UNIVERSITÉ DU QUÉBEC À MONTRÉAL

BICARACTÈRES D'ESPACES DE POLYNÔMES HARMONIQUES MULTIVARIÉS

THÈSE
PRÉSENTÉE
COMME EXIGENCE PARTIELLE
DU DOCTORAT EN MATHÉMATIQUES

PAR
PAULINE HUBERT

MARS 2020

REMERCIEMENTS

J'aimerais tout d'abord remercier mes deux directeurs François Bergeron et Nicolas Thièry. François, merci de raconter les mathématiques comme tu le fais et de les rendre si passionnantes, merci de m'avoir conseillée et supportée tout au long de ce doctorat, merci pour ton soutien financier et tes encouragements et enfin pour tes conseils qui m'ont poussée à m'améliorer plus que je ne m'en croyais capable. Nicolas, merci pour ton accueil lors de mon séjour en France, merci pour ton temps, ton écoute et ta patience face à mes (très) nombreuses questions, merci pour tout ce que tu m'as appris sur Sage et en informatique en général. Merci à vous deux pour votre supervision sans laquelle ce projet ne serait pas allé si loin.

Je souhaite également remercier les membres de mon jury, A. Blondin-Massé, F. Saliola et M. Zabrocki d'avoir accepté de lire ma thèse et pour les commentaires constructifs dont ils m'ont fait part.

Je tiens également à remercier du fond du cœur les merveilleux amis que j'ai rencontrés au LaCIM. Merci à Nadia pour tout ce chemin fait ensemble (les prédocs, les soupers, les vacances, les conférences, ...), merci pour ton soutien et merci d'aimer la neige autant que moi! Merci à Mélodie pour les longues discussions dans notre bureau, pour ton bel accent du Saguenay et tes encouragements. Rien n'aurait été pareil au PK-4528 sans vous deux. Merci aussi à Antoine de toujours trouver le moyen de me faire rire et de me redonner le sourire. Merci à Florence de toujours être là pour moi et pour ton écoute en toutes circonstances. Merci également à Stéphanie, ta bonne humeur et ton humour sont précieux, ton amour pour les quiz aussi! Merci à Aram d'avoir cru en moi plus que moi-même et de

toujours m'encourager comme tu le fais. Merci aussi à Carole, Élise, Émile, Emine, Fanny, Herman, Nathan, Véronique, Guillaume, Laura et à tous ceux qui font et ont fait du LaCIM ce qu'il est.

Je souhaite également remercier les professeurs du LaCIM, Alexandre, Christophe H., Christophe R., Franco et Hugh ainsi que les secrétaires Johanne et Linda, de contribuer à l'ambiance chaleureuse et positive qui fait la marque du LaCIM. Enfin, je remercie Alexandra qui, au travers de l'ISM, m'a permis de participer à plusieurs projets très enrichissants tout au long de mon doctorat.

J'aimerais aussi remercier Monica, Balthazar, Justine, Joël et Hugo pour votre accueil lors mon séjour à Paris. Merci pour les bons moments partagés avec vous, souvent autour d'un jeu de société. Merci également aux professeurs du LRI, Viviane et Florent de m'avoir accueillie parmi vous au LRI l'espace de quelques mois.

J'aimerais ensuite remercier ma famille, mes parents, Sylvie et Philippe, ma sœur Mélanie et mon frère Rémi, pour votre support et votre amour inconditionnel. Depuis toujours vous avez cru en moi et sans vous je ne serais jamais arrivée si loin. Je voudrais aussi remercier ma meilleure amie, Audrey. Malgré la distance, tu es toujours là pour moi, prête à m'écouter ou à rire de mon accent.

Un grand merci également à toute la famille Laroche, vous êtes comme une deuxième famille pour moi. Merci de m'avoir accueillie dans votre famille alors que la mienne était si loin. Un merci particulier à Gabrielle et Philippe pour votre écoute et votre hospitalité. Merci aussi à Marc-Alexandre et Josiane pour tous les bons moments passés ensembles à la Chapelle et dans le 5 à 7.

Je ne pourrai pas non plus manquer de remercier toutes les personnes qui m'ont soutenues et qui ont tant prié pour moi durant ces quatre années et en particulier

dans les derniers mois. Merci à Luis, Sylvie, Jackson, Firmina, Amélie, Annie, Laurie, Aurélie, à toutes les personnes du 5 à 7 et à tous les autres. Merci à ma colocataire Wendy pour sa présence encourageante dans les dernières et non moins difficiles semaines. Merci à Gabrielle pour nos belles discussions, pour ton écoute et ton cœur débordant d'amour pour les autres. Et pour finir merci à Celui qui est au-dessus de toutes choses d'avoir marché à mes côtés sans jamais faillir.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

TABLE DES FIGURES

RÉSUMÉ

Cette thèse porte sur le calcul du caractère de l'action simultanée du groupe général linéaire $\mathrm{GL_r}$ et du groupe symétrique $\mathbb{S}_\mathrm{n}$ sur des sous-espaces des polynômes harmoniques multivariés en plusieurs jeux de variables. Les espaces auxquels nous nous intéressons sont engendrés par un déterminant de type Vandermonde associé à un diagramme γ et clos par dérivation et polarisation. On expose dans cette thèse les stratégies mises en place afin de rendre possible le calcul de ces caractères. On obtient une décomposition complète en composantes irréductibles pour $\mathrm{GL_r}$ et $\mathbb{S}_\mathrm{n}$ des espaces étudiés en exploitant la quasi-commutativité des dérivées partielles et des opérateurs de polarisation, l'idempotent de Young et la graduation par le degré. En exploitant également les symétries sur les jeux de variables et les antisymétries qui apparaissant naturellement dans les polynômes considérés, on réduit encore la difficulté du calcul. De plus, on montre que lorsque γ est un vecteur d'entier, on peut se ramener au cas d'un partage. Enfin, on présente le code implémenté en utilisant les stratégies décrites ainsi que les caractères obtenus.

ABSTRACT

This thesis is on the computation of the character of subspaces of harmonic multivariate polynomials on multiple sets of variables by the simultaneous action of the general linear group $\mathrm{GL_r}$ and the symmetric group $\mathbb{S}_\mathrm{n}$. The subspaces we are interested in are spanned by a Vandermonde like determinant associated to a diagram γ and closed by derivation and polarization. In this thesis, we discuss the strategies used to make the computation of those characters possible. We obtain a complete decomposition into $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-irreducibles of those spaces by exploiting the quasi-commutativity of the partial derivatives and the polarization operators, the Young idempotent and the graduation by the degree. By also exploiting the symmetries on sets of variables and antisymmetries that naturally appear in the polynomials, the computation complexity is reduced again. Moreover, we show that when γ is a vector of integers, we can reduce to the partition case. Finally, we present the implemented code using the described strategies as well as the characters we managed to compute.

INTRODUCTION

Dans un article de 1987, I. Macdonald décrit une nouvelle famille de fonctions symétriques aujourd'hui connues sous le nom de polynômes de Macdonald, (Macdonald, 1987). L'expression de ces fonctions dans une base des fonctions de Schur transformées fait apparaître des coefficients $K_{\lambda,\mu}(q,t)$ qui semblent posséder des propriétés remarquables. I. Macdonald formule alors la conjecture qui affirme que les $K_{\lambda,\mu}(q,t)$ sont des polynômes en q et t à coefficients entiers positifs. Comme l'explique M. Haiman dans (Haiman, 2003), afin de prouver cette conjecture, A. Garsia et M. Haiman proposent, dans (Garsia et Haiman, 1993), une interprétation d'une version modifiée $\widetilde{H}_\mu$ des polynômes de Macdonald, comme transformée de Frobenius graduée du caractère de modules $\mathcal{M}_\mu$, aujourd'hui connus sous le nom de modules de Garsia-Haiman. Ils conjecturent également que $\dim(\mathcal{M}_\mu) = n!$. Cette conjecture devient connue sous le nom de conjecture $n!$.

Les modules de Garsia-Haiman sont tous des sous-espaces du plus large module des polynômes harmoniques diagonaux (en deux jeux de variables). En 2002, M. Haiman prouve la conjecture $n!$ qui devient alors le théorème $n!$ (Haiman, 2002). Dans le même article, il démontre également que la transformée de Frobenius graduée de l'espace des polynômes harmoniques diagonaux en deux jeux de variables est donnée par $\nabla(e_n)$, où ∇ désigne un opérateur introduit par F. Bergeron et A. Garsia (Bergeron et Garsia, 1996), qui admet les $\widetilde{H}_\mu$ comme fonctions propres. Cependant, l'impact des modules de Garsia-Haiman et des polynômes harmoniques diagonaux ne se limite pas à la démonstration de la conjecture $n!$. Comme le dit J. Haglund dans la préface de son livre (Haglund, 2008), l'étude des polynômes

de Macdonald et de ces modules a révélée des liens avec la géométrie algébrique et les schémas de Hilbert (Haiman, 1999), la physique mathématique et d'autres objets connus de combinatoire algébrique. L'étude des polynômes harmoniques diagonaux s'est depuis élargie au contexte de plusieurs jeux de variables, mais la dimension de ces espaces rend les calculs difficiles et les conjectures ardues à formuler.

C'est dans ce contexte que s'inscrit cette thèse, le but étant de développer des outils de calcul pour les $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractères des espaces de polynômes harmoniques diagonaux pour de plus grands nombres de jeux de variables. On veut également étudier d'autres espaces de polynômes multivariés en plusieurs jeux de variables, construits à partir d'un déterminant de type Vandermonde et clos par dérivation et polarisation, de manière analogue aux polynômes harmoniques diagonaux et aux modules de Garsia-Haiman. Dans cette thèse, on travaille donc sur des espaces de polynômes multivariés avec entre autre des variables dites *inertes* qui se comportent comme des constantes à la dérivation. Le but du travail mené dans ce doctorat est de développer une stratégie mathématique qui permet de calculer informatiquement les caractères de ces espaces. Ces calculs sont difficiles à réaliser à cause de la dimension des espaces considérés et de la taille (degré et nombre de termes) des polynômes qu'ils contiennent. Il faut donc bien comprendre comment ces espaces sont construits et mettre en place différentes techniques qui rendent le calcul faisable en un temps raisonnable et en prenant en compte les limites de mémoire d'un ordinateur.

La principale contribution de cette thèse est double. Tout d'abord, on montre que l'on peut obtenir les caractères des espaces étudiés en les décomposant en $(\mathrm{GL}_r \times \mathbb{S}_\mathrm{n})$-composantes irréductibles fermées par polarisation (théorème 4.2). Dans un deuxième temps, en utilisant les résultats théoriques établis, on décrit

une implémentation d'un code modulable et réutilisable qui permet de calculer les $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractères des espaces mentionnés précédemment jusqu'à 5 jeux de variables (plus un jeu de variables dites *inertes*). De plus, on prouve que dans le cas d'un espace associé à un vecteur d'entiers, on peut se ramener au cas d'un partage, obtenu en réordonnant les parts du vecteur. On montre ainsi que dans le cas de diagrammes, c'est-à-dire de sous-ensembles de cases de $\mathbb{N} \times \mathbb{N}$, justifiés à gauche, on peut se réduire uniquement à l'étude des espaces associés à des partages. Enfin, quelques résultats sur le cas de diagrammes qui ne sont ni des partages, ni des vecteurs d'entiers, sont présentés dans le dernier chapitre.

Dans le chapitre 1, on rappelle des notions de combinatoire algébrique nécessaires au développement des chapitres suivants. On rappelle entre autres ce que sont les partages, les chemins de Dyck et les fonctions symétriques. C'est également dans ce chapitre que l'on rappelle des définitions classiques sur les anneaux de polynômes gradués, les définitions de polynômes harmoniques et de polynômes harmoniques diagonaux. Les concepts de théorie de la représentation et des caractères sont rappelés au chapitre 2. On se concentre plus particulièrement sur le groupe symétrique $\mathbb{S}_\mathrm{n}$ et sur le groupe général linéaire $\mathrm{GL_r}$ ainsi que sur leurs représentations irréductibles (polynomiales dans le cas de $\mathrm{GL_r}$). Dans le cas du groupe symétrique, on introduit un projecteur, l'idempotent de Young. Dans la première partie du chapitre 3, on présente le contexte et les résultats déjà connus sur les polynômes harmoniques diagonaux. Dans une deuxième partie, on expose les liens avec l'algèbre elliptique de Hall et on motive l'étude d'espaces généralisant les polynômes harmoniques diagonaux au cas à plusieurs jeux de variables. Le chapitre 4 décrit les stratégies de calculs mises en place pour calculer les caractères voulus. On voit, en particulier, comment obtenir ces caractères en utilisant les projecteurs de Young, la graduation par le degré, la quasi-commutativité des opérateurs utilisés dans la construction des espaces et une borne de stabilité sur

le nombre de jeux de variables. De plus, on voit comment améliorer l'efficacité du calcul en exploitant les symétries et antisymétries qui apparaissent dans ces espaces. Pour finir, on utilise les stratégies discutées pour obtenir une formule explicite du caractère pour deux types de partages et on montre que dans le cas de vecteurs d'entiers, on peut toujours se ramener à un partage. Enfin, on termine cette thèse en présentant, dans le chapitre 5, le code implémenté d'un point de vue plus informatique ainsi que quelques tests de temps de calcul. On y retrouve également les résultats obtenus en exploitant le code implémenté et on aborde le cas de diagrammes plus généraux et des essais de calculs effectués pour ces diagrammes. En annexe, on présente les explications permettant d'accéder au code implémenté dans le cadre de cette thèse, des tables de résultats obtenus grâce à ce code ainsi que des exemples de calculs effectués.

CHAPITRE I

PRÉLIMINAIRES

Le but de ce chapitre est de rappeler des notions et résultats de base et de fixer les notations employées. Ces notations et résultats se retrouvent pour la plupart dans (Macdonald, 1995).

1.1 Partages et tableaux de Young

Un *partage* $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_k)$, d'un entier n, est une suite décroissante de *parts* entières $\lambda_j > 0$, telle que $\sum_{j=1}^{k} \lambda_j = n$. On note $\lambda \vdash n$ ou $|\lambda| = n$. De plus, on dit que $\ell(\lambda) = k$ est la *longueur* de λ. Il existe plusieurs façons d'ordonner les partages, on donne ici deux ordres possibles, l'ordre *lexicographique*, noté $<$, qui est un ordre total et l'ordre de la *dominance*, noté $\preccurlyeq$, qui est un ordre partiel. Soit λ et μ deux partages de n, $\lambda < \mu$ si λ est avant μ dans l'ordre lexicographique. Autrement dit, $\lambda < \mu$ si et seulement si $\lambda_j < \mu_j$ pour j le plus petit entier tel que $\lambda_j \neq \mu_j$. D'autre part, $\lambda \preccurlyeq \mu$ si et seulement si pour tout $1 \leq N \leq k$, $\lambda_1 + \lambda_2 + \cdots + \lambda_N \leq \mu_1 + \mu_2 + \cdots + \mu_N$.

Pour $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k) \in \mathbb{N}_+^k$ (resp. $\mathbb{N}^k$) tel que $\sum_{j=1}^{k} \alpha_j = n$, on dit que α est une *composition* (resp. un *vecteur d'entiers*) de (*poids*) n. On note $\lambda(\alpha)$ le partage obtenu en ordonnant les parts de α (et en supprimant les 0 dans le cas où α est

un vecteur d'entiers). Enfin, pour α une composition de n ou un vecteur d'entiers de poids n, on note $|\alpha| = n$.

On omet parfois les parenthèses dans la notation des partages, des compositions et des vecteurs d'entiers lorsque cela simplifie l'écriture et ne porte pas à confusion. Par exemple, les partages de 3 sont $(3) = 3$; $(2,1) = 21$ et $(1,1,1) = 111$. Les compositions de 3 sont 3, 21, 12 et 111. Les vecteurs d'entiers de poids 3 sont : 3, 21, 12, 111, 03, 201, etc. De plus $\lambda(12) = \lambda(201) = 21$.

Un *diagramme* est un sous-ensemble fini de $\mathbb{N} \times \mathbb{N}$ de *cases* (i,j). Traditionnellement, on représente chaque case (i,j) par un carré 1×1 dont le coin inférieur gauche est de coordonnées (i,j). La *colonne* i d'un diagramme est l'ensemble des cases dont la première coordonnée est i et la ligne j est l'ensemble des cases dont la deuxième coordonnée est j. Un partage λ est souvent présenté sous forme d'un *diagramme de Ferrer*, c'est-à-dire du sous-ensemble de $\mathbb{N} \times \mathbb{N}$ constitué des cases (i,j) où $0 \leq j \leq k-1$ et $0 \leq i \leq \lambda_{j+1} - 1$. Le diagramme de Ferrer de λ contient donc λ_{j+1} cases dans la ligne j pour tout $0 \leq j \leq \ell(\lambda)-1$, en numérotant les lignes de bas en haut (voir figure 1.1). Par abus de notation, on dénote par λ le partage et son diagramme associé donc λ est à la fois un k-uplet et un sous-ensemble de $\mathbb{N}^2$.

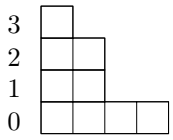


FIGURE 1.1 Diagramme associé au partage $\lambda = 4221$

Le *conjugué* λ' d'un partage λ est le partage formé des cases (j,i) avec $(i,j) \in \lambda$. Autrement dit, chaque part λ'_j de λ' est égale au nombre de cases dans la colonne j de λ. Par exemple, le conjugué de $\lambda = 4221$ est $\lambda' = 4311$.

Notons qu'une composition ou un vecteur d'entiers $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ peut également être représenté sous forme de diagramme en suivant des règles analogues à celles des partages. Ainsi, de bas en haut, la ligne j contient α_j cases justifiées à gauche. Le diagramme associé à α est donc constitué de l'ensemble des cases $\{(i, j) \mid 0 \leq j \leq k-1, 0 \leq i \leq \alpha_{j+1} - 1\}$.

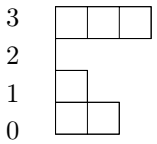


FIGURE 1.2 Diagramme associé au vecteur d'entier $\alpha = 2103$

Un *tableau de Young* est une fonction $\tau : \lambda \to \{1, 2, \ldots, N\} \subseteq \mathbb{N}$ décrivant un *remplissage* du diagramme λ par des entiers inférieurs ou égaux à N. Lorsque les valeurs sont strictement croissantes sur les colonnes du bas vers le haut et faiblement croissantes sur lignes de gauche à droite, on dit que le tableau est *semi-standard*. Autrement dit, dans le cas d'un remplissage semi-standard, on a que $\tau(i, j) < \tau(i', j)$ pour $i < i'$ et $\tau(i, j) \leq \tau(i, j')$ pour $j < j'$ (voir figure 1.3). De plus, si $\tau : \lambda \to \{1, 2, \ldots n\}$ est bijectif, alors le tableau est dit *standard* (voir figure 1.4). On note SSYT(λ) l'ensemble des tableaux de Young semi-standards de forme λ et SYT(λ) l'ensemble des tableaux de Young standards de forme λ.

7			
5	5		
2	3		
1	1	2	4

FIGURE 1.3 $\tau \in$ SSYT(4221)

8			
5	9		
3	6		
1	2	4	7

FIGURE 1.4 $\tau \in$ SYT(4221)

On introduit un ordre sur les cases de λ et on écrit $(a, b) < (c, d)$ si $b < d$ ou si $b = d$ et $a < c$. Le *tableau canonique* de forme λ est le tableau standard de forme

λ dont les cases sont numérotées de 1 à n dans l'ordre (voir figure 1.5).

9			
7	8		
5	6		
1	2	3	4

FIGURE 1.5 Tableau canonique de forme 4221

1.2 Chemins de Dyck et diagrammes associés

En plus des diagrammes de Ferrer et des diagrammes associés à des vecteurs d'entiers, on définit maintenant des diagrammes associés des chemins de Dyck dans une grille rectangulaire.

On considère la *grille* de taille $m \times n$, c'est-à-dire le sous-ensemble de $\mathbb{N} \times \mathbb{N}$ défini par $\{(i,j) \mid 0 \leq i \leq m \text{ et } 0 \leq j \leq n\}$. Un *chemin de Dyck* dans cette grille est une suite de pas sud $(0,-1)$ et est $(1,0)$ allant de $(0,n)$ à $(m,0)$ qui reste toujours en dessous de la droite d d'équation $y = -\frac{n}{m}x + n$. La droite d est aussi appelée *diagonale*. Notons que l'on a choisi de définir les chemins de Dyck en fonction de la diagonale $y = -\frac{n}{m}x+n$ et non de la diagonale $y = \frac{n}{m}x$ comme c'est généralement le cas. On a fait ce choix car, de cette façon, les chemins de Dyck obtenus définissent des partages pour lesquels le diagramme de Ferrer correspondant apparaît sur la représentation du chemin. Le chemin de Dyck formé de a_1 pas sud, suivis de b_1 pas est, suivis de a_2 pas sud, etc. est noté $S^{a_1}E^{b_1}S^{a_2}E^{b2}\ldots S^{a_k}E^{b_k}$, avec $\sum_i a_i = n$ et $\sum_i b_i = m$.

Un chemin de Dyck dans une grille $m \times n$ induit un diagramme $\gamma(m,n)$ comme suit. On considère le chemin de Dyck le plus proche de la diagonale, c'est-à-dire le chemin de Dyck pour lequel il n'existe pas de point entier $(i,j) \in \mathbb{N} \times \mathbb{N}$ contenu entre le chemin et la diagonale. On met en évidence toutes les cases situées immédiatement à droite d'un pas sud et on les fait tomber jusqu'au bas de la grille.

On obtient ainsi le diagramme constitué des cases (i, j) avec $0 \leq j \leq a_i - 1$ et $0 \leq i \leq n - 1$.

Exemple 1.1.

On construit les diagrammes pour $(m, n) = (3, 4)$, $(m, n) = (4, 6)$ *et* $(m, n) = (7, 4)$.

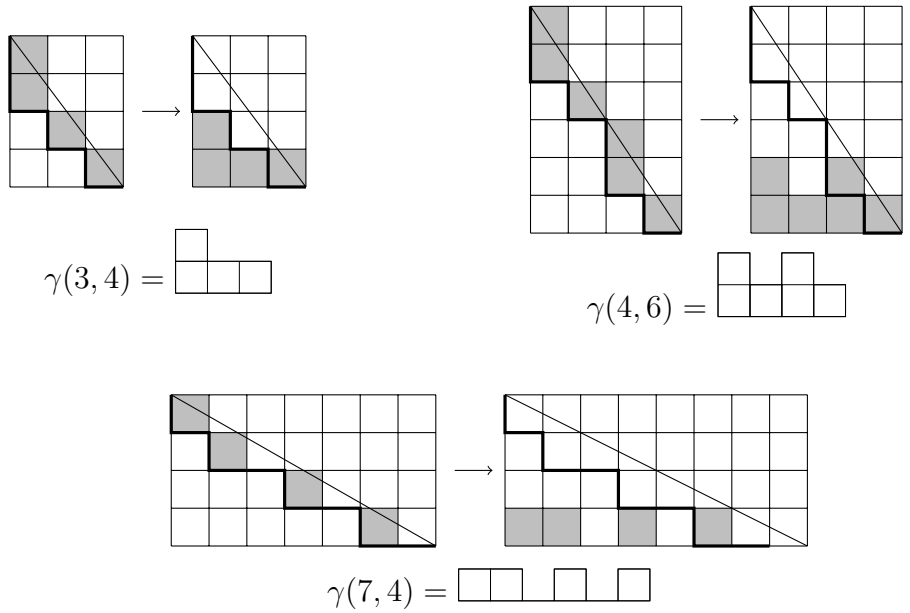


FIGURE 1.6 Exemples de diagrammes

On remarque que dans certains cas $\gamma(m, n)$ correspond au diagramme d'un partage. C'est le cas pour certaines valeurs de m et n pour lesquelles $m \leq n$. Dans l'exemple 1.1, le cas $m = 4$ et $n = 6$ est l'exemple avec $m \leq n$ et $m + n$ minimal pour lequel $\gamma(m, n)$ n'est pas un partage. Lorsque $m = n$, on obtient le partage formé d'une seule part de taille n. Et enfin, dans le cas $m > n$, on obtient n cases avec des coordonnées de la forme $(a_i, 0)$, avec $0 \leq a_i \leq m$ et $0 \leq i \leq n - 1$. C'est le cas de $\gamma(7, 4)$ dans l'exemple 1.1. Pour finir, notons qu'il possible lorsque $m \geq n$ que deux chemins de Dyck différents induisent le même diagramme. Dans ce cas,

la valeur de n est identique pour les deux diagrammes, la valeur de m diffère de 1 et l'un des chemins de Dyck est composé d'un pas $(1, 0)$ supplémentaire à la fin. On a par exemple $\gamma(8, 4) = \gamma(9, 4)$.

1.3 Anneaux de polynômes

On note $R = \mathbb{Q}[X]$ l'anneau des polynômes en r jeux de n variables à coefficients dans $\mathbb{Q}$ muni de l'addition et de la multiplication usuelle sur les polynômes. Les jeux de variables sont représentés dans une matrice X de taille $r \times n$ où chaque ligne correspond à l'un de ces jeux de n variables.

$$X := \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{r,1} & x_{r,2} & \cdots & x_{r,n} \end{pmatrix}$$

On dénote par $\boldsymbol{x}_i$ le i-ème jeu de variables de X, c'est donc dire que $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ pour $1 \leq i \leq r$. De manière générale, les jeux de variables sont notés par des caractères gras et les variables par des caractères usuels. Un monôme dans R est donné par $X^\alpha = \boldsymbol{x}_1^{\alpha_1} \boldsymbol{x}_2^{\alpha_2} \cdots \boldsymbol{x}_n^{\alpha_n}$ où $\alpha \in \mathbb{N}^{r \times n}$ et α_i correspond à la i-ème ligne de α de sorte que $\boldsymbol{x}_i^{\alpha_i} = x_{i1}^{\alpha_{i1}} x_{i2}^{\alpha_{i2}} x_{in}^{\alpha_{in}}$. Le *degré* de X^α, noté $\deg(X^\alpha)$, est un vecteur d'entiers $\boldsymbol{d} \in \mathbb{N}^r$ tel que $\boldsymbol{d} = (|\alpha_1|, |\alpha_2|, \ldots, |\alpha_n|)$ et le *degré total* de X^α est $\mathrm{Deg}(X^\alpha) = |\boldsymbol{d}| = d$. Le degré total d'un polynôme p de R est

$$\mathrm{Deg}(p) = \max_{X^\alpha \in p} \mathrm{Deg}(X^\alpha)$$

et p est dit *homogène* de degré total d si tous ses termes sont de degré d et *multihomogène* de degré $\boldsymbol{d}$ si tous ses termes sont de degré $\boldsymbol{d}$. Par exemple, $x_{11}^2 x_{21} + x_{12}^2 x_{22}$ est homogène de degré total 3 et multihomogène de degré $(2, 1)$ mais $x_{11}^2 x_{21} + x_{12} x_{22}^2$ est uniquement homogène de degré total 3.

Soit M un monoïde commutatif (souvent $\mathbb{N}^k$ dans ce qui suit). Un anneau M-*gradué* (on dit souvent simplement *gradué*) est un anneau A qui admet une décomposition en somme directe $A = \bigoplus_i A_i$ telle que $A_i A_j \subseteq A_{i+j}$ pour tout $i, j \in M$. On définit alors la notion de sous-anneau gradué de manière naturelle. L'anneau R est gradué par le degré. En effet, le produit d'un polynôme multihomogène de degré $\boldsymbol{d}_1$ par un polynôme multihomogène de degré $\boldsymbol{d}_2$ est un polynôme multihomogène de degré $\boldsymbol{d}_1 + \boldsymbol{d}_2$. On a donc $R = \bigoplus_{\boldsymbol{d} \in \mathbb{N}^r} R^{\langle \boldsymbol{d} \rangle}$ où $R^{\langle \boldsymbol{d} \rangle}$ est la *composante multihomogène* de degré $\boldsymbol{d}$, c'est-à-dire l'ensemble des polynômes de R multihomogènes de degré $\boldsymbol{d}$. Ainsi, tout polynôme $p \in R$ s'écrit sous la forme $p = p_1 + \cdots + p_k$ où les p_i sont des éléments multihomogènes appartenant à des composantes multihomogènes distinctes. On dit également que p_i est une *composante multihomogène* de p. L'anneau R est également gradué par le degré total : $R = \bigoplus_{d \in \mathbb{N}} R^{\langle d \rangle}$ où $R^{\langle d \rangle}$ est l'ensemble de polynômes de degré total d. La graduation par le degré constitue un raffinement de la graduation simple par le degré total. En effet,

$$R^{\langle d \rangle} = \bigoplus_{\substack{\boldsymbol{d} \in \mathbb{N}^r \\ |\boldsymbol{d}| = d}} R^{\langle \boldsymbol{d} \rangle}.$$

De plus, toutes les remarques précédentes sur les composantes multihomogènes s'appliquent également pour la graduation par le degré total et on parle alors de *composantes homogènes*. Enfin, un idéal $I \subseteq R$ est dit *multihomogène* (resp. *homogène*) si pour tout $q \in I$, les composantes multihomogènes (resp. homogènes) de q appartiennent aussi à I. Si I est multihomogène (resp. homogène), alors R/I est aussi un anneau gradué par le degré (resp. degré total). Pour plus de détails sur les anneaux gradués et les anneaux de polynômes, voir (Lang, 2002).

Dans le cas à un jeu de variables, on note $\mathbb{Q}[\boldsymbol{x}]$ l'anneau des polynômes en les variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ à coefficients dans $\mathbb{Q}$. Bien évidemment, l'anneau de polynômes $\mathbb{Q}[\boldsymbol{x}]$ est gradué par le degré.

Pour finir, dans les prochaines sections et prochains chapitres, nous serons amenés à dériver des polynômes, aussi pour alléger la notation, on écrit $\partial x^k := \frac{\partial^k}{\partial x^k}$ pour la k-ième dérivée partielle en la variable x. De plus, pour $\alpha \in \mathbb{N}^n$, on écrit $\partial \boldsymbol{x}^\alpha := \partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_n^{\alpha_n}$.

1.4 Fonctions symétriques

Rappelons qu'une permutation σ dans $\mathbb{S}_n$ est une bijection de l'ensemble $\{1, 2, \dots, n\}$ vers lui-même. L'ensemble des permutations $\mathbb{S}_n$ muni de la composition forme un groupe. On rappelle également qu'une *inversion* dans une permutation σ est un couple (i, j) tel que $i < j$ et $\sigma(i) > \sigma(j)$. On dénote par $\mathrm{inv}(\sigma)$ le nombre d'inversions de σ et le *signe* d'une permutation σ est $\mathrm{sgn}(\sigma) = (-1)^{\mathrm{inv}(\sigma)}$.

On considère l'action par permutation du groupe des permutations $\mathbb{S}_\mathrm{n}$ sur le vecteur de variables $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$. Pour ce faire, on considère σ comme matrice de permutation, et σ agit par multiplication à droite. Autrement dit, $\boldsymbol{x}.\sigma = (x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(n)})$. On étend cette action aux polynômes dans $\mathbb{Q}[\boldsymbol{x}]$ en posant que $f(\boldsymbol{x}) \mapsto f(\boldsymbol{x}.\sigma)$.

Définition 1.1. *Soit $f \in \mathbb{Q}[\boldsymbol{x}]$. On dit que f est* symétrique *si $f(\boldsymbol{x}.\sigma) = f(\boldsymbol{x})$ pour tout $\sigma \in \mathbb{S}_\mathrm{n}$ et* antisymétrique *si $f(\boldsymbol{x}.\sigma) = \mathrm{sgn}(\sigma) f(\boldsymbol{x})$ pour tout $\sigma \in \mathbb{S}_\mathrm{n}$.*

Le sous-ensemble des polynômes symétriques en n variables dans $\mathbb{Q}[\boldsymbol{x}]$ est fermé pour l'addition et la multiplication. On dénote par $\Lambda(\boldsymbol{x})$ l'anneau correspondant. Pour $\boldsymbol{x} \subseteq \boldsymbol{x}'$, on a un homomorphisme surjectif naturel obtenu en annulant les variables de $\boldsymbol{x}'$ n'apparaissant pas dans $\boldsymbol{x}$:

$$\mathbb{Q} \xleftarrow[x_1=0]{} \Lambda(x_1) \xleftarrow[x_2=0]{} \Lambda(x_1, x_2) \xleftarrow[x_3=0]{} \Lambda(x_1, x_2, x_3) \xleftarrow[x_4=0]{} \cdots \tag{1.1}$$

Ces morphismes sont compatibles avec la graduation par le degré. Autrement dit, les projections de (1.1) sont des morphismes d'anneau gradué. Toute équation dans $\Lambda(x_1, x_2, \cdots, x_N)$ se spécialise en une équation dans $\Lambda(x_1, x_2, \cdots, x_n)$ en fixant $x_k = 0$ pour $n < k \leq N$. On peut alors écrire $\Lambda(x_1, x_2, \cdots, x_n) = \Lambda(x_1, x_2, \cdots, x_n, 0, \cdots, 0)$. Ceci permet de considérer la limite inverse

$$\Lambda := \varprojlim_{n \to \infty} \Lambda(\boldsymbol{x}),$$

c'est-à-dire qu'on passe au contexte d'une infinité dénombrable de variables. On parle alors de *fonctions symétriques* et Λ est l'anneau des fonctions symétriques. C'est un anneau gradué par le degré $\Lambda = \bigoplus_{d \geq 0} \Lambda^{(d)}$, où $\Lambda^{(d)}$ est l'ensemble des combinaisons linéaires de *monomiales* indicées par un partage $\lambda \vdash d$,

$$m_\lambda(\boldsymbol{x}) := \sum_{\substack{\alpha \in \mathbb{N}^\infty \\ \lambda(\alpha) = \lambda}} \boldsymbol{x}^\alpha. \tag{1.2}$$

On rappelle brièvement la définition de plusieurs bases usuelles de l'anneau des fonctions symétriques. Pour une part k, la fonction symétrique $p_k(\boldsymbol{x}) = p_k := m_k(\boldsymbol{x}) = x_1^k + x_2^k + \cdots + x_n^k$ est appelée *somme de puissances*. On étend ensuite aux partages de façon multiplicative en posant $p_\lambda := p_{\lambda_1} p_{\lambda_2} \ldots p_{\lambda_\ell}$. On dit aussi que p_k est une base *multiplicative*. Les fonctions symétriques *élémentaires* sont aussi définies multiplicativement en posant d'abord, pour $k \in \mathbb{N}$, que $e_k(\boldsymbol{x}) = e_k := m_{1^k}(\boldsymbol{x})$, puis en étendant aux partages en posant $e_\lambda := e_{\lambda_1} e_{\lambda_2} \ldots e_{\lambda_\ell}$. Il en va de même pour les fonctions symétriques *homogènes*, on pose $h_k(\boldsymbol{x}) = h_k := \sum_{\lambda \vdash k} m_\lambda(\boldsymbol{x})$, puis $h_\lambda := h_{\lambda_1} h_{\lambda_2} \ldots h_{\lambda_\ell}$.

Exemple 1.2.

$$m_{21}(x_1, x_2, x_3) = x_1^2 x_2 + x_1^2 x_3 + x_1 x_2^2 + x_2^2 x_3 + x_1 x_3^2 + x_2 x_3^2$$

$$m_{411}(x_1, x_2, x_3) = x_1^4 x_2 x_3 + x_1 x_2^4 x_3 + x_1 x_2 x_3^4$$

$$p_2(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2$$
$$p_{211}(x_1, x_2, x_3) = (x_1^2 + x_2^2 + x_3^2)(x_1 + x_2 + x_3)^2$$

$$e_2(x_1, x_2, x_3) = x_1 x_2 + x_1 x_3 + x_2 x_3$$
$$e_{21}(x_1, x_2, x_3) = (x_1 x_2 + x_1 x_3 + x_2 x_3)(x_1 + x_2 + x_3)$$

$$h_2(x_1, x_2, x_3) = x_1^2 + x_1 x_2 + x_1 x_3 + x_2^2 + x_2 x_3 + x_3^2$$
$$h_{22}(x_1, x_2, x_3) = (x_1^2 + x_1 x_2 + x_1 x_3 + x_2^2 + x_2 x_3 + x_3^2)^2$$

La dernière base que nous définissons pour l'instant revêt une grande importance pour la suite, il s'agit de la base des *fonctions de Schur*. Il en existe plusieurs descriptions équivalentes. On utilise ici comme définition la description combinatoire que l'on retrouve dans plusieurs ouvrages incluant (Stanley, 1999).

Soit λ un partage de n et $\tau : \lambda \to \{1, 2, \ldots, k\}$ un remplissage quelconque de λ. Alors l'*évaluation monomiale* de τ est donnée par

$$x_\tau := \prod_{c \in \lambda} x_{\tau(c)}$$

où $\tau(c)$ est la valeur attribuée à la case c dans le remplissage τ. Par exemple, l'évaluation monomiale du tableau de la figure 1.3 est $x_\tau = x_1^2 x_2^2 x_3 x_4 x_5^2 x_7$.

Définition 1.2. *Soit λ un partage de n et $\boldsymbol{x} = (x_1, x_2, \ldots)$, alors les* fonctions de Schur *sont définies par*

$$s_\lambda(\boldsymbol{x}) := \sum_{\tau \in \mathrm{SSYT}(\lambda)} x_\tau \tag{1.3}$$

Exemple 1.3. *Tous les tableaux semi-standards de forme $\lambda = (2, 1)$ à valeur dans $\{1, 2, 3\}$ sont les suivants :*

$$\begin{array}{|c|c|}\hline 2 & \\ \hline 1 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 3 & \\ \hline 1 & 1 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 2 & \\ \hline 1 & 2 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 3 & \\ \hline 1 & 2 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 2 & \\ \hline 1 & 3 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 3 & \\ \hline 1 & 3 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 3 & \\ \hline 2 & 2 \\ \hline \end{array} \quad \begin{array}{|c|c|}\hline 3 & \\ \hline 2 & 3 \\ \hline \end{array}$$

On a alors $s_{21}(x_1, x_2, x_3) = x_1^2x_2 + x_1^2x_3 + x_1x_2^2 + 2x_1x_2x_3 + x_1x_3^2 + x_2^2x_3 + x_2x_3^2$.

Un élégant argument combinatoire permet de montrer que $s_\lambda(\boldsymbol{x})$ est symétrique (Stanley, 1999). Pour d'autres descriptions équivalentes des fonctions de Schur, voir (Macdonald, 1995).

Le produit scalaire usuel sur les fonctions symétriques, aussi dit de Hall, est défini dans (Macdonald, 1995) par

$$\langle p_\lambda, p_\mu \rangle = \delta_{\lambda,\mu} z_\lambda$$

où $z_\lambda := 1^{d_1} d_1! 2^{d_2} d_2! \ldots j^{d_j} d_j!$ avec d_i le nombre de parts de taille i dans λ, et où

$$\delta_{\lambda,\mu} = \begin{cases} 1 & \text{si } \lambda = \mu \\ 0 & \text{sinon.} \end{cases}$$

Il est bien connu que fonctions de Schur sont orthogonales pour ce produit scalaire.

