

Université Paris-Sud
Département de mathématiques

Mémoire de magistère

Édouard ROUSSEAU



Comprendre le monde,
construire l'avenir®

Remerciements

Avant d'inonder le lecteur avec des symboles plus ou moins obscurs, j'aimerais remercier quelques personnes. Tout d'abord, Nathalie Carrierre, qui a toujours été à la fois disponible pour papoter et pour me sortir d'affaire lorsque j'appelais à l'aide. Le magistère et la licence 3 ne seraient pas les mêmes sans elle! Merci aussi pour avoir été si si bon public lors de nos spectacles à *Maths en folie* et à la TIPS.

Bien sûr, je voudrais aussi remercier tous les professeurs qui m'ont encadré à l'université. J'ai eu la chance de travailler avec des encadrants bienveillants et passionnants, qui m'ont donné envie de continuer à faire des mathématiques. Merci à Stéphane Fischler et Pierre-Guy Plamondon pour les encadrements sur des sujets "de recherche". Merci également à Nicolas Thiéry pour m'avoir encadré en stage et pour m'avoir fait découvrir et sentir l'intérêt de l'algèbre effective. C'est aussi via Nicolas que j'ai rencontré Luca De Feo, qui m'a par la suite encadré lors d'un stage puis en thèse, et que je remercie d'ailleurs pour tout ça. Merci à Michaël Quisquater pour m'avoir fait découvrir le joli sujet des bases normales en projet de programmation. Pour finir, merci à Éric Schost de m'avoir accueilli à l'université de Waterloo et de m'avoir fait travailler dans la bonne humeur.

Finalement, merci à tous mes camarades de promo, pour les repas, les discussions, les rires, avec une mention spéciale pour les Versaillais Pascal et Simon. Les maths aussi sont des sciences humaines. En particulier merci à mes colocs Benoît, Emile, Erwan, Jérémy, Maxime et Thibault pour le bonheur quotidien! Merci aussi à mes camarades de la TIPS (puis des Gibbons) pour nos soirées ou spectacles improvisés.

Et finalement merci à ma famille et à Amandine pour avoir toujours été là.

Préambule

Ce document contient 8 parties, dont la plupart sont indépendantes, en anglais ou en français, en fonction de la situation dans lesquelles elles ont été écrites. Je retrace d'abord mon parcours au sein du magistère de mathématiques. Puis je présente le domaine de recherche dans lequel j'ai évolué en stage, et qui constitue toujours un sujet connexe à mon sujet de thèse. Enfin, je joins tous les documents que j'ai pu écrire lors de mon cursus de magistère, chronologiquement. Plus de détails seront donnés dans un résumé en début de partie.

Table des matières

1	Cursus en magistère	6
1.1	Licence 3 - Première année	6
1.1.1	Premier semestre	6
1.1.2	Second semestre	7
1.2	Master 1 - Deuxième année	7
1.2.1	Premier semestre	7
1.2.2	Second semestre	7
1.3	Préparation à l'agrégation	8
1.4	Master 2 "Algèbre Appliquée" - Troisième année	8
1.5	Conclusion	9
2	Présentation d'un domaine de recherche	10
2.1	Cryptologie	10
2.2	Le problème du logarithme discret	11
2.3	Quelques avancées	11
2.4	Les mathématiques du logarithme discret (dans les corps finis)	12
2.5	La petite caractéristique	14
3	Projet de L3	18
3.1	Résultats préliminaires	18
3.2	Le cas irrationnel	22
3.3	Le théorème	26
3.4	Irrationnalité de e	31
4	Stage de magistère	32
4.1	Préambule	32
4.2	Some theory	32
4.2.1	Coxeter groups	32
4.2.2	Markov chains	33
4.3	Commutation classes of the reduced words	33
4.4	The bijection between tilings and commutation classes	34
4.5	Exchange walk	37
4.6	Directed graph and transition matrix	39

4.7	Appendix : the code of the programs	41
4.7.1	Ismaxi	42
4.7.2	Sorting networks	42
4.7.3	Maximise	42
4.7.4	Tiling	42
4.7.5	Walk	43
4.7.6	Transition matrix	43
4.7.7	Walk graph	44
5	Travail Encadré de Recherche de M1	45
5.1	Préambule	45
5.2	Notions préliminaires	46
5.2.1	Carquois	46
5.2.2	Représentations de carquois	46
5.2.3	Théorie des catégories	51
5.2.4	Diagrammes de Dynkin	51
5.3	Foncteurs de réflexion et foncteurs de Coxeter	53
5.3.1	Foncteurs de réflexion	53
5.3.2	Foncteurs de Coxeter	67
5.4	Graphes, groupe de Weyl et transformations de Coxeter	72
5.4.1	Graphes et groupe de Weyl	72
5.4.2	Racines	78
5.5	Le théorème de Gabriel	80
6	Stage hors parcours	84
6.1	Préambule	84
6.2	Julia	84
6.2.1	Overview of Julia’s characteristics	84
6.2.2	Nemo	85
6.3	A bit of theory	85
6.3.1	General background	85
6.3.2	Trace and duality	86
6.3.3	Transposition principle	87
6.3.4	The algorithms	88
6.4	Julia/Nemo in practice	93
6.4.1	Benchmarks	93
6.5	Conclusion	95
7	Projet de M2	96
7.1	Préambule	96
7.2	Introduction	96
7.2.1	Normal bases	96
7.2.2	Recalls and notations	97
7.2.3	Flint	97

7.3	Basics on normal bases	98
7.4	Computation of normal bases	101
7.4.1	Randomized algorithm	102
7.4.2	Deterministic algorithms	103
7.5	Experimental results	107
7.5.1	Random algorithms	108
7.5.2	Lüneburg’s and Lenstra’s algorithms	108
8	Stage de M2	110
8.1	Préambule	111
8.2	Introduction	111
8.3	The index calculus method	113
8.4	Quasi-polynomial algorithms	114
8.4.1	Setup	114
8.4.2	Background ideas	115
8.5	The BGJT algorithm	116
8.5.1	Setup of the BGJT algorithm	116
8.5.2	Main results and complexity	117
8.5.3	Proof of Proposition 8.5.1 and Proposition 8.5.2	118
8.5.4	Some concluding remarks	122
8.6	The powers-of-2 algorithm	123
8.6.1	Setup of the algorithm	123
8.6.2	On-the-fly degree 2 elimination	124
8.6.3	The descent	127
8.6.4	Some concluding remarks	129
8.7	Experimental results	130
8.7.1	Julia and Nemo	130
8.7.2	The BGJT algorithm	130
8.7.3	The powers-of-2 algorithm	132

Chapitre 1

Cursus en magistère

Avant de m'attarder sur le contenu de mon parcours au sein du magistère, il est possible qu'il soit intéressant de mentionner les raisons qui m'ont motivées à y entrer.

Je souhaite faire des mathématiques depuis un moment déjà, et c'est pourquoi j'ai fait deux années de classe prépa après le lycée. Je ne connaissais pas l'existence des formations "magistère" mais la publicité réalisée dans la classe m'a permis de découvrir ce parcours. J'ai donc été tout naturellement attiré par la possibilité de faire un petit peu plus que la formation universitaire classique et me suis inscrit en magistère. J'ai par la suite eu un parcours tout à fait standard : L3 Mathématiques Fondamentales et Appliquées (MFA), M1 MFA, une pause d'un an pour passer le concours de l'agrégation, puis un M2 Recherche.

1.1 Licence 3 - Première année

La première année de magistère, une licence 3 de Mathématiques Fondamentales et Appliquées (MFA), comportait des cours fondamentaux et généralistes, couvrant des domaines essentiels des mathématiques, et offrant un bon bagage de départ.

1.1.1 Premier semestre

Au premier semestre, j'ai suivi les cours obligatoires

- algèbre ;
- théorie de l'intégration de Lebesgue ;
- topologie et calcul différentiel ;

ainsi que les cours optionnels algorithmique et théorie des graphes. Je le savais déjà avant d'arriver en licence, mais il m'a été confirmé que j'appréciais beaucoup plus les cours à tendances algébriques que les autres. J'ai également découvert l'algorithmique, que j'ai énormément apprécié.

1.1.2 Second semestre

Au second semestre, il y a eu un peu plus de cours

- théorie des équations différentielles ordinaires ;
- analyse de Fourier ;
- probabilités ;
- fonctions holomorphes ;
- algèbre effective ;

ainsi qu'un "Projet" qui consistait en un petit travail de recherche. C'est lors de ce projet que j'ai étudié, avec Stéphane Fishler, la transcendance de e et π . C'est également pendant ce semestre que j'ai utilisé pour la première fois le logiciel Sagemath, un logiciel libre de calcul formel que j'ai beaucoup utilisé par la suite. Enfin, j'ai également pu découvrir la topologie générale, dans le cadre du cours supplémentaire du Magistère, que j'ai énormément apprécié. Enfin, le stage *hors-murs* imposé par le magistère m'a permis de découvrir le Laboratoire de Recherche en Informatique (LRI) d'Orsay, où j'ai pu de nouveau utiliser Sage. C'est réellement pendant ce stage que j'ai pu découvrir les problématiques liées au calcul formel, et que j'ai été charmé par ce domaine.

1.2 Master 1 - Deuxième année

Le M1 Mathématiques Fondamentales et Appliquées est lui aussi généraliste, mais on commence tout de même à se spécialiser un peu. J'ai décidé de suivre les cours généraux qui préparaient à l'agrégation, ainsi que les cours de probabilité et statistiques.

1.2.1 Premier semestre

Ainsi, au premier semestre, j'ai suivi les cours

- probabilités ;
- mathématiques Générales et Analyse I ;
- introduction à la théorie spectrale ;

le cours de théorie spectrale étant le cours supplémentaire du magistère, et qui a l'intérêt de faire aimer la dimension infinie (en tous cas quand ça se passe bien et qu'on est dans un espace de Hilbert).

1.2.2 Second semestre

Lors de la seconde partie de l'année, j'ai étudié

- statistiques ;
- mathématiques Générales et Analyse II ;
- introduction aux systèmes dynamiques ;

le cours de système dynamique étant une fois encore lié au magistère et très intéressant, bien qu'un peu loin de mes amours algébriques. Pendant cette période, je me suis également intéressé au théorème de Gabriel, en compagnie de Maxime Buron et sous la direction de Pierre-Guy Plamondon. Ce sujet un peu abstrait, et faisant intervenir de très beaux objets (que je ne connaissais pas) m'a énormément plu.

1.3 Préparation à l'agrégation

Comme beaucoup d'autres avant moi, j'ai fait une pause d'un an afin de passer le concours de l'agrégation. Cela a permis de solidifier mes bases dans tous les domaines des mathématiques et j'ai pu à nouveau faire de l'algèbre effective, au sein de l'option C de l'agrégation : "Calcul formel". C'est en me trouvant de nouveau confronté au calcul formel que j'ai décidé de faire un stage supplémentaire entre mon M2 agrégation et mon M2 de recherche, afin de travailler dans le domaine. C'est aussi cette option qui m'a poussé à choisir le master "Algèbre Appliquée" de l'université de Versailles.

1.4 Master 2 "Algèbre Appliquée" - Troisième année

J'ai choisi de suivre ce master car il réunit des cours de mathématiques fondamentales (courbes algébriques, courbes elliptiques) et des cours d'informatique pratique (Algorithmique et langage C) en passant par des cours intermédiaires (algèbre effective). Étant donné mon souhait d'étudier l'algèbre de manière pratique, ce master me convenait à la perfection. Plus précisément, j'ai suivi les cours

- courbes algébriques ;
- courbes elliptiques ;
- algorithmique et langage C ;
- algèbre effective ;
- complexité algébrique et cryptographie ;
- algorithmes avancés de la cryptographie et de la cryptanalyse ;

pendant la première période. J'ai également eu l'occasion de travailler sur un projet d'informatique via le cours d'algorithmique et langage C : j'ai étudié la génération des bases normales avec l'aide de Michaël Quisquater. Je suis ensuite parti en stage à l'université de Waterloo, au Canada, sous la direction d'Éric Schost, pour travailler sur le problème du logarithme discret dans les corps finis de petite caractéristique. Cette année a été pour moi l'occasion de concrétiser mon intérêt pour le calcul formel, et de m'initier à la recherche dans ce domaine. De plus, j'ai acquis des compétences à la fois en mathématiques (mon domaine de compétence initial) et en informatique, me plaçant ainsi à la frontière entre ces deux compétences.

1.5 Conclusion

Avec un peu de recul, on voit clairement que c'est le magistère qui m'a permis de découvrir de nouveaux domaines des mathématiques, notamment via le stage *hors-murs* de l'année de licence. C'est notamment pendant ce stage que j'ai fait des rencontres (humaines et scientifiques) importantes pour les années qui ont suivies, et qui m'ont menées aujourd'hui à faire une thèse dans le domaine du calcul formel. In fine, la formation m'a également apporté un spectre de connaissance un peu plus large (théorie spectrale, systèmes dynamiques, topologie générale) et m'a également fait travailler des exercices de styles récurrents (présentation orale, rapports écrits...). Ce qui me laisse une bonne impression du parcours Magistère et un slogan. Vous aussi, avec le magistère, devenez un-e super étudiant-e !

Chapitre 2

Présentation d'un domaine de recherche

2.1 Cryptologie

Vocabulaire et étymologie Le problème du logarithme discret, qu'on précisera tout à l'heure, est aujourd'hui partie intégrante de la *cryptologie*. La cryptologie, du grec ancien *kruptós* ("couvert, caché"), c'est la science qui étudie - entre autre - les communications sécurisées. Au sein de la cryptologie, on distingue la cryptographie ("la défense"), qui met au point des protocoles sécurisés, de la cryptanalyse ("l'attaque") qui étudie les faiblesses éventuelles issues de la cryptographies. Bien qu'il y ait deux concepts différents, il est assez rare de tomber sur un cryptographe qui ne soit pas cryptanalyste, et inversement...

Histoire La cryptologie existe depuis longtemps, car il n'est pas nouveau que les individus aient des secrets à protéger. On comprend facilement que les militaires, par exemple, aient eu besoin de communiquer sans que l'ennemi puisse comprendre le message dans le cas où il passerait entre ses mains. Un exemple célèbre est le chiffre de César, qui consiste à décaler toutes les lettres de quatre unités. Du côté civil, la reine Marie-Antoinette a chiffré certaines de ses lettres, qui n'ont été décryptées que récemment. Ces deux protocoles sont des exemples de *cryptographie symétrique*. C'est-à-dire qu'on suppose que les individus qui veulent communiquer possèdent tous les deux une même clé, un même secret, qui permet à la fois de chiffrer et déchiffrer le message.

Cryptographie asymétrique Mais il existe un autre type de cryptographie, dite *asymétrique*, qui a été inventée plus récemment, en 1976, dans le révolutionnaire article de Diffie et Hellman [14]. Supposons qu'on ait deux individus, Alice et Bob, qui veulent communiquer. Contrairement au cas

symétrique, où Alice et Bob partagent une même clé (par exemple de combien d'unités on décale l'alphabet, dans le cas d'un chiffrement par décalage comme le chiffre de César), ici Alice et Bob ne connaissent pas les mêmes choses, d'où la dénomination "asymétrique". L'inconvénient de la cryptographie symétrique, c'est qu'on suppose qu'Alice et Bob se ont déjà rencontrés au moins une fois pour s'échanger la clé, or ceci n'est pas toujours possible. Un objectif de la cryptographie asymétrique est donc de proposer des protocoles permettant d'établir des communications sécurisées entre individus qui n'ont pas pu communiquer auparavant. Parmi les exemples célèbres, on trouve le protocole RSA (de Rivest, Shamir et Adleman), et le protocole de Diffie-Hellman (celui exposé dans [14]). Ils sont respectivement basés sur le problème de la factorisation des entiers et sur le problème du logarithme discret.

2.2 Le problème du logarithme discret

Le problème est relativement simple à expliquer : soit $G = \langle g \rangle$ un groupe cyclique engendré par un élément g , de cardinal N . On a alors un isomorphisme :

$$\begin{aligned} \exp_g : (\mathbb{Z}/N\mathbb{Z}, +) &\rightarrow (\mathcal{G}, \times) \\ n &\mapsto g^n, \end{aligned}$$

dont on note $\exp_g^{-1} = \log_g$ (ou plus simplement \log) l'isomorphisme inverse. L'essence du problème vient du fait que, en pratique, on dispose d'algorithmes efficaces pour calculer $g^n = \exp_g(n)$ (les algorithmes de types *square and multiply*), mais que cela n'est plus vrai pour les calculs de l'isomorphisme inverse. En effet, étant donné un élément $y = g^n$, on ne dispose pas d'algorithme aussi efficace pour calculer $n = \log y$. Historiquement, Gauss s'intéressait déjà à des calculs de logarithmes discrets, qu'il appelait "indices", mais le problème est véritablement devenu célèbre en 1976 suite à l'invention du protocole de Diffie et Hellman. La sécurité de ce protocole s'appuie sur la difficulté supposée de calculer des logarithmes discrets dans le groupe $(\mathbb{Z}/p\mathbb{Z})^\times$, où p est un nombre premier. D'autres groupes ont un intérêt cryptographique, par exemple le groupe des points d'une courbe elliptique, ou bien le groupe des éléments inversibles d'un corps fini. C'est d'ailleurs ce dernier cas auquel nous nous intéressons. Plus précisément, nous étudions les corps finis de *petite caractéristique*, c'est-à-dire les corps \mathbb{F}_{q^n} où q est petit devant la taille du corps ($q \ll q^n$).

2.3 Quelques avancées

Complexité Le problème du logarithme discret est un problème algorithmique. Le but n'est donc pas de trouver une solution (il suffit de tester toutes

les possibilités), mais de trouver une solution efficacement. Pour évaluer l'efficacité d'un algorithme, on parle de complexité, c'est-à-dire grossièrement le nombre d'opérations élémentaires que doit faire l'algorithme. On introduit la notation

$$L_N(\alpha, c) = \exp((c + o(1))(\log N)^\alpha (\log \log N)^{1-\alpha})$$

pour évaluer cette complexité, et comparer les complexités entre elles. On note plus simplement $L_N(\alpha)$ lorsqu'on ne désire pas spécifier la constante, voire $L(\alpha)$ lorsqu'il n'y a pas d'ambiguïté concernant le paramètre N . Cette notation peut être vue comme une interpolation entre la complexité polynomiale en $\log N$, à savoir $L_N(0) = (\log N)^{c+o(1)}$ et la complexité exponentielle en $\log N$, à savoir $L_N(1) = N^{c+o(1)}$. Les algorithmes ayant une complexité de type $L_N(\alpha)$ pour $0 < \alpha < 1$ sont dits *sous-exponentiels*.

Quelques résultats En 1976, seuls des algorithmes à complexité exponentielles étaient connus pour attaquer le problème du logarithme discret dans les corps finis. Il faut attendre 1979 pour qu'Adleman [3] propose et analyse le premier algorithme sous-exponentiel, avec une complexité $L(1/2)$, cet algorithme est utilisable dans les corps finis premiers, *i.e.* de type $\mathbb{Z}/p\mathbb{Z}$ où p est un nombre premier. En 1984, Coppersmith propose le premier algorithme ayant une complexité de $L(1/3)$ [12], valable dans les corps finis de caractéristique 2. Puis, entre 1984 et 2006, tous les types de corps finis ont peu à peu été dotés d'un algorithme pour résoudre le problème du logarithme discret avec une complexité $L(1/3)$.

2.4 Les mathématiques du logarithme discret (dans les corps finis)

Le calcul d'indice Au cœur des algorithmes de logarithme discret, on retrouve la technique du *calcul d'indice*, dont les étapes sont toujours les mêmes. Supposons qu'on veuille trouver le logarithme discret d'un élément h dans le corps fini \mathbb{F}_{q^n} . On va alors :

1. choisir $\mathcal{F} \subset (\mathbb{F}_{q^n})^\times$ tel que \mathcal{F} soit assez grand pour que h puisse être exprimé en fonction des éléments de \mathcal{F} (*i.e.* $h \in \langle \mathcal{F} \rangle$);
2. trouver des relations multiplicatives entre les éléments de \mathcal{F} , du type $f_1^2 f_2 = f_3^5$, car une relation de ce genre se traduit en l'équation linéaire $2 \log f_1 + \log f_2 - 5 \log f_3 = 0$;
3. une fois que le nombre de relations multiplicatives (*i.e.* d'équations linéaires) est assez important, résoudre le système linéaire associé en les inconnues $\log f_i$ où les f_i sont les éléments de \mathcal{F} ;
4. exprimer h en fonction des éléments de \mathcal{F} .

Derrière l'étape 3, il y a les algorithmes classiques de résolution de systèmes linéaires (éventuellement creux si les relations sont creuses). Les étapes 2 et 4 requièrent d'engendrer des relations entre les éléments de notre corps finis \mathbb{F}_{q^n} . Les techniques pour y arriver sont inspirées du crible de corps de nombre, ou crible algébrique, qui est notamment utilisé pour factoriser les entiers avec une complexité $L(1/3)$. On y croise les objets classiques de la théorie (algorithmique) des nombres : corps de nombres, corps finis, corps de fonction, ordres, idéaux, anneaux de polynômes, théorie de Galois...

Un exemple Regardons l'exemple de l'algorithme d'Adleman. Soit $\mathcal{G} = \mathbb{F}_p^\times$ le groupe multiplicatif du corps premier $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, et on pose

$$\mathcal{F} = \{f \mid f \leq B, f \text{ prime}\}$$

où $B \in \mathbb{N}$ est un certain entier. On utilise un abus de notation consistant à écrire f aussi bien pour l'élément $f \in \mathbb{N}$ et sa classe $\bar{f} \in \mathbb{F}_p$. Et on suppose de plus que $g \in \mathcal{F}$, quitte à ajouter g à \mathcal{F} ($\mathcal{G} = \langle g \rangle$). Afin de générer des relations multiplicatives, on choisit au hasard $e \in \mathbb{Z}/(p-1)\mathbb{Z}$ et on calcule g^e . On dit que e génère une relation si l'élément $g^e \in \mathbb{N}$ est B -friable, *i.e.* si g^e ne possède que des diviseurs premiers $\leq B$. Dans ce cas on a la relation

$$g^e \equiv \prod_{f \in \mathcal{F}} f^{e_f} \pmod{p}$$

dans les entiers, qui devient

$$g^e = \prod_{f \in \mathcal{F}} f^{e_f}$$

dans \mathbb{F}_p (avec le même abus de notation), et où les exposants sont des entiers $e_f \in \mathbb{N}$. Au niveau des logarithmes, on a ainsi la relation

$$e = \sum_{f \in \mathcal{F}} e_f \log f.$$

Lorsque l'on a suffisamment de relations, c'est-à-dire plus que $|\mathcal{F}|$, on est capable de retrouver toutes les inconnues $\log f$ en utilisant de l'algèbre linéaire classique.

Si on cible un certain élément $h \in \mathcal{G}$, on procède de manière analogue pour retrouver $\log h$. On choisit au hasard $e \in \mathbb{Z}/(p-1)\mathbb{Z}$ et on calcule hg^e . Ici encore, si l'élément hg^e (vu comme entier) est B -friable, on a une équation du type

$$\log h + e = \sum_{f \in \mathcal{F}} h_f \log f$$

où les $h_f \in \mathbb{N}$ sont des entiers et tous les $\log f$ sont connus. On a ainsi calculé le logarithme de h . Plus B est grand et plus il est facile de trouver des relations, mais en même temps il faudra plus de relations pour résoudre le système linéaire. Adleman [3] a montré qu'on pouvait choisir B de telle sorte que cet algorithme soit de complexité $L(1/2)$.

2.5 La petite caractéristique

Théorie Dans le cas des corps finis de petite caractéristique, on a eu plus récemment de grosses avancées. Intuitivement, un corps fini \mathbb{F}_{q^n} de petite caractéristique est plus simple à gérer car les opérations dans \mathbb{F}_q ne coûtent pas cher (comme q est petit) et les éléments de \mathbb{F}_{q^n} sont représentés par des polynômes à coefficients dans \mathbb{F}_q , on peut donc utiliser les algorithmes efficaces sur les polynômes dans $\mathbb{F}_q[X]$. Ces mêmes algorithmes (dont la complexité dépend de q) sont trop coûteux lorsque q n'est pas petit. Ainsi, en 2013, Atoine Joux a combiné de nouvelles idées à d'autres, issues de l'algorithme de Coppersmith et du crible de corps de fonction, pour créer un algorithme de complexité $L(1/4)$ [23]. Dans la foulée, ces idées novatrices ont permis de découvrir deux algorithmes [6, 18] de complexité *quasi-polynomiale*, quasiment au même moment et indépendamment. Si on note ℓ la taille en bit d'un corps fini \mathbb{F}_{q^n} , c'est-à-dire $\ell = \log(q^n)$, on parle de complexité quasi-polynomiale pour une complexité de type $\ell^{O(\log \ell)}$ (alors que polynomiale serait $\ell^{O(1)}$). Cela a été une énorme avancée car une complexité quasi-polynomiale est plus petite que n'importe quelle complexité $L(\varepsilon)$ où $\varepsilon > 0$, c'est pourquoi on la note aussi parfois $L(o(1))$.

Essayons de présenter (succinctement) les grandes idées derrière l'algorithme de Joux et les algorithmes quasi-polynomiaux. Pour fixer le cadre, comparons le à l'algorithme d'Adleman que nous avons déjà présenté. On travaillera ici avec un corps fini \mathbb{F}_{q^n} , contre \mathbb{F}_p auparavant. En conséquences, les éléments du corps ne seront plus représentés par des entiers mais par des polynômes, car on a

$$\mathbb{F}_{q^n} \cong \mathbb{F}_q[X]/(P),$$

où P est un polynôme irréductible de degré n . On utilisera en outre le même genre d'abus de notation et on aura des égalités multiplicatives basées non plus sur la décomposition en facteur premier des entiers, mais sur la décomposition en facteurs irréductibles des polynômes. Enfin, l'ensemble \mathcal{F} ne sera plus composé d'entiers premiers inférieurs à un entier B mais de polynômes irréductibles de degré inférieur à B . Dans ce contexte, on cherchera à créer des relations impliquant des polynômes ayant des facteurs irréductibles de petit degré.

Il existe trois grandes idées pour arriver à cet objectif.

1. Supposons qu'on ait $Q(X)$ un polynôme qui se factorise bien, ou plus formellement un polynôme B -friable (*i.e.* dont les facteurs irréductibles sont de degré inférieur à B). On remarque facilement qu'après un changement de variable linéaire, le polynôme $Q(aX + b)$ est toujours B -friable. Mais on peut en réalité appliquer cette idée avec une famille de changement de variable encore plus grosse, les homographies $h(X) = \frac{aX+b}{cX+d}$. Bien entendu l'élément $Q(\frac{aX+b}{cX+d})$ n'est pas un

polynôme, mais en notant

$$h \cdot Q = (cX + d)^{\deg Q} Q\left(\frac{aX + b}{cX + d}\right),$$

on s'aperçoit que le polynôme $h \cdot Q$ est bien B -friable. Ces homographies permettent d'engendrer beaucoup de polynômes intéressants à partir d'un seul polynôme initial.

2. La deuxième idée consiste à utiliser un polynôme qui se factorise agréablement par nature. On va ainsi appliquer la première idée avec le polynôme $X^q + X$ qui se factorise en polynômes linéaires sur $\mathbb{F}_q[X]$.
3. Enfin, cette stratégie va disséminer des termes en X^q un peu partout dans nos équations, on va donc simplifier ce terme. Pour cela, on va utiliser une représentation particulière de notre corps $\mathbb{F}_{q^n} \cong \mathbb{F}_q[X]/(P)$. On va choisir un polynôme P qui soit un diviseur irréductible de degré n de $h_1(X)X^q - h_0(X)$, où h_0 et h_1 sont des polynômes de petit degré. En pratique, on arrive systématiquement à trouver des polynômes h_i convenables de degré 2. Cependant, il n'existe pas à ce jour de preuve permettant d'affirmer que tout corps fini peut-être représenté de cette façon. Avec ce choix, on obtient que

$$X^q \equiv \frac{h_0(X)}{h_1(X)} \pmod{P}.$$

Ainsi, si on note x la classe de X dans \mathbb{F}_{q^n} , on a $x^q = \frac{h_0(x)}{h_1(x)}$, ce qui permet de simplifier les équations en abaissant les degrés intervenants. On appelle cette représentation *représentation de Frobenius* car on demande à l'automorphisme de Frobenius d'avoir une expression très simple.

En utilisant ces idées, on est capable de trouver des relations ne faisant intervenir que des éléments de l'ensemble \mathcal{F} , composé, selon les cas, des (éléments représentés par des) polynômes irréductibles de degré inférieur à 1, 2 ou encore 4. On peut en fait même faire cette étape en temps polynômial. La nature quasi-polynomiale ne vient donc pas de l'étape 1 du calcul d'indice. L'étape 2 est elle aussi polynômiale, il ne s'agit que de la résolution d'un système linéaire dont la taille est modérée. Par élimination, l'étape difficile est donc l'étape 3.

En effet, la situation n'est pas la même que dans le cas de l'algorithme d'Adleman, où l'on pouvait exprimer le logarithme de notre cible h en fonction des logarithmes des éléments de \mathcal{F} en une seule étape. Ici, si l'on cible un élément $Q \in \mathbb{F}_{q^m}$, on va l'exprimer en fonctions d'éléments "plus petits" (*i.e.* représentés par des polynômes de degrés plus petits), mais pas nécessairement dans \mathcal{F} . Formellement, on a la proposition

Proposition 2.5.1

Soit \mathbb{F}_{q^n} un corps fini admettant une représentation de Frobenius. Il existe un algorithme polynômial (mais basé sur des heuristiques) tel que, ayant pour entrée un élément de \mathbb{F}_{q^n} représenté par un polynôme Q , avec $2 \leq \deg Q \leq n - 1$, l'algorithme retourne une expression de $\log Q$ comme combinaison linéaire de \mathcal{N} logarithmes $\log Q_j$ où les polynômes Q_j vérifient l'inégalité $\deg Q_j \leq \lceil \frac{1}{2} \deg Q \rceil$.

On n'a pas précisé le paramètre \mathcal{N} , car il dépend de l'algorithme utilisé. Cependant, ce qu'il est important de noter est que ce paramètre est lui aussi polynômial en $\ell = \log(q^n)$, la taille en bit du corps fini en entrée. Ainsi, en appliquant l'algorithme de la proposition par récurrence, on obtient ce qu'on appelle une descente, ou un arbre de descente, où chaque nœud correspond à l'application de l'algorithme. En analysant le nombre de nœud dans l'arbre, on montre que la complexité totale du processus est bien quasi-polynomiale.

En fonction de si l'on utilise l'algorithme de Barbulescu, Gaudry, Joux et Thomé [6] ou celui de Granger, Kleinjung et Zumbrägel [18], les mécanismes cachés derrière la Proposition 2.5.1 sont différents. Dans le premier cas, il s'agit de générer beaucoup de relations multiplicatives et de les combiner intelligemment, de manière analogue au calcul des logarithmes des éléments de l'ensemble \mathcal{F} . Dans le second cas, on va plonger Q dans une (peut-être grosse) extension de \mathbb{F}_q pour avoir une factorisation intéressante de Q . Puis on va exprimer les facteurs de Q en fonction de polynômes de plus petit degré (toujours dans notre extension). Enfin, ces calculs sont faits de telle manière que l'on puisse exprimer le résultat dans $\mathbb{F}_q[X]$.

En pratique, on doit faire des calculs dans des tours d'extension de \mathbb{F}_q , et un sujet intéressant lié à ce problème est la réalisation de réseaux compatibles d'extensions de corps finis, c'est-à-dire un réseau d'extension comportant des plongements, isomorphismes, ou projections entre ces corps, de telle sorte que toutes les flèches du diagramme ainsi formé commutent. Il existe une implémentation de ce type de réseau dans le logiciel de calcul formel MAGMA, et un projet de début de thèse sera pour nous de regarder si l'on peut améliorer la façon de traiter le problème.

Heuristiques Néanmoins, les algorithmes quasi-polynomiaux découverts sont *heuristiques*. En d'autres termes, lors de certaines étapes de ces algorithmes, on s'appuie sur un résultat conjectural. Le caractère heuristique de ces algorithmes n'est en fait pas nouveau, les algorithmes ayant une complexité plus grande sont eux aussi heuristiques. Avec le temps, la communauté du logarithme discret a décidé que certaines hypothèses heuristiques, très utilisées, étaient recevables. Bien entendu, un des objectifs du domaine est de proposer des algorithmes sans aucune heuristique. Cependant, dans un souci de praticité, on préfère un algorithme heuristique rapide à un algorithme prouvé un peu plus lent.

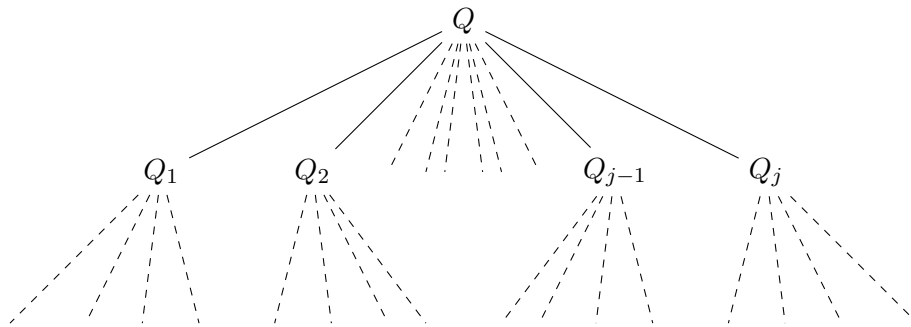


FIGURE 2.1 – L’arbre de descente

Pratique On a parfois des résultats théoriques asymptotiques révolutionnaires, mais inutilisables en pratique. Un exemple vient de l’algorithme permettant de prouver qu’un nombre est premier en temps polynomial, qui n’est pas utilisé en pratique car le degré du polynôme intervenant dans la complexité est trop élevé. Dans le cas du logarithme discret dans les corps finis de petite caractéristique, les résultats théoriques ont eu un réel impact pratique. Certains calculs, complètement hors de portée avec les précédents algorithmes, ont été (difficilement) réalisés. Certains protocoles cryptographiques “sécurisés” ont également été attaqués en utilisant ces nouveaux résultats. On peut tout de même nuancer le propos, car s’il est vrai qu’il y a eu des retombées pratiques, les constantes cachées dans la notation $\ell^{O(\log \ell)}$ (non précisément analysées à ma connaissance) sont assez grosses pour rendre les calculs coûteux.

Problèmes ouverts Comme on l’a déjà dit, un objectif naturel est de prouver les heuristiques utilisées dans les algorithmes. Un deuxième objectif, tout aussi naturel, est de diminuer les temps de calcul. Pour y arriver, on peut chercher à changer la nature de la complexité asymptotique, c’est-à-dire trouver un nouvel algorithme avec une complexité véritablement polynomiale. Une autre solution est d’améliorer les présents algorithmes quasi-polynomiaux, ou en d’autres termes diminuer les constantes cachées dans la notation $O(\cdot)$, en retravaillant les étapes essentielles des algorithmes.

Chapitre 3

Projet de L3

Introduction

On ne démontre pas la transcendance de e et π dans les pages qui suivent, mais simplement l'irrationalité de e . Cependant, la méthode employée pour arriver à ce résultat vient de la simplification du théorème 3.3.2 ci-dessous, dans lequel on travaille initialement dans un contexte plus large. Le présent document présente les grandes idées de la preuve écrite par S. Lang dans son livre Algebra.

3.1 Résultats préliminaires

On va ici donner quelques définitions et présenter des résultats.

Définition 3.1.1. Soit A un anneau, qui soit de plus un \mathbb{Q} -espace vectoriel. On dit que l'application $\bar{D} : A \rightarrow A$ de A dans lui même est une dérivation si elle est \mathbb{Q} -linéaire et si elle vérifie :

$$\forall (x, y) \in A \times A, \bar{D}(xy) = x\bar{D}(y) + y\bar{D}(x) \quad (3.1)$$

Ce qui nous amène à poser un lemme :

Lemme 3.1.2

Supposons qu'il existe une dérivation \bar{D} sur $A = \mathbb{Q}[T_1, \dots, T_N]$ et des polynômes P_i sur A tels que :

$$\forall i \in \llbracket 1, N \rrbracket, \bar{D}(T_i) = P_i(T_1, \dots, T_N)$$

une telle dérivation est alors unique.

Preuve :

Soit deux dérivations \bar{D} et \tilde{D} sur A vérifiant

$$\forall i \in \llbracket 1, N \rrbracket, \bar{D}(T_i) = \tilde{D}(T_i) = P_i(T_1, \dots, T_N).$$

Soit Q un polynôme de A , on note d son degré, et on écrit $Q = \sum_{\nu} q_{\nu} M_{\nu}(T_1, \dots, T_N)$

où $\nu = (\nu_1, \dots, \nu_N) \in \mathbb{N}^N$ est un multi-indice tel que $\sum_{i=1}^N \nu_i \leq d$ et où les

$M_{\nu}(T_1, \dots, T_N) = \prod_{i=1}^N T_i^{\nu_i}$ sont des monômes de degré au plus d .

Par hypothèse, on a que pour tout monôme M de degré 1, $\overline{D}(M) = \tilde{D}(M)$.

Supposons qu'il existe un entier k tel que si M est un monôme de degré k , alors $\overline{D}(M) = \tilde{D}(M)$.

Soit alors un monôme \overline{M} de degré $k+1$, on décompose \overline{M} en un monôme de degré 1 multiplié par un monôme de degré k . On choisit donc $i \in \llbracket 1, N \rrbracket$ tel que $\overline{M} = T_i M'$ où M' est un monôme de degré k .

$$\begin{aligned} \text{Il vient } \overline{D}(\overline{M}) &= T_i \overline{D}(M') + M' \overline{D}(T_i) \text{ d'après (3.1)} \\ &= T_i \tilde{D}(M') + M' \tilde{D}(T_i) \text{ par hypothèse de récurrence} \\ &= \tilde{D}(\overline{M}) \end{aligned}$$

On a donc prouvé par récurrence que si M est un monôme, on a $\overline{D}(M) = \tilde{D}(M)$, on a donc :

$$\begin{aligned} \overline{D}(Q) &= \overline{D}\left(\sum_{\nu} q_{\nu} M_{\nu}(T_1, \dots, T_N)\right) \\ &= \sum_{\nu} q_{\nu} \overline{D}(M_{\nu}(T_1, \dots, T_N)) \text{ par } \mathbb{Q}\text{-linéarité} \\ &= \sum_{\nu} q_{\nu} \tilde{D}(M_{\nu}(T_1, \dots, T_N)) \\ &= \tilde{D}(Q) \end{aligned}$$

Et ce pour tout polynôme $Q \in A$, d'où $\overline{D} = \tilde{D}$.

□

Définition 3.1.3. Soit $P \in \mathbb{Q}[T_1, \dots, T_N]$ et $Q \in \mathbb{R}[T_1, \dots, T_N]$ tels que :

$$\begin{aligned} P &= \sum_{\nu} \alpha_{\nu} M_{\nu}(T_1, \dots, T_N) \\ Q &= \sum_{\nu} \beta_{\nu} M_{\nu}(T_1, \dots, T_N) \end{aligned}$$

et où $0 \leq \beta_\nu$ pour tout ν .

On dit que Q domine P et on note $P \preceq Q$, si pour tout multi-indice ν , $|\alpha_\nu| \leq \beta_\nu$.

Lemme 3.1.4

Soit $P, P' \in \mathbb{Q}[T_1, \dots, T_N]$ et $Q, Q' \in \mathbb{R}[T_1, \dots, T_N]$ avec Q et Q' à coefficients positifs. On suppose que $P \preceq Q$ et $P' \preceq Q'$, alors :

$$P + P' \preceq Q + Q'$$

$$PP' \preceq QQ'$$

Preuve :

Supposons qu'on est dans les conditions du lemme, on note :

$$P = \sum_{\nu} \alpha_{\nu} M_{\nu}(T_1, \dots, T_N), \quad P' = \sum_{\nu} \alpha'_{\nu} M_{\nu}(T_1, \dots, T_N)$$

$$Q = \sum_{\nu} \beta_{\nu} M_{\nu}(T_1, \dots, T_N), \quad Q' = \sum_{\nu} \beta'_{\nu} M_{\nu}(T_1, \dots, T_N)$$

on a alors :

$$\begin{aligned} \forall \nu \quad |\alpha_{\nu} + \alpha'_{\nu}| &\leq |\alpha_{\nu}| + |\alpha'_{\nu}| \text{ d'après l'inégalité triangulaire} \\ &\leq \beta_{\nu} + \beta'_{\nu} \text{ par hypothèse de domination} \end{aligned}$$

ainsi $P + P' \preceq Q + Q'$.

On a également

$$PP' = \sum_{\nu} \left(\sum_{\nu_1 + \nu_2 = \nu} \alpha_{\nu_1} \alpha'_{\nu_2} \right) M_{\nu}(T_1, \dots, T_N)$$

$$QQ' = \sum_{\nu} \left(\sum_{\nu_1 + \nu_2 = \nu} \beta_{\nu_1} \beta'_{\nu_2} \right) M_{\nu}(T_1, \dots, T_N)$$

or

$$\begin{aligned} |\alpha_{\nu_1} \alpha'_{\nu_2}| &= |\alpha_{\nu_1}| |\alpha'_{\nu_2}| \\ &\leq \beta_{\nu_1} \beta'_{\nu_2} \end{aligned}$$

d'où $PP' \preceq QQ'$ par inégalité triangulaire.

□

Lemme 3.1.5

Soit $P \in \mathbb{Q}[T_1, \dots, T_N]$ et $Q \in \mathbb{R}[T_1, \dots, T_N]$ tels que Q ne possède que des coefficients positifs, et tels que $P \preceq Q$. On pose $\omega = (\omega_1, \dots, \omega_N) \in \mathbb{Q}^N$.

Alors $|P(\omega_1, \dots, \omega_N)| \leq Q(|\omega_1|, \dots, |\omega_N|)$.

Preuve :

Soit P et Q comme dans le lemme. On considère $\omega = (\omega_1, \dots, \omega_N) \in \mathbb{Q}^N$ et on note :

$$P = \sum_{\nu} \alpha_{\nu} M_{\nu}(T_1, \dots, T_N), \quad Q = \sum_{\nu} \beta_{\nu} M_{\nu}(T_1, \dots, T_N)$$

Il vient :

$$\begin{aligned} |P(\omega_1, \dots, \omega_N)| &= \left| \sum_{\nu} \alpha_{\nu} M_{\nu}(\omega_1, \dots, \omega_N) \right| \\ &\leq \sum_{\nu} |\alpha_{\nu}| |M_{\nu}(\omega_1, \dots, \omega_N)| \\ &\leq \sum_{\nu} |\beta_{\nu}| |M_{\nu}(\omega_1, \dots, \omega_N)| \end{aligned}$$

or, si k est un entier, on a $|\omega_i|^k = |\omega_i^k|$ pour tout $i \in \llbracket 1, N \rrbracket$.

D'où $|M_{\nu}(\omega_1, \dots, \omega_N)| = M_{\nu}(|\omega_1|, \dots, |\omega_N|)$. Ainsi :

$$\begin{aligned} |P(\omega_1, \dots, \omega_N)| &\leq \sum_{\nu} \beta_{\nu} M_{\nu}(|\omega_1|, \dots, |\omega_N|) \\ &\leq Q(|\omega_1|, \dots, |\omega_N|). \end{aligned}$$

□

On définit maintenant les fonctions holomorphes d'ordre inférieur à ρ , qui seront importantes dans la suite.

Définition 3.1.6. Soit f une fonction entière, on dit que f est d'ordre inférieur à ρ si :

$$\exists C > 1 \exists r \in \mathbb{R} \forall R > r \forall z \in \mathbb{C}, |z| \leq R \Rightarrow |f(z)| \leq C^{R^{\rho}}$$

On dit qu'une fonction méromorphe est d'ordre inférieur à ρ si elle est le quotient de deux fonctions entières d'ordres inférieurs à ρ .

Lemme 3.1.7

Si f et g sont deux fonctions méromorphes d'ordres inférieurs à ρ , alors fg est d'ordre inférieur à ρ .

Preuve :

f est d'ordre inférieur à ρ , d'où :

$$\exists C_f > 1 \exists r_f \in \mathbb{R} \forall R > r_f \forall z \in \mathbb{C}, |z| \leq R \Rightarrow |f(z)| \leq C_f^{R^\rho}$$

g est d'ordre inférieur à ρ , d'où :

$$\exists C_g > 1 \exists r_g \in \mathbb{R} \forall R > r_g \forall z \in \mathbb{C}, |z| \leq R \Rightarrow |g(z)| \leq C_g^{R^\rho}$$

on a alors :

$$\begin{aligned} \forall R > \max(r_f, r_g) \forall z \in \mathbb{C}, |z| \leq R \Rightarrow |f(z)g(z)| &= |f(z)||g(z)| \\ &\leq C_f^{R^\rho} C_g^{R^\rho} \\ &\leq \max(C_f, C_g)^{R^\rho} \end{aligned}$$

donc fg est d'ordre inférieur à ρ .

□

3.2 Le cas irrationnel

Lemme 3.2.1 (de Siegel)

On considère le système linéaire à coefficients entiers a_{ij} (où $n > r$) suivant :

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = 0 \\ \vdots \\ a_{r1}x_1 + \cdots + a_{rn}x_n = 0 \end{cases}$$

Soit A un nombre tel que $\forall (i, j) \in \llbracket 1, r \rrbracket \times \llbracket 1, n \rrbracket, |a_{ij}| \leq A$.

Alors il existe une solution non triviale (i.e. non nulle) à coefficients entiers telle que $\forall j \in \llbracket 1, n \rrbracket, |x_j| \leq 2(3nA)^{\frac{r}{n-r}}$.

Preuve :

On considère la matrice M formée des coefficients a_{ij} , et L l'application linéaire de \mathbb{Z}^n dans \mathbb{Z}^r canoniquement associée.

On note, pour X un vecteur de \mathbb{Z}^n , $|X|$ le maximum de la valeur absolue de ses coefficients. Et si B est un nombre positif, on note $\mathbb{Z}^n(B)$ l'ensemble des vecteurs X de \mathbb{Z}^n tels que $|X| \leq B$.

On a alors $L(\mathbb{Z}^n(B)) \subset \mathbb{Z}^r(nAB)$, en effet :

$$\forall (i, j) \in \llbracket 1, r \rrbracket \times \llbracket 1, n \rrbracket, |a_{ij}| \leq A \text{ et } \forall k \in \llbracket 1, r \rrbracket, |x_k| \leq B$$

$$\begin{aligned} \text{d'où } \forall i \in \llbracket 1, r \rrbracket, \left| \sum_{j=1}^n a_{ij} x_j \right| &\leq \sum_{j=1}^n |a_{ij}| |x_j| \\ &\leq \sum_{j=1}^n AB \\ &\leq nAB. \end{aligned}$$

On remarque ensuite que $\text{Card}(\mathbb{Z}^n(B)) = (2\lfloor B \rfloor + 1)^n$: on a $2\lfloor B \rfloor + 1$ choix pour chaque coordonnée. On en déduit :

$$\begin{aligned} B^n &\leq \text{Card}(\mathbb{Z}^n(B)) \\ &\leq (2\lfloor B \rfloor + 1)^n. \end{aligned}$$

On va désormais chercher deux vecteurs distincts X et Y tels que $L(X) = L(Y)$, pour cela, il suffit qu'il y ait strictement plus d'éléments dans $\mathbb{Z}^n(B)$ que dans $\mathbb{Z}^r(nAB)$. Or $B^n \leq \text{Card}(\mathbb{Z}^n(B))$ et $\text{Card}(\mathbb{Z}^r(nAB)) \leq (2nAB + 1)^r$.

Comme $1 < 2nAB$, on a $(2nAB + 1)^r < (3nAB)^r$, il suffit que $(3nAB)^r \leq B^n$, ainsi $B = (3nAB)^{\frac{r}{n-r}}$ convient. On a alors $L(X) - L(Y) = L(X - Y) = 0$, avec $X - Y$ non nul, et, en notant x_j (respectivement y_j) les coordonnées de X (respectivement Y), il vient :

$$\begin{aligned} \forall j \in \llbracket 1, n \rrbracket, |x_j - y_j| &\leq |x_j| + |y_j| \\ &\leq 2(3nAB)^{\frac{r}{n-r}}. \end{aligned}$$

$X - Y$ satisfait donc aux conditions du lemme. □

Ce deuxième et dernier lemme avant la démonstration finale du théorème (qui n'a pas encore été énoncé) traite de polynômes et de bornes sur les coefficients de leurs dérivées.

Définition 3.2.2. Si P est un polynôme à coefficients dans \mathbb{Q} , on note $\|P\|$ le maximum de la valeur absolue de ses coefficients. On note également $\text{den}(P)$ un dénominateur de P , c'est-à-dire un entier positif tel que le produit de $\text{den}(P)$ avec n'importe quel coefficient de P soit un entier.

Lemme 3.2.3

Soit $N \in \mathbb{N}$, $N \geq 2$ et f_1, \dots, f_N des fonctions holomorphes au voisinage d'un point $\omega \in \mathbb{C}$. On suppose que l'anneau $\mathbb{Q}[f_1, \dots, f_N]$ est stable pour la dérivation complexe $D = \frac{d}{dz}$ et que $\forall i \in \llbracket 1, N \rrbracket$ $f_i(\omega) \in \mathbb{Q}$.

Alors il existe un nombre C_3 possédant les propriétés suivantes :

Soit $P \in \mathbb{Q}[T_1, \dots, T_N]$ un polynôme de degré inférieur ou égal à r . Si on pose $f = P(f_1, \dots, f_N)$, alors on a :

$$\forall k \in \mathbb{N}, |D^k f(\omega)| \leq \|P\| r^k k! C_3^{k+r}.$$

De plus, il existe un dénominateur pour $D^k f(\omega)$ majoré par $\text{den}(P) C_3^{k+r}$.

Preuve :

Comme $\mathbb{Q}[f_1, \dots, f_N]$ est stable par D , il existe des polynômes $P_i \in \mathbb{Q}[T_1, \dots, T_N]$ tels que $Df_i = P_i(f_1, \dots, f_N)$. On note alors h le maximum de leurs degrés.

On admet qu'il existe une dérivation \overline{D} sur l'anneau $\mathbb{Q}[T_1, \dots, T_N]$ telle que $\forall i \in \llbracket 1, N \rrbracket$, $\overline{D}(T_i) = P_i(T_1, \dots, T_N)$. Une telle dérivation est alors unique d'après le lemme 3.1.2.

On considère désormais les relations de dominations sur les polynômes, dont on a parlé dans la définition 3.1.3.

Comme le polynôme $(1 + T_1 + \dots + T_N)^d$ fait apparaître tous les monômes de degré inférieur à d , on a :

$$\forall Q \in \mathbb{Q}[T_1, \dots, T_N], Q \preceq \|Q\| (1 + T_1 + \dots + T_N)^{\text{deg}(Q)}.$$

D'où :

$$P \preceq \|P\| (1 + T_1 + \dots + T_N)^r \tag{3.2}$$

$$\forall i \in \llbracket 1, N \rrbracket P_i \preceq \|P_i\| (1 + T_1 + \dots + T_N)^h \tag{3.3}$$

On va prouver par récurrence que pour tout entier k , on a :

$$D^k(P) \preceq \|P\| C_1^k r^k k! (1 + T_1 + \dots + T_N)^{r+kh} \tag{3.4}$$

avec C_1 une constante qui vérifie :

$$h \sum_{i=1}^N \|P_i\| \leq C_1. \tag{3.5}$$

La propriété est vraie pour $k = 0$ d'après (3.2).

Supposons désormais qu'il existe un entier k tel que la relation (3.4) soit vraie.

On va alors utiliser la relation suivante (qui découle de la définition d'une dérivation) :

$$\begin{aligned}\overline{D}(P) &= \sum_{i=1}^N D_i(P) \overline{D}(T_i) \\ &= \sum_{i=1}^N D_i(P) P_i \quad (*)\end{aligned}$$

avec D_i les dérivées partielles par rapport aux T_i .

Lorsqu'on dérive partiellement un polynôme Q de degré d par rapport à une indéterminée, on multiplie les coefficients par d au plus, en effet les monômes qui constituent Q sont de degré au plus d . Ainsi, il vient :

$$D_i(\overline{D}^k(P)) \preceq (r + kh) \|P\| C_1^k r^k k! (1 + T_1 + \dots + T_N)^{r+kh}$$

car $\overline{D}^k(P)$ est de degré inférieur à $r + kh$.

En réutilisant la relation (*), les relations de dominations (3.3), ainsi que les résultats du lemme 3.1.4, on a donc :

$$\begin{aligned}\overline{D}^{k+1}(P) &\preceq \left(\sum_{i=1}^N \|P_i\| \right) \|P\| C_1^k (r + kh) r^k k! (1 + T_1 + \dots + T_N)^{r+kh+k} \\ &\preceq \left(\sum_{i=1}^N \|P_i\| \right) \|P\| C_1^k (r^{k+1} k! + r^k k h k!) (1 + T_1 + \dots + T_N)^{r+(k+1)h} \\ &\preceq \left(\sum_{i=1}^N \|P_i\| \right) \|P\| C_1^k (r^{k+1} k! h + r^{k+1} k h k!) (1 + T_1 + \dots + T_N)^{r+(k+1)h} \quad \text{car } h, r \geq 1 \\ &\preceq h \left(\sum_{i=1}^N \|P_i\| \right) \|P\| C_1^k r^{k+1} (k+1)! (1 + T_1 + \dots + T_N)^{r+(k+1)h} \\ &\preceq \|P\| C_1^{k+1} r^{k+1} (k+1)! (1 + T_1 + \dots + T_N)^{r+(k+1)h} \quad \text{d'après (3.5)}\end{aligned}$$

ce qui achève la récurrence.

Or on a $\overline{D}^k(P)(f_1(\omega), \dots, f_N(\omega)) = D^k(f)(\omega)$.

En remplaçant les T_i par des $f_i(\omega)$ dans la relation de domination (3.4),

il vient d'après le lemme 3.1.5 :

$$\begin{aligned}
|D^k(f)(\omega)| &\leq \|P\| C_1^k r^k k! (1 + |f_1(\omega)| + \cdots + |f_N(\omega)|)^{r+kh} \\
&\leq \|P\| C_1^k r^k k! C_2^{r+kh} \text{ avec } C_2 = 1 + |f_1(\omega)| + \cdots + |f_N(\omega)| \\
&\leq \|P\| C_1^k r^k k! C_2^r (C_2^h)^k \\
&\leq \|P\| (C_1 C_2^h)^k r^k k! C_2^r \\
&\leq C_3^{k+h} \|P\| r^k k! \text{ où } C_3 = \max(C_1 C_2^h, C_2)
\end{aligned}$$

On admet le résultat sur les dénominateurs, qui se prouve par récurrence. □

3.3 Le théorème

On va maintenant énoncer le théorème (dans le cas irrationnel) et le démontrer, grâce aux lemmes précédemment établis. Mais une définition s'impose tout d'abord :

Définition 3.3.1. On dit que deux fonctions f et g sont algébriquement indépendantes sur \mathbb{Q} s'il n'existe aucun polynôme non nul P de $\mathbb{Q}[T_1, T_2]$ tel que :

$$\forall z \in \mathbb{C}, P(f(z), g(z)) = 0.$$

Théorème 3.3.2

Soit f_1, \dots, f_N des fonctions méromorphes d'ordre inférieur à ρ , on suppose que l'anneau $\mathbb{Q}[f_1, \dots, f_N]$ est stable pour la dérivation complexe $D = \frac{d}{dz}$, et qu'il existe deux fonctions parmi les f_1, \dots, f_N qui sont algébriquement indépendantes sur \mathbb{Q} . Soit de plus $\omega_1, \dots, \omega_m$ des complexes n'appartenant pas aux pôles des f_1, \dots, f_N et tels que :

$$\forall i \in [1, N] \forall \nu \in [1, m], f_i(\omega_\nu) \in \mathbb{Q}.$$

Alors $m \leq 10\rho$.

Preuve :

Soit f et g deux fonctions parmi les f_1, \dots, f_N algébriquement indépendantes sur \mathbb{Q} , soit $r \in \mathbb{N}$ divisible par $2m$. On pose alors :

$$F = \sum_{i,j=1}^r b_{ij} f^i g^j$$

où les coefficients b_{ij} appartiennent à \mathbb{Q} .

On pose maintenant $n = \frac{r^2}{2m}$. On souhaite choisir les coefficients b_{ij} de telle sorte que :

$$\forall k \in \llbracket 0, n-1 \rrbracket \forall \nu \in \llbracket 1, m \rrbracket, D^k F(\omega_\nu) = 0$$

ce qui équivaut à résoudre un système linéaire de mn équations à r^2 inconnues (les coefficients b_{ij}) :

$$\forall k \in \llbracket 0, n-1 \rrbracket \forall \nu \in \llbracket 1, m \rrbracket, D^k(F)(\omega_\nu) = \sum_{i,j=1}^r b_{ij} D^k(f^i g^j)(\omega_\nu) = 0.$$

Or on a $r^2 = 2mn > mn$, on peut appliquer le lemme 3.2.1 : il existe une solution non triviale qui possède des coefficients bornés par $2(6mn\mathfrak{C}^{\frac{mn}{2mn-mn}})$ où :

$$\mathfrak{C}^{\frac{mn}{2mn-mn}} = \mathfrak{C} = \max_{\substack{k \in \llbracket 0, n-1 \rrbracket \\ \nu \in \llbracket 1, m \rrbracket \\ (i,j) \in \llbracket 1, r \rrbracket^2}} |D^k(f^i g^j)(\omega_\nu)|.$$

Alors, $\forall \nu \in \llbracket 1, m \rrbracket$ et $\forall (i, j) \in \llbracket 1, r \rrbracket \times \llbracket 1, r \rrbracket$, d'après les estimations du lemme 3.2.3, on a :

$$\begin{aligned} \forall k \in \mathbb{N}, |D^k(f^i g^j)(\omega_\nu)| &\leq (2r)^k k! C_3^{k+2r} \\ &\leq r^k k! 2^{k+r} C_3^{2k+2r} \\ &\leq r^k k! (2C_3^2)^{k+r}. \end{aligned}$$

On pose alors $C_4 = 2C_3^2 f$.

Il vient donc :

$$\mathfrak{C} \leq r^{n-1} (n-1)! C_4^{n-1+r}$$

puis :

$$\begin{aligned}
\forall (i, j) \in \llbracket 1, r \rrbracket \times \llbracket 1, r \rrbracket, |b_{ij}| &\leq 12mnr^{n-1}(n-1)!C_4^{m-1+r} \\
&\leq \frac{12m}{rC_4}r^n n!C_4^{n+r} \\
&\leq O(r^n n!C_4^{n+r}) \\
&\leq O(n^{2n}) \text{ lorsque } n \rightarrow +\infty
\end{aligned}$$

Comme f et g sont algébriquement indépendantes et que les coefficients b_{ij} ne sont pas tous nuls, la fonction F n'est pas identiquement nulle.

On pose alors s l'unique entier tel que :

$$\forall k \in \llbracket 0, s-1 \rrbracket \forall \nu \in \llbracket 1, m \rrbracket, D^k(F)(\omega_\nu) = 0$$

et tel qu'il existe un point parmi les $\omega_1, \dots, \omega_N$ qui n'annule pas $D^s(F)$. Quitte à réindexer les $\omega_1, \dots, \omega_N$, on peut supposer que ce point est ω_1 .

Suite à notre choix des b_{ij} , on a alors $n \leq s$, d'où :

$$\forall (i, j) \in \llbracket 1, r \rrbracket \times \llbracket 1, r \rrbracket, |b_{ij}| \leq O(s^{2s}) \quad (3.6)$$

Puis on pose $\gamma = D^s(F)(\omega_1) \neq 0$, $\gamma \in \mathbb{Q}$, il existe donc $(p, q) \in \mathbb{Z} \setminus \{0\} \times \mathbb{N}^*$ tels que $\gamma = \frac{p}{q}$ et $\text{pgcd}(p, q) = 1$.

On a alors $1 \leq q|\gamma|$. Puis, d'après le résultat sur les dénominateurs du lemme 3.2.3 :

$$q \leq O(C_3^s)$$

On va désormais chercher à majorer $|\gamma|$. On admet qu'il existe θ une fonction entière d'ordre inférieur à ρ telle que θf et θg soient entières et que $\theta(\omega_1) \neq 0$. Il vient alors que $\theta^{2r}F$ est entière, on considère alors :

$$H(z) = \frac{\theta(z)^{2r}F(z)}{\prod_{\nu=1}^m (z - \omega_\nu)^s}$$

qui est entière, en effet $\forall k \in \llbracket 0, s-1 \rrbracket \forall \nu \in \llbracket 1, m \rrbracket, D^k F(\omega_\nu) = 0$, les ω_ν sont donc des zéros d'ordre s de F , ce qui assure que H est bien définie et holomorphe sur \mathbb{C} .

Cherchons désormais à comparer $|D^s(F)(\omega_1)| = |\gamma|$ à $|H(\omega_1)|$. Les fonctions f et g sont holomorphes au voisinage de ω_1 , ainsi F est aussi holomorphe au voisinage de ω_1 . F est donc développable en série entière en ω_1 :

$$F(z) = \sum_{n=0}^{+\infty} \frac{D^n F(\omega_1)}{n!} (z - \omega_1)^n$$

Mais on a $\forall n \in \llbracket 0, s-1 \rrbracket$, $D^n F(\omega_1) = 0$ et $D^s F(\omega_1) \neq 0$, il vient alors :

$$\begin{aligned} F(z) &\underset{\omega_1}{\sim} \frac{D^s F(\omega_1)}{s!} (z - \omega_1)^s \\ \text{Puis } H(z) &\underset{\omega_1}{\sim} \frac{\theta(z)^{2r}}{\prod_{\nu=1}^m (z - \omega_\nu)^s} \frac{D^s F(\omega_1)}{s!} (z - \omega_1)^s \\ &\underset{\omega_1}{\sim} \frac{\theta(\omega_1)^{2r}}{\prod_{\nu=2}^m (\omega_1 - \omega_\nu)^s} \frac{D^s F(\omega_1)}{s!} \\ &\underset{\omega_1}{\sim} H(\omega_1) \end{aligned}$$

$$\text{D'où } D^s F(\omega_1) = \frac{s! \prod_{\nu=2}^m (\omega_1 - \omega_\nu)^s}{\theta(\omega_1)^{2r}} H(\omega_1).$$

Il vient alors $|D^s(F)(\omega_1)| = |\gamma| \leq C_5^s s! |H(\omega_1)|$ où C_5 est une constante.

H étant holomorphe sur \mathbb{C} , on peut appliquer le principe du maximum à tout disque $D(0, R)$ afin de majorer la valeur de $|H(\omega_1)|$. On remarque que :

$$\forall z \in \partial D(0, R), |z - \omega_\nu| \geq ||z| - |\omega_\nu|| = R \left| 1 - \frac{|\omega_\nu|}{R} \right| \text{ avec } \frac{|\omega_\nu|}{R} \xrightarrow{R \rightarrow +\infty} 0$$

Ainsi, il existe un réel r_0 tel que $\forall z \in \mathbb{C} \forall \nu \in \llbracket 1, m \rrbracket$, $|z| \geq r_0 \Rightarrow |z - \omega_\nu| \geq \frac{R}{2}$.
D'où $|z - \omega_\nu|^s \geq \frac{R^s}{2^s}$, puis enfin :

$$\frac{1}{\prod_{\nu=1}^m |z - \omega_\nu|^s} \leq \frac{2^{sm}}{R^{sm}} \quad (3.7)$$

De plus, θ , f et g sont d'ordre inférieur à ρ , le lemme 3.1.7 nous permet donc de conclure que θf et θg sont d'ordre inférieur à ρ .

On a alors :

$$\exists C_\theta > 1 \exists r_\theta \in \mathbb{R} \forall R > r_\theta \forall z \in \mathbb{C}, |z| \leq R \Rightarrow |\theta(z)| \leq C_\theta^{R^p}$$

$$\exists C_{\theta f} > 1 \exists r_{\theta f} \in \mathbb{R} \forall R > r_{\theta f} \forall z \in \mathbb{C}, |z| \leq R \Rightarrow |\theta(z)f(z)| \leq C_{\theta f}^{R^p}$$

$$\exists C_{\theta g} > 1 \exists r_{\theta g} \in \mathbb{R} \forall R > r_{\theta g} \forall z \in \mathbb{C}, |z| \leq R \Rightarrow |\theta(z)g(z)| \leq C_{\theta g}^{R^p}$$

on pose $C_6 = \max(C_\theta, C_{\theta f}, C_{\theta g})$, il nous vient donc, pour tout couple (i, j) appartenant à $[[1, r]] \times [[1, r]]$:

$$\forall R > \max(r_\theta, r_{\theta f}, r_{\theta g}) \forall z \in \mathbb{C}, |z| \leq R \Rightarrow |\theta(z)^{2r} f(z)^i g(z)^j| \leq C_6^{2rR^p} \quad (3.8)$$

D'où, en prenant R assez grand,

$$\begin{aligned} \forall z \in \partial D(0, R) \quad \frac{|\theta(z)^{2r} \sum_{i,j=1}^r b_{ij} f(z)^i g(z)^j|}{\prod_{\nu=1}^m |z - \omega_\nu|^s} &= \frac{|\sum_{i,j=1}^r b_{ij} \theta(z)^{2r} f(z)^i g(z)^j|}{\prod_{\nu=1}^m |z - \omega_\nu|^s} \\ &\leq \frac{\sum_{i,j=1}^r |b_{ij}| |\theta(z)^{2r} f(z)^i g(z)^j|}{\prod_{\nu=1}^m |z - \omega_\nu|^s} \\ &\leq \frac{2^{sm} \sum_{i,j=1}^r C s^{2s} C_6^{2rR^p}}{R^{sm}} \text{ d'après (3.6), (3.7), (3.8)} \\ &\leq \frac{s^{2s} r^2 C 2^{sm} C_6^{2rR^p}}{R^{sm}} \text{ or } r^2 = 2mn \\ &\leq \frac{s^{2s} 2mn C 2^{sm} C_6^{2rR^p}}{R^{sm}} \\ &\leq \frac{s^{2s+1} 2(2^m)^s m C C_6^{2rR^p}}{R^{sm}} \text{ car } s \geq n \\ &\leq \frac{s^{2s+1} 2(2^m)^s (m C C_6)^{2rR^p}}{R^{sm}} \\ &\leq \frac{s^{3s} C_7^{2rR^p}}{R^{sm}} \text{ pour un } s \text{ assez grand} \end{aligned}$$

On peut donc écrire, d'après le principe du maximum :

$$H(z) \leq \frac{s^{3s} C_7^{2rR^p}}{R^{ms}}$$

Puis on prend $R = s^{\frac{1}{2\rho}}$, il vient alors :

$$\begin{aligned}
|\gamma| &\leq C_5^s s! \frac{s^{3s} C_7^{2rR^\rho}}{R^{ms}} \\
&\leq C_5^s s! \frac{s^{3s} C_7^{2rs \frac{1}{2}}}{s^{\frac{ms}{2\rho}}} \\
&\leq \frac{s^s s^{3s} C_5^s C_7^{2rs}}{s^{\frac{ms}{2\rho}}} \\
&\leq \frac{s^{4s} C_8^s}{s^{\frac{ms}{2\rho}}}
\end{aligned}$$

On fait alors tendre r vers l'infini, ce qui implique que n et s tendent vers l'infini également. On a donc :

$$1 \leq q|\gamma| \leq C_3^s \frac{s^{4s} C_8^s}{s^{\frac{ms}{2\rho}}}$$

Or $C_3^s \frac{s^{4s} C_8^s}{s^{\frac{ms}{2\rho}}}$ a la même limite que $\frac{s^{4s}}{s^{\frac{ms}{2\rho}}} = s^{4s - \frac{ms}{2\rho}}$

On en déduit que $4s - \frac{ms}{2\rho} \geq 0$ (sinon $s^{4s - \frac{ms}{2\rho}} \xrightarrow{s \rightarrow +\infty} 0$) On a alors l'inégalité souhaitée :

$$m \leq 10\rho$$

Ce qui conclut la preuve du théorème.

□

3.4 Irrationalité de e

On considère les fonctions identité Id et exponentielle exp. Elles sont toutes deux holomorphes donc méromorphes, d'ordre inférieur à 1, elles sont algébriquement indépendantes et l'anneau $\mathbb{Q}[\text{Id}, \text{exp}]$ est stable pour la dérivation complexe. Les hypothèses du théorème sont ainsi satisfaites.

Supposons par l'absurde que e est rationnel. On a alors :

$$\forall \alpha \in \mathbb{N} \alpha, e^\alpha \in \mathbb{Q}$$

Or m est borné, on a donc une contradiction. Ainsi e est irrationnel.

□

Chapitre 4

Stage de magistère

4.1 Préambule

This report presents a magistère internship realised in L3 MFA. It was effected at the LRI, Laboratoire de Recherche en Informatique, the Laboratory for Computer Science at Université Paris-Sud. We first give some theory on Coxeter groups and Markov chains, then we recall a bijection between rhombic tilings of the regular $2n$ -gon and the commutation classes of the reduced words of the longest element of \mathfrak{S}_n . We next present a random walk on the set of the commutation classes and we study the properties of the associated Markov Chains, called “exchange walk”. All the computing parts, in particular the implementation of the programs, were realised with the mathematics free software Sage.

4.2 Some theory

4.2.1 Coxeter groups

The Coxeter groups are groups which have a certain structure, in all the paper we will consider the symmetric group \mathfrak{S}_n , which is a simple example of Coxeter group.

Définition 4.2.1. Let W be a group, we say that W is a Coxeter group if $W = \langle S \rangle$ with

$$S = \{s_1, \dots, s_n \mid \forall (i, j) \in \llbracket 1, n \rrbracket^2 \exists m(i, j) \in \mathbb{N}, (s_i s_j)^{m(i, j)} = 1\}$$

where $m(i, i) = 1$ and $m(i, j) = m(j, i) \geq 2$ if $i \neq j$. Moreover, (W, S) is called a Coxeter system.

Définition 4.2.2. For any $w \in W$, the least k such that w can be written as $w = s_{i_1} s_{i_2} \dots s_{i_k}$ is called the length of w and denoted by $l(w)$. An ordered k -tuple $z = (i_1, \dots, i_k)$ with $w = s_{i_1} \dots s_{i_k}$ and $k = l(w)$ is called a reduced word for w .

We can see that the symmetric group \mathfrak{S}_n is a Coxeter group, generated by the elementary transpositions $\tau_i = (i \ i + 1)$. Indeed $\tau_i \tau_i = Id$ and either τ_i and τ_j commute or $\tau_i \tau_j$ is a cycle if $i \neq j$.

4.2.2 Markov chains

We consider a sequence of random variables $\{X_n\}_{n \in \mathbb{N}}$ which take their values in E . E is called the state space, in all the document we will work with finite state spaces. We define Markov chains as follow.

Définition 4.2.3. The sequence $\{X_n\}_{n \in \mathbb{N}}$ is called a Markov chain if for all integer $n \in \mathbb{N}$, for all states $i_0, \dots, i_{n-1}, i, j \in E$:

$$P(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \dots, X_0 = i_0) = P(X_{n+1} = j \mid X_n = i).$$

We also say that the Markov chain is time-homogeneous if the second term of the expression above is independant of n .

Définition 4.2.4. The transition matrix of the Markov Chain is the matrix $P = \{p_{ij}\}_{i,j \in E}$ where

$$p_{ij} = P(X_1 = j \mid X_0 = i)$$

is the probability to go from state i to state j .

4.3 Commutation classes of the reduced words

Let $\sigma \in \mathfrak{S}_n$ be a permutation, we consider $V(\sigma)$ the set of reduced words for σ , which are defined in definition 4.2.2. We know that τ_i and τ_j commute if and only if $|i - j| > 1$, we call that a trivial commutation. Let $z = (z_1, \dots, z_k) \in V(\sigma)$ be a reduced word, assuming that there exists i such that $|z_i - z_{i+1}| > 1$, we have :

$$\sigma = \tau_{z_1} \dots \tau_{z_i} \tau_{z_{i+1}} \dots \tau_{z_k} = \tau_{z_1} \dots \tau_{z_{i+1}} \tau_{z_i} \dots \tau_{z_k}$$

so $z' = (z_1, \dots, z_{i+1}, z_i, \dots, z_k)$ is another reduced word, we say that z and z' are equivalent and we denote this relation by $z \mathcal{R} z'$. More generally, if we have two reduced words x and y , we say that $x \mathcal{R} y$ if we can obtain y from x using only trivial commutations. We can see that \mathcal{R} is an equivalence relation, thus we can consider the classes of equivalence, denoted by $W(\sigma)$. For each commutation class, we chose the maximal element of the class with respect to the lexical order and we identify the commutation class with this element.

Proposition 4.3.1

Let $z = (z_1, \dots, z_k) \in V(\sigma)$ be a reduced word of some permutation σ , z is maximal (with respect to the lexical order) if and only if

$$\forall i \in \llbracket 1, k - 1 \rrbracket, z_{i+1} - z_i < 2.$$

Démonstration. Let $z = (z_1, \dots, z_k) \in V(\sigma)$ be a reduced word of a certain permutation. Assume that $\forall i \in \llbracket 1, k-1 \rrbracket$, $z_{i+1} - z_i < 2$, if z is not maximal, there exists an index $j \in \llbracket 1, k-1 \rrbracket$ such that $z_j < z_{j+1}$ and τ_{z_j} and $\tau_{z_{j+1}}$ commute. Thus we have $z_j < z_{j+1}$ and $|z_j - z_{j+1}| > 1$, hence $z_{j+1} - z_j > 1$ and finally $z_{j+1} - z_j \geq 2$: contradiction. Assume now that z is maximal, and that there exists an index $j \in \llbracket 1, k-1 \rrbracket$ such that $z_{j+1} - z_j \geq 2$, hence $|z_j - z_{j+1}| > 1$ and $z_j < z_{j+1}$ so $z' = (z_1, \dots, z_{j+1}, z_j, \dots, z_k)$ is equivalent to z and greater : contradiction with the maximality of z . \square

This characterisation is a constructive way to test if a word is maximal or not. In a program that returns the set of the reduced word for a permutation, adding this condition permits to obtain only the maximal words, and therefore only the classes of commutation of the reduced words for that permutation. You can find the code of the program that tests if an element is maximal or not in appendix 4.7.1, and the code of the program that creates the list of the commutation classes, adapted from another program that returns all the reduced words, in appendix 4.7.2.

4.4 The bijection between tilings and commutation classes

We consider in all that part a positive integer $n \in \mathbb{N}$. For any permutation $\sigma \in \mathfrak{S}_n$, we build a $2n$ -gon $X(\sigma)$ as follow :

- Let M be the uppermost vertex, and M' the vertex at its antipode
- We build the left part of a regular $2n$ -gon linking M to M' .
- We label the sizes $1, \dots, n$ counter-clockwise from M
- We label the sizes $\sigma(1), \dots, \sigma(n)$ clockwise from M , and we arrange them such that the sizes with equal labels are parallel and have the same size.

For $\sigma \in \mathfrak{S}_n$ and $X(\sigma)$ the associated polygon, we denote by $W(\sigma)$ the set of classes of commutation of the reduced words for σ and by $T(\sigma)$ the set of rhombic tilings of $X(\sigma)$.

Théorème 4.4.1

There exists a bijection between $W(\sigma)$ and $T(\sigma)$.

We will now present a constructive way to implement the bijection in one direction : how to obtain a tiling from a reduced word, the code is foundable in 4.7.4. Nevertheless, details of the entire proof, in particular how to produce a reduced word from a tiling, can be found in [15].

Let σ be a permutation of \mathfrak{S}_n , and $z \in W(\sigma)$ a reduced word for σ . We have $z = (z_1, z_2, \dots, z_m)$ with $z_1, z_2, \dots, z_m \in \llbracket 1, n-1 \rrbracket$. We define as a

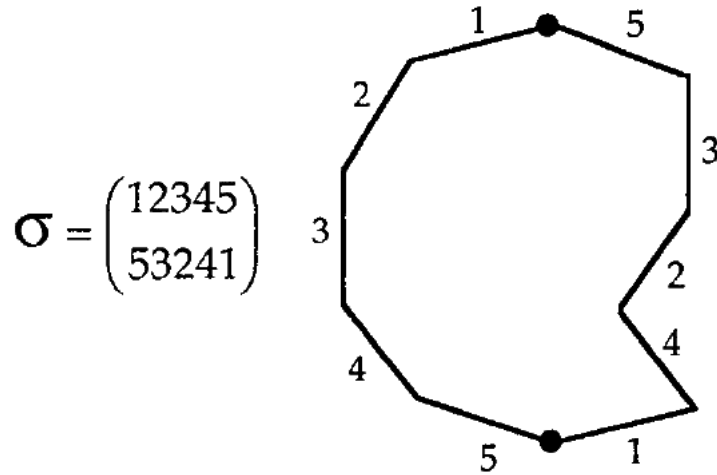


FIGURE 4.1 – The $2n$ -gon, an example for $n = 5$. Image taken from [15].

border a path from M to M' , composed of labeled sizes and viewed as a permutation of \mathfrak{S}_n read from M to M' . For example the left part of a regular $2n$ -gon would be $(1 \ 2 \ \dots \ n)$. The figure 4.2 shows two superposed borders. In order to build a rhombic tiling of $X(\sigma)$ we draw the border $Id = (1 \ \dots \ n)$, then we superpose to it the border $Id \circ \tau_{z_1}$ where τ_i is the elementary transposition $(i \ i+1)$. For example, in figure 4.2, we are in the case $z_1 = 2$. Then we apply τ_{z_2} to the border and superpose it to the drawing, and we do the same with z_3, \dots, z_m . Finally we get the polygon $X(\sigma)$ tiled with rhombi. In fact, applying τ_i to the border and then superpose it to the drawing corresponds to create a rhombus using the i -th and the $(i+1)$ -th part of the last-drawn border. If two words z and z' are equivalent, they produce the same tiling : indeed two indices z_i and z_{i+1} that commute in the word z verify $|z_i - z_{i+1}| > 1$, hence they create rhombi that have no size in common. Since we obtain z' from z only using trivial commutations, z and z' produce the same tiling.

In particular, for the permutation $w_0 = (n \ \dots \ 1) \in \mathfrak{S}_n$, which is the longest element of \mathfrak{S}_n , the polygon $X(w_0)$ obtained is the regular $2n$ -gon, thus the reduced words for w_0 are in bijection with tilings of the regular $2n$ -gon by rhombi. In the figure 4.3, the rhombi are labeled with their order of appearance, the reduced word considered here is a reduced word for $(5 \ 4 \ 3 \ 2 \ 1)$, the longest element of \mathfrak{S}_5 , thus we obtain a tiling of a decagone.

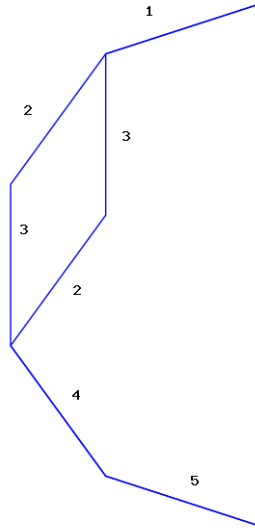


FIGURE 4.2 – The borders $(1\ 2\ 3\ 4\ 5)$ and $(1\ 3\ 2\ 4\ 5)$.

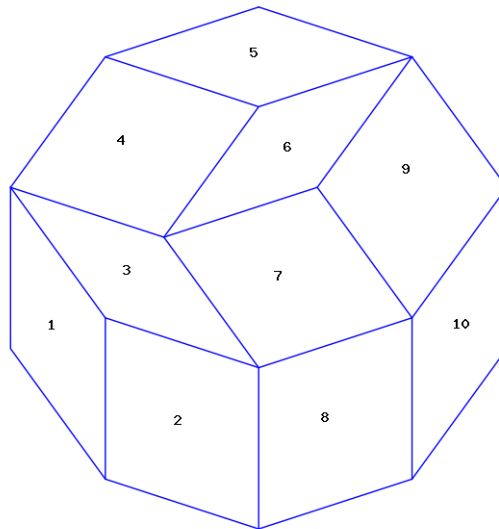


FIGURE 4.3 – The tiling corresponding to the word $(3, 4, 3, 2, 1, 2, 3, 4, 2, 3)$.

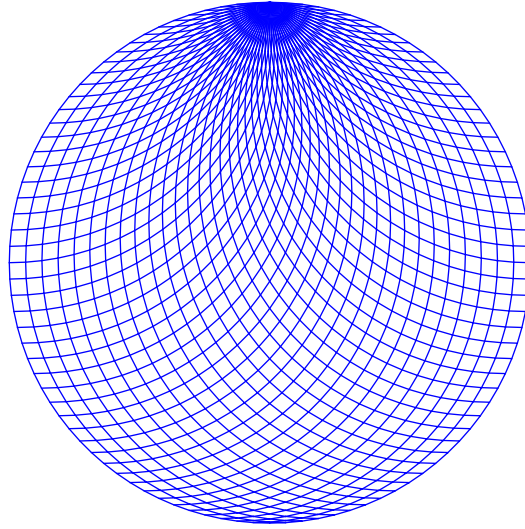


FIGURE 4.4 – A tiling of an hectogon (the regular 100-gon).

4.5 Exchange walk

We first need to recall a result from Coxeter groups theory, the exchange property :

Théorème 4.5.1 (Exchange property)

Let (W, S) be a Coxeter system,

$w = s_1 s_2 \dots s_k \in W$ a reduced word for w , and $s \in S$. If $l(sw) \leq l(w)$ then $sw = s_1 \dots \widehat{s}_i \dots s_k$ for $i \in \llbracket 1, k \rrbracket$ where \widehat{s}_i means that s_i has been deleted. Furthermore, the index i is unique.

In the theorem above, we have identified the reduced words $(1, 2, \dots, k)$, that is an ordered k -tuple, with the product $s_1 s_2 \dots s_k \in W$. We will use the same identification in the following paragraphs.

Démonstration. We need a little more tools to prove it, details can be found in [8]. \square

We now consider (W, S) a Coxeter system, with W a finite Coxeter group, it has a longest element (ie an element w such that $l(w)$ is maximal) denoted by w_0 (for example with $W = \mathfrak{S}_n$ we have $w_0 = (n \ n-1 \ \dots \ 1)$). Let $V(w_0)$ be the set of the reduced words for w_0 , $\alpha = s_1 s_2 \dots s_k \in V(w_0)$ a reduced word for w_0 , and $s \in S$. Since w_0 is maximal, we have $l(sw_0) \leq l(w_0)$ and, by the exchange property : $sw_0 = s_1 \dots \widehat{s}_i \dots s_k$, but we have also $s^2 = 1$ so we obtain a new reduced word for w_0 denoted by $e_s(\alpha) = s s_1 \dots \widehat{s}_i \dots s_k$. We can now introduce the exchange walk : consider a probability P on S

and the Markov chain which state space is $V(w_0)$ and transitions are given by moving from state α to state $e_s(\alpha)$ with probability $P(s)$. This Markov chain is called the exchange walk on (W, S) and is studied in [5].

The Markov chain we study here is slightly different because its state space is not $V(w_0)$, the set of the reduced words for w_0 , but $W(w_0)$, the commutation classes of the reduced words. To define this new exchange walk, we use the identification between the commutation class and its maximal element in the lexical order. Every commutation class $\beta \in W(w_0)$ is represented by a reduced word $\alpha_\beta \in V(w_0)$, we can apply the former exchange walk with $s \in S$ (the one defined on $V(w_0)$) to this reduced word α_β and obtain a new one : $e_s(\alpha_\beta) \in V(w_0)$. This reduced word is not necessarily maximal, if it is not, we can simply maximise it using the condition mentionned in proposition 4.3.1 : each time maximality condition is not respected, we commute the elements that do not respect it (you can see the corresponding code in appendix 4.7.3). At the end of this process, we obtain a reduced word whose elements respect the condition, hence we obtain a maximal element that we can identify with a commutation class $e'_s(\beta) \in W(w_0)$. Considering a probability P on S and the Markov chain which state space is $W(w_0)$ and which transitions are given by moving from stage β to state $e'_s(\beta)$ with probability $P(s)$, we have an exchange walk on the commutation classes.

Since we have a bijection between the commutation classes of the reduced words for w_0 , the longest element of \mathfrak{S}_n , and the rhombic tilings of the regular $2n$ -gon, we have also a random walk on the set of the tilings $T(w_0)$. For example take the figure 4.4, that represents the tiling corresponding to the reduced word $(1, 2, \dots, 50, 1, 2, \dots, 49, 1, \dots, 48, \dots, 1, 2, 3, 1, 2, 1)$, and that creates a very symmetric tiling, after 10000 random walks with uniform probability on S , we obtain the figure 4.5, which looks like figure 4.4 which would have been a little mixed up.

In the exchange walk we have presented, we do not know which index i will be deleted during the process, so we have to build an algorithm that is able to find that index i if we want to work on the exchange walk. We will now study that algorithm. We consider a border $b = (b(1) b(2) \dots b(n))$ of a $2n$ -gon, viewed like in part 4.4, as a permutation of \mathfrak{S}_n . Let $j \in \llbracket 1, n-1 \rrbracket$, we can build a rhombus with the sizes $b(j)$ and $b(j+1)$ if and only if the angle (at the right) formed by these sizes is lower than 180 degrees. Since the labels on the sizes correspond to the sizes of a regular $2n$ -gon, counter-clockwise from the uppermost vertex, this condition on the angle is equivalent to $b(j) < b(j+1)$. Finally, we can build a rhombus in place j (ie with the sizes $b(j)$ and $b(j+1)$ of the border) if and only if j is not a descent for b . Take the example of figure 4.2, and the border $(1 3 2 4 5)$: there is a descent in position 2 ($3 > 2$) so it is impossible to apply τ_2 to the border because we

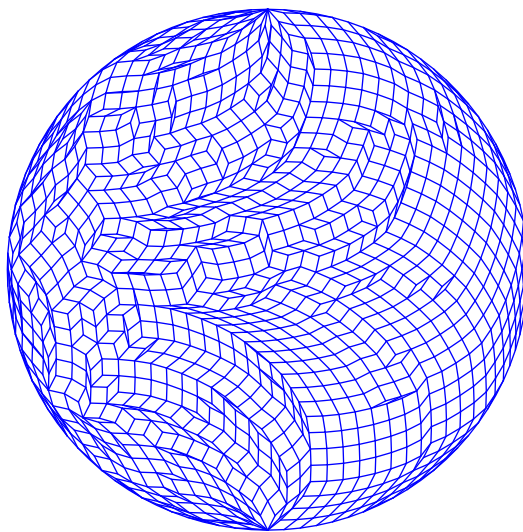


FIGURE 4.5 – The precedent tiling of the heptagon, after 10000 walks.

cannot create a tile with the sizes 3 and 2 of the border. The advantage of the characterisation with the descents is that it is easy to implement, since the operations to know the descents are just comparisons. We consider now a reduced word $\alpha = (\alpha_1, \dots, \alpha_k) \in W(w_0)$ where w_0 is the longest element of \mathfrak{S}_n , and $s \in \llbracket 1, n-1 \rrbracket$, we concatenate s and α to create a new word $w = (s, \alpha_1, \dots, \alpha_k)$. We would like to know which α_i we should delete to obtain a reduced word from w , so we apply the same steps as those of the bijection described in part 4.4, except that we do not draw anything, we just check that we could draw a new rhombus at each step with the characterisation using the descents. Then we just delete α_j when applying τ_{α_j} is impossible, and we have a new reduced word $(s, \alpha_1, \dots, \alpha_{j-1}, \alpha_{j+1}, \dots, \alpha_k)$ (we maximise it if needed). This algorithm is presented in appendix 4.7.5.

4.6 Directed graph and transition matrix

Now that we have a way to implement the exchange walk, we can build new tools based on this implementation in order to study this walk. Let w_0 be the longest element of \mathfrak{S}_n and $W(w_0)$ its set of commutation classes of its reduced words. We will study the directed graph G associated with the exchange walk on $W(w_0)$, defined as follow. The set of vertices of G are the elements of $W(w_0)$, let $\beta, \beta' \in W(w_0)$, there exists an edge from β to β' if and only if there exists $s \in S$ such that $\beta' = e'_s(\beta)$. Here, since $S = \{\tau_1, \dots, \tau_{n-1}\}$, the elementary transpositions, we label the edge from β

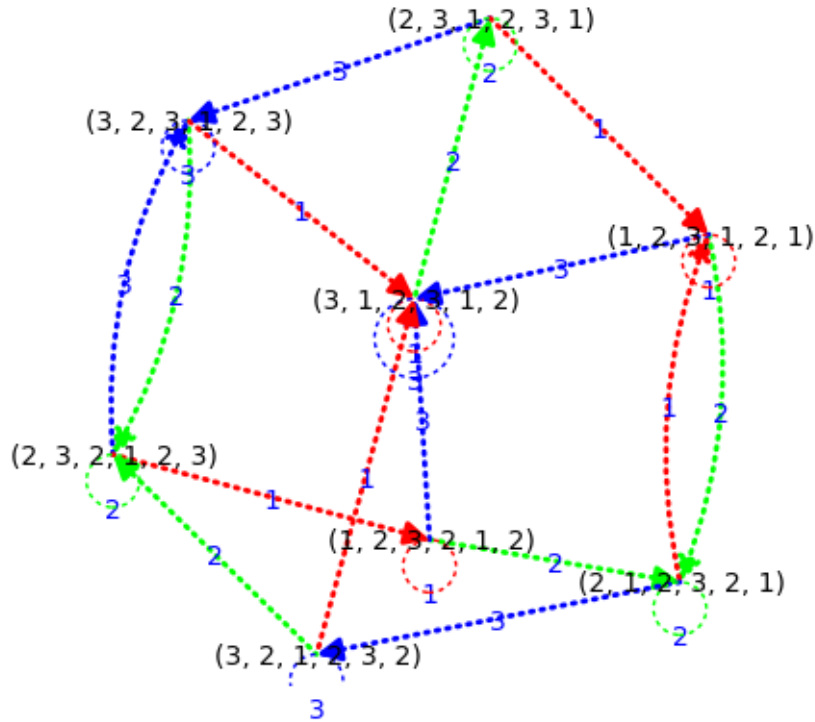


FIGURE 4.6 – The directed graph associated to the exchange walk ($n = 4$).

to $\beta' = e'_s(\beta)$ with the index j such that $s = \tau_j$. To implement a program that creates the graph, we apply the exchange walk with $\beta \in W(w_0)$ and the j -th element of S and we look at the result β' , then we just create an edge from β to β' labeled j (see appendix 4.7.7 for the code). Figure 4.6 is an example in the case of $w_0 \in \mathfrak{S}_4$, where the graph has 8 vertices and the edges are labeled from 1 to 3. We conjecture that G is strongly connected (ie for any i, k there exists a path from i to k).

The last object, the associated graph of the exchange walk, as nothing to do with the probability P on S , it just shows the different ways among $W(w_0)$ (certain walks could have a probability equal to 0). But we will now study the transition matrix of the exchange walk, which is directly linked to P and introduced in definition 4.2.4.

Here is an example of the transition matrix of the exchange walk on the commutation classes of the longest element of \mathfrak{S}_4 with uniform probability

on the set $\{1, 2, 3\}$.

$$\begin{pmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \end{pmatrix}$$

To build the transition matrix in the case of \mathfrak{S}_n and probability P , we use the exchange walk program again. We create the list of all the commutation classes, and for each one of them, we apply the exchange walk with every element of $S = \llbracket 1, n-1 \rrbracket$. If the i -th element of the list is sent to the j -th element, with $s \in S$, we just add $P(s)$ to the box ij (at the beginning there are only zeros). The corresponding program is presented in appendix 4.7.6. As it has been done in [5] in the case of the exchange walk on the reduced words, we study the properties of the transition matrix in the case of the commutation classes. For the first values of n , we use the already implemented function about eigenvalues in Sage to obtain the eigenvalues and their multiplicity. Thanks to these results, we conjecture that the eigenvalues are the

$$\gamma_J = \sum_{j \in J} P(j)$$

where $J \subseteq S$ and $|J| \neq |S| - 1$. And that the multiplicity of the eigenvalue γ_J is given by

$$\sum_{K \supseteq J} (-1)^{|K|-|J|} |W(w_J w_0)|$$

where w_0 is the longest element of \mathfrak{S}_n , w_J is the longest element of the subgroup of \mathfrak{S}_n generated by $\{\tau_j \mid j \in J\}$, and $W(w_J w_0)$ is the set of the commutation classes of the reduced words of $w_J w_0$.

4.7 Appendix : the code of the programs

Sometimes, the code of a program is more efficient to describe it than paraphrases, that is why we write all the programs mentioned in the paper below. All this programs are coded with Sage, a mathematics software using formal computation and written in the langage python. That is why the codes below look a lot like python 2, the small differences you could encounter are due to Sage's own langage.

4.7.1 Ismaxi

This program tests if the element “elem”, which is a list, is lexicographically maximal.

```
1 def ismaxi(elem):
2     n=len(elem)
3     for k in [0..n-2]:
4         if (elem[k+1]-elem[k]>=2):
5             return false
6     return true
```

4.7.2 Sorting networks

This program, adapted from another program that was already implemented in Sage, creates the list of the reduced words for the permutation “L” that are maximal, all these elements can be interpreted as the commutation classes.

```
1 def sorting_networks(L,k=0):
2     descents = L.descents()
3     if descents == []:
4         return [[]]
5     else:
6         return [ r + [i] for i in descents if (k-i<2) for r in
                sorting_networks(L.apply_simple_reflection(i),i)]
```

4.7.3 Maximise

This program permits to transform any word into the maximal word (with respect to the lexical order) of the same commutation class.

```
1 def maximise(mot):
2     while (ismaxi(mot)==false):
3         for k in [0..len(mot)-2]:
4             if (mot[k+1]-mot[k]>=2):
5                 mot[k],mot[k+1]=mot[k+1],mot[k]
6     return mot
```

4.7.4 Tiling

The program below creates the $2n$ -gon corresponding to the reduced word “mot” following the steps presented in the section 4.4, except that we don’t superpose the drawings representing the borders, we just add the new elements one after the others to be more efficient.

```
1 def tiling(mot,n=0):
2     if (n==0):
3         n=max(mot)+1
```

```

4     vectors=[]
5     order=[1..n] #the vectors will be drawn in this order
6     s=I
7     vertices=[[0,1]]
8     for k in [0..n-1]:
9         vectors.append((exp((2*I*pi*(k+1))/(2*n)+I*pi/2)-exp((2*I*pi*k
10            )/(2*n)+I*pi/2)) #creation of the vectors
11     for l in [0..n-1]:
12         s=s+vectors[l]
13         vertices.append([s.real(),s.imag()]) #vertices
14     figure=line(vertices) #the leftmost part of the 2n-gon
15     for m in [0..len(mot)-1]:
16         c=mot[m]
17         order[c-1],order[c]=order[c],order[c-1]
18         depart=I+sum([vectors[x-1] for x in order[0:(c-1)]])
19         milieu=depart+vectors[order[c]-1]
20         fin=milieu+vectors[order[c]-1]
21         figure=figure+line([[real(depart),imag(depart)],[real(milieu)
22            ,imag(milieu)],[real(fin),imag(fin)]])
23     return figure.show(aspect_ratio=1,axes=false)

```

4.7.5 Walk

This program gives the result of the exchange walk of the reduced word “mot” (a list) with the generator τ_k .

```

1     def walk(mot,k,n=0):
2         if (n==0):
3             n=max(mot)+1
4             L=[1..n]
5             word=[k]+mot
6             for n in [0..len(word)-1]:
7                 if (L[word[n]]<L[word[n]-1]):
8                     word[n:n+1]=[]
9                     word=maximise(word)
10                return word
11            L=switch(word[n]-1,word[n],L)

```

4.7.6 Transition matrix

This program builds the transition matrix corresponding to the exchange walk on the commutation classes of the reduced word of the longest element of \mathfrak{S}_p with probability P on $S = \{1, \dots, p-1\}$, the second element “proba” is the list $[P(1), \dots, P(p-1)]$.

```

1     def transition_matrix(p,proba=[]):
2         if (p<=1):
3             return "p must be >1"
4         if (proba==[]):

```

```

5     proba=[1/(p-1)]*(p-1)
6     L=sorting_networks(SymmetricGroup(p).long_element())
7     n=len(L)
8     M=[]
9     for k in [0..n-1]:
10        M.append([0]*n)
11    for j in [0..n-1]:
12        for q in [1..p-1]:
13            M[j][L.index(walk(L[j],q))]=M[j][L.index(walk(L[j],q))]+
                proba[q-1]
14    return Matrix(M)

```

4.7.7 Walk graph

This programs build the directed graph corresponding to the exchange walk on the commutation classes of the reduced word of the longest element of \mathfrak{S}_p

```

1 def walk_graph(p):
2     if (p<=1):
3         return "p must be >1"
4     L=sorting_networks(SymmetricGroup(p).long_element())
5     n=len(L)
6     edges=[]
7     for j in [0..n-1]:
8         for q in [1..p-1]:
9             edges.append([tuple(L[j]),tuple(walk(L[j],q)),q])
10    G=DiGraph(loops=true,multiedges=true)
11    G.add_edges(edges)
12    return G

```

Chapitre 5

Travail Encadré de Recherche de M1

5.1 Préambule

Ce mémoire a été écrit dans le cadre des Travaux Encadrés de Recherche (TER) au cours du M1 Mathématiques Fondamentales et Appliquées (MFA) de l'université Paris-Sud. Nous y démontrons un théorème dû à Pierre Gabriel datant de 1972 ; cependant, la preuve utilisée ici n'est pas la preuve originale de Gabriel, nous suivons la preuve de Bernstein, Gel'fand et Ponomarev issue de [7]. On introduit tout d'abord des notions fondamentales à la compréhension de cette preuve, comme les représentations de carquois, la théorie des catégories, et les systèmes de racines. Puis on met en place des outils, tels que foncteurs de Coxeter. Ces outils nous permettront, finalement, de prouver le théorème de Gabriel.

Quelques notations

Rassemblons ici quelques notations que nous utiliserons sans rappel dans la suite du document. Nous noterons fg plutôt que $f \circ g$, la composition entre les fonctions f et g ; fg ne sera jamais un produit point par point. Si f est une application, on note $\text{Im } f$ son image et $\text{Ker } f$ son noyau. Si f est une application définie sur un ensemble X et que Y est un sous-ensemble de X , on note $f|_Y$ la restriction de f à Y . On utilise la flèche \hookrightarrow pour désigner une injection, et la flèche \twoheadrightarrow pour désigner une surjection. Nous utiliserons l'abus de notation $x = 0$ où x peut représenter des objets de natures diverses, 0 désignera alors l'élément nul correspondant à la nature de l'objet x . Lorsqu'il n'y a pas d'ambiguïté, ou par souci de simplification, nous noterons V plutôt que (V, f) pour une représentation de carquois. Les espaces vectoriels utilisés sont tous des \mathbb{K} -espaces vectoriels, où \mathbb{K} est un corps fixé. Si V et V' sont deux espaces vectoriels, on note $V \oplus V'$ leur somme directe. Si f et g sont

deux applications linéaires, on note $f \oplus g$ leur somme directe. Si A et B sont deux objets, on note A/B leur quotient, la définition de quotient varie alors en fonction de la nature de A et B . Si X est un ensemble, on note Id_X l'application identité de X dans lui-même. On note $\llbracket 1, n \rrbracket$ l'ensemble des entiers naturels compris entre 1 et n .

5.2 Notions préliminaires


Afin de comprendre l'énoncé, puis la preuve, du théorème de Gabriel, nous avons besoin de notions. Dans toute cette partie, nous introduirons les concepts nécessaires à la bonne compréhension du lecteur.

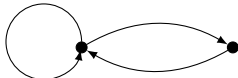
5.2.1 Carquois

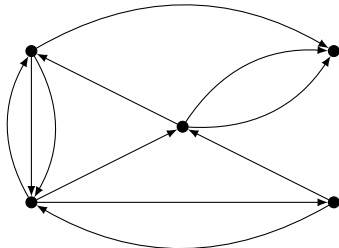
Le théorème de Gabriel donne de très beaux résultats sur les représentations de carquois. Cette notion est donc primordiale pour toute la suite de ce document.

Définition 5.2.1 (Carquois). Un *carquois* (Γ, Λ) est constitué d'un graphe quelconque non orienté Γ et d'une orientation Λ . On note Γ_0 l'ensemble des sommets de Γ et Γ_1 l'ensemble des arêtes non orientées de Γ . L'orientation Λ est un couple de fonctions (s, t) nommées respectivement source et destination définis de Γ_1 dans Γ_0 .

On considérera seulement des carquois connexes dans les exemples ci-dessous.

Exemple 5.2.2. 

Exemple 5.2.3. 

Exemple 5.2.4. 

5.2.2 Représentations de carquois

Maintenant que nous connaissons la définition d'un carquois, et que nous pouvons nous représenter l'objet, nous sommes capables d'aborder le coeur du sujet : les représentations de carquois. À l'issue de cette partie, nous

serons même capables d'énoncer, et comprendre, une partie du théorème de Gabriel.

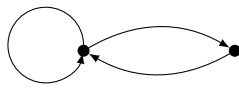
Définition 5.2.5 (Représentation de carquois). Une *représentation de carquois* d'un carquois (Γ, Λ) est un couple (V, f) où V est une liste indexée sur Γ_0 d'espaces vectoriels de dimension finie et f est une liste indexée sur Γ_1 de morphismes d'espace vectoriel tels que pour tout $l \in \Gamma_1$, f_l est définie de $V_{s(l)}$ dans $V_{t(l)}$.

Exemple 5.2.6 (Une représentation). Soit $\bullet \longrightarrow \bullet$ le carquois dessiné dans l'exemple 5.2.2. Si l'on numérote les sommets de 1 à 2 de la gauche vers la droite, on pose $V_1 = \mathbb{R}^2$ et $V_2 = \mathbb{R}^3$, que l'on voit comme des \mathbb{R} -espaces vectoriels. Posons une application linéaire f de V_1 dans V_2 , par exemple $f : (x, y) \mapsto (x, y, x + y)$. On obtient alors la représentation suivante.

$$\mathbb{R}^2 \xrightarrow{f} \mathbb{R}^3$$

Si l'on choisi des bases dans les espaces vectoriels V_1 et V_2 , par exemple les bases canoniques, on peut réécrire cette représentation de la manière suivante.

$$\mathbb{R}^2 \xrightarrow{\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}} \mathbb{R}^3$$

Exemple 5.2.7 (Une deuxième représentation). Soit  le carquois de l'exemple 5.2.3. Si l'on numérote les sommets de 1 à 2 de la gauche vers la droite, on pose $V_1 = \mathbb{R}^2$ et $V_2 = \mathbb{R}$, vus comme des \mathbb{R} -espaces vectoriels. On choisi pour bases de V_1 et V_2 les bases canoniques, afin d'exprimer les applications linéaires à l'aide de matrices. La figure ci-dessous est alors une représentation du carquois dessiné à l'exemple 5.2.3.

$$\begin{pmatrix} -1 & 1 \\ -3 & -1 \end{pmatrix} \begin{array}{c} \text{self-loop} \\ \mathbb{R}^2 \end{array} \begin{array}{c} \xrightarrow{\begin{pmatrix} 1 & 2 \end{pmatrix}} \\ \mathbb{R} \\ \xleftarrow{\begin{pmatrix} 0 \\ 1 \end{pmatrix}} \end{array}$$

Définition 5.2.8 (Représentation nulle). Soit (Γ, Λ) un carquois, la *représentation nulle* de (Γ, Λ) est la représentation (V, f) où $\forall \gamma \in \Gamma_0$, $V_\gamma = \{0\}$ et où $\forall l \in \Gamma_1$, f_l est l'application linéaire nulle. On note $(V, f) = 0$.

Définition 5.2.9 (Morphisme de représentations). Soit (Γ, Λ) un carquois et $(V, f), (W, g)$ deux représentations de ce carquois. On dit que $\varphi = (\varphi_\gamma)_{\gamma \in \Gamma_0}$ est un *morphisme de représentations* de (V, f) dans (W, g) si, pour tout $\gamma \in \Gamma_0$, φ_γ est une application linéaire de V_γ dans W_γ , et pour tout $l \in \Gamma_1$, le diagramme suivant commute :

$$\begin{array}{ccc} V_{s(l)} & \xrightarrow{f_l} & V_{t(l)} \\ \varphi_{s(l)} \downarrow & & \downarrow \varphi_{t(l)} \\ W_{s(l)} & \xrightarrow{g_l} & W_{t(l)} \end{array}$$

c'est-à-dire si $\forall l \in \Gamma_1, \varphi_{t(l)} f_l = g_l \varphi_{s(l)}$.

Exemple 5.2.10 (Morphisme de représentations). Soit $(\Gamma, \Lambda) = \bullet \longrightarrow \bullet$ le carquois dessiné dans l'exemple 5.2.2, et soient (V, f) et (W, g) deux représentations de (Γ, Λ) , on considère les ensembles aux sommets comme des \mathbb{R} -espaces vectoriels, munis de leurs bases canoniques respectives. Ainsi on peut écrire ces représentations à l'aide de matrices.

$$\begin{array}{ccc} \mathbb{R} & \xrightarrow{1} & \mathbb{R} \\ (V, f) & & \end{array} \quad \begin{array}{ccc} & & \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \\ \mathbb{R}^2 & \xrightarrow{\quad} & \mathbb{R}^3 \\ (W, g) & & \end{array}$$

En appelant 1 le sommet de gauche et 2 celui de droite, on pose φ définie par $\forall x \in \mathbb{R}, \varphi_1(x) = (x, x), \varphi_2(x) = (x, x, 2x)$. Ainsi, on exprime φ_1 et φ_2 sous forme matricielle, et comme $\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix}$, le diagramme suivant commute :

$$\begin{array}{ccc} \mathbb{R} & \xrightarrow{1} & \mathbb{R} \\ \begin{pmatrix} 1 \\ 1 \end{pmatrix} \downarrow & & \downarrow \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \\ \mathbb{R}^2 & \xrightarrow{\quad} & \mathbb{R}^3 \\ & & \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \end{array},$$

et ainsi φ est un morphisme de représentations de (V, f) dans (W, g) .

Définition 5.2.11 (Isomorphisme de représentations). Soit (Γ, Λ) un carquois et $(V, f), (W, g)$ deux représentations de ce carquois. Soit φ un morphisme de représentations de (V, f) dans (W, g) . On dit que φ est un isomorphisme de représentations entre (V, f) et (W, g) si $\forall \gamma \in \Gamma_0, \varphi_\gamma$ est inversible. Si on note $\varphi^{-1} = (\varphi_\gamma^{-1})_{\gamma \in \Gamma_0}$, on a que φ^{-1} est un morphisme de (W, g) dans (V, f) . En effet, φ étant un morphisme de (V, f) dans (W, g) on a $\forall l \in \Gamma_1, \varphi_{t(l)}f_l = g_l\varphi_{s(l)}$, cette égalité s'écrit également $\forall l \in \Gamma_1, \varphi_{t(l)}^{-1}g_l = f_l\varphi_{s(l)}^{-1}$, donc on a bien que φ^{-1} est un morphisme de (W, g) dans (V, f) .

Exemple 5.2.12 (Condition nécessaire et suffisante d'isomorphisme entre deux représentations simples). On considère les deux représentations suivantes, où l'on a les espaces vectoriels \mathbb{R} à chaque sommet, et où $\lambda, \mu \in \mathbb{R}$ sont des constantes.



Supposons que les deux représentations ci-dessus soient isomorphes. Les applications linéaires de \mathbb{R} dans \mathbb{R} peuvent être représentées par de simples constantes. Ainsi il existe a et b deux constantes non nulles telles que :

$$\begin{cases} 1 \times b & = & a \times 1 \\ b\lambda & = & \mu a \end{cases}$$

d'où :

$$\begin{cases} a & = & b \\ \lambda & = & \mu. \end{cases}$$

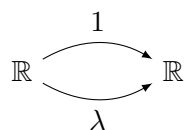
Ainsi les deux représentations sont isomorphes seulement si $\lambda = \mu$. Si l'on suppose maintenant que $\lambda = \mu$, les représentations sont égales donc a fortiori isomorphes. On a donc prouvé que les deux représentations sont isomorphes si et seulement si $\lambda = \mu$.

Définition 5.2.13 (Somme directe de deux représentations de carquois). Soit (Γ, Λ) un carquois, (V, f) et (V', f') deux représentations de ce carquois. La *somme directe* de (V, f) et (V', f') est la représentation (W, g) définie par $\forall \gamma \in \Gamma_0, W_\gamma = V_\gamma \oplus V'_\gamma$ et $\forall l \in \Gamma_1, g_l = f_l \oplus f'_l$. On note $(W, g) = (V, f) \oplus (V', f')$.

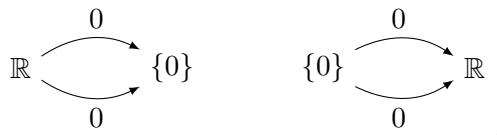
Définition 5.2.14 (Représentation indécomposable). Soit (Γ, Λ) un carquois et (V, f) une représentation non nulle de ce carquois. On dit que (V, f) est une *représentation indécomposable* si on ne peut pas l'écrire comme somme directe de deux représentations non nulles. De manière équivalente, on dit que (V, f) est une représentation indécomposable si pour toutes représentations $(V^1, f^1), (V^2, f^2), (V, f) = (V^1, f^1) \oplus (V^2, f^2)$ implique $(V^1, f^1) = 0$ ou $(V^2, f^2) = 0$.

Exemple 5.2.15 (Représentations irréductibles associées aux sommets). Soit (Γ, Λ) un carquois. On appelle *représentations irréductibles associées aux sommets*, et on note L_γ , où $\gamma \in \Gamma_0$, les représentations telles que $\forall \delta \neq \gamma, (L_\gamma)_\delta = \{0\}$, $(L_\gamma)_\gamma = \mathbb{K}$ et $\forall l \in \Gamma_1, f_l = 0$. L_γ est alors une représentation indécomposable.


Exemple 5.2.16 (Infinité des classes d'isomorphismes de représentations indécomposables dans un carquois simple). Soit (V, f) la représentation donnée par le dessin suivant, où l'on a \mathbb{R} , considéré comme un \mathbb{R} -espace vectoriel, muni de sa base canonique, et où $\lambda \in \mathbb{R}$ est un réel quelconque. Nous allons prouver qu'elle est indécomposable.



Supposons que (V, f) s'écrit comme somme directe de deux représentations non nulles (V^1, f^1) et (V^2, f^2) . Si les espaces vectoriels aux sommets de (V^1, f^1) sont \mathbb{R} et \mathbb{R} , alors ceux aux sommets de (V^2, f^2) sont $\{0\}$ et $\{0\}$, et alors (V^2, f^2) est la représentation nulle. Ainsi (V^1, f^1) et (V^2, f^2) sont les deux représentations :



dont la somme directe n'est pas isomorphe à (V, f) . Ainsi (V, f) est une représentation indécomposable. En cumulant ce résultat avec celui de l'exemple

5.2.12, on obtient que le carquois  possède une infinité non dénombrable de représentations indécomposables, même à isomorphisme près. Le théorème de Gabriel (numéroté 5.5.2 dans ce document) permet de classer tous les carquois possédant un nombre fini, à isomorphisme près, de représentations indécomposables.

Définition 5.2.17 (Sous-représentations). Soit (Γ, Λ) un carquois, $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ une représentation de ce carquois, et $\mathcal{V} = (\mathcal{V}_\gamma)_{\gamma \in \Gamma_0}$ une famille d'espaces vectoriels sur les sommets de Γ telle que pour tout sommet $\gamma \in \Gamma_0$, \mathcal{V}_γ est un sous-espace vectoriel de V_γ et pour toute arête $l \in \Gamma_1$ on a $f_l(\mathcal{V}_{s(l)}) \subset \mathcal{V}_{t(l)}$. On appelle alors *sous-représentation* la représentation $((\mathcal{V}_\gamma)_{\gamma \in \Gamma_0}, (f_l|_{\mathcal{V}_{s(l)}})_{l \in \Gamma_1})$, qu'on notera plus simplement \mathcal{V} .

Définition 5.2.18 (Représentation quotient). Soit (Γ, Λ) un carquois, $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ une représentation de ce carquois, et \mathcal{V} une sous-représentation de

(V, f) . On définit alors la *représentation quotient* $(V/\mathcal{V}, \bar{f})$ par $\forall \gamma \in \Gamma_0$, $(V/\mathcal{V})_\gamma = V_\gamma/\mathcal{V}_\gamma$ et pour toute arête $l \in \Gamma_1$, \bar{f} est l'application définie par $\bar{f}(v + \mathcal{V}_{s(l)}) = f(v) + \mathcal{V}_{t(l)}$.

5.2.3 Théorie des catégories

Avant de traiter la preuve du théorème de Gabriel, nous introduisons le langage de la théorie des catégories, qui nous aidera au cours de la preuve. Les notations utilisées ici sont celles de [4], et de plus amples détails sur les catégories peuvent y être trouvés, nous ne donnons ici qu'une brève présentation des notions qui nous seront utiles par la suite.

Définition 5.2.19 (Catégorie). Une *catégorie* \mathcal{C} est définie par :

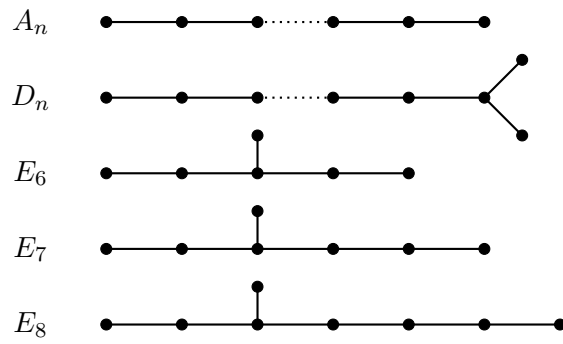
1. une classe \mathcal{C}_0 , appelée classe des objets de \mathcal{C} ;
2. pour chaque paire (X, Y) d'objets de \mathcal{C} , un ensemble noté $\text{Hom}(X, Y)$ dont les éléments sont appelés les morphismes de X vers Y , tel que si $(X, Y) \neq (X', Y')$ alors $\text{Hom}(X, Y) \cap \text{Hom}(X', Y') = \emptyset$;
3. pour chaque triplet X, Y, Z d'objets de \mathcal{C} , il existe \circ définie de $\text{Hom}(X, Y) \times \text{Hom}(Y, Z)$ dans $\text{Hom}(X, Z)$, notée $\circ(f, g) = g \circ f$, et satisfaisant :
 - (a) Pour $f \in \text{Hom}(U, V)$, $g \in \text{Hom}(V, W)$ et $h \in \text{Hom}(W, X)$: $(h \circ g) \circ f = h \circ (g \circ f)$;
 - (b) Pour $X \in \mathcal{C}_0$, il existe $1_X \in \text{Hom}(X, X)$ tel que pour tout $g \in \text{Hom}(W, X)$, $1_X \circ g = g$ et $h \in \text{Hom}(X, Y)$, $h \circ 1_X = h$.

Définition 5.2.20 (Foncteur). Soit \mathcal{C} et \mathcal{D} deux catégories, un *foncteur* F de \mathcal{C} dans \mathcal{D} associe à chaque objet X de \mathcal{C} , un objet $F(X)$ de \mathcal{D} et à chaque morphisme $f \in \text{Hom}_{\mathcal{C}}(X, Y)$, un morphisme $F(f) \in \text{Hom}_{\mathcal{D}}(F(X), F(Y))$ tel que :

1. Pour $f \in \text{Hom}_{\mathcal{C}}(X, Y)$ et $g \in \text{Hom}_{\mathcal{C}}(Y, Z)$, $F(f \circ g) = F(f) \circ F(g)$;
2. Pour X objet de \mathcal{C} , $F(1_X) = 1_{F(X)}$.

5.2.4 Diagrammes de Dynkin

Nous faisons ici un inventaire de graphes non orientés et sans boucles, qui joueront un rôle fondamental dans le théorème de Gabriel. L'indice de ces graphes correspond au nombre de sommets qu'ils contiennent.



On appelle ces graphes les diagrammes de Dynkin de type A_n , D_n , E_6 , E_7 et E_8 , ou encore diagrammes de Dynkin simplement lacés.

5.3 Foncteurs de réflexion et foncteurs de Coxeter

5.3.1 Foncteurs de réflexion

Soit (Γ, Λ) un carquois, on se place dans la catégorie $\mathcal{L}(\Gamma, \Lambda)$. Nous allons construire des foncteurs de réflexion, associés aux sommets $\gamma \in \Gamma_0$ de Γ pour lesquels la direction des flèches le contenant est identique. On note $\Gamma^\gamma = \{l \in \Gamma_1 : s(l) = \gamma \text{ ou } t(l) = \gamma\}$ l'ensemble des arêtes contenant γ .

Définition 5.3.1 (Puits et sources). On dit qu'un sommet $\beta \in \Gamma_0$ est un *puits*, ou qu'il est (+)-accessible, si $\forall l \in \Gamma^\beta$, $t(l) = \beta$ et $s(l) \neq \beta$, c'est-à-dire si toutes les arêtes contenant β arrivent en β et qu'il n'y a pas de boucles de β dans lui-même.

On dit de même qu'un sommet $\alpha \in \Gamma_0$ est une *source*, ou qu'il est (-)-accessible, si $\forall l \in \Gamma^\alpha$, $s(l) = \alpha$ et $t(l) \neq \alpha$, c'est-à-dire si toutes les arêtes contenant α partent de α et qu'il n'y a pas de boucles de α dans lui-même.

Si $\gamma \in \Gamma_0$ est un sommet, on note $\sigma_\gamma \Lambda$ l'orientation obtenue à partir de Λ en inversant le sens des flèches contenant γ .

Supposons que le sommet $\alpha \in \Gamma_0$ soit un puits avec l'orientation Λ . Nous allons construire une application qui à un objet (V, f) de $\mathcal{L}(\Gamma, \Lambda)$ associe un objet (W, g) de $\mathcal{L}(\Gamma, \sigma_\alpha \Lambda)$. De même si $\beta \in \Gamma_0$ est une source, nous allons construire une autre application qui à un objet de $\mathcal{L}(\Gamma, \Lambda)$ associe un objet de $\mathcal{L}(\Gamma, \sigma_\beta \Lambda)$.

Définition 5.3.2 (Foncteur de réflexion). Soit β un puits, on appelle *foncteur de réflexion* l'application F_β^+ définie comme suit. En notant $F_\beta^+(V, f) = (W, g)$, on pose, $\forall \gamma \neq \beta$, $W_\gamma = V_\gamma$, et $\forall l \notin \Gamma^\beta$, $g_l = f_l$. Ainsi on laisse inchangés tous les sommets différents de β et toutes les arêtes ne faisant pas intervenir β . On considère les arêtes l_1, \dots, l_n qui vont en β et on note $h : \bigoplus_{j=1}^n V_{s(l_j)} \rightarrow V_\beta$ l'application qui à un vecteur (v_1, \dots, v_n) associe la somme $f_{l_1}(v_1) + \dots + f_{l_n}(v_n)$. On pose alors $W_\beta = \text{Ker}(h)$. On définit maintenant les applications g_l pour $l \in \Gamma^\beta$. Soit $i \in \llbracket 1, n \rrbracket$, alors g_{l_i} est la composition de l'injection de $\text{Ker}(h)$ dans $\bigoplus_{j=1}^n V_{s(l_j)}$ et de la projection de $\bigoplus_{j=1}^n V_{s(l_j)}$ dans $V_{s(l_i)}$.

Soit α une source, on a un deuxième *foncteur de réflexion* qu'on notera F_α^- et qu'on définit comme suit. On note $F_\alpha^-(V, f) = (W, g)$, et on pose, $\forall \gamma \neq \alpha$, $W_\gamma = V_\gamma$, et $\forall l \notin \Gamma_\alpha$, $f_l = g_l$. En considérant les arêtes l_1, \dots, l_n qui partent de α , on note $\tilde{h} : V_\alpha \rightarrow \bigoplus_{j=1}^n V_{t(l_j)}$ et qui à un vecteur $v \in V_\alpha$ associe le vecteur $(f_1(v), \dots, f_n(v))$. On pose alors $W_\alpha = \bigoplus_{j=1}^n V_{t(l_j)} / \text{Im}(\tilde{h})$. On pose pour $i \in \llbracket 1, n \rrbracket$, alors g_{l_i} comme la composition de l'inclusion naturelle de V_{l_i}

dans $\bigoplus_{j=1}^n V_{t(l_j)}$ et de la projection dans $\bigoplus_{j=1}^n V_{t(l_j)} / \text{Im}(\tilde{h})$.

Exemple 5.3.3 (Application des foncteurs sur L_γ). Soit (Γ, Λ) le carquois définie à l'exemple 5.2.2 par $\bullet \longrightarrow \bullet$, on note 1 et 2 les sommets de gauche à droite et on considère la représentation L_2 .

$$\{0\} \xrightarrow{0} \mathbb{R}$$

On va donc appliquer le foncteur F_2^+ à cette représentation. Les sommets autres que 2 ne changent pas, le sens des flèches est inversé, et le sommet 2 est $\text{Ker } f$ où $f : \{0\} \rightarrow \mathbb{R}$ est l'application nulle. Ainsi on a $\text{Ker } f = \{0\}$, et l'application de l'arête $2 \rightarrow 1$ est la composée de l'injection de $\{0\}$ dans $\{0\}$ et de la projection sur la première coordonnée (mais il n'y en a qu'une), donc c'est l'application nulle. Ainsi $F_2^+(L_2) = 0$, c'est la représentation nulle. On utilisera d'ailleurs ce résultat dans le théorème 5.3.8.

Proposition 5.3.4

L'application $F_\beta^+ : \mathcal{L}(\Gamma, \Lambda) \rightarrow \mathcal{L}(\Gamma, \sigma_\beta \Lambda)$ est un foncteur. De même l'application $F_\alpha^- : \mathcal{L}(\Gamma, \Lambda) \rightarrow \mathcal{L}(\Gamma, \sigma_\alpha \Lambda)$ est un foncteur.

Démonstration. On se place dans le même cadre que lors des définitions, ainsi on a $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ une représentation du carquois (Γ, Λ) . On note toujours $F_\beta^+(V, f) = (W, g) \in \mathcal{L}(\Gamma, \sigma_\beta \Lambda)$ la représentation image de (V, f) par l'application F_β^+ . On a déjà la définition de F_β^+ sur les objets des catégories $\mathcal{L}(\Gamma, \Lambda)$ et $\mathcal{L}(\Gamma, \sigma_\beta \Lambda)$, et pour tout $(\gamma, l) \in \Gamma_0 \times \Gamma_1$, on a bien que W_γ est un espace vectoriel, et que g_l est une application linéaire. Il nous reste donc à définir F_β^+ sur les morphismes de représentations pour obtenir son caractère fonctoriel. Ainsi, soient (V, f) et (V', f') deux représentations de Γ , on note encore $l_1, \dots, l_n \in \Gamma^\beta$ les arêtes en direction du puits β , et soit $\varphi = (\varphi_\gamma)_{\gamma \in \Gamma_0}$ un morphisme de représentations entre (V, f) et (V', f') . On note (W, g) et (W', g') les représentations images par F_β^+ de (V, f) et (V', f') . On note également $\psi = (\psi_\gamma)_{\gamma \in \Gamma_0}$ le morphisme image de φ par F_β^+ , qu'il nous faut encore définir. On pose donc :

$$\psi_\beta(v_1, \dots, v_n) = (\varphi_{s(l_1)}(v_1), \dots, \varphi_{s(l_n)}(v_n))$$

$$\forall \gamma \neq \beta, \psi_\gamma = \varphi_\gamma.$$

Assurons nous tout d'abord du caractère bien défini de ψ . Pour tout $\gamma \neq \beta$, on a $V_\gamma = W_\gamma$, $V'_\gamma = W'_\gamma$ et $\psi_\gamma = \varphi_\gamma$, le morphisme ψ est donc bien défini pour tout sommet $\gamma \neq \beta$. Il nous reste à vérifier que ψ_β est bien défini. Soit

$(v_1, \dots, v_n) \in \text{Ker}(h)$, il nous faut voir que $\psi_\beta(v_1, \dots, v_n) \in \text{Ker}(k)$, où

$$\begin{aligned} h : \bigoplus_{j=1}^n V_{s(l_j)} &\rightarrow V_\beta \\ (v_1, \dots, v_n) &\mapsto \sum_{i=1}^n f_{l_i}(v_i) \end{aligned}$$

$$\begin{aligned} k : \bigoplus_{j=1}^n V'_{s(l_j)} &\rightarrow V'_\beta \\ (v'_1, \dots, v'_n) &\mapsto \sum_{i=1}^n f'_{l_i}(v'_i). \end{aligned}$$

Ainsi :

$$\begin{aligned} k(\psi_\beta(v_1, \dots, v_n)) &= \sum_{i=1}^n f'_{l_i}(\varphi_{s(l_i)}(v_i)) \\ &= \sum_{i=1}^n \varphi_\beta(f_{l_i}(v_i)) \\ &= \varphi_\beta\left(\sum_{i=1}^n f_{l_i}(v_i)\right) \\ &= \varphi_\beta(0) \\ &= 0 \end{aligned}$$

donc ψ_β est bien définie. Voyons désormais que ψ est bien un morphisme de représentations entre (W, g) et (W', g') . Comme pour tout $\gamma \neq \beta$, on a $\psi_\gamma = \varphi_\gamma$, et pour tout $l \notin \Gamma^\beta$, on a $g_l = f_l$ et $g'_l = f'_l$, il vient facilement que $g'_l \psi_{s(l)} = \psi_{t(l)} g_l$ dès lors que $l \notin \Gamma^\beta$. Regardons maintenant le cas où $l \in \Gamma^\beta$, il existe alors $i \in \llbracket 1, n \rrbracket$ tel que $l = l_i$, et on a $t(l_i) = \beta$. Soit $(v_1, \dots, v_n) \in W_\beta$, il vient, d'une part :

$$\begin{aligned} g'_i \psi_\beta(v_1, \dots, v_n) &= g'_{l_i}(\varphi_{s(l_1)}(v_1), \dots, \varphi_{s(l_n)}(v_n)) \\ &= \varphi_{s(l_i)}(v_i) \end{aligned}$$

et d'autre part :

$$\begin{aligned} \psi_{s(l_i)} g_{l_i}(v_1, \dots, v_n) &= \psi_{s(l_i)}(v_i) \\ &= \varphi_{s(l_i)}(v_i). \end{aligned}$$

Ainsi, on a bien que $\forall l \in \Gamma_1$, $g'_l \psi_{t(l)} = \psi_{s(l)} g_l$, ce qui prouve que ψ est un morphisme de représentations de (W, g) vers (W', g') . Il nous reste donc à voir que F_β^+ respecte la composition des morphismes et qu'il envoie l'identité sur l'identité. À cet effet, posons $\text{Id} = (\text{Id}_{V_\gamma})_{\gamma \in \Gamma_0}$ le morphisme identité de (V, f) dans (V, f) , c'est-à-dire que pour tout sommet $\gamma \in \Gamma_0$, Id_{V_γ} est l'application identité de V_γ dans lui-même. On a :

$$\forall \gamma \neq \beta, F_\beta^+(\text{Id})_\gamma = \text{Id}_{V_\gamma}$$

$$F_\beta^+(\text{Id})_\beta = (\text{Id}_{V_{s(l_1)}}, \dots, \text{Id}_{V_{s(l_n)}}) = \text{Id}_{\text{Ker}(h)}.$$

Le morphisme identité de (V, f) est donc bien envoyé sur le morphisme identité de $F_\beta^+(V, f)$. Posons désormais $(V, f), (V', f')$ et (V'', f'') trois représentations de (Γ, Λ) , φ (respectivement φ') un morphisme de représentation de (V, f) dans (V', f') (respectivement de (V', f') dans (V'', f'')), ainsi que $(W, g), (W', g'), (W'', g''), \psi$ et ψ' leurs images par F_β^+ . Il vient :

$$\forall \gamma \neq \beta, F_\beta^+(\varphi')_\gamma F_\beta^+(\varphi)_\gamma = \varphi'_\gamma \varphi_\gamma = F_\beta^+(\varphi' \varphi)_\gamma$$

$$\begin{aligned} \forall (v_1, \dots, v_n) \in \text{Ker}(h), \\ F_\beta^+(\varphi')_\beta F_\beta^+(\varphi)_\beta (v_1, \dots, v_n) &= F_\beta^+(\varphi')_\beta (\varphi_{s(l_1)}(v_1), \dots, \varphi_{s(l_n)}(v_n)) \\ &= (\varphi'_{s(l_1)}(\varphi_{s(l_1)}(v_1)), \dots, \varphi'_{s(l_n)}(\varphi_{s(l_n)}(v_n))) \\ &= F_\beta^+(\varphi' \varphi)_\beta (v_1, \dots, v_n) \end{aligned}$$

Ainsi la composition de deux morphismes est bien respectée par F_β^+ , et cela conclut la preuve pour cette application.

On a déjà défini l'action de F_α^- sur les représentations de la catégorie $\mathcal{L}(\Gamma, \Lambda)$ à valeurs dans la catégorie $\mathcal{L}(\Gamma, \sigma_\alpha \Lambda)$. Il s'agit maintenant de définir l'action de F_α^- sur les morphismes de représentations. Soit (V, f) et (V', f') deux représentations et $\varphi = (\varphi_\gamma)_{\gamma \in \Gamma_0}$ un morphisme entre ces deux représentations dans $\mathcal{L}(\Gamma, \Lambda)$. Posons pour tout $\gamma \neq \alpha, F_\alpha^-(\varphi)_\gamma = \varphi_\gamma$, car F_α^- laisse inchangés les espaces vectoriels associés aux sommets différents de α . On définit alors l'action de $F_\alpha^-(\varphi)_\alpha$ sur $\oplus_{l \in \Gamma^\alpha} V_{t(l)} / \text{Im } \tilde{h}$ dans $\oplus_{l \in \Gamma^\alpha} V'_{t(l)} / \text{Im } \tilde{h}'$ avec \tilde{h}' la fonction de V_α dans $\oplus_{l \in \Gamma^\alpha} V'_{t(l)}$ qui à v associe $(f'_1(v), \dots, f'_n(v))$ par :

$$\forall (y_1, \dots, y_n) + \text{Im } \tilde{h} \in \oplus_{i=1}^n V_{t(l_i)} / \text{Im } \tilde{h}, F_\alpha^-(\varphi)_\alpha ((y_1, \dots, y_n) + \text{Im } \tilde{h}) = (\varphi_{t(l_1)}(y_1), \dots, \varphi_{t(l_n)}(y_n)) + \text{Im } \tilde{h}'$$

Vérifions alors que cette définition a un sens, soit (y_1, \dots, y_n) et (z_1, \dots, z_n) deux éléments de $\oplus_{i=1}^n V_{t(l_i)}$ ayant la même classe dans $\oplus_{i=1}^n V_{t(l_i)} / \text{Im } \tilde{h}$ c'est-à-dire qu'il existe $v \in V_\alpha$ tel que $(y_1, \dots, y_n) = (z_1, \dots, z_n) + (f_1(v), \dots, f_n(v))$.

$$\begin{aligned} (\varphi_{t(l_1)}(y_1), \dots, \varphi_{t(l_n)}(y_n)) &= ((\varphi_{t(l_1)}(z_1 + f_1(v)), \dots, \varphi_{t(l_n)}(z_n + f_n(v))) \\ &= (\varphi_{t(l_1)}(z_1), \dots, \varphi_{t(l_n)}(z_n)) + (\varphi_{t(l_1)}(f_1(v)), \dots, \varphi_{t(l_n)}(f_n(v))) \\ &= (\varphi_{t(l_1)}(z_1), \dots, \varphi_{t(l_n)}(z_n)) + (f'_1((\varphi_\alpha(v)), \dots, f'_n((\varphi_\alpha(v)))) \end{aligned}$$

car pour tout $i \in \llbracket 1, n \rrbracket$, on a $\varphi_{t(l_i)} f_{l_i} = f'_{l_i} \varphi_\alpha$, donc finalement $F_\alpha^-(\varphi)_\alpha$ est bien définie.

Soit g_{l_i} comme définie ci-dessus et pour tout $i \in \llbracket 1, n \rrbracket, g'_{l_i}$ la composition de l'inclusion naturelle de $V'_{t(l_i)}$ dans $\oplus_{i=1}^n V'_{t(l_i)}$ avec la projection de $\oplus_{i=1}^n V'_{t(l)}$ dans $\oplus_{i=1}^n V'_{t(l)} / \text{Im } \tilde{h}'$. Montrons alors que pour tout $i \in \llbracket 1, n \rrbracket$ et $x \in V_{t(l_i)}$, on ait $F_\alpha^-(\varphi)_\alpha g_{l_i}(x) = g'_{l_i} \varphi_{l_i}(x)$ pour vérifier que $F_\alpha^-(\varphi)$ est bien un morphisme de représentation dans $\mathcal{L}(\Gamma, \sigma_\alpha \Lambda)$. Et en effet,

$$\begin{aligned} F_\alpha^-(\varphi)_\alpha g_{l_i}(x) &= F_\alpha^-(\varphi)_\alpha ((0, \dots, x, \dots, 0) + \text{Im } \tilde{h}) \\ &= (0, \dots, \varphi_{t(l_i)}(x), \dots, 0) + \text{Im } \tilde{h}' \\ &= g'_{l_i} \varphi_{t(l_i)}(x) \end{aligned}$$

Montrons dès à présent que la composition des morphismes de représentations est respectée par F_α^- . Soit (V, f) , (V', f') et (V'', f'') trois représentations de $\mathcal{L}(\Gamma, \Lambda)$ et (W, g) , (W', g') et (W'', g'') leur image par F_α^- , et φ (respectivement φ') un morphisme de (V, f) dans (V', f') (respectivement (V', f') dans (V'', f'')). Ainsi, pour tout $\gamma \neq \alpha$, on a par définition de F_α^- , $F_\alpha^-(\varphi'\varphi)_\gamma = (\varphi'\varphi)_\gamma = \varphi'_\gamma\varphi_\gamma = F_\alpha^-(\varphi')_\gamma F_\alpha^-(\varphi)_\gamma$.

$$\begin{aligned} \forall (y_1, \dots, y_n) + \text{Im } \tilde{h} \in \bigoplus_{i=1}^n V_{t(i)}/\text{Im } \tilde{h}, \\ F_\alpha^-(\varphi'\varphi)_\alpha((y_1, \dots, y_n) + \text{Im } \tilde{h}) &= ((\varphi'\varphi)_{t(l_1)}(y_1), \dots, (\varphi'\varphi)_{t(l_n)}(y_n)) + \text{Im } \tilde{h}'' \\ &= F_\alpha^-(\varphi')_\alpha F_\alpha^-(\varphi)_\alpha((y_1, \dots, y_n) + \text{Im } \tilde{h}) \end{aligned}$$

On a bien montré que $F_\alpha^-(\varphi'\varphi) = F_\alpha^-(\varphi')F_\alpha^-(\varphi)$.

Il reste à voir Id le morphisme identité d'une représentation (V, f) est envoyé sur le morphisme identité de $F_\alpha^-(V, f)$ par F_α^- . Pour tout $\gamma \neq \alpha$, on a $F_\alpha^-(\text{Id})_\gamma = \text{Id}_\gamma$ et :

$$\begin{aligned} \forall (y_1, \dots, y_n) + \text{Im } \tilde{h} \in \bigoplus_{i=1}^n V_{t(i)}/\text{Im } \tilde{h}, \\ F_\alpha^-(\text{Id})_\alpha((y_1, \dots, y_n) + \text{Im } \tilde{h}) &= (y_1, \dots, y_n) + \text{Im } \tilde{h} \end{aligned}$$

Donc on a le résultat. \square

Définition 5.3.5. Soit $\beta \in \Gamma_0$ un puits suivant l'orientation Λ , alors β est une source suivant l'orientation $\sigma_\beta\Lambda$. On peut donc définir le foncteur : $F_\beta^- F_\beta^+ : \mathcal{L}(\Gamma, \Lambda) \rightarrow \mathcal{L}(\Gamma, \Lambda)$. Soit $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$, on note $(Z, e) = F_\beta^- F_\beta^+(V, f)$ l'image de (V, f) par $F_\beta^- F_\beta^+$ et on construit alors une application $i_V^\beta : (Z, e) \rightarrow (V, f)$ de la manière suivante :

$$\forall \gamma \neq \beta, Z_\gamma = V_\gamma, \text{ on pose donc : } (i_V^\beta)_\gamma = \text{Id}_{V_\gamma}.$$

Pour $(i_V^\beta)_\beta$, on prend l'application :

$$\begin{aligned} Z_\beta = \bigoplus_{l \in \Gamma^\beta} V_{s(l)}/\text{Im } \tilde{h} = \bigoplus_{l \in \Gamma^\beta} V_{s(l)}/\text{Ker } h &\rightarrow V_\beta \\ (v_1, \dots, v_n) + \text{Ker } h &\mapsto \sum_{i=1}^n f_{l_i}(v_i) \end{aligned}$$

où $\Gamma^\beta = \{l_1, \dots, l_n\}$.

Soit $\alpha \in \Gamma_0$ une source suivant l'orientation Λ , alors α est un puits suivant l'orientation $\sigma_\alpha\Lambda$. On peut donc définir le foncteur : $F_\alpha^+ F_\alpha^- : \mathcal{L}(\Gamma, \Lambda) \rightarrow \mathcal{L}(\Gamma, \Lambda)$. Soit $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$, on note $(Z, e) = F_\alpha^+ F_\alpha^-(V, f)$ l'image de (V, f) par $F_\alpha^+ F_\alpha^-$ et on construit alors une application $p_V^\alpha : (V, f) \rightarrow (Z, e)$ de la manière suivante :

$$\forall \gamma \neq \alpha, Z_\gamma = V_\gamma, \text{ on pose donc : } (p_V^\alpha)_\gamma = \text{Id}_{Z_\gamma}.$$

Pour $(p_V^\alpha)_\alpha$, on prend l'application :

$$\begin{array}{ccc} V_\alpha & \rightarrow & Z_\alpha \\ v & \mapsto & (f_{l_1}(v), \dots, f_{l_n}(v)) \end{array}$$

où $\Gamma^\alpha = \{l_1, \dots, l_n\}$.

Proposition 5.3.6

L'application i_V^β est un morphisme de (Z, e) dans (V, f) et l'application p_V^α est un morphisme de (V, f) dans (Z, e) .

Démonstration. Notons $(W, g) = F_\beta^+(V, f)$ l'image de (V, f) par F_β^+ . Dans la définition de $(i_V^\beta)_\beta$, on a implicitement déclaré que $\text{Ker } h = \text{Im } \tilde{h}$, nous allons le prouver. Considérons la suite d'applications suivante :

$$W_\beta \xrightarrow{\tilde{h}} \bigoplus_{l \in \Gamma^\beta} V_{s(l)} \xrightarrow{h} V_\beta,$$

on a alors :

$$\begin{aligned} \text{Im } \tilde{h} &= \{(g_{l_1}(v), \dots, g_{l_n}(v)) \mid v = (v_1, \dots, v_n) \in W_\beta\} \\ &= \{(v_1, \dots, v_n) \mid (v_1, \dots, v_n) \in \text{Ker } h\} \\ &= \text{Ker } h. \end{aligned}$$

Car $W_\beta = \text{Ker } h$ par définition et pour tout $i \in \llbracket 1, n \rrbracket$, g_{l_i} est la composée de l'injection $\text{Ker } h \hookrightarrow \bigoplus_{l \in \Gamma^\beta} V_{s(l)}$ avec la projection $\bigoplus_{l \in \Gamma^\beta} V_{s(l)} \rightarrow V_{s(l_i)}$. Prouvons désormais que i_V^β est un morphisme, c'est-à-dire que le diagramme suivant commute :

$$\begin{array}{ccc} Z_{s(l)} & \xrightarrow{e_l} & Z_{t(l)} \\ (i_V^\beta)_{s(l)} \downarrow & & \downarrow (i_V^\beta)_{t(l)} \\ V_{s(l)} & \xrightarrow{f_l} & V_{t(l)} \end{array}$$

ou encore que $\forall l \in \Gamma_1$, on a $(i_V^\beta)_{t(l)} e_l = f_l (i_V^\beta)_{s(l)}$. Commençons par prendre $l \notin \Gamma^\beta$, on a alors $(i_V^\beta)_{s(l)} = \text{Id}_{V_{s(l)}}$, $(i_V^\beta)_{t(l)} = \text{Id}_{V_{t(l)}}$, et $e_l = f_l$, d'où l'égalité souhaitée. Voyons maintenant le cas où $l \in \Gamma^\beta$, on alors $t(l) = \beta$. Vérifions dans un premier temps que l'application $(i_V^\beta)_\beta$ est bien définie. Pour cela prenons deux représentants de la même classe dans Z_β : (v_1, \dots, v_n) et (v'_1, \dots, v'_n) , et vérifions qu'ils ont la même image par $(i_V^\beta)_\beta$. On a $(v_1 - v'_1, \dots, v_n - v'_n) \in \text{Ker } h$, d'où :

$$\begin{aligned} 0 &= \sum_{i=1}^n f_{l_i}(v_i - v'_i) \\ &= \sum_{i=1}^n f_{l_i}(v_i) - \sum_{i=1}^n f_{l_i}(v'_i) \\ &= (i_V^\beta)_\beta((v_1, \dots, v_n) + \text{Ker } h) - (i_V^\beta)_\beta((v'_1, \dots, v'_n) + \text{Ker } h). \end{aligned}$$

Et on a bien $(i_V^\beta)_\beta((v_1, \dots, v_n) + \text{Ker } h) = (i_V^\beta)_\beta((v'_1, \dots, v'_n) + \text{Ker } h)$, d'où le caractère bien définie de $(i_V^\beta)_\beta$. Notons $l = l_j$ pour un certain $j \in \llbracket 1, n \rrbracket$ et vérifions que $f_{l_j} \text{Id}_{V_{s(l_j)}} = (i_V^\beta)_\beta e_{l_j}$. Soit $v_j \in V_{s(l_j)}$, on a d'une part :

$$f_{l_j} \text{Id}_{V_{s(l_j)}}(v_j) = f_{l_j}(v_j)$$

et d'autre part :

$$\begin{aligned} (i_V^\beta)_\beta e_{l_j}(v_j) &= (i_V^\beta)_\beta((0, \dots, 0, v_j, 0, \dots, 0) + \text{Ker } h) \\ &= f_{l_j}(v_j). \end{aligned}$$

Et on a encore l'égalité souhaitée, ainsi i_V^β est bien un morphisme.

Montrons de même que p_V^α est un morphisme. Commençons par montrer que $(p_V^\alpha)_\alpha$ est bien définie, c'est-à-dire que son image est contenue dans $F_\alpha^+ F_\alpha^-(V)$. Notons $(W, g) = F_\alpha^-(V, f)$, alors si $\Gamma^\alpha = \{l_1, \dots, l_n\}$, on sait pour tout $j \in \llbracket 1, n \rrbracket$, g_{l_j} est la composition de l'inclusion naturelle de $W_{t(l_j)}$ dans $\bigoplus_{i=1}^n W_{t(l_i)}$ et de la projection dans $\bigoplus_{i=1}^n W_{t(l_i)} / \text{Im}(\tilde{h})$.

$$\begin{aligned} \text{Ker } h &= \left\{ (w_1, \dots, w_n) \in \bigoplus_{i=1}^n W_{t(l_i)} \mid \sum_{j=1}^n g_{l_j}(w_j) = 0 \right\} \\ &= \left\{ (v_1, \dots, v_n) \in \bigoplus_{i=1}^n V_{t(l_i)} \mid (v_1, \dots, v_n) + \text{Im } \tilde{h} = 0 \right\} \\ &= \text{Im } \tilde{h} \\ &= \{(f_{l_1}(v), \dots, f_{l_n}(v)), v \in V_\alpha\} \end{aligned}$$

Donc p_V^α est bien définie, on peut alors montrer que c'est un morphisme.

Soit $l \in \Gamma_1$, on souhaite que le diagramme suivant commute.

$$\begin{array}{ccc} V_{s(l)} & \xrightarrow{f_l} & V_{t(l)} \\ (p_V^\alpha)_{s(l)} \downarrow & & \downarrow (p_V^\alpha)_{t(l)} \\ Z_{s(l)} & \xrightarrow{e_l} & Z_{t(l)} \end{array}$$

Si $l \notin \Gamma^\alpha$ alors $(p_V^\alpha)_{s(l)} = \text{Id}_{Z_{s(l)}}$, $(p_V^\alpha)_{t(l)} = \text{Id}_{Z_{t(l)}}$ et $e_l = f_l$ donc $e_l(p_V^\alpha)_{s(l)} = (p_V^\alpha)_{t(l)} f_l$. Si $l \in \Gamma^\alpha$ alors $s(l) = \alpha$, $(p_V^\alpha)_{t(l)} = \text{Id}_{Z_{t(l)}}$ et pour tout $v \in V_\alpha$,

$$\begin{aligned} e_l(p_V^\alpha)_\alpha(v) &= e_l(f_{l_1}(v), \dots, f_{l_n}(v)) \\ &= f_l(v) \end{aligned}$$

Car on sait que pour tout e_l est la composition de l'inclusion naturelle de $\text{Ker } h$ dans $\bigoplus_{i=1}^n Z_{t(l_i)}$ et de la projection dans $Z_{t(l)}$. □

Lemme 5.3.7

On a les résultats suivants :

1. $F_\alpha^\pm((V_1, f) \oplus (V_2, g)) \cong F_\alpha^\pm(V_1, f) \oplus F_\alpha^\pm(V_2, g)$;
2. p_V^α est surjectif et i_V^β est injectif ;
3. (a) si i_V^β est un isomorphisme, alors on a les formules :

$$\begin{aligned} \forall \gamma \neq \beta, \dim F_\beta^+(V, f)_\gamma &= \dim V_\gamma \\ \dim F_\beta^+(V, f)_\beta &= -\dim V_\beta + \sum_{l \in \Gamma^\beta} \dim V_{s(l)} \end{aligned}$$

- (b) si p_V^α est un isomorphisme, alors on a les formules :

$$\begin{aligned} \forall \gamma \neq \alpha, \dim F_\alpha^-(V, f)_\gamma &= \dim V_\gamma \\ \dim F_\alpha^-(V, f)_\alpha &= -\dim V_\alpha + \sum_{l \in \Gamma^\alpha} \dim V_{t(l)}; \end{aligned}$$

4. $\forall \gamma \neq \alpha, (\text{Ker } p_V^\alpha)_\gamma = \{0\}$ et $\forall \gamma \neq \beta, (V/\text{Im } i_V^\beta)_\gamma = \{0\}$;
5. si V est de la forme $F_\alpha^+(W, g)$ (respectivement $F_\beta^-(W, g)$), alors p_V^α (i_V^β) est un isomorphisme ;
6. V est isomorphe à $F_\beta^- F_\beta^+(V, f) \oplus V/\text{Im } i_V^\beta$, de même V est isomorphe à $F_\alpha^+ F_\alpha^-(V, f) \oplus \text{Ker } p_V^\alpha$.

Démonstration. Prouvons le point 1. Soit $\beta \in \Gamma_0$ un puits de Γ suivant l'orientation Λ , soient de plus (V, f) et (V', f') deux représentations de (Γ, Λ) . On veut montrer que $F_\beta^+((V, f) \oplus (V', f')) = F_\beta^+(V, f) \oplus F_\beta^+(V', f')$. Il vient :

$$\begin{aligned} \forall \gamma \neq \beta, F_\beta^+((V, f) \oplus (V', f'))_\gamma &= ((V, f) \oplus (V', f'))_\gamma \\ &= V_\gamma \oplus V'_\gamma \\ &= F_\beta^+(V, f)_\gamma \oplus F_\beta^+(V', f')_\gamma \end{aligned}$$

$$\begin{aligned} F_\beta^+((V, f) \oplus (V', f'))_\beta &= \text{Ker}(h \oplus h') \\ &\cong \text{Ker } h \oplus \text{Ker } h' \\ &= F_\beta^+(V, f)_\beta \oplus F_\beta^+(V', f')_\beta, \end{aligned}$$

où :

$$\begin{aligned} h : \quad \bigoplus_{j=1}^n V_{s(l_j)} &\rightarrow V_\beta \\ (v_1, \dots, v_n) &\mapsto \sum_{i=1}^n f_{l_i}(v_i) \end{aligned}$$

$$\begin{aligned} h' : \quad \bigoplus_{j=1}^n V'_{s(l_j)} &\rightarrow V'_\beta \\ (v'_1, \dots, v'_n) &\mapsto \sum_{i=1}^n f'_{l_i}(v'_i). \end{aligned}$$

et où $\Gamma^\beta = \{l_1, \dots, l_n\}$. Ainsi on a bien l'égalité annoncée.

Supposons maintenant que α est un source dans (Γ, Λ) , (V, f) et (V', f') deux représentations de $\mathcal{L}(\Gamma, \Lambda)$, on a :

$$\begin{aligned} \forall \gamma \neq \alpha, F_\alpha^-((V, f) \oplus (V', f'))_\gamma &= ((V, f) \oplus (V', f'))_\gamma \\ &= V_\gamma \oplus V'_\gamma \\ &= F_\alpha^-(V, f)_\gamma \oplus F_\alpha^-(V', f')_\gamma \\ \\ F_\alpha^-((V, f) \oplus (V', f'))_\alpha &= \bigoplus_{i=1}^n (V_{l_i} \oplus V'_{l_i}) / \text{Im } \tilde{h}_\oplus \\ &\cong \left(\bigoplus_{i=1}^n V_{l_i} / \text{Im } \tilde{h} \right) \oplus \left(\bigoplus_{i=1}^n V'_{l_i} / \text{Im } \tilde{h}' \right) \\ &= F_\alpha^-(V, f)_\alpha \oplus F_\alpha^-(V', f')_\alpha \end{aligned}$$

où :

$$\begin{aligned} \tilde{h} : \quad V_\alpha &\rightarrow \bigoplus_{j=1}^n V_{t(l_j)} \\ v &\mapsto (f_{l_1}(v), \dots, f_{l_n}(v)) \\ \\ \tilde{h}' : \quad V'_\alpha &\rightarrow \bigoplus_{j=1}^n V'_{t(l_j)} \\ v' &\mapsto (f'_{l_1}(v'), \dots, f'_{l_n}(v')) \\ \\ \tilde{h}_\oplus : \quad V_\alpha \oplus V'_\alpha &\rightarrow \bigoplus_{j=1}^n (V_{t(l_j)} \oplus V'_{t(l_j)}) \\ (v, v') &\mapsto ((f_{l_1}(v), f'_{l_1}(v')), \dots, (f_{l_n}(v), f'_{l_n}(v'))) \end{aligned}$$

Prouvons désormais le point 2. Dire que le morphisme i_V^β est injectif signifie que pour tout $\gamma \in \Gamma_0$, l'application $(i_V^\beta)_\gamma$ est injective. Comme $\forall \gamma \neq \beta$, $(i_V^\beta)_\gamma = \text{Id}_{V_\gamma}$, l'injectivité de $(i_V^\beta)_\gamma$ est assurée pour tout $\gamma \neq \beta$. Regardons maintenant le cas $\gamma = \beta$. Soient $a = (v_1, \dots, v_n) + \text{Ker } h$ et $b = (v'_1, \dots, v'_n) + \text{Ker } h$ tels que $(i_V^\beta)_\beta(a) = (i_V^\beta)_\beta(b)$. On alors :

$$\begin{aligned} \sum_{i=1}^n f_{l_i}(v_i) &= \sum_{i=1}^n f_{l_i}(v'_i) \\ \sum_{i=1}^n f_{l_i}(v_i - v'_i) &= 0 \end{aligned}$$

d'où $(v_1 - v'_1, \dots, v_n - v'_n) \in \text{Ker } h$, donc $a = b$, et on a l'injectivité de $(i_V^\beta)_\beta$ puis de i_V^β .

Le morphisme p_V^α est quant à lui surjectif, car pour tout $\gamma \neq \alpha$, $(p_V^\alpha)_\gamma = \text{Id}_{V_\gamma}$ est surjective. On a montré que $\text{Ker}(h) = \text{Im } \tilde{h}$, on observe que $\text{Im}(p_V^\alpha)_\alpha = \text{Im } \tilde{h}$ et que $F_\alpha^+ F_\alpha^-(V)_\alpha = \text{Ker}(h)$, donc $(p_V^\alpha)_\alpha$ est surjective.

Prouvons maintenant le point 3. Supposons que i_V^β est un isomorphisme, en particulier, $(i_V^\beta)_\beta$ est un isomorphisme et on a $\bigoplus_{l \in \Gamma^\beta} V_{s(l)} / \text{Ker } h \cong V_\beta$. On a égalité entre les dimensions d'espaces vectoriels isomorphes, et ainsi il vient :

$$\begin{aligned} \dim(\bigoplus_{l \in \Gamma^\beta} V_{s(l)} / \text{Ker } h) &= \dim V_\beta \\ \sum_{l \in \Gamma^\beta} \dim V_{s(l)} - \dim \text{Ker } h &= \dim V_\beta \\ \sum_{l \in \Gamma^\beta} \dim V_{s(l)} - \dim V_\beta &= \dim \text{Ker } h \\ \sum_{l \in \Gamma^\beta} \dim V_{s(l)} - \dim V_\beta &= \dim F_\beta^+(V, f)_\beta. \end{aligned}$$

Comme $\forall \gamma \neq \beta$, $F_\beta^+(V, f)_\gamma = V_\gamma$, on a immédiatement $\forall \gamma \neq \beta$, $\dim F_\beta^+(V, f)_\gamma = \dim V_\gamma$.

Si p_V^α est un isomorphisme alors :

$$\forall \gamma \neq \alpha, \dim F_\alpha^-(V)_\gamma = \dim V_\gamma$$

Comme on a $V_\alpha \cong \text{Im } \tilde{h}$,

$$\begin{aligned} \dim F_\alpha^-(V)_\alpha &= \dim \left(\bigoplus_{i=1}^n V_{l_i} / \text{Im } \tilde{h} \right) \\ &= \sum_{i=1}^n \dim V_{l_i} - \dim \text{Im } \tilde{h} \\ &= \sum_{i=1}^n \dim V_{l_i} - \dim V_\alpha \end{aligned}$$

Prouvons à présent le point 4. Pour tout $\gamma \neq \beta$, on a $(V / \text{Im } i_V^\beta)_\gamma = V_\gamma / \text{Im}((i_V^\beta)_\gamma)$ et $(i_V^\beta)_\gamma = \text{Id}_{V_\gamma}$, d'où $\text{Im}((i_V^\beta)_\gamma) = V_\gamma$ et ainsi $V_\gamma / V_\gamma = \{0\}$.

De même, pour tout $\gamma \neq \beta$, on a $(\text{Ker } p_V^\alpha)_\gamma = \text{Ker}((p_V^\alpha)_\gamma)$ et $(p_V^\alpha)_\gamma = \text{Id}_{V_\gamma}$, d'où $\text{Ker}((p_V^\alpha)_\gamma) = \{0\}$.

Passons à la preuve du point 5. Pour cela, supposons que $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ est de la forme $F_\beta^-(W, g)$, il nous faut alors voir que i_V^β est surjectif. Pour tout $\gamma \neq \beta$, on a $(i_V^\beta)_\gamma = \text{Id}_{V_\gamma}$, la surjectivité de $(i_V^\beta)_\gamma$ est donc immédiate. Intéressons-nous maintenant au cas où $\gamma = \beta$. On se place dans l'orientation Λ où β est un puits et on note $\Gamma^\beta = \{l_1, \dots, l_n\}$, on a alors $V_\beta = \bigoplus_{i=1}^n V_{s(l_i)} / \text{Im } \tilde{h}$. On a aussi :

$$\begin{aligned} h : \quad \bigoplus_{i=1}^n V_{s(l_i)} &\rightarrow V_\beta \\ (v_1, \dots, v_n) &\mapsto \sum_{i=1}^n f_{l_i}(v_i) \end{aligned}$$

où, pour tout $i \in \llbracket 1, n \rrbracket$:

$$\begin{aligned} f_{l_i} : \quad V_{s(l_i)} &\rightarrow V_\beta \\ v_i &\mapsto (0, \dots, 0, v_i, 0, \dots, 0) + \text{Im } \tilde{h} \end{aligned}$$

d'où :

$$\begin{aligned} h : \quad \bigoplus_{i=1}^n V_{s(l_i)} &\rightarrow V_\beta \\ (v_1, \dots, v_n) &\mapsto (v_1, \dots, v_n) + \text{Im } \tilde{h} \end{aligned}$$

On a alors immédiatement que $\text{Ker } h = \text{Im } \tilde{h}$, et on a

$$(i_V^\beta)_\beta : \begin{array}{ccc} \bigoplus_{i=1}^n V_{s(l_i)} / \text{Ker } h & \rightarrow & V_\beta \\ (v_1, \dots, v_n) + \text{Ker } h & \mapsto & (v_1, \dots, v_n) + \text{Im } \tilde{h}, \end{array}$$

la surjectivité de $(i_V^\beta)_\beta$ est donc prouvée. Ainsi i_V^β est surjectif, puis bijectif, donc est un isomorphisme.

De manière similaire, montrons que p_V^α est un isomorphisme, c'est-à-dire que $(p_V^\alpha)_\alpha$ est injective, dans le cas où $(V, f) = F_\alpha^+(W, g)$. Écrivons $\Gamma^\alpha = \{l_1, \dots, l_n\}$, on sait que $V_\alpha = \{(v_1, \dots, v_n) \in \bigoplus_{i=1}^n V_{t(l_i)} \mid \sum_{i=1}^n f_{l_i}(v_i) = 0\}$, et que $\forall j \in \llbracket 1, n \rrbracket$, f_{l_j} est la composition de l'inclusion de V_α dans $\bigoplus_{i=1}^n V_{t(l_i)}$ et la projection sur $V_{t(l_j)}$. Donc soit $v = (v_1, \dots, v_n) \in V_\alpha$, on a $f_{l_j}(v) = v_j$. Or si $(p_V^\alpha)_\alpha(v) = 0$ c'est-à-dire $(f_{l_1}(v), \dots, f_{l_n}(v)) = 0$ alors $v = 0$ donc $(p_V^\alpha)_\alpha$ est injective.

Enfin, prouvons le point 6. Notons $\tilde{V} = V / \text{Im } i_V^\beta$, il nous faut alors prouver que V est isomorphe à $F_\beta^- F_\beta^+(V) \oplus \tilde{V}$. Notons également $\varphi'_\beta : V_\beta \rightarrow \tilde{V}_\beta$ la projection de V_β dans \tilde{V}_β . Cette représentation admet une section, c'est-à-dire une application $\varphi_\beta : \tilde{V}_\beta \rightarrow V_\beta$ telle que $\varphi'_\beta \varphi_\beta = \text{Id}_{\tilde{V}_\beta}$: en effet il suffit d'envoyer chaque classe d'équivalence dans \tilde{V}_β sur un unique représentant de cette classe dans V_β . On posera également, pour tout $\gamma \neq \beta$, $\varphi_\gamma = 0$, ce qui fait de $\varphi = (\varphi_\gamma)_{\gamma \in \Gamma_0}$ un morphisme de $V / \text{Im } i_V^\beta$ dans V . En effet, le diagramme suivant commute :

$$\begin{array}{ccc} \tilde{V}_{s(l)} & \xrightarrow{\tilde{f}_l} & \tilde{V}_{t(l)} \\ \varphi_{s(l)} \downarrow & & \downarrow \varphi_{t(l)} \\ V_{s(l)} & \xrightarrow{f_l} & V_{t(l)} \end{array}$$

car on a d'une part $\forall l \in \Gamma_0$, $s(l) \neq \beta$ (β étant un puits) et ainsi $\varphi_{s(l)} = 0$ d'où $f_l \varphi_{s(l)} = 0$. Et d'autre part, comme on a :

$$\tilde{f}_l : \begin{array}{ccc} \tilde{V}_{s(l)} & \rightarrow & \tilde{V}_{t(l)} \\ v + \text{Im} \left[(i_V^\beta)_{s(l)} \right] & \mapsto & f_l(v) + \text{Im} \left[(i_V^\beta)_{t(l)} \right] \end{array}$$

où $\tilde{V}_{s(l)} = \{0\}$ car $\forall l \in \Gamma_0$, $s(l) \neq \beta$, il vient $\tilde{f}_l = 0$ puis $\varphi_{t(l)} \tilde{f}_l = 0$. Ainsi φ est bien un morphisme. On pose désormais $\xi = (\xi_\gamma)_{\gamma \in \Gamma_0}$ où $\forall \gamma \neq \beta$, $\xi_\gamma = \text{Id}_{V_\gamma}$ et ξ_β est définie par :

$$\xi_\beta : \begin{array}{ccc} \tilde{V}_\beta \oplus F_\beta^- F_\beta^+(V, f)_\beta & \rightarrow & V_\beta \\ (x, y) & \mapsto & \varphi_\beta(x) + (i_V^\beta)_\beta(y). \end{array}$$

Nous allons prouver que ξ est un isomorphisme. Tout d'abord, ξ est bien définie car $\forall \gamma \neq \beta$, $\xi_\gamma = \text{Id}_{V_\gamma}$, et comme φ_β et $(i_V^\beta)_\beta$ sont bien définies, on a que ξ_β est bien définie également. Prouvons maintenant que ξ est un morphisme, c'est-à-dire que le diagramme suivant commute :

$$\begin{array}{ccc} \left[\tilde{V} \oplus F_\beta^- F_\beta^+(V, f) \right]_{s(l)} & \xrightarrow{\tilde{f}_l \oplus e_l} & \left[\tilde{V} \oplus F_\beta^- F_\beta^+(V, f) \right]_{t(l)} \\ \xi_{s(l)} \downarrow & & \downarrow \xi_{t(l)} \\ V_{s(l)} & \xrightarrow{f_l} & V_{t(l)} \end{array}$$

où les e_l sont les morphismes de $F_\beta^- F_\beta^+(V, f)$. Soit $(x, y) \in \left[\tilde{V} \oplus F_\beta^- F_\beta^+(V, f) \right]_{s(l)}$:

$$\begin{aligned} \xi_{t(l)}(\tilde{f}_l \oplus e_l(x, y)) &= \xi_{t(l)}(\tilde{f}_l(x), e_l(y)) \\ &= \varphi_{t(l)} \tilde{f}_l(x) + (i_V^\beta)_{t(l)} e_l(y) \\ &= f_l \varphi_{s(l)}(x) + f_l (i_V^\beta)_{s(l)}(y) \text{ car } \varphi \text{ et } i_V^\beta \text{ sont des morphismes} \\ &= f_l(\varphi_{s(l)}(x) + (i_V^\beta)_{s(l)}(y)) \\ &= f_l \xi_{s(l)}(x, y) \end{aligned}$$

ainsi ξ est bien un morphisme. Prouvons son injectivité. Celle-ci est immédiatement vérifiée dans le cas où $\gamma \neq \beta$, vérifions qu'elle l'est aussi pour $\gamma = \beta$. Soient $(x, y), (x', y') \in \tilde{V}_\beta \oplus F_\beta^- F_\beta^+(V, f)_\beta$ tels que $\xi_\beta(x, y) = \xi_\beta(x', y')$, a fortiori on a $\varphi'_\beta(\xi_\beta(x, y)) = \varphi'_\beta(\xi_\beta(x', y'))$, d'où $x = x'$. Ainsi $(i_V^\beta)_\beta(y) = (i_V^\beta)_\beta(y')$, mais par injectivité de i_V^β , il vient $y = y'$. Ainsi $(x, y) = (x', y')$ et ξ est injective. Prouvons donc la surjectivité de ξ , une fois encore, elle est immédiate dès que $\gamma \neq \beta$, nous nous intéressons donc au cas $\gamma = \beta$. Soit donc $v \in V_\beta$, on a alors $v = \varphi_\beta \varphi'_\beta(v) + v - \varphi_\beta \varphi'_\beta(v)$, d'où il vient immédiatement que $\varphi_\beta \varphi'_\beta(v) \in \text{Im } \varphi_\beta$. De plus, comme $\varphi'_\beta \varphi_\beta = \text{Id}_{\tilde{V}_\beta}$, on sait que $\varphi'_\beta(v) = \varphi'_\beta \varphi_\beta \varphi'_\beta(v)$ d'où $v - \varphi_\beta \varphi'_\beta(v) \in \text{Im } [(i_V^\beta)_\beta]$. On a donc la surjectivité de ξ_β , donc la surjectivité de ξ et la bijectivité de ξ , qui est un morphisme. On a donc prouvé que ξ est un isomorphisme, et ainsi (V, f) est isomorphe à $F_\beta^- F_\beta^+(V, f) \oplus V / \text{Im } i_V^\beta$.

Considérons la représentation (\tilde{V}, \tilde{f}) où $\tilde{V} = \text{Ker } p_V^\alpha$ et \tilde{f} est formée par :

- \tilde{f}_l est un morphisme de l'espace nul dans l'espace nul, lorsque $l \notin \Gamma^\alpha$;
- \tilde{f}_l est un morphisme de \tilde{V}_α dans l'espace nul, lorsque $l \in \Gamma^\alpha$.

On construit φ , le morphisme de (V, f) dans (\tilde{V}, \tilde{f}) défini par $\forall \gamma \in \Gamma_0 \setminus \{\alpha\}$, on ait $\varphi_\gamma(V_\gamma) = \tilde{V}_\gamma = \{0\}$ et $\varphi_\alpha(V_\alpha) = \tilde{V}_\alpha$ c'est-à-dire que φ_α est la projection de \tilde{V}_α dans V_α . On vérifie alors que le diagramme suivant commute

quelque soit $l \in \Gamma_1$.

$$\begin{array}{ccc} V_{s(l)} & \xrightarrow{f_l} & V_{t(l)} \\ \varphi_{s(l)} \downarrow & & \downarrow \varphi_{t(l)} \\ \tilde{V}_{s(l)} & \xrightarrow{\tilde{f}_l} & \tilde{V}_{t(l)} \end{array}$$

On construit alors le morphisme ξ de (V, f) dans $(Z, e) \oplus (\tilde{V}, \tilde{f})$ en sommant l'action des morphismes p_V^α et φ , c'est-à-dire que pour tout $\gamma \in \Gamma_0$, $\xi_\gamma(V, f) = p_V^\alpha(V, f) \oplus \varphi_\gamma(V, f)$. Comme p_V^α et φ sont respectivement des morphismes de (V, f) dans (Z, e) et de (V, f) dans (\tilde{V}, \tilde{f}) , alors ξ est un morphisme de (V, f) dans $(Z, e) \oplus (\tilde{V}, \tilde{f})$.

Il s'agit maintenant de montrer que ξ est un isomorphisme, c'est-à-dire que pour tout $\gamma \in \Gamma_0$, ξ_γ est un isomorphisme d'espace vectoriel. Lorsque $\gamma \in \Gamma_0 \setminus \alpha$, $(p_V^\alpha)_\gamma = \text{Id}_{V_\gamma}$ et φ_γ est l'application nulle, donc ξ_γ est un isomorphisme de V_γ dans $V_\gamma \oplus \{0\}$. Le morphisme ξ_α est un isomorphisme, en effet il est surjectif, car p_V^α et φ_α sont surjectifs d'après ce qui précède et parce que φ_α est une projection. De plus, ξ_α est injectif, soit $v \in V_\alpha$ tel que $((p_V^\alpha)_\alpha(v), \varphi_\alpha(v)) = 0$, alors v appartient à $(\text{Ker } p_V^\alpha)_\alpha$, donc $\varphi_\alpha(v) = v$ d'où $v = 0$. Donc ξ est bien un isomorphisme et $(V, f) \cong (Z, e) \oplus (\tilde{V}, \tilde{f})$. \square

Théorème 5.3.8

Soit (Γ, Λ) un carquois.

1. Soit $\beta \in \Gamma_0$ un puits suivant l'orientation Λ . Soit $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ une représentation indécomposable. On a alors deux cas possibles :
 - (a) $V \cong L_\beta$ (où L_β a été défini à l'exemple 5.2.15) et $F_\beta^+(V, f) = 0$;
 - (b) $F_\beta^+(V, f)$ est indécomposable, $F_\beta^- F_\beta^+(V, f) = (V, f)$ et les dimensions des espaces $F_\beta^+(V, f)_\gamma$ sont données par :

$$\begin{aligned} \forall \gamma \neq \beta, \dim F_\beta^+(V, f)_\gamma &= \dim V_\gamma \\ \dim F_\beta^+(V, f)_\beta &= -\dim V_\beta + \sum_{l \in \Gamma^\beta} \dim V_{s(l)}. \end{aligned}$$

2. Soit $\alpha \in \Gamma_0$ un source suivant l'orientation Λ . Soit $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ une représentation indécomposable. On a alors deux cas possibles :
 - (a) $V \cong L_\alpha$ (où L_α a été défini à l'exemple 5.2.15) et $F_\alpha^-(V, f) = 0$;
 - (b) $F_\alpha^-(V, f)$ est indécomposable, $F_\alpha^+ F_\alpha^-(V, f) = (V, f)$ et les dimensions des espaces $F_\alpha^-(V, f)_\gamma$ sont données par :

$$\begin{aligned} \forall \gamma \neq \alpha, \dim F_\alpha^-(V, f)_\gamma &= \dim V_\gamma \\ \dim F_\alpha^-(V, f)_\alpha &= -\dim V_\alpha + \sum_{l \in \Gamma^\alpha} \dim V_{t(l)}. \end{aligned}$$

Démonstration. Soit (Γ, Λ) un carquois et $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ une représentation indécomposable de ce carquois. Supposons que β est un puits suivant l'orientation Λ . D'après le point 6 du lemme 5.3.7, on a $V \cong F_\beta^- F_\beta^+(V, f) \oplus V/\text{Im } i_V^\beta$, or V est indécomposable, donc V est isomorphe à l'un des deux termes.

- Cas 1. $V \cong V/\text{Im } i_V^\beta : \forall \gamma \neq \beta, V_\gamma = \{0\}, \forall l \in \Gamma_1, f_l = 0$ et $V_\beta = \mathbb{K}$, car (V, f) est indécomposable, et si on avait $\dim V_\beta > 1$, on pourrait décomposer (V, f) en $\dim V_\beta$ copies de L_β . Ainsi $V \cong L_\beta$.
- Cas 2. $V \cong F_\beta^- F_\beta^+(V, f)$: autrement dit, i_V^β est un isomorphisme. Ainsi, d'après le point 3 du lemme 5.3.7, on a les égalités sur les dimensions des espaces vectoriels. Montrons à présent que $(W, g) = F_\beta^+(V, f)$ est indécomposable, pour cela supposons qu'il existe (W_1, g_1) et (W_2, g_2) des représentations telles que $(W, g) = (W_1, g_1) \oplus (W_2, g_2)$. On a alors, d'après le point 1 du lemme 5.3.7 que $(V, f) \cong F_\beta^-(W_1, g_1) \oplus F_\beta^-(W_2, g_2)$. Comme (V, f) est indécomposable, un des termes est la représentation nulle, supposons, sans perte de généralité, que $F_\beta^-(W_2, g_2) = 0$. D'après la forme de W et le point 5 du lemme 5.3.7, on sait que $p_V^\beta : (W, g) \rightarrow F_\beta^+ F_\beta^-(W, g)$ est un isomorphisme, or $p_V^\beta(W_2, g_2) \subset F_\beta^+ F_\beta^-(W_2, g_2) = 0$, d'où $(W_2, g_2) = 0$. Ainsi $F_\beta^+(V)$ est indécomposable.

Cela termine la démonstration dans le cas où β est un puits. Le cas d'une source est similaire. \square

Définition 5.3.9. On dit qu'une suite de sommets $\beta_1, \beta_2, \dots, \beta_k$ est une suite *(+)-accessible*, ou une *suite de puits*, suivant l'orientation Λ , si β_1 est un puits suivant l'orientation Λ , β_2 est un puits suivant l'orientation $\sigma_{\beta_1} \Lambda$, β_3 est un puits suivant l'orientation $\sigma_{\beta_2} \sigma_{\beta_1} \Lambda$, et ainsi de suite.

On dit qu'une suite de sommets $\alpha_1, \alpha_2, \dots, \alpha_k$ est une suite *(-)-accessible*, ou une *suite de sources*, suivant l'orientation Λ , si α_1 est une source suivant l'orientation Λ , α_2 est une source suivant l'orientation $\sigma_{\alpha_1} \Lambda$, α_3 est une source suivant l'orientation $\sigma_{\alpha_2} \sigma_{\alpha_1} \Lambda$, et ainsi de suite.

Corollaire 5.3.10

Soit (Γ, Λ) un carquois et β_1, \dots, β_k une suite de puits.

1. $\forall i \in \llbracket 1, k \rrbracket, F_{\beta_1}^- \dots F_{\beta_{i-1}}^-(L_{\beta_i})$ est soit la représentation nulle, soit une représentation indécomposable de $\mathcal{L}(\Gamma, \Lambda)$ (ici $L_{\beta_i} \in \mathcal{L}(\Gamma, \sigma_{\beta_{i-1}} \sigma_{\beta_{i-2}} \dots \sigma_{\beta_1} \Lambda)$).
2. Soit $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ une représentation indécomposable, et telle que $F_{\beta_k}^+ F_{\beta_{k-1}}^+ \dots F_{\beta_1}^+(V, f) = 0$, alors il existe $i \in \llbracket 1, k \rrbracket$ tel que $V \cong F_{\beta_1}^- F_{\beta_2}^- \dots F_{\beta_{i-1}}^-(L_{\beta_i})$.

Démonstration. Soit, comme dans l'énoncé, (Γ, Λ) un carquois et β_1, \dots, β_k une suite de puits. Soit $i \in \llbracket 1, n \rrbracket$, alors L_{β_i} est une représentation indécomposable. D'après le théorème 5.3.8, on sait donc que $F_{\beta_{i-1}}^-(L_{\beta_i})$ est une

représentation indécomposable. Par récurrence, il vient que $F_{\beta_1}^- \dots F_{\beta_{i-1}}^-(L_{\beta_i})$ est soit nulle, soit une représentation indécomposable de $\mathcal{L}(\Gamma, \Lambda)$. Prenons maintenant $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ une représentation indécomposable, et telle que $F_{\beta_i}^+ F_{\beta_{i-1}}^+ \dots F_{\beta_1}^+(V, f) = 0$. Soit $i \in \llbracket 1, k \rrbracket$ le plus petit entier tel que $F_{\beta_i}^+ \dots F_{\beta_1}^+(V, f) = 0$, alors d'après le théorème 5.3.8 on a $F_{\beta_{i-1}}^+ \dots F_{\beta_1}^+(V, f) \cong L_{\beta_i}$, et ainsi :

$$\begin{aligned} F_{\beta_{i-1}}^- F_{\beta_{i-1}}^+ F_{\beta_{i-2}}^+ \dots F_{\beta_1}^+(V, f) &\cong F_{\beta_{i-2}}^+ \dots F_{\beta_1}^+(V, f) \text{ (d'après le lemme 5.3.7)} \\ F_{\beta_{i-1}}^- F_{\beta_{i-1}}^+ F_{\beta_{i-2}}^+ \dots F_{\beta_1}^+(V, f) &\cong F_{\beta_{i-1}}^-(L_{\beta_i}) \end{aligned}$$

D'où $F_{\beta_{i-2}}^+ \dots F_{\beta_1}^+(V, f) \cong F_{\beta_{i-1}}^-(L_{\beta_i})$, et $F_{\beta_{i-1}}^-(L_{\beta_i})$. En itérant ce raisonnement il vient $V \cong F_{\beta_1}^- F_{\beta_2}^- \dots F_{\beta_{i-1}}^-(L_{\beta_i})$. \square

5.3.2 Foncteurs de Coxeter

Définition 5.3.11. On dit qu'une numérotation $(\gamma_1, \dots, \gamma_n)$ des sommets d'un carquois (Γ, Λ) est un *tri topologique* des sommets de Γ pour l'orientation Λ si pour toute arête l de Γ telle que $s(l) = \gamma_i$ et $t(l) = \gamma_j$ avec $i, j \in \llbracket 1, n \rrbracket$ alors $i > j$.

Proposition 5.3.12

Soit (Γ, Λ) un carquois ne contenant pas de cycle orienté.

1. (Γ, Λ) admet un tri topologique de ces sommets.
2. Un tri topologique des sommets de (Γ, Λ) est une suite de puits suivant l'orientation Λ .

Démonstration. Montrons d'abord qu'il existe toujours un puits dans un carquois (Γ, Λ) sans cycle orienté. En effet, dans le cas contraire, on peut construire par récurrence une suite infinie $(\gamma_n)_{n \in \mathbb{N}}$ de sommets de Γ telle que pour tout $n \in \mathbb{N}$, il existe une arête orientée l de (Γ, Λ) avec $s(l) = \gamma_n$ et $t(l) = \gamma_{n+1}$. Seulement le nombre de sommets de Γ est fini donc il existe deux entiers k et l distincts tels que $\gamma_k = \gamma_l$. Donc il existe un cycle orienté dans (Γ, Λ) .

Montrons maintenant le point 1 par récurrence sur le nombre de sommets de Γ . Si Γ n'admet qu'un unique sommet et aucun cycle pour l'orientation Λ , c'est-à-dire aucune boucle, alors il existe un tri topologique évident de Γ pour l'orientation Λ . Soit $n \in \mathbb{N}^*$, supposons maintenant que tout carquois (Γ', Λ') sans cycle orienté ayant n sommets admet un tri topologique. Soit (Γ, Λ) un carquois avec $n+1$ sommets sans cycle orienté. On sait qu'il existe s un puits dans Γ . Considérons (Γ', Λ') le carquois dont les sommets sont $\Gamma_0 \setminus \{s\}$ et les arêtes sont $\Gamma_1 \setminus \Gamma^s$ avec l'orientation Λ réduite. Par hypothèse de récurrence, il existe un tri topologique $\gamma'_1, \dots, \gamma'_n$ sur les sommets de Γ' pour l'orientation Λ' . En posant $\gamma_1 = s$ et pour tout $i \in \llbracket 2, n+1 \rrbracket$, $\gamma_i = \gamma'_{i-1}$, on obtient que $\gamma_1, \dots, \gamma_{n+1}$ est un tri topologique des sommets de Γ pour l'orientation Λ .

Montrons finalement le point 2, soit $\gamma_1, \dots, \gamma_n$ un tri topologique de Γ pour l'orientation Λ . On remarque que γ_1 est un puits dans (Γ, Λ) car toute arête de Γ^{γ_1} a pour destination γ_1 . Sinon une arête de source γ_1 aura une destination d'indice strictement inférieure à 1, c'est impossible. Soit $i \in \llbracket 2, n \rrbracket$ montrons que γ_i est un puits dans $(\Gamma, \sigma_{\gamma_{i-1}} \dots \sigma_{\gamma_1} \Lambda)$. Soit $l \in \Gamma^{\gamma_i}$ une arête reliant γ_{i-1} et γ_j un autre sommet. Si $i < j$ alors l'orientation de l dans $\sigma_{\gamma_{i-1}} \dots \sigma_{\gamma_1} \Lambda$ est la même que dans Λ , donc la destination de l est γ_i . Sinon l'orientation de l dans $\sigma_{\gamma_{i-1}} \dots \sigma_{\gamma_1} \Lambda$ est l'inverse de celle dans Λ , donc la destination de l est encore γ_i pour l'orientation $\sigma_{\gamma_{i-1}} \dots \sigma_{\gamma_1} \Lambda$. Finalement γ_i est toujours un puits dans $(\Gamma, \sigma_{\gamma_{i-1}} \dots \sigma_{\gamma_1} \Lambda)$. \square

Théorème 5.3.13

Soit Γ un graphe ne contenant pas de cycle, et soient Λ, Λ' deux orientations de ce graphe.

1. *Il existe $\beta_1, \beta_2, \dots, \beta_k$ une suite de puits suivant Λ telle que $\sigma_{\beta_k} \sigma_{\beta_{k-1}} \dots \sigma_{\beta_1} \Lambda = \Lambda'$;*
2. *Soit \mathcal{M} , (respectivement \mathcal{M}') l'ensemble des représentations indécomposables (à isomorphisme près) de $\mathcal{L}(\Gamma, \Lambda)$ (respectivement de $\mathcal{L}(\Gamma, \Lambda')$). $\widetilde{\mathcal{M}} \subset \mathcal{M}$ l'ensemble des classes des objets $F_{\beta_1}^- F_{\beta_2}^- \dots F_{\beta_{i-1}}^- (L_{\beta_i})$ pour $i \in \llbracket 1, k \rrbracket$ et $\widetilde{\mathcal{M}}' \subset \mathcal{M}'$ l'ensemble des classes d'objets $F_{\beta_k}^+ F_{\beta_{k-1}}^+ \dots F_{\beta_i}^+ (L_{\beta_i})$ pour $i \in \llbracket 1, k \rrbracket$, alors le foncteur $F_{\beta_k}^+ F_{\beta_{k-1}}^+ \dots F_{\beta_1}^+$ est une bijection entre $\mathcal{M} \setminus \widetilde{\mathcal{M}}$ et $\mathcal{M}' \setminus \widetilde{\mathcal{M}}'$.*

Démonstration. Montrons qu'il suffit de faire la preuve dans le cas où Λ et Λ' ne diffèrent que sur une arête. Pour ce faire, démontrons que l'on peut passer d'une arête à deux arêtes de différence. Le cas général se déduit alors par récurrence. Soit Λ et Λ' deux orientations différentes sur l et l' deux arêtes distinctes de Γ . Posons $\widetilde{\Lambda}$ différente de Λ sur l uniquement. Alors on sait qu'il existe par hypothèse une suite de puits β_1, \dots, β_k dans (Γ, Λ) telle que $\sigma_{\beta_1} \dots \sigma_{\beta_k} \Lambda = \widetilde{\Lambda}$. De même, il existe une suite de puits $\beta'_1, \dots, \beta'_{k'}$ dans $(\Gamma, \widetilde{\Lambda})$ telle que $\sigma_{\beta'_1} \dots \sigma_{\beta'_{k'}} \widetilde{\Lambda} = \Lambda'$. Finalement, on a que $\sigma_{\beta'_1} \dots \sigma_{\beta'_{k'}} \sigma_{\beta_1} \dots \sigma_{\beta_k} \Lambda = \Lambda'$.

Montrons maintenant l'existence d'une séquence β_1, \dots, β_k de puits dans (Γ, Λ) telle que $\sigma_{\beta_1} \dots \sigma_{\beta_k} \Lambda = \Lambda'$ lorsque Λ et Λ' sont deux orientations différents sur une seule arête l . Considérons le graphe $\Gamma \setminus l$, il est scindé en deux composantes connexes car Γ ne contient pas de cycle. Nommons Γ' la composante contenant le sommet $t(l)$, elle ne contient toujours pas de cycle, donc il existe un tri topologique de ces sommets $\gamma_1, \dots, \gamma_m$ pour l'orientation Λ . Montrons alors que $\sigma_{\gamma_1} \dots \sigma_{\gamma_m} \Lambda = \Lambda'$. En effet, l'orientation des arêtes de Γ' différentes de l est modifiée deux fois donc reste inchangée, alors que l'orientation de l est modifiée une unique fois. D'après la proposition 5.3.12, $\gamma_1, \dots, \gamma_m$ est une suite de puits pour orientation Λ , donc on a le résultat.

Montrons maintenant le point 2, soit β_1, \dots, β_k une suite de puits de Γ tels que $\sigma_{\beta_1} \dots \sigma_{\beta_k} \Lambda = \Lambda'$ et posons $\Psi^+ = F_{\beta_k}^+ F_{\beta_{k-1}}^+ \dots F_{\beta_1}^+$ et $\Psi^- = F_{\beta_1}^- F_{\beta_2}^- \dots F_{\beta_k}^-$. Les foncteurs Ψ^+ et Ψ^- sont respectivement définis de $\mathcal{L}(\Gamma, \Lambda)$ dans $\mathcal{L}(\Gamma, \Lambda')$ et $\mathcal{L}(\Gamma, \Lambda)$ dans $\mathcal{L}(\Gamma, \Lambda)$ d'après le point 1. Si V est une représentation de $\mathcal{L}(\Gamma, \Lambda)$ indécomposable et non isomorphe à $F_{\beta_1}^- F_{\beta_2}^- \dots F_{\beta_{i-1}}^- (L_{\beta_i})$ pour un certain $i \in \llbracket 1, k \rrbracket$, d'après le corollaire 5.3.10, on sait que $\Psi^+(V) \neq 0$ sinon il existe $i \in \llbracket 1, k \rrbracket$ tel que V soit isomorphe à $F_{\beta_1}^- F_{\beta_2}^- \dots F_{\beta_{i-1}}^- (L_{\beta_i})$. Donc en utilisant le théorème 5.3.8, on obtient que $\Psi^+(V)$ est indécomposable. Montrons que $\Psi^+(V)$ n'est pas isomorphe à $F_{\beta_k}^+ F_{\beta_{k-1}}^+ \dots F_{\beta_{j+1}}^+ (L_{\beta_j})$ pour un certain $j \in \llbracket 1, k \rrbracket$. Sinon on obtient que $F_{\beta_j}^- \dots F_{\beta_k}^- \Psi^+(V)$ isomorphe à $F_{\beta_{j-1}}^+ \dots F_{\beta_1}^+ (V)$ et isomorphe à 0. On utilise à nouveau le corollaire 5.3.10 pour en déduire la contradiction V isomorphe à $F_{\beta_1}^- \dots F_{\beta_{i-1}}^- (L_{\beta_i})$ pour un certain $i \in \llbracket 1, k \rrbracket$. Finalement, on a montré que $\Psi^+(V)$ est indécomposable et non isomorphe à $F_{\beta_k}^+ F_{\beta_{k-1}}^+ \dots F_{\beta_{j+1}}^+ (L_{\beta_j})$ pour un certain $j \in \llbracket 1, k \rrbracket$, c'est-à-dire que Ψ^+ est bien défini entre $\mathcal{M} \setminus \widetilde{\mathcal{M}}$ et $\mathcal{M}' \setminus \widetilde{\mathcal{M}'}$. De plus, on sait que $\Psi^+ \Psi^-(V)$ est isomorphe à V donc Ψ^+ est une bijection entre $\mathcal{M} \setminus \widetilde{\mathcal{M}}$ et $\mathcal{M}' \setminus \widetilde{\mathcal{M}'}$. \square

Définition 5.3.14. Soit (Γ, Λ) un carquois sans cycle orienté et $\gamma_1, \dots, \gamma_n$ un tri topologique sur les sommets de Γ pour l'orientation Λ . On appelle les *foncteurs de Coxeter* les foncteurs $\Phi^+ = F_{\gamma_n}^+ F_{\gamma_{n-1}}^+ \dots F_{\gamma_1}^+$ et $\Phi^- = F_{\gamma_1}^- F_{\gamma_2}^- \dots F_{\gamma_n}^-$.

Lemme 5.3.15

1. Les foncteurs Φ^+ et Φ^- sont définis de (Γ, Λ) dans lui-même.
2. Les foncteurs Φ^+ et Φ^- ne dépendent pas du choix du tri topologique.

Démonstration. Démontrons le point 1, soit $(\gamma_1, \dots, \gamma_n)$ un tri topologique sur les sommets de Γ pour l'orientation Λ , on remarque que l'orientation de chaque arête de Γ est modifiée deux fois par le foncteur Ψ^+ obtenu par ce tri topologique. Ainsi on a $\mathcal{L}(\Gamma, \sigma_{\gamma_1} \dots \sigma_{\gamma_n} \Lambda) = \mathcal{L}(\Gamma, \Lambda)$; c'est-à-dire que Ψ^+ envoie la catégorie $\mathcal{L}(\Gamma, \Lambda)$ dans elle-même.

Passons au point 2, et remarquons d'abord que si δ_1 et δ_2 sont deux puits de $(\Gamma, \widetilde{\Lambda})$ pour $\widetilde{\Lambda}$ une certaine orientation, non joints par une arête, alors $F_{\delta_1}^+$ et $F_{\delta_2}^+$ commutent. Soit $\gamma_1, \dots, \gamma_n$ et $\gamma'_1, \dots, \gamma'_n$ deux tris topologiques sur les sommets de Γ pour l'orientation Λ , et soit $m \in \llbracket 1, n \rrbracket$ tel que $\gamma_1 = \gamma'_m$. Montrons que $\gamma_1 = \gamma'_m$ n'est joint à aucun des sommets $\gamma'_1, \dots, \gamma'_{m-1}$. En effet, si $i \in \llbracket 1, m-1 \rrbracket$, il n'existe pas d'arête de γ'_i vers γ'_m car $i < m$ et si $j \in \llbracket 2, n \rrbracket$ tel que $\gamma_j = \gamma'_i$, alors il n'existe pas d'arête de γ_j vers γ_1 car $1 < j$. Avec la remarque ci-dessus, on obtient $F_{\gamma'_m}$ et $F_{\gamma'_i}$ commutent pour $i \in \llbracket 1, m-1 \rrbracket$ et $F_{\gamma'_m}^+ F_{\gamma'_{m-1}}^+ \dots F_{\gamma'_1}^+ = F_{\gamma'_{m-1}}^+ \dots F_{\gamma'_1}^+ F_{\gamma'_m}^+ = F_{\gamma'_{m-1}}^+ \dots F_{\gamma'_1}^+ F_{\gamma_1}^+$. En utilisant successivement, le même argument pour γ_2 puis γ_3 et ainsi de

suite. On obtient :

$$F_{\gamma'_n} \dots F_{\gamma'_1} = F_{\gamma_n} \dots F_{\gamma_1}.$$

□

Définition 5.3.16. Soit (Γ, Λ) un carquois sans cycle orienté.

1. On dit qu'un objet $V \in \mathcal{L}(\Gamma, \Lambda)$ est *(+)-irrégulier* (respectivement *(-)-irrégulier*) si $(\Phi^+)^k(V) = 0$ (respectivement $(\Phi^-)^k(V) = 0$) pour un certain k entier.
2. On dit qu'un objet $V \in \mathcal{L}(\Gamma, \Lambda)$ est *régulier* si V est isomorphe à $(\Phi^-)^k(\Phi^+)^k(V)$ et à $(\Phi^+)^k(\Phi^-)^k(V)$ pour tout entier k .

Théorème 5.3.17

Soit (Γ, Λ) un carquois sans cycle orienté.

1. *Chaque $V \in \mathcal{L}(\Gamma, \Lambda)$ indécomposable est soit régulier soit (+) ou (-)-irrégulier.*
2. *Soit $\gamma_1, \dots, \gamma_n$ un tri topologique des sommets de Γ pour l'orientation Λ et posons $V_i = F_{\gamma_1}^- F_{\gamma_2}^- \dots F_{\gamma_{i-1}}^- (L_{\gamma_i}) \in \mathcal{L}(\Gamma, \Lambda)$ et $\widehat{V}_i = F_{\gamma_n}^+ F_{\gamma_{n-1}}^+ \dots F_{\gamma_{i+1}}^+ (L_{\gamma_i}) \in \mathcal{L}(\Gamma, \Lambda)$ pour $i \in \llbracket 1, n \rrbracket$. Alors $\Phi^+(V_i) = 0$ et chaque objet indécomposable $V \in \mathcal{L}(\Gamma, \Lambda)$ tel que $\Phi^+(V) = 0$ est isomorphe à l'un des V_i . De même, $\Phi^-(\widehat{V}_i) = 0$ et si V est indécomposable avec $\Phi^-(V) = 0$, alors V est isomorphe à l'un des \widehat{V}_i .*
3. *Chaque objet V (+)-irrégulier (respectivement (-)-irrégulier) indécomposable a la forme de $(\Phi^-)^k(V_i)$ (respectivement $(\Phi^+)^k(\widehat{V}_i)$) pour i et k entiers.*

Démonstration. Démontrons le point 1, on remarque premièrement que $V \in \mathcal{L}(\Gamma, \Lambda)$ irréductible ne peut être régulier et (+)- ou (-)-irrégulier. En effet, s'il existe un certain k tel que $(\Phi^+)^k(V) = 0$ ou $(\Phi^-)^k(V) = 0$, alors on ne peut pas avoir V , $(\Phi^-)^k(\Phi^+)^k(V)$ et $(\Phi^+)^k(\Phi^-)^k(V)$ isomorphes car l'un des deux derniers est nul et V est indécomposable. Supposons V est non (-)-irrégulier et non (+)-irrégulier et V indécomposable, montrons alors que V est régulier. Comme $(\Phi^+)^k(V)$ est non nul pour tout k entier, on sait d'après le corollaire 5.3.10 que $(\Phi^+)^k(V)$ est indécomposable pour tout k entier. Et ainsi, en composant à gauche k fois par Φ^- , on obtient par le théorème 5.3.8 que $(\Phi^-)^k(\Phi^+)^k(V)$ est isomorphe à V . Comme V n'est pas (-)-irrégulier et par le même raisonnement, on en déduit que $(\Phi^+)^k(\Phi^-)^k(V)$ est isomorphe à V , d'où le résultat espéré.

Démontrons maintenant le point 2, les V_i et les \widehat{V}_i sont soit indécomposables, soit nuls d'après le corollaire 5.3.10. D'où, $\Phi^+(V_i) = 0$ et $\Phi^-(\widehat{V}_i) = 0$, si V_i et \widehat{V}_i sont nuls. S'ils sont indécomposables, $\Phi^+(V_i)$ et $\Phi^-(\widehat{V}_i)$ sont respectivement isomorphes à $F_{\gamma_n}^+ \dots F_{\gamma_i}^+(L_{\gamma_i})$ et $F_{\gamma_1}^- \dots F_{\gamma_i}^-(L_{\gamma_i})$, tous deux isomorphes à 0. Par une utilisation directe du deuxième point du corollaire

5.3.10, si V est indécomposable, $\Phi^+(V) = 0$ implique que V est isomorphe à l'un des V_i et $\Phi^-(V) = 0$ implique que V est isomorphe à l'un des \widehat{V}_i .

Démontrons le point 3, soit V un objet indécomposable et k l'entier minimal tel que $(\Phi^+)^k(V) = 0$, c'est-à-dire tel que $(\Phi^+)^{k-1}(V) \neq 0$. On pose $W = (\Phi^+)^{k-1}(V)$, W est indécomposable et $\Phi^+(W) = 0$. D'après le point précédent, W est isomorphe à un V_i et en plus, V est isomorphe à $(\Phi^-)^{k-1}(W)$, d'où le résultat pour (+)-irrégulier. Et on procède de même pour obtenir le résultat pour les (-)-irréguliers. \square

5.4 Graphes, groupe de Weyl et transformations de Coxeter

Dans cette section nous définirons un *système de racine* utile, ainsi que des résultats qui nous permettront dans la dernière partie de prouver le théorème de Gabriel. Il n'est pas nécessaire de connaître la théorie sur les systèmes de racine pour comprendre cette partie, néanmoins les résultats obtenus ici sont très généraux et sont traités dans [25].

5.4.1 Graphes et groupe de Weyl

Définition 5.4.1. Soit Γ un graphe ne possédant pas de boucles.

1. On note \mathcal{E}_Γ le \mathbb{Q} -espace vectoriel \mathbb{Q}^{Γ_0} composé des familles de rationnels $x = (x_\gamma)_{\gamma \in \Gamma_0}$. On associe à chaque sommet $\gamma \in \Gamma_0$, le vecteur $\bar{\gamma} \in \mathcal{E}_\Gamma$ tel que $(\bar{\gamma})_\gamma = 1$ et $\forall \delta \neq \gamma, (\bar{\gamma})_\delta = 0$. On dit qu'un vecteur x est *entier* si $\forall \gamma \in \Gamma_0, x_\gamma \in \mathbb{Z}$. On dit qu'un vecteur x est *positif*, et l'on note $x > 0$, si $x \neq 0$ et si $\forall \gamma \in \Gamma_0, x_\gamma \geq 0$.
2. On pose :

$$B : \mathcal{E}_\Gamma \longrightarrow \mathbb{Q}$$

$$x \longmapsto \sum_{\gamma \in \Gamma_0} x_\gamma^2 - \sum_{l \in \Gamma_1} x_{e_1(l)} x_{e_2(l)}$$

où $e_1(l)$ et $e_2(l)$ sont les extrémités de l .

3. Pour chaque sommet $\gamma \in \Gamma_0$, on pose l'application σ_γ de \mathcal{E}_Γ à valeurs dans \mathcal{E}_Γ définie par $\forall \delta \neq \gamma, (\sigma_\gamma x)_\delta = x_\delta$ et $(\sigma_\gamma x)_\gamma = -x_\gamma + \sum_{l \in \Gamma_1} x_{e(l)}$, où $e(l)$ est l'extrémité de l différente de γ . On note W l'ensemble engendré en composant les applications σ_γ .

Proposition 5.4.2

L'application B est une forme quadratique, dont on notera \langle, \rangle la forme bilinéaire symétrique associée. Pour tout sommet $\gamma \in \Gamma_0$, l'application σ_γ est linéaire.

Démonstration. Soient $x, y \in \mathcal{E}_\Gamma, \lambda \in \mathbb{Q}$, on remarque tout d'abord que $B(\lambda x) = \sum_{\gamma \in \Gamma_0} (\lambda x_\gamma)^2 - \sum_{l \in \Gamma_1} \lambda x_{e_1(l)} \lambda x_{e_2(l)} = \lambda^2 B(x)$. On pose $\langle x, y \rangle = \frac{1}{2}(B(x+y) - B(x) - B(y))$, on a donc bien $\langle x, x \rangle = \frac{1}{2}(B(2x) - 2B(x)) = B(x)$, et il vient :

$$2 \langle x, y \rangle = \sum_{\gamma \in \Gamma_0} (x+y)_\gamma^2 - \sum_{l \in \Gamma_1} (x+y)_{e_1(l)} (x+y)_{e_2(l)}$$

$$- \sum_{\gamma \in \Gamma_0} x_\gamma^2 + \sum_{l \in \Gamma_1} x_{e_1(l)} x_{e_2(l)} - \sum_{\gamma \in \Gamma_0} y_\gamma^2 + \sum_{l \in \Gamma_1} y_{e_1(l)} y_{e_2(l)}$$

$$\text{d'où } \langle x, y \rangle = \sum_{\gamma \in \Gamma_0} x_\gamma y_\gamma - \frac{1}{2} \left(\sum_{l \in \Gamma_1} x_{e_1(l)} y_{e_2(l)} + \sum_{l \in \Gamma_1} y_{e_1(l)} x_{e_2(l)} \right).$$

De cette dernière égalité on tire immédiatement :

- $\forall x, y \in \mathcal{E}_\Gamma, \langle x, y \rangle = \langle y, x \rangle$
- $\forall x, y, z \in \mathcal{E}_\Gamma, \langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
- $\forall x, y \in \mathcal{E}_\Gamma, \lambda \in \mathbb{Q}, \langle \lambda x, y \rangle = \lambda \langle x, y \rangle$

Ainsi, l'application \langle, \rangle est bilinéaire et symétrique, l'application B est donc une forme quadratique. Prouvons maintenant la seconde partie de la proposition, à cet effet, soient $\gamma \in \Gamma_0, x, y \in \mathcal{E}_\Gamma$ et $a, b \in \mathbb{Q}$. On a $\forall \delta \neq \gamma, (\sigma_\gamma(ax + by))_\delta = ax_\delta + by_\delta = a(\sigma_\gamma x)_\delta + b(\sigma_\gamma y)_\delta$ et $(\sigma_\gamma(ax + by))_\gamma = -(ax_\gamma + by_\gamma) + \sum_{l \in \Gamma^\gamma} (ax_{e(l)} + by_{e(l)}) = a(\sigma_\gamma x)_\gamma + b(\sigma_\gamma y)_\gamma$. On a donc bien $\sigma_\gamma(ax + by) = a\sigma_\gamma x + b\sigma_\gamma y$, or le sommet $\gamma \in \Gamma_0$ est arbitraire, cela conclut donc la démonstration. \square

Lemme 5.4.3

Soit Γ un graphe sans boucle, $\gamma, \delta \in \Gamma_0, x \in \mathcal{E}_\Gamma$:

1. on a $\langle \bar{\gamma}, \bar{\gamma} \rangle = 1$, et si $\gamma \neq \delta$, alors $-2\langle \bar{\gamma}, \bar{\delta} \rangle$ est le nombre d'arêtes liant γ et δ ;
2. on a $\sigma_\gamma(x) = x - 2\langle \bar{\gamma}, x \rangle \bar{\gamma}$ et $\sigma_\gamma^2 = \text{Id}$; en particulier W est un groupe ;
3. le groupe W préserve les éléments entiers de \mathcal{E}_Γ et la forme quadratique B ;
4. si la forme B est définie positive, alors le groupe W est fini.

Démonstration. Commençons par prouver le point 1. Soient $\gamma, \delta \in \Gamma_0$, avec $\gamma \neq \delta$, alors $\langle \bar{\gamma}, \bar{\gamma} \rangle = B(\bar{\gamma}) = 1$, en effet la somme sur les arêtes vaut 0, et dans la somme sur les sommets, seul le terme en γ est non nul et il vaut 1. Calculons maintenant $\langle \bar{\gamma}, \bar{\delta} \rangle$, comme $\gamma \neq \delta$, on a $\forall \varepsilon \in \Gamma_0, \bar{\gamma}_\varepsilon \bar{\delta}_\varepsilon = 0$. Ainsi $-2\langle \bar{\gamma}, \bar{\delta} \rangle = \sum_{l \in \Gamma_1} (\bar{\gamma}_{e_1(l)} \bar{\delta}_{e_2(l)} + \bar{\gamma}_{e_2(l)} \bar{\delta}_{e_1(l)})$. Notons $\Gamma^\gamma \cap \Gamma^\delta = \{l_1, \dots, l_n\}$ les arêtes liant les sommets γ et δ . Alors $\forall l \notin \{l_1, \dots, l_n\}, \bar{\gamma}_{e_1(l)} = \bar{\gamma}_{e_2(l)} = 0$ ou $\bar{\delta}_{e_1(l)} = \bar{\delta}_{e_2(l)} = 0$, d'où $\bar{\gamma}_{e_1(l)} \bar{\delta}_{e_2(l)} + \bar{\gamma}_{e_2(l)} \bar{\delta}_{e_1(l)} = 0$. Et, $\forall i \in \llbracket 1, n \rrbracket$, soit $\bar{\gamma}_{e_1(l_i)} \bar{\delta}_{e_2(l_i)} = 1$ et $\bar{\gamma}_{e_2(l_i)} \bar{\delta}_{e_1(l_i)} = 0$, soit $\bar{\gamma}_{e_1(l_i)} \bar{\delta}_{e_2(l_i)} = 0$ et $\bar{\gamma}_{e_2(l_i)} \bar{\delta}_{e_1(l_i)} = 1$. D'où :

$$\begin{aligned}
-2\langle \bar{\gamma}, \bar{\delta} \rangle &= \sum_{l \in \Gamma_1} (\bar{\gamma}_{e_1(l)} \bar{\delta}_{e_2(l)} + \bar{\gamma}_{e_2(l)} \bar{\delta}_{e_1(l)}) \\
&= \sum_{i=1}^n (\bar{\gamma}_{e_1(l_i)} \bar{\delta}_{e_2(l_i)} + \bar{\gamma}_{e_2(l_i)} \bar{\delta}_{e_1(l_i)}) \\
&= \sum_{i=1}^n 1 \\
&= n,
\end{aligned}$$

ce qui termine la démonstration du premier point, passons au deuxième. Soient $x \in \mathcal{E}_\Gamma, \gamma \in \Gamma_0$, on a $x = \sum_{\delta \in \Gamma_0} x_\delta \bar{\delta}$. Ainsi on a $(\sigma_\gamma x)_\delta = x_\delta =$

$(x - 2 \langle \bar{\gamma}, x \rangle \bar{\gamma})_\delta$, dès que $\gamma \neq \delta$ car $\bar{\gamma}_\delta = 0$, et :

$$\begin{aligned}
(x - 2 \langle \bar{\gamma}, x \rangle \bar{\gamma})_\gamma &= x_\gamma - 2 \langle \bar{\gamma}, x \rangle \\
&= x_\gamma - 2 \left\langle \bar{\gamma}, \sum_{\delta \in \Gamma_0} x_\delta \bar{\delta} \right\rangle \\
&= x_\gamma - 2 \langle \bar{\gamma}, \bar{\gamma} \rangle x_\gamma + \sum_{\delta \in \Gamma_0, \delta \neq \gamma} -2 \langle \bar{\gamma}, \bar{\delta} \rangle x_\delta \\
&= -x_\gamma + \sum_{\delta \in \Gamma, \delta \neq \gamma} \text{Card}(\Gamma^\gamma \cap \Gamma^\delta) x_\delta \\
&= -x_\gamma + \sum_{\delta \in \mathcal{V}_\gamma} \sum_{l \in \Gamma^\delta \cap \Gamma^\gamma} x_{e(l)} \\
&= -x_\gamma + \sum_{l \in \Gamma^\gamma} x_{e(l)} \\
&= (\sigma_\gamma x)_\gamma
\end{aligned}$$

où \mathcal{V}_γ est l'ensemble des sommets voisins de γ . Ainsi on a $\sigma_\gamma x = x - 2 \langle \bar{\gamma}, x \rangle \bar{\gamma}$.
Calculons maintenant σ_γ^2 :

$$\begin{aligned}
\sigma_\gamma(\sigma_\gamma(x)) &= \sigma_\gamma(x) - 2 \langle \bar{\gamma}, \sigma_\gamma(x) \rangle \bar{\gamma} \\
&= \sigma_\gamma(x) - 2 \langle \bar{\gamma}, x - 2 \langle \bar{\gamma}, x \rangle \bar{\gamma} \rangle \bar{\gamma} \\
&= x - 2 \langle \bar{\gamma}, x \rangle \bar{\gamma} - 2 \langle \bar{\gamma}, x \rangle \bar{\gamma} + 4 \langle \bar{\gamma}, x \rangle \bar{\gamma} \\
&= x
\end{aligned}$$

d'où $\sigma_\gamma^2 = \text{Id}$. Ainsi $\text{Id} \in W$, la composition est associative, et tous les éléments de W sont inversibles. En effet soit $\omega \in W$, alors ω s'écrit $\omega = \prod_{i=1}^n \sigma_{\gamma_i}$ avec $\gamma_i \in \Gamma_0$ et ainsi $\omega' = \prod_{i=1}^n \sigma_{\gamma_{n+1-i}}$ est l'inverse de ω , les termes s'annulant deux à deux. On a donc que W est un groupe et ceci conclut la preuve du point 2. Pour prouver le point 3, soit $x \in \mathcal{E}_\Gamma$ un vecteur entier et $\gamma \in \Gamma_0$ un sommet de Γ . On a $(\sigma_\gamma(x))_\delta = x_\delta \in \mathbb{Z}$ dès que $\delta \neq \gamma$ et $(\sigma_\gamma(x))_\gamma = -x_\gamma + \sum_{l \in \Gamma^\gamma} x_{e(l)} \in \mathbb{Z}$ car $\forall \delta \in \Gamma_0, x_\delta \in \mathbb{Z}$. Comme les éléments de W sont des compositions des applications σ_γ , on a immédiatement que W préserve les éléments entiers de \mathcal{E}_Γ . Voyons maintenant que W préserve B :

$$\begin{aligned}
B(\sigma_\gamma(x)) &= B(x - 2 \langle \bar{\gamma}, x \rangle \bar{\gamma}) \\
&= B(x) + 4 \langle \bar{\gamma}, x \rangle^2 B(\bar{\gamma}) - 2 \langle x, 2 \langle \bar{\gamma}, x \rangle \bar{\gamma} \rangle \\
&= B(x).
\end{aligned}$$

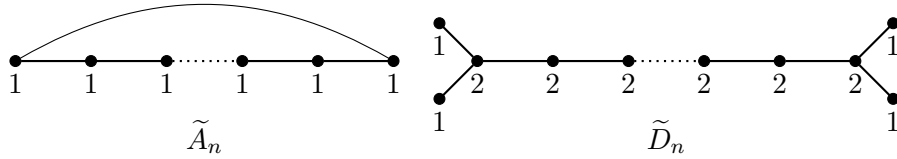
Or W est engendré par les applications σ_γ , on a donc que W préserve la forme quadratique B . Enfin, prouvons le point 4. À cet effet, on pose $B_1 = \{x \in \mathcal{E}_\Gamma : x \text{ est entier et } B(x) = 1\}$. Supposons un instant que B_1 est un ensemble fini. D'après le point 3, W est un groupe qui préserve B_1 , c'est donc un sous-groupe du groupe des permutations des éléments de B_1 . W est ainsi un sous-groupe d'un groupe fini donc W est un groupe fini. Prouvons

maintenant que B_1 est fini. On note $[B]$ la matrice représentant la forme quadratique B . Cette dernière est symétrique, ainsi, d'après le théorème spectral, il existe une base orthonormée \mathcal{B} dans laquelle $[B]$ est diagonale, notons $\lambda_1, \dots, \lambda_n$ ses valeurs propres, elles sont strictement positives car B est définie positive. Soit $v \in B_1$, on note v_1, \dots, v_n ses coordonnées dans \mathcal{B} . On a donc $\lambda_1 v_1^2 + \dots + \lambda_n v_n^2 = 1$, d'où $\forall i \in \llbracket 1, n \rrbracket, |v_i| \leq \frac{1}{\sqrt{\lambda_i}}$. Ainsi les coordonnées des éléments de B_1 sont bornées dans la base \mathcal{B} , elle sont donc également bornées dans la base canonique de \mathcal{E}_Γ , utilisée pour définir les éléments entiers. Ainsi B_1 est une sous-partie bornée de \mathbb{Z}^n donc est finie. Or on a prouvé précédemment que cela implique que W est fini, la preuve est donc complète. \square

Proposition 5.4.4

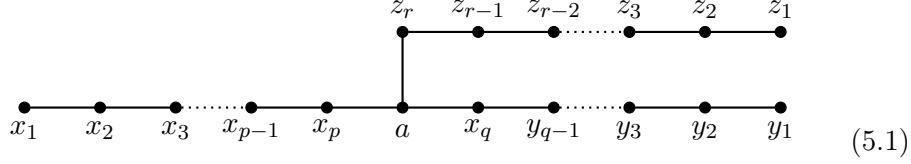
Soit Γ un graphe connexe, alors la forme quadratique B est définie positive si et seulement si Γ est de la forme A_n, D_n, E_6, E_7 ou E_8 .

Démonstration. Soit Γ un graphe. On va d'abord prouver que si B est définie positive, Γ est nécessairement de la forme A_n, D_n, E_6, E_7 ou E_8 . Tout d'abord, remarquons que Γ ne peut contenir de sous-graphe d'une des formes suivantes : \tilde{A}_n ou \tilde{D}_n , où n représente le nombre de sommets. \tilde{A}_n est défini pour $n \geq 2$ et \tilde{D}_n est défini pour $n \geq 5$.



Pour cela, supposons par l'absurde que Γ contient un sous-graphe de la forme \tilde{A}_n ou \tilde{D}_n . Si Γ contient un sous-graphe de la forme \tilde{A}_n , on prend $x \in \mathcal{E}_\Gamma$ tel que $x_\gamma = 0$ lorsque γ n'est pas un élément du sous-graphe \tilde{A}_n , et $x_\gamma = 1$ lorsque γ appartient à \tilde{A}_n . Alors, comme \tilde{A}_n contient n sommets et n arêtes, on a $B(x) = n - n = 0$ alors que x n'est pas nul. Donc B n'est pas définie positive, ce qui est une contradiction. Supposons maintenant que Γ contient un sous-graphe de la forme \tilde{D}_n , on prend dans ce cas $x \in \mathcal{E}_\Gamma$ tel que $x_\gamma = 0$ dès que $\gamma \notin \tilde{D}_n$, $x_\gamma = 1$ si $\gamma \in \tilde{D}_n$ possède un seul voisin dans \tilde{D}_n , ou de manière équivalente si γ est une des quatre extrémités de \tilde{D}_n , et $x_\gamma = 2$ si $\gamma \in \tilde{D}_n$ possède au moins deux voisins dans \tilde{D}_n , c'est-à-dire si γ est un point non extrémal de \tilde{D}_n . Comme \tilde{D}_n possède $n - 4$ sommets non extrémaux, de valeurs 2 dans x , 4 extrémités, de valeurs 1 dans x , la somme sur les sommets dans la définition de $B(x)$ vaut $4(n - 4) + 4 \times 1 = 4(n - 3)$. D'autre part \tilde{D}_n contient $n - 5$ arêtes centrales (c'est-à-dire dont aucune des extrémités de l'arête n'est un point extrémal), et 4 arêtes extrémales (ou non centrales), ainsi la somme sur les arêtes dans la définition de $B(x)$ vaut $4 \times 2 + 4(n - 5) = 4(n - 3)$. D'où $B(x) = 4(n - 3) - 4(n - 3) = 0$ alors que x est non nul, donc B n'est pas définie positive, ce qui est une contradiction. Ainsi, si B est définie

positive, Γ est nécessairement de la forme :



où $p, q, r \in \mathbb{N}$ sont des entiers positifs. Posons maintenant, pour un certain entier $p \in \mathbb{N}$, l'application C_p en $p + 1$ variables :

$$C_p(x_1, \dots, x_{p+1}) = -x_1x_2 - x_2x_3 - \dots - x_px_{p+1} + x_1^2 + \dots + x_p^2 + \frac{p}{2(p+1)}x_{p+1}^2,$$

et prouvons que C_p est une forme quadratique positive, de noyau de dimension 1 et qui vérifie que si $x \neq 0$ est tel que $C_p(x) = 0$ alors x n'a aucune coordonnée nulle. Pour prouver tout cela, nous allons démontrer que

$$C_p(x) = \sum_{i=1}^p \frac{i}{2(i+1)} \left(x_{i+1} - \frac{i+1}{i}x_i\right)^2. \quad (5.2)$$

Faisons le calcul :

$$\begin{aligned} \sum_{i=1}^p \frac{i}{2(i+1)} \left(x_{i+1} - \frac{i+1}{i}x_i\right)^2 &= \sum_{i=1}^p \frac{i}{2(i+1)} \left(x_{i+1}^2 - 2\frac{i+1}{i}x_ix_{i+1} + \left(\frac{i+1}{i}x_i\right)^2\right) \\ &= -\sum_{i=1}^p x_ix_{i+1} + \sum_{i=1}^p \frac{i}{2(i+1)}x_{i+1}^2 + \sum_{i=1}^p \frac{i+1}{2i}x_i^2 \\ &= -\sum_{i=1}^p x_ix_{i+1} + \frac{p}{2(p+1)}x_{p+1}^2 + x_1^2 + \sum_{i=2}^p \frac{(i-1) + (i+1)}{2i}x_i^2 \\ &= C_p(x), \end{aligned}$$

ainsi on a bien l'égalité annoncée plus haut. Grâce à cette nouvelle écriture (5.2), on a immédiatement la positivité de C_p . Soit $x \in \mathcal{E}_\Gamma$ tel que $C_p(x) = 0$, d'après (5.2), on a donc $\forall i \in \llbracket 1, n \rrbracket$, $x_{i+1} = \frac{i+1}{i}x_i$, d'où $\forall i \in \llbracket 1, n \rrbracket$, $x_i = ix_1$ par récurrence. Ainsi $\text{Ker } C_p \subset \text{Vect} \{(1, 2, \dots, n)\}$, et la réciproque est immédiate. Donc $\text{Ker } C_p = \text{Vect} \{(1, 2, \dots, n)\}$, et ainsi le noyau de C_p est de dimension 1. De plus, comme $(1, 2, \dots, n)$ ne possède pas de coordonnée nulle, $x \neq 0$ et $x \in \text{Vect} \{(1, 2, \dots, n)\}$ implique $\forall i \in \llbracket 1, n \rrbracket$, $x_i \neq 0$. Donc si $x \neq 0$ et $C_p(x) = 0$, alors toutes les coordonnées de x sont non nulles. Enfin prouvons que C_p est une forme quadratique, pour cela posons :

$$\langle x, y \rangle_p = \sum_{i=1}^p \frac{i}{2(i+1)} \left(x_{i+1} - \frac{i+1}{i}x_i\right) \left(y_{i+1} - \frac{i+1}{i}y_i\right),$$

on a ainsi :

1. $\forall x \in \mathcal{E}_\Gamma, \langle x, x \rangle_p = C_p(x)$
2. $\forall x, y \in \mathcal{E}_\Gamma, \langle x, y \rangle = \langle y, x \rangle$
3. $\forall x, y, z \in \mathcal{E}_\Gamma, \langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
4. $\forall x, y \in \mathcal{E}_\Gamma, \lambda \in \mathbb{Q}, \langle \lambda x, y \rangle = \lambda \langle x, y \rangle,$

ce qui prouve que C_p est bien une forme quadratique. Et on a ainsi prouv   toutes les propri  t  s annonc  es    propos de C_p . On place maintenant les nombres $x_1, \dots, x_p, y_1, \dots, y_q, z_1, \dots, z_r$ et a sur le graphe Γ comme repr  sent   sur le sch  ma (5.1), formant ainsi un vecteur $t \in \mathcal{E}_\Gamma$. On a alors :

$$\begin{aligned}
B(t) &= \sum_{\gamma \in \Gamma_0} t_\gamma^2 - \sum_{l \in \Gamma_1} t_{e_1(l)} t_{e_2(l)} \\
&= x_1^2 + \dots + x_p^2 + y_1^2 + \dots + y_q^2 + \dots + z_1^2 + \dots + z_r^2 + a^2 \\
&\quad - x_1 x_2 - \dots - x_p a - y_1 y_2 - \dots - y_q a - \dots - z_1 z_2 - \dots - z_r a \\
&= C_p(x_1, \dots, x_p, a) + C_q(y_1, \dots, y_q, a) + C_r(z_1, \dots, z_r, a) \\
&\quad + \left(1 - \frac{p}{2(p+1)} - \frac{q}{2(q+1)} - \frac{r}{2(r+1)} \right) a^2.
\end{aligned}$$

Gr  ce    cette derni  re formule, on prouve que B est d  finie postivie si et seulement si $\frac{p}{2(p+1)} + \frac{q}{2(q+1)} + \frac{r}{2(r+1)} < 1$. Supposons que $\frac{p}{2(p+1)} + \frac{q}{2(q+1)} + \frac{r}{2(r+1)} \geq 1$, alors on prend $a > 0$, $x_i = \frac{i}{p+1}a$, $y_i = \frac{i}{q+1}a$ et $z_i = \frac{i}{r+1}a$, ce qui nous donne que $(x_1, \dots, x_p, a) \in \text{Ker } C_p$, $(y_1, \dots, y_q, a) \in \text{Ker } C_q$, $(z_1, \dots, z_r, a) \in \text{Ker } C_r$, d'o   $B(t) \leq 0$ et ainsi B n'est pas d  finie positive. Supposons maintenant que $\frac{p}{2(p+1)} + \frac{q}{2(q+1)} + \frac{r}{2(r+1)} < 1$, et qu'on a $B(t) = 0$. Comme $B(t)$ est une somme de termes positifs, ils doivent tous   tre nulles, on obtient ainsi $a = 0$, puis, d'apr  s les propri  t  s des noyaux de C_p, C_q, C_r , si l'un des termes est non nul, ils le sont tous, or $a = 0$, on a donc $x_1 = \dots = x_p = y_1 = \dots = y_q = z_1 = \dots = z_r = 0$. On vient de voir que $B(t) = 0$ implique $t = 0$ donc B est d  finie positive. Ainsi B est d  finie postivie si et seulement si $\frac{p}{2(p+1)} + \frac{q}{2(q+1)} + \frac{r}{2(r+1)} < 1$. Cette in  galit     quivaut en fait    $\frac{1}{p+1} + \frac{1}{q+1} + \frac{1}{r+1} > 1$, en effet les in  galit  s suivantes sont   quivalentes :

$$\begin{aligned}
\frac{p}{2(p+1)} + \frac{q}{2(q+1)} + \frac{r}{2(r+1)} &< 1 \\
\frac{\frac{2(p+1)}{2} - 1}{2(p+1)} + \frac{\frac{2(q+1)}{2} - 1}{2(q+1)} + \frac{\frac{2(r+1)}{2} - 1}{2(r+1)} &< 1 \\
\frac{3}{2} - 1 &< \frac{1}{2(p+1)} + \frac{1}{2(q+1)} + \frac{1}{2(r+1)} \\
1 &< \frac{1}{p+1} + \frac{1}{q+1} + \frac{1}{r+1}
\end{aligned}$$

On peut supposer, quitte      changer les r  les de p, q, r , que $p \leq q \leq r$. On pose $A = \frac{1}{p+1} + \frac{1}{q+1} + \frac{1}{r+1}$ et on   tudie si $A > 1$ (c'est-  -dire si B est d  finie positive) :

1. $p = 0$, q et r sont arbitraires, alors $A > 1$ et on est dans le cas A_n ;

2. $p = 1, q = 1$, et r est arbitraire, alors $A > 1$ et on est dans le cas D_n ;
3. $p = 1, q = 2$ et $r = 2, 3, 4$, alors $A > 1$ et on est dans les cas E_6, E_7, E_8 ;
4. $p = 1, q = 2$, et $r \geq 5$, alors $A \leq 1$;
5. $p = 1, q \geq 3$ et $r \geq 3$ alors $A \leq 1$;
6. $p \geq 2, q \geq 2$ et $r \geq 2$ alors $A \leq 1$;

On a ainsi énuméré tous les cas possibles pour p, q, r et les seuls graphes pour lesquels $A > 1$ sont les graphes de type A_n, D_n, E_6, E_7 et E_8 . Donc la forme quadratique B est définie positives si et seulement si Γ est de type A_n, D_n, E_6, E_7 ou E_8 . \square

5.4.2 Racines

Définition 5.4.5. Soit Γ un graphe. Un vecteur $x \in \mathcal{E}_\Gamma$ est appelé *racine* s'il existe $\gamma \in \Gamma_0, w \in W$, tel que $x = w(\bar{\gamma})$. Les vecteurs $\bar{\gamma}$, où $\gamma \in \Gamma_0$, sont appelés *racines simples*. Une racine x est dite *positive* si $x > 0$.

Lemme 5.4.6

Soit Γ un graphe et $x \in \mathcal{E}_\Gamma$ un vecteur.

1. Si x est une racine, x est entier et $B(x) = 1$;
2. si x est une racine, $-x$ est une racine ;
3. si x est une racine, soit $x > 0$, soit $-x > 0$.

Démonstration. Soit $x \in \mathcal{E}_\Gamma$ une racine, il existe alors $\gamma \in \Gamma_0, w \in W$, tel que $x = w(\bar{\gamma})$. $B(\bar{\gamma}) = 1$ et $\bar{\gamma}$ est entier, ainsi d'après le point 3 du lemme 5.4.3, $B(w(\bar{\gamma})) = B(x) = 1$ et $w(\bar{\gamma}) = x$ est entier, ce qui prouve le point 1. Le point 2 suit du fait que $\forall \gamma \in \Gamma_0, \sigma_\gamma(\bar{\gamma}) = -\bar{\gamma}$, ainsi $w(\sigma_\gamma(\bar{\gamma})) = -w(\bar{\gamma}) = -x$, et comme $w\sigma_\gamma \in W$, $-x$ est une racine. On peut écrire la racine x sous la forme $\sigma_{\gamma_1}\sigma_{\gamma_2}\dots\sigma_{\gamma_k}(\bar{\gamma})$, il est donc suffisant, pour prouver le point 3, de vérifier que si $y \in \mathcal{E}_\Gamma$ est une racine positive, et si $\gamma \in \Gamma_0$ est un sommet quelconque de Γ , alors $\sigma_\gamma(y)$ est une racine et soit $\sigma_\gamma(y) > 0$, soit $-\sigma_\gamma(y) > 0$. Par définition, si y est une racine, on a que $\sigma_\gamma(y)$ est une racine. Comme $B(y) = 1$ (d'après le point 1), et $B(\bar{\gamma}) = 1$, on sait d'après l'inégalité de Cauchy-Schwarz que $|\langle \bar{\gamma}, y \rangle| \leq 1$. De plus, si on écrit $y = \sum_{i=1}^n y_i \bar{\gamma}_i$, on a $2 \langle \bar{\gamma}, y \rangle = 2 \sum_{i=1}^n y_i \langle \bar{\gamma}, \bar{\gamma}_i \rangle$, où $y_i, \langle \bar{\gamma}, \bar{\gamma}_i \rangle \in \mathbb{Z}$, car y est une racine donc est entier d'après le point 1, et $\langle \bar{\gamma}, \bar{\gamma}_i \rangle$ est entier d'après le lemme 5.4.3, d'où $2 \langle \bar{\gamma}, y \rangle \in \mathbb{Z}$. Ainsi on a 5 valeurs possibles pour $2 \langle \bar{\gamma}, y \rangle$: $-2, -1, 0, 1, 2$. Regardons les différents cas possibles :

- si $2 \langle \bar{\gamma}, y \rangle \leq 0$, alors $\sigma_\gamma(y) = y - 2 \langle \bar{\gamma}, y \rangle \bar{\gamma}$ est positif car y est positif et $-2 \langle \bar{\gamma}, y \rangle \bar{\gamma}$ est positif ou nul ;
- si $2 \langle \bar{\gamma}, y \rangle = 1$, alors $1 = 2 \langle \bar{\gamma}, y \rangle = 2y_\gamma - \sum_{l \in \Gamma_\gamma} y_{e(l)}$ et ainsi $y_\gamma > 0$ (sinon $2 \langle \bar{\gamma}, y \rangle \leq 0$) d'où $y_\gamma \geq 1$ comme y est entier. Ainsi, $\sigma_\gamma(y) = y - \bar{\gamma}$ est positif (on ne peut pas avoir $y = \bar{\gamma}$ car sinon $2 \langle \bar{\gamma}, y \rangle = 2$) ;

- si $2 \langle \bar{\gamma}, y \rangle = 2$, alors $\langle \bar{\gamma} - y, \bar{\gamma} - y \rangle = \langle \bar{\gamma}, \bar{\gamma} \rangle - 2 \langle \bar{\gamma}, y \rangle + \langle y, y \rangle = 1 - 2 + 1 = 0$, ainsi, si B est définie positive, on conclut que $\bar{\gamma} - y = 0$, donc $y = \bar{\gamma}$ et $-\sigma_\gamma(y) = -(-\bar{\gamma}) = \bar{\gamma} > 0$.

On a ainsi prouvé le point 3 dans le cas où B est définie positive, et on se contentera de cette démonstration car on utilisera ce résultat uniquement lorsque B sera définie positive. Pour une preuve dans un cas plus général, on pourra regarder dans [25]. \square

Définition 5.4.7. Soit Γ un graphe sans boucle, et soit $\gamma_1, \dots, \gamma_n$ une énumération de ses sommets. Un élément $c = \sigma_{\gamma_n} \dots \sigma_{\gamma_1} \in W$ est appelé *transformation de Coxeter*, c dépend de l'énumération choisie.

Lemme 5.4.8

Soit Γ un graphe sans boucle, $\gamma_1, \dots, \gamma_n$ une énumération de ses sommets, et supposons que la forme quadratique B est définie positive pour ce graphe Γ . On note $c = \sigma_{\gamma_n} \dots \sigma_{\gamma_1}$.

1. La transformation de Coxeter c n'admet aucun point fixe non nul dans \mathcal{E}_Γ ;
2. si $x \in \mathcal{E}_\Gamma$ et $x \neq 0$, alors il existe $i \in \mathbb{N}$ tel que $c^i(x)$ n'est pas positif.

Démonstration. Supposons qu'il existe $y \in \mathcal{E}_\Gamma$ avec $y \neq 0$ et $c(y) = y$. Comme les applications $\sigma_{\gamma_n}, \sigma_{\gamma_{n-1}}, \dots, \sigma_{\gamma_2}$ ne changent pas la coordonnée en γ_1 (c'est-à-dire $\forall i \in \llbracket 2, n \rrbracket \forall z \in \mathcal{E}_\Gamma, (\sigma_{\gamma_i}(z))_{\gamma_1} = z_{\gamma_1}$), on a $(\sigma_{\gamma_1}(y))_{\gamma_1} = c(y)_{\gamma_1} = y_{\gamma_1}$. Comme σ_{γ_1} laisse les coordonnées différentes de γ_1 inchangées, on a $\sigma_{\gamma_1}(y) = y$. Puis, par récurrence, on prouve que $\forall i \in \llbracket 1, n \rrbracket, \sigma_{\gamma_i}(y) = y$. De plus, on a $\forall \gamma \in \Gamma_0, \sigma_\gamma(y) = y = y - 2 \langle \bar{\gamma}, y \rangle \bar{\gamma}$, d'après le point 2 du lemme 5.4.3, d'où $\forall \gamma \in \Gamma_0, \langle \bar{\gamma}, y \rangle = 0$. Les $\bar{\gamma}$ formant une base de \mathcal{E}_Γ , cela nous donne $\forall x \in \mathcal{E}_\Gamma, \langle x, y \rangle = 0$, ainsi $y = 0$, ce qui est une contradiction. Le point 1 est ainsi prouvé, passons au point 2. Comme B est définie positive, on sait d'après le point 4 du lemme 5.4.3 que le groupe W est fini, ainsi il existe $h \in \mathbb{N}$ tel que $c^h = \text{Id}$. Soit $x \in \mathcal{E}_\Gamma$, avec $x \neq 0$, et supposons par l'absurde que $\forall i \in \mathbb{N}, c^i(x) > 0$, alors les vecteurs $x, c(x), \dots, c^{h-1}(x)$ sont positifs, donc le vecteur $y = x + c(x) + \dots + c^{h-1}(x)$ est positif. Or $c(y) = c(x) + c^2(x) + \dots + c^{h-1}(x) + c^h(x)$, et $c^h(x) = x$, donc $c(y) = y$ et y est non nul car positif, ce qui contredit le point 1. Le point 2 est ainsi prouvé. \square

5.5 Le théorème de Gabriel

Soit (Γ, Λ) un carquois.

Définition 5.5.1 (vecteur dimension). Pour chaque objet $V \in \mathcal{L}(\Gamma, \Lambda)$, on note $\dim V \in \mathcal{E}_\Gamma$ le vecteur $(\dim V_\gamma)_{\gamma \in \Gamma_0}$, qu'on appellera *vecteur dimension* de V .

Théorème 5.5.2 (Gabriel [16])

Supposons que Γ est un graphe connexe.

1. *Si $\mathcal{L}(\Gamma, \Lambda)$ ne contient, à isomorphisme près, qu'un nombre fini de représentations indécomposables, alors Γ est un graphe de Dynkin simplement lacé, c'est-à-dire un graphe de type A_n, D_n, E_6, E_7 , ou E_8 .*
2. *Soit Γ un graphe de Dynkin simplement lacé, c'est-à-dire de type A_n, D_n, E_6, E_7 , ou E_8 , et soit Λ une orientation quelconque de Γ . Alors $\mathcal{L}(\Gamma, \Lambda)$ ne contient, à isomorphisme près, qu'un nombre fini de représentations indécomposables. De plus, l'application $V \mapsto \dim V$ est une bijection entre les classes d'équivalence des représentations indécomposables de $\mathcal{L}(\Gamma, \Lambda)$ et les racines positives de \mathcal{E}_Γ .*

Démonstration. On considère les représentations $(V, f) \in \mathcal{L}(\Gamma, \Lambda)$ dont la dimension $\dim V = m = (m_\gamma)_{\gamma \in \Gamma_0}$ est fixée. Si l'on fixe une base dans chacun des espaces vectoriels V_γ , alors la représentation (V, f) est entièrement donnée par la famille de matrices $\{A_l \mid l \in \Gamma_1\}$, où A_l est la matrice représentative de l'application $f_l : V_{s(l)} \rightarrow V_{t(l)}$ dans les bases choisies. Dans chaque espace vectoriel V_γ , on peut changer de base au moyen d'une matrice de changement de base g_γ (de taille m_γ^2), les matrices A_l sont alors remplacées par les matrices :

$$A'_l = g_{t(l)}^{-1} A_l g_{s(l)}. \quad (5.3)$$

Notons A la variété composée de toutes les familles de matrices A_l ($l \in \Gamma_1$) et G le groupe des familles de matrices de changement de base g_γ ($\gamma \in \Gamma_0$). Alors le groupe G agit sur A par la formule (5.3), et on a que deux objets de $\mathcal{L}(\Gamma, \Lambda)$ de même dimension m sont isomorphes si et seulement si les familles de matrices correspondantes appartiennent à une même orbite. S'il n'existe qu'un nombre fini de représentations indécomposables à isomorphisme près dans $\mathcal{L}(\Gamma, \Lambda)$, alors, a fortiori, il n'existe qu'un nombre fini de représentations indécomposables de dimension m , à isomorphisme près. Ainsi, A est la réunion d'un nombre fini d'orbites, et il suit que $\dim A \leq \dim G - 1$ ¹. Or on a $\dim G = \sum_{\gamma \in \Gamma_0} m_\gamma^2$ et $\dim A = \sum_{l \in \Gamma_1} m_{s(l)} m_{t(l)}$. Ainsi la condition $\dim A \leq \dim G - 1$ se réécrit $\sum_{\gamma \in \Gamma_0} m_\gamma^2 - \sum_{l \in \Gamma_1} m_{s(l)} m_{t(l)} \geq 1$, ou encore $B(m) \geq 1$. Comme m est un vecteur entier de \mathcal{E}_Γ , on a donc $B(m) > 0$

1. Ce résultat suit d'une étude faisant intervenir de la géométrie algébrique, nous ne précisons pas comment l'obtenir. Le terme -1 est expliqué par l'existence d'un sous-groupe de dimension 1 agissant trivialement sur A .

dès que $m \neq 0$, car si $x \in \mathcal{E}_\Gamma$ est entier, alors $B(x) \in \mathbb{Z}$ (on ne fait que des multiplications et additions d'entiers et \mathbb{Z} est un anneau). On sait donc que B est définie positive sur les vecteurs entiers positifs de \mathcal{E}_Γ . De plus, si $x \in \mathcal{E}_\Gamma$ et si on note $|x|$ le vecteur tel que $\forall \gamma \in \Gamma_0, |x|_\gamma = |x_\gamma|$, alors on a toujours $B(x) \geq B(|x|)$. En effet :

$$\begin{aligned} B(x) - B(|x|) &= \sum_{\gamma \in \Gamma_0} (x_\gamma^2 - |x|_\gamma^2) - \sum_{l \in \Gamma_1} (x_{s(l)}x_{t(l)} - |x|_{s(l)}|x|_{t(l)}) \\ &= - \sum_{l \in \Gamma_1} (x_{s(l)}x_{t(l)} - |x|_{s(l)}|x|_{t(l)}), \end{aligned}$$

or $\forall l \in \Gamma_1, x_{s(l)}x_{t(l)} \leq |x|_{s(l)}|x|_{t(l)}$, d'où $-\sum_{l \in \Gamma_1} (x_{s(l)}x_{t(l)} - |x|_{s(l)}|x|_{t(l)}) \geq 0$ et ainsi $B(x) \geq B(|x|)$. On a ainsi prouvé que B était définie positive sur les vecteurs entiers de signe quelconque de \mathcal{E}_Γ . Soit maintenant $x \in \mathcal{E}_\Gamma$ un vecteur quelconque, on a $\forall \gamma \in \Gamma_0, x_\gamma \in \mathbb{Q}$, on écrit alors $x_\gamma = \frac{p_\gamma}{q_\gamma}$ et notons k le plus petit commun multiple des dénominateurs q_γ , de sorte que kx soit un vecteur entier de \mathcal{E}_Γ . Alors $B(kx) = k^2B(x)$, et comme $k \neq 0$, on a que $B(kx) = 0$ si et seulement si $B(x) = 0$, or kx est entier donc $B(x) = 0$ équivaut à $kx = 0$, B étant définie positive sur les vecteurs entiers de \mathcal{E}_Γ . Puis $kx = 0$ équivaut à $x = 0$ car $k \neq 0$; finalement, on a prouvé que $B(x) = 0$ si et seulement si $x = 0$, et $B(x)$ est du signe de $B(kx)$ qui est positif, donc B est définie positive sur \mathcal{E}_Γ . Ainsi, on a prouvé que s'il y a, à isomorphisme près, un nombre fini de représentations indécomposables dans $\mathcal{L}(\Gamma, \Lambda)$, alors B est définie positive. Donc, d'après la proposition 5.4.4, cela prouve que Γ est de la forme A_n, D_n, E_6, E_7 , ou E_8 .

Prouvons maintenant la seconde partie du théorème. On se place donc dans le cas où le graphe Γ est du type A_n, D_n, E_6, E_7 ou E_8 , c'est-à-dire un diagramme de Dynkin simplement lacé. Donnons pour commencer un lemme.

Lemme 5.5.3

Soit (Γ, Λ) un carquois, $\beta \in \Gamma_0$ un puits suivant l'orientation Λ , et $V \in \mathcal{L}(\Gamma, \Lambda)$ une représentation indécomposable du carquois (Γ, Λ) . Alors :

- soit $F_\beta^+(V)$ est une représentation indécomposable, et $\dim F_\beta^+(V) = \sigma_\beta(\dim V)$;
- soit $V \cong L_\beta$, $F_\beta^+(V) = 0$, et $\dim F_\beta^+(V) \neq \sigma_\beta(\dim V) < 0$.

On a un résultat similaire pour $\alpha \in \Gamma_0$ une source et le foncteur F_α^- .

Démonstration du lemme 5.5.3. Il s'agit en réalité d'une réécriture du théorème 5.3.8, en utilisant le formalisme introduit dans la partie 5.4. En effet, les formules concernant les dimensions des espaces vectoriels données dans le théorème 5.3.8 correspondent exactement à la définition de σ_β . Le seul point nouveau est que lorsque $V \cong L_\beta$, on a $\dim F_\beta^+(V) \neq \sigma_\beta(\dim V) < 0$, prouvons donc cela. On a $\dim V = \dim(L_\beta) = \bar{\beta}$, et $\sigma_\beta(\bar{\beta}) = -\bar{\beta} < 0$, or

$\dim F_\beta^+(V)$ est un vecteur dimension, donc ne peut pas être négatif, ainsi on a bien prouvé que $\dim F_\beta^+(V) \neq \sigma_\beta(\dim V) < 0$. \square

Corollaire 5.5.4

Soit (Γ, Λ) un carquois, β_1, \dots, β_k une suite de puits suivant l'orientation Λ , et $V \in \mathcal{L}(\Gamma, \Lambda)$ une représentation indécomposable. On note, pour $j \in \llbracket 1, k \rrbracket$, $V_j = F_{\beta_j}^+ \dots F_{\beta_1}^+(V)$, $m_j = \sigma_j \dots \sigma_1(\dim V)$, $V_0 = V$ et $m_0 = \dim V$. On pose $i \in \llbracket 0, n \rrbracket$ le plus grand entier tel que $\forall j \leq i$, $m_j > 0$. Alors, pour $j \leq i$, V_j est indécomposable et $V = F_{\beta_1}^- \dots F_{\beta_j}^-(V_j)$. De plus, si $i < k$, alors $V_{i+1} = V_{i+2} = \dots = V_k = 0$, $V_i \cong L_{i+1}$ et $V \cong F_{\beta_1}^- \dots F_{\beta_i}^-(L_{i+1})$. On a des résultats similaires avec une suite de sources.

Démonstration du corollaire 5.5.4. On reprend les mêmes notations que dans l'énoncé. L'indice i existe car l'indice 0 vérifie les propriétés souhaitées. Supposons par l'absurde que $A = \{j \in \mathbb{N}, j \leq i : V_j \text{ est une représentation non indécomposable}\}$ est non vide. C'est une sous-partie non vide de \mathbb{N} donc elle admet un minimum, qu'on notera $j_0 = \min A$. On a $j_0 \geq 1$ car $V_0 = V$ est indécomposable par hypothèse, ainsi on sait que V_{j_0} n'est pas indécomposable alors que V_{j_0-1} est indécomposable. D'après le lemme 5.5.3, on sait donc que m_{j_0} n'est pas positif, ce qui est une contradiction avec la définition de i . On a donc prouvé que $\forall j \leq i$, V_j est indécomposable. De plus, pour tout entier $j \leq i$, on a $V_j = F_{\beta_j}^+ \dots F_{\beta_1}^+(V) = F_{\beta_j}^+(V_{j-1})$, avec V_{j-1} indécomposable, ainsi d'après le théorème 5.3.8, comme V_j n'est pas nul, on a $F_{\beta_j}^- F_{\beta_j}^+(V_{j-1}) = V_{j-1}$. On en tire donc $F_{\beta_j}^+(V_j) = V_{j-1}$; et, par récurrence, il vient $V = F_{\beta_1}^- \dots F_{\beta_j}^-(V_j)$. Supposons à présent que $i < k$, comme on sait que V_i est indécomposable et que V_{i+1} ne l'est pas, on a d'après le lemme 5.5.3 que $V_i \cong L_{i+1}$, donc que $V \cong F_{\beta_1}^- \dots F_{\beta_i}^-(L_{i+1})$, et que $F_{\beta_{i+1}}^+(V_i) = 0 = V_{i+1}$. Puis, pour tout puits $\beta \in \Gamma_0$, on a $F_\beta^+(0) = 0$, donc $V_{i+2} = \dots = V_k = 0$, ce qui conclut la démonstration du corollaire. \square

Reprenons la démonstration du théorème de Gabriel. Nous allons maintenant prouver que les classes d'isomorphismes des représentations indécomposables du carquois (Γ, Λ) sont en bijection avec les racines positives de \mathcal{E}_Γ , avec pour bijection $V \mapsto \dim V$.

Soit $V \in \mathcal{L}(\Gamma, \Lambda)$ une représentation indécomposable et $\gamma_1, \gamma_2, \dots, \gamma_n$ un tri topologique sur les sommets de Γ suivant l'orientation Λ . Posons c la transformation de Coxeter associée. Le vecteur $\dim V$ est non nul, car V est indécomposable. D'après le point 2 du lemme 5.4.8, il existe un entier k tel que $c^k(\dim V)$ n'est pas positif. En posant $(\delta_1, \delta_2, \dots, \delta_{nk}) = (\gamma_1, \gamma_2, \dots, \gamma_n, \dots, \gamma_1, \gamma_2, \dots, \gamma_n)$ où la suite $\gamma_1, \gamma_2, \dots, \gamma_n$ est répétée k fois, on a $\sigma_{\delta_{nk}} \dots \sigma_{\delta_1}(\dim V) = c^k(\dim V)$. Soit i le plus petit entier tel que pour tout entier $j \leq i$, $\sigma_{\delta_j} \dots \sigma_{\delta_1}(\dim V)$ soit positif. L'entier i est déterminé uniquement par le vecteur $\dim V$; on peut utiliser un raisonnement similaire à celui fait dans la démonstration du lemme 5.3.15 pour obtenir ce résultat.

On a clairement $i < kn$, donc d'après le corolaire 5.5.4, V est isomorphe à $F_{\delta_1}^- F_{\delta_2}^- \dots F_{\delta_n}^- (L_{\delta_{i+1}})$ et $\dim V = \sigma_{\delta_1} \dots \sigma_{\delta_i}(\bar{\delta}_{i+1})$. On a bien que $\dim V$ est racine positive. De plus, on peut, à partir de $\dim V$, obtenir la classe d'isomorphisme de V , ainsi on a l'injectivité souhaitée.

Soit $x \in \mathcal{E}_\Gamma$ une racine positive. On considère comme ci-dessus un tri topologique $\gamma_1, \dots, \gamma_n$ et $c = \sigma_{\gamma_1} \dots \sigma_{\gamma_n}$ la transformation de Coxeter associée. D'après le lemme 5.4.8 il existe $k \in \mathbb{N}$ un entier tel que $c^k(x)$ n'est pas positif. On considère alors la suite de sommets $\delta_1, \dots, \delta_{nk} = (\gamma_1, \dots, \gamma_n, \dots, \gamma_1, \dots, \gamma_n)$ (k fois), qui est une suite de puits d'après la proposition 5.3.12. On a alors que $\sigma_{\delta_{nk}} \dots \sigma_{\delta_1} = c^k(x)$ n'est pas positif. On note alors i le plus grand indice tel que $\forall j \leq i, \sigma_{\delta_j} \dots \sigma_{\delta_1}(x) > 0$. On remarque comme dans la preuve du lemme 5.4.6 qu'on a nécessairement $\sigma_{\delta_i} \dots \sigma_{\delta_1}(x) = \bar{\delta}_{i+1}$. On considère désormais la suite de sommets $\delta_i, \delta_{i-1}, \dots, \delta_1$, comme $\delta_1, \dots, \delta_i$ est une suite de puits suivant l'orientation Λ , on a que $\forall j \in \llbracket 1, n \rrbracket$, δ_j est un puits suivant l'orientation $\sigma_{\delta_{j-1}} \dots \sigma_{\delta_1} \Lambda$, ainsi $\forall j \in \llbracket 1, n \rrbracket$, δ_j est une source suivant l'orientation $\sigma_{\delta_j} \sigma_{\delta_{j-1}} \dots \sigma_{\delta_1} \Lambda$, c'est-à-dire que $\delta_i, \delta_{i-1}, \dots, \delta_1$ est une suite de sources suivant l'orientation $\sigma_{\delta_i} \dots \sigma_{\delta_1} \Lambda$. On considère alors la représentation indécomposable $L_{\delta_{i+1}} \in \mathcal{L}(\Gamma, \sigma_{\delta_i} \dots \sigma_{\delta_1} \Lambda)$, et on va désormais appliquer le corolaire 5.5.4. On a $\dim L_{\delta_{i+1}} = \bar{\delta}_{i+1}$, et $\forall j \leq i, \sigma_{\delta_{i+1-j}} \dots \sigma_{\delta_i}(\bar{\delta}_{i+1}) = \sigma_{\delta_{i-j}} \dots \sigma_{\delta_1}(x) > 0$ par définition de i . Ainsi, d'après le corolaire 5.5.4 on sait que $\forall j \leq i, V_j := F_{\delta_{i+1-j}}^- \dots F_{\delta_i}^- (L_{\delta_{i+1}})$ est une représentation indécomposable, et d'après le lemme 5.5.3, $\dim V_j = \sigma_{\delta_{i+1-j}} \dots \sigma_{\delta_i}(\bar{\delta}_{i+1})$, en particulier $V := V_i = F_{\delta_1}^- \dots F_{\delta_i}^- (L_{\delta_{i+1}}) \in \mathcal{L}(\Gamma, \Lambda)$ est une représentation indécomposable et $\dim V = \sigma_{\delta_1} \dots \sigma_{\delta_i}(\bar{\delta}_{i+1}) = x$. On a ainsi prouvé que les classes d'isomorphismes des représentations indécomposables du carquois (Γ, Λ) sont en bijection avec les racines positives de \mathcal{E}_Γ . Or, nous sommes dans le cas où le graphe Γ est un diagramme de Dynkin simplement lacé, donc la forme quadratique B est définie positive d'après la proposition 5.4.4, donc le groupe W est fini d'après le point 4 du lemme 5.4.3, ainsi les racines, et a fortiori les racines positives, sont en nombre fini, donc les classes d'isomorphisme des représentations indécomposables sont également en nombre fini. Cela conclut donc la preuve du théorème de Gabriel. \square

Et par la même occasion, cela conclut ce mémoire.

Chapitre 6

Stage hors parcours

6.1 Préambule

This internship took place during July and August 2016, the aim was to study the algorithms presented in [13], and to implement them in Nemo [21], a computer algebra package for the Julia [1] programming language. These algorithms were designed to deal with the algebraic closure of finite fields, and often have a quasi-linear cost, whereas traditional algorithms have quadratic cost. This paper intends to compare the Julia and the C implementations of the code, in terms of speed, but also genericity and ease of writing and reading such code. As we will see, writing code in Julia is really easy, and the tools available to develop the code are sharp, but the speed of the implementation in Julia does not match the speed of the C implementation.

6.2 Julia

6.2.1 Overview of Julia's characteristics

Julia is a free and open-source, high-level programming language developed since 2012, with dynamic type system and high-performance. It is a compiled language, with a *just-in-time* (jit) compilation. This means that when one writes a function, e.g.

```
julia> function myFunction(n)
    return n^3+n+2
end
```

the function will be compiled during the very first call. After this first call, it will be much faster to run the compiled function

```
julia> myFunction(BigInt(2)^300)
```

than the expression below.

```
julia> (BigInt(2)^300)^3 + BigInt(2)^300 + 2
```

In order to optimize the code, the compiler creates a new function for each type of input that one can think of when calling `myFunction`. For example, in the code below, Julia will compile `myFunction` three times, which will result in three functions, each one of them optimized for `Int64` (the default type of 17), `Float64`, or `BigInt`.

```
julia> myFunction(17)
4932
julia> myFunction(17.)
4932.0
julia> myFunction(BigInt(17))
4932
```

Julia is a very easy-to-learn language, the syntax is intuitive for someone who already worked with high-level languages, like Python. The code written in `FastArithmetic.jl` is very generic, thanks to the type system of Julia, and works for fields \mathbb{F}_{p^n} with a small p as well as large p . More importantly, the *jit* compiler creates a function for each kind of p . Indeed the elements of \mathbb{F}_{p^n} have type `fq_nmod` for p small or `fq` for p large, so one can write only one function and have two compiled functions : one for each type, which will be optimized by the compiler for this type.

6.2.2 Nemo

Nemo is a computer algebra package of Julia, it can be installed with

```
julia> Pkg.add("Nemo")
julia> using Nemo
```

and tested with the following.

```
julia> Pkg.test('Nemo')
```

Nemo is based on C/C++ libraries such as Flint, Antic, Pari, etc. and is also written in Julia. It provides a lot of features, and all the work of this internship is based on Nemo. Indeed, we work with Nemo finite fields and Nemo polynomials over these fields. `fq`, `fq_nmod`, and `fq_nmod_poly` (the type of the polynomials over \mathbb{F}_{p^n}) are all Nemo types. The interested reader can learn more in Nemo's manual, which is very well documented.

6.3 A bit of theory

6.3.1 General background

In most of the computer algebra systems, like Sage, Magma, Flint, etc., it is possible to work with finite fields and their algebraic closure. Let p be a

prime number and $k = \mathbb{F}_p$ the field with p elements, the algebraic closure of k is infinite and is $\bar{k} = \cup_{i \geq 0} \mathbb{F}_{p^i}$, dealing with \bar{k} means being able to compute the finite fields \mathbb{F}_{p^i} , and also being able to embed small fields into large ones, or project elements of large fields into smaller ones (when possible).

These algorithms often rely on linear algebra, and the cost is at least quadratic in the degree of the extension used. The algorithms presented in [13] rely on polynomial arithmetic, and have a quasi-linear cost for most of the algorithms needed to compute in the algebraic closure of a finite field. In this section, we will introduce (briefly) the tools used in the algorithms, and explain how they work. The most important result is the *transposition principle*, an algorithmic proposition that guarantees that with any linear algorithm performing a matrix-vector product $v \mapsto Mv$, one can *transpose* it to obtain another algorithm, which has essentially the same cost, performing the transposed matrix-vector product $v \mapsto M^t v$. But we first need to talk about *duality*, because the transposition principle also changes the used basis.

6.3.2 Trace and duality

Let k be a finite field, and E, F be two k -vector spaces with the same finite dimension $\dim E = \dim F < \infty$. Let also $\langle \cdot, \cdot \rangle : E \times F \rightarrow k$ be a non-degenerate bilinear form. Then, for any basis $\boldsymbol{\xi} = (\xi_i)_i$ of E , there exists a unique basis $\boldsymbol{\xi}^* = (\xi_j^*)_j$ of F such that $\forall (i, j), \langle \xi_i, \xi_j^* \rangle = \delta_{i,j}$ where $\delta_{i,j}$ is the KRONECKER symbol and $\delta_{i,j} = 1$ if $i = j$ and 0 otherwise. The basis $\boldsymbol{\xi}^*$ is called the *dual basis* of $\boldsymbol{\xi}$. Similarly, let E', F' be two other k -vector spaces with $\dim E' = \dim F' < \infty$ and $\langle \cdot, \cdot \rangle' : E' \times F' \rightarrow k$ another non-degenerate bilinear form. Then, for any linear map $u : E \rightarrow E'$, there exists a unique map $u^t : F' \rightarrow F$ such that $\forall (a, b) \in E \times F', \langle u(a), b \rangle' = \langle a, u^t(b) \rangle$. This map is called the *dual map* of u , with respect to $\langle \cdot, \cdot \rangle$ and $\langle \cdot, \cdot \rangle'$.

It is now time to speak about the *trace form*, it will be our candidate to be $\langle \cdot, \cdot \rangle$, the non-degenerate bilinear form. Let P be an irreducible polynomial of $k[x]$ of degree m , let $K = k[x]/(P)$ be a finite extension of k , and $a \in K$. We denote by μ_a the application of multiplication-by- a , $\mu_a : K \rightarrow K, b \mapsto ab$. K is a k -vector space of dimension m and μ_a is an endomorphism of K . For all $a, b \in K$, we also denote by $\tau_P(ab)$ the trace of the endomorphism μ_{ab} , and we define $\langle a, b \rangle_P = \tau_P(ab)$. This defines a non-degenerate bilinear form on $K \times K$. Thus, if $\boldsymbol{\xi}$ is a basis of K , one can obtain a dual basis $\boldsymbol{\xi}^*$ with respect to the bilinear form $\langle \cdot, \cdot \rangle_P$. Hence, if $a \in K$ has the coordinates $(a_i)_i$ in the basis $\boldsymbol{\xi}^*$, these coordinates are given by $a_i = \langle \xi_i, a \rangle$, so $a = \sum a_i \xi_i^* = \sum \langle \xi_i, a \rangle \xi_i^*$. This formula will be important for the change-of-basis algorithms.

Let now Q be an irreducible polynomial of $k[y]$ of degree n , with n coprime to m , then $L = k[x, y]/(P, Q) \cong \mathbb{F}_{p^{mn}}$ is also a finite extension of k , of degree mn . Since L is a k -vector space (of dimension mn), it is again possible to construct the trace $\tau_{P,Q}$ of an element of L , and we define the same way a non-degenerate bilinear form $\langle \cdot, \cdot \rangle_{P,Q}$ on $L \times L$. If $\boldsymbol{\xi} = (x^i)_{0 \leq i \leq m-1}$

and $\mathbf{v} = (y^j)_{0 \leq j \leq n-1}$ are the monomial bases of respectively $k[x]/(P)$ and $k[y]/(Q)$, then $\boldsymbol{\xi} \otimes \mathbf{v} = (x^i y^j)_{0 \leq i \leq m-1, 0 \leq j \leq n-1}$ is the canonical monomial basis of L . But, if $\boldsymbol{\xi}^*$ and \mathbf{v}^* are respectively the dual basis of $\boldsymbol{\xi}$ with respect to τ_P and the dual basis of \mathbf{v} with respect to τ_Q , $\boldsymbol{\xi} \otimes \mathbf{v}^*$, $\boldsymbol{\xi}^* \otimes \mathbf{v}$, and $\boldsymbol{\xi}^* \otimes \mathbf{v}^*$, are also bases of L . What's more, $\boldsymbol{\xi} \otimes \mathbf{v}$ and $\boldsymbol{\xi}^* \otimes \mathbf{v}^*$ are dual with respect to $\tau_{P,Q}$, and so are $\boldsymbol{\xi} \otimes \mathbf{v}^*$ and $\boldsymbol{\xi}^* \otimes \mathbf{v}$.

Constructing $\mathbb{F}_{p^{mn}}$ this way leads to a bivariate representation of the elements. We prefer a univariate representation, because the algorithms we have in this case are more efficient, even for simple operations such as multiplication. Hence, we can represent $\mathbb{F}_{p^{mn}}$ by $k[z]/(R)$ where $R = P \odot Q$ is the composed product of P and Q . It means that if $(p_i)_{0 \leq i \leq m-1}$ and $(q_j)_{0 \leq j \leq n-1}$ are respectively the roots of P and Q in a algebraic closure of k , R is the polynomial whose roots are the products $p_i q_j$. So we have embeddings φ_x , φ_y and an isomorphism Φ of the form :

$$\begin{aligned} \varphi_x : \quad & k[x]/(P) \rightarrow k[z]/(R), \\ \varphi_y : \quad & k[y]/(Q) \rightarrow k[z]/(R), \text{ and} \\ \Phi : \quad & A = k[x, y]/(P, Q) \rightarrow k[z]/(R) \\ & \quad \quad \quad xy \mapsto z. \end{aligned}$$

We will provide algorithms for φ_x , φ_y and Φ in Section 6.3.4.

6.3.3 Transposition principle

Let, as in the previous section, $\boldsymbol{\xi}$ be a basis of E , and $\boldsymbol{\xi}^*$ the basis of F which is dual to $\boldsymbol{\xi}$. Let also \mathbf{v} and \mathbf{v}^* be, respectively, a basis of E' and its dual basis of F' . If M is the matrix of a linear map $u : E \rightarrow E'$ in the bases $(\boldsymbol{\xi}, \mathbf{v})$, then the matrix of the linear map $u^t : F' \rightarrow F$ in the bases $(\mathbf{v}^*, \boldsymbol{\xi}^*)$ is M^t . Given an algorithm to compute a linear map $u : E \rightarrow E'$ in the bases $(\boldsymbol{\xi}, \mathbf{v})$, the transposition principle says that one can transpose it to obtain a new algorithm computing the dual map $u^t : F' \rightarrow F$ in the bases $(\mathbf{v}^*, \boldsymbol{\xi}^*)$. The costs of the two algorithms only differ by a constant. Roughly speaking, transposing an algorithm consists on transposing all the subroutines and inverse their orders.

Exemple 6.3.1. Let A be an algorithm taking x_1 and x_2 as input, and computing $y_1 = ax_1 + bx_2$ and $y_2 = cx_1 + dx_2$. A is linear and we obtain its transposition A^t by reversing all the arrows and exchanging \times and $+$ in the graph representing A . We obtain an algorithm that takes y_1 and y_2 as input and compute $x_1 = ay_1 + cy_2$ and $x_2 = by_1 + dy_2$.

Let us introduce some operations and their transposes. Let k be a field and $k[x]_m$ the vector space of polynomials of degree at most m , the reversal operator of $k[x]_m$ is the map such that

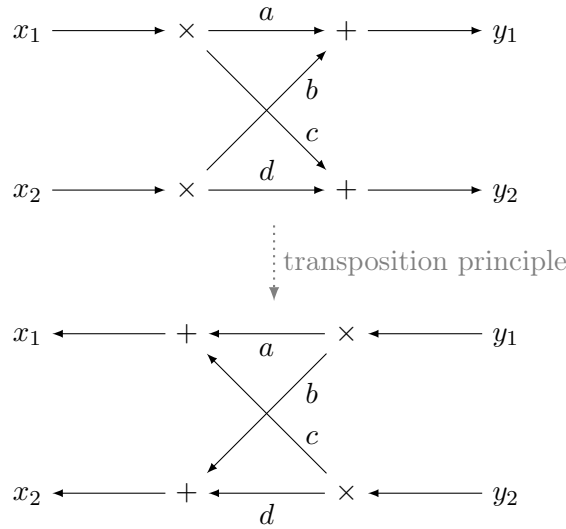


FIGURE 6.1 – Transposition of a simple arithmetic circuit

$$\mathbf{rev}(\cdot, m) : \begin{array}{l} k[x]_m \rightarrow k[x]_m \\ Q \mapsto x^m Q(1/x) \end{array} .$$

We can see that \mathbf{rev} is its own transpose, indeed \mathbf{rev} is a linear map and its matrix in the canonical basis of $k[x]_m$ is symmetric : it is the matrix with 1's on the anti-diagonal. Let now $P \in k[x]$ be a monic polynomial of degree m and $n \in \mathbb{N}$ an integer, we denote by $\mathbf{rem}(\cdot, P, n)$ the remainder by P with length n , defined by

$$\mathbf{rem}(\cdot, P, n) : \begin{array}{l} k[x]_n \rightarrow k[x]_{m-1} \\ Q \mapsto Q \bmod P \end{array} .$$

As explained in [10], $\mathbf{rem}^t(\cdot, P, n)$ is the linear sequence extension : it takes as input the m first terms of a linear recurring sequence and computes the $n + 1$ first terms using P as the minimal polynomial of the sequence.

6.3.4 The algorithms

All the algorithms described here, and their complexity analysis, have been discussed in [13], that is why we will not show all the pseudo-codes but rather try to explain how they work and on which result they are based. The complexities will also be given without any proofs.

As we saw in the previous section, transposing an algorithm results in a change of basis. Thus, it is important to be able to go from a basis to another. Here again k is a finite field, P an irreducible monic polynomial of $k[x]$, and $K = k[x]/(P)$ is a finite extension of k . We will be considering two

bases of K : the monomial basis $\xi = (x^i)_{0 \leq i \leq m-1}$ (where m is the degree of P), and its dual basis ξ^* with respect to $\langle \cdot, \cdot \rangle_P$.

Change of basis. The two first algorithms allow us to juggle between the monomial and the dual basis. They are both based on a lemma which describes the generating series of the traces $\tau_P(ax^i)_{0 \leq i \leq m-1}$, where $a \in K$.

Lemme 6.3.2

Let $a \in K$ be an element of K . In the formal power series ring $k[[x]]$, we have :

$$\sum_{i \geq 0} \tau_P(ax^i)x^i = \frac{\text{rev}(P'a \text{ mod } P, m)}{\text{rev}(P, m+1)}.$$

Going from the monomial basis to the dual basis can be done by computing $\text{rev}(P'a \text{ mod } P, m) \text{ mod } x^m$, $(\text{rev}(P, m+1) \text{ mod } x^m)^{-1}$, and the product of them. We obtain, by Lemma 6.3.2 :

$$\frac{\text{rev}(P'a \text{ mod } P, m)}{\text{rev}(P, m+1)} \text{ mod } x^m = \sum_{i=0}^{m-1} \tau_P(ax^i)x^i.$$

Since we said that $(\tau_P(ax^i))_{0 \leq i \leq m-1}$ are the coordinates of a in the dual basis ξ^* , taking the coefficient of the polynomial $\sum_{i=0}^{m-1} \tau_P(ax^i)x^i$ gives a solution to go from the monomial basis to the dual basis, that can be written as follows.

`monomialToDual(a, P)`

Input $\mathbf{a} = (a_i)_{0 \leq i \leq m-1} \in k^m$, P an irreducible monic polynomial of degree m .

Output $(\tau_P(ax^i))_{0 \leq i \leq m-1}$, with $a = \sum_{0 \leq i \leq m-1} a_i x^i$

1. $T = 1/\text{rev}(P, m+1) \text{ mod } x^m$
2. $b = \text{rev}(P'a = \sum_{0 \leq i \leq m-1} a_i x^i \text{ mod } P, m)T \text{ mod } x^m$
3. **return** $(\text{coefficient}(b, x^i))_{0 \leq i \leq m-1}$

Obtaining the coefficients in the monomial basis ξ knowing the coefficients in the dual basis ξ^* is done using the same equality, but starting from the other half. So, computing $\sum_{0 \leq i \leq m-1} \tau_P(ax^i)x^i \text{ mod } x^m$, $\text{rev}(P, m+1)$, and multiplying them gives, by Lemma 6.3.2 again :

$$\text{rev}(P, m+1) \sum_{0 \leq i \leq m-1} \tau_P(ax^i)x^i \text{ mod } x^m = \text{rev}(P'a \text{ mod } P, m) \text{ mod } x^m.$$

Then, applying $\text{rev}(\cdot, m)$ and multiplying by $(P')^{-1} \text{ mod } P$ gives $a \text{ mod } P$, so taking the coefficients of a gives the coefficients in the monomial basis ξ .

`dualToMonomial`(\mathbf{b}, P)

Input $\mathbf{b} = (b_i)_{0 \leq i \leq m-1} \in k^m$, P an irreducible monic polynomial of degree m .

Output $(a_i)_{0 \leq i \leq m-1}$ such that $\tau_P(\sum_{0 \leq i \leq m-1} a_i x^{i+j}) = b_j$ for all j

1. $S = 1/P \pmod{P}$
2. $b = \text{rev}(P, m+1) \sum_{0 \leq i \leq m-1} b_i x^i \pmod{x^m}$
3. $c = \text{rev}(b, m)$
4. $d = cS \pmod{P}$
5. **return** $(\text{coefficient}(d, x^i))_{0 \leq i \leq m-1}$

As we see, even if the lemma holds in $k[[x]]$, we just need to work $\pmod{x^m}$ so the inverses can be obtained with fast algorithms for EUCLIDE division. `monomialToDual` uses $O(M(m))$ operations in k , where $M(m)$ is the number of operations in A of a multiplication of polynomials of degree at most m on any ring A . The cost of `dualToMonomial` is $O(M(m) \log(m))$, but can be reduced to $O(M(m))$ with precomputations. M is a function that is quasi-linear, so the change-of-basis algorithms are quasi-linear too.

Embedding and computing R . Let Q be an irreducible monic polynomial of $k[y]$ of degree n , with n coprime to m . Let also $R = P \odot Q$ be the composed product of P and Q . We have the extensions $k[x]/(P)$, $k[y]/(Q)$, $k[x, y]/(P, Q)$, and $k[z]/(R)$ which respectively have the traces τ_P , τ_Q , $\tau_{P,Q}$, τ_R and the associated bilinear forms. These extensions are respectively endowed with the monomial bases ξ , \mathbf{v} , $\xi \otimes \mathbf{v}$ and ζ . Finally, we denote by ξ^* , \mathbf{v}^* and ζ^* the dual bases of ξ , \mathbf{v} and ζ with respect to $\langle \cdot, \cdot \rangle_P$, $\langle \cdot, \cdot \rangle_Q$ and $\langle \cdot, \cdot \rangle_R$. In this paragraph, we will give algorithms to compute the embeddings φ_x , φ_y , their sections, and to compute the polynomial R . The algorithms follow from the next lemma.

Lemma 6.3.3

Let b be in $k[x]/(P)$ and c in $k[y]/(Q)$. Then we have $\tau_R(\Phi(bc)) = \tau_{P,Q}(bc) = \tau_P(b)\tau_Q(c)$.

This lemma will permit us to compute the restriction of Φ to the set

$$\Pi = \{bc \mid b \in k[x]/(P), c \in k[y]/(Q)\} \subset k[x, y]/(P, Q).$$

Indeed, if we have the coordinates $(\tau_P(bx^i))$ and $(\tau_Q(cy^j))$ of b and c in the dual bases, we can extend them up to $mn - 1$ using rem^t , then, by Lemma 6.3.3, we have $(\tau_P(bx^i)\tau_Q(cy^j))_{0 \leq i \leq mn-1} = (\tau_R(\Phi(bc)z^i))_{0 \leq i \leq mn-1}$ which gives us the coordinates of $\Phi(bc)$ in the dual basis ζ^* . So we can obtain φ_x and φ_y by applying the last formula with respectively $c = 1$ and $b = 1$. This can be written as follows.

$\text{embed}(\mathbf{b}, \mathbf{c}, r)$

Input $\mathbf{b} = (b_i)_{0 \leq i \leq m-1} \in k^m$, $\mathbf{c} = (c_i)_{0 \leq i \leq n-1} \in k^n$, an optional integer $r \geq mn$ set to $r = mn$ by default

Output $\mathbf{a} = (a_i)_{0 \leq i \leq r-1} \in k^r$

1. $(t_i)_{0 \leq i \leq r-1} = \text{rem}^t(\mathbf{b}, P, r)$
2. $(u_i)_{0 \leq i \leq r-1} = \text{rem}^t(\mathbf{c}, P, r)$
3. **return** $(t_i u_i)_{0 \leq i \leq r-1}$

embed uses $O(r(M(m)/m + M(n)/n))$ operations in k . If u_P is the vector of the coordinates of $1 \in k[x]/(P)$ in the dual basis $\boldsymbol{\xi}^*$ and u_Q the vector defined similarly, we can compute $\varphi_x = \text{embed}(\cdot, u_Q)$ and $\varphi_y = \text{embed}(u_P, \cdot)$. Since R is the minimal polynomial of the sequence $(\tau_R(z^i)) = \text{embed}(u_P, u_Q)$, we can obtain R by computing the BERLEKAMP-MASSEY algorithm (which computes minimal polynomials of linear sequences).

As we have just seen, the theory of duality gives us a very simple algorithm for embed , but together with the transposition principle, it will also prove useful to find algorithms for the section of φ_x , and, as we will see later, for Φ^{-1} . Let $c \in k[y]/(Q)$ such that $\tau_Q(c) = 1$ (for example $c = v_0^*$ is convenient), $\varepsilon : k[x]/(P) \rightarrow k[z]/(R)$, $b \mapsto \Phi(bc)$, and $\varepsilon^t : k[z]/(R) \rightarrow k[x]/(P)$ its dual map with respect to $\langle \cdot, \cdot \rangle_P$ and $\langle \cdot, \cdot \rangle_R$. Then, for any b, b' in $k[x]/(P)$, we have

$$\begin{aligned} \langle b, b' \rangle_P &= \tau_P(bb') = \tau_P(bb')\tau_Q(c) = \tau_R(\Phi(bb'c)) \\ &= \langle \varepsilon(b), \Phi(b') \rangle_R = \langle b, \varepsilon^t(\Phi(b')) \rangle_P. \end{aligned}$$

Since $\langle \cdot, \cdot \rangle_P$ is non-degenerate, this means that $b' = \varepsilon^t(\Phi(b')) = \varepsilon^t(\varphi_x(b'))$ for all $b' \in k[x]/(P)$. In other words, ε^t is an inverse of φ_x on its image, also called a section of φ_x . Remarking that ε is nothing else than $\text{embed}(\cdot, c)$, where $c = (1, 0, \dots, 0)$, we have an algorithm for ε , and, by transposing this algorithm, we have an algorithm for ε^t , which is the section of φ_x we were searching. This algorithm uses $O(nM(m) + nM(n))$ operations in k .

Isomorphism. We will apply the same kind of strategy to deduce an algorithm for Φ^{-1} from an algorithm for Φ . First, recall that $\boldsymbol{\xi} \otimes \mathbf{v}$, $\boldsymbol{\xi}^* \otimes \mathbf{v}^*$, $\boldsymbol{\xi}^* \otimes \mathbf{v}$ and $\boldsymbol{\xi} \otimes \mathbf{v}^*$ are all bases of $k[x, y]/(P, Q)$ and that the second is dual to the first and the third to the fourth, with respect to $\langle \cdot, \cdot \rangle_{P, Q}$. Let Φ^t be the dual map to Φ with respect to $\langle \cdot, \cdot \rangle_{P, Q}$ and $\langle \cdot, \cdot \rangle_R$. For any b and b' in $k[x, y]/(P, Q)$, we have

$$\begin{aligned} \langle b, b' \rangle_{P, Q} &= \tau_{P, Q}(bb') = \tau_R(\Phi(bb')) \\ &= \langle \Phi(b), \Phi(b') \rangle_R = \langle b, \Phi^t(\Phi(b')) \rangle_{P, Q}. \end{aligned}$$

So, here again, thanks to the non-degeneracy of $\langle \cdot, \cdot \rangle_{P, Q}$, we have $\Phi^t = \Phi^{-1}$. Hence, if we have an algorithm computing Φ in some bases \mathbf{a} and \mathbf{b} , we can deduce, using the transposition principle, an algorithm computing Φ^{-1} in

the dual bases \mathbf{b}^* and \mathbf{a}^* . We have two algorithms for computing Φ , the first is better in cases where m is small compared to n , and uses the linearity of Φ and the work already done. The idea is to express $b \in k[x, y]/(P, Q)$ in the form $b = \sum_i^{m-1} b_i x^i$ with $b_i \in k[y]/(Q)$ for all $0 \leq i \leq m-1$. Since the $b_i x^i$ are all in Π , we can apply `embed` and sum the result to obtain $\Phi(b) = \sum_i^{m-1} \Phi(b_i x^i)$. To apply this strategy, the b_i must be expressed in the dual basis \mathbf{v}^* of $k[y]/(Q)$, because `embed` takes its input in the dual bases. In the end, exploiting the fact that the coordinates of the x^i in the dual basis $\boldsymbol{\xi}^*$ are $(\tau_P(x^{i+j}))_{0 \leq j \leq m-1}$ (u_P shifted i times), we have an algorithm to compute Φ in the basis $\boldsymbol{\xi} \otimes \mathbf{v}^*$ and giving the result in the basis $\boldsymbol{\zeta}^*$. As we noticed, transposing this algorithm will create a new algorithm computing Φ^{-1} and expressed in the bases $\boldsymbol{\zeta}$ and $\boldsymbol{\xi}^* \otimes \mathbf{v}$. These algorithms both have a quadratic cost in the degree m : $O(m^2 M(n))$.

The second way of computing Φ does not depend on previous algorithms, but we will still be able to deduce an algorithm for Φ^{-1} from it using the transposition principle. It is best suited when dealing with m and n of the same magnitude and it uses baby steps / giant steps : a technique consisting of sacrificing memory to gain speed. Recall that $\Phi(xy) = z$, then, if $b \in k[x, y]/(P, Q)$, we can write

$$\begin{aligned} b &= \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} b_{i,j} x^i y^j = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} b_{i,j} x^i y^i y^{j-i} \\ &= \sum_{h=-m+1}^{n-1} \sum_{i=0}^{m-1} b_{i,i+h} (xy)^i y^h = \frac{1}{y^{m-1}} \sum_{h=0}^{m+n-2} c_h(xy) y^h, \end{aligned}$$

with $c_h(xy) = \sum_{0 \leq i \leq m-1} b_{i,i+h-m+1} (xy)^i$, and where undefined indices $b_{i,i+h-m+1}$ equal zero. So $\Phi(xy) = z$ is known, but also $T = \Phi(y)$, because $y \in \Pi$ so we can use `embed` to compute it. Then, we have

$$a = \frac{1}{T^{m-1}} \tilde{a} \quad \text{mod } R \quad \text{with } \tilde{a} = \sum_{h=0}^{m+n-2} c_h(z) T^h.$$

This is the equality that will be exploited to compute Φ , and, thanks to baby steps / giant steps techniques, this will be reduced to matrix multiplication (but the coefficients of those matrices are polynomials). This technique gives an algorithm expressed in the bases $\boldsymbol{\xi} \otimes \mathbf{v}$ and $\boldsymbol{\zeta}$. Details can be found in [13]. As we said earlier, we immediatly deduce an algorithm for Φ^{-1} using the transposition principle, this last algorithm will be expressed in the bases $\boldsymbol{\zeta}^*$ and $\boldsymbol{\xi}^* \otimes \mathbf{v}^*$. Both these algorithms use $O(M(mn)n^{1/2} + M(m)n^{(\omega+1)/2})$ operations in k , where ω is a constant such that one can multiply two matrices of size $n \times n$ over any ring A in $O(n^\omega)$ operations in A (we know that $2 < \omega \leq 2.38$).

6.4 Julia/Nemo in practice

Since Julia is easy-to-write, we hope it is also easy-to-read. If not, it is probably due to the way `FastArithmetic.jl` is written, rather than to Julia itself. The code can be found at <https://github.com/erou/FastArithmetic.jl>. What's more, it is really easy to install personal packages, one just has to clone the repository inside Julia, using

```
julia> Pkg.clone('https://github.com/edouardRousseau/FastArithmetic.jl')
```

and the package can be easily tested by running the following.

```
julia> Pkg.test('FastArithmetic')
```

The code is very likely to change, but it is possible to update the package as follows.

```
julia> Pkg.update()
```

6.4.1 Benchmarks

We will now look at the speed of the code in Julia, and compare it with the speed in C. All the benchmarks in Julia were realised with the `@benchmark` macro available in the `BenchmarkTools` package. This macro can be used to benchmark any function with

```
julia>@benchmark 1+1
BenchmarkTools.Trial:
  samples:          10000
  evals/sample:     1000
  time tolerance:   5.00%
  memory tolerance: 1.00%
  memory estimate:  0.00 bytes
  allocs estimate:  0
  minimum time:    3.00 ns (0.00% GC)
  median time:     3.00 ns (0.00% GC)
  mean time:       3.04 ns (0.00% GC)
  maximum time:    10.00 ns (0.00% GC)
```

which runs the desired function and measures the time it takes. It is possible to choose all the parameters (number of samples, number of evaluation per sample, ...) of the benchmark and one can use different estimators. For the following benchmarks, we use the median time, because it is more stable than the mean time and still gives an idea of the average time the function will take. The stability we are talking about is linked with the memory management in Julia. Indeed, in Julia, we do not have to concern about

memory, and the variables we assign are cleared by the Garbage Collector (GC). The time Julia passes in the GC is not very clear nor predictable, and the operations in the GC take a significant time. That is why the time taken by a function can change between two samples, depending on the time spent in the GC. By default, before every benchmark, the garbage is automatically collected, but there is still a bit of instability. We benchmark our functions in the case $p = 5$, as in [13], because no difference is seen between different values of p . Since all the functions written in `FastArithmetic.jl` depend on at least one polynomial P , sometimes also Q and $R = P \odot Q$, we let $m = \deg(P)$ grow from 1 to 200 and benchmark the functions for all these values of m . We choose $\deg(Q) = m + 1$, and both P and Q are irreducible polynomials. Irreducible polynomials of degree 1 to 201 were precomputed, by choosing random elements in $\mathbb{F}_p[X]$, in order to always use the same list and to be sure that we are not using special polynomials that could be speeding up the functions (like sparse polynomials). When only one field is involved in an operation (such as in modular multiplication or in change of basis) the field used is $\mathbb{F}_{p^{m(m+1)}} \cong k[z]/(R)$.

The benchmarks of the C code were realized thanks to a python script written by the authors of the code, that can be found at https://github.com/defeo/ff_compositum. The benchmarks work the same but not all the values of m were tested, so we just chose to compare the results for certain values of m tested (we also could have change the script...).

In the tables, the speed of the C code is set to 1.00, and smaller is better. One of the functions benchmarked in the C code, the modular multiplication `mulmod`, was already implemented in Nemo, so we start to compare the results. We see no difference in terms of speed and it was predictable : the Julia function is calling C code.

mulmod	
m	Julia
10	0.99
51	1.04
104	1.03
140	1.05
187	1.07

The speed of `monomialToDual` and of `dualToMonomial` are the same, either in C or Julia, so we show the results only for the first. All the Julia functions differ from the C ones by a constant factor, which depends on the functions but is around 8. Surprisingly, `inversePhi1` is three times faster than its transposition `phi1`, and `inversePhi2` is slower than `phi2` for small values of m but equals its speed at $m \approx 100$ and is two times faster for $m \geq 170$. In Julia, the cross point between `phi1` and `phi2` is earlier than in C, it occurs at $m = 30$.

We did not mention the time to compute R in these tables, the reason is because the implementation of the BERLEKAMP-MASSEY algorithm used in it is naive, so the algorithm `computeR` can be improved easily by improving BERLEKAMP-MASSEY.

monomialToDual		monomialToDual_pre		mulModT	
m	Julia	m	Julia	m	Julia
10	10.50	10	3.13	10	19.92
51	10.92	51	2.99	51	15.62
104	10.26	104	1.96	104	16.23
140	10.43	140	2.06	140	17.73
187	11.03	187	3.06	187	17.97

embed		project		phi1		phi2	
m	Julia	m	Julia	m	Julia	m	Julia
10	5.29	10	5.73	10	5.39	10	1.90
51	4.51	51	3.54	51	6.29	51	3.40
104	3.85	104	3.00	104	5.26	104	5.78
140	6.78	140	2.65	140	4.91	140	8.28
187	5.42	187	6.44	187	5.74	187	9.94

6.5 Conclusion

As we see, we are not able to reach C speed, and the Julia code is slower by a constant factor, approximately 8. Part of this factor is due to precomputations in C that are not done in Julia. With further investigations and understanding of the Julia language, we could probably diminish this constant. Anyway, Julia provides a much more generic code and we could have benchmarked the code for $p = 2^{127} - 1$ without changing a single line. On top of that, Julia provides a variety of tools that makes life easier when coding : the benchmark package, the package manager, a high-level syntax. . . It will be up to reader to decide if the price of the constant factor 8 is too much.

Any error in this report or in the code of `FastArithmetic.jl` can be signaled by e-mail (edouard.rousseau@u-psud.fr). Finally, this work would not have been done without the help of Luca DE FEO, who took the time to answer all the naive questions he received.

Chapitre 7

Projet de M2

7.1 Préambule

This is the report of a C project about the generation of normal elements in finite fields. Consider a field extension $\mathbb{F}_{p^d}/\mathbb{F}_p$, we say that $\alpha \in \mathbb{F}_{p^d}$ is normal if $\{\alpha, \alpha^p, \alpha^{p^2}, \dots, \alpha^{p^{d-1}}\}$ is a basis of \mathbb{F}_{p^d} over \mathbb{F}_p . We first give some theory to characterize normal elements, then we describe three algorithms to compute normal elements : a randomized algorithm, Lüneburg's algorithm, and Lenstra's algorithm. Finally, we give experimental results about our implementation. All this work is based on Gao's PhD thesis [17].

7.2 Introduction

7.2.1 Normal bases

With the rise of electronics and computer science, areas like cryptography and coding theory have been largely studied. In both these domains, finite fields often have a fundamental role. Normal bases can be used to implement efficiently the finite fields arithmetic in the hardware, consuming less power than other bases. But normal bases can also be used as a theoretic tool to understand field extensions, and it is probably why normal bases were studied since the 19th century. Gauss, for example, used normal bases to study when regular polygons can be drawn with ruler and compass alone, a *very* old problem. The notion of normal bases is not linked with finite fields, if \mathbb{K} is a field and \mathbb{L} is a finite Galois extension of \mathbb{K} of Galois group G , a normal basis of \mathbb{L} over \mathbb{K} is a basis of the form $\{\sigma(\alpha) \mid \sigma \in G\}$ where $\alpha \in \mathbb{L}$. In other words, it is a basis composed of an element α and all its conjugates. In the case of finite fields, the definition that we give is indeed the same as this one, since, in this case, G is generated by the Frobenius automorphism. Given a finite Galois extension \mathbb{L}/\mathbb{K} , normal bases can also be used to realize the Galois correspondence of \mathbb{L}/\mathbb{K} . Theory tells us that there exists a correspondence

between intermediate extensions of \mathbb{L}/\mathbb{K} and subgroups of G , but it does not give an *effective* way of realising that correspondence. Normal bases are a way to solve that problem. In this report, we focus on the problem of finding normal elements in finite fields. That is why, from now on, we do not look at cases other than finite extensions of finite fields (*i.e.* extensions of type $\mathbb{F}_{q^d}/\mathbb{F}_q$).

7.2.2 Recalls and notations

In all the text, we denote by \mathbb{F}_n the field with n elements. We recall that n must be a *prime power* (*i.e.* $n = p^d$ where p is a prime number and d is a positive number). We have $\mathbb{F}_{p^d} \cong \mathbb{F}_p[X]/(P)$, where P is an irreducible polynomial of degree d in $\mathbb{F}_p[X]$, and this is the representation that will be used in the C code. From now, we note X both for the indeterminate X and for its image \bar{X} in the quotient $\mathbb{F}_p[X]/(P)$. \mathbb{F}_{p^d} is a vector space over \mathbb{F}_p , of dimension d , the basis $\{1, X, X^2, \dots, X^{d-1}\}$ is called the *polynomial basis* in the following. We also recall that \mathbb{F}_{p^d} is a *field extension* of \mathbb{F}_p and the characteristic of \mathbb{F}_{p^d} is p . Lastly, we denote by σ the *Frobenius map*, defined by $\sigma(x) = x^p$ for all $x \in \mathbb{F}_{p^d}$. This map is a \mathbb{F}_p -automorphism of \mathbb{F}_{p^d} (*i.e.* σ is a field morphism, a bijection, and $\forall y \in \mathbb{F}_p, \sigma(y) = y$), as a consequence, σ is also a linear map.

Note that we look only at *prime extensions*, (*i.e.* extensions of type $\mathbb{F}_{p^d}/\mathbb{F}_p$). This choice has been made because the theory of normal elements in extensions of type $\mathbb{F}_{q^n}/\mathbb{F}_q$ is not different (replacing p by q in the following demonstrations would be sufficient), but the implementation of the algorithms is simpler in the case of prime extensions. Speaking about implementation, we must introduce Flint (Fast Library for Number Theory), because *every* function wrote in this project uses Flint.

7.2.3 Flint

Flint [20] is a C library maintained by William Hart, and developed by him and many others since 2007. In Flint, we use the type `fq_t`, where the elements of $\mathbb{F}_{p^d} \cong \mathbb{F}_p[X]/(P)$ are represented as polynomials of degree less than d . The underlying data structure is `fmpz_poly_t`, that is a `struct` containing

1. a pointer to arbitrary large integers `fmpz`, representing the coefficients of the polynomial;
2. the number of memory allocations for the pointer of type `slong` : Flint's own `unsigned long`;
3. the degree of the polynomial, a `slong` too.

In Flint, it is possible to work with finite fields of arbitrary large order and with polynomials or matrices over these fields. The basic functions are

already implemented, such as gcd or derivative for polynomials, or reduced row echelon form for matrices. Using Flint save a lot of time, but it also has its limits. It is possible to deal with arbitrary *prime* extensions. For extension of type $\mathbb{F}_{q^n}/\mathbb{F}_q$, it is still possible, using polynomial arithmetic over \mathbb{F}_q . But using polynomials over \mathbb{F}_{q^n} would have required polynomial in two indeterminates, and everything would be more complicated. That is the reason we look only at prime extensions.

All these recalls being made, and the true hero (Flint) of our functions being introduced, we can begin our journey.

7.3 Basics on normal bases

In all this section, we set p a prime number and d a positive number. We work with the extension $\mathbb{F}_{p^d}/\mathbb{F}_p$ and the Frobenius morphism σ defined above. We are now able to give a formal definition of a normal element and a normal basis.

Définition 7.3.1 (normal element, normal basis). Let $\alpha \in \mathbb{F}_{p^d}$, we say that α is a *normal element* if $\{\alpha, \sigma(\alpha), \dots, \sigma^{d-1}(\alpha)\} = \{\alpha, \alpha^p, \dots, \alpha^{p^{d-1}}\}$ is a basis of \mathbb{F}_{p^d} . Such a basis is called a *normal basis*.

In order to recognize a normal element α , we can compute the dimension of the linear span of $\{\sigma^i(\alpha) \mid 0 \leq i < d\}$. A way of doing that is to construct the matrix M which columns are the coordinates of the elements $\sigma^i(\alpha)$ in the polynomial basis, and to check if M is non-singular. This can be done using Gauss algorithm. This method is not efficient so we work on other characterizations of normal elements. We need two more definitions to be able to state our results.

Définition 7.3.2 (trace function). The *trace function* $\text{Tr}_{p^d|p} : \mathbb{F}_{p^d} \rightarrow \mathbb{F}_p$ of the extension $\mathbb{F}_{p^d}/\mathbb{F}_p$ is the function defined by

$$\text{Tr}_{p^d|p}(\alpha) = \sum_{i=0}^{d-1} \alpha^{p^i}.$$

It takes its values in \mathbb{F}_p since $(\text{Tr}_{p^d|p}(\alpha))^p = \text{Tr}_{p^d|p}(\alpha)$.

The trace is a tool that permit to know when a family of elements $\alpha_1, \dots, \alpha_d$ forms a basis of \mathbb{F}_{p^d} over \mathbb{F}_p , using the discriminant of these elements.

Définition 7.3.3 (discriminant). Let $\alpha_1, \dots, \alpha_d$ be elements in \mathbb{F}_{p^d} , the

discriminant $\Delta(\alpha_1, \dots, \alpha_d)$ of these elements is the determinant

$$\Delta(\alpha_1, \dots, \alpha_d) = \det \begin{pmatrix} \text{Tr}(\alpha_1\alpha_1) & \text{Tr}(\alpha_1\alpha_2) & \dots & \text{Tr}(\alpha_1\alpha_d) \\ \text{Tr}(\alpha_2\alpha_1) & \text{Tr}(\alpha_2\alpha_2) & \dots & \text{Tr}(\alpha_2\alpha_d) \\ \vdots & \vdots & & \vdots \\ \text{Tr}(\alpha_d\alpha_1) & \text{Tr}(\alpha_d\alpha_2) & \dots & \text{Tr}(\alpha_d\alpha_d) \end{pmatrix}$$

where Tr is the same trace as before. We omit the indices because the extension is always the same.

With this last tool, we can state our first theorem.

Théorème 7.3.4 (Theorem 2.2.1, [17])

For any n elements $\alpha_1, \dots, \alpha_d$ in \mathbb{F}_{p^d} , they form a basis of \mathbb{F}_{p^d} over \mathbb{F}_p if and only if $\Delta(\alpha_1, \dots, \alpha_d) \neq 0$.

Démonstration. First assume that $\alpha_1, \dots, \alpha_d$ form a basis of \mathbb{F}_{p^d} over \mathbb{F}_p . We prove that $\Delta(\alpha_1, \dots, \alpha_d) \neq 0$ by showing that the row vectors L_1, \dots, L_d of the matrix in the definition of $\Delta(\alpha_1, \dots, \alpha_d)$ are linearly independent over \mathbb{F}_p , and thus the matrix is nonsingular. Suppose that there exists $c_1, \dots, c_d \in \mathbb{F}_p$ with $\sum_i c_i L_i = 0$, it means

$$c_1 \text{Tr}(\alpha_1\alpha_j) + \dots + c_n \text{Tr}(\alpha_d\alpha_j) = 0 \text{ for } 1 \leq j \leq d.$$

Then with $\beta = c_1\alpha_1 + \dots + c_d\alpha_d$, by linearity of the trace function, we get $\text{Tr}(\beta\alpha_j) = 0$ for $1 \leq j \leq d$. Since $\alpha_1, \dots, \alpha_d$ is a basis of \mathbb{F}_{p^d} , it follows that $\text{Tr}(\beta\alpha) = 0$ for all $\alpha \in \mathbb{F}_{p^d}$. If $\beta \neq 0$, it means that $\text{Tr}(\gamma) = 0$ for all $\gamma \in \mathbb{F}_{p^d}$: we can see that this is impossible by thinking of $\text{Tr}(\gamma)$ as the trace of the multiplication-by- γ linear map. So we have $\beta = 0$, and then $c_1\alpha_1 + \dots + c_d\alpha_d = 0$ implies $c_1 = \dots = c_d = 0$.

Conversely, assume that $\Delta(\alpha_1, \dots, \alpha_d) \neq 0$ and $c_1\alpha_1 + \dots + c_d\alpha_d = 0$ for some $c_1, \dots, c_d \in \mathbb{F}_p$. By multiplying by α_j , we have

$$c_1\alpha_1\alpha_j + \dots + c_d\alpha_d\alpha_j \text{ for } 1 \leq j \leq d,$$

and by applying the trace function, we get

$$c_1 \text{Tr}(\alpha_1\alpha_j) + \dots + c_n \text{Tr}(\alpha_d\alpha_j) \text{ for } 1 \leq j \leq d.$$

But this is the same as $\sum c_i L_i = 0$ where the L_i are the rows of the matrix in $\Delta(\alpha_1, \dots, \alpha_d)$. Since this matrix is nonsingular by hypothesis, its rows are linearly independent and it follow that $c_1 = \dots = c_d$. Therefore $\alpha_1, \dots, \alpha_d$ are linearly independant over \mathbb{F}_p , and form a family of d vectors in a d -dimensional vector space, so $\alpha_1, \dots, \alpha_d$ form basis of \mathbb{F}_{p^d} over \mathbb{F}_p . \square

This characterization is interesting in itself, but the matrix defined in $\Delta(\alpha_1, \dots, \alpha_d)$ is quite complicated. Fortunately, we can use a much simpler matrix.

Corollaire 7.3.5

The elements $\alpha_1, \dots, \alpha_d$ form a basis of \mathbb{F}_{p^d} over \mathbb{F}_p if and only if the matrix A is nonsingular, where

$$A = \begin{pmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_d \\ \alpha_1^p & \alpha_2^p & \dots & \alpha_d^p \\ \vdots & \vdots & & \vdots \\ \alpha_1^{p^{d-1}} & \alpha_2^{p^{d-1}} & \dots & \alpha_d^{p^{d-1}} \end{pmatrix}$$

Démonstration. We have $\Delta(\alpha_1, \dots, \alpha_d) = \det(A^t A) = (\det A)^2$. □

We now have a simpler matrix to study, but it is still a matrix, so the nature of the problem is the same as before. Next lemma allow us to change the nature of the problem, by working with polynomials.

Lemme 7.3.6

For any n elements $a_0, a_1, \dots, a_{d-1} \in \mathbb{F}_{p^d}$, the $d \times d$ circulant matrix

$$c[a_0, a_1, \dots, a_{d-1}] = \begin{pmatrix} a_0 & a_1 & a_2 & \dots & a_{d-1} \\ a_{d-1} & a_0 & a_1 & \dots & a_{d-2} \\ a_{d-2} & a_{d-1} & a_0 & \dots & a_{d-3} \\ \vdots & \vdots & \vdots & & \vdots \\ a_1 & a_2 & a_3 & \dots & a_0 \end{pmatrix}$$

is nonsingular if and only if the polynomial $\sum a_i X^i$ is relatively prime to $X^d - 1$.

Démonstration. Let A be the following $d \times d$ permutation matrix

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}.$$

Then we see that $c[a_0, a_1, \dots, a_{d-1}] = \sum_{i=0}^{d-1} a_i A^i = f(A)$, where $f(X) = \sum_{i=0}^{d-1} a_i X^i$. On the first line of A^i , there is only one nonzero value, it is a 1 in position $i + 1$. Hence the matrices A^0, A, \dots, A^{d-1} , are linearly independant, and since $A^d = I_d$, we know that the minimal polynomial of A is $X^d - 1$. Assume that $f(X)$ is relatively prime to $X^d - 1$. Then there are polynomials $a(X), b(X)$ such that

$$a(X)f(X) + b(X)(X^d - 1) = 1,$$

and so

$$a(A)f(A) = I_d,$$

as $A^d - I_d = 0$. This implies that $f(A)$ is invertible and so nonsingular. Now assume that $f(X)$ and $X^d - 1$ are not relatively prime, and note $d(X) \neq 1$ their gcd. Let $f(X) = f_1(X)d(X)$ and $X^d - 1 = h(X)d(X)$. Since $\deg h < n$, we have $h(A) \neq 0$. But $h(A)d(A) = 0$, so we see that $d(A)$ is singular. Therefore $f(A) = f_1(A)d(A)$ is also singular. We have shown that $f(A)$ is singular if and only if $f(X)$ is relatively prime to $X^d - 1$. \square

We are now able to state the last result oh this section.

Théorème 7.3.7 (Hensel, 1888)

Let $\alpha \in \mathbb{F}_{p^d}$, α is a normal element of the extension $\mathbb{F}_{p^d}/\mathbb{F}_p$ if and only if the polynomial $\alpha^{p^{d-1}}X^{d-1} + \dots + \alpha^p X + \alpha \in \mathbb{F}_{p^d}[X]$ is relatively prime to $X^d - 1$.

Démonstration. α is a normal element, if, by definition, the elements $\alpha, \alpha^p, \dots, \alpha^{p^{d-1}}$ form a basis of \mathbb{F}_{p^d} over \mathbb{F}_p . By Corollary 7.3.5, this is true if and only if the matrix

$$A = \begin{pmatrix} \alpha & \alpha^p & \alpha^{p^2} & \dots & \alpha^{p^{d-1}} \\ \alpha^p & \alpha^{p^2} & \alpha^{p^3} & \dots & \alpha \\ \vdots & \vdots & \vdots & & \vdots \\ \alpha^{p^{d-1}} & \alpha & \alpha^p & \dots & \alpha^{p^{d-2}} \end{pmatrix}$$

is nonsingular. But, reversing the order of the rows in A , from the second row to the last, we get the matrix $c[\alpha, \alpha^p, \dots, \alpha^{p^{d-1}}]$, that is nonsingular if and only if A is nonsingular. By Lemma 7.3.6, $c[\alpha, \alpha^p, \dots, \alpha^{p^{d-1}}]$ is nonsingular if and only if $X^d - 1$ and $\alpha^{p^{d-1}}X^{d-1} + \dots + \alpha^p X + \alpha$ are relatively prime. This proves the theorem. \square

We now have a new characterization of normal elements. We can decide if an element is normal or not by computing a gcd. It is more efficient than the naive matrix method because we have efficient algorithms to compute the gcd. The function `is_normal` is exactly the implementation of this method. The code of this function, and of all the others, is available at github.com/erou/normalBases.

7.4 Computation of normal bases

Before trying to compute normal elements, a natural question is to wonder if normal elements always exist in extensions of type $\mathbb{F}_{p^d}/\mathbb{F}_p$. We do not prove this result here (it is done in [17]), but the answer is yes. Now that this natural fear has been eliminated, we give some ways to compute normal elements. We use the same notations as in the previous section.

7.4.1 Randomized algorithm

First, we give a randomized algorithm. Such algorithms are often based on the same strategy.

1. Take a random element, following a certain protocol.
2. Check if this element verify the wanted property.

Our algorithm also uses this strategy, that is why our first function `is_normal` is very important. But the protocol is also essential. For example, if we just take an element completely at random in \mathbb{F}_{p^d} , our algorithm will not be very efficient. The following theorem helps us defining a better protocol.

Théorème 7.4.1

Let $f(X)$ be an irreducible polynomial of degree d over \mathbb{F}_p and α a root of $f(X)$. Let

$$g(X) = \frac{f(X)}{(X - \alpha)f'(\alpha)}.$$

Then there are at least $p - d(d - 1)$ elements u in \mathbb{F}_p such that $g(u)$ is a normal element of \mathbb{F}_{p^d} over \mathbb{F}_p .

Démonstration. Let $\sigma_i = \sigma^i$ be the automorphism $\theta \rightarrow \theta^{p^i}$, $\theta \in \mathbb{F}_{p^d}$, for $0 \leq i < d$. Then $\alpha_i = \sigma_i(\alpha)$ is also a root of $f(X)$, for $0 \leq i < d$. The automorphism $\sigma_i : \mathbb{F}_{p^d} \rightarrow \mathbb{F}_{p^d}$ can be extended into a ring morphism $\sigma_i : \mathbb{F}_{p^d}[X] \rightarrow \mathbb{F}_{p^d}[X]$ by setting $\sigma_i(X) = X$. We get

$$g_i(X) := \sigma_i(g(X)) = \frac{f(X)}{(X - \alpha_i)f'(\alpha_i)},$$

and we note that $\sigma_i\sigma_i(g(X)) = \sigma_{i+j}(g(X))$. Then $g_i(X)$ is a polynomial in $\mathbb{F}_{p^d}[X]$ having α_k as a root for $k \neq i$ and $g_i(\alpha_i) = 1$. Hence, for $i \neq k$, we have that every root of $f(X)$ is also a root of $g_i(X)g_k(X)$, so

$$g_i(X)g_k(X) \equiv 0 \pmod{f(X)}, \text{ for } i \neq k. \quad (7.1)$$

Note that

$$g_1(X) + g_2(X) + \cdots + g_d(X) - 1 = 0, \quad (7.2)$$

since the left side is a polynomial of degree at most $d - 1$ (all the g_i are of degree $d - 1$) and has $\alpha_0, \alpha_1, \dots, \alpha_{d-1}$ as roots. Multiplying (7.2) by $g_i(X)$ and using (7.1) to simplify, we have

$$g_i(X)^2 \equiv g_i(X) \pmod{f(x)}. \quad (7.3)$$

We next set the matrix

$$D = (\sigma_{i+j}(g(X)))_{0 \leq i, j < d}$$

and we study its determinant, $\Delta(X)$. From the equations (7.1), (7.2) and (7.3), we see that the entries of $D^t D$ modulo $f(X)$ are all 0, except on the main diagonal where they are all 1. It follows that

$$\Delta(X)^2 = \det(D^t D) \equiv 1 \pmod{f(x)}.$$

This proves that $\Delta(X)$ is a nonzero polynomial of degree at most $d(d-1)$, since it is a sum of d products of polynomials of degree $d-1$. Therefore $\Delta(X)$ has at most $d(d-1)$ roots in \mathbb{F}_p . The result follows from Corollary 7.3.5, since the matrix $D(u)$ is exactly the one defined in the corollary applied to the elements $g(u), \sigma(g(u)), \dots, \sigma^{d-1}(g(u))$. \square

Now we have our protocol. We take $u \in \mathbb{F}_p$ at random, then we check if $g(u)$ is normal. If p is large enough, for example $p > 2d(d-1)$, then $g(u)$ is normal with probability at least $1/2$. We will not discuss it, but the entire computation takes $O((n + \log q)(n \log q)^2)$ bit operations. A reference for this result is given in [17]. As always, we are not *completely* satisfied with randomized algorithm, so we give two deterministic algorithms.

7.4.2 Deterministic algorithms

First, we define the σ -Order of an element $\theta \in \mathbb{F}_{p^d}$, both Lenstra's and Lüneburg's algorithms use it.

The σ -Order polynomials

Définition 7.4.2 (σ -Order). Let $\theta \in \mathbb{F}_{p^d}$ be an arbitrary element. Let k be the least positive integer such that $\sigma^k(\theta) = \theta^{p^k}$ belongs to \mathbb{F}_p -linear span of $\{\sigma^i \theta \mid 0 \leq i < k\}$. If $\sigma^k \theta = \sum_{i=0}^{k-1} c_i \sigma^i \theta$ for that k , then the σ -Order of θ is the polynomial

$$\text{Ord}_\theta(X) = X^k - \sum_{i=0}^{k-1} c_i X^i.$$

The σ -Order polynomials are widely used in our functions. We compute them using Gauss algorithm and the row reduced echelon form. To know if $\sigma^j(\theta)$ is in the linear span of $\{\sigma^i(\theta) \mid 0 \leq i < j\}$, we construct the matrix M whose columns are the vectors $\theta, \sigma(\theta), \dots, \sigma^j(\theta)$, we compute the rank using Gauss algorithm. If the rank is $j+1$ then $\theta, \sigma(\theta), \dots, \sigma^j(\theta)$ are linearly independent. In order to compute k , we do this operation for $j = 1$ and while the vectors are linearly independent, we increase the value of j by 1. When we get that the vectors are dependant, we use the reduced row echelon form

to get a $d \times (k + 1)$ matrix of type

$$\begin{pmatrix} 1 & 0 & 0 & \dots & 0 & c_0 \\ 0 & 1 & 0 & \dots & 0 & c_1 \\ 0 & 0 & 1 & \dots & 0 & c_2 \\ \vdots & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \dots & 1 & c_{k-1} \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}$$

where the coefficients c_i are in the last column. This strategy is implemented in the function `sigma_order`.

We also see that an element $\alpha \in \mathbb{F}_{p^d}$ is normal if and only if $\text{Ord}_\alpha(X) = X^d - 1$. This characterization is the same as the definition, thus it is not very interesting. The following property is used in further demonstrations.

Proposition 7.4.3

Let $P \in \mathbb{F}_p[X]$, if $P(\sigma)\zeta = 0$, then P is divisible by Ord_ζ .

Démonstration. Let I_ζ be the set defined by $I_\zeta = \{P \in \mathbb{F}_p[X] \mid P(\sigma)\zeta = 0\}$. This is an ideal of $\mathbb{F}_p[X]$, so I_ζ is principal. The polynomial Ord_ζ belongs to I_ζ and has minimum degree in I_ζ by definition. Hence $I_\zeta = (\text{Ord}_\zeta)$. \square

We always have $(X^d - 1)(\sigma)\theta = 0$, so for any $\theta \in \mathbb{F}_{p^d}$, we have that $X^d - 1$ is divisible by Ord_θ . The result below is also used both in Lenstra's and Lüneburg's algorithm.

Proposition 7.4.4

Let α and β be two elements in \mathbb{F}_{p^d} . Then we have

$$\text{Ord}_{\alpha+\beta} \mid \text{lcm}(\text{Ord}_\alpha, \text{Ord}_\beta).$$

Démonstration. Let $P = \text{lcm}(\text{Ord}_\alpha, \text{Ord}_\beta)$. By linearity, we have

$$P(\sigma)(\alpha + \beta) = P(\sigma)\alpha + P(\sigma)\beta = 0 + 0 = 0.$$

It follows from Proposition 7.4.3 that $\text{Ord}_{\alpha+\beta}$ divides P . \square

Corollaire 7.4.5

Let α and β be two elements in \mathbb{F}_{p^d} such that Ord_α and Ord_β are relatively prime. Then we have $\text{Ord}_{\alpha+\beta} = \text{Ord}_\alpha \text{Ord}_\beta$.

Démonstration. Since Ord_α and Ord_β are relatively prime, we have $\text{lcm}(\text{Ord}_\alpha, \text{Ord}_\beta) = \text{Ord}_\alpha \text{Ord}_\beta$. It follows from Proposition 7.4.4 that $\text{Ord}_{\alpha+\beta}$ divides $\text{Ord}_\alpha \text{Ord}_\beta$. Now, note that for any $\theta \in \mathbb{F}_{p^d}$, $\text{Ord}_{-\theta} = \text{Ord}_\theta$, then we also have

$$\text{Ord}_\alpha \mid \text{lcm}(\text{Ord}_{\alpha+\beta}, \text{Ord}_\beta)$$

$$\text{Ord}_\beta \mid \text{lcm}(\text{Ord}_{\alpha+\beta}, \text{Ord}_\alpha).$$

Since Ord_α and Ord_β are relatively prime, it follows that

$$\text{Ord}_\alpha \mid \text{Ord}_{\alpha+\beta}$$

$$\text{Ord}_\beta \mid \text{Ord}_{\alpha+\beta}.$$

Therefore, $\text{Ord}_\alpha \text{Ord}_\beta$ divides $\text{Ord}_{\alpha+\beta}$. Both polynomial are monic so we have the wanted equality. \square

Lenstra's algorithm

Lenstra's algorithm is based on linear algebra, so our implementation is based on the matrices module `fq_mat` of Flint. Before describing the algorithm, we need two lemmas.

Lemme 7.4.6

Let $\theta \in \mathbb{F}_{p^d}$ with $\text{Ord}_\theta(X) \neq X^d - 1$. Let $g(X) = (X^d - 1)/\text{Ord}_\theta(X)$. Then there exists $\beta \in \mathbb{F}_{p^d}$ such that

$$g(\sigma)\beta = \theta. \tag{7.4}$$

Démonstration. Let γ be a normal element of \mathbb{F}_{p^d} over \mathbb{F}_p . Then, by definition of being a normal element, there exists a polynomial $f(X) \in \mathbb{F}_p[X]$ such that $f(\sigma)\gamma = \theta$. Since $\text{Ord}_\theta(\sigma)\theta = 0$, we have $(\text{Ord}_\theta(\sigma)f(\sigma))\gamma = 0$. So, by Proposition 7.4.3, $\text{Ord}_\theta(X)f(X)$ is divisible by $X^d - 1$. We have $g(X) = (X^d - 1)/\text{Ord}_\theta(X)$ and $X^d - 1 \mid \text{Ord}_\theta(X)f(X)$, so $g(X) \mid f(X)$. Let $f(X) = g(X)h(X)$. Then

$$g(\sigma)(h(\sigma)\gamma) = \theta.$$

This proves that $\beta = h(\sigma)\gamma$ is a solution of (7.4). \square

Lemme 7.4.7

Let $\theta \in \mathbb{F}_d$ with $\text{Ord}_\theta(X) \neq X^d - 1$. Assume that there exists a solution β of (7.4) such that $\deg \text{Ord}_\beta \leq \deg \text{Ord}_\theta$. Then there exists a nonzero element $\zeta \in \mathbb{F}_{p^d}$ such that

$$g(\sigma)\zeta = 0, \tag{7.5}$$

where $g(X) = (X^d - 1)/\text{Ord}_\theta(X)$. Moreover any such ζ has the property that

$$\deg \text{Ord}_{\theta+\zeta} > \deg \text{Ord}_\theta \tag{7.6}$$

Démonstration. Let γ be a normal element in \mathbb{F}_{p^d} over \mathbb{F}_p . We can see that $\zeta = \text{Ord}_\theta(\sigma)\gamma \neq 0$ is a solution of (7.5). In fact, we prove that (7.6) is true for any solution ζ of (7.5). From (7.4) and Proposition 7.4.3, it follows that Ord_θ divides Ord_β , so the hypothesis that $\deg \text{Ord}_\beta \leq \deg \text{Ord}_\theta$ implies that $\text{Ord}_\beta = \text{Ord}_\theta$. It follows that g and Ord_θ are relatively prime : suppose

$\gcd(g, \text{Ord}_\theta) = d$ and $\deg d > 0$, let $\text{Ord}_\theta = \text{Ord}'_\theta d = \text{Ord}_\beta$ and $g = g'd$, then $\text{Ord}'_\theta(\sigma)\theta = (\text{Ord}'_\theta g)(\sigma)\beta = (\text{Ord}_\beta g')(\sigma)\beta = 0$ and $\deg \text{Ord}'_\theta < \deg \text{Ord}_\beta$, that is a contradiction. Since Ord_ζ is a divisor of g , Ord_ζ and Ord_θ are also relatively prime. Therefore, by Corollary 7.4.5

$$\text{Ord}_{\theta+\zeta} = \text{Ord}_\theta \text{Ord}_\zeta,$$

and then (7.6) follows from the fact that $\zeta \neq 0$ and then $\deg \text{Ord}_\zeta \geq 1$. \square

We now have a deterministic algorithm to compute a normal element of \mathbb{F}_{p^d} over \mathbb{F}_p .

1. Take an element $\theta \in \mathbb{F}_{p^d}$ and compute Ord_θ .
2. If $\text{Ord}_\theta(X) = X^d - 1$ then the algorithm stops (θ is normal).
3. Calculate $g(X) = (X^d - 1)/\text{Ord}_\theta$ and then find $\beta \in \mathbb{F}_{p^d}$ such that $g(\sigma)\beta = \theta$.
4. Determine Ord_β . If $\deg \text{Ord}_\beta > \deg \text{Ord}_\theta$ then replace θ by β and go to 2; otherwise if $\deg \text{Ord}_\beta \leq \deg \text{Ord}_\theta$ then find a nonzero element ζ such that $g(\sigma)\zeta = 0$, replace θ by $\theta + \zeta$ and determine the σ -Order of the new θ , then go to 2.

This algorithm ends because with each replacement of θ , the degree of Ord_θ increases by at least one, by Lemma 7.4.7, and because this degree is bounded by d since the only possible σ -Order of degree d is $X^d - 1$. The existence of the elements β of step 3 is guaranteed by Lemma 7.4.6 and the existence of ζ of step 4 by Lemma 7.4.7. Since $g(\sigma)$ is a linear map, we compute β and ζ , using Gauss algorithm and reduced row echelon form again. Computing elements of type $g(\sigma)\theta$ is not just polynomial evaluation, and is widely used in our algorithms, so we have a special function `sigma_composition` to do it. The rest of the algorithm is essentially linear algebra, it uses Flint matrices and is in the function `lenstra`.

Lüneburg's algorithm

Lüneburg algorithm is more elementary, it is not based on a powerful theory like linear algebra. Let f be an irreducible polynomial of degree d over \mathbb{F}_p and α a root of f . Then $\{1, \alpha, \dots, \alpha^{d-1}\}$ is a basis of \mathbb{F}_{p^d} over \mathbb{F}_p . Let $f_i = \text{Ord}_{\alpha^i}$, for $0 \leq i < d$. Let $\gamma = \sum_{i=0}^{d-1} a_i \alpha^i$ be a normal element of \mathbb{F}_{p^d} over \mathbb{F}_p . Note that for any $a_i \in \mathbb{F}_p$, $\text{Ord}_{a_i \alpha^i} = \text{Ord}_{\alpha^i} = f_i$, and as seen in the demonstration of Corollary 7.4.5, $\text{Ord}_\gamma = X^d - 1$ divides $\text{lcm}(f_0, \dots, f_{d-1})$. Since for all i , f_i divides $X^d - 1$, we have that $\text{lcm}(f_0, \dots, f_{d-1})$ divides $X^d - 1$. Therefore $\text{lcm}(f_0, \dots, f_{d-1}) = X^d - 1$. Then we apply the *factor refinement* [26]. It is an algorithm that, given a list of polynomials f_0, \dots, f_{d-1} , compute a new list g_1, \dots, g_r of polynomials, pairwise relatively prime, such

that

$$\prod_{i=0}^{d-1} f_i = \prod_{j=1}^r g_j^{e_j}.$$

The factor refinement works that way : we first set $h_i = f_i$ and $e_i = 1$ for $0 \leq i < d$ and we consider the list $(h_i, e_i)_i$. While there exists $i \neq j$ with $p := \gcd(h_i, h_j) \neq 1$, we delete (h_i, e_i) and (h_j, e_j) from the list and we add $(p, e_i + e_j)$, $(h_i/p, e_i)$, $(h_j/p, e_j)$, except for the cases where the first entry is 1. This algorithm stops and compute what we expect. In our case, we use it to compute the pairwise relatively prime polynomials g_j ($1 \leq j \leq r$), and then we compute the integers e_{ij} such that

$$f_i = \prod_{1 \leq j \leq r} g_j^{e_{ij}}, \text{ for all } 0 \leq i < d.$$

This algorithm is implemented in the function `factor_refinement`, it uses Flint's type `fq_poly_factor` to represent the list $(h_i, e_i)_i$ and to let Flint deal with the memory allocation.

Next, for each j , $1 \leq j \leq r$, we find an index $i(j)$ such that e_{ij} is maximized. Let

$$h_j = f_{i(j)} / g_j^{e_{i(j)j}}$$

and take $\beta_j = h_j(\sigma)\alpha^{i(j)}$. Then

$$\beta = \sum_{j=1}^r \beta_j$$

is a normal element of \mathbb{F}_{p^d} over \mathbb{F}_p . In fact, the σ -Order of β_j is $g_j^{e_{i(j)j}}$ (β_j has been constructed in that purpose) for $1 \leq j \leq r$. As g_1, \dots, g_r are pairwise relatively prime, Corollary 7.4.5 states that the σ -Order of β must

$$\prod_{j=1}^r g_j^{e_{i(j)j}} = \text{lcm}(f_0, \dots, f_{d-1}) = X^d - 1,$$

meaning that β is a normal element.

To implement this algorithm, we use our functions `sigma_composition`, `sigma_order`, `factor_refinement`, and Flint's basic functions to deal with polynomials. The implementation is a translation of the steps we followed to obtain β .

7.5 Experimental results

In the source code, one can find two directories named `fq` and `fq_nmod` with very little differences between the files inside these directories. The name of the file determines the type used in the functions. The type `fq` supports arbitrary large integers p for the characteristic of the fields \mathbb{F}_{p^d} , whereas `fq_nmod` supports only small integers for p but is faster.

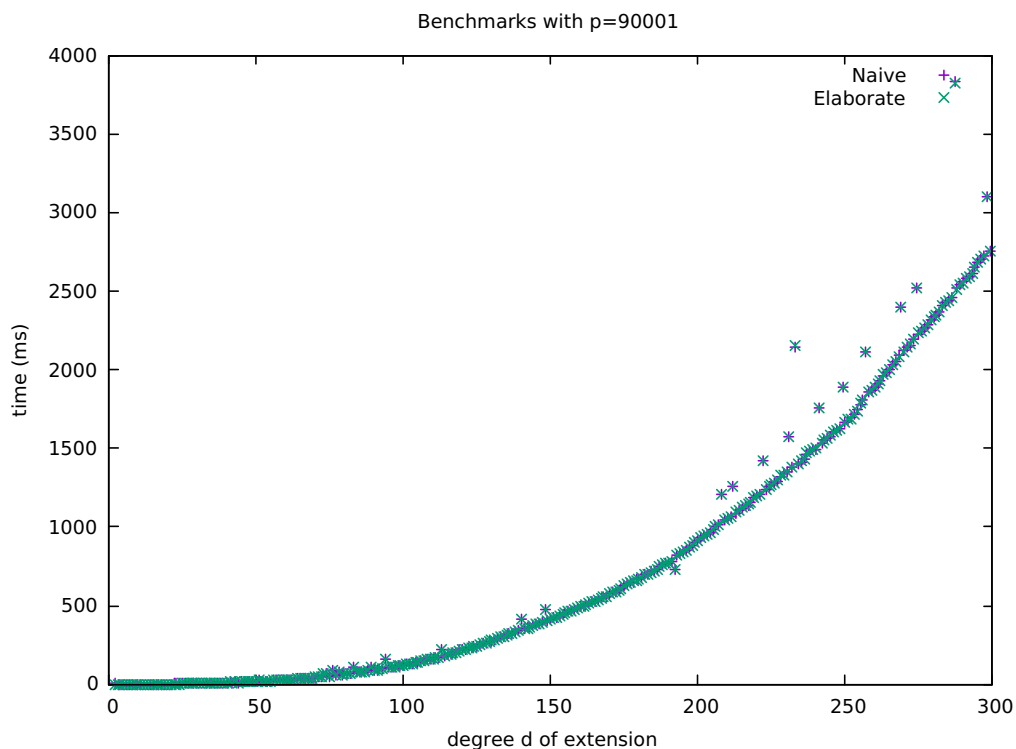


FIGURE 7.1 – Benchmark of naive and elaborate algorithms with $p = 90001$.

7.5.1 Random algorithms

We implemented the naive algorithm for computing normal elements, in order to compare this algorithm to the more elaborate one. Surprisingly, as we see in Figure 7.1, the speed is often the same, because when we chose an element randomly in \mathbb{F}_{p^d} , it is often a normal element. If the naive algorithm finds a normal element with its first pick, there is no way that the the elaborate one is faster, since it does more calculus. There are still a few cases where the naive algorithm does not find a normal element easily, and where the elaborate one is faster.

The time spent in those algorithms is essentially to time spent to perform the tests.

7.5.2 Lüneburg’s and Lenstra’s algorithms

Lenstra’s algorithm is faster than Lüneburg algorithm, though the complexity obtained seems to match with the theory in both algorithms. Also, Lenstra’s algotihm is mush more stable than Lüneburg’s, the latter can have really different behaviours for entries of the same sizes. The core of both

d	Lüneburg	Lenstra
50	1,87 s	0,65 s
114	93,0 s	11,7 s
205	290 s	187 s
249	1960 s	253 s

FIGURE 7.2 – Benchmark of Lüneburg’s and Lenstra’s algorithms with $p = 90001$.

Lüneburg’s and Lenstra’s algorithms is the computation of the σ -order polynomials.

Chapitre 8

Stage de M2

```
      _
     _  _  _  _
    (_)  | (_) (_)  | A fresh approach to technical computing
      _  _  _  _  | Documentation: http://docs.julialang.org
     | | | | | | | / _ ' | Type "?help" for help.
     | | | | | | | ( _ | | |
    _/ | \ _ ' _ | | | \ _ ' _ | Version 0.5.1 (2017-03-05 13:25 UTC)
    | _ /         | x86_64-pc-linux-gnu Official http://julialang.org/ release
```

```
julia> using DlogGF
```

Welcome to

```
      8I ,dPYb,
      8I IP' 'Yb
      8I I8 8I
      8I I8 8'
,gggg,8I I8 dP ,ggggg, ,gggg,gg 888 .oooooo. oooooooooooooo
dP' 'Y8I I8dP dP' 'Y8ggg dP' 'Y8I 888 d8P' 'Y8b '888' '8
i8' ,8I I8P i8' ,8I i8' ,8I 888 oooooo 888 '
d8, ,d8b,,d8b,_ ,d8, ,d8' d8, ,d8I '88. .88' 888
'Y8888P' 'Y88P' 'Y88P' Y8888P' 'Y8888P' 888 'Y8bood8P' o888o
      ,d8I'
      ,dP'8I
      ,8' 8I
      I8 8I
      '8, ,8I
      'Y8P'
```

8.1 Préambule

This internship took place from March to September 2017 in the Symbolic Computation Group of the University of Waterloo. We studied the recent breakthroughs in the discrete logarithm world, in the finite field of small characteristic case. As part of this study, we implemented the encountered algorithms and made them available as an open-source library written in Julia/Nemo [1, 21].

8.2 Introduction

With the rise of new technologies and new communication media, the importance of cryptography has been steadily growing. In 1976, a new era of cryptography was born, with the invention of *public-key* cryptography by Diffie and Hellman, in their article “New Directions in Cryptography” [14]. Today, cryptography is omnipresent, though mostly invisible to the user. Even if millions of executions of public-key protocols happen each second, there are only two main families those protocols are drawn from. Indeed, they are very often based either on the hardness of integer factorization (like the well known and extensively used RSA protocol), or on discrete logarithms. The Discrete Logarithm Problem (DLP) is fairly easy to state. Let $\mathcal{G} = \langle g \rangle$ be a cyclic group written multiplicatively, generated by an element g , and denote by $N = |\mathcal{G}|$ its cardinal. We have the isomorphism :

$$\begin{aligned} \exp_g : (\mathbb{Z}/N\mathbb{Z}, +) &\rightarrow \mathcal{G} \\ n &\mapsto g^n, \end{aligned}$$

and we denote by $\log_g = \exp_g^{-1}$ the inverse isomorphism (sometimes only denoted by \log). In practice, the *square and multiply* algorithm allows us to compute $g^n = \exp_g(n)$ efficiently, *i.e.* in polynomial time in the bitsize of n . But, given $y = g^k$, the computation of $k = \log_g(y)$ is not as easy. This kind of function f , where f is easy to compute but f^{-1} is (presumed to be) hard to compute, is called *one-way* function. It is typically used in cryptology to make the encryption fast and the deciphering slow.

In fact, discrete logarithms were studied long before their utilization in cryptography. Indeed, Gauß, in its *Disquisitiones Arithmeticae*, back in 1801, was already computing discrete logarithms, that he referred to as *indices*. Nevertheless, the discrete logarithm really became a practically relevant problem with the invention of the Diffie-Hellman key exchange protocol [14]. The security of the protocol originally relied on the hardness of the discrete logarithm problem in $(\mathbb{Z}/N\mathbb{Z})^\times$. This group is no longer secure, but other interesting groups can be used, such as the points of an elliptic curve or the multiplicative group of a finite field. In this internship, we focused on that last case. Over the years, two types of algorithms emerged :

- the *generic* algorithms, that can be used for any group, with an exponential complexity of type $O(\sqrt{N})$, where $N = |\mathcal{G}|$ is the cardinal of the group ;
- the *index calculus* algorithms, that are built on the structure of the group considered, and that have very different complexities, depending on the kind of group.

To express the complexity of an algorithm, one typically uses the notation

$$L_N(\alpha, c) = \exp((c + o(1))(\log N)^\alpha (\log \log N)^{1-\alpha})$$

where $\alpha \in [0, 1]$, $c > 0$, and \log represents the natural logarithm. We may omit the subscript N if there is no ambiguity, or even write $L(\alpha)$ to indicate $L(\alpha, c)$ for some constant $c > 0$, if we do not want to specify the constant. This notation can be seen as an interpolation between the polynomial complexity (in $\log N$) $L(0) = (\log N)^{c+o(1)}$ and the exponential complexity $L(1) = N^{c+o(1)}$. An algorithm with complexity $L(\alpha)$, for $0 < \alpha < 1$, is said to have *sub-exponential* complexity.

When Diffie and Hellman wrote their article in 1976, the known algorithms had exponential complexities. The first sub-exponential algorithm for the discrete logarithm in finite fields was analyzed in 1979 by Adleman [3], it has complexity $L(1/2)$. In 1984, Coppersmith designed an algorithm to handle the case of binary fields [12] with complexity $L(1/3)$. Other algorithms were invented, generalized, or improved, among them the FFS (Function Field Sieve) and NFS (Number Field Sieve), so that in 2006, every type of finite field was covered with complexity $L(1/3)$. This complexity remains the best in medium and large characteristic. In the small characteristic case, however, there has been a series of dramatic improvements that leave us with *quasi-polynomial* algorithms. If we denote by l the bitsize of the finite field in which we work, a quasi-polynomial complexity is of the form

$$l^{O(\log l)}.$$

This complexity is smaller than $L(\varepsilon)$ for any $\varepsilon > 0$, but larger than the polynomial complexity $L(0)$, so it is sometimes denoted as $L(o(1))$.

The recent algorithms that achieve quasi-polynomial complexity are index calculus algorithms, so we introduce this method in Section 8.3. We recall the new ideas that led to a new family of algorithms achieving quasi-polynomial complexity, the so-called Frobenius representation algorithms, in Section 8.4. We study the first quasi-polynomial algorithm, due to Barbulescu, Gaudry, Joux and Thomé in Section 8.5. In particular we show how to compute the logarithms of the factor base in this section. We study the second quasi-polynomial algorithm, due to Granger, Kleinjung and Zumbrägel, which is more efficient in practice, in Section 8.6. Finally, we give our experimental results in Section 8.7.

8.3 The index calculus method

Here again, we consider a group $\mathcal{G} = \langle g \rangle$, an element $g^k = h \in \mathcal{G}$, and we want to compute k . The index calculus algorithms always follow the same pattern :

0. we choose a subset $\mathcal{F} \subset \mathcal{G}$ such that $\langle \mathcal{F} \rangle = \mathcal{G}$ (we often have $g \in \mathcal{F}$), this subset is called the *factor base* ;
1. we generate multiplicative relations between the elements of \mathcal{F} , *i.e.* we find $f_1, \dots, f_n \in \mathcal{F}$ and $e_1, \dots, e_n \in \mathbb{Z}$ such that $\prod_i f_i^{e_i} = 1$, that is equivalent to the linear equation $\sum_i e_i \log_g(f_i) = 0$;
2. we solve the linear system with unknowns $\log_g(f_i)$ arising from step 1 to obtain $\log(f)$ for all $f \in \mathcal{F}$;
3. we find a multiplicative relation between h and elements of \mathcal{F} , or equivalently, we express $\log_g h = k$ as a linear combination of the $\log_g(f_i)$.

As a first example, we introduce Adleman's algorithm. We let $\mathcal{G} = \mathbb{F}_p^\times$ be the multiplicative group of a prime field $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$, and we let our factor base be

$$\mathcal{F} = \{f \mid f \leq B, f \text{ prime}\}$$

where $B \in \mathbb{N}$ is an integer that has yet to be determined, and where we make the abuse of denoting by f both the integer $f \in \mathbb{N}$ and its class $\bar{f} \in \mathbb{F}_p$. We assume that $g \in \mathcal{F}$, otherwise we add g to \mathcal{F} . In order to generate multiplicative relations, we take a random $e \in \mathbb{Z}/(p-1)\mathbb{Z}$ and we compute g^e . We say that e yields a relation if the lift $g^e \in \mathbb{N}$ is B -smooth, *i.e.* if g^e has only prime divisors $\leq B$. In that case we have a relation

$$g^e \equiv \prod_{f \in \mathcal{F}} f^{e_f} \pmod{p}$$

in the integers, that becomes

$$g^e = \prod_{f \in \mathcal{F}} f^{e_f}$$

in \mathbb{F}_p (with the same abuse of notation), and where the exponents are just some integers $e_f \in \mathbb{N}$. In terms of the logarithms, we write this relation

$$e = \sum_{f \in \mathcal{F}} e_f \log f.$$

Once we have enough relations, meaning more than $|\mathcal{F}|$, we are able to recover the unknowns $\log f$ using classic linear algebra algorithms. Note that the system is usually sparse, since the relations hardly contain all the elements $f \in \mathcal{F}$. If the target element is some $h \in \mathcal{G}$, we proceed in the same way to

recover $\log h$. We take some $e \in \mathbb{Z}/(p-1)\mathbb{Z}$ at random and compute hg^e . Once again, if the lift of hg^e is B -smooth, we have an equation of type

$$\log h + e = \sum_{f \in \mathcal{F}} h_f \log f$$

where the $h_f \in \mathbb{N}$ are integers and all the $\log f$ are now known, so we are able to recover the logarithm of h . The larger B is, the easier it is to obtain multiplicative relations. But at the same time, the size of \mathcal{F} will also become larger and we will need more multiplicative relations to compute all the unknowns $\log f$. Adleman [3] showed that with a suitable choice of B , this algorithm has an $L(1/2)$ complexity.

Even if the pattern is almost identical in every index calculus algorithm, there are a lot of available strategies to generate multiplicative relations and express the target element as a combination of elements in the factor base. This has led to a rich variety of algorithms, among which are the quasi-polynomial algorithms. We introduce them briefly in the next section.

8.4 Quasi-polynomial algorithms

There are two quasi-polynomial algorithms, the BGJT algorithm [6] and the powers-of-2 algorithm [18], they are respectively studied in Section 8.5 and 8.6. They were published almost at the same time, in 2013 and 2014, and are independent. The main ideas for these algorithms were already present in a previous work of Joux [23], published earlier in 2013 and leading to an algorithm with complexity $L(1/4 + o(1))$. Before giving those ideas, we recall the general setup in which we perform the algorithms.

8.4.1 Setup

Let $\mathcal{G} = (\mathbb{F}_{q^n})^\times$ be the multiplicative group of a finite field of small characteristic \mathbb{F}_{q^n} , where $q = p^l$ is a prime power. By *small characteristic* we mean that q must be a polynomial in the bitsize of the field q^n , *i.e.* q is a polynomial in $\log(q^n)$. Asymptotically, we want $q = L_{q^n}(0)$. In other documents, the term “small characteristic” may indicate that $q = L_{q^n}(\alpha)$ for $\alpha < \frac{1}{3}$, but this assumption would break the quasi-polynomial nature of these algorithms. We represent elements of \mathbb{F}_{q^n} by polynomials over \mathbb{F}_q of degree less than n . The factor base $\mathcal{F} \subset \mathcal{G}$ will contain elements represented by irreducible polynomials up to a certain degree, depending on the algorithm. We have not precised what polynomial P we use to represent $\mathbb{F}_{q^n} \cong \mathbb{F}_q[X]/(P)$, but this is done in Section 8.4.2. Again, we will make a notation abuse : if we have $Q \in \mathbb{F}_q[X]$, we will still write Q for the class $\bar{Q} \in \mathbb{F}_q[X]/(P)$. Conversely, if we have an element $Q \in \mathbb{F}_{q^n}$, we will also denote by Q the polynomial in $\mathbb{F}_q[X]$ representing the element.

8.4.2 Background ideas

In the case of $\mathbb{Z}/p\mathbb{Z}$, we used the decomposition of integers into prime factors to generate multiplicative relations. The situation here is analogous : we decompose some polynomials into irreducible polynomials of lower degree to get relations. Before going into details, we list three ideas (first stated in [23]) used in the algorithms.

Idea 1 : homographies. Given a polynomial Q that factors *nicely* (and thus is susceptible to give *nice* relations), we can produce a lot of other polynomials using changes of variables induced by homographies :

$$X \rightarrow \frac{aX + b}{cX + d}.$$

We see that $Q(\frac{aX+b}{cX+d})$ is not a polynomial, so by change of variable using homographies, we mean homogeneous evaluation at $\frac{aX+b}{cX+d}$, that is, the polynomial :

$$Q_{abcd}(X) = (cX + d)^{\deg Q} Q\left(\frac{aX + b}{cX + d}\right).$$

If we have the decomposition $Q = \gamma \prod_j Q_j$, with γ a constant and Q_j irreducible monic polynomials, then we also have

$$Q_{abcd}(X) = \gamma \prod_j (cX + d)^{\deg Q_j} Q_j\left(\frac{aX + b}{cX + d}\right)$$

but the new factors in this decomposition are not necessary irreducible, not necessary monic and may even have a lower degree than the original Q_j .

Idea 2 : systematic polynomial splitting. The second idea is to use the polynomial $X^q - X$ because we already know that it factors into linear polynomials over \mathbb{F}_q . Thus, instead of looking for a random candidate to apply the first idea, we always use $X^q - X$.

Idea 3 : defining polynomial. We want to take advantage of the freedom that we have for the irreducible polynomial P defining $\mathbb{F}_{q^n} = \mathbb{F}_q[X]/(P)$. In fact, by using the ideas 1 and 2, we somehow favor the term X^q and spread it in our equations. That is why we make a choice of P that transforms X^q into something simpler. To do that, we ask P to be an irreducible factor of degree n of $h_1(X)X^q - h_0(X)$, where $h_0, h_1 \in \mathbb{F}_q[X]$ are polynomials of low degree δ (in practice, we take $\delta \leq 2$). If we denote by x the class of X in \mathbb{F}_{q^n} , we have

$$x^q = \frac{h_0(x)}{h_1(x)}.$$

Since this construction requires a simple expression of the map $x \mapsto x^q$, it is called *Frobenius representation*. It is not known if it is always possible to find suitable polynomials h_0, h_1 , the fact that every finite field can be represented that way is a *heuristic*. This construction might be impossible simply because $n > q + \delta$, but in fact, we can always assume that we have $n \leq q + \delta$ by embedding our field \mathbb{F}_{q^n} into a bigger one. Let us start again from scratch : let \mathbb{F}_{p^n} be the finite field in which we want to solve the DLP and let $q = p^l$ be the smallest power of p such that $q \geq n$. Then we embed \mathbb{F}_{p^n} into $\mathbb{F}_{p^{ln}} = \mathbb{F}_{q^n}$ and we actually solve the DLP in \mathbb{F}_{q^n} . From now on, we will always assume that \mathbb{F}_{q^n} has been constructed that way, so that there is no obstruction to the Frobenius representation.

These ideas are used both to find relations between the elements in the factor base and to express a target element as a combination of elements in the base. In general, this last operation cannot be done in one single step. Instead, the target element will be expressed as “simpler” elements, *i.e.* elements represented by polynomials with lower degrees. This strategy is applied recursively until all the elements are in the factor base \mathcal{F} . This whole operation is called the *descent*. We give more details in next section.

8.5 The BGJT algorithm

The BGJT algorithm gets its name from its authors : Barbulescu, Gaudry, Joux and Thomé. It was first published in 2013 [6] and it is the first (heuristic) quasi-polynomial algorithm. The setup is slightly different from the general one given in Section 8.4.1, it is the same as the original one given in the article.

8.5.1 Setup of the BGJT algorithm

Let $\mathcal{G} = (\mathbb{F}_{q^n})^\times$ the multiplicative group of a finite field of small characteristic \mathbb{F}_{q^n} , where $q = p^l$ is a prime power. We also assume that there is a “medium subfield” in \mathbb{F}_{q^n} , *i.e.* a field of size \mathbb{F}_{q^2} . In other words, we ask that $n = 2m$ for some $m > 0$. If this is not the case, we first embed our field \mathbb{F}_{q^n} in a larger field. Thus, we represent elements of $\mathbb{F}_{q^n} = \mathbb{F}_{q^{2m}} = \mathbb{F}_{q^2}[X]/(P)$ by polynomials over \mathbb{F}_{q^2} of degree less than m . As described in Section 8.4.2, the polynomial P is an irreducible polynomial of degree m dividing $h_1(X)X^q - h_0(X)$, where $h_0, h_1 \in \mathbb{F}_{q^2}[X]$ are polynomials of small degree. Heuristically, it seems that taking polynomials of degree 2 is sufficient. Our factor base $\mathcal{F} \subset \mathcal{G}$ will be the set of elements represented by degree one polynomials. Using the relation $X^q = \frac{h_0(X)}{h_1(X)}$, a power of h_1 appears in all equations, so we also add h_1 to the factor base \mathcal{F} . When we target an element $h \in \mathbb{F}_{q^{2m}}$, we implicitly assume that a basis for the discrete logarithm has been chosen and that h belongs to a subgroup of \mathcal{G} whose order has no

small irreducible factor. This assumption is possible thanks to the use of the Pohlig-Hellman algorithm [11].

8.5.2 Main results and complexity

Now that we have the setup for the algorithm, we can state the main results and explain how to use them to recover the discrete logarithm of an element in $\mathbb{F}_{q^{2m}}$. The following result is used to express an element as “simpler elements”.

Proposition 8.5.1

Let $K = \mathbb{F}_{q^{2m}}$ be a finite field with the structure discussed in Section 8.5.1. There exists a heuristic algorithm whose complexity is polynomial in q and m and which can be used for the following task. Given an element of K represented by a polynomial $Q \in \mathbb{F}_{q^2}[X]$ with $2 \leq \deg Q \leq m-1$, the algorithm returns an expression of $\log Q$ as a linear combination of at most $O(q^2m)$ logarithms $\log Q_j$ with $\deg Q_j \leq \lceil \frac{1}{2} \deg Q \rceil$ and of $\log h_1$.

In order to compute the logarithms of the linear elements and h_1 , i.e. the factor base, we use this second proposition.

Proposition 8.5.2

Let $K = \mathbb{F}_{q^{2m}}$ be a finite field with the structure discussed in Section 8.5.1. There exists a heuristic algorithm whose complexity is polynomial in q and m and that returns the logarithm of h_1 and of the logarithms of all the elements of K of the form $X + a$ for $a \in \mathbb{F}_{q^2}$.

Before explaining how to perform the algorithms, we show how to use them to recover the discrete logarithm of an element, and we discuss the complexity of such a process. Let $h \in \mathbb{F}_{q^{2m}}$ be a target element. We denote by $Q \in \mathbb{F}_{q^2}[X]$ the polynomial representing h , and we assume that $2 \leq \deg Q \leq m-1$, otherwise h is already in the factor base \mathcal{F} . We start by applying Proposition 8.5.1 to Q , so we obtain a relation of the form

$$\log Q = q_0 \log h_1 + \sum_j q_j \log Q_j$$

where the sum has at most $O(q^2m)$ terms and where $\deg Q_j \leq \lceil \frac{1}{2} \deg Q \rceil$ for all j . We then apply Proposition 8.5.1 recursively to the Q_j 's, creating a descent process (see Figure 8.1) in which the logarithm of each element is expressed as a linear combination of the logarithms of other elements with degrees twice as small (rounded up). By Proposition 8.5.1, the arity of the descent tree is in $O(q^2m)$. At the end of the process, the logarithm of Q is expressed as a linear combination of linear polynomials and of h_1 . These are the elements of the factor base, and we can find their logarithm thanks to Proposition 8.5.2.

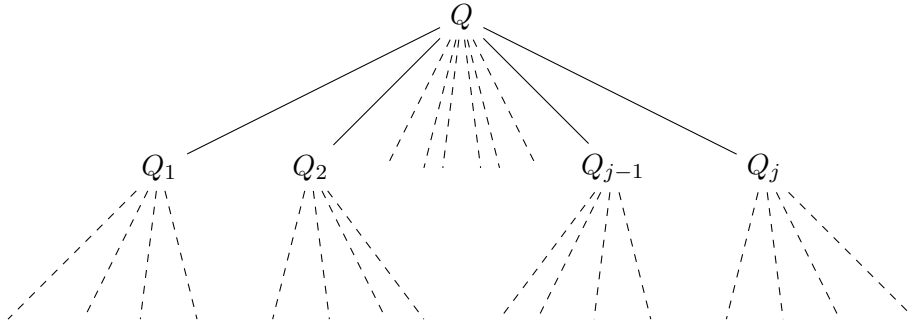


FIGURE 8.1 – The descent tree

The cost of each use of Proposition 8.5.1 is bounded by a polynomial in q and m , and each node of the descent tree corresponds to one application of the proposition. The complexity of the descent is therefore bounded by the number of nodes times a polynomial in q and m . Since the degree of the polynomials involved is divided by two during each step, the depth of the tree is $O(\log m)$.

The number of nodes in a tree is bounded by its arity raised to the power of its depth, so the number of nodes is bounded by $(q^2 m)^{O(\log m)}$. Since any polynomial is absorbed in the $O()$ notation in the exponent, we can conclude that the descent has a complexity bounded by

$$\max(q, m)^{O(\log m)}.$$

In the original article [6], the authors discuss the consequences of such a result for various ranges of parameters. In our case, where we assume that q is polynomial in the size of the input field $l = \log |\mathbb{F}_{q^{2m}}|$ and $q \geq m$, we can write the complexity

$$l^{O(\log l)}.$$

Therefore, we have the claimed quasi-polynomial complexity. Note that if we had $q = L_{q^{2m}}(\alpha)$ with some value $\alpha < \frac{1}{3}$, the complexity of the BGJT algorithm would be in $L_{q^{2m}}(\alpha + o(1))$, so it would no longer be quasi-polynomial. Nevertheless, even if less impressive than quasi-polynomial complexity, $L_{q^{2m}}(\alpha + o(1))$ is still better than previously known algorithms for large families of finite fields. Now that we have a global view of how to compute a discrete logarithm, we explain the algorithms the propositions refer to.

8.5.3 Proof of Proposition 8.5.1 and Proposition 8.5.2

We start with Proposition 8.5.1, but as we will see, the method used in the second proposition is essentially the same. Let Q be the element to be eliminated, *i.e.* to be expressed as “simpler” elements. The strategy is to find

relations involving Q and its translates by a constant $Q - \xi$, where $\xi \in \mathbb{F}_{q^2}$. We use a projective version of the *systematic equation* :

$$X^q - X = \prod_{u \in \mathbb{F}_q} X - u$$

obtained via homogenization that we write as a product on the projective line $\mathbb{P}_1(\mathbb{F}_q)$:

$$X^q Y - X Y^q = \prod_{\alpha \in \mathbb{P}_1(\mathbb{F}_q)} X - \alpha Y. \quad (8.1)$$

In this slightly abusive notation, when $\alpha = (u : 1)$ is not the point at infinity, the expression $X - \alpha Y$ simply means $X - uY$, whereas when $\alpha = (1 : 0)$ is the point at infinity, the expression $X - \alpha Y$ means Y .

To make the translates appear, we will use homographies, denoted by matrices $m = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, on Equation (8.1), where $a, b, c, d \in \mathbb{F}_{q^2}$. With each element m , we associate the equation (E_m) by substituting X by $aQ + b$ and Y by $cQ + d$.

$$\begin{aligned} (aQ + b)^q (cQ + d) - (aQ + b)(cQ + d)^q &= \prod_{\alpha \in \mathbb{P}_1(\mathbb{F}_q)} (aQ + b) - \alpha(cQ + d) \\ &= \prod_{\alpha \in \mathbb{P}_1(\mathbb{F}_q)} (a - \alpha c)Q + b - \alpha d \\ &= \lambda \prod_{\alpha \in \mathbb{P}_1(\mathbb{F}_q)} Q - m^{-1} \cdot \alpha. \quad (E_m) \end{aligned}$$

Again, in the products of the two first lines, some care has to be taken about the meaning of the terms, but the explicit meaning follows directly from the explanations given above. In the last line, that gives our Equation (E_m) , the notation $Q - m^{-1} \cdot \alpha$ means $Q - u$ if $m^{-1} \cdot \alpha = (u : 1)$; it means 1 if $m^{-1} \cdot \alpha = \infty = (1 : 0)$. Finally, $\lambda \in \mathbb{F}_{q^2}$ is the constant such that the equality holds.

In the end, we just reorganized the terms by factoring the constants multiplying Q . Note that one of the constants could be zero and the term Q could then vanish, this is happening when $m^{-1} \cdot \alpha = \infty$. Therefore, the right hand of Equation (E_m) is, up to a multiplicative constant, a product of $q + 1$ or q translates of Q .

At this point, one may wonder what we mean by m^{-1} , since we did not specify anything about m . In fact, we have some constraints on this matrix. If m is not invertible, we have only one translate appearing in the right hand side, leading to a trivial relation that we will not exploit. If the constants a, b, c, d are all in \mathbb{F}_q , the relation generated is, up to a constant in \mathbb{F}_q , always the same, so we consider this case trivial too. Finally, we can see that multiplying all the coefficients by a non-zero constant will lead to the

same relation, up to a constant, so we consider this trivial again. We consider that the relations are the same when they differ by a constant because the logarithms of the constants are easy to find : since \mathbb{F}_{q^2} contains q^2 elements, and since q is a polynomial in the size of the input, we can find the logarithms of all the constants in polynomial time by computing ζ^j for $1 \leq j \leq q^2 - 1$ and ζ a generator of $\mathbb{F}_{q^2}^\times$. All that being said, we conclude that we have to take m in the set of cosets :

$$\mathcal{P}_q = \text{PGL}_2(\mathbb{F}_{q^2}) / \text{PGL}_2(\mathbb{F}_q).$$

Note that in general $\text{PGL}_2(\mathbb{F}_q)$ is not a normal subgroup of $\text{PGL}_2(\mathbb{F}_{q^2})$, therefore \mathcal{P}_q is not a quotient group, but only a set of cosets. We do not have a nice way of iterating through its elements perfectly, *i.e.* visiting each element only once and in $|\mathcal{P}_q|$ steps, but an efficient algorithm has been written in [28]. It is a general result that $|\text{PGL}_2(\mathbb{F}_{q^j})| = q^{3j} - q^j$, so the number of elements in the set of cosets \mathcal{P}_q is $|\mathcal{P}_q| = (q^6 - q^2) / (q^3 - q) = q^3 + q$.

Now, let us go back to our Equation (E_m). On the left side of the equation, we have a polynomial of high degree that seems hard to exploit. But recall that, with the choice of defining polynomial for our field $\mathbb{F}_{q^{2m}}$, we have $X^q \equiv \frac{h_0(X)}{h_1(X)}$. This will be used to lower the degree on the left hand side. Let us denote by \tilde{a} the element a^q for any $a \in \mathbb{F}_q^2$. Similarly, we denote by \tilde{Q} the polynomial Q with all its coefficients raised to the power q . We can now write the left hand of Equation (E_m)

$$(\tilde{a}\tilde{Q}(X^q) + \tilde{b})(cQ(X) + d) - (aQ(X) + b)(\tilde{c}\tilde{Q}(X^q) + \tilde{d}),$$

and using the defining polynomial for $\mathbb{F}_{q^{2m}}$, the left hand side of Equation (E_m) is congruent to the following.

$$\left(\tilde{a}\tilde{Q}\left(\frac{h_0(X)}{h_1(X)}\right) + \tilde{b} \right) (cQ(X) + d) - (aQ(X) + b) \left(\tilde{c}\tilde{Q}\left(\frac{h_0(X)}{h_1(X)}\right) + \tilde{d} \right) \quad (\mathcal{L}_m)$$

The denominator of (\mathcal{L}_m) is a power of h_1 , whose logarithm is supposed to be known, and its numerator has a much smaller degree than the original one in (E_m). Indeed, the numerator of (\mathcal{L}_m) has degree at most $(1 + \delta) \deg Q$, where $\delta = \max(\deg h_0, \deg h_1)$ (we can assume $\delta = 2$ in practice), whereas the original degree in (E_m) is at most $(1 + q) \deg Q$ (and we can have $q = 3^5 = 243$ in practice, see [2] for an example).

But we did not explain why we want small degree polynomials so much in our equations. The answer is probabilistic : it is more likely that a polynomial has only irreducible factors of small degree if it has small degree itself. We say that a polynomial Q is D -smooth is it has only irreducible factors of degree smaller than D . Hence, the probability that Q is D -smooth, for a fixed D , increases when the degree of Q decreases. In our case, we are interested in $\lceil \frac{1}{2} \deg Q \rceil$ -smooth polynomials. We say that $m \in \mathcal{P}_q$ yields a relation if the

numerator of (\mathcal{L}_m) is $\lceil \frac{1}{2} \deg Q \rceil$ -smooth. We will keep only the equations such that m yields a relation, and we will remember the translates involved in the right hand side of Equation (E_m) .

To any $m \in \mathcal{P}_q$, we associate a row vector $v(m)$ of size $q^2 + 1$ indexed by $\xi \in \mathbb{P}_1(\mathbb{F}_{q^2})$. Each coordinate is either 1 or 0, depending on if the term $Q - u$ for $\xi = (u : 1)$ appears or not in the right hand side of Equation (E_m) . If there are only q terms in the product, we put 1 in position $\xi = \infty$. Hence, we always have exactly $q + 1$ non-zero coordinates in $v(m)$. Equivalently, we can write

$$v(m)_{\xi \in \mathbb{P}_1(\mathbb{F}_{q^2})} = \begin{cases} 1 & \text{if } \xi = m^{-1} \cdot \alpha \text{ with } \alpha \in \mathbb{P}_1(\mathbb{F}_q) \\ 0 & \text{otherwise} \end{cases}.$$

We construct a matrix $H(Q)$ whose rows are the $v(m)$ for which m yields a relation, taking at most one matrix m in each coset. We see that $H(Q)$ has $q^2 + 1$ columns but the number of rows depends on Q . In fact, we are able to bound from below the probability that m yields a relation by a constant κ (see [6] for details), therefore, since $m \in \mathcal{P}_q$ and $|\mathcal{P}_q| = q^3 + q$, the number of rows in $H(Q)$ is $\Theta(q^3)$. In order to finish the proof of Proposition 8.5.1, we need the following heuristic.

Heuristic 8.5.3

For any polynomial $Q \in \mathbb{F}_{q^2}[X]$, the set of rows $v(m)$ for cosets $m \in \mathcal{P}_q$ that yield a relation forms a matrix $H(Q)$ of full rank $q^2 + 1$.

We implicitly ask for q to be large enough for $\Theta(q^3)$ to be larger than $q^2 + 1$, otherwise the heuristic cannot hold. In other words, we ask the matrix $H(Q)$ to be sufficiently random to be invertible. Proposition 8.5.1 follows from linear algebra on $H(Q)$. Since the matrix has full rank, there is a linear combination of the rows leading to the vector $(1, 0, \dots, 0)$. We can also assume that the first element in our iteration of $\mathbb{P}_1(\mathbb{F}_{q^2})$ is $\xi = (0 : 1)$, such that a 1 in the first position corresponds to the term Q in the right hand side of Equation (E_m) . Thus, saying that $(1, 0, \dots, 0)$ can be written as a linear combination of the rows of $H(Q)$ is equivalent to saying that the polynomial Q can be written as a product of the right hand side of those Equations (E_m) for which m yields a relation. On the left hand side of this combination, we obtain a product of $\lceil \frac{1}{2} \deg Q \rceil$ -smooth polynomials, which is thus $\lceil \frac{1}{2} \deg Q \rceil$ -smooth too. In the end, we have expressed $\log Q$ as a linear combination of $\log Q_i$ where the Q_i are the irreducible polynomials occurring in the left hand side. Since there are $O(q^2)$ columns in $H(Q)$, the elimination process involves at most $O(q^2)$ rows. Each row corresponds to an Equation (E_m) whose left hand side is congruent to (\mathcal{L}_m) , and the degree of the numerator of (\mathcal{L}_m) is $(1 + \delta) \deg Q$. In the end, Q is expressed as a product of $O(q^2 \deg Q)$ polynomials of degree less than $\lceil \frac{1}{2} \deg Q \rceil$. The polynomial h_1 also appears as the denominator of (\mathcal{L}_m) , but this is not an issue since its logarithm is known.

In this process, we perform some polynomial manipulations and smoothness tests to know whether $m \in \mathcal{P}_q$ yields a relation. All these manipulations can be done in polynomial time in q and $\deg Q \leq m$. The linear algebra part, performed on a matrix with $q^2 + 1$ columns, has a cost of $O(q^{2\omega})$, using asymptotically fast matrix multiplication, or $O(q^5)$ using sparse matrix techniques (remember we have only $q + 1$ non-zero coefficients out of $q^2 + 1$). Therefore, we have an algorithm with polynomial time in q and m , as claimed, and Proposition 8.5.1 is proven.

In order to prove Proposition 8.5.2, we use the same process with Q replaced by the polynomial X . In this particular case, the right hand side of Equation (E_m) involves only linear polynomials, and the condition “ $m \in \mathcal{P}_q$ yields a relation” means that the numerator of (\mathcal{L}_m) factors into linear polynomials. Thus, the generated relations involve only linear polynomials and h_1 . That gives us a linear system whose unknowns are the $\log X + \xi$ for $\xi \in \mathbb{F}_{q^2}$ and $\log h_1$. We cannot rely on Heuristic 8.5.3 to know whether this system has solutions or not, so we need to state another similar heuristic. In practice, we find that the matrix associated with the system has a kernel of dimension 1; this is the best we can hope for because a dimension of 0 would mean that the only solution is $\log(X + \xi) = 0 = \log h_1$ for all $\xi \in \mathbb{F}_{q^2}$ and a dimension larger than 2 would mean too many solutions. If we choose the basis of the logarithm to be a linear element $X + \xi_0$, we are able to recover all the logarithms by choosing the only solution such that $\log(X + \xi_0) = 1$. This concludes the proof of Proposition 8.5.1 and Proposition 8.5.2.

8.5.4 Some concluding remarks

There are three kinds of heuristics used in this algorithm.

- The existence of suitable polynomials h_0 and h_1 , which give the structure needed to perform the degree reduction in the left hand side of (E_m) and obtain (\mathcal{L}_m).
- Heuristic 8.5.3 and its equivalent in the case $Q(X) = X$, needed for the linear algebra step.
- Finally, the analysis leading to the bound on the probability that $m \in \mathcal{P}_q$ yields a relation is based on the assumption that the numerator of (\mathcal{L}_m) has the same probability to be $\lceil \frac{1}{2} \deg Q \rceil$ -smooth as a random polynomial of the same degree.

Supporting arguments and experimental results are given in [6]. Another important fact is that the logarithm of certain polynomials cannot be computed by that process. They are called *problematic polynomials* and a result is given in [6] to overcome this issue without changing the quasi-polynomial nature of the algorithm. It happens when the polynomial to be eliminated $Q \neq P$, where P is the defining polynomial of $\mathbb{F}_{q^{2m}}$, divides the polynomial $h_1(X)X^q - h_0(X)$. In that case, Q also divides the numerator of (\mathcal{L}_m), making it impossible to link the logarithm of Q with smaller degree polynomials,

or preventing Proposition 8.5.2 to succeed if Q is a linear polynomial.

The main obstacle preventing this algorithm to be very efficient in practice, in spite of its quasi-polynomial complexity, is the $O(q^2 \deg Q)$ arity of the descent tree. In the powers-of-2 algorithm, studied in Section 8.6, the arity is smaller and one heuristic is not needed anymore, making one step towards a heuristic-free algorithm.

8.6 The powers-of-2 algorithm

The algorithm gets its name from its design, particularly suited to target elements with a degree of the form 2^j . It was written in 2014 by Granger, Kleinjung and Zumbrägel. Like the BGJT algorithm, it is based on a heuristic Frobenius representation and on the other ideas of Section 8.4.2. The setup in which we present the algorithm is slightly different from the original, thanks to the work of Joux and Pierrot in [24]. Indeed, they remarked that working with polynomials of degree up to d over \mathbb{F}_q is essentially equivalent to working with linear polynomials over \mathbb{F}_{q^d} . Thus, instead of working with a factor base composed of linear polynomials over \mathbb{F}_{q^d} (with d usually between 2 and 4), we work with a factor base composed of irreducible polynomials up to degree d . Joux and Pierrot precisely give a way to compute such a factor base in [24], that is based on the technique used in the BGJT algorithm [23, 6] and has a complexity of $O(q^6)$. Since we work in the case where q is polynomial in the bitsize of the input field, this precomputation time is polynomial.

8.6.1 Setup of the algorithm

Let $\mathcal{G} = (\mathbb{F}_{q^n})^\times$ be the multiplicative group of a finite field of small characteristic \mathbb{F}_{q^n} , where $q = p^l$ is a prime power, and where we represent elements of \mathbb{F}_{q^n} by polynomials over \mathbb{F}_q of degree less than n . More precisely, we consider the field $\mathbb{F}_{q^n} \cong \mathbb{F}_q[X]/(P)$ where $P \in \mathbb{F}_q[X]$ is an irreducible polynomial of degree n over \mathbb{F}_q dividing $h_1 X^q - h_0$, and $h_0, h_1 \in \mathbb{F}_q[X]$ are polynomials of degree at most 2 over \mathbb{F}_q . Different options are available to choose h_0 and h_1 , and the respective results are discussed in [24]. The existence of such parameters h_0 and h_1 is, again, heuristic and critical for the algorithm.

We set the factor base \mathcal{F} to be composed of all the elements of \mathcal{G} represented by an irreducible polynomial of degree at most 4. We assume that we know the logarithm of each element in \mathcal{F} . In other words, we assume that the steps 1 and 2 of the index calculus method have already been performed. This can be done by using the BGJT-like techniques in the paper of Joux and Pierrot [24]. As we already pointed out in a discussion above, in the original article the authors proposed to apply Proposition 8.5.2 to the field \mathbb{F}_{q^4} . The only remaining step is then 3, where we express the logarithm of an element in \mathcal{G} as the linear combination of the logarithms of elements in \mathcal{F} .

In order to perform that step, we use a descent strategy, like in the BGJT algorithm. We express the logarithm of the element that we want to study, denoted by $Q \in \mathcal{G}$, as a linear combination of $q + 2$ “simpler” elements Q_j , *i.e.* elements represented by polynomials of degree twice as small. Proceeding recursively, we are able to express $\log Q$ as a linear combination of elements of degree at most 4, *i.e.* elements in \mathcal{F} . We have to precompute the polynomials up to degree 4 because the descent does not work to express elements of degree 4 as elements of degree 2. This is because in the descent, we exploit polynomials of a certain form, and there are less available polynomials when working with elements of small degree. With elements of degree 4 or less, the lack of polynomials prevent us from using the descent. Before explaining the descent, we have to introduce its building block, the *on-the-fly* elimination. Unlike the building block of the BGJT algorithm, namely Proposition 8.5.1, there is no linear algebra in that step.

8.6.2 On-the-fly degree 2 elimination

The crucial part of the descent is called the *on-the-fly* elimination. It allows one to express a degree 2 polynomial $Q \in \mathbb{F}_{q^m}[X]$ over \mathbb{F}_{q^m} as a product of degree 1 polynomials over \mathbb{F}_{q^m} . The name “on the fly” is due to the fact that no relation gathering or linear algebra is performed during this elimination. At the core of this elimination we also use the fact that polynomials of the form $\mathcal{P}_B = X^{q+1} - BX + B$ split completely over \mathbb{F}_{q^m} (*i.e.* they are a product of polynomials of degree 1 over \mathbb{F}_{q^m}) very often. We denote by \mathcal{B} the set of elements $B \in \mathbb{F}_{q^m}$ such that \mathcal{P}_B splits completely. In [9], the roots and the splitting field of \mathcal{P}_B are studied and it is shown that the cardinality of \mathcal{B} is approximately q^{m-3} .¹ Thus, if $B \in \mathcal{B}$, $a, b, c \in \mathbb{F}_{q^m}$ such that $c \neq ab$, $a^q \neq b$ and $B = \frac{(b-a^q)^{q+1}}{(c-ab)^q}$, we see by the change of variable $X \rightarrow \frac{ab-c}{b-a^q}X - a$ in $X^{q+1} + aX^q + bX + c$ that the polynomial $X^{q+1} + aX^q + bX + c$ splits completely whenever \mathcal{P}_B does.

To see that, let us consider the change of variable $X \mapsto uX + v$ in $X^{q+1} + aX^q + bX + c$, where u and v are yet to be determined. We obtain the new polynomial

$$u^{q+1}X^{q+1} + (u^qv + au^q)X^q + (uv^q + bu)X + v^{q+1} + av^q + bv + c.$$

If we want this polynomial to be of the form \mathcal{P}_B , we must impose that

$$\begin{cases} u^qv + au^q = 0 \\ uv^q + bu + v^{q+1} + av^q + bv + c = 0 \end{cases} .$$

1. $|\mathcal{B}| = \begin{cases} (q^{m-1} - 1)/(q^2 - 1) & \text{if } m \text{ is odd} \\ (q^{m-1} - q)/(q^2 - 1) & \text{if } m \text{ is even} \end{cases}$

We know that $u \neq 0$, otherwise the change of variable would be trivial, so we deduce that $v = -a$. Then, we note that

$$v^{q+1} + av^q = ((-1)^{q+1} + (-1)^q)a^{q+1} = 0.$$

So the second equation becomes

$$u(-1)^q a^q + bu - ba + c = 0,$$

and we deduce that $u = \frac{ab-c}{b+(-1)^q a^q}$. Assuming that we work with odd characteristic (otherwise the formula can be adapted), we obtain the announced change of variable. Dividing by the leading coefficient u^{q+1} , we also have the equality $B = \frac{(b-a^q)^{q+1}}{(c-ab)^q}$. Note that using [9, 22, 18], we also know that \mathcal{B} is the image of $\mathbb{F}_{q^m} \setminus \mathbb{F}_{q^2}$ under the map

$$u \mapsto \frac{(u - u^{q^2})^{q+1}}{(u - u^q)^{q^2+1}},$$

so we are able to compute elements in \mathcal{B} .²

Let Q be the polynomial to be eliminated. We have to introduce one last tool, linked to Q . We define the lattice $L_Q \subset \mathbb{F}_{q^m}[X]^2$ by

$$L_Q = \{(w_0, w_1) \in \mathbb{F}_{q^m}[X]^2 \mid w_0 h_0 + w_1 h_1 \equiv 0 \pmod{Q}\}.$$

If Q divides $w_0 h_0 + w_1 h_1 \neq 0$ for $w_0, w_1 \in \mathbb{F}_{q^m}$, then we have

$$Q = w(w_0 h_0 + w_1 h_1)$$

for some $w \in \mathbb{F}_{q^m}^\times$ because the degree of the right hand side is at most 2. Hence, using the equality $X^q \equiv \frac{h_0(X)}{h_1(X)}$, we see that the logarithm of Q is linked with the logarithm of

$$w_0 X^q + w_1 = (w_0^{q^{m-1}} X + w_1^{q^{m-1}})^q,$$

and so the logarithm of Q is linked to the logarithm of a linear polynomial over \mathbb{F}_{q^m} . Otherwise, and this is the generic case, we can find a basis of L_Q of the form $(u_0, X + u_1), (X + v_0, v_1)$ with $u_i, v_i \in \mathbb{F}_{q^m}$ using linear algebra.

To find such a basis, we write :

- $Q = X^2 + dX + e$ (it does not change the divisibility results to suppose that Q is monic),
- $\bar{h}_0 = h_0 \pmod{Q} = d_0 X + e_0$,
- $\bar{h}_1 = h_1 \pmod{Q} = d_1 X + e_1$,
- (w_0, w_1) a solution, with
 - $w_0 = \alpha_0 X + \beta_0$,

2. There is also a proof in [19].

$$- w_1 = \alpha_1 X + \beta_1.$$

We seek a basis of the form $(u_0, X + u_1), (X + v_0, v_1)$, that is to say that the solutions (w_0, w_1) can be written

$$(w_0, w_1) = \lambda(u_0, X + u_1) + \mu(X + v_0, v_1)$$

for some $\lambda, \mu \in \mathbb{F}_{q^m}$. Looking at the leading coefficients, we deduce that $\mu = \alpha_0$ and $\lambda = \alpha_1$. Now, writing the conditions such that $w_0 \bar{h}_0 + w_1 \bar{h}_1 \equiv 0 \pmod{Q}$, we obtain a linear system of two equations in 4 unknowns $\alpha_0, \alpha_1, \beta_0, \beta_1$. We solve the system by expressing the β_i 's as a linear combination of the α_i 's :

$$\begin{cases} \beta_0 = f_{0,0}(d, d_0, d_1, e, e_0, e_1)\alpha_0 + f_{0,1}(d, d_0, d_1, e, e_0, e_1)\alpha_1 \\ \beta_1 = f_{1,0}(d, d_0, d_1, e, e_0, e_1)\alpha_0 + f_{1,1}(d, d_0, d_1, e, e_0, e_1)\alpha_1 \end{cases}$$

where the expressions $f_{i,j}(d, d_0, d_1, e, e_0, e_1)$ depend only on the known constants d, d_0, d_1, e, e_0, e_1 and appear while solving the system. But we also have $\beta_0 = \mu v_0 + \lambda u_0$, and recall that $\mu = \alpha_0$ and $\lambda = \alpha_1$. It follows that $\beta_0 = v_0 \alpha_0 + u_0 \alpha_1$. By identification, we obtain that $v_0 = f_{0,0}(d, d_0, d_1, e, e_0, e_1)$ and $u_0 = f_{0,1}(d, d_0, d_1, e, e_0, e_1)$. The same discussion allows us to find u_1 and v_1 .

Now that the tools are in place, let us use them to eliminate our target Q . We have

$$h_1(X^{q+1} + aX^q + bX + c) \equiv (X + a)h_0 + (bX + c)h_1 \pmod{P},$$

where $P|h_1X^q - h_0$ is the defining polynomial of our field. Therefore, if Q divides $(X + a)h_0 + (bX + c)h_1$, a polynomial of degree at most 3, then we have that

$$(X + a)h_0 + (bX + c)h_1 = Q \times R,$$

where R is a linear polynomial. If on top of that, the polynomial

$$X^{q+1} + aX^q + bX + c$$

splits completely over \mathbb{F}_{q^m} , then we have a relation between Q , polynomials of degree 1, and h_1 (which logarithm is supposed to be known). The goal is thus to find a triple (a, b, c) satisfying the two conditions. To do that, we choose $B \in \mathcal{B}$ and we remark that $(X + a, bX + c) \in L_Q$ implies that

$$(X + a, bX + c) = b(u_0, X + u_1) + (X + v_0, v_1).$$

Hence, we have $a = bu_0 + v_0$ and $c = bu_1 + v_1$. Rewriting a and c in the conditions on $(a, b, c) : c \neq ab, b \neq a^q$ and $\frac{(b-a^q)^{q+1}}{(c-ab)^q} = B$, we get the condition

$$B = \frac{(-u_0^q b^q + b - v_0^q)^{q+1}}{(-u_0 b^2 + (u_1 - v_0)b + v_1)^q}$$

and we can find a suitable b in polynomial time in q and m by finding a root to the degree $q^2 + q$ polynomial

$$\Delta(X) = (-u_0^q X^q + X - v_0^q)^{q+1} - B(-u_0 X^2 + (u_1 - v_0)X + v_1)^q.$$

Expanding the polynomial, we can see that it is quite sparse, indeed the only non-zero coefficients are those of degree $q^2 + q, q^2 + 1, q^2, 2q, q + 1, q, 1$ and the constant :

$$\begin{aligned} \Delta(X) = & u_0^{q^2+q} X^{q^2+q} - u_0^{q^2} X^{q^2+1} + u_0^{q^2} v_0^q X^{q^2} + (B u_0^q - u_0^q) X^{2q} + X^{q+1} \\ & + (v_0^{q^2} u_0^q - v_0^q + B(v_0^q - u_1^q)) X^q - v_0^{q^2} X + v_0^{q^2+q} - B v_1^q. \end{aligned}$$

Alternatively, we can write b in a $\mathbb{F}_{q^m}/\mathbb{F}_q$ basis and solve a quadratic system in m variables with Gröbner bases algorithms, with an exponential time in m . In practice [2], the second option is used, because this on-the-fly elimination is used with small m .

We also emphasize the fact that this elimination works if a suitable element b is found. For each $B \in \mathcal{B}$, the probability of finding a suitable b is about $\frac{1}{2}$, so if $m \geq 4$, then $|\mathcal{B}| \approx q$ and the probability that the elimination works is overwhelming.

8.6.3 The descent

Let now $Q \in \mathbb{F}_q[X]$ be a monic irreducible polynomial over \mathbb{F}_q of degree $2m$, with $m \geq 4$. We have seen in Section 8.6.2 how to express a quadratic polynomial in an extension \mathbb{F}_{q^m} of \mathbb{F}_q as a product of linear polynomials with coefficients in the same extension. Therefore, we express Q as a product of quadratic polynomials. We first introduce a notation : if $P \in \mathbb{F}_{q^m}$ is a polynomial, we denote by $P^{[i]}$ the polynomial P with all coefficients raised to the power q^i . Hence, if

$$P = \sum_{j=0}^d p_j X^j,$$

we have

$$P^{[i]} = \sum_{j=0}^d p_j^{q^i} X^j$$

and we call $P^{[i]}$ the i -th conjugate of P .

Since Q is an irreducible polynomial of degree $2m$ over \mathbb{F}_q , we know that $\mathbb{F}_{q^{2m}} \cong \mathbb{F}_q[X]/(Q)$ and that $\mathbb{F}_{q^{2m}}$ is a splitting field of Q . Furthermore, we know that if $\alpha \in \mathbb{F}_{q^{2m}}$ is a root of Q and σ is a \mathbb{F}_q -automorphism of $\mathbb{F}_{q^{2m}}$, then $\sigma(\alpha)$ is again a root of Q . By Galois theory, we know that the \mathbb{F}_q -automorphisms are the Frobenius iterates σ_0^i where $\sigma_0 : x \mapsto x^q$, hence we can write

$$Q = \prod_{i=0}^{2m-1} X - \alpha^{q^i}.$$

By gathering the terms $X - \alpha^{q^i}$ and $X - \alpha^{q^{m+i}}$ for $0 \leq i \leq m - 1$, we produce quadratic polynomials with coefficients in the field \mathbb{F}_{q^m} . They do not have any roots in the field \mathbb{F}_{q^m} and are quadratic so they are all irreducible polynomials. Finally, we have

$$Q = \prod_{i=0}^{m-1} Q_i = Q_0^{[i]}$$

where the polynomials $Q_i \in \mathbb{F}_{q^m}[X]$ are irreducible polynomials over \mathbb{F}_{q^m} of degree 2, and are all conjugates. By applying the on-the-fly elimination to Q_0 , we obtain

$$R_0 Q_0 = \prod_{j=1}^{q+1} R_j$$

where, for all $0 \leq j \leq q + 1$, $R_j \in \mathbb{F}_{q^m}[X]$ is a linear polynomial over \mathbb{F}_{q^m} . We also deduce that

$$R_0^{[i]} Q_i = \prod_{j=1}^{q+1} R_j^{[i]}$$

and it follows that

$$\left(\prod_{i=0}^{m-1} R_0^{[i]} \right) \times Q = \prod_{j=1}^{q+1} \left(\prod_{i=0}^{m-1} R_j^{[i]} \right).$$

Now recall that $N_j := \prod_{i=0}^{m-1} R_j^{[i]}$ is the norm of the polynomial R_j , and that the norm of a linear polynomial over $\mathbb{F}_{q^m}/\mathbb{F}_q$ is an irreducible polynomial over \mathbb{F}_q of degree e , raised to the power f , such that $ef = m$. Hence, we have $N_j = T_j^{f_j}$, with $T_j \in \mathbb{F}_q[X]$, $\deg T_j = e_j$, and $e_j f_j = m$. In the end, we have expressed the polynomial Q as a product of polynomials with degrees dividing m .

To be able to really perform this descent, we must have an irreducible polynomial Q with a power of 2 degree. This is possible thanks to Theorem 5.1 proved in [27] by Wan that roughly states that we can represent any element in \mathcal{G} by a monic irreducible polynomial with a power of 2 degree that is not too large, in order to keep the quasi-polynomial complexity. Assuming that the input polynomial is of degree $2^e = 2 \times 2^{e-1}$, note that the descent can be performed in a slightly different way than stated above. Indeed, instead of taking norms with respect to the base field \mathbb{F}_q , it is possible to take norms with respect to a subfield of index 2, hence obtaining quadratic polynomial with coefficient in $\mathbb{F}_{q^{e-2}}$ and being able to recurse. Figure 8.2 illustrates this strategy and this is what we use in practice.

The discussion about the complexity is very similar to the one we had for the BGJT algorithm. Performing the descent, a tree is built in which the logarithm of a node is expressed as a linear combination of the logarithms

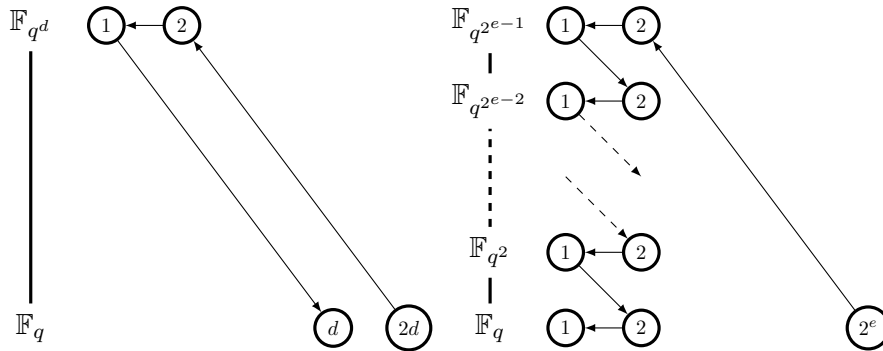


Fig. 2(a)

Fig. 2(b)

FIGURE 8.2 – Two different possible strategies to perform the descent, the arrows \nwarrow , \leftarrow , and \searrow stands respectively for factorization and embedding in a larger field, on-the-fly degree 2 elimination, and taking a norm and projecting in a subfield. The numbers represent the degrees of the involved elements.

of its children. The total complexity of the algorithm is then the number of nodes of this tree. We have seen that the arity of the tree is $q + 2$, and the height of the tree is $O(\log n)$, the number of nodes is then $(q + 2)^{\log n}$. Since we are in the small characteristic case, we can write this complexity $l^{O(\log l)}$, where $l = \log(q^n)$ is the bitsize of \mathcal{G} , and we see that the algorithm is quasi-polynomial.

8.6.4 Some concluding remarks

Similarly to the BGJT algorithm case, there are polynomials that cannot be descended using the on-the-fly elimination. Once again, it happens when dealing with a polynomial dividing the equation that defines our field. Nevertheless, these polynomials can be dealt with and details can be found in [18]. Two heuristics remain in this algorithm :

- The existence of suitable polynomials h_0 and h_1 , in order to be able to perform the on-the-fly elimination.
- The existence of a polynomial time algorithm to compute the logarithm of the elements of degree up to 4. Indeed, we recall that the polynomial time algorithm of Proposition 8.5.1, *i.e.* the building block of the BGJT algorithm, is heuristic.

We now study the practical results obtained in our implementation of the quasi-polynomial algorithms.

8.7 Experimental results

All the material discussed in this section is available at <https://github.com/erou/DlogGF.jl>. It is published as a Julia [1] package, depending on the Nemo package [21]. We first give a very brief introduction to Julia and Nemo.

8.7.1 Julia and Nemo

Julia. Julia is a free and open-source, high-level programming language developed since 2012, with dynamic type system and high-performance. It is a compiled language, with a just-in-time (jit) compilation, meaning that the compilation is almost invisible for the user but performances are comparable with other compiled languages. It is easy to learn and to read, especially for someone who already worked with a high-level language like Python. There are many other interesting things to say about this new language, the interested reader can find additional informations on Julia's website and, of course, in the documentation of the language. Julia is designed for numerical computing, but there are some symbolic computer algebra packages, such as Nemo.

Nemo. Nemo is a computer algebra package for Julia. It aims to cover commutative algebra, number theory and group theory. It contains wrappers for MPIR, Flint, Arb and Antic, and other features written in Julia. Among these libraries, the C library Flint [20] (Fast library for number theory) is the only one that we use.

Why Julia/Nemo? One of the advantages of Julia is that it is fairly easy to directly call C functions contained in other libraries. Therefore, all the Nemo functions that we used in `DlogGF.jl` were in fact Flint functions. This provides efficiency, the main reason why we use the duo Julia/Nemo, as well as simplicity, because we are not directly using the C programming language. On top of that, if a Flint function is not available in Nemo, we can directly call it from Julia.

8.7.2 The BGJT algorithm

We recall that for Heuristic 8.5.3 to hold, we must have that q is not too small, because the expected number of equations is $\Theta(q^3)$, and the number of unknowns is $q^2 + 1$ or q^2 . When we compute the factor base, our unknowns are the $\log(X + a)$, for $a \in \mathbb{F}_{q^2}$, and h_1 . In our experiments, the heuristic holds as soon as $q \geq 7$, but not for every choice of parameters h_0 and h_1 . When $q \geq 11$, it holds in every experiment. In the descent part, the constant hidden in the $\Theta(q^3)$ depends on the degree of the target element, but it

is definitely smaller than when working with linear elements. Therefore the integers q for which the heuristic holds are larger.

We give timing results for the computation of the factor base, *i.e.* the linear elements and h_1 , in fields of the form $\mathbb{F}_{q^{2q}}$. Since elements are represented as polynomials with coefficients in \mathbb{F}_{q^2} , there are $q^2 + 1$ elements in the factor base.

Field	Bitsize	Timing (seconds)	Linear algebra
$\mathbb{F}_{7^{14}}$	39	0.20	5%
$\mathbb{F}_{11^{22}}$	76	1.5	23%
$\mathbb{F}_{17^{34}}$	139	16	59%
$\mathbb{F}_{23^{46}}$	208	140	79%
$\mathbb{F}_{29^{58}}$	282	990	79%

We see that the linear algebra is the main operation. For example, in the last computation (in $\mathbb{F}_{29^{58}}$), the other main steps were finding the factors of the order of the group, *i.e.* factoring $29^{58} - 1$, (10% of the time), computing the logarithms modulo the small factors with the Pohlig-Hellman algorithm (5% of the time), the other 6% being various polynomial manipulations, such as determining if a polynomial is smooth. The linear algebra step uses standard Gaussian elimination modulo each large factor of the cardinal of the group, performed on a matrix of size $\Theta(q^3) \times (q^2 + 1)$, for an expected complexity of $O(q^6)$. In examples of cryptographic size, for example with a bitsize of 4000, the linear algebra is known to be the bottleneck of the computations. Note that, since the matrix is sparse, we can obtain a complexity of $O(q^5)$ using sparse linear algebra. With some optimisations, and assuming that the factorization of the cardinal of the group is given, the other manipulations should be negligible.

In order to measure running times of the descent, we take a random monic irreducible polynomial of degree 3 in $\mathbb{F}_{q^{2q}}$ and we apply the algorithm of Proposition 8.5.1. We obtain polynomials of degree 1 and 2, and we count them to illustrate the $O(q^2m)$ arity. The number between parentheses is the number of polynomial of degree 2. In other words, this is the number of additional calls to Proposition 8.5.1 that are necessary to be able to compute the logarithm of our target element of degree 3.

Field	Bitsize	Timing (seconds)	Linear algebra	Polynomials obtained
$\mathbb{F}_{17^{34}}$	139	9.6	42%	794 (542)
$\mathbb{F}_{23^{46}}$	208	80	76%	1477 (1014)
$\mathbb{F}_{29^{58}}$	282	570	92%	2360 (1628)

The smaller field in which Heuristic 8.5.3 holds is $q = 17$, so we do not have results for smaller values of q . The main operation is again linear algebra, but we see that other operations are not negligible with small fields. Among these other operations, the smoothness tests are the most expensive.

This is probably because our test is just a factorization. A better test, that does not rely on factorization, is described in [2].

As already discussed earlier, the main drawback of this algorithm is its large arity.

8.7.3 The powers-of-2 algorithm

We do not analyze the precomputation of the factor base. Using the algorithm of Joux and Pierrot [24], the complexity of the precomputation is $O(q^6)$. Since the manipulations are inspired from the one found in the BGJT algorithm, the linear algebra is also the main part of the precomputation. Contrary to the BGJT algorithm, the descent of the powers-of-2 algorithm does not require anything about the size of q . It allows us to test the descent in smaller fields. In this descent, we know exactly the arity. We always express an element Q as $q + 2$ other elements. We apply the descent algorithm on elements of degree 8, meaning that only one on-the-fly elimination is needed. Hence, we can estimate the running time of the building block of the descent algorithm. In the table below, we write for example $\mathbb{F}_{3^5 \times 2^5}$ to specify that we apply the algorithm to \mathbb{F}_{q^n} with $q = 3^5$ and $n = 25$.

Field	Bitsize	Timing (seconds)	Root finding
$\mathbb{F}_{3^3 \times 10}$	47	1.5	77%
$\mathbb{F}_{17^2 \times 17}$	139	34	93%
$\mathbb{F}_{3^5 \times 2^3}$	182	47	91%
$\mathbb{F}_{23^2 \times 2^3}$	208	140	95%
$\mathbb{F}_{29^2 \times 2^9}$	282	400	97%
$\mathbb{F}_{3^5 \times 2^{10}}$	1585	49	91%

Note that the root finding is the bottleneck of the on-the-fly elimination. Our root finding is basic, so there is still hopes for improvement. In order to find a root in \mathbb{F}_{q^m} of the polynomial $\Delta(X)$ appearing in the elimination, since $\Delta(X)$ has degree $q^2 + q \leq q^m$, we first compute

$$S = X^{q^m} \pmod{\Delta(X)}.$$

Then we compute

$$\gamma(X) = \gcd(\Delta(X), X^{q^m} - X) = \gcd(\Delta(X), S - X)$$

and we factor the polynomial $\gamma(X)$. The computation of S typically takes the majority of the time, the rest being the computation of the gcd $\gamma(X)$. The factorization of $\gamma(X)$ is very fast because $\gamma(X)$ typically has a low degree.

Also, when proceeding to this elimination, we have to keep in mind that the polynomial $\Delta(X)$ may not have any roots. If no root is found, we create another polynomial $\Delta(X)$ and proceed to the root finding once again. The

timings that we give in the table are obtained when the root finding succeed with the first trial. Since almost all the time is spent in the root finding, k trials would result in k times the timing reported in the table.

We notice that the computations in $\mathbb{F}_{3^5 \times 23}$ and $\mathbb{F}_{3^5 \times 200}$ take approximately the same time. This is to be expected because even if the parameters change, such as h_0, h_1 , and the defining polynomial P , the computations are essentially the same. In both cases, we take a polynomial Q with coefficients in \mathbb{F}_{3^5} , and we perform the on-the-fly elimination in $\mathbb{F}_{3^5 \times 4}$. We now fix the field $\mathbb{F}_{3^4 \times 65}$ and we use the on-the-fly elimination on elements of different degrees.

Degree of the element	Timing (seconds)	Root finding
8	6	84%
16	19	92%
32	51	92%
64	185	90%

As expected, the elimination is longer with elements of high degree because the root finding is harder in bigger extensions of \mathbb{F}_{3^5} .

Because the polynomial $\Delta(X)$ has a high degree, the root finding is long and is the main obstacle preventing this algorithm to be used in practical works. In these works, summarised in Gora Adj's PhD thesis [2], we can see that the powers-of-2 algorithm is not really used all the way from elements of high degree (*i.e.* ≥ 200 in [2]) to elements in the factor base. Instead, other descents are first used to reduce the degree of the elements until having elements of small degree (*i.e.* ≤ 16 in [2]). Furthermore, even when using the powers-of-2 algorithm, it is mixed with Joux's $L(1/4)$ algorithm, in order to deal with the element of odd degree. If this mixed algorithm was not used with elements of higher degree, it is probably because the on-the-fly elimination was using Gröbner bases algorithms. These algorithms have an exponential complexity, but are efficient to eliminate small degree elements.

If we were able to use the sparse structure of the polynomials $\Delta(X)$, or the fact the the extensions $\mathbb{F}_{q^m}/\mathbb{F}_q$ are of degree a power of two, to implement a more efficient root finding, we could try to use the powers-of-2 algorithm in practice.

Acknowledgements

Of course, this work could not have been done without the help and advices of my supervisors, and I am very grateful that they let me work on this very exciting subject. I would also like to warmly thank Gora ADJ for the precious explanations he gave me during all the internship, his expertise on the subject was helpful at the very least. Finally, I would like to thank

all the people in the SCG lab for their welcome, and for the times chatting,
having lunch, or **defusing bombs**.

Bibliographie

- [1] Julia : a high-level, high-performance dynamic language for technical computing. <http://julialang.org>.
- [2] Gora Adj. *Logaritmo discreto en campos finitos de característica pequeña : atacando la criptografía basada en emparejamientos de Tipo 1*. PhD thesis, 2016.
- [3] L. Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pages 55–60, Oct 1979.
- [4] Ibrahim Assem. *Algèbres et modules*. Masson edition, 1997.
- [5] Arvind Ayyer, Anne Schilling, Benjamin Steinberg, and Nicolas M. Thiéry. Markov chains, r-trivial monoids and representation theory. *arXiv :1401.4250*, jan 2014.
- [6] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. *CoRR*, abs/1306.4244, 2013.
- [7] I.N. Bernstein, I.M. Gel'fand, and V.A. Ponomarev. Coxeter functors and Gabriel's theorem. Décembre 1972.
- [8] Anders Björner and Francesco Brenti. *Combinatorics of Coxeter Groups*. Springer edition, september 2004.
- [9] Antonia W Bluhner. On $x^{q+1} + ax + b$. *Finite Fields and Their Applications*, 10(3) :285–305, 2004.
- [10] A. Bostan, G. Lecerf, and É. Schost. Tellegen's principle into practice. In *ISSAC'03*, pages 37–44. ACM, 2003.
- [11] Stephen C. Pohlig and Martin E. Hellman. Hellman, m. : An improved algorithm for computing logarithms over $\text{gf}(p)$ and its cryptographic significance. *IEEE transactions on information theory*, it 24, 106-110. 24 :106 – 110, 02 1978.
- [12] Don Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE transactions on information theory*, 30(4) :587–594, 1984.

- [13] Luca De Feo, Javad Doliskani, and Éric Schost. Fast arithmetic for the algebraic closure of finite fields. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 122–129, New York, NY, USA, 2014. ACM.
- [14] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6) :644–654, Nov 1976.
- [15] Serge Elnitsky. Rhombic tilings of polygons and classes of reduced words in coxeter groups. *JCTA*, mar 1997.
- [16] Pierre Gabriel. *Unzerlegbare Darstellungen I*. Manuscripta Math. edition, 1972.
- [17] Shuhong Gao. *Normal Bases over Finite Fields*. PhD thesis, University of Waterloo, 1993.
- [18] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. On the powers of 2. *IACR Cryptology ePrint Archive*, 2014 :300, 2014.
- [19] Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel. On the discrete logarithm problem in finite fields of fixed characteristic. *Transactions of the American Mathematical Society*, 2016.
- [20] William Hart. Fast library for number theory : an introduction. *Mathematical Software-ICMS 2010*, pages 88–91, 2010.
- [21] William Hart et al. Nemo Package (Version 0.5.0). <http://nemocas.org>, 2016.
- [22] Tor Hellesest and Alexander Kholosha. $x^{2^l+1} + x + a$ and related affine polynomials over $\text{GF}(2^k)$. *Cryptography and Communications*, 2(1) :85–109, 2010.
- [23] Antoine Joux. A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic. *IACR Cryptology ePrint Archive*, 2013 :95, 2013.
- [24] Antoine Joux and Cécile Pierrot. Improving the polynomial time pre-computation of frobenius representation discrete logarithm algorithms - simplified setting for small characteristic finite fields. *IACR Cryptology ePrint Archive*, 2014 :924, 2014.
- [25] Jean-Pierre Serre. *Algèbres de Lie semi-simples complexes*. W.A. Benjamin, Inc. edition, 1966.
- [26] E. Bach ; J. Driscoll ; J. Shallit. Factor refinement. *Journal of Algorithms*, 15, 1993.
- [27] Daqing Wan. Generators and irreducible polynomials over finite fields. *Mathematics of Computation of the American Mathematical Society*, 66(219) :1195–1212, 1997.
- [28] Jincheng Zhuang and Qi Cheng. On generating coset representatives of $\text{PGL}_2(\mathbb{F}_q)$ in $\text{PGL}_2(\mathbb{F}_{q^2})$. In *Revised Selected Papers of the 11th*

International Conference on Information Security and Cryptology - Volume 9589, Inscrypt 2015, pages 167–177, New York, NY, USA, 2016. Springer-Verlag New York, Inc.