pour chaque `j = 0,1,2,3` les quatre produits `I` $\otimes$ `SPos[lb[j]]` correspondant à la dernière colonne "`returned solutions`" ci-dessous :

```
j       I            lb[j]      SPos[j]              returned solutions
0    [-1,-1]         0       set([(1/y, 'R+')])        set([(1/y, 'R-')])
1    [-1,1]          1       set([])                   set([])
```

```
2    [1,-1]          1       SPos[1]                  set([])
3    [1,1]           0       SPos[0]                  set([(1/y, 'R+')])
```

Les solutions non vides sont ajoutées dans la variable `sol` contenant les solutions réelles du système flou F_1 :

```
sol =  set([(1/y, 'R+'), (1/y, 'R-')])
```

Nous trouvons bien la variété $V(F_1) = \{(x = \frac{1}{y}, y) \mid y \in \mathbb{R}\backslash\{0\})\}$. Comme dans la version séquentielle, seuls deux systèmes distincts sont résolus.

IV.7 Conclusion

Pour résoudre un système flou (S) de s équations et k variables, jusqu'à présent les méthodes algébriques existantes exécutaient des calculs avec la représentation paramétrique des coefficients pour obtenir la forme tranchée collectée de (S) composée de $4s$ équations réelles. Nous montrons que ces calculs sont superflus en exhibant une formule qui définit un système équivalent à $3s$ équations réelles, que nous appelons la transformation réelle $\mathcal{T}(S)$ du système (S). Elle possède comme propriété principale d'avoir les mêmes solutions réelles positives que (S) (Théorème IV.3.4) et a donné lieu à une publication à la conférence Fuzzy Computation Theory and Applications (FCTA) de la $11^{ème}$ International Joint Conference on Computational Intelligence (IJCCI).

Alors que les méthodes antérieures ne traitaient que les systèmes à coefficients flous triangulaires, nos résultats s'appliquent à toute famille $\mathfrak{F}(L, R)$ où les fonctions de dispersion L et R sont bijectives. De plus, il est inutile de calculer l'inverse des fonctions de dispersion puisque la transformation réelle est une formule universelle indépendante de L et R. Une adaptation de ces résultats aux nombres flous trapézoïdaux a été donnée.

Pour résoudre des équations avec des coefficients flous symétriques appartenant à une même famille $\mathfrak{F}(L, L)$, il faut se poser la question du signe des solutions. Cette question est intrinsèque aux nombres flous et provient du fait que la multiplication d'un nombre flou par un scalaire réel s'exprime différemment selon le signe de ce scalaire. Notre stratégie a consisté à nous concentrer sur les solutions positives en reportant a priori sur les coefficients flous ce problème de signes. Le Théorème IV.4.1 a rendu possible cette stratégie puisqu'il exprime l'ensemble des solutions réelles de (S) à partir des solutions positives d'au plus 2^k systèmes réels. À partir de ce théorème, nous concevons un premier algorithme qui automatise la recherche de solutions de (S) en évitant les études de signes indispensables avec les méthodes antérieures. Notre approche est tout à fait indépendante du choix de la méthode employée pour calculer les solutions positives d'un système de polynômes à coefficients réels.

Parmi les 2^k systèmes induits de (S), certains sont identiques. Nos exemples montrent qu'il n'est pas rare de réduire substantiellement le nombre de systèmes induits à résoudre. Nous décrivons une stratégie pour éviter des branches de calcul redondantes qui conduit à un algorithme optimisé, `SolvingFuzzySystem`, implémenté dans la bibliothèque `Fuzzy` du logiciel de calcul formel `SageMath` (voir Annexe B). Ce travail a mené à la publication d'un article dans la revue Fuzzy Sets and Systems [AMV20].

La partie coûteuse de cet algorithme réside dans la recherche des solutions positives de la transformation réelle des systèmes induits distincts de (S). Nous décrivons dans la Section IV.5.2 une parallélisation de l'algorithme `SolvingFuzzySystem` permettant d'exécuter ces calculs mutuellement indépendants en parallèle, et nous en illustrons le fonctionnement sur le second exemple de la Section IV.6.

Conclusion

L'arithmétique modulaire est l'élément central de la majorité des protocoles de chiffrement en cryptographie asymétrique, et sur lequel repose leur sécurité. Pour assurer une arithmétique modulaire efficace, Knuth est le premier à proposer une classe de moduli particuliers dès les années 80. Bien que de nombreuses améliorations aient été effectuées depuis pour étendre l'ensemble des moduli exploitables, la construction de systèmes permettant une arithmétique efficace pour un large choix de moduli restait difficilement réalisable en pratique au début de cette thèse, limitée par leur dépendance à la forme des moduli utilisés, ou bien par des paramètres trop restrictifs. Dans cette thèse, nous nous sommes concentré sur le besoin de proposer de nombreuses bases de systèmes pour de nombreux premiers, en s'affranchissant de la forme du premier tout en garantissant une arithmétique efficace, et de rendre cette dernière sûre face aux attaques par canaux cachés. En parallèle de ce travail, nous proposons une nouvelle méthode dans le domaine de la modélisation incertaine afin d'optimiser la résolution réelle de systèmes polynomiaux flous.

Une amélioration des algorithmes généralistes de multiplication modulaire de Montgomery et Barrett est proposée par Thomas Plantard en 2004, avec le système de représentation adapté polynomial (PMNS) permettant l'implémentation d'une arithmétique modulaire efficace impliquant uniquement des additions et des multiplications. Cependant, la construction de ces systèmes est limitée par les paramètres restrictifs du théorème d'existence et la recherche de bases viables n'est pas triviale, conduisant à peu de systèmes en pratique. Dans cette thèse, nous avons proposé un théorème général d'existence, en garantissant une borne sur les chiffres, qui peut être calculée simplement à partir de la norme 1 du réseau euclidien associé. Ce théorème nous a conduit à considérer les systèmes PMNS définis par un polynôme irréductible pour lesquels définir une base du réseau est aisé. Ces théorèmes, propositions et corollaires nous ont permis de produire des PMNS avec des polynômes de réduction spécifiques permettant des réductions efficaces et dont les racines fournissent les bases de ces systèmes. Nous avons à présent la possibilité d'offrir pour un modulo premier donné une grande variété de PMNS.

Le système PMNS peut devenir très redondant quand la taille des chiffres augmente, laissant suggérer une possible exploitation de cette redondance afin de changer aléatoirement de représentation au cours des calculs, au sein d'une même exécution, pour empêcher tout hypothèse de la part d'un attaquant sur les représentations utilisées. Dans le deuxième chapitre, nous avons montré comment utiliser cette redoncance pour construire des protections arithmétiques efficaces contre les attaques DPA, en randomisant les données en entrée lors de la conversion vers le PMNS via deux méthodes, et en introduisant deux multiplications modulaires randomisées sur le PMNS. Ces méthodes peuvent être utilisées pour appliquer des contre-mesures classiques sur la multiplication scalaire $k\mathcal{P}$ sur les courbes elliptiques. Contrairement à certaines contre-mesures existantes nécessitant une ECSM supplémentaire $k\mathcal{R}$ pour un point aléatoire $\mathcal{R}$, nous avons montré que la randomisation du processus de conversion suffit à elle seule à protéger les données contre l'attaque de Goubin. Ces résultats constituent une première étape dans l'utilisation de la randomisation des opérations arithmétiques sur le PMNS et ouvrent de nouvelles perspectives dans le domaine des contre-mesures aux attaques SCA.

Bien qu'il soit possible d'implémenter l'addition et de la multiplication de façon efficaces via le système RNS, ce système se prête moins aux réductions modulaires. L'approche traditionnelle de Montgomery sur ces systèmes exige de revenir dans la représentation classique, une étape impliquant de grandes extensions de bases qui augmentent le coût de la réduction. Le système hybride HPR proposé par Bigou et Tisserand rend les algorithmes de multiplication avec une complexité en temps sous-quadratique viables pour ECC, mais reste limité à des premiers d'une forme spéciale, empêchant d'étendre cette approche aux opérations de groupe sur des courbes elliptiques actuellement standardisées. Dans la troisième contribution, nous avons proposé un nouveau système hybride HyPoRes qui, via un affaiblissement des hypothèses sous-jacentes, atteint une complexité en temps sous-quadratique similaire, et supporte n'importe quelle valeur première, ce qui le rend compatible avec les courbes elliptiques standardisées. Il en résulte une accélération pouvant atteindre 1,4 lorsqu'il est comparé aux approches basées sur RNS pour des nombres premiers des standards ECC. La complexité des extensions de base se trouve réduite par rapport à une approche purement RNS, rendant HyPoRes plus adapté au parallélisme à une plus petite échelle. La possibilité de généraliser le polynôme de réduction permet également d'introduire des représentations redondantes, obtenues par des changements de systèmes, procurant une protection contre les attaques SCA.

Dans le domaine de la modélisation incertaine, de nombreuses méthodes locales sont utilisées pour résoudre des systèmes à coefficients flous. Étant donné un système flou de s équations et k indéterminées, les méthodes algébriques globales existantes effectuent

des calculs avec la représentation paramétrique des coefficients en se limitant au cas des nombres flous triangulaires. Dans le dernier chapitre, nous avons montré que ces calculs sont superflus et avons présenté une formule, la transformation réelle, qui définit un système équivalent avec moins d'équations et possédant les mêmes solutions positives que le système de départ. Cette méthode peut s'appliquer à n'importe quelle famille de nombre flous $L - R$, et ce, sans avoir à calculer l'inverse des fonctions de dispersions L et R. Afin d'obtenir les solutions réelles d'un système à coefficients flous symétriques, nous avons fourni un algorithme optimisé qui automatise la recherche de solutions en évitant les études des signes nécessaires dans les méthodes précédentes et avons décrit une parallélisation de cet algorithme.

Publications

1. On Polynomial Modular Number Systems over $\mathbb{Z}/p\mathbb{Z}$,

   avec Jean-Claude Bajard, Pascal Véron et Thomas Plantard, soumis, 2020.

2. Randomization of arithmetic over Polynomial Modular Number System,

   avec Laurent-Stéphane Didier, Fangan-Yssouf Dosso, Nadia El Mrabet et Pascal Véron,
   26th IEEE International Symposium on Computer Arithmetic (ARITH-26), 2019.

3. HyPoRes : An Hybrid Representation System for ECC,

   avec Paulo Martins, Jean-Claude Bajard et Leonel Sousa,
   26th IEEE International Symposium on Computer Arithmetic (ARITH-26), 2019.

4. The real transform : Computing positive solutions of fuzzy polynomial systems,

   avec Philippe Aubry et Annick Valibouze,
   11th International Joint Conference on Computational Intelligence (IJCCI), 2019.

5. Computing real solutions of fuzzy polynomial systems,

   avec Philippe Aubry et Annick Valibouze,
   Fuzzy Sets and Systems, 2020.

Annexe A

Construction de PMNS et normes

Exemple A.0.1. Construction de 38 systèmes PMNS pour un premier sur 256 bits
$p = 57896044618658097711785492504343953926634992332820282019728792003956566811073$,
$n = 8$ et $\|E(X)\|_1 \leq 1$.
Dans cet exemple, pour un premier p fixé, nous décrivons le nombre de systèmes contructibles à partir de nos classes de polynômes irréductibles, et donnons pour chacun d'eux la borne sur les chiffres en fonction de la base utilisée dans le calcul de la majoration.

Systèmes obtenus à partir de `ClassCyclo(8)` :

► **Nous obtenons** 8 **systèmes à partir de** $X^8 + 1$ **et des racines suivantes :**

— 52818609237435594463690459959484136748225065326409518886622180278882883124379
— 46315455537225662701141288508287261834557155748741463271299445489880439213302
— 42376218727930553321757239285528470365939306462672670785163952383551129520660
— 36047555367693598423871824123063010900898267680890473144205935496444119276882
— 21848489250964499287913668381280943025736724651929808875522856507512447534191
— 15519825890727544390028253218815483560695685870147611234564839620405437290413
— 11580589081432435010644203996056692092077836584078818748429346514076127597771
— 50774353812225032480950325448598171784099270064107631331066117250736836866694

où chaque système satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + 1$	4735529113.5	4735529113.5	4735529113.5	4735529113.5	4735529113.5

▶ **Nous obtenons 8 systèmes à partir de $X^8 - X^4 + 1$ et des racines suivantes :**

— 444006578676198869371887116189923515433740164927571437358822030464018192140 68

— 134953867510382107745967808853516023832609758400631382838465889575547475970 05

qui satisfont tous les deux

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^4 + 1$	6722646939.5	6722646939.5	6894456814	6236219964.5	6236219964.5

— 405987702628549127720410901864313921764813160821077054273344681131130597144 93

— 172972743558031849397444023179125617501536762507125765923943238908435070965 80

qui satisfont tous les deux

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^4 + 1$	5273608652	5273608652	5684931070	5684931070	5684931070

— 398396270355910696888712465334588994287154030004521447520149671243523712726 46

— 180564175830670280229142459708850544979195893323681372677138248796041955384 27

qui satisfont tous les deux

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^4 + 1$	5195123262.5	5195123262.5	6755855835.5	6684641449.5	6684641449.5

— 367142022173678025957960681821785933631834248468910004121478020253370332101 01

— 211818424012902951159894243221653605634515674859292816075809899786195336009 72

qui satisfont tous les deux

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^4 + 1$	6475848368	6475848368.0	6606088651	6684641449.5	6684641449.5

Systèmes obtenus à partir de `TrinomialClass(8)` :

▶ **Nous obtenons 2 systèmes à partir de $X^8 + X^2 - 1$ et des racines suivantes :**

— 44582051656053013167989761859406033059354424105701596615619354088632683329441

— 13313992962605084543795730644937920867280568227118685404109437915323883481632

où chaque système satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + X^2 - 1$	5258063115	5258063115	6178332792	6178332792	6178332792

▶ **Nous obtenons 2 systèmes à partir de $X^8 - X^2 - 1$ et des racines suivantes :**

— 353701556705986148646579502905446263661628470298061192231373713434838338686 65

— 22525888948059482847127542213799327560472145303014162796591420660472732942408

qui satisfont tous les deux

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^2 - 1$	6009904930	6009904930	8206098942.5	8469257863	8469257863

▶ **Nous obtenons 4 systèmes à partir de $X^8 + X^3 - 1$ et des racines suivantes :**

— 43997453786947348529790827133316902895351440438169700909565215527663162723526

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + X^3 - 1$	5167087144.5	5167087144.5	5167087144.5	5167087144.5	5167087144.5

— 40999693908757945794387931767478954756186658977527670083978362890400572114760

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + X^3 - 1$	6531907633	6531907633	6324138231.5	6324138231.5	6324138231.5

— 375155237221877868316522798514388420177781454783272778307522516080686624646 79

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^8+X^3-1	6775675061	6775675061	6784596631.5	6808409971	6808409971

— 344719171794150025287841627127972488369782709346923926008874054629492680567 17

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^8+X^3-1	7714141361	7714141361	6688700366	6688700366	6688700366

► **Nous obtenons** 2 **systèmes à partir de** X^8+X^3+1 **et des racines suivantes :**

— 549588129494991890464089121395792779038373760494233311590036246444290777327 65

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^8+X^3+1	6151661217	6151661217	6225488093	6225488093	6225488093

— 165709670888410407318993921077910015804068383263841396856813186808990304326 49

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^8+X^3+1	6616862362.5	6616862362.5	6802019395	6509777056	6509777056

► **Nous obtenons** 4 **systèmes à partir de** X^8-X^3-1 **et des racines suivantes :**

— 234241274392430951830013297915467050896567213981278894188413865410072987543 56

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
X^8-X^3-1	7714141361	7714141361	6688700366	6688700366	6688700366

— 203805208964703108801332126529051119088568468544930041889765403958879043463 94

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^3 - 1$	6775675061	6775675061	6784596631.5	6808409971	6808409971

— 168963507099001519173975607368649991704483333552926119357504291135559946963 13

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^3 - 1$	6531907633	6531907633	6324138231.5	6324138231.5	6324138231.5

— 138985908317107491819946653710270510312835518946505811101635764762934040875 47

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^3 - 1$	5167087144.5	5167087144.5	5167087144.5	5167087144.5	5167087144.5

► **Nous obtenons 2 systèmes à partir de $X^8 - X^3 + 1$ et des racines suivantes :**

— 413250775298170569798861003965529523462281540064361423340474733230575363784 24

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^3 + 1$	6616862362.5	6616862362.5	6802019395	6509777056	6509777056

— 293723166915890866537658036476467602279761628339695086072516735952748907830 8

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 - X^3 + 1$	6151661217	6151661217	6225488093	6225488093	6225488093

Systèmes obtenus à partir de `QuadrinomialClass(8)` :

▶ **Nous obtenons 1 système à partir de $X^8 + X^2 + X + 1$ et de la racine suivante :**

— 418205177121105737302723723634547915274250725923933547849720519022258849015 08

et 1 système à partir de $X^8 + X^2 - X + 1$ et de la racine suivante :

— 160755269065475239815131201408891623992099197404269272347567401017306819095 65

qui satisfont tous les deux

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + X^2 \pm X + 1$	6758830414.5	6758830414.5	6918797903	6918797903	6918797903

▶ **Nous obtenons 2 systèmes à partir de $X^8 + X^4 - X^3 + 1$ et des racines suivantes :**

— 38677869801997893370801143168900060768812782758356142077839390422580983571771

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + X^4 - X^3 + 1$	8455439312	8455439312	7759250111.5	6952404457	6952404457

— 53462548971576404293339781025117581398699407255035974344079794025761020933706

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + X^4 - X^3 + 1$	6583114770	6583114770	6490060412.5	6490060412.5	6490060412.5

▶ **Nous obtenons 2 systèmes à partir de $X^8 + X^4 + X^3 + 1$ et des racines suivantes :**

— 19218174816660204340984349335443893157822095744641399418894015813755832393 02

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + X^4 + X^3 + 1$	8455439312	8455439312	7759250111.5	6952404457	6952404457

— 443349564708169341844571147922637252793558507778430767564899797819554587736 7

qui satisfait

$E(X)$	$\frac{1}{2}\|\mathbf{B}\|_1$	$\frac{1}{2}\|\mathbf{D}\|_1$	$\frac{1}{2}\|\mathrm{LLL}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{BKZ}(\mathbf{A})\|_1$	$\frac{1}{2}\|\mathrm{HKZ}(\mathbf{A})\|_1$
$X^8 + X^4 + X^3 + 1$	6583114770	6583114770	6490060412.5	6490060412.5	6490060412.5

Annexe B

Documentation

Bibliothèque Fuzzy en SageMath

Jérémy Marrez

SageMath est un logiciel libre de mathématiques sous licence GPL, permettant entre autres d'effectuer du calcul formel, du calcul numérique, de résoudre des équations et de tracer des courbes. Il est basé sur le langage de programmation Python.

La bibliothèque Fuzzy 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 ou réels [Mar19].

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

B.1 Les classes des nombres flous

B.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 1-coupe, aussi appelée noyau, n'est pas forcément réduite à un point.

Le constructeur est `NombreFlou(modeGauche, modeDroit, dispersionGauche, dispersionDroite, typeGauche, typeDroit)`.

Une instance de cette classe possède les six propriétés suivantes :

— modeGauche, ou leftMode (un entier) : la borne gauche du noyau du nombre flou

— modeDroit, ou rightMode (un entier) : la borne droite du noyau du nombre flou

— dispersionGauche, ou leftSpread (un entier positif ou nul) : la propagation du nombre flou à gauche du noyau

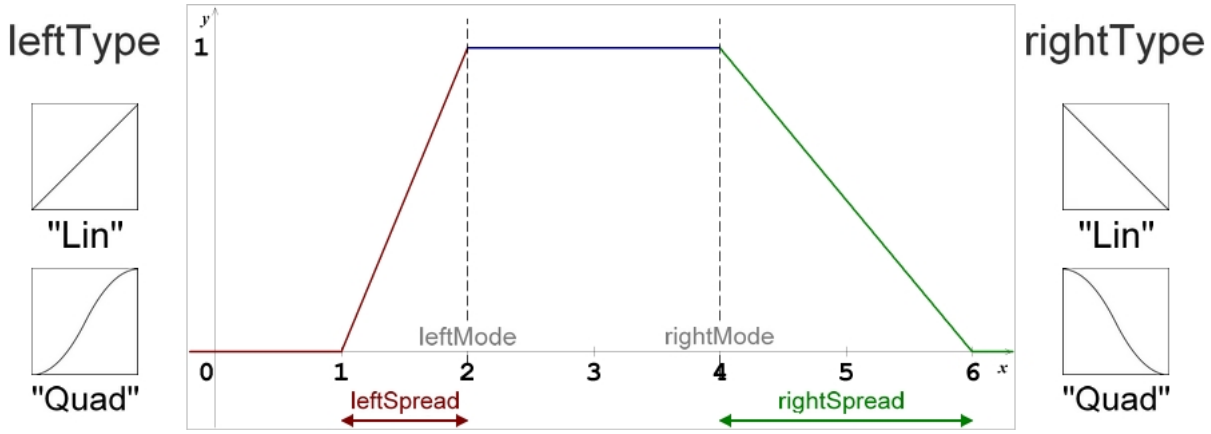
— dispersionDroite, ou rightSpread (un entier positif ou nul) : la propagation du nombre flou à droite du noyau

— typeGauche, ou leftType (une chaîne de caractères) : le type de la restriction de la fontion d'appartenance à gauche du noyau

— typeDroit, ou 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 réfèrencés par abus de langage comme des nombres flous trapézoïdaux.

Une instance peut étre 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'effectuer 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 (modeGauche, modeDroit, dispersionGauche, dispersionDroite) et précise sa famille en fonction du type de ses restrictions.

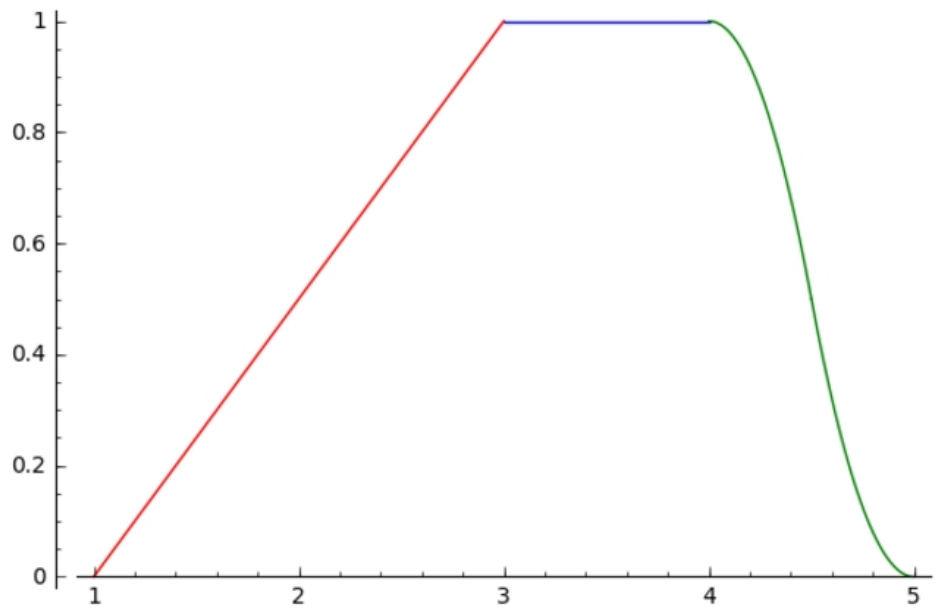
```
print(n1)
```

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

Les fonctions `get_mode(self)`, `get_dispersionGauche(self)` et `get_dispersionDroite(self)` renvoient respectivement le mode, le dispersionGauche et le dispersionDroite du nombre flou.

La fonction `trace(self)` 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 s'effectuant normalement sur le mode sont réalisés sur un intervalle. Pour ce faire, nous utilisons l'arithmétique des intervalles explicitée dans [TR11] 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 effectuent les calculs sur les instances de la classe. 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.

```
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
```

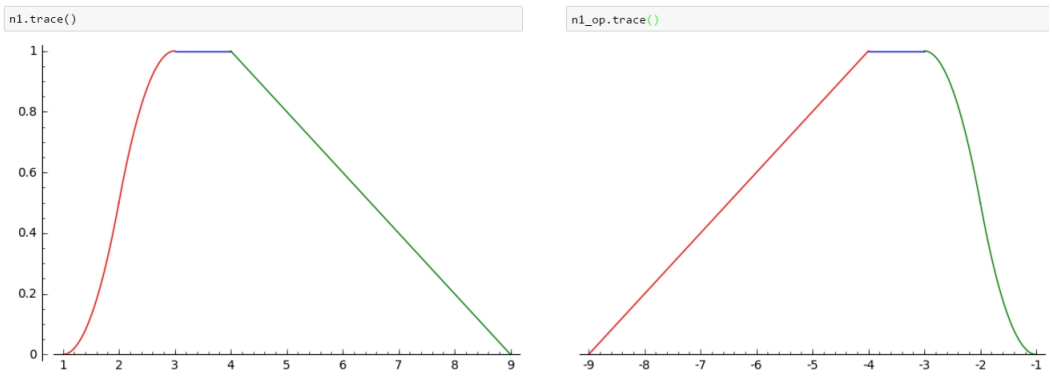
```
n1_inv = ~n1
print(n1_inv)
```

```
(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 effectuent les calculs sur les instances de la classe. L'opérateur - calcule l'opposé de l'opérande de droite et l'additionne avec l'opérande de gauche (mème 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, 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 si toutes les propriétés des deux opérandes sont identiques.

```
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 $\widetilde{n}$ dans sa forme paramétrique $[\underline{n}, \overline{n}]$ avec

$$\begin{cases} \underline{n}(r) = n - \alpha\, L^{-1}(r) \\ \overline{n}(r) = n + \beta\, R^{-1}(r) \end{cases},$$

où α et β sont respectivement les dispersions gauche et droite de $\widetilde{n}$. L et R sont les fonctions de dispersion de $\widetilde{n}$, qui définissent le type des restrictions de sa fonction d'appartenance.

Pour ce faire, la fonction calcule la fonction inverse en r de chaque restriction (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 sous la forme d'une chaîne de caractère "typeGauche - typeDroit", 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
```

B.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 1-coupe, ou noyau, est réduite à un point qui est leur mode.

Le constructeur est `NombreFlouRed(mode, dispersionGauche, dispersionDroite, typeGauche, typeDroit)`.

Une instance de cette classe possède les cinq propriétés suivantes :

— mode (un entier) : le mode du nombre flou

— dispersionGauche (un entier positif ou nul) : la propagation du nombre flou à gauche du noyau

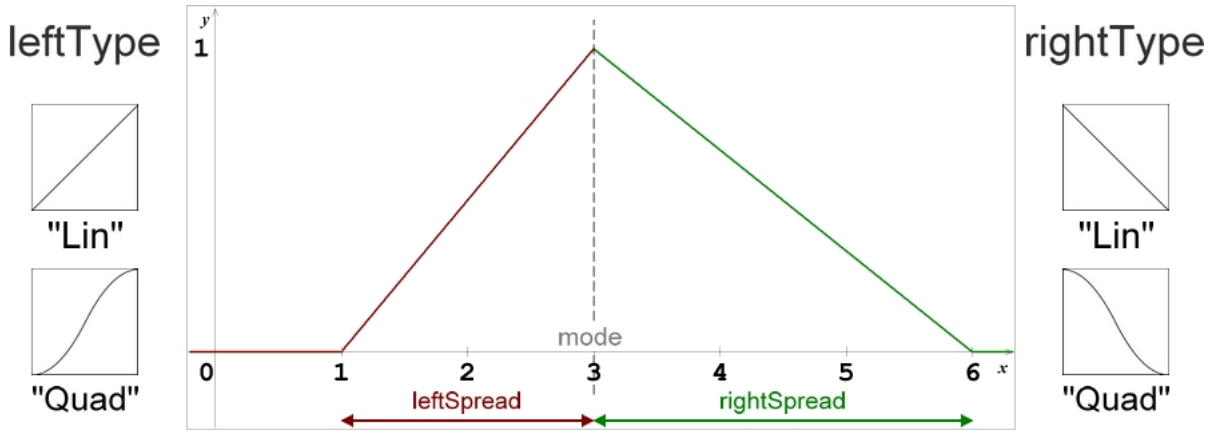
— dispersionDroite (un entier positif ou nul) : la propagation du nombre flou à droite du noyau

— typeGauche (une chaîne de caractères) : le type de la restriction de la fontion d'appartenance à gauche du noyau

— typeDroit (une chaîne de caractères) : le type de la restriction de la fontion d'appartenance à droite du noyau

On a donc ici modeGauche = modeDroit = 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, dispersionGauche, dispersionDroite) et précise sa famille.

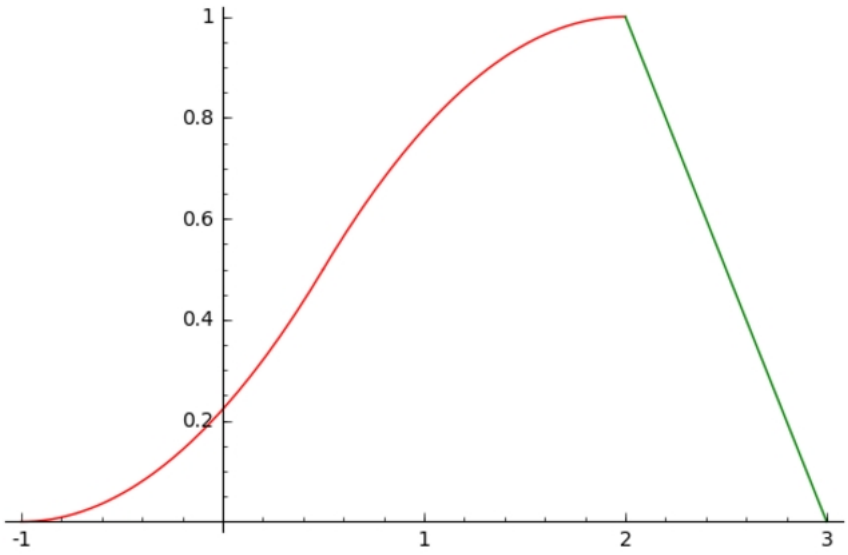
Les redéfinitions des opérations unaires - et $\sim$ sont aussi redéfinies pour prendre en compte le fait que le noyau est unique, et renvoient bien une instance de la classe `NombreFlouRed`.

Il en va de même avec les opérations binaires + et * dont les calculs sont exactement ceux vus en IV.2.3 avec le noyau réduit à un élèment.

```
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
```

B.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 r-coupe 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 r-coupe 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 des 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.

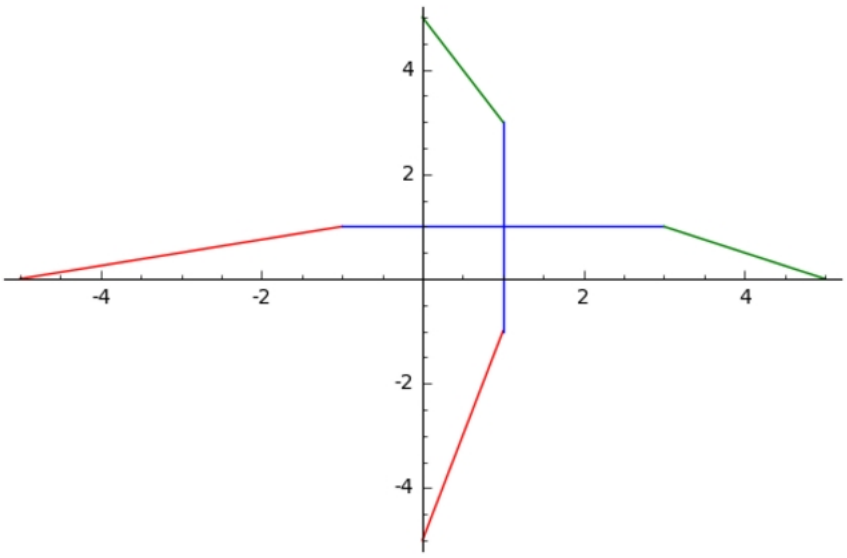
Dans cette classe, les différentes méthodes permettent à la fois d'afficher et de dessiner les graphes de ses instances, d'effectuer des calculs sur ces instances et de les ordonner.

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 90 degrés par rapport à l'origine suivie d'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 "typeGauche - typeDroit", la liste de ses types [typeGauche, typeDroit]. 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 IV.2.2 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
```

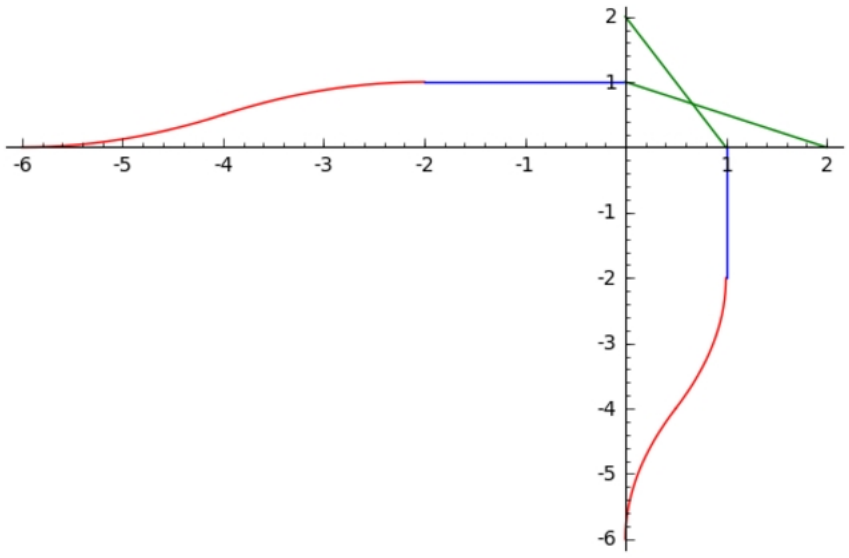
```
p3 = p1 - p2
print(p3)
```

```
[[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 [EKZ15]. 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 :

— $R(a,\lambda) > R(b,\lambda) \Leftrightarrow a > b$,

— $R(a,\lambda) < R(b,\lambda) \Leftrightarrow a < b$,

— $R(a,\lambda) = R(b,\lambda) \Leftrightarrow a \sim b$.

B.2 Méthodes de résolution algébrique

Les méthodes listées ci-après sont celles implantées pour la résolution algébrique de systèmes de polynômes réels et 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

B.3 Algorithme principal

L'utilisateur entre son système polynomial à coefficients flous sous la forme d'une liste de couples :

$$\begin{aligned} System = [&(\text{membre de gauche equation } 1, \text{membre de droite equation } 1),\\ &(\text{membre de gauche equation } 2, \text{membre de droite equation } 2),\\ &\vdots\\ &(\text{membre de gauche equation } s, \text{membre de droite equation } s)] \end{aligned}$$

où chaque membre d'une équation est également donné par une liste de couples :

membre d'une equation $=$ [(Coefficient flou monôme 1, terme monôme 1), ..., (Coefficient flou monôme m, terme monôme m)]

Par exemple, pour le système

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

on écrit

```
systeme = [([(NombreFlouRed(1,0,0,"Quad","Quad"),x), (NombreFlouRed(-1,1,1,"Quad","Quad"),1)],
[(NombreFlouRed(-2,1,1,"Quad","Quad"),y**2)]),
([(NombreFlouRed(1,0,0,"Quad","Quad"),x), (NombreFlouRed(3,1,1,"Quad","Quad"),1)],
[(NombreFlouRed(2,1,1,"Quad","Quad"),y**2)])]
```

`resolution_reelle_systemes_flous(Systeme, n)` : Renvoie, à partir d'un système flou `Systeme` à s équations, donné comme ci-dessus, et d'un nombre de variables n, l'ensemble des solutions réelles exactes de `Systeme`. Pour cela, elle utilise principalement les fonctions suivantes.

`RealTransform(dep,I)` : À partir d'un système flou `dep` à s équations donné

comme ci-dessus et d'une liste de signes `I` $\in \{-1, 1\}^n$, construit le système `SI` à coefficients réels à $3s$ équations.

`SolPos(SI)` : Renvoie à partir d'un système `SI` à coefficients réels à $3s$ équations, l'ensemble des solutions positives de `SI`. Elle fait appel pour cela à la fonction `Wu(systeme)` qui implante l'algorithme de Wu, i.e. renvoie l'ensemble Z des ensembles caractéristiques B du système réel passé en paramètre, puis à la fonction `get_zeros(`Z`)` qui à partir de l'ensemble Z des ensembles caractéristiques renvoie les zéros de la variété $V(\texttt{SI}) = \bigcup_{B \in Z} V(B/I_B)$ où $I_B = \prod_{b \in B} init(b)$, i.e. les solutions positives de `SI`.

B.4 Tests

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

B.4.1 Exemple 1 :

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

```
systeme =  [([(NombreFlouRed(1,0,0,"Quad","Quad"),x), (NombreFlouRed(-1,1,1,"Quad","Quad"),1)], [(NombreFlouRed(-2,1,1,"Quad","Quad"),y**2)]),\
    ([(NombreFlouRed(1,0,0,"Quad","Quad"),x), (NombreFlouRed(3,1,1,"Quad","Quad"),1)], [(NombreFlouRed(2,1,1,"Quad","Quad"),y**2)])]
```

```
resolution_reelle_systemes_flous(systeme2,2)
```

```
Pour j =  0  :  I =  [-1, -1] ,  x négatif  et  y négatif
ColonnesDistinctes =  []
ListeSystemes =  []
C =  [-1, 1, 1, -1, 1, 1]
La colonne C est une nouvelle colonne, cpt est incrémenté, cpt =  0  et on ajoute C à ColonnesDistinctes à l'indice  0
S(I) = [-y^2 + 1, 2*y^2 - x - 1, -2*y^2 - x + 3]
Le système S(I) est un nouveau système, on l'ajoute à ListeSystemes à l'indice  0
On résout S(I)
Solutions positives de S(I) : set([(1, 1)])
On insère ces solutions dans lb à  l'incide 0
On ajoute dans sol le produit de Kronecker de I =  [-1, -1]  avec lb[ 0 ]
sol =  set([(-1, -1)])


Pour j =  1  :  I =  [-1, 1] ,  x négatif  et  y positif
ColonnesDistinctes =  [[-1, 1, 1, -1, 1, 1]]
ListeSystemes =  [[y^2 - 1, 2*y^2 - x - 1, 2*y^2 + x - 3]]
C =  [-1, 1, 1, -1, 1, 1]
La colonne C est déjà présente dans ColonnesDistinctes à l'indice  0
On ajoute dans sol le produit de Kronecker de I =  [-1, 1]  avec lb[ 0 ]
sol =  set([(-1, 1), (-1, -1)])


Pour j =  2  :  I =  [1, -1] ,  x positif  et  y négatif
ColonnesDistinctes =  [[-1, 1, 1, -1, 1, 1]]
ListeSystemes =  [[y^2 - 1, 2*y^2 - x - 1, 2*y^2 + x - 3]]
C =  [1, 1, 1, 1, 1, 1]
La colonne C est une nouvelle colonne, cpt est incrémenté, cpt =  1  et on ajoute C à ColonnesDistinctes à l'indice  1
S(I) = [2*y^2 + x - 1, -2*y^2 + x + 3, -y^2 + 1]
Le système S(I) est un nouveau système, on l'ajoute à ListeSystemes à l'indice  1
On résout S(I)
Solutions positives de S(I) : set([])
On insère ces solutions dans lb à  l'incide 1
On ajoute dans sol le produit de Kronecker de I =  [1, -1]  avec lb[ 1 ]
sol =  set([(-1, 1), (-1, -1)])


Pour j =  3  :  I =  [1, 1] ,  x positif  et  y positif
ColonnesDistinctes =  [[-1, 1, 1, -1, 1, 1], [1, 1, 1, 1, 1, 1]]
ListeSystemes =  [[y^2 - 1, 2*y^2 - x - 1, 2*y^2 + x - 3], [2*y^2 + x - 1, 2*y^2 - x - 3, y^2 - 1]]
C =  [1, 1, 1, 1, 1, 1]
La colonne C est déjà présente dans ColonnesDistinctes à l'indice  1
On ajoute dans sol le produit de Kronecker de I =  [1, 1]  avec lb[ 1 ]
sol =  set([(-1, 1), (-1, -1)])


{(-1, -1), (-1, 1)}
```

On obtient pour solution la variété $V = \{(x = -1, y \pm 1)\}$.

B.4.2 Exemple 2 :

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

```
systeme2 =[([(NombreFlouRed(2,1,1,"Quad","Quad"),x*y), (NombreFlouRed(3,1,1,"Quad","Quad"),x**2*y**2), (NombreFlouRed(2,1,1,"Quad","Quad"),x**3*y**3)],\
      [(NombreFlouRed(7,3,3,"Quad","Quad"),1)]), ([(NombreFlouRed(5,1,1,"Quad","Quad"),x*y), (NombreFlouRed(2,3,1,"Quad","Quad"),x**2*y**2)\
      , (NombreFlouRed(2,2,1,"Quad","Quad"),x**3*y**3)], [(NombreFlouRed(9,6,3,"Quad","Quad"),1)])]
```

less than a minute ago
0.018 seconds

```
resolution_reelle_systemes_flous(systeme,2)
```

less than a minute ago
0.348 seconds

```
Pour j =  0  :  I =  [-1, -1] ,  x négatif  et  y négatif
ColonnesDistinctes =  []
ListeSystemes =  []
C =  [1, 1, 1, 1, 1, 1, 1, 1]
La colonne C est une nouvelle colonne, cpt est incrémenté, cpt =  0  et on ajoute C à ColonnesDistinctes à l'indice  0
S(I) = [2*x^3*y^3 + 2*x^2*y^2 + 5*x*y - 9, 2*x^3*y^3 + 3*x^2*y^2 + 2*x*y - 7, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6, x^3*y^3 + x^2*y^2 + x*y - 3]
Le système S(I) est un nouveau système, on l'ajoute à ListeSystemes à l'indice  0
On résout S(I)
Solutions positives de S(I) : set([(1/y, 'R+')])
On insère ces solutions dans lb à  l'incide 0
On ajoute dans sol le produit de Kronecker de I =  [-1, -1]  avec lb[ 0 ]
sol =  set([(1/y, 'R-')])


Pour j =  1  :  I =  [-1, 1] ,  x négatif  et  y positif
ColonnesDistinctes =  [[1, 1, 1, 1, 1, 1, 1, 1]]
ListeSystemes =  [[2*x^3*y^3 + 2*x^2*y^2 + 5*x*y - 9, 2*x^3*y^3 + 3*x^2*y^2 + 2*x*y - 7, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6, x^3*y^3 + x^2*y^2 + x*y - 3]]
C =  [-1, 1, -1, 1, -1, 1, -1, 1]
La colonne C est une nouvelle colonne, cpt est incrémenté, cpt =  1  et on ajoute C à ColonnesDistinctes à l'indice  1
S(I) = [-2*x^3*y^3 + 3*x^2*y^2 - 2*x*y - 7, -2*x^3*y^3 + 3*x^2*y^2 - x*y - 6, -x^3*y^3 + x^2*y^2 - x*y - 3, -2*x^3*y^3 + 2*x^2*y^2 - 5*x*y - 9]
Le système S(I) est un nouveau système, on l'ajoute à ListeSystemes à l'indice  1
On résout S(I)
Solutions positives de S(I) : set([])
On insère ces solutions dans lb à  l'incide 1
On ajoute dans sol le produit de Kronecker de I =  [-1, 1]  avec lb[ 1 ]
sol =  set([(1/y, 'R-')])


Pour j =  2  :  I =  [1, -1] ,  x positif  et  y négatif
ColonnesDistinctes =  [[1, 1, 1, 1, 1, 1, 1, 1], [-1, 1, -1, 1, -1, 1, -1, 1]]
ListeSystemes =  [[2*x^3*y^3 + 2*x^2*y^2 + 5*x*y - 9, 2*x^3*y^3 + 3*x^2*y^2 + 2*x*y - 7, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6, x^3*y^3 + x^2*y^2 + x*y - 3], [2*x^3*y^3
- 3*x^2*y^2 + 2*x*y + 7, 2*x^3*y^3 - 3*x^2*y^2 + x*y + 6, x^3*y^3 - x^2*y^2 + x*y + 3, 2*x^3*y^3 - 2*x^2*y^2 + 5*x*y + 9]]
C =  [-1, 1, -1, 1, -1, 1, -1, 1]
La colonne C est déjà présente dans ColonnesDistinctes à l'indice  1
On ajoute dans sol le produit de Kronecker de I =  [1, -1]  avec lb[ 1 ]
sol =  set([(1/y, 'R-')])


Pour j =  3  :  I =  [1, 1] ,  x positif  et  y positif
ColonnesDistinctes =  [[1, 1, 1, 1, 1, 1, 1, 1], [-1, 1, -1, 1, -1, 1, -1, 1]]
ListeSystemes =  [[2*x^3*y^3 + 2*x^2*y^2 + 5*x*y - 9, 2*x^3*y^3 + 3*x^2*y^2 + 2*x*y - 7, 2*x^3*y^3 + 3*x^2*y^2 + x*y - 6, x^3*y^3 + x^2*y^2 + x*y - 3], [2*x^3*y^3
- 3*x^2*y^2 + 2*x*y + 7, 2*x^3*y^3 - 3*x^2*y^2 + x*y + 6, x^3*y^3 - x^2*y^2 + x*y + 3, 2*x^3*y^3 - 2*x^2*y^2 + 5*x*y + 9]]
C =  [1, 1, 1, 1, 1, 1, 1, 1]
La colonne C est déjà présente dans ColonnesDistinctes à l'indice  0
On ajoute dans sol le produit de Kronecker de I =  [1, 1]  avec lb[ 0 ]
sol =  set([(1/y, 'R+'), (1/y, 'R-')])


{(1/y, 'R+'), (1/y, 'R-')}
```

On obtient pour solution la variété $V = \{(x = \frac{1}{y}, y) \mid y \in \mathbb{R}\backslash\{0\})\}$.

Bibliographie

[AA06] S. Abbasbandy and M. Amirfakhrian. Numerical approximation of fuzzy functions by fuzzy polynomials. *Applied Mathematics and Computation*, 174(2) :1001 – 1006, 2006. 92

[ABPR13] Diego F. Aranha, Paulo S. L. M. Barreto, Geovandro C. C. F. Pereira, and Jefferson E. Ricardini. A note on high-security general-purpose elliptic curves. Cryptology ePrint Archive, Report 2013/647, 2013. `https://eprint.iacr.org/2013/647`. 83, 84

[ABS12] Samuel Antao, Jean Claude Bajard, and Leonel Sousa. RNS-based elliptic curve point multiplication for massive parallel architectures. *The Computer Journal*, 55 :629–647, 05 2012. 70, 81, 82, 83, 86

[AJMAH11] Noor'ani Ahmad, Kavikumar Jacob, Mustafa Mamat, and Nor Shamsidah Amir Hamzah. Solving dual fuzzy polynomial equation by ranking method. *Far East Journal of Mathematical Sciences*, 51 :151 –163, 04 2011. 92

[Ami08] Majid Amirfakhrian. Numerical solution of algebraic fuzzy equations with crisp variable by gauss–newton method. *Applied Mathematical Modelling*, 32(9) :1859 – 1868, 2008. 92

[AMM99] Philippe Aubry and Marc Moreno Maza. Triangular sets for solving polynomial systems : a comparative implementation of four methods. *Journal of Symbolic Computation*, 28 :125–154, 07 1999. 8, 92

[AMV20] Philippe Aubry, Jérémy Marrez, and Annick Valibouze. Computing real solutions of fuzzy polynomial systems. *Fuzzy Sets and Systems*, January 2020. 128

[AO06] Saeid Abbasbandy and Mahmood Otadi. Numerical solution of fuzzy polynomials by fuzzy neural network. *Applied Mathematics and Computation*, 181(2) :1084–1089, 2006. 8, 92, 101

[AOM08] S. Abbasbandy, M. Otadi, and M. Mosleh. Numerical solution of a system of fuzzy polynomials by fuzzy neural network. *Information Sciences*, 178(8) :1948 – 1960, 2008. 8, 92, 121

[AS99] Jerome A. Solinas. Generalized mersenne numbers. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.46.2133, 10 1999. 81

[AT03] T. Akishita and T. Takagi. Zero-value point attacks on elliptic curve cryptosystem. In *Information Security, 6th International Conference, ISC 2003*, pages 218–233, 2003. 56

[ATT94] J. Aluja, A. Tacu, and H. Teodorescu, editors. *Fuzzy Systems in Economy and Engineering*. Publishing House of The Romanian Academy, 1994. 7, 92

[AVL19] R. Abarzúa, C. Valencia, and J. López. Survey for performance and security problems of passive side-channel attacks countermeasures in ECC. Cryptology ePrint Archive, Report 2019/010, 2019. 56

[Bab86] László Babai. On lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1) :1–13, 1986. 21

[Bar87] P.D. Barrett. Implementing the rivest shamir and adleman public key encryption algorithm on a standard digital signal processor. In Springer-Verlag, editor, *Advances in Cryptology, Proc. Crypto'86*, volume 263 of *LNCS*, pages 311–323, 1987. 7, 14

[BBRV16] 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. 8, 10, 92, 101, 109, 111, 112, 123, 125

[BE97] J.J. Buckley and E. Eslami. Neural net solutions to fuzzy problems : The quadratic equation. *Fuzzy Sets and Systems*, 86(3) :289 – 298, 1997. 92

[BEHZ16] Jean-Claude Bajard, Julien Eynard, Anwar Hasan, and Vincent Zucca. A full RNS variant of FV like somewhat homomorphic encryption schemes. Cryptology ePrint Archive, Report 2016/510, 2016. https://eprint.iacr.org/2016/510. 74

[BFH02] J. J. Buckley, T. Feuring, and Y. Hayashi. Solving fuzzy equations using evolutionary algorithms and neural nets. *Soft Computing*, 6(2) :116–123, Apr 2002. 92

[BI04] Jean-Claude Bajard and Laurent Imbert. A full RNS implementation of rsa. *IEEE Trans. Comput.*, 53(6) :769–774, June 2004. 70, 74

[BIP05a] J.-C. Bajard, L. Imbert, and T. Plantard. Arithmetic operations in the polynomial modular number system. In *17th IEEE Symposium on Computer Arithmetic (ARITH'05)*, pages 206–213, June 2005. 9, 15, 16, 50, 54, 57, 75, 81, 82

[BIP05b] J.C. Bajard, L. Imbert, and T. Plantard. Modular number systems : Beyond the mersenne family. In *Selected Areas in Cryptography*, pages 159–169. Springer, 2005. 14, 80, 81

[BJ02] E. Brier and M. Joye. Weierstraß elliptic curves and side-channel attacks. In *Public Key Cryptography*, volume 2274 of *LNCS*, pages 335–345. Springer, 2002. 56

[Bla83] George R Blakely. A computer algorithm for calculating the product ab modulo m. *IEEE Transactions on Computers*, 100(5) :497–500, 1983. 6

[Bon10] Nicolae Ciprian Bonciocat. On an irreducibility criterion of perron for multivariate polynomials. *Bulletin mathématique de la Société des Sciences Mathématiques de Roumanie*, pages 213–217, 2010. 31

[Bon15] N. C. Bonciocat. Schönemann–eisenstein–dumas-type irreducibility conditions that use arbitrarily many prime numbers. *journal Communications in Algebra*, 43(8), 2015. 25, 31

[BQ90] J.J. Buckley and Y. Qu. Solving linear and quadratic fuzzy equations. *Fuzzy Sets and Systems*, 38(1) :43 – 59, 1990. 92, 101

[BSS16] V. Bahadur, D. Selvakumar, and P. M. Sobha. Reconfigurable side channel attack resistant true random number generator. In *International Conference on VLSI Systems, Architectures, Technology and Applications*, pages 1–6, 2016. 56

[BT16] K. Bigou and A. Tisserand. Hybrid position-residues number system. In *IEEE 23nd Symposium on Computer Arithmetic (ARITH)*, pages 126–133, July 2016. 5, 6, 70, 80, 81, 82, 83, 86

[Buh01] Joe Buhler. Resultants, discriminants, bezout, nullstellensatz, etc. `http://people.reed.edu/~jpb/alg/notes/101.pdf`, 2001. Reed College. 77

[BW93] Becker and Weispfenning. *Grobner bases*, volume 141 of *Graduate texts in math.* Springer-Verlag, 1993. 8, 92

[CFG+10] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil. Horizontal correlation analysis on exponentiation. In *ICICS*, volume 6476 of *LNCS*, pages 46–61. Springer, 2010. 56

[CG90] 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. 123

[CH03] Jaewook Chung and Anwar Hasan. More generalized mersenne numbers. In *International Workshop on Selected Areas in Cryptography*, pages 335–347. Springer, 2003. 5

[Che04] B. Chevallier-Mames. Self-randomized exponentiation algorithms. In *CT-RSA*, volume 2964 of *LNCS*, pages 236–249. Springer, 2004. 56

[CJ01] C. Clavier and M. Joye. Universal exponentiation algorithm. In *CHES*, volume 2162 of *LNCS*, pages 300–308. Springer, 2001. 56

[CJ03] M. Ciet and M. Joye. (virtually) Free randomization techniques for elliptic curve cryptography. In *ICICS*, volume 2836 of *LNCS*, pages 348–359. Springer, 2003. 56

[Cor99] J.-S. Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *CHES*, volume 1717 of *LNCS*, pages 292–302, 1999. 56

[CP01] Richard Crandall and Carl Pomerance. *Prime Numbers : a Computational Perspective*. 2001. 83

[Cra92] Richard E Crandall. Method and apparatus for public key exchange in a cryptographic system, October 27 1992. US Patent 5,159,632. 5

[DDEM+19] L.-S. Didier, F.-Y. Dosso, N. El Mrabet, J. Marrez, and P. Véron. Randomization of arithmetic over polynomial modular number system. June 2019. to appear. 67

[DDV18] L.-S. Didier, F.-Y. Dosso, and P. Véron. Efficient and secure modular operations using the adapted modular number system. https ://arxiv.org/abs/1901.11485, 2018. 51, 65

[DH76] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6) :644–654, 1976. 3, 4

[DKMP00] Didier Dubois, Etienne Kerre, Radko Mesiar, and Henri Prade. *Fuzzy Interval Analysis*, pages 483–581. Springer US, Boston, MA, 2000. 93

[DP+] Giambattista Della Porta et al. *De Furtivis Literarum notis, vulgo de Ziferis libri IIII*. Apud Ioa. Mariam Scotum. 2

[DP78] Didier Dubois and Henri Prade. Operations on fuzzy numbers. *International Journal of systems science*, 9(6) :613–626, 1978. 96

[Dum06] G. Dumas. Sur quelques cas d'irreductibilité des polynômes á coefficients rationnels. *journal de Mathématique Pure et Appliquée*, 2, 1906. 25

[EKZ15] 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. 154

[ElG85a] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4) :469–472, 1985. 4

[ElG85b] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18. Springer Berlin Heidelberg, 1985. 85

[Feo17] Luca De Feo. Mathematics of isogeny based cryptography. *CoRR*, abs/1711.04062, 2017. 81

[Fip00] PUB Fips. 186-2. digital signature standard (dss). *National Institute of Standards and Technology (NIST)*, 20 :13, 2000. 5

[FJ06] C. Finch and L. Jones. On the irreducibility of -1,0,1-quadrinomials. *INTEGERS : Electronic journal of Combinatorial number Theory*, 6, 2006. 27

[FMK98] Menahem Friedman, Ma Ming, and Abraham Kandel. Fuzzy linear systems. *Fuzzy Sets and Systems*, 96(2) :201 – 209, 1998. 101

[FPA19] Hamed Farahani, Mahmoud Paripour, and Saeid Abbasbandy. Resolution of single-variable fuzzy polynomial equations and an upper bound on the number of solutions. *Soft Computing*, pages 1–9, 2019. 92, 101

[FRBM15] Hamed Farahani, Sajjad Rahmany, Abdolali Basiri, and Ali Abbasi Molai. Resolution of a system of fuzzy polynomial equations using eigenvalue method. *Soft Comput.*, 19 :283–291, 2015. 92, 101

[FV12] J. Fan and I. Verbauwhede. An updated survey on secure ECC implementations : Attacks, countermeasures and cost. In *Cryptography and Security : From Theory to Applications*, pages 265–282. Springer, 2012. 50, 65, 67

[Gal11] Steven Galbraith. Algorithms for the closest and shortest vector problem. *Mathematics of Public Key Cryptography*, 2011. 53

[Gal12] S.D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012. 14

[GJM+11] R. Goundar, M. Joye, A. Miyaji, M. Rivain, and A. Venelli. Scalar multiplication on weierstraß elliptic curves from co-Z arithmetic. *J. Cryptographic Engineering*, 1(2) :161–176, 2011. 56

[Gou03] L. Goubin. A refined power-analysis attack on elliptic curve cryptosystems. In *Public Key Cryptography*, volume 2567 of *LNCS*, pages 199–210. Springer, 2003. 1, 5, 50, 56, 67

[GV86] R Goetschel and W Voxman. Elementary fuzzy calculus. *Fuzzy Sets Syst.*, 18(1) :31–43, January 1986. 95

[Ham15] Mike Hamburg. Ed448-goldilocks, a new elliptic curve. Cryptology ePrint Archive, Report 2015/625, 2015. https://eprint.iacr.org/2015/625. 83, 84

[Hea00] Daniel R Headrick. *When information came of age : Technologies of knowledge in the age of reason and revolution, 1700-1850.* Oxford University Press, 2000. 2

[HPS11] G. Hanrot, X. Pujol, and D. Stehlé. Algorithms for the shortest and closest lattice vector problems. In *International Conference on Coding and Cryptology*, pages 159–190. Springer, 2011. 14

[Joh99] Anna M. Johnston. A generalized qth root algorithm. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '99, pages 929–930, Philadelphia, PA, USA, 1999. Society for Industrial and Applied Mathematics. 72

[JQ01] Marc Joye and Jean-Jacques Quisquater. Hessian elliptic curves and side-channel attacks. volume 2162, pages 402–410, 01 2001. 86

[JY02] M. Joye and S.-M. Yen. The montgomery powering ladder. In *CHES*, volume 2523 of *LNCS*, pages 291–302. Springer, 2002. 56

[Kah80] David Kahn. *La guerre des codes secrets.* Inter Editions, 1980. 2

[Kah96] David Kahn. *The Codebreakers : The comprehensive history of secret communication from ancient times to the internet.* Simon and Schuster, 1996. 1, 2

[KAV05] M. Tavassoli Kajani, B. Asady, and A. Hadi Vencheh. An iterative method for solving dual fuzzy nonlinear equations. *Applied Mathematics and Computation*, 167(1) :316 – 323, 2005. 92

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg. 1, 86

[KMVOV96] Jonathan Katz, Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography.* CRC press, 1996. 14

[Knu81] Donald E Knuth. The art of computer programming. vol. 2 : Seminumerical algorithms. *Atmospheric Chemistry & Physics*, 1981. 5, 7

[KO62] A Karatsuba and Yu Ofman. Multiplication of multiplace numbers on automata. In *Dokl. Akad. Nauk SSSR*, volume 145, pages 293–294, 1962. 7

[Kob87] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177) :203–209, 1987. 4

[Koc96] P. Kocher. Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In *CRYPTO*, LNCS, pages 104–113. Springer, 1996. 1, 50, 56

[LD99] J. López and R. Dahab. Fast multiplication on elliptic curves over gf(2^{m}) without precomputation. In *CHES*, LNCS, pages 316–327. Springer, 1999. 56

[LD15] Weldon A. Lodwick and Didier Dubois. Interval linear systems as a necessary step in fuzzy linear systems. *Fuzzy Sets and Systems*, 281 :227 – 251, 2015. Special Issue Celebrating the 50th Anniversary of Fuzzy Sets. 101

[Liu02] Puyin Liu. Analysis of approximation of continuous fuzzy functions by multivariate fuzzy polynomials. *Fuzzy Sets and Systems*, 127(3) :299–313, 2002. 8, 92

[Lju60] W. Ljunggren. On the irreducibility of certain trinomials and quadrinomials. *Mathematica Scandinavica*, volume 8(n^o 1) :65–70, 1960. 27, 28, 30

[Lod07] W. Lodwick. *Fuzzy Surfaces in GIS and Geographical Analysis : Theory, Analytical Methods, Algorithms and Applications*. CRC Press, 2007. 7, 92

[LS03] Weldon A. Lodwick and Jorge Santos. Constructing consistent fuzzy surfaces from fuzzy data. *Fuzzy Sets and Systems*, 135(2) :259 – 277, 2003. 8, 92

[LvdPdW12] Thijs Laarhoven, Joop van de Pol, and Benne de Weger. Solving hard lattice problems and the security of lattice-based cryptosystems. *IACR Cryptology EPrint Archive*, 2012 :533, 2012. 21

[Mar17] Jérémy Marrez. Bibliothèque Fuzzy en SageMath, Documentation. working paper or preprint, December 2017. 93, 121

[Mar19] Jérémy Marrez. Fuzzy package SageMath. `https://github.com/JeremyMarrez/Fuzzy`, 2019. 8, 143

[Mat02] Jiří Matoušek. *Lattices and Minkowski's Theorem*, pages 17–28. Springer New York, New York, NY, 2002. 77

[MBR13] 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. 8, 10, 92, 101, 109, 111

[MG02] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems : a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic publishers, 2002. 19, 20

[Mil85] W. H. Mills. The factorization of certain quadrinomials. *Mathematica Scandinavica*, 57, 1985. 27, 28

[MM19] Paulo Martins and Jérémy Marrez. HyPoRes Multiplication C++. `https://github.com/JeremyMarrez/HyPoRes`, 2019. 83

[MMBS19] Paulo Martins, Jérémy Marrez, Jean-Claude Bajard, and Leonel Sousa. Hypores : An hybrid representation system for ecc. June 2019. 89

[Mon85] P.L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44 :519–521, 1985. 7, 14

[Mon87] P. Montgomery. Speeding the Pollard and elliptic curve method of factorization. In *Mathematics of computation*, pages 243–264. Springer, 1987. 50, 56

[MR06] S. Muzzioli and H. Reynaerts. Fuzzy linear systems of the form a1x+b1=a2x+b2. *Fuzzy Sets and Systems*, 157(7) :939 – 951, 2006. 101

[MZMS17] A. S. Molahosseini, A. A. E. Zarandi, P. Martins, and L. Sousa. A multifunctional unit for designing efficient RNS-based datapaths. *IEEE Access*, 5 :25972–25986, 2017. 85

[NP08] C. Nègre and T. Plantard. Efficient modular arithmetic in adapted modular number system using Lagrange representation. In *13th Australasian conference on Information Security and Privacy*, pages 463–477. Springer, 2008. 51

[NQ98] P. Naudin and C. Quitté. Univariate polynomial factorization over finite fields. *Theoretical Computer Science*, 191(1-2) :1–36, January 1998. 38

[Odl84] Andrew M Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 224–314. Springer, 1984. 4

[OS12] D. O'Shea and R. Seroul. *Programming for Mathematicians*. Universitext. Springer Berlin Heidelberg, 2012. 71

[Pla05] T. Plantard. *Arithmétique modulaire pour la cryptographie*. Theses, Université Montpellier II - Sciences et Techniques du Languedoc, 2005. 7, 14, 19, 50

[Pol78] John M Pollard. Monte carlo methods for index computation (mod p). *Mathematics of computation*, 32(143) :918–924, 1978. 4

[PSZ15] T. Plantard, W. Susilo, and Z. Zhang. Lll for ideal lattices : re-evaluation of the security of gentry–halevi's fhe scheme. *Designs, Codes and Cryptography*, volume 76(n^o 2) :325–344, 2015. 14, 52, 53, 61

[Rou99] Fabrice Rouillier. Solving Zero-Dimensional Systems through the Rational Univariate Representation. *Applicable Algebra in Engineering, Communication and Computing*, 9(5) :433–461, May 1999. Article dans revue scientifique avec comité de lecture. 8, 92

[Rou07] H. Rouhparvar. Solving fuzzy polynomial equation by ranking method. First Joint Congress on Fuzzy and Intelligent Systems, Ferdowsi University of Mashhad, Iran, 2007. 8, 92

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, February 1978. 3, 85

[S+99] Jerome A Solinas et al. *Generalized Mersenne numbers*. Citeseer, 1999. 5, 16

[SAM16] L. Sousa, S. Antao, and P. Martins. Combining residue arithmetic to design efficient cryptographic circuits and systems. *IEEE Circuits and Systems Magazine*, 16(4) :6–32, Fourthquarter 2016. 6, 70

[SEC00] SECG SEC. 2 : Recommended elliptic curve domain parameters. *Standards for Efficient Cryptography Group, Certicom Corp*, 2000. 5

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3) :379–423, 1948. 2

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 256–266. Springer, 1997. 4

[Sin03] Simon Singh. The code book. the science of secrecy from ancient egypt to quantum cryptography. *Swiat Ksiazki*, pages 19–21, 2003. 2

[SS71] Arnold Schönhage and Volker Strassen. Schnelle multiplikation grosser zahlen. *Computing*, 7(3-4) :281–292, 1971. 7

[SS09] Luciano Stefanini and Laerte Sorini. Fuzzy arithmetic with parametric lr fuzzy numbers. In *IFSA/EUSFLAT Conf.*, pages 600–605, 2009. 96

[Tak92] Naofumi Takagi. A radix-4 modular multiplication hardware algorithm for modular exponentiation. *IEEE Transactions on Computers*, 41(8) :949–956, 1992. 6

[Tay81] F Taylor. Large moduli multipliers for signal processing. *IEEE Transactions on circuits and systems*, 28(7) :731–736, 1981. 6

[TB02] E. Trichina and A. Bellezza. Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks. In *CHES*, LNCS, pages 98–113. Springer, 2002. 56

[TR11] A Taleshian and S Rezvani. Multiplication operation on trapezoidal fuzzy numbers. 2011. 145

[Wal01] C. D. Walter. Sliding windows succumbs to big mac attack. In *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 286–299. Springer Berlin Heidelberg, 2001. 86, 87

[Wu87] Wen-tsun Wu. A zero structure theorem for polynomial-equations-solving and its applications. In *EUROCAL*, page 44, 1987. 8, 116, 123

[Zad65] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3) :338–353, 1965. 94