Le *pléthysme*, $f[A]$, consiste à considérer f comme opérateur sur les éléments A d'une $\mathbb{Q}$-algèbre $\mathcal{A}$. Il est définit à partir des règles suivantes. Soient $A, B \in \mathcal{A}$ et f et g des fonctions symétriques, alors

i) $p_k[\boldsymbol{x}] = x_1^k + x_2^k + ... + x_n^k$

ii) $p_k[c] = c$ si $c \in \mathbb{Q}$

iii) $(f + g)[A] = f[A] + g[A]$

iv) $(fg)[A] = f[A]g[A]$

v) $p_k[A + B] = p_k[A] + p_k[B]$

vi) $p_k[AB] = p_k[A]p_k[B]$

Soit $\mathbb{Q}(q, t)$ le corps des fractions rationnelles en les variables q et t à coefficients dans $\mathbb{Q}$. Dans le contexte de l'anneau $\Lambda(q, t)$ des fonctions symétriques à coeffi-

cients dans $\mathbb{Q}(q,t)$, I. Macdonald définit, dans (Macdonald, 1995), de nouvelles bases des fonctions symétriques que l'on appelle les *polynômes de Macdonald*. Ceux qui nous intéressent plus particulièrement sont les *polynômes de Macdonald modifiés*, habituellement noté $\widetilde{H}_\lambda(\boldsymbol{x};q,t)$. On reprend ici la description donnée par M. Haiman dans (Haiman, 2003) (voir définition 3.5.2).

Définition 1.3. *Les polynômes de Macdonald modifiés $\widetilde{H}_\lambda(\boldsymbol{x};q,t)$ sont uniquement caractérisés par les trois conditions suivantes :*

(i) $\widetilde{H}_\lambda[(1-q)\boldsymbol{x};q,t] \in \mathbb{Q}(q,t)\{s_\mu : \mu \succcurlyeq \lambda\}$,

(ii) $\widetilde{H}_\lambda[(1-t)\boldsymbol{x};q,t] \in \mathbb{Q}(q,t)\{s_\mu : \mu \succcurlyeq \lambda'\}$ *et*

(iii) $\widetilde{H}_\lambda[1;q,t] = 1$.

On rappelle que λ' est le conjugué de λ (voir section 1.1).

Exemple 1.4.

$$\begin{aligned}\widetilde{H}_{21}(x_1,x_2,x_3) &= x_1^3 + (q+t+1)\,(x_1^2x_2 + x_1x_2^2 + x_1^2x_3 + x_2^2x_3 + x_1x_3^2 + x_2x_3^2) \\ &\quad + x_2^3 + (qt+2q+2t+1)\,x_1x_2x_3 + x_3^3 \\ &= qts_{111} + (q+t)s_{21} + s_3\end{aligned}$$

$$\begin{aligned}\widetilde{H}_3(x_1,x_2,x_3) &= x_1^3 + (q^2+q+1)\,(x_1^2x_2 + x_1x_2^2 + x_1^2x_3 + x_2^2x_3 + x_1x_3^2 + x_2x_3^2) \\ &\quad + x_2^3 + (q^3+2q^2+2q+1)\,x_1x_2x_3 + x_3^3 \\ &= q^3s_{111} + (q^2+q)s_{21} + s_3\end{aligned}$$

Pour terminer cette section, on définit brièvement un opérateur bien connu sur les fonctions symétriques. Il s'agit de l'opérateur ∇ (se lit « nabla »), introduit par F. Bergeron et A. Garsia en 1996 dans (Bergeron et Garsia, 1996). Cet opérateur est caractérisé par le fait que ses fonctions propres sont les polynômes de Macdonald $\widetilde{H}_\lambda$, avec valeur propre :

$$\nabla(\widetilde{H}_\lambda) := \prod_{(i,j)\in\lambda} q^i t^j \widetilde{H}_\lambda. \tag{1.4}$$

L'opérateur ∇ joue un rôle fondamental dans la théorie des fonctions symétriques comme on peut le voir dans (Haglund *et al.*, 2005).

1.5 Polynômes harmoniques

Dans cette section, nous introduisons les espaces de polynômes harmoniques en un jeu de variables ainsi qu'une construction simple de ces espaces. Pour plus de détails, voir (Haglund, 2008).

Définition 1.4. *Soit $f \in \mathbb{Q}[\boldsymbol{x}]$, alors f est un* polynôme $\mathbb{S}_n$-harmonique *si et seulement si, pour tout $k > 0$,*

$$\partial x_1^k f + \partial x_2^k f + \cdots + \partial x_n^k f = 0.$$

Par la suite, on dit simplement *harmonique* pour $\mathbb{S}_n$-harmonique et on note $\mathcal{H}_n$ l'espace des polynômes harmoniques en n variables. Par exemple, pour $n = 2$, les constantes et $x_1 - x_2$ sont des polynômes harmoniques, mais $x_1^2 - x_2^2$ n'est pas harmonique. De plus, pour tout n, le déterminant de Vandermonde

$$V_n := \det(x_i^a)_{\substack{1 \le i \le n \\ 0 \le a \le n-1}} = \prod_{i<j} (x_i - x_j) \tag{1.5}$$

est harmonique. En effet, il est connu que V_n est l'unique polynôme antisymétrique non nul de plus petit degré contenu dans $\mathbb{Q}[\boldsymbol{x}]$. Or, $\deg(\sum_i \partial x_i^k V_n) < \deg(V_n)$ et $\sum_i \partial x_i^k V_n$ est antisymétrique pour tout $k > 0$ donc $\sum_i \partial x_i^k V_n = 0$. Enfin, l'espace des polynômes harmoniques en deux variables $\mathcal{H}_2$ est engendré par $\{V_2 = x_1 - x_2, 1\}$. Et l'espace des polynômes harmoniques en trois jeux de variables $\mathcal{H}_3$ est engendré par $\{V_3 = -x_1^2 x_2 + x_2 x_2^2 + x_1^2 x_3 - x_2^2 x_3 - x_1 x_3^2 + x_2 x_3^2, x_1^2 - 2x_1 x_2 + 2x_2 x_3 - x_3^2, -2x_1 x_2 + x_2^2 + 2x_1 x_3 - x_3^2, x_1 - x_3, x_2 - x_3, 1\}$.

Il est facile de voir que si f est harmonique alors $\partial x_i f$ est aussi harmonique. En utilisant cette remarque avec un argument de dimension, on montre que $\mathcal{H}_n$ admet la description explicite suivante.

Théorème 1.1 (Classique)**.** *L'espace des polynômes harmoniques $\mathcal{H}_n$ est le plus petit sous-espace contenant le déterminant de Vandermonde qui soit clos par dérivation.*

De fait, il admet la base $\{\partial \boldsymbol{x}^d V_n \mid d = (d_1, d_2, \ldots, d_n), 0 \leq d_i \leq i-1\}$. L'une des versions équivalentes du théorème de Sheppard-Todd, Chevallay dans le cas du groupe symétrique affirme que $\mathbb{Q}[\boldsymbol{x}] \simeq \Lambda(\boldsymbol{x}) \otimes \mathcal{H}_n$, (Shephard et Todd, 1954) et (Chevalley, 1955) . Autrement dit, tout polynôme $f(\boldsymbol{x})$ s'écrit de façon unique sous la forme

$$f(\boldsymbol{x}) = \sum_{\substack{d=(d_1,d_2,\ldots,d_n) \\ 0 \leq d_i \leq i-1}} c_d(\boldsymbol{x}) \partial \boldsymbol{x}^d V_n,$$

où les $c_n(\boldsymbol{x})$ sont des polynômes symétriques. Pour plus de détails et la démonstration, voir (Humphreys, 1978).

1.6 Polynômes harmoniques diagonaux

Dans cette section, on travaille maintenant avec plusieurs jeux de variables et on introduit les espaces de polynômes harmoniques diagonaux. On rappelle que R est l'espace des polynômes en r jeux de n variables où X est la matrice de taille $r \times n$ dans laquelle chaque ligne correspond à l'un de ces r jeux de n variables. Les polynômes harmoniques diagonaux sont les solutions d'un système d'équations en plusieurs jeux de variables, décrit en terme des opérateurs suivants.

Définition 1.5. *Soit $\alpha \in \mathbb{N}^r$, alors D_α est l'opérateur définit par*

$$D_\alpha := \sum_{i=1}^{n} \partial x_{1,i}^{\alpha_1} \partial x_{2,i}^{\alpha_2} \ldots \partial x_{r,i}^{\alpha_r}.$$

Par exemple, pour

$$X = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix}$$

et $\alpha = (2,0,1)$, on a $D_{(2,0,1)} = \partial x_1^2 \partial z_1 + \partial x_2^2 \partial z_2 + \partial x_3^2 \partial z_3$.

Définition 1.6. *Soit $f \in R$, alors f est un* polynôme harmonique diagonal *en r jeux de n variables si pour tout $\alpha \in \mathbb{N}^r$ on a $D_\alpha f = 0$.*

On note $\mathcal{H}_n^{(r)}$ l'espace des polynômes harmoniques diagonaux en r jeux de n variables. Par exemple, $\mathcal{H}_2^{(2)}$ est engendré par $\{V_2(\boldsymbol{x}), V_2(\boldsymbol{y}), 1\}$ où $V_2(\boldsymbol{x}) = x_1 - x_2$ et $V_2(\boldsymbol{y}) = y_1 - y_2$.

Remarquons que $\mathcal{H}_n^{(1)} = \mathcal{H}_n \subseteq \mathcal{H}_n^{(r)}$. De plus, on définit les opérateurs suivants.

Définition 1.7. L'opérateur de polarisation $P^k_{\boldsymbol{u},\boldsymbol{v}}$ *est défini par*

$$P^k_{\boldsymbol{u},\boldsymbol{v}} = \sum_{i=1}^{n} v_i \partial u_i^k$$

où $\boldsymbol{u}$ et $\boldsymbol{v}$ sont deux jeux de variables de X.

Intuitivement, l'opérateur $P^1_{\boldsymbol{u},\boldsymbol{v}}$ permet de remplacer des variables de $\boldsymbol{u}$ par les variables de $\boldsymbol{v}$ correspondantes dans le polynôme sur lequel il est appliqué.

Exemple 1.5. *Par exemple, on part du déterminant de Vandermonde*

$$V_3(\boldsymbol{x}) = x_1^2 x_2 - x_1 x_2^2 - x_1^2 x_3 + x_2^2 x_3 + x_1 x_3^2 - x_2 x_3^2$$

que l'on polarise à plusieurs reprises.

$$\begin{aligned} P^1_{\boldsymbol{x},\boldsymbol{y}} V_3(\boldsymbol{x}) &= 2x_1x_2y_1 - x_2^2y_1 - 2x_1x_3y_1 + x_3^2y_1 + x_1^2y_2 - 2x_1x_2y_2 + 2x_2x_3y_2 \\ &\quad -x_3^2y_2 - x_1^2y_3 + x_2^2y_3 + 2x_1x_3y_3 - 2x_2x_3y_3 = \tfrac{1}{2}(P^1_{\boldsymbol{y},\boldsymbol{x}})^2 V_3(\boldsymbol{y}) \end{aligned}$$

$$\begin{aligned} (P^1_{\boldsymbol{x},\boldsymbol{y}})^2 V_3(\boldsymbol{x}) &= 2x_2y_1^2 - 2x_3y_1^2 + 4x_1y_1y_2 - 4x_2y_1y_2 - 2x_1y_2^2 + 2x_3y_2^2 - 4x_1y_1y_3 \\ &\quad +4x_3y_1y_3 + 4x_2y_2y_3 - 4x_3y_2y_3 + 2x_1y_3^2 - 2x_2y_3^2 = 2P^1_{\boldsymbol{y},\boldsymbol{x}} V_3(\boldsymbol{y}) \end{aligned}$$

$$(P^1_{\boldsymbol{x},\boldsymbol{y}})^3 V_3(\boldsymbol{x}) = 6y_1^2y_2 - 6y_1y_2^2 - 6y_1^2y_3 + 6y_2^2y_3 + 6y_1y_3^2 - 6y_2y_3^2 = 6V_3(\boldsymbol{y})$$

On pourrait également repartir de $V_3(\boldsymbol{y})$ et faire les polarisations inverses jusqu'à obtenir de nouveau un multiple de $V_3(\boldsymbol{x})$.

Théorème 1.2. *(Haiman, 2002) L'espace des polynômes harmoniques diagonaux* $\mathcal{H}_n^{(2)}$ *en deux jeux de variables* $\boldsymbol{x}$ *et* $\boldsymbol{y}$ *est le plus petit espace engendré par* V_n*, fermé par dérivation partielle et par les opérateurs de polarisation* $P^k_{\boldsymbol{x},\boldsymbol{y}}$ *et* $P^k_{\boldsymbol{y},\boldsymbol{x}}$*.*

Exemple 1.6. $\mathcal{H}_3^{(2)}$ *est engendré par* $\{V_3(\boldsymbol{x}), \partial x_2 V_3(\boldsymbol{x}), \partial x_2^2 V_3(\boldsymbol{x}), \partial x_3 \partial x_2^2 V_3(\boldsymbol{x}), P^1_{\boldsymbol{x},\boldsymbol{y}} V_3(\boldsymbol{x}), (P^1_{\boldsymbol{x},\boldsymbol{y}})^2 V_3(\boldsymbol{x}), P^2_{\boldsymbol{x},\boldsymbol{y}} V_3(\boldsymbol{x}), (P^1_{\boldsymbol{x},\boldsymbol{y}})^3 V_3(\boldsymbol{x}) = V_3(\boldsymbol{y}), \partial x_2 V_3(\boldsymbol{y}), \partial x_2^2 V_3(\boldsymbol{y})\}$ *où*

$V_3(\boldsymbol{x})$*,* $V_3(\boldsymbol{y})$*,* $P^1_{\boldsymbol{x},\boldsymbol{y}} V_3(\boldsymbol{x})$ *et* $(P^1_{\boldsymbol{x},\boldsymbol{y}})^2 V_3(\boldsymbol{x})$ *sont donnés à l'exemple 1.5,*

$\partial x_2 V_3(\boldsymbol{x}) = x_1^2 - 2x_1x_2 + 2x_2x_3 - x_3^2$*,*

$\partial x_2^2 V_3(\boldsymbol{x}) = 2(x_3 - x_1)$*,*

$\partial x_3 \partial x_2^2 V_3(\boldsymbol{x}) = 2$*,*

$\partial x_2^2 V_3(\boldsymbol{y}) = 2(y_3 - y_1)$*,*

$\partial x_2 V_3(\boldsymbol{y}) = y_1^2 - 2y_1y_2 + 2y_2y_3 - y_3^2$ *et*

$P^2_{\boldsymbol{x},\boldsymbol{y}} V_3(\boldsymbol{x}) = 2(x_1y_1 - x_2y_1 - x_1y_2 + x_3y_2 + x_2y_3 - x_3y_3)$*.*

On note par $\mathcal{P}(\mathcal{H}_n^{(1)})$ le plus petit espace engendré par $\mathcal{H}_n^{(1)}$ et fermé par les opérateurs de polarisation. Dans la section 4.2, on démontre que l'ordre dans lequel on applique les dérivées partielles et les opérateurs de polarisation n'a pas d'importance pour la construction de l'espace. On peut donc en conclure que $\mathcal{P}(\mathcal{H}_n^{(1)}) \subseteq \mathcal{H}_n^{(r)}$. Cependant, on ne sait pas si ces deux espaces sont les mêmes pour $r > 2$. C'est une question intéressante mais difficile qui n'est pas discutée dans cette thèse.

CHAPITRE II

REPRÉSENTATIONS LINÉAIRES DU GROUPE SYMÉTRIQUE ET DU GROUPE GÉNÉRAL LINÉAIRE

Le but de ce chapitre est de rappeler brièvement les concepts de la théorie de la représentation de $\mathbb{S}_n$ et de $\mathrm{GL_r}$ sur les corps algébriquement clos, et de la théorie des caractères nécessaires pour la suite. On trouve dans (Sagan, 2001) un exposé plus spécifiquement adapté à $\mathbb{S}_\mathrm{n}$ et on pourra se référer à (Green, 2007) pour les représentations polynomiales de $\mathrm{GL_r}$. Avant de s'intéresser aux cas de ces deux groupes, on rappelle la définition d'une représentation pour un groupe G en général.

Définition 2.1. *Soit $\mathcal{V}$ un espace vectoriel sur un corps $\mathbb{K}$ et G un groupe. Alors $\mathcal{V}$ est un G-*module *s'il existe un homomorphisme*

$$\begin{array}{rccc} \rho: & G & \to & \mathrm{GL}(\mathcal{V}) \\ & g & \mapsto & \rho(g) \end{array}$$

qui associe à chaque élément de G une matrice de $\mathrm{GL}(\mathcal{V})$.

On dit aussi que $\mathcal{V}$ est une *représentation* de G. Tout groupe admet au moins une représentation, la représentation triviale, notée $\mathbb{1}_G$, qui associe à chaque élément la matrice identité de dimension 1. De plus, on rappelle qu'une *sous-représentation* de $\mathcal{V}$ est un sous-espace vectoriel stable par l'action de G.

2.1 Représentations usuelles du groupe symétrique $\mathbb{S}_n$

Rappelons d'abord quelques représentations bien connues du groupe symétrique $\mathbb{S}_n$. Notons que la notation en cycle est utilisée pour les permutations et que ε représente la permutation identité.

Tout d'abord la *représentation régulière* (à gauche) est la représentation pour laquelle $\mathbb{S}_n$ agit sur lui-même par multiplication (à gauche). La représentation régulière ρ est donnée par

$$\begin{array}{rccc} \rho: & \mathbb{S}_n & \to & \mathrm{GL}(\mathbb{C}\,\mathbb{S}_n) \\ & \sigma & \mapsto & \rho_\sigma \end{array}$$

où $\rho_\sigma(\tau) := \sigma\tau$ pour tout $\tau \in \mathbb{S}_n$ et $\mathbb{C}\mathbb{S}_n$ désigne l'algèbre du groupe $\mathbb{S}_n$ sur $\mathbb{C}$.

Exemple 2.1. *Dans le cas de $\mathbb{S}_3$, pour $\sigma = (12)$, on a*

$$\rho_{(12)}(\varepsilon) = (12), \qquad \rho_{(12)}((13)) = (132), \qquad \rho_{(12)}((123)) = (23),$$

$$\rho_{(12)}((12)) = \varepsilon, \qquad \rho_{(12)}((23)) = (123), \qquad \rho_{(12)}((132)) = (13).$$

Bien entendu, on peut exprimer ces résultats sous forme de matrice.

$$\rho_{(12)} = \begin{array}{c} \\ \varepsilon \\ (12) \\ (13) \\ (23) \\ (123) \\ (132) \end{array} \begin{array}{c} \begin{array}{cccccc} \varepsilon & (12) & (13) & (23) & (123) & (132) \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

La *représentation signe* est donnée par

$$\begin{array}{rccc} \epsilon: & \mathbb{S}_n & \to & \mathrm{GL}_1(\mathbb{Z}_2) \\ & \sigma & \mapsto & \mathrm{sgn}(\sigma) \end{array}.$$

Dans le cas de $\mathbb{S}_3$, on a $\epsilon_\varepsilon = 1$, $\epsilon_{(12)} = -1$, $\epsilon_{(123)} = 1$.

Enfin la *représentation standard* associe chaque permutation de $\mathbb{S}_\mathrm{n}$ à sa matrice de permutation dans GL_n. Cette représentation est donnée par

$$\begin{array}{rccc} \varphi : & \mathbb{S}_\mathrm{n} & \to & \mathrm{GL}_n \\ & \sigma & \mapsto & \varphi_\sigma \end{array}$$

où φ_σ représente la matrice de permutation associée à σ.

Exemple 2.2. *Dans le cas de* $\mathbb{S}_3$*, on a*

$$\varphi_\varepsilon = \mathrm{Id}_3 \qquad \varphi_{(12)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \varphi_{(123)} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

où Id_3 *est la matrice identité de dimension 3.*

Définition 2.2. *Soit* $\mathcal{V}$ *une représentation de* G*, alors* $\mathcal{V}$ *est dite* irréductible *s'il n'existe pas de sous-*G*-module non trivial de* $\mathcal{V}$*, c'est-à-dire de sous-espace stable sous l'action de* G *qui ne soit ni le sous-espace vide, ni l'espace vectoriel* $\mathcal{V}$ *lui-même.*

Pour le groupe symétrique, les représentations triviale, signe et standard sont des représentations particulières appelées *représentations irréductibles*. Par contre, la représentation régulière n'est pas une représentation irréductible de $\mathbb{S}_\mathrm{n}$. Dans le cas de $\mathbb{S}_3$, les trois représentations irréductibles citées précédemment forment l'ensemble de toutes les représentations irréductibles de $\mathbb{S}_3$. On peut se référer à (Sagan, 2001) pour plus de détails.

Définition 2.3. *Soit* $\rho_1 : G \to \mathrm{GL}(\mathcal{V}_1)$ *et* $\rho_2 : G \to \mathrm{GL}(\mathcal{V}_2)$ *deux représentations de* G*. On dit que* ρ_1 *et* ρ_2 *sont* isomorphes *s'il existe un morphisme* $\psi : \mathcal{V}_1 \to \mathcal{V}_2$ *tel que* $\rho_2 \circ \psi = \psi \circ \rho_1$*.*

Afin de manipuler plus aisément les représentations, on utilise un outil plus puissant que les matrices et les morphismes, il s'agit du *caractère*.

Définition 2.4. *Soit ρ une représentation de G, alors le* caractère *de la représentation est la fonction $\chi_\rho : G \to \mathbb{C}$ définie par $\chi_\rho(g) = \mathrm{tr}(\rho(g))$ pour tout $g \in G$.*

Exemple 2.3. *Pour les représentations définies précédemment, on obtient respectivement $\chi_\rho((12)) = 0$; $\chi_\epsilon(\varepsilon) = 1$, $\chi_\epsilon((12)) = -1$, $\chi_\epsilon((123)) = 1$ et $\chi_\varphi(\varepsilon) = 3$, $\chi_\varphi((12)) = 1$,$\chi_\varphi((123)) = 0$.*

Le caractère d'une représentation irréductible est également appelé *caractère irréductible*. Le caractère vérifie plusieurs propriétés bien connues que l'on rappelle ici. Pour plus de détails et les démonstrations voir (Sagan, 2001).
Pour tout $\rho : G \to \mathrm{GL}(\mathcal{V})$, on a

1. $\chi_{\mathbb{1}}(g) = 1$ pour tout $g \in G$, avec $\chi_{\mathbb{1}}$ le caractère de la représentation triviale de G ;
2. $\chi_\rho(g) = \chi_\rho(hgh^{-1})$ pour tout $g, h \in G$;
3. $\chi_\rho(\varepsilon) = \dim(\mathcal{V})$;
4. $\chi_{\rho+\rho'} = \chi_\rho + \chi_{\rho'}$;
5. $\chi_\rho = \chi_{\rho'}$ si et seulement si ρ et ρ' sont isomorphes.

D'après la propriété 2 ci-dessus, il est suffisant de donner le caractère d'un élément par classe de conjugaison. De plus, le nombre de caractères irréductibles de G correspond au nombre de classes de conjugaisons de G. Pour $\mathbb{S}_\mathrm{n}$, cela signifie qu'il suffit de connaître le caractère d'une permutation de chaque type cyclique. Comme les types cycliques sont en bijection avec les partages, on indice les caractères irréductibles de $\mathbb{S}_\mathrm{n}$ par les partages de n.

La table des caractères d'un groupe permet de synthétiser toute cette information. Pour le groupe symétrique, les caractères irréductibles sont bien connus, voir par exemple (Sagan, 2001) ou (Fulton et Harris, 1991). On donne ci-dessous les tables de caractères pour $n = 3$ et $n = 4$. On rappelle que $\chi_{\mathbb{1}}$ est la représentation triviale, χ_ϵ est la représentation signe et χ_φ est la représentation standard.

TABLE 2.1 Table de caractères de $\mathbb{S}_3$

	ε	(12)	(123)
$\chi_{\mathbb{1}}$	1	1	1
χ_ϵ	1	-1	1
χ_φ	2	0	-1

TABLE 2.2 Table de caractères de $\mathbb{S}_4$

	ε	(12)	(123)	(12)(34)	(1234)
$\chi_{\mathbb{1}}$	1	1	1	1	1
χ_ϵ	1	-1	1	1	-1
χ_φ	3	1	0	-1	-1
χ_4	3	-1	0	-1	1
χ_5	2	0	-1	2	0

2.2 Représentations irréductibles de $\mathbb{S}_n$

Dans cette sous-section, on introduit les symétriseurs de Young et on décrit une façon de construire toutes les représentations irréductibles du groupe symétrique en utilisant ces symétriseurs de Young. On explore le fait que la représentation régulière contient des copies de toutes les représentations irréductibles, (Fulton et Harris, 1991). En utilisant un projecteur isotypique, on peut extraire chacune de ces représentations irréductibles. Les définitions, lemmes et démonstrations de cette sous-section sont issues de (Fulton et Harris, 1991) et (Naimark et Štern, 1982).

Soit λ un partage de n. Dans cette section, on note également par λ le tableau canonique associé au partage considéré. Une permutation $\sigma \in \mathbb{S}_n$ agit sur un tableau par permutation des valeurs des entrées du tableau. On dit d'une permutation σ qu'elle stabilise les lignes (respectivement les colonnes) si appliquée au tableau canonique λ, les entrées sont conservées dans la même ligne (respectivement colonne) que dans le tableau de départ λ mais dans un ordre qui peut être différent. On définit alors les deux sous-groupes de $\mathbb{S}_n$ suivants :

$$P_\lambda := \{\sigma \in \mathbb{S}_n \mid \sigma \text{ stabilise chacune des lignes de } \lambda\} \text{ et}$$

$$Q_\lambda := \{\sigma \in \mathbb{S}_n \mid \sigma \text{ stabilise chacune des colonnes de } \lambda\}.$$

Par exemple, pour $\lambda = 21$ avec le tableau canonique $\begin{array}{|c|c|}\hline 3 \\ \hline 1 & 2 \\ \hline \end{array}$, on a alors $P_\lambda = \{\varepsilon, (12)\}$ et $Q_\lambda = \{\varepsilon, (13)\}$.

On rappelle que l'algèbre du groupe symétrique $\mathbb{C}\,\mathbb{S}_n$ correspond aux combinaisons linéaires formelles d'éléments de $\mathbb{S}_n$ à coefficients dans $\mathbb{Q}$. On définit alors deux éléments de l'algèbre du groupe symétrique $\mathbb{C}\,\mathbb{S}_n$ correspondant à P_λ et Q_λ.

$$r_\lambda := \sum_{\sigma \in P_\lambda} \sigma \qquad \text{et} \qquad c_\lambda := \sum_{\sigma \in Q_\lambda} \operatorname{sgn}(\sigma)\sigma$$

Toujours pour $\lambda = 21$, on a alors $r_{21} = \varepsilon + (12)$ et $c_{21} = \varepsilon - (13)$. L'élément r_λ permet de symétriser par rapport aux lignes de λ et c_λ permet d'antisymétriser par rapport aux colonnes de λ. Enfin, on définit un dernier élément, appelé le *symétriseur de Young* qui est la composition des deux précédents :

$$y_\lambda := r_\lambda . c_\lambda. \tag{2.1}$$

Par exemple, pour $n = 3$, on a $y_{111} = \sum_{\sigma \in \mathbb{S}_3} \sigma$, $y_{21} = (\varepsilon + (12)).(\varepsilon - (13)) = \varepsilon + (12) - (13) - (132)$ et $y_3 = \sum_{\sigma \in \mathbb{S}_3} \mathrm{sgn}(\sigma)\sigma$. Le symétriseur de Young permet de construire les représentations irréductibles du groupe symétrique. On énonce ce résultat dans le théorème suivant.

Théorème 2.1 (Classique)**.** *Le symétriseur de Young y_λ est idempotent à un scalaire près et l'image de y_λ par multiplication à droite sur $\mathbb{C}\,\mathbb{S}_\mathrm{n}$ est une représentation irréductible $\mathcal{V}_\lambda$ de $\mathbb{S}_\mathrm{n}$. Toute représentation irréductible peut être obtenue de cette manière pour un unique partage.*

Notons que les représentations irréductibles de $\mathbb{S}_\mathrm{n}$ peuvent aussi être définies sur $\mathbb{Q}$ puisque $c_\lambda \in \mathbb{Q}\,\mathbb{S}_\mathrm{n}$. La suite de cette section est consacrée à donner une esquisse de la démonstration ce théorème. Pour plus de détails voir (Fulton et Harris, 1991) et (Naimark et Štern, 1982).

On note $A = \mathbb{C}\,\mathbb{S}_\mathrm{n}$ et $\mathcal{V}_\lambda = Ay_\lambda$. Remarquons que $P_\lambda \cap Q_\lambda = \{\varepsilon\}$. Alors pour toute permutation $\sigma \in \mathbb{S}_\mathrm{n}$, il existe au plus une façon d'écrire σ comme produit d'un élément p de P_λ et d'un élément q de Q_λ. Ainsi, on a que y_λ s'écrit

$$y_\lambda = \sum_{\sigma \in P_\lambda} \sigma . \sum_{\sigma \in Q_\lambda} \mathrm{sgn}(\sigma)\sigma = \sum_{\sigma} \pm\sigma$$

où la somme est sur toutes les permutations σ qui s'écrivent comme un produit pq avec $p \in P_\lambda$ et $q \in Q_\lambda$ et le signe ± 1 correspond au signe de q.

Lemme 2.1.

(i) Pour tout $p \in P_\lambda$, $p.r_\lambda = r_\lambda.p = r_\lambda$.

(ii) Pour tout $q \in Q_\lambda$, $\mathrm{sgn}(q)q.c_\lambda = c_\lambda.(\mathrm{sgn}(q)q) = c_\lambda$.

(iii) Soit $a \in A$ tel que $p.a.\,\mathrm{sgn}(q)q = a$ pour tout $p \in P_\lambda$ et pour tout $q \in Q_\lambda$, alors $a = ny_\lambda$ où $n \in \mathbb{C}$.

Démonstration. Pour (i) et (ii), puisque P_λ et Q_λ sont des sous-groupes de $\mathbb{S}_n$, on a bien

$$p.r_\lambda = p.\sum_{\sigma\in P_\lambda}\sigma = \sum_{\sigma\in P_\lambda} p.\sigma = \sum_{\sigma\in P_\lambda}\sigma = r_\lambda.$$

Le même argument fonctionne également pour $r_\lambda.p$ et pour c_λ puisque pour tout $\sigma, \sigma' \in \mathbb{S}_n$, $\mathrm{sgn}(\sigma).\mathrm{sgn}(\sigma') = \mathrm{sgn}(\sigma\sigma')$. Il nous reste donc à prouver (iii). Soit $a = \sum_{\sigma\in\mathbb{S}_n} a_\sigma\sigma$, on a alors

$$\sum_{\sigma\in\mathbb{S}_n} a_\sigma\sigma = a = p.a.\,\mathrm{sgn}(q)q = \sum_{\sigma\in\mathbb{S}_n}\mathrm{sgn}(q)a_\sigma p\sigma q =$$

En comparant terme à terme les deux sommes de l'égalité précédente, pour $\sigma = pq$ dans la somme de droite, on a que $\sigma = \varepsilon$ dans la somme de gauche et $\mathrm{sgn}(q)a_\varepsilon = a_{pq}$. On peut alors montrer que si $\sigma_0 \neq pq$ alors $a_{\sigma_0} = 0$ et donc on obtient que

$$a = \sum_{\sigma\in\mathbb{S}_n} a_\sigma\sigma = \sum_{p,q}\mathrm{sgn}(q)a_{pq}pq = a_\varepsilon y_\lambda.$$

□

Lemme 2.2.

(i) Si $\lambda > \mu$ alors $r_\lambda A c_\mu = \{0\}$.

(ii) Si $\lambda \neq \mu$ alors $y_\lambda y_\mu = 0$.

(iii) Pour tout $x \in A$, $y_\lambda x y_\lambda = n_\lambda y_\lambda$, pour un certain $n_\lambda \in \mathbb{C}$. En particulier, $y_\lambda y_\lambda = n_\lambda y_\lambda$, $n_\lambda \in \mathbb{C}$.

Démonstration. Pour montrer (i), commençons par remarquer que si $\lambda > \mu$ alors $r_\lambda c_\mu = 0$. En effet, puisque λ et μ sont les tableaux canoniques de formes respectives λ et μ et que $\lambda > \mu$, il existe deux entrées i et j telles que i et j se trouvent dans la même ligne dans λ et dans la même colonne dans μ. Soit $t = (ij) \in \mathbb{S}_n$ la

transposition qui échange i et j. On a alors $r_\lambda t = r_\lambda$ et $tc_\mu = -c_\mu$ d'après (i) et lemme 2.1 (ii). Donc $r_\lambda c_\mu = -r_\lambda t t c_\mu = -r_\lambda c_\mu$ d'où $r_\lambda c_\mu = 0$.

Soit $\sigma \in \mathbb{S}_n$, on note $\sigma\mu$ le tableau de forme μ construit à partir du tableau canonique en permutant les entrées selon σ. Alors $\sigma c_\mu \sigma^{-1}$ correspond à c_μ construit pour le tableau $\sigma\mu$ plutôt que pour le tableau canonique. Soit $a_\sigma\sigma \in A$, avec $a_\sigma \in \mathbb{C}$, on multiplie à droite par $a_\sigma\sigma$ les deux côtés de l'égalité précédente et on somme sur toutes les permutations σ, on obtient alors

$$\sum_\sigma r_\lambda \sigma c_\mu \sigma^{-1} a_\sigma \sigma = r_\lambda \left(\sum_\sigma a_\sigma \sigma \right) c_\mu = 0.$$

Or $\sum_\sigma a_\sigma\sigma$ est un élément quelconque de A, donc $r_\lambda A c_\mu = \{0\}$.

Pour prouver (ii) dans le cas où $\lambda > \mu$, il suffit de voir que $y_\lambda y_\mu = r_\lambda c_\lambda r_\mu c_\mu \in r_\lambda A c_\mu = \{0\}$ d'après (i).
Dans le cas où $\lambda < \mu$, on a cette fois qu'il existe deux entrées i et j telles que i et j se trouvent dans la même colonne dans λ et dans la même ligne dans μ. Soit $t = (ij) \in \mathbb{S}_n$, alors $c_\lambda t = -c_\lambda$ et $t r_\mu = r_\mu$. On a alors $y_\lambda y_\mu = r_\lambda c_\lambda r_\mu c_\mu = -r_\lambda c_\lambda t t r_\mu c_\mu = -y_\lambda y_\mu$. D'où $y_\lambda y_\mu = 0$.

Pour (iii), il faut remarquer que $y_\lambda x y_\lambda$ satisfait la condition permettant d'appliquer (iii) du lemme 2.1. En effet, pour tout $p \in P_\lambda$ et tout $q \in Q_\lambda$,

$$p.y_\lambda x y_\lambda . \operatorname{sgn}(q) q = \underbrace{p.r_\lambda}_{=r_\lambda} c_\lambda . x . r_\lambda \underbrace{c_\lambda \operatorname{sgn}(q) q}_{=c_\lambda} = y_\lambda x y_\lambda$$

d'après (i) et (ii) du lemme 2.1. Donc $y_\lambda x y_\lambda = n_\lambda y_\lambda$, $n_\lambda \in \mathbb{C}$. □

Le lemme 2.2 prouve la première partie du théorème 2.1. Il nous reste maintenant à montrer que les représentations construites avec le symétriseur de Young sont

bien des représentations irréductibles. Il est facile de voir que $\mathcal{V}_\lambda$ est un idéal à gauche de A. En effet, soit $x \in A$ et $v \in \mathcal{V}_\lambda$, alors $v = ay_\lambda$ où $a \in A$ et donc $x.v = xay_\lambda$ est aussi un élément de $\mathcal{V}_\lambda$.

Lemme 2.3. *$\mathcal{V}_\lambda$ est un idéal minimal à gauche de A.*

Démonstration. Notons tout d'abord que $y_\lambda \mathcal{V}_\lambda \subset \mathbb{C}y_\lambda$. En effet, pour tout $b \in y_\lambda \mathcal{V}_\lambda$, on a que b s'écrit $b = y_\lambda a y_\lambda$ pour un certain $a \in A$. Ainsi, d'après le lemme 2.2 (ii), $b = ny_\lambda$ avec $n \in \mathbb{C}$ et donc $b \in \mathbb{C}y_\lambda$. De plus, on a déjà vu que $\mathcal{V}_\lambda$ est un idéal à gauche de A, il nous reste donc à montrer qu'il est minimal.
Soit I_l un idéal à gauche de A contenu dans $\mathcal{V}_\lambda$. On a donc $y_\lambda I_l \subset y_\lambda \mathcal{V}_\lambda \subset \mathbb{C}y_\lambda$. Or comme $\mathbb{C}y_\lambda$ est de dimension 1, soit $y_\lambda I_l = \mathbb{C}y_\lambda$, soit $y_\lambda I_l = \{0\}$. Dans le premier cas, on a alors $\mathcal{V}_\lambda = Ay_\lambda \subset A\mathbb{C}y_\lambda = Ay_\lambda I_l \subset I_l$, la dernière inclusion découle du fait que I_l est un idéal à gauche de A. Donc $\mathcal{V}_\lambda \subset I_l$ ce qui implique que $I_l = \mathcal{V}_\lambda$. Dans le deuxième cas, on a que $I_l^2 \subset \mathcal{V}_\lambda I_l = Ay_\lambda I_l = \{0\}$ puisque par hypothèse $y_\lambda I_l = \{0\}$. Or $V_\lambda \neq \{0\}$ donc $I_l = \{0\}$.
Ainsi, $\mathcal{V}_\lambda$ est bien un idéal minimal à gauche de A. □

Comme on sait que le nombre de représentations irréductibles de $\mathbb{S}_\mathrm{n}$ est égal au nombre de classes de conjugaisons de $\mathbb{S}_\mathrm{n}$, pour obtenir un ensemble complet de représentations irréductibles de $\mathbb{S}_\mathrm{n}$, il suffit de construire des représentations irréductibles indicées par les partages de n qui soient toutes non isomorphes. Nous venons ici de construire des représentations $\mathcal{V}_\lambda$ indicées par les partages de n et nous avons montré qu'elles sont irréductibles puisque $\mathcal{V}_\lambda$ est un idéal minimal.

Lemme 2.4. *Si $\lambda \neq \mu$, alors $\mathcal{V}_\lambda \not\simeq \mathcal{V}_\mu$.*

Démonstration. Sans perte de généralité on peut supposer que $\lambda > \mu$. On a alors $r_\lambda \mathcal{V}_\mu = r_\lambda A y_\mu = r_\lambda \underbrace{A r_\mu}_{\in A} c_\mu = \{0\}$, d'après le lemme 2.2 (i). De plus, $r_\lambda \mathcal{V}_\lambda \neq \{0\}$,

en effet, $y_\lambda \in \mathcal{V}_\lambda$ et $r_\lambda y_\lambda \neq 0$.

Supposons que $\mathcal{V}_\lambda \simeq \mathcal{V}_\mu$. Alors les représentations de $\mathbb{S}_\mathrm{n}$ associées à chaque module sont équivalentes et les représentations correspondantes de A sont aussi équivalentes. On note ces représentations de A respectivement φ_λ et φ_μ. Ainsi, il existe une application bijective P qui permet de passer de $\mathcal{V}_\mu$ à $\mathcal{V}_\lambda$, $\varphi_\lambda(a) = P^{-1}\varphi_\mu(a)P$. En particulier, pour $a = r_\lambda$, on a $\varphi_\lambda(r_\lambda) = P^{-1}\varphi_\mu(r_\lambda)P$. Or, en utilisant le lemme 2.1 et les remarques précédentes, on obtient que les deux égalités suivantes sont en contradiction,

1. $\varphi_\lambda(r_\lambda)\mathcal{V}_\lambda = r_\lambda\mathcal{V}_\lambda \neq \{0\}$ et
2. $P^{-1}\varphi_\mu(r_\lambda)P\mathcal{V}_\lambda = P^{-1}\varphi_\mu(r_\lambda)\mathcal{V}_\mu P = P^{-1}r_\lambda\mathcal{V}_\mu P = \{0\}$.

On en conclut donc que $\mathcal{V}_\lambda \not\simeq \mathcal{V}_\mu$. □

Nous venons ainsi de construire un ensemble complet de représentations irréductibles de $\mathbb{S}_\mathrm{n}$. Notons que pour construire ces représentations irréductibles, nous sommes partis du tableau canonique pour chaque partage λ. Nous aurions pu utiliser un autre tableau standard de forme λ, nous aurions alors obtenu d'autres représentations irréductibles isomorphes à celles obtenues à partir du tableau canonique. Enfin, le théorème 2.1 nous dit que le symétriseur de Young est idempotent à un scalaire près. La valeur de ce scalaire est donnée par la proposition suivante dont la démonstration peut être trouvée dans (Fulton et Harris, 1991).

Proposition 2.1 (Classique)**.** *La valeur de n_λ dans le lemme 2.2 est donnée par*

$$n_\lambda = \frac{n!}{\dim \mathcal{V}_\lambda}.$$

Démonstration. Soit $F_\lambda : A \to A$ l'opérateur définit sur $A = \mathbb{C}\,\mathbb{S}_\mathrm{n}$ par $x \mapsto xy_\lambda$. D'après (iii) du lemme 2.2, $F_\lambda = n_\lambda\,\mathrm{Id}$ sur $\mathcal{V}_\lambda$. En effet, pour tout $v \in \mathcal{V}_\lambda$, il existe

$a \in A$ tel que $x = ay_\lambda$, donc $F(x) = ay_\lambda y_\lambda = n_\lambda a y_\lambda = n_\lambda x$.
Soit $a_1, a_2, \ldots, a_p$ une base de $\mathcal{V}_\lambda$, on étend cette base en une base $a_1, a_2, \ldots, a_{n!}$ de A. On a alors que la matrice associée à F_λ dans cette base est de la forme

$$\begin{pmatrix} n_\lambda \operatorname{Id} & M \\ 0 & 0 \end{pmatrix}$$

où Id est la matrice identité d'ordre $\dim \mathcal{V}_\lambda$ et M une matrice de format $\dim \mathcal{V}_\lambda \times (n! - \dim \mathcal{V}_\lambda)$. Ainsi, $\operatorname{tr}(F_\lambda) = n_\lambda \dim \mathcal{V}_\lambda$. D'autre part, on a que le coefficient de σ dans σy_λ est 1 pour tout $\sigma \in \mathbb{S}_\mathrm{n}$. Donc on a aussi que $\operatorname{tr}(F_\lambda) = n!$ puisque $\mathbb{S}_\mathrm{n}$ est la base standard de A. Finalement, on obtient bien que $n_\lambda = \dfrac{n!}{\dim \mathcal{V}_\lambda}$. □

Les caractères irréductibles de $\mathbb{S}_\mathrm{n}$ sont efficacement encodés par des fonctions symétriques, via la transformée de Frobenius définie de la façon suivante.

Définition 2.5. *La* transformée de Frobenius *est l'application qui associe à une représentation la fonction symétrique*

$$\rho \mapsto \sum_{\lambda \vdash n} \chi_\rho(\sigma_\lambda) \frac{p_\lambda}{z_\lambda}$$

où $\chi_\rho(\sigma_\lambda)$ est la valeur du caractère χ_ρ sur la permutation σ_λ de type cyclique λ et $z_\lambda := 1^{d_1} d_1! 2^{d_2} d_2! \ldots j^{d_j} d_j!$ où d_i est le nombre de parts de taille i dans λ.

On sait que dans le cas du groupe symétrique, les caractères irréductibles sont associés aux fonctions de Schur par la transformée de Frobenius. En fait, on a exactement que le caractère de la représentation irréductible $\mathcal{V}_\lambda$ est associé à la fonction de Schur s_λ. Par exemple, dans le cas de $\mathbb{S}_3$, le caractère de la représentation triviale est associé à s_3, celui de la représentation standard est associé à s_{21} et enfin s_{111} correspond au caractère de la représentation signe. En général dans $\mathbb{S}_\mathrm{n}$, le caractère de la représentation triviale est toujours associé à s_n, la re-

présentation signe est toujours associée à $s_{11\ldots1}$ et la représentation standard est toujours associée à $s_{n-1,1}$.

2.3 Représentations polynomiales du groupe général linéaire $\mathrm{GL_r}$

Soit $\mathrm{GL_r}$ le groupe des matrices carrées inversibles de taille r à coefficients dans $\mathbb{C}$. Une représentation *polynomiale* de $\mathrm{GL_r}$ est un morphisme

$$\begin{array}{rccc} \varphi: & \mathrm{GL_r} & \to & \mathrm{GL}_N \\ & A & \mapsto & \varphi(A) \end{array}$$

tel que les entrées de la matrice $\varphi(A)$ sont des polynômes en les entrées de A. De plus, une représentation polynomiale est dite *homogène* de degré d si les entrées de $\varphi(A)$ sont des polynômes homogènes de degré d.

Exemple 2.4. *On donne pour exemple une représentation polynomiale de* GL_2 *homogène de degré 2. On définit le morphisme suivant*

$$\begin{array}{rccc} \varphi: & \mathrm{GL}_2 & \to & \mathrm{GL}_4 \\ A = & \begin{pmatrix} a & b \\ c & d \end{pmatrix} & \mapsto & \begin{pmatrix} aA & bA \\ cA & dA \end{pmatrix} = \varphi(A). \end{array}$$

On a donc

$$\varphi(A) = \begin{pmatrix} a^2 & ab & ba & b^2 \\ ac & ad & bc & bd \\ ca & cb & da & db \\ c^2 & cd & dc & d^2 \end{pmatrix}.$$

Chaque entrée de la matrice $\varphi(A)$ s'écrit bien comme un polynôme de degré 2 en les entrées de la matrice A.

Le morphisme qui associe à une matrice son déterminant est aussi un exemple de représentation polynomiale, cette fois de dimension 1. En effet, le déterminant est bien un polynôme en les entrées de la matrice et comme $\det(AB) = \det(A)\det(B)$,

il s'agit bien d'un morphisme de groupe.

Le caractère d'une représentation polynomiale φ de $\mathrm{GL_r}$ est, par définition, la fonction qui à toute matrice $A \in \mathrm{GL_r}$ associe la trace de la matrice $\varphi(A)$. Ainsi, dans l'exemple 2.4, le caractère de la représentation est $\mathrm{tr}(\varphi(A)) = a^2+ad+da+d^2$. Notons qu'il s'agit d'une fonction symétrique en a et d, $\mathrm{tr}(\varphi(A)) = s_2(a,d) + s_{11}(a,d) = s_1^2(a,d)$. En fait, on peut montrer que le caractère d'une représentation polynomiale de $\mathrm{GL_r}$ est toujours une fonction symétrique. Cette affirmation se base sur le fait que le caractère est le même pour tous les éléments d'une même classe de conjugaison. Dans le cas de $\mathrm{GL_r}$, les matrices semblables vont donc avoir le même caractère. De plus, il est facile de voir que la trace d'une matrice diagonalisable est un polynôme symétrique puisqu'il s'agit de la somme de ses valeurs propres. Enfin, on sait que l'ensemble des matrices diagonalisables est dense dans $\mathrm{GL_r}$ ce qui permet de conclure. Pour plus de détails voir (Green, 2007).

Tout comme dans le cas de $\mathbb{S}_\mathrm{n}$, on s'intéresse aux représentations et aux caractères irréductibles de $\mathrm{GL_r}$ et cette fois encore, on peut associer les caractères irréductibles aux fonctions de Schur. Dans (Prasad, 2015), on trouve les deux théorèmes suivants qui nous donnent exactement ces propriétés.

Théorème 2.2 (Classique)**.** *Les représentations polynomiales irréductibles de degré d de* $\mathrm{GL_r}$ *sont indicées par les partages de d ayant au plus r parts.*

Théorème 2.3 (Classique)**.** *Les caractères des représentations polynomiales irréductibles de degré d de* $\mathrm{GL_r}$ *correspondent aux fonctions de Schur s_λ, $\lambda \vdash d$ et $\ell(\lambda) \leq r$.*

Pour plus de détails sur les représentations de $\mathrm{GL_r}$, voir (James et Kerber, 1981), (Green, 2007) et (Bergeron, 2009).

2.4 Composantes isotypiques

Dans cette sous-section, on s'intéresse à la décomposition des représentations en composantes isotypiques. Une représentation ρ est *isotypique* si elle est une somme directe de sous-représentations irréductibles ρ_i toutes isomorphes. Autrement dit, soit $\mathcal{V}$ une représentation de G et S une représentation irréductible de G, la composante isotypique de $\mathcal{V}$ de type S, notée $\mathcal{V}^{[S]}$ est la somme de toutes les représentations irréductibles de $\mathcal{V}$ isomorphes à S. Ainsi, $\mathcal{V}^{[S]}$ est l'ensemble des éléments qui s'écrivent sous la forme $v_1+v_2+\cdots+v_l$ avec $v_i \in \mathcal{V}_i$ où $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_l$ est la liste de toutes les représentations irréductibles de $\mathcal{V}$ isomorphes à S.

Définition 2.6 (Classique)**.** *Soit ρ une représentation d'un groupe G et soit $\rho_1, \rho_2, \ldots \rho_k$ un ensemble complet de représentations irréductibles de G. Si $\rho = m_1\rho_1 \oplus m_2\rho_2 \oplus \cdots \oplus m_k\rho_k$ alors m_i est la multiplicité de ρ_i dans ρ et $m_i\rho_i$ est la* composante isotypique *de type ρ_i de ρ.*

Il est connu que la décomposition en composantes isotypiques est unique à ordre près dans la somme, voir par exemple (Sagan, 2001).

Théorème 2.4 (Classique)**.** *La décomposition de la représentation régulière en G en composantes isotypiques est $\bigoplus_{i=1}^{k} n_i\rho_i$ où ρ_i pour $1 \leq i \leq k$ sont les représentations irréductibles de G et $n_i = \dim \rho_i$.*

Dans notre cas, on souhaite trouver la décomposition en composantes isotypiques de certains $\mathbb{S}_n$-modules. En effet, si on connaît les composantes isotypiques et leurs dimensions, on connaît alors la multiplicité de la représentation irréductible associée dans le module étudié.

CHAPITRE III

$(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-CARACTÈRES D'ESPACES DE POLYNÔMES MULTIVARIÉS

On s'intéresse aux espaces de polynômes harmoniques diagonaux, et à des généralisations de ces espaces contenant des variables « inertes ». Dans ce chapitre, on définit les $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractères sur ces espaces et on en motive l'étude en exposant les résultats déjà connus et les liens entre la théorie des caractères des espaces de polynômes harmoniques diagonaux et d'autres domaines des mathématiques.

3.1 Actions simultanées de $\mathrm{GL_r}$ et $\mathbb{S}_\mathrm{n}$ sur R

L'anneau de polynômes $R = \mathbb{Q}[X]$, où X est la matrice contenant r jeux de n variables, est muni d'une action du groupe symétrique $\mathbb{S}_\mathrm{n}$ et du groupe général linéaire $\mathrm{GL_r}$.

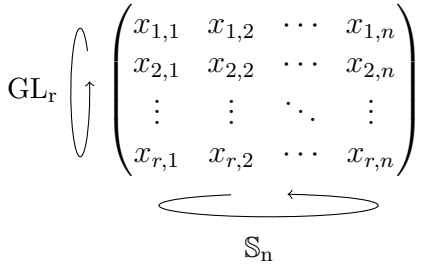


FIGURE 3.1 Actions simultanées de $\mathrm{GL_r}$ et $\mathbb{S}_\mathrm{n}$ sur X.

L'action de $\mathbb{S}_\mathrm{n}$ permute les colonnes de X, et $\mathrm{GL_r}$ agit par combinaison linéaire sur les lignes de X. Ces deux actions commutent. Une description plus détaillée

de l'action de $\mathbb{S}_n$ sur les polynômes a été donnée à la section 1.4. De la même façon, l'action de GL_r sur $f \in R$ est donnée par $f(X) \mapsto f(T.X)$ pour tout $T \in \mathrm{GL}_r$. On désigne par $R^{\mathbb{S}_n}$ l'anneau des *polynômes diagonalement symétriques*, c'est-à-dire les polynômes dans R qui sont invariants pour l'action sur les variables correspondant aux permutation des colonnes de X. L'*anneau coinvariant* $R_{\mathbb{S}_n}$ est le quotient de R par l'idéal $\mathcal{I}_n = \langle {R^{\mathbb{S}_n}}^+ \rangle$ des polynômes diagonalement symétriques sans terme constant. On rappelle que R est gradué par le degré (voir section 1.3). Ainsi, puisque $\mathcal{I}_n$ est un idéal homogène de R, $R_{\mathbb{S}_n}$ hérite de R une structure de $(\mathrm{GL}_r \times \mathbb{S}_n)$-module gradué par le degré. De plus, le multidegré est préservé par l'action de $\mathbb{S}_n$ (mais pas par l'action de GL_r). Le caractère de GL_r peut alors être déduit entièrement du degré de chaque composante multihomogène.

Pour $r = 1$, l'action de GL_1 est triviale, et $R_{\mathbb{S}_n}$ est de dimension $n!$. En fait, $R_{\mathbb{S}_n}$ correspond dans ce cas à la représentation régulière de $\mathbb{S}_n$. Son caractère bigradué est donné par un polynôme de Hall-Littlewood. Ces polynômes sont définis dans le chapitre 3 de (Macdonald, 1995). Pour $r = 2$, $R_{\mathbb{S}_n}$ est de dimension $(n+1)^{n-1}$ (16807 pour $n = 6$) ; comme conséquence de la démonstration de Haiman de la conjecture $n!$ (Haiman, 2002), le caractère bigradué correspond à $\nabla(e_n)$. On décrit plus en détails ces résultats dans la prochaine section.

On rappelle que les caractères irréductibles de $\mathbb{S}_n$ et de GL_r peuvent être exprimés en terme des fonctions de Schur (voir chapitre 2 pour plus de détails). Le caractère sera donc toujours donné sous cette forme. Ainsi le caractère associé à l'action simultanée de GL_r et de $\mathbb{S}_n$ sur un espace de polynômes en les variables de X est un couple de fonctions de Schur $(s_\mu(\boldsymbol{q}), s_\lambda(\boldsymbol{w}))$ où $s_\mu(\boldsymbol{q})$, avec $\boldsymbol{q} = (q_1, q_2, \ldots, q_r)$, encode l'action de GL_r et $s_\lambda(\boldsymbol{w})$, avec $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$, encode l'action de $\mathbb{S}_n$. On écrit ce couple sous la forme d'un produit tensoriel formel. On obtient alors une expression de la forme suivante pour le caractère de GL_r et de $\mathbb{S}_n$

$$\sum_{\lambda \vdash n} \sum_{\mu} c_{\lambda\mu} s_\mu(\boldsymbol{q}) \otimes s_\lambda(\boldsymbol{w}), \tag{3.1}$$

où $c_{\lambda\mu}$ sont des entiers positifs correspondant aux multiplicités des représentations irréductibles et sont les coefficients que l'on cherche à déterminer. Dans l'expression (3.1), le produit tensoriel permet d'identifier clairement quelle fonction de Schur encode l'action de $\mathrm{GL_r}$ et laquelle encode l'action de $\mathbb{S}_\mathrm{n}$. Pour simplifier l'écriture, on peut alors omettre les variables $\boldsymbol{q}$ et $\boldsymbol{w}$ et on écrit

$$\sum_{\lambda,\mu} c_{\lambda\mu} s_\mu \otimes s_\lambda. \tag{3.2}$$

Par exemple, le $(\mathrm{GL}_r \times \mathbb{S}_n)$-caractère de l'espace des polynômes harmoniques diagonaux en 2 jeux de 3 variables $\mathcal{H}_3^{(2)}$ est $1 \otimes s_3 + (s_1 + s_2) \otimes s_{21} + (s_{11} + s_3) \otimes s_{111}$. (Voir chapitre 4 pour les détails du calcul.) Ainsi, si les variables sont mentionnés dans l'expression du caractère, on suppose qu'il est alors sous la forme d'une somme de fonctions de Schur à coefficients dans $\mathbb{Q}[\boldsymbol{q}]$ où les polynômes en $\boldsymbol{q}$ représentent l'action de $\mathrm{GL_r}$. Si les variables sont omises, le caractère est exprimé sous forme de produit tensoriel. Dans les sections suivantes, on s'intéresse à des sous-espaces de R pour lesquels on veut une expression du caractère sous la même forme que l'équation (3.2).

3.2 Résultats connus sur les polynômes harmoniques diagonaux

Les caractères de certains sous-$(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-modules de R, incluant les polynômes harmoniques diagonaux $\mathcal{H}_n^{(r)}$, ont déjà été étudiés et des résultats pour $r \leq 3$ ont été établis. Dans cette section, on rappelle ces résultats. Dans le cas des polynômes harmoniques en un jeu de variables $\mathcal{H}_n$, on a le résultat suivant qui est une reformulation, adaptée à nos notations, d'un cas particulier.

Théorème 3.1. *(Shephard et Todd, 1954) Soit $\mathcal{H}_n(q;\boldsymbol{w})$ le $(\mathrm{GL}_1 \times \mathbb{S}_\mathrm{n})$-caractère de $\mathcal{H}_n$, alors $\mathcal{H}_n(q;\boldsymbol{w}) = \widetilde{H}_n(q;\boldsymbol{w})$.*

Dans le théorème 3.1, $\widetilde{H}_n(q;\boldsymbol{w})$ est le polynôme de Macdonald correspondant au partage à une part n. Il en résulte en particulier que la dimension de $\mathcal{H}_n$ est $n!$.

Dans le cas de deux jeux de variables, M. Haiman, dans (Haiman, 2003), a prouvé le théorème suivant sur les polynômes harmoniques diagonaux $\mathcal{H}_n^{(2)}$, en utilisant l'opérateur ∇ (voir équation 1.4) introduit par F. Bergeron et A. Garsia dans (Bergeron et Garsia, 1996) et des arguments de géométrie algébrique.

Théorème 3.2. *(Haiman, 2003) La dimension de $\mathcal{H}_n^{(2)}$ est $(n+1)^{n-1}$ et son $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractère est $\mathcal{H}_n^{(2)}(q,t;\boldsymbol{w}) = \nabla(e_n(\boldsymbol{w}))$.*

Dans le même article, M. Haiman prouve également des résultats sur des sous-espaces de $\mathcal{H}_n^{(2)}$ appelés les *modules de Garsia-Haiman* et notés $\mathcal{M}_\mu$. Il avait auparavant introduit ces modules en collaboration avec A. Garsia dans (Garsia et Haiman, 1993) dans le but de résoudre une conjecture posée par I. Macdonald sur la positivité des coefficients des polynômes qu'il introduit dans (Macdonald, 1995). Dans (Garsia et Haiman, 1993), A. Garsia et M. Haiman formulent la conjecture que la transformée de Frobenius des caractères irréductibles de $\mathbb{S}_\mathrm{n}$ sur $\mathcal{M}_\mu$ correspond aux polynômes définis par I. Macdonald et que la dimension de $\mathcal{M}_\mu$ est $n!$. Ces modules introduits par A. Garsia et M. Haiman ont des liens avec la géométrie des variétés de drapeau (Humphreys, 1975).

Les modules de Garsia-Haiman sont définis à partir d'une version généralisée du déterminant de Vandermonde sur deux jeux de variables. Soit $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ et $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ deux jeux de variables et μ un partage de n, alors

$$V_\mu(\boldsymbol{x};\boldsymbol{y}) := \det(x_i^a y_i^b)_{\substack{1\le i\le n \\ (a,b)\in\mu}}.$$

Cette définition suppose un ordre sur les cases de μ. On utilise l'ordre sur les cases d'un diagramme défini à la section 1.1. Cependant, l'ordre choisi importe peu puisqu'on on obtient le même déterminant au signe près et donc l'espace engendré reste le même. On note $\mathcal{M}_\mu$ le plus petit module contenant $V_\mu(\boldsymbol{x};\boldsymbol{y})$ et ses dérivées partielles en toutes les variables de $\boldsymbol{x}$ et de $\boldsymbol{y}$. On a alors le résultat suivant conjecturé dans (Garsia et Haiman, 1993) et prouvé dans (Haiman, 2003).

Théorème 3.3. *(Haiman, 2003) Le* $(\mathrm{GL}_2 \times \mathbb{S}_\mathrm{n})$*-caractère de* $\mathcal{M}_\mu$ *est* $\mathcal{M}_\mu(q,t;\boldsymbol{w}) = \widetilde{H}_\mu(q,t;\boldsymbol{w})$*. La dimension de* $\mathcal{M}_\mu$ *est* $n!$*.*

Le cas de trois jeux de variables a été étudié par F. Bergeron et L.-F. Préville-Ratelle dans (Bergeron et Préville-Ratelle, 2012). Dans cet article, les auteurs mettent en évidence des liens entre le caractère gradué de $\mathcal{H}_n^{(3)}$, les treillis de Tamari généralisés et les fonctions de parking sur les chemins de Dyck dans le but de donner une description combinatoire du caractère. Ils conjecturent une formule pour le caractère $\mathcal{H}_n^{(3)}(q,t,1;\boldsymbol{w})$ et M. Haiman conjecture que la dimension de $\mathcal{H}_n^{(3)}$ est $2^n(n+1)^{n-2}$. Cependant, dans le cas d'un nombre quelconque r de jeux de variables, il ne semble pas y avoir de « belle » formule donnant la dimension de $\mathcal{H}_n^{(r)}$.

3.3 Motivations : Algèbre de Hall elliptique

Dans les dernières années, il a été mis en évidence qu'il existe de nombreux liens entre la théorie des polynômes de Macdonald, les représentations des polynômes harmoniques diagonaux et la combinatoire des fonctions de parking d'une part et l'algèbre de Hall elliptique de Shiffmann-Vasserot (Schiffmann et Vasserot, 2011), les algèbres de Hecke doublement affines, la géométrie algébrique de fibres de Hikita (Hikita, 2014), ou encore le polynôme de HOMFLY en théorie des nœuds (Freyd *et al.*, 1985) d'autre part. En combinant ces différents travaux avec leurs

propres découvertes (Gorsky et Neguţ, 2015), E. Gorsky et A. Neguţ ont formulé la conjecture qu'on pourrait appeler « (m,n)-shuffle », avec $\text{pgcd}(m,n)=1$. Le travail de F. Bergeron, A. Garsia, E. Leven et G. Xin, dans (Bergeron *et al.*, 2016) correspond à formuler une extension à la conjecture de E. Gorsky et A. Neguţ, dite conjecture « compositional (km,kn)-shuffle ». On présente dans cette section une partie de leur travail.

Soit $f(\boldsymbol{x})$ une fonction symétrique, on donne l'opérateur suivant

$$D_k(f(\boldsymbol{x})) := f[\boldsymbol{x}+M/z]\sum_{i\geq 0}(-z)^i e_i[\boldsymbol{x}]\big|_{z^k} \tag{3.3}$$

avec $M=(1-q)(1-t)$ et le symbole $\big|_{z^k}$ signifie que l'on garde uniquement le coefficient devant z^k dans l'expression. Les opérateurs $\{D_k\}_{k\geq 0}$ ont été introduits pour l'étude des polynômes de Macdonald (Bergeron *et al.*, 1999). Ils sont à l'origine de l'algèbre de Hall elliptique de Shiffmann et Vasserot (Schiffmann et Vasserot, 2011).

Proposition 3.1 (Classique)**.** *Si $m,n>1$ sont copremiers, alors il existe une unique décomposition $(m,n)=(a,b)+(c,d)$ telle que $\det\begin{pmatrix}a & b\\ c & d\end{pmatrix}=1$ avec $a,b,c,d\in\mathbb{N}$.*

On note alors $\text{Split}(m,n):=(a,b)+(c,d)$ avec $\text{Split}(1,n):=(1,n-1)+(0,1)$ et $\text{Split}(m,1):=(1,0)+(m-1,1)$.

Exemple 3.1. $\text{Split}(4,3)=(3,2)+(1,1)$ *et* $\text{Split}(3,2)=(2,1)+(1,1)$

Définition 3.1. *(Bergeron* et al.*, 2016) Pour tous m,n copremiers, on définit l'opérateur suivant sur les fonctions symétriques*

$$Q_{m,n}:=\begin{cases}\frac{1}{M}[Q_{c,d},Q_{a,b}] & \text{si } m>1 \text{ et } \text{Split}(m,n)=(a,b)+(c,d)\\ D_n & \text{si } m=1\end{cases}$$

où $[A, B] = AB - BA$ *et* D_n *est l'opérateur défini en* (3.3).

Exemple 3.2. *Calculons* $Q_{4,3}$.
À l'exemple 3.1, on a vu que $\mathrm{Split}(4,3) = (3,2)+(1,1)$, *donc* $Q_{4,3} = \frac{1}{M}[Q_{1,1}, Q_{3,2}]$. *De plus,* $\mathrm{Split}(3,2) = (2,1) + (1,1)$ *donc* $Q_{3,2} = \frac{1}{M}[Q_{1,1}, Q_{2,1}]$ *et* $Q_{1,1} = D_1$. *Enfin,* $\mathrm{Split}(2,1) = (1,1) + (1,0)$ *donc* $Q_{2,1} = \frac{1}{M}[Q_{1,0}, Q_{1,1}] = \frac{1}{M}[D_0, D_1]$. *Donc finalement* $Q_{4,3} = \frac{1}{M^3}[D_1, [D_1, [D_0, D_1]]] = \frac{1}{M^3}(D_1^3 D_0 - 3D_1^2 D_0 D_1 + 3D_1 D_0 D_1^2 - D_0 D_1^3)$.

Notons que si m et n ne sont pas premiers entre eux, il existe plusieurs solutions qui satisfont les conditions de la proposition 3.1, dans ce cas, on considère la solution pour laquelle $b - an/m$ est minimal et alors les opérateurs $Q_{m,n}$ sont bien définis. Les auteurs ont également montré que $\{\prod_i Q_{0,\lambda_i}\}_\lambda$ forme une base de l'anneau des fonctions symétriques Λ. Enfin, on rappelle l'algorithme donné dans (Bergeron *et al.*, 2016) pour construire l'opérateur qui nous intéresse.

Algorithme. *(Bergeron* et al.*, 2016) Soit* f *une fonction symétrique homogène de degré* k *et* m *et* n *des entiers premiers entre eux.*

1. *On exprime* f *dans la base* $\{\prod_i Q_{0,\lambda_i}\}_\lambda$ *:* $f = \sum_{\lambda \vdash k} c_\lambda(q,t) \prod_{i=1}^{l(\lambda)} Q_{0,\lambda_i}$

2. *À l'aide des coefficients* $c_\lambda(q,t)$ *apparaissant dans le développement de* f*, on définit*

$$\boldsymbol{f}_{km,kn} := \sum_{\lambda \vdash k} c_\lambda(q,t) \prod_{i=1}^{\ell(\lambda)} Q_{m\lambda_i, n\lambda_i}. \tag{3.4}$$

Définition 3.2. *On définit l'opérateur* $\boldsymbol{e}_{km,kn}$ *en fixant* $f = e_k$ *dans (3.4).*

Par la suite, on note simplement cet opérateur $\boldsymbol{e}_{m,n}$ où m et n ne seront pas nécessairement premiers entre eux. Dans ce cas, la valeur de k correspondante peut être retrouvée en calculant le pgcd de m et n. Le travail mené dans la suite

est motivé par le souhait de trouver des modules dont le $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractère est $\boldsymbol{e}_{m,n}$.

3.4 Polynômes harmoniques diagonaux à variables inertes

On introduit maintenant des espaces de polynômes multivariés contenant des variables « inertes ». On ajoute à la matrice X une ligne contenant un jeu de variables particulières dites *inertes*. Plus précisément, soit X' la matrice contenant r jeux de n variables et ce jeu de variables inertes, $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$,

$$X' := \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{r1} & x_{r2} & \cdots & x_{rn} \\ \theta_1 & \theta_2 & \cdots & \theta_n \end{pmatrix}.$$

Le groupe $\mathrm{GL_r}$ agit toujours par multiplication à gauche sur les r premiers jeux de variables et fixe le jeu de variables inertes. Ainsi $\mathrm{GL_r}$ apparaît comme un sous-groupe de GL_{r+1} fixant la dernière ligne de X'. Cependant, le groupe symétrique $\mathbb{S}_\mathrm{n}$ agit toujours par multiplication à droite, en permutant les colonnes, ce qui inclut les variables inertes.

$$\mathrm{GL_r} \circlearrowleft \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{r,1} & x_{r,2} & \cdots & x_{r,n} \\ \hline \theta_1 & \theta_2 & \cdots & \theta_n \end{pmatrix} \circlearrowleft \mathbb{S}_\mathrm{n}$$

FIGURE 3.2 Actions simultanées $\mathrm{GL_r}$ et $\mathbb{S}_\mathrm{n}$ sur les variables classiques et inertes.

On dit que les θ_i sont inertes, car on considère que leur degré est 0, elles ne sont donc pas prises en compte dans le calcul du degré des polynômes et n'influencent pas le caractère de $\mathrm{GL_r}$. Ainsi, l'anneau de polynômes $R' = \mathbb{Q}[X']$ possède une structure de $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-module gradué par le degré et, encore une fois, le caractère de $\mathrm{GL_r}$ est donné par le degré des composantes multihomogènes. Toutes les remarques faites à la section 3.1 sont également valables dans le cas de R'.

L'avantage d'ajouter les variables inertes est que l'on va obtenir dans le $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractère des termes qui n'apparaissent pas si on utilise uniquement des variables classiques. Les variables inertes, puisqu'elles sont de degré 0, n'apparaissent pas dans les opérateurs de dérivations ou de polarisation. Or les espaces que l'on souhaite étudier sont construits à partir d'un générateur, exprimé en terme de variables classiques et de variables inertes, et clos par dérivation et polarisation. Ainsi, ces variables inertes vont apparaître dans tous les polynômes obtenus par dérivation ou polarisation de ce générateur. Pour certains multidegrés, les composantes multihomogènes correspondantes vont contenir des polynômes non nuls, ce qui ne serait pas le cas avec uniquement les variables classiques qui « disparaissent » progressivement au cours des dérivations et polarisations successives. Comme ces variables inertes portent de l'information sur l'action de $\mathbb{S}_\mathrm{n}$, le caractère associé contient alors des termes qui ne sont pas présents si on utilise uniquement des variables classiques.

Définition 3.3. *Soit γ un diagramme tel que $|\gamma| = n$. On définit alors le déterminant associé à γ de la façon suivante*

$$V_\gamma := \det(x_{1,i}^a \theta_i^b)_{\substack{1 \leq i \leq n \\ (a,b) \in \gamma}}.$$

Bien qu'une définition précise de V_γ dépende de l'ordre choisi sur les cases de γ, cela ne change rien pour ce qui est du sous-espace obtenu. Par exemple, on

peut choisir d'utiliser l'ordre défini à la section 1.1. Bien entendu, lorsque γ est le partage réduit à une part de taille n, alors V_γ coïncide avec le déterminant de Vandermonde classique V_n.

Exemple 3.3. *Dans le cas du diagramme* $\gamma =$  , *on obtient le déterminant*

$$V_\gamma = \begin{vmatrix} 1 & x_1 & \theta_1 \\ 1 & x_2 & \theta_2 \\ 1 & x_3 & \theta_3 \end{vmatrix} = x_1\theta_2 - x_2\theta_1 - x_1\theta_3 + x_3\theta_1 + x_2\theta_3 - x_3\theta_2.$$

Notons que pour alléger l'écriture nous avons nommé les variables x_i *au lieu de* $x_{1,i}$ *dans cet exemple.*

Définition 3.4. *On définit* $\mathcal{E}_\gamma^{(r)}$ *comme le plus petit sous-espace de* R' *contenant* V_γ *et clos par dérivation et polarisation.*

Notre but est de calculer le caractère de $\mathcal{E}_\gamma^{(r)}$ pour l'action simultanée de $\mathrm{GL_r}$ et de $\mathbb{S}_\mathrm{n}$. On ne connaît pas de formule explicite pour ces caractères dans le cas d'un nombre quelconque r de jeux de variables et leur calcul nécessite potentiellement beaucoup d'espace mémoire et un temps très important. Par exemple, la dimension de $\mathcal{E}_6^{(5)}$ est 5.8×10^5 et fait intervenir des polynômes jusqu'au degré 15 en 30 variables contenant plusieurs milliers de termes.

Les espaces $\mathcal{E}_\gamma^{(r)}$ associés à des diagrammes γ particuliers liés à la combinatoire de Catalan rectangulaire, nous intéressent principalement. Il s'agit des diagrammes $\gamma(m,n)$ issus de chemins de Dyck définis à la section 1.2. Dans un premier temps, on s'intéresse uniquement aux diagrammes $\gamma(m,n)$ qui correspondent à des partages et on souhaite calculer $\mathcal{E}_{\gamma(m,n)}$. Les autres cas de diagrammes, à savoir les diagrammes obtenus de chemins de Dyck qui ne correspondent pas à des partages, les diagrammes correspondant à des vecteurs d'entiers et les diagrammes quelconques, sont considérés au chapitre 5.

Notons que dans le cas où $\gamma = \mu$ est un partage, le caractère des modules de Garsia-Haiman $\mathcal{M}_\mu$ nous donne directement de l'information sur le caractère de $\mathcal{E}_\mu^{(2)}$. En effet, si on garde uniquement la partie du caractère de $\mathcal{M}_\mu(q,t;\boldsymbol{w})$ pour laquelle t est de degré maximal, cela correspond aux polynômes pour lesquels le degré en $\boldsymbol{y}$ est maximal, c'est-à-dire les polynômes obtenus de $V_\mu(\boldsymbol{x};\boldsymbol{y})$ sans dériver ou polariser par rapport à $\boldsymbol{y}$. Les variables $\boldsymbol{y}$ jouent alors le même rôle que les variables inertes dans ces polynômes et on obtient ainsi le caractère de $\mathcal{E}_\mu^{(1)}$.

Le but de plusieurs recherches menées récemment par F. Bergeron (Bergeron, 2013) est de trouver des $(\mathrm{GL}_2 \times \mathbb{S}_\mathrm{n})$-modules dont le caractère correspond à $\boldsymbol{e}_{m,n}$.

Conjecture 1. *[F. Bergeron, non publié] Si $\gamma(m,n)$ est un partage alors*

$$\mathcal{E}_{\gamma(m,n)}(q,t;\boldsymbol{w}) = \boldsymbol{e}_{m,n}(q,t;\boldsymbol{w}).$$

De plus, en combinant les résultats de (Gorsky et Neguţ, 2015), (Bergeron *et al.*, 2016) et (Bergeron, 2017), on a que $\nabla(\boldsymbol{e}_{m,n}(q,t;\boldsymbol{w})) = \boldsymbol{e}_{m+n,n}(q,t;\boldsymbol{w})$. Enfin, comme $\boldsymbol{e}_{n,n} = \nabla(e_n)$, on en déduit que $\boldsymbol{e}_{jn,n} = \nabla^j(e_n)$. On peut alors formuler un corollaire à la conjecture précédente qui est que $\mathcal{E}_{\gamma(jn,n)}(q,t;\boldsymbol{w}) = \nabla^j(e_n)(q,t;\boldsymbol{w})$.

CHAPITRE IV

STRATÉGIES DE CALCUL

L'objectif principal du travail mené dans cette thèse est de développer une stratégie de calcul du $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractère des espaces décrits au chapitre 3. Comme nous l'avons vu précédemment (section 3.2), la dimension de ces espaces augmente rapidement en fonction de m et n. Cela rend le calcul du caractère difficile et, même à l'ordinateur, on atteint rapidement les limites de temps et d'espace machine. On a estimé que le temps de calcul pour le caractère de $\mathcal{E}_6^{(5)}$ en utilisant uniquement de l'algèbre linéaire serait d'environ 1600 ans. Il est donc indispensable de mettre en place des stratégies de calcul basées sur une réflexion mathématique permettant de gagner en efficacité. Dans ce chapitre, on présente les optimisations et stratégies utilisées pour parvenir à faire des calculs pour des valeurs de m et n allant jusqu'à 7 et qui constituent la partie centrale de cette thèse.

4.1 Stabilisation du caractère sur le nombre de jeux de variables

On rappelle que l'on souhaite calculer le $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractère de $\mathcal{E}_\gamma^{(r)}$, l'espace engendré par le déterminant V_γ et clos par dérivation et par polarisation sur r jeux de variables (voir section 3.4). Dans la formule du caractère, les termes obtenus pour l'action de $\mathrm{GL_r}$ dépendent du nombre r de jeux de variables utilisés. En effet, d'après le théorème 2.3, avec r jeux de variables, on obtient dans le

$\mathrm{GL_r}$-caractère des fonctions de Schur indicées par des partages ayant au plus r parts. Cependant, lorsque r est suffisamment grand (voir théorème ci-dessous), la formule du caractère se stabilise et contient alors toute l'information concernant les représentations irréductibles de $\mathrm{GL_r}$. Ce résultat a été formulé par F. Bergeron.

Théorème 4.1. *(Bergeron, 2013) La formule du caractère de $\mathcal{E}_n^{(r)}$ se stabilise pour $r \geq n$. Autrement dit, pour tout $r \geq n$, $\mathcal{E}_n^{(r)} = \mathcal{E}_n^{(n)}$.*

Idée de la démonstration. Soit $R = \mathbb{Q}[X]$ l'anneau des polynômes en r jeux de n variables où X est la matrice contenant r lignes de n variables. L'espace $\mathcal{E}_n^{(r)}$ est un sous-module de R, sa décomposition en irréductibles est donc contenue dans celle de R. Or le bicaractère χ de R est donné par la formule pléthystique

$$\chi = h_n\left[\left(\sum_{j\geq 0} h_j(\mathbf{q})\right)\boldsymbol{z}\right] \qquad \text{avec} \qquad \sum_{j\geq 0} h_j(\mathbf{q}) = \prod_{j=0}^{r} \frac{1}{1-q_i},$$

où $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$ sont les variables formelles intervenant dans les fonctions de Schur qui encodent le caractère de $\mathbb{S}_\mathrm{n}$, et $\mathbf{q} = (q_1, q_2, \ldots, q_r)$ sont les variables formelles intervenant dans les fonctions de Schur qui encodent le caractère de $\mathrm{GL_r}$. On rappelle que q_i encode le degré en les variables de la ligne i. Avec la notation sous forme de produit tensoriel, on obtient la formule suivante écrite en terme de fonctions de Schur $\chi = h_n\left[\sum_{j\geq 0} h_j \otimes s_1\right]$. On peut montrer que le résultat du pléthysme écrit dans la base des fonctions de Schur est de la forme $\sum_{\mu\vdash n}\sum_{\lambda} a_{\lambda\mu} s_\lambda \otimes s_\mu$, avec $\ell(\lambda) \leq n$ essentiellement parce que le développement dans la base des fonctions de Schur du pléthysme $h_a[h_b]$ ne fait intervenir que des fonctions de Schur indicées par des partages ayant au plus a parts. □

Le théorème 4.1 donne une indication précieuse pour le calcul du caractère. En effet, la cardinalité des espaces étudiés et donc le temps de calcul du caractère

dépendent du nombre de jeux de variables utilisés. En connaissant la borne pour laquelle la formule du caractère se stabilise, cela nous évite d'effectuer des calculs avec un nombre inutilement trop grand de jeux de variables. Ainsi, on a besoin d'utiliser uniquement $(n-1)$ jeux de variables dans le calcul du caractère pour obtenir toute l'information. On se réduit ainsi à un calcul fini, en termes de nombre de jeux de variables. Bien entendu, cela reste vrai pour $R' = \mathbb{Q}[X']$ puisque $\mathrm{GL_r}$ n'agit pas sur les variables inertes. Le théorème 4.1 est donc valable pour $\mathcal{E}_\gamma^{(r)}$, pour tout γ. De plus, puisque le caractère de $\mathcal{E}_\gamma^{(r)}$ se stabilise pour r assez grand, il n'est pas toujours nécessaire de préciser sur combien de jeux de variables on effectue le calcul. Ainsi, si la précision n'est pas nécessaire, on omet l'exposant r et on note simplement $\mathcal{E}_\gamma$. Il est alors implicite que $\mathcal{E}_\gamma$ est la forme stable du caractère associé au diagramme γ exprimé à l'aide de produits tensoriels.

4.2 Exploitation de la quasi-commutativité des opérateurs

Dans cette section, on s'intéresse à la quasi-commutativité des opérateurs introduits au chapitre 1. On souhaite étudier l'impact sur la construction de l'espace $\mathcal{E}_\gamma$ de l'ordre dans lequel on applique ces opérateurs. Bien évidemment, si on applique à un polynôme f des opérateurs agissant sur des jeux de variables différents, peu importe l'ordre dans lequel on les applique, cela donne le même résultat puisqu'il s'agit de dérivées partielles. Ainsi, les compositions d'opérateurs qu'il est intéressant de regarder sont celles contenant des opérateurs agissant sur les mêmes jeux de variables. On calcule alors le commutateur $[A, B] = AB - BA$ dans chacun des cas. Pour rappel, la dérivée k-ième d'un produit est donnée par la formule suivante

$$(fg)^{(k)} = \sum_{i=0}^{n} \binom{k}{i} f^{(k-i)} g^{(i)},$$

et donc on a que $\partial_x^k(xf) = k\partial_x^{k-1} f + x\partial_x^k f$.

Premièrement, on s'intéresse à la composition de deux opérateurs de polarisation ayant des variables en commun. Soit f un polynôme multivarié en plusieurs jeux de variables, $f \in R'$, on a alors,

$$P^k_{\boldsymbol{x},\boldsymbol{y}} P^l_{\boldsymbol{y},\boldsymbol{z}} f - P^l_{\boldsymbol{y},\boldsymbol{z}} P^k_{\boldsymbol{x},\boldsymbol{y}} f = \sum_{i=1}^{n} y_i \partial^k_{x_i} \sum_{j=1}^{n} z_j \partial^l_{y_j} f - \sum_{j=1}^{n} z_j \partial^l_{y_j} \sum_{i=1}^{n} y_i \partial^k_{x_i} f. \tag{4.1}$$

Lorsque $i \neq j$, les dérivées commutent et les termes s'annulent. Le seul cas à regarder est donc $i = j$. Afin de simplifier l'écriture, on omet temporairement les indices i et j. On obtient alors $y\partial^k_x z\partial^l_y f - z\partial^l_y y\partial^k_x f = yz\partial^k_x \partial^l_y f - z\left(l\partial^{l-1}_y \partial^k_x f + y\partial^l_y \partial^k_x f\right) = -lz\partial^{l-1}_y \partial^k_x f$. En utilisant ce résultat dans l'équation 4.1, on obtient

$$P^k_{\boldsymbol{x},\boldsymbol{y}} P^l_{\boldsymbol{y},\boldsymbol{z}} f - P^l_{\boldsymbol{y},\boldsymbol{z}} P^k_{\boldsymbol{x},\boldsymbol{y}} f = \sum_{i=1}^{n} -l z_i \partial^{l-1}_{y_i} \partial^k_{x_i} f. \tag{4.2}$$

On en conclut donc qu'il est nécessaire d'appliquer ces deux compositions de polarisateurs lors de la construction de $\mathcal{E}_\gamma$ pour que l'espace soit bel et bien clos par polarisation.

Regardons maintenant ce qu'il se passe pour un opérateur de polarisation et une dérivée partielle ayant des variables communes.

$$\partial^l_{y_i} P^k_{\boldsymbol{x},\boldsymbol{y}} f - P^k_{\boldsymbol{x},\boldsymbol{y}} \partial^l_{y_i} f = \partial^l_{y_i} \sum_{j=1}^{n} y_j \partial^k_{x_j} f - \sum_{j=1}^{n} y_j \partial^k_{x_j} \partial^l_{y_i} f \tag{4.3}$$

Pour $i = j$, on omet temporairement les indices et on obtient, $\partial^l_y y\partial^k_x f - y\partial^k_x \partial^l_y = l\partial^{l-1}_y \partial^k_x f + y\partial^l_y \partial^k_x f - y\partial^k_x \partial^l_y f = l\partial^{l-1}_y \partial^k_x f$. En utilisant ce résultat dans l'équation 4.3, on a

$$\partial^l_{y_i} P^k_{\boldsymbol{x},\boldsymbol{y}} f - P^k_{\boldsymbol{x},\boldsymbol{y}} \partial^l_{y_i} f = l \sum_{i=1}^{n} \partial^{l-1}_{y_i} \partial^k_{x_i} f. \tag{4.4}$$

Ainsi, on obtient une somme de compositions de dérivées partielles. Puisque les

espaces étudiés sont clos par dérivation et polarisation, la commutation du polarisateur et de la dérivée partielle donne des opérateurs déjà présents dans la construction de l'espace, cela ne créera donc pas de nouveaux éléments linéairement indépendants. Ce phénomène s'explique par la commutativité de l'action de $\mathrm{GL_r}$, représentée par les opérateurs de polarisation, et de l'action de $\mathbb{S}_\mathrm{n}$ sur V_γ et ses dérivées partielles. On revient sur cette observation dans les prochaines sections. De plus, on remarque que l'opérateur obtenu par commutation des deux précédents reste symétrique. Nous revenons également sur ce point plus loin.

Soit $\mathcal{W}$ un sous-espace de R'. On note $\mathcal{D}(\mathcal{W})$ la fermeture de $\mathcal{W}$ par dérivation, $\mathcal{P}(\mathcal{W})$ la fermeture de $\mathcal{W}$ par polarisation et enfin $\mathcal{P}_\mathcal{D}(\mathcal{W})$ la fermeture de $\mathcal{W}$ par dérivation et polarisation. Des remarques précédentes, on peut alors formuler la proposition suivante.

Proposition 4.1. *Pour tout $\mathcal{W}$, sous-espace de R',*

$$\mathcal{P}_\mathcal{D}(\mathcal{W}) = \mathcal{D} \circ \mathcal{P}(\mathcal{W}) = \mathcal{P} \circ \mathcal{D}(\mathcal{W}).$$

Dans notre cas, $\mathcal{W} = \mathbb{Q}\{V_\gamma\}$ et on veut calculer une base de l'espace $\mathcal{E}_\gamma^{(r)} = \mathcal{P}_\mathcal{D}(\mathbb{Q}\{V_\gamma\})$. La proposition 4.1 implique que l'on peut tout d'abord calculer une base $\mathcal{B}_\gamma^{(1)}$ de l'espace $\mathcal{E}_\gamma^{(1)} = \mathcal{D}(\mathbb{Q}\{V_\gamma\})$. Puis dans un deuxième temps, on cherche à obtenir une base de $\mathcal{E}_\gamma^{(r)}$ à partir de $\mathcal{P}(\mathcal{B}_\gamma^{(1)})$. En effectuant les calculs de cette façon, on utilise uniquement les dérivées partielles en le premier jeu de variables et on réduit le nombre de compositions d'opérateurs à effectuer.

4.3 Utilisation du symétriseur de Young

Dans la section précédente, on a obtenu une façon plus efficace de trouver une base $\mathcal{B}_\gamma$ de $\mathcal{E}_\gamma$, mais rappelons que notre but général est de calculer le $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-

caractère de cet espace. Pour le $\mathbb{S}_n$-caractère plus précisément, notre stratégie est de calculer, pour chaque composante isotypique $\mathcal{I}_\lambda^{(r)}$ de $\mathcal{E}_\gamma^{(r)}$, une base multihomogène $\mathcal{B}_\lambda^{(r)}$ de $\mathcal{I}_\lambda^{(r)}$. Cela correspond à décomposer les modules considérés de la façon suivante. Soit $\mathcal{G}$ un sous-module gradué de R' clos par dérivation, $\mathcal{G} = \mathcal{D}(\mathcal{W})$. Et, soit $\mathcal{F}$ le sous-module de R' obtenu de $\mathcal{G}$ par polarisation sur r jeux de variables, $\mathcal{F} = \mathcal{P}(\mathcal{G})$ où $\mathcal{P}$ est la fermeture par polarisation pour les r jeux de variables classiques de X'. On décompose $\mathcal{F}$ en représentations irréductibles pour l'action de $\mathbb{S}_n$ en utilisant l'idempotent de Young y_λ. On note $\mathcal{F}_\lambda$ l'image de $\mathcal{F}$ par l'idempotent de Young, $\mathcal{F} \xrightarrow{\pi_\lambda} \!\!\!\!\rightarrow \mathcal{F}_\lambda$ où $\pi_\lambda(f) = f.y_\lambda$ pour tout $f \in \mathcal{F}$. Le projecteur π_λ projette chaque composante irréductible sur un espace de dimension 1 (voir théorème 2.1 et lemme 2.2 (iii)). On obtient alors pour chaque partage λ un sous-espace, dit de type λ, de dimension égale à la multiplicité de la composante irréductible associée.

Proposition 4.2. *Pour tout $\lambda \vdash n$ le sous-espace de type λ est stable par polarisation, autrement dit $\pi_\lambda \circ \mathcal{P}(\mathcal{G}) = \mathcal{P} \circ \pi_\lambda(\mathcal{G})$.*

On peut également reformuler la proposition 4.2 par $\mathcal{F}_\lambda = \mathcal{P}(\mathcal{G}_\lambda)$ où $\mathcal{G}_\lambda$ le sous-espace de type λ de $\mathcal{G}$, $\mathcal{G}_\lambda = \pi_\lambda(\mathcal{G})$.

Démonstration. Soit $g \in \mathcal{G}_\lambda$, on peut alors montrer que $P^k_{\boldsymbol{u},\boldsymbol{v}} g \in \mathcal{F}_\lambda$ pour tous jeux de variables $\boldsymbol{u}$ et $\boldsymbol{v}$. En effet, on a que

$$\begin{aligned}
(P^k_{\boldsymbol{u},\boldsymbol{v}} g).y_\lambda &= \sum_{i=1}^{n} v_i \partial u_i^k g. \sum_{\sigma \in P_\lambda} \sigma \sum_{\tau \in Q_\lambda} \operatorname{sgn}(\tau)\tau \\
&= \sum_{\substack{\sigma \in P_\lambda \\ \tau \in Q_\lambda}} \operatorname{sgn}(\tau) \sum_{i=1}^{n} v_{\tau\sigma(i)} \partial u^k_{\tau\sigma(i)} \tau\sigma(g) \\
&= \sum_{\substack{\sigma \in P_\lambda \\ \tau \in Q_\lambda}} \operatorname{sgn}(\tau) \sum_{i=1}^{n} v_i \partial u_i^k \tau\sigma(g) \\
&= \sum_{i=1}^{n} v_i \partial u_i^k (g.y_\lambda) = P^k_{\boldsymbol{u},\boldsymbol{v}}(g.y_\lambda).
\end{aligned}$$

Comme $g \in \mathcal{G}_\lambda$ et que π_λ est idempotent à un scalaire près (théorème 2.1 et proposition 2.1), on a $g.y_\lambda = n_\lambda g$. Donc $(P^k_{\boldsymbol{u},\boldsymbol{v}}g).y_\lambda = P^k_{\boldsymbol{u},\boldsymbol{v}}(g.y_\lambda) = P^k_{\boldsymbol{u},\boldsymbol{v}}(n_\lambda g) = n_\lambda(P^k_{\boldsymbol{u},\boldsymbol{v}}g)$. Ainsi, $P^k_{\boldsymbol{u},\boldsymbol{v}}g \in \mathcal{F}_\lambda$ et $\mathcal{F}_\lambda = \mathcal{P}(\mathcal{G}_\lambda)$. □

Dans notre cas, $\mathcal{W} = \mathbb{Q}\{V_\gamma\}$, $\mathcal{G} = \mathcal{D}(\mathcal{W}) = \mathcal{E}^{(1)}_\gamma$ et $\mathcal{F} = \mathcal{P}(\mathcal{G}) = \mathcal{E}^{(r)}_\gamma$. D'après les propositions 4.1 et 4.2, on a donc

$$\mathcal{E}^{(r)}_\gamma = \mathcal{P}\left(\bigoplus_{\lambda\vdash n} \pi_\lambda(\mathcal{E}^{(1)}_\gamma)\right) = \bigoplus_{\lambda\vdash n} \mathcal{P}\circ\pi_\lambda(\mathcal{E}^{(1)}_\gamma) \tag{4.5}$$

Exemple 4.1. *On calcule, par exemple, pour tout $\lambda \vdash 3$ les sous-espaces de type λ de $\mathcal{E}_{21}$. On rappelle que $\mathcal{E}^{(1)}_{21}$ est engendré par le déterminant $V_{21} = -x_2\theta_1 + x_3\theta_1 + x_1\theta_2 - x_3\theta_2 - x_1\theta_3 + x_2\theta_3$ et clos par dérivation. On va tout d'abord projeter $\mathcal{E}^{(1)}_{21}$ sur ses sous-espaces de type λ, $\lambda \vdash 3$:*

$$\begin{aligned}
\pi_{111}(\mathcal{E}^{(1)}_{21}) &= \mathbb{Q}\{x_2\theta_1 - x_3\theta_1 - x_1\theta_2 + x_3\theta_2 + x_1\theta_3 - x_2\theta_3\},\\
\pi_{21}(\mathcal{E}^{(1)}_{21}) &= \mathbb{Q}\{-\theta_1 + \theta_3\},\\
\pi_3(\mathcal{E}^{(1)}_{21}) &= \{0\}.
\end{aligned}$$

Ensuite, on polarise chaque sous-espace $\pi_\lambda(\mathcal{E}^{(1)}_{21})$. Le sous-espace $\pi_{21}(\mathcal{E}^{(1)}_{21})$ est homogène de degré 0, donc la polarisation n'a pas d'effet sur ce sous-espace. La clôture par polarisation de $\pi_{111}(\mathcal{E}^{(1)}_{21})$, décomposée en composantes homogènes pour le degré, est

$$\begin{aligned}
\pi_{111}(\mathcal{E}_{21}) &= \mathbb{Q}\{x_2\theta_1 - x_3\theta_1 - x_1\theta_2 + x_3\theta_2 + x_1\theta_3 - x_2\theta_3\}\\
&\quad \oplus \mathbb{Q}\{-y_2\theta_1 + y_3\theta_1 + y_1\theta_2 - y_3\theta_2 - y_1\theta_3 + y_2\theta_3\}.
\end{aligned}$$

Dans l'exemple 4.1 ci-dessus, le sous-espace de type 111 se décompose en composantes multihomogènes pour le degré. Dans la section suivante, on montre que cela est vrai pour tous les sous-espaces $\pi_\lambda(\mathcal{E}^{(r)}_\gamma)$.

4.4 Exploitation de la graduation par le degré

Grâce au projecteur de Young, on obtient une base multihomogène $\mathcal{B}_\lambda^{(r)}$ de chaque sous-espace de type λ, $\mathcal{I}_\lambda^{(r)}$, de $\mathcal{E}_\gamma^{(r)}$. De plus, la projection π_λ préserve le degré. En effet, $\deg(f.y_\lambda) = \deg(f)$ pour tout $f \in R'$ puisque $y_\lambda \in \mathbb{Q}\mathbb{S}_\mathrm{n}$. En outre, on a montré (proposition 4.2) que la projection par π_λ et la clôture par polarisation commutent. Donc comme $\mathcal{E}_\gamma^{(r)}$ est gradué par le degré, les sous-espace $\pi_\lambda(\mathcal{E}_\gamma^{(r)})$ le sont aussi. On note $\pi_{\boldsymbol{d}} : \mathcal{V} \to \mathcal{V}$ la projection sur la composante multihomogène de degré $\boldsymbol{d}$. D'après les remarques précédentes et l'équation 4.5, on peut en déduire une décomposition de $\mathcal{E}_\gamma^{(r)}$.

Théorème 4.2. *L'espace $\mathcal{E}_\gamma$ engendré par V_γ et clos par dérivation et polarisation se décompose de la façon suivante :*

$$\mathcal{E}_\gamma = \bigoplus_{\boldsymbol{d}\in\mathbb{N}^r} \bigoplus_{\lambda\vdash n} \pi_{\boldsymbol{d}} \circ \mathcal{P} \circ \pi_\lambda(\mathcal{E}_\gamma^{(1)})$$

De plus, en tant que $\mathrm{GL_r}$-module, le sous-espace de type λ, $\mathcal{P} \circ \pi_\lambda(\mathcal{E}_\gamma^{(1)})$, admet alors une description simple de son caractère comme la somme $\sum_{f\in\mathcal{B}_\lambda^{(r)}} \boldsymbol{q}^{\deg(f)}$, et cette somme est une fonction symétrique Schur-positive (voir théorème 2.3). Ainsi, comme conséquence du théorème 4.2, on obtient finalement le caractère de $\mathcal{E}_\gamma^{(r)}$.

Corollaire 4.1. *Le $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractère de l'espace engendré par V_γ et clos par dérivation et polarisation est $\mathcal{E}_\gamma^{(r)} = \sum_{\lambda\vdash n} \sum_\mu c_{\lambda\mu} s_\mu \otimes s_\lambda$, où les $c_{\lambda\mu}$ sont donnés par $\sum_\mu c_{\lambda\mu} s_\mu(\boldsymbol{q}) = \sum_{f\in\mathcal{B}_\lambda^{(r)}} \boldsymbol{q}^{\deg(f)}$.*

Exemple 4.2. *On décompose $\mathcal{E}_3^{(1)}$. Dans le tableau suivant, on donne pour chaque type $\lambda \vdash n$, une base de chaque composante homogène.*

Type	*Degré*	*Base de la composante homogène*
111	(3)	$-x_1^2x_2 + x_1x_2^2 + x_1^2x_3 - x_2^2x_3 - x_1x_3^2 + x_2x_3^2$
21	(2)	$x_1^2 - 2x_1x_2 + 2x_2x_3 - x_3^2$, $-2x_1x_2 + x_2^2 + 2x_1x_3 - x_3^2$
	(1)	$x_1 - x_3$, $x_2 - x_3$
3	(0)	*1*

Par polarisation, on obtient la décomposition en irréductibles de $\mathcal{E}_3^{(2)}$.

Type	*Degré*	*Base de la composante multihomogène*
111	(3, 0)	$x_1^2x_2 - x_1x_2^2 - x_1^2x_3 + x_2^2x_3 + x_1x_3^2 - x_2x_3^2$
	(2, 1)	$-x_1x_2y_1 + \frac{1}{2}x_2^2y_1 + x_1x_3y_1 - \frac{1}{2}x_3^2y_1 - \frac{1}{2}x_1^2y_2 - x_2x_3y_2$ $+x_1x_2y_2 + \frac{1}{2}x_3^2y_2 + \frac{1}{2}x_1^2y_3 - \frac{1}{2}x_2^2y_3 - x_1x_3y_3 + x_2x_3y_3$
	(1, 2)	$-\frac{1}{2}x_2y_1^2 + \frac{1}{2}x_3y_1^2 - x_1y_1y_2 + x_2y_1y_2 + \frac{1}{2}x_1y_2^2 - \frac{1}{2}x_3y_2^2$ $+x_1y_1y_3 - x_3y_1y_3 - x_2y_2y_3 + x_3y_2y_3 - \frac{1}{2}x_1y_3^2 + \frac{1}{2}x_2y_3^2$
	(0, 3)	$y_1^2y_2 - y_1y_2^2 - y_1^2y_3 + y_2^2y_3 + y_1y_3^2 - y_2y_3^2$
21	(2, 0)	$x_1x_2 - \frac{1}{2}x_2^2 - x_1x_3 + \frac{1}{2}x_3^2$, $\frac{1}{2}x_1^2 - \frac{1}{2}x_2^2 - x_1x_3 + x_2x_3$
	(1, 1)	$-x_1y_1 + x_3y_1 + x_2y_2 - x_3y_2 + x_1y_3 - x_2y_3$, $-x_1y_1 + x_2y_1 + x_1y_2 - x_3y_2 - x_2y_3 + x_3y_3$
	(1, 0)	$x_1 - x_2$, $-x_2 + x_3$
	(0, 2)	$-\frac{1}{2}y_1^2 + \frac{1}{2}y_2^2 + y_1y_3 - y_2y_3$, $-\frac{1}{2}y_1^2 + y_1y_2 - y_2y_3 + \frac{1}{2}y_3^2$
	(0, 1)	$-y_1 + y_3$, $-y_1 + y_2$
3	(0, 0)	1

Trouver une façon efficace de générer les sous-espaces de type λ pour tout $\lambda \vdash |\gamma|$ de $\mathcal{E}_\gamma$ et de les décomposer en composantes multihomogènes constitue le cœur du calcul. On en déduit immédiatement le caractère. Cependant, il est possible d'améliorer encore l'efficacité du calcul.

4.5 Exploitation des symétries sur les jeux de variables

Si on restreint à $\mathbb{S}_r$ l'action de $\mathrm{GL_r}$, alors l'espace $\mathcal{E}_\gamma$ est stable par permutation des lignes de X'. Ces permutations permettent de passer d'une composante multihomogène à une autre. Par exemple, pour $r = 3$, les composantes multihomogènes de degrés respectifs $(4, 2, 1)$ et $(2, 1, 4)$ sont isomorphes et peuvent être obtenues l'une de l'autre en appliquant la permutation appropriée sur les lignes de X'. Autrement dit, on a que

$$\pi_{\boldsymbol{d}}(\mathcal{E}_\gamma) \simeq \pi_{\lambda(\boldsymbol{d})}(\mathcal{E}_\gamma). \tag{4.6}$$

Notons que $\pi_{(d_1,d_2,\cdots,d_k,0,\cdots,0)} = \pi_{(d_1,d_2,\cdots,d_k)}$. Ainsi, il n'est pas nécessaire de calculer toutes les composantes multihomogènes pour obtenir le $\mathrm{GL_r}$-caractère. Par exemple, le polynôme $x_1^2 - 2x_1x_2 + 2x_2x_3 - x_3^2$ appartient à $\mathcal{E}_3^{(2)}$ (voir exemple 4.2). Comme ce polynôme est de degré $(1, 0)$, le polynôme correspondant de degré $(0, 1)$ c'est-dire $y_1^2 - 2y_1y_2 + 2y_2y_3 - y_3^2$, appartient aussi à $\mathcal{E}_3^{(2)}$. Cependant, il n'est pas nécessaire de calculer ce dernier explicitement. On va donc déduire le $\mathrm{GL_r}$-caractère des composantes multihomogènes de degré décroissant uniquement.

On a vu dans la section précédente que le $\mathrm{GL_r}$-caractère de $\mathcal{E}_\gamma$ est donné par le degré des composantes multihomogènes. En exploitant les symétries sur les jeux de variables, on sait que si on a une composante de degré $\boldsymbol{d}$, elle contribue au terme $\sum_{\sigma\in\mathbb{S}_r} q^{\boldsymbol{d}.\sigma} = m_{\lambda(\boldsymbol{d})}$ dans le caractère. Ainsi, on peut utiliser les fonctions symétriques monomiales au lieu d'une sommation pour récupérer le caractère de

$\mathrm{GL_r}$ dans chaque composante de λ, pour tout $\lambda \vdash n$. Ainsi, dans l'expression du caractère donnée au corollaire 4.1, les $c_{\lambda\mu}$ sont donnés par

$$\sum_{\mu} c_{\lambda\mu} s_\mu(\boldsymbol{q}) = \sum_{\substack{f \in \mathcal{B}_\lambda^{(r)} \\ \deg(f) \text{ décroissant}}} m_{\lambda(\deg(f))}. \tag{4.7}$$

Pour obtenir uniquement les composante multihomogènes de degré décroissant, les opérateurs de polarisation sont modifiés de sorte que chaque fois que le résultat d'une polarisation est un élément dont le degré n'est pas décroissant, on le corrige en appliquant la permutation appropriée sur les lignes. On note $\widetilde{P}^k_{\boldsymbol{u},\boldsymbol{v}}$ les opérateurs de polarisation modifiés. En effet, on ne peut pas simplement ignorer un élément s'il n'est pas de degré décroissant. Puisque les opérateurs de polarisation ne commutent pas tous entre eux (voir équation (4.2)), en tronquant un élément f, appartenant à une composante de degré non décroissant, dès son apparition dans le calcul, certains éléments de degré décroissant construits par polarisation de f ne seraient jamais calculés.

Exemple 4.3. *Prenons l'exemple de $V_3(\boldsymbol{x})$ que l'on a déjà vu dans l'exemple 1.5. Si on le polarise deux fois avec l'opérateur de polarisation classique $P^1_{\boldsymbol{x},\boldsymbol{y}}$, on obtient après la première polarisation*

$$\begin{aligned} P^1_{\boldsymbol{x},\boldsymbol{y}} V_3(\boldsymbol{x}) &= 2x_1x_2y_1 - x_2^2y_1 - 2x_1x_3y_1 + x_3^2y_1 + x_1^2y_2 - 2x_1x_2y_2 + 2x_2x_3y_2 \\ &\quad -x_3^2y_2 - x_1^2y_3 + x_2^2y_3 + 2x_1x_3y_3 - 2x_2x_3y_3 \end{aligned}$$

qui est de degré $(2,1)$. Dans ce cas, $P^1_{\boldsymbol{x},\boldsymbol{y}} V_3(\boldsymbol{x}) = \tilde{P}^1_{\boldsymbol{x},\boldsymbol{y}} V_3(\boldsymbol{x})$. Cependant, après la deuxième polarisation, on obtient

$$\begin{aligned} (P^1_{\boldsymbol{x},\boldsymbol{y}})^2 V_3(\boldsymbol{x}) &= 2x_2y_1^2 - 2x_3y_1^2 + 4x_1y_1y_2 - 4x_2y_1y_2 - 2x_1y_2^2 + 2x_3y_2^2 - 4x_1y_1y_3 \\ &\quad +4x_3y_1y_3 + 4x_2y_2y_3 - 4x_3y_2y_3 + 2x_1y_3^2 - 2x_2y_3^2 \end{aligned}$$

qui est de degré $(1,2)$. Alors que si on applique le polarisateur modifié $\tilde{P}^1_{\boldsymbol{x},\boldsymbol{y}}$, c'est-à-dire qu'on applique $P^1_{\boldsymbol{x},\boldsymbol{y}}$ puis on permute les jeux de variables pour obtenir un

degré décroissant, on obtient cette fois

$$\begin{aligned}\tilde{P}^1_{\boldsymbol{x},\boldsymbol{y}}(P^1_{\boldsymbol{x},\boldsymbol{y}}V_3(\boldsymbol{x})) &= 2y_2x_1^2 - 2y_3x_1^2 + 4y_1x_1x_2 - 4y_2x_1x_2 - 2y_1x_2^2 + 2y_3x_2^2 - 4y_1x_1x_3 \\ &\quad +4y_3x_1x_3 + 4y_2x_2x_3 - 4y_3x_2x_3 + 2y_1x_3^2 - 2y_2x_3^2\end{aligned}$$

dont le degré $(2,1)$ *est bien un degré décroissant.*

Conjecture 2. *Pour tout* $\mathcal{W}$ *sous-espace de* R'*, on note* $\widetilde{\mathcal{P}}(\mathcal{W})$ *la clôture de* $\mathcal{W}$ *par les opérateurs de polarisation modifiés* $\widetilde{P}^k_{\boldsymbol{u},\boldsymbol{v}}$*. Alors, pour tout* $\boldsymbol{d} \in \mathbb{N}^r$ *tel que* $d_1 \geq d_2 \geq \cdots \geq d_r$ *on a que* $\pi_{\boldsymbol{d}} \circ \mathcal{P}(\mathcal{E}^{(1)}_\gamma) = \pi_{\boldsymbol{d}} \circ \widetilde{\mathcal{P}}(\mathcal{E}^{(1)}_\gamma)$ *.*

Le résultat ci-dessus est présenté comme une conjecture car, bien que vérifié par le calcul, nous n'en avons pas de démonstration. Nous avons cependant certains éléments de preuve permettant de supporter cette conjecture. En effet, on peut remarquer qu'appliquer une transposition (ij) des jeux de variables $\boldsymbol{x}_i$ et $\boldsymbol{x}_j$ dans $f \in R'$ est équivalent à appliquer à f des opérateurs de polarisation de la forme $P^1_{\boldsymbol{u},\boldsymbol{v}}$. Ces opérateurs ne changent pas le degré de f et en utilisant les opérateurs appropriés autant de fois que nécessaire, on peut transformer toutes les variables $\boldsymbol{x}_i$ en variables $\boldsymbol{x}_j$, en passant par un jeu de variables intermédiaire $\boldsymbol{x}_k$ pour $k \neq i, j$. Il faudrait donc montrer que si $f \in \pi_{\boldsymbol{d}} \circ \mathcal{P}(\mathcal{E}^{(1)}_\gamma)$ est obtenu en polarisant un polynôme g dont le degré n'est pas décroissant alors il est possible d'obtenir f ou un multiple de f d'un polynôme g' de degré décroissant et d'une succession de polarisateurs dont les résultats intermédiaires sont de degré décroissant.

Comme on a vu qu'il est suffisant de considérer uniquement les composantes multihomogènes de degré décroissant pour obtenir le $\mathrm{GL_r}$-caractère, si ce résultat est vrai, on obtient bien le caractère voulu en utilisant la clôture par les opérateurs de polarisation modifiés mais en évitant de calculer de l'information redondante. Le gain apporté par cette stratégie en terme de coût de calcul est loin d'être négligeable (voir 5.1 pour plus de détails), il serait donc particulièrement utile d'avoir une preuve de cette conjecture.

4.6 Exploitation des antisymétries sur les colonnes

Le calcul du caractère de $\mathcal{E}_\gamma$ est difficile, non seulement à cause de sa dimension mais également à cause de la taille (degré et nombre de termes) des polynômes contenus dans cet espace. Par exemple, pour $\lambda \vdash 6$, le déterminant V_λ est un polynôme homogène de degré allant jusqu'à $\sum_{i=1}^{5} i = 15$ avec $6! = 720$ monômes. Cependant, il est possible de diminuer l'effort de calcul et la difficulté de manipulation en restreignant le nombre de termes des polynômes.

En effet, pour tout $f \in \pi_\lambda(\mathcal{E}_\gamma^{(r)})$, par définition de l'idempotent de Young (équation 2.1), f est antisymétrique pour les permutations qui préservent les colonnes de λ. Autrement dit, $f.\sigma = \mathrm{sgn}(\sigma) f$ pour tout $\sigma \in Q_\lambda$. Il est donc suffisant de conserver un représentant par orbite puisque si le monôme X^α apparaît dans f, alors les termes $\mathrm{sgn}(\sigma) X^\alpha.\sigma$ pour tout $\sigma \in Q_\lambda$ apparaissent également dans f. On choisit de garder comme représentant le plus petit monôme dans l'ordre lexicographique.

Définition 4.1. *Pour tout $f \in R'_\lambda$, avec R'_λ le sous-espace de type λ de R', on appelle* forme normale réduite *de f, l'expression obtenue de f en gardant uniquement le plus petit représentant pour l'ordre lexicographique de chaque orbite de Q_λ. On note π' l'application qui à f associe sa forme normale réduite.*

Par exemple, $V_{21} = -x_2\theta_1 + x_3\theta_1 + x_1\theta_2 - x_3\theta_2 - x_1\theta_3 + x_2\theta_3$ est de type 111 et a pour forme normale réduite $\pi'(V_{21}) = x_1\theta_2$. De même, $\partial x_2 V_{21} = \theta_1 - \theta_2$ est de type 21 et à pour forme normale réduite $\pi'(\partial x_2 V_{21}) = \theta_1$. On a alors que pour tout $f \in \pi_\lambda(\mathcal{E}_\gamma^{(r)})$,

$$f = \sum_{\sigma \in Q_\lambda} \mathrm{sgn}(\sigma) \pi'(f).\sigma = \pi'(f).c_\lambda. \tag{4.8}$$

De l'équation (4.8), on en déduit que $\pi_\lambda(\mathcal{E}_\gamma^{(r)}) \simeq \pi'(\pi_\lambda(\mathcal{E}_\gamma^{(r)}))$. De plus, il est clair que $\deg(f) = \deg(\pi'(f))$, donc $\pi_{\boldsymbol{d}} \circ \pi_\lambda(\mathcal{E}_\gamma^{(r)}) \simeq \pi_{\boldsymbol{d}} \circ \pi'(\pi_\lambda(\mathcal{E}_\gamma^{(r)}))$. Le $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-

caractère est donc conservé lors du passage aux formes normales réduites. Pour finir, il reste à montrer que la forme normale réduite $\pi'(f)$ est compatible avec la polarisation afin de s'assurer que l'on peut appliquer π' avant de polariser et ainsi manipuler le plus tôt possible dans le calcul la forme normale réduite des polynômes.

Proposition 4.3. *Pour tout* $f \in \pi_\lambda(\mathcal{E}_\gamma^{(r)})$, $\pi' \circ P^k_{\boldsymbol{u},\boldsymbol{v}} \circ \pi'(f) = \pi' \circ P^k_{\boldsymbol{u},\boldsymbol{v}}(f)$ *pour tous* $\boldsymbol{u}, \boldsymbol{v} \in X'$ *et pour tout* $k \in \mathbb{N}$.

Démonstration. Soit $f \in \pi_\lambda(\mathcal{E}_\gamma^{(r)})$, on a alors

$$\begin{aligned} \pi' \circ P^k_{\boldsymbol{u},\boldsymbol{v}}(f) &= \pi' \circ P^k_{\boldsymbol{u},\boldsymbol{v}}(\pi'(f).c_\lambda) \quad \text{(équation (4.8))} \\ &= \pi'\left(\left(P^k_{\boldsymbol{u},\boldsymbol{v}} \circ \pi'(f)\right).c_\lambda\right) \quad \text{(voir démonstration proposition 4.2)} \end{aligned}$$

De plus, par définition de π', on a que $\pi'(g.c_\lambda) = \pi'(g)$ pour tout $g \in R'_\lambda$. On a donc bien que $\pi' \circ P^k_{\boldsymbol{u},\boldsymbol{v}} \circ \pi'(f) = \pi' \circ P^k_{\boldsymbol{u},\boldsymbol{v}}(f)$. □

Exemple 4.4. *On reprend l'exemple 4.2 et on exploite à la fois les antisymétries sur les colonnes et les symétries sur les lignes, les composantes multihomogènes de* $\mathcal{E}_3$ *deviennent les suivantes :* $\pi_{111}(\mathcal{E}_3) = \mathbb{Q}\{x_1^2x_2\} \oplus \mathbb{Q}\{x_1x_2y_1 + \frac{1}{2}x_1^2y_2\}$ *;* $\pi_{21}(\mathcal{E}_3) = \mathbb{Q}\{x_1\} \oplus \mathbb{Q}\{-\frac{1}{2}x_1^2 + x_1x_2\} \oplus \mathbb{Q}\{x_1y_1 - x_2y_1 - x_1y_2\}$ *et* $\pi_3(\mathcal{E}_3) = \mathbb{Q}$. *On observe que seules les composantes de multidegré décroissant sont calculées. Les composantes de multidegré* $(1,2)$, $(0,3)$, $(0,2)$ *et* $(0,1)$ *ne sont pas présentes. De plus, les éléments de chaque base sont sous forme normale réduite*

Exemple 4.5. *De la même façon, pour l'exemple 4.1 les bases des composantes multihomogènes deviennent* $\pi_{111}(\mathcal{E}_{21}) = \mathbb{Q}\{x_1\theta_2\} \oplus \mathbb{Q}\{y_1\theta_2\}$ *et* $\pi_{21}(\mathcal{E}_{21}) = \mathbb{Q}\{\theta_1\}$.

4.7 Caractères associés aux partages 1^n et 21^{n-2}

Dans cette section, on donne les formules explicites du caractère pour deux types de partages. Tout d'abord, les caractères associés à des partages de la forme 1^n sont simples à calculer. En effet, $\mathcal{E}_{1^n} = 1 \otimes s_{1^n}$. Dans ce cas, V_{1^n} est de degré 0, seules des variables inertes interviennent dans ce déterminant. Il n'y a donc aucune dérivée partielle, ni aucun opérateur de polarisation qui puisse être appliqué. Donc $\mathcal{E}_{1^n} = \mathbb{Q}\{V_{1^n}\}$, et comme V_{1^n} est antisymétrique, le caractère est bien $1 \otimes s_{1^n}$.

Ensuite, afin d'illustrer notre stratégie de calcul, on montre comment calculer le caractère de $\mathcal{E}_{21^{n-2}}$ en passant par la décomposition en composantes isotypiques et en composantes multihomogènes.

Proposition 4.4. $\mathcal{E}_{21^{n-2}} = 1 \otimes s_{21^{n-2}} + s_1 \otimes s_{1^n}$

Démonstration. Rappelons que $V_{21^{n-2}} = \det(x_i^a \theta_i^b)_{\substack{1 \le i \le n \\ (a,b) \in 21^{n-2}}}$, autrement dit

$$V_{21^{n-2}} = \begin{vmatrix} 1 & x_1 & \theta_1 & \theta_1^2 & \dots & \theta_1^{n-2} \\ 1 & x_2 & \theta_2 & \theta_2^2 & \dots & \theta_2^{n-2} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \theta_n & \theta_n^2 & \dots & \theta_n^{n-2} \end{vmatrix}.$$

Donc $V_{21^{n-2}}$ est de la forme $V_{21^{n-2}}(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{i=1}^n x_i V_{n-1}(\boldsymbol{\theta}^{(i)})$ où $\boldsymbol{\theta}^{(i)}$ signifie que θ_i n'apparaît pas, $\boldsymbol{\theta}^{(i)} = (\theta_1, \theta_2, \cdots, \hat{\theta_i}, \cdots, \theta_n)$. Le polynôme $V_{n-1}(\boldsymbol{\theta}^{(i)})$ est donc antisymétrique pour les permutations qui fixent i. Les polarisations possibles sont $P^1_{\boldsymbol{u},\boldsymbol{v}} = \sum_i v_i \partial_{u_i}$ pour tout couple de jeux de variables $\boldsymbol{u}$ et $\boldsymbol{v}$. Or $P^1_{\boldsymbol{x},\boldsymbol{v}} V_{21^{n-2}}(\boldsymbol{x}, \boldsymbol{\theta}) = V_{21^{n-2}}(\boldsymbol{v}, \boldsymbol{\theta})$ pour tout $\boldsymbol{v}$ puisque $V_{21^{n-2}}(\boldsymbol{x}, \boldsymbol{\theta})$ est de degré 1. De même, pour tous $\boldsymbol{u}$ et $\boldsymbol{v}$, $P^1_{\boldsymbol{u},\boldsymbol{v}} V_{21^{n-2}}(\boldsymbol{u}, \boldsymbol{\theta}) = V_{21^{n-2}}(\boldsymbol{v}, \boldsymbol{\theta})$. Donc la composante de $\mathcal{E}_{21^{n-2}}$ de degré 1 est $\mathcal{E}^{\langle 1 \rangle}_{21^{n-2}} = \mathbb{Q}\{V_{21^{n-2}}(\boldsymbol{u}, \boldsymbol{\theta}); \boldsymbol{u} \in X', \boldsymbol{u} \neq \boldsymbol{\theta}\}$, d'où $\mathcal{E}^{\langle 1 \rangle}_{21^{n-2}} = s_1 \otimes s_{1^n}$.

Considérons maintenant la composante de degré 0. Les seules dérivées partielles

que l'on peut appliquer sont les dérivées partielles de degré 1 : $\partial_{x_i} V_{21^{n-2}} = V_{n-1}(\boldsymbol{\theta}^{(i)})$. Ainsi $\mathcal{E}^{\langle 0\rangle}_{21^{n-2}} = \mathbb{Q}\{V_{n-1}(\boldsymbol{\theta}^{(i)}), 1 \leq i \leq n\}$. Montrons que cette composante appartient à la composante isotypique de type 21^{n-2}. Pour cela, on utilise l'idempotent de Young associé au partage 21^{n-2} : $y_{21^{n-2}} = r_{21^{n-2}} c_{21^{n-2}}$ avec $r_{21^{n-2}} = (12)$ et $c_{21^{n-2}} = \sum_{\substack{\sigma\in\mathbb{S}_n \\ \sigma(2)=2}} \mathrm{sgn}(\sigma)\sigma$; que l'on applique à $V_{n-1}(\boldsymbol{\theta}^{(i)})$. Mais tout d'abord, calculons $V_{n-1}(\boldsymbol{\theta}^{(i)}).\sigma = V_{n-1}(\boldsymbol{\theta}^{(i)}.\sigma)$. Pour tout $\sigma \in \mathbb{S}_n$,

$$V_{n-1}(\boldsymbol{\theta}^{(i)}).\sigma = \det(\theta^b_{\sigma(j)})_{\substack{1\leq j\leq n \\ j\neq\sigma(i) \\ 0\leq b\leq n-2}} = \mathrm{sgn}(\sigma)\det(\theta^b_j)_{\substack{1\leq j\leq n \\ j\neq\sigma(i) \\ 0\leq b\leq n-2}} = \mathrm{sgn}(\sigma) V_{n-1}(\boldsymbol{\theta}^{(\sigma(j))}) \tag{4.9}$$

Appliquons maintenant l'idempotent de Young. Pour $i = 1$, puisque $V_{n-1}(\boldsymbol{\theta}^{(2)})$ est antisymétrique pour les permutations qui fixent 2 et d'après l'équation 4.9, on a

$$\begin{aligned} V_{n-1}(\boldsymbol{\theta}^{(1)}).y_{21^{n-2}} &= \sum_{\substack{\sigma\in\mathbb{S}_n \\ \sigma(2)=2}} \mathrm{sgn}(\sigma) V_{n-1}(\boldsymbol{\theta}^{(1)}).(12).\sigma \\ &= \sum_{\substack{\sigma\in\mathbb{S}_n \\ \sigma(2)=2}} \mathrm{sgn}(\sigma)(-V_{n-1}(\boldsymbol{\theta}^{(2)})).\sigma \\ &= -\sum_{\substack{\sigma\in\mathbb{S}_n \\ \sigma(2)=2}} \mathrm{sgn}(\sigma)^2 V_{n-1}(\boldsymbol{\theta}^{(2)}) \\ &= -(n-1)! V_{n-1}(\boldsymbol{\theta}^{(2)}). \end{aligned} \tag{4.10}$$

Lorsque $i = 2$, on a

$$\begin{aligned} V_{n-1}(\boldsymbol{\theta}^{(2)}).y_{21^{n-2}} &= \sum_{\substack{\sigma\in\mathbb{S}_n \\ \sigma(2)=2}} \mathrm{sgn}(\sigma) V_{n-1}(\boldsymbol{\theta}^{(2)}).(12).\sigma \\ &= \sum_{\substack{\sigma\in\mathbb{S}_n \\ \sigma(2)=2}} \mathrm{sgn}(\sigma)(-V_{n-1}(\boldsymbol{\theta}^{(1)})).\sigma \\ &= -\sum_{\substack{\sigma\in\mathbb{S}_n \\ \sigma(2)=2}} V_{n-1}(\boldsymbol{\theta}^{(\sigma(1))}) \\ &= -(n-2)! \sum_{\substack{i=1 \\ i\neq 2}}^{n} V_{n-1}(\boldsymbol{\theta}^{(i)}). \end{aligned} \tag{4.11}$$

De manière similaire, lorsque $i > 2$ on obtient que

$$\begin{aligned} V_{n-1}(\boldsymbol{\theta}^{(i)}).y_{21^{n-2}} &= -\sum_{\substack{\sigma\in\mathbb{S}_n \\ \sigma(2)=2}} V_{n-1}(\boldsymbol{\theta}^{(\sigma(i))}) \\ &= -(n-2)!\sum_{\substack{i=1 \\ i\neq 2}}^{n} V_{n-1}(\boldsymbol{\theta}^{(i)}). \end{aligned} \tag{4.12}$$

De plus, par construction, $V_{21^{n-2}}$ est un polynôme harmonique, donc $\sum_i \partial_{x_i} V_{21^{n-2}} = 0$, d'où $\sum_i V_{n-1}(\boldsymbol{\theta}^{(i)}) = 0$. Ainsi, d'après les équations 4.11 et 4.12, lorsque $i \neq 1$, on a $V_{n-1}(\boldsymbol{\theta}^{(i)}).y_{21^{n-2}} = (n-2)!V_{n-1}(\boldsymbol{\theta}^{(2)})$. Ainsi, $\mathbb{Q}\{V_{n-1}(\boldsymbol{\theta}^{(i)}), 1 \leq i \leq n\}$ est de type 21^{n-2}. De plus, d'après les calculs effectués, cette composante irréductible est de multiplicité 1 dans la composante isotypique. Donc, $\mathcal{E}^{\langle 0\rangle}_{21^{n-2}} = 1 \otimes s_{21^{n-1}}$ et finalement $\mathcal{E}_{21^{n-2}} = 1 \otimes s_{21^{n-1}} + s_1 \otimes s_{1^n}$. □

4.8 Caractères associés à des vecteurs d'entiers

Développer une stratégie de calcul efficace du caractère permet d'explorer et de calculer les caractères pour plusieurs familles de diagrammes. Cette exploration permet alors de conjecturer et de démontrer d'autres propriétés des caractères des espaces étudiés. Dans cette section, on s'intéresse à des diagrammes possédant au moins deux lignes dont toutes les cases ont un voisin immédiat à gauche (excepté les cases de la bordure gauche du diagramme). Dans ce cas, γ est le diagramme d'un vecteur d'entiers et on a des espaces dans lesquels les variables inertes jouent un rôle dans le caractère de $\mathrm{GL_r}$.

Proposition 4.5. *Soit γ un vecteurs d'entiers de poids n. On note γ^{ij} le vecteur obtenu de γ en échangeant les parts γ_i et γ_j, alors*

$$\mathcal{E}_{\gamma^{ij}}(\boldsymbol{q};\boldsymbol{w}) = \mathcal{E}_{\gamma}(\boldsymbol{q};\boldsymbol{w}),$$

avec $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ *et* $\boldsymbol{q} = (q_1, q_2, \cdots, q_r)$.

Démonstration. On rappelle que les colonnes du déterminant V_γ sont indicées par les cases du diagramme γ. On rappelle également que le caractère associé à γ ne dépend pas de l'ordre de lecture des cases du diagramme (voir définition 3.3). Supposons que $\gamma_i < \gamma_j$. Ainsi, les cases de γ ayant pour coordonnées $(\gamma_i + k, j)$ pour $1 \leq k \leq \gamma_j - \gamma_i$ (voir les cases colorées dans la figure 4.1) vont devenir dans γ^{ij} des cases ayant pour coordonnées $(\gamma_i + k, i)$ pour $1 \leq k \leq \gamma_j - \gamma_i$ mais on peut les lire à la même position que dans γ. Dans la figure 4.1 ci-dessous, les valeurs indiquées dans les cases correspondent à l'ordre de lecture des cases.

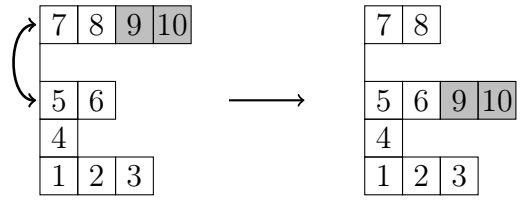


FIGURE 4.1 Positions des cases dans γ^{ij} (à droite) par rapport à γ (à gauche).

Soit $\boldsymbol{x_1} = (x_{11}, x_{12}, \ldots, x_{1n})$ et $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_n)$. On peut écrire V_γ sous la forme $V_\gamma = \sum_{k=1}^{n!} f_k(\boldsymbol{x_1}) g_k(\boldsymbol{\theta})$, où f_k et g_k sont des polynômes. De même, on écrit $V_{\gamma^{ij}}$ sous la forme $V_{\gamma^{ij}} = \sum_{k=1}^{n!} \tilde{f}_k(\boldsymbol{x_1}) \tilde{g_k}(\boldsymbol{\theta})$, où $\tilde{f}_k$ et $\tilde{g_k}$ sont également des polynômes. D'après la remarque précédente, on a que $f_k(\boldsymbol{x_1}) = \tilde{f}_k(\boldsymbol{x_1})$ pour tout k puisque $f_k(\boldsymbol{x_1})$ et $\tilde{f}_k(\boldsymbol{x_1})$ dépendent uniquement des abscisses des cases et qu'elles sont les mêmes, lues dans le même ordre, dans γ et dans γ^{ij}. De plus, pour tout k, il existe $\boldsymbol{d} = (d_1, d_2, \ldots, d_n) \in \mathbb{N}^n$ avec $\sum_{i=1}^{n} d_i = \gamma_j - \gamma_i$ tel que $\tilde{g_k}(\boldsymbol{\theta}) = g_k(\boldsymbol{\theta^d})$ où $\boldsymbol{\theta^d} = (\theta_1^{d_1}, \theta_2^{d_2}, \ldots, \theta_n^{d_n})$. En effet, à chaque monôme de V_γ, on peut associer un monôme de $V_{\gamma^{ij}}$ dans lequel $f_k(\boldsymbol{x_1}) = \tilde{f}_k(\boldsymbol{x_1})$. Alors, pour un tel couple de monômes, les indices des variables de $\boldsymbol{\theta}$ dans $g_k(\boldsymbol{\theta})$ et $\tilde{g_k}(\boldsymbol{\theta})$ sont les mêmes, seuls les degrés, qui dépendent de l'ordre des lignes dans le diagramme, sont différents.

Enfin, soit $a \in \mathbb{N}$, $1 \leq a \leq n$, on a $\partial^l_{x_{1a}} V_{\gamma^{ij}} = \sum_{k=1}^{n!} \partial^l_{x_{1a}}(f_k(\boldsymbol{x_1}))\tilde{g}_k(\boldsymbol{\theta})$. Donc $\mathcal{E}^{(1)}_{\gamma} \simeq \mathcal{E}^{(1)}_{\gamma^{ij}}$ en tant que $(\mathrm{GL_1} \times \mathbb{S}_\mathrm{n})$-modules.

De plus, d'après la proposition 4.2, $\mathcal{E}^{(r)}_{\gamma} = \mathcal{P}(\mathcal{E}^{(1)}_{\gamma})$ et le $\mathrm{GL_r}$-caractère est déterminé par le degré des composantes multihomogènes (corollaire 4.1). Or puisque $f_k(\boldsymbol{x_1}) = \tilde{f}_k(\boldsymbol{x_1})$ pour tout k, le multidegré est préservé lors de l'échange de lignes dans γ. D'où $\mathcal{E}_{\gamma^{ij}}(\boldsymbol{q}; \boldsymbol{w}) = \mathcal{E}_{\gamma}(\boldsymbol{q}; \boldsymbol{w})$. □

Corollaire 4.2. *Soit γ un vecteur d'entiers de poids n et $\lambda(\gamma)$ le partage obtenu de γ en réordonnant les parts, alors*

$$\mathcal{E}_{\lambda(\gamma)}(\boldsymbol{q}; \boldsymbol{w}) = \mathcal{E}_{\gamma}(\boldsymbol{q}; \boldsymbol{w}).$$

Démonstration. Il suffit d'appliquer successivement la proposition 4.5 jusqu'à avoir ordonné les parts de γ. □

Le corollaire 4.2 nous permet donc de conclure qu'il n'est pas nécessaire d'étudier le cas d'espaces associés à des vecteurs d'entiers (ou à des compositions), il est suffisant de s'intéresser uniquement aux partages. On a ainsi réduit le nombre de caractères qu'il est nécessaire de calculer et d'étudier.

CHAPITRE V

IMPLÉMENTATION ET RÉSULTATS EXPÉRIMENTAUX

Dans ce chapitre, on décrit l'implémentation effectuée en appliquant les stratégies établies au chapitre 4. L'outil ainsi développé a permis de réaliser de nombreux calculs jamais effectués auparavant à notre connaissance. On a mentionné au chapitre 4 que le temps de calcul sans optimisation pour des espaces associés à des partages de taille 6 était estimé à plusieurs centaines d'années. On a actuellement réduit ce temps de calcul à moins de deux jours. Cela a ainsi permis de confirmer la conjecture 1 pour les diagrammes qui sont des partages obtenus à partir de chemins de Dyck pour des valeurs de m et n allant jusqu'à 7. On a également pu explorer de nouvelles idées dans le cas des diagrammes qui ne représentent pas des partages ou des vecteurs d'entiers. On présente dans ce chapitre les résultats obtenus et les avancées réalisées.

5.1 Implémentation et coût de calcul

Pour l'implémentation des calculs, nous avons utilisé le logiciel SageMath qui est un logiciel de calcul formel gratuit de code source ouvert. Le choix de ce logiciel nous a semblé évident dans la mesure où il s'agit d'un logiciel gratuit et donc accessible à tous, il est ainsi facile de partager le code implémenté. SageMath est aussi un logiciel de code source ouvert, il est donc possible d'accéder au code

déjà développé et même de le modifier au besoin. Enfin, les outils déjà présents et performants, comme les fonctions symétriques, nous ont permis de nous concentrer uniquement sur les calculs nouveaux effectués dans le cadre de ce travail.

Dans les chapitres 1 et 3, nous avons défini $\mathcal{E}_\gamma^{(r)}$ comme l'espace engendré par V_γ et clos par dérivation et polarisation, $\mathcal{E}_\gamma^{(r)} = \mathcal{P}_\mathcal{D}(\mathbb{Q}\{V_\gamma\})$. Ainsi, nous avons tout d'abord implémenté un utilitaire général, `Subspace(vectors, operators)`, qui calcule une base du plus petit espace vectoriel contenant les vecteurs donnés et stable sous l'action des opérateurs linéaires donnés. Il s'agit principalement d'algèbre linéaire. Les lignes d'une matrice échelonnée représentent une base de l'espace calculé, et on dispose d'une liste d'éléments de la forme $op(v)$, où op est un opérateur et v un élément du sous-espace en construction, qu'il reste à calculer et à insérer dans la matrice. Le coût du calcul est donc de $\mathcal{O}(N^2M)$ où N est la dimension du sous-espace résultant et M la dimension de l'espace ambiant.

Dans notre cas, le vecteur donné en argument est V_γ et les opérateurs sont les dérivées partielles et les polarisateurs. Bien évidemment, on ne va pas effectuer le calcul en une seule étape puisque nous avons vu que l'on peut calculer en deux étapes la clôture par dérivation puis par polarisation (proposition 4.1). Ainsi, on va tout d'abord calculer $\mathcal{E}_\gamma^{(1)} = \mathcal{D}(\mathbb{Q}\{V_\gamma\})$.

Exemple 5.1. *Calculons une base de* $\mathcal{E}_3^{(1)}$ *en un jeu de variables* $\boldsymbol{x} = (x_1, x_2, x_3)$. *On a que* $V_3 = (x_2 - x_3)(-x_1 + x_2)(x_1 - x_3)$ *et on va utiliser les opérateurs* ∂x_1, ∂x_2 *et* ∂x_3. *On obtient alors la base suivante pour* $\mathcal{E}_3^{(1)}$ *:* $\{-x_1^2x_2 + x_1x_2^2 + x_1^2x_3 - x_2^2x_3 - x_1x_3^2 + x_2x_3^2, -x_1x_2 + \frac{1}{2}x_2^2 + x_1x_3 - \frac{1}{2}x_3^2, \frac{1}{2}x_1^2 - x_1x_2 + x_2x_3 - \frac{1}{2}x_3^2, x_1 - x_3, x_2 - x_3, 1\}$. *De plus l'utilitaire implémenté permet aussi d'obtenir la dimension qui ici est 6. Les commandes à utiliser pour obtenir ce résultat sont données en annexe B.*

Puis, il reste à calculer la clôture par polarisation de $\mathcal{E}_\gamma^{(1)}$. Ce calcul peut également être effectué à l'aide de l'utilitaire `Subspace`. Cependant, d'après la proposition

4.2, il est plus avantageux de décomposer d'abord $\mathcal{E}_\gamma^{(1)}$ en composantes isotypiques avant de polariser. En effet, si on polarise d'abord, on a vu que le coût de calcul est $\mathcal{O}(N^2M)$ où N est la dimension du sous-espace résultat et M la dimension de l'espace ambiant. Mais si on projette d'abord sur les composantes isotypiques, on va ensuite polariser des espaces de plus petite dimension. On note N_λ la dimension de chaque composante isotypique, le coût de calcul devient alors $\mathcal{O}(\sum_{\lambda\vdash n} N_\lambda^2 M)$ au lieu de $\mathcal{O}((\sum_{\lambda\vdash n} N_\lambda)^2 M)$. Nous avons donc implémenté l'application π_λ de paramètre $\lambda \vdash n$ qui projette les polynômes sur leur composante de type λ. Utilisée avec `Subspace`, on obtient alors une base de chaque composante isotypique.

Exemple 5.2. *On a effectué des tests sur le calcul de* $\mathcal{E}_{31}^{(3)}$ *et on obtient que le temps de calcul pour obtenir une base de* $\mathcal{E}_{31}^{(3)}$ *en calculant directement* $\mathcal{P}_\mathcal{D}(\mathbb{Q}\{V_{31}\})$ *est de* $5.72s$*, alors que le temps de calcul sur la même machine pour calculer* $\mathcal{P}\circ\mathcal{D}(\mathbb{Q}\{V_{31}\})$ *est de* $3.45s$ *et enfin le temps de calcul, toujours dans les mêmes conditions, de* $\bigoplus_{\lambda\vdash n}\mathcal{P}\circ\pi_\lambda\circ\mathcal{D}(\mathbb{Q}\{V_{31}\})$ *est de* $1.3s$.

Une fois que l'on a une base de $\mathcal{E}_\gamma^{(r)}$, il reste à retrouver son bicaractère. Pour $\mathrm{GL_r}$, nous avons vu qu'il est donné par le multidegré et pour $\mathbb{S}_\mathrm{n}$ par le type de chaque composante isotypique. À cette étape, on a décomposé $\mathcal{E}_\gamma^{(r)}$ en composante isotypiques, il nous reste donc à projeter sur les composantes multihomogènes pour obtenir le caractère de $\mathrm{GL_r}$ dans chaque composante isotypique (corollaire 4.1).

On définit le *degré* d'un opérateur comme la variation du degré obtenue après application de cet opérateur sur un polynôme de R'. Par exemple, le degré de ∂x_i^k est $\alpha \in \mathbb{N}^r$ avec $\alpha_i = -k$ et $\alpha_j = 0$ pour tout $j \neq i$, et le degré de $P^k_{\boldsymbol{x}_i,\boldsymbol{x}_j}$ est $\beta \in \mathbb{N}^r$ où $\beta_i = -k$, $\beta_j = 1$ et $\beta_\ell = 0$ pour tout $\ell \neq i, j$. L'utilitaire `Subspace` peut être adapté pour projeter automatiquement sur les composantes multihomogènes en tout temps. Les entrées sont alors une collection de vecteurs homogènes et

d'opérateurs avec leurs degrés, et une matrice échelonnée différente est maintenue pour chaque degré. On obtient ainsi immédiatement le $\mathrm{GL_r}$-caractère par lecture sur le résultat. En procédant ainsi, on réduit encore le temps de calcul puisque l'on a alors $N_\lambda = \sum_{\boldsymbol{d}\in\mathbb{N}^r} N_\lambda^{\boldsymbol{d}}$ où $N_\lambda^{\boldsymbol{d}}$ est la dimension de la composante de degré $\boldsymbol{d}$ dans la composante de type λ.

Exemple 5.3. *Dans les mêmes condition qu'à l'exemple 5.2, si on utilise la projection sur les composantes multihomogènes, on réduit cette fois le temps de calcul d'une base de* $\mathcal{E}_{31}^{(3)}$ *à* $930ms$. *Le caractère est ensuite récupéré en quelques millisecondes supplémentaires (*$29ms$*).*

De plus, comme on peut travailler sur les composantes isotypiques indépendamment pour appliquer les opérateurs de polarisation, cela nous permet de paralléliser les calculs. Alors puisque $\mathcal{E}_\gamma^{(1)}$ se décompose en au plus $p(n)$ composantes isotypiques, où $p(n)$ est le nombre de partages de $n = |\gamma|$, si on effectue en parallèle la polarisation de ces composantes isotypiques, on réduit le temps de calcul de la polarisation au temps de calcul maximum nécessaire pour polariser une de ces composantes isotypiques.

À la section 4.5, nous avons vu qu'il n'est pas nécessaire de calculer toutes les composantes multihomogènes puisqu'on peut déduire le caractère de $\mathrm{GL_r}$ à partir des composantes de degré décroissant uniquement. Nous avons donc implémenté les opérateurs de polarisation modifiés $\widetilde{P}_{\boldsymbol{u},\boldsymbol{v}}^k$ afin de ne conserver que des polynômes de degré décroissant. Ainsi, on réduit la complexité en espace puisque les composantes multihomogènes ne sont pas toutes conservées en mémoire. En effet, pour un degré total donné d, on conserve uniquement une composante multihomogène par partage de d au lieu d'une composante par vecteur d'entier de poids d à r parts. De plus, on réduit la complexité en temps en diminuant le nombre de polynômes sur lesquels appliquer des polarisateurs.

Exemple 5.4. *Toujours à la suite des exemples 5.2 et 5.3 et dans les mêmes conditions, en exploitant en plus les symétries sur les lignes de X', on passe d'un temps de calcul de $930ms$ à $560ms$ pour obtenir une base de $\mathcal{E}^{(3)}_{31}$.*

Exemple 5.5. *En exploitant la commutativité des dérivées et des polarisateurs, la projection sur les composantes isotypiques et la projection sur les composantes multihomogènes, le temps de calcul pour obtenir une base de $\mathcal{E}^{(4)}_{211}$ est de $5.12s$. En exploitant en plus les symétries sur les lignes de X', on réduit ce temps de calcul à $1.29s$.*

Il serait encore plus avantageux de prévoir quelles suites d'opérateurs il n'est pas nécessaire d'appliquer. Cela pourrait être fait en étudiant les espaces de plus haut poids et en s'intéressant à la théorie des cristaux mais nous n'avons pas eu l'occasion de regarder plus en détails dans cette direction.

La dernière stratégie de calcul, abordée dans la section 4.6, concerne les antisymétries dans les polynômes eux-mêmes. On a montré que l'on peut travailler avec la forme normale réduite des polynômes. En travaillant sur des polynômes possédant un nombre réduit de termes, on réduit le temps de calcul lors de l'application des opérateurs, et l'espace nécessaire pour conserver les polynômes en mémoire. Les antisymétries sont exploitées dès l'étape de projection de $\mathcal{E}^{(1)}_{\gamma}$ sur ses composantes isotypiques (proposition 4.3). En effet, une fois le type isotypique d'un polynôme f identifié, on connaît l'ensemble des permutations pour lesquelles f est antisymétrique. En pratique, au moment où l'on applique l'idempotent de Young y_λ, la forme normale réduite est calculée pour chaque polynôme obtenu. Ensuite, les polynômes de la base de chaque composante isotypique sont maintenus sous leur forme réduite durant la polarisation. En théorie, le coût de calcul de la polarisation peut ainsi être jusqu'à $n!$ fois moins élevé. En pratique, on gagne un peu moins à cause du redressement des monômes à effectuer à chaque étape. Notons

également qu'une attention particulière a été portée à l'implémentation afin d'obtenir un code le plus proche possible des mathématiques. Ainsi, il est possible en tout temps de vérifier les antisymétries pour lesquelles une forme normale réduite à été calculée et ainsi retrouver le polynôme d'origine.

Exemple 5.6. *On reprend l'exemple* 5.5 *du calcul d'une base de* $\mathcal{E}^{(3)}_{221}$. *Si on exploite les antisymétries, on passe de* $5.12s$ *à* $567ms$ *et si on exploite simultanément les symétries sur les jeux de variables et les antisymétries, on obtient un temps de calcul de* $430ms$.

Exemple 5.7. *On prend maintenant l'exemple du calul de* $\mathcal{E}^{(4)}_{32}$. *On exploite dans tous les cas la commutativité des dérivées et des polarisateurs, la projection sur les composantes isotypiques et la projection sur les composantes multihomogènes. Si on utilise les symétries sur les jeux de variables mais pas les antisymétries dans les polynômes, le temps de calcul est de* $31.2s$. *Si on utilise les antisymétries mais pas les symétries sur les jeux de variables, le temps de calcul est de* $5.4s$. *Si on exploite les deux stratégies, le temps de calcul diminue à* $1.91s$.

Toutes les stratégies d'optimisations mises en œuvres peuvent être utilisées indépendamment les unes des autres dans le package Sage implémenté. Si on veut utiliser toutes les stratégies et les opérateurs usuels, une fonction par défaut est également implémentée. Elle renvoie le résultat final du caractère. Voir les annexes A et B pour plus de détails et d'exemples.

5.2 Tables de caractères

Grâce au code implémenté, avec les stratégies décrites au chapitre 4, nous avons été en mesure d'obtenir le caractère de $\mathcal{E}_\lambda$ pour les partages λ avec $1 \leq |\lambda| \leq 6$. De plus, on a également obtenu $\mathcal{E}^{(2)}_\lambda$ pour certains partages de 7. On présente ici les tables de caractères 5.1 et 5.2 obtenues grâce au code implémenté pour $1 \leq n \leq 4$.

TABLE 5.1 Table de caractères de $\mathcal{E}_\lambda$ pour $1 \leq |\lambda| \leq 3$

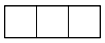
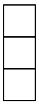
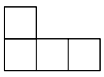
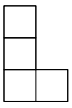
Diagramme	Caractère
	$1 \otimes s_1$
	$1 \otimes s_2 + s_1 \otimes s_{11}$
	$1 \otimes s_{11}$
	$1 \otimes s_3 + (s_1 + s_2) \otimes s_{21} + (s_{11} + s_3) \otimes s_{111}$
	$1 \otimes s_{21} + s_1 \otimes s_{111}$
	$1 \otimes s_{111}$

TABLE 5.2 Table de caractères de $\mathcal{E}_\lambda$ pour $|\lambda| = 4$

Diagramme	Caractère
	$1 \otimes s_4 + (s_1 + s_2 + s_3) \otimes s_{31} + (s_2 + s_{21} + s_4) \otimes s_{22}$ $+(s_{11} + s_{21} + s_3 + s_{31} + s_4 + s_5) \otimes s_{211}$ $+(s_{111} + s_{31} + s_{41} + s_6) \otimes s_{1111}$
	$1 \otimes s_{31} + s_1 \otimes s_{22} + (s_1 + s_2) \otimes s_{211} + (s_{11} + s_3) \otimes s_{1111}$
	$1 \otimes s_{22} + s_1 \otimes s_{211} + s_2 \otimes s_{1111}$
	$1 \otimes s_{211} + s_1 \otimes s_{1111}$
	$1 \otimes s_{1111}$

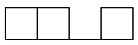
Les autres tables étant trop longues pour être présentées ici, elles peuvent être trouvées en annexe C. Notons également que le résultat est ici donné en termes de fonctions de Schur mais qu'il est également possible d'obtenir le caractère dans d'autres bases des fonctions symétriques (monomiales, sommes de puissances, homogènes ou élémentaires) en le spécifiant en argument.

5.3 Diagrammes troués

Jusqu'à présent, seuls les diagrammes de partages ou de vecteurs d'entiers ont été considérés. Le point commun de ces deux types de diagrammes est que dans chaque ligne les cases sont justifiées à gauche. Il n'y a pas de trous dans les lignes. Cependant, les diagrammes obtenus des chemins de Dyck définis dans la section 3.4 ne sont pas tous des diagrammes de partages. Certains de ces diagrammes possèdent des trous (voir $\gamma(4,6)$ dans la figure 1.6). Pour ces diagrammes, la définition utilisée jusqu'à présent pour $\mathcal{E}_\gamma$ ne donne pas le résultat souhaité, à savoir $\mathcal{E}_{\gamma(m,n)}(q,t;\boldsymbol{w}) = \boldsymbol{e}_{m,n}(q,t;\boldsymbol{w})$ (conjecture 1). Dans cette section, on présente les expérimentations menées pour trouver une définition de $\mathcal{E}_{\gamma(m,n)}$ qui donne le caractère souhaité.

La première observation que l'on peut faire lorsque l'on compare les valeurs obtenues pour $\mathcal{E}_{\gamma(m,n)}(q,t;\boldsymbol{w})$ et $\boldsymbol{e}_{m,n}(q,t;\boldsymbol{w})$, dans le cas de diagrammes troués, est que l'expression obtenue pour $\mathcal{E}_{\gamma(m,n)}(q,t;\boldsymbol{w})$ contient plus de termes que $\boldsymbol{e}_{m,n}(q,t;\boldsymbol{w})$.

Exemple 5.8. *Dans le cas $n = 3$ et $m = 5$, on a le diagramme suivant.*



On obtient alors l'expression suivante pour le caractère associé à ce diagramme $\mathcal{E}_{\gamma(5,3)}(q,t;\boldsymbol{w}) = (s_{1,1}+s_{2,1}+s_3+s_4)\otimes s_{1,1,1}+(s_1+s_{1,1}+2s_2+s_3)\otimes s_{2,1}+(1+s_1)\otimes s_3$. *Cependant,* $\boldsymbol{e}_{5,3}(q,t;\boldsymbol{w}) = (s_{2,1}+s_4)\otimes s_{1,1,1} + (s_{1,1}+s_2+s_3)\otimes s_{2,1} + s_1\otimes s_3$. *Si on exprime la composante alternante dans la base des monomiales sur deux jeux de variables, on obtient, dans l'expression du caractère, le terme* $(m_{1,1}+2m_{2,1}+m_{2,2}+m_3+m_{3,1}+m_4)\otimes s_{1,1,1}$, *alors que dans* $\boldsymbol{e}_{5,3}(q,t;\boldsymbol{w})$, *on obtient le terme* $(m_{2,1}+m_{2,2}+m_{3,1}+m_4)\otimes s_{1,1,1}$.

Exemple 5.9. *Dans le cas $n = 3$ et $m = 6$, on a le diagramme suivant.*

On obtient alors l'expression suivante pour le caractère associé à ce diagramme.

$$\begin{aligned}\mathcal{E}_{\gamma(6,3)}(q,t;\boldsymbol{w}) = {} & (s_{1,1}+s_{2,1}+s_3+2s_{3,1}+s_4+s_{4,1}+2s_5+s_6)\otimes s_{1,1,1}\\ & +(s_1+s_{1,1}+2s_2+2s_{2,1}+3s_3+s_{3,1}+3s_4+s_5)\otimes s_{2,1}\\ & +(1+s_1+2s_2+s_3)\otimes s_3\end{aligned}$$

Cependant, $\boldsymbol{e}_{6,3}(q,t;\boldsymbol{w}) = (s_{2,2}+s_{4,1}+s_6)\otimes s_{1,1,1}+(s_{2,1}+s_{3,1}+s_4+s_5)\otimes s_{2,1}+(s_{1,1}+s_3)\otimes s_3$. Si on exprime la composante alternante dans la base des monomiales sur deux jeux de variables, les termes apparaissant dans l'expression du caractère mais pas dans $\boldsymbol{e}_{6,3}(q,t;\boldsymbol{w})$ sont : $m_{1,1}+2m_{2,1}+m_{2,2}+m_3+2m_{3,1}+2m_{3,2}+2m_4+2m_{4,1}+2m_5$.

À partir de maintenant, on note $\mathcal{M}_\gamma^{(r)}$ le plus petit sous-espace contenant V_γ, ses dérivées partielles et fermé par polarisation pour r jeux de variables. On cherche à quotienter $\mathcal{M}_\gamma$ afin que le caractère de l'espace quotient ainsi construit soit égal à $\boldsymbol{e}_{m,n}(q,t;\boldsymbol{w})$.

Définition 5.1. *Soit f un polynôme diagonalement symétrique de degré total au moins 1, on appelle* dérivée symétrique *l'opérateur noté $f(\partial_X)$ et défini par la fonction f dans laquelle on remplace x_{ij}^k par $\partial_{x_{ij}}^k$ pour tout (i,j) et pour tout k.*

On note $\Lambda^*\mathcal{M}_\gamma^{(r)}$ l'union des images de $\mathcal{M}_\gamma^{(r)}$ par $f(\partial_X)$ pour tout f diagonalement symétrique.

$$\Lambda^*\mathcal{M}_\gamma^{(r)} := \bigcup_f f(\partial_X)\mathcal{M}_\gamma^{(r)}$$

Enfin, on note $\mathcal{E}_\gamma^{(r)}$ le sous-espace de $R' = \mathbb{Q}[X']$ défini pour tout diagramme γ

par $\mathcal{E}_\gamma^{(r)} := \mathcal{M}_\gamma^{(r)}/\Lambda^*\mathcal{M}_\gamma^{(r)}$. Encore une fois, s'il n'y a pas de confusion possible, on omet l'exposant (r) si r est suffisamment grand (voir théorème 4.1). Notons que les opérateurs de différentiation D_α (voir définition 1.5) forment une base des dérivées symétriques. On peut donc se restreindre à considérer uniquement des dérivées symétriques de la forme D_α. De plus, on va pouvoir se limiter à appliquer les opérateurs D_α pour lesquels α est un vecteur d'entiers de poids N où N est au plus le degré de V_γ. La raison pour laquelle on souhaite quotienter $\mathcal{M}_\gamma$ par l'espace des polynômes générés par les dérivées symétriques apparaît ici plus clairement. En effet, on souhaite obtenir une construction qui généralise les polynômes harmoniques diagonaux, or ces derniers sont définis comme étant solutions de systèmes de la forme $D_\alpha f = 0$ pour α un vecteur d'entiers ayant le nombre de parts approprié et $f \in R'$.

Nous allons maintenant vérifier la commutation des dérivées symétriques avec les opérateurs de polarisation. Il est suffisant de le vérifier pour des compositions de la forme $\alpha = (0, \ldots, 0, \alpha_j, 0, \ldots, 0)$ avec $\alpha_j \neq 0$ et où la position j de la part non nulle de α correspond à la position dans X' du jeu de variables $\boldsymbol{y}$ intervenant dans le polarisateur $P^k_{\boldsymbol{x},\boldsymbol{y}}$. Soit f un polynôme de R' et k un entier strictement positif, on a alors

$$D_\alpha P^k_{\boldsymbol{x},\boldsymbol{y}} f - P^k_{\boldsymbol{x},\boldsymbol{y}} D_\alpha f = \sum_{i=1}^{n} \partial y_i^{\alpha_j} \sum_{j=1}^{n} y_j \partial x_j^k f - \sum_{j=1}^{n} y_j \partial x_j^k \sum_{i=1}^{n} \partial y_i^{\alpha_j} f \tag{5.1}$$

On considère les termes pour lesquels $i = j$ et on omet temporairement les indices.

$$\partial y^{\alpha_j} y \partial x^k f - y \partial x^k \partial y^{\alpha_j} f = \alpha_j \partial y^{\alpha_j - 1} \partial x^k f + y \partial y^{\alpha_j} \partial x^k f - y \partial x^k \partial y^{\alpha_j} f = \alpha_j \partial_y^{\alpha_j - 1} \partial x^k f.$$

En utilisant ce résultat dans 5.1, on obtient finalement

$$D_\alpha P^k_{\boldsymbol{x},\boldsymbol{y}} f - P^k_{\boldsymbol{x},\boldsymbol{y}} D_\alpha f = \alpha_j \sum_{i=1}^{n} \partial {y_i}^{\alpha_j - 1} \partial x_i^k f = \alpha_j D_\beta f, \tag{5.2}$$

avec $\beta = (\dots, k, \dots, \alpha_j - 1, \dots)$, où $\alpha_j - 1$ et k sont les seules parts non nulles de β et leurs positions correspondent respectivement aux positions dans X_θ des jeux de variables $\boldsymbol{y}$ et $\boldsymbol{x}$. On peut donc en conclure que l'ordre d'application de ces opérateurs n'aura pas d'incidence sur le résultat final.

Proposition 5.1. *Pour tout $\mathcal{W}$, sous-espace de R', $\mathcal{P}(\Lambda^*\mathcal{W}) = \Lambda^*\mathcal{P}(\mathcal{W})$.*

On remarque qu'appliquer une dérivée symétrique de la forme $\sum_{i=1}^n \partial x_i^c$ à V_γ revient à faire la somme de tous les diagrammes obtenus en bougeant une case de γ de c cases vers la gauche.

Proposition 5.2. *Soit γ un diagramme quelconque de taille n alors*

$$\sum_{i=1}^{n} \partial x_i^c V_\gamma = \sum_{\tilde{\gamma}} \beta V_{\tilde{\gamma}},$$

où $\tilde{\gamma}$ est un diagramme obtenu de γ en remplaçant une case (a, b) de γ par la case $(a - c, b)$. De plus, $|\beta| = a(a-1)\dots(a-c+1)$ si $a \geq c$ et $\beta = 0$ si $a < c$ et le signe de β est positif si la case $(a - c, b)$ est lue à la même position dans $\tilde{\gamma}$ que la case (a, b) dans γ.

Démonstration. Fixons un entier i dans la somme de gauche, et développons le déterminant V_γ selon la i-ème ligne, on obtient alors

$$\partial x_i^c V_\gamma = \sum_{(a,b)\in\gamma} \partial x_i^c(x_i^a)\theta_i^b \operatorname{com}(V_\gamma)_{i,(a,b)} = \sum_{(a,b)\in\gamma} |\beta| x_i^{a-c}\theta_i^b \operatorname{com}(V_\gamma)_{i,(a,b)},$$

où $\operatorname{com}(V_\gamma)_{i,(a,b)}$ est le cofacteur de V_γ d'indice $(i, (a, b))$. On somme à nouveau sur i et on a alors

$$\sum_{i=1}^{n} \partial x_i^c V_\gamma = \sum_{i=1}^{n} \sum_{(a,b)\in\gamma} |\beta| x_i^{a-c}\theta_i^b \operatorname{com}(V_\gamma)_{i,(a,b)} = \sum_{(a,b)\in\gamma} |\beta| \sum_{i=1}^{n} x_i^{a-c}\theta_i^b \operatorname{com}(V_\gamma)_{i,(a,b)}. \tag{5.3}$$

Or la somme (5.3) correspond au développement selon la colonne indicée (a, b) du déterminant $V_{\tilde{\gamma}}$ avec $\tilde{\gamma}$ obtenu de γ en remplaçant la case (a, b) par la case $(a - c, b)$, multiplié par -1 si l'ordre de lecture des cases diffère entre les deux diagrammes.

De plus, on remarque que la somme va être nulle si la case $(a - c, b)$ est une case déjà existante dans γ puisque dans ce cas, on aurait deux colonnes identiques dans $V_{\tilde{\gamma}}$. Ainsi, $\sum_{i=1}^{n} \partial x_i^c V_\gamma = \sum_{(a,b)\in\gamma} \beta V_{\tilde{\gamma}} = \sum_{\tilde{\gamma}} \beta V_{\tilde{\gamma}}$. □

Exemple 5.10. *Pour $n = 2$ et $m = 4$, $\gamma(4, 2) = \{(0, 0), (2, 0)\}$ et*

$$V_{\gamma(4,2)} = \begin{vmatrix} 1 & x_1^2 \\ 1 & x_2^2 \end{vmatrix} = x_2^2 - x_1^2.$$

On applique $p_1(\partial_{x_1}, \partial_{x_2})$ à $V_{\gamma(4,2)}$ et on obtient $\sum_{i=1}^{2} \partial_{x_i} V_{\gamma(4,2)} = 2(x_2 - x_1)$. On calcule ensuite le membre de gauche de l'égalité de la proposition 5.2. Le seul diagramme $\tilde{\gamma}$ qu'il est possible d'obtenir à partir de γ est construit en remplaçant la case $(2, 0)$ par $(1, 0)$. On a alors que β est positif et $\beta = 2$, d'où

$$\sum_{\tilde{\gamma}} \beta V_{\tilde{\gamma}} = 2 \begin{vmatrix} 1 & x_1 \\ 1 & x_2 \end{vmatrix} = 2(x_2 - x_2) = \sum_{i=1}^{2} \partial_{x_i} V_{\gamma(4,2)}.$$

Dans le cas des diagrammes avec des trous, lorsque l'on construit $\mathcal{M}_\gamma$, certaines séries d'opérateurs correspondent à déplacer des cases vers des trous. En quotientant $\mathcal{M}_\gamma$, on souhaite fixer les cases de γ et donc imposer que toute suite d'opérateurs correspondant à déplacer une case de γ vers un trou donne un résultat nul. Cela ne change rien dans le cas où γ ne contient pas de trous.

Corollaire 5.1. *Lorsque γ est le diagramme associé à un partage ou à un vecteur d'entiers, $\mathcal{E}_\gamma = \mathcal{M}_\gamma$.*

Démonstration. En effet, si γ est un partage ou un vecteur d'entiers, pour toute

case $(a,b) \in \gamma$ et pour tout entier c, soit $(a-c,b)$ est une case de γ, soit $a < c$ et dans les deux cas, le déterminant résultant du mouvement de la case (a,b) est nul. Donc $\sum_{i=1}^{n} \partial x_i^c V_\gamma = 0$ pour tout $c \in \mathbb{N}$. De plus, on a vu que l'ordre d'application des opérateurs n'influe pas sur le résultat (proposition 5.1). Alors, puisque les dérivées symétriques en un jeu de variables sont toutes de la forme $\sum_{i=1}^{n} \partial x_i^c$, on a que $\Lambda^* \mathcal{M}_\gamma = \emptyset$ et donc $\mathcal{E}_\gamma = \mathcal{M}_\gamma$. □

Le fait de quotienter $\mathcal{M}_\gamma$ donne un résultat plus proche de celui attendu mais l'espace obtenu n'est toujours pas celui espéré en général.

Exemple 5.11. *Pour $\gamma(5,3)$, on obtient que le caractère associé à $\Lambda^* \mathcal{M}_{\gamma(5,3)}$ est* $1 \otimes s_3 + (s_1 + s_2) \otimes s_{2,1} + (s_{1,1} + s_3) \otimes s_{1,1,1}$. *Et alors* $\mathcal{E}_{\gamma(5,3)}(q,t;\boldsymbol{w}) = s_1 \otimes s_3 + (s_{1,1} + s_2 + s_3) \otimes s_{2,1} + (s_{2,1} + s_4) \otimes s_{1,1,1} = \boldsymbol{e}_{5,3}(q,t;\boldsymbol{w})$.

Par contre, pour $\gamma(6,3)$, on obtient $\mathcal{E}_{\gamma(6,3)}(q,t;\boldsymbol{w}) = (s_2 + s_3) \otimes s_3 + (s_{2,1} + s_3 + s_{3,1} + 2s_4 + s_5) \otimes s_{2,1} + (s_{3,1} + s_{4,1} + s_5 + s_6) \otimes s_{1,1,1}$. *On exprime la différence des composantes alternantes en terme de fonctions monomiales sur deux jeux de variables :*

$$\begin{aligned}\mathcal{E}_{\gamma(6,3)}(q,t;\boldsymbol{w}) - \boldsymbol{e}_{6,3}(q,t;\boldsymbol{w}) =& m_2 \otimes s_3 + (m_{2,1} + m_{2,2} + m_3 + m_{3,1} + m_4) \otimes s_{2,1} \\ &+ (m_{3,1} + m_{3,2} + m_{4,1} + m_5) \otimes s_{1,1,1}.\end{aligned}$$

On remarque donc que même si dans certains cas, quotienter par la somme des images des dérivées symétriques suffit à nous donner le résultat recherché, cette définition de $\mathcal{E}_\gamma$ n'est toujours pas satisfaisante. L'exemple du diagramme $\gamma(6,3)$ est en réalité le plus petit exemple pour lequel cela ne fonctionne pas. À l'aide du code implémenté pour le calcul des caractères, on explore donc la possibilité de faire intervenir d'autres opérateurs.

5.4 Opérateurs de Steenrod

On considère une nouvelle famille d'opérateurs, celle des opérateurs de Steenrod. On voit apparaître ces opérateurs naturellement lorsque l'on calcule des commutateurs de la forme $\left[P^l_{\boldsymbol{y},\boldsymbol{x}} - P^k_{\boldsymbol{x},\boldsymbol{y}}\right]$. Soit $\mathcal{C}_\gamma$ le sous-espace de R' engendré par un déterminant de type Vandermonde V_γ et clos par dérivation, polarisation et par les opérateurs de Steenrod. On souhaite comparer $\mathcal{C}_\gamma$ et $\mathcal{M}_\gamma$ afin d'évaluer l'impact des opérateurs de Steenrod.

Définition 5.2. L'opérateur de Steenrod $St^k_{\boldsymbol{x}}$ *pour* $k > 1$ *est défini par*

$$St^k_{\boldsymbol{x}} = \sum_{i=1}^{n} x_i \partial^k_{x_i}$$

où $\boldsymbol{x}$ *est un jeu de variables de* X' *(*$\boldsymbol{x} \neq \boldsymbol{\theta}$*).*

Proposition 5.3 (Classique)**.** *Il est suffisant de considérer uniquement les opérateurs de Steenrod pour* $k = 2$ *et* $k = 3$*. Les autres opérateurs de Steenrod peuvent être obtenus par composition des deux premiers.*

Démonstration. On peut vérifier par un simple calcul que $St^4_{\boldsymbol{x}} = St^3_{\boldsymbol{x}} St^2_{\boldsymbol{x}} - St^2_{\boldsymbol{x}} St^3_{\boldsymbol{x}}$. De plus,

$$\begin{aligned}
St^k_{\boldsymbol{x}} St^2_{\boldsymbol{x}} - St^2_{\boldsymbol{x}} St^k_{\boldsymbol{x}} &= \sum_{i=1}^{n} x_i \partial^k_{x_i} \sum_{j=1}^{n} x_j \partial^2_{x_j} - \sum_{j=1}^{n} x_j \partial^2_{x_j} \sum_{i=1}^{n} x_i \partial^k_{x_i} \\
&= \sum_{i=1}^{n} (x_i \partial {x_i}^k x_i \partial^2_{x_i} - x_i \partial^2_{x_i} x_i \partial^k_{x_i}) \\
&= \sum_{i=1}^{n} x_i (k \partial {x_i}^{k+1} + x_i \partial^{k+2}_{x_i}) - x_i ((k-1)\partial^{k+1}_{x_i} + x_i \partial^{k+2}_{x_i}) \\
&= \sum_{i=1}^{n} x_i \partial {x_i}^{k+1} = St^{k+1}_{\boldsymbol{x}}
\end{aligned}$$

□

Notons que $St^k_{\boldsymbol{x}} = P^k_{\boldsymbol{x},\boldsymbol{x}}$ mais on préfère utiliser des notations différentes car ces deux familles d'opérateurs jouent des rôles différents. Comme avec les opérateurs précédents, on veut s'assurer de l'impact de l'ordre dans lequel ils sont appliqués. On commence par calculer le commutateur des opérateurs de Steenrod $St^k_{\boldsymbol{x}}$ et des polarisateurs. On a alors deux polarisateurs différents à prendre en compte $P^k_{\boldsymbol{x},\boldsymbol{y}}$ et $P^k_{\boldsymbol{y},\boldsymbol{x}}$.

$$St^l_{\boldsymbol{x}} P^k_{\boldsymbol{x},\boldsymbol{y}} f - P^k_{\boldsymbol{x},\boldsymbol{y}} St^l_{\boldsymbol{x}} f = \sum_{i=1}^{n} x_i \partial^l_{x_i} \sum_{j=1}^{n} y_j \partial^k_{x_j} f - \sum_{j=1}^{n} y_j \partial^k_{x_j} \sum_{i=1}^{n} \partial^l_{x_i} f \tag{5.4}$$

Par un calcul similaire aux précédents (voir section 4.2), on obtient que

$$St^l_{\boldsymbol{x}} P^k_{\boldsymbol{x},\boldsymbol{y}} f - P^k_{\boldsymbol{x},\boldsymbol{y}} St^l_{\boldsymbol{x}} f = -k \sum_{i=1}^{n} y_i \partial^{k+l-1}_{x_i} f = -k P^{k+l-1}_{\boldsymbol{y},\boldsymbol{x}} f. \tag{5.5}$$

Dans ce cas, on obtient à nouveau un opérateur de polarisation, on n'a donc pas besoin de porter attention à l'ordre dans lequel sont appliqués ces opérateurs. On remarque également que l'on a effectué le calcul avec un polarisateur de la forme $P^{k_1}_{\boldsymbol{x},\boldsymbol{y}}$ et que l'on a obtenu un multiple d'un polarisateur de la forme $P^{k_2}_{\boldsymbol{y},\boldsymbol{x}}$. Cela nous permet d'en déduire que considérer ensemble les polarisateurs descendants, c'est-dire les polarisateurs de la forme $P^k_{\boldsymbol{x}_i,\boldsymbol{x}_j}$ avec $i < j$, et les opérateurs Steenrod, est équivalent à utiliser tous les polarisateurs sans les opérateur de Steenrod. Enfin, pour le second polarisateur considéré, on a

$$St^l_{\boldsymbol{x}} P^k_{\boldsymbol{y},\boldsymbol{x}} f - P^k_{\boldsymbol{y},\boldsymbol{x}} St^l_{\boldsymbol{x}} f = l \sum_{i=1}^{n} x_i \partial^{l-1}_{x_i} \partial^k_{y_i} f. \tag{5.6}$$

Dans ce cas, on obtient un nouvel opérateur non considéré auparavant dans la construction de $\mathcal{E}_\gamma$, on peut donc en conclure que l'ordre dans lequel on applique ces opérateurs a une importance et que l'ajout des opérateurs de Steenrod aura un impact sur $\mathcal{C}_\gamma$. De plus, l'image de $\Lambda^* \mathcal{C}_\gamma$ par les opérateurs de Steenrod n'est

pas stable par polarisation et donc n'est pas nécessairement une représentation de $\mathrm{GL_r}$.

On termine par vérifier la commutativité des opérateurs de Steenrod et des dérivées partielles. Il n'est pas nécessaire d'étudier la commutativité des opérateurs de Steenrod avec les dérivées symétriques puisque ce calcul nous ramènerait au cas des dérivées partielles. Pour les opérateurs de Steenrod et les dérivées partielles, on en conclut que l'ordre d'application n'a pas d'importance puisqu'on obtient

$$\partial_{x_i}^k St_{\boldsymbol{x}}^l f - St_{\boldsymbol{x}}^l \partial_{x_i}^k f = \partial_{x_i}^k \sum_{j=1}^{n} x_j \partial_{x_j}^l f - \sum_{j=1}^{n} x_j \partial_{x_j}^l \partial_{x_i}^k f = k \partial_{x_i}^{k+1} f. \tag{5.7}$$

L'ajout des opérateurs de Steenrod lors du calcul de $\mathcal{C}_\gamma$ est pertinent car cela permet d'avoir un équivalent de la polarisation en un seul jeu de variables. En effet, le caractère en un jeu de variables n'est pas affecté par ces opérateurs mais l'espace ainsi construit est cohérent avec la construction en plusieurs jeux de variables.

Deux définitions de $\Lambda^* \mathcal{C}_\gamma$ ont été testées : l'union des images de $\mathcal{C}_\gamma$ par les dérivées symétriques et l'union des images par les dérivées symétriques et les opérateurs de Steenrod. Aucune de ces deux définitions n'a donné un résultat qui a semblé plus intéressant que l'autre. Dans le cas du diagramme $\gamma(6,3)$ de l'exemple 5.9, des exemples de calculs sont données en annexe B.

Notons que l'un des problèmes que nous avons actuellement provient des espaces de plus haut poids. Les espaces de plus haut poids sont les composantes $\mathcal{E}_\gamma^{\langle \mu \rangle}$ de degré $\boldsymbol{d}$ avec $\boldsymbol{d} = \lambda(\mu)$. Ces composantes contribuent par un terme m_μ au caractère de $\mathrm{GL_r}$ et ce terme contribue au terme s_μ dans le caractère de $\mathrm{GL_r}$. Par exemple, nous n'arrivons pas à obtenir le terme $s_{2,2} \otimes s_{1,1,1}$ qui apparaît dans $\boldsymbol{e}_{\gamma(6,3)}(q,t;\boldsymbol{w})$, or ce terme est contenu dans le terme $s_{3,1} \otimes s_{1,1,1}$: $s_{3,1} =$

$s_{2,2} + m_{1,1,1,1} + m_{2,1,1} + m_{3,1}$. Pour le moment, le résultat du calcul pour $\gamma(6,3)$ contient les termes $(m_{1,1,1,1} + m_{2,1,1} + m_{3,1}) \otimes s_{1,1,1}$ et le quotient calculé ne nous permet pas de les supprimer de manière à obtenir uniquement s_{22}.

Nous n'avons donc pas encore trouvé de solution qui fonctionne dans le cas général. Ainsi la définition de $\mathcal{E}_\gamma$ reste encore à affiner. Cependant, les calculs menés nous permettent d'avoir déjà le caractère de $\mathcal{M}_\gamma$ ou de $\mathcal{C}_\gamma$ pour tout diagramme γ. De plus, les efforts mis dans la conception d'un package modulable et réutilisable ont permis de mener de nombreuses expérimentations avec différentes combinaisons d'opérateurs comme les dérivées symétriques et les opérateurs de Steenrod mais également des opérateurs de polarisation multiple (sur plusieurs jeux de variables en même temps) qui apparaissent dans l'équation (5.6). Si le calcul de $\mathcal{E}_{\gamma(6,3)}(q,t;\boldsymbol{w})$ n'a pas encore pu être mené à bien, nous avons cependant avancé dans la compréhension de ces espaces.

CONCLUSION

Dans cette thèse, on a introduit des sous-modules de $\mathbb{Q}[X']$, les polynômes en r jeux de n variables et un jeu de variables inertes. Ces sous-modules, $\mathcal{E}_\gamma^{(r)}$, sont associés à des diagrammes γ et construits à partir d'un déterminant de type Vandermonde et par clôture par dérivation et polarisation. Le but était de développer des stratégies pour le calcul des $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractères de ces espaces. On s'intéresse à ces caractères, entre autre, pour leur liens avec d'autres domaines comme l'algèbre elliptique de Hall par exemple. On souhaite donc les calculer pour mener à bien des expérimentations informatiques à des fins de catégorisation.

La difficulté majeure dans le calcul de ces caractères est la grande dimension des espaces étudiés ainsi que le haut degré et le grand nombre de termes des polynômes qu'ils contiennent. On a donc présenté les stratégies de calcul qui ont été exploitées et implémentées afin de rendre ces calculs possibles pour des cas qui n'avaient pas déjà été obtenus à la main ou à l'ordinateur. Tout d'abord, on peut donner une borne sur le nombre de jeux de variables à partir de laquelle le caractère se stabilise. Puis, on a vu comment utiliser l'idempotent de Young pour obtenir le caractère de $\mathbb{S}_\mathrm{n}$ et la graduation par le degré afin d'obtenir directement le caractère de $\mathrm{GL_r}$. Combiné aux propriétés de quasi-commutativité des opérateurs de dérivation et de polarisation, cela a permis de réduire considérablement le temps de calcul. En effet, le théorème 4.2 montre qu'il est possible de décomposer l'espace $\mathcal{E}_\gamma^{(r)}$ en $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-composantes irréductibles calculées sur un jeu de variables puis polarisées indépendamment. Ensuite, on a exploité les symétries sur les lignes de la matrice X' ainsi que les antisymétries présentes dans les polynômes, afin de

gagner encore en espace et en temps. Pour finir, on a vu qu'il était possible de paralléliser le calcul du caractère par composante isotypique de $\mathbb{S}_n$. Cette stratégie est implémentée mais pour le moment, elle n'a pas encore toute son utilité car le calcul est trop exigeant en terme de mémoire pour les diagrammes de 6 cases et plus, là où elle apporterait une réelle contribution.

De plus, la réflexion menée sur ces stratégies de calcul a permis de mieux comprendre certaines propriétés de ces espaces. On a, entre autres, pu démontrer que les espaces associés à des diagrammes de vecteurs d'entiers peuvent être ramenés à des espaces associés à des partages, réduisant ainsi le nombre des possibilités à étudier.

Dans le futur, il serait intéressant de continuer dans cette direction d'optimisation du code. Comme évoqué précédemment, il serait avantageux de pouvoir mettre en place une parallélisation efficace. De plus, il a été mentionné dans le chapitre 4, que l'on n'exploite pas encore au maximum les propriétés des opérateurs de polarisation dans la construction des espaces En particulier, on pourrait espérer exploiter les représentations de l'algèbre de Lie de GL_r, données par les opérateurs de polarisation simples, pour restreindre encore le calcul aux espaces de plus haut poids.

Le travail effectué jusqu'à maintenant a permis de confirmer et de suggérer des conjectures. Par exemple, on a pu confirmer la conjecture 1 pour les partages allant jusqu'à 6, ainsi que d'autres non citées dans ce document et non publiées. La flexibilité du code produit a permis de mener une campagne d'expérimentations qui, même si elle n'a pas abouti, a nourri la réflexion sur la bonne définition de $\mathcal{E}_\gamma$. Il reste cependant beaucoup de potentiel d'exploration que l'amélioration du code rendrait possible. Cela permettrait alors de pouvoir obtenir d'autres exemples et ainsi de formuler, confirmer et prouver d'autres conjectures. En particulier, dans

le cas des diagrammes troués évoqués au chapitre 5, on aimerait disposer d'un outil de calcul rapide afin de générer plus d'exemples et de poursuivre la réflexion présentée au chapitre 5.

Qui plus est, nous aimerions dans un futur proche intégrer le module implémenté dans Sage afin qu'il soit utilisable par d'autres. Le code est modulaire, avec des parties utilisables dans un cadre plus large, comme par exemples l'utilitaire `Subspace` de calcul de sous-espaces gradués. Nous avons donc espoir qu'il puisse être réutilisé et adapté à d'autres contextes.

Enfin, il serait intéressant de calculer les caractères de ces modules pour des groupes de Coxeter W autres que le groupe symétrique. Le code actuel peut être facilement adapté à ces groupes pour peu que l'on ait un moyen efficace pour extraire les représentations irréductibles des polynômes harmoniques en un jeu de variables, tel que les modules de Specht, un projecteur isotypique ou mieux un projecteur analogue au projecteur de Young. Pour terminer, mentionnons que dans un article récent, R. Orellana et M. Zabrocki (Orellana et Zabrocki, 2019) introduisent des modules similaires à $\mathcal{E}_\gamma$, appelés supers espaces, contenant un jeu de variables non commutatif au lieu d'un jeu de variables inertes. Il serait donc intéressant de voir les liens entre ces deux approches et d'adapter le code implémenté pour calculer les caractères de ces espaces.

ANNEXE A

CODE

L'entièreté du code implémenté dans le cadre de cette thèse est disponible dans un répertoire public sur le site GitHub qui est accessible au lien suivant : `https://github.com/nthiery/harmonic-modules`. Les utilisateurs sont libres de le télécharger, de l'utiliser et de le modifier (licence GPL).

Le code est documenté et contient de nombreux exemples. Pour l'utiliser, le module `pypersist` développé par M. Torpey (`https://github.com/mtorpey/pypersist`) est nécessaire. Le code peut ensuite être téléchargé et installé comme un module Sage à l'aide de la commande : `sage -pip install -e .` (le point fait partie de la commande).

Pour de simples essais, sans installation, ni sauvegarde du travail, une feuille de démonstration Jupyter est accessible par le service Binder. Pour y accéder, il suffit de cliquer sur le lien Binder présent dans le README sur le dépôt GitHub. Il sera nécessaire d'attendre quelques minutes que le chargement s'effectue puis les feuilles Jupyter présentes dans le répertoire pourront être ouvertes et manipulées. La feuille intitulée `demo.ipynb` contient des explications et une introduction au fonctionnement du code. Il est conseillé de commencer par regarder cette démonstration avant de regarder le code ou les autres exemples plus avancés présents dans les autres feuilles de travail Jupyter.

À titre de référence, le restant de cette annexe contient une partie du code implémenté dans le cadre de cette thèse. Afin de s'en tenir à un nombre de pages raisonnable, on omet le code de certaines fonctions et on en donne seulement une description, on omet également les importations de packages auxiliaires et la documentation a été significativement réduite. Le code complet et intégralement documenté est disponible en ligne.

Par soucis de transparence, le nom du ou des auteur.s est mentionné pour chaque fichier. Précisons tout de même que le code contenu dans les fichiers `utilities.pyx` et `matrix_of_vector.py` a été entièrement écrit par N. Thiéry et le code contenu dans les fichiers `antisymetric_utilities.pyx`, `subspace.py` et `young_idempotent.py` a été majoritairement écrit par N. Thiéry et amélioré et adapté adapté aux besoins des calculs décrits dans cette thèse par P. Hubert. Ensuite, le code contenu dans les fichiers `isomorphic_object.py` et `diagonal_polynomial_ring.py` a été ébauché par N. Thiéry mais a majoritairement été développé jusqu'à sa version actuelle par P. Hubert. Enfin, les fichiers `diagram.py` et `character.py` ont été entièrement écrits par P. Hubert.

Listing A.1 Fichier utilities.pyx

```
# Author : Nicolas Thiery

cpdef items_of_vector(Element v):
    """
    Return an iterator over the pairs ``(index, coefficient)``
       for ``v``.

    INPUT::
    - ``v`` -- an element of some some vector space or free
       module

    EXAMPLES:
    This handles indexed free module elements::
        sage: E = CombinatorialFreeModule(QQ, [1,2,4,8,16])
        sage: v = E.an_element(); v
        2*B[1] + 2*B[2] + 3*B[4]
        sage: list(items_of_vector(v))
        [(1, 2), (2, 2), (4, 3)]
```

```
17
18      multivariate polynomials::
19          sage: P = QQ['x,y,z']
20          sage: x,y,z = P.gens()
21          sage: p = (x+y+1)^2; p
22          x^2 + 2*x*y + y^2 + 2*x + 2*y + 1
23          sage: list(items_of_vector(p))
24          [((1, 0, 0), 2),
25           ((1, 1, 0), 2),
26           ((0, 0, 0), 1),
27           ((2, 0, 0), 1),
28           ((0, 1, 0), 2),
29           ((0, 2, 0), 1)]
30      """
31      if isinstance(v, CombinatorialFreeModule.Element):
32          return v
33      else:
34          try:
35              return v.dict().items()
36          except AttributeError:
37              return items_of_vector(v.lift())
38
39  cpdef act_on_polynomial(p, PermutationGroupElement sigma):
40      """
41      EXAMPLES::
42          sage: x,y,z,t = QQ['x,y,z,t'].gens()
43          sage: s = PermutationGroupElement([(1,2,3,4)])
44          sage: p = 2*x^2*y+3*z
45          sage: act_on_polynomial(p, s)
46          2*x*t^2 + 3*y
47      """
48      R = p.parent()
49      return R({ETuple(sigma._act_on_list_on_position(list(<ETuple>
           t))): c
50                for t, c in p.dict().iteritems() })
51
52  cpdef list diagonal_swap(list exponents, int n, int r, int j1,
       int j2):
53      """
54      Swap in place two columns of a diagonal exponent vector.
55
56      EXAMPLES::
57          sage: l = [1,2,3,4,5,6,7,8]
58          sage: diagonal_swap(l, 4, 2, 1, 3)
59          sage: l
60          [1, 4, 3, 2, 5, 8, 7, 6]
61      """
62      cdef int i
63      for i in range(r):
```

```
        exponents[i*n+j1], exponents[i*n+j2] = exponents[i*n+j2],
            exponents[i*n+j1]

cpdef int diagonal_cmp(list exponents, int n, int r, int j1, int
    j2):
    """
    Compare lexicographically two columns of a diagonal exponent
        vector.

    EXAMPLES::
        sage: l = [1, 1, 2, 2, 0, 1, 1, 0]
        sage: diagonal_cmp(l, 4, 2, 0, 1)
        -1
        sage: diagonal_cmp(l, 4, 2, 1, 0)
        1
        sage: diagonal_cmp(l, 4, 2, 3, 3)
        0
    """
    cdef int i
    cdef int c
    for i in range(r):
        c = cmp(exponents[i*n+j1], exponents[i*n+j2])
        if c:
            return c
    return 0

cpdef reverse_sorting_permutation(t):
    """
    Return a permutation `p` such that is decreasing

    EXAMPLES::
        sage: t = [3, 3, 1, 2]
        sage: s = reverse_sorting_permutation(t); s
        [1, 2, 4, 3]
        sage: [t[s[i]-1] for i in range(len(t))]
        [3, 3, 2, 1]
    """
    return ~(Word([-i for i in t]).standard_permutation())

cpdef destandardize(self):
    """
    Return the smallest word whose standard permutation is ``self
        ``

    EXAMPLES::
        sage: for p in Permutations(3): print(p, destandardize(p)
            )
        [1, 2, 3] [0, 0, 0]
        [1, 3, 2] [0, 1, 0]
```

```
108         [2, 1, 3] [1, 0, 1]
109         [2, 3, 1] [1, 1, 0]
110         [3, 1, 2] [1, 0, 0]
111         [3, 2, 1] [2, 1, 0]
112     """
113     n = len(self)
114     sigma = ~self
115     c = 0
116     w = [None] * n
117     for i in range(1,n+1):
118         w[sigma(i)-1] = c
119         if i < n and sigma(i+1) < sigma(i):
120             c += 1
121     return w
122 
123 cpdef index_filling(t):
124     """
125     Return the index filling of this standard tableau.
126     The index filling of ‘t‘ is the semi standard tableau with
        lowest content whose standardized row reading coincides
        with the row reading of ‘t‘.
127     """
128     return sage.combinat.tableau.from_shape_and_word(t.shape(),
        destandardize(t.reading_word_permutation()))
```

Listing A.2 Fichier antisymmetric_utilities.pyx

```
1  # Authors : Pauline Hubert, Nicolas Thiery
2  
3  cpdef diagonal_antisort(exponents, int n, int r, tuple
       positions_list):
4      """
5      Sort columns decreasingly at the given positions.
6      See :func:‘antisymmetric_normal‘ for details.
7  
8      INPUT:
9      - ‘‘exponents ‘‘ -- a list, seen as an ‘r\times n‘ array
10     - ‘‘r‘‘, ‘‘n‘‘ -- nonnegative integers
11     - ‘‘positions_list‘‘ -- a tuple of tuples of all distinct
           column indices
12 
13     EXAMPLES::
14         sage: diagonal_antisort([1,2,3,4], 2, 2, ((0,1),))
15         ((2, 1, 4, 3), -1)
16         sage: diagonal_antisort([1,2,4,3], 2, 2, ((0,1),))
17         ((2, 1, 3, 4), -1)
18         sage: diagonal_antisort([2,1,4,3], 2, 2, ((0,1),))
19         ((2, 1, 4, 3), 1)
20     """
21     cdef int sign = 1
```

```
22      cdef int i, j
23      cdef tuple positions
24      cdef list _exponents = list(exponents)
25      for positions in positions_list:
26          for i in range(1, len(positions)):
27              for j in range(i-1, -1, -1):
28                  c = utilities.diagonal_cmp(_exponents, n, r,
                        positions[j], positions[j+1])
29                  if not c:
30                      return None
31                  if c < 0:
32                      utilities.diagonal_swap(_exponents, n, r,
                            positions[j], positions[j+1])
33                      sign = -sign
34                  else:
35                      continue
36      return ETuple(_exponents), sign
37
38  cpdef is_diagonal_antisorted(exponents, int n, int r, tuple
        positions_list):
39      """
40      Return True if the columns are decreasingly sorted according
            to positions.
41
42      EXAMPLES::
43          sage: is_diagonal_antisorted([2,1], 2, 1, ((0,1),))
44          True
45          sage: is_diagonal_antisorted([1,2], 2, 1, ((0,1),))
46          False
47      """
48      cdef int i, j
49      cdef tuple positions
50      cdef list _exponents = list(exponents)
51      for positions in positions_list:
52          for i in range(1, len(positions)):
53              for j in range(i-1, -1, -1):
54                  c = utilities.diagonal_cmp(_exponents, n, r,
                        positions[j], positions[j+1])
55                  if c < 0:
56                      return False
57                  else:
58                      continue
59      return True
60
61  def antisymmetric_normal(p, int n, int r, tuple positions):
62      """
63      Return the `I` antisymmetric normal form of `b_I(p)`.
64
65      INPUT:
```

```
66
67      - `p` -- a polynomial in `r` sets of `n` variables
68      - `r`, `n` -- nonnegative integers
69      - `positions` -- a tuple of tuples of all distinct column
            indices `(I_i)_i`
70
71      A polynomial `b_I(p)` in `P_I` can be uniquely written in the
            form `b_I(q)`, where the monomials of `q` have exponent
            vectors whose columns are decreasingly sorted at positions
            `i,i+1` for `i \in I`.
72      We call `q` the `I` *antisymmetric normal form* of (the
            antisymmetrized polynomial) `b_I(p)`.
73
74      EXAMPLES::
75          sage: load("diagonal_polynomial_ring.py")
76          sage: R = DiagonalPolynomialRing(QQ, 4, 2)
77          sage: X = R.algebra_generators()
78          sage: p = 2 * X[0,0]*X[0,3]^2*X[1,1]*X[1,0]^3 + X[1,3] +
                3
79          sage: antisymmetric_normal(p, 4, 2, ((0,1,2,3),))
80          -2*x00^2*x01*x11^3*x12
81      """
82      cdef Parent R = p.parent()
83      cdef dict d = {}
84      cdef ETuple exponent
85      cdef Element c
86      for exponent, c in utilities.items_of_vector(p):
87          res = diagonal_antisort(exponent, n, r, positions)
88          if res:
89              exponent, sign = res
90              d.setdefault(exponent, 0)
91              d[exponent] += sign*c
92      return R(d)
93
94  def reduce_antisymmetric_normal(p, int n, int r, tuple positions)
        :
95      """
96      Return the terms of `p` which are diagonally antisorted.
97
98      INPUT:
99      - ``p`` -- a polynomial
100     - ``r``, ``n`` -- nonnegative integers
101     - ``positions`` -- a tuple of tuple of positions
102
103     This coincides with antisymmetric_normal (and is faster) if `
            p` is antisymmetric.
104
105     EXAMPLES::
106         sage: load("diagonal_polynomial_ring.py")
```

```
107         sage: R = DiagonalPolynomialRing(QQ, 4, 2)
108         sage: x = R.algebra_generators()
109         sage: p = -2*x[0,0]^2*x[0,1]*x[1,1]^3*x[1,2]-2*x[0,0]^2*x
                [0,1]*x[1,1]^3*x[1,2]
110         sage: reduce_antisymmetric_normal(p,4,1,((0,1,2,3),))
111         -4*x00^2*x01*x11^3*x12
112     """
113     X = p.parent().gens()
114     res = 0
115     for exposent, c in utilities.items_of_vector(p) :
116         if is_diagonal_antisorted(exposent,n,r,positions) :
117             product = 1
118             for (x,a) in zip(X,exposent) :
119                 product = product*(x**a)
120             res += c*product
121     return res
122
123 def antisymmetries_of_tableau(Q):
124     if not isinstance(Q,StandardTableau) :
125         Q = Partition(Q).initial_tableau()
126     return tuple(tuple(i-1 for i in column) for column in Q.
           conjugate())
127
128 def row_permutation(n, sigma):
129     """
130     Return the permutation of the variables induced by a
           permutation of the rows
131
132     EXAMPLES::
133         sage: s = PermutationGroupElement([(1,2,4),(3,5)])
134         sage: row_permutation(3,s)
135         (1,4,10)(2,5,11)(3,6,12)(7,13)(8,14)(9,15)
136     """
137     return PermutationGroupElement([tuple((i-1)*n + 1 + j for i
           in c)
138                                     for c in sigma.cycle_tuples()
139                                     for j in range(n) ])
```

Listing A.3 Fichier matrix_of_vectors

```
1 # Author : Nicolas Thiery
2
3 class MatrixOfVectors:
4     """
5     A mutable data structure representing a collection of vectors
          as a matrix
6     """
7     def __init__(self, vectors=None, ambient=None, stats={}):
8         if vectors is None and not isinstance(ambient, Parent):
9             vectors = ambient
```

```
10              ambient = None
11          if ambient is None:
12              if vectors is not None:
13                  ambient = vectors[0].parent()
14          self._ambient = ambient
15          self._base_ring = ambient.base_ring()
16          self._rank, self._unrank = on_fly()
17          self._matrix = matrix(self._base_ring, 0, 0)
18          self._basis = []
19          self._words = []
20          self._is_echelon = True
21          stats.setdefault("add_vector", 0)
22          stats.setdefault("extend", 0)
23          stats.setdefault("zero", 0)
24          stats.setdefault("dimension", 0)
25          self._stats = stats
26          if vectors:
27              for v in vectors:
28                  self.add_vector(v)
29
30      def __repr__(self):
31          m = self._matrix
32          return "A %sx%s matrix of vectors in %s"%(m.nrows(), m.
               ncols(), self.ambient())
33
34      def ambient(self):
35          return self._ambient
36
37      def plain_vector(self, v):
38          """
39          Return `v` as a plain vector
40          """
41          if not self._ambient.is_parent_of(v):
42              raise ValueError("Expected vector in %s; got %s"%(
                   self._ambient, v))
43          rank = self._rank
44          d = dict((rank(i), c) for i, c in items_of_vector(v))
45          return vector(self._base_ring, len(self._rank.cache), d,
               sparse=False)
46
47      def vector(self, v):
48          R = self.ambient()
49          unrank = self._unrank
50          return R.sum(v[i]* R.monomial(*unrank(i)) for i in range(
               len(v)))
51
52      def vectors(self):
53          return tuple(self.vector(v) for v in self._matrix)
54
```

```
55      def _add_vector_to_matrix(self, m, v):
56          r = self.plain_vector(v)
57          if len(r) > m.ncols():
58              m = m.augment(matrix(self._base_ring, m.nrows(), len(
                    r)-m.ncols()))
59          return m.stack(r)
60
61      def add_vector(self, v):
62          """
63          Add `v` at the bottom of ``self``
64          """
65          self._stats["add_vector"] += 1
66          self._is_echelon = False
67          self._matrix = self._add_vector_to_matrix(self._matrix, v
                )
68
69      def cardinality(self):
70          return self._matrix.nrows()
71
72  class EchelonMatrixOfVectors(MatrixOfVectors):
73      """
74      A mutable data structure representing a collection of vectors
            in row echelon form
75      """
76
77      def __repr__(self):
78          m = self._matrix
79          return "A %sx%s echelon matrix of vectors in %s"%(m.nrows
                (), m.ncols(), self.ambient())
80
81      def extend(self, v, word=None):
82          self._stats["extend"] += 1
83          if not v:
84              self._stats["zero"] += 1
85              return False
86          assert self._is_echelon
87          m = self._add_vector_to_matrix(self._matrix, v)
88          m.echelonize()
89          if m[-1]:
90              self._stats['dimension'] += 1
91              self._matrix = m
92              self._basis.append(v)
93              if word is not None:
94                  self._words.append(word)
95              return True
96          return False
97
98  def annihilator_basis(B, S, action=operator.mul, side='right',
        ambient=None):
```

```
99      """
100     A generalization of :meth:‘Modules.FiniteDimensional.
           WithBasis.ParentMethods.annihilator_basis‘
101     Return a basis of the annihilator of a finite set of elements
            in the span of ‘‘B‘‘
102     See :meth:‘annihilator‘ for the assumptions and definition of
            the annihilator.
103     """
104     if side == 'right':
105         action_left = action
106         action = lambda b,s: action_left(s, b)
107     B = list(B)
108     S = list(S)
109     if ambient is None:
110         ambient = action(S[0], B[0]).parent()
111     mat = matrix(ambient.base_ring(), len(B), 0)
112     for s in S:
113         mat = mat.augment(
114             MatrixOfVectors([action(s, b) for b in B], ambient=
                   ambient)._matrix)
115     return tuple(sum(c * B[i] for i,c in v.iteritems())
116                  for v in mat.left_kernel().basis())
```

Listing A.4 Fichier subspace.py

```
1  # Author : Nicolas Thiery
2  # Contributor : Pauline Hubert
3
4  class Subspace(object):
5      """
6      Construct a subspace from generators and linear operators
7
8      INPUT:
9      - ‘‘generators‘‘ -- a list of vectors in some ambient vector
           space ‘V‘
10     - ‘‘operators‘‘ -- a list of linear endomorphism ‘V‘ (default
           : ‘‘[]‘‘)
11
12     EXAMPLES::
13     Computing a subspace of a multivariate polynomial ring::
14         sage: P = QQ['x,y,z']
15         sage: x,y,z = P.gens()
16         sage: F = Subspace([x-y, y-z, x-z])
17         sage: F.dimension()
18         2
19         sage: F.matrix()
20         [ 1  0 -1]
21         [ 0  1 -1]
22
23     The derivatives of the Van-der-Monde determinant in ‘n‘
```

```
        variables
    spans a space of dimension `n!`::
        sage: Delta = (x-y)*(y-z)*(x-z)
        sage: F = Subspace([Delta], [attrcall("derivative", x)
            for x in P.gens()])
        sage: F.dimension()
        6
    """
    def __init__(self, generators, operators={},
                 add_degrees=operator.add,
                 extend_word=ConstantFunction([]),
                 hilbert_parent=None,
                 degree=None,
                 ambient=None,
                 verbose=False):
        self._stats={}
        self._verbose=verbose

        if self._verbose is not False:
            import tqdm
            if isinstance(self._verbose, tqdm.tqdm):
                self._bar = self._verbose
            else:
                self._bar = tqdm.tqdm(leave=True, unit="
                    extensions")
            if isinstance(self._verbose, str):
                self._bar.set_description(self._verbose)

        self._degree = degree
        if not isinstance(generators, dict):
            if self._degree is None:
                generators = {0: generators}
            else:
                gens = dict()
                for g in generators:
                    d = self._degree(g)
                    gens.setdefault(d, [])
                    gens[d].append(g)
                generators = gens

        self._generators = generators

        if ambient is not None:
            assert all( ambient.is_parent_of(g) for gens in
                generators.values() for g in gens )
        else:
            ambient = {g.parent() for gens in generators.values()
                for g in gens}
            assert len(ambient) == 1
```

```
68              ambient = ambient.pop()
69          self._ambient = ambient
70          self._base_ring = ambient.base_ring()
71
72          if hilbert_parent is None:
73              if generators.keys()[0] in NN:
74                  hilbert_parent = QQ['q']
75          self._hilbert_parent = hilbert_parent
76
77          if not isinstance(operators, dict):
78              operators = {0: operators}
79          self._operators = operators
80
81          self._bases = {}
82          self._todo = []
83          self._add_degrees = add_degrees
84          self._extend_word = extend_word
85          for d, gens in generators.iteritems():
86              basis = EchelonMatrixOfVectors(ambient=self._ambient,
                        stats=self._stats)
87              for g in gens:
88                  if basis.extend(g):
89                      self.todo(g, d, [])
90              self._bases[d] = basis
91
92      def todo(self, vector, d1, word):
93          todo = self._todo
94          for d2, ops in self._operators.iteritems():
95              try:
96                  d3 = self._add_degrees(d1, d2)
97              except ValueError:
98                  continue
99              for op in ops:
100                 new_word = self._extend_word(word, op)
101                 if new_word is not None:
102                     todo.append((vector, op, d3, new_word))
103
104     def dimension(self):
105         self.finalize()
106         return sum(basis.cardinality() for basis in self._bases.
                values())
107
108     def basis(self):
109         self.finalize()
110         basis = {}
111         for i, val in self._bases.iteritems() :
112             if val.vectors() != () :
113                 basis[i] = val.vectors()
114         return basis
```

```
115
116     def hilbert_polynomial(self):
117         return self._hilbert_parent(self.dimensions())
118
119     def dimensions(self):
120         self.finalize()
121         return {d: basis.cardinality() for d, basis in self.
                _bases.iteritems()}
122
123     def matrix(self):
124         self.finalize()
125         assert self._bases.keys() == [0]
126         return self._bases[0]._matrix
127
128     def extend(self, v, d=None, word=None):
129         if d is None and self._degree is not None:
130             d = self._degree(v)
131         if d not in self._bases:
132             self._bases[d] = EchelonMatrixOfVectors(ambient=self.
                    _ambient, stats=self._stats)
133         if self._bases[d].extend(v):
134             self.todo(v, d, word)
135         if self._verbose is not False:
136             self._bar.update()
137             self._bar.set_postfix({'todo': len(self._todo), '
                    dimension': self._stats['dimension'],  'zero':
                    self._stats['zero']})
138
139     @cached_method
140     def finalize(self):
141         todo = self._todo
142         if not todo:
143             return
144         while todo:
145             v,op,d,word = todo.pop()
146             w = op(v)
147             if not isinstance(w, (list, tuple)):
148                 w = [w]
149             for w2 in w:
150                 self.extend(w2, d, word)
151         if self._verbose is not False:
152             self._bar.set_postfix({'dimension': self._stats['
                    dimension'], 'zero': self._stats['zero']})
153             self._bar.close()
```

Listing A.5 Fichier add_degree.py

```
1 Authors : Pauline Hubert, Nicolas Thiery
2
3 def add_degree(d1,d2):
```

```
4       """
5       Compute the sum componentwise of d1 and d2 and return a
           grading set with no negative component as result.
6
7       INPUT: ``d1``,``d2`` -- lists of integers
8
9       EXAMPLES::
10          sage: D = cartesian_product([ZZ for i in range(3)])
11          sage: add_degree(D([3,2,1]), D([-2,0,0]))
12          (1, 2, 1)
13          sage: add_degree(D([3,2,1]), D([-2,1,4]))
14          (1, 3, 5)
15      """
16      d = d1 + d2
17      if not all(i>=0 for i in d):
18          raise ValueError("invalid degree")
19      return d
20
21  def add_degrees_isotypic(gen_deg, op_deg):
22      """
23      Compute the sum componentwise of the lists of integrers
           contained in d1 and d2  and return a grading set and the
           partition contained in d2 as result.
24
25      EXAMPLES::
26          sage: add_degrees_isotypic(([3,0],[2,1]), [-1,0])
27          ((2, 0), [2, 1])
28          sage: add_degrees_isotypic(([2,0,1],[3,2]), [-1,0,0])
29          ((1, 0, 1), [3, 2])
30      """
31      D = cartesian_product([ZZ for i in range(len(gen_deg[0]))])
32      d = D(gen_deg[0])+D(op_deg)
33      return d, gen_deg[1]
34
35  def add_degrees_symmetric(gen_deg, op_deg):
36      """
37      Compute the sum componentwise of the lists of integrers
           contained in d1 and d2
38      and return an ordered grading set and the partition contained
            in d2 as result.
39
40      EXAMPLES::
41          sage: add_degrees_symmetric(([2,0,1],[3,2]), [-1,0,0])
42          ((1, 1, 0), [3, 2])
43      """
44      D = cartesian_product([ZZ for i in range(len(gen_deg[0]))])
45      d1, d2 = add_degrees_isotypic(gen_deg, op_deg)
46      return D(sorted(d1, reverse=True)), d2
```

Listing A.6 Fichier isomorphic_object.py

```
# Authors : Pauline Hubert, Nicolas Thiery

# Stuff here everything currently missing in SageMath about
    isomorphic algebras
class Algebras_IsomorphicObjects(IsomorphicObjectsCategory):
    class ParentMethods:
        def summation(self, x, y):
            return self.retract(x.lift() + y.lift())
        def gens(self):
            return [self.retract(g) for g in self.ambient().gens
                ()]
        def is_field(self):
            return self._ambient.is_field()
        def is_integral_domain(self):
            return self._ambient.is_integral_domain()
        def is_commutative(self):
            return self._ambient.is_commutative()

    class ElementMethods:
        def _sub_(x, y):
            return x.parent().retract(x.lift() - y.lift())
        def _neg_(self):
            return self.parent().retract(-self.lift())
        def _div_(x, y):
            return x.parent().retract(x.lift()/y.lift())

Algebras.IsomorphicObjects = Algebras_IsomorphicObjects

class IsomorphicObject(UniqueRepresentation, Parent):
    def ambient(self):
        return self._ambient
    def lift(self, p):
        return p.value
    def retract(self, p):
        return self.element_class(self,p)
    def base_ring(self):
        return self.ambient().base_ring()
    def __init__(self, ambient, category):
        self._ambient = ambient
        Parent.__init__(self, category=category.IsomorphicObjects
            ())
        # Bricolage
        self._remove_from_coerce_cache(ambient)
        phi = SetMorphism(Hom(ambient, self, category), self.
            retract)
        phi.register_as_coercion()

    class Element(ElementWrapper):
```

```
45          pass
```

Listing A.7 Fichier diagonal_polynomial_ring.py

```
# Authors : Pauline Hubert, Nicolas Thiery

#### Polynomial ring with diagonal action

class DiagonalPolynomialRing(IsomorphicObject):
    """
    The ring of diagonal polynomials in $n \times r$ variables
       and $n \times k$ inert variables.
    In order to distinguish the inert variables among the others,
        they are named `theta_{i,j}`.
    """
    def __init__(self, R, n, r, inert=0):
        names = ["x%s%s"%(i,j) for i in range(r) for j in range(n
           )]+["theta%s%s"%(i,j) for i in range(inert) for j in
           range(n)]
        P = PolynomialRing(R, n*(r+inert), names)
        IsomorphicObject.__init__(self, P, Algebras(R))
        self._n = n
        self._r = r
        self._inert = inert
        self._P = P
        self._R = R
        self._grading_set = cartesian_product([ZZ for i in range(
           r)])
        self._hilbert_parent = PolynomialRing(ZZ, r, 'q')

    def _element_constructor_(self, data):
        if isinstance(data, DiagonalPolynomialRing.Element):
            new_data = self._P(data.lift())
        else:
            new_data = self._P(data)
        return self.element_class(self, new_data)

    def variable_names(self):
        # Return the variable names. This is use when calling
           function inject_variables() on self.

    def nrows(self):
        # Return the number of sets of classic variables of self.

    def ncols(self):
        # Return the number of variables in each set of self.

    def ninert(self):
        # Return the number of sets if inert variables of self.

```

```
41      def grading_set(self):
42          # Return the grading set of self.
43
44      def algebra_generators(self):
45          # Return all the variables of self including the inert
               variables.
46
47      def variables(self):
48          # Return only the classic variables.
49
50      def inert_variables(self):
51          # Return only the inert variables of self.
52
53      def multivar_pol_ring_variables(self):
54          """
55          Return only the classic variables as variables of a
               multivariate polynomial ring.
56          """
57          vars = self._P.gens()
58          n = self.ncols()
59          r = self.nrows()
60          inert = self.ninert()
61          return matrix([[vars[i*n+j] for j in range(n)] for i in
               range(r)])
62
63      def multipower(self, d):
64          # Return the product of the terms $q_i^{d_i}$ for all
               $d_i \in d$ .
65
66      def monomial(self, *args):
67          # Return the monomial with given exponents.
68
69      def random_monomial(self, D):
70          """
71          Return a random monomial of multidegree `D`.
72          """
73          X = self.algebra_generators()
74          X_by_rows = [Set(list(X[i,j] for j in range(X.ncols())))
               for i in range(X.nrows())]
75          return prod( X_by_rows[i].random_element() for i in range
               (len(D)) for j in range(D[i]) )
76
77      def random_element(self, D, l=10):
78          """
79          Return a "random" multi homogeneous polynomial of
               multidegree `D`.
80          """
81          K = self.base_ring()
82          return sum(K.random_element() * self.random_monomial(D)
```

```
83                  for i in range(l))
84
85      @cached_method
86      def row_permutation(self, sigma):
87          """
88          Return the permutation of the variables induced by a
               permutation of the rows.
89
90          EXAMPLES::
91              sage: s = PermutationGroupElement([(1,2,4),(3,5)])
92              sage: P = DiagonalPolynomialRing(QQ,3,5)
93              sage: P.row_permutation(s)
94              (1,4,10)(2,5,11)(3,6,12)(7,13)(8,14)(9,15)
95          """
96          n = self._n
97          return PermutationGroupElement([tuple((i-1)*n + 1 + j for
                i in c) for c in sigma.cycle_tuples() for j in range(
               n) ])
98
99      @cached_method
100     def derivative_input(self, D, j):
101         r = self._r
102         X = self.multivar_pol_ring_variables()
103         res = []
104         for i in range(r):
105             res.extend([X[i,j],D[i]])
106         return res
107
108     class Element(IsomorphicObject.Element):
109         def multidegree(self):
110             """
111             Return the multidegree of a multihomogeneous
                   polynomial.
112             """
113             if not self:
114                 return -1
115             n = self.parent().ncols()
116             r = self.parent().nrows()
117             v = self.parent()._P(self).exponents()[0]
118             return self.parent().grading_set()([sum(v[n*i+j] for
                   j in range(n)) for i in range(r)])
119
120         def degree(self):
121             """
122             Return the total degree of a multihomogeneous
                   polynomial.
123             """
124             return sum(self.multidegree())
125
```

```
126         def subs(self, *args):
127             """
128             Substitute some variables of the polynomial by a
                    specified value or variable.
129             """
130             p = self.lift()
131             return self.parent(p.subs(*args))
132 
133         def factor(self, *args):
134             """
135             Return the factorization of the polynomial self.
136             See method func:‘factor‘ of multivariate polynomial
                    ring.
137             """
138             p = self.lift()
139             return p.factor(*args)
140 
141         def derivative(self, *args):
142             """
143             Return the derivative of self w.r.t the given
                    arguments.
144             """
145             p = self.parent()._P(self)
146             return self.parent()(p.derivative(*args))
147 
148         @cached_method
149         def apply_permutation(self):
150             """
151             Strainghten the polynomial self by applying the right
                     permutation
152             to obtain the equivalent polynomial of decreasing
                    multidegree.
153 
154             EXAMPLES::
155                 sage: P = DiagonalPolynomialRing(QQ, 3, 3, inert
                        =1)
156                 sage: P.inject_variables()
157                 Defining x00, x01, x02, x10, x11, x12, x20, x21,
                        x22, theta00, theta01, theta02
158                 sage: p = x00*x11^3*x22^2 + x01*x12^3+x20^2
159                 sage: p.apply_permutation()
160                 x01^3*x12^2*x20 + x02^3*x21 + x10^2
161             """
162             d = self.multidegree()
163             d = tuple(d) + tuple(0 for i in range(self.parent().
                    ninert()))
164             result = self.lift()
165             if list(d) != sorted(d, reverse=True):
166                 s = reverse_sorting_permutation(d)
```

```
167                 ss = self.parent().row_permutation(s)
168                 result = act_on_polynomial(result, ss)
169             return self.parent()(result)
170 
171         def polarization(self, i1, i2, d, row_symmetry=None):
172             """
173             Return the polarization `P_{d,i_1,i_2}. p` of `p`.
174 
175             Recall that the polarization operator is defined by
176             .. MATH:: P_{d,i_1,i_2} := \sum_j x_{i_2,j} \partial_
                    {i_1,j}^d
177             """
178             n = self.parent().ncols()
179             r = self.parent().nrows()
180             X = self.parent().multivar_pol_ring_variables()
181             if i1>=r or i2 >=r:
182                 print "Row number out of range"
183                 return None
184             else:
185                 result = sum(X[i2,j]*self.derivative(X[i1,j],d)
                        for j in range(n))
186                 if row_symmetry=="permutation" and result:
187                     result = result.apply_permutation()
188                 return result
189 
190         def symmetric_derivative(self, d, row_symmetry=None):
191             """
192             Return the symmetric derivative of p w.r.t the
                    degrees d.
193             """
194             n = self.parent().ncols()
195             r = self.parent().nrows()
196             X = self.parent().multivar_pol_ring_variables()
197             result = 0
198             if not isinstance(d, (tuple, list)):
199                 d = [d]
200             for i in range(n): #columns
201                 interm_result = self
202                 for j in range(len(d)): #rows
203                     interm_result = interm_result.derivative(X[j,
                            i],d[j])
204                 result += interm_result
205 
206             if row_symmetry=="permutation" and result:
207                 result = result.apply_permutation()
208             return result
209 
210         def steenrod_op(self, i, k, row_symmetry=None):
211             """
```

```
212             Apply the Steenrod operator of degree `k` for the `i`
                    th set of variables to `p`.
213             """
214             n = self.parent().ncols()
215             X = self.parent().multivar_pol_ring_variables()
216             result = sum(X[i,j]*self.derivative(X[i,j], k) for j
                    in range(0, n))
217             if row_symmetry=="permutation" and result:
218                 result = result.apply_permutation()
219             return result
220
221 #### Polynomial ring with diagonal action and antisymmetries
222
223 class DiagonalAntisymmetricPolynomialRing(DiagonalPolynomialRing)
        :
224     """
225     The ring of diagonal antisymmetric polynomials in $n \times
            r$ variables
226     and $n \times k$ inert variables.
227     """
228     def __init__(self, R, n, r, inert=0, antisymmetries=None):
229         DiagonalPolynomialRing.__init__(self, R, n, r, inert=
                inert)
230         if isinstance(antisymmetries, Partition):
231             self._antisymmetries = antisymmetries_of_tableau(
                    antisymmetries.initial_tableau())
232         else:
233             self._antisymmetries = antisymmetries
234
235     def antisymmetries(self):
236         # Return the antisymmetries of `self`.
237
238     def set_antisymmetries(self, antisymmetries):
239         # Set the antisymmetries of self to be the given ones.
240
241     class Element(DiagonalPolynomialRing.Element):
242         def antisymmetries(self):
243             # Return the antisymmetries of `self`.
244
245         def set_antisymmetries(self, antisymmetries):
246             # Set the antisymmetries of the parent of self to be
                    the given ones.
247
248         def polarization(self, i1, i2, d, row_symmetry=None):
249             """
250             Return the polarization `P_{d,i_1,i_2}` of `self`.
251             The result is reduced with respect to the ring
                    antisymmetries.
252
```

```
253             EXAMPLES::
254                 sage: mu = Partition([2,1])
255                 sage: antisymmetries = antisymmetries_of_tableau(
                        mu.initial_tableau())
256                 sage: antisymmetries = tuple(tuple(a) for a in
                        antisymmetries)
257                 sage: P = DiagonalAntisymmetricPolynomialRing(QQ,
                         4, 3, antisymmetries = antisymmetries)
258                 sage: x = P.algebra_generators()
259                 sage: v = -x[0,0]^2*x[0,1] + x[0,0]*x[0,1]^2 + x
                        [0,0]^2*x[0,2] - x[0,1]^2*x[0,2] - x[0,0]*x
                        [0,2]^2 + x[0,1]*x[0,2]^2
260                 sage: v.polarization(0, 1, 1)
261                 -4*x00*x01*x10 + 2*x01^2*x10 + 4*x00*x02*x10 - 2*
                        x00^2*x11 + 4*x00*x01*x11 + 2*x00^2*x12
262             """
263             antisymmetries = self.parent().antisymmetries()
264             n = self.parent().ncols()
265             r = self.parent().nrows()
266             inert = self.parent().ninert()
267             result = super(DiagonalAntisymmetricPolynomialRing.
                    Element, self).polarization(i1, i2, d,
                    row_symmetry=row_symmetry)
268             if antisymmetries and result:
269                 result = antisymmetric_normal(self.parent()._P(
                        result), n, r+inert, antisymmetries)
270             return self.parent()(result)
271
272         def steenrod_op(self, i, k, row_symmetry=None):
273             """
274             Apply the Steenrod operator `St_i^k` to `self`.
275             The result is reduced with respect to the ring
                    antisymmetries.
276             """
277             antisymmetries = self.parent().antisymmetries()
278             n = self.parent().ncols()
279             r = self.parent().nrows()
280             inert = self.parent().ninert()
281             result = super(DiagonalAntisymmetricPolynomialRing.
                    Element, self).steenrod_op(i, k, row_symmetry=
                    row_symmetry)
282             if antisymmetries and result:
283                 result = antisymmetric_normal(result, n, r+inert,
                         antisymmetries)
284             return result
```

Listing A.8 Fichier young_idempotent.py

```
1 # Authors : Pauline Hubert, Nicolas Thiery
2
```

```
#### Young idempotent and related functions

def apply_young_idempotent(p, t):
    """
    Apply the Young idempotent indexed by `t` on the polynomial `p`

    INPUT::
    - `t` -- a standard tableau or a partition
    - `p` -- a polynomial on as many variables as there are cells in `t`

    EXAMPLES::
        sage: x,y,z = QQ['x,y,z'].gens()
        sage: p = x^2 * y
        sage: t = StandardTableau([[1],[2],[3]])
        sage: apply_young_idempotent(p, t)
        x^2*y - x*y^2 - x^2*z + y^2*z + x*z^2 - y*z^2
        sage: apply_young_idempotent(p, Partition([1,1,1]))
        x^2*y - x*y^2 - x^2*z + y^2*z + x*z^2 - y*z^2
        sage: t = StandardTableau([[1,2],[3]])
        sage: p = x*y*y^2
        sage: apply_young_idempotent(p, t)
        x^3*y + x*y^3 - y^3*z - y*z^3
    """
    if isinstance(t, Partition):
        t = t.initial_tableau()
    res = sum(act(Permutation(sigma),p) for sigma in t.row_stabilizer())
    if isinstance(p.parent(), DiagonalAntisymmetricPolynomialRing):
        antisymmetries = antisymmetries_of_tableau(t)
        P = p.parent()
        D = DiagonalAntisymmetricPolynomialRing(P._R,  P.ncols(), P.nrows(), P.ninert(), antisymmetries = antisymmetries)
        res = res.lift()
        res = antisymmetric_normal(res, t.size(), 2, antisymmetries)
        res = D(res)
    else:
        res = sum(sigma.sign()*act(Permutation(sigma),res) for sigma in t.column_stabilizer())
    return res

@cached_function
def act(sigma, v) :
    """
    Compute the action of the permutation sigma on the element v.
```

```
44
45      EXAMPLES::
46          sage: P = PolynomialRing(QQ,5,'x')
47          sage: X = P.gens()
48          sage: X
49          (x0, x1, x2, x3, x4)
50          sage: v = X[0]*X[1]+X[2]^2-X[4]
51          sage: v
52          x0*x1 + x2^2 - x4
53          sage: sigma = (2,1,3,4,5)
54          sage: act(sigma,v)
55          x0*x1 + x2^2 - x4
56      """
57      if isinstance(v, DiagonalPolynomialRing.Element):
58          X = v.parent()._P.gens()
59          r = v.parent().nrows() + v.parent().ninert()
60          n = v.parent().ncols()
61      else:
62          X = v.parent().gens()
63          r = 1
64          n = len(X)
65      sub = {}
66      for j in range(0,r) :
67          sub.update({X[j*n+i]:X[j*n+sigma[i]-1] for i in range (0,
               n) if i!=sigma[i]-1})
68      return v.subs(sub)
```

Listing A.9 Fichier diagram.py

```
1  # Author : Pauline Hubert
2
3  class Diagram():
4      """
5      A list of cells in the grid $\mathbb{N} \times \mathbb{N}$ :
6      $[(a_1, b_1), (a_2, b_2), ..., (a_n, b_n)]$.
7
8      EXAMPLES ::
9          sage: d = Diagram([(0,0),(3,0),(6,0)])
10         sage: d = Diagram([(0,0),(1,0),(2,0),(3,0),(0,1),(2,1)])
11     """
12
13     def __init__(self, cells):
14         c = True
15         for cell in cells:
16             if len(cell) != 2:
17                 c = False
18         self._cells = []
19         if c:
20             for cell in cells:
21                 self._cells += [(cell[1],cell[0])]
```

```
22                 self._cells.sort()
23         else:
24             print("Wrong format for class Diagram")
25
26     def size(self) :
27         """
28         The size of the diagram, which is the number of cells.
29
30         EXAMPLES:
31             sage: Diagram([(0,0),(1,0),(2,0),(3,0),(0,1),(2,1)]).
                   size()
32             sage: d.size()
33             6
34         """
35         return Integer(len(self._cells))
36
37     def cells(self):
38         """
39         Gives the list of the matrix coordinates of the cells of
               the diagram.
40
41         EXAMPLES ::
42             sage: d = Diagram([(0,0),(3,0),(6,0)])
43             sage: d.cells()
44             [(0, 0), (0, 3), (0, 6)]
45         """
46         return self._cells
47
48     def nb_cols(self):
49         """
50         Gives the number of columns of the diagram including the
               empty ones that lie between unempty ones.
51         """
52         return max([c[1] for c in self.cells()])
```

Listing A.10 Fichier character.py

```
1 # Author: Pauline Hubert
2
3 ### Vandermonde like determinant
4
5 def vandermonde(gamma, r=0):
6     """
7     It returns the determinant of the matrix $(x_i^a\theta_i^b)$
          for $1 \leq i \leq n$ and $(a,b) the cells of ‘mu‘.
8
9     EXAMPLES::
10        sage: gamma = Diagram([(0,0),(1,0),(3,0)])
11        sage: vandermonde(gamma)
12        -x00^3*x01 + x00*x01^3 + x00^3*x02 - x01^3*x02 - x00*x02
```

```
            ^3 + x01*x02^3
13          sage: v = vandermonde(Partition([2,1]), r=2)
14          sage: v
15          -x01*theta00 + x02*theta00 + x00*theta01 - x02*theta01 -
               x00*theta02 + x01*theta02
16      """
17      if isinstance(gamma, Integer):
18          gamma = Partition([gamma])
19      n = gamma.size()
20      if r == 0:
21          r = 1
22      P = DiagonalPolynomialRing(QQ, n, r, inert=1)
23      X = P.variables()
24      Theta = P.inert_variables()
25      return matrix([[x**i[1]*theta**i[0] for i in gamma.cells()]
           for x,theta in zip(X[0],Theta[0])]).determinant()
26
27  ### Operators
28
29  def partial_derivatives(P):
30      """
31      Return the partial derivative functions in all the variables
           of ‘P‘ allowed for derivation (ie not in the inert
           variables).
32      """
33      n = P.ncols()
34      r = P.nrows()
35      D = P.grading_set()
36      X = P.multivar_pol_ring_variables()
37      op = {}
38      for i in range(r):
39          op[D((-1 if j==i else 0 for j in range(r)))] = [attrcall(
               "derivative", X[i,k]) for k in range(n)]
40      return op
41
42  def polarization_operators(r, max_deg=1, side=None, row_symmetry=
        None):
43      """
44      Return the polarization operator functions in ‘r‘ sets of
           variables with maximum degree ‘max_deg‘.
45      """
46      D = cartesian_product([ZZ for i in range(r)])
47      return {D([-d if i==i1 else 1 if i==i2 else 0 for i in range(
           r)]):
48              [attrcall("polarization", i1=i1, i2=i2, d=d,
                   row_symmetry=row_symmetry)]
49              for d in range(1, max_deg+1)
50              for i1 in range(0, r)
51              for i2 in range(0, r)
```

```
52              if (i1<i2 if side == 'down' else i1!=i2)
53             }
54
55 def steenrod_operators(r, degree=1, row_symmetry=None):
56     """
57     Return the Steenrod operator functions in `r` sets of
          variables of  degree `degree`.
58     """
59     if degree < 1:
60         degree = 1
61     D = cartesian_product([ZZ for i in range(r)])
62     op = {}
63     for i in range(r):
64         op[D((-degree if j==i else 0 for j in range(r)))] = [
65             attrcall("steenrod_op", i=i, k=degree+1, row_symmetry
                =row_symmetry)]
66     return op
67
68 def merge(dict1, dict2):
69     """
70     Merge two dictionnaries whose values are lists into the first
          one.
71     """
72
73 ### Projection on isotypic components
74
75 def IsotypicComponent(S, arg, use_antisymmetry=False):
76     """
77     Project the basis of the given subspace S on the isotypic
          components of $S_n$
78     or on the isotypic component of the given type.
79
80     INPUT:
81         -``S`` -- Subspace
82         -``arg`` -- an integer or a partition
83
84     EXAMPLES::
85         sage: v = vandermonde(Partition([3]))
86         sage: gen = {v.multidegree() : [v]}
87         sage: deriv = partial_derivatives(v.parent())
88         sage: S = Subspace(gen, deriv)
89         sage: V = IsotypicComponent(S, 3)
90         sage: for value in V.values():
91         ....:     print(value.basis())
92         {((3,), (1, 1, 1)): (-x00^2*x01 + x00*x01^2 + x00^2*x02 -
              x01^2*x02 - x00*x02^2 + x01*x02^2,)}
93         {((0,), (3,)): (1,)}
94         {((2,), (2, 1)): (x00^2 - 2*x00*x01 + 2*x01*x02 - x02^2,)
              , ((1,), (2, 1)): (x00 - x02,)}
```

```
95          sage: V = IsotypicComponent(S, 3, use_antisymmetry=True)
96          sage: for value in V.values():
97          ....:     print(value.basis())
98          {((3,), (1, 1, 1)): (x00^2*x01,)}
99          {((0,), (3,)): (1,)}
100         {((2,), (2, 1)): (x00^2 - 2*x00*x01,), ((1,), (2, 1)): (
                x00,)}
101     """
102     if isinstance(arg, Partition):
103         list_partitions = [arg]
104     elif isinstance(arg, Integer):
105         list_partitions = Partitions(arg)
106     else :
107         print("Error: arg should be a partition or an integer.")
108     basis = S.basis()
109     result = {}
110     P1 = basis.values().pop()[0].parent()
111     for nu in list_partitions:
112         result_nu = {}
113         if use_antisymmetry == True:
114             antisymmetries = antisymmetries_of_tableau(nu.
                    initial_tableau())
115             P2 = DiagonalAntisymmetricPolynomialRing(P1._R, P1.
                    ncols(), P1.nrows(), P1.ninert(), antisymmetries)
116         for deg, value in basis.iteritems():
117             if use_antisymmetry:
118                 gen = []
119                 for p in value:
120                     temp = apply_young_idempotent(P2(p), nu)
121                     if temp != 0:
122                         gen += [temp]
123             else:
124                 gen = []
125                 for p in value:
126                     temp = apply_young_idempotent(p, nu)
127                     if temp != 0:
128                         gen += [temp]
129             if gen != []:
130                 result_nu[(deg, tuple(nu))] = Subspace(gen, {}).
                        basis()[0]
131         if result_nu != {}:
132             result[nu] = Subspace(result_nu, operators={})
133
134     if len(result.keys()) == 1:
135         key = result.keys()[0]
136         return result[key]
137     else:
138         return result
139
```

```
#### Polarization Space

def PolarizedSpace(S, operators, add_degrees=add_degrees_isotypic):
    """
    Polarize the subspace S with the given operators on the polynomial ring P.

    INPUT:
    - ``P`` -- a polynomial ring
    - ``S`` -- a subspace
    - ``operators`` -- a list or a dictionnary of operators acting on elements of P
    - ``add_degrees`` -- a function that will be used to determine the degrees of the elements computed

    EXAMPLES::
        sage: P = PolarizedSpace(V, polarizators)
        sage: for value in P.values():
        ....:     print(value.basis())
        {((2, 1), (1, 1, 1)): (x00*x01*x10 + 1/2*x00^2*x11,), ((1, 1), (1, 1, 1)): (x00*x11,), ((3, 0), (1, 1, 1)): (x00^2*x01,), ((1, 2), (1, 1, 1)): (x00*x10*x11 - 1/2*x00*x11^2,), ((0, 3), (1, 1, 1)): (x10^2*x11,)}
        {((0, 0), (3,)): (1,)}
        {((1, 1), (2, 1)): (x00*x10 - x01*x10 - x00*x11,), ((0, 1), (2, 1)): (x10,), ((1, 0), (2, 1)): (x00,), ((0, 2), (2, 1)): (-1/2*x10^2 + x10*x11,), ((2, 0), (2, 1)): (-1/2*x00^2 + x00*x01,)}
    """
    if isinstance(S, dict):
        return {key : PolarizedSpace(value, operators, add_degrees=add_degrees) for key, value in S.iteritems()}
    else:
        basis = S.basis()
        basis_element = basis.values().pop()[0]
        P1 = basis_element.parent()
        if operators != {}:
            r = len(operators.keys().pop())
            row_symmetry = operators.values().pop()[0].kwds['row_symmetry']
        else:
            r=1
            row_symmetry = None
        if row_symmetry == "permutation":
            add_degrees = add_degrees_symmetric
        D = cartesian_product([ZZ for i in range(r)])
        generators = {}
```

```
176         if isinstance(P1, DiagonalAntisymmetricPolynomialRing):
177             P2 = DiagonalAntisymmetricPolynomialRing(P1._R, P1.
                    ncols(), r , P1.ninert(), P1.antisymmetries())
178             for key, value in basis.iteritems():
179                 d = (D((key[0][0] if i==0 else 0 for i in range
                        (0,r))), key[1])
180                 generators[d] = tuple(reduce_antisymmetric_normal
                        (P2(b), b.parent().ncols(), b.parent().nrows()
                        +b.parent().ninert(), b.antisymmetries()) for
                        b in value)
181         else :
182             P2 = DiagonalPolynomialRing(P1._R, P1.ncols(), r , P1
                    .ninert())
183             for key, value in basis.iteritems():
184                 if isinstance(key[0], Integer):
185                     d = (D((key[0] if i==0 else 0 for i in range
                            (0,r))))
186                     add_degrees = add_degree
187                 else:
188                     d = (D((key[0][0] if i==0 else 0 for i in
                            range(0,r))), key[1])
189                 generators[d] = tuple(P2(b) for b in value)
190         return Subspace(generators, operators, add_degrees=
                add_degrees)
191
192 ### Quotient
193
194 def Range(S, operators, add_degrees=add_degrees_isotypic):
195     """
196     Return the basis of the image of the subspace S by the given
            operators.
197
198     INPUT:
199     - ``S`` -- a subspace
200     - ``operators`` -- a list or a dictionnary of operators
            acting on elements of P
201     - ``add_degrees`` -- a function that will be used to
            determine the degrees of the elements computed
202
203     """
204     if isinstance(S, dict):
205         return {key : Range(value, operators, add_degrees=
                    add_degrees) for key, value in S.iteritems()}
206
207     result = {}
208     basis = S.basis()
209     for key, b in basis.iteritems():
210         result = merge(result, {add_degrees(key, deg): [op(p) for
                    p in b for op in op_list if op(p)!=0] for deg,
```

```
            op_list in operators.iteritems()})
    if result != {} :
        return Subspace(result, operators, add_degrees) #{} <->
            operators
    else :
        return None

#### Character

def character(S, left_basis=s, right_basis=s, row_symmetry=None):
    """
    Return the bicharacter of the subspace ‘S‘ into the given
       bases. The subspace ‘S‘ must be a multivariate polynomial
       subspace projected on isotypic components of ‘S_n‘ or a
       dictionnary of subspaces projected on isotypic components.

    INPUT:
    - ‘‘S‘‘ -- a subspace or a dictionnary of subspaces

    EXAMPLES::
        sage: v = vandermonde(Partition([2,2]))
        sage: V = Subspace({v.multidegree(): [v]},
            partial_derivatives(v.parent()))
        sage: V_iso = IsotypicComponent(V, 4, use_antisymmetry=
            True)
        sage: V_pol = PolarizedSpace(V_iso,
            polarization_operators(2, max_deg = v.degree()))
        sage: character(V_pol)
        s[] # s[2, 2] + s[1] # s[2, 1, 1] + s[2] # s[1, 1, 1, 1]
    """
    if isinstance(S, dict):
        return sum(character(V, left_basis=left_basis,
            right_basis=right_basis, row_symmetry=row_symmetry)
            for V in S.values())
    else:
        basis = S.basis()
        basis_element = basis.values().pop()[0]
        P = basis_element.parent()
        n = P.ncols()
        r = P.nrows()
        charac = 0
        if row_symmetry != "permutation":
            q = PolynomialRing(QQ,'q',r).gens()
        for nu in Partitions(n):
            basis_nu = {}
            charac_nu = 0
            for key, value in basis.iteritems():
                if Partition(key[1]) == nu:
                    basis_nu[key[0]] = value
```

```
250             if row_symmetry == "permutation":
251                 for deg, b in basis_nu.iteritems():
252                     charac_nu += sum(m(Partition(deg)) for p in b
                        )
253                 if charac_nu != 0 :
254                     if left_basis == s :
255                         charac_nu = s(charac_nu).
                            restrict_partition_lengths(r,exact=
                            False)
256                     elif left_basis != m :
257                         charac_nu = left_basis(charac_nu)
258             else:
259                 for deg, b in basis_nu.iteritems():
260                     charac_nu += sum(prod(q[i]**deg[i] for i in
                        range(0,len(deg))) for p in b)
261                 if charac_nu != 0 :
262                     if left_basis == s :
263                         charac_nu = s.from_polynomial(charac_nu).
                            restrict_partition_lengths(r,exact=
                            False)
264                     else:
265                         charac_nu = left_basis.from_polynomial(
                            charac_nu)
266             if charac_nu != 0:
267                 charac += tensor([charac_nu, right_basis(s(nu))])
268         return charac
269
270 @parallel
271 def parallel_character(S, operators, row_symmetry="permutation",
    add_degrees=add_degrees_isotypic):
272     """
273     Return a dictionnary corresponding to the character of the
        polarized version of the subspace S.
274     """
275     V = PolarizedSpace(S, operators, add_degrees=add_degrees)
276     charac = character(V, row_symmetry=row_symmetry)
277     res = {tuple((tuple(support[0]),tuple(support[1]))): coef for
         support, coef in charac}
278     return res
279
280 #### Tools on character
281
282 def factorize(f, n=0):
283     """
284     Return the factorization of the tensor product `f` w.r.t the
        right symmetric functions. The right symmetric functions
        have their supports in the partitions on `n`.
285     """
286
```

```
287 def latex_output_character(f):
288     """
289     Return the latex code of the character ‘f‘.
290     """
291 
292 #### Main function
293 
294 @persist(hash=lambda k: 'character_%s_%s' % (k[0][1].size(),''.
    join(str(i) for i in k[0][1])), funcname='character')
295 def E_mu(mu, use_antisymmetry=True, row_symmetry="permutation",
    parallel=False, r=0):
296     """
297     Given a diagram ‘mu‘, compute the character associated to
        this diagram.
298     Compute the subspace span by the Vandermonde determinant
        associated to ‘mu‘
299     and closed by partial derivatives and polarization, and
        return its bicaracter.
300     If ‘use_antisymmetry‘ is ‘True‘, use the optimisation on
        antisymmetries, and if
301     ‘row_symmetry‘ is "permutation", use the optimisation on row
        permutation.
302 
303     EXAMPLES::
304         sage: E_mu(Partition([2,1,1]))
305         s[] # s[2, 1, 1] + s[1] # s[1, 1, 1, 1]
306         sage: for mu in Partitions(3):
307         ....:     print(E_mu(mu))
308         s[] # s[3] + s[1] # s[2, 1] + s[1, 1] # s[1, 1, 1] + s[2]
              # s[2, 1] + s[3] # s[1, 1, 1]
309         s[] # s[2, 1] + s[1] # s[1, 1, 1]
310         s[] # s[1, 1, 1]
311 
312     """
313     n = Integer(mu.size())
314     v = vandermonde(mu)
315     generator = {v.multidegree() : [v]}
316     list_op = partial_derivatives(v.parent())
317     V = Subspace(generators=generator, operators=list_op,
        add_degrees=add_degree)
318     V_iso = IsotypicComponent(V, n, use_antisymmetry=
        use_antisymmetry)
319     if r == 0 :
320         r = max(n-1, 1)
321     deg = v.degree()
322     if deg == 0:
323         deg = 1
324     op_pol = polarization_operators(r, deg, row_symmetry=
        row_symmetry)
```

```
325     if parallel:
326         charac = 0
327         for (((_,_),_), V_pol) in parallel_character([(subspace,
                op_pol) for subspace in V_iso.values()]):
328             for key, coeff in V_pol.iteritems():
329                 charac += coeff*tensor([s(key[0]), s(key[1])])
330         return charac
331     else:
332         V_pol = PolarizedSpace(V_iso, op_pol)
333         return character(V_pol, row_symmetry=row_symmetry)
```

ANNEXE B

EXEMPLES DE CALCULS

Dans cette annexe, on présente quelques exemples d'utilisation du code et de tests effectués sur les diagrammes $\gamma(5,3)$ et $\gamma(6,3)$ qui n'ont pas été inclus dans le texte pour en alléger la lecture. Ces tests et exemples peuvent être retrouvés en version interactive sur GitHub, pour y accéder voir les explications données en annexe A.

Exemples du chapitre 4

Les commandes suivantes permettent de retrouver les résultats mentionnés à l'exemple 5.1.

```
sage: R = QQ['x1,x2,x3']
sage: M = matrix.vandermonde(R.gens(), R)
sage: M
[   1   x1 x1^2]
[   1   x2 x2^2]
[   1   x3 x3^2]
sage: derivatives = [attrcall('derivative', x) for x in R.
    gens()]
sage: E_3 = Subspace([M.determinant()], derivatives)
sage: E_3.basis()
```

```
10 {0: (-x1^2*x2 + x1*x2^2 + x1^2*x3 - x2^2*x3 - x1*x3^2 + x2*
     x3^2, -x1*x2 + 1/2*x2^2 + x1*x3 - 1/2*x3^2, 1/2*x1^2 - x1
     *x2 + x2*x3 - 1/2*x3^2, x1 - x3, x2 - x3, 1)}
11 sage: E_3.dimension()
12 6
```

Les commandes suivantes permettent de retrouver les résultats mentionnés à l'exemple 4.2 à propos de l'exploitation du multidegré.

```
13 sage: v = vandermonde(3)
14 sage: v
15 -x00^2*x01 + x00*x01^2 + x00^2*x02 - x01^2*x02 - x00*x02^2 +
       x01*x02^2
16 sage: derivatives = partial_derivatives(v.parent())
17 sage: derivatives
18 {(-1,): [*.derivative(x00), *.derivative(x01), *.derivative(
     x02)]}
19 sage: E_3 = Subspace({v.multidegree(): [v]}, derivatives)
20 sage: E_3.basis()
21 {(2,): (x00^2 - 2*x00*x01 + 2*x01*x02 - x02^2, -2*x00*x01 +
     x01^2 + 2*x00*x02 - x02^2), (0,): (1,), (3,): (-x00^2*x01
      + x00*x01^2 + x00^2*x02 - x01^2*x02 - x00*x02^2 + x01*
     x02^2,), (1,): (x00 - x02, x01 - x02)}
22 sage: E_3.dimension()
23 6
24 sage: polarizators = polarization_operators(2)
25 sage: polarizators
26 {(-1, 1): [*.polarization(i1=0, row_symmetry=None, i2=1, d
     =1)], (1, -1): [*.polarization(i1=1, row_symmetry=None,
     i2=0, d=1)]}
```

```
27 sage: E_3 = PolarizedSpace(E_3, polarizators)
28 sage: E_3.dimension()
29 15
30 sage: E_3.dimensions()
31 {(0, 1): 2, (1, 2): 1, (0, 0): 1, (3, 0): 1, (2, 1): 1, (1,
       1): 2, (2, 0): 2, (1, 0): 2, (0, 3): 1, (0, 2): 2}
```

Notons ensuite que la fonction `PolariazedSpace` prend en argument un élément de type `Subspace` duquel une base est extraite et retourne un autre élément de type `Subsapce` construit à partir de cette base et des opérateurs donnés. La différence est que, cette fois, le calcul est adapté pour prendre en compte le changement d'anneau de polynômes entre un et deux (ou plus) jeux de variables.

Les commandes suivantes permettent de retrouver les résultats mentionnés à l'exemple 4.1 à propos des composantes isotypiques.

```
32 sage: v = vandermonde(Partition([2,1]))
33 sage: derivatives = partial_derivatives(v.parent())
34 sage: E_21 = Subspace({v.multidegree(): [v]}, derivatives)
35 sage: E_21 = IsotypicComponent(E_21, 3)
36 sage: E_21
37 {[1, 1, 1]: <subspace.Subspace object at 0x7f68add79f10>,
       [2, 1]: <subspace.Subspace object at 0x7f68add79910>}
```

Le résultat est maintenant un dictionnaire de `Subspace`, un pour chaque composante isotypique.

```
38 sage: E_21[Partition([2,1])].basis()
39 {((0,), (2, 1)): (-theta00 + theta02,)}
40 sage: E_21[Partition([1,1,1])].basis()
41 {((1,), (1, 1, 1)): (x01*theta00 - x02*theta00 - x00*theta01
```

```
     + x02*theta01 + x00*theta02 - x01*theta02,)}
```

Notons que `PolarizedSpace` peut prendre en argument un dictionnaire de `Subspace` qui sont alors considérés séparément.

```
sage: E_21 = PolarizedSpace(E_21, polarization_operators(2))
sage: E_21[Partition([2,1])].basis()
{((0, 0), (2, 1)): (theta00 - theta02,)}
sage: E_21[Partition([1,1,1])].basis()
{((0, 1), (1, 1, 1)): (-x11*theta00 + x12*theta00 + x10*
     theta01 - x12*theta01 - x10*theta02 + x11*theta02,), ((1,
      0), (1, 1, 1)): (x01*theta00 - x02*theta00 - x00*theta01
      + x02*theta01 + x00*theta02 - x01*theta02,)}
```

Les commandes suivantes permettent de retrouver les résultats sur les antisymétries mentionnés à l'exemple 4.5.

```
sage: v = vandermonde(Partition([2,1]))
sage: derivatives = partial_derivatives(v.parent())
sage: E_21 = Subspace({v.multidegree(): [v]}, derivatives)
```

À cette étape, on projette sur les composantes isotypiques de $\mathbb{S}_3$ en exploitant les antisymétries. On peut vérifier la différence entre ce calcul et l'exemple 4.2.

```
sage: E_21 = IsotypicComponent(E_21, 3, use_antisymmetry=
     True)
sage: E_21[Partition([2,1])].basis()
{((0,), (2, 1)): (theta00,)}
sage: E_21[Partition([1,1,1])].basis()
{((1,), (1, 1, 1)): (x00*theta01,)}
sage: p = E_21[Partition([2,1])].basis().values()[0][0]
sage: p.parent()
```

```
Diagonal Antisymmetric Polynomial Ring with 1 rows of 3 variables and inert variables over Rational Field with antisymmetries ((0, 2), (1,))
```

On polarise chaque composante isotypiques en exploitant les antisymétries.

```
sage: E_21 = PolarizedSpace(E_21, polarization_operators(2))
sage: E_21[Partition([2,1])].basis()
{((0, 0), (2, 1)): (theta00,)}
sage: E_21[Partition([1,1,1])].basis()
{((0, 1), (1, 1, 1)): (x10*theta01,), ((1, 0), (1, 1, 1)): (x00*theta01,)}
```

Enfin, les commandes suivantes peuvent être utilisées pour retrouver les résultats obtenus à l'exemple 4.4 à propos de l'exploitation des symétries sur les lignes.

```
sage: E_3 = IsotypicComponent(E_3, 3, use_antisymmetry=True)
sage: E_3 = PolarizedSpace(E_3, polarization_operators(2, row_symmetry="permutation"))
sage: E_3[Partition([1,1,1])].basis()
{((2, 1), (1, 1, 1)): (x00*x01*x10 + 1/2*x00^2*x11,), ((3, 0), (1, 1, 1)): (x00^2*x01,)}
sage: E_3[Partition([2,1])].basis()
{((1, 1), (2, 1)): (x00*x10 - x01*x10 - x00*x11,), ((1, 0), (2, 1)): (x00,), ((2, 0), (2, 1)): (-1/2*x00^2 + x00*x01,)}
sage: E_3[Partition([3])].basis()
{((0, 0), (3,)): (1,)}
```

Calculs directs exploitant toutes les stratégies

Exemples de calculs directs pour les partages de 3.

```
sage: E_mu(Partition([3]))
s[] # s[3] + s[1] # s[2, 1] + s[1, 1] # s[1, 1, 1] + s[2] #
    s[2, 1] + s[3] # s[1, 1, 1]
sage: E_mu(Partition([2,1]))
s[] # s[2, 1] + s[1] # s[1, 1, 1]
sage: E_mu(Partition([1,1,1]))
s[] # s[1, 1, 1]
```

On calcule les caractères $\mathcal{E}_{31}$, $\mathcal{E}_{221}$ et $\mathcal{E}_{32}$, en exploitant toutes les stratégies de calul et on donne le temps de calcul. (Voir exemples 5.2, 5.3, 5.4, 5.5, 5.6 et 5.7 pour plus de détails.)

```
sage: E_mu(Partition([3,1]))
s[] # s[3, 1] + s[1] # s[2, 1, 1] + s[1] # s[2, 2] + s[1, 1]
     # s[1, 1, 1, 1] + s[2] # s[2, 1, 1] + s[3] # s[1, 1, 1,
    1]
sage: timeit('E_mu(Partition([3,1]))')
5 loops, best of 3: 282 ms per loop
sage: E_mu(Partition([2,2,1]))
s[] # s[2, 2, 1] + s[1] # s[2, 1, 1, 1] + s[2] # s[1, 1, 1,
    1, 1]
sage: timeit('E_mu(Partition([2,2,1]))')
5 loops, best of 3: 417 ms per loop
sage: E_mu(Partition([3,2]))
s[] # s[3, 2] + s[1] # s[2, 2, 1] + s[1] # s[3, 1, 1] + s[1,
     1] # s[2, 1, 1, 1] + s[2] # s[2, 1, 1, 1] + s[2] # s[2,
```

```
     2, 1] + s[2, 1] # s[1, 1, 1, 1, 1] + s[3] # s[2, 1, 1, 1]
      + s[4] # s[1, 1, 1, 1, 1]
sage: timeit('E_mu(Partition([3,2]))')
5 loops, best of 3: 2.27 s per loop
```

Exemples de calculs pour des diagrammes troués

Calcul de $\mathcal{E}_{\gamma(5,3)}$.

```
sage: v = vandermonde(Diagram([(0,0),(1,0),(3,0)]))
sage: v
-x00^3*x01 + x00*x01^3 + x00^3*x02 - x01^3*x02 - x00*x02^3 +
     x01*x02^3
sage: r=1
sage: generator = {v.multidegree() : [v]}
sage: list_op = merge(merge(partial_derivatives(v.parent()),
     steenrod_operators(r, 1)), steenrod_operators(r, 2))
sage: W1 = Subspace(generators=generator, operators=list_op,
     add_degrees=add_degree)
sage: W1.basis()
{(2,): (x00^2 - x02^2, x01^2 - x02^2, -x00*x01 + x00*x02, -
     x00*x01 + x01*x02), (0,): (1,), (3,): (1/3*x00^3 - 1/3*
     x01^3 - x00*x02^2 + x01*x02^2, -1/3*x00^3 + x00*x01^2 -
     x01^2*x02 + 1/3*x02^3, -x00^2*x01 + 1/3*x01^3 + x00^2*x02
      - 1/3*x02^3), (1,): (x00, x01, x02), (4,): (-x00^3*x01 +
      x00*x01^3 + x00^3*x02 - x01^3*x02 - x00*x02^3 + x01*x02
     ^3,)}
sage: character(IsotypicComponent(W1, 3))
s[] # s[3] + s[1] # s[2, 1] + s[1] # s[3] + 2*s[2] # s[2, 1]
```

```
      + s[3] # s[1, 1, 1] + s[3] # s[2, 1] + s[4] # s[1, 1, 1]
sage: r = 2
sage: op_pol = polarization_operators(r, max_deg=v.degree())
sage: W2 = PolarizedSpace(IsotypicComponent(W1, 3), op_pol)
sage: character(W2)
s[] # s[3] + s[1] # s[2, 1] + s[1] # s[3] + s[1, 1] # s[1,
    1, 1] + s[1, 1] # s[2, 1] + 2*s[2] # s[2, 1] + s[2, 1] #
    s[1, 1, 1] + s[3] # s[1, 1, 1] + s[3] # s[2, 1] + s[4] #
    s[1, 1, 1]
sage: d = v.degree()
sage: list_degrees = [tuple((i,j)) for i in range(0,d) for j
     in range(0,d) if i+j>0]
sage: sym_diff = symmetric_derivatives(list_degrees)
sage: character(Range(W2, sym_diff))
s[] # s[3] + s[1] # s[2, 1] + s[1, 1] # s[1, 1, 1] + s[2] #
    s[2, 1] + s[3] # s[1, 1, 1]
sage: charac = character(W2) - character(Range(W2, sym_diff)
    )
sage: charac
s[1] # s[3] + s[1, 1] # s[2, 1] + s[2] # s[2, 1] + s[2, 1] #
     s[1, 1, 1] + s[3] # s[2, 1] + s[4] # s[1, 1, 1]
```

Exemples d'essais de calcul pour $\mathcal{E}_{\gamma(6,3)}$. On se concentre uniquement sur la omposante isotypique alternante (de type 111). Tout d'abord, on calcule $\mathcal{E}_{\gamma(6,3)}$ en construisant $\Lambda^*\mathcal{M}_{\gamma(6,3)}$ après polarisation de $\mathcal{M}_{\gamma(6,3)}$.

```
sage: v = vandermonde(Diagram([(0,0),(2,0),(4,0)]))
sage: generator = {v.multidegree() : [v]}
sage: list_op = partial_derivatives(v.parent())
sage: W1 = Subspace(generators=generator, operators=list_op,
```

```
        add_degrees=add_degree)
117 sage: W1 = IsotypicComponent(W1, Partition([1,1,1]))
118 sage: r=2
119 sage: op_pol = polarization_operators(r, max_deg=v.degree())
120 sage: W1 = PolarizedSpace(W1, op_pol)
121 sage: deg = v.degree()
122 sage: list_degrees = [tuple((i,j)) for i in range(0, deg)
      for j in range(0, deg) if i+j>0 and i+j<deg]
123 sage: sym_diff = symmetric_derivatives(list_degrees)
124 sage: W2 = Range(W1, sym_diff, add_degrees=
      add_degrees_isotypic)
125 sage: charac = character(W1) - character(W2)
126 sage: charac
127 s[3, 1] # s[1, 1, 1] + s[4, 1] # s[1, 1, 1] + s[5] # s[1, 1,
       1] + s[6] # s[1, 1, 1]
```

Ensuite, on calcule $\mathcal{E}_{\gamma(6,3)}$ en construisant $\Lambda^*\mathcal{M}_{\gamma(6,3)}$ avant de polariser de $\mathcal{M}_{\gamma(6,3)}$ et on polarise les deux espaces séparément. Ici on fait également intervenir les opérateurs de Steenrod.

```
128 sage: v = vandermonde(Diagram([(0,0),(2,0),(4,0)]))
129 sage: deg = v.degree()
130 sage: generator = {v.multidegree() : [v]}
131 sage: list_op = partial_derivatives(v.parent())
132 sage: list_op = merge(merge(list_op, steenrod_operators(1,
      1)), steenrod_operators(1,2))
133 sage: W1 = Subspace(generators=generator, operators=list_op,
       add_degrees=add_degree)
134 sage: W1 = IsotypicComponent(W1, Partition([1,1,1]))
135 sage: list_degrees = [tuple((i,)) for i in range(1,4)]
```

```
sage: sym_diff = symmetric_derivatives(list_degrees)
sage: sym_diff = merge(merge(sym_diff, steenrod_operators(1,
      1)), steenrod_operators(1, 2))
sage: W2 = Range(W1, sym_diff)
sage: r=2
sage: op_pol = polarization_operators(r, max_deg=4)
sage: list_degrees = [tuple((i,j)) for i in range(0,deg) for
      j in range(0,deg) if i+j!=0 and i+j<=deg]
sage: op_pol = merge(merge(op_pol, steenrod_operators(r, 1))
     , steenrod_operators(r,2))
sage: W1 = PolarizedSpace(W1, op_pol)
sage: W2 = PolarizedSpace(W2, op_pol)
sage: charac = character(W1) - character(W2)
sage: charac
s[4, 1] # s[1, 1, 1] + s[6] # s[1, 1, 1]
```

Dans le but d'obtenir le terme $s_{2,2} \otimes s_{1,1,1}$ qui apparaît dans $\boldsymbol{e}_{6,3}$, d'autres tests ont été menés avec les opérateurs de Steenrod. En voici un exemple.

```
sage: v = vandermonde(Diagram([(0,0),(2,0),(4,0)]))
sage: generator = {v.multidegree() : [v]}
sage: list_op = {}
sage: list_op = merge(list_op, steenrod_operators(1, 1))
sage: W1 = Subspace(generators=generator, operators=list_op,
      add_degrees=add_degree)
sage: W1 = IsotypicComponent(W1, Partition([1,1,1]))
sage: r=2
sage: op_pol = polarization_operators(r, max_deg=2)
sage: op_pol = merge(merge(op_pol, steenrod_operators(r, 1))
     , steenrod_operators(r, 2))
```

```
157 sage: W1 = PolarizedSpace(W1, op_pol)
158 sage: list_degrees = [tuple((i,j)) for i in range(0, deg)
      for j in range(0, deg) if i+j>0 and i+j<deg]
159 sage: sym_diff = steenrod_operators(r, 1)
160 sage: W2 = Range(W1, sym_diff, add_degrees=
      add_degrees_isotypic)
161 sage: charac = character(W1) - character(W2)
162 sage: charac
163 s[2, 2] # s[1, 1, 1] + s[6] # s[1, 1, 1]
```

Plus d'exemples de calculs peuvent être retrouvés dans la feuille de calcul `quotient_tests.ipynb` qui se trouve dans le dépôt Git du projet.

ANNEXE C

TABLES DE CARACTÈRES

Cette annexe contient les caractères qui ont pu être calculés avec le code implémenté. On donne chaque fois le diagramme et le $(\mathrm{GL_r} \times \mathbb{S}_\mathrm{n})$-caractère associé à l'espace construit à partir du déterminant de Vandermonde indicé par ce diagramme.

On commence par les partages jusqu'à 7. On rappelle que si γ est le diagramme d'un vecteur d'entiers, on peut se ramener au cas d'un partage (voir corollaire 4.2). Pour les partages jusqu'à 3 voir table 5.1 et pour les partages de 4 voir table 5.2 dans le chapitre 5.

TABLE C.1 Table de caractères de $\mathcal{E}_\lambda$ pour $|\lambda| = 5$

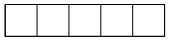
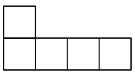
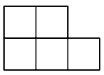
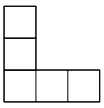
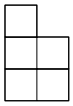
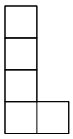
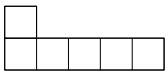
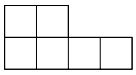
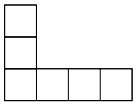
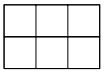
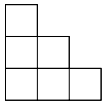
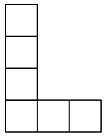
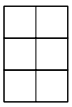
$n = 5$	
Diagramme	Caractère
	$1 \otimes s_5 + (s_1 + s_2 + s_3 + s_4) \otimes s_{41}$ $+(s_2 + s_{21} + s_{22} + s_3 + s_{31} + s_4 + s_{41} + s_5 + s_6) \otimes s_{32}$ $+(s_{11} + s_{21} + s_3 + 2s_{31} + s_{32} + s_4 + s_{41} + 2s_5 + s_{51} + s_6 + s_7) \otimes s_{311}$ $+(s_{21} + s_{211} + s_{22} + s_{31} + s_{311} + s_{32} + s_4 + 2s_{41} + s_{42} + s_5$ $+2s_{51} + s_6 + s_{61} + s_7 + s_8) \otimes s_{221}$ $+(s_{111} + s_{211} + s_{31} + s_{311} + s_{32} + s_{33} + 2s_{41} + s_{411} + s_{42} + 2s_{51}$ $+s_{52} + s_6 + 2s_{61} + s_7 + s_{71} + s_8 + s_9) \otimes s_{2111}$ $+(s_{1111} + s_{311} + s_{411} + s_{42} + s_{43} + s_{511} + s_{61} + s_{62}$ $+s_{71} + s_{81} + s_{10}) \otimes s_{11111}$
	$1 \otimes s_{41} + (s_1 + s_2) \otimes s_{32} + (s_1 + s_2 + s_3) \otimes s_{311}$ $+(s_{11} + s_2 + s_{21} + s_3 + s_4) \otimes s_{221}$ $+(s_{11} + s_{21} + s_3 + s_{31} + s_4 + s_5) \otimes s_{2111}$ $+(s_{111} + s_{31} + s_{41} + s_6) \otimes s_{11111}$
	$1 \otimes s_{32} + s_1 \otimes s_{311} + (s_1 + s_2) \otimes s_{221}$ $+(s_{11} + s_2 + s_3) \otimes s_{2111} + (s_{21} + s_4) \otimes s_{11111}$
	$1 \otimes s_{311} + s_1 \otimes s_{221} + (s_1 + s_2) \otimes s_{2111} + (s_{11} + s_3) \otimes s_{11111}$
	$1 \otimes s_{221} + s_1 \otimes s_{2111} + s_2 \otimes s_{11111}$
	$1 \otimes s_{2111} + s_1 \otimes s_{11111}$
	$1 \otimes s_{11111}$

TABLE C.2 Table de caractères de $\mathcal{E}_\lambda$ pour $|\lambda| = 6$

$n = 6$	
Diagramme	Caractère
	$1 \otimes s_{51} + (s_1 + s_2 + s_3) \otimes s_{42}$ $+(s_1 + s_2 + s_3 + s_4) \otimes s_{411} + (s_2 + s_{21} + s_4) \otimes s_{33}$ $+(s_{11} + s_2 + 2s_{21} + s_{22} + 2s_3 + 2s_{31} + 2s_4 + s_{41} + 2s_5 + s_6) \otimes s_{321}$ $+(s_{11} + s_{21} + s_3 + 2s_{31} + s_{32} + s_4 + s_{41} + 2s_5 + s_{51} + s_6 + s_7) \otimes s_{3111}$ $+(s_{21} + s_{211} + s_{31} + s_{32} + s_4 + s_{41} + s_5 + s_{51} + s_7) \otimes s_{222}$ $+(s_{111} + s_{21} + s_{211} + s_{22} + 2s_{31} + s_{311} + s_{32} + s_4 + 3s_{41}$ $+s_{42} + s_5 + 2s_{51} + 2s_6 + s_{61} + s_7 + s_8) \otimes s_{2211}$ $+(s_{111} + s_{211} + s_{31} + s_{311} + s_{32} + s_{33} + 2s_{41} + s_{411} + s_{42} + 2s_{51}$ $+s_{52} + s_6 + 2s_{61} + s_7 + s_{71} + s_8 + s_9) \otimes s_{21111}$ $+(s_{1111} + s_{311} + s_{411} + s_{42} + s_{43} + s_{511} + s_{61} + s_{62}$ $+s_{71} + s_{81} + s_{10}) \otimes s_{111111}$
	$1 \otimes s_{42} + s_1 \otimes s_{411} + s_1 \otimes s_{33} + (s_1 + s_{11} + 2s_2 + s_3) \otimes s_{321}$ $+(s_{11} + s_2 + s_{21} + s_3 + s_4) \otimes s_{3111} + (s_2 + s_{21} + s_4) \otimes s_{222}$ $+(s_{11} + 2s_{21} + 2s_3 + s_{31} + s_4 + s_5) \otimes s_{2211}$ $+(s_{111} + s_{21} + s_{22} + 2s_{31} + s_4 + s_{41} + s_5 + s_6) \otimes s_{21111}$ $+(s_{211} + s_{32} + s_{41} + s_{51} + s_7) \otimes s_{111111}$
	$1 \otimes s_{411} + (s_1 + s_2) \otimes s_{321} + (s_1 + s_2 + s_3) \otimes s_{3111}$ $+(s_{11} + s_3) \otimes s_{222} + (s_{11} + s_2 + s_{21} + s_3 + s_4) \otimes s_{2211}$ $+(s_{11} + s_{21} + s_3 + s_{31} + s_4 + s_5) \otimes s_{21111}$ $+(s_{111} + s_{31} + s_{41} + s_6) \otimes s_{111111}$
	$1 \otimes s_{33} + (s_1 + s_2) \otimes s_{321} + (s_{11} + s_3) \otimes s_{3111} + (s_{11} + s_3) \otimes s_{222}$ $+(s_2 + s_{21} + s_3 + s_4) \otimes s_{2211} + (s_{21} + s_{31} + s_4 + s_5) \otimes s_{21111}$ $+(s_{22} + s_{41} + s_6) \otimes s_{111111}$
	$1 \otimes s_{321} + s_1 \otimes s_{3111} + s_1 \otimes s_{222} + (s_1 + s_2) \otimes s_{2211}$ $+(s_{11} + s_2 + s_3) \otimes s_{21111} + (s_{21} + s_4) \otimes s_{111111}$
	$1 \otimes s_{3111} + s_1 \otimes s_{2211} + (s_1 + s_2) \otimes s_{21111} + (s_{11} + s_3) \otimes s_{111111}$
	$1 \otimes s_{222} + s_1 \otimes s_{2211} + s_2 \otimes s_{21111} + s_3 \otimes s_{111111}$

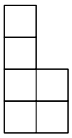
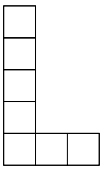
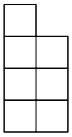
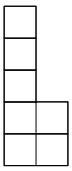
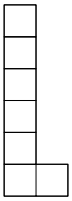
	$1 \otimes s_{2211} + s_1 \otimes s_{21111} + s_2 \otimes s_{111111}$
	$1 \otimes s_{21111} + s_1 \otimes s_{111111}$
	$1 \otimes s_{111111}$

TABLE C.3 Table de caractères de $\mathcal{E}_\lambda^{(2)}$ pour $|\lambda| = 7$

$n = 7$ et $r = 2$	
Diagramme	Caractère
	$1 \otimes s_{31111} + (s_1 + s_2) \otimes s_{211111} + s_1 \otimes s_{22111} + (s_{11} + s_3) \otimes s_{1111111}$
	$1 \otimes s_{2,2,2,1} + s_1 \otimes s_{2,2,1,1,1} + s_2 \otimes s_{2,1,1,1,1,1} + s_3 \otimes s_{1,1,1,1,1,1,1}$
	$1 \otimes s_{22111} + s_1 \otimes s_{211111} + s_2 \otimes s_{1111111}$
	$1 \otimes s_{211111} + s_1 \otimes s_{1111111}$
	$1 \otimes s_{1111111}$

On a également fait les calculs de $\mathcal{M}_\gamma$ pour des diagrammes troués obtenus à partir d'un partage de la forme $(n, 1)$ auquel on enlève une case $(a, 0)$ avec $1 \leq a \leq n-1$. On a pu remarquer que le caractère associé à un tel diagramme est égal à celui associé au diagramme contenant les cases $(0, j)$ pour $1 \leq j \leq n-2$ auquel on ajoute la case $(b, 1)$ avec $b = n - a - 1$. On présente ici les résultats obtenus pour $n = 4$ et $n = 5$.

TABLE C.4 Table de caractères de $\mathcal{M}\gamma$ pour $|\gamma| = 4$

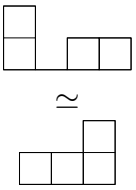
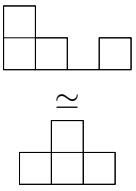
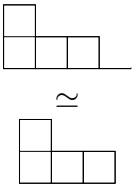
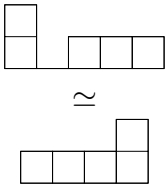
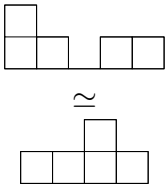
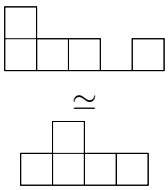
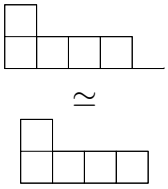
$n = 4$	
Diagramme	Caractère
	$(1 + s_1 + s_2) \otimes s_{31} + (s_1 + s_{11} + s_2 + s_3) \otimes s_{22}$ $+(s_1 + s_{11} + 2s_2 + s_{21} + 2s_3 + s_4) \otimes s_{211}$ $+(s_{11} + s_{21} + s_3 + s_{31} + s_4 + s_5) \otimes s_{1111}$
	$(1 + s_1) \otimes s_{31} + (s_1 + s_2) \otimes s_{22} + (s_1 + s_{11} + 2s_2 + s_3) \otimes s_{211}$ $+(s_{11} + s_{21} + s_3 + s_4) \otimes s_{1111}$
	$1 \otimes s_{31} + s_1 \otimes s_{22} + (s_1 + s_2) \otimes s_{211} + (s_{11} + s_3) \otimes s_{1111}$

TABLE C.5 Table de caractères de $\mathcal{M}\gamma$ pour $|\gamma| = 5$

$n = 5$	
Diagramme	Caractère
	$(1 + s_1 + s_2 + s_3) \otimes s_{41}$ $+(s_1 + s_{11} + 2s_2 + 2s_{21} + 2s_3 + s_{31} + 2s_4 + s_5) \otimes s_{32}$ $+(s_1 + s_{11} + 2s_2 + 2s_{21} + s_{22} + 3s_3 + 2s_{31} + 3s_4 + s_{41} + 2s_5 + s_6) \otimes s_{311}$ $+(s_{11} + s_{111} + s_2 + 3s_{21} + s_{211} + s_{22} + 2s_3 + 4s_{31} + s_{32} + 3s_4 + 3s_{41}$ $+3s_5 + s_{51} + 2s_6 + s_7) \otimes s_{221}$ $+(s_{11} + s_{111} + 2s_{21} + s_{211} + s_{22} + s_3 + 4s_{31} + s_{311} + 2s_{32} + 2s_4 + 4s_{41}$ $+s_{42} + 3s_5 + 3s_{51} + 3s_6 + s_{61} + 2s_7 + s_8) \otimes s_{2111}$ $+(s_{111} + s_{211} + s_{31} + s_{311} + s_{32} + s_{33} + 2s_{41} + s_{411} + s_{42}$ $+2s_{51} + s_{52} + s_6 + 2s_{61} + s_7 + s_{71} + s_8 + s_9) \otimes s_{11111}$
	$(s + s_1 + s_2) \otimes s_{41} + (s_1 + s_{11} + 2s_2 + s_{21} + 2s_3 + s_4) \otimes s_{32}$ $+(s_1 + s_{11} + 2s_2 + 2s_{21} + 3s_3 + s_{31} + 2s_4 + s_5) \otimes s_{311}$ $+(s_{11} + s_{111} + s_2 + 3s_{21} + s_{22} + 2s_3 + 3s_{31} + 3s_4 + s_{41} + 2s_5 + s_6) \otimes s_{221}$ $+(s_{11} + s_{111} + 2s_{21} + s_{211} + s_{22} + s_3 + 4s_{31} + s_{32} + 2s_4 + 3s_{41}$ $+3s_5 + s_{51} + 2s_6 + s_7) \otimes s_{2111}$ $+(s_{111} + s_{211} + s_{31} + s_{311} + s_{32} + 2s_{41} + s_{42} + 2s_{51}$ $+s_6 + s_{61} + s_7 + s_8) \otimes s_{11111}$
	$(s + s_1) \otimes s_{41} + (s_1 + s_{11} + 2s_2 + s_3) \otimes s_{32}$ $+(s_1 + s_{11} + 2s_2 + s_{21} + 2s_3 + s_4) \otimes s_{311}$ $+(s_{11} + s_2 + 3s_{21} + 2s_3 + s_{31} + 2s_4 + s_5) \otimes s_{221}$ $+(s_{11} + s_{111} + 2s_{21} + s_{22} + s_3 + 3s_{31} + 2s_4 + s_{41} + 2s_5 + s_6) \otimes s_{2111}$ $+(s_{111} + s_{211} + s_{31} + s_{32} + 2s_{41} + s_{51} + s_6 + s_7) \otimes s_{11111}$
	$1 \otimes s_{41} + (s_1 + s_2) \otimes s_{32} + (s_1 + s_2 + s_3) \otimes s_{311}$ $+(s_{11} + s_2 + s_{21} + s_3 + s_4) \otimes s_{221}$ $+(s_{11} + s_{21} + s_3 + s_{31} + s_4 + s_5) \otimes s_{2111}$ $+(s_{111} + s_{31} + s_{41} + s_6) \otimes s_{11111}$

BIBLIOGRAPHIE

Bergeron, F. (2009). *Algebraic combinatorics and coinvariant spaces*. CMS Treatises in Mathematics. Canadian Mathematical Society, Ottawa, ON ; A K Peters, Ltd., Wellesley, MA.

Bergeron, F. (2013). Multivariate diagonal coinvariant spaces for complex reflection groups. *Adv. Math.*, *239*, 97–108.

Bergeron, F. (2017). Open questions for operators related to rectangular Catalan combinatorics. *J. Comb.*, *8*(4), 673–703.

Bergeron, F., Garsia, A., Sergel Leven, E. et Xin, G. (2016). Compositional (km, kn)-shuffle conjectures. *Int. Math. Res. Not. IMRN*, (14), 4229–4270.

Bergeron, F. et Garsia, A. M. (1996). Science fiction and Macdonald's polynomials. In *Algebraic methods and q-special functions (Montréal, QC, 1996)*, volume 22 de *CRM Proc. Lecture Notes* 1–52. Amer. Math. Soc., Providence, RI.

Bergeron, F., Garsia, A. M., Haiman, M. et Tesler, G. (1999). Identities and positivity conjectures for some remarkable operators in the theory of symmetric functions. *Methods Appl. Anal.*, *6*(3), 363–420. Dedicated to Richard A. Askey on the occasion of his 65th birthday, Part III.

Bergeron, F. et Préville-Ratelle, L.-F. (2012). Higher trivariate diagonal harmonics via generalized Tamari posets. *J. Comb.*, *3*(3), 317–341.

Chevalley, C. (1955). Invariants of finite groups generated by reflections. *Amer. J. Math.*, *77*, 778–782.

Curtis, C. (1999). *Pioneers of Representation Theory : Frobenius, Burnside, Schur, and Brauer*.

Freyd, P., Yetter, D., Hoste, J., Lickorish, W. B. R., Millett, K. et Ocneanu, A. (1985). A new polynomial invariant of knots and links. *Bull. Amer. Math. Soc. (N.S.)*, *12*(2), 239–246.

Fulton, W. et Harris, J. (1991). *Representation theory*, volume 129 de *Graduate Texts in Mathematics*. Springer-Verlag, New York. A first course, Readings in Mathematics.

Garsia, A. M. et Haiman, M. (1993). A graded representation model for Macdonald's polynomials. *Proc. Nat. Acad. Sci. U.S.A.*, *90*(8), 3607–3610.

Gorsky, E. et Neguţ, A. (2015). Refined knot invariants and Hilbert schemes. *J. Math. Pures Appl. (9)*, *104*(3), 403–435.

Green, J. A. (2007). *Polynomial representations of* GL_n (augmented éd.), volume 830 de *Lecture Notes in Mathematics*. Springer, Berlin. With an appendix on Schensted correspondence and Littelmann paths by K. Erdmann, Green and M. Schocker.

Haglund, J. (2008). *The q,t-Catalan numbers and the space of diagonal harmonics*, volume 41 de *University Lecture Series*. American Mathematical Society, Providence, RI. With an appendix on the combinatorics of Macdonald polynomials.

Haglund, J., Haiman, M., Loehr, N., Remmel, J. B. et Ulyanov, A. (2005). A combinatorial formula for the character of the diagonal coinvariants. *Duke Math. J.*, *126*(2), 195–232.

Haiman, M. (1999). Macdonald polynomials and geometry. In *New perspectives in algebraic combinatorics (Berkeley, CA, 1996–97)*, volume 38 de *Math. Sci. Res. Inst. Publ.* 207–254. Cambridge Univ. Press, Cambridge.

Haiman, M. (2002). Vanishing theorems and character formulas for the Hilbert scheme of points in the plane. *Invent. Math.*, *149*(2), 371–407.

Haiman, M. (2003). Combinatorics, symmetric functions, and Hilbert schemes. In *Current developments in mathematics, 2002* 39–111. Int. Press, Somerville, MA.

Hikita, T. (2014). Affine Springer fibers of type A and combinatorics of diagonal coinvariants. *Adv. Math.*, *263*, 88–122.

Humphreys, J. E. (1975). *Linear algebraic groups*. Springer-Verlag, New York-Heidelberg. Graduate Texts in Mathematics, No. 21.

Humphreys, J. E. (1978). *Introduction to Lie algebras and representation theory*, volume 9 de *Graduate Texts in Mathematics*. Springer-Verlag, New York-Berlin. Second printing, revised.

James, G. et Kerber, A. (1981). *The representation theory of the symmetric*

group, volume 16 de *Encyclopedia of Mathematics and its Applications*. Addison-Wesley Publishing Co., Reading, Mass.

Lang, S. (2002). *Algebra* (third éd.), volume 211 de *Graduate Texts in Mathematics*. Springer-Verlag, New York.

Macdonald, I. G. (1987). Commuting differential operators and zonal spherical functions. In *Algebraic groups Utrecht 1986*, volume 1271 de *Lecture Notes in Math.* 189–200. Springer, Berlin.

Macdonald, I. G. (1995). *Symmetric functions and Hall polynomials* (second éd.). Oxford Classic Texts in the Physical Sciences. The Clarendon Press, Oxford University Press, New York.

Naimark, M. A. et Štern, A. I. (1982). *Theory of group representations*, volume 246 de *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences*. Springer-Verlag, New York. Translated from the Russian by Elizabeth Hewitt, Translation edited by Edwin Hewitt.

Orellana, R. et Zabrocki, M. (2019). A combinatorial model for the decomposition of multivariate polynomials rings as an s_n-module.

Prasad, A. (2015). *Representation theory*, volume 147 de *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Delhi. A combinatorial viewpoint.

Sagan, B. E. (2001). *The symmetric group* (second éd.), volume 203 de *Graduate Texts in Mathematics*. Springer-Verlag, New York. Representations, combinatorial algorithms, and symmetric functions.

Schiffmann, O. et Vasserot, E. (2011). The elliptic Hall algebra, Cherednik Hecke algebras and Macdonald polynomials. *Compos. Math.*, *147*(1), 188–234.

Shephard, G. C. et Todd, J. A. (1954). Finite unitary reflection groups. *Canadian J. Math.*, *6*, 274–304.

Stanley, R. P. (1999). *Enumerative combinatorics. Vol. 2*, volume 62 de *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, Cambridge